ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόβλεψη πωλήσεων σε προϊόντα ταχείας κατανάλωσης με τη χρήση Μηχανικής Μάθησης

Machine Learning Sales forecasting for fast-moving consumer goods

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΠΟΥΛΙΔΗ ΣΤΕΦΑΝΟΥ

Επιβλέπων : Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλιος 2021

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόβλεψη πωλήσεων σε προϊόντα ταχείας κατανάλωσης με τη χρήση Μηχανικής Μάθησης

Machine Learning Sales forecasting for fast-moving consumer goods

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΠΟΥΛΙΔΗ ΣΤΕΦΑΝΟΥ

Επιβλέπων : Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Ιουνίου 2021.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
Ανδρέας-Γεώργιος Σταφυλοπάτης Καθηγητής Ε.Μ.Π.	Στέφανος Κόλλιας Καθηγητής Ε.Μ.Π.	Γεώργιος Στάμου Καθηγητής Ε.Μ.Π.

Αθήνα, Απρίλιος 2021

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή.Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....

Στέφανος Πουλίδης

Απρίλιος 2021

Περίληψη

Το αντικείμενο αυτής της διατριβής είναι η ανάλυση του ρόλου της μηχανικής μάθησης στην πρόβλεψη πωλήσεων, και συγκεκριμένα την πρόβλεψη πωλήσεων προϊόντων ταχείας κατανάλωσης με χρήση μηχανικής και βαθείας μάθησης.

Η μηχανική μάθηση είναι ένα εξαιρετικό εργαλείο για προβλέψεις, καθώς έχει τη δυνατότητα να αναλύει σε βάθος τα πραγματικά δεδομένα και χρονοσειρές και να δημιουργεί βελτιστοποιημένα μοντέλα. Σε αντίθεση με τις παραδοσιακές μεθοδολογίες πρόβλεψης πωλήσεων που σχεδιάζουν και προτείνουν γενικά μοντέλα και πρακτικές, χάρη στη μηχανική μάθηση, μπορούμε να αναπτύξουμε εργαλεία πλήρως προσαρμοσμένα στις συνθήκες και τα χαρακτηριστικά κάθε επιχείρησης και οργανισμού. Επιλέχθηκε το παρόν θέμα, διότι οι προβλέψεις πωλήσεων αποτελούν ταυτόχρονα μερικές από τις πιο σημαντικές αποφάσεις για μία επιχείρηση, και με τη χρήση μηχανικής μάθησης, υπάρχει σημαντικό περιθώριο βελτίωσης των προβλέψεων συγκριτικά με τα παραδοσιακά μοντέλα λήψης αποφάσεων.

Συγκεκριμένα, στην παρούσα διπλωματική αναλύονται οι εξελίξεις στον σχετικό ερευνητικό τομέα και κάνουμε τη δική μας ανάλυση και εκπαίδευση μοντέλων, με έμφαση στην πρόβλεψη πωλήσεων για προϊόντα ταχείας κατανάλωσης. Ο λόγος είναι ότι τα προϊόντα ταχείας κατανάλωσης αφορούν μία πληθώρα επιχειρήσεων, αλλά και η πρόβλεψη πωλήσεων τους, λόγω συγκεκριμένων χαρακτηριστικών τους, έχει μεγάλη σημασία για την κερδοφορία των σχετικών επιχειρήσεων.

Στόχος είναι τελικά η συγκριτική ανάλυση μοντέλων και μεθόδων για την επίλυση του παραπάνω προβλήματος, αλλά και η κατασκευή ενός μετα-μοντέλου μηχανικής μάθησης με πολύ καλή ακρίβεια για την πρόβλεψη πωλήσεων προϊόντων ταχείας κατανάλωσης. Για να το πετύχουμε αυτό, δοκιμάσαμε μερικά από τα πιο σύγχρονα και αποδοτικά μοντέλα στην πρόβλεψη πωλήσεων και τελικά κρατήσαμε τα καλύτερα μοντέλα, υψηλότερης ακρίβειας, και με χρήση τεχνικών Συνολικής Μάθησης (Ensemble learning) και Μέτα-μάθησης (Meta-learning) παράξαμε ενιαία συγκεντρωτικά μοντέλα προβλέψεων. Δείξαμε, όπως περιμέναμε, ότι με μικρό επιπλέον κόστος, μπορούμε να έχουμε συνδυαστικά αποτελέσματα καλύτερα από τα επιμέρους αποτελέσματα κάθε μοντέλου. Τέλος, συγκρίναμε διαφορετικές τεχνικές συνολικής μάθησης μεταξύ τους, για διαφορετικά μοντέλα, για να ελέγξουμε ποια δίνει τα καλύτερα δυνατά αποτελέσματα και, συνεπώς, προτείνουμε σε επιχειρίσεις του χώρου.

Συνολικά, η μεθοδολογία μας κατάφερε να συγκρίνει διαφορετικά μοντέλα τόσο μηχανικής, όσο και βαθειάς μάθησης για την περίπτωση πρόβλεψης πωλήσεων προϊόντων ταχείας κατανάλωσης και να βρούμε τα βέλτιστα μοντέλα και τεχνικές με ανάλυση κόστους-αξίας για το συγκεκριμένο πρόβλημα. Έτσι, η μέθοδος και τα αποτελέσματα μας είναι πολύ χρήσιμα για τις επιχειρήσεις που παράγουν και εμπορεύονται προϊόντα ταχείας κατανάλωσης, αφού δημιουργούμε για αυτές έναν χάρτη για να αυξήσουν την ακρίβεια των μοντέλων τους.

Ταυτόχρονα, αξιοποιώντας τα αποτελέσματα μας, μπορούν να γλιτώσουν εξαιρετικά μεγάλο κομμάτι της επένδυσης τους, αφού μπορούν να δουν ποια δεδομένα είναι σημαντικά για την πρόβλεψη των πωλήσεων τους και ποιοι αλγόριθμοι μηχανικής μάθησης δίνουν την βέλτιστη ακρίβεια, και άρα αξίζει να δοκιμαστούν και στην δική τους περίπτωση.

Η μέθοδος και τα αποτελέσματά μας μπορούν να γίνουν οδηγός για την ανάπτυξη μοντέλων πρόβλεψης πωλήσεων στις εταιρείες προϊόντων ταχείας κατανάλωσης, αλλά μπορούν και να χρησιμοποιηθούν ως σημείο αναφοράς από επιχειρήσεις σε άλλες βιομηχανίες, αλλά και πρόσθετα πειράματα και έρευνα σε πρόσθετες κατηγορίες προϊόντων.

Λέξεις Κλειδιά: μηχανική μάθηση, τεχνητή νοημοσύνη, βαθιά μάθηση, ανάλυση δεδομένων, μηχανική δεδομένων, καθαρισμός δεδομένων, οπτικοποίηση δεδομένων, μηχανική χαρακτηριστικών, λήψη αποφάσεων, πρόβλεψη πωλήσεων, στρατηγική τιμολόγησης, μοντέλα τιμολόγησης, προϊόντα ταχείας κατανάλωσης, στρατηγική επιχειρήσεων

Abstract

The object of this thesis is the analysis of the role of machine learning in sales forecasting and, in particular, of sales forecasting for fast-moving consumer goods.

Machine learning is an excellent tool for forecasting, as it has the ability to analyse in-depth real data and time-series, and build optimised models. In contrast to traditional sales forecasting methodologies that design and propose general models and practices, thanks to machine learning, we can develop tools fully adapted to the conditions and characteristics of each business and organisation. This topic was chosen because sales forecasting is both a valuable, value-adding tool for some of the most important business decisions and there is considerable room for improvement by effectively using machine learning implementations.

Specifically, in this dissertation we analyse the previous work and the developments in the relevant research field, and we execute our own analysis and model building, with emphasis on sales forecasting for fast-moving consumer goods (FMCGs). The reason is that fast-moving consumer goods concern a plethora of companies, but also FMCGs' sales forecasting is of great importance for the profitability of these companies.

Our goal was to construct a comparative analysis of models and methods to solve the above problem, but also the synthesis of machine learning models with very good accuracy to predict sales of FMCGs. To achieve this, we tested some of the most efficient models in sales forecasting and finally synthesised the best models (those of higher accuracy), by using Ensemble learning and Meta-learning techniques. So, we finally produced a single aggregate forecast model based on the champion models of our earlier analysis. We show that with negligible additional cost, we can have better results with stacking than with any other individual model and, therefore, meta-learners are an excellent investment for businesses' sales forecasting development departments. We also compare different meta-learning algorithms and techniques, for different models, to test which gives the best possible results and so to recommend it to the businesses in the FMCGs industries.

Overall, our methodology was able to compare both machine learning and deep learning models in the case of predicting sales of FMCGs and we present the best models and techniques with a cost-benefit analysis. Therefore, our method and results are very useful for companies that produce, promote or sell FMCGs, as they can increase the accuracy of their models for forecasting with smaller investments, and so achieving much better ROIs in their sales forecasting investments.

Our method and results can be a guide for the development of sales forecasting models and processes for businesses in the FMCGs industries, but it can also be used by additional experiments and research, as a reference point for sales forecasting at the same, or additional, product categories.

Keywords: machine learning, artificial intelligence, deep learning, data analysis, data engineering, data cleaning, data visualisation, feature engineering, decision making, sales forecasting, pricing strategy, pricing models, business intelligence, fast-moving consumer goods, business strategy

Ευχαριστίες

Η παρούσα διπλωματική εκπονήθηκε στο πλαίσιο του προπτυχιακού προγράμματος σπουδών της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσοβίου Πολυτεχνείο για την ολοκλήρωση των σπουδών μου και την προετοιμασία μου για σπουδές διδακτορικού επίπεδου.
Το ενδιαφέρον μου για την έρευνα στο σχετικό αντικείμενο, αλλά και γενικά η αγάπη μου για γνώση και έρευνα, προέκυψε από μία σειρά ανθρώπων που δεν πέρασαν απλά από τη ζωή μου, αλλά με επηρέασαν και βοήθησαν σημαντικά. Σε αυτούς ακριβώς τους ανθρώπους θέλω να αναφερθώ σε αυτές τις ευχαριστίες. Τους ευχαριστώ θερμά για τη βοήθεια και την συνεργασία μας, τις συμβουλές τους, το feedback, τις ιδέες τους, τον χρόνο τους, και πάνω από όλα, την διάθεσή τους να με στηρίξουν και υποστηρίξουν.

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα μου, κ. Ανδρέα-Γεώργιο Σταφυλοπάτη, Καθηγητή Ε.Μ.Π., ο οποίος μου έδωσε τη δυνατότητα να εκπονήσω την σχετική διπλωματική στο εργαστήριο του και μου παρείχε κάθε δυνατή βοήθεια και υποστήριξη όποτε την χρειάστηκα.

Ιδιαίτερες ευχαριστίες θέλω επίσης να αποδώσω στον κ. Γεώργιο Σιόλα, ΕΔΙΠ Ε.Μ.Π., για τον πολύτιμο χρόνο του όλον αυτόν τον καιρό, τον οποίο και μου έδωσε απλόχερα. Η καθοδήγηση του για την συγγραφή αυτής της διπλωματικής διατριβής ήταν καθοριστική, καθώς με στήριξε τόσο υλικά, με πόρους του εργαστηρίου Τεχνητής Νοημοσύνης και Συστημάτων Γνώσης, όσο και επιστημονικά και πνευματικά με εξαιρετικές προτάσεις για άρθρα, βιβλία και δημοσιεύσεις, αλλά και εποικοδομητική κριτική όταν την είχα ανάγκη.

Θα ήθελα, επίσης, να ευχαριστήσω τους κ.κ. Στέφανο Κόλλια, Καθηγητή Ε.Μ.Π. και Γεώργιο Στάμου, Καθηγητή Ε.Μ.Π. για την τιμή που μου έκαναν να είναι μέλη της επιτροπής εξέτασης της διπλωματικής μου εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια, τους φίλους και όλους τους ανθρώπους που με στήριξαν και στηρίζουν όλα αυτά τα χρόνια και είναι, άμεσα και έμμεσα, συνδημιουργοί και συμμέτοχοι της δουλειάς μου.

Στέφανος Πουλίδης
Απρίλιος, 2021

Πίνακας περιεχομένων

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Συστημάτων Τεχνητής Νοημοσύνης και Μάθησης (AILS) της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου στην Αθήνα, κατά το ακαδημαϊκό έτος 2020-2021.

0. Εκτεταμένη Περίληψη

Ο ρόλος της εκτεταμένης περίληψης είναι να δοθεί μία πλήρης περίληψη των επιμέρους κεφαλαίων της παρούσας διπλωματικής εργασίας στα ελληνικά. Τα κεφάλαια που ακολουθούν περιλαμβάνουν περιληπτικά το περιεχόμενο της εργασίας, το οποίο στα επόμενα κεφάλαια θα αναπτυχθεί στα αγγλικά. Σε αυτήν την εκτεταμένη περίληψη λοιπόν, περιλαμβάνονται όλα τα κύρια κομμάτια της εργασίας, από την περιγραφή των θεωρητικών εννοιών που πραγματεύεται και την αναφορά των μοντέλων που χρησιμοποιήθηκαν, μέχρι τα πειράματα που διατελέστηκαν, τα αποτελέσματα που προέκυψαν και τα τελικά μας συμπεράσματα.

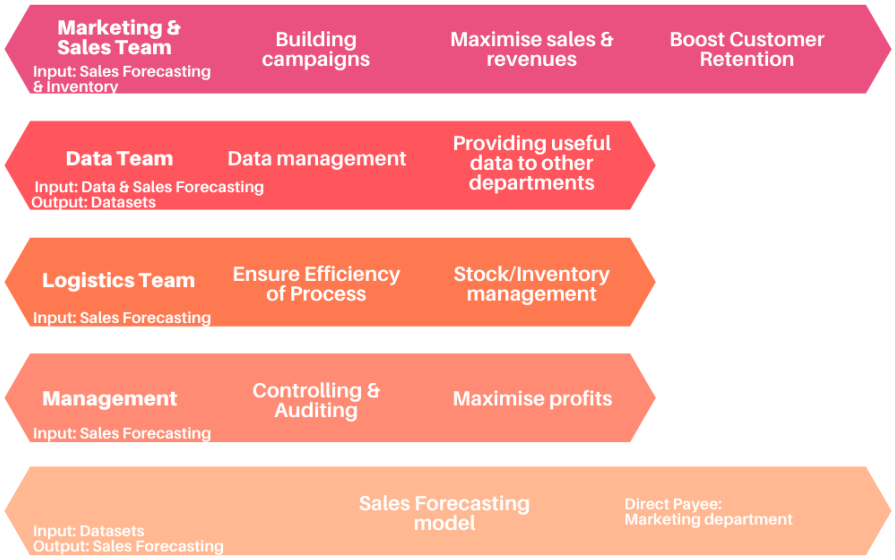
0.1 Εισαγωγή

Σκοπός της παρούσας διπλωματικής είναι η συγκριτική ανάλυση μοντέλων μηχανικής και βαθιάς μάθησης για το πρόβλημα της πρόβλεψης πωλήσεων προϊόντων ταχείας κατανάλωσης και η κατασκευή ενός συνολικού μοντέλου μηχανικής μάθησης με πολύ υψηλή ακρίβεια, και ταυτόχρονα μικρό κόστος. Για να το πετύχουμε αυτό, δοκιμάσαμε μερικά από τα πιο σύγχρονα και αποδοτικά μοντέλα στην πρόβλεψη πωλήσεων. Τα μοντέλα που τελικά κρατήσαμε και στα συμπεράσματα τις συγκεκριμένης διπλωματικής εργασίας προτείνουμε, είναι τα μοντέλα με τη μέγιστη δυνατή ακρίβεια από αυτά που δοκιμάστηκαν. Η δοκιμή περισσότερων μοντέλων δεν θα αύξανε υποχρεωτικά την ακρίβεια του τελικού μοντέλου μας, παρά μόνο, εάν βρίσκαμε μοντέλο υψηλότερης ακρίβειας από τα παρόντα καλύτερα μοντέλα και ταυτόχρονα μικρότερης συμφωνίας με τα καλύτερα μοντέλα μας, ώστε το τελικό μετά-μοντέλο μας να πετυχαίνει αισθητά καλύτερα αποτελέσματα από τα μεμονωμένα μοντέλα. Είδαμε, και δείχνουμε, ότι, όπως περιμέναμε, τα τελικά μετά-μοντέλα μας, έχουν υψηλότερη ακρίβεια από κάθε άλλο μεμονωμένο μοντέλο για την ακρίβεια πρόβλεψης πωλήσεων στα συγκεκριμένα δεδομένα, και μεταξύ των διαφορετικών μεθόδων για την κατασκευή μετά-μοντέλων που δοκιμάσαμε, αναδεικνύουμε στα αποτελέσματα μας αυτή που είχε την καλύτερη απόδοση στην ανάλυση κόστους-κέρδους.

Το πρώτο και κύριο σημαντικό πλεονέκτημα αυτής της μεθόδου, δηλαδή, η αρχική ανάλυση μοντέλων (Στάδιο 1) και η σύνθεση των καλύτερων εξ' αυτών σε ένα μετα-μοντέλο (Στάδιο 2) είναι ότι για την βελτίωση της ακρίβειας των μοντέλων που χρησιμοποιεί ο κάθε οργανισμός το επιπλέον κόστος που πληρώνεται είναι το ελάχιστο δυνατό. Το δεύτερο κύριο πλεονέκτημα είναι ότι, σε σχέση με άλλες μεθόδους, μπορεί να εφαρμοστεί όχι μόνο από οργανισμούς που δεν έχουν και θέλουν να δημιουργήσουν μοντέλα για την πρόβλεψη των πωλήσεων τους, αλλά και από οργανισμούς με ήδη υπάρχοντα μοντέλα. Για παράδειγμα, αν ένας οργανισμός/επιχείρηση έχει ήδη επενδύσει και χρησιμοποιεί μοντέλα μηχανικής μάθησης, αν θέλει να οδηγηθεί σε βελτίωση της ακρίβειας των προβλέψεων του με την αξιοποίηση νέων καλύτερων μοντέλων μηχανικής ή βαθείας μάθησης, για την αντικατάσταση των ήδη υπαρχόντων του, η προηγούμενη επένδυση του πέφτει εν κενώ. Ωστόσο, με τη χρήση της μεθόδου μας για τη δημιουργία των υποεξέταση μετά-μοντέλων, ο ίδιος οργανισμός μπορεί να αυξήσει την ακρίβεια των προβλέψεων του χωρίς να χρειαστεί να επανεπενδύσει σημαντικά κεφάλαια για την βελτίωση των μοντέλων πρόβλεψης των πωλήσεων του.

Καθώς προχωρούμε σε αυτήν την εργασία πρέπει διαρκώς να έχουμε στο μυαλό μας ότι ανεξάρτητα από τις βέλτιστες ακαδημαϊκά πρακτικές οπού μπορεί να υπάρχει ο χρόνος, η γνώση και οι πόροι να κυνηγήσουμε το θεωρητικά βέλτιστο αποτέλεσμα, στα πλαίσια ενός οργανισμού/ επιχείρησης τα πράγματα είναι αρκετά διαφορετικά. Η χρηματική επένδυση, ο συνολικά απαιτούμενος χρόνος, η ικανότητα και η απαιτούμενη εκπαίδευση των ανθρώπων που παράγουν, συντηρούν και χρησιμοποιούν ένα εργαλείο είναι καθοριστικής σημασίας. Το τεράστιο πλεονέκτημα της μηχανικής μάθησης είναι ότι μπορούμε να αντιμετωπίζουμε μαθηματικά μοντέλα και μερικές διαφορικές εξισώσεις σαν ένα πολύ χρήσιμο "μαύρο κουτί", αλλά αυτό δεν

ισχύει για τον ανθρώπινο παράγοντα. Τα μοντέλα μηχανικής μάθησης δίνουν κάποια αποτελέσματα, υψηλότερης ή χαμηλότερης ακρίβειας, τα οποία θα πρέπει να αποκτηθούν με συγκεκριμένο τρόπο από ανθρώπους, δεδομένα, μηχανήματα και άλλους αλγορίθμους. Την ροή πληροφορίας και ευθύνης των οργανισμών και προϊόντων που εξετάζουμε πρέπει να μπορούμε πλήρως να την παρακολουθήσουμε προκειμένου να κατανοήσουμε τις απαιτήσεις, τις ανάγκες, τα πιθανά προβλήματα και το ποιος είναι ο τελικός χειριστής των μοντέλων μας. Στην περίπτωση της πρόβλεψης πωλήσεων για προϊόντα ταχείας κατανάλωσης η πιο συνηθισμένη δομή είναι η εξής:



Σχήμα 1. Αξιοποίηση μοντέλων πρόβλεψης πωλήσεων ανά τμήμα επιχείρησης

Το τμήμα Μάρκετινγκ είναι αυτό που πληρώνει για τη δημιουργία των σχετικών εργαλείων πρόβλεψης πωλήσεων, αφού συγκριτικά με όλα τα τμήματα που κοστίζουν χρήματα σε έναν οργανισμό (cost centers), το τμήμα Μάρκετινγκ (πολλές φορές "Μάρκετινγκ και Πωλήσεων") είναι αυτό που δημιουργεί τα εταιρικά έσοδα και για αυτό το λόγο έχει και σημαντικό budget και ευελιξία για να πετύχει τους στόχους του.

Συνεπώς, συνήθως όλα τα εργαλεία μηχανικής μάθησης για πρόβλεψη των πωλήσεων πληρώνονται από το τμήμα Μάρκετινγκ του εκάστοτε οργανισμού και για αυτό το λόγο χτίζονται με λειτουργικότητες και χαρακτηριστικά που πρέπει τελικά να εξυπηρετούν τα ίδια.

Στο πλαίσιο της παρούσας διπλωματικής συνεργαστήκαμε με δύο μεγάλες επιχειρήσεις (marketplaces) που εμπορεύονται και προωθούν προϊόντα ταχείας κατανάλωσης. Συνεπώς, η γνώση μας επί του σχετικού χώρου δεν παρέμεινε στα στενά όρια θεωρητικής και ακαδημαϊκοκεντρικής προσέγγισης, αλλά από την επαφή μας με πραγματικές επιχειρήσεις του χώρου καταφέραμε να αποκτήσουμε μία πιο ολοκληρωμένη εικόνα του κλάδου.

Πριν προχωρήσουμε στο κομμάτι του θεωρητικού υποβάθρου, είναι ιδιαίτερα σημαντικό να γνωρίζουμε για ποια προϊόντα μιλάμε όταν χρησιμοποιούμε τον όρο "προϊόντα ταχείας κατανάλωσης". Στα προϊόντα ταχείας κατανάλωσης υπάγονται οι εξής κατηγορίες προϊόντων:

- Μόδα και ρουχισμός
- Ηλεκτρονικά/Ηλεκτρονικές συσκευές
- Ροφήματα
- Καλλυντικά
- Προϊόντα περιποίησης σώματος και προσώπου
- Απορρυπαντικά
- Καθαριστικά
- Προμαγειρεμένα και έτοιμα γεύματα
- Προϊόντα φούρνου
- Κατεψυγμένα
- Φάρμακα
- Αναλώσιμα γραφείου
- Άλλα προϊόντα ταχείας κατανάλωσης

Αν και, συνολικά δεν έχει μελετηθεί εκτενώς ο συγκεκριμένος τομέας, έχει διεξαχθεί πολλή έρευνα στην πρόβλεψη πωλήσεων για προϊόντα μόδας (περισσότερα στα επόμενα κεφάλαια).

Μέσα στην ευρεία περιοχή προϊόντων ταχείας κατανάλωσης, επέλεξα να επικεντρωθώ στα απορρυπαντικά και καθαριστικά για τους ακόλουθους λόγους:

1. Παρόλο που υπάρχει πολλή έρευνα σχετικά με την πρόβλεψη πωλήσεων για τη βιομηχανία μόδας, υπάρχει σχεδόν μηδενική έρευνα σχετικά με την πρόβλεψη πωλήσεων για απορρυπαντικά και καθαριστικά.
2. Η βιομηχανία απορρυπαντικών και καθαριστών είναι μια ταχέως αναπτυσσόμενη βιομηχανία με CAGR άνω του 4% YoY (χρόνο με το χρόνο).
3. Είναι μια βιομηχανία με μεγάλη ευαισθησία στις τιμές που δίνει ακόμη μεγαλύτερη σημασία στις στρατηγικές τιμολόγησης και στην ακριβή πρόβλεψη πωλήσεων για τη μεγιστοποίηση των εσόδων και κερδών.
4. Μπορούν να αντιπροσωπεύουν αποτελεσματικά το σύνολο των προϊόντων ταχείας κατανάλωσης.

Με την επιλογή των Καθαριστικών και Απορρυπαντικών σαν αντιπροσωπευτική ομάδα προϊόντων για τα προϊόντα ταχείας κατανάλωσης και την επαφή με δύο μεγάλες ευρωπαϊκές επιχειρήσεις, συλλέξαμε τα απαραίτητα δεδομένα με τα οποία και κατασκευάσαμε τα μοντέλα μας και πραγματοποιήσαμε την πειραματική ανάλυση μας.

Οργάνωση κεφαλαίων Εκτεταμένης Περίληψης

Στα επόμενα κεφάλαια θα καλύψουμε το θεωρητικό υπόβαθρο, θα παρουσιάσουμε την αναλυτική μέθοδο και μοντέλα που χρησιμοποιήσαμε, τις μετρήσεις, τα δεδομένα και τα αποτελέσματα μας.

Πιο συγκεκριμένα:

- Στο *Κεφάλαιο 0.2* δίνουμε το θεωρητικό υπόβαθρο της πρόβλεψης πωλήσεων. Θα σχολιάσουμε μαθηματικά μοντέλα, τι εφαρμόζεται στην πράξη, και την σχετική έρευνα στην πρόβλεψη πωλήσεων προϊόντων ταχείας κατανάλωσης.

- Στο *Κεφάλαιο 0.3* αναλύουμε το θεωρητικό υπόβαθρο της μηχανικής μάθησης και των μοντέλων που χρησιμοποιήσαμε.

- Στο Κεφάλαιο 0.4 θα σχολιάσουμε αναλυτικά τα αποτελέσματα μας με τους σχετικούς πίνακες και γραφήματα για την ευκολότερη κατανόηση και επεξήγηση τους. Συγκριτικά, παρατηρούμε την αξία όλων των μοντέλων μας αναλύοντας τα στο πλαίσιο της αλγοριθμικής οικογένειας τους, και την αξία των μετα-μοντέλων μέσω της ανάλυση κόστους-αξίας που διεξάγουμε.

- Στο Κεφάλαιο 0.5 αναλύουμε τα συμπεράσματα μας και τις μελλοντικές κατευθύνσεις της δουλειάς μας, τόσο για επέκταση της έρευνας σε προϊόντα ταχείας κατανάλωσης, όσο και πιθανές μελλοντικές προεκτάσεις της έρευνας μας σε άλλες κατηγορίες προϊόντων.

0.2 Θεωρητικό Υπόβαθρο

Σε αυτήν την ενότητα αναλύουμε τα ιδιαίτερα χαρακτηριστικά των προϊόντων ταχείας κατανάλωσης, καθώς και το θεωρητικό υπόβαθρο της πρόβλεψης πωλήσεων.

Η πρόβλεψη πωλήσεων μπορεί να έχει καθοριστικό αντίκτυπο στην επιτυχία και την απόδοση των εταιρειών. Οι ανακριβείς προβλέψεις πωλήσεων οδηγούν σε μεγάλα αποθέματα που αυξάνουν τα κόστη της διαχείρισης και αποθήκευσης (logistics) ή την ανεπάρκεια αποθεμάτων, που έχουν ως αποτέλεσμα την απώλεια κερδών για τις επιχειρήσεις. Ειδικότερα, για τις εταιρείες που παράγουν και εμπορεύονται προϊόντα ταχείας κατανάλωσης, όπως ηλεκτρονικά, καθαριστικά ή απορρυπαντικά και για την βιομηχανία της μόδας, είναι απαραίτητες οι ακριβείς προβλέψεις πωλήσεων. Η σημασία της έρευνας στα μοντέλα πρόβλεψης πωλήσεων υψηλής ακρίβειας προκύπτει από το γεγονός ότι η πρόβλεψη πωλήσεων δεν είναι μόνο καθοριστικής σημασίας για μία τεράστια πληθώρα εταιρειών, αλλά και το ότι οι περισσότερες επιχειρήσεις αντιμετωπίζουν τεράστιες προκλήσεις στο να πετύχουν ακριβείς προβλέψεις.

Ειδικά για τα προϊόντα ταχείας κατανάλωσης αυτή η ανάγκη είναι πολλαπλασιαστικά αυξημένη. Για παράδειγμα, οι επιχειρήσεις του χώρου πρέπει να υποβάλουν τα σχέδια παραγωγής τους προτού έχουν ακριβείς γνώσεις σχετικά με τις μελλοντικές πωλήσεις τους. Αυτό απαιτείται λόγω του γεγονότος ότι τα περισσότερα εργοστάσια παραγωγής βρίσκονται σε χώρες της Ασίας και, ως εκ τούτου, ο χρόνος αγοράς, παραγωγής και διάθεσης των προϊόντων είναι μεγαλύτερος από την περίοδο πώλησης.

Όσον αφορά το πρόβλημα δημιουργίας προβλέψεων υψηλής ακρίβειας, παράγοντες όπως οι μεταβαλλόμενες καιρικές συνθήκες, οι αργίες, οι δημόσιες εκδηλώσεις καθώς και η γενική οικονομική κατάσταση, μπορούν να έχουν αντίκτυπο στις μελλοντικές απαιτήσεις (Thomassey, 2010). Λόγω των μικρών κύκλων ζωής και της υψηλής μεταβλητότητας στα προϊόντα ταχείας κατανάλωσης και της γενικής αβεβαιότητας της ζήτησης, οι εταιρείες αντιμετωπίζουν συχνά υψηλές προκλήσεις όσον αφορά τις ακριβείς προβλέψεις.

Επιπλέον παράγοντες που συμβάλλουν στη δυσκολία παραγωγής προβλέψεων ακριβείας αφορούν τα ιδία τα δεδομένα πωλήσεων, στα οποία μπορούμε να παρατηρήσουμε διάφορους τύπους παραμέτρων και μοτίβων. Μερικές από αυτές είναι η τάση, η εποχικότητα, η αυτοσυσχέτιση και μοτίβα που προκαλούνται από την επίδραση εξωτερικών παραγόντων όπως η προσφορά, η τιμολόγηση ή η συμπεριφορά των ανταγωνιστών. Παρατηρούμε επίσης σημαντικό θόρυβο στις πωλήσεις, ειδικά για τα προϊόντα ταχείας κατανάλωσης. Ο θόρυβος αυτός προκαλείται από παράγοντες που δεν περιλαμβάνονται στην ανάλυση μας και μπορεί να αφορούν ανθρώπινα λάθη, αλγοριθμικά και τεχνολογικά λάθη, εταιρικές πολιτικές συλλογής δεδομένων, προκλήσεις και νομοθεσίες που αφορούν συλλογή δεδομένων, και πολλούς ακόμα εξωτερικούς παράγοντες. Επίσης, στα δεδομένα πωλήσεων, σχεδόν σε όλες τις περιπτώσεις έχουμε αρκετές καταγραφές με ακραίες τιμές (outliers). Οι outliers μπορεί να προκληθούν από ορισμένους συγκεκριμένους παράγοντες, π.χ. εκδηλώσεις προώθησης, μείωση τιμών, καιρικές συνθήκες κ.λπ. Εάν αυτά τα συγκεκριμένα συμβάντα επαναλαμβάνονται περιοδικά, μπορούμε να προσθέσουμε νέες δυνατότητες στα μοντέλα μας που θα δείχνουν αυτά τα ειδικά συμβάντα και θα περιγράφουν τις ακραίες τιμές της μεταβλητής-στόχου.

Συνεπώς, αν πρέπει να πραγματοποιήσουμε υψηλής ακρίβειας προβλέψεις πωλήσεων, πρέπει να λάβουμε υπόψιν μας τις αβεβαιότητες, τον θόρυβο και τις ακραίες τιμές. Οι αβεβαιότητες

σχετίζονται εν μέρει με τις πωλήσεις και συχνά οφείλονται σε έλλειψη γνώσεων ή εσφαλμένες πληροφορίες. Επιπλέον, οι συχνά μεταβαλλόμενες σειρές προϊόντων και η διαφορετική ζήτηση λόγω διαφορετικών παραγόντων που επηρεάζουν (εποχιακές επιρροές, πολιτικές τιμών και επιλογών κ.λπ.) είναι μόνο μερικοί από πολλούς παράγοντες που καθιστούν δύσκολη τη χρήση συμβατικών μοντέλων στατιστικών προβλέψεων. Επιπλέον, η πραγματική συσχέτιση μεταξύ των παραγόντων που επηρεάζουν είναι συχνά δύσκολο να κατανοηθεί ή να μην αναγνωριστεί από τον άνθρωπο και την παραδοσιακή χρήση χρονοσειρών, γεγονός που οδηγεί σε αυξημένη επιθυμία και αύξηση της χρήσης μεθόδων τεχνητής νοημοσύνης στο σχεδιασμό της ζήτησης [2].

Μια έρευνα στην οποία συμμετείχαν κορυφαίες εταιρίες διαφόρων κλάδων έδειξε ότι οι εταιρείες που βασίζονται στη λήψη αποφάσεων βάσει δεδομένων (Data Driven Decision Management - DDDM) επιτυγχάνουν καλύτερα αποτελέσματα (Provost και Fawcett, 2013). Κατά μέσο όρο, οι εταιρείες που χρησιμοποιούν συστήματα πρόβλεψης πωλήσεων στο DDDM τους είναι καλύτερες από το μέσο όρο της βιομηχανίας τους και συγκεκριμένα είναι κατά μέσο όρο 5% πιο παραγωγικές και 6% πιο κερδοφόρες σε σύγκριση με τον ανταγωνισμό τους (Brynjolfsson, Hitt and Kim, 2011; McAfee and Brynjolfsson, 2012). Οι εταιρείες που βασίζονται στη λήψη αποφάσεων βάσει δεδομένων (DDDM) χρησιμοποιούν κυρίως στατιστικούς τρόπους για την πρόβλεψη των πωλήσεων τους, όπως τη μέθοδο έρευνας εμπειρογνωμόνων [21] και αλγόριθμους που σχετίζονται με χρονοσειρές. Η μέθοδος έρευνας εμπειρογνωμόνων βασίζεται πλήρως στην ανθρώπινη εμπειρία και η ακρίβεια δεν είναι αρκετά σταθερή. Από την άλλη, οι αλγόριθμοι χρονοσειρών που περιλαμβάνουν αυτόματη παλινδρόμηση [21], εκθετική μέθοδο εξομάλυνσης [21] και μοντέλο ARIMA [12], χρησιμοποιούν ιστορικά δεδομένα πωλήσεων για την κατασκευή μοντέλων. Αυτές οι μέθοδοι δεν μπορούν να κάνουν πλήρη χρήση σχετικών παραγόντων στις πωλήσεις προϊόντων, όπως για παράδειγμα, τιμή, προωθητικές ενέργειες, αργίες κ.λπ., ώστε να καθίσταται δύσκολο να διασφαλιστεί η ακρίβεια των προβλέψεων σε σύνθετες αλλαγές.

Αυτά τα παραδοσιακά μοντέλα πρόβλεψης πωλήσεων λοιπόν, που βασίζονται στην προσέγγιση και ανάλυση χρονοσειρών (Box-Jenkins, Autoregressive ενσωματωμένος κινούμενος μέσος όρος (ARIMA), γενικευμένες αυτοεκτελεστικές συνθήκες (GARCH) κτλ.) έχουν ορισμένους εγγενείς περιορισμούς των προσεγγίσεων και εκτιμήσεων χρονοσειρών για τις προβλέψεις πωλήσεων και ειδικά για τις περιπτώσεις των προϊόντων ταχείας κατανάλωσης, όπως:

• Πρέπει να έχουμε ιστορικά δεδομένα για μεγάλο χρονικό διάστημα για να καταγράψουμε την εποχικότητα. Ωστόσο, συχνά δεν διαθέτουμε ιστορικά δεδομένα για μια μεταβλητή στόχου, παραδείγματος χάριν, στις περιπτώσεις που λανσάρουμε ένα νέο προϊόν. Μπορεί να έχουμε όμως χρονοσειρές πωλήσεων για παρόμοια προϊόντα και να μπορούμε να εκτιμήσουμε ότι το νέο μας προϊόν θα έχει παρόμοιο μοτίβο πωλήσεων, οπότε και οι αλγόριθμοι και μοντέλα πρόβλεψης πωλήσεων μας θα έπρεπε να λαμβάνουν υπόψιν τους αυτήν την πληροφορία. Σε αντίθεση με τα καθαρά μαθηματικά μοντέλα που αγνοούν αυτού του είδους δεδομένα (πληροφορίες και δεδομένα για παρόμοια προϊόντα) η μηχανική μάθηση για την πρόβλεψη πωλήσεων αποδεικνύεται ιδιαίτερα χρήσιμη, αφού μπορούμε με υψηλότερη ακρίβεια να χρησιμοποιούμε παρελθοντικά δεδομένα και δεδομένα από παρόμοια προϊόντα στην πρόβλεψη των πωλήσεων μας.

• Τα δεδομένα πωλήσεων μπορεί να έχουν πολλούς outliers και ελλειπή δεδομένα. Πρέπει να αντιμετωπίσουμε τα ακραία σημεία και να ελέγξουμε αναλυτικά τα δεδομένα πριν χρησιμοποιήσουμε μία προσέγγιση χρονοσειρών. Τα μαθηματικά μοντέλα δυσκολεύονται αρκετά να ανταποκριθούν σε αυτές τις περιπτώσεις και πέφτει εξαιρετικά πολύ η ακρίβεια τους. Σε αντίθεση, στην Προεργασία Δεδομένων που κάνουμε (Data Pre-procesing) (Κεφάλαιο 5.3) μπορούμε να έχουμε την ανάλυση που χρειαζόμαστε και να δίνουμε τελικά καλές λύσεις για τα μοντέλα μηχανικής μάθησης.

• Πρέπει να λάβουμε υπόψιν πολλούς εξωγενείς παράγοντες που επηρεάζουν τις πωλήσεις. Η πολυπλοκότητα των μαθηματικών μοντέλων αυξάνεται σχεδόν εκθετικά με την αύξηση των εξωγενών παραγόντων και επιπλέον παραμέτρων, αλλά η πολυπλοκότητα των μοντέλων μηχανικής μάθησης επηρεάζεται αισθητά λιγότερο. Συνεπώς, στις εφαρμογές μηχανικής και

βαθιάς μάθησης μπορούμε να έχουμε πολύ πιο σύνθετα μοντέλα προβλέψεων με σημαντικά μικρότερο κόστος.

Λόγω των παραπάνω δυσκολιών, τις τελευταίες τρεις δεκαετίες πολλές επιχειρήσεις μεταπήδησαν από τα μοντέλα ανάλυσης χρονοσειρών σε προσεγγίσεις παλινδρόμησης (Thomassey, 2010). Η πρόβλεψη πωλήσεων είναι μάλλον πρόβλημα παλινδρόμησης παρά πρόβλημα χρονοσειρών. Η πρακτική δείχνει ότι η χρήση των προσεγγίσεων παλινδρόμησης μπορεί συχνά να μας δώσει καλύτερα αποτελέσματα σε σύγκριση με τις μεθόδους χρονοσειρών και οι αλγόριθμοι μηχανικής μάθησης καθιστούν δυνατή την εύρεση μοτίβων στα δεδομένα. Μία από τις κύριες παραδοχές των μεθόδων παλινδρόμησης είναι ότι τα μοτίβα των δεδομένων στο παρελθόν, θα επαναληφθούν στο μέλλον, κάτι που συνάδει εξαιρετικά καλά με τα μοντέλα μηχανικής μάθησης και για αυτό και επιλέξαμε να δουλέψουμε με μοντέλα παλινδρόμησης μηχανικής μάθησης (Machine Learning Regression).

Προκειμένου να επιτευχθούν οι στόχοι μας για ακριβή μοντέλα πρόβλεψης με άμεση χρήση τους στην αγορά, χρησιμοποιούνται μέθοδοι μηχανικής μάθησης (ML), αλλά και βαθιάς μάθηση (DL) [4, 5]. Και οι δύο τύποι μεθόδων μπορούν να οριστούν ως υποπεριοχές της τεχνητής νοημοσύνης. Το πλεονέκτημα των μεθόδων που βασίζονται σε τεχνητή νοημοσύνη είναι μια αυτόματη ανάλυση των προτύπων και των εξαρτήσεων στα δεδομένα εισαγωγής, προκειμένου να χρησιμοποιηθούν για επόμενες προβλέψεις. Φυσικά, όπως συμβαίνει και με τις στατιστικές μεθόδους, δεν υπάρχει γενικά μέθοδος και μοντέλα τεχνητής νοημοσύνης που δημιουργούν βελτιωμένη πρόβλεψη για κάθε κατάσταση. Αντίθετα, κάθε μέθοδος μπορεί να χρησιμοποιηθεί για την επίτευξη διαφορετικών ιδιοτήτων ανάλογα με την εφαρμογή.

Οι Xia και Wong (2014) πρότειναν τις διαφορές μεταξύ των κλασικών μεθόδων (βάσει μαθηματικών και στατιστικών μοντέλων) και των σύγχρονων ευρετικών μεθόδων. Στην πρώτη ομάδα, αναφέρουν λύσεις όπως εκθετική εξομάλυνση, παλινδρόμηση, Box-Jenkins, αυτοπαλιδνρόμενος ενσωματωμένος κινούμενος μέσος όρος (ARIMA), γενικευμένες αυτοεκτελεστικές συνθήκες (GARCH) και άλλα. Τα περισσότερα από αυτά τα μοντέλα είναι γραμμικά και δεν είναι σε θέση να αντιμετωπίσουν την ασυμμετρική συμπεριφορά των περισσότερων δεδομένων πωλήσεων του πραγματικού κόσμου (Makridakis, Wheelwright, & Hyndman, 1998). Αντίθετα, οι σύγχρονες ευρετικές μέθοδοι μηχανικής μάθησης είναι συνήθως σε θέση να αντιμετωπίσουν αυτές τις προκλήσεις. Στον τομέα της πρόβλεψης των προϊόντων ταχείας κατανάλωσης, αυτές οι στατιστικές τεχνικές στις αρχικές τους μορφές αντιμετωπίζουν προκλήσεις στην παραγωγή ακριβών αποτελεσμάτων πρόβλεψης, λόγω παραγόντων όπως ακανόνιστα μοτίβα και υψηλή μεταβλητότητα (Choi, Hui, & Yu, 2011) των δεδομένων πωλήσεων.

Επίσης, πρέπει να θυμόμαστε ότι η πρόβλεψη πωλήσεων είναι ένα πρόβλημα πρόβλεψης πολλαπλών παραλλαγών χρονοσειρών. Οι κύριες προκλήσεις του έργου πρόβλεψης είναι οι μεταβλητές επιρροής υψηλής διάστασης με θόρυβο και οι περίπλοκες σχέσεις μεταξύ των χρονοσειρών. Οι επιχειρήσεις ανησυχούν διαρκώς για την κερδοφορία τους και, επομένως, επειδή οι πωλήσεις σχετίζονται άμεσα με τα κέρδη, έχουν ερευνηθεί ευρέως. Όπως αναφέραμε, η πρόβλεψη πωλήσεων είναι μια σημαντική πρόκληση, αφού στην πράξη, η συνδυασμένη δράση των πολύπλοκων παραγόντων επιρροής, της επιχειρηματικής στρατηγικής και των κανόνων της αγοράς προσθέτει δυσκολία στην πρόβλεψη του κέρδους (Chi-Jie Lu, "Πρόβλεψη πωλήσεων προϊόντων υπολογιστών με βάση μεταβλητό σχήμα επιλογής και υποστήριξη παλινδρόμησης φορέα", 128: 491–499, 2014). Για αυτό το λόγο, η αδυναμία ικανοποιητικής πρόβλεψης του επιχειρηματικού κέρδους, μεταβίβασε το ερευνητικό ενδιαφέρον στην πρόβλεψη πωλήσεων.

Ειδικά για τις περιπτώσεις των εταιρειών που παράγουν και εμπορεύονται προϊόντα ταχείας κατανάλωσης, έχουμε επιπλέον δυσκολίες στις προβλέψεις πωλήσεων αφού έχουν τα εξής χαρακτηριστικά:

Σχήμα 2. Προϊόντα ταχείας κατανάλωσης (FMCGs).

- Έχουν μικρή διάρκεια ζωής.
- Πωλούνται διαρκώς και σε διαφορετικές ποσότητες.
- Αγοράζονται από όλους και συνεπώς οι πωλήσεις τους επηρεάζονται από εξαιρετικά πολλούς παράγοντες.
- Αγοράζονται συχνά, αλλά είναι αγορές χαμηλής συμμετοχής, δηλαδή πολύ εύκολα οι αγοραστές των σχετικών προϊόντων μεταπηδούν σε νέα ή ανταγωνιστικά προϊόντα της αγοράς.

Επειδή τα προϊόντα ταχείας κατανάλωσης έχουν τόσο υψηλό κύκλο εργασιών και εξαιρετικά γρήγορες πωλήσεις, η αγορά τους δεν είναι μόνο πολύ μεγάλη, αλλά και πολύ ανταγωνιστική. Ορισμένες από τις μεγαλύτερες εταιρείες του κόσμου ανταγωνίζονται για μερίδιο αγοράς σε αυτόν τον κλάδο, συμπεριλαμβανομένων των Coca-Cola, Unilever, Procter & Gamble, Nestlé, PepsiCo και Danone. Για αυτό το λόγο οι συγκεκριμένες επιχειρήσεις αναγκάζονται να επικεντρώσουν τις προσπάθειές τους στο μάρκετινγκ για τα προϊόντα τους ταχείας κατανάλωσης για να δελεάσουν και να προσελκύσουν τους καταναλωτές να τα αγοράσουν. Τα προϊόντα ταχείας κατανάλωσης πωλούνται σε μεγάλες ποσότητες, επομένως θεωρούνται αξιόπιστη πηγή εσόδων, αλλά αυτός ο μεγάλος όγκος πωλήσεων αντισταθμίζει από τα χαμηλά περιθώρια κέρδους σε μεμονωμένες πωλήσεις. Για αυτό και η ακριβής πρόβλεψη πωλήσεων είναι τόσο μείζονος σημασίας για αυτές. Τα συστήματα εφοδιαστικής και διανομής τους απαιτούν τεράστιους πόρους και ιδιαίτερη προσοχή, και η ακριβέστερη πρόβλεψη πωλήσεων, τους επιτρέπει μεγέθυνση των περιθωρίων κέρδους τους, ενώ η απουσία ικανοποιητικά καλών προβλέψεων πωλήσεων τους κοστίζει εκατομμύρια από επιπλέον κόστη στη διαχείριση της εφοδιαστικής τους αλυσίδας.

Αν και υπάρχει ένα τεράστιο σώμα λογοτεχνίας και τεχνολογικής προόδου στο θέμα της πρόβλεψης (Fildes, Goodwin and Lawrence, 2006; McCarthy, Davis Golicic and Mentzer, 2006; Armstrong, Green and Graefe, 2015), υπάρχει σημαντική αδυναμία να εφαρμοστεί με επιτυχία σε επιχειρηματικές υλοποιήσεις. Οι υπεύθυνοι λήψης αποφάσεων παραμένουν δύσπιστοι σχετικά με τις συστάσεις που προσφέρουν τα συστήματα υποστήριξης προβλέψεων (FSS) και βασίζονται

στην εφαρμογή των δικών τους διανοητικών μοντέλων (Goodwin, Fildes, Lawrence και Stephens, 2011) των οποίων οι προκύπτουσες προβλέψεις είναι συχνά μη βέλτιστες. Εάν οι οργανισμοί επιθυμούν να βελτιώσουν την αποτελεσματικότητά τους (δηλαδή να μειώσουν το χάσμα μεταξύ των προβλέψεων και της πραγματοποίησης), πρέπει να σκεφτούν αυτά τα αγκυροβολημένα νοητικά μοντέλα.

Ευτυχώς, πολλοί ειδικοί και μελετητές συνέβαλαν στις μεθόδους πρόβλεψης πωλήσεων με την πάροδο των ετών. Αν και τα δεδομένα πωλήσεων είναι δεδομένα χρονοσειρών, ο αντίκτυπος των παραγόντων στις πωλήσεις δεν μπορεί να αγνοηθεί. Για την επίλυση αυτού του προβλήματος, το κλασικό μοντέλο παλινδρόμησης εφαρμόζεται, το οποίο και βασίζεται στη λήψη εύλογων παραγόντων επιρροής για τις πωλήσεις. Ωστόσο, είναι πολύ δύσκολο να κατανοηθούν και προσδιορισθούν οι παράγοντες που έχουν γραμμική σχέση με τις πωλήσεις. Στο Κεφάλαιο 2 (“Theoretical Background”), εξηγούμε μια μέθοδο για μικρά δείγματα (λίγα δεδομένα) που αξιοποιεί ανάλυση και μοντέλα μηχανικής μάθησης. Μπορεί να λύσει αποτελεσματικά τα προβλήματα μικρού δείγματος, μη γραμμικής και υψηλής διάστασης αναγνώρισης προτύπων και μπορεί να εφαρμοστεί σε άλλα προβλήματα μηχανικής μάθησης. Δυστυχώς όμως, η δυνατότητα εφαρμογής αυτής της μεθόδου μειώνεται, επειδή οι περισσότερες προβλέψεις πωλήσεων βασίζονται σε μεγάλο αριθμό δειγμάτων δεδομένων. Ειδικά για τα προϊόντα ταχείας κατανάλωσης, εκατομμύρια ή και δισεκατομμύρια πωλήσεις συμβαίνουν κάθε χρόνο και $10\text{-}10^3$ GB πωλήσεων δημιουργούνται κάθε χρόνο ανά επιχείρηση.

Προκειμένου να ξεχωρίζουν από τους ανταγωνιστές, οι εταιρείες επικεντρώνονται στην ευέλικτη εξυπηρέτηση πελατών, την ταχύτητα και την τήρηση των ημερομηνιών παράδοσης σε λογικές τιμές [7]. Οι συντομευμένοι κύκλοι ζωής των προϊόντων, οι διακυμάνσεις της συμπεριφοράς των πελατών και η ανάγκη άμεσης αντίδρασης στις διακυμάνσεις της αγοράς είναι μερικές μόνο από τις προκλήσεις σε αυτό το θέμα. Προκειμένου να μειωθούν οι βραχυπρόθεσμες αλλαγές στην αλυσίδα εφοδιασμού, είναι ζωτικής σημασίας να εφαρμοστούν αποτελεσματικά μοντέλα πρόβλεψης πωλήσεων που επιτρέπουν στις εταιρείες να προετοιμαστούν για μελλοντικές καταστάσεις εκ των προτέρων [11]. Οι διαθέσιμοι αλγόριθμοι πρόβλεψης στη βιβλιογραφία καθώς και στα εμπορικά (ERP) συστήματα αυξάνονται συνεχώς από την άποψη της ποσότητας και της πολυπλοκότητας.

Επιπλέον, η υπολογιστική ισχύς και η χωρητικότητα αποθήκευσης έχουν γίνει πολύ λιγότερο δαπανηρές, γεγονός που ανοίγει νέες δυνατότητες πρόβλεψης για εταιρείες [32]. Ωστόσο, τόσο τα ποσοτικά όσο και τα ποιοτικά μοντέλα πρόβλεψης πωλήσεων σε ορισμένες περιπτώσεις δεν είναι κατάλληλα για την παραγωγή πρόβλεψης επαρκούς ποιότητας λόγω της υψηλής και ταχύτατης διακύμανσης της αγοράς. Για αυτό το λόγο, θα μετρήσουμε το χρονικό κόστος εκτέλεσης όλων των μοντέλων μας, κάνοντας την υπόθεση ότι χρειάζονται τον ίδιο χρόνο προετοιμασίας και προγραμματισμού. Παραπάνω για τα θέματα υπολογιστικής ισχύς και των αναγκών των μοντέλων μπορούν να βρεθούν στα Κεφάλαια 5 (“Πειράματα”) και 6 (“Συμπεράσματα”) της παρούσας διατριβής.

Απλές μέθοδοι βαθείας μάθησης έχουν εφαρμοστεί στις προβλέψεις πωλήσεων στη βιβλιογραφία και έχουν επιτευχθεί υποσχόμενα αποτελέσματα. Ωστόσο, η ακρίβεια πρόβλεψης των παραπάνω μεθόδων δεν είναι ικανοποιητική όταν τα χαρακτηριστικά του προβλήματος πρόβλεψης είναι ασαφείς παράγοντες επιρροής, τεράστια δείγματα με πολύπλοκη δομή και μεγάλα χρονικά διαστήματα. Με βάση την ανάπτυξη της βαθιάς μάθησης, οι R. G. Hiranya Pemathilake et al. παρέχουν ένα υβριδικό μοντέλο με ενσωματωμένο κινούμενο μέσο όρο και επαναλαμβανόμενο νευρωνικό δίκτυο [6]. Συνδύασαν παραδοσιακά στατιστικά μοντέλα με βαθιά μάθηση και πέτυχαν πολλά υποσχόμενα αποτελέσματα. Ωστόσο, δεν υπάρχει περιγραφή του τρόπου αντιμετώπισης των δυναμικών μεταβλητών επιρροής. Αυτά τα προβλήματα ακρίβειας πρόβλεψης των μοντέλων θα τα αντιμετωπίσουμε και επιβεβαιώσουμε και στα δικά μας πειράματα.

Συνοψίζοντας, εξακολουθούν να υπάρχουν τρεις βασικές δυσκολίες στο πρόβλημα πρόβλεψης πωλήσεων:

- Τα μαζικά δεδομένα αυξάνουν τη δυσκολία υπολογισμού και μοντελοποίησης.

- Υπάρχει μια περίπλοκη μη γραμμική σχέση μεταξύ μεταβλητών επηρεασμού και πωλήσεων.

- Οι πωλήσεις προφανώς επηρεάζονται από τον παράγοντα του χρόνου, αλλά η επίδραση του παράγοντα του χρόνου στις πωλήσεις είναι δύσκολο να ποσοτικοποιηθεί.

Για αυτό το λόγο στη διεθνή έρευνα και βιβλιογραφία, εισάγονται διάφορες παραδοσιακές τεχνικές μηχανικής εκμάθησης (ML), όπως το Τυχαίο Δάσος (Random Forest) [31], Γραμμικής Παλινδρόμησης (Linear Regression) [34], XGBoost [5]. Οι παραπάνω μέθοδοι χρησιμοποιούν συνολικά παράγοντες που σχετίζονται με τις πωλήσεις και βελτιώνουν την ακρίβεια της πρόβλεψης. Ωστόσο, αυτά τα μοντέλα δεν μπορούν να επεξεργαστούν απευθείας δεδομένα χρονοσειρών, ούτε μπορούν να εξαγάγουν τους κρυμμένους κανόνες των δεδομένων. Πιο πρόσφατα, ξεκίνησαν να δοκιμάζονται και οι τεχνικές βαθιάς μάθησης, όπως το CNN [6] και το RNN [7] οι οποίες έχουν αποδειχθεί ανταγωνιστικές σε αυτόν τον τομέα. Από τα RNN μοντέλα, το LSTM [8] είναι ανώτερο από άλλες μεθόδους στην ακρίβεια των προβλέψεων. Για αυτό το λόγο, στα δικά μας πειράματα, με βάση και την προηγούμενη ερευνητική δουλειά που έχει πραγματοποιηθεί, θα επενδύσουμε στην εκπαίδευση ενός LSTM δικτύου. Περισσότερα για όλα τα μοντέλα μπορούν να βρεθούν στα Κεφάλαια 4 (“Μέθοδοι και Μοντέλα”) και 5 (“Πειράματα”).

Πίσω στις προκλήσεις μας στην πρόβλεψη πωλήσεων, θα αντιμετωπίσουμε τρία επιπλέον προβλήματα: αραιά δεδομένα, προτιμήσεις χρηστών και την ύπαρξη ενός ενιαίου μοντέλου με καλή απόδοση.

Τα αραιά δεδομένα εμφανίζονται συχνά στην πρόβλεψη πωλήσεων αφού:

1. Μεταξύ όλων των προϊόντων, μόνο ένα μικρό μέρος έχει καθημερινές πωλήσεις. Συνήθως υπάρχει πολύ μεγάλη διακύμανση στην συχνότητα πωλήσεων κάθε προϊόντος και συνεπώς δεν υπάρχουν εύκολοι τρόποι εξομάλυνσης των δεδομένων για κοινή αντιμετώπιση και ύπαρξη ενιαίου μοντέλου για όλα τα προϊόντα μιας επιχείρησης. Για αυτό το λόγο, επικεντρωνόμαστε μόνο στα προϊόντα ταχείας κατανάλωσης, εξετάζοντας προϊόντα με πολύ υψηλή συχνότητα πωλήσεων. Σε περίπτωση που οι συγκεκριμένες επιχειρήσεις έχουν και προϊόντα αργούς κατανάλωσης, αυτά θεωρούνται outliers από το μοντέλα μας και εξαιρούνται. Επίσης, για αυτό το λόγο, δοκιμάζουμε δύο διαφορετικές μεθόδους για feature scaling, τόσο standardisation, όσο και normalisation των δεδομένων μας.

2. Όσο μικρότερη είναι η διάσταση διαίρεσης των προϊόντων, τόσο πιο εμφανής είναι η αραιότητα των δεδομένων. Χρειαζόμαστε όμως μικρότερα δεδομένα αναλυτικότητας, διότι είναι ακριβή για τη λήψη αποφάσεων της καθημερινής λειτουργίας. Για αυτό το λόγο είναι τόσο σημαντική η αρχική, ακριβής ανάλυση των δεδομένων μας (Data Pre-processing). Για αυτό το σκοπό, ένα σημαντικό κομμάτι του Κεφαλαίου 5, επενδύεται στην ανάλυση των δεδομένων που είχαμε στην διάθεση μας και χρησιμοποιήσαμε για να επιλύσουμε τα παραπάνω προβλήματα.

Δεδομένου ότι κανένα μοντέλο δεν μπορεί να πετύχει την θεωρητική 100% ακρίβεια είναι ιδιαίτερα σημαντικό να επιλέγουμε σωστά το “bias” των προβλέψεων του μοντέλου μας. Δηλαδή, +0.1 και -0.1 RMSE μπορεί θεωρητικά να είναι ισοδύναμο σφάλμα κατά απόλυτη τιμή, αλλά στην πράξη οδηγούν σε πολύ διαφορετικά κέρδη για έναν οργανισμό, αφού, για παράδειγμα, όταν το κόστος αποθήκευσης είναι μικρότερο από το κόστος εξάντλησης, η κατάλληλη υπερεκτίμηση της πρόβλεψης είναι επωφελής για άμεση επαναφορά. Αντίθετα, όταν το κόστος αποθήκευσης είναι μεγαλύτερο από το κόστος εξάντλησης των αποθεμάτων, τότε θα προτιμούσαμε το μοντέλο μας να εκτιμά μικρότερες, από ότι υψηλότερες, πωλήσεις. Ωστόσο, υπάρχει ελάχιστη βιβλιογραφία και πολύ λίγοι ειδικοί και ερευνητές μελετούν αυτά τα όρια και το bias κατά την πρόβλεψη πωλήσεων.

Δεδομένης της πολυπλοκότητας και της αναγκαιότητας του προβλήματος, διάφοροι συγγραφείς και ερευνητές έχουν εργαστεί στην έρευνα για τις προβλέψεις πωλήσεων. Σε αυτή τη διατριβή, μελετάμε τη χρήση μοντέλων μηχανικής μάθησης για αναλυτικές προβλέψεις πωλήσεων προϊόντων ταχείας κατανάλωσης. Ο κύριος στόχος είναι να εξεταστούν οι κύριες προσεγγίσεις και οι μελέτες περιπτώσεων της χρήσης μηχανικής και βαθειάς μάθησης για την πρόβλεψη πωλήσεων σε προϊόντα ταχείας κατανάλωσης.

0.3 Μέθοδος και Μοντέλα

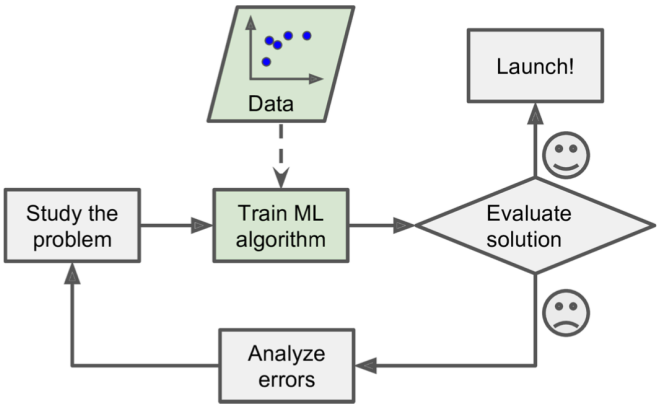
Σε αυτήν την ενότητα παρουσιάζουμε τα θεμέλια της μηχανικής μάθησης και της βαθιάς μάθησης και περιγράφουμε τους βασικούς αλγόριθμους στους οποίους βασίζονται τα μοντέλα της έρευνας μας. Μέχρι το τέλος αυτού του κεφαλαίου, ο αναγνώστης θα πρέπει να είναι εξοικειωμένος με όλες τις έννοιες και τα μοντέλα που χρησιμοποιούνται στην παρούσα εργασία.

Προκειμένου να επιτευχθεί ακριβής πρόβλεψη παρά τις υψηλές απαιτήσεις της αγοράς, χρησιμοποιούνται μέθοδοι μηχανικής (ML) και βαθειάς (DL) μάθησης. Και οι δύο τύποι μεθόδων μπορούν να οριστούν ως υπο-περιοχές της τεχνητής νοημοσύνης. Το πλεονέκτημα των μεθόδων που βασίζονται σε AI είναι μια αυτόματη ανάλυση των προτύπων και των εξαρτήσεων στα δεδομένα εισαγωγής, προκειμένου να επαναχρησιμοποιηθούν για μελλοντικές προβλέψεις.

Στο Κεφάλαιο 4, θα συζητήσουμε λεπτομερώς τη μέθοδο και τα μοντέλα μηχανικής και βαθείας μάθησης που χρησιμοποιούμε σε αυτήν τη διατριβή, αλλά, προς το παρόν, θα δώσουμε μερικές θεωρητικές πληροφορίες για την τεχνητή νοημοσύνη και την μηχανική μάθηση γενικά.

Η **Τεχνητή Νοημοσύνη (AI)**, που αναφέρεται επίσης ως μηχανική νοημοσύνη, περιγράφει τη νοημοσύνη που επιδεικνύουν οι μηχανές, η οποία διαφέρει από τη φυσική νοημοσύνη, το είδος της νοημοσύνης που επιδεικνύουν οι άνθρωποι.

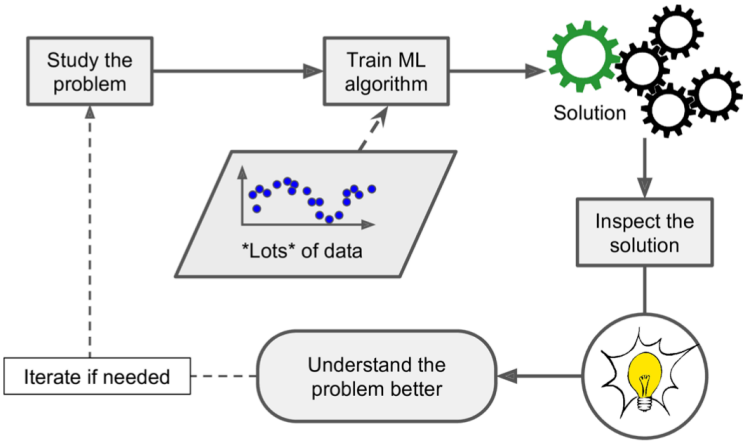
Η **Μηχανική Μάθηση (ML)** (ή μηχανική εκμάθηση) είναι ένα υπο-πεδίο της τεχνητής νοημοσύνης και της επιστήμης των υπολογιστών και αφορά τους αλγόριθμους δημιουργίας οι οποίοι, για να είναι χρήσιμοι, βασίζονται σε μια συλλογή δεδομένων κάποιου φαινομένου. Αυτά τα δεδομένα μπορεί να προέρχονται από τη φύση, να είναι χειροποίητα από ανθρώπους ή να δημιουργούνται από έναν άλλο αλγόριθμο. Η μηχανική εκμάθηση μπορεί επίσης να οριστεί ως η διαδικασία επίλυσης ενός πρακτικού προβλήματος με 1) συλλογή ενός συνόλου δεδομένων και 2) αλγοριθμική δημιουργία ενός στατιστικού μοντέλου βάσει αυτού του συνόλου δεδομένων.



Σχήμα 3. Μηχανική Μάθηση [48]

Η μηχανική μάθηση αφορά την εξαγωγή γνώσεων από δεδομένα. Βρίσκεται στη διασταύρωση της στατιστικής, της τεχνητής νοημοσύνης και της επιστήμης των υπολογιστών και είναι επίσης γνωστή ως προγνωστική ανάλυση ή στατιστική μάθηση. Η εφαρμογή μεθόδων μηχανικής μάθησης έχει γίνει τα τελευταία χρόνια πανταχού παρούσα στην καθημερινή ζωή. Από τις αυτόματες προτάσεις για ποιες ταινίες να παρακολουθήσουμε, μέχρι ποια φαγητά να παραγγείλουμε ή ποια προϊόντα να αγοράσουμε, έως την αναγνώριση ανθρώπων σε φωτογραφίες. Πολλοί σύγχρονοι ιστότοποι και συσκευές έχουν στον πυρήνα τους αλγόριθμους μηχανικής μάθησης. Εκτός των εμπορικών εφαρμογών, η μηχανική μάθηση είχε τεράστια επίδραση στον τρόπο με τον οποίο πραγματοποιείται η έρευνα βάσει δεδομένων σήμερα.

Επιπλέον, η Μηχανική Μάθηση μπορεί να βοηθήσει τους ανθρώπους να μάθουν. Για παράδειγμα, όταν το φίλτρο ανεπιθύμητων μηνυμάτων έχει εκπαιδευτεί σε αρκετά ανεπιθύμητα μηνύματα, μπορεί εύκολα να ελεγχθεί για να αποκαλυφθεί ο κατάλογος των λέξεων και οι συνδυασμοί λέξεων που πιστεύει ότι είναι οι καλύτεροι προγνωστικοί παράγοντες του spam. Μερικές φορές αυτό θα αποκαλύψει ανυποψίαστους συσχετισμούς ή νέες τάσεις, και έτσι θα οδηγήσει σε καλύτερη κατανόηση του προβλήματος.



Σχήμα 4. Μαθαίνοντας με τη χρήση μηχανικής μάθησης [48]

Η εφαρμογή τεχνικών μηχανικής μάθησης για την αξιοποίηση μεγάλων ποσοτήτων δεδομένων μπορεί να μας βοηθήσει να ανακαλύψουμε μοτίβα που δεν είναι άμεσα εμφανή με την χρήση παραδοσιακών μαθηματικών μεθόδων.

Έτσι, η μηχανική μάθηση είναι ιδανική για:

- Προβλήματα για τα οποία οι υπάρχουσες λύσεις απαιτούν πολλές ρυθμίσεις χειρός ή μεγάλες λίστες κανόνων (ένας αλγόριθμος Machine Learning μπορεί συχνά να απλοποιήσει τον κώδικα και να έχει καλύτερη απόδοση).
- Πολύπλοκα προβλήματα για τα οποία δεν υπάρχει καθόλου καλή λύση χρησιμοποιώντας μια παραδοσιακή προσέγγιση (οι καλύτερες τεχνικές μηχανικής μάθησης μπορούν να βρουν μια λύση).
- Κυμαινόμενα περιβάλλοντα (ένα σύστημα μηχανικής μάθησης μπορεί να προσαρμοστεί σε νέα δεδομένα).
- Λήψη πληροφοριών σχετικά με πολύπλοκα προβλήματα και μεγάλες ποσότητες δεδομένων.

Η Μηχανική Μάθηση μπορεί να Επιβλεπόμενη (Supervised), Ημι-επιβλεπόμενη (Semi-Supervised), Μη Επιβλεπόμενη (Unsupervised) και Ενισχυτικής Μάθησης (Reinforcement Learning).

Επιβλεπόμενη Μάθηση (Supervised Learning) είναι η διαδικασία όπου ο αλγόριθμος κατασκευάζει μια συνάρτηση που απεικονίζει δεδομένα εισόδου (σύνολο εκπαίδευσης) σε γνωστές επιθυμητές εξόδους, με απώτερο στόχο τη γενίκευση της συνάρτησης αυτής και για εισόδους με άγνωστη έξοδο.

Ημι-επιβλεπόμενη (Semi-Supervised Learning) στην οποία, το σύνολο δεδομένων περιέχει παραδείγματα με ετικέτα και χωρίς σήμανση. Συνήθως, η ποσότητα των μη επισημασμένων παραδειγμάτων είναι πολύ μεγαλύτερη από τον αριθμό των επισημασμένων παραδειγμάτων. Ο στόχος ενός αλγορίθμου Ημι-επιβλεπόμενη μάθησης είναι ο ίδιος με τον στόχο του αλγορίθμου της επιβλεπόμενης μάθησης. Η ελπίδα εδώ είναι ότι η χρήση πολλών παραδειγμάτων χωρίς ετικέτα μπορεί να βοηθήσει τον αλγόριθμο εκμάθησης να βρει, παράξει ή υπολογίσει ένα καλύτερο μοντέλο.

Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning), όπου ο αλγόριθμος κατασκευάζει ένα μοντέλο για κάποιο σύνολο εισόδων υπό μορφή παρατηρήσεων χωρίς να γνωρίζει τις επιθυμητές εξόδους.

Ενισχυτική Μάθηση (Reinforcement Learning), όπου ο αλγόριθμος μαθαίνει μια στρατηγική ενεργειών μέσα από άμεση αλληλεπίδραση με το περιβάλλον. Χρησιμοποιείται κυρίως σε προβλήματα Σχεδιασμού (Planning), όπως για παράδειγμα ο έλεγχος κίνησης ρομπότ και η βελτιστοποίηση εργασιών σε εργοστασιακούς χώρους.

Ρηχή vs Βαθειάς Μάθησης (Shallow vs Deep Learning)

Ένας ρηχός αλγόριθμος μάθησης μαθαίνει τις παραμέτρους του μοντέλου απευθείας από τα χαρακτηριστικά των παραδειγμάτων εκπαίδευσης. Οι περισσότεροι εποπτευόμενοι αλγόριθμοι μάθησης είναι ρηχοί. Οι διαβόητες εξαιρέσεις είναι οι αλγόριθμοι εκμάθησης νευρωνικών δικτύων, ειδικά αυτοί που δημιουργούν νευρωνικά δίκτυα με περισσότερα από ένα επίπεδα μεταξύ εισόδου και εξόδου. Τέτοια νευρωνικά δίκτυα ονομάζονται βαθιά νευρωνικά δίκτυα. Στην μάθηση ενός τέτοιου δικτύου (ή πιο απλά στην βαθειά μάθηση), σε αντίθεση με τη ρηχή μάθηση, οι περισσότερες παράμετροι μοντέλου μαθαίνονται όχι απευθείας από τα χαρακτηριστικά των παραδειγμάτων εκπαίδευσης, αλλά από τα αποτελέσματα των προηγούμενων επιπέδων.

Ταξινόμηση vs Παλινδρόμηση (Classification vs Regression)

Η ταξινόμηση είναι ένα πρόβλημα της αυτόματης εκχώρησης μιας ετικέτας σε ένα μη επισημασμένο παράδειγμα. Η ανίχνευση ανεπιθύμητων στοιχείων (παραδείγματος χάριν spam email) είναι ένα διάσημο παράδειγμα ταξινόμησης.

Στη μηχανική εκμάθηση, το πρόβλημα ταξινόμησης επιλύεται από έναν αλγόριθμο εκμάθησης ταξινόμησης που λαμβάνει μια συλλογή από επισημασμένα παραδείγματα ως εισόδους και παράγει ένα μοντέλο που μπορεί να πάρει ένα μη επισημασμένο παράδειγμα ως είσοδο, οπότε και είτε εξάγει απευθείας μια ετικέτα είτε εξάγει έναν αριθμό που μπορεί να χρησιμοποιηθεί από έναν αναλυτή για να συμπεράνει την ετικέτα.

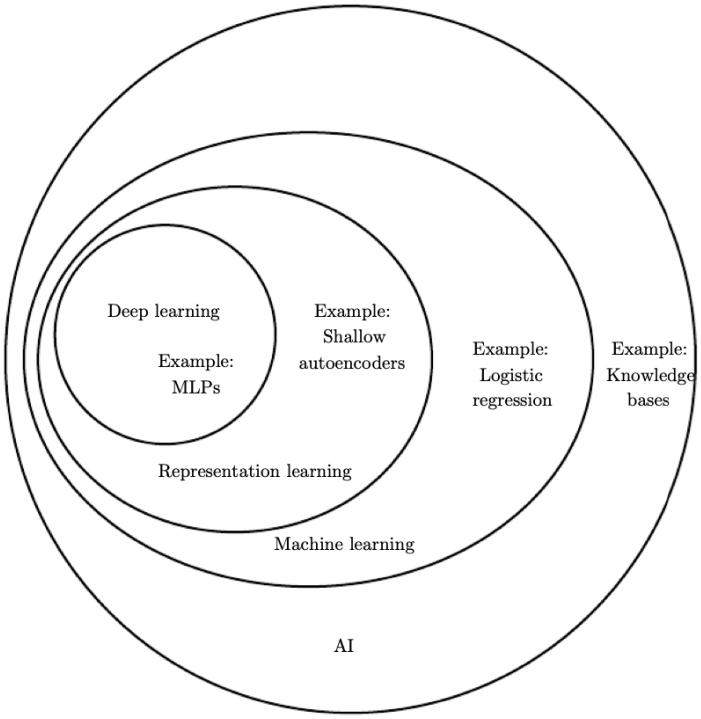
Σε ένα πρόβλημα ταξινόμησης, μια ετικέτα είναι μέλος ενός πεπερασμένου συνόλου τάξεων. Εάν το μέγεθος του συνόλου των τάξεων είναι δύο ("άρρωστο"/"υγιές", "ανεπιθύμητο"/"επιθυμητό"), μιλάμε για δυαδική ταξινόμηση (λέγεται και διωνυμική). Η ταξινόμηση πολλαπλών κατηγοριών είναι ένα πρόβλημα ταξινόμησης με τρεις ή περισσότερες κατηγορίες.

Ενώ μερικοί αλγόριθμοι μηχανικής μάθησης επιτρέπουν φυσικά περισσότερες από δύο τάξεις, άλλοι είναι από τη φύση τους δυαδικοί αλγόριθμοι ταξινόμησης. Υπάρχουν στρατηγικές που επιτρέπουν τη μετατροπή ενός αλγορίθμου εκμάθησης δυαδικής ταξινόμησης σε έναν πολλαπλών κατηγοριών.

Η παλινδρόμηση είναι ένα πρόβλημα πρόβλεψης μιας πραγματικής αξίας ετικέτας (συχνά ονομάζεται στόχος), δεδομένου ενός μη επισημασμένου παραδείγματος. Η εκτίμηση της τιμής της κατοικίας βάσει των χαρακτηριστικών του σπιτιού, όπως η περιοχή, ο αριθμός των υπνοδωματίων, η τοποθεσία και ούτω καθεξής είναι ένα διάσημο παράδειγμα παλινδρόμησης.

Το πρόβλημα παλινδρόμησης επιλύεται από έναν αλγόριθμο εκμάθησης παλινδρόμησης που παίρνει μια συλλογή από επισημασμένα παραδείγματα ως εισόδους και παράγει ένα μοντέλο που μπορεί να πάρει ένα μη επισημασμένο παράδειγμα ως είσοδο και έχουμε έξοδο ενός στόχου.

Έτσι, μια καλή αναπαράσταση του χώρου των AI, ML και DL θα ήταν η ακόλουθη:



Σχήμα 5. Τεχνητή νοημοσύνη, μηχανική εκμάθηση και βαθιά μάθηση [51]

Καθώς η πρόβλεψη πωλήσεων σχετίζεται άμεσα με τα προβλήματα παλινδρόμησης, εστιάζουμε συγκεκριμένα στην παλινδρόμηση. Στο Κεφάλαιο 2, παρουσιάζω μερικούς από τους πιο θεμελιώδεις αλγόριθμους παλινδρόμησης (Γραμμική παλινδρόμηση, Λογιστική παλινδρόμηση κ.λπ.) που θα μας βοηθήσουν να κατανοήσουμε σε βάθος τους αλγόριθμους που χρησιμοποιήσαμε στην εργασία μας για την πρόβλεψη πωλήσεων προϊόντων ταχείας κατανάλωσης (FMCGs).

Όσον αφορά τις προκλήσεις που αντιμετωπίζει κανείς για την πρόβλεψη πωλήσεων των FMCGs, θα χρησιμοποιήσουμε αλγόριθμους (θα εξηγηθούν λεπτομερώς στο Κεφάλαιο 4) με βάση τις

κύριες ιδέες δημιουργίας αλγορίθμων μηχανικής μάθησης, των επιλεγμένων μετρήσεων μας, και της αξιολόγησης αλγορίθμων βάση της υπάρχουσας έρευνας και βιβλιογραφίας.

Βαθειά Μάθηση (Deep Learning)

Οι απλοί αλγόριθμοι μηχανικής μάθησης που περιγράφονται στο Κεφάλαιο 4 λειτουργούν πολύ καλά σε μια μεγάλη ποικιλία σημαντικών προβλημάτων. Ωστόσο, δεν έχουν καταφέρει να επιλύσουν τα κεντρικά προβλήματα της τεχνητής νοημοσύνης, όπως η αναγνώριση ομιλίας ή η αναγνώριση αντικειμένων. Η ανάπτυξη της βαθιάς μάθησης οφείλεται εν μέρει στην αποτυχία των παραδοσιακών αλγορίθμων να γενικευθούν καλά σε τέτοιες εργασίες.

Αυτή η ενότητα αναφέρεται στο πώς η πρόκληση της γενίκευσης σε νέα παραδείγματα καθίσταται εκθετικά πιο δύσκολη όταν εργαζόμαστε με δεδομένα υψηλής διάστασης και πώς οι μηχανισμοί που χρησιμοποιούνται για την επίτευξη της γενίκευσης στην παραδοσιακή μηχανική μάθηση είναι ανεπαρκείς για να μάθουν περίπλοκες λειτουργίες σε χώρους υψηλών διαστάσεων. Τέτοιοι χώροι επιβάλλουν επίσης συχνά υψηλό υπολογιστικό κόστος. Η βαθιά μάθηση σχεδιάστηκε για να ξεπεράσει αυτά και άλλα εμπόδια.

Η βαθιά μάθηση είναι ένα συγκεκριμένο υπο-πεδίο της μηχανικής μάθησης: μια νέα ανάληψη μαθησιακών αναπαραστάσεων από δεδομένα που δίνει έμφαση στη μάθηση διαδοχικών επιπέδων ολοένα και πιο σημαντικών αναπαραστάσεων. Η σε βάθος μάθηση δεν αναφέρεται σε οποιοδήποτε είδος βαθύτερης κατανόησης που επιτυγχάνεται με μία αρχική προσέγγιση. Αντίθετα, ενσαρκώνει αυτή την ιδέα των διαδοχικών επιπέδων αναπαραστάσεων. Πόσα επίπεδα συμβάλλουν σε ένα μοντέλο των δεδομένων ονομάζεται βάθος του μοντέλου. Άλλα κατάλληλα ονόματα για το πεδίο θα μπορούσαν να ήταν η εκμάθηση σε στρώσεις και η εκμάθηση ιεραρχικών αναπαραστάσεων. Η σύγχρονη βαθιά μάθηση συχνά περιλαμβάνει δεκάδες ή ακόμα και εκατοντάδες διαδοχικά στρώματα αναπαραστάσεων και όλα μαθαίνουν αυτόματα από την έκθεση σε δεδομένα εκπαίδευσης.

Στη βαθιά μάθηση, αυτές οι πολυεπίπεδες αναπαραστάσεις μαθαίνονται (σχεδόν πάντα) μέσω μοντέλων που ονομάζονται νευρωνικά δίκτυα, δομημένα σε κυριολεκτικά στρώματα που στοιβάζονται το ένα πάνω στο άλλο. Ο όρος νευρωνικό δίκτυο είναι μια αναφορά στη νευροβιολογία, αλλά παρόλο που μερικές από τις κεντρικές έννοιες στη βαθιά μάθηση αναπτύχθηκαν εν μέρει αντλώντας έμπνευση από την κατανόηση μας για τον ανθρώπινο εγκέφαλο, τα μοντέλα βαθιάς μάθησης δεν είναι μοντέλα του εγκεφάλου. Δεν υπάρχουν στοιχεία ότι ο εγκέφαλος μοιάζει ή εφαρμόζει διεργασίες με τον ίδιο τρόπο με τους μηχανισμούς μάθησης στα σύγχρονα μοντέλα βαθιάς μάθησης. Παρόλο λοιπόν που πολλοί άνθρωποι πιστεύουν ότι τα νευρωνικά δίκτυα λειτουργούν σαν τον εγκέφαλο ή έχουν διαμορφωθεί σύμφωνα με τον ανθρώπινο εγκέφαλο, αυτό είναι αναληθές.

Στην πραγματικότητα, ένα νευρωνικό δίκτυο (NN), όπως και ένα μοντέλο παλινδρόμησης, είναι μια μαθηματική συνάρτηση: $y = f_{NN}(x)$. Η συνάρτηση f_{NN} έχει μια συγκεκριμένη μορφή: είναι μια σύνθετη συνάρτηση. Αυτές οι σύνθετες/εμφωλιασμένες συναρτήσεις σχηματίζουν τα επίπεδα του νευρωνικού δικτύου. Έτσι, για ένα νευρωνικό δίκτυο 3 επιπέδων που επιστρέφει μία βαθμίδα, η συνάρτηση f_{NN} μοιάζει ως εξής:

$$y = f_{NN}(x) = f_3(f_2(f_1(x)))$$

Στην παραπάνω εξίσωση, τα f_1 και f_2 είναι διανυσματικές συναρτήσεις της ακόλουθης μορφής:

$$\boldsymbol{f}_l(\mathbf{z}) \stackrel{\text{def}}{=} \boldsymbol{g}_l(\mathbf{W}_l\mathbf{z} + \mathbf{b}_l)$$

Η συνάρτηση g_l ονομάζεται συνάρτηση ενεργοποίησης. Είναι μια σταθερή, συνήθως μη γραμμική συνάρτηση που επιλέγεται από τον αναλυτή δεδομένων πριν ξεκινήσει η μάθηση. Οι

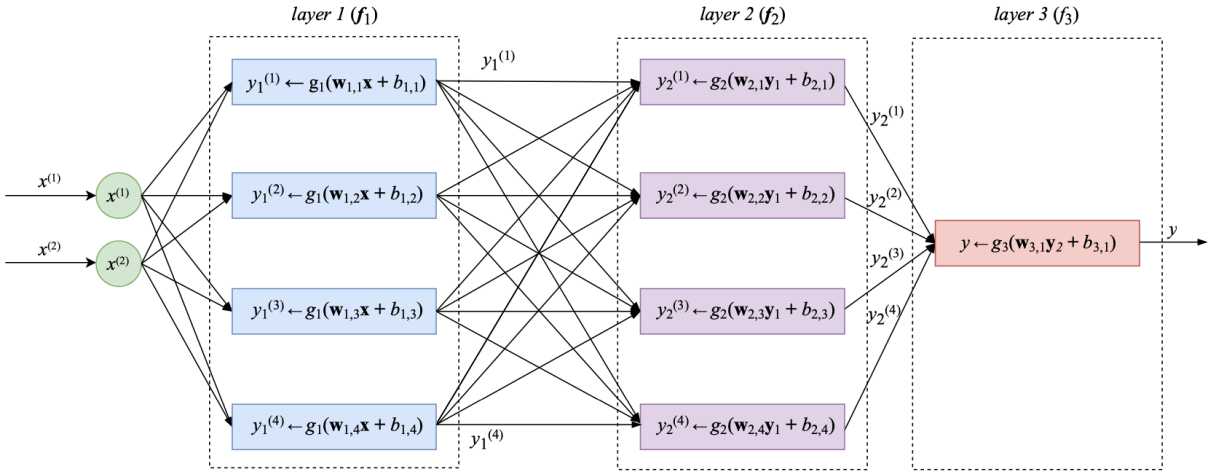
παράμετροι W_l (μήτρα) και b_l (φορέας) για κάθε στρώμα μαθαίνονται χρησιμοποιώντας τη γνωστή διαβάθμιση βελτιστοποιώντας, ανάλογα με την εργασία, μια συγκεκριμένη συνάρτηση κόστους (όπως MAPE, RMSE ή MSE).

Ο λόγος που χρησιμοποιείται ένας πίνακας W_l και όχι ένας φορέας w_l είναι ότι το g_l είναι μια διανυσματική συνάρτηση. Κάθε σειρά $w_{l,u}$ (u για μονάδα) του πίνακα W_l είναι ένας φορέας της ίδιας διάστασης με το z. Η έξοδος του $f_l(z)$ είναι ένας φορέας $[g_l(a_l,1), g_l(a_l,2),. . . , g_l(a_l,size_l)]$, όπου το g_l είναι κάποια βαθμίδα και το $size_l$ είναι ο αριθμός μονάδων στο επίπεδο l. Για να το κάνουμε πιο συγκεκριμένο, θα εξετάσουμε μια αρχιτεκτονική νευρωνικών δικτύων που ονομάζεται πολυεπίπεδη perceptron και συχνά αναφέρεται ως το πιο απλό νευρωνικό δίκτυο.

Για να το κάνουμε αυτό, θα εμβαθύνουμε σε μια συγκεκριμένη διαμόρφωση νευρωνικών δικτύων που ονομάζονται τροφοδοτικά νευρωνικά δίκτυα (FFNN), και πιο συγκεκριμένα στην αρχιτεκτονική που ονομάζεται multilayer perceptron (MLP). Για παράδειγμα, ας εξετάσουμε ένα MLP με τρία επίπεδα. Το δίκτυό μας λαμβάνει ένα δισδιάστατο διάνυσμα ως είσοδο και έξοδο έχει έναν αριθμό. Αυτό το FFNN μπορεί να είναι ένα μοντέλο παλινδρόμησης ή ταξινόμησης, ανάλογα με τη λειτουργία ενεργοποίησης που χρησιμοποιείται στο τρίτο επίπεδο εξόδου. Αυτό το MLP απεικονίζεται στο Σχήμα 6. Το νευρωνικό δίκτυο αντιπροσωπεύεται γραφικά ως συνδεδεμένος συνδυασμός μονάδων λογικά οργανωμένων σε ένα ή περισσότερα επίπεδα. Κάθε μονάδα αντιπροσωπεύεται είτε από έναν κύκλο είτε από ένα ορθογώνιο. Το εισερχόμενο βέλος αντιπροσωπεύει μια είσοδο μιας μονάδας και υποδεικνύει από πού προήλθε αυτή η είσοδος. Το εξερχόμενο βέλος υποδεικνύει την έξοδο μιας μονάδας.

Η έξοδος κάθε μονάδας είναι το αποτέλεσμα της μαθηματικής λειτουργίας που γράφτηκε μέσα στο ορθογώνιο. Οι μονάδες κύκλου δεν κάνουν τίποτα με την είσοδο, στέλνουν απλώς την είσοδο τους απευθείας στο αποτέλεσμα.

Τα ακόλουθα συμβαίνουν σε κάθε ορθογώνιο μονάδα. Πρώτον, όλες οι είσοδοι της μονάδας ενώνονται για να σχηματίσουν ένα φορέα εισόδου. Στη συνέχεια, η μονάδα εφαρμόζει έναν γραμμικό μετασχηματισμό στο φορέα εισόδου, όπως ακριβώς το μοντέλο γραμμικής παλινδρόμησης με το διάνυσμα χαρακτηριστικών εισόδου. Τέλος, η μονάδα εφαρμόζει μια συνάρτηση ενεργοποίησης g στο αποτέλεσμα του γραμμικού μετασχηματισμού και λαμβάνει την τιμή εξόδου, έναν πραγματικό αριθμό. Σε ένα απλό FFNN, η τιμή εξόδου μιας μονάδας κάποιου επιπέδου γίνεται με τιμή εισαγωγής καθεμιάς από τις μονάδες του επόμενου στρώματος. Συνήθως, όλες οι μονάδες ενός επιπέδου χρησιμοποιούν την ίδια λειτουργία ενεργοποίησης, αλλά δεν είναι κανόνας. Κάθε στρώμα μπορεί να έχει διαφορετικό αριθμό μονάδων. Κάθε μονάδα έχει τις παραμέτρους $w_{l,u}$ και $b_{l,u}$, όπου u είναι ο δείκτης της μονάδας και l είναι ο δείκτης του επιπέδου. Το διάνυσμα y_l σε κάθε μονάδα ορίζεται ως $[y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)}]$. Το διάνυσμα x στο πρώτο επίπεδο ορίζεται ως $[x^{(1)}, . . . , x^{(D)}]$.



Σχήμα 6: Ένα πολυεπίπεδο perceptron με δισδιάστατη είσοδο, δύο στρώματα με τέσσερις μονάδες και ένα επίπεδο εξόδου με μία μονάδα. [54]

Όπως μπορείτε να δείτε στο Σχήμα 6, σε πολυστρωματικό perceptron όλες οι έξοδοι ενός επιπέδου συνδέονται με κάθε είσοδο του επόμενου επιπέδου. Αυτή η αρχιτεκτονική ονομάζεται πλήρως συνδεδεμένη. Ένα νευρωνικό δίκτυο μπορεί να περιέχει πλήρως συνδεδεμένα επίπεδα. Αυτά είναι τα επίπεδα των οποίων οι μονάδες λαμβάνουν ως εισόδους τις εξόδους καθεμιάς από τις μονάδες του προηγούμενου επιπέδου.

Επίσης, ένα από τα δίκτυα βαθείας μάθησης που χρησιμοποιήσαμε στην έρευνα μας και στην δημιουργία αυτής της διπλωματικής είναι τα δίκτυα μακράς βραχυπρόθεσμης μνήμης (LSTM). Τα LSTM δίκτυα ξεκίνησαν ως πολύπλοκες λύσεις σε πολύ συγκεκριμένα προβλήματα που αφορούν μοτίβα, αλλά τα τελευταία χρόνια γίνονται όλο και πιο χρήσιμα σε ένα ευρύ φάσμα εφαρμογών. Ειδικά στην πρόβλεψη πωλήσεων φαίνονται πολύ υποσχόμενα καθώς είναι ένας τύπος τεχνητού νευρωνικού δικτύου που έχει σχεδιαστεί για να αναγνωρίζει μοτίβα σε ακολουθίες δεδομένων, όπως αριθμητικά δεδομένα χρονοσειρών. Για αυτόν τον λόγο, εκπαιδεύσαμε ένα δίκτυο LSTM για προβλέψεις πωλήσεων για FMCGs. Αυτό που ανακαλύψαμε είναι ότι, όπως όλα τα πιο προηγμένα μοντέλα, κάνει over-fit πολύ γρήγορα. Η ακρίβεια της πρόβλεψης που παρείχε ήταν πολύ καλή, αλλά δεν άξιζε τον επιπλέον χρόνο και την απαραίτητα εξειδικευμένη γνώση ανάπτυξης τους.

0.4 Αποτελέσματα

Σε αυτήν την ενότητα παρουσιάζουμε τα αποτελέσματα και την ακρίβεια των μοντέλων μας μαζί με τους σχετικούς πίνακες και γραφήματα για την ευκολότερη σύγκριση τους. Επί των αποτελεσμάτων μας θα βασιστούν τα συμπεράσματα μας, αλλά και οι πιθανές μελλοντικές κατευθύνσεις της συγκεκριμένης εργασίας. Για το πως καταλήξαμε σε αυτά, δείτε το Κεφάλαιο 5 (“Πειράματα”).

Από τα πειράματα μας, διαπιστώσαμε ότι τα 4 κορυφαία μοντέλα μηχανικής μάθησης για την πρόβλεψη πωλήσεων στις κατηγορίες προϊόντων ταχείας κατανάλωσης είναι τα: 1. CatBoost, 2. LightGBM, 3. XGBoost, 4. HuberRegression. Το καλύτερο μέτα-μοντέλο μας είναι αυτό που συνδυάζει τα δύο καλύτερα μοντέλα μας (2-best Stacking Regressor), αφού συνδυάζει υψηλή ακρίβεια προβλέψεων, τεράστια ευελιξία και χαμηλό χρόνο εκπαίδευσης, καθιστώντας το εξαιρετική επιλογή για λύσεις πρόβλεψης πωλήσεων στον πραγματικό κόσμο.

Αυτά είναι τα αποτελέσματα και τα χαρακτηριστικά καθενός από τα μοντέλα μας:

	MAPE (%) (Standard)	MAPE (%) (Normal)	RMSE (Standard)	RMSE (Normal)	MSE (Standard)	MSE (Normal)	Execution Time (sec) (Standard)	Execution Time (sec) (Normal)	Best Parameters (Standard)	Best Parameters (Normal)
HuberRegression	17,413	19,654	3,596	3,516	12,929	12,362	895	1512	{'epsilon': 1.4}	{'epsilon': 1.4}
KNNReg	45,208	51,240	3,192	3,331	10,190	11,097	19542	19722	{'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'distance'}	{'metric': 'manhattan', 'n_neighbors': 15, 'weights': 'distance'}
Pass/AggReg	57,822	64,591	3,590	3,525	12,891	12,425	7	10	{'C': 0.1}	{'C': 0.1}
LassoRegression	89,261	89,261	4,033	4,033	16,264	16,264	1	1	{'selection': 'random'}	{'selection': 'random'}
RidgeRegression	62,120	57,160	3,496	3,288	12,221	10,809	1	1	{'alpha': 0.2}	{'alpha': 0.05}
XGBoost	4,580	5,454	0,494	0,822	0,244	0,676	1634	1792	{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 6, 'min_child_weight': 4, 'n_estimators': 300, 'nthread': 4, 'objective': 'reg:squarederror', 'silent': 1, 'subsample': 0.7}	{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 5, 'min_child_weight': 4, 'n_estimators': 300, 'nthread': 4, 'objective': 'reg:squarederror', 'silent': 1, 'subsample': 0.7}
RandomForestReg	53,246	42,092	2,061	1,986	4,248	3,945	727	727	{'max_depth': 6, 'max_features': 'sqrt', 'min_impurity_decrease': 0.001, 'n_estimators': 150}	{'max_depth': 6, 'max_features': 'sqrt', 'min_impurity_decrease': 0.001, 'n_estimators': 150}
CatBoost	2,486	1,816	0,429	0,566	0,184	0,321	133	178	estimator=catb_reg, param_grid=param_grid, cv= 5, n_jobs=-1, verbose=0	estimator=catb_reg, param_grid=param_grid, cv= 5, n_jobs=-1, verbose=0
LightGBM	2,981	3,075	0,514	0,655	0,264	0,429	17	15	{'boosting_type': 'goss'}	{'boosting_type': 'goss'}
2-best SC	2,000	1,643	0,373	0,485	0,139	0,235	153	187	model1 = lightgbm.LGBMRegressor (boosting_type = 'goss') model2 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('lightgbm', model1), ('catboost', model2)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)	model1 = lightgbm.LGBMRegressor (boosting_type = 'goss') model2 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('lightgbm', model1), ('catboost', model2)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)
3-best SC	1,990	1,622	0,358	0,467	0,128	0,218	191	249	model1 = xgboost.XGBRegressor (objective = 'reg: squarederror', colsample_bytree = 0.7, learning_rate = 0.03, max_depth = 5, min_child_weight = 4, n_estimators = 300, nthread = 4, silent = 1, subsample = 0.7) model2 = lightgbm.LGBMRegressor (boosting_type = 'goss') model3 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('xgb_reg', model1), ('lightgbm', model2), ('catboost', model3)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)	model1 = xgboost.XGBRegressor (objective = 'reg: squarederror', colsample_bytree = 0.7, learning_rate = 0.03, max_depth = 5, min_child_weight = 4, n_estimators = 300, nthread = 4, silent = 1, subsample = 0.7) model2 = lightgbm.LGBMRegressor (boosting_type = 'goss') model3 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('xgb_reg', model1), ('lightgbm', model2), ('catboost', model3)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)
MoE	24,440	24,264	4,228	4,228	17,878	17,877	850	830	model = MoE(8, 1)	model = MoE(8, 1)
KerasRegressor	38,093	38,093					628	653	model = Sequential (), model.add (Dense (128, input_dim = X_train.shape [1], activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (256, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (128, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (1)), opt = Nadam (1e-2), model.compile (optimizer = opt, loss = 'mean_squared_error')	model = Sequential (), model.add (Dense (128, input_dim = X_train.shape [1], activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (256, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (128, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (1)), opt = Nadam (1e-2), model.compile (optimizer = opt, loss = 'mean_squared_error')
MLP	39,073	34,273	2,100	2,470	4,408	6,103	2011	1772	mlp_reg = MLPRegressor (), param_grid = { 'hidden_layer_sizes': [(100,), (20, 20), (256, 256, 128)], 'activation': ['logistic', 'tanh']} reg_cv = GridSearchCV (estimator = mlp_reg, param_grid = param_grid, cv = 5, n_jobs = -1)	mlp_reg = MLPRegressor (), param_grid = { 'hidden_layer_sizes': [(100,), (20, 20), (256, 256, 128)], 'activation': ['logistic', 'tanh']} reg_cv = GridSearchCV (estimator = mlp_reg, param_grid = param_grid, cv = 5, n_jobs = -1)
LSTM	24,091	24,090	4,231	4,231	17,899	17,899	818	847	loss_function = nn.L1Loss (), optimizer = torch.optim.Adam (model.parameters (), lr = 0.001), epochs = 100, best_mse = np.inf, model = LSTM (X_train_norm.shape [1], 100) , sss_norm = ShuffleSplit (n_splits = 1, test_size = 0.1)	loss_function = nn.L1Loss (), optimizer = torch.optim.Adam (model.parameters (), lr = 0.001), epochs = 100, best_mse = np.inf, model = LSTM (X_train_norm.shape [1], 100) , sss_norm = ShuffleSplit (n_splits = 1, test_size = 0.1)

Και για να δούμε καλύτερα την εκπαίδευση των μοντέλων μας και τις καλύτερες παραμέτρους τους:

	Best Parameters (Standard)	Best Parameters (Normal)
HuberRegression	{'epsilon': 1.4}	{'epsilon': 1.4}
KNNReg	{'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'distance'}	{'metric': 'manhattan', 'n_neighbors': 15, 'weights': 'distance'}
Pass/AggReg	{'C': 0.1}	{'C': 0.1}
LassoRegression	{'selection': 'random'}	{'selection': 'random'}
RidgeRegression	{'alpha': 0.2}	{'alpha': 0.05}
XGBoost	{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 6, 'min_child_weight': 4, 'n_estimators': 300, 'nthread': 4, 'objective': 'reg:squarederror', 'silent': 1, 'subsample': 0.7}	{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 5, 'min_child_weight': 4, 'n_estimators': 300, 'nthread': 4, 'objective': 'reg:squarederror', 'silent': 1, 'subsample': 0.7}
RandomForestReg	{'max_depth': 6, 'max_features': 'sqrt', 'min_impurity_decrease': 0.001, 'n_estimators': 150}	{'max_depth': 6, 'max_features': 'sqrt', 'min_impurity_decrease': 0.001, 'n_estimators': 150}
CatBoost	estimator=catb_reg, param_grid=param_grid, cv= 5, n_jobs=-1, verbose=0	estimator=catb_reg, param_grid=param_grid, cv= 5, n_jobs=-1, verbose=0
LightGBM	{'boosting_type': 'goss'}	{'boosting_type': 'goss'}
2-best SC	model1 = lightgbm.LGBMRegressor (boosting_type = 'goss') model2 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('lightgbm', model1), ('catboost', model2)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)	model1 = lightgbm.LGBMRegressor (boosting_type = 'goss') model2 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('lightgbm', model1), ('catboost', model2)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)
3-best SC	model1 = xgboost.XGBRegressor (objective = 'reg: squarederror', colsample_bytree = 0.7, learning_rate = 0.03, max_depth = 5, min_child_weight = 4, n_estimators = 300, nthread = 4, silent = 1, subsample = 0.7) model2 = lightgbm.LGBMRegressor (boosting_type = 'goss') model3 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('xgb_reg', model1), ('lightgbm', model2), ('catboost', model3)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)	model1 = xgboost.XGBRegressor (objective = 'reg: squarederror', colsample_bytree = 0.7, learning_rate = 0.03, max_depth = 5, min_child_weight = 4, n_estimators = 300, nthread = 4, silent = 1, subsample = 0.7) model2 = lightgbm.LGBMRegressor (boosting_type = 'goss') model3 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('xgb_reg', model1), ('lightgbm', model2), ('catboost', model3)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)
MoE	model = MoE(8, 1)	model = MoE(8, 1)
KerasRegressor	model = Sequential (), model.add (Dense (128, input_dim = X_train.shape [1], activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (256, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (128, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (1)), opt = Nadam (1e-2), model.compile (optimizer = opt, loss = 'mean_squared_error')	model = Sequential (), model.add (Dense (128, input_dim = X_train.shape [1], activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (256, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (128, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (1)), opt = Nadam (1e-2), model.compile (optimizer = opt, loss = 'mean_squared_error')
MLP	mlp_reg = MLPRegressor (), param_grid = { 'hidden_layer_sizes': [(100,), (20, 20), (256, 256, 128)], 'activation': ['logistic', 'tanh']} reg_cv = GridSearchCV (estimator = mlp_reg, param_grid = param_grid, cv = 5, n_jobs = -1)	mlp_reg = MLPRegressor (), param_grid = { 'hidden_layer_sizes': [(100,), (20, 20), (256, 256, 128)], 'activation': ['logistic', 'tanh']} reg_cv = GridSearchCV (estimator = mlp_reg, param_grid = param_grid, cv = 5, n_jobs = -1)
LSTM	loss_function = nn.L1Loss (), optimizer = torch.optim.Adam (model.parameters (), lr = 0.001), epochs = 100, best_mse = np.inf, model = LSTM (X_train_norm.shape [1], 100) , sss_norm = ShuffleSplit (n_splits = 1, test_size = 0.1)	loss_function = nn.L1Loss (), optimizer = torch.optim.Adam (model.parameters (), lr = 0.001), epochs = 100, best_mse = np.inf, model = LSTM (X_train_norm.shape [1], 100) , sss_norm = ShuffleSplit (n_splits = 1, test_size = 0.1)

Απλά μοντέλα παλινδρόμησης (simple Regression models):

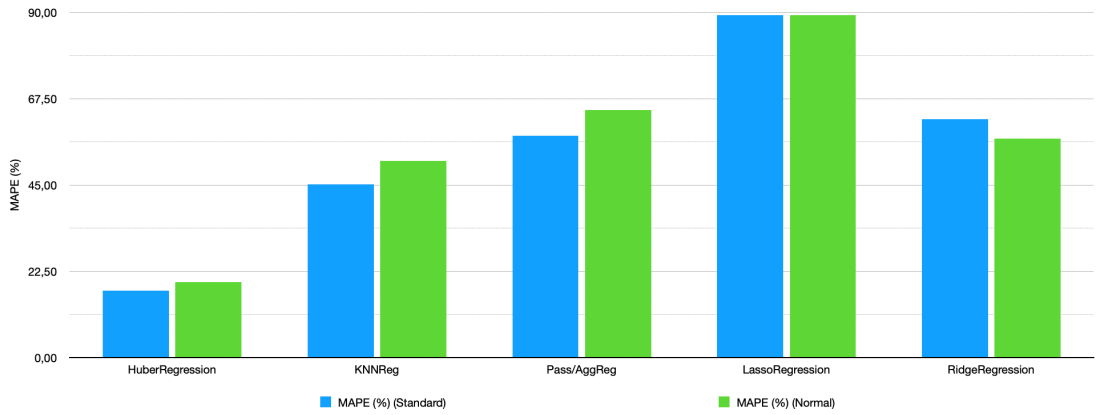
Τα μοντέλα που καλύπτουμε εδώ είναι:

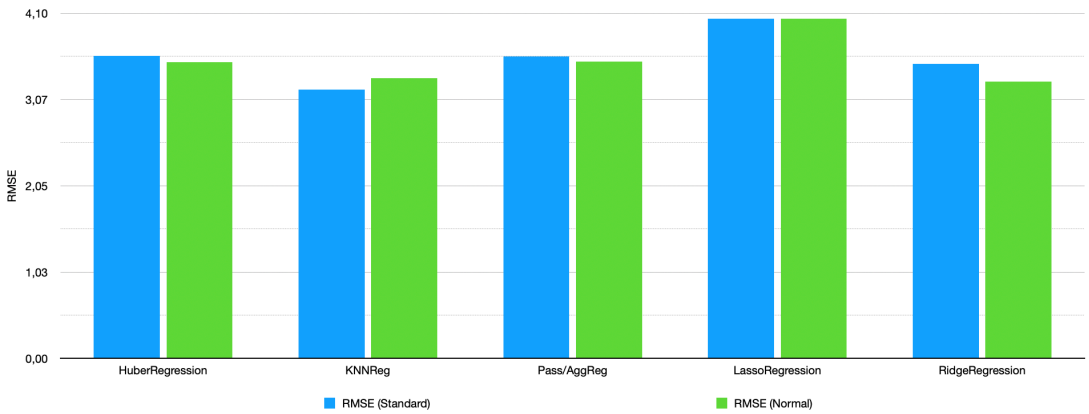
- HuberRegression
- KNN Regressor
- Passive Aggressive Regressor
- Lasso Regression
- Ridge Regression

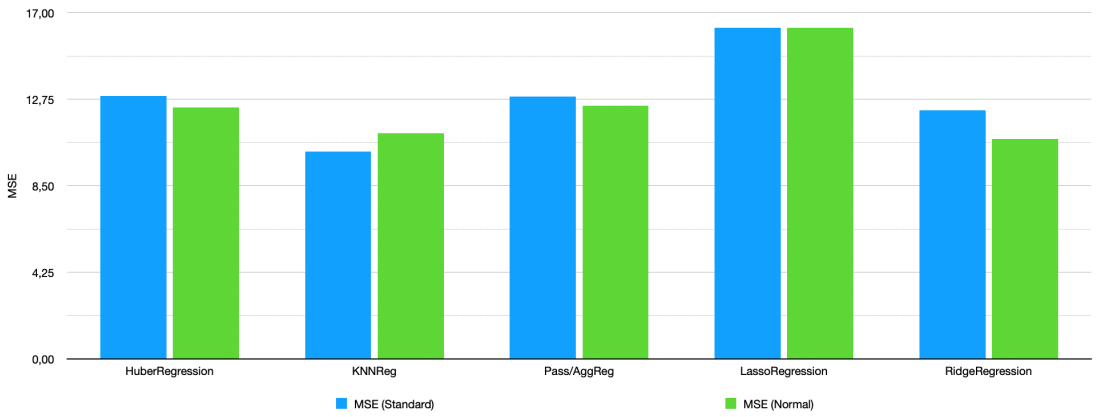
Και έχουμε τα ακόλουθα αποτελέσματα:

	MAPE (%) (Standard)	MAPE (%) (Normal)	RMSE (Standard)	RMSE (Normal)	MSE (Standard)	MSE (Normal)	Execution Time (sec) (Standard)	Execution Time (sec) (Normal)
HuberRegression	17,41	19,65	3,60	3,52	12,93	12,36	895	1512
KNNReg	45,21	51,24	3,19	3,33	10,19	11,10	19542	19722
Pass/AggReg	57,82	64,59	3,59	3,53	12,89	12,43	7	10
LassoRegression	89,26	89,26	4,03	4,03	16,26	16,26	1	1
RidgeRegression	62,12	57,16	3,50	3,29	12,22	10,81	1	1

Και όπως βλέπουμε και από τις γραφικές παραστάσεις:







- Δεν υπάρχουν σταθερές διαφορές μεταξύ του τυποποιημένου (standardized) συνόλου δεδομένων και του κανονικοποιημένου (normalized) μας. Ορισμένα μοντέλα επιτυγχάνουν καλύτερη ακρίβεια σε ένα από αυτά, αλλά ακόμη και αυτά βλέπουμε ότι για διαφορετικές μετρήσεις-δείκτες αλλάζουν την προτιμώμενη μέθοδο κλιμάκωσης χαρακτηριστικών.
- Το HuberRegression μοντέλο έχει μακράν το χαμηλότερο MAPE, αλλά χάνει σε άλλες βασικές μετρήσεις.
- Με βάση τις ανάγκες της βιομηχανίας και τι γενικά θεωρείται χρήσιμο από την βιομηχανία, MAPEs >20% δεν είναι αρκετά καλά για FMCGs. Έτσι, από απλούς παλινδρόμους, μόνο το HuberRegression περνά το αποδεκτό όριο και αξίζει να δοκιμαστεί από τον επιχειρησιακό σχεδιασμό πρόβλεψης πωλήσεων των διαφόρων οργανισμών.

Μοντέλα ενίσχυσης κλίσης (GBDT models):

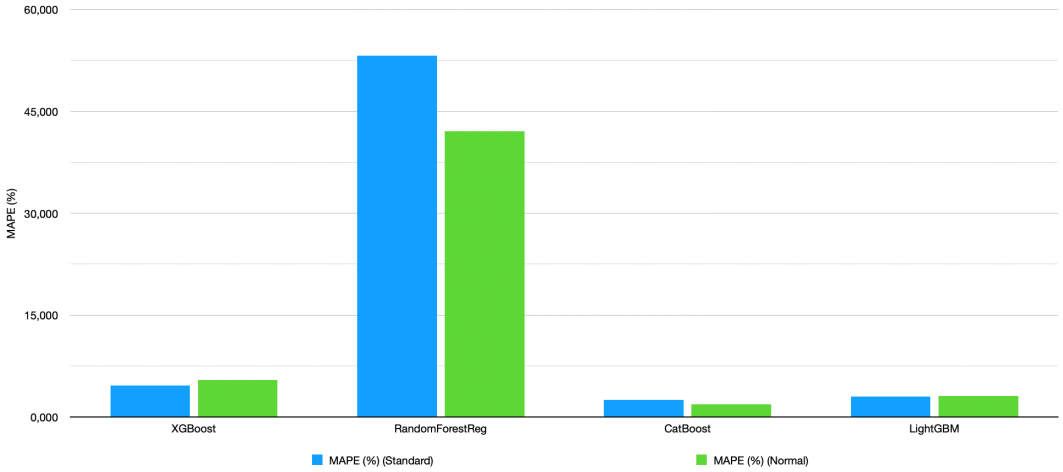
Τα μοντέλα που καλύπτουμε εδώ είναι:

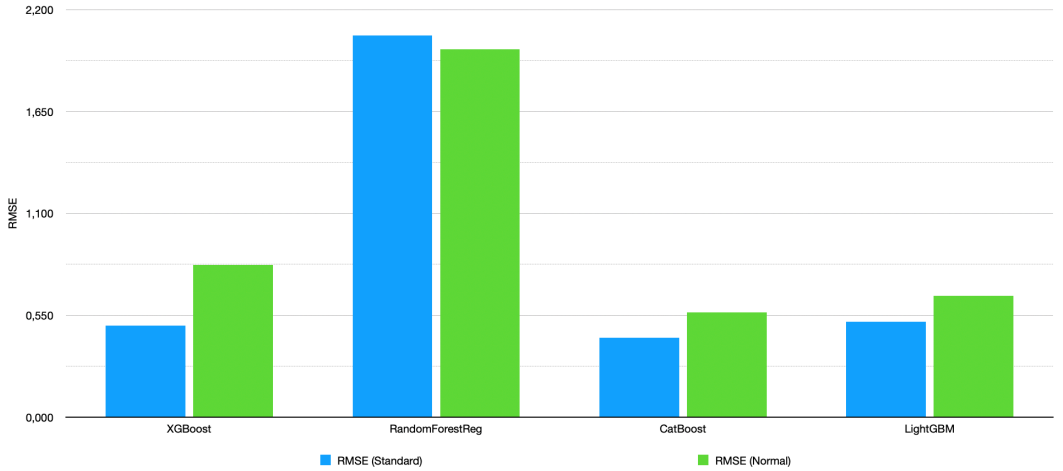
- XGBoost
- Random Forest Regressor
- CatBoost
- LightGBM

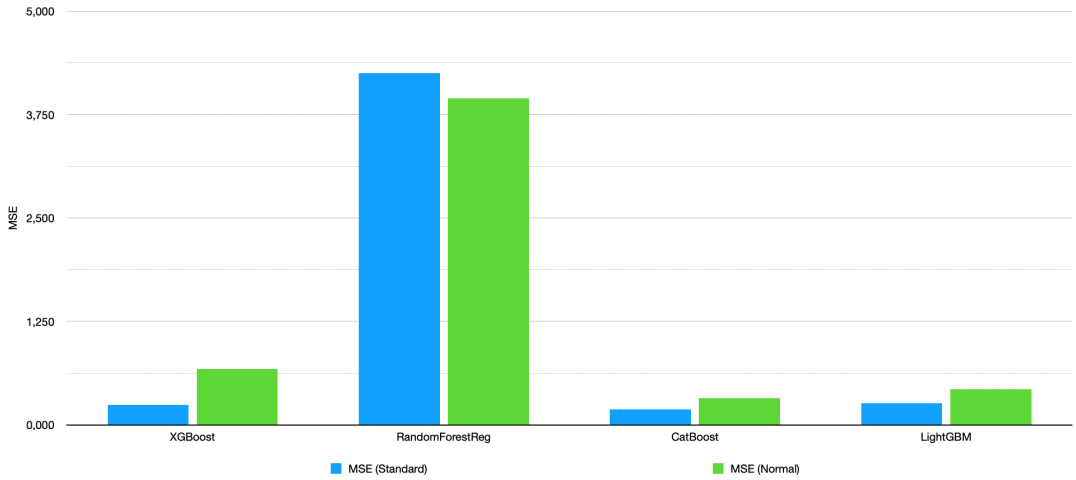
Και έχουμε τα ακόλουθα αποτελέσματα:

	MAPE (%) (Standard)	MAPE (%) (Normal)	RMSE (Standard)	RMSE (Normal)	MSE (Standard)	MSE (Normal)	Execution Time (sec) (Standard)	Execution Time (sec) (Normal)
XGBoost	4,580	5,454	0,494	0,822	0,244	0,676	1634	1792
RandomForestReg	53,246	42,092	2,061	1,986	4,248	3,945	727	727
CatBoost	2,486	1,816	0,429	0,566	0,184	0,321	133	178
LightGBM	2,981	3,075	0,514	0,655	0,264	0,429	17	15

Και όπως βλέπουμε και από τις γραφικές παραστάσεις:







- Το τυποποιημένο σύνολο δεδομένων αποδίδει λίγο καλύτερα από το κανονικοποιημένο στα GBDT μοντέλα. Ειδικά στο XGBoost η διαφορά είναι σημαντική.

- Τα XGBoost, CatBoost και LightGBM έχουν εξαιρετική απόδοση σε όλες τις μετρήσεις. Με ένα MAPE 2.5-5%, τα GBDT μοντέλα δείχνουν γιατί πρέπει να βρίσκονται σε διαφορετική κατηγορία από τα απλά μοντέλα παλινδρόμησης, αλλά και γιατί είναι τα πιο πολυχρησιμοποιούμενα από την βιομηχανία.
- Το RandomForest Regressor είναι το μόνο GBDT μοντέλο με χαμηλή ακρίβεια πρόβλεψης. Αν και είναι πολύ απλό στη χρήση και ενδεχομένως καλό για συγκριτική αξιολόγηση (benchmarking), δεν φαίνεται κατάλληλο για τις προβλέψεις πωλήσεων FMCGs.

Μοντέλα Βαθειάς Μάθησης (Deep Learning models):

Τα μοντέλα που καλύπτουμε εδώ είναι:

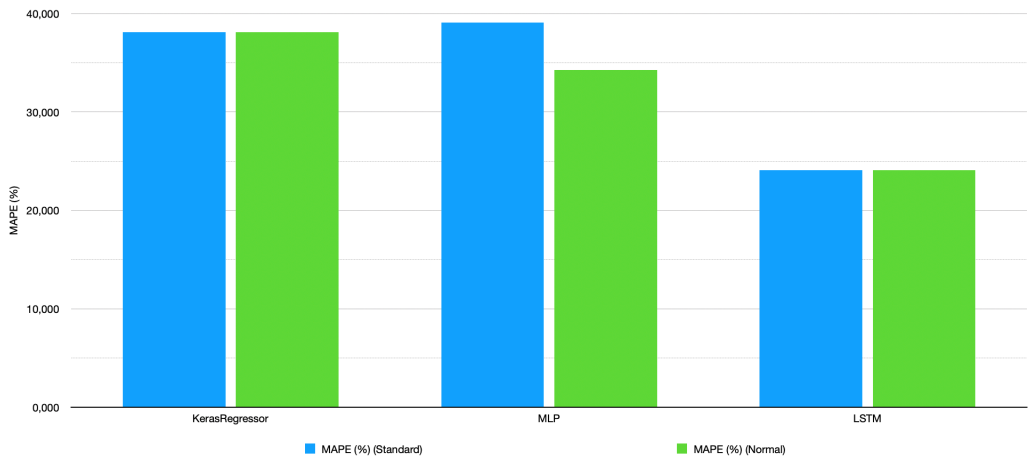
- KerasRegressor
- MLP
- LSTM

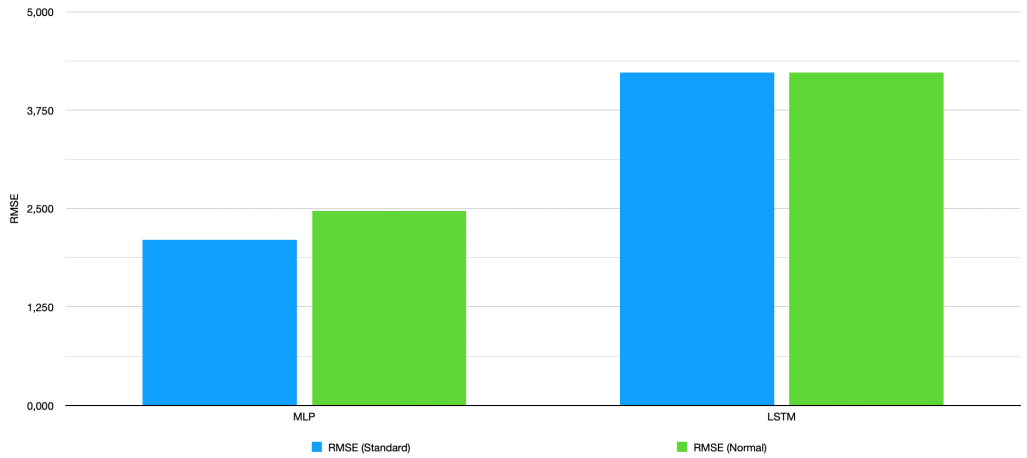
Και έχουμε τα ακόλουθα αποτελέσματα:

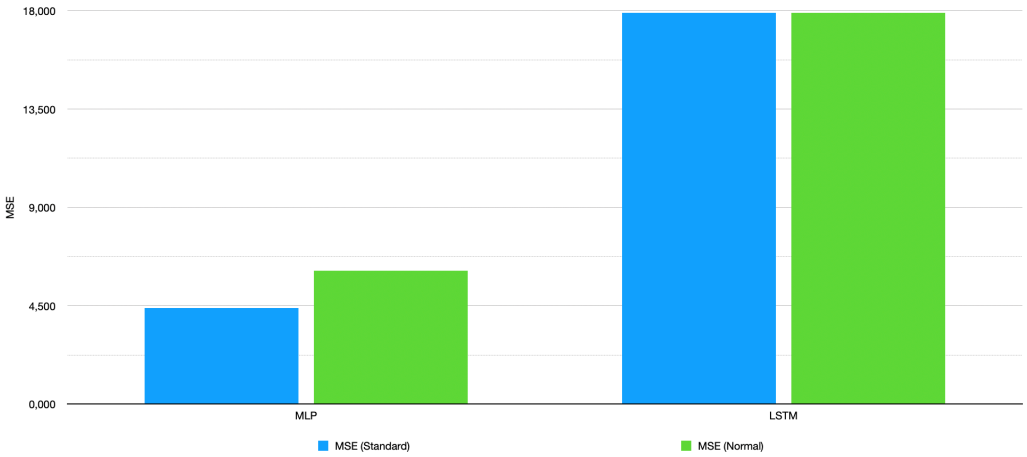
	MAPE (%) (Standard)	MAPE (%) (Normal)	RMSE (Standard)	RMSE (Normal)	MSE (Standard)	MSE (Normal)	Execution Time (sec) (Standard)	Execution Time (sec) (Normal)
KerasRegressor	38,093	38,093					628	653
MLP	39,073	34,273	2,100	2,470	4,408	6,103	2011	1772
LSTM	24,091	24,090	4,231	4,231	17,899	17,899	818	847

Μετρήσαμε μόνο το MAPE για το KerasRegressor για να το χρησιμοποιήσουμε μόνο ως σημείο αναφοράς DL.

Και όπως βλέπουμε και από τις γραφικές παραστάσεις:







- Δεν υπάρχουν σταθερές διαφορές μεταξύ του τυποποιημένου (standardized) συνόλου δεδομένων και του κανονικοποιημένου (normalized) μας. Ορισμένα μοντέλα επιτυγχάνουν καλύτερη ακρίβεια σε ένα από αυτά, αλλά ακόμη και αυτά βλέπουμε ότι για διαφορετικές μετρήσεις-δείκτες αλλάζουν την προτιμώμενη μέθοδο κλιμάκωσης χαρακτηριστικών.
- Ενώ το MLP έχει χαμηλότερο RMSE και MSE, έχει μεγαλύτερο MAPE από το LSTM.
- Με βάση τα βιομηχανικά πρότυπα, τόσο τα δίκτυα MLP όσο και τα δίκτυα LSTM θα μπορούσαν να χρησιμοποιηθούν για την πρόβλεψη πωλήσεων FMCGs.
- Σε σύγκριση με τους απλούς παλινδρόμους, τα μοντέλα Deep Learning δίνουν ακριβέστερα, αλλά όχι απαραίτητα καλύτερα αποτελέσματα από τα υπόλοιπα μοντέλα καθώς είναι πιο χρονοβόρα, πολύ πιο απαιτητικά στην ανάλυση και ποιότητα των δεδομένων μας και συνολικά χειρότερα από αρκετά μοντέλα μηχανικής μάθησης.

Επιπλέον αποτελέσματα και συμπεράσματα για τα μοντέλα Βαθείας μάθησης:

Για τις επιχειρήσεις που θέλουν να εφαρμόσουν λύσεις βαθιάς μάθησης, στα πειράματα μας (Κεφάλαιο 5) μπορούμε να δούμε ότι πολλά μοντέλα μηχανικής μάθησης είναι καλύτερα από τις λύσεις με χρήση μοντέλων βαθείας μάθησης.

Το τελικό αποτέλεσμα δείχνει ότι ακόμη και οι προσεγγίσεις νευρωνικών δικτύων δεν είναι σε θέση να παρέχουν βελτιώσεις σε σχέση με τα μοντέλα μηχανικής μάθησης για τις περιπτώσεις των προβλέψεων πωλήσεων προϊόντων ταχείας κατανάλωσης. Αυτό οφείλεται στα χαρακτηριστικά των δεδομένων του προβλήματος, όπως είναι η χαμηλή διάσταση (low

dimensionality) , τα οποία μπορούν να οδηγήσουν σε γρήγορη υπερφόρτωση (overfitting) τα πολύ ευέλικτα μοντέλα. Επίσης, το μέγεθος εισόδου μπορεί να επηρεάσει την εξαγωγή χαρακτηριστικών στο μοντέλο LSTM. Για αυτόν τον λόγο, πρέπει να γίνουν περισσότερες δοκιμές σε διαφορετικά σύνολα δεδομένων, αλλά καθώς τα σύνολα δεδομένων μας (τόσο από τις επιχειρήσεις/marketplaces όσο και από τον Kaggle διαγωνισμό) είναι εξαιρετικές αναπαραστάσεις και εκπρόσωποι των συνόλων δεδομένων του πραγματικού κόσμου, που χρησιμοποιούνται για προβλέψεις πραγματικών επιχειρησιακών πωλήσεων, τα αποτελέσματά μας είναι ως όσο το δυνατόν πιο κοντά στην πραγματικότητα.

Ωστόσο, μεταξύ των διαφορετικών νευρωνικών δικτύων, μπορεί να παρατηρηθεί ότι το LSTM παρείχε τα καλύτερα αποτελέσματα. Το LSTM και το MoE παρουσίασαν παρόμοια αποτελέσματα σε όλες τις μετρήσεις. Το μοντέλο Mixture of Experts σημείωσε βελτίωση σε σχέση με το μοντέλο MLP, το οποίο αναμένεται λόγω της ομοιότητας της αρχιτεκτονικής.

Για σύγκριση, όπως αναφέραμε και παραπάνω, μεταξύ των μοντέλων μηχανικής μάθησης το καλύτερο αποτέλεσμα επιτεύχθηκε με τα μοντέλα XGBoost, LGBM και CatBoost, καθώς και τα μετα-μοντέλα που βασίζονται σε αυτά. Δεδομένου ότι αυτά τα μοντέλα φέρουν δύο βασικούς παράγοντες που απαιτούνται για ένα σύνολο μοντέλων (υψηλή απόδοση και χαμηλή συμφωνία), το τελικό μοντέλο προσφέρει βελτίωση σε σχέση με τα μεμονωμένα αποτελέσματα του κάθε μοντέλου. Αυτό συμβαίνει λόγω της επεκτασιμότητάς του και της ακριβούς εφαρμογής της ενίσχυσης της κλίσης, καθώς δημιουργήθηκε με σκοπό τη βελτιστοποίηση τόσο της απόδοσης όσο και της υπολογιστικής πολυπλοκότητας. Επίσης, το HuberRegression οδηγεί σε μια πολύ καλή πρόβλεψη, η οποία, με βάση την ανάλυση των δεδομένων μας, συμβαίνει λόγω της ευαισθησίας του αλγορίθμου στα ακραία σημεία (outliers).

- Όλα τα μοντέλα Deep Learning (KerasRegressor, MLP, LSTM) θα μπορούσαν να χρησιμοποιηθούν για την πρόβλεψη πωλήσεων FMCGs.
- Σε σύγκριση με τους απλούς παλινδρόμους, τα μοντέλα Deep Learning δίνουν καλύτερα αποτελέσματα, αλλά όχι απαραίτητα καλύτερα αποτελέσματα από τα μοντέλα GBDT καθώς γρήγορα υπερφορτώνονται. Δυνητικά θα μπορούσαν να επιτύχουν ακόμα καλύτερη εκπαίδευση και επιδόσεις αν είχαμε σύνολα δεδομένων υψηλότερων διαστάσεων, αλλά καθώς το πρόβλημα της πρόβλεψης πωλήσεων χρησιμοποιεί συμπαγή σύνολα δεδομένων με χαμηλή διάσταση, οι εκτελέσεις βαθείας μάθησης αποδίδουν χειρότερα ή οριακά ισάξια με τα GBDT μοντέλα σχεδόν σε όλες τις περιπτώσεις.
- Όπως περιμέναμε, όλα τα μοντέλα Deep Learning έχουν σημαντικό χρόνο εκτέλεσης. Ενώ ο χρόνος εκτέλεσης δεν αποτελεί πρόβλημα για αυτήν την περίπτωση προβλημάτων (πρόβλεψη πωλήσεων), μπορεί να επηρεάσει σοβαρά το κόστος της αρχικής επένδυσης, καθώς επηρεάζει τον απαιτούμενο χρόνο εργασίας, το οποίο στην περίπτωση των μικρομεσαίων επιχειρήσεων μπορεί να είναι υψηλής σημασίας. Η χρήση επιπλέον υπολογιστικών πόρων σίγουρα μειώνει τον χρόνο εκτέλεσης, αλλά με υψηλότερο κόστος λειτουργίας / επεξεργασίας.
- Το μοντέλο MLP εκτελεί αναζήτηση πλέγματος που αλλάζει τη λειτουργία ενεργοποίησης καθώς και τις ρυθμίσεις των κρυφών επιπέδων.
- Όπως επισημαίνουμε παραπάνω, το απλούστερο μοντέλο επιλέγεται πάντα στο GridSearch. Για παράδειγμα, το MLP μας σχεδιάστηκε για να δοκιμάσει ένα 1-hidden επίπεδο με 100 νευρώνες, ένα 2-hidden επίπεδο με 20 νευρώνες το καθένα και 3-hidden επίπεδο με 256, 256 και 128 νευρώνες (η ίδια ακριβώς προσέγγιση που χρησιμοποιήσαμε για το KerasRegressor). Στα πειράματα μας, ο αλγόριθμος επέλεξε να χρησιμοποιήσει την απλούστερη εφαρμογή, δηλαδή το 1-hidden επίπεδο με 100 νευρώνες.
- Το LSTM που δημιουργήσαμε είναι ένα LSTM ενός επιπέδου που εξαγάγει χαρακτηριστικά σε ένα πλήρως συνδεδεμένο επίπεδο.
- Το μοντέλο LSTM συμπεριφέρθηκε ακριβώς όπως προβλέψαμε, δηλαδή πολύ γρήγορα υπερφορτώθηκε. Είδαμε ότι ακόμη και οι απλούστερες αρχιτεκτονικές είχαν πρόβλημα με την υπερφόρτωση/overfitting λόγω των μικρών διαστάσεων (dimensionality). Παίρνουμε πολύ χαμηλό σκορ στην επικύρωση των μοντέλων βαθείας μάθησης, λοιπόν, ακόμη και όταν εφαρμόζονται τέλεια. Ένα τόσο ισχυρό, μη στατικό μοντέλο, όπως το LSTM, είναι πολύ

ισχυρό και, επομένως λογικό, να προσαρμόζεται υπερβολικά και υπερφορτώνεται ακόμη και με κανονικοποίηση.

Μοντέλα Μετα-μάθησης (Meta-learning models):

Τέλος, χρησιμοποιούμε τις μεθοδολογίες κατασκευής μετα-μοντέλων που αναφέραμε. Όπως θα αναφέρουμε στα Κεφάλαιο 2 («Θεωρητικό Ιστορικό»), Κεφάλαιο 3 («Προηγούμενες Εργασίες») και Κεφάλαιο 4 («Μέθοδος και Μοντέλα»), το Ensemble Learning και η κατασκευή μετα-μοντέλων παίζει τεράστιο ρόλο στην αύξηση της ακρίβειας άλλων, απλούστερων, μοντέλων. Για αυτόν τον λόγο, αφού εκτελέσουμε τα πειράματα 1ου Σταδίου με τους απλούς παλινδρόμους, τα μοντέλα GBDT και τα μοντέλα Deep Learning, προγραμματίσαμε τα σχετικά μετα-μοντέλα στο Στάδιο 2 και εδώ παρουσιάζουμε τα αποτελέσματά τους.

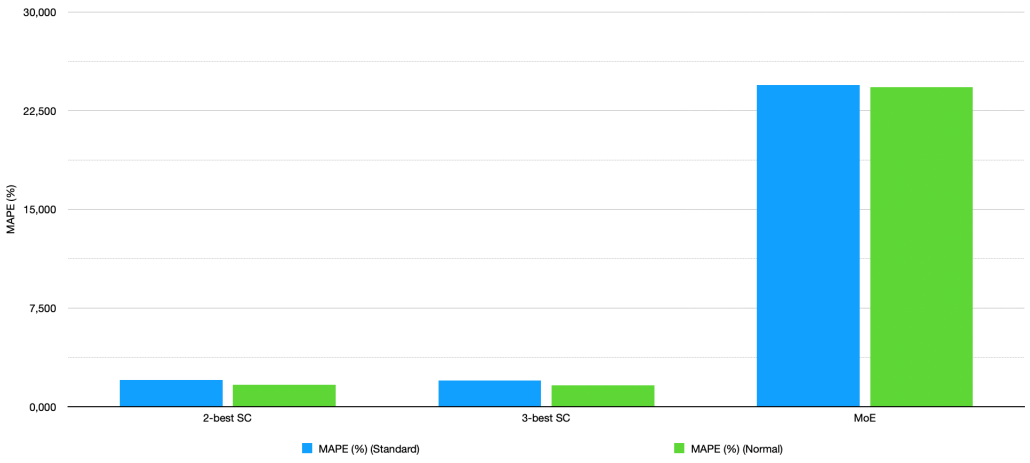
Χρησιμοποιήσαμε τα ακόλουθα 3 μετα-μοντέλα:

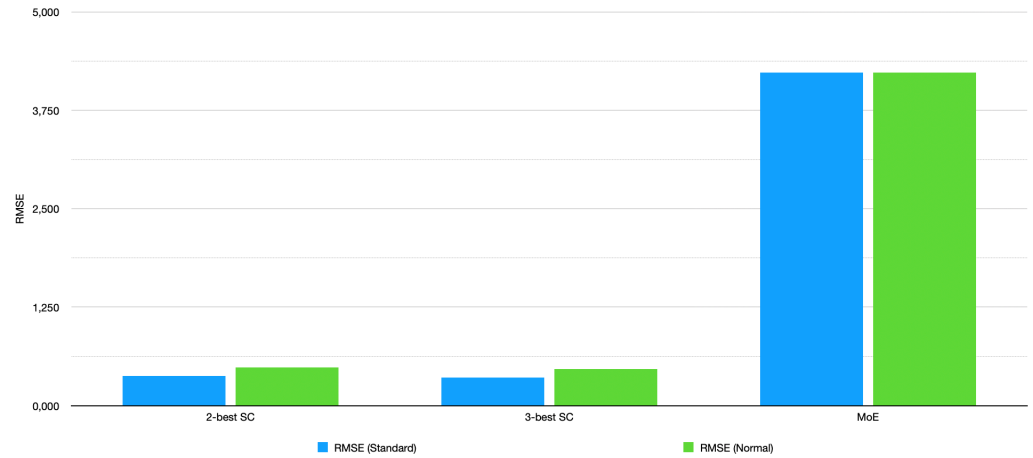
- 2-best Stacking Classifier/Regressor
- 3-best Stacking Classifier/Regressor
- Mixture of Experts (only for Deep Learning models)

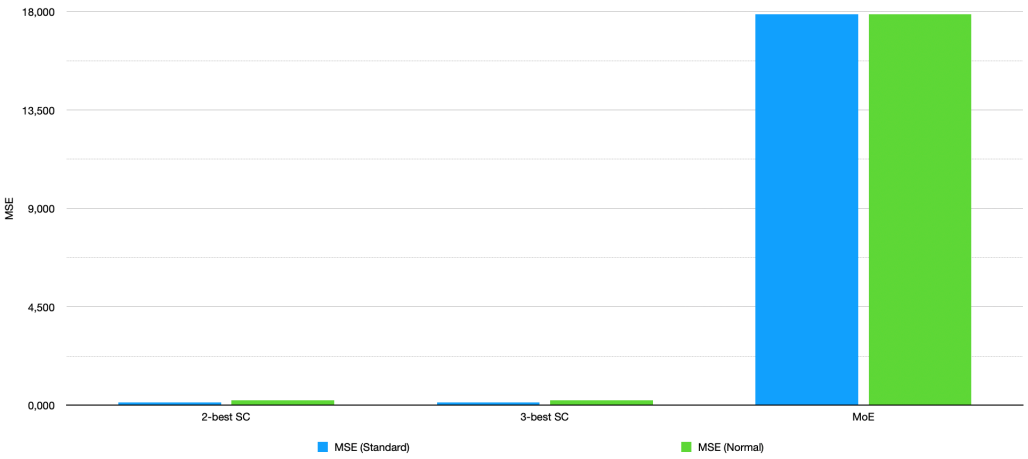
Και έχουμε τα ακόλουθα αποτελέσματα:

	MAPE (%) (Standard)	MAPE (%) (Normal)	RMSE (Standard)	RMSE (Normal)	MSE (Standard)	MSE (Normal)	Execution Time (sec) (Standard)	Execution Time (sec) (Normal)
2-best SC	2,000	1,643	0,373	0,485	0,139	0,235	153	187
3-best SC	1,990	1,622	0,358	0,467	0,128	0,218	191	249
MoE	24,440	24,264	4,228	4,228	17,878	17,877	850	830

Και όπως βλέπουμε και από τις γραφικές παραστάσεις:







- Δεν υπάρχουν σταθερές διαφορές μεταξύ του τυποποιημένου (standardized) συνόλου δεδομένων και του κανονικοποιημένου (normalized) μας. Ορισμένα μοντέλα επιτυγχάνουν καλύτερη ακρίβεια σε ένα από αυτά, αλλά ακόμη και αυτά βλέπουμε ότι για διαφορετικές μετρήσεις-δείκτες αλλάζουν την προτιμώμενη μέθοδο κλιμάκωσης χαρακτηριστικών.
- Το μείγμα εμπειρογνωμόνων (MoE) ήταν λιγότερο από 1% καλύτερο από το LSTM μοντέλο σε όλες τις βασικές μετρήσεις για την αξιολόγηση των μοντέλων βαθείας μάθησης.
- Οι Συνδυαστές Παλινδρόμησης (Stacking Regressor) των 2 καλύτερων και των 3 καλύτερων μοντέλων βρίσκονται πολύ κοντά σε όλες τις βασικές μετρήσεις.
- Σε σύγκριση με το καλύτερο απλό μοντέλο μας (CatBoost), τα μετα-μοντέλα μας έδωσαν καλύτερα αποτελέσματα κατά 25% και σε σύγκριση με το 3ο καλύτερο μοντέλο μας τα μετα-μοντέλα μας έδωσαν καλύτερα αποτελέσματα κατά 130%.

Επιπλέον αποτελέσματα και συμπεράσματα για τα μοντέλα μετα-μάθησης:

Ένας από τους κύριους στόχους αυτής της διατριβής, εκτός από τη συνολική μελέτη της πρόβλεψης πωλήσεων για FMCGs, ήταν να εξετάσουμε και αναλύσουμε τη διαδικασία και το θεωρητικό υπόβαθρο δημιουργίας μετα-μοντέλων, και ιδιαίτερα των Συνδυαστών Ταξινόμησης/Παλινδρόμησης (Stacking Classifier/Regressor), αλλά και το Μίγμα Εμπειρογνωμόνων (MoE).

Τα αποτελέσματα και συμπεράσματα που προκύπτουν από τα πειράματά μας είναι:

- Οι 2-καλύτεροι και οι 3-καλύτεροι Συνδυαστές Ταξινόμησης/Παλινδρόμησης (Stacking Classifier/Regressor) βρίσκονται πολύ κοντά σε όλες τις βασικές μετρήσεις. Έτσι, λαμβάνοντας υπόψιν την ανάλυση χρόνου εκτέλεσης, φαίνεται ότι ο 2-καλύτερος StackingRegressor είναι η βέλτιστη λύση για την περίπτωση των προβλέψεων πωλήσεων για FMCGs.
- Σε σύγκριση με το καλύτερο απλό μοντέλο μας (CatBoost), τα μετα-μοντέλα μας έδωσαν καλύτερα αποτελέσματα κατά 25% και σε σύγκριση με το 3ο καλύτερο μοντέλο μας τα μετα-μοντέλα μας έδωσαν καλύτερα αποτελέσματα κατά 130%. Αυτές είναι βελτιώσεις μεγάλης σημασίας και έτσι μπορούμε να καταλήξουμε στο συμπέρασμα ότι η χρήση του StackingRegressor είναι απαραίτητη για κάθε υλοποίηση προβλέψεων πωλήσεων.
- Ένας από τους βασικούς περιορισμούς που έχει κάποιος στη χρήση ενός Μείγματος Εμπειρογνωμόνων (MoE), είναι ότι χρειάζεται μοντέλα από την ίδια ομάδα (π.χ. νευρωνικά δίκτυα), ενώ ένας StackingClassifier μπορεί να συγκροτηθεί από οποιοδήποτε είδος μοντέλου. Έτσι, καθώς ένα ποσοστό των δεδομένων μας μπορεί πιο εύκολα να προβλεφθεί από μια συγκεκριμένη αρχιτεκτονική (π.χ. CatBoost), το stacking θα έχει το πάνω χέρι σε σύγκριση με τη χρήση ενός MoE.
- Το μοντέλο μίξης εμπειρογνωμόνων που χρησιμοποιήσαμε βασίζεται στις ίδιες ρυθμίσεις MLP, με τη λειτουργία softmax στο δίκτυο πύλης. Το MoE μας, όπως αναμενόταν, βελτίωσε το μοντέλο MLP, αλλά είχε απλώς ένα μέσο αποτέλεσμα σε σύγκριση με τα προηγούμενα. Προσπαθήσαμε να βελτιώσουμε την εφαρμογή του MoE, αλλά τα αποτελέσματα δείχνουν ότι είναι καλύτερο να επιλέξουμε ένα απλούστερο μοντέλο, καθώς τα πιο ευέλικτα μοντέλα είναι πολύ ισχυρά και λόγω της μικρής διάστασης υπερφορτώνονται γρήγορα. Χρησιμοποιήσαμε μια ιεραρχική υλοποίηση, η οποία μας έδωσε καλύτερα αποτελέσματα από μια κανονική λόγω της φύσης του προβλήματος.
- Το μείγμα εμπειρογνωμόνων (MoE) ήταν λιγότερο από 1% καλύτερο από το LSTM σε όλες τις βασικές μετρήσεις για την αξιολόγηση μοντέλων βαθιάς μάθησης.

Φαίνεται ότι το MoE δεν είναι πολύ χρήσιμο στην πρόβλεψη πωλήσεων, καθώς η αρχιτεκτονική του βασίζεται σε καμπύλες Gauss και, επομένως, μπορεί να είναι εξαιρετικά ευέλικτο, κάτι που συνήθως αποτελεί πλεονέκτημα, αλλά στην περίπτωση της πρόβλεψης πωλήσεων για FMCGs, λόγω της χαμηλής διάστασης εισόδου, υπερφορτώνεται (overfitting). Συνολικά, όμως, βλέπουμε ότι οι υλοποιήσεις των μετα-μοντέλων, και συγκεκριμένα οι StackingRegressors, έχουν πολλά πλεονεκτήματα, όπως:

- Οδηγούν σε εξαιρετικά ακριβείς προβλέψεις πωλήσεων, επειδή συνδυάζουν αποτελεσματικά τα καλύτερα μεμονωμένα μοντέλα σε καλύτερα συνδυαστικά μοντέλα.
- Συνδυάζουν την υψηλή ακρίβεια προβλέψεων με χαμηλό χρόνο εκπαίδευσης, καθιστώντας τους μια εξαιρετική επιλογή για λύσεις πρόβλεψης πωλήσεων σε πραγματικό κόσμο που εφαρμόζονται από εταιρείες στους κλάδους των προϊόντων ταχείας κατανάλωσης.
- Μπορούν να χρησιμοποιηθούν τόσο από επιχειρήσεις που δεν έχουν επενδύσει ακόμη στις διαδικασίες πρόβλεψης πωλήσεων (π.χ. νέες επιχειρήσεις, νεοσύστατες επιχειρήσεις κ.λπ.), αλλά και από επιχειρήσεις που διαθέτουν μοντέλα και διαδικασίες πρόβλεψης πωλήσεων και θέλουν να βελτιώσουν την ακρίβεια των μοντέλων τους με επιπλέον επενδύσεις.

Χρόνοι εκτέλεσης:

Γνωρίζουμε ότι δεν είναι συνηθισμένο να συγκρίνουμε τον χρόνο εκτέλεσης για εργασίες που δεν πρέπει να εκτελούνται πολλές φορές. Η πρόβλεψη πωλήσεων είναι, τουλάχιστον θεωρητικά, μία από αυτές. Ωστόσο, σε πραγματικές εφαρμογές, η πρόβλεψη πωλήσεων πρέπει να επαναληφθεί πολλές φορές. Ειδικά για τα προϊόντα ταχείας κατανάλωσης, λόγω της ιδιαίτερης φύσης τους, ο χρόνος εκτέλεσης των σχετικών μοντέλων είναι υψηλής σημασίας.

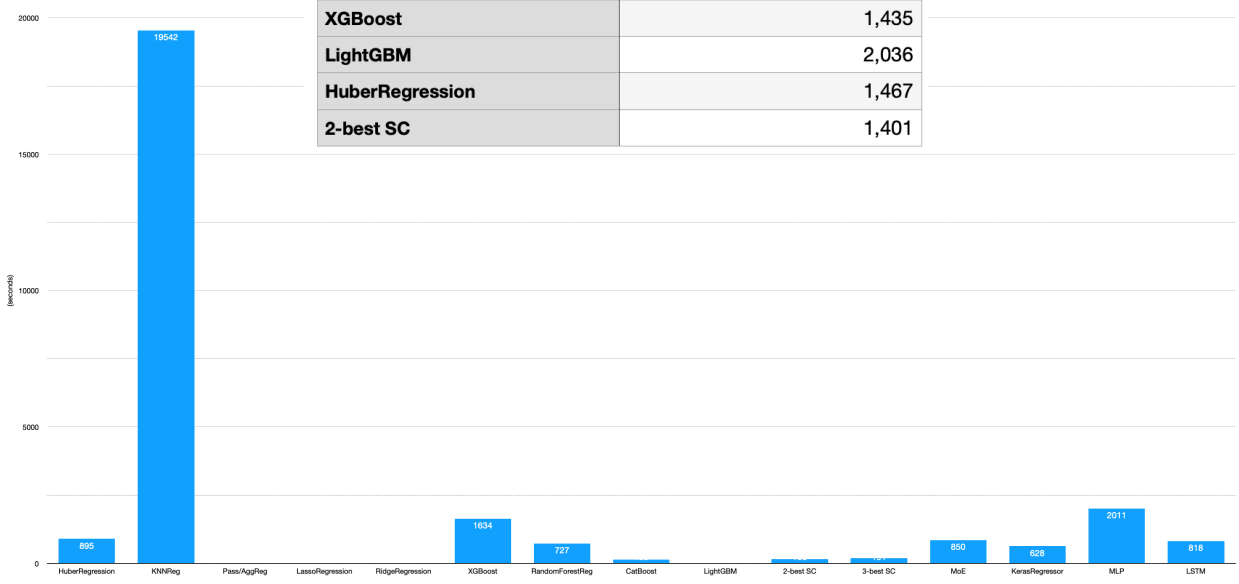
Αυτό συμβαίνει διότι:

- Οι επιχειρήσεις αποφασίζουν να βελτιώσουν τα παλαιότερα μοντέλα πρόβλεψης πωλήσεων.
- Οι επιχειρήσεις αλλάζουν τις διαδικασίες πρόβλεψης πωλήσεων.
- Νέα έρευνα που προτείνει πιο ακριβή μοντέλα βλέπει το φως και οι επιχειρήσεις θέλουν να τα δοκιμάσουν.
- Οι αλλαγές στο προσωπικό στα τμήματα μάρκετινγκ ή στις ομάδες δεδομένων ή στη διαχείριση των επιχειρήσεων επηρεάζουν τις μεθόδους πρόβλεψης πωλήσεων και τις εσωτερικές διαδικασίες.
- Δημιουργούνται νέα δεδομένα για υπάρχοντα προϊόντα.
- Κυκλοφορούν νέα προϊόντα.
- Λαμβάνονται υπόψιν νέα χαρακτηριστικά (features).
- Οι επιχειρήσεις αποφασίζουν να αυξήσουν την επένδυσή τους στην πρόβλεψη πωλήσεων και, επομένως, επενδύουν για να εκτελέσουν εκ νέου και να βελτιώσουν τα υπάρχοντα μοντέλα, μεθόδους και διαδικασίες τους.

Για αυτόν τον λόγο μετρήσαμε τον χρόνο εκτέλεσης στα μοντέλα μας και τον παρουσιάσαμε ανά κατηγορία μοντέλου στο Κεφάλαιο 5 ("Πειράματα").

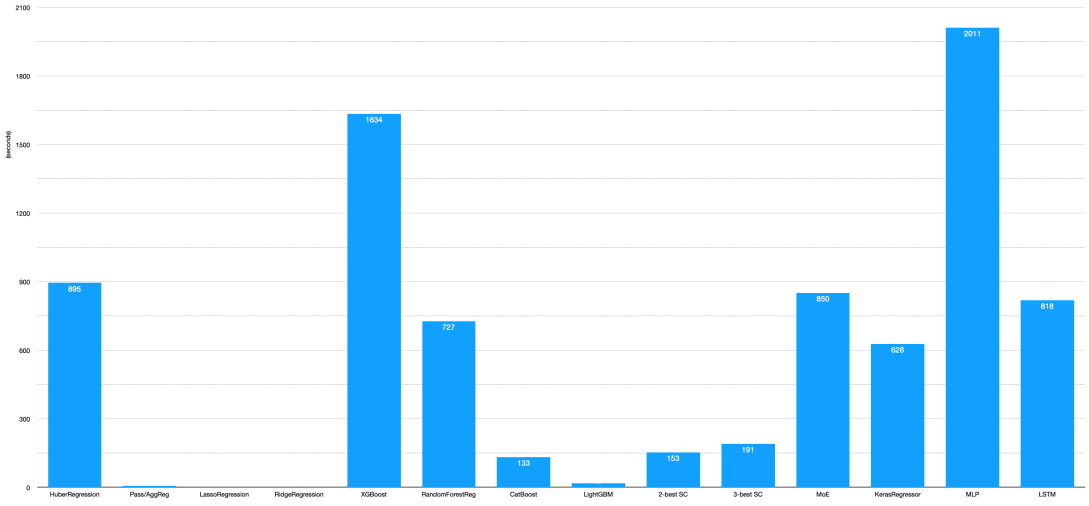
Μια γρήγορη εικόνα του χρόνου εκτέλεσης όλων των μοντέλων μας:

Model	RMSE
CatBoost	3,367
XGBoost	1,435
LightGBM	2,036
HuberRegression	1,467
2-best SC	1,401

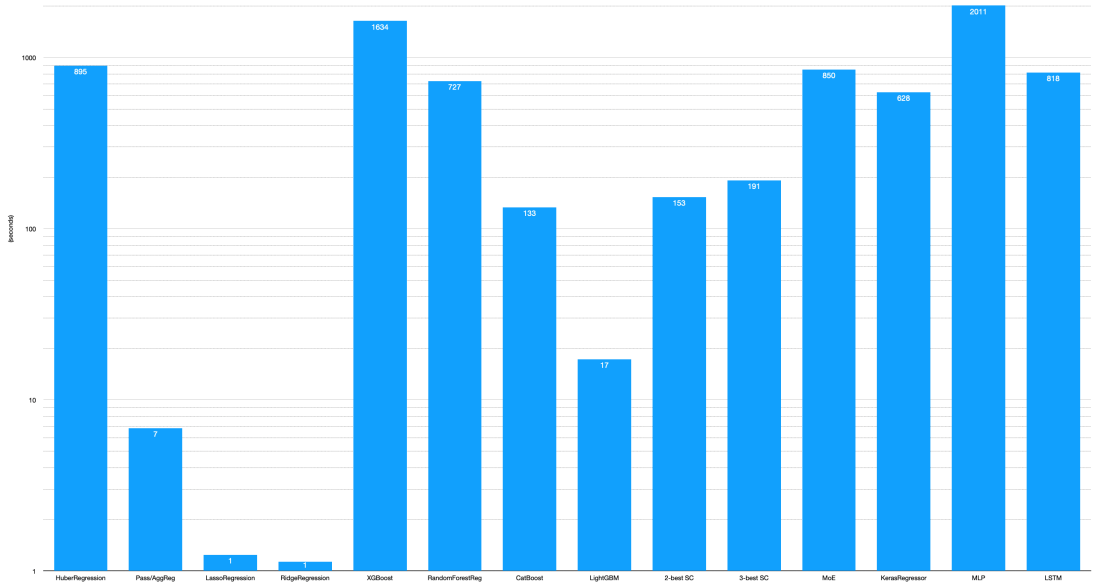


Βλέπουμε ότι το KNN Regression διαρκεί πολλές φορές περισσότερο χρόνο από όλα τα άλλα μοντέλα μας. Για αυτόν τον λόγο, το αποκλείουμε από τα χρονοδιαγράμματα εκτέλεσης.

Με κλίμακα γραμμικού άξονα:

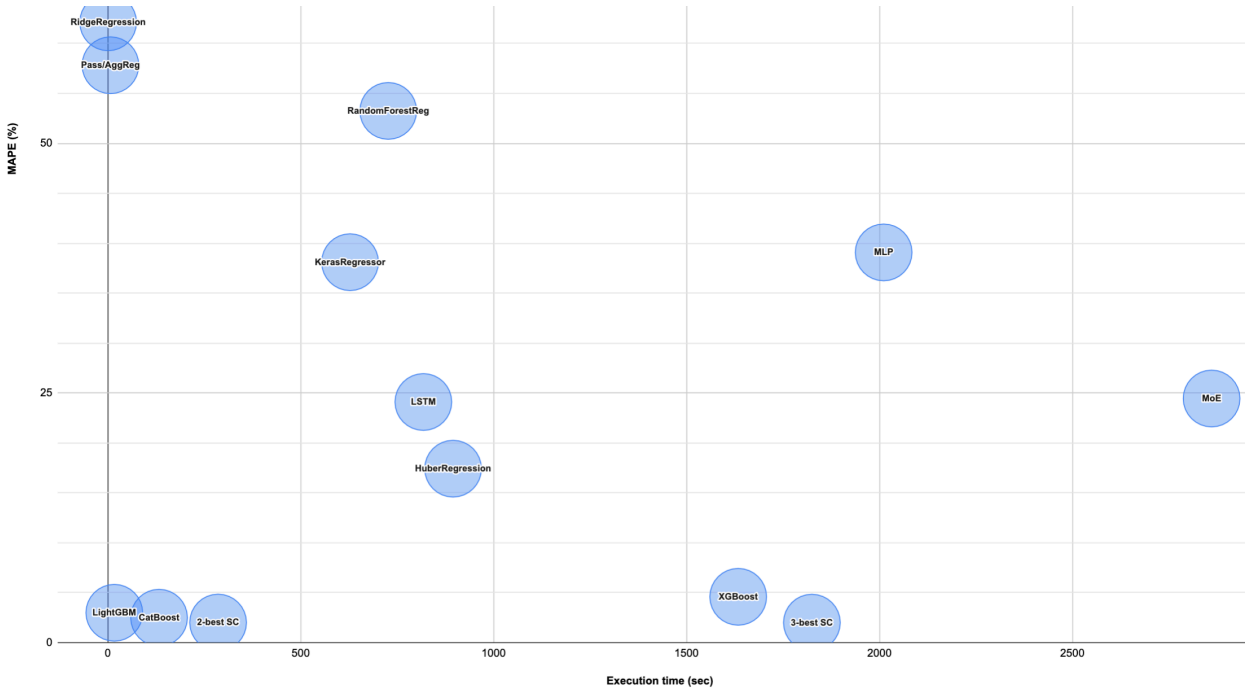


Με λογαριθμική κλίμακα άξονα:



- Είναι εύκολο να δούμε ότι τα μοντέλα με τις καλύτερες επιδόσεις (CatBoost και LightGBM) χρειάζονται πολύ λίγο χρόνο, καθιστώντας τα ιδανικούς υποψήφιους για προβλέψεις πωλήσεων για FMCGs σε κάθε επιχείρηση του κλάδου.
- Επίσης, για να εφαρμόσουμε τους Stacking Regressors, ο χρόνος εκτέλεσης τους είναι «επιπλέον» σε όλα τα άλλα μοντέλα, κάτι που πρέπει να λάβουμε υπόψιν μας. Ενώ η εκτέλεση όλων των άλλων μοντέλων μπορεί θεωρητικά να παραλληλιστεί, αυτό δεν ισχύει για τους N-best StackingRegressors καθώς υλοποιούνται σε 2 βήματα και πρέπει πρώτα να έχουν τα αποτελέσματα από όλα τα άλλα μοντέλα και να επιλέξουν τα καλύτερα μοντέλα για ενοποίηση.

Έτσι, ένα καλό τελικό γράφημα των μοντέλων μας μοιάζει με αυτό:



Και όπως μπορούμε να δούμε:

- Τα CatBoost και LightGBM δεν είναι μόνο εξαιρετικά μοντέλα για προβλέψεις πωλήσεων για προϊόντα ταχείας κατανάλωσης, αλλά και πολύ γρήγορα.
- Εάν εφαρμόσουμε μόνο τις λύσεις για τα CatBoost και LightGBM μοντέλα, ο 2-καλύτερος StackingRegressor «περιμένει» μόνο για αυτά και με τον πρόσθετο χρόνο υλοιποίησης για στοίβαξη, μπορούμε να δούμε τον τελικό χρόνο εκτέλεσης. Φυσικά, αυτή είναι η βέλτιστη περίπτωση, γιατί σε μια πραγματική ανάλυση μελέτης, θα είχαμε τρέξει πολλά μοντέλα και θα επιλέγαμε τα δύο καλύτερα μοντέλα, με πιθανώς υψηλότερο χρόνο εκτέλεσης από τα δύο καλύτερα μοντέλα μας. Ωστόσο, εάν γνωρίζουμε από την αρχή ποια είναι τα κορυφαία μοντέλα, τότε μπορούμε απλά να τρέξουμε ένα 2-καλύτερο StackingRegressor με βάση αυτά. Έτσι, ο πραγματικός χρόνος λειτουργίας θα είναι αυτός που απεικονίζεται στα παραπάνω διαγράμματα, γεγονός που καθιστά το Stacking Classifier / Regressor ένα από τα ταχύτερα μοντέλα.
- Ομοίως με το 2-καλύτερο StackingRegressor, ο βέλτιστος χρόνος για το 3-καλύτερο StackingRegressor είναι πολύ υψηλότερος, καθώς πρέπει να "περιμένει" για το XGBoost. Και πάλι, σε μια πραγματική μελέτη περίπτωσης, ο πραγματικός χρόνος εκτέλεσης θα ήταν πολύ μεγαλύτερος, καθώς θα έπρεπε να περιμένει περισσότερα μοντέλα. Από την άλλη, αν γνωρίζαμε από την αρχή τα βέλτιστα μοντέλα, θα μπορούσαμε να τρέξουμε τον 3-καλύτερο Stacking Regressor στο Στάδιο 1. Αυτό είναι εξαιρετικά ενδιαφέρον και σημαντικό, καθώς ολόκληρο το StackingRegressor θα είχε συνολικό χρόνο εκτέλεσης μικρότερο από ορισμένα από τα μεμονωμένα μοντέλα, και ταυτόχρονα αποτελέσματα υψηλότερης ακρίβειας. Έτσι, φαίνεται εδώ και η σημασία αυτής της διατριβής, καθώς έχοντας μία ακριβής αρχική πρόβλεψη σχετικά με το ποια είναι τα καλύτερα μοντέλα για την πρόβλεψη πωλήσεων για FMCGs, μπορεί να χρησιμοποιηθεί από τις επιχειρήσεις. Απλώς ένας StackingRegressor, με μεγαλύτερη ακρίβεια και μικρότερο κόστος από ένα συνηθισμένο έργο μηχανικής ή βαθιάς μάθησης φαίνεται να είναι η βέλτιστη επιλογή.
- Από τα μοντέλα βαθιάς μάθησης, το MLP έχει τη χειρότερη ακρίβεια, αλλά και το χειρότερο χρόνο εκτέλεσης. Για το λόγο αυτό, το Μείγμα Εμπειρογνωμόνων είναι επίσης αργό καθώς πρέπει να περιμένει να ολοκληρωθεί το νευρωνικό δίκτυο MLP.
- Το HuberRegression, όπως είπαμε και νωρίτερα, μας εντυπωσίασε, καθώς είναι ένας απλός αλγόριθμος παλινδρόμησης που κατάφερε να επιτύχει πολύ ικανοποιητική ακρίβεια σε χαμηλό χρόνο εκτέλεσης. Είναι σίγουρα ένα μοντέλο που πρέπει να ληφθεί περισσότερο υπόψη, ειδικά

στην περίπτωση της πρόβλεψης πωλήσεων, όπου υπάρχουν πάντα ακραίες τιμές. Η αντίσταση του HuberRegression στα ακραία σημεία (outliers) το καθιστά πολύ χρήσιμο, ακόμη και όταν έχουμε λίγα δεδομένα για την εκπαίδευση πιο περίπλοκων μοντέλων.

Διαγωνισμός Kaggle για σύγκριση μεθοδολογίας και αποτελεσμάτων

Προκειμένου να έχουμε ακόμη περισσότερα δεδομένα και αποτελέσματα σχετικά με τα μοντέλα και μετα-μοντέλα που δημιουργήσαμε, διαγωνιστήκαμε σε έναν Kaggle διαγωνισμό σχετικά με τις προβλέψεις πωλήσεων. Ήταν επίσης ένα καλό έδαφος για να δούμε εάν τα μοντέλα που αποδείχθηκαν οι κορυφαίες επιλογές για την πρόβλεψη πωλήσεων FMCGs είναι επίσης κορυφαίες επιλογές για άλλους τύπους προϊόντων και πιο ευρείες προβλέψεις πωλήσεων. Τα προτεινόμενα μοντέλα και μεθοδολογία οδηγούν στα ίδια αποτελέσματα, δείχνοντας ότι η ακρίβεια του Stacking Classifier με βάση τα δύο ή τρία καλύτερα μοντέλα είναι σημαντικά υψηλότερη από την ακρίβεια κάθε μεμονωμένου μοντέλου. Στην πραγματικότητα, η λύση βρισκόταν στις κορυφαίες 50 λύσεις του Kaggle (top 1%), σε έναν διαγωνισμό με περισσότερες από 10.000 υποβληθείσες λύσεις (46η θέση στον Πίνακα βαθμολογίας του διαγωνισμού).

Τα μοντέλα με την καλύτερη ακρίβεια πρόβλεψης για τον διαγωνισμό ήταν: α. XGBoost, β. HuberRegression. Το XGBoost έχει εξαιρετική απόδοση λόγω της επεκτασιμότητάς του και της ακριβούς εφαρμογής της ενίσχυσης κλίσης, πετυχαίνοντας υψηλή απόδοση και χαμηλή υπολογιστική πολυπλοκότητα. Επίσης, το HuberRegression οδηγεί σε πολύ καλά αποτελέσματα, λόγω της αντίστασής του στις ακραίες τιμές. Όσον αφορά τον διαγωνισμό Kaggle, ξοδέψαμε λιγότερο χρόνο για την προεπεξεργασία και τη μηχανική χαρακτηριστικών σε σχέση με τα κύρια μοντέλα μας, βασισμένα στα marketplaces, οπότε και είναι φυσικό το HuberRegression να έχει καλύτερη ακρίβεια από άλλα, πιο εξελιγμένα, μοντέλα. Υποθέτουμε ότι, αν δεν είχαμε κάνει τόσο ολοκληρωμένη επεξεργασία δεδομένων και ανάλυση χαρακτηριστικών στα κύρια σύνολα δεδομένων μας, θα είχαμε δει το ίδιο αποτέλεσμα από το HuberRegression. Αυτό σημαίνει ότι το HuberRegression θα έχει χαμηλότερη «ποινή» από την χειρότερη προεπεξεργασία δεδομένων και τη λειτουργία χαρακτηριστικών, επιτυγχάνοντας τα ίδια ή ακόμα καλύτερα αποτελέσματα σε σύγκριση με τα μοντέλα GBDT. Ωστόσο, με την πλήρη επεξεργασία δεδομένων και τη λειτουργία χαρακτηριστικών, είναι φυσικό τα μοντέλα GBDT (XGBoost, LightGBM & CatBoost) να είναι εκπληκτικά αποτελεσματικά.

Οπότε, τα μοντέλα που εκπαιδεύσαμε είναι:

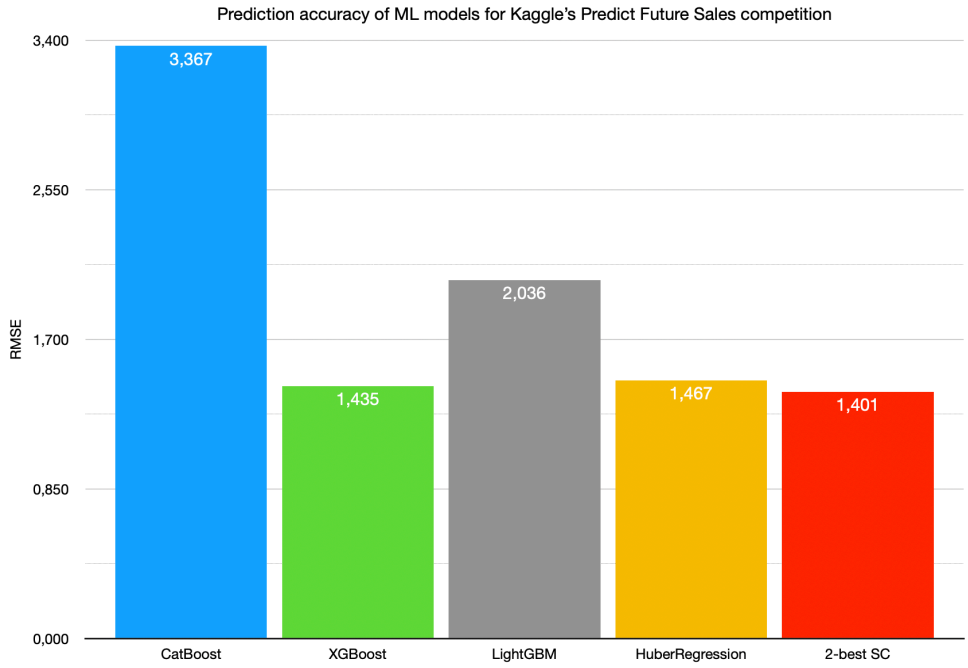
ML models:

1. HuberRegression
2. XGBoost
3. CatBoost
4. Lightgbm

Μοντέλο μετα-μάθησης: 2-best Stacking Classifier

Ο στόχος είναι να δοκιμάσουμε εντελώς ανεξάρτητα την αξία των κορυφαίων μοντέλων μας. Για να το κάνουμε αυτό, επιλέξαμε αυτόν τον διαγωνισμό Kaggle που μοιάζει με τα αρχικά σύνολα δεδομένων μας και έτσι, η βαθμολογία των μοντέλων ML θα δοκιμαστεί σε αυτό το παγκόσμιο, ανεξάρτητο περιβάλλον.

Μετράμε μόνο το RMSE κάθε μοντέλου, καθώς θα δείξουμε στο Κεφάλαιο 5 ότι τα GBDT μοντέλα είναι εξαιρετικά συνεπή σε όλες τις βασικές μετρήσεις (έτσι η μέτρηση και η σύγκριση μόνο του RMSE του κάθε μοντέλου είναι αρκετή). Τα αποτελέσματα ήταν:



Έτσι:

- Το CatBoost ενώ ήταν η κορυφαία επιλογή για τα αρχικά δεδομένα μας, ήταν πολύ χειρότερο σε αυτό το σύνολο δεδομένων. Όπως εξηγήσαμε, αυτό το περιμέναμε λόγω της μικρότερης προεπεξεργασίας που έγινε, αλλά και από την ίδια την φύση του προβλήματος. Είδαμε αναλυτικά γιατί τα FMCGs βοηθούν τον CatBoost αλγόριθμο να τους δώσει τα καλύτερα δυνατά αποτελέσματα.
- Το HuberRegression απέδειξε για άλλη μια φορά ότι είναι ένας εξαιρετικά απλός, αλλά εξαιρετικά ισχυρός αλγόριθμος παλινδρόμησης, ο οποίος έχει εξαιρετική απόδοση ακόμη και σε λιγότερο ομοιογενή σύνολα δεδομένων πωλήσεων. Έτσι, μπορεί να χρησιμοποιηθεί αποτελεσματικά σε σχεδόν οποιοδήποτε είδος πρόβλεψης πωλήσεων και είναι απαραίτητο για την ανάπτυξη προβλέψεων πωλήσεων από μελετητές και ομάδες δεδομένων επιχειρήσεων.
- Το XGBoost ήταν το πιο ακριβές μοντέλο και υπερασπίστηκε τη θέση του ως ένα από τα πιο χρήσιμα και πιο χρησιμοποιημένα μοντέλα μηχανικής μάθησης. Είναι ανώτερο τόσο στην ακρίβεια όσο και στην ποικιλία χρήσης (χρησιμότητα) του.
- Ο 2-καλύτερος Stacking Regressor μας οδήγησε σε λίγο καλύτερα αποτελέσματα, όπως αναμενόταν, αλλά όχι αρκετά για να ικανοποιήσει το επιπλέον επίπεδο ανάπτυξης και δέσμευσης χρόνου. Βλέπουμε ότι πήγε μόλις 2.4% καλύτερα. Το αν αυτή είναι πολύτιμη διαφορά, παραμένει ελαφρώς υποκειμενικό. Συνιστούμε σίγουρα σε επιχειρήσεις που θέλουν την καλύτερη δυνατή ακρίβεια πρόβλεψης να δοκιμάσουν οπωσδήποτε μετα-μοντέλα, αλλά βλέπουμε εδώ ότι με το να μην δοκιμάζουμε αρκετά μοντέλα, με μη επαρκές data pre-processing, με το να μην συνδυάζονται αρκετά τα μοντέλα μας και με παρόμοια αποτελέσματα από τα κορυφαία μοντέλα, η βελτίωση της ακρίβειας των προβλέψεων που παρέχουν τα μετα-μοντέλα μπορεί να θεωρηθεί ακόμα και αμελητέα.

Φαίνεται ότι η δουλειά μας ήταν αρκετά καλή για να τοποθετηθεί στις κορυφαίες λύσεις παγκοσμίως. Οι προβλέψεις μας ανήκουν στο top 1% των υποβολών από όλο τον κόσμο και συγκεκριμένα την 46η του διαγωνισμού, με περισσότερες από 10.000 υποβολές (μέχρι την στιγμή που λήφθηκε το παρακάτω στιγμιότυπο για την ολοκλήρωση αυτής της διπλωματικής εργασίας, έχουμε πέσει στην 70η θέση, αλλά παραμένουμε στο top 1%).

Συνολικά, φαίνεται ότι ο χρόνος που εξοικονομήσαμε δοκιμάζοντας τα μοντέλα υψηλότερης ακρίβειας από τα βασικά δεδομένα μας για την πρόβλεψη πωλήσεων στο πλαίσιο του διαγωνισμού είναι σημαντικό και η ακρίβεια των προβλέψεων που επιτύχαμε είναι κάτι παραπάνω από καλή. Αυτό δείχνει για άλλη μια φορά, ότι βάσει αυτής της διατριβής μπορούν να επωφεληθούν πολλές επιχειρήσεις, τόσο αυτές που έχουν ήδη εφαρμόσει μοντέλα πρόβλεψης πωλήσεων και διαδικασίες διανομής προβλέψεων, αλλά και εκείνες που θα αναπτύξουν τέτοια μοντέλα και διαδικασίες στο μέλλον.

Γνωρίζοντας λόγω αυτής, και άλλων εργασιών, ποια μοντέλα έχουν την βέλτιστη απόδοση και ακρίβεια πριν την δημιουργία των δικών τους μοντέλων, μειώνουν τα κόστη επένδυσης τους σημαντικά και αυξάνουν την ταχύτητα υλοποίησης του σχετικού έργου, εξασφαλίζοντας ταυτόχρονα την δημιουργία διαδικασιών και μοντέλων με εκπληκτικά υψηλή ακρίβεια προβλέψεων.

0.5 Συμπεράσματα και μελλοντικές κατευθύνσεις

Σε αυτήν την ενότητα αναλύουμε τα συμπεράσματα μας και τις μελλοντικές κατευθύνσεις της σχετικής δουλειάς τόσο για τα προϊόντα ταχείας κατανάλωσης, όσο και γενικότερα μελλοντικές προεκτάσεις της έρευνας επί της πρόβλεψης πωλήσεων.

Όσον αφορά τις μελλοντικές κατευθύνσεις, ας εξετάσουμε, πρώτα, πιθανές βελτιώσεις στα μοντέλα μας, αλλά και την ανάγκη για επιπλέον πειράματα. Το Μίγμα Εμπειρογνωμόνων που χρησιμοποιήσαμε, περιορίστηκε στην αρχιτεκτονική MLP που δεν ήταν κοντά στις βέλτιστες περιπτώσεις. Στο μέλλον, θα μπορούσαν να γίνουν περισσότερες εργασίες για την υλοποίηση ενός MoE που βασίζεται σε καλύτερα νευρωνικά δίκτυα ή GBDT μοντέλα. Αποφύγαμε να συγκρίνουμε άμεσα και να εξαγάγουμε συμπεράσματα σχετικά με την απόδοση των StackingRegressors με αυτό του MoE, επειδή βασίστηκαν σε διαφορετικά μοντέλα. Έτσι, είναι λογικό να παράγουν διαφορετικά αποτελέσματα και να πετυχαίνουν διαφορετικές ακρίβειες. Ωστόσο, από αυτή τη διαφορά στην εκπαίδευση μοντέλων καταφέραμε να επισημάνουμε ότι ενώ το MoE χρειάζεται μοντέλα από την ίδια ομάδα (π.χ. νευρωνικά δίκτυα), ένας StackingClassifier μπορεί να ενσωματώσει οποιοδήποτε είδος μοντέλου. Σε κάθε περίπτωση, φαίνεται να απαιτείται περισσότερη δουλειά για την καλύτερη επιλογή ενός μετα-μοντέλου σε διαφορετικές περιπτώσεις πρόβλεψης πωλήσεων.

Απαιτείται επίσης περισσότερη δουλειά σε πρόσθετα σύνολα δεδομένων. Ενώ χρησιμοποιήσαμε καλές αναπαραστάσεις της τρέχουσας υπόθεσης της πρόβλεψης πωλήσεων σε πραγματικό επιχειρηματικό περιβάλλον, δεν πειραματιστήκαμε αρκετά με εντελώς διαφορετικές προσεγγίσεις. Κάποιος θα μπορούσε να υποστηρίξει ότι περισσότερα ή διαφορετικά χαρακτηριστικά θα μπορούσαν και θα έπρεπε να χρησιμοποιηθούν στην πρόβλεψη πωλήσεων για FMCGs και έτσι οι επιχειρήσεις πρέπει να κάνουν αλλαγές στην ποσότητα ή την ποιότητα των δεδομένων που διατηρούν. Επιπλέον, ο μετασχηματισμός δεδομένων, η επεξεργασία δεδομένων και η μηχανική χαρακτηριστικών διαδραματίζουν πάντα σημαντικό ρόλο σε οποιοδήποτε έργο και υλοποίηση μηχανικής εκμάθησης και έτσι ένας τρόπος για να δείτε και να ελέγξετε γενικά την πρόβλεψη πωλήσεων μηχανικής μάθησης είναι να ελέγξετε όλα τα βήματα πριν από τη δημιουργία μοντέλων, από τη διατήρηση και διαχείριση δεδομένων, έως την ανάλυση δεδομένων, έως την επιχειρηματική και διαχειριστική γνώση, ικανότητα, πρωτοβουλία και ευφυΐα. Παρόλο που εργαστήκαμε με δύο είδη κλιμάκωσης χαρακτηριστικών καθώς παίζει

σημαντικό ρόλο στην προεπεξεργασία, και φαίνεται ότι η βιομηχανία διχάζεται σχετικά με το ποιο να χρησιμοποιήσει, δυνητικά ακόμη και διαφορετικές τεχνικές κλίμακας χαρακτηριστικών μπορούν να βελτιώσουν τη μηχανική χαρακτηριστικών και βελτιώσουν ή χειροτερέψουν την απόδοση ορισμένων μοντέλων. Βέβαια, από τα πειράματα μας είδαμε ότι η επιλογή της τεχνικής κλίμακας χαρακτηριστικών δεν έπαιξε σημαντικό ρόλο, αφού δεν υπήρξε τεχνική που απέδωσε ομοιόμορφα καλύτερα από κάποια άλλη.

Επιπλέον, ενώ δοκιμάσαμε τα κορυφαία μοντέλα από την πιο πρόσφατη έρευνα στον τομέα, υπάρχουν άλλες παρόμοιες μέθοδοι και άλλες αρχιτεκτονικές νευρωνικών δικτύων που μπορούν να εφαρμοστούν. Για παράδειγμα, το CNN φαίνεται μια εξαιρετική προσθήκη στη δημιουργία και συγκριτική αξιολόγηση μοντέλων νευρωνικών δικτύων. Ταυτόχρονα, το Facebook Prophet έχει δείξει ουσιαστική βελτίωση σε ερευνητικές εργασίες, αλλά και σε πολλές εφαρμογές μηχανικής μάθησης. Η πρόβλεψη του Prophet φαίνεται να είναι μια υποχρεωτική συμπερίληψη σε οποιοδήποτε είδος μοντέλου μηχανικής και βαθιάς μάθησης συγκριτικής αξιολόγησης στο μέλλον. Αυτή τη στιγμή παράγεται περισσότερη έρευνα σχετικά με αυτό, και περιμένουμε νέες υλοποιήσεις και προτάσεις, όπως ο αλγόριθμος DAE-LSTM.

Σε κάθε περίπτωση, η πρόβλεψη πωλήσεων γίνεται όλο και πιο σημαντική κάθε μέρα σε όλα τα τμήματα μιας επιχείρησης και σύντομα θα είναι απαραίτητο εργαλείο για εταιρείες κάθε κλάδου, αλλά ειδικά από τις επιχειρήσεις που παράγουν και εμπορεύονται προϊόντα ταχείας κατανάλωσης, όπου η πρόβλεψη πωλήσεων παίζει ακόμη πιο θεμελιώδη ρόλο στη βιωσιμότητα και την κερδοφορία τους.

Τελικά συμπεράσματα

Οι στόχοι αυτής της διατριβής επιτυγχάνονται πλήρως, επειδή καταφέραμε να διακρίνουμε τα μοντέλα που λειτουργούν τόσο σε θεωρητικό περιβάλλον όσο και στον πραγματικό κόσμο για το πρόβλημα της πρόβλεψης πωλήσεων για τα προϊόντα ταχείας κατανάλωσης. Τα CatBoost και LightGBM μοντέλα είναι καταπληκτικές επιλογές για την πρόβλεψη πωλήσεων για FMCGs και χρησιμοποιώντας μόνο ένα Συνδυαστή Ταξινόμησης/Παλινδρόμησης (Stacking Classifier/ Regressor) μπορούμε να εγγυηθούμε περισσότερο από ικανοποιητικά αποτελέσματα, σε σχεδόν οποιεσδήποτε συνθήκες. Έτσι, κατασκευάσαμε ένα γενικά καλό, ενοποιημένο μοντέλο για γρήγορη και ακριβή πρόβλεψη πωλήσεων για FMCGs: έναν StackingRegressor βασισμένο σε CatBoost και LightGBM.

Σε περίπτωση που οι επιχειρήσεις και οι σύμβουλοι τους πρέπει να πειραματιστούν με παραπάνω μοντέλα σε μια συγκεκριμένη περίπτωση/έργο για την πρόβλεψη πωλήσεων, είτε λόγω σημαντικής διαφοροποίησης του προβλήματος, είτε διαφοροποίησης στην ποσότητα ή/και την ποιότητα των δεδομένα τους, μπορούμε να τους προτείνουμε τις υλοποιήσεις με XGBoost και HuberRegression, για να έχουν πλήρη εικόνα και να βεβαιωθούν ότι χρησιμοποίησαν βέλτιστα το χρόνο τους για να δοκιμάσουν τα καλύτερα δυνατά μοντέλα. Τέλος, εάν υπάρχει ανάγκη να δοκιμαστούν επίσης αρχιτεκτονικές νευρωνικών δικτύων, το LSTM είναι μακράν το πιο ελπιδοφόρο. Ωστόσο, λόγω το ότι αντιμετωπίζουμε εκ φύσεως ένα πρόβλημα χαμηλών διαστάσεων, φαίνεται ότι δεν αποδίδουν καλύτερα οι υλοποιήσεις βαθείας μάθησης από τα μοντέλα και μετα-μοντέλα που προτείνουμε εδώ.

1. Introduction

1.1 Sales Forecasting

Due to the strong and growing competition existing nowadays, the majority of retailers are in a continuous effort for increasing profits and reducing costs [19]. In addition, the variations in consumers demand, which are caused by many factors like price, promotion, changing consumers' preference, seasonality, or weather changes, contribute to a fluctuating market behaviour [2]. In that sense, an accurate sales forecasting system is an efficient way to achieve higher profits and lower costs, by improving customers satisfaction, increasing sales revenue and designing production plans efficiently [9]. Sales forecasting is the starting point for planning various phases of a firms operations [13], and a crucial task in supply chain management under dynamic market demand which, ultimately, affects retailers and other channel members in various ways [19]. Industry forecasts are especially useful to big retailers who may have a greater market share [3]. Due to ever-increasing global competition and an environment characterised by very short product life cycles and high market volatility [2], this subject plays an even more prominent role in supply chain management when the profitability and the long-term viability of a firm relies on effective and efficient sales forecasts [6]. In particular, within the retail and consumer-oriented industries, and especially for fast-moving consumer goods such as the electronic market or the fashion industry, accurate forecasts are essential.

Companies face several challenges regarding accurate forecasts. For instance, they have to place their production plans before exact knowledge about future demand is available. This is required due to the fact that most production plants are located in Asian countries and therefore the time-to-market is longer than the selling period of fast-moving consumer goods. [5] The production and distribution of products is longer than the period of sale of other, domestically produced ,products. In addition, other factors, such as changing weather conditions, holidays, public events as well as the general economic situation, can have an impact on future demand (Thomassey, 2010). Summing up, due to short life cycles, high variability in products and demand uncertainties, businesses in fast-moving consumer goods industries face high challenges with regard to precise forecasts.

Inaccurate sales forecasts are likely to lead to large inventories that increase management and logistics costs or stock shortages that result in loss of profits for companies. The importance of research into high-precision sales forecasting models stems from the fact that sales forecasting is not only crucial for a huge number of companies, but also that most companies face huge challenges in achieving accurate forecasts.

Sales prediction is rather a regression problem than a time series problem. Practice shows that the use of regression approaches can often give us better results compared to time series methods. Machine-learning algorithms make it possible to find patterns in the time series. We can find complicated patterns in the sales dynamics, using supervised machine learning methods. One of the main assumptions of regression methods is that the patterns in the past data will be repeated in future.

In the sales data, we can observe several types of patterns and effects, such as, trend, seasonality, autocorrelation, patterns caused by the impact of such external factors as marketing, pricing, competitors' behaviour. We also observe noise in the sales. Noise is caused by the factors which are not included into our consideration. In the sales data, we can also observe extreme values (outliers). If we need to perform risk assessment, we should to take into account noise and extreme values. Outliers can be caused by some specific factors, e.g., promotion events, price

reduction, weather conditions, etc. If these specific events are repeated periodically, we can add a new feature which will indicate these special events and describe the extreme values of the target variable.

Also, there are some inherent limitations of time series approaches for sales forecasting:

• We need to have historical data for a long time period to capture seasonality. However, often we do not have historical data for a target variable, for example in case when a new product is launched. At the same time we have sales time series for a similar product and we can expect that our new product will have similar sales pattern.
• Sales data can have a lot of outliers and missing data. We must clean outliers and interpolate data before using a time series approach.
• We need to take into account a lot of exogenous factors which have impact on sales.

Due to the above limitations, in the last three decades many companies have switched from time series analysis models to regression approaches (Thomassey, 2010). Nowadays, many businesses change or want to change to machine learning implementations as, with machine learning, one can find intricate patterns in sales dynamics and get models with high prediction accuracy at a fraction of the cost in comparison with traditional sales forecasting analysis.

1.2 Thesis' Scope

As Sales Forecasting is an extremely big area and is severely impacted by the industry it is implemented for, and the type of product/service a business offers, I chose to study the fast-moving consumer goods industries. These include: 1. Fashion, 2. Electronics, 3. Beverages, 4. Cosmetics, 5. Detergents, 6. Cleaners, 7. Other fast-moving consumer goods. Although, there isn't enough research on this area, a lot of research has been conducted on Sales Forecasting specifically for fashion products (see also Chapter 3, "Previous Work").

As the fast-moving consumer goods area is also very big and diverse, we needed to further focus on a specific category of FMCGs. We chose to focus to work on Detergents & Cleaners for the following reasons:

1. Although there is a lot of research on Sales Forecasting for the Fashion industry there is nearly zero research on Sales Forecasting for Detergents & Cleaners.
2. The Detergents & Cleaners industry is a very fast growing one with a CAGR of more than 4% YoY.
3. They are a heavily price-sensitive industry which gives even more importance to pricing strategies and accurate Sales Forecasting to maximise revenues and profits.
4. They can efficiently represent the entirety of the fast-moving consumer goods area.

So, this Thesis' goal is to evaluate the most promising machine learning and deep learning models for sales forecasting for FMCGs and to build a unified model (meta-learner) with very good accuracy for Machine Learning Sales forecasting for fast-moving consumer goods. So, we research the effectiveness of both individual models and meta-learners for sales forecasting for FMCGs based on other models' prediction accuracy (hyper-parameter tuning with the top models).

The key value proposition is the following: Ensembling/Meta-learning is easy to execute and not extremely time-consuming. If a simple Meta-learning mechanism, based on our test models, is efficiently more accurate than every individual model, then we have a value-for-money solution for increased Sales prediction accuracy.

So, we created Meta-learners based on the best models on our data. Then we tested whether the proposed Stacking classifiers/regressors (SC/R) and/or Mixture of Experts (MoE) produce significant model improvements for this type of Sales Forecasting. It turns out that in the case of different models based on different algorithms and data, one can get a real gain in accuracy by using the right ensembling (meta-learning), as it manages to get the benefits from "all the worlds", combining the individual advantages of each model for the construction of a super-model.

Of course, at the same time, we conduct a comparison analysis of the most important Machine Learning and Deep Learning Sales Forecasting models and present our experiments in results in the next chapters.

1.3 Thesis Outline

In the following chapters we will cover the theoretical background, the previous research work, the method and models we used, the results and our final conclusions.

- *Chapter 2* introduces the main concepts of fast-moving consumer goods (FMCGs), Sales Forecasting and Machine Learning. We cover some basic terminology, relevant research and benchmark models. More focus is given on theoretically outlining Sales Forecasting as without deep understanding of the business case, the already tried solutions and the current implementations we will not be able to follow the main concept and goal.

- *Chapter 3* provides a review of previous work on Machine Learning implementations specifically for Sales Forecasting and Ensembling and Meta-learning.

- *Chapter 4* is wholly focused on the models we used. We provide a thorough theoretical background of each of the benchmark models we used and of our meta-learners. Reading Chapter 4 can be optional for Machine Learning engineers and Machine Learning professionals but it is a must for readers who are not fully accustomed to ML regression models or those who want to remember specifics and details of some of the ML models we used.

- *Chapter 5* presents the details of our actual work. We present all the data analysis, pre-processing, feature engineering and model training we did. Also, we present the key metrics which will be used to evaluate every model’s prediction accuracy.

- *Chapter 6* concludes the outcomes of our work and presents the final conclusion and ideas about future work based on this Thesis.

- *Chapter 7* is the Bibliography section (references to books, papers and publications we studied and used for this Thesis).

2. Theoretical Background

2.1 Fast-moving consumer goods

Fast-moving consumer goods (FMCGs) are consumer goods that have a short shelf life because of high consumer demand (e.g., soft drinks, cleaners, medicine) or because they are perishable (e.g., meat, dairy products, and baked goods). These goods are purchased frequently, are consumed rapidly, are priced low, and are sold in large quantities. They also have a high turnover when they're on the shelf at the store. For comparison. slow-moving consumer goods, are consumer goods that have a longer shelf life and are purchased over time, include items like furniture and appliances. [46]

FMCGs are purchased for consumption by the average consumer. Products, in general, are divided into three different categories: durable, nondurable goods, and services. Durable goods have a shelf life of three years or more while nondurable goods have a shelf life of less than one year. Fast-moving consumer goods are the largest segment of consumer goods. They fall into the nondurable category, as they are consumed immediately and have a short shelf life.

Fig. 1. Fast-Moving Consumer Goods (FMCGs)

Because fast-moving consumer goods (FMCGs) have such a high turnover and extremely fast sales, their market is not only very large, but also very competitive. Some of the world's largest

companies are competing for market share in this industry, including Coca-Cola, Unilever, Procter & Gamble, Nestlé, PepsiCo and Danone. For this reason, these companies are forced to focus their marketing efforts on their fast-moving consumer goods in order to entice and attract consumers to buy their products.

FMCGs are sold in large quantities, so they can generate extremely high revenues for businesses in these industries. This large sales volume also offsets the low profit margins in individual sales. This is why accurate sales forecasting is so important to them. These businesses' logistics and distribution systems require huge resources and special attention and having more accurate sales forecasting allows them to increase their profit margins, while the absence of satisfactory sales forecasting adds millions of additional costs in their supply chain.

2.2 Sales Forecasting

Sales forecasting refers to the prediction of future sales based on past historical data. Due to competition and globalisation, sales forecasting plays an even more important role as part of the commercial enterprise. Accurate sales forecasting is vital for profitable retail operations, because without good forecasting, either much more or much less stocks would result, directly affecting revenue and competitive position.

Mentzer and Moon (2005) defines a sales forecast as "a projection into the future of expected demand, given a stated set of environmental conditions". A sales forecast can therefore be explained as a way of using different factors (e.g. sales history, sales promotion, seasonality, and so on) to predict future sales and then use information from the forecast to develop plans for resources and capacity to meet the demand in the best possible way (Herbig et al., 1993; Jonsson & Mattson, 2011; Mentzer & Moon, 2005). Uncertainties are partly sales-related and often the result of a lack of knowledge or incorrect information. Further, frequently changing product ranges and varying demand due to different interfering influencing factors (seasonal influences, price and assortment policies etc.) are only a few of many factors that make it difficult to use conventional statistical forecasting models. In addition, the actual correlation between the influencing factors is often difficult to grasp or not recognisable by humans, which leads to an increased desire for and rising usage of artificial intelligence methods in demand planning [2].

Sales forecasting can also be used in significant managerial decision-making within companies (Herbig et al., 1993; Lee, 2000; Mentzer & Moon, 2005). If managers get more reliable information about future demand the simpler the decision-making about customer requirements is going to be (Herbig et al., 1993; Mentzer & Moon, 2005). For companies to be able to share the information for their sales forecasting, a well-designed forecasting process must exist. A well-designed forecasting process helps the company share information within the company, not just the outgoing data but also information about which data to use when forecasting, for example: marketing activities, market dynamics, consumer behaviour, and so on (Danese & Kalchschmidt, 2011). With a well-designed forecasting process companies can increase their sales forecasting accuracy by ensuring that the ingoing data is as accurate as possible (Ramanathan, 2012). Without sales forecasting, operations can only respond retroactively, leading to poor production planning, lost orders, inadequate customer service, and poorly utilised resources.

Therefore, companies need to make predictions of the products' future demand. These predictions can then be used for the planning, budgeting and scheduling of the companies resources. These kinds of predictions are also known as sales forecasting (Bovee et al., 2006). Regardless of industry, whether the company is a manufacturer, wholesaler, retailer or service provider, it is important to effectively forecast demand. This helps companies identify market opportunities, increase customer satisfaction, reduce inventory and obsolescence products and make scheduling more effective (Linderman et al., 2003; McIntyre et al., 1993). Because of this, forecasting is an

important part of all industries when it comes to business planning and management (Armstrong et al., 1997; Fildes & Hastings, 1994) and is in many cases what the basis of the corporate strategy is built upon (Mentzer et al., 1999). Sales Forecasts accuracy is even more necessary for Fast Moving Consumer Goods (FMCGs), as these products have large sales volume to offset the low profit margins in individual sales. Thus, bad forecasting would lead to even lower or even negative profit margins per individual sale which would make their production and retailing unattainable.

Sales Forecasting should be a high priority for FMCG businesses' management, as it includes decision-making about information gathering, data management and decision making tools. For instance, what information should be collected and how it should be used. It also includes organisational issues as to which department is responsible for creating the forecasts. Decisions also need to be taken in regard to the cooperation of information flow, both between the companies departments but also within the supply chain. Information from multiple sources can be used to improve the accuracy of the forecast (Fildes & Hastings, 1994; Fisher et al., 1994; Remus et al., 1995). This means that; to improve the understanding of how to reduce forecasting miscalculations companies must study the connections between forecasting techniques and different factors (Danese & Kalchschmidt, 2011).

These factors can be grouped quantitatively (based on data collected over a longer period of time) and qualitatively, based on judgment, intuition, and up-to-date (and inherently subjective) views (Davis and Mentzer, 2007; Kerkkänen and Huiskonen, 2007; Ingram, LaForge, Avila, Schwepker and Williams, 2012; Armstrong and Green, 2014). If a company has a large number of stored transactions, it is possible to use probability estimation techniques based on the development of opportunities (Lodato, 2006; Duran, 2008; Söhnchen and Albers, 2010). Such an approach is less applicable when there are fewer sales opportunities. In addition, the size of the opportunity also matters, as the company must allocate its resources (Duran, 2008).

Despite the advanced tools available over-discussing the models a business should use for sales forecasting might be unnecessary, or even hurtful. It is common that companies spend too much time on advanced tools, when focus instead should be on processes, procedures, and educating forecasters (the developers of a forecast) (Lawrence et al., 2000). It is important that all departments provide input for the forecast and that all essential data is regarded when making decisions (Gilmore & Lewis, 2006; Mentzer & Moon, 2005). A common problem in the making of a forecast is that not all departments realise the advantages and are therefore not interested in being part of the forecast creation. Thus it is important to make the departments understand in what way each department can benefit from an accurate forecast. A difficult task is also to balance it evenly so that the forecast does not become too extensive. It is important to acknowledge the differences between core needs of sales and wants of sales. The process of forecasting is often overcomplicated because of these differences (Gilmore & Lewis, 2006).

There are also two types of costs that result from inaccurate forecasting. The first one is if there is an over-forecast that results in overstock and obsolescence products and the second one is if there is an under-forecast; which means that the company does not have enough products and the cost of lost sales increases (Dalrymple, 1987; Huang et al., 2010; Lawrence et al., 2000; Mahmoud et al., 1988). Sales forecasting is therefore an important factor of a company's profitability and market share (Huang et al., 2010; Lee, 2000). Forecasts are also of significance in management decisions as they are explicitly or implicitly based on predictions about the future (Herbig et al., 1993; Lee, 2000). For a business to survive it is important to reach the customers at least as fast as competitors. The more accurate manager decisions about the future are, the more successful will companies be when adapting to different situations (Herbig et al., 1993). It is therefore important that all decisions in a company or even the supply chain are based on a single, shared and accurate forecast (Mentzer & Bienstock, 1998). Technologies in forecasting have developed from having the same technique for all products to instead using technology that adapts techniques for each product in order to achieve higher accuracy (Mentzer & Kahn, 1997; Mentzer & Schroeter, 1993). However, if compared to other areas within the company, the performance of

forecasts has improved remarkably little, even among more successful companies (Mentzer et al., 1999; Moon et al., 2003).

Many techniques and solutions support sales forecasting both for the business to consumer (B2C) and B2B segments. They can be grouped as quantitative (relying on data collected over longer period of time) and qualitative, based on judgment, intuition and informed opinions (and inherently subjective) (Davis and Mentzer, 2007; Kerkkänen and Huiskonen, 2007; Ingram, LaForge, Avila, Schwepker and Williams, 2012; Armstrong and Green, 2014). If a company has a large number of stored transactions, it is possible to use probability estimation techniques based on the development of opportunity, i.e. Sales funnel (Lodato, 2006; Duran, 2008; Söhnchen and Albers, 2010). A survey of the leading companies in various industries has shown that companies relying on data-driven decision-making (DDDM) achieve better results (Provost and Fawcett, 2013). On average, the top one-third DDDM companies from their industry are on average 5% more productive and 6% more profitable compared to their competition (Brynjolfsson, Hitt and Kim, 2011; McAfee and Brynjolfsson, 2012)

Companies that rely on data-driven decision-making (DDDM) mainly use statistical ways, such as the expert inquiry method and time-series related algorithms. The expert inquiry method completely relies on human experience and the accuracy is not stable enough. The time-series algorithms including auto-regression, exponential smoothing method, and ARIMA model, which are using historical sales data to construct the model. These methods cannot make full use of related factors, for example, price, holiday, events/anomalies etc. so that it is hard to guarantee optimal prediction accuracy in environments' with complex changes. Various traditional machine learning (ML) techniques are introduced into the field of sales forecast. These methods can comprehensively use factors related to sales and improve the prediction accuracy. However, these models cannot directly process time-series data, nor can they well extract the hidden rules of the data. More recently, deep learning techniques, such as CNN and RNN have also shown to be competitive in this domain. And, LSTM [1, 65] is superior to other methods in prediction accuracy.

Although there is a huge body of literature and technological advancement on the subject of prediction (Fildes, Goodwin and Lawrence, 2006; McCarthy, Davis Golicic and Mentzer, 2006; Armstrong, Green and Graefe, 2015), there is a significant impossibility of successfully applying it to business implementations. Decision makers remain skeptical of the recommendations provided by forecast support systems (FSS) and rely on the application of their own mental models (Goodwin, Fildes, Lawrence and Stephens, 2011) whose resulting forecasts are often inadequate. If organisations want to improve their effectiveness and reduce the gap between prediction and realisation, they must consider these anchored mental models.

Fortunately, many experts and researchers have contributed to sales forecasting methods over the years. Although sales data is time series data, the impact of factors on sales cannot be ignored. To solve this problem, the classic regression model is applied, which is based on taking reasonable factors of influence for sales. However, it is very difficult to understand and identify the factors that are linearly related to sales. In order to stand out from the competition, companies focus on flexible customer service, speed and adherence to delivery dates at reasonable prices [7]. Shortened product life cycles, fluctuations in customer behaviour and the need to respond immediately to market fluctuations are just some of the challenges in this regard. In order to reduce short-term changes in the supply chain, it is vital to implement effective sales forecasting models that allow companies to prepare for future situations in advance [2]. The available prediction algorithms in the literature as well as in commercial (ERP) systems are constantly increasing in terms of quantity and complexity.

In addition, computing power and storage capacity have become much less expensive, which opens up new forecasting possibilities for companies [18]. However, both quantitative and

qualitative sales forecasting models in some cases are not suitable for producing sufficient quality forecasting due to the high and rapid market fluctuations.

To sum up, there are still three main difficulties in sales forecasting problem:

a. Massive data increase the difficulty of computing and modelling.
b. There is a complex non-linear relationship between influencing variables and sales.
c. Sales is influenced by a time factor. But the influence of time factor on sales is difficult to quantify.

2.3 Machine Learning

2.3.1 Introduction to Machine Learning

In order to obtain a precise sales forecasting despite high market requirements, methods of machine learning (ML) and the subcategory deep learning (DL) are increasingly used [2, 4, 5]. Both method types can be defined as sub-areas of artificial intelligence. The advantage of AI-based methods is an automatic analysis of patterns and dependencies in the input data, in order to use them for the subsequent forecast. As is the case with statistical methods, there is no generally valid AI-based method that generates an improved forecast for every situation. Rather, each method can be used to achieve different qualities depending on the application.

In Chapter 4, we will discuss in detail the method and ML models we use in this Thesis, but, first, let's give some theoretical information on AI and ML in general.

Artificial Intelligence (AI), also referred to as machine intelligence, describes intelligence demonstrated by machines, which is different from natural intelligence, the kind of intelligence displayed by humans. [47]

Machine learning (ML) is a subfield of AI, and computer science in that sense, that is concerned with building algorithms which, to be useful, rely on a collection of examples of some phenomenon. These examples can come from nature, be handcrafted by humans or generated by another algorithm. Machine learning can also be defined as the process of solving a practical problem by 1) gathering a dataset, and 2) algorithmically building a statistical model based on that dataset. [48]

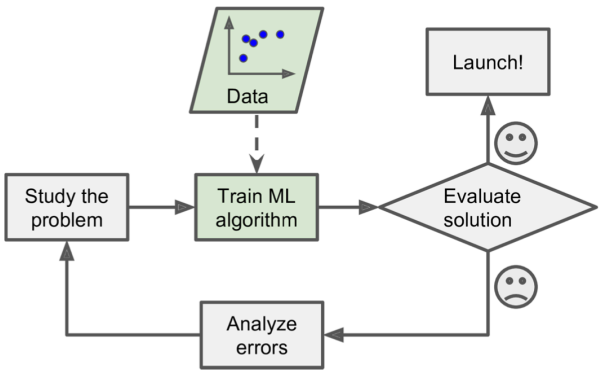


Fig. 2. Machine Learning approach [54]

Machine learning is about extracting knowledge from data. It is at the intersection of statistics, artificial intelligence, and computer science and is also known as predictive analytics or statistical learning. The application of machine learning methods has in recent years become ubiquitous in everyday life. From automatic recommendations of which movies to watch, to what food to order or which products to buy, to personalised online radio and recognising people in photos, many modern websites and devices have machine learning algorithms at their core. Outside of commercial applications, machine learning has had a tremendous influence on the way data-driven research is done today.

Finally, Machine Learning can help humans learn: ML algorithms can be inspected to see what they have learned (although for some algorithms this can be tricky). For instance, once the spam filter has been trained on enough spam, it can easily be inspected to reveal the list of words and combinations of words that it believes are the best predictors of spam. Sometimes this will reveal unsuspected correlations or new trends, and thereby lead to a better understanding of the problem.

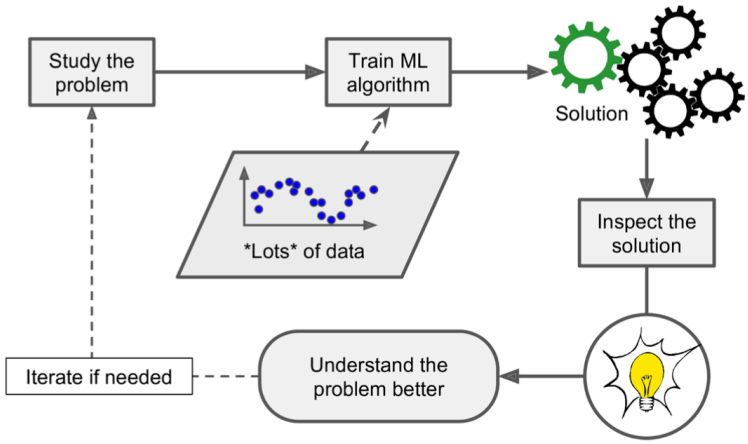


Fig. 3. Learning from Machine Learning [54]

Applying ML techniques to dig into large amounts of data can help discover patterns that were not immediately apparent. This is called data mining.

So, Machine Learning is great for:

- Problems for which existing solutions require a lot of hand-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better.
- Complex problems for which there is no good solution at all using a traditional approach: the best Machine Learning techniques can find a solution.
- Fluctuating environments: a Machine Learning system can adapt to new data.
- Getting insights about complex problems and large amounts of data.

Machine Learning can be based on supervised, semi-supervised, unsupervised and reinforcement learning.

Supervised Learning

In supervised learning, the dataset is the collection of labeled examples $\{(x_i,y_i)\}^N_{i=1}$. Each element x_i among N is called a feature vector. A feature vector is a vector in which each dimension j = 1,…, D contains a value that describes the example somehow. That value is called a feature and is denoted as $x^{(j)}$. For instance, if each example x in our collection represents a person, then the first feature, $x^{(1)}$, could contain height in cm, the second feature, $x^{(2)}$, could contain weight in kg, $x^{(3)}$ could contain gender, and so on. For all examples in the dataset, the feature at position j in the feature vector always contains the same kind of information. It means that if $x^{(2)}$ contains weight in kg in some example x_i, then $x_k^{(2)}$ will also contain weight in kg in every example x_k, k = 1,…, N. The label y_i can be either an element belonging to a finite set of classes {1, 2,…, C}, or a real number, or a more complex structure, like a vector, a matrix, a tree, or a graph. Unless otherwise stated, in this book y_i is either one of a finite set of classes or a real number. One can see a class as a category to which an example belongs.

The goal of a supervised learning algorithm is to use the dataset to produce a model that takes a feature vector x as input and outputs information that allows deducing the label for this feature vector. For instance, the model created using the dataset of people could take as input a feature vector describing a person and output a probability that the person has cancer.

Unsupervised Learning

In unsupervised learning, the dataset is a collection of unlabelled examples $\{x_i\}^N_{i=1}$. Again, x is a feature vector, and the goal of an unsupervised learning algorithm is to create a model that takes a feature vector x as input and either transforms it into another vector or into a value that can be used to solve a practical problem. For example, in clustering, the model returns the id of the cluster for each feature vector in the dataset. In dimensionality reduction, the output of the model is a feature vector that has fewer features than the input x; in outlier detection, the output is a real number that indicates how x is different from a typical example in the dataset.

Semi-Supervised Learning

In semi-supervised learning, the dataset contains both labeled and unlabeled examples. Usually, the quantity of unlabeled examples is much higher than the number of labeled examples. The goal of a semi-supervised learning algorithm is the same as the goal of the supervised learning algorithm. The hope here is that using many unlabeled examples can help the learning algorithm to produce a better model.

It seems counter-intuitive that learning could benefit from adding more unlabeled examples, as, unlabeled examples increase the problem's uncertainty. However, by adding unlabeled examples, we add more information: a larger data sample provides a more accurate picture of the probability distribution function of the labelled data. Theoretically, a learning algorithm should be able to leverage this additional information.

Reinforcement Learning

In reinforcement learning we have live feedback from the environment our machine runs and lives in. Here, our implementation is able to perceive the state of its environment as a features' vector and execute actions in every state. Different actions bring different rewards (or penalties) and change the environment's state. The goal of a reinforcement learning algorithm is to learn a policy.

A policy is a function (similar to the model in supervised learning) that takes the feature vector of a state as input and outputs an optimal action to execute in that state. The action is optimal if it maximises the expected average reward. Reinforcement learning solves a particular kind of problem where decision making is sequential, and the goal is long-term, such as game playing, robotics, resource management, or logistics.

Shallow vs. Deep Learning

A shallow learning algorithm learns the model parameters directly from the features of the training examples. Most supervised learning algorithms are shallow. The characteristic exceptions are neural network learning algorithms, especially those that create neural networks with more than one layer between input and output. Such neural networks are called deep neural networks. In deep neural network learning (or, simply, deep learning), unlike shallow learning, most model parameters are learnt not directly from the features of the training examples, but from the outputs of the previous levels. We will dive into Deep Learning and Neural Network architectures at the end of this chapter.

Classification vs. Regression

Classification is a problem of automatically assigning a label to an unlabelled example. Spam detection is a famous example of classification.

In machine learning, the classification problem is solved by a classification learning algorithm that takes a collection of labeled examples as inputs, and produces a model that can receive an unlabeled example and either directly output a label, or output a number that can be used by the analyst to deduce the label. In a classification problem, a label is a member of a finite set of classes. If the size of the set of classes is two (“sick”/“healthy”, “spam”/“not_spam”), we talk about binary classification (also called binomial). Multi-class classification (also called multinomial) is a classification problem with three or more classes.

Regression is a problem of predicting a real-valued label (often called a target) given an unlabeled example. For example, we can estimate a house price valuation based on house features, such as area, the number of bedrooms, location etc. This kind of target estimation is the essence of regression algorithms. A regression learning algorithm that takes a collection of labeled examples as inputs and produces a model that can receive an unlabeled example and output a target value. It is obvious that forecasting is related to regression (as we have already pointed out Sales Forecasting is more of a regression than a time-series problem) and, thus, we will mainly focus on the theoretical background of regression algorithms, as they will be used in the construction of our forecasting models.

So, a good representation of the AI, ML and DL space would be the following:

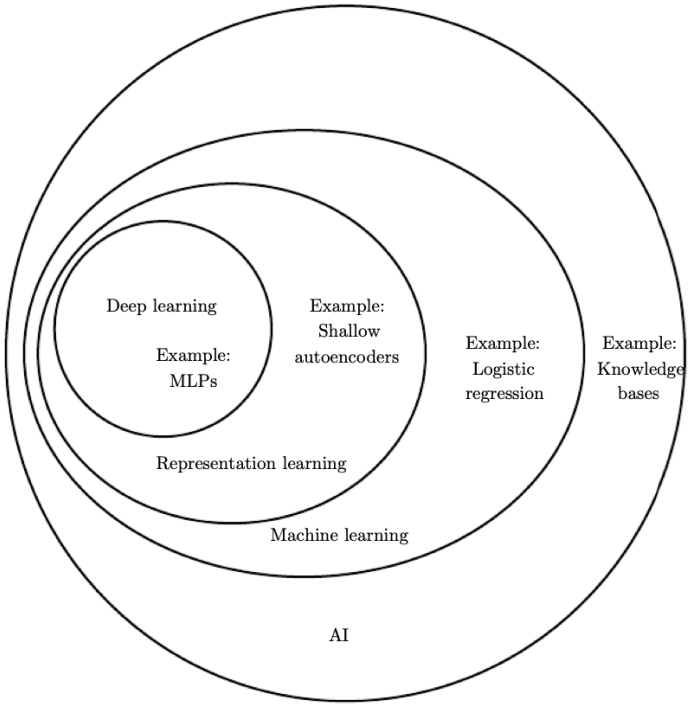


Fig. 4. Artificial Intelligence, Machine Learning and Deep Learning schematic [49]

Next, we present some of the most fundamental regression algorithms (Linear regression, Logistic Regression etc.) which will help us understand in depth the algorithms that we used in our work for FMCGs Sales Forecasting.

2.3.2 Linear Regression

Linear regression is a popular regression learning algorithm that learns a model which is a linear combination of features of the input example.

Problem Statement

We have a collection of labeled examples $\{(x_i,y_i)\}^N_{i=1}$, where N is the size of the collection, x_i is the D-dimensional feature vector of the example i = 1,…,N, y_i is a real-valued target and every feature $x_i(j)$, j=1,...,D, is also a real number. We want to build a model $f_{w,b}(x)$ as a linear combination of features of example x:

$$f_{w,b}(x) = wx + b \qquad (1)$$

where w is a D-dimensional vector of parameters and b is a real number. The notation $f_{w,b}$ means that the model f is parametrised by two values: w and b.

We will use the model to predict the unknown y for a given x like this: $y \leftarrow f_{w,b}(x)$. Two models parametrised by two different pairs (w, b) will likely produce two different predictions when applied to the same example. We want to find the optimal values (w^*,b^*). Obviously, the optimal values of parameters define the model that makes the most accurate predictions.

Linear regression is chosen to be as close to all training examples as possible. This is essential by looking at the illustration in Figure 5. It displays the regression line (in red) for one-dimensional examples (blue dots).

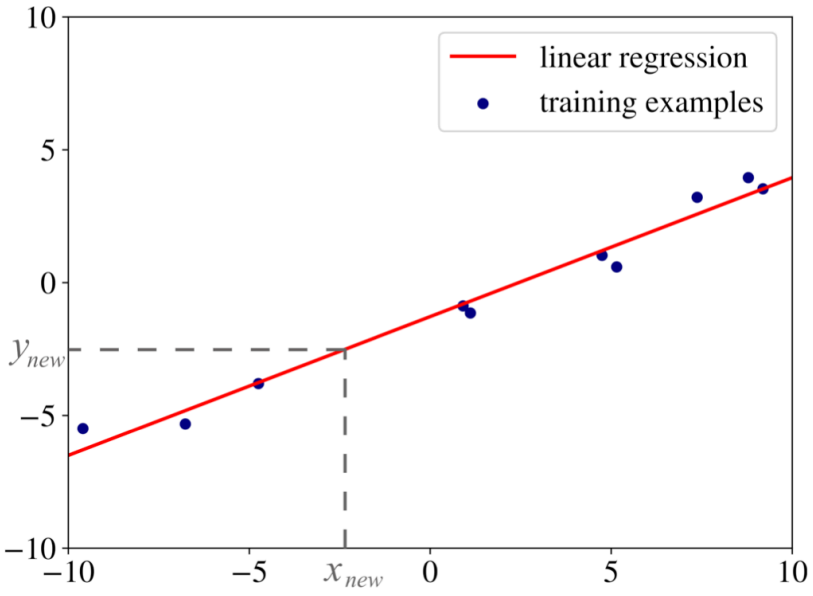


Fig. 5. Linear Regression for one-dimensional examples [54].

We can use this line to predict the value of the target y_{new} for a new unlabelled input example x_{new}. If our examples are D-dimensional feature vectors (for D > 1), the only difference with the one-dimensional case is that the regression model is not a line but a plane (for two dimensions) or a hyperplane (for D > 2). So, this is the reason it is essential to have the requirement that the regression hyperplane lies as close to the training examples as possible: if the red line in Figure 5 was far from the blue dots, the prediction y_{new} would have fewer chances to be correct.

Solution

So, that means that the optimisation procedure, which we use to find the optimal values for w^* and b^*, tries to minimise the following objective function:

$$\frac{1}{N}\sum_{i=1\ldots N}(f_{\mathbf{w},b}(\mathbf{x}_i)-y_i)^2.$$

The expression $(f_{w,b}(x_i) - y_i)^2$ in the above objective is called the loss function. It's a measure of penalty for misclassification of example i. This particular choice of the loss function is called squared error loss. All model-based learning algorithms have a loss function and what we do is to find the best model by minimising the objective function known as the cost function. In linear regression, the cost function is given by the average loss, which is also called "empirical risk". The average loss, or empirical risk of a model is the average of all penalties obtained by applying the model to the training data.

Based on different decisions about the form of the model, the form of the loss function, and about the choice of the algorithm that minimises the average loss to find the best values of parameters,

we would end up inventing a different algorithm. However, the fact that it's different doesn't mean that it will work better.

We invent new learning algorithms for one of the two main reasons:

1. The new algorithm solves a specific practical problem better than the existing algorithms.
2. The new algorithm has better theoretical guarantees on the quality of the model it produces.

One practical justification of the choice of the linear form for the model is that it's simple. Another consideration is that linear models rarely overfit. Overfitting is the property of a model such that the model predicts very well labels of the examples used during training but frequently makes errors when applied to examples that weren't seen by the learning algorithm during training. An example of overfitting in regression is shown in Figure 6.

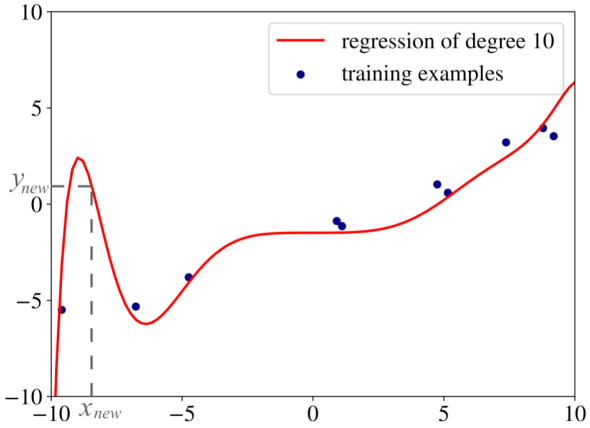


Fig. 6. Example of Overfitting. [54]

The data used to build the red regression line is the same as in Figure 5. The difference is that this time, this is the polynomial regression with a polynomial of degree 10. The regression line predicts almost perfectly the targets almost all training examples, but will likely make significant errors on new data, as you can see in Figure 5 for x_{new}. The case of overfitting becomes more and more relevant the more complicated and models we implement. We will see that especially in Deep Learning implementations it quickly becomes a problem, as these very flexible solutions (just like the tenth degree polynomial) tend to easily overfit.

2.3.3 Logistic Regression

Logistic regression has a misguiding name, as it is not really a regression, but a classification learning algorithm. The name comes from statistics and is due to the fact that the mathematical formulation of logistic regression is similar to that of linear regression. Here we give an explanation of logistic regression on the case of binary classification. However, it can naturally be extended to multi-class classification.

Problem Statement

In logistic regression, we want to model y_i as a linear function of x_i (just like in linear regression). However, with a binary y_i this is not straightforward. The linear combination of features such as $wx_i + b$ is a function that spans from minus infinity to plus infinity, while y_i has only two possible values.

At the time where the absence of computers required scientists to perform manual calculations, they were eager to find a linear classification model. They figured out that if we define a negative label as 0 and the positive label as 1, we would just need to find a simple continuous function whose codomain is (0,1). In such a case, if the value returned by the model for input x is closer to 0, then we assign a negative label to x; otherwise, the example is labeled as positive. One function that has such a property is the standard logistic function (also known as the sigmoid function):

$$f(x) = \frac{1}{1 + e^{-x}},$$

where e is the base of the natural logarithm (also called Euler's number; e^x is also known as the exp(x) function in programming languages).

The logistic regression model looks like this:

$$f_{\mathbf{w},b}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{1 + e^{-(\mathbf{w}\mathbf{x}+b)}}.$$

By looking at the graph of the standard logistic function, we can see how well it fits our classification purpose: if we optimise the values of w and b appropriately, we could interpret the output of f(x) as the probability of y_i being positive. For example, if it's higher than or equal to the threshold 0.5 we would say that the class of x is positive; otherwise, it's negative. In practice, the choice of the threshold could be different depending on the problem. Now, the main question is how one finds the optimal w^* and b^*. In linear regression, the goal was to minimise the empirical risk which was defined as the average squared error loss, also known as the mean squared error or MSE (which will be one of the main key metrics in our models too).

Solution

In logistic regression, on the other hand, the aim is to maximise the likelihood of our training set according to the model. In statistics, the likelihood function defines how likely the observation is according to a specific model.

$$L_{\mathbf{w},b} \stackrel{\text{def}}{=} \prod_{i=1\ldots N} f_{\mathbf{w},b}(\mathbf{x}_i)^{y_i}(1 - f_{\mathbf{w},b}(\mathbf{x}_i))^{(1-y_i)}. \quad (4)$$

The expression $f_{\mathbf{w},b}(\mathbf{x})^{y_i}(1 - f_{\mathbf{w},b}(\mathbf{x}))^{(1-y_i)}$ may look scary but it's just a fancy mathematical way of saying: "$f_{\mathbf{w},b}(\mathbf{x})$ when $y_i = 1$ and $(1 - f_{\mathbf{w},b}(\mathbf{x}))$ otherwise". Indeed, if $y_i = 1$, then $(1 - f_{\mathbf{w},b}(\mathbf{x}))^{(1-y_i)}$ equals 1 because $(1 - y_i) = 0$ and we know that anything power 0 equals 1. On the other hand, if $y_i = 0$, then $f_{\mathbf{w},b}(\mathbf{x})^{y_i}$ equals 1 for the same reason.

For instance, let's have a labeled example (x_i,y_i) in our training data. Assume also that we found (guessed) some specific values w^* and b^* of our parameters. By applying our model f_{w^*,b^*} to x_i we will get some value $0 < p < 1$ as output. If y_i is the positive class, the likelihood of y_i being the

positive class, according to our model, is given by p. Similarly, if y_i is the negative class, the likelihood of it being the negative class is given by 1-p.
The optimisation criterion in logistic regression is that of maximum likelihood. Instead of minimising the average loss, like in linear regression, we maximise the likelihood of the training data.

You may have noticed that we used the product operator $\prod$ in the objective function instead of the sum operator $\sum$ which was used in linear regression. It's because the likelihood of observing N labels for N examples is the product of likelihoods of each observation (assuming that all observations are independent of one another, which is the case). You can draw a parallel with the multiplication of probabilities of outcomes in a series of independent experiments in the probability theory. Because of the exp function used in the model, in practice, it's more convenient to maximise the log-likelihood instead of likelihood. The log-likelihood is defined like follows:

$$LogL_{\mathbf{w},b} \stackrel{\text{def}}{=} \ln(L(_{\mathbf{w},b}(\mathbf{x})) = \sum_{i=1}^{N} y_i \ln f_{\mathbf{w},b}(\mathbf{x}) + (1 - y_i) \ln (1 - f_{\mathbf{w},b}(\mathbf{x})).$$

Because ln is a strictly increasing function, maximising this function is the same as maximising its argument, and the solution to this new optimisation problem is the same as the solution to the original problem.

Contrary to linear regression, there's no closed form solution to the above optimisation problem. A typical numerical optimisation procedure used in such cases is gradient descent. We will see more about gradient descent in this chapter.

2.3.4 Decision Tree Learning

A decision tree is an acyclic graph that can be used to make decisions. In each branching node of the graph, a specific feature j of the feature vector is examined. If the value of the feature is below a specific threshold, then the left branch is followed; otherwise, the right branch is followed. As the leaf node is reached, the decision is made about the class to which the example belongs. As the title of the section suggests, a decision tree can be learned from data.

Problem Statement

Like previously, we have a collection of labeled examples; labels belong to the set $\{0,1\}$. We want to build a decision tree that would allow us to predict the class given a feature vector.

Solution

There are various formulations of the decision tree learning algorithm. For the purpose of the theoretical background of this Thesis, we consider just one, called ID3.
The optimisation criterion, in this case, is the average log-likelihood:

$$\frac{1}{N} \sum_{i=1\ldots N} (f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2.$$

where f_{ID3} is a decision tree.

By now, it looks very similar to logistic regression. However, contrary to the logistic regression learning algorithm which builds a parametric model f_{w^*,b^*} by finding an optimal solution to the optimization criterion, the ID3 algorithm optimizes it approximately by constructing a nonparametric model $f_{ID3}(x) = Pr(y = 1|x)$.

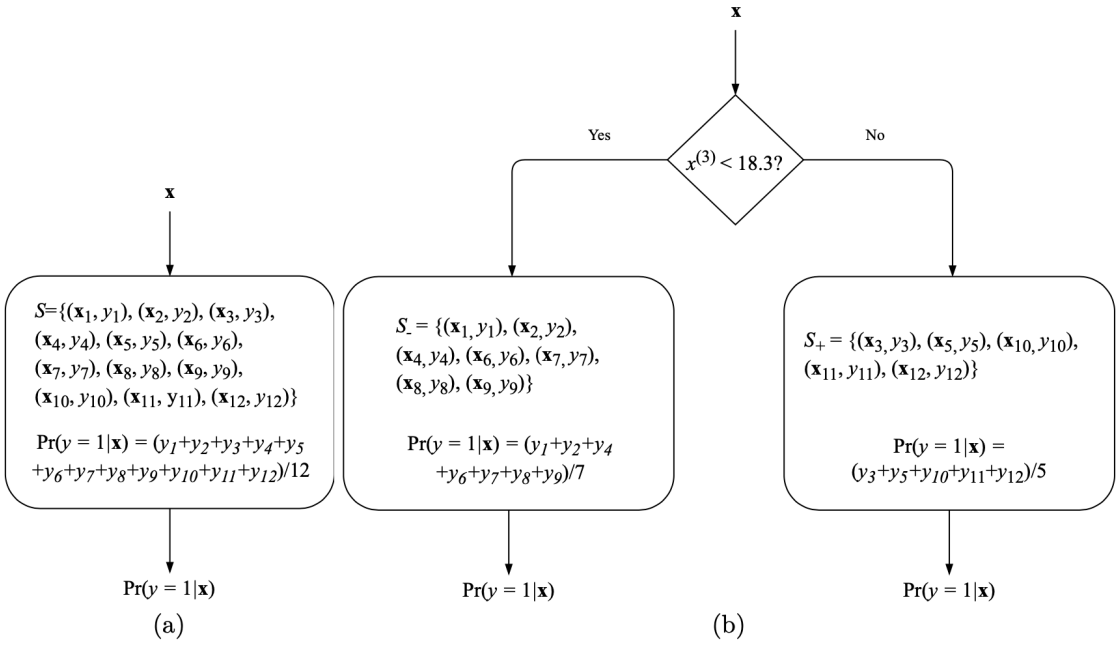


Fig. 7. An illustration of a decision tree building algorithm. The set S contains 12 labeled examples. (a) In the beginning, the decision tree only contains the start node; it makes the same prediction for any input. (b) The decision tree after the first split; it tests whether feature 3 is less than 18.3 and, depending on the result, the prediction is made in one of the two leaf nodes. [54]

The ID3 learning algorithm works as follows. Let S denote a set of labeled examples. In the def beginning, the decision tree only has a start node that contains all examples: $S = \{(x_i, y_i)\}^N_{i=1}$. Start with a constant model f^S_{ID3} defined as:

$$f^{\mathcal{S}}_{ID3} \stackrel{\text{def}}{=} \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},y)\in\mathcal{S}} y.$$

The prediction given by the above model, $f^S_{ID3}(x)$, would be the same for any input x. The corresponding decision tree built using a toy dataset of twelve labeled examples is shown in Fig.7.a.

Then we search through all features j = 1,…, D and all thresholds t, and split the set S $^{\text{def}}$ (j) into two subsets: $S_- = \{(x,y)|(x,y) \in S, x < t\}$ and $S_+ = \{(x,y)|(x,y) \in S, x \geq t\}$. The two new subsets would go to two new leaf nodes, and we evaluate, for all possible pairs (j, t) how good the split with pieces S_- and S_+ is. Finally, we pick the best such values (j, t), split S into S_+ and S_-, form two new leaf nodes, and continue recursively on S_+ and S_- (or quit if no split produces a model that's sufficiently better than the current one). A decision tree after one split is illustrated in Fig.7.b. In ID3, the goodness of a split is estimated by using the criterion called entropy. Entropy

is a measure of uncertainty about a random variable. It reaches its maximum when all values of the random variables are equiprobable. Entropy reaches its minimum when the random variable can have only one value. The entropy of a set of examples S is given by,

$$H(\mathcal{S}) \stackrel{\text{def}}{=} -f_{ID3}^{\mathcal{S}} \ln f_{ID3}^{\mathcal{S}} - (1 - f_{ID3}^{\mathcal{S}}) \ln(1 - f_{ID3}^{\mathcal{S}})$$

When we split a set of examples by a certain feature j and a threshold t, the entropy of a split, $H(S_{-},S_{+})$, is simply a weighted sum of two entropies:

$$H(\mathcal{S}_{-}, \mathcal{S}_{+}) \stackrel{\text{def}}{=} \frac{|\mathcal{S}_{-}|}{|\mathcal{S}|} H(\mathcal{S}_{-}) + \frac{|\mathcal{S}_{+}|}{|\mathcal{S}|} H(\mathcal{S}_{+})$$

So, in ID3, at each step, at each leaf node, we find a split that minimises the entropy or we stop at this leaf node.

The algorithm stops at a leaf node in any of the below situations:

- All examples in the leaf node are classified correctly by the one-piece model.
- We cannot find an attribute to split upon.
- The split reduces the entropy less than some ε (the value for which has to be found experimentally).
- The tree reaches some maximum depth d (also has to be found experimentally).

Because in ID3, the decision to split the dataset on each iteration is local (doesn't depend on future splits), the algorithm doesn't guarantee an optimal solution. The model can be improved by using techniques like backtracking during the search for the optimal decision tree at the cost of possibly taking longer to build a model.

The most widely used formulation of a decision tree learning algorithm is called C4.5. It has several additional features as compared to ID3:

- it accepts both continuous and discrete features.
- it handles incomplete examples.
- it solves overfitting problem by using a bottom-up technique known as "pruning".

Pruning consists of going back through the tree once it's been created and removing branches that don't contribute significantly enough to the error reduction by replacing them with leaf nodes.
The entropy-based split criterion intuitively makes sense: entropy reaches its minimum of 0 when all examples in S have the same label; on the other hand, the entropy is at its maximum of 1 when exactly one-half of examples in S is labeled with 1, making such a leaf useless for classification. The only remaining question is how this algorithm approximately maximises the average log-likelihood criterion.

2.3.5 Gradient Descent

We will show how gradient descent works as it can effectively provide us with a background to later discuss most of the algorithms in this Thesis. Our example here, is taken from [54] and it's about finding the solution to a linear regression problem, using a dataset with only one feature. So, the optimisation criterion will have two parameters: w and b. The extension to multi-dimensional training data is straightforward; instead of having variables $w^{(1)}$, $w^{(2)}$, and b for two-

dimensional data, we would have $w^{(1)}$, $w^{(2)}$, $w^{(3)}$, and b for three-dimensional data and so on for higher dimensions.

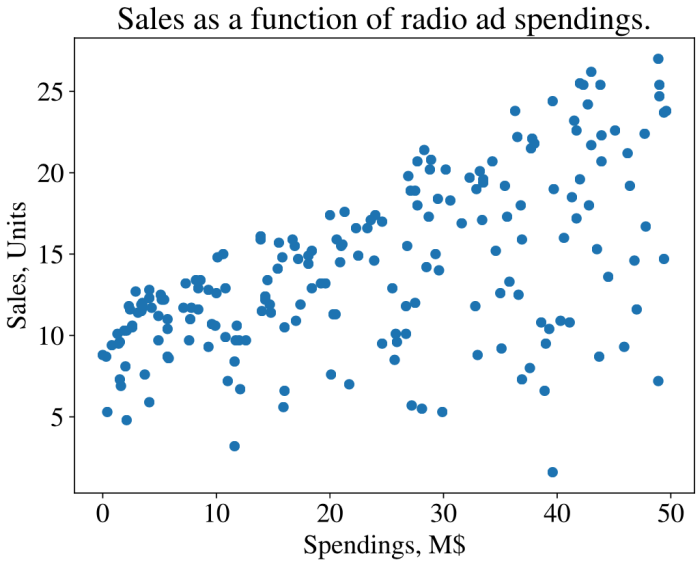


Fig. 8: The Y-axis corresponds to the sales in units (the quantity we want to predict), the X-axis corresponds to our feature: the spendings on radio ads in M$. [54]

Our dataset has two columns: the spendings of various companies on radio advertising each year and their annual sales in terms of units sold. We want to build a regression model that can be used to predict units sold based on how much a company spends on radio advertising. Each row in the dataset represents one specific company:

Company	Spendings, M$	Sales, Units
1	37.8	22.1
2	39.3	10.4
3	45.9	9.3
4	41.3	18.5
..	..	..

We have data for 200 companies, so we have 200 training examples in the form (x_i,y_i) = $(Spendings_i,Sales_i)$. Figure 8 shows all examples on a 2D plot.

Remember that the linear regression model looks like this: f(x) = wx + b. We don't know what the optimal values for w and b are and we want to learn them from data. To do that, we look for such values for w and b that minimize the mean squared error:

$$l \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N}(y_i - (wx_i + b))^2$$

Gradient descent starts with calculating the partial derivative for every parameter and then we start to subtract partial derivatives from the values of parameters. If a derivative is positive at some point, then the function grows at this point. And, so, because we want to minimise the objective function, when the derivative is positive we know that we need to move our parameter

in the opposite direction (to the left on the axis of coordinates). When the derivative is negative (the function is decreasing), we need to move our parameter to the right to decrease the value of the function even more. Subtracting a negative value from a parameter moves it to the right.

At the next epoch, we recalculate partial derivatives using eq. 1 with the updated values of w and b; we continue the process until convergence. Typically, we need many epochs until we start seeing that the values for w and b don't change much after each epoch; then we stop.

$$\begin{aligned} \frac{\partial l}{\partial w} &= \frac{1}{N}\sum_{i=1}^{N} -2x_i(y_i - (wx_i + b)); \\ \frac{\partial l}{\partial b} &= \frac{1}{N}\sum_{i=1}^{N} -2(y_i - (wx_i + b)). \end{aligned} \tag{1}$$

To find the partial derivative of the term $(y_i - (wx + b))^2$ with respect to w I applied the *chain rule*. Here, we have the chain $f = f_2(f_1)$ where $f_1 = y_i - (wx+b)$ and $f_2 = f_1^2$. To find a partial derivative of f with respect to w we have to first find the partial derivative of f with respect to f_2 which is equal to $2(y_i - (wx + b))$ (from calculus, we know that the derivative $\frac{\partial}{\partial x}x^2 = 2x$) and then we have to multiply it by the partial derivative of $y_i - (wx + b)$ with respect to w which is equal to $-x$. So overall $\frac{\partial l}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} -2x_i(y_i - (wx_i + b))$. In a similar way, the partial derivative of l with respect to b, $\frac{\partial l}{\partial b}$, was calculated.

Gradient descent proceeds in **epochs**. An epoch consists of using the training set entirely to update each parameter. In the beginning, the first epoch, we initialize[2] $w \leftarrow 0$ and $b \leftarrow 0$. The partial derivatives, $\frac{\partial l}{\partial w}$ and $\frac{\partial l}{\partial b}$ given by eq. 1 equal, respectively, $\frac{-2}{N}\sum_{i=1}^{N} x_i y_i$ and $\frac{-2}{N}\sum_{i=1}^{N} y_i$. At each epoch, we update w and b using partial derivatives. The learning rate α controls the size of an update:

$$\begin{aligned} w &\leftarrow w - \alpha\frac{\partial l}{\partial w}; \\ b &\leftarrow b - \alpha\frac{\partial l}{\partial b}. \end{aligned} \tag{2}$$

At the next graph (Figure 9), we see the evolution of the regression line through gradient descent epochs.

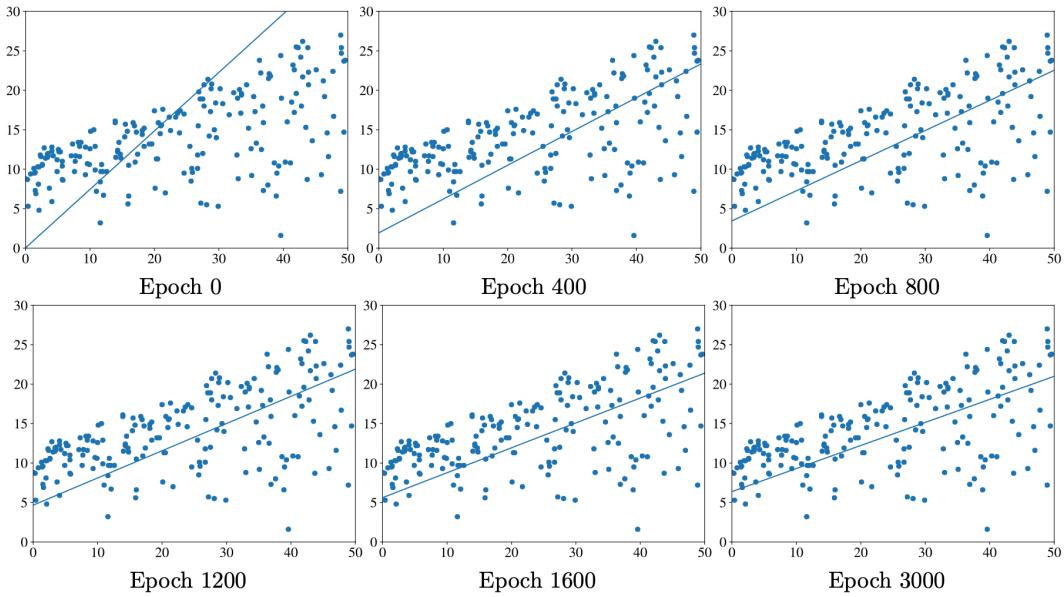


Fig. 9: The evolution of the regression line through gradient descent epochs [54].

Gradient descent is sensitive to the choice of the learning rate α. It is also slow for large datasets. Fortunately, several significant improvements to this algorithm have been proposed.

Notice that gradient descent and its variants are not machine learning algorithms. They are solvers of minimisation problems in which the function to minimise has a gradient (in most points of its domain).

At the end of this chapter (Chapter 2, "Theoretical Background") we will also establish Deep Learning (2.4) which is strongly related to Gradient Descent logic and epochs.

2.3.6 Ensemble Learning

The fundamental algorithms that we have already considered (Linear Regression, Logistic Regression, Random Trees etc.) have their limitations. Because of their simplicity, sometimes they cannot produce a model accurate enough for your problem. To achieve higher accuracy, one can try deep neural networks. However, in practice, deep neural networks require a significant amount of labeled data which you might not have. Another approach to boost the performance of simple learning algorithms is ensemble learning. Later, we will talk in detail about ensemble learning/meta-learning and meta-learners' building techniques in methodologies.

Ensemble learning is a learning paradigm that, instead of trying to learn one extremely accurate model, focuses on training a large number of low-accuracy models and then combining the predictions given by those weak models to obtain a high-accuracy meta-model.

Low-accuracy models are usually learned by weak learners, that is learning algorithms that cannot learn complex models, and thus are typically fast at the training and at the prediction time. The most frequently used weak learner is a decision tree learning algorithm in which we often stop splitting the training set after just a few iterations. The obtained trees are shallow and not particularly accurate, but the idea behind ensemble learning is that if the trees are not identical and each tree is at least slightly better than random guessing, then we can obtain high accuracy by combining a large number of such trees.

To obtain the prediction for input x, the predictions of each weak model are combined using some sort of weighted voting. The specific form of vote weighting depends on the algorithm, but, independently of the algorithm, the idea is the same: if the council of weak models predicts that the message is spam, then we assign the label spam to x.

Two principal ensemble learning methods are boosting and bagging.

- Boosting consists of using the original training data and iteratively create multiple models by using a weak learner. Each new model would be different from the previous ones in the sense that the weak learner, by building each new model tries to fix previous models' errors. The final ensemble model is a certain combination of those multiple weak models built iteratively.

- Bagging consists of creating many "copies" of the training data (each copy is slightly different from another) and then apply the weak learner to each copy to obtain multiple weak models and then combine them. A widely used and effective machine learning algorithm based on the idea of bagging is random forest.

Random Forest

Random Forest is the closest algorithm to the most "vanilla' bagging algorithm. About the "vanilla" bagging algorithm, consider the following: Given a training set, we create B random samples S_b (for each b = 1,...,B) of the training set and build a decision tree model f_b using each sample S_b as the training set. To sample S_b for some b, we do the sampling with replacement. This means that we start with an empty set, and then pick at random an example from the training set and put its exact copy to S_b by keeping the original example in the original training set. We keep picking examples at random until the $|S_b| = N$.

After training, we have B decision trees. The prediction for a new example x is obtained as the average of B predictions:

$$y \leftarrow \hat{f}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{B} \sum_{b=1}^{B} f_b(\mathbf{x})$$

in the case of regression, or by taking the majority vote in the case of classification.

Random forest is different from the vanilla bagging in just one way. It uses a modified tree learning algorithm that inspects, at each split in the learning process, a random subset of the features. The reason for doing this is to avoid the correlation of the trees: if one or a few features are very strong predictors for the target, these features will be selected to split examples in many trees. This would result in many correlated trees in our "forest." Correlated predictors cannot help in improving the accuracy of prediction. The main reason behind a better performance of model ensembling is that models that are good will likely agree on the same prediction, while bad models will likely disagree on different ones. Correlation will make bad models more likely to agree, which will hamper the majority vote or the average.

The most important hyper-parameters to tune are the number of trees, B, and the size of the random subset of the features to consider at each split.

Random forest is one of the most widely used ensemble learning algorithms. The reason it is that popular is that by using multiple samples of the original dataset, we reduce the variance of the final model. We need to constantly take into account that low variance means low overfitting. Overfitting happens when our model tries to explain small variations in the dataset because our dataset is just a small sample of the population of all possible examples of the phenomenon we try to model. If we were unlucky with how our training set was sampled, then it could contain some undesirable (but unavoidable) artifacts: noise, outliers and overrepresented or

underrepresented examples. By creating multiple random samples with replacement of our training set, we reduce the effect of these artifacts.

Gradient Boosting

Another effective ensemble learning algorithm is gradient boosting. As the name points out, it is based on the idea of boosting. Let's first look at gradient boosting for regression. To build a strong regressor, we start with a constant model $f = f_0$ (just like we did in ID3):

$$f = f_0(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N} y_i$$

Then we modify labels of each example i = 1,…, N in our training set like follows:

$$\hat{y}_i \leftarrow y_i - f(\mathbf{x}_i)$$

where $y_{\text{hat-i}}$, called the residual, is the new label for example x_i. Now we use the modified training set, with residuals instead of original labels, to build a new decision tree model, f_1. The boosting model is now defined as $f = f_0 + \alpha f_1$, where α is the learning rate (hyperparameter).

Then we recompute the residuals and replace the labels in the training data once again, train the new decision tree model f_2, redefine the boosting model as $f = f_0 + \alpha f_1 + \alpha f_2$ and the process continues until the predefined maximum M of trees are combined.

So, by computing the residuals, we find how well (or poorly) the target of each training example is predicted by the current model f. We then train another tree to fix the errors of the current model (this is why we use residuals instead of real labels) and add this new tree to the existing model with some weight α. Therefore, each additional tree added to the model partially fixes the errors made by the previous trees until the maximum number M (another hyperparameter) of trees are combined.

In gradient boosting, we don't calculate any gradient contrary to what we did for linear regression. To see the similarity between gradient boosting and gradient descent remember why we calculated the gradient in linear regression: we did that to get an idea on where we should move the values of our parameters so that the MSE cost function reaches its minimum. The gradient showed the direction, but we didn't know how far we should go in this direction, so we used a small step α at each iteration and then reevaluated our direction. The same happens in gradient boosting. However, instead of getting the gradient directly, we use its proxy in the form of residuals: they show us how the model has to be adjusted so that the error (the residual) is reduced.

The three principal hyperparameters to tune in gradient boosting are the number of trees, the learning rate, and the depth of trees. Apart from affecting the model accuracy, the depth of trees also affects the speed of training and prediction: the shorter, the faster.

It can be shown that training on residuals optimizes the overall model f for the mean squared error criterion. You can see the difference with bagging here: boosting reduces the bias (or underfitting) instead of the variance. As such, boosting can overfit. However, by tuning the depth and the number of trees, overfitting can be largely avoided.

Gradient boosting for classification is similar, but the steps are slightly different. Let's consider the binary case. Assume we have M regression decision trees. Similarly to logistic regression, the prediction of the ensemble of decision trees is modeled using the sigmoid function:

$$\Pr(y = 1|\mathbf{x}, f) \stackrel{\text{def}}{=} \frac{1}{1 + e^{-f(\mathbf{x})}},$$

where $f(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{m=1}^{M} f_m(\mathbf{x})$ and f_m is a regression tree.

Again, like in logistic regression, we apply the maximum likelihood principle by trying to find such an f that maximizes $L_f = \sum_{i=1}^{N} \ln\left[\Pr(y_i = 1|\mathbf{x}_i, f)\right]$. Again, to avoid numerical overflow, we maximize the sum of log-likelihoods rather than the product of likelihoods.

The algorithm starts with the initial constant model $f = f_0 = \frac{p}{1-p}$, where $p = \frac{1}{N}\sum_{i=1}^{N} y_i$. (It can be shown that such initialization is optimal for the sigmoid function.) Then at each iteration m, a new tree f_m is added to the model. To find the best f_m, first the partial derivative g_i of the current model is calculated for each $i = 1, \ldots, N$:

$$g_i = \frac{dL_f}{df},$$

where f is the ensemble classifier model built at the previous iteration $m - 1$. To calculate g_i we need to find the derivatives of $\ln\left[\Pr(y_i = 1|\mathbf{x}_i, f)\right]$ with respect to f for all i. Notice that $\ln\left[\Pr(y_i = 1|\mathbf{x}_i, f)\right] \stackrel{\text{def}}{=} \ln\left[\frac{1}{1+e^{-f(\mathbf{x}_i)}}\right]$. The derivative of the right-hand term in the previous equation with respect to f equals to $\frac{1}{e^{f(\mathbf{x}_i)}+1}$.

We then transform our training set by replacing the original label y_i with the corresponding partial derivative g_i, and build a new tree f_m using the transformed training set. Then we find the optimal update step ρ_m as:

$$\rho_m \leftarrow \arg\max_{\rho} L_{f+\rho f_m}.$$

At the end of iteration m, we update the ensemble model f by adding the new tree f_m:

$$f \leftarrow f + \alpha\rho_m f_m.$$

We iterate until m = M, then we stop and return the ensemble model f.

Gradient boosting is one of the most powerful machines learning algorithms. Not just because it creates very accurate models, but also because it is capable of handling huge datasets with millions of examples and features. It usually outperforms random forest in accuracy but, because of its sequential nature, can be significantly slower in training. More about Gradient boosting algorithms and Ensemble Learning and real-world Ensembling implementations will be shown in Chapter 3 (“Previous Work”). Also, the power of Gradient boosting algorithms will become obvious in our Experiments (Chapter 5).

2.3.7 Real-world problems of ML implementations

Regarding our challenges in Sales Forecasting for FMCGs, we will use algorithms (all of which will be explained in detail in Chapter 4) based on the main ideas of ML algorithm building, metrics and algorithm evaluation to those that have mentioned here.

However, we will face three problems: sparse data, user preference and developing a single model with good performance.

Sparse data appears as:

- Among all the products, only a small part belongs to the best-selling with sales records every day. Most products only have sales records in certain periods and none in other periods.
- The smaller the division dimension of the products, the more obvious the sparseness of the data. But we need smaller granularity data because it is accurate for decision-making of daily operation.

More about the real-world problems of ML Sales Forecasting will be mentioned in Chapter 3 (“Previous Work”), Chapter 4 (“Method & Models”) and Chapter 5 (“Experiments”).

To complete this chapter, we need to give one more aspect of ML Sales Forecasting. Since no model can achieve the theoretical 100% it is very important to choose the correct "bias" of our model predictions. That is, +0.1 and -0.1 MSE may theoretically be an equivalent error at an absolute value, but in practice they lead to very different profits for an organisation, since, for example, when the storage cost is less than the exhausted cost, the appropriate overestimation of forecast is beneficial for immediate recovery. Conversely, when the cost of storage is greater than the cost of depleting inventory, then we would prefer our model to value lower than higher sales. Also, when the warehousing cost is less than the out-of-stock cost, appropriate overestimate the prediction is beneficial for promptly restock. However, there is little literature and very few experts and researchers study these limits and bias when forecasting sales.

In Chapter 5, we will further discuss model bias, as we build, run and evaluate our models. With a Stacking Classifier/Regressor (SC/R) and a Mixture of Experts (MoE) we will show how we successfully did fast and improved Ensembling. In a real prediction system, although deep learning models are effective and can be extremely accurate, they are harder to maintain and need much more data to start with [37]. A single ML model with good performance is much more valuable [19]. So, this is the reason this Thesis’ final goal is to build a unified Machine Learning method with very high accuracy for Sales forecasting for fast-moving consumer goods.

2.4 Deep Learning

The simple machine learning algorithms described in this chapter work very well on a wide variety of important problems. However, they have not succeeded in solving the central problems in AI, such as recognising speech or recognising objects. The development of deep learning was motivated in part by the failure of traditional algorithms to generalize well on such AI tasks.

This section is about how the challenge of generalising to new examples becomes exponentially more difficult when working with high-dimensional data, and how the mechanisms used to achieve generalisation in traditional machine learning are insufficient to learn complicated functions in high-dimensional spaces. Such spaces also often impose high computational costs. Deep learning was designed to overcome these and other obstacles.

Deep learning is a specific subfield of machine learning: a new take on learning representations from data that puts an emphasis on learning successive layers of increasingly meaningful representations. The deep in deep learning isn’t a reference to any kind of deeper understanding achieved by the approach; rather, it stands for this idea of successive layers of representations. How many layers contribute to a model of the data is called the depth of the model. Other appropriate names for the field could have been layered representations learning and hierarchical representations learning. Modern deep learning often involves tens or even hundreds of successive layers of representations, and they’re all learned automatically from exposure to

training data. Meanwhile, other approaches to machine learning tend to focus on learning only one or two layers of representations of the data; hence, they're sometimes called shallow learning. [49]

In deep learning, these layered representations are (almost always) learned via models called neural networks, structured in literal layers stacked on top of each other. The term neural network is a reference to neurobiology, but although some of the central concepts in deep learning were developed in part by drawing inspiration from our understanding of the brain, deep-learning models are not models of the brain. There's no evidence that the brain implements anything like the learning mechanisms used in modern deep-learning models. Many people have the misbelief that neural networks work like the brain or are modeled after the brain, but that isn't true. In fact, a neural network (NN), just like a regression model, is a mathematical function: $y = f_{NN}(x)$.

The function f_{NN} has a particular form: it's a nested function. These nested functions form neural network layers. So, for a 3-layer neural network that returns a scalar, f_{NN} looks like this:

$$y = f_{NN}(x) = f_3(f_2(f_1(x)))$$

In the above equation, f_1 and f_2 are vector functions of the following form: $\boldsymbol{f}_l(\mathbf{z}) \stackrel{\text{def}}{=} \boldsymbol{g}_l(\mathbf{W}_l\mathbf{z} + \mathbf{b}_l)$

where l is called the layer index and can span from 1 to any number of layers. The function g_l is called an activation function. It is a fixed, usually nonlinear function chosen by the data analyst before the learning is started. The parameters W_l (a matrix) and b_l (a vector) for each layer are learned using the familiar gradient descent by optimising, depending on the task, a particular cost function (such as MSE). Compare the equation above with the equation for logistic regression, where you replace g_l by the sigmoid function, and you will not see any difference. The function f_3 is a scalar function for the regression task, but can also be a vector function depending on your problem.

We use a matrix W_l instead of a vector w_l because g_l is a vector function. Each row $w_{l,u}$ (u for unit) of the matrix W_l is a vector of the same dimensionality as z. Let $a_{l,u} = w_{l,u}z + b_{l,u}$. The output of $f_l(z)$ is a vector $[g_l(a_l,1), g_l(a_l,2), \ldots, g_l(a_l,size_l)]$, where g_l is some scalar function, and $size_l$ is the number of units in layer l. To make it more concrete, let's consider one architecture of neural networks called multilayer perceptron and often referred to as a vanilla neural network.

To give more details about deep learning, and to focus on the architectures we will use in Chapter 5, let's have a closer look at one particular configuration of neural networks called feed-forward neural networks (FFNN), and more specifically the architecture called a multilayer perceptron (MLP). As an illustration, let's consider an MLP with three layers. Our network takes a two-dimensional feature vector as input and outputs a number. This FFNN can be a regression or a classification model, depending on the activation function used in the third, output layer.

Our MLP is depicted in Figure 10. The neural network is represented graphically as a connected combination of units logically organised into one or more layers. Each unit is represented by either a circle or a rectangle. The inbound arrow represents an input of a unit and indicates where this input came from. The outbound arrow indicates the output of a unit.

The output of each unit is the result of the mathematical operation written inside the rectangle. Circle units don't do anything with the input; they just send their input directly to the output.

The following happens in each rectangle unit. Firstly, all inputs of the unit are joined together to form an input vector. Then the unit applies a linear transformation to the input vector, exactly like linear regression model does with its input feature vector. Finally, the unit applies an activation function g to the result of the linear transformation and obtains the output value, a real number. In

a vanilla FFNN, the output value of a unit of some layer becomes an input value of each of the units of the subsequent layer.

In Figure 10, the activation function g_l has one index: l, the index of the layer the unit belongs to. Usually, all units of a layer use the same activation function, but it's not a rule. Each layer can have a different number of units. Each unit has its parameters $w_{l,u}$ and $b_{l,u,}$ where u is the index of the unit, and l is the index of the layer. The vector y_{l-1} in each unit is defined as $[y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)}]$. The vector x in the first layer is defined as $[x^{(1)},\ldots, x^{(D)}]$.

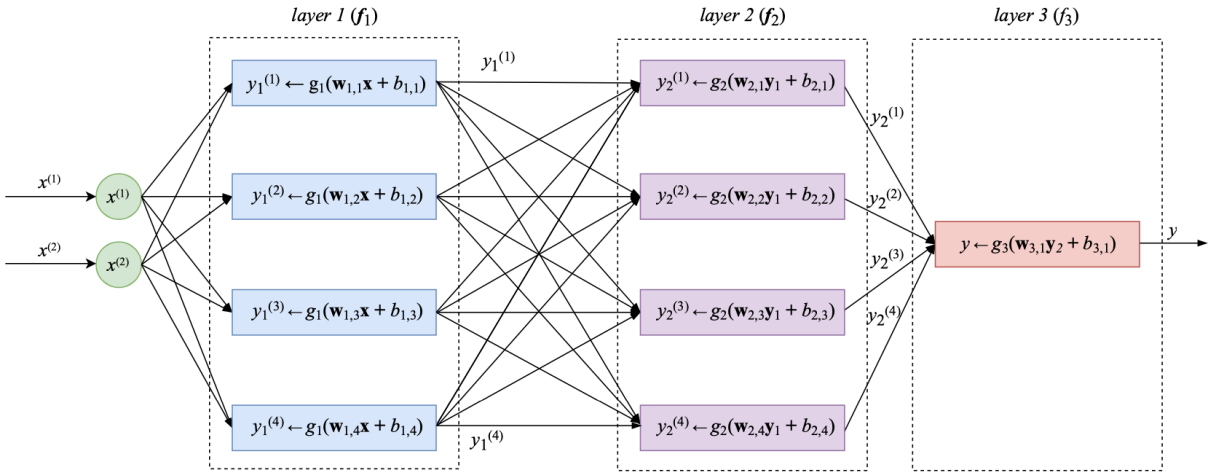


Fig. 10: A multilayer perceptron with two-dimensional input, two layers with four units and one output layer with one unit [54]

As you can see in Figure 10, in multilayer perceptrons all outputs of one layer are connected to each input of the succeeding layer. This architecture is called fully-connected. A neural network can contain fully-connected layers. Those are the layers whose units receive the outputs of each of the units of the previous layer.

Feed-Forward Neural Network Architecture

If we want to solve a regression or a classification problem discussed in previous chapters, the last (the rightmost) layer of a neural network usually contains only one unit. If the activation function g_{last} of the last unit is linear, then the neural network is a regression model. If the g_{last} is a logistic function, the neural network is a binary classification model.

The data analyst can choose any mathematical function as $g_{l,u}$, assuming it's differentiable. The latter property is essential for gradient descent used to find the values of the parameters $w_{l,u}$ and $b_{l,u}$ for all l and u. The primary purpose of having nonlinear components in the function f_{NN} is to allow the neural network to approximate nonlinear functions. Without nonlinearities, f_{NN} would be linear, no matter how many layers it has. The reason is that $W_l z + b_l$ is a linear function and a linear function of a linear function is also linear.

Popular choices of activation functions are the logistic function, already known to you, as well as TanH and ReLU. The former is the hyperbolic tangent function, similar to the logistic function but ranging from −1 to 1 (without reaching them). The latter is the rectified linear unit function, which equals to zero when its input z is negative and to z otherwise:

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$

$$relu(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise} \end{cases}$$

As we explained above, W_l in the expression $W_l z + b_l$, is a matrix, while b_l is a vector. That looks different from linear regression's wz+b. In matrix W_l, each row u corresponds to a vector of parameters $w_{l,u}$. The dimensionality of the vector $w_{l,u}$ equals to the number of units in the layer l−1. The operation $W_l z$ results in a vector al = [$w_{l,1}z$, $w_{l,2}z$, . . . , $w_{l,size_l}z$]. Then the sum $a_l + b_l$ gives a vector c_l. Finally, the function $g_l(c_l)$ produces the vector y_l = [y, y,…, y] as output.

Deep learning refers to training neural networks with more than two non-output layers. In the past, it became more difficult to train such networks as the number of layers grew. The two biggest challenges were referred to as the problems of exploding gradient and vanishing gradient as gradient descent was used to train the network parameters.

While the problem of exploding gradient was easier to deal with by applying simple techniques like gradient clipping and L1 or L2 regularization, the problem of vanishing gradient remained intractable for decades.

To update the values of the parameters in neural networks the algorithm called backpropagation is typically used. Backpropagation is an efficient algorithm for computing gradients on neural networks using the chain rule. During gradient descent, the neural network's parameters receive an update proportional to the partial derivative of the cost function with respect to the current parameter in each iteration of training. The problem is that in some cases, the gradient will be vanishingly small, effectively preventing some parameters from changing their value. In the worst case, this may completely stop the neural network from further training.

Traditional activation functions, such as the hyperbolic tangent function we mentioned above, have gradients in the range (0, 1), and backpropagation computes gradients by the chain rule. That has the effect of multiplying n of these small numbers to compute gradients of the earlier (leftmost) layers in an n-layer network, meaning that the gradient decreases exponentially with n. That results in the effect that the earlier layers train very slowly, if at all.

However, the modern implementations of neural network learning algorithms allow us to effectively train very deep neural networks (up to hundreds of layers). This is due to several improvements combined together, including ReLU, LSTM and other gated units, as well as techniques such as skip connections used in residual neural networks, as well as advanced modifications of the gradient descent algorithm.

Therefore, today, since the problems of vanishing and exploding gradient are mostly solved (or their effect diminished) to a great extent, the term "deep learning" refers to training neural networks using the modern algorithmic and mathematical toolkit independently of how deep the neural network is. In practice, many business problems can be solved with neural networks having 2-3 layers between the input and output layers. The layers that are neither input nor output are often called hidden layers. We will see in Chapter 5 ("Experiments"), that we implement different neural networks with a different number of (hidden) layers.

Convolutional Neural Network

As we make the network bigger, the number of parameters an MLP can grow exponentially fast. More specifically, as one adds one layer, there is an addition of $(size_l - 1 + 1) \cdot size_l$ parameters (our matrix W_l plus the vector b_l). That means that if you add another 1000-unit layer to an existing

neural network, then you add more than 1 million additional parameters to your model. Optimizing such big models is a very computationally intensive problem.

When our training examples are images, the input is very high-dimensional. If you want to learn to classify images using an MLP, the optimization problem is likely to become intractable.

A convolutional neural network (CNN) is a special kind of FFNN that significantly reduces the number of parameters in a deep neural network with many units without losing too much in the quality of the model. CNNs have found applications in image and text processing where they beat many previously established benchmarks.

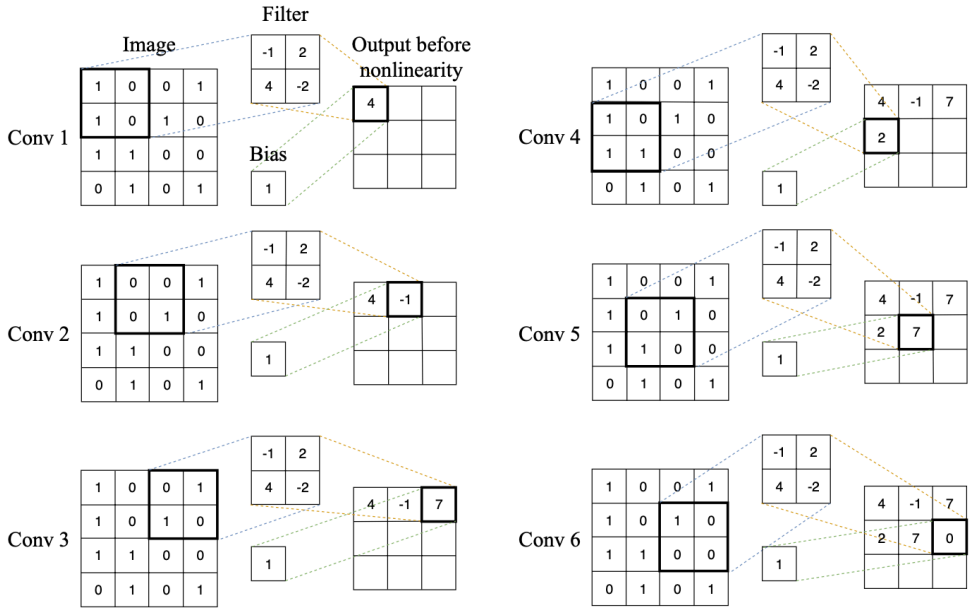


Fig. 11: A filter convolving across an image. [54]

One layer of a CNN consists of multiple convolution filters (each with its own bias parameter), just like one layer in a vanilla FFNN consists of multiple units. Each filter of the first (leftmost) layer slides -or convolves- across the input image, left to right, top to bottom, and convolution is computed at each iteration. An illustration of the process is given in Figure 11 steps of one filter convolving across an image are shown.

The filter matrix (one for each filter in each layer) and bias values are trainable parameters that are optimized using gradient descent with backpropagation.

A nonlinearity is applied to the sum of the convolution and the bias term. Typically, the ReLU activation function is used in all hidden layers. The activation function of the output layer depends on the task.

Since we can have $size_l$ filters in each layer l, the output of the convolution layer l would consist of $size_l$ matrices, one for each filter.

If the CNN has one convolution layer following another convolution layer, then the subsequent layer l + 1 treats the output of the preceding layer l as a collection of $size_l$ image matrices. Such a collection is called a volume. The size of that collection is called the volume's depth. Each filter

of layer l + 1 convolves the whole volume. The convolution of a patch of a volume is simply the sum of convolutions of the corresponding patches of individual matrices the volume consists of.

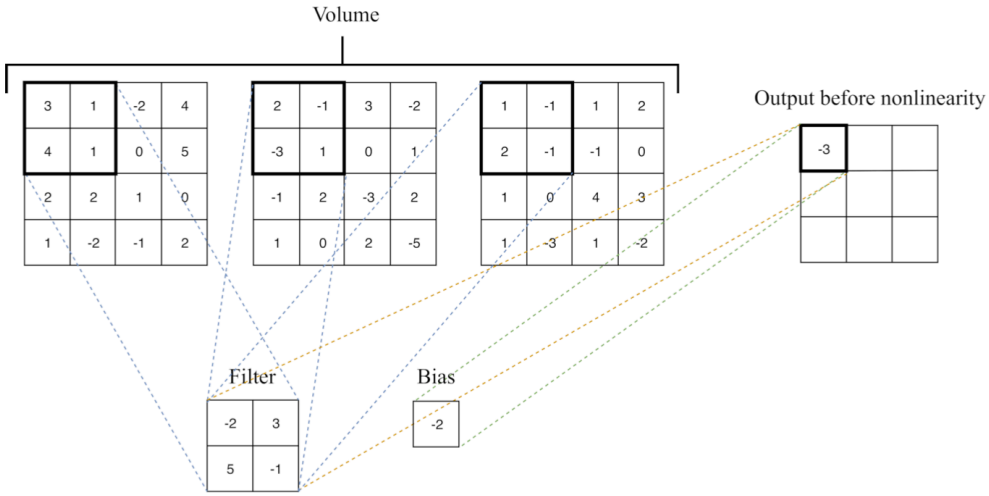


Fig. 12: Convolution of a volume consisting of three matrices. [54]

The value of the convolution, −3, was obtained as (−2·3+3·1+5·4+−1·1)+(−2·2+ 3·(−1)+5·(−3)+ −1·1)+(−2·1+3·(−1)+5·2+−1·(−1))+(−2). In computer vision, CNNs often get volumes as input, since an image is usually represented by three channels: R, G, and B, each channel being a monochrome picture.

Two important properties of convolution are stride and padding. Stride is the step size of the moving window. If the stride is 1, this means that the filter slides to the right and to the bottom by one cell at a time. With stride equals 2, the filter slides two cells at a time. Also, it is obvious that the output matrix is smaller when the stride is bigger.

Padding allows getting a larger output matrix; it's the width of the square of additional cells with which you surround the image (or volume) before you convolve it with the filter. The cells added by padding usually contain zeroes. When the padding is 0, no additional cells are added to the image. When the stride is 2 and padding is 1, a square of width 1 of additional cells are added to the image. We can see that the output matrix is bigger when padding is bigger.

Recurrent Neural Network

Recurrent neural networks (RNNs) are used to label, classify, or generate sequences. A sequence is a matrix, each row of which is a feature vector and the order of rows matters. To label a sequence is to predict a class for each feature vector in a sequence. To classify a sequence is to predict a class for the entire sequence. To generate a sequence is to output another sequence (of a possibly different length) somehow relevant to the input sequence.

RNNs are often used in text processing because sentences and texts are naturally sequences of either words/punctuation marks or sequences of characters. For the same reason, recurrent neural networks are also used in speech processing.

A recurrent neural network is not feed-forward: it contains loops. The idea is that each unit u of recurrent layer l has a real-valued state $h_{l,u}$. The state can be seen as the memory of the unit. In

RNN, each unit u in each layer l receives two inputs: a vector of states from the previous layer l − 1 and the vector of states from this same layer l from the previous time step.

To illustrate the idea, let's consider the first and the second recurrent layers of an RNN. The first (leftmost) layer receives a feature vector as input. The second layer receives the output of the first layer as input.

This situation is schematically depicted in Figure 13. As we said above, each training example is a matrix in which each row is a feature vector. So, let's illustrate this matrix as a sequence of vectors $X=[x^1,x^2,\ldots,x^t,x^{t+1},\ldots,x^{\text{lengthX}}]$,where lengthX is the length of the input sequence. If our input example X is a text sentence, then feature vector x_t for each t = 1,…, lengthX represents a word in the sentence at position t.

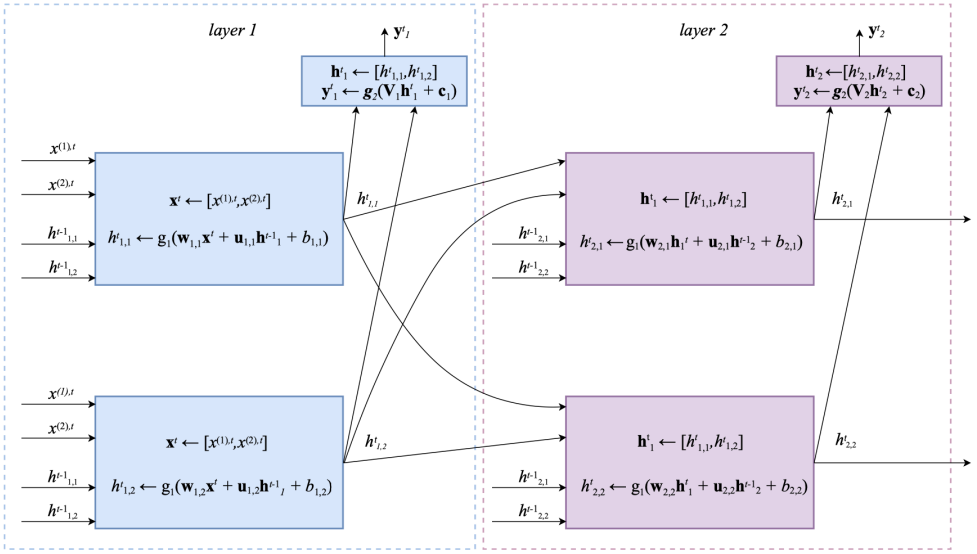


Fig. 13: The first two layers of an RNN. The input feature vector is two-dimensional; each layer has two units. [54]

As depicted in Figure 13, in an RNN, the feature vectors from an input example are read by the neural network sequentially in the order of the timesteps. The index t denotes a timestep. To update the state $h^t_{l,u}$ at each timestep t in each unit u of each layer l we first calculate a linear combination of the input feature vector with the state vector h^{t-1} of this same layer l,u from the previous timestep, t-1. The linear combination of two vectors is calculated using two parameter vectors $w_{l,u}$, $u_{l,u}$ and a parameter $b_{l,u}$. The value of $h^t_{l,u}$ is then obtained by applying activation function g_1 to the result of the linear combination. A typical choice for function g_1 is tanh. The output y^{lt} is typically a vector calculated for the whole layer l at once. To obtain ylt, we use activation function g_2 that takes a vector as input and returns a different vector of the same dimensionality. The function g_2 is applied to a linear combination of the state vector values $h_{lt,u}$ calculated using a parameter matrix V_l and a parameter vector $c_{l,u}$. In classification, a typical choice for g_2 is the softmax function:

$$\boldsymbol{\sigma}(\mathbf{z}) \stackrel{\text{def}}{=} [\sigma^{(1)},\ldots,\sigma^{(D)}],\ \text{where}\ \sigma^{(j)} \stackrel{\text{def}}{=} \frac{\exp\left(z^{(j)}\right)}{\sum_{k=1}^{D}\exp\left(z^{(k)}\right)}.$$

The softmax function is a generalization of the sigmoid function to multidimensional outputs. The dimensionality of V_l is chosen by the data analyst such that multiplication of matrix V_l by the vector h^{lt} results in a vector of the same dimensionality as that of the vector c. This choice depends on the dimensionality for the output label y in your training data. (Until now we only saw one-dimensional labels, but we will see in the future chapters that labels can be multidimensional as well.)

The values of $w_{l,u}$, $u_{l,u}$, $b_{l,u}$, $V_{l,u}$, and $c_{l,u}$ are computed from the training data using gradient descent with backpropagation. To train RNN models, a special version of backpropagation is used called backpropagation through time.

Both tanh and softmax suffer from the vanishing gradient problem. Even if our RNN has just one or two recurrent layers, because of the sequential nature of the input, backpropagation has to "unfold" the network over time. From the point of view of the gradient calculation, in practice this means that the longer is the input sequence, the deeper is the unfolded network.

Another problem RNNs have is that of handling long-term dependencies. As the length of the input sequence grows, the feature vectors from the beginning of the sequence tend to be "forgotten," because the state of each unit, which serves as network's memory, becomes significantly affected by the feature vectors read more recently. Therefore, in text or speech processing, the cause-effect link between distant words in a long sentence can be lost.

The most effective recurrent neural network models used in practice are gated RNNs. These include the long short-term memory (LSTM) networks and networks based on the gated recurrent unit (GRU).

The beauty of using gated units in RNNs is that such networks can store information in their units for future use, much like bits in a computer's memory. The difference with the real memory is that reading, writing, and erasure of information stored in each unit is controlled by activation functions that take values in the range (0, 1). The trained neural network can read the input sequence of feature vectors and decide at some early time step t to keep specific information about the feature vectors. That information about the earlier feature vectors can later be used by the model to process the feature vectors from near the end of the input sequence. For instance, if the input text starts with the word "she", a language processing RNN model could decide to store the information about the gender to interpret correctly the word "their" when it will be seen later in the text.

Units make decisions about what information to store, and when to allow reads, writes, and erasures. Those decisions are learned from data and implemented through the concept of gates. There are several architectures of gated units. A simple but effective one is called the minimal gated GRU and is composed of a memory cell, and a forget gate.

Let's look at the math of a GRU unit on an example of the first layer of the RNN (the one that takes the sequence of feature vectors as input). A minimal gated GRU unit u in layer l takes two inputs: the vector of the memory cell values from all units in the same layer from the previous timestep, h^{t-1}, and a feature vector x^t. It then uses these two vectors like l follows (all operations in the below sequence are executed in the unit one after another):

$$
\begin{aligned}
\tilde{h}^t_{l,u} &\leftarrow g_1(\mathbf{w}_{l,u}\mathbf{x}^t + \mathbf{u}_{l,u}\mathbf{h}^{t-1}_l + b_{l,u}),\\
\Gamma^t_{l,u} &\leftarrow g_2(\mathbf{m}_{l,u}\mathbf{x}^t + \mathbf{o}_{l,u}\mathbf{h}^{t-1} + a_{l,u}),\\
h^t_{l,u} &\leftarrow \Gamma^t_{l,u}\tilde{h}^t_l + (1-\Gamma^t_{l,u})h^{t-1}_l,\\
\mathbf{h}^t_l &\leftarrow [h^t_{l,1}, \ldots, h^t_{l,size_l}]\\
\mathbf{y}^t_l &\leftarrow \boldsymbol{g}_3(\mathbf{V}_l\mathbf{h}^t_l + \mathbf{c}_{l,u}),
\end{aligned}
$$

where g_1 is the tanh activation function, g_2 is called the gate function and is implemented as the sigmoid function taking values in the range (0, 1). If the gate $\Gamma_{l,u}$ is close to 0, then the memory cell keeps its value from the previous time step, h^{t-1}. On the other hand, if the gate $\Gamma_{l,u}$ is close to 1, the value of the memory cell is overwritten by a new value $h^t_{l,u}$. Just like in standard RNNs, g_3 is usually softmax.

A gated unit takes an input and stores it for some time. This is equivalent to applying the identity function ($f(x) = x$) to the input. Because the derivative of the identity function is constant, when a network with gated units is trained with backpropagation through time, the gradient does not vanish.

Other important extensions to RNNs include bi-directional RNNs, RNNs with attention and sequence-to-sequence RNN models. The latter, in particular, are frequently used to build neural machine translation models and other models for text to text transformations. A generalization of an RNN is a recursive neural network.

We will see implementations of RNNs in Chapter 5 (“Experiments”), as we implemented an LSTM neural architecture to achieve high accuracy predictions in sales forecasting for FMCGs.

3. Previous Work

3.1 Machine Learning in Sales Forecasting

From a historical perspective, exponential smoothing methods and decomposition methods were the first forecasting approaches to be developed back in the mid-1950s. During the 1960s, as computer power became more available and cheaper, more sophisticated forecasting methods appeared.

Box-Jenkins [14] methodology gave rise to the ARIMA models Later on, during the 1970s and 1980s, even more sophisticated forecasting approaches were developed including econometric methods and Bayesian methods.

Time series forecasting models have been widely applied in sales forecasting, such as exponential smoothing models [2], ARIMA models [2, 6, 7], expert systems [1, 3, 8], fuzzy systems [1, 3, 8, 9], and NN models [23, 24, 25, 49, 53, 54, 55].

So, Sales Forecasting is a well-researched subject, especially in the context of time series. Although there is a vast body of literature and technological advancement on the topic of forecasting (Fildes, Goodwin and Lawrence, 2006; McCarthy, Davis Golicic and Mentzer, 2006; Armstrong, Green and Graefe, 2015), there is a weak evidence on successful business implementations. Decision makers remain skeptical about recommendations offered by forecasting support systems (FSS) and rather rely on applying their own mental models (Goodwin, Fildes, Lawrence and Stephens, 2011) the resulting forecasts are often ‘sub-optimal’ because many judgmental adjustments are made when they are not required.

The research of forecasts now focuses more on how to take the knowledge gained about forecasting and implement that in companies to improve forecasts (Moon et al., 2003). So as attention of the importance about forecasting increases, focus is moved from having focused mainly on different techniques, to focusing on the forecasting process (Bunn & Taylor, 2001). Previous studies have focused on an adoption between different techniques, both quantitative and qualitative, to achieve higher accuracy. Moon et al. (2003) showed that technique adoption will not guarantee a good accuracy and therefore companies should also focus on how the forecasting process is managed and organised. The main objective of a forecasting process is to maintain a good information flow within the company and also organise the work around the creation of the sales forecast (Danese & Kalchschmidt, 2011). According to Danese and Kalchschmidt (2011) a forecasting process can be divided into four different steps information-gathering and tools (what information should be collected and how it should be collected), organisational (who should be in charge of forecasting and what roles should be designed), interfunctional and intercompany (using different sources of information within the company or supply network and a joint elaboration of forecasts), and measurement of accuracy (using the proper metric and defining proper incentive mechanisms). These steps should always be under continuous improvement to ensure a high sales forecast accuracy. A company that has a well-designed forecasting process provides a better opportunity to understand market dynamics and consumer behaviour, reduce uncertainty of future events, and provide the company’s departments with useful analyses and information (Danese & Kalchschmidt, 2011). If a company can develop a well-designed forecasting process and use an adoption between different techniques, the company will facilitate the work to achieving higher sales forecast accuracy (Danese & Kalchschmidt, 2011).
However, regardless the forecasting process, the need for models with higher accuracy for immediate improvement in prediction accuracy is stronger than ever. In [4], a method for small samples based on Support Vector Machine is proposed. It can effectively solve the problems of small sample, nonlinear and high-dimensional pattern recognition, and can be applied to other

machine learning problems such as function fitting. But regrettably the applicability of this method is reduced because most sales forecasting is based on a large number of sample data. Meanwhile, there is a limitation in the above statistical methods: the need to transform qualitative data into quantitative data. On this foundation, paper [7] puts forward the method combining sentiment analysis and Bass model which is aimed at processing online review data for sales forecasting. Moreover, the shallow neural network method has been applied to the sales forecasting in literature and promising results have been obtained. However, the prediction accuracy of the above method is not satisfactory when the characteristics of the prediction problem are fuzzy influence factors, massive samples with complex structure and a large time interval. It is an urgent challenge that older models of shallow learning (or simply machine learning) cannot clearly depict the relationship between variables of sales prediction problem. Based on the development of deep learning, R. G. Hiranya Pemathilake et al. provided a hybrid model with integrated moving average and recurrent neural network.

To sum up, many publications over the years specifically study Machine Learning in Sales Forecasting. However, there is no publication (at least to my knowledge) that focuses specifically on FMCGs. The closest to this are the publications focusing on Sales Forecasting for fashion goods. But fashion goods are not a good representative of the FMCG industries as, as it was established in the Theoretical Background (Chapter 2), fashion goods are quite unique in terms of seasonability and market trends. To give a benchmark of publications on new Machine Learning solutions for Sales Forecasting, we provide the follow 5 publications:

- Shouwen Ji, Xiaojing Wang, Wenpeng Zhao, Dong Guo, "An Application of a Three-Stage XGBoost-Based Model to Sales Forecasting of a Cross-Border E-Commerce Enterprise", *Mathematical Problems in Engineering*, vol. 2019, Article ID 8503252, 15 pages, 2019. https://doi.org/10.1155/2019/8503252
- Yonghe Zhao et al, "Optimization of a Comprehensive Sequence Forecasting Framework Based on DAE-LSTM Algorithm", J. Phys., Conf. Ser. 1746 012087, 2021. https://iopscience.iop.org/article/10.1088/1742-6596/1746/1/012087
- Nikolas Ulrich Moroff, Ersin Kurt, Josef Kamphues, "Machine Learning and Statistics: A Study for assessing innovative Demand Forecasting Models", Procedia Computer Science, Volume 180, 2021, Pages 40-49, ISSN 1877-0509. https://doi.org/10.1016/j.procs.2021.01.127
- McCarthy, T. M., Davis, D. F., Golicic, S. L. & Mentzer, J. T., "The evolution of sales forecasting manage- ment: a 20-year longitudinal study of forecasting practices", *Journal of Forecasting*, 2006, 25 (5), 303-324. http://dx.doi.org/10.1002/for.989
- Yun Dai, Jinghao Huang, "A Sales Prediction Method Based on LSTM with Hyper-Parameter Search", 2021, *J. Phys.: Conf. Ser.* 1756 012015. https://doi:10.1088/1742-6596/1756/1/012015

So, we hope that this Thesis feels that gap, focusing specifically on Machine Learning Sales Forecasting for FMCGs. As we pointed out at this Thesis' scope, our datasets on Detergents and Cleaners are excellent representatives of the FMCGs' class characteristics. More about our datasets' characteristics can be found at Chapter 5, "Experiments", where we show why our data and data analysis are an adequate representative of the FMCGs industries.

3.2 Ensemble Learning and Meta-Learning

For the purpose of this Thesis, many Ensemble Learning papers have been studied. The main publications that helped us develop a good idea about Ensemble Forecasting are mainly from Weather Forecasting and Price Forecasting. To the best of our knowledge the worth-reading publications in Ensemble Forecasting for Sales Forecasting are amazingly little. The following

publications helped us develop good idea about Ensemble Forecasting and we should briefly look into them in this section to get a good idea about the state of research in Ensemble Forecasting and its historical use in industry applications.

The following publications will be briefly discussed here:

- M. Leutbecher, T.N. Palmer, "Ensemble forecasting", Journal of Computational Physics, Volume 227, Issue 7, 2008, Pages 3515-3539, ISSN 0021-9991. https://doi.org/10.1016/j.jcp.2007.02.014
- Jujie Wang, Xin Sun, Qian Cheng, Quan Cui, "An innovative random forest-based nonlinear ensemble paradigm of improved feature extraction and deep learning for carbon price forecasting", Science of The Total Environment, Volume 762, 2021, 143099, ISSN 0048-9697. https://doi.org/10.1016/j.scitotenv.2020.143099
- Ling Tang, Lean Yu, Shuai Wang, Jianping Li, Shouyang Wang, "A novel hybrid ensemble learning paradigm for nuclear energy consumption forecasting", Applied Energy, Volume 93, 2012, Pages 432-443, ISSN 0306-2619. https://doi.org/10.1016/j.apenergy.2011.12.030
- Guoqiang Zhang, Jifeng Guo, "A novel ensemble method for residential electricity demand forecasting based on a novel sample simulation strategy", Energy, Volume 207, 2020, 118265, ISSN 0360-5442. https://doi.org/10.1016/j.energy.2020.118265
- Ramon Gomes da Silva, Matheus Henrique Dal Molin Ribeiro, Sinvaldo Rodrigues Moreno, Viviana Cocco Mariani, Leandro dos Santos Coelho, "A novel decomposition-ensemble learning framework for multi-step ahead wind energy forecasting", Energy, Volume 216, 2021, 119174, ISSN 0360-5442. https://doi.org/10.1016/j.energy.2020.119174
- Sonia Kahiomba Kiangala, Zenghui Wang, "An effective adaptive customization framework for small manufacturing plants using extreme gradient boosting-XGBoost and random forest ensemble learning algorithms in an Industry 4.0 environment", Machine Learning with Applications, Volume 4, 2021, 100024, ISSN 2666-8270. https://doi.org/10.1016/j.mlwa.2021.100024

Ensemble Learning refers to the procedures employed to train multiple learning machines and combine their outputs, treating them as a "committee" of decision makers. The principle is that the committee decision, with individual predictions combined appropriately, should have better overall accuracy, on average, than any individual committee member. Numerous empirical and theoretical studies have demonstrated that ensemble models very often attain higher accuracy than single models.

The members of the ensemble might be predicting real-valued numbers, class labels, posterior probabilities, rankings, clusterings, or any other quantity. Therefore, their decisions can be combined by many methods, including averaging, voting, and probabilistic methods. The majority of ensemble learning methods are generic, applicable across broad classes of model types and learning tasks.

An ensemble consists of a set of models and a method to combine them. So, we have a set of models, generated by any of the learning algorithms in this Thesis; we explore popular methods of combining their outputs, for classification and regression problems. Following this, we review some of the most popular ensemble algorithms, for learning a set of models given the knowledge that they will be combined, including extensive pointers for further reading. Finally, we take a theoretical perspective, and review the concept of ensemble diversity, the fundamental property which governs how well an ensemble can perform.

The ensemble learning method found its early origin in a typical human principle for best decision-making to seek several experts' opinions in a specific area before concluding (Re & Valentini, 2012). Ensemble learning is a process that combines several base predictors such as

individual learning algorithms to produce improved results in terms of accuracy or stability (Dietterich, 2000; Kuncheva, 2004). In the area of machine learning (ML), various researchers have given the ensemble learning technique the merit of significantly improving the general performance of ML models (Dietterich, 2000; Hansen & Salamon, 1990; Kuncheva, 2004; Zhou, Wu, & Tang, 2002) and producing some of the best learning system prototypes (Fernández-Delgado, Cernadas, Barro, & Amorim, 2014; Wu et al., 2008). The worth of ensemble learning methods has also been experienced in real-life applications producing outstanding results (Ahneman, Estrada, Lin, Dreher & Doyle, 2018; Lee, Jeong, Lee, & Jeong, 2019). In order to improve the performance of ML models, the ensemble learning technique exposes the individual learning algorithms (base predictors) to learn from a different perspective of the dataset either by heterogeneous ensemble learning (utilising different learning algorithms) or by homogeneous ensemble learning (using one single-learning algorithm that learns on random subsets coming from the original training set) (Pham, Kim, Park, & Choi, 2021). It is worth mentioning that base predictors' diversity contributes to the result accuracy of the ensemble learning model (Kuncheva & Whitaker, 2003; Liu, Zhang, Luo, & Cai, 2017). Kuncheva (2004) suggested four levels to categorise the different ensemble learning methods: classifier level, data level, feature level, and combination level.

(a) Classifier level: Ensemble learning methods in this category utilise homogeneous or heterogeneous classifiers created by applying a certain randomness level into the same classifiers. The final result of the ensemble learning method is a combination of classifiers' results grouped by a method that reduces each classifier's level of bias. The well-known ''boosting ensemble learning algorithm'' falls under this group. Fig. 14 presents an ensemble learning algorithm introduced to create a powerful classifier model, with accurate classification results, based on several weaker classifiers. The boosting classifier model is incrementally constructed by correcting errors of previous weak models.

(b) *Data level:* In this ensemble learning category, resampling methods such as leave one out, random selection(with or without replacement) are used to create training sample subsets from which the base classifiers are trained. Various voting methods are used to combine the base classifiers results. A prevalent ensemble learning method in the data level is the bagging algorithm. Breiman (1996) introduced ''the bagging'' algorithm to improve model classification results. He achieved this by combining classification results from several independently trained classifiers. The independent classifiers originate from the random training set (from the original dataset). Bagging is a form of ''bootstrap aggregating''. The random forest algorithm (RF) used in this research is a form of a bagging algorithm. It groups several decision trees (from random training sets) to obtain better classification accuracies (Deng et al., 2020).

In the most common machine learning setting, one predicts the value of a single target attribute, categorical or numeric. A natural generalisation of this setting is to predict multiple target attributes simultaneously. The task comes in two slightly different flavours. In multi-target prediction (Blockeel et al., 1998), all target attributes are (equally) important and predicted simultaneously with a single model. Multi-task learning (Caruana, 1997), on the other hand, originally focused on a single target attribute and used the rest for assistance only. Nowadays, however, multi-task models typically predict each target attribute individually but with at least partially distinct models.

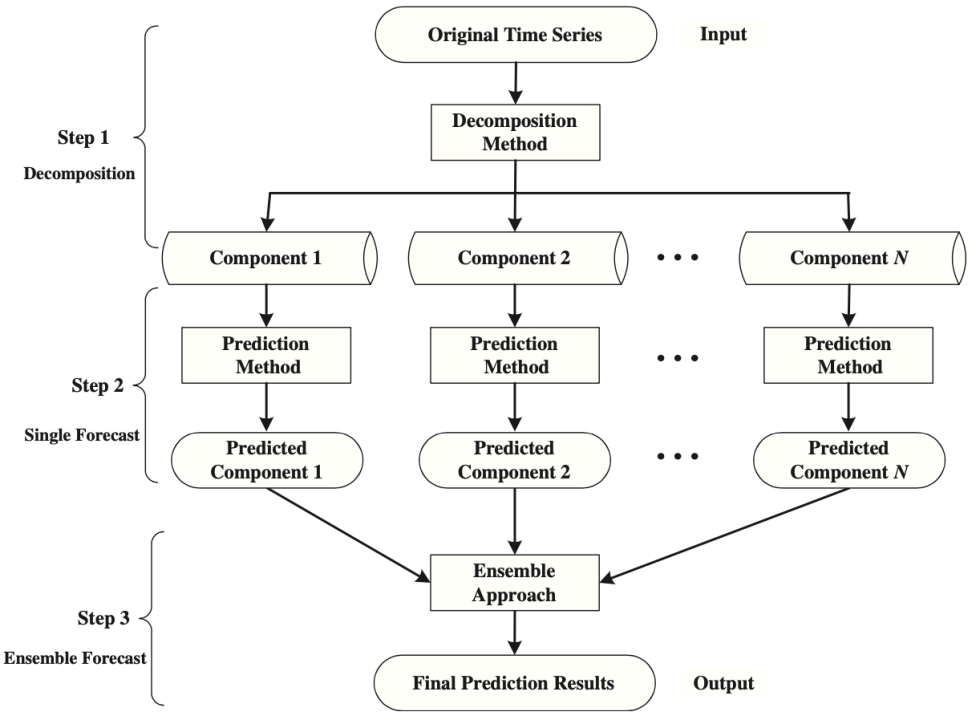


Fig. 14. General framework of the hybrid ensemble learning paradigm. [37]

There are many different methods for model combination, such as the linear combiner, the product combiner, and the voting combiner are by far the most commonly used in practice. Though a combiner could be specifically chosen to optimise performance in a particular application, these three rules have shown consistently good behaviour across many problems, and are simple enough that they are amenable to theoretical analysis.

The linear combiner is used for models that output real-valued numbers, so is applicable for regression ensembles, or for classification ensembles producing class probability estimates. Here we only show notation for the latter case. We have a model $f_t(y|x)$, an estimate of the probability of class y given input x. For a set of these, t = {1,…,T }, the ensemble probability estimate is,

$$\bar{f}(y|\mathbf{x}) = \sum_{t=1}^{T} w_t f_t(y|\mathbf{x}).$$

If the weights w_t =1/T, ∀t, this is a simple uniform averaging of the probability estimates. The notation clearly allows for the possibility of a non-uniformly weighted average. If the classifiers have different accuracies on the data, a non-uniform combination could in theory give a lower error than a uniform combination. However, in practice, the difficulty of estimating the w parameters without overfitting, and the relatively small gain that is available have meant that in practice the uniformly weighted average is by far the most commonly used. A notable exception, which will be analysed in detail in Chapter 5 as it is used in our Experiments, is the Mixture of Experts. In MoE, weights are non-uniform, but are learnt and dependent on the input value x. An alternative combiner is the product rule:

$$\bar{f}(y|\mathbf{x}) = \frac{1}{Z}\prod_{t=1}^{T} f_t(y|\mathbf{x})^{w_t}.$$

Where Z is a normalisation factor to ensure f is a valid distribution. Note that Z is not required to make a valid decision, as the order of posterior estimates will remain unchanged before/after normalisation. Under the assumption that the class-conditional probability estimates are independent, this is the theoretically optimal combination strategy. However, this assumption is highly unlikely to hold in practice, and again the weights w are difficult to reliably determine. Interestingly, the linear and product combiners are in fact special cases of the generalised mean allowing for a continuum of possible combining strategies.

The linear and product combiners are applicable when our models output real-valued numbers. When the models instead output class labels, a majority (or plurality) vote can be used. Here, each classifier votes for a particular class, and the class with the most votes is chosen as the ensemble output. For a two-class problem the models produce labels, $h_t(x) \in \{-1,+1\}$. In this case the ensemble output for the voting combiner can be written:

$$H(\mathbf{x}) = \operatorname{sign}\Big(\sum_{t=1}^{T} w_t h_t(\mathbf{x})\Big).$$

The weights w can be uniform for a simple majority vote, or non-uniform for a weighted vote.
We have discussed only a small fraction of the possible combiner rules. Numerous other rules exist, including methods for combining rankings of classes, and unsupervised methods to combine clustering results. For details of the wider literature, see references [37] or [38].

A typical example coming from the environmental sciences is the task of predicting species distribution or community structure (Demsar et al., 2006), where we are interested in predicting the abundances of a set of different species living in the same environment. These species represent the target attributes, which might, but need not be related. Examples from other areas, ranging from natural language processing to bioinformatics and medicine are also plentiful (Jeong and Lee, 2009; Liu et al., 2010; Bickel et al., 2008).

With multiple targets, a typical solution is to create a collection of single-target models. Nevertheless, especially if we are interested in the interpretability of the model, the collection of single-target models is more complex and harder to interpret than a single model that jointly predicts all target attributes (Blockeel, 1998; Suzuki et al., 2001; Zenko and Dzeroski, 2008). Furthermore, learning several tasks together may increase the predictive performance for the individual tasks due to inductive transfer, where the knowledge from one task is transferred to the other tasks (Piccart et al., 2008; Kocev et al., 2007; Suzuki et al., 2001). An additional benefit of the multi-target models is that they are less likely to overfit the data than the corresponding collections of single-target models (Blockeel, 1998; Caruana, 1997).

Rule sets, together with decision trees, are one of the most expressive and human readable model representations. They are frequently used when an interpretable model is desired. The majority of rule learning methods are based on the sequential covering algorithm (Michalski, 1969), originally designed for learning ordered rule lists for binary classification domains. This is also the case with the existing methods for learning multi-target rules (Zenko and Dzeroski, 2008). Unfortunately, on both single-target and multi-target regression problems, the accuracy of rule sets that are learned by the sequential covering approach is considerably lower than that of other regression methods, like for example, regression trees.

An alternative rule learning method that performs well also on (single-target) regression problems is the approach of rule ensembles (Friedman and Popescu, 2005, 2008; Dembczynski et al., 2008). It creates a collection of rules and uses an optimisation procedure with the purpose of finding a small (and therefore interpretable) subset of rules. Optionally, rules can be combined with simple linear functions of descriptive attributes.

In the multi-target prediction task, we are given a set of training examples E of the form (x,y), where $x = (x_1,x_2,...,x_K)$ is a vector of K descriptive attributes and $y = (y_1,y_2,...,y_T)$ is a vector of T target attributes. Our task is to learn a model that, given a new unlabelled instance x, can predict the values of all target attributes y simultaneously. Several standard learning methods such as neural networks, decision trees, model trees, classification rules and random forests have been extended towards multi-target prediction (Caruana, 1997; Blockeel et al., 1998; Appice and Dzeroski, 2007; Suzuki et al., 2001; Zenko and Dzeroski, 2008; Kocev et al., 2007).

An approach related to multi-target learning is multi-task learning (Caruana, 1997; Argyriou et al., 2008; Chapelle et al., 2010; Jalali et al., 2011; Rakotomamonjy et al., 2011; Parameswaran and Weinberger, 2011). In multi-task learning, the aim is to solve multiple single-target learning tasks $(x, y)^T_{t=1}$ with different training sets Et (and in general with different descriptive attributes) at the same time. Multi-task learning should be able to benefit from relationships between tasks, just like multi-target prediction. The result of multi-task training is a distinct trained model $f^t(x_t)$ for each of the T tasks.

While it is true that multi-target and multi-task learning have some common background, there are also some clear differences between them. The most obvious one is the number of trained models: a separate model for each of the tasks versus a single model trained for the entire problem. Multi-target learning aims to predict the target features and explicitly describe their relationship with the descriptive features. Moreover, it implicitly describes the relationships among the target features. The multi-task model, on the other hand, does not specifically aim to describe the relation- ships between the target features.

Multi-target learning implicitly captures the dependencies among the targets and represents them in the single model generated. By going through this model, we can determine the effect of the descriptive features on all the targets, and analyse the relationships, either linear or nonlinear, between targets (or groups of targets). In case the targets are related, we can obtain information about these relationships.

To place our final Ensemble model into a broader context, we further analyse the cases of Stacking Classifier/Regressor and Mixture of Experts in Chapter 4 (“Method and Models”). This Thesis final Stacking Classifier/Regressor will be discussed in detail in Chapter 5 (“Experiments”). However, we give here information about the latest multi-task linear regression algorithm to put all the algorithms in perspective.

As our final Meta-learning models are based on combining predictions of the best models through weighted voting (Stacking Classifier/Regressors), we give some extra information about rule ensembles. A rule ensemble is understood to be a set of unordered rules whose predictions are combined through weighted voting, which is the approach introduced by the RULEFIT (Friedman and Popescu, 2005, 2008) and REGENDER methods (Dembczynski et al., 2008).

Most of the multi-task algorithms are originally designed for classification purposes (Chapelle et al., 2010; Rakotomamonjy et al., 2011; Parameswaran and Weinberger, 2011), but the method by Jalali et al. (2011) is readily suitable for our regression tasks. Jalali et al. (2011) tried to find a compromise between selecting important weights for separate tasks and for all tasks together.

That is, they are searching for both shared features and features important for each task separately. The authors do this by using both separate element-wise L_1 and block L_1/L_q regularisation and alternate between the two during optimisation. Here L_1/L_q is matrix

regularisation, with q > 1 in the latter case. Because of mixing up the two "clean" regularisation terms, Jalali et al. (2011) call their method "dirty", therefore we refer to their algorithm as DIRTY.

Since our method learns regression rules, it is closely related to rule learning (Flach and Lavrac, 2003). A method for learning multi-target rules has been recently developed (Zenko and Dzeroski, 2008). It employs the standard covering approach (Michalski, 1969) and can learn ordered or unordered rule sets for classification and regression domains. Its accuracy on classification domains is comparable to other classification methods, such as (multi-target) decision trees. However, on regression domains, the approach performs significantly worse than the alternatives (Zenko, 2007).

An alternative approach to rule learning is called rule ensembles (Friedman and Popescu, 2005, 2008; Dembczyn ski et al., 2008). Strictly speaking, any set of (unordered) rules can be called a rule ensemble, as for example, in Indurkhya and Weiss (2001).

The RULEFIT algorithm starts by generating a set of decision trees in much the same way as ensembles are generated in methods like bagging (Breiman, 1996) and random forests (Breiman, 2001). Because such large ensembles are hard or even impossible to interpret, all the trees are transcribed into a collection of rules, and an optimisation procedure is used to select a small subset of the rules and to determine their weights. As a result, we get a relatively small set of weighted rules combined in a linear fashion. In addition to rules, we can also use descriptive attributes in the linear combination if we add them to the initial set of rules, and likewise determine their weights in the optimisation step. The final prediction for a given example is obtained by a weighted voting of all linear terms and those rules that apply (cover the example). The resulting model can thus be written as:

$$\hat{y} = f(\boldsymbol{x}) = w_0 + \sum_{i=1}^{M} w_i r_i(\boldsymbol{x}) + \underbrace{\sum_{j=1}^{K} w_{(M+j)} x_j}_{\text{optional}},$$

where w_0 is the baseline prediction, the first sum is the correction value obtained from the M rules, and the second sum is the correction value obtained from the (optional) K linear terms. The rules r_i are functions, which have a value of 1 for all examples that they cover, and 0 otherwise. During the learning phase, all the weights wi are optimised by a gradient directed optimisation algorithm. The linear terms part of the model is global, that is, it covers the entire example space.

Note that this is different from model trees (Quinlan, 1992; Karalic, 1992; Wang and Witten, 1997), where we may also have local linear models in tree leaves, where each such model only applies to the specific examples covered by the leaf.

Last but not least, let's point out once more that our final Stacking classifier will be discussed in detail in Chapter 5 ("Experiments"). Our method and our final model is based on combining predictions of the 2-best and 3-best models through weighted voting, and the showcase of previous work on rule ensembles we hope that helped to provide both the necessary theoretical background and research perspective. Also, we implement a Mixture of Experts for our Deep Learning models to showcase its power, but also its weaknesses.

All in all, the analysis of the Previous Work, both about Machine Learning in Sales Forecasting and Ensemble Learning prepared us to dive into the most important sections of this Thesis. For more information and details about relevant work, we provide the full list of References (Bibliography) at the end of this Thesis.

4. Method & models

4.1 Method

Unlike models that predict option prices by comparing their results with the Black & Scholes option pricing formula, in consumer products the underlying partial differential equation that explains the pricing-discount-sales relationship is unknown. For this reason, machine learning can be particularly useful in this area, as by considering the underlying partial differential equations a black-box we lose nearly zero information and model accuracy, in comparison with the traditional mathematical models who are based on supply and demand equations. As we have seen from Previous Work and we will see in our Experiments too, Machine Learning can produce extremely good, high-precision models, which are particularly useful in both relevant research and industry applications.

Our goal is the comparative analysis of models and methods to solve the problem of High Accuracy Sales Forecasting Predictions, but also the construction of a unified machine learning model (meta-learner) with very good accuracy to do sales forecasting for fast-moving consumer goods (FMCGs). To achieve this, we tested some of the most modern and efficient models in sales forecasting and finally kept the best models (those with the highest accuracy). Then, by using Ensemble learning techniques (meta-learning) we produced aggregated forecasting models. We showed, as expected, that with negligible additional cost, we can have combined results better than any other's model's. Also, we compared Ensemble Learning techniques for different models to test which gives the best possible results and, thus, have a final model and methodology to recommend to businesses in the field.

As FMCGs are products with massive production, campaigns and sales, there is a necessity for extremely fast forecasting with the maximum possible accuracy. The final Sales Forecasting process that is given as an advice to businesses in the FMCG industries takes into account both time and accuracy, and so it provides our final meta-learner for improvements in accuracy, but at the same time, with no penalty at execution time and overall performance. So, regardless of theoretical improvements in ML algorithms, business-wise, the method of benchmarking some ML models and Ensembling appropriately the best models is the optimal from a cost-benefit perspective. Improvements in accuracy may be achieved by using Deep Learning models but with significant additional cost. However, from our Experiments (see Chapter 5, "Experiments") this is not the case, as Deep Learning models overfit exceptionally fast and, thus, score lower, than much simpler models. Therefore, our method is very useful for businesses, who need to know which models achieve high accuracy with low cost, and so they can use our final results and conclusions to increase the accuracy of their sales forecasting models at nearly zero cost.

Overall, the benefits of our final Meta-learning solution has many advantages, such as:

- It leads to extremely efficient Sales Forecasting, as it efficiently combines some of the best single ML models
- It has very high prediction accuracy, but with relatively low training time, making it an excellent choice for real-world Sales Forecasting solutions implemented by companies in the fast-moving consumer goods industries.
- It can be used both by businesses who haven't yet invested in their Sales Forecasting process (e.g. new businesses, startups etc.) and by businesses that have a Sales Forecasting process and want to improve the accuracy of their models but with worth-investing additional cost.

To complete our Experiments and benchmark different Ensembling techniques we tried 3 different Meta-learning techniques, 2 Stacking Classifiers and a Mixture of Experts (MoE) which will be explained theoretically in this section, but the results of which will be shown in Chapter 5 ("Experiments").

4.2 Regression Models

As we discussed in Chapter 2, the regression problem is solved by a regression learning algorithm that takes a collection of labeled examples as inputs and produces a model that can receive an unlabelled example and output a target. In this section, we present the 5 simple Regression ML models we used in our experiments for the purpose of this Thesis.

4.2.1 Huber Regression

Huber Regression is based on the generalized version of the Huber loss function which can be incorporated with Generalized Linear Models (GLM) and is well-suited for heteroscedastic regression problems. We will start with a brief recap of the Huber loss function and the basics of Generalized Linear Models (GLM) and then we will show how to optimise this loss function with gradient boosted trees and compare the results to classical loss functions on an artificial data set.

The Huber loss is a robust loss function for regression problems defined as, (1):

$$\mathcal{L}(y, \hat{y}) = \begin{cases} (y - \hat{y})^2 & \dots \quad |y - \hat{y}| \leq \alpha \\ |y - \hat{y}| & \dots \quad |y - \hat{y}| > \alpha \end{cases}$$

where y is the target variable, $\hat{y}$ are the corresponding predictions and $\alpha \in \mathbb{R}^+$ is a hyper-parameter. It is tempting to look at this loss as the log-likelihood function of an underlying heavy tailed error distribution. Indeed, for absolute errors smaller than α the corresponding distribution resembles the normal distribution, outside this region it coincides with the more heavy-tailed Laplace distribution. This is precisely the reason why this loss is robust against outliers.

In linear regression one often assumes that the error term in the linear relationship between the dependent variable Y and some feature vector X is normally distributed with mean zero and constant variance σ^2, i.e. $Y|X \sim X^\top \beta + \varepsilon$ with $\varepsilon \in \mathcal{N}(0,\sigma^2)$ and β being a set of variational parameters. One is interested in finding the best estimate β_{hat} that minimises a quadratic cost function (corresponding to the log-likelihood of the distribution of ε). The estimate $\hat{y}$ for a given X is then simply $\hat{y}(X)=\mathrm{E}[Y|X]$. Note that (in a maximum-likelihood interpretation) Huber regression replaces the normal distribution with a more heavy tailed distribution but still assumes a constant variance.

The GLM approach on the other hand relaxes the assumptions of linear regression in the following way:

A. Non-normality of the random component: $Y|X \sim \text{some distribution from the exponential family}$

B. Non-linearity introduced by a link function g: $g(E[Y|X]) = X^\top \beta$

The exponential family contains a variety of distributions and in particular some where the variance is a function of the mean like the Poisson or Gamma distribution. This feature comes in handy especially for heteroscedastic problems where the assumption of a constant variance of the error term does not hold anymore, as is, for example, often the case for a target variable whose range spans several orders of magnitude. The link function additionally enhances the model complexity by contributing non-linear effects.

Note also that the link function is never applied to y. This stands in contrast with the common practice of fitting a model to the transformed target variable, which often leads to underestimating the mean once the predictions are back-transformed. This can be seen from Jensen's inequality which states that $\mathrm{E}[g(Y)|X] \leq g(\mathrm{E}[Y|X])$ for any concave function g, as it is the case for the logarithm or the Box-Cox function. It is further possible to derive a closed-form expression for the GLM likelihood function which results in a broad class of loss functions. However, it seems that there is no continuous distribution with negative support and non-constant variance within the exponential family. Generalised Huber Loss Function combines the idea of a link function with the Huber loss, while still having a non-constant variance.

Generalised Huber Loss Function

For any invertible function $g: \mathbb{R} \mapsto \mathbb{R}$ we define the Generalized Huber Loss (GHL) function as (2):

$$\mathcal{L}(y, \hat{y}) = \begin{cases} (y - \hat{y})^2 & \dots \quad |y - \hat{y}| \leq \alpha \\ |y - \hat{y}| & \dots \quad |y - \hat{y}| > \alpha \end{cases}$$

with $\alpha \in \mathbb{R}^+$, y the target variable and $\hat{y}$ the continuous output of some predictive model. The most important observation here is that the case distinction is taken on the "link scale" defined by g(y), whereas the range is on the original scale. This loss function can not be transformed to a single variable problem.

Let us now discuss what would happen if we took $g(y)$ instead of $g^{-1}(\hat{y})$ on the right hand side of the equation. This would simply correspond to first transforming the target variable and thus estimating $\mathrm{E}[g(Y)|X]$. However, since $\mathrm{E}[g(Y)|X] \leq g(\mathrm{E}[Y|X])$ for any concave function $g(y)$ we would end up underestimating the mean.
The second option of taking $g^{-1}(\hat{y})$ on the right hand side of equation (2) and thus applying the case distinction on the original scale wouldn't help much either. Keep in mind that we would like to address problems where the range of y can vary over several orders of magnitude. In such a case we would in general not be able to find an appropriate value of α to guarantee that for all ranges of y both case distinctions are applied. In other words, only by the choice in equation (2) we do get a distribution of non-constant variance.

The loss function in equation however has jump discontinuities at the lines $|g(y) - \hat{y}| = \alpha$ which can be removed in one way or another. The following smoothed version of the previous equation turned out to work nicely in practice:

$$\mathcal{L}_{\mathrm{s}}(y, \hat{y}) = \begin{cases} (y - g^{-1}(\hat{y}))^2 \left(\frac{1}{|g^{-1}(\hat{y} \pm \alpha) - g^{-1}(\hat{y})|} + \right. \\ \qquad \left. \frac{1}{|y - g^{-1}(g(y) \mp \alpha)|} \right) & \dots \quad |g(y) - \hat{y}| \leq \alpha \\ 4|y - g^{-1}(\hat{y})| - \left(|g^{-1}(\hat{y} \pm \alpha) - g^{-1}(\hat{y})| + \right. \\ \qquad \left. |y - g^{-1}(g(y) \mp \alpha)| \right) & \dots \quad |g(y) - \hat{y}| > \alpha \end{cases}$$

where $\mp = \mathrm{sgn}(g(y) - \hat{y})$.

The function $\mathscr{L}$_s(y_0, $\hat{y}$) for fixed y_0 has no local minima but is not convex either. Moreover, $\mathscr{L}$_s(y_0, $\hat{y}$) exhibits a region of little slope which can lead to convergence issues in gradient based optimisation routines. However, such issues can typically be overcome by the choice of a good starting vector.

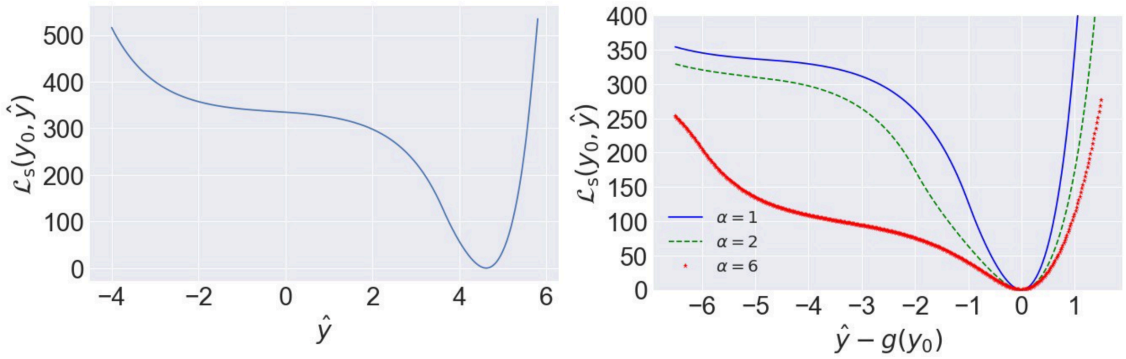


Fig. 15: Left: Smoothed generalised Huber function with y_0 = 100 and α =1. Right: Smoothed generalized Huber function for different values of α at y_0 = 100. Both with link function g(x) = sgn(x) log(1+|x|). [51]

In Figure 15, we illustrate the aforementioned increase of the scale of $\mathscr{P}(y, \hat{y}\_0)$ with increasing $\hat{y}$_0. It is precisely this feature that makes the GHL function robust and applicable to heteroscedastic problems. Note that the scale of $\mathscr{P}(y, \hat{y}\_0)$ does also increase with increasing α as shown on the right hand side of Figure 15. Note that we did not normalise $\mathscr{P}(y, \hat{y}\_0)$. The corresponding normalisation factor would depend on $\hat{y}$_0 and it would be interesting to investigate whether or not a closed-form expression could be derived.

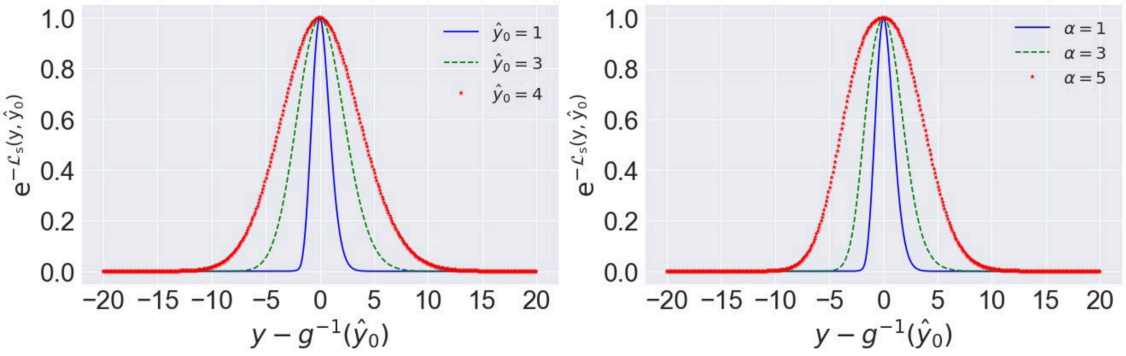


Fig. 16: Left: (Unnormalized) Error distribution for different values of ŷ_0 at α = 1. Right: (Unnormalized) Error distribution at ŷ_0 = 1 for different values of α. Both with link function g(x) = sgn(x) log(1+|x|). [51]

In order to optimise $\mathscr{L}$_s(y, $\hat{y}$) with gradient methods we need the gradient and the Hessian of this function.

The gradient of $\mathscr{L}\_s(y, \hat{y})$ is given by

$$\partial_{\hat{y}} \mathcal{L}_s(y, \hat{y}) = \begin{cases} \Big(-2(y - g^{-1}(\hat{y}))(\partial_{\hat{y}} g^{-1}(\hat{y}))\Big(\frac{1}{|A|} + \frac{1}{|B|}\Big) \\ \quad -(y - g^{-1}(\hat{y}))^2 \mathrm{sgn(A)} \frac{\partial_{\hat{y}} \mathrm{A}}{\mathrm{A}^2}\Big) & \dots\ |g(y) - \hat{y}| \le \alpha \\ \\ -4\ \mathrm{sgn(y - g^{-1}(\hat{y}))}(\partial_{\hat{y}} \mathrm{g}^{-1}(\hat{y})) - \ \mathrm{sgn(A)} \partial_{\hat{y}} \mathrm{A} & \dots\ |g(y) - \hat{y}| > \alpha \end{cases} \quad (4)$$

and the Hessian is found to be

$$\partial_{\hat{y}}^2 \mathcal{L}_s(y, \hat{y}) = \begin{cases} 2\Big((\partial_{\hat{y}} g^{-1}(\hat{y}))^2 - (y - g^{-1}(\hat{y}))(\partial_{\hat{y}}^2 g^{-1}(\hat{y}))\Big)\Big(\frac{1}{|A|} + \frac{1}{|B|}\Big) \\ +4(y - g^{-1}(\hat{y}))(\partial_{\hat{y}} g^{-1}(\hat{y}))\mathrm{sgn}(A) \frac{\partial_{\hat{y}} A}{A^2} \\ +(y - g^{-1}(\hat{y}))^2\Big(2\frac{(\partial_{\hat{y}} A)^2}{|A|^3} - \mathrm{sgn}(A)\ \frac{\partial_{\hat{y}}^2 A}{A^2}\Big) & \dots\ |g(y) - \hat{y}| \le \alpha \\ \\ -4\ \mathrm{sgn}(y - g^{-1}(\hat{y}))(\partial_{\hat{y}}^2 g^{-1}(\hat{y})) - \ \mathrm{sgn}(A)\partial_{\hat{y}}^2 A & \dots\ |g(y) - \hat{y}| > \alpha \end{cases} \quad (5)$$

where we have defined $A = (g^{-1}(\hat{y} \pm \alpha)$ - $g^{-1}(\hat{y}))$ and $B = (\mathrm{y}$ - $g^{-1}(g(y) \mp \alpha))$ with $\mp = \mathrm{sgn}(g(y)$ - $\hat{y})$.

In general one needs a good starting vector in order to converge to the minimum of the GHL loss function. Our execution of the Huber Regression algorithm can be found in the next chapter (Chapter 5, "Experiments") along with the model building with all the other models that are explained in this chapter.

4.2.2 KNN Regression

k-Nearest Neighbors (kNN) is a non-parametric learning algorithm. Contrary to other learning algorithms that allow discarding the training data after the model is built, kNN keeps all training examples in memory. Once a new, previously unseen example x comes in, the kNN algorithm finds k training examples closest to x and returns the majority label, in case of classification, or the average label, in case of regression.

The closeness of two examples is given by a distance function. For example, Euclidean distance seen above is frequently used in practice. Another popular choice of the distance function is the negative cosine similarity. Cosine similarity defined as,

$$s(\mathbf{x}_i, \mathbf{x}_k) \stackrel{\text{def}}{=} \cos(\angle(\mathbf{x}_i, \mathbf{x}_k)) = \frac{\sum_{j=1}^{D} x_i^{(j)} x_k^{(j)}}{\sqrt{\sum_{j=1}^{D} \left(x_i^{(j)}\right)^2} \sqrt{\sum_{j=1}^{D} \left(x_k^{(j)}\right)^2}},$$

is a measure of similarity of the directions of two vectors. If the angle between two vectors is 0 degrees, then two vectors point to the same direction, and cosine similarity is equal to 1. If the vectors are orthogonal, the cosine similarity is 0. For vectors pointing in opposite directions, the

cosine similarity is −1. If we want to use cosine similarity as a distance metric, we need to multiply it by −1. Other popular distance metrics include Chebychev distance, Mahalanobis distance, and Hamming distance. The choice of the distance metric, as well as the value for k, are the choices the analyst makes before running the algorithm. So these are hyperparameters. The distance metric could also be learned from data, instead of making a guess.

KNN Regression allows its users to understand and interpret what's happening inside the model, and it's very fast to develop. This makes kNN a great model for many machine learning use cases that don't require highly complex techniques. However, the main drawback of kNN is its capacity to adapt to highly complex relationships between independent and dependent variables. kNN is less likely to perform well on advanced tasks. We can try to push the performance of kNN as far as possible, potentially by adding other techniques from machine learning. That is why we discussed bagging, as it is a way to improve predictive performances. At a certain point of complexity, though, kNN will probably be less effective than other models regardless of the way it was tuned.

In Chapter 5, "Experiments" we confirmed these theoretical predictions on the accuracy of kNN Regression, as it was one of the worst-behaving algorithms (low prediction accuracy). However, we had to use it as a benchmark, as it is one of the oldest and most used ML algorithms in a variety of problems and in use by many Sales Forecasting publications as a benchmark algorithm, and, thus, we included it too for wholeness and comparison.

4.2.3 Passive Aggressive Regression

The passive-aggressive algorithms are a family of algorithms for large-scale learning. They are similar to the Perceptron in that they do not require a learning rate. However, contrary to the Perceptron, they include a regularisation parameter C.

Passive-Aggressive algorithms are called so because of the following characteristics:

- Passive: If the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model.
- Aggressive: If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.

And based on a slightly different Hinge loss function (called ε-insensitive):

$$L(\bar{\theta}, \varepsilon) = \max\left(0, \left|y_t - f(\bar{x}_t; \bar{\theta})\right| - \varepsilon\right)$$

The parameter ε determines a tolerance for prediction errors. The update conditions are the same adopted for classification problems and the resulting update rule is:

$$\bar{w}_{t+1} = \bar{w}_t + \frac{\max\left(0, |y_t - \bar{w}^T \cdot \bar{x}_t| - \varepsilon\right)}{\|x_t\|^2 + \frac{1}{2C}} sign(y_t - \bar{w}^T \cdot \bar{x}_t)\bar{x}_t$$

Scikit-learn implements a Passive Aggressive Regression, so our simple model building will be presented in Chapter 5 ("Experiments").

4.2.4 Lasso regression

In LASSO Regression, LASSO stands for Least Absolute Shrinkage & Selection Operator. Lasso regression is a regularisation technique. It is used over regression methods for a more accurate prediction and it uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The LASSO procedure encourages simple, sparse models, such as models with fewer parameters. This particular type of regression is well-suited for models showing high levels of multicollinearity or when we want to automate certain parts of model selection, like variable selection or parameter elimination.

Lasso Regression uses an L1 regularisation technique, which is used when we have a lot of features because it automatically performs feature selection. Considering only a single feature, linear regression looks for optimising w (slope) and b (intercept) such that it minimises the cost function. The cost function can be written as:

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 \tag{1.2}$$

In the equation above, we have assumed the dataset has M instances and p features. Once we use linear regression on a dataset divided in to training and test set, calculating the scores on training and test set can give us a rough idea about whether the model is suffering from over-fitting or under-fitting. If we have very few features on a data-set and the score is poor for both training and test set then it's a problem of under-fitting. On the other hand if we have large number of features and test score is relatively poor than the training score then it's the problem of over-generalisation or overfitting. Ridge (explained below) and Lasso regression are some of the simple techniques to reduce model complexity and prevent over-fitting which may result from simple linear regression.

The cost function for Lasso regression can be written as:

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} |w_j| \tag{1.4}$$

Where,

- λ denotes the amount of shrinkage.
- λ = 0 implies all features are considered and it is equivalent to the linear regression where only the residual sum of squares is considered to build a predictive model
- λ = ∞ implies no feature is considered i.e, as λ closes to infinity it eliminates more and more features
- The bias increases with increase in λ
- variance increases with decrease in λ

This is equivalent to saying minimising the cost function in equation 1.2 under the condition:

$$\text{For some } t > 0, \sum_{j=0}^{p} |w_j| < t$$

The only difference with the general case is that instead of taking the square of the coefficients, magnitudes are taken into account. This type of regularisation (L1) can lead to zero coefficients i.e. some of the features are completely neglected for the evaluation of output. So, Lasso regression not only helps in reducing overfitting but it helps us in feature selection too. The regularisation parameter (lambda) can be controlled and so feature selection using Lasso regression can be depicted well by changing the regularization parameter.

4.2.5 Ridge regression

Similar to Lasso Regression, Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization (contrary to Lasso's L1 regularization explained above). When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values.

In ridge regression, the first step is to standardise the variables (both dependent and independent) by subtracting their means and dividing by their standard deviations. This causes a challenge in notation since we must somehow indicate whether the variables in a particular formula are standardised or not. As far as standardisation is concerned, all ridge regression calculations are based on standardised variables. When the final regression coefficients are displayed, they are adjusted back into their original scale. However, the ridge trace is on a standardised scale.

Bias and variance trade-off is generally complicated when it comes to building ridge regression models on an actual dataset. However, following the general trend which one needs to remember is:

1. The bias increases as λ increases.
2. The variance decreases as λ increases.

In ridge regression, the cost function is altered by adding a penalty equivalent to square of the magnitude of the coefficients.

$$\sum_{i=1}^{M} (y_i - \hat{y_i})^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} w_j^2 \qquad (1.3)$$

This is equivalent to saying minimising the cost function in equation (1.3) under the condition:

$$\text{For some } c > 0, \sum_{j=0}^{p} w_j^2 < c$$

So Ridge Regression puts constraint on the coefficients (w). The penalty term (lambda) regularises the coefficients such that if the coefficients take large values the optimisation function is penalised. So, ridge regression shrinks the coefficients and it helps to reduce the model complexity and multi-collinearity. Going back to eq. 1.3 one can see that when $\lambda \to 0$, the cost function becomes similar to the linear regression cost function. So lower the constraint (low λ) on the features and the model will resemble linear regression model.

How Lasso Regularization Leads to Feature Selection?

So far we have gone through the basics of Ridge and Lasso regression and seen some examples to understand the applications. Now, I will try to explain why the Lasso regression can result in feature selection and Ridge regression only reduces the coefficients close to zero, but not zero.
In Figure 17, we assume a hypothetical data-set with only two features. Using the constrain for the coefficients of Ridge and Lasso regression, we can plot the figure below:

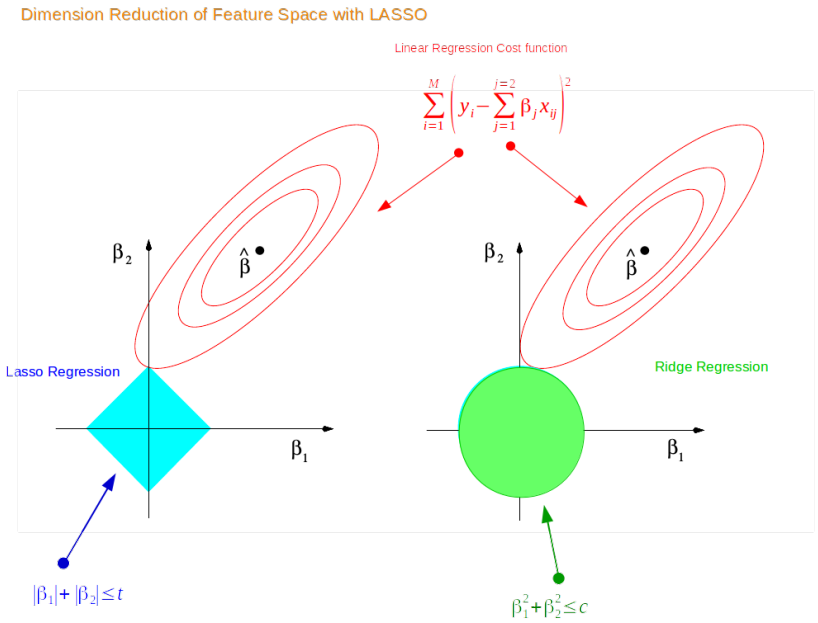
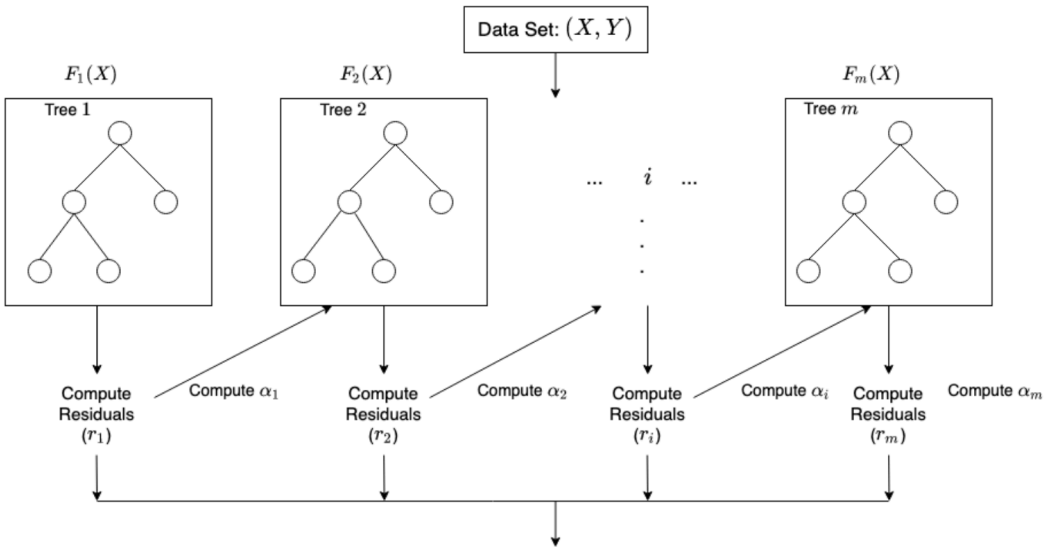


Fig. 17: Why LASSO can reduce dimension of feature space? Example on 2D feature space. Modified from the plot used in 'The Elements of Statistical Learning' [55]

For a two dimensional feature space, the constraint regions are plotted for Lasso and Ridge regression with cyan and green colours. The elliptical contours are the cost function of linear regression. Now if we have relaxed conditions on the coefficients, then the constrained regions can get bigger and eventually they will hit the centre of the ellipse. This is the case when Ridge and Lasso regression resembles linear regression results. Otherwise, both methods determine coefficients by finding the first point where the elliptical contours hit the region of constraints. The diamond (Lasso) has corners on the axes, unlike the disk, and whenever the elliptical region hits such point, one of the features completely vanishes. For higher dimensional feature space there can be many solutions on the axis with Lasso regression and thus we get only the important features selected.

4.3 Gradient Boosting Models (GBDT)

Gradient-boosted decision trees are a machine learning technique for optimizing the predictive value of a model through successive steps in the learning process. Each iteration of the decision tree involves adjusting the values of the coefficients, weights, or biases applied to each of the input variables being used to predict the target value, with the goal of minimising the loss function (the measure of difference between the predicted and actual target values). The gradient is the incremental adjustment made in each step of the process; boosting is a method of accelerating the improvement in predictive accuracy to a sufficiently optimum value.



Gradient-boosted decision trees are a popular method for solving prediction problems in both classification and regression domains. The approach improves the learning process by simplifying the objective and reducing the number of iterations to get to a sufficiently optimal solution. Gradient-boosted models have proven themselves time and again in various competitions grading on both accuracy and efficiency, making them a fundamental component in the data scientist's tool kit.

In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now. We will see that also in our experiments in Chapter 5, as, indeed, our decision tree based algorithms achieved the optimal accuracy.

4.3.1 XGBoost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. XGBoost algorithm was developed as a research project by Tianqi Chen and Carlos Guestrin in 2016 and revolutionised machine learning. Since its introduction, XGBoost

has not only been credited with winning numerous Kaggle competitions, but also for being the driving force under the hood for several cutting-edge industry applications. It can be used in a wide range of applications, such as regression, classification, ranking, and user-defined prediction problems.

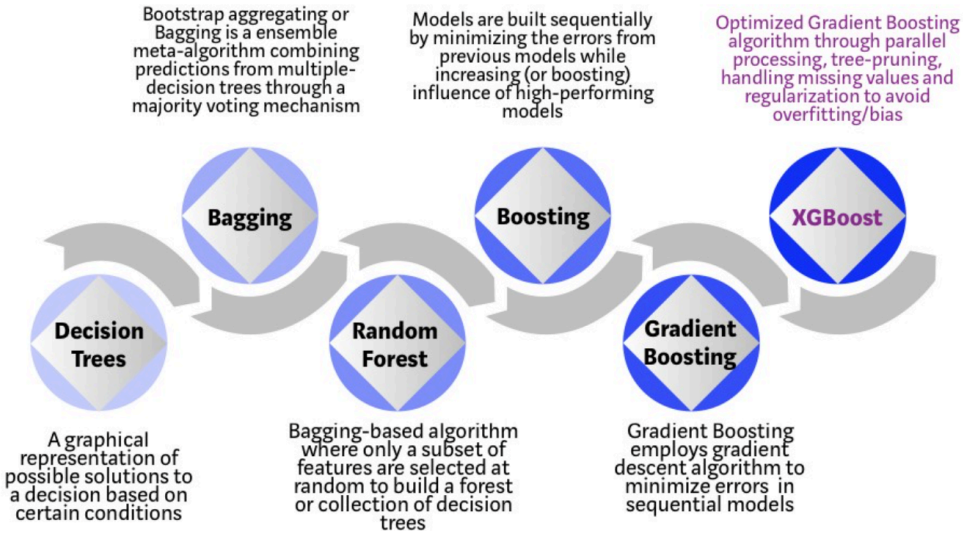


Fig. 18: From Decision Trees to XGBoost [56]

- **Decision Tree:** Every hiring manager has a set of criteria. A decision tree is analogous to a hiring manager interviewing candidates based on its own set of criteria.
- **Bagging:** In this step we have something like an interview panel where each interviewer has a vote. Bagging or bootstrap aggregating involves combining inputs from all interviewers for the final decision through a democratic voting process.
- **Random Forest:** It is a bagging-based algorithm with a key difference that only a subset of features is selected at random. In other words, every interviewer will only test the interviewee on certain randomly selected qualifications, such as a technical interview for testing programming skills or a behavioural interview for evaluating non-technical skills.
- **Boosting:** This is an alternative approach where each interviewer alters the evaluation criteria based on the previous interviewer's feedback. This step boosts the interview process' overall efficiency by deploying a more dynamic evaluation process.
- **Gradient Boosting:** This is a special case of boosting, where errors are minimised by a gradient descent algorithm. So, it is the analogous to a strategy consulting firm using old case interviews to weed out less qualified candidates.
- **XGBoost:** The XGBoost algorithm uses extreme gradient boosting (that is its name origin). It is a perfect combination of software and hardware optimization techniques to yield superior results using less computing resources in the shortest amount of time.

XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

So, with XGBoost, we predict the target label using all of the trees within the ensemble. Each sample passes through the decision nodes of the newly formed tree until it reaches a given lead.

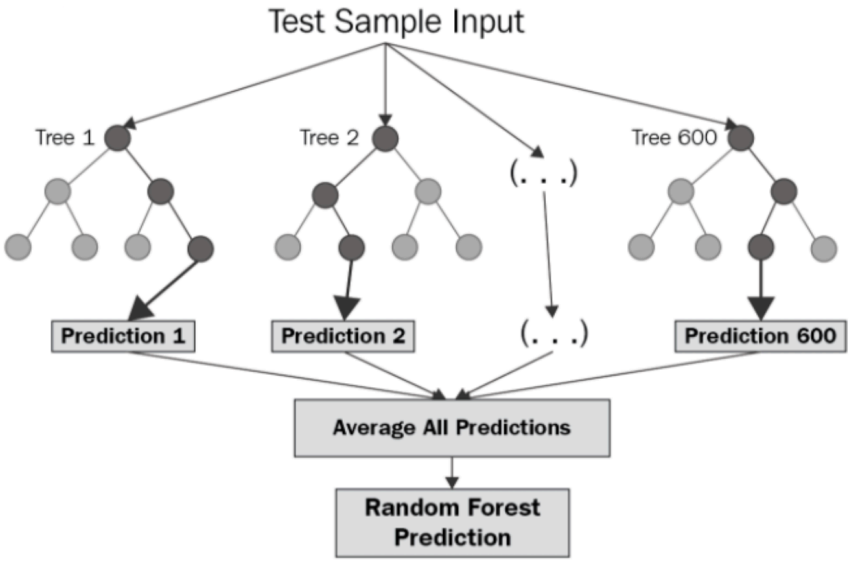
It's been shown through experimentation that taking small incremental steps towards the solution achieves a comparable bias with a lower overall variance (a lower variance leads to better accuracy on samples outside of the training data). Thus, to prevent overfitting, in the XGBoost algorithm a new hyperparameter was introduced: the learning rate. When we make a prediction, each residual is multiplied by the learning rate and this leads to using more decision trees, each taking a small step towards the final solution.

So, in Sales Forecasting we calculate a new set of residuals by subtracting the actual number and/ or value of sales from the predictions made in the previous step. The residuals will then be used for the leaves of the next decision tree. And once trained, by combining all of the trees in the ensemble we make a final prediction as to the value of the target variable (in our case sales volume). The final prediction will be equal to the mean we computed in the first step, plus all of the residuals predicted by the trees that make up the forest multiplied by the learning rate.

We will use the XGBoost algorithm both in our original datasets and experiments and also in our work for a Kaggle competition for sales forecasting, all of which are in Chapter 5 of this Thesis.

4.3.2 Random Forest Regressor

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.



The diagram above shows the structure of a Random Forest. One can notice that the trees run in parallel with no interaction amongst them. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

To get a better understanding of the Random Forest algorithm, one can think of the following algorithm:

- Pick at random k data points from the training set.
- Build a decision tree associated to these k data points.

- Choose the number N of trees you want to build and repeat steps 1 and 2.
- For a new data point, make each one of your N-tree trees predict the value of y for the data point in question and assign the new data point to the average across all of the predicted y values.

A Random Forest Regression model is powerful and accurate. It usually performs great on many problems, including features with non-linear relationships. Disadvantages, however, include the following: there is no interpretability, overfitting may easily occur, we must choose the number of trees to include in the model. Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. One can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

Another great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction. Sklearn provides a great tool for this that measures a feature's importance by looking at how much the tree nodes that use that feature reduce impurity across all trees in the forest. It computes this score automatically for each feature after training and scales the results so the sum of all importance is equal to one.

By looking at the feature importance we can decide which features to possibly drop because they don’t contribute enough to the prediction process. This is important because a general rule in machine learning is that the more features we have the more likely our model will suffer from overfitting.

While random forest is a collection of decision trees, there are some differences. If we input a training dataset with features and labels into a decision tree, it will formulate some set of rules, which will be used to make the predictions. For example, to predict whether a person will click on an online advertisement, we might collect the ads the person clicked on in the past and some features that describe his/her decision. If we put the features and labels into a decision tree, it will generate some rules that help predict whether the advertisement will be clicked or not. In comparison, the random forest algorithm randomly selects observations and features to build several decision trees and then averages the results.

Another difference is "deep" decision trees might suffer from overfitting. Most of the time, random forest prevents this by creating random subsets of the features and building smaller trees using those subsets. Afterwards, it combines the subtrees. It's important to note this doesn’t work every time and it also makes the computation slower, depending on how many trees the random forest builds.

Random forest is a great algorithm to train early in the model development process, to see how it performs. Its both very simple, but also effective, as it makes highly unlikely the possibility of building a low-accuracy random forest. The algorithm is also a great choice for anyone who needs to develop a model quickly and it provides a pretty good indicator of the importance it assigns to your features.

Random forests are also very hard to beat performance wise. That’s why they are the most popular benchmarking model for machine learning projects. Of course, we can probably always

find a model that can perform better, like a neural network for example, but these usually take more time to be developed and executed.

Overall, Random Forest is a flexible tool, based on a simple, yet effective algorithm, but with significant limitations. At our experiments (Chapter 5, “Experiments”) we validate the theoretical assumptions about Random Forests, as we show that Random Forests can be implemented really easily, have extremely low execution time and have rather good prediction accuracy at Sales Forecasting for FMCGs, but definitely not one of the best accuracies overall. Thus, they can be used as a start and benchmark model in these applications to give a general idea about the Sales Forecasting, and be replaced by other models we analyse in this Thesis, which achieve much better accuracy at different levels of data amounts.

4.3.3 Catboost

CatBoost is an algorithm for gradient boosting on decision trees. It is developed by Yandex researchers and engineers, it is universal, and it can be applied across a wide range of areas and to a variety of problems. Gradient Boosted Decision Trees are the best ML models for tabular heterogeneous datasets (such as our sales data for sales forecasting). These models are the top performers on Kaggle competitions and in widespread use in the industry.

Catboost achieves the best results on the benchmark. However, it hasn’t yet seen big integration into businesses’ ML models, as replacing a working production model for only a fraction of a log-loss improvement alone does not considered as a good investment by big companies data divisions. We hope to give extra value and motivation to this kind of businesses with this Thesis.

However, for datasets where categorical features play a large role this improvement becomes even more significant and undeniable. That is the main reason we explore it in this Thesis, as categorical features are extremely significant in Sales Forecasting and even more important for FMCGs Sales Forecasting. Our assumption that Catboost can have a big impact in future Sales Forecasting ML models was validated, as it was the best model for our dataset in our ML Sales Forecasting.

Catboost’s has the following strong points:

- While training time can take up longer than other GBDT implementations, prediction time is 13-16 times faster than the other libraries according to the Yandex benchmark.
- Catboost’s default parameters are a better starting point than in other GBDT algorithms. And this is good news for beginners who want a plug and play model to start experience tree ensembles or Kaggle competitions.
- Some more noteworthy advancements by Catboost are the features interactions, object importance and the snapshot support.
- In addition to classification and regression, Catboost supports ranking out of the box.
- Catboost introduces two critical algorithmic advances - the implementation of ordered boosting, a permutation-driven alternative to the classic algorithm, and an innovative algorithm for processing categorical features. Both techniques are using random permutations of the training examples to fight the prediction shift caused by a special kind of target leakage present in all existing implementations of gradient boosting algorithms.

Ordered Target Statistic

Most of the GBDT algorithms use the idea of Target Statistic (or target mean encoding). It’s a simple yet effective approach in which we encode each categorical feature with the estimate of the expected target y conditioned by the category. Well, it turns out that applying this encoding

carelessly (average value of y over the training examples with the same category) results in a target leakage.

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
   1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

   2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
   3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

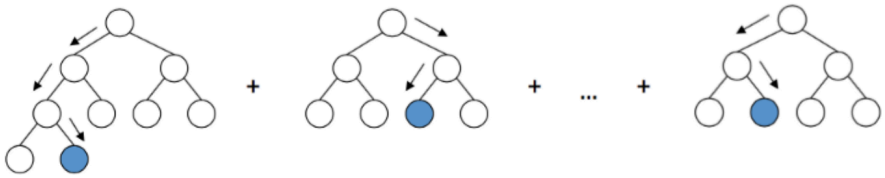
$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Gradient Boosting on Wikipedia



To fight this prediction shift, CatBoost uses a more effective strategy. It relies on the ordering principle and is inspired by online learning algorithms which get training examples sequentially in time. In this setting, the values of the Target Statistic (TS) for each example rely only on the observed history.

To adapt this idea to a standard offline setting, Catboost introduces an artificial “time”-a random permutation σ1 of the training examples. Then, for each example, it uses all the available history to compute its Target Statistic. Note that, using only one random permutation, results in preceding examples with higher variance in Target Statistic than subsequent ones. To this end, CatBoost uses different permutations for different steps of gradient boosting.

One Hot Encoding

Catboost uses a one-hot encoding for all the features with at most one_hot_max_size unique values. The default value is 2. Catboost’s algorithm is the classic Gradient Boosting with the aforementioned advantages.

Let’s take a look at its algorithm:

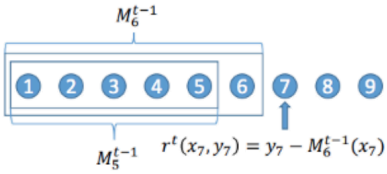


Figure 1: Ordered boosting principle.

Algorithm 1: Ordered boosting

input : $\{(\mathbf{x}_k, y_k)\}_{k=1}^n$, I;

$\sigma \leftarrow$ random permutation of $[1, n]$;
$M_i \leftarrow 0$ for $i = 1..n$;
for $t \leftarrow 1$ **to** I **do**
  for $i \leftarrow 1$ **to** n **do**
    $r_i \leftarrow y_i - M_{\sigma(i)-1}(i)$;
  for $i \leftarrow 1$ **to** n **do**
    $\Delta M \leftarrow$ $LearnModel((\mathbf{x}_j, r_j) : \sigma(j) \leq i)$;
    $M_i \leftarrow M_i + \Delta M$;
return M_n

Algorithm 2: Building a tree in CatBoost

input : $M, \{y_i\}_{i=1}^n, \alpha, L, \{\sigma_i\}_{i=1}^s, Mode$

$grad \leftarrow CaclGradient(L, M, y)$;
$r \leftarrow random(1, s)$;
$G \leftarrow (grad_r(1), \ldots, grad_r(n))$ for $Plain$;
$G \leftarrow (grad_{r,\sigma_r(1)-1}(i)$ for $i = 1$ to $n)$ for $Ordered$;
$T \leftarrow$ empty tree;
foreach *step of top-down procedure* **do**
  foreach *candidate split* c **do**
    $T_c \leftarrow$ add split c to T;
    if $Mode == Plain$ **then**
      $\Delta(i) \leftarrow \text{avg}(grad_r(p)$ for $p : leaf(p) = leaf(i))$ for all i;
    if $Mode == Ordered$ **then**
      $\Delta(i) \leftarrow \text{avg}(grad_{r,\sigma_r(i)-1}(p)$ for $p : leaf(p) = leaf(i), \sigma_r(p) < \sigma_r(i))$ $\forall i$;
    $loss(T_c) \leftarrow ||\Delta - G||_2$
  $T \leftarrow \text{argmin}_{T_c}(loss(T_c))$
if $Mode == Plain$ **then**
  $M_{r'}(i) \leftarrow M_{r'}(i) - \alpha\,\text{avg}(grad_{r'}(p)$ for $p : leaf(p) = leaf(i))$ for all r', i;
if $Mode == Ordered$ **then**
  $M_{r',j}(i) \leftarrow M_{r',j}(i) - \alpha\,\text{avg}(grad_{r',j}(p)$ for $p : leaf(p) = leaf(i), \sigma_{r'}(p) \leq j$ for all r', j, i;
return T, M

CatBoost has two modes for choosing the tree structure, Ordered and Plain. Plain mode corresponds to a combination of the standard GBDT algorithm with an ordered Target Statistic. In Ordered mode boosting we perform a random permutation of the training examples - σ_2, and maintain n different supporting models - $M_1, \ldots, M_n$ such that the model M_i is trained using only the first i samples in the permutation. At each step, in order to obtain the residual for j-th sample, we use the model $M_j{-}1$. Unfortunately, this algorithm is not feasible in most practical tasks due to the need of maintaining n different models, which increase the complexity and memory requirements by n times. Catboost implements a modification of this algorithm, on the basis of the gradient boosting algorithm, using one tree structure shared by all the models to be built.

In order to avoid prediction shift, Catboost uses permutations such that $\sigma_1 = \sigma_2$. This guarantees that the target-y is not used for training M_i neither for the Target Statistic calculation nor for the gradient estimation.

4.3.4 LightGBM

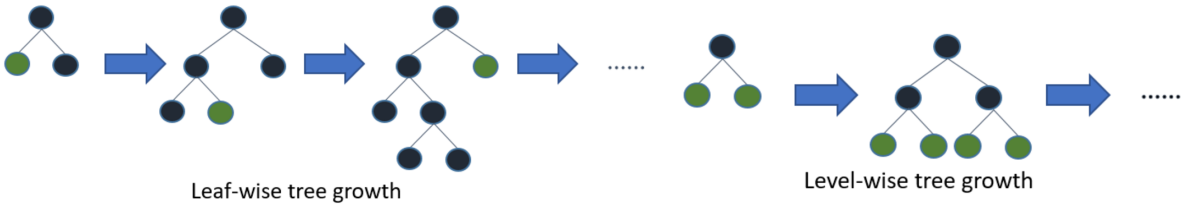
LightGBM is a gradient boosting framework that uses tree based learning algorithms.

It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel, distributed, and GPU learning.

- Capable of handling large-scale data.

It's main difference from other GBDT algorithms is that while other algorithms' trees grow horizontally, LightGBM (LGBM) algorithm grows vertically. This means that while other algorithms grow level-wise, LGBM grows leaf-wise. LGBM chooses the leaf with large loss to grow and it can lower down more loss than a level wise algorithm when growing the same leaf.



How LBGM works / How other GBDT work

One of the main reasons for its increased popularity, is that it has become difficult for a lot of algorithms to give results fast for big bunches of data. LightGBM is called "Light" because of its computation power and giving results faster. It takes less memory to run and is able to deal with large amounts of data. Another reason of why Light GBM is popular is because it focuses on accuracy of results. LGBM also supports GPU learning and thus data scientists are widely using LGBM for data science application development.

So, LightGBM is not for a small volume of datasets. It can easily overfit small data due to its sensitivity. It can be used for data having more than 10,000+ rows (such as our datasets for sales forecasting). There is no fixed threshold that helps in deciding the usage of LightGBM. It can be used for large volumes of data especially when one needs to achieve a high accuracy. Thus, LGBM seems to be one of the most promising models/candidates for Sales Forecasting.

Especially in our case, where we study Sales Forecasting for FMCGs, LGBM is a promising model as:

- It is perfect for big amounts of data which is one of the main characteristics of working with FMCGs
- It gives results extremely fast and so it is perfect for constant re-runs and a lot of model tweaking that is usually required when working with FMCGs
- It has all the advantages of GBDT algorithms which are very fast and some of the best models, regarding accuracy of predictions for forecasting

4.4 Meta-Learning

As we explained in Chapter 2, Ensemble Learning is a learning paradigm that, instead of trying to learn one super-accurate model, focuses on training a large number of low-accuracy models and then combining the predictions given by those weak models to obtain a high-accuracy meta-model.

In this section, we study a specific class of ensemble learning methods which produce models called meta-learning models (hence the title of this section). That is models that attempt to learn from the output or learn how to best combine the output of other lower-level models. Meta-learning is a process of learning from learners/classifiers.

In order to induce a meta-classifier, first the base classifiers are trained (stage one), and then the Meta-classifier (second stage). We will distinctively show these two stages for training Meta-Learning models in Chapter 5 (“Experiments”).

4.4.1 Stacking Classifier/Regressor

Stacking is an ensemble learning technique to combine multiple classification models via a meta-classifier. The individual classification models are trained based on the complete training set; then, the meta-classifier is fitted based on the outputs (meta-features) of the individual classification models in the ensemble. The meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.

For details see Chapter 3, “3.2 Ensemble Learning”.

4.4.2 Mixture of Experts

Mixture of experts is an ensemble learning technique developed in the field of neural networks.
It involves decomposing predictive modeling tasks into sub-tasks, training an expert model on each, developing a gating model that learns which expert to trust based on the input to be predicted, and combines the predictions.

Although the technique was initially described using neural network experts and gating models, it can be generalized to use models of any type. As such, it shows a strong similarity to stacked generalization and belongs to the class of ensemble learning methods referred to as meta-learning.

There are four elements MoE’s approach:

- Division of a task into subtasks. (Step 1: Subtasks)
- Develop an expert for each subtask. (Step 2: Expert Models)
- Use a gating model to decide which expert to use. (Step 3: Gating Model)
- Pool predictions and gating model output to make a prediction. (Step 4: Pooling Method)

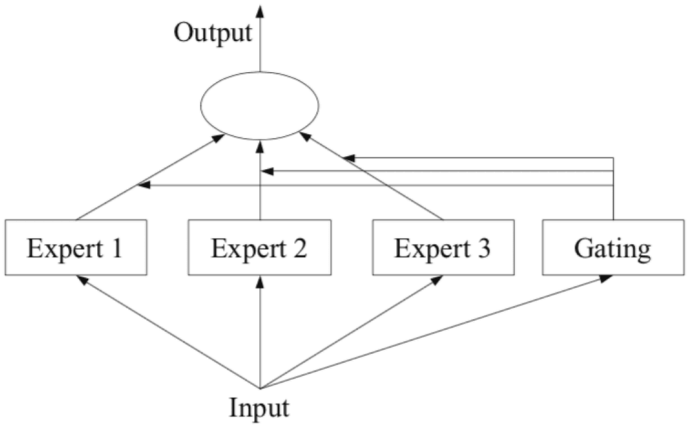


Fig. 19: Example of a MoE Model with Expert Members and a Gating Network [54]

Step1: Subtasks
The first step is to divide the predictive modeling problem into subtasks. This often involves using domain knowledge. For example, an image could be divided into separate elements such as background, foreground, objects, colours, lines etc. So, with the MoE we start with a divide-and-conquer strategy where a complex task is broken up into several simpler and smaller subtasks, and individual learners (called experts) are trained for different subtasks.

For those problems where the division of the task into subtasks is not obvious, a simpler and more generic approach could be used. For example, one could imagine an approach that divides the input feature space by groups of columns or separates examples in the feature space based on distance measures, inliers, and outliers for a standard distribution, and much more.

Step 2: Expert Models
Next, an expert is designed for each subtask. The Mixture of Experts approach was initially developed and explored within the field of artificial neural networks, so traditionally, experts themselves are neural network models used to predict a numerical value in the case of regression or a class label in the case of classification. Experts each receive the same input pattern (row) and make a prediction.

Step 3: Gating Model
A model is used to interpret the predictions made by each expert and to aid in deciding which expert to trust for a given input. This is called the gating model, or the gating network, given that it is traditionally a neural network model. The gating network takes as input the input pattern that was provided to the expert models and outputs the contribution that each expert should have in making a prediction for the input. So, the weights determined by the gating network are dynamically assigned based on the given input, as the MoE effectively learns which portion of the feature space is learned by each ensemble member.

The gating network is key to the approach and effectively the model learns to choose the type subtask for a given input and, in turn, the expert to trust to make a strong prediction. MoE can also be seen as a classifier selection algorithm, where individual classifiers are trained to become experts in some portion of the feature space.

When neural network models are used, the gating network and the experts are trained together such that the gating network learns when to trust each expert to make a prediction. This training procedure was traditionally implemented using expectation maximization (EM). The gating network might have a softmax output that gives a probability-like confidence score for each

expert. In general, the training procedure tries to achieve two goals: for given experts, to find the optimal gating function; for a given gating function, to train the experts on the distribution specified by the gating function.

Step 4: Pooling Method
Finally, the Mixture of Experts models must make a prediction, and this is achieved using a pooling or aggregation mechanism. This might be as simple as selecting the expert with the largest output or confidence provided by the gating network. Alternatively, a weighted sum prediction could be made that explicitly combines the predictions made by each expert and the confidence estimated by the gating network. So, the pooling/combining system chooses a single classifier, the one with the highest weight, or calculates a weighted sum of the classifier outputs for each class, and picks the class that receives the highest weighted sum.

We can also see a relationship between a mixture of experts to Classification And Regression Trees, often referred to as CART. Decision trees are fit using a divide and conquer approach to the feature space. Each split is chosen as a constant value for an input feature and each sub-tree can be considered a sub-model. We could take a similar recursive decomposition approach to decomposing the predictive modeling task into subproblems when designing the mixture of experts. This is generally referred to as a hierarchical mixture of experts.

The hierarchical mixtures of experts (HME) procedure can be viewed as a variant of tree-based methods. The main difference is that the tree splits are not hard decisions but rather soft probabilistic ones. Unlike decision trees, the division of the task into subtasks is often explicit and top-down. Also, unlike a decision tree, the mixture of experts attempts to survey all of the expert sub-models rather than a single model. There are other differences between HMEs and the CART implementation of trees. In an HME, a linear (or logistic regression) model is fit in each terminal node, instead of a constant as in CART. The splits can be multiway, not just binary, and the splits are probabilistic functions of a linear combination of inputs, rather than a single input as in the standard use of CART.

The application of the technique does not have to be limited to neural network models and a range of standard machine learning techniques can be used in place seeking a similar end. In this way, the MoE method belongs to a broader class of ensemble learning methods that would also include stacking. Like a MoE, stacking trains a diverse ensemble of machine learning models and then learns a higher-order model to best combine the predictions.

Unlike a Mixture of Experts, stacking models (Stacking Classifier/Regressors) are often all fit on the same training dataset, having no decomposition of the task into subtasks. And also unlike a mixture of experts, the higher-level model that combines the predictions from the lower-level models typically does not receive the input pattern provided to the lower-level models and instead takes as input the predictions from each lower-level model. Meta-learning methods are best suited for cases in which certain classifiers consistently correctly classify, or consistently misclassify, certain instances.

Nevertheless, there is no reason why hybrid stacking and mixture of expert models cannot be developed that may perform better than either approach in isolation on a given predictive modelling problem. For both of those reasons we assumed that meta-learning models are an extremely good addition in Sales Forecasting for FMCGs.

4.5 Deep Learning models

We have seen both the Theoretical Background and the Basics of Deep Learning models in Chapter 2 (“2.4 Deep Learning”). Here we analyse a bit more on the specifics of the Deep Learning models we used in Chapter 5 (“Experiments”) for Deep Learning Sales Forecasting for FMCGs.

4.5.1 Keras Regressor

There are many deep learning libraries out there, but the most popular ones are TensorFlow, Keras, and PyTorch. Although TensorFlow and PyTorch are immensely popular, they are not easy to use and have a steep learning curve. So, for many practitioners, Keras is the preferred choice.

The Keras library is a high-level API for building deep learning models that has gained favour for its ease of use and simplicity facilitating fast development. Often, building a very complex deep learning network with Keras can be achieved with only a few lines of code. However, we need to say that we will also use PyTorch to develop our LSTM model. The reasons why the PyTorch library is beneficial for our LSTM model will be found in the next sections.

The basic architecture of the deep learning neural network, which we will be following, consists of three main components.

- Input Layer: This is where the training observations are fed. The number of predictor variables is also specified here through the neurons.
- Hidden Layers: These are the intermediate layers between the input and output layers. The deep neural network learns about the relationships involved in data in this component.
- Output Layer: This is the layer where the final output is extracted from what’s happening in the previous two layers. In case of regression problems, the output later will have one neuron.

These are the main steps for implementing Regression models with Keras:

- Step 1: Loading the required libraries and modules.
- Step 2: Loading the data and performing basic data checks.
- Step 3: Creating arrays for the features and the response variable.
- Step 4: Creating the training and test datasets.
- Step 5: Define, compile, and fit the Keras regression model.
- Step 6: Predict on the test data and compute evaluation metrics.

Both the basic architecture of the deep learning neural network we used and the main steps for implementing Regression models (in this case Sales Forecasting) with Keras will be shown in Chapter 5 (“Experiments”).

4.5.2 MLP

We gave a full analysis of MLP in Chapter 2 (“2.4 Deep Learning”). However, for completeness, let’s explain again the basic characteristics of Neural Networks (NN) and Multilayer Perceptrons

(MLP) as they play a huge part in the development of the modern deep learning models and networks.

We've seen here that the Perceptron, that neural network whose name evokes how the future looked from the perspective of the 1950s, is a simple algorithm intended to perform binary classification; i.e. it predicts whether input belongs to a certain category of interest or not. The perceptron is a linear classifier, with input typically a feature vector x multiplied by weights w and added to a bias b: $y = w*x + b$.

Perceptrons produce a single output based on several real-valued inputs by forming a linear combination using input weights (and sometimes passing the output through a non-linear activation function). Rosenblatt built a single-layer perceptron; it did not include multiple layers, which allow neural networks to model a feature hierarchy. It was, therefore, a shallow neural network, which ended up preventing his perceptron from performing non-linear classification, such as the classic logic XOR function (an XOR operator trigger when input exhibits either one trait or another, but not both; it stands for "exclusive OR").

Fast forward to 1986, when Hinton, Rumelhart, and Williams published a paper "Learning representations by back-propagating errors", introducing backpropagation and hidden layers concepts, and so giving birth to Multilayer Perceptrons (MLPs):

- Backpropagation, a procedure to repeatedly adjust the weights so as to minimise the difference between actual output and desired output
- Hidden Layers, which are neuron nodes stacked in between inputs and outputs, allowing neural networks to learn more complicated features (such as XOR logic)

Therefore, an MLP can be thought of as a deep artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP.

Multilayer perceptrons train on a set of input-output pairs and learn to model the correlation (or dependencies) between those inputs and outputs. Training involves adjusting the parameters, or the weights and biases, of the model in order to minimise error. Backpropagation is used to make those weigh and bias adjustments relative to the error, and the error itself can be measured in a variety of ways, including by root mean squared error (RMSE). Of course, RMSE is one of the key metrics we will use in evaluating all of our models in Chapter 5 ("Experiments").

Feedforward networks such as MLPs are mainly involved in two motions, a constant back and forth (forward and backward passes):

- In the forward pass, the signal flow moves from the input layer through the hidden layers to the output layer, and the decision of the output layer is measured against the ground truth labels.
- In the backward pass, using backpropagation and the chain rule of calculus, partial derivatives of the error function regarding the various weights and biases are back-propagated through the MLP. That act of differentiation gives us a gradient, or a landscape of error, along which the parameters may be adjusted as they move the MLP one step closer to the error minimum. (this can be done with any gradient-based optimization algorithm such as stochastic gradient descent).

The network keeps moving back and forth until the error reaches a minimum, and this is the moment of convergence. Of course, the algorithm is completes and finishes when it reaches convergence.
To benchmark Machine Learning and Deep Learning models, we built a simple MLP network which seems to be a simple, yet overall effective, solution for Sales Forecasting for FMCGs.

4.5.3 LSTM

Long Short-Term Memory (LSTM) networks started as complicated solutions to very specific problems involving patterns, but they have become more and more useful. Especially in Sales Forecasting they seem very promising as they are a type of artificial neural network designed to recognise patterns in sequences of data, such as numerical times series data.

Especially algorithms that improve the standard LSTM networks or create additional states to make them perfect for Sales Forecasting have been built and introduced in the last 2 years. Yonghe Zhao et al publication "Optimization of a Comprehensive Sequence Forecasting Framework Based on DAE-LSTM Algorithm" that was published just a few months before this Thesis totally change the field of Sales Forecasting algorithms and methodologies.

(Yonghe Zhao et al, "Optimization of a Comprehensive Sequence Forecasting Framework Based on DAE-LSTM Algorithm", J. Phys., Conf. Ser. 1746 012087, 2021. https://iopscience.iop.org/article/10.1088/1742-6596/1746/1/012087)

For that reason, we had to train an LSTM network for Sales Forecasting for FMCGs. What we found out is that, along with all the more advanced models, it over-fits really fast. The prediction accuracy it provided was very good, but not worth the extra time and skills investment. Of course, as it happens with all the models included in this Chapter, their final built, their optimal parameters and their results are presented in Chapter 5 ("Experiments").

So, let's give a theoretical analysis and background of Long Short-Term Memory (LSTM) networks and we will see them in action trying to create predictions for Sales Forecasting for FMCGs in the next Chapter.

LSTMs are the most powerful and well known subset of Recurrent Neural Networks (RNNs), which have been explained in detail in Chapter 2 ("Theoretical Background"). What differentiates RNNs and LSTMs from other neural networks is that they take time and sequence into account, they have a temporal dimension.

One of the appeals of RNNs was always the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they'd be extremely useful. However, it was found out that there is a plethora of cases where they can't.

Sometimes, we only need to look at recent information to perform the present task. In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information. But there are also cases where we need more context. It's entirely possible for the gap between the relevant information and the point where it is needed to become very large, and, unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

In theory, RNNs are absolutely capable of handling such "long-term dependencies." A human could carefully pick parameters for them to solve toy problems of this form. However, in practice, RNNs don't seem to be able to learn them. The problem was explored in depth by Hochreiter (1991) and Bengio, et al. (1994), who found some pretty fundamental reasons why it might be difficult. That's why LSTM networks came into the light, as they don't face this kind of problems. LSTMs were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behaviour and their main and extremely effective advantage.

All recurrent neural networks have the form of a chain of repeating modules of a neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

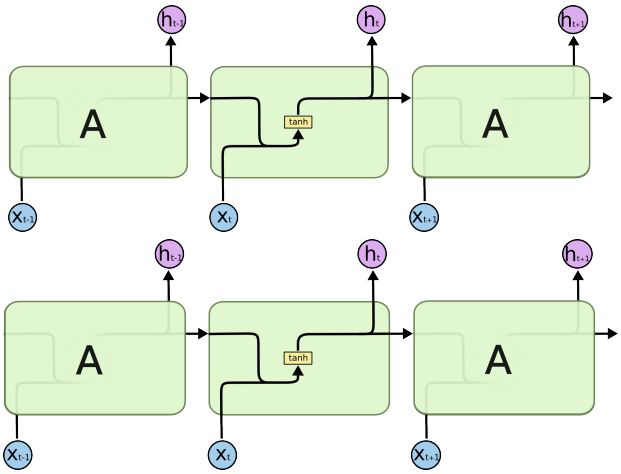


Fig 20: The repeating module in a standard RNN contains a single layer. [57]

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

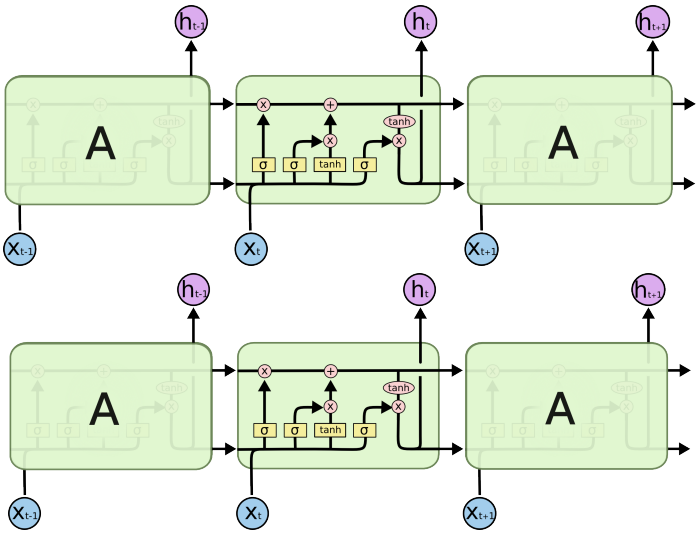
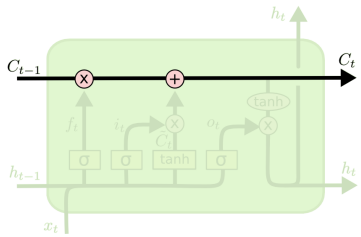


Fig 21: The repeating module in an LSTM contains four interacting layers. [57]

In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent point-wise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.



The LSTM has the ability to remove or add information to the cell state, carefully regulated by structures called gates (we discussed about gates in Chapter 2). Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a point-wise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through”. An LSTM has three of these gates, to protect and control the cell state.

LSTMs were a big step in what we can accomplish with RNNs. But a lot of room for improvement is also there. Future improvements in LSTM networks seem to be the solution in a raft of problems that cannot be effectively addressed with current models and networks.

We discussed about Yonghe Zhao et al publication of a new DAE-LSTM framework that was published just few months before the completion of this Thesis. LSTM networks and their variants seem to be great for Sales Forecasting, maybe not for FMCGs due to the specific characteristics of fast-moving consumer goods, but for other industries and product categories. where we don’t have heterogeneous tabular data structures, and, thus, there is a need to use deep learning models.

We further analyse the results of our experiments and our thoughts of the use of LSTM variants in Future Work for Sales Forecasting in the next chapters.

5. Experiments

5.1 Data

We managed to gather full data from two big european marketplaces with millions in sales in detergents and cleaning products. In case you need access to our direct data, feel free to email us and we will provide them to you.

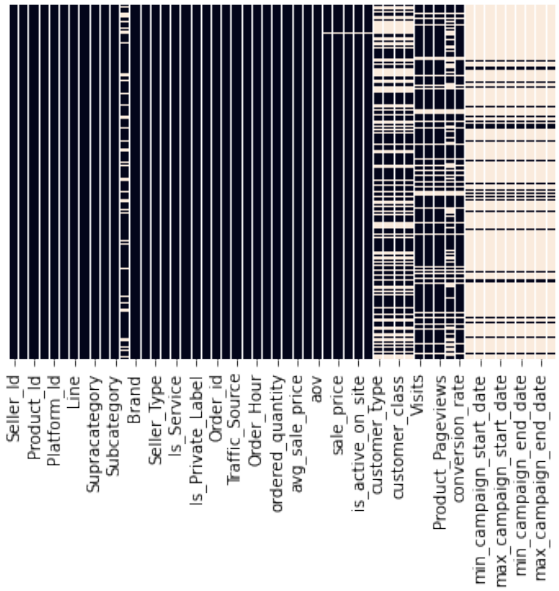
The accumulated data's analysis is on the following sections.

5.1.1 EDA & Feature Selection

dtypes: float64(16), int64(10), object(28)
memory usage: 157.0+ MB

Also, we can see that there are no duplicates.

Checking for missing values:



We can see from the heat map above that some columns have missing values (shaded in white). We will fill the missing values later in the pre-processing stage (“5.3 Preprocessing & Feature Engineering”).

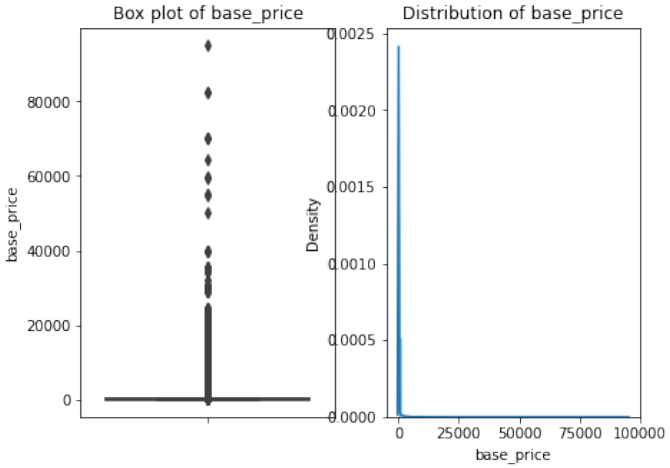
5.1.2 Individual Columns Analysis

Now, let's check for outliers.

- **Outliers in Base Price:**

```
In [ ]: analyze(df.base_price)
```

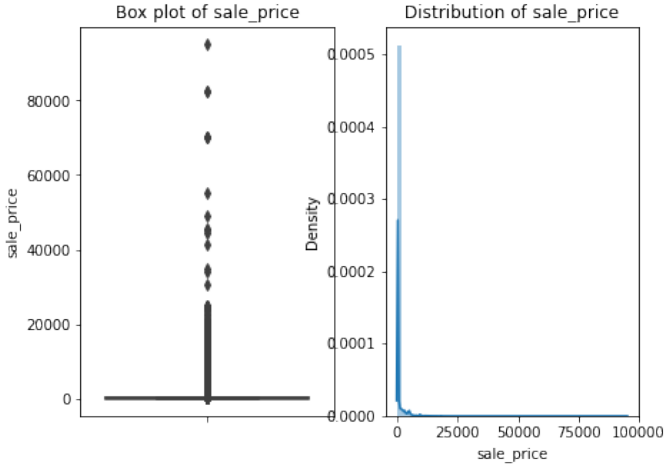
```
The number of outliers is: 39803
The mean value is: 283.77861862617914
The Median is: 29.41
The number of missing values is: 1458
The % of the missing values is: 0.38%
```



- **Outliers in Sale Price:**

```
In [ ]: analyze(df.sale_price)
```

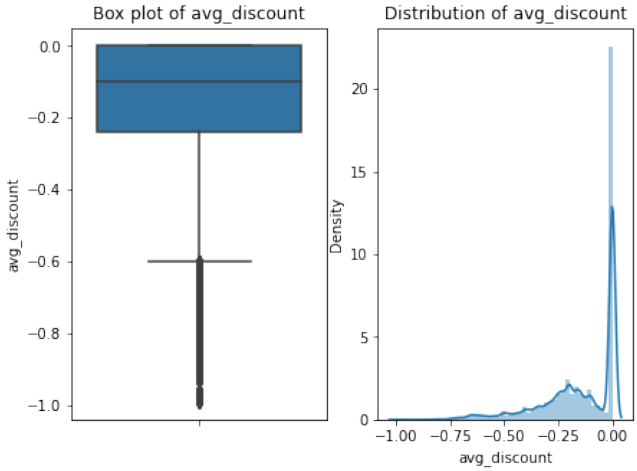
```
The number of outliers is: 38148
The mean value is: 190.62579482764735
The Median is: 25.0
The number of missing values is: 1458
The % of the missing values is: 0.38%
```



- **Outliers in Average Discount:**

```
In [ ]: analyze(df.avg_discount)
```

```
The number of outliers is: 12492
The mean value is: -0.14731612235553615
The Median is: -0.09991019308486748
The number of missing values is: 1465
The % of the missing values is: 0.38%
```



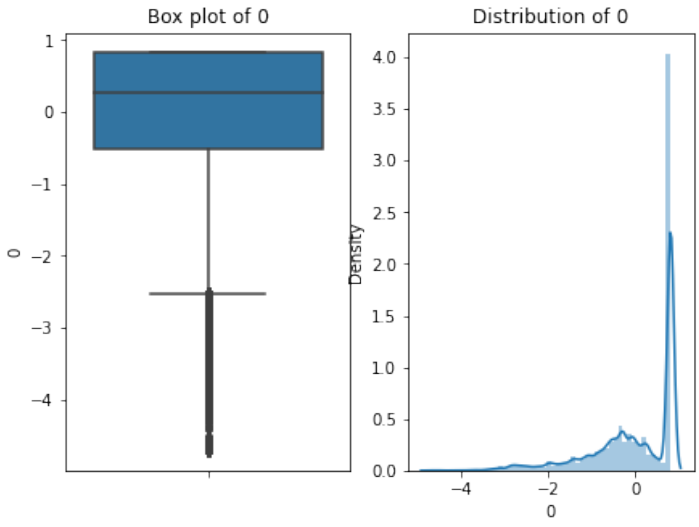
Here, it seems that we should do standardising. One of the problems of our datasets is that the majority of data in those cases do not have big discounts. As FMCGs are products who can be sold without great effort, extremely fast and in big scales discounts are more rare in comparison with slow-moving consumer goods. So, the vast majority of datasets, which contain data of the overall sales, have only a very small percentage of discounts.

So, average discount with standardizing:

```
In [ ]: # Standardizing a column

        sc = StandardScaler()
        avg_discount_std = sc.fit_transform(df[['avg_discount']])
        analyze(pd.DataFrame(avg_discount_std).iloc[:, 0])
```

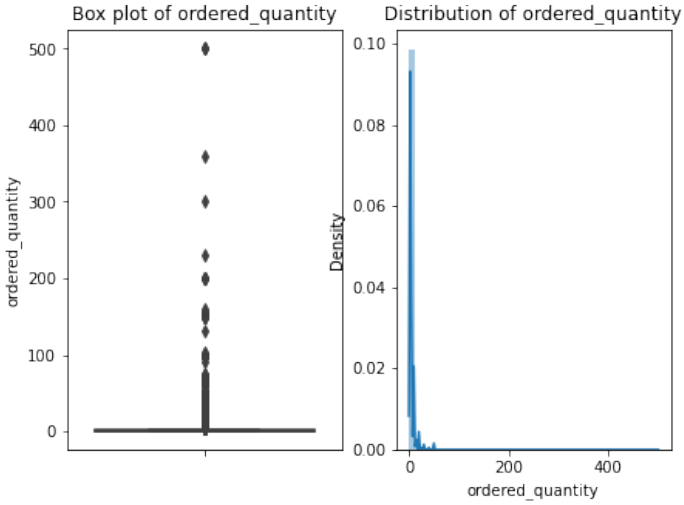
```
The number of outliers is: 12492
The mean value is: 1.7374504359880938e-14
The Median is: 0.2647511450819803
The number of missing values is: 1465
The % of the missing values is: 0.38%
```



- **Outliers in Ordered Quantity:**

```
In [ ]: analyze(df.ordered_quantity)
```

```
The number of outliers is: 47286
The mean value is: 2.252977150907903
The Median is: 1.0
The number of missing values is: 0
The % of the missing values is: 0.0%
```



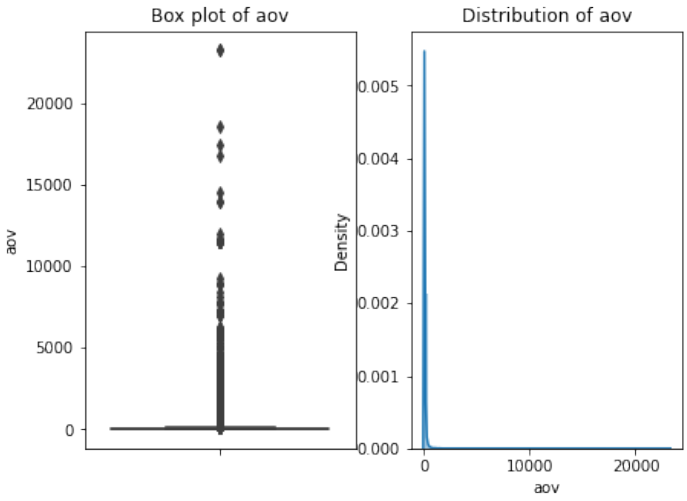
The outliers in the Ordered Quantity seem to be from the difference between B2C and B2B sales. While B2C customers order relatively small amounts of FMCGs (logical for a household) in B2B we see bulk purchases (outliers as they are huge ordered quantities).

We will omit these outliers (B2B customers) to be sure that we only take into account B2C customers/sales in our analysis.

- **Outliers in Average Order Value (AOV):**

```
In [ ]: analyze(df.aov)
```

```
The number of outliers is: 30343
The mean value is: 73.75259207804608
The Median is: 40.0
The number of missing values is: 0
The % of the missing values is: 0.0%
```



Again, the outliers in the Average Order Value (AOV) are the B2B sales. We will omit these outliers (B2B customers) to be sure that we only take into account B2C customers/sales in our analysis.

5.1.3 Categorical Features Analysis

In Chapter 2, we discussed about Categorical Features and how to handle them. Categorical features handling is extremely important in Sales Forecasting, as usually more than half of products' features are categorical.

Our data's overall features:

- Seller ID
- MKT ID
- Product ID
- Product Name
- Platform ID
- Country ID
- Line
- Division
- Supracategory
- Category
- SubCategory
- SubSubCategory
- Brand
- Product Status
- Order ID
- Order Source
- Traffic Source
- Order Date
- Order Hour
- No of Orders
- Ordered Quantity
- No of Products
- Avg Sale Price
- Ordered Value
- AOV
- Base Price
- Sale Price
- Avg Discount
- Is Active
- Is Visible on Site
- Customer Type
- Client Type
- Customer Class
- Order Type
- Visits
- Pageviews
- Product Pageviews
- Conversion Rate
- No of Campaigns
- Min Campaign Start Date
- Days since min Campaign Start
- Max Campaign Start Date
- Days since max Campaign start
- Min Campaign End date
- Days since min Campaign End
- Max Campaign End Date
- Days since max Campaign end
- Base Price STD
- Sales Price STD
- AVG Discount STD
- Week

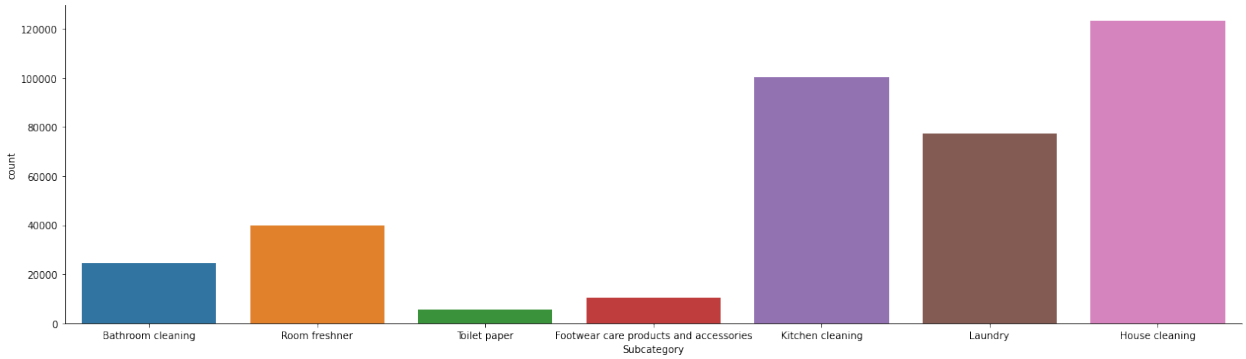
We see that 28 out of our 54 total features are categorical (>50%) as expected. This will help us to benchmark our experiments and models with publications on Sales Forecasting as we will use the same (or similar) features to the ones they use for Sales Forecasting for FMCGs.

Further exploring our Data

In Chapter 1 ("Introduction"), we explained that from all the FMCGs product categories we chose to work on Detergents & Cleaners for the following reasons:

1. Although there is a lot of research on Sales Forecasting for the fashion industry there is nearly zero research on Sales Forecasting for Detergents & Cleaners.
2. The Detergents & Cleaners industry is a very fast growing one with a CAGR of more than 4% YoY.
3. They are a heavily price-sensitive industry which gives even more importance to pricing strategies and accurate Sales Forecasting to maximise revenues and profits.
4. They can efficiently represent the entirety of the fast-moving consumer goods area.

So, let's dive even further by taking a look at the exact Subcategories of Detergents & Cleaners we have at hand:

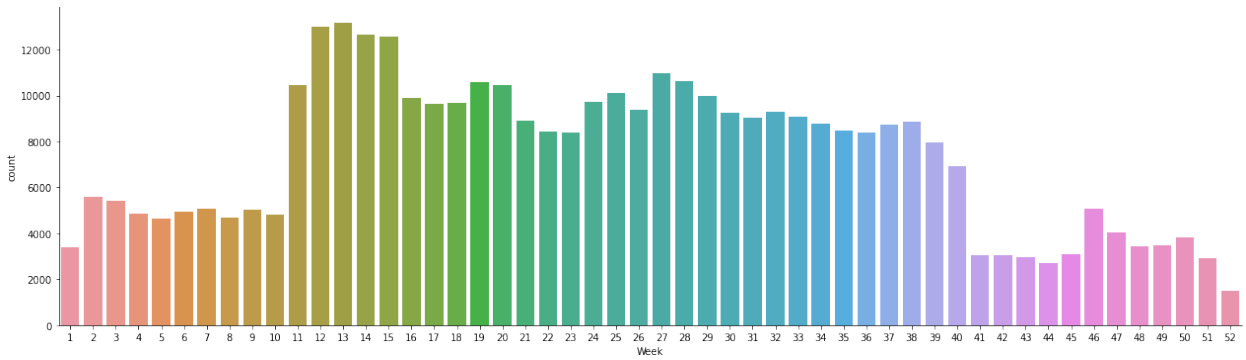


And, in detail, we have the following amount of products in each Subcategory:

- House cleaning 123405
- Kitchen cleaning 100276
- Laundry 77196
- Room freshner 39751
- Bathroom cleaning 24557
- Footwear care products and accessories 10270
- Toilet paper 5698

Also, apart from having enough amount of data from many different Detergents & Cleaners subcategories (to have as little bias as possible in choosing the data to work with), we need to have enough data from many different weeks to avoid cases of bias from extreme seasonality.

The amount of products sold per week is:



As we have, no less than 2000 products sold per week, we seem to have enough data from all different weeks/periods. Although, we can easily detect a seasonality pattern in Detergents & Cleaners during the spring and summer, they are products which make sales all year long, which is an additional reason why they are the optimal choice for studying FMCGs overall.

Last but not least, one of the most important aspects of modern Sales Forecasting is order source and traffic source. As we study marketplaces, they have a number of different sales and distribution channels and we want to study the correlation between order/traffic source and actual

sales. This analysis is becoming increasingly important to businesses' marketing departments and in recent marketing research, especially regarding targeting.
In our data, more than 99% of our order source comes form:

```
In [ ]: df.Order_source.value_counts()

Out[ ]: MOBILE APP                143004
        SITE                      135772
        MOBILE                     99964
```

So, the 3 top order sources are the marketplaces':

- Mobile app (Android & iOS)
- Website (desktop/laptop view)
- Website (mobile view)

Also, more than 90% of our traffic source comes form:

```
In [ ]: df.Traffic_Source.value_counts()

Out[ ]: Direct Pure               150554
        SEO                       111930
        Direct Unknown             56522
        AdWords – PLA              14718
        AdWords – Search           11058
        Trigger                     7024
        Facebook Paid               4137
```

So, the 4 top order sources are the marketplaces':

- Direct (website & app)
- SEO
- AdWords
- Facebook Ad

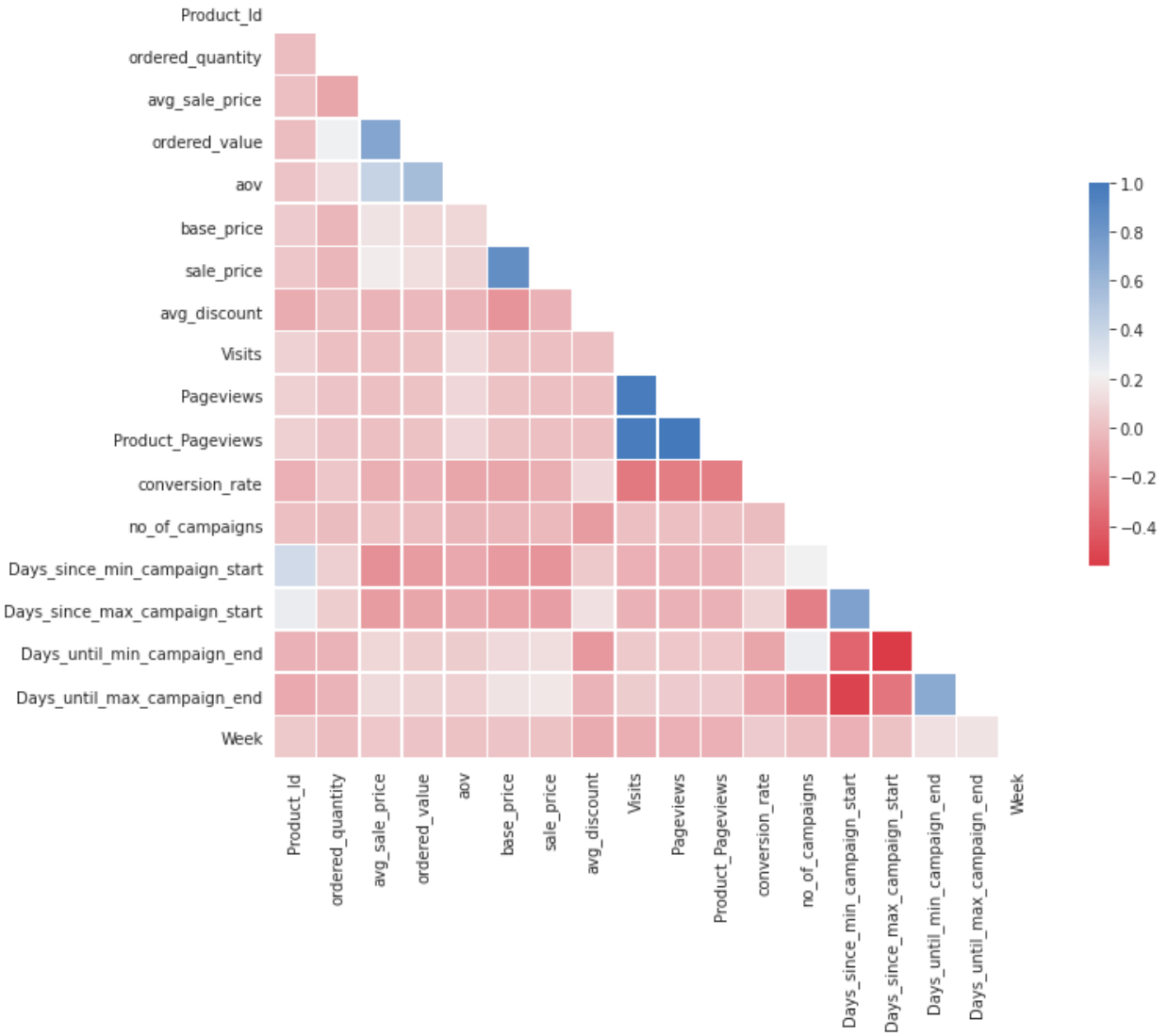
5.1.4 Bivariate Analysis/Pearson's correlation

The Pearson product-moment correlation coefficient (or Pearson correlation coefficient, for short) is a measure of the strength of a linear association between two variables and is denoted by r. Basically, a Pearson product-moment correlation attempts to draw a line of best fit through the data of two variables, and the Pearson correlation coefficient, r, indicates how far away all these data points are to this line of best fit (i.e., how well the data points fit this new model/line of best fit).

It can take a range of values from +1 to -1. A value of 0 indicates that there is no association between the two variables. A value greater than 0 indicates a positive association; that is, as the value of one variable increases, so does the value of the other variable. A value less than 0 indicates a negative association; that is, as the value of one variable increases, the value of the other variable decreases. The stronger the association of the two variables, the closer the Pearson correlation coefficient, r, will be to either +1 or -1 depending on whether the relationship is positive or negative, respectively. Achieving a value of +1 or -1 means that all your data points

are included on the line of best fit – there are no data points that show any variation away from this line. Values for r between +1 and -1 indicate that there is variation around the line of best fit. The closer the value of r to 0 the greater the variation around the line of best fit.

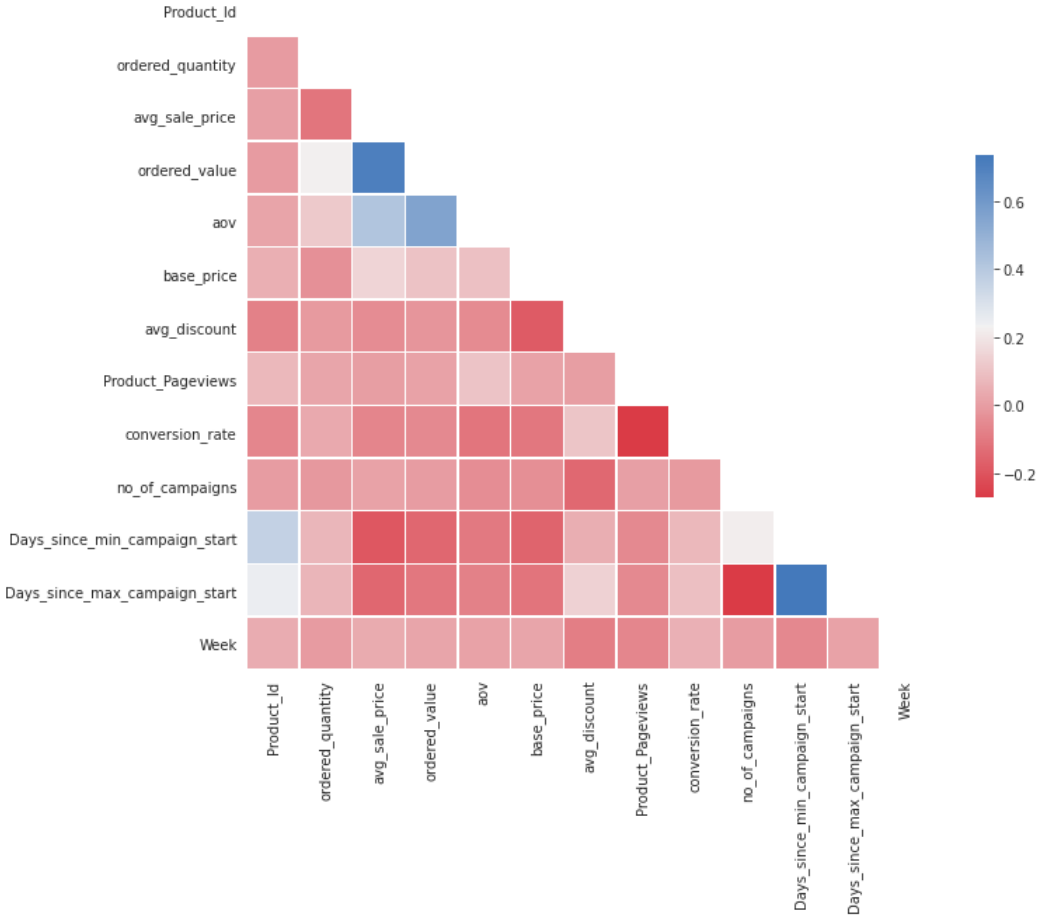
Our data:



From the matrix above we can make the following conclusions:

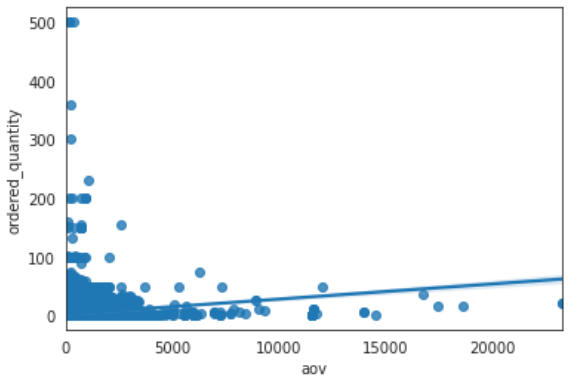
- We see that Visits and Pageviews, Visits and Product_Pageviews, and Pageviews and Product_Pageviews are closely correlated. Therefore, we can remove Visits and Page_Views columns, keeping Product_Pageviews.
- We can also see that base_price and sale_price are correlated, and therefore we can remove one of them.
- We notice that Days_since_min_campaign_start and Days_until_max_campaign_end are strongly negatively correlated and, therefore, we can also remove one of them (we will remove the one that is less correlated to the target feature). The same applies to Days_since_max_campaign_start and Days_until_min_campaign_end.

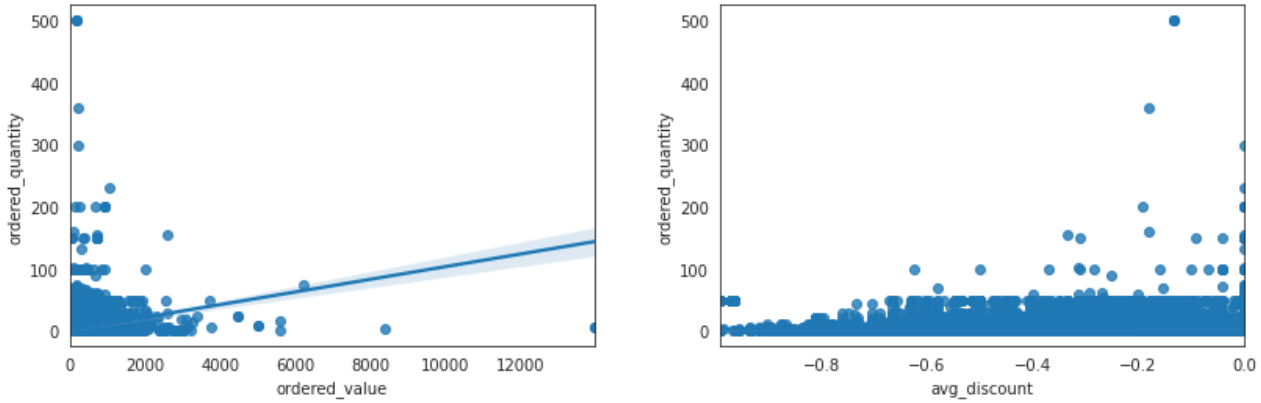
From removing the one that is less correlated to the target feature (ordered_quantity) we have the final Pearson's correlation:



5.1.5 Target Value & Input Features

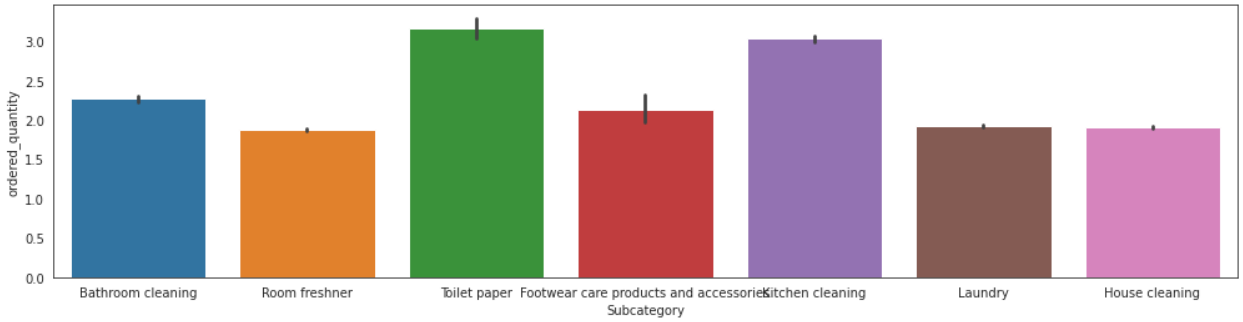
Just to complete our data analysis, we provide further visualisations between our target value (ordered_quantity) and some of the most important Sales Forecasting features (based on other publications and our Bi-Variate Analysis).





Our data have the characteristics we expected. As we work with FMCGs we have small average price per product and small to medium order sizes (0-100). Excluding the outliers, the vast amount of data are for products that:

- Have a base price of 0-20€.
- Sold in small to medium quantities (order size).
- The buyers are individuals and maybe small businesses that buy retail.
- We have a diversity of discounts (average discount per product id), but the majority of campaigns are based on small discounts (<50%).
- Customers don't buy just one product, which means that by the time they send their order they have many items in their cart, as it usually happens at marketplaces and especially for FMCGs.



So, once again we see that our data are an excellent representation of FMCGs' sales and, thus, can be effectively used for Sales Forecasting for nearly every FMCG product.

5.2 Key Metrics

To better benchmark our models with the Bibliography's research and results we need to chose similar key metrics. Also, we need to choose key metrics that can be used in a variety of models that we used in our experiments in order to be able to accurately compare them. Thus, the key metrics that we used are: 1. MAPE, 2. RMSE, 3. MSE

MAPE, RMSE & MSE are some of the most used key metrics in Sales Forecasting.

Evaluation indexes	Expression	Description
ME	$ME = 1/(b-a+1)\sum_{k=a^b}(y_k-\hat{y}_k)$	The mean sum error
MSE	$MSE = 1/(b-a+1)\sum_{k=a}^{b}(y_k-\hat{y}_k)^2$	The mean squared error
RMSE	$RMSE = \sqrt{1/(b-a+1)\sum_{k=a}^{b}(y_k-\hat{y}_k)^2}$	The root mean squared error
MAE	$MAE = 1/(b-a+1)\sum_{k=a}^{b}\lvert y_k-\hat{y}_k\rvert$	The mean absolute error

y_k is the sales of the k-th sample. $\hat{y}_k$ denotes the corresponding prediction.

And because mean_squared_error is a standard function in sklearn we need to define the two others:

```
In [ ]: from sklearn.metrics import mean_squared_error

def mean_absolute_percentage_error(y_t, y_p):
    return np.mean(np.abs((y_t - y_p) / y_t)) * 100

def root_mean_squared_error(y_t, y_p):
  mse = mean_squared_error(y_t, y_p)
  return np.sqrt(mse)
```

So, now we have ready to use for every one of our model our key metrics, which are called with the following names:

- MAPE: mean_absolute_percentage_error
- MSE: mean_squared_error
- RMSE: root_mean_squared_error

5.3 Preprocessing & Feature Engineering

Before we start building the ML models with target value the ordered_quantity, we have to do some pre-processing. After that, we will try to transform existing features, and construct new features to improve the performance of the model.

The data pre-processing we have done includes:

- Handling missing values
- Data encoding
- Removing outliers
- Feature Scaling

For more information about the exact Preprocessing & Feature Engineering we conducted, please ask for our relevant notebook (.ipynb).

Regarding Feature Scaling, we decided to do double work and double experiments, both with Standardized and Normalized data. This helped us take everything into account and guarantee that the results of our experiments will be unbiased from the Feature Scaling technique/case that will be used from businesses' data departments. Also, regarding Sales Forecasting it is not clear which feature scaling technique works best, as we have studied top publications that use either of these methods. However, there may be a simpler answer to this, which is that Sales Forecasting is unbiased from the feature scaling method (due to the specific characteristics of the features that are being used for Sales Forecasting). However, in any case, it seems absolutely necessary to use a feature scaling technique.

According to our results, the two feature scaling methods had a comparable behaviour, pointing towards the assumption that Sales Forecasting problems are unbiased to the choice of feature scaling. Nevertheless, we give some details about both Standardisation and Normalisation for completeness in our analysis and theoretical background of necessary data science and feature engineering for ML in Sales Forecasting.

5.3.1 Standardisation

Standardisation (or z-score normalisation) is the procedure during which the feature values are rescaled so that they have the properties of a standard normal distribution with μ = 0 and σ = 1, where μ is the mean (the average value of the feature, averaged over all examples in the dataset) and σ is the standard deviation from the mean. Standard scores (or z-scores) of features are calculated as follows:

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}}$$

In [ ]:
```
# 1. Standardisation
sc = StandardScaler()
enc_df_std = enc_df.copy()
enc_df_std[columns_to_scale] = sc.fit_transform(enc_df_std[columns_to_scale])
enc_df_std
```

Out[ ]:

	Product_Id	Subcategory	SubSubcategory	Brand	Product_Status	Seller_Type	Is_Accessory	Order_source	Traffic_Source	ordered_quantity	avg_sale_
0	45392776	0	20.000000	2495.0	0	1	0	5.0	7.0	1	-0.45
1	45495458	0	20.000000	2222.0	0	1	0	5.0	7.0	1	-0.21
2	56790176	5	11.137896	2357.0	0	1	0	25.0	2.0	1	-0.14
3	26095097	6	11.137896	2357.0	0	1	0	25.0	2.0	1	1.95
4	41344737	5	11.137896	1433.0	0	1	0	4.0	20.0	1	0.34
...	...	...	...	...	...	...	...	...	...	...	...
381148	40923336	0	0.000000	1195.0	0	1	0	5.0	7.0	5	-0.42
381149	17174689	2	10.000000	2222.0	0	1	0	5.0	20.0	1	-0.24
381150	17174688	2	10.000000	2222.0	0	1	0	5.0	20.0	1	-0.07
381151	16643891	2	10.000000	1296.0	0	1	0	5.0	7.0	1	-0.15
381152	31145066	2	10.000000	977.0	0	1	0	4.0	20.0	3	-0.57

381153 rows × 27 columns

5.3.2 Normalisation

Normalisation is the process of converting an actual range of values which a numerical feature can take, into a standard range of values, typically in the interval [−1, 1] or [0, 1]. For example, suppose the natural range of a particular feature is 120 to 2520. By subtracting 120 from every value of the feature, and dividing the result by 2400, one can normalise those values into the range [0, 1]. More generally, the normalisation formula looks like this:

$$\bar{x}^{(j)} = \frac{x^{(j)} - min^{(j)}}{max^{(j)} - min^{(j)}}$$

Normalising the data is not a strict requirement. However, in practice, it can lead to an increased speed of learning. Additionally, it's useful to ensure that our inputs are roughly in the same relatively small range to avoid problems which computers have when working with very small or very big numbers (overflow).

In [ ]:
```
# 2. Normalisation
norm = Normalizer()
enc_df_norm = enc_df.copy()
enc_df_norm[columns_to_scale] = norm.fit_transform(enc_df_norm[columns_to_scale])
enc_df_norm
```

Out[ ]:

	Product_Id	Subcategory	SubSubcategory	Brand	Product_Status	Seller_Type	Is_Accessory	Order_source	Traffic_Source	ordered_quantity	avg_sale_
0	45392776	0	20.000000	2495.0	0	1	0	5.0	7.0	1	0.05
1	45495458	0	20.000000	2222.0	0	1	0	5.0	7.0	1	0.12
2	56790176	5	11.137896	2357.0	0	1	0	25.0	2.0	1	0.14
3	26095097	6	11.137896	2357.0	0	1	0	25.0	2.0	1	0.41
4	41344737	5	11.137896	1433.0	0	1	0	4.0	20.0	1	0.37
...	...	...	...	...	...	...	...	...	...	...	...
381148	40923336	0	0.000000	1195.0	0	1	0	5.0	7.0	5	0.06
381149	17174689	2	10.000000	2222.0	0	1	0	5.0	20.0	1	0.11
381150	17174688	2	10.000000	2222.0	0	1	0	5.0	20.0	1	0.16
381151	16643891	2	10.000000	1296.0	0	1	0	5.0	7.0	1	0.13
381152	31145066	2	10.000000	977.0	0	1	0	4.0	20.0	3	0.02

381153 rows × 27 columns

5.4 Initial model training & Results

We do train-test splitting and now we are ready to run our models (which were analysed in detail in Chapter 4, “Method & Models”).

In order to be able to compare our models (both between each other, and with other publications’ results) we will review them into 4 categories:

- (Simple) Regression Models
- Ensemble models/GBDT
- Meta-Learners
- Deep Learning models/Neural Networks

An additional reason we will do that is that by having 15 models (9 ML, 3 meta-learners, 3 deep learning models) in total we face the problem of having difficult visual comparisons. So, the only way to compare our models is by benchmarking them with other models from their “algorithmic family”.

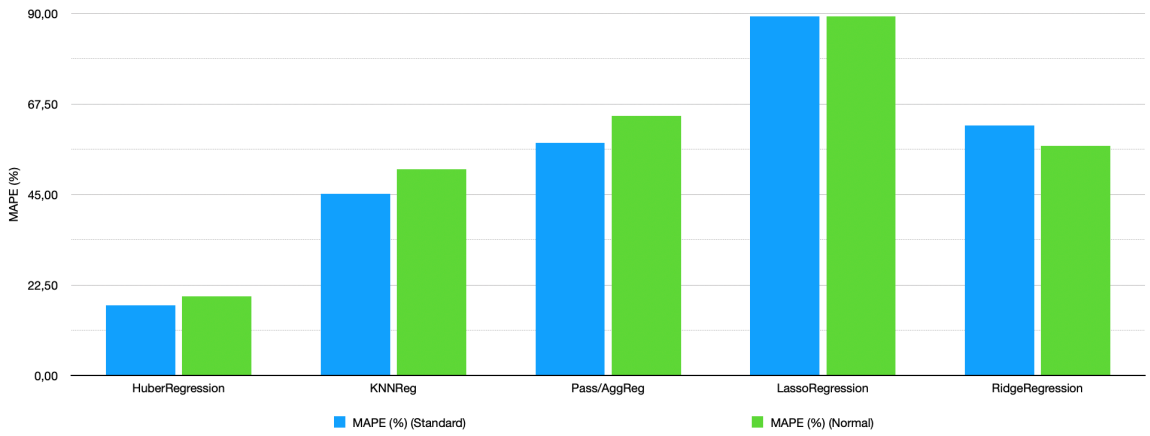
5.4.1 Regression models

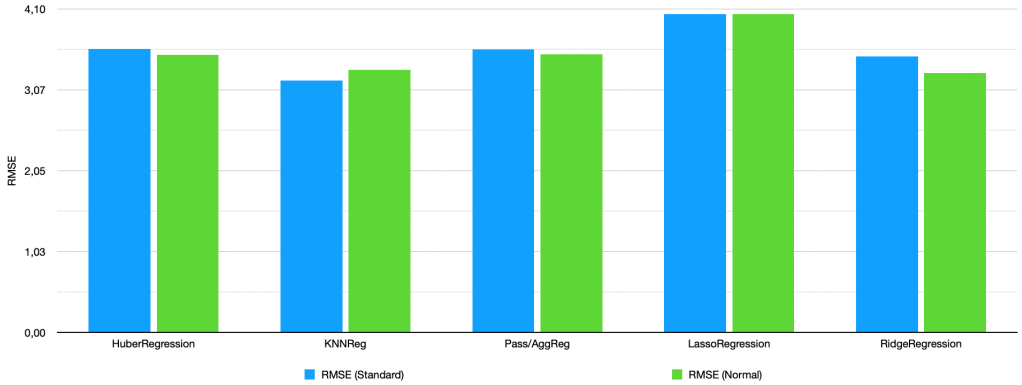
First, we use the standard regression models. Relative results and further use of them can be found in publications [2], [4], [17] and [46] in bibliography.

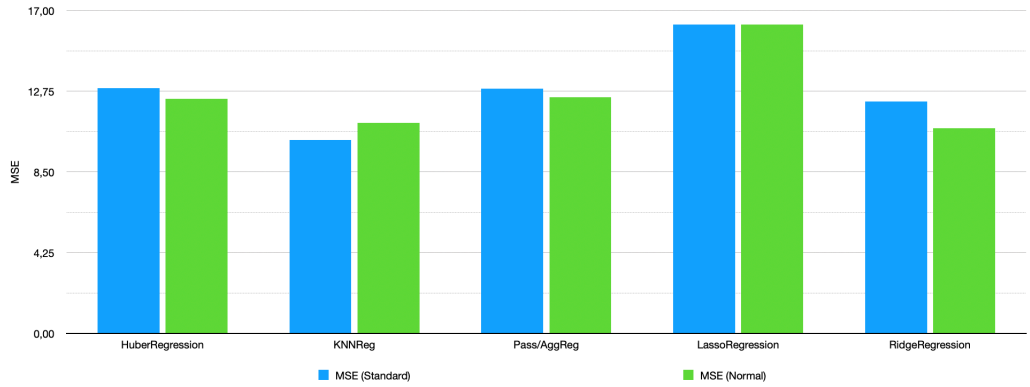
The models we cover are:

- HuberRegression
- KNN Regressor
- Passive Aggressive Regressor
- Lasso Regression
- Ridge Regression

	MAPE (%) (Standard)	MAPE (%) (Normal)	RMSE (Standard)	RMSE (Normal)	MSE (Standard)	MSE (Normal)	Execution Time (sec) (Standard)	Execution Time (sec) (Normal)
HuberRegression	17,41	19,65	3,60	3,52	12,93	12,36	895	1512
KNNReg	45,21	51,24	3,19	3,33	10,19	11,10	19542	19722
Pass/AggReg	57,82	64,59	3,59	3,53	12,89	12,43	7	10
LassoRegression	89,26	89,26	4,03	4,03	16,26	16,26	1	1
RidgeRegression	62,12	57,16	3,50	3,29	12,22	10,81	1	1



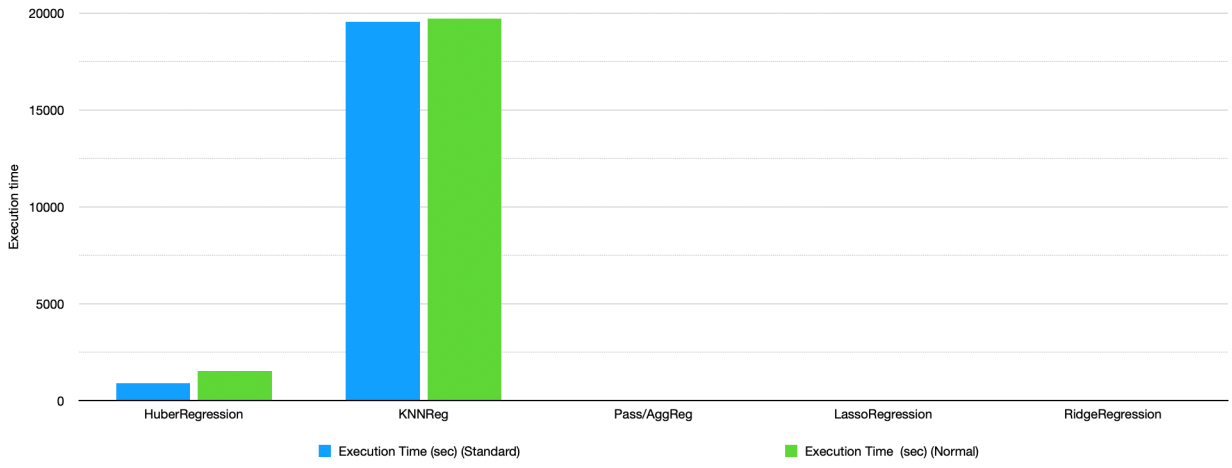




We have the following results:

As we see from the barplots above:

- There are no consistent differences between our standardized dataset and our normalised one. Some models achieve better accuracy for one of them, but even in this case, we see that across different key metrics, nearly all models change the preferred feature scaling method. So, as we have predicted, for our FMCGs' dataset, the choice of feature scaling seems to be irrelevant.
- HuberRegression has by far the lowest MAPE, but loses in other key metrics.
- Based on industry standards, MAPE >20% is not good enough for FMCGs. So, from simple



regressors, only HuberRegression passes the acceptable limit.

- Nowadays, all those models are run online (cloud-based services). So, the absolute value of running/execution time is insignificant. What maybe matters is the relative execution time to get a good sense about each model's need for computing resources and time for execution.

- KNN takes an extreme amount of time to be executed.
- HuberRegression has about 20-30 minutes execution time.
- All the other models have practically zero execution time.

5.4.2 GBDT models

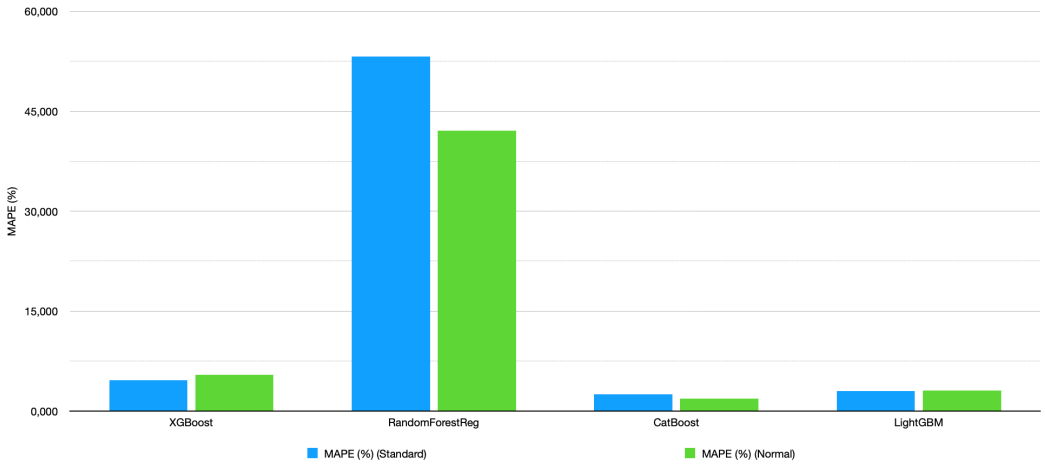
Then, we need to check the experimental results from our GBDT models. Relative results and further use of them can be found in publications [1], [3], [5], [6], [8], [10], [12], [22], [23], [36], [37] and [42] in bibliography.

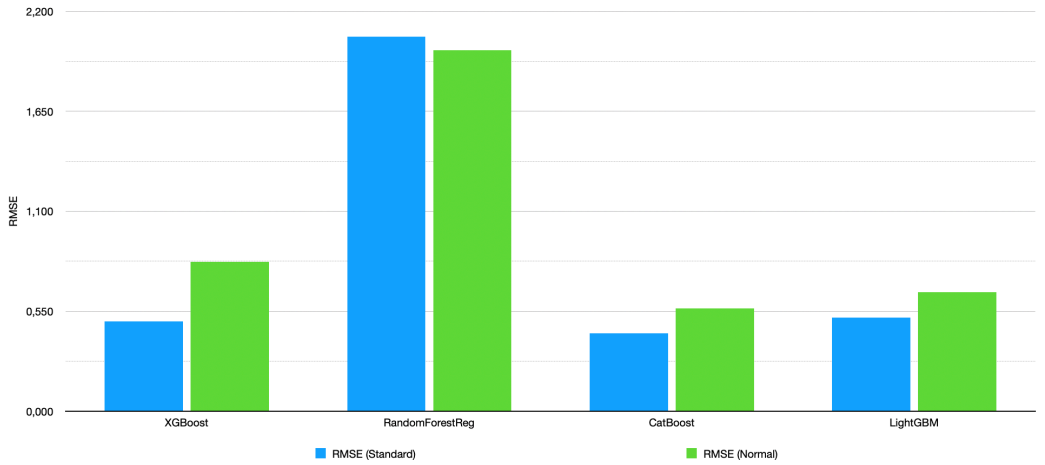
The models we cover are:

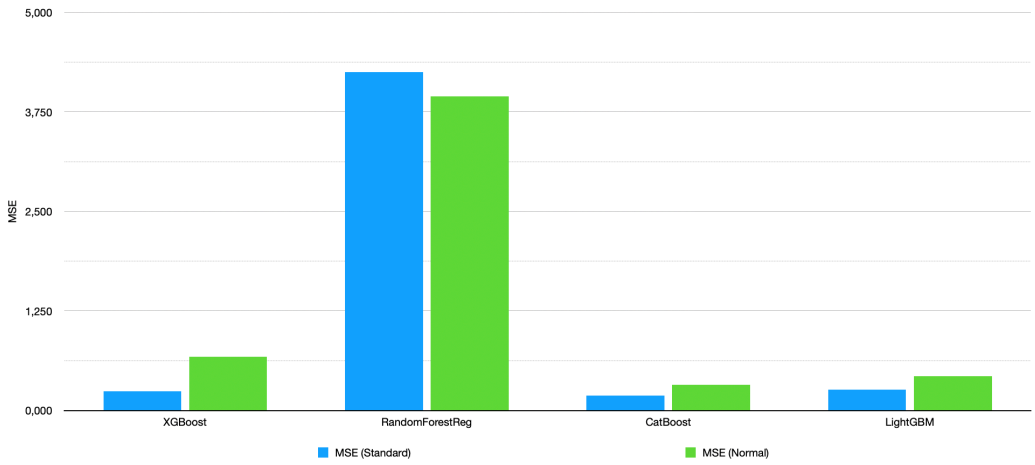
- XGBoost
- Random Forest Regressor
- CatBoost
- LightGBM

We have the following results:

	MAPE (%) (Standard)	MAPE (%) (Normal)	RMSE (Standard)	RMSE (Normal)	MSE (Standard)	MSE (Normal)	Execution Time (sec) (Standard)	Execution Time (sec) (Normal)
XGBoost	4,580	5,454	0,494	0,822	0,244	0,676	1634	1792
RandomForestReg	53,246	42,092	2,061	1,986	4,248	3,945	727	727
CatBoost	2,486	1,816	0,429	0,566	0,184	0,321	133	178
LightGBM	2,981	3,075	0,514	0,655	0,264	0,429	17	15

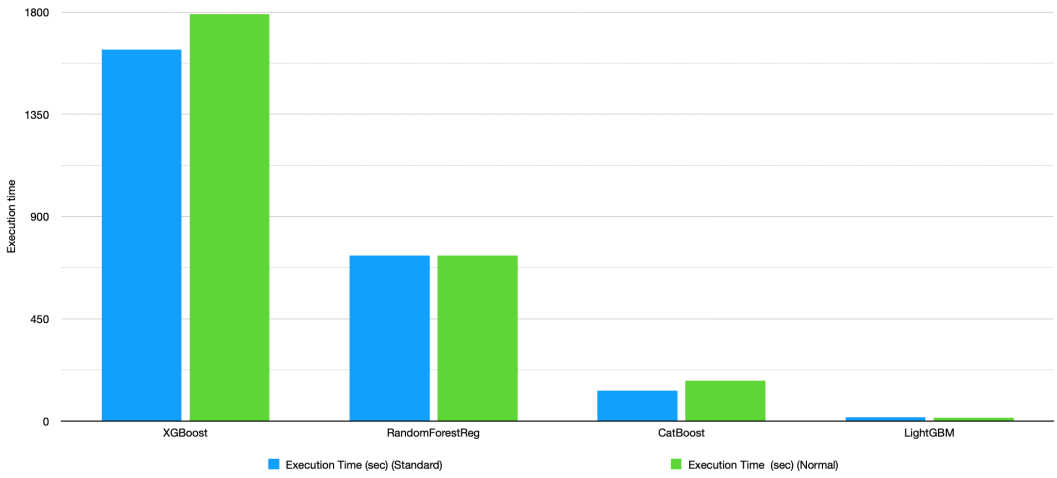






As we see from the barplots above:

- The standardized dataset is doing a bit better than the normalised one in GBDT models. Especially in XGBoost, the difference is significant. So, XGBoost has a preference for standardisation.
- XGBoost, CatBoost and LightGBM do extremely well across all metrics. With a MAPE of just 2.5-5% GBDT models show why they are considered the best models for problems with heterogeneous tabular data structures.
- RandomForest Regressor is the only GBDT model with low prediction accuracy. While it is very simple to use and potentially good for benchmarking, it doesn't seem appropriate for FMCGs Sales Forecasting.
- Based on industry standards, XGBoost, CatBoost and LightGBM are great models for Sales Forecasting, especially for FMCGs Sales Forecasting.
- We see that GBDT models are extremely consistent across all key metrics, meaning that the model ranking is the same across all the different key metrics we chose.



- XGBoost takes a bigger amount of time to be executed (~27-30min to be executed).
- CatBoost and LightGBM give very good results and have a low execution time too.

5.4.3 Deep Learning models

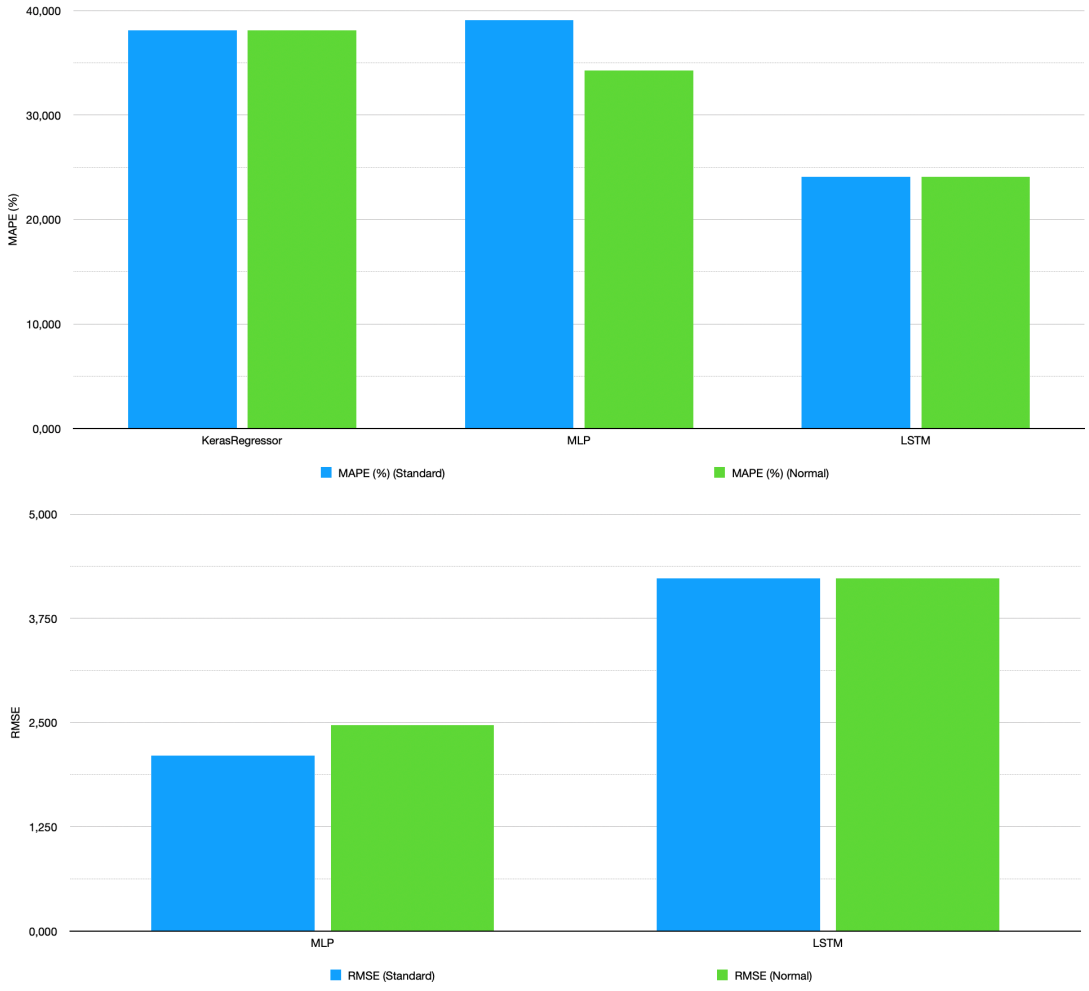
Let's take a look into our Deep Learning models. Relative results and further use of them can be found in publications [31], [33], [34], [36], [38], [40] and [43] in bibliography.

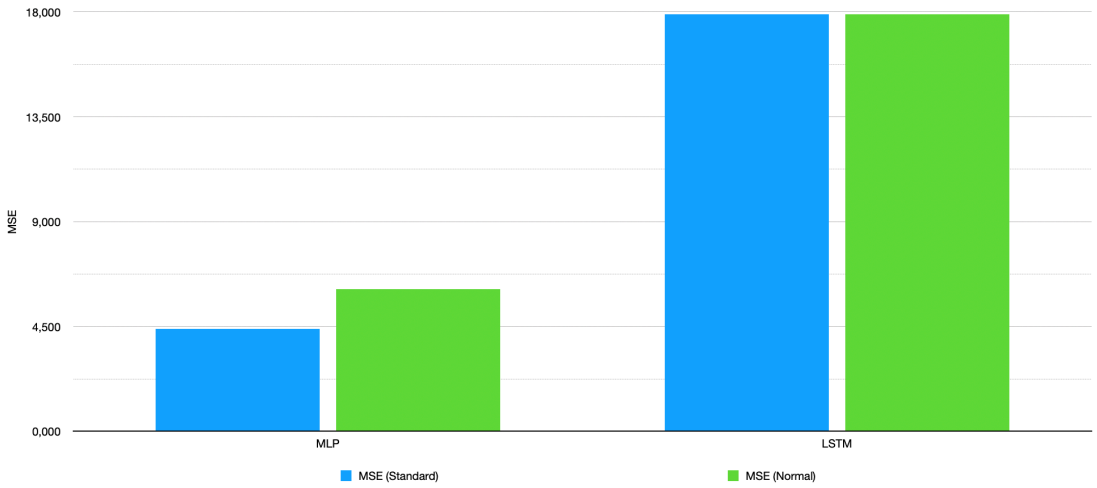
The models we cover are:

- KerasRegressor
- MLP
- LSTM

It is also important to point out that as the optimization of our Deep Learning models is gradient based, the standard label encoding we have done for other models can be problematic. This is a standard data analysis problem, and as we explained in Chapter 2 ("Theoretical Background") it happens because the numeric values can be misinterpreted by the deep learning algorithms as having some sort of hierarchy or order in them. For that reason, as we established earlier, we used one-hot encoding to troubleshoot this potential issue.

	MAPE (%) (Standard)	MAPE (%) (Normal)	RMSE (Standard)	RMSE (Normal)	MSE (Standard)	MSE (Normal)	Execution Time (sec) (Standard)	Execution Time (sec) (Normal)
KerasRegressor	38,093	38,093					628	653
MLP	39,073	34,273	2,100	2,470	4,408	6,103	2011	1772
LSTM	24,091	24,090	4,231	4,231	17,899	17,899	818	847



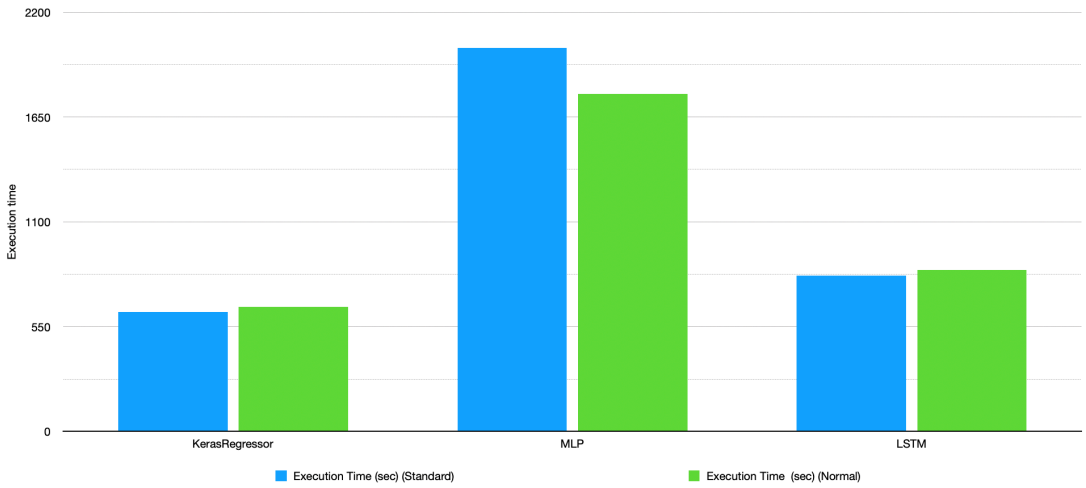


We have the following results:

We only measured MAPE for KerasRegressor to just use it as a DL benchmark.

As we see from the barplots above:

- There are no consistent differences between our standardized dataset and our normalised one. Some models achieve better accuracy for one of them, but even in this case, we see that across different key metrics, nearly all models change the preferred feature scaling method. So, as we have predicted, for our FMCGs' dataset, the choice of feature scaling seems to be irrelevant.
- While MLP has lower RMSE and lower MSE, it has bigger MAPE than LSTM.
- Based on industry standards all of the aforementioned networks could be used for FMCGs Sales Forecasting, but, as expected, the LSTM is clearly the most favorable.
- In comparison to simple regressors, Deep Learning models give better results, but not



necessarily better results from GBDT models as they quickly overfit.

- As we expected, all Deep Learning models have a significant execution time. However, as it happens in those cases, businesses use extra computational resources.

- MLP takes a bigger amount of time to be executed (~30-34min to be executed).

5.5 Ensembling/Meta-Learning

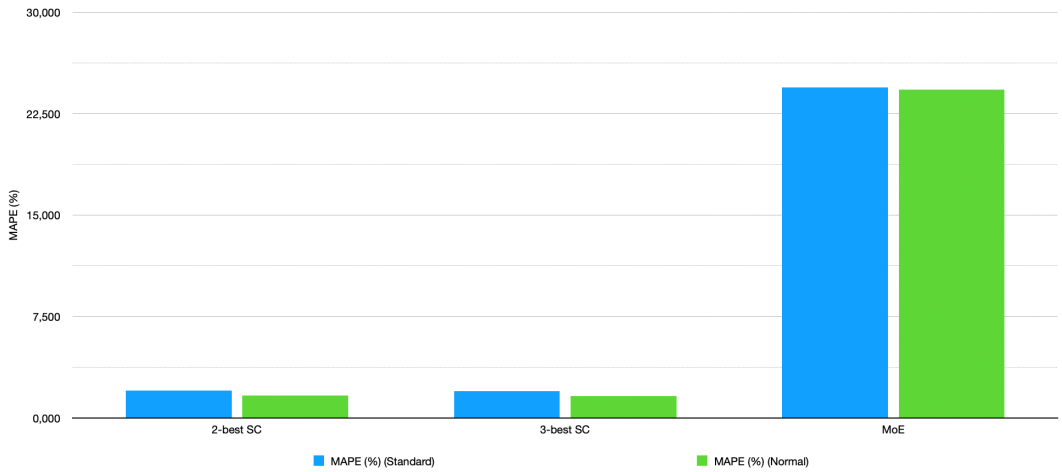
Finally, we use the meta-learners we discussed about. As we discussed in Chapter 2 (“Theoretical Background”), Chapter 3 (“Previous Work”) and Chapter 4 (“Method and Models”), Ensemble learning plays a huge part in increasing simple models accuracy. For that reason, after we run our 1st stage experiments with our simple regressors, GBDT models and Deep Learning models, we need to use and test the relative meta-learners and showcase our results.

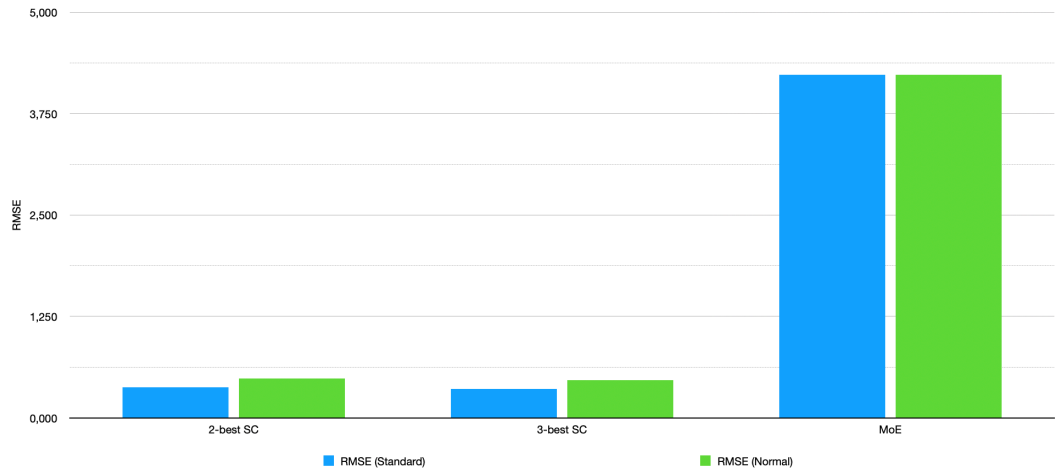
For that reason we use the following 3 meta-learners:

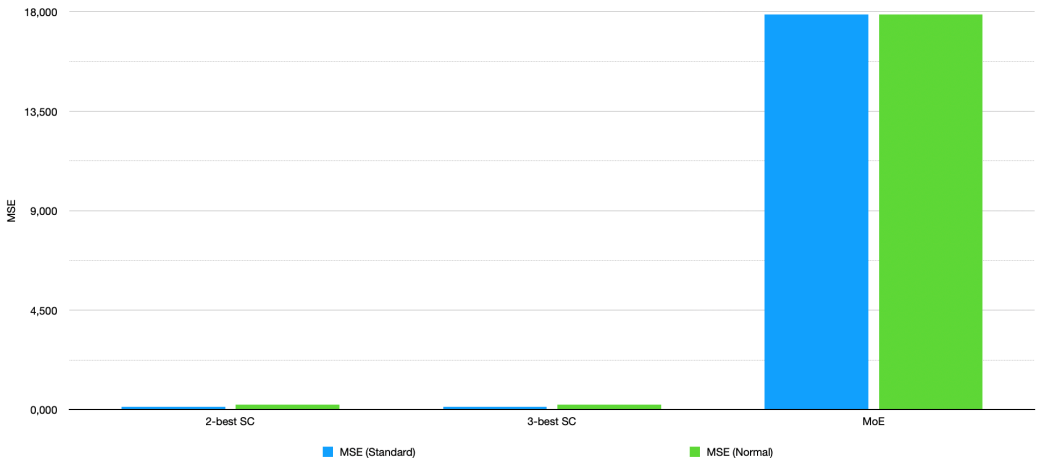
- 2-best Stacking Classifier
- 3-best Stacking Classifier
- Mixture of Experts (only for Deep Learning models)

And we have the following results:

	MAPE (%) (Standard)	MAPE (%) (Normal)	RMSE (Standard)	RMSE (Normal)	MSE (Standard)	MSE (Normal)	Execution Time (sec) (Standard)	Execution Time (sec) (Normal)
2-best SC	2,000	1,643	0,373	0,485	0,139	0,235	153	187
3-best SC	1,990	1,622	0,358	0,467	0,128	0,218	191	249
MoE	24,440	24,264	4,228	4,228	17,878	17,877	850	830

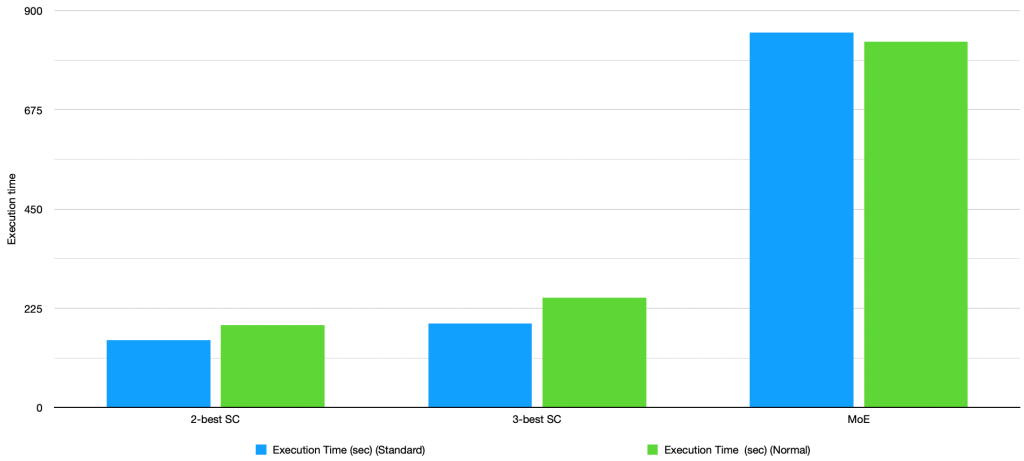






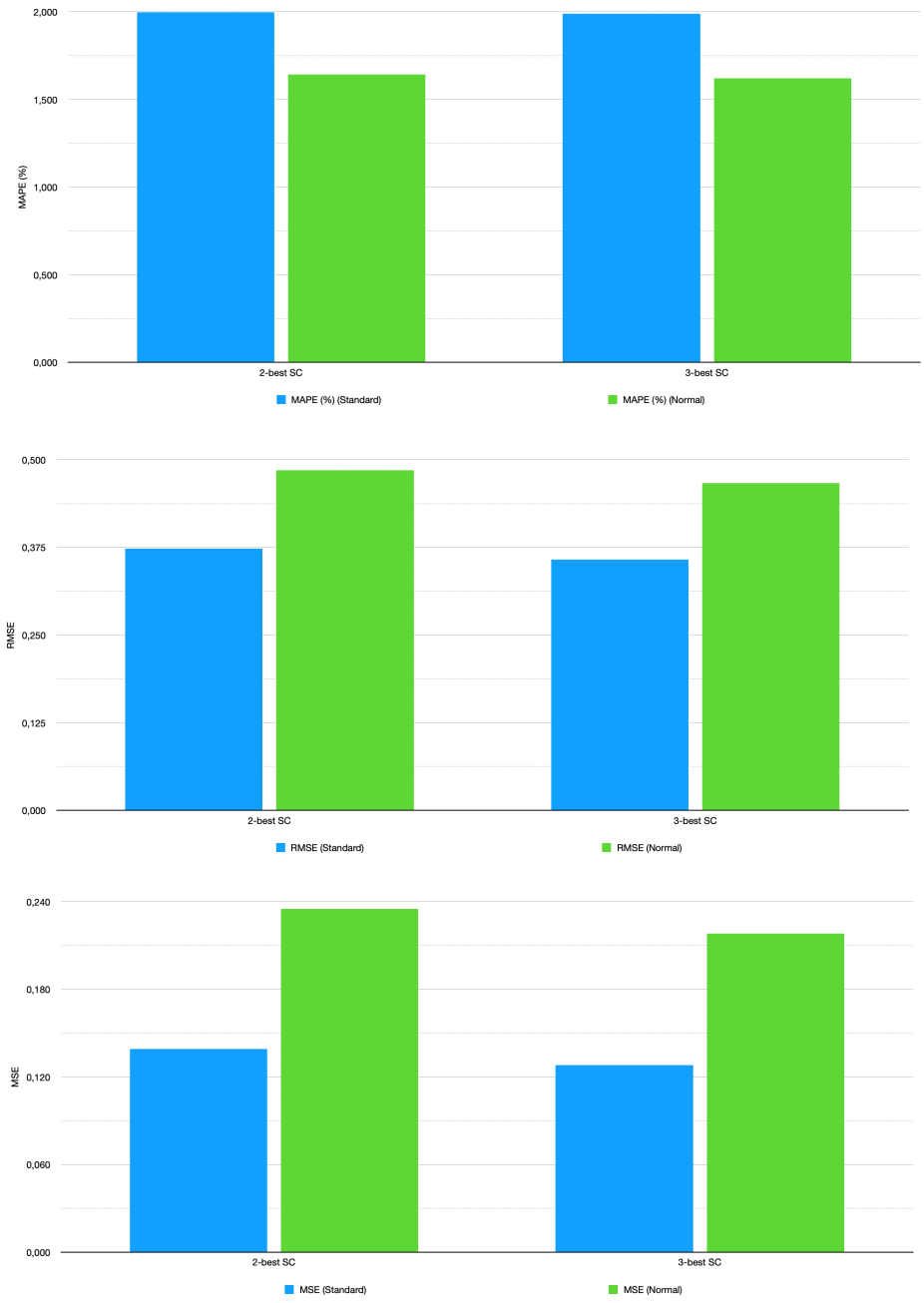
As we see from the barplots above:

- There are no consistent differences between our standardized dataset and our normalised one. Some models achieve better accuracy for one of them, but even in this case, we see that across different key metrics, nearly all models change the preferred feature scaling method. So, as we have predicted, for our FMCGs' dataset, the choice of feature scaling seems to be irrelevant.
- The Mixture of Experts (MoE) was about 1% better than LSTM across all key metrics for evaluating Deep Learning models.
- 2-best and 3-best Stacking Classifiers are very close across all key metrics.
- In comparison to our best model, our Stacking Classifier/Regressor gave better results by 25% and in comparison to our 3rd best model, our meta-learners gave better results by 130%.



- The 2-best Stacking Classifier, with our computational resources, takes about 2.5-3min to be executed.
- MoE takes a bigger amount of time to be executed (~14min).
- For the additional increase in prediction accuracy, Stacking Classifiers need little extra time and seem to be very good investments even when we regard the additional computational power.

To be able to better compare the two Stacking Classifiers and draw conclusions about which Stacking Classifier/Regressor is better for industry use (based on the increase in prediction accuracy, additional execution time and theoretical optimal forecasting limits), let's see them side by side:

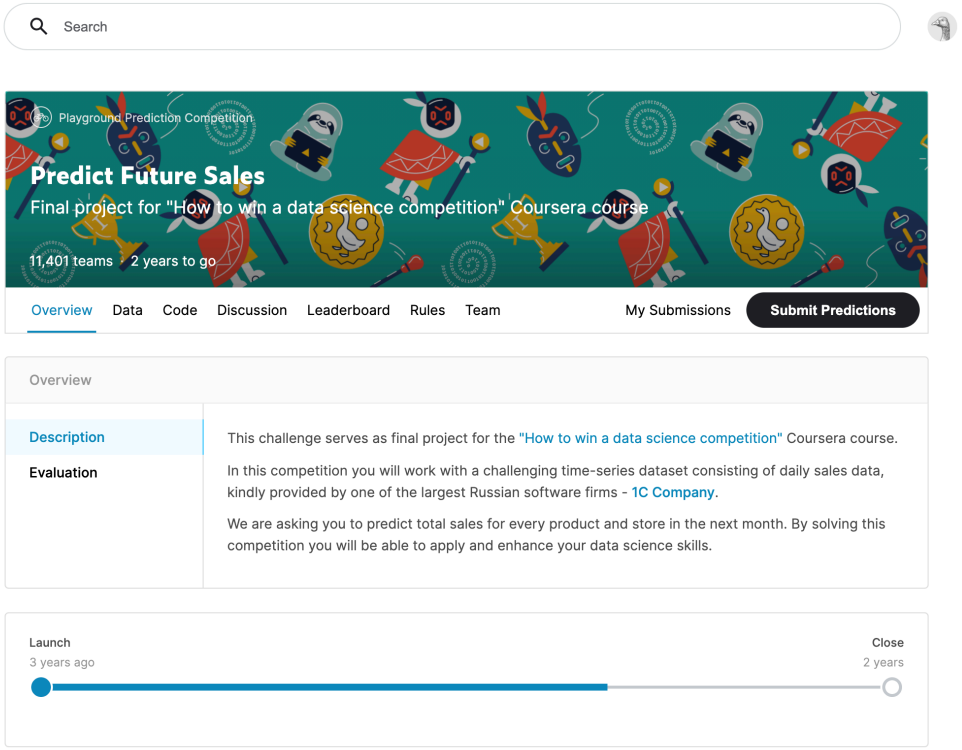


The 3-best Stacking Classifier is about 0-1% better than the 2-best Stacking Classifier across all key metrics.

In the next section, we will use the same method and models for a Kaggle competition. Along with this analysis, we will present our final experimental results and conclusions in Chapter 6 ("Conclusion").

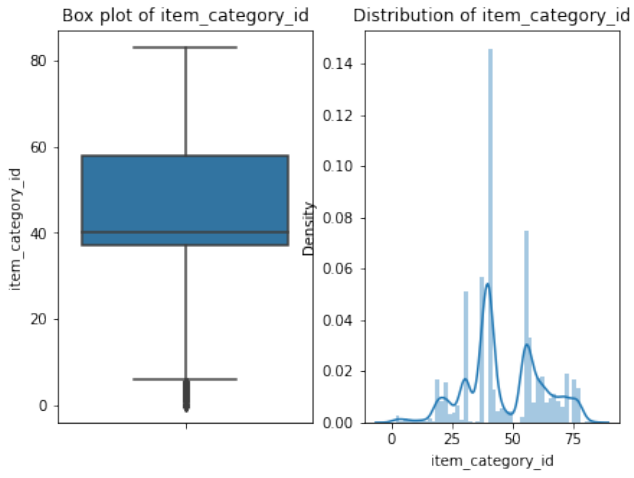
5.6 Benchmarking with Kaggle competition

In order to have even more data and results about the meta-learners we tried, we decided to compete in a Kaggle competition about Sales Forecasting. It was also a good ground to test if the models that proved to be good top choices for FMCGs Sales Forecasting are also top choices for other types of products and more diverse Sales Forecasting.

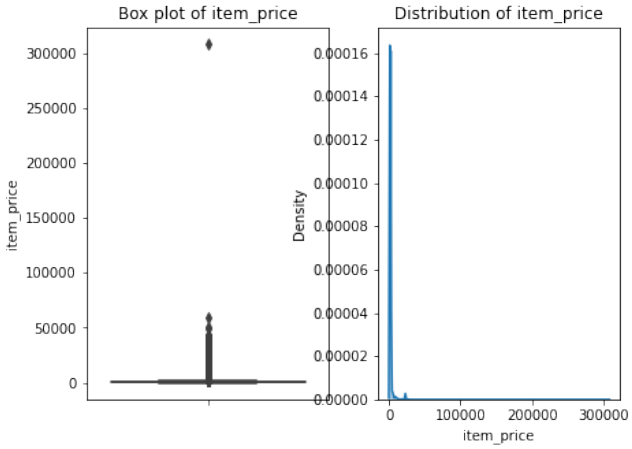


To be able to compare the results of our models in a different setting, first let's take a quick view about the data (products) in the specific dataset.

Opposite to our original dataset, here we have many different products from many different categories, and we have the following distribution of items per item category:

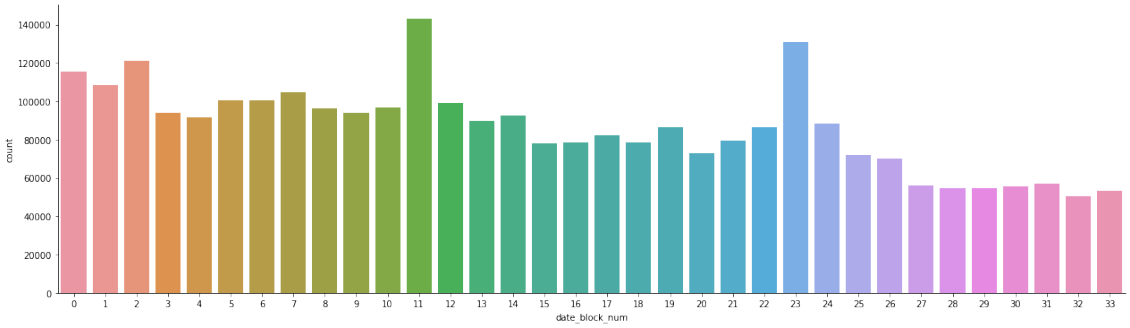


Next, let's take a look on the prices of the products at hand, and see how much they resemble FMCGs:



The fact that the huge majority of the products in this dataset have very small price is good for efficient comparison, as we have products that at least have some common characteristics with FMCGs.

Last but not least, let's take a look at the amount of data we have per week:



It seems that we have enough data from many different weeks (all year round), so, also in that sense, the Kaggle competition's data, have some common characteristics to ours.

So, now that we have provided insights about the nature of the competition and the fact that these data resemble satisfyingly well our initial data, let's see the model building we did and how it went.

- We tested the 4 best models from our original data.
- We tested our best meta-learner, the "2-best Stacking Classifier" to test again how efficient it is to use meta-learners and what kind of meta-learners should someone use for Sales Forecasting.

So, the models we tested are:

- ML models:

1. HuberRegression
2. XGBoost
3. CatBoost
4. Lightgbm

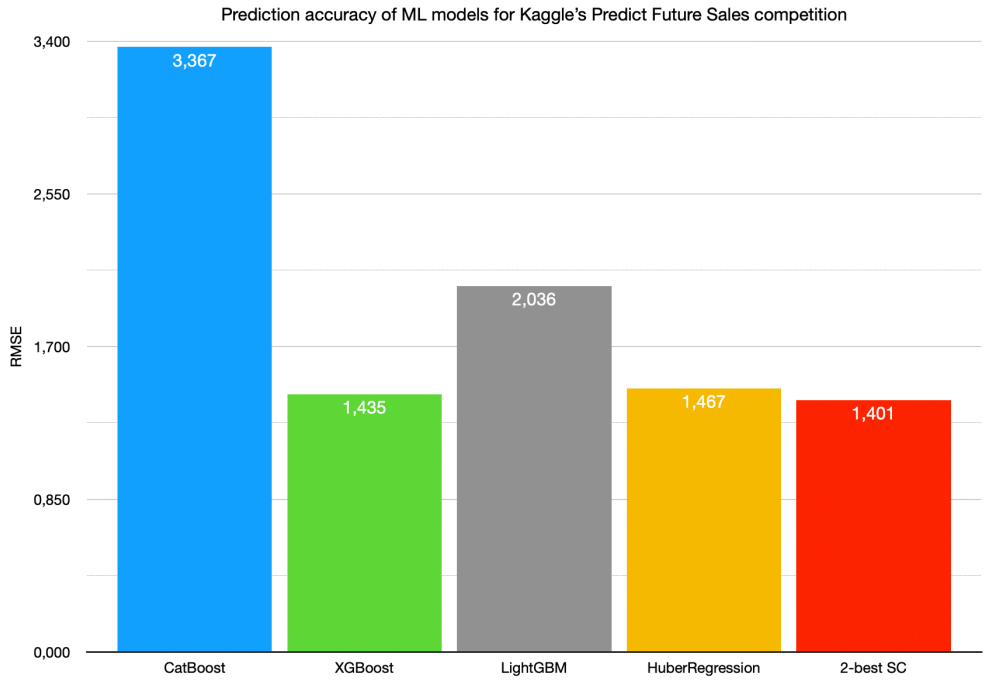
• Meta-learners:

2-best Stacking Classifier

The goal is to completely independently test the value of our top models from our original experiments. To do so, we chose this Kaggle competition that resembles our initial datasets, and so, the score of our ML models will be tested in this globally measured, independent setting. We only measure each model's RMSE, as we have shown that GBDT models seem to be extremely consistent across all key metrics (so measuring and comparing just their RMSE is sufficient).

The results were:

Model	RMSE
CatBoost	3,367
XGBoost	1,435
LightGBM	2,036
HuberRegression	1,467
2-best SC	1,401



So:

- CatBoost, while did extremely well in our data, was by far worse in this dataset.
- HuberRegression proved once more that is an extremely simple, but extremely powerful regressor, which scores amazingly well even in less homogeneous sales datasets. Thus, it can be efficiently used in nearly any kind of Sales Forecasting and is a must-go for the Sales Forecasting development by scholars and businesses' data teams.
- XGBoost was the most accurate model and defended its position as one of the most useful and most used ML models, being superior both in accuracy and diversity of use (usefulness).
- Our 2-best Stacking Regressor did a bit better, as expected, but not enough to satisfy the extra level of development and time commitment. Here it did just 2.4% better. Whether or not this is a valuable prediction difference remains is a bit subjective. We definitely recommend to businesses that want the best possible prediction accuracy to at least try meta-learners, but we see here that with not enough models to be combined and similar results

from the top models, the prediction accuracy improvement the meta-learners provide may be regarded as insignificant. As we can see, in a competition setting this improvement, although small, may be sufficient to skyrocket our performance in the competition's setting.

It seems that our work was good enough to be placed in the top submitted solutions worldwide. We scored in the top 1% of submissions from all over the world and specifically 46th out of more than 10,000 submissions (by the time of the screenshot we have fallen to the 70th place, but still in the top 1%).

All in all, it seems that the amount of time we saved by testing this Thesis' top scoring models for Sales Forecasting is significant (as we developed only the top 4 models and the best meta-learner, not blindly developing countless models as it regularly happens in relevant projects), and the prediction accuracy we achieved is exquisite.

This shows once more, that on the basis of this Thesis many businesses can be benefitted; both those who have already implemented Sales Forecasting models and prediction distribution processes, but also those who will develop such models and processes in the future. By having a clear idea which methods and which models perform optimally for sales forecasting for FMCGs, businesses can save a tremendous percentage of their time and investments and simultaneously develop models with surprisingly high prediction accuracy.

6. Conclusion

6.1 Results

The main goal was to study in depth the field of Sales Forecasting and the role of Machine Learning in modern sales forecasting, with a focus on fast-moving consumer goods (FMCGs).

For that reason, we started by providing the theoretical background of both FMCGs and Machine Learning in Sales Forecasting. This is necessary, as knowing the nature of the problem is extremely important in Machine Learning and Deep Learning implementations.

By having the full picture about the problem and the previous work was done about it (Chapter 3, "Previous Work") we can improve our models (e.g. better data-preprocessing and feature engineering), but also have good initial assumptions about the accuracy of every solution. As it happens in every relevant project, we are benchmarking different ML & DL models for our specific datasets (which, as we have proved, accurately represent the whole field of FMCGs) to find improvements in Sales Forecasting for FMCGs, draw important conclusions and give useful recommendations to businesses in these industries. We hope that with this Thesis, businesses can more effectively build sales forecasting models and sales predictions processes, and so they can improve their inventory management and modify their products' future prices (give discounts, create campaigns etc.) to maximise their revenues and profits.

The models with the best prediction accuracy from every category were:

- (simple) Regressors: HuberRegression
- GBDT: Catboost
- Neural Networks: LSTM
- Meta-Learners: 3-best Stacking Regressor

As expected, the Stacking Classifiers/Regressors (2-best SC/R and 3-best SC/R) achieved the maximum prediction accuracy, as they achieved lower errors than the best models they consist of and stack (CatBoost, Lightgbm, XGBoost, HuberRegression).

Kaggle competition

In order to have even more data and results about the models and meta-learners we tried in our main experiments, we competed in a Kaggle competition about Sales Forecasting. It was a good ground to test if the models that proved to be good top choices for FMCGs Sales Forecasting are also top choices for other types of products and more diverse Sales Forecasting.

The proposed models and methodology lead to the same outcomes, showing that the accuracy of the Stacking Classifier based on the the two or three best models is significantly higher than every individual model's accuracy. As a matter of fact, the solution was in the top 50, top 1% Kaggle solutions, out of more than 10,000 submitted solutions (46th place at the competition Leaderboard). Also, the models with the best prediction accuracy for the Kaggle competition were: a. XGBoost and b. HuberRegression.

XGBoost has excellent performance due to its scalability and accurate implementation of gradient boosting, optimizing high performance and low computational complexity. Accordingly, HuberRegression leads to very good results, because of its insensitivity to outliers.

As we spent less time for pre-processing and feature engineering at our Kaggle datasets, it is natural that HuberRegression scored better than other, more sophisticated, models, such as CatBoost and LGBM. We assume that if we haven't done such elaborate and complete data-preprocessing and feature engineering in our main datasets, we would have seen the same relevant scoring by HuberRegression, as HuberRegression would have lower "penalty" from worse data-preprocessing and feature engineering achieving the same or even better results in comparison with the GBDT models. However, with complete data-preprocessing and feature engineering it is natural that the GBDT models (XGBoost, LightGBM and CatBoost) are amazingly effective (the most effective algorithmic family) for problems with heterogeneous tabular data.

More detailed Results about our main datasets:

From all our case-studies we have seen that trying our top 4 basic ML models (HuberRegression, XGBoost, LightGBM and CatBoost) and having a "2-best Stacking Classifier solution" leads to extremely efficient Sales Forecasting, as it efficiently combines incredibly high prediction accuracy with relatively low training time, making it an excellent choice for real-world Sales Forecasting solutions implemented by companies in the fast-moving consumer goods industries.

These are the results and the characteristics of each one of our models:

	MAPE (%) (Standard)	MAPE (%) (Normal)	RMSE (Standard)	RMSE (Normal)	MSE (Standard)	MSE (Normal)	Execution Time (sec) (Standard)	Execution Time (sec) (Normal)	Best Parameters (Standard)	Best Parameters (Normal)
HuberRegression	17,413	19,654	3,596	3,516	12,929	12,362	895	1512	{'epsilon': 1.4}	{'epsilon': 1.4}
KNNReg	45,208	51,240	3,192	3,331	10,190	11,097	19542	19722	{'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'distance'}	{'metric': 'manhattan', 'n_neighbors': 15, 'weights': 'distance'}
Pass/AggReg	57,822	64,591	3,590	3,525	12,891	12,425	7	10	{'C': 0.1}	{'C': 0.1}
LassoRegression	89,261	89,261	4,033	4,033	16,264	16,264	1	1	{'selection': 'random'}	{'selection': 'random'}
RidgeRegression	62,120	57,160	3,496	3,288	12,221	10,809	1	1	{'alpha': 0.2}	{'alpha': 0.05}
XGBoost	4,580	5,454	0,494	0,822	0,244	0,676	1634	1792	{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 6, 'min_child_weight': 4, 'n_estimators': 300, 'nthread': 4, 'objective': 'reg:squarederror', 'silent': 1, 'subsample': 0.7}	{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 5, 'min_child_weight': 4, 'n_estimators': 300, 'nthread': 4, 'objective': 'reg:squarederror', 'silent': 1, 'subsample': 0.7}
RandomForestReg	53,246	42,092	2,061	1,986	4,248	3,945	727	727	{'max_depth': 6, 'max_features': 'sqrt', 'min_impurity_decrease': 0.001, 'n_estimators': 150}	{'max_depth': 6, 'max_features': 'sqrt', 'min_impurity_decrease': 0.001, 'n_estimators': 150}
CatBoost	2,486	1,816	0,429	0,566	0,184	0,321	133	178	estimator=catb_reg, param_grid=param_grid, cv= 5, n_jobs=-1, verbose=0	estimator=catb_reg, param_grid=param_grid, cv= 5, n_jobs=-1, verbose=0
LightGBM	2,981	3,075	0,514	0,655	0,264	0,429	17	15	{'boosting_type': 'goss'}	{'boosting_type': 'goss'}
2-best SC	2,000	1,643	0,373	0,485	0,139	0,235	153	187	model1 = lightgbm.LGBMRegressor (boosting_type = 'goss') model2 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('lightgbm', model1), ('catboost', model2)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)	model1 = lightgbm.LGBMRegressor (boosting_type = 'goss') model2 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('lightgbm', model1), ('catboost', model2)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)
3-best SC	1,990	1,622	0,358	0,467	0,128	0,218	191	249	model1 = xgboost.XGBRegressor (objective = 'reg: squarederror', colsample_bytree = 0.7, learning_rate = 0.03, max_depth = 5, min_child_weight = 4, n_estimators = 300, nthread = 4, silent = 1, subsample = 0.7) model2 = lightgbm.LGBMRegressor (boosting_type = 'goss') model3 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('xgb_reg', model1), ('lightgbm', model2), ('catboost', model3)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)	model1 = xgboost.XGBRegressor (objective = 'reg: squarederror', colsample_bytree = 0.7, learning_rate = 0.03, max_depth = 5, min_child_weight = 4, n_estimators = 300, nthread = 4, silent = 1, subsample = 0.7) model2 = lightgbm.LGBMRegressor (boosting_type = 'goss') model3 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('xgb_reg', model1), ('lightgbm', model2), ('catboost', model3)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)
MoE	24,440	24,264	4,228	4,228	17,878	17,877	850	830	model = MoE(8, 1)	model = MoE(8, 1)
KerasRegressor	38,093	38,093					628	653	model = Sequential (), model.add (Dense (128, input_dim = X_train.shape [1], activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (256, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (128, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (1)), opt = Nadam (1e-2), model.compile (optimizer = opt, loss = 'mean_squared_error')	model = Sequential (), model.add (Dense (128, input_dim = X_train.shape [1], activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (256, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (128, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (1)), opt = Nadam (1e-2), model.compile (optimizer = opt, loss = 'mean_squared_error')
MLP	39,073	34,273	2,100	2,470	4,408	6,103	2011	1772	mlp_reg = MLPRegressor (), param_grid = { 'hidden_layer_sizes': [(100,), (20, 20), (256, 256, 128)], 'activation': ['logistic', 'tanh']} reg_cv = GridSearchCV (estimator = mlp_reg, param_grid = param_grid, cv = 5, n_jobs = -1)	mlp_reg = MLPRegressor (), param_grid = { 'hidden_layer_sizes': [(100,), (20, 20), (256, 256, 128)], 'activation': ['logistic', 'tanh']} reg_cv = GridSearchCV (estimator = mlp_reg, param_grid = param_grid, cv = 5, n_jobs = -1)
LSTM	24,091	24,090	4,231	4,231	17,899	17,899	818	847	loss_function = nn.L1Loss (), optimizer = torch.optim.Adam (model.parameters (), lr = 0.001), epochs = 100, best_mse = np.inf, model = LSTM (X_train_norm.shape [1], 100) , sss_norm = ShuffleSplit (n_splits = 1, test_size = 0.1)	loss_function = nn.L1Loss (), optimizer = torch.optim.Adam (model.parameters (), lr = 0.001), epochs = 100, best_mse = np.inf, model = LSTM (X_train_norm.shape [1], 100) , sss_norm = ShuffleSplit (n_splits = 1, test_size = 0.1)

Some useful comments:

- The HuberRegression got the best results by having the highest regularization parameter, and the same happened in the case of MLP. In our experiments, the simplest model was chosen in most models (we will explain in a bit why).
- HuberRegression impressed us, as it achieved low errors and good prediction accuracy. It is a very useful model for Sales Forecasting implementations, where there are always significant

outliers. Its extreme resistance to outliers makes it very useful, even when we have little data. In those cases, we can see that is better than more complicated models.

- The KNN model could be executed only one time, as, since the prediction is based on finding the closest elements from the test element, it takes an enormous computational time (5+ hours execution time, more than any other model). With a quick estimation, we can see that it will take around 32 billion operations for each prediction fold in each combination. Thus, it is by far the most computational heavy ML algorithm we tried, and by far slower even than complicated neural networks implementations.
- We didn't perform a GridSearch at our Deep Learning models due to the extreme cost, but also because GridSearch impacts the task at hand (searching for relations at the entire architecture instead of changing the size of a specific layer). So, instead of doing GridSearch, we implemented an early-stop technique to prevent overfitting. As we didn't use any kind of validation set at our other models, we took a 10% split for GridSearch, but also kept our test set in pristine condition.

To have a better look at the training of our models and their best parameters:

	Best Parameters (Standard)	Best Parameters (Normal)
HuberRegression	{'epsilon': 1.4}	{'epsilon': 1.4}
KNNReg	{'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'distance'}	{'metric': 'manhattan', 'n_neighbors': 15, 'weights': 'distance'}
Pass/AggReg	{'C': 0.1}	{'C': 0.1}
LassoRegression	{'selection': 'random'}	{'selection': 'random'}
RidgeRegression	{'alpha': 0.2}	{'alpha': 0.05}
XGBoost	{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 6, 'min_child_weight': 4, 'n_estimators': 300, 'nthread': 4, 'objective': 'reg:squarederror', 'silent': 1, 'subsample': 0.7}	{'colsample_bytree': 0.7, 'learning_rate': 0.03, 'max_depth': 5, 'min_child_weight': 4, 'n_estimators': 300, 'nthread': 4, 'objective': 'reg:squarederror', 'silent': 1, 'subsample': 0.7}
RandomForestReg	{'max_depth': 6, 'max_features': 'sqrt', 'min_impurity_decrease': 0.001, 'n_estimators': 150}	{'max_depth': 6, 'max_features': 'sqrt', 'min_impurity_decrease': 0.001, 'n_estimators': 150}
CatBoost	estimator=catb_reg, param_grid=param_grid, cv= 5, n_jobs=-1, verbose=0	estimator=catb_reg, param_grid=param_grid, cv= 5, n_jobs=-1, verbose=0
LightGBM	{'boosting_type': 'goss'}	{'boosting_type': 'goss'}
2-best SC	model1 = lightgbm.LGBMRegressor (boosting_type = 'goss') model2 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('lightgbm', model1), ('catboost', model2)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)	model1 = lightgbm.LGBMRegressor (boosting_type = 'goss') model2 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('lightgbm', model1), ('catboost', model2)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)
3-best SC	model1 = xgboost.XGBRegressor (objective = 'reg: squarederror', colsample_bytree = 0.7, learning_rate = 0.03, max_depth = 5, min_child_weight = 4, n_estimators = 300, nthread = 4, silent = 1, subsample = 0.7) model2 = lightgbm.LGBMRegressor (boosting_type = 'goss') model3 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('xgb_reg', model1), ('lightgbm', model2), ('catboost', model3)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)	model1 = xgboost.XGBRegressor (objective = 'reg: squarederror', colsample_bytree = 0.7, learning_rate = 0.03, max_depth = 5, min_child_weight = 4, n_estimators = 300, nthread = 4, silent = 1, subsample = 0.7) model2 = lightgbm.LGBMRegressor (boosting_type = 'goss') model3 = CatBoostRegressor (loss_function = 'RMSE', verbose = 0), estimators = [('xgb_reg', model1), ('lightgbm', model2), ('catboost', model3)] stack_reg = StackingRegressor (estimators = estimators, final_estimator = HuberRegressor (), cv = 5, verbose = 1)
MoE	model = MoE(8, 1)	model = MoE(8, 1)
KerasRegressor	model = Sequential (), model.add (Dense (128, input_dim = X_train.shape [1], activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (256, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (128, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (1)), opt = Nadam (1e-2), model.compile (optimizer = opt, loss = 'mean_squared_error')	model = Sequential (), model.add (Dense (128, input_dim = X_train.shape [1], activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (256, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (128, activation = 'relu')), model.add (Dropout (0.2)), model.add (Dense (1)), opt = Nadam (1e-2), model.compile (optimizer = opt, loss = 'mean_squared_error')
MLP	mlp_reg = MLPRegressor (), param_grid = { 'hidden_layer_sizes': [(100,), (20, 20), (256, 256, 128)], 'activation': ['logistic', 'tanh']} reg_cv = GridSearchCV (estimator = mlp_reg, param_grid = param_grid, cv = 5, n_jobs = -1)	mlp_reg = MLPRegressor (), param_grid = { 'hidden_layer_sizes': [(100,), (20, 20), (256, 256, 128)], 'activation': ['logistic', 'tanh']} reg_cv = GridSearchCV (estimator = mlp_reg, param_grid = param_grid, cv = 5, n_jobs = -1)
LSTM	loss_function = nn.L1Loss (), optimizer = torch.optim.Adam (model.parameters (), lr = 0.001), epochs = 100, best_mse = np.inf, model = LSTM (X_train_norm.shape [1], 100) , sss_norm = ShuffleSplit (n_splits = 1, test_size = 0.1)	loss_function = nn.L1Loss (), optimizer = torch.optim.Adam (model.parameters (), lr = 0.001), epochs = 100, best_mse = np.inf, model = LSTM (X_train_norm.shape [1], 100) , sss_norm = ShuffleSplit (n_splits = 1, test_size = 0.1)

Details about Execution Time:

We know that it is not common to compare execution time for tasks that are not to be performed multiple times. Sales Forecasting is, at least theoretically, one of them. However, in real-world implementations Sales Forecasting models need to run several times.

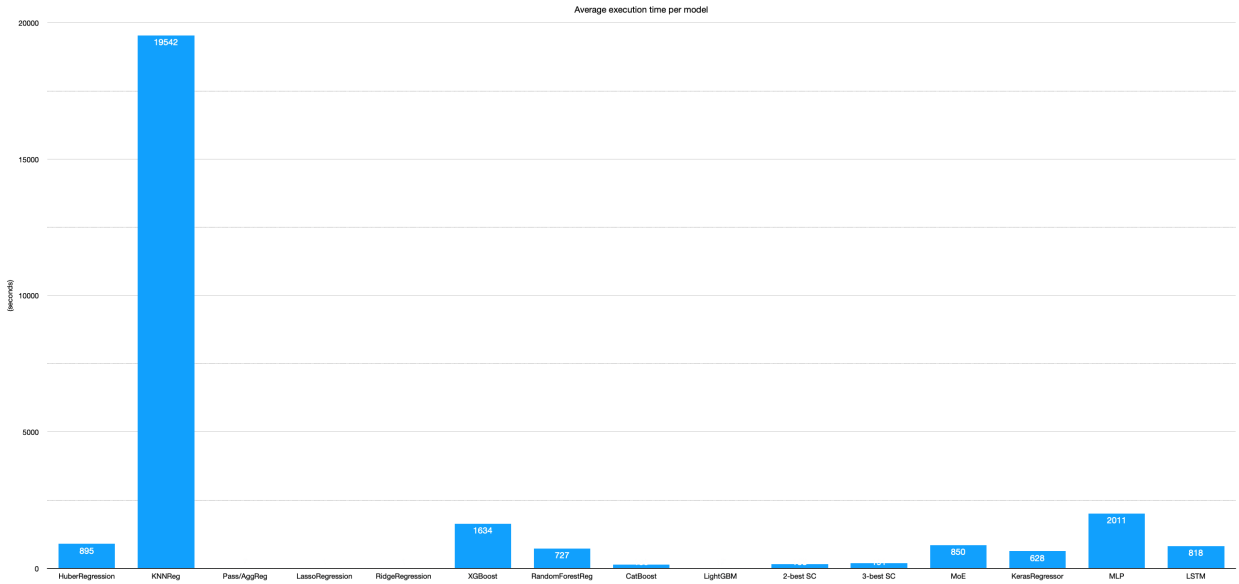
This happens because:

- Businesses decide to improve their older Sales Forecasting models.
- Businesses change their Sales Forecasting processes.

- New research that proposes more accurate models sees the light, and businesses want to try them out.
- Changes in personnel in marketing or data teams or in business management impacts Sales Forecasting methods and interior processes.
- New data about existent products are being created.
- New products are being launched.
- New features are taken into account.
- Businesses decide to increase their investment on their Sales Forecasting and, so, they invest to re-run and improve their existent models, methods and processes.

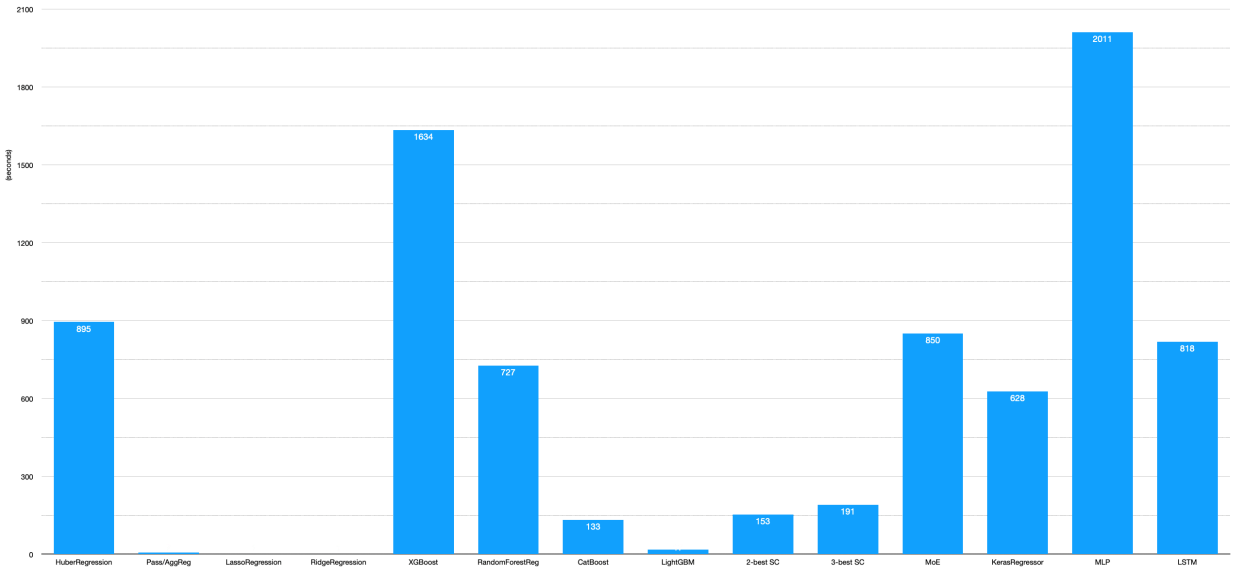
For that reason we measured the execution time in our models and presented it per model category in Chapter 5 (“Experiments”).

To take a quick view of the execution time of all our models:

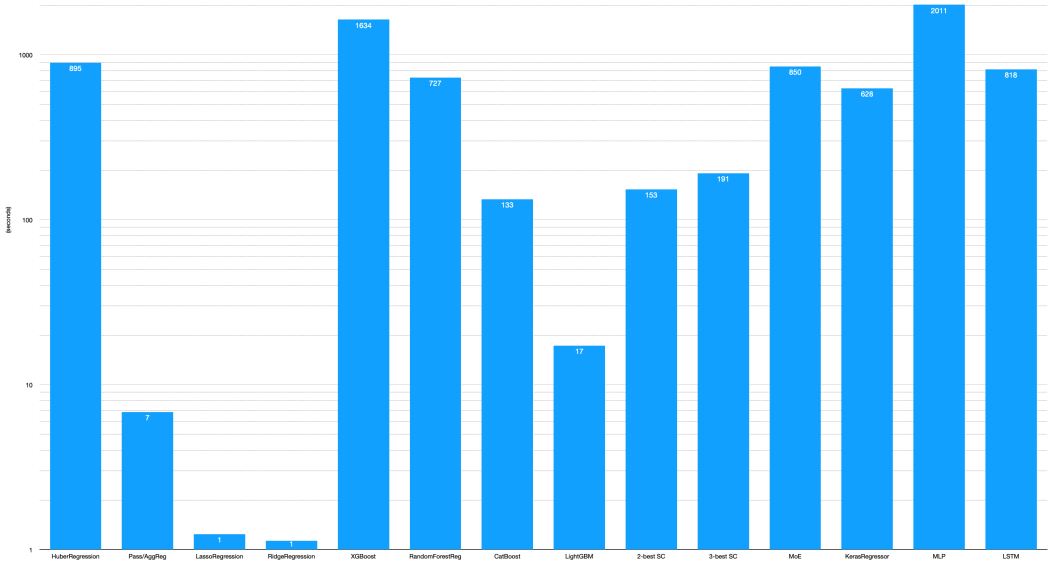


We see that KNN Regression takes nearly infinitely times more time than all our other models. For that reason, we exclude it from our execution time plots.

With linear axis scale:

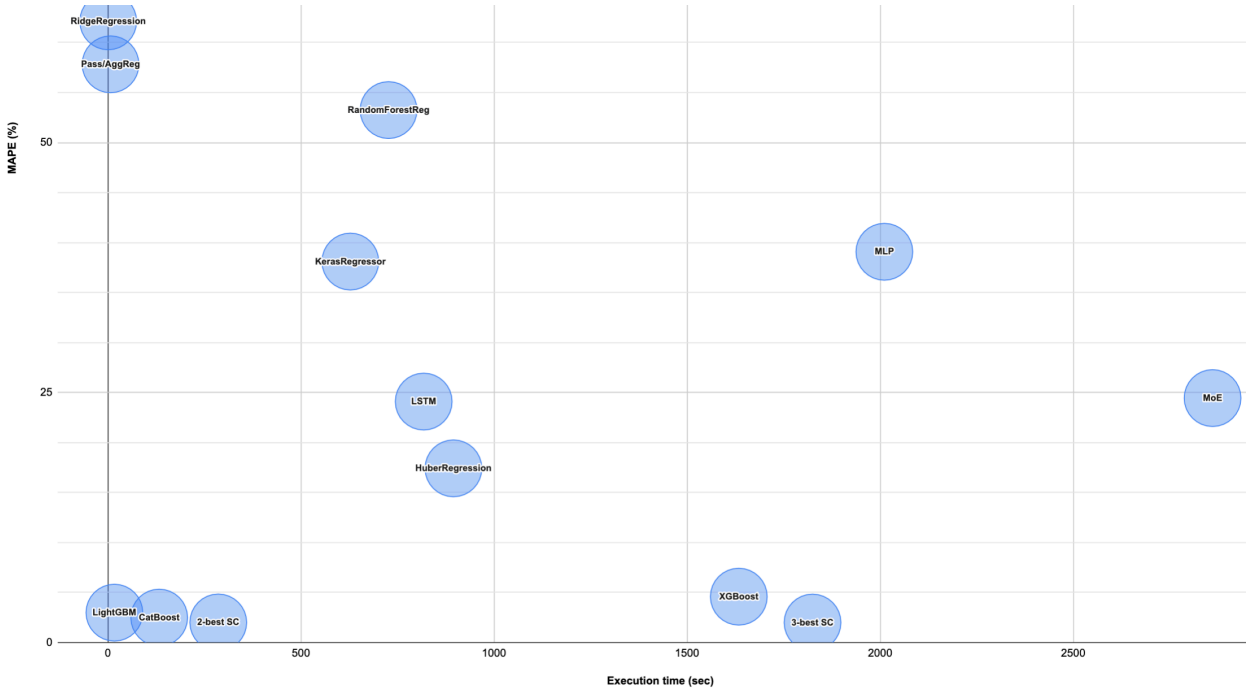


With logarithmic axis scale:



- It is easy to see that our best performing models (CatBoost and LightGBM) take also very little time, thus, making them ideal candidates for Sales Forecasting for FMCGs to every business in the industry.
- Also, please take into account that in order to implement the Stacking Regressors, their execution time is “extra” to all the other models. While the execution of all the other models can theoretically be parallelized, that is not the case for N-best StackingRegressors as they are being implemented in 2-steps, and they first need to have the results from all the other models and choose the best models to ensemble/stack together.

So, a good bubble plot of our models looks like this:



And as we can see:

- CatBoost and LightGBM are not just great models for Sales Forecasting for FMCGs, but also very fast.
- If we have just implemented the CatBoost and LightGBM solutions, the 2-best StackingRegressor would have waited only for those, and with the extra time-penalty for stacking/ensembling, we can see its final execution time at the plot above. Of course, this is the optimal time, because in a real case study, we would have to run multiple models, and choose the 2-best, but if other models had a higher execution time than our two best models, which is extremely probable, our total execution time would be higher. However, if we know from the start which are our top models, then we can just run the 2-best StackingRegressor based on those. So, the actual running time would be the one depicted in the barplots above, which makes the Stacking Classifier/Regressor one of the fastest models overall.
- As with the 2-best StackingRegressor, the optimal time for the 3-best StackingRegressor is higher, as it has to wait for the XGBoost model. Again, in a real case study the actual execution time would be much bigger as it would have to wait for more models. On the other side of the coin, if we knew from the start which are the optimal models, we could run the 3-best Stacking Regressor in Stage 1 (not implementing individual models first, but directly running our Stacking Regressor). This is extremely interesting and important, as the StackingRegressor would have had a total execution time lower than some of the individual models, and at the same time could provide greater results. So, this point also shows why the benchmarking we conducted is important, as by having an accurate initial prediction about the best models for Sales Forecasting for FMCGs, a business can build better models; just a StackingRegressor, with greater accuracy and less costs than a usual ML/DL project. (More on "Details about our Meta-learners results").
- From the Deep Learning models, MLP has the worst accuracy, but also the worst execution time. For that reason, the Mixture of Experts, also is slow as it needs to wait for the MLP neural network to complete.
- HuberRegression, as we also said earlier, impressed us, as it is a simple regressor that managed to achieve very satisfying accuracy at a low execution time. It is definitely a model that needs to be taken more into account, especially in the case of Sales Forecasting where there are always significant outliers. Its extreme resistance to outliers makes it very useful, even when we have little data to train more complicated models.

Details about our Deep Learning models' results:

For the businesses who want to work with Deep Learning implementations, one can see from our experiments that many machine learning models outperform the DL solutions.

The final results show that even the neural network approaches are not able to provide improvements over the ensemble models for the cases of Sales Forecasting. This can be due to a few data and model features, such as low dimension, which can lead to a fast overfitting in very flexible models. As we said in Chapter 4, Gradient Boosted Decision Trees are the best ML models for tabular heterogeneous datasets. Also, the input size can impact the feature extraction on the LSTM model. For that reason, more tests on different datasets need to be done, but as our datasets (both from our Marketplaces and the Kaggle competition) are excellent representations of the real-world business datasets that are used for actual real sales forecasting, our results are as close to reality as possible.

However, among the neural methods, it can be observed that the LSTM provided the best results. LSTM and MoE presented similar results over all metrics. The Mixture of Experts model made an improvement over the MLP model, which is expected due to the architecture similarity.

- All our Deep Learning models (KerasRegressor, MLP, LSTM) could be used for FMCGs Sales Forecasting.
- In comparison to simple regressors, Deep Learning models give better results, but not necessarily better results from GBDT models as they quickly overfit. Potentially they could do better if we had datasets with more features and higher dimensions, but as the problem of Sales Forecasting uses tabular datasets, DL executions perform worse than GBDT models.
- As we expected, all Deep Learning models have significant execution time. While execution time is usually not a problem for the cases of sales forecasting, it can severely impact the cost of the initial investment, as it impacts the amount of working hours needed, which in the case of small to medium businesses' may be of high significance. The use of extra computational resources (as they have become a commodity) definitely lowers the execution time, but at potentially higher running/processing costs.
- The MLP model performs a GridSearch that change its activation function as well as the hidden layers settings.
- As we pointed out above, the simplest model is always being chosen in GridSearch. For instance, our MLP was designed to try a 1-hidden layer with 100 neurons, a 2-hidden layers with 20 neurons each and 3-hidden layers with 256, 256 and 128 neurons (the exact same approach we used for our KerasRegressor). It chosen to use the simplest implementation; the 1-hidden layer with 100 neurons.
- Our LSTM execution is a one-layer LSTM that will extract features to a fully connected layer.
- The LSTM model behaved exactly as we predicted; even its simplest architectures had real problem with overfitting. This usually happens to non-stationary models, as they can't define a function to properly represent the data. So, such a powerful, non-stationary model, such as LSTM, is too powerful for simple sales forecasting, especially that of FMCGs where we have tabular data, and, thus, it will overfit even with optimal regularisation.

Details about our Meta-learners' results:

One of the main goals of this Thesis, apart from studying in total the case of Sales Forecasting for FMCGs, was to take a closer look into meta-learners, and especially Stacking Classifier/ Regressors and Mixture of Experts.

What we have seen from our experiments is that:

- 2-best and 3-best Stacking Classifiers/Regressors are very close across all key metrics. So, by also taking into account our execution time analysis, it seems that the 2-best StackingRegressor is the optimal solution for the case of Sales Forecasting for FMCGs.
- In comparison to our best model, our Stacking Classifier/Regressor gave better results by 25% and in comparison to our 3rd best model, our meta-learners gave better results by 130%. These are improvements of great significance and so we can draw the conclusion that the use of StackingRegressors is a must for every Sales Forecasting implementation.
- One of the main restrictions in using a Mixture of Experts (MoE), is that it needs models from the same group (e.g. neural networks). However, a StackingClassifier can stack/ensemble any kind of models. So, as different parts or cases of our data can more easily be predicted by a specific architecture (e.g. CatBoost) than another that is better in a different part or feature, the stacking will have the upper hand in comparison with using a MoE.
- The Mixture of Experts model we used is based on the same MLP settings, with softmax function to the gate network. Our MoE, as expected, improved the MLP model, but didn't provide sufficiently good results. However, there is definitely room for improvement as they seem to be a great investment overall. We tried to improve our MoE implementation, but the results show that it is better to choose a simpler model, as more flexible models are too

powerful, and due to low dimensionality overfit quickly. We used a hierarchical one, which gave us better results than a regular one due to the nature of the problem.

- The Mixture of Experts (MoE) was about 1% better than LSTM across all key metrics for evaluating Deep Learning models.

It seems that the Mixture of Experts is not very useful in Sales Forecasting, as its architecture is based on gaussian curves and, thus, it can be extremely flexible, which is usually an advantage, but in the case of Sales Forecasting for FMCGs, due to low dimensional input and the fact that we have heterogeneous tabular data structures (as we have many sales, similar prices, specific features to be taken into account etc.) it overfits very fast. However, overall, we see that meta-learners' implementations, and specifically StackingRegressors, have many advantages, such as:

- They lead to extremely accurate Sales Forecasting, because they efficiently combine the best single models into better unified models.
- They combine high prediction accuracy with low training time, making them an excellent choice for real-world Sales Forecasting solutions implemented by companies in the fast-moving consumer goods industries.
- They can be used both by businesses who haven't yet invested in their Sales Forecasting processes (e.g. new businesses, startups etc.) and by businesses that have Sales Forecasting models and processes, but they want to improve the accuracy of their models with additional optimal investments.

6.2 Future Work

Regarding future work, first, let's discuss potential improvements to our models and additional experiments. The Mixture of Experts we used, was limited to the MLP architecture which performed sub-optimally. In the future, more work could be done to implement a MoE based on better neural networks, or GBDT models. We avoided to directly compare and draw conclusions about the performance of StackingRegressors to that of the MoE because they were based on different models; thus, producing different results and achieving different accuracies. However, from this difference in model training we were able to point out that while MoE needs models from the same group (e.g. neural networks), a Stacking Classifier/Regressor can ensemble any kind of models. Also, it would be a good idea to have a measurement of models agreement with each other. Meta-learners are strongly influenced by combining models agreement. Having accurate models with low agreement is the optimal scenario as the ensemble model is much better than every individual model. The idea of N-best SC/R we used here satisfies per se the requirement for strong models, but in order to have even better results, a measurement of models' agreement seems necessary. Maybe a tradeoff between combining the best models and combining models with low agreement can be beneficial (e.g. combining the 1st and 4th model, if they have lower agreement than that of the 1st with the 2nd). In any case, more work about the best choice for a meta-learner in more diverse Sales Forecasting cases, seems very welcome.

More work on additional datasets seems also useful. While we used good representations of the current case of Sales Forecasting in real-world business settings, we didn't experimented enough with completely different approaches. One could argue that more or different features could and should be used in Sales Forecasting for FMCGs and so businesses need to make changes in the amount or quality and quantity of data they are keeping.

Additionally, data transformation, data-preprocessing and feature engineering always play a gargantuan role in any Machine Learning project and implementation, and so one way to view and review Machine Learning Sales Forecasting models is to review all the steps before model building; from data keeping to data management to data analysis to business intelligence.

Although we worked with double data to implementation two feature scaling methods (standardisation and normalisation) we didn't find significant differences between the two in our experiments. However, we had to test both, as feature scaling plays an important role in pre-processing, and potentially even different feature scaling techniques can improve feature engineering and the performance of some of our models. So, as it seems that the choice of our feature scaling method doesn't play a very important role, the fact that the industry is divided about which one to use makes perfect sense.

Furthermore, while we tried the top performing models from the most recent research in the field, there are other similar methods and other neural networks architectures that can be implemented. For example, CNN seems a fine addition to our neural networks building and benchmarking. Simultaneously, the Facebook Prophet predictor has shown substantial improvements in benchmark tasks in many ML implementations. The Prophet predictor seems to be a must-go inclusion in any kind of Machine Learning and Deep Learning models benchmarking in the future. Also, GAN models become more and more popular in Forecasting, as they can be perfect adaptive prediction models. Training a GAN based on the best models found in this Thesis seems like a great way to go. An N-best GAN with a mixture of generators (MGAN as characterised by Quan Hoang et. al, 2018), can be the evolution of our N-best SC/R methodology, as it not just combines the optimal models, but pushes each model to its "prediction limits", achieving optimal forecasting. More research about sales forecasting algorithms and methods is currently being produced and we await new implementations and suggestions, such as the DAE-LSTM algorithm.

In any case, Sales Forecasting is becoming more and more important every day across all departments of a business, and soon it will be a must-have tool for companies of all industries, but especially for businesses in the FMCGs industries where Sales Forecasting plays a foundational role to these companies' viability and profitability.

6.3 Conclusion

The goals of this Thesis have being completely met, as we managed to build, test and benchmark the top methods, models and meta-learners for sales forecasting, that are being used both in a theoretical environment and in the real world. CatBoost and LightGBM are amazing choices for Sales Forecasting for FMCGs and by using a StackingRegressor with them, we can guarantee optimal results, under nearly any circumstances, as long we do sufficient data-preprocessing and feature engineering. So, there we have it, an overall good, unified model for fast and accurate Sales Forecasting for FMCGs: a Stacking Classifier/Regressor based on CatBoost and LightGBM.

In case businesses and consultancies need to experiment more on a specific case/project for Sales Forecasting, we can recommend them to also use XGBoost and HuberRegression to get the full picture, and to also be sure that they optimally used their time to build and test the best possible models. Finally, if there is a need to also try neural networks architectures, LSTM is by far the most promising. However, due to the problem's occasional low dimensionality and full tabular datasets' availability, it seems that GBDT models are the undisputed champions for FMCGs Sales Forecasting.

7. Bibliography/Βιβλιογραφία

[1] Shouwen Ji, Xiaojing Wang, Wenpeng Zhao, Dong Guo, "An Application of a Three-Stage XGBoost-Based Model to Sales Forecasting of a Cross-Border E-Commerce Enterprise", *Mathematical Problems in Engineering*, vol. 2019, Article ID 8503252, 15 pages, 2019. https://doi.org/10.1155/2019/8503252

[2] Hoffmann, Kira and Mahlendorf, Matthias D. and Pettersson, Kim, "The Cost Impact of Sales Prediction Accuracy", 2021. http://dx.doi.org/10.2139/ssrn.3782336

[3] Yonghe Zhao et al, "Optimization of a Comprehensive Sequence Forecasting Framework Based on DAE-LSTM Algorithm", J. Phys., Conf. Ser. 1746 012087, 2021. https://iopscience.iop.org/article/10.1088/1742-6596/1746/1/012087

[4] Nikolas Ulrich Moroff, Ersin Kurt, Josef Kamphues, "Machine Learning and Statistics: A Study for assessing innovative Demand Forecasting Models", Procedia Computer Science, Volume 180, 2021, Pages 40-49, ISSN 1877-0509. https://doi.org/10.1016/j.procs.2021.01.127

[5] A Raiyani et al, Usage of time series forecasting model in Supply chain sales prediction, 2021, IOP Conf. Ser.: Mater. Sci. Eng. 1042 012022. https://doi:10.1088/1757-899X/1042/1/012022

[6] McAfee, A. & Brynjolfsson, E., "Big data. The management revolution", 2012, *Harvard Business Review*, 90 (10), 61–67. https://hbr.org/2012/10/big-data-the-management-revolution

[7] McCarthy, T. M., Davis, D. F., Golicic, S. L. & Mentzer, J. T., "The evolution of sales forecasting manage- ment: a 20-year longitudinal study of forecasting practices", *Journal of Forecasting*, 2006, 25 (5), 303-324. http://dx.doi.org/10.1002/for.989

[8] Yun Dai, Jinghao Huang, "A Sales Prediction Method Based on LSTM with Hyper-Parameter Search", 2021, *J. Phys.: Conf. Ser.* 1756 012015. https://doi:10.1088/1742-6596/1756/1/012015

[9] Samaneh Beheshti-Kashi, Hamid Reza Karimi, Klaus-Dieter Thoben, Michael Lütjen & Michael Teucke, "A survey on retail sales forecasting and prediction in fashion markets", 2015, Systems Science & Control Engineering, 3:1, 154-161, https://doi.org/10.1080/21642583.2014.999389

[10] He Wei, QingTao Zeng, "Research on sales Forecast based on XGBoost-LSTM algorithm Model", 2021, *J. Phys.: Conf. Ser.* 1754 012191. https://doi:10.1088/1742-6596/1754/1/012191

[11] Punit Gupta et al, "Implementation of Demand Forecasting – A Comparative Approach", 2021, J. Phys.: Conf. Ser. **1714** 012003. https://doi:10.1088/1742-6596/1714/1/012003

[12] Josep-Francesc Valls, María José Andrade and Raquel Arribas, "Consumer attitudes towards brands in times of great price sensitivity", 2011, Four case studies, *Innovative Marketing* , *7*(2).

[13] Niladri Syam, Arun Sharma, "Waiting for a sales renaissance in the fourth industrial revolution: Machine learning and artificial intelligence in sales research and practice", Industrial Marketing Management, Volume 69, 2018, Pages 135-146, ISSN 0019-8501. https://doi.org/10.1016/j.indmarman.2017.12.019

[14] Marko Bohanec, Mirjana Kljajić Borštnar, Marko Robnik-Šikonja, "Explaining machine learning models in sales predictions", Expert Systems with Applications, Volume 71, 2017, Pages 416-428, ISSN 0957-4174. https://doi.org/10.1016/j.eswa.2016.11.010

[15] Tsan-Ming Choi, Chi-Leung Hui, Na Liu, Sau-Fun Ng, Yong Yu, "Fast fashion sales forecasting with limited data and time", Decision Support Systems, Volume 59, 2014, Pages 84-92, ISSN 0167-9236. https://doi.org/10.1016/j.dss.2013.10.008

[16] F.L. Chen, T.Y. Ou, "Sales forecasting system based on Gray extreme learning machine with Taguchi method in retail industry", Expert Systems with Applications, Volume 38, Issue 3, 2011, Pages 1336-1345, ISSN 0957-4174. https://doi.org/10.1016/j.eswa.2010.07.014

[17] Zhang, C. and Y. Ma, "*Ensemble machine learning: methods and applications",* Springer, 2012

[18] Zliobaite, I., J. Bakker, and M. Pechenizkiy (2009). "Towards Context Aware Food Sales Prediction." In: *2009 IEEE International Conference on Data Mining Workshops*. IEEE, pp. 94–99. ISBN: 978-1-4244-5384-9. https://doi.org/10.1109/ICDMW.2009.60

[19] Z.X. Guo, W.K. Wong, Min Li, "A multivariate intelligent decision-making model for retail sales forecasting", Decision Support Systems, Volume 55, Issue 1, 2013, Pages 247-255, ISSN 0167-9236. https://doi.org/10.1016/j.dss.2013.01.026

[20] Sébastien Thomassey, Antonio Fiordaliso, A hybrid sales forecasting system based on clustering and decision trees, Decision Support Systems, Volume 42, Issue 1, 2006, Pages 408-421, ISSN 0167-9236. https://doi.org/10.1016/j.dss.2005.01.008

[21] Anderson, J. L., 1996: A method for producing and evaluating probabilistic forecasts from ensemble model integrations. *J. Climate,* 9**,** 1518–1530.

[22] M. Leutbecher, T.N. Palmer, "Ensemble forecasting", Journal of Computational Physics, Volume 227, Issue 7, 2008, Pages 3515-3539, ISSN 0021-9991. https://doi.org/10.1016/j.jcp.2007.02.014

[23] Vislocky, R. L., and J. M. Fritsch, "Improved model output statistics forecasts through model consensus", 1995, *Bull. Amer. Meteor. Soc.,* 76**,** 1157–1164

[24] Jujie Wang, Xin Sun, Qian Cheng, Quan Cui, "An innovative random forest-based nonlinear ensemble paradigm of improved feature extraction and deep learning for carbon price forecasting", Science of The Total Environment, Volume 762, 2021, 143099, ISSN 0048-9697. https://doi.org/10.1016/j.scitotenv.2020.143099

[25] Ling Tang, Lean Yu, Shuai Wang, Jianping Li, Shouyang Wang, "A novel hybrid ensemble learning paradigm for nuclear energy consumption forecasting", Applied Energy, Volume 93, 2012, Pages 432-443, ISSN 0306-2619. https://doi.org/10.1016/j.apenergy.2011.12.030

[26] Guoqiang Zhang, Jifeng Guo, "A novel ensemble method for residential electricity demand forecasting based on a novel sample simulation strategy", Energy, Volume 207, 2020, 118265, ISSN 0360-5442. https://doi.org/10.1016/j.energy.2020.118265

[27] Ramon Gomes da Silva, Matheus Henrique Dal Molin Ribeiro, Sinvaldo Rodrigues Moreno, Viviana Cocco Mariani, Leandro dos Santos Coelho, "A novel decomposition-ensemble learning framework for multi-step ahead wind energy forecasting", Energy, Volume 216, 2021, 119174, ISSN 0360-5442. https://doi.org/10.1016/j.energy.2020.119174

[28] Sonia Kahiomba Kiangala, Zenghui Wang, "An effective adaptive customization framework for small manufacturing plants using extreme gradient boosting-XGBoost and random forest ensemble learning algorithms in an Industry 4.0 environment", Machine Learning with Applications, Volume 4, 2021, 100024, ISSN 2666-8270. https://doi.org/10.1016/j.mlwa.2021.100024

[29] Christian Johansson, Markus Bergkvist, Davy Geysen, Oscar De Somer, Niklas Lavesson, Dirk Vanhoudt, "Operational Demand Forecasting In District Heating Systems Using Ensembles Of Online Machine Learning Algorithms", Energy Procedia, Volume 116, 2017, Pages 208-216, ISSN 1876-6102. https://doi.org/10.1016/j.egypro.2017.05.068

[30] Shaoze Cui, Yunqiang Yin, Dujuan Wang, Zhiwu Li, Yanzhang Wang, "A stacking-based ensemble learning method for earthquake casualty prediction", Applied Soft Computing, Volume 101, 2021, 107038, ISSN 1568-4946. https://doi.org/10.1016/j.asoc.2020.107038

[31] Tiago Pinto, Isabel Praça, Zita Vale, Jose Silva, "Ensemble learning for electricity consumption forecasting in office buildings", Neurocomputing, Volume 423, 2021, Pages 747-755, ISSN 0925-2312. https://doi.org/10.1016/j.neucom.2020.02.124

[32] Lean Yu, Yueming Ma, Mengyao Ma, “An effective rolling decomposition-ensemble model for gasoline consumption forecasting”, Energy, Volume 222, 2021, 119869, ISSN 0360-5442. https://doi.org/10.1016/j.energy.2021.119869

[33] K.H. van Donselaar, J. Peters, A. de Jong, R.A.C.M. Broekmeulen, “Analysis and forecasting of demand during promotions for perishable items”, International Journal of Production Economics, Volume 172, 2016, Pages 65-75, ISSN 0925-5273. https://doi.org/10.1016/j.ijpe.2015.10.022

[34] Philip Doganis, Alex Alexandridis, Panagiotis Patrinos, Haralambos Sarimveis, “Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing”, Journal of Food Engineering, Volume 75, Issue 2, 2006, Pages 196-204, ISSN 0260-8774. https://doi.org/10.1016/j.jfoodeng.2005.03.056

[35] Bangzhu Zhu, Dong Han, Ping Wang, Zhanchi Wu, Tao Zhang, Yi-Ming Wei, “Forecasting carbon price using empirical mode decomposition and evolutionary least squares support vector regression”, Applied Energy, Volume 191, 2017, Pages 521-530, ISSN 0306-2619. https://doi.org/10.1016/j.apenergy.2017.01.076

[36] Tao Huang, Robert Fildes, Didier Soopramanien, “The value of competitive information in forecasting FMCG retail product sales and the variable selection problem”, European Journal of Operational Research, Volume 237, Issue 2, 2014, Pages 738-748, ISSN 0377-2217. https://doi.org/10.1016/j.ejor.2014.02.022

[37] Shaohui Ma, Robert Fildes, “Retail sales forecasting with meta-learning”, European Journal of Operational Research, Volume 288, Issue 1, 2021, Pages 111-128, ISSN 0377-2217. https://doi.org/10.1016/j.ejor.2020.05.038

[38] Tal Peretz, “Mastering The New Generation of Gradient Boosting”, https://www.kdnuggets.com/2018/11/mastering-new-generation-gradient-boosting.html

[39] Abhishek Sharma, “What makes LightGBM lightning fast?”,https://towardsdatascience.com/what-makes-lightgbm-lightning-fast-a27cf0d9785e

[40] Vishal Morde, Venkat Anurag Setty, “XGBoost Algorithm: Long May She Reign!”, https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[41] Christopher Olah, “Understanding LSTM Networks”, 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[42] Eric M. Burger, Scott J. Moura, "Gated ensemble learning method for demand-side electricity load forecasting”, Energy and Buildings, Volume 109, 2015, Pages 23-34, ISSN 0378-7788. https://doi.org/10.1016/j.enbuild.2015.10.019

[43] Goodness C. Aye, Mehmet Balcilar, Rangan Gupta, Anandamayee Majumdar, "Forecasting aggregate retail sales: The case of South Africa”, International Journal of Production Economics, Volume 160, 2015, Pages 66-79, ISSN 0925-5273. https://doi.org/10.1016/j.ijpe.2014.09.033

[44] Indrė Žliobaitė, Jorn Bakker, Mykola Pechenizkiy, “Beating the baseline prediction in food sales: How intelligent an intelligent predictor is?”, Expert Systems with Applications, Volume 39, Issue 1, 2012, Pages 806-815, ISSN 0957-4174. https://doi.org/10.1016/j.eswa.2011.07.078

[45] Ullah, Fath U M.; Khan, Noman; Hussain, Tanveer; Lee, Mi Y.; Baik, Sung W. 2021. "Diving Deep into Short-Term Electricity Load Forecasting: Comparative Analysis and a Novel Framework" *Mathematics* 9, no. 6: 611. https://doi.org/10.3390/math9060611

[46] Lakshman Krishnamurthi, Purushottam Papatla, “Accounting for heterogeneity and dynamics in the loyalty–price sensitivity relationship”, Journal of Retailing, Volume 79, Issue 2, 2003, Pages 121-135, ISSN 0022-4359. https://doi.org/10.1016/S0022-4359(03)00010-1

[47] Esteban Casado, Juan-Carlos Ferrer, "Consumer price sensitivity in the retail industry: Latitude of acceptance with heterogeneous demand”, European Journal of Operational Research, Volume 228, Issue 2, 2013, Pages 418-426, ISSN 0377-2217. https://doi.org/10.1016/j.ejor.2013.01.010

[48] Aurélien Géron, “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow”, 2nd Edition, 2019, O’Reilly Media, Inc., ISBN: 9781492032649. https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/

[49] Saptashwa Bhattacharyya, “Ridge and Lasso Regression: L1 and L2 Regularization”, https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b

[50] Jason Brownlee, “A Gentle Introduction to Mixture of Experts Ensembles”. https://machinelearningmastery.com/mixture-of-experts/

[51] Will Kenton, Gordon Scott, “Fast-Moving Consumer Goods (FMCGs), Investopedia, 2021. https://www.investopedia.com/terms/f/fastmoving-consumer-goods-fmcg.asp

[52] Andreas C. Müller, Sarah Guido, “Introduction to Machine Learning with Python”, O’Reilly Media, Inc., 2020. https://www.oreilly.com/catalog/errata.csp?isbn=9781449369415

[53] Donna F. Davis, John T. Mentzer, “Organizational factors in sales forecasting management”, International Journal of Forecasting, Volume 23, Issue 3, 2007, Pages 475-495, ISSN 0169-2070. https://doi.org/10.1016/j.ijforecast.2007.02.005

[54] Andriy Burkov, “The hundred-page Machine Learning Book”, 2019, http://ema.cri-info.cm/wp-content/uploads/2019/07/2019BurkovTheHundred-pageMachineLearning.pdf

[55] Oliveira Martelo Lobo da Costa, Luis Francisco, “Modeling of Sales Forecasting in Retail Using Soft Computing Techniques”, Dissertacion¸ Grau de Mestreem, Engenharia Mecanica

[56] Mark A. Moon, John T. Mentzer, Carlo D. Smith, "Conducting a sales forecasting audit”, International Journal of Forecasting, Volume 19, Issue 1, 2003, Pages 5-25, ISSN 0169-2070. https://doi.org/10.1016/S0169-2070(02)00032-8

[57] Philip Hans Franses, Rianne Legerstee, “Do statistical forecasting models for SKU-level data benefit from including past expert knowledge?”, International Journal of Forecasting, Volume 29, Issue 1, 2013, Pages 80-87, ISSN 0169-2070. https://doi.org/10.1016/j.ijforecast.2012.05.008

[58] Michael Lawrence, Marcus O'Connor, Bob Edmundson, “A field study of sales forecasting accuracy and processes”, European Journal of Operational Research, Volume 122, Issue 1, 2000, Pages 151-160, ISSN 0377-2217. https://doi.org/10.1016/S0377-2217(99)00085-5

[59] Eleonora Bottani, Antonio Rizzi, “Economical assessment of the impact of RFID technology and EPC system on the fast-moving consumer goods supply chain”, International Journal of Production Economics, Volume 112, Issue 2, 2008, Pages 548-569, ISSN 0925-5273. https://doi.org/10.1016/j.ijpe.2007.05.007

[60] Chong Liu, Wanli Xie, Wen-Ze Wu, Hegui Zhu, “Predicting Chinese total retail sales of consumer goods by employing an extended discrete grey polynomial model”, Engineering Applications of Artificial Intelligence, Volume 102, 2021, 104261, ISSN 0952-1976. https://doi.org/10.1016/j.engappai.2021.104261

[61] Lukas Morand, Dirk Helm, “A mixture of experts approach to handle ambiguities in parameter identification problems in material modeling”, Computational Materials Science, Volume 167, 2019, pg. 85-91, ISSN 0927-0256. https://doi.org/10.1016/j.commatsci.2019.04.003

[62] Chelsea Finn, Pieter Abbeel, Sergey Levine, “Proceedings of the 34th International Conference on Machine Learning”, 2017, PMLR 70:1126-1135. http://proceedings.mlr.press/v70/finn17a

[63] Philip K. Chan, Salvatore J. Stolfo, “Experiments on multistrategy learning by meta-learning”, CIKM '93: Proceedings of the second international conference on Information and knowledge management,1993, Pages 314–323. https://doi.org/10.1145/170088.170160

[64] Nicolas Schweighofer, Kenji Doya, Meta-learning in Reinforcement Learning, Neural Networks, Volume 16, Issue 1, 2003, Pages 5-9, ISSN 0893-6080. https://doi.org/10.1016/S0893-6080(02)00228-9

[65] Masoudnia, S., Ebrahimpour, R., "Mixture of experts: a literature survey", 2014, Artificial Intell Review 42, 275–293. https://doi.org/10.1007/s10462-012-9338-y

[66] Jinsung Yoon, Daniel Jarrett, Mihaela van der Schaar, **"**Time-series Generative Adversarial Networks", 2019, NeurIPS 2019. https://openreview.net/forum?id=rJeZq4reLS

[67] Ran Avnimelech, Nathan Intrator; Boosted Mixture of Experts: An Ensemble Learning Scheme. *Neural Comput* 1999; 11 (2): 483–497. https://doi.org/10.1162/089976699300016737