ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

Διπλωματική Εργασία

Πρόβλεψη επιτυχίας προώθησης τραπεζικών προϊόντων με μεθόδους μηχανικής μάθησης

Ονοματεπώνυμο : Μπότσιου Χριστίνα
Αριθμός Μητρώου: ge14012
E-mail : christinebotsiou@gmail.com
Σχολή: Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών, ΕΜΠ
Επιβλέπων : Κουσουρής Κωνσταντίνος
Ημερομηνία : 7/7/2021

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

Διπλωματική Εργασία

Πρόβλεψη επιτυχίας προώθησης τραπεζικών προϊόντων με μεθόδους μηχανικής μάθησης

Ονοματεπώνυμο : Μπότσιου Χριστίνα
Αριθμός Μητρώου: ge14012
E-mail : christinebotsiou@gmail.com
Σχολή: Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών, ΕΜΠ
Επιβλέπων : Κουσουρής Κωνσταντίνος
Ημερομηνία : 7/7/2021

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 7η Ιουλίου 2021.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
…….	…….	…….
Κωνσταντίνος Κουσουρής, Αναπληρωτής Καθηγητής Ε.Μ.Π.	Γ. Τσιπολίτης, Καθηγητής Ε.Μ.Π.	Α. Τζαμαριουδάκη, Ερευνήτρια Ά ΕΚΕΦΕ Δημόκριτος

Αθήνα, Ιούλιος 2021

(*Υπογραφή*)

……

ΜΠΟΤΣΙΟΥ ΧΡΙΣΤΙΝΑ

Διπλωματούχος Εφαρμοσμένων Μαθηματικών & Φυσικών Επιστημών

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

"*A breakthrough in machine learning would be worth 10 Microsofts*"

~Bill Gates

Ευχαριστίες

Θα ήθελα πραγματικά να ευχαριστήσω τον καθηγητή μου, τον Δρ. Κωνσταντίνο Κουσουρή, για την εμπιστοσύνη που έδειξε αναθέτοντας μου την εκπόνηση της συγκεκριμένης διπλωματικής εργασίας. Κυρίως όμως θέλω να εκφράσω την εκτίμηση μου για την υποστήριξη που μου έδωσε, καθώς διανύσαμε μια δύσκολη, με προκλήσεις περίοδο λόγω της πανδημίας Covid19.

Φυσικά, δεν μπορώ να παραλείψω να ευχαριστήσω την οικογένεια μου & τους φίλους μου, που ήταν και είναι πάντα δίπλα μου σε κάθε εμπόδιο μεγάλο & μικρό. Είμαι ευγνώμον για την στήριξη τους σε κάθε μάθημα, εξεταστική, ψυχολογική δυσκολία και φυσικά σ' αυτή τη διπλωματική εργασία.

Η εργασία είναι αφιερωμένη στον μικρό μου αδερφό, τον Κωνσταντίνο.

Περίληψη

Η «έκρηξη» δεδομένων στη σύγχρονη εποχή σε συνδυασμό με την ανάγκη εξόρυξης χρήσιμων πληροφοριών από αυτά, έχουν συμβάλει στην εξέλιξη της επιστήμης των δεδομένων (data science) και της μηχανικής μάθησης (machine learning). Η ανάγκη ταξινόμησης αντικειμένων οδήγησε στην ανάπτυξη υπολογιστικών μοντέλων μηχανικής μάθησης, τα οποία επιτελούν λήψη αποφάσεων βασιζόμενα σε εμπειρικά δεδομένα, γνωστά και ως ταξινομητές.

Η παρούσα διπλωματική εργασία έχει ως αντικείμενο ένα πρόβλημα ταξινόμησης και παρουσιάζονται μέθοδοι μηχανικής μάθησης που επιλύουν το πρόβλημα. Κατασκευάστηκαν γραμμικά και μη γραμμικά μοντέλα με σκοπό την πρόβλεψη για το αν ένας πελάτης ενός πορτογαλικού τραπεζικού ιδρύματος θα ανοίξει προθεσμιακή κατάθεση ή όχι. Χρησιμοποιήθηκαν μέθοδοι ενδυνάμωσης, νευρωνικά δίκτυα και linear discriminant analysis. Το σύνολο δεδομένων προς επεξεργασία αποτελείται από χαρακτηριστικά των πελατών, όπως ηλικία φύλο εκπαίδευση, αλλά και οικονομικούς δείκτες.

Για την επεξεργασία των δεδομένων και την εφαρμογή των μεθόδων χρησιμοποιήθηκε η γλώσσα προγραμματισμού *Python.* Αρχικά πραγματοποιήθηκε η επεξεργασία των δεδομένων για να είναι σε κατάλληλη μορφή και στη συνέχεια με τις αντίστοιχες βιβλιοθήκες που παρέχει η Python, έγινε υλοποίηση των μεθόδων.

Λέξεις κλειδιά

Μηχανική Μάθηση, Ανομοιογενή Δεδομένα ,Μέθοδοι ενδυνάμωσης, Νευρωνικά Δίκτυα, Ταξινόμηση , Δέντρα απόφασης, Ενδυναμωμένα δέντρα απόφασης, Τεχνική Μείωσης Διαστάσεων

Abstract

The "explosion" of data in modern times combined with the need to extract useful information from them, have contributed to the evolution of data science and machine learning. The need to classify objects has led to the development of computer machine learning models, which make decisions based on empirical data, also known as classifiers.

The present dissertation deals with a classification problem and presents machine learning methods that solve the problem. Linear and non-linear models were constructed to predict whether a customer of a Portuguese banking institution would open a time deposit or not. Boosting trees, neural networks and linear discriminant analysis were used in order to accomplish this task. The dataset consists of customer characteristics such as age, sex, education, but also financial indicators.

Python (programming language) was used in order to to process the data and apply the methods. Initially the data was processed to be in a suitable format and then with the corresponding libraries provided by Python, the methods were implemented.

Keywords

Machine Learning, Imbalanced Data, Boosting techniques, Neural Networks, Classification, Decision Trees, Boosting Decision Trees, Dimensionality Reduction Techniques

Περιεχόμενα

Κεφάλαιο 1: Θεωρία Μηχανικής Μάθησης

1.1 Μηχανική Μάθηση & Κατηγορίες

Μηχανική Μάθηση

Η μηχανική μάθηση αποτελεί κομμάτι της επιστήμης των υπολογιστών, και αναπτύχθηκε από τη μελέτη της αναγνώρισης προτύπων και της υπολογιστικής θεωρίας μάθησης στην τεχνητή νοημοσύνη. Ο Arthur Samuel το 1959 όρισε τη μηχανική μάθηση ως "*Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί*".

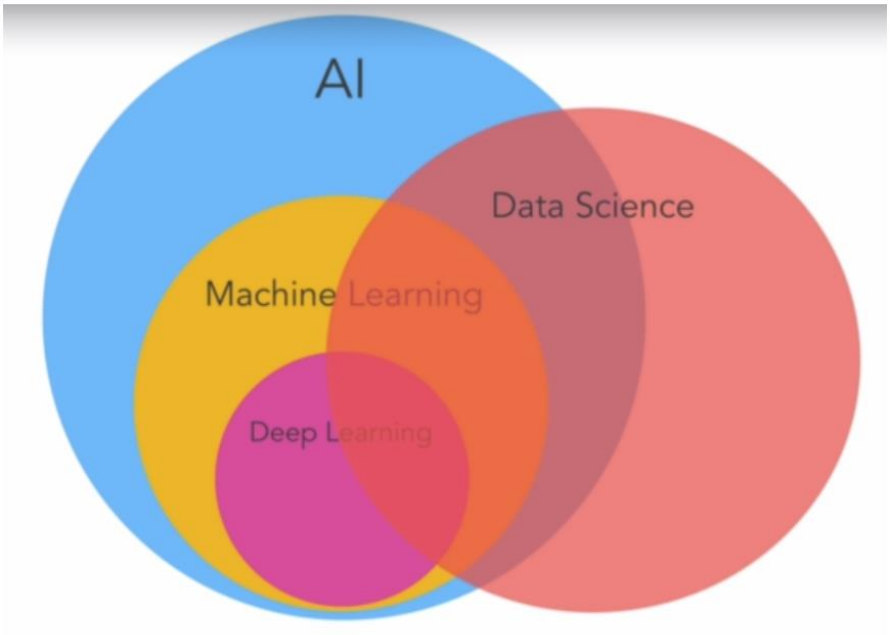
Η μηχανική μάθηση μελετά και κατασκευάζει αλγορίθμους που εκπαιδεύουν/ μαθαίνουν από τα δεδομένα και στο τέλος κάνουν προβλέψεις σχετικά με αυτά. Αυτοί οι αλγόριθμοι κατασκευάζουν μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις βασιζόμενες στα δεδομένα ή να εξάγουν αποφάσεις που εκφράζονται ως το αποτέλεσμα.

Τεχνητή Νοημοσύνη

Είναι σημαντικό να αναφερθεί πως η μηχανική μάθηση αποτελεί κομμάτι της τεχνητής νοημοσύνης. Από τη στιγμή που η τεχνολογία εισχώρησε στο κόσμο, οι άνθρωποι επιδιώκουν να εντάξουν την αυτοματοποίηση σε όλο και περισσότερα κομμάτια της ζωής μας. Η τεχνητή νοημοσύνη κάνει τις μηχανές να σκέφτονται χωρίς ανθρώπινη παρέμβαση.

Βαθιά Μάθηση

Η βαθιά μάθηση αποτελεί υποσύνολο των δύο παραπάνω και έχει εμπνευστεί από τον τρόπο που ο ανθρώπινος εγκέφαλος φιλτράρει πληροφορίες. Τα συστήματα βαθιάς μάθησης επιτρέπουν σε ένα υπολογιστικό μοντέλο να φιλτράρει τα input δεδομένα μέσω στρωμάτων (*layers*) με σκοπό τη πρόβλεψη & ταξινόμηση πληροφοριών. Τα Convolutional Neural Networks, Recurrent Neural Networks και Recursive Neural Networks συγκαταλέγονται στις αρχιτεκτονικές δικτύου βαθιάς μάθησης.



(**Πηγή**: "*https://lotuslabs.medium.com/clarifying-ai-machine-learning-deep-learning-data-science-with-venn-diagrams-c94198faa063*")

Τύποι Μηχανικής Μάθησης

Η Μηχανικής Μάθηση, ανάλογα με τον τρόπο μάθησης, χωρίζεται σε τρεις κατηγορίες:

1. *Επιβλεπόμενη Μάθηση (Supervised Learning)*

Ουσιαστικά, ένας αλγόριθμος επιβλεπόμενης μάθησης μαθαίνει από το ένα σύνολο εκπαίδευσης (βλ. 2.3.3) που εμπεριέχει και την αντίστοιχη ετικέτα, με σκοπό να γενικεύσει αυτή τη διαδικασία και για άγνωστα δεδομένα.

Η επιβλεπόμενη μάθηση συναντάται σε προβλήματα:

- *Ταξινόμησης (Classification)*: Κατά την εκπαίδευση του αλγορίθμου τα δεδομένα εισόδου αντιπροσωπεύονται από μία μοναδική τιμή κλάσης και ο αλγόριθμος καλείται να κατασκευάσει ένα μοντέλο που τα ταξινομήσει στις κλάσεις. Ένα από τα πιο κλασσικά real life problems ταξινόμησης αποτελούν τα φίλτρα Spam των emails, που καλούνται να ταξινομήσουν τα τελευταία σε «spam» και «όχι spam».

- *Παλινδρόμησης (Regression)*: Τα μοντέλα παλινδρόμησης προβλέπουν μία συνεχή και όχι διακριτή τιμή. Η τιμή εξόδου ονομάζεται μεταβλητή απόκρισης.

2. *Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)*

Στη μη επιβλεπόμενη μάθηση, το μοντέλο εκπαιδεύεται γνωρίζοντας μόνο τις μεταβλητές εισόδου. Έχει ονομαστεί μη επιβλεπόμενη γιατί σε αντίθεση με την επιβλεπόμενη, δεν έχει ετικέτες δεδομένων για να τα κατατάξει, αλλά προσπαθεί να «μάθει» από αυτά ώστε να τα μοντελοποιήσει τη δομή τους ή την κατανομή τους.

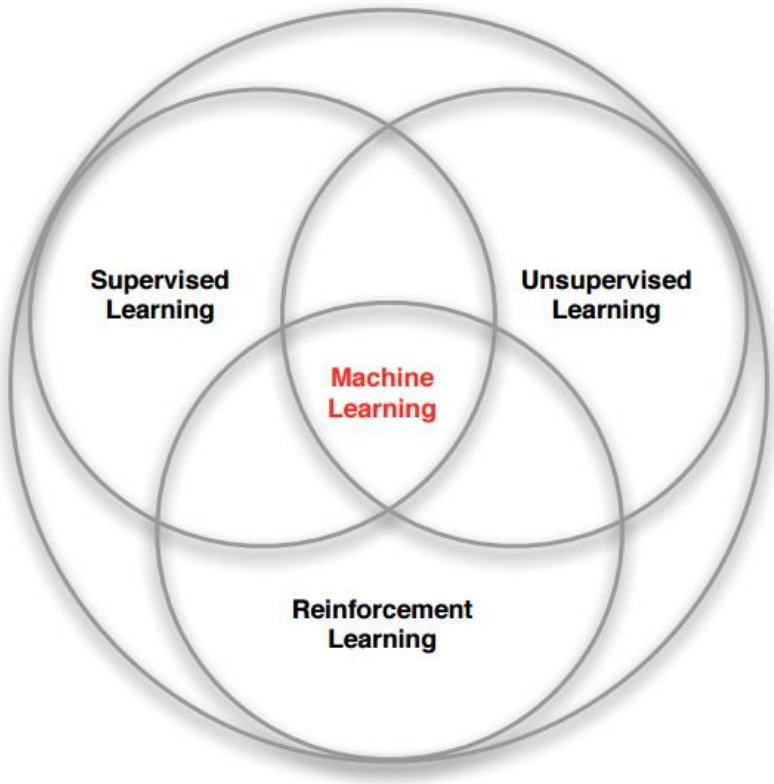
Η μη επιβλεπόμενη μάθηση συναντάται σε προβλήματα:

- *Ομαδοποίησης (Clustering)*: Το μοντέλο διαχωρίζει τα δεδομένα με βάση τις ομοιότητες τους. Αποτελεί κατηγορία της μη επιβλεπόμενης μάθησης, καθώς δεν υπάρχει προγενέστερη γνώση για το κριτήριο διαχωρισμού σε ομάδες.

- *Ανάλυσης Συσχετισμών (Association Analysis)*: Ανιχνεύουν συσχετίσεις σε μεγάλα σύνολα δεδομένων.

3. *Ενισχυτική Μάθηση (Reinforcement Learning)*:

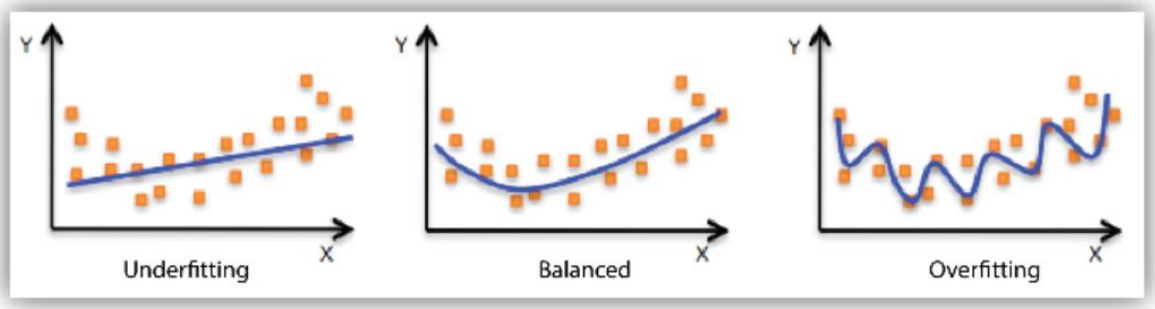
Ο αλγόριθμος μαθαίνει μια στρατηγική ενεργειών μέσα από άμεση αλληλεπίδραση με το περιβάλλον. Ένα παράδειγμα είναι η αυτόματη οδήγηση οχήματος ή να παίζει ένα παιχνίδι εναντίον ενός αντιπάλου.



(Πηγή: *https://gr.pinterest.com/pin/219761656794012482/*)

1.2 Υπερεκπαίδευση

Ένα από τα πιο συνήθης προβλήματα στους αλγορίθμους επιβλεπόμενης μάθησης είναι η *υπερεκπαίδευση (overfitting)* και η *υποεκπαίδευση (underfitting).* Η υπερεκπαίδευση συμβαίνει όταν το μοντέλο έχει προσαρμοστεί τέλεια στο σύνολο εκπαίδευσης και αποτυγχάνει να έχει καλή απόδοση σε καινούργια δεδομένα. Από την άλλη, η υποεκπαίδευση αφορά ένα μοντέλο που δεν μπορεί ούτε να προσαρμοστεί στο σύνολο εκπαίδευσης ούτε να γενικευθεί σε καινούργια δεδομένα. Συμβαίνει όταν ένα μοντέλο είναι πολύ απλό, δηλαδή βασίζεται σε πολύ λίγες μεταβλητές ή δεν υπάρχει συσχέτιση μεταξύ αυτών και της μεταβλητής εξόδου, με αποτέλεσμα να μην έχει την ικανότητα να εκπαιδευθεί.



(Πηγή: *https://mrrajeshrai.com/2019/06/16/model-fitting-overfitting-underfitting-and-balanced/)*

Κεφάλαιο 2: Περιγραφή προβλήματος, Επεξεργασία & Ανάλυση Δεδομένων

2.1 Περιγραφή Προβλήματος & Δεδομένων

Οι καμπάνιες marketing με στόχο τις πωλήσεις αποτελούν μια συχνή στρατηγική ενίσχυσης των επιχειρήσεων. Οι εταιρείες χρησιμοποιούν άμεσο marketing με σκοπό τη στόχευση συγκεκριμένων πληθυσμών/πελατών, επικοινωνώντας μαζί τους για την επίτευξη ενός συγκεκριμένου στόχου. Το marketing που στηρίζεται σε μία βάση επαφών (contact center) ονομάζεται telemarketing λόγω του χαρακτηριστικού της εξ αποστάσεως επικοινωνίας.

Επιπλέον, τα συστήματα υποστήριξης αποφάσεων (DSS) χρησιμοποιούν την τεχνολογία πληροφοριών για την υποστήριξη της διαχείρισης αποφάσεων. Υπάρχουν δύο είδη DSS, τα personal & intelligence DSS. Το τελευταίο χρησιμοποιεί τεχνικές τεχνητής νοημοσύνης (artificial intelligence) για τη λήψη αποφάσεων.

Το συγκεκριμένο πρόβλημα είναι *Classification Problem* και σκοπός είναι να δημιουργηθούν μοντέλα πρόβλεψης των κλάσεων. Ειδικότερα, η πρόβλεψη αφορά το αν ένας πελάτης τράπεζας θα ανοίξει προθεσμιακή κατάθεση ή όχι, με βάση κάποια χαρακτηριστικά/ μεταβλητές που δίνονται. Τα δεδομένα αυτά βασίζονται σε τηλεφωνικές κλήσεις πελατών, από το 2008 έως το 2013, διαφημιστικής καμπάνιας ενός πορτογαλικού τραπεζικού ιδρύματος. Επιπλέον, πολλές φορές χρειάστηκε να επαναληφθούν οι κλήσεις στους ίδιους πελάτες για να οριστικοποιηθεί η κλάση στην οποία ανήκουν.

Η πηγή εύρεσης των δεδομένων είναι το αποθετήριο του ιστότοπου για Machine Learning & Intelligent Systems του UCI (University of California, Irvine). Το πλήθος δεδομένων είναι 41.188.

Δεδομένα : https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

2.2 Σχολιασμός Μεταβλητών

Οι μεταβλητές των δεδομένων είναι συνολικά 20, χωρίς να έχει συμπεριληφθεί η μεταβλητή που καθορίζει την κλάση. Οι 10 από αυτές είναι αριθμητικές, ενώ οι υπόλοιπες κατηγορικές.

- **Age** : ηλικία πελάτη *(αριθμητική μεταβλητή)*
- **Job** : τύπος επαγγέλματος *(κατηγορική μεταβλητή*
- **Marital** : οικογενειακή κατάσταση *(κατηγορική μεταβλητή)*
- **Education** : εκπαιδευτικό επίπεδο *(κατηγορική μεταβλητή)*
- **Default** : έχει ασφάλιστρα κινδύνου; *(κατηγορική μεταβλητή)*
- **Housing** : έχει στεγαστικό δάνειο; *(κατηγορική μεταβλητή)*
- **Loan** : έχει προσωπικό δάνειο; *(κατηγορική μεταβλητή)*
- **Contact**: τρόπος επικοινωνίας (κινητό ή σταθερό) *(κατηγορική μεταβλητή)*
- **Month** : μήνας τελευταίας επικοινωνίας *(κατηγορική μεταβλητή)*
- **Day_of_week**: ημέρα τελευταίας επικοινωνίας *(κατηγορική μεταβλητή)*
- **Duration** : διάρκεια τηλεφωνικής κλήσης *(αριθμητική μεταβλητή)*
- **Campaign** : συνολικό πλήθος τηλεφωνημάτων της συγκεκριμένης διαφημιστικής καμπάνιας ανά πελάτη *(αριθμητική μεταβλητή)*
- **Pdays** : πλήθος ημερών που πέρασαν από τότε που υπήρξε επικοινωνία με τον πελάτη από προηγούμενη καμπάνια *(αριθμητική μεταβλητή)*
- **Previous**: πλήθος τηλεφωνημάτων που έγιναν πριν τη συγκεκριμένη καμπάνια *(αριθμητική μεταβλητή)*
- **Poutcome**: αποτέλεσμα προηγούμενης καμπάνιας *(κατηγορική μεταβλητή)*
- **Emp.var.rate** : Employment Variation Rate (τρίμηνος δείκτης) *(αριθμητική μεταβλητή)*
- **Cons.price.idx** : Μηνιαίος Δείκτης Τιμών Καταναλωτή *(αριθμητική μεταβλητή)*
- **Cons.conf.idx** : Μηνιαίος Δείκτης Εμπιστοσύνης Καταναλωτών *(αριθμητική μεταβλητή)*
- **Euribor.3m** : Ημερήσιος Δείκτης Euribor- 3month rate *(αριθμητική μεταβλητή)*
- **Nr.employed** : πλήθος εργαζομένων (τρίμηνος δείκτης) *(αριθμητική μεταβλητή)*

2.3 Σύνολο Εκπαίδευσης & Αξιολόγησης

Πριν τον έλεγχο απόδοσης ενός μοντέλου, είναι σημαντικό να έχει δοθεί σε αυτό σύνολο δεδομένων, που θα διαφέρει από τα δεδομένα στα οποία θα δοκιμαστεί. Δηλαδή, αν δημιουργηθεί ένα μοντέλο βασισμένο σε ένα συγκεκριμένο dataset και έχει υψηλό accuracy, πρέπει κάπως να ελεγχθεί η αποδοτικότητα του και σε κάποιο διαφορετικό dataset.

Ένας συνηθισμένος τρόπος δημιουργίας έξτρα δεδομένων είναι μέσω της αποκοπής ενός τυχαίου τμήματος των συνολικών δεδομένων. Έτσι, χρησιμοποιείται το μεγαλύτερο μέρος των δεδομένων για την εκπαίδευση του μοντέλου (*training set*) και το υπόλοιπο αποκομμένο σύνολο δεδομένων για την αξιολόγηση του μοντέλου (*test set*).

Συνηθίζεται το Training Set να υπερέχει του Test Set, αν και σε ορισμένες περιπτώσεις είναι 50-50. Στο συγκεκριμένο πρόβλημα, το Dataset χωρίστηκε σε ποσοστά 70-30.

2.4 Προεπεξεργασία Δεδομένων

Η προεπεξεργασία των δεδομένων αποτελεί ίσως ένα από τα πιο κουραστικά και χρονοβόρα βήματα μιας εργασίας, πριν την εφαρμογή των μεθόδων μηχανικής μάθησης πάνω σε αυτά. Χωρίς «καθαρά» δεδομένα ή αν δεν είναι σε κατάλληλη μορφή, τα αποτελέσματα δεν θα είναι ποιοτικά.

2.4.1 Κανονικοποίηση Δεδομένων

Πριν την εκπαίδευση των ταξινομητών, είναι σημαντικό να γίνει κανονικοποίηση των δεδομένων. Καθώς στις διακριτές μεταβλητές εντοπίζονται είτε μεγάλες αποκλίσεις στα εύρη τιμών τους ή ακόμα και διαφορετικές μονάδες μέτρησης, μπορεί να μην γίνεται ίδια συνεισφορά όλων στο fit modeling του εκάστοτε μοντέλου (πολλά από αυτά βασίζονται σε υπολογισμούς αποστάσεων) και καταλήγουν να δημιουργούν bias. Έτσι, για κάποιες μεθόδους είναι σημαντικό να τοποθετηθούν σε μια κοινή κλίμακα ώστε να μπορούν εύκολα όλα τα δεδομένα να συγκρίνονται μεταξύ τους.

Συγκεκριμένα, στην LDA και στα νευρωνικά δίκτυα εφαρμόστηκε η κανονικοποίηση μεγίστου – ελαχίστου (*Python:MinMaxScaler()*) στις αριθμητικές μεταβλητές, ώστε οι αριθμοί προς επεξεργασία να κυμαίνονται στην περιοχή $[0, 1]$. Η φόρμουλα πάνω στην οποία βασίζεται η μέθοδος είναι η εξής :

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

2.4.2 Διαχείριση Κατηγορικών Μεταβλητών

Τα κατηγορικά δεδομένα είναι μεταβλητές που περιέχουν τιμές "ετικέτας"(label values) και όχι αριθμητικές τιμές. Ο αριθμός των δυνατών τιμών/κατηγοριών που μπορούν να πάρουν συχνά περιορίζεται σε ένα σταθερό σύνολο (π.χ. female or male). Το συγκεκριμένο πρόβλημα περιέχει 10 κατηγορικές μεταβλητές, τις οποίες πρέπει να διαχειριστούμε για να έχουμε καλύτερη απόδοση των μοντέλων πρόβλεψης που θα χρησιμοποιηθούν.

Για κατηγορικές μεταβλητές όπου υπάρχει συσχέτιση μεταξύ των κατηγοριών της, η απλή κωδικοποίηση τους σε ακέραιους αριθμούς δεν είναι αρκετή (π.χ. μεταβλητή education: basic_6y →1, basic_9y →2, high_school →3, illitirate →4, professional_course →5, degree →6) . Μάλιστα, χρησιμοποιώντας αυτού του είδους την κωδικοποίηση, επιτρέπουμε στο μοντέλο να θεωρήσει μια φυσική 'σειρά προτεραιότητας' μεταξύ κατηγοριών και αυτό μπορεί να οδηγήσει σε κακή απόδοση ή απροσδόκητα αποτελέσματα.

Για να αποφευχθεί κάτι τέτοιο, μπορεί να εφαρμοστεί *one-hot encode κωδικοποίηση*. Με αυτή τη μέθοδο αφαιρείται η μεταβλητή που κωδικοποιείται και προστίθεται μια νέα δυαδική μεταβλητή για κάθε μοναδική κατηγορία της. Στην παρακάτω εικόνα φαίνεται ένα απλό παράδειγμα μετατροπής κατηγορικών δεδομένων μέσω της μεθόδου one-hot encode.

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

2.4.3 Ανομοιογενή Δεδομένα

Υπάρχει η περίπτωση ένα σύνολο δεδομένων να θεωρηθεί ανομοιογενές (imbalanced), καθώς οι κλάσεις δεν είναι ίσα κατανεμημένες, δηλαδή πολύ περισσότερα data ανήκουν στη μία κλάση, συγκριτικά με την άλλη.

Το πρόβλημα που προκύπτει είναι ότι τα ανομοιογενή δεδομένα τείνουν να παράγουν υψηλή ακρίβεια πρόβλεψης στην κλάση που υπερέχει και αντίστοιχα πολύ χαμηλή ακρίβεια για την άλλη κλάση. Αυτό συμβαίνει επειδή οι αλγόριθμοι ταξινόμησης θεωρούν ότι οι κλάσεις είναι ομοιόμορφα κατανεμημένες στο dataset και έτσι η κλάση με τα λιγότερα data συνεισφέρει ελάχιστα στον υπολογισμό σφάλματος. Συνεπώς στα προβλήματα ταξινόμησης μηχανικής μάθησης, αν δεν «εξισορροπηθεί» το σύνολο εκπαίδευσης, το μοντέλο δεν θα δουλέψει καλά και το πρόβλημα ουσιαστικά δεν θα επιλυθεί.

Στο συγκεκριμένο πρόβλημα, έχουμε imbalanced data, καθώς :

Target	**Πλήθος**
“Yes”	36548
“No”	4640

Τρόπος Επίλυσης Προβλήματος

Ένω υπάρχουν διάφοροι τρόποι ώστε να εξισορροπηθούν οι δύο κλάσεις, στο συγκεκριμένο πρόβλημα χρησιμοποιήθηκε η έτοιμη συνάρτηση της Python *make_imbalance* της *Imblearn* , η οποία ουσιαστικά διαγράφει rows της υπερέχουσας κλάσης και ανάλογα με το *sampling_strategy* του χρήστη επιστρέφει τελικά ένα balanced dataset.

Πριν την εφαρμογή της συνάρτησης, το training set χωριζόταν ως εξής:

Target	**Πλήθος**
“Yes”	25579
“No”	3252

Αφού η «υποδεέστερη» κλάση είχε 3252 rows, επιλέχθηκαν 3000 rows κάθε κλάσης, δηλ.: *sampling_strategy={0: 3000, 1: 3000}* . Οπότε, μετά την εφαρμογή της συνάρτησης, το training set ανακατανέμεται ως εξής:

Target	**Πλήθος**
“Yes”	3000
“No”	3000

Αντίστοιχα το Test set πριν την εφαρμογή της συνάρτησης:

Target	Πλήθος
"Yes"	10969
"No"	1388

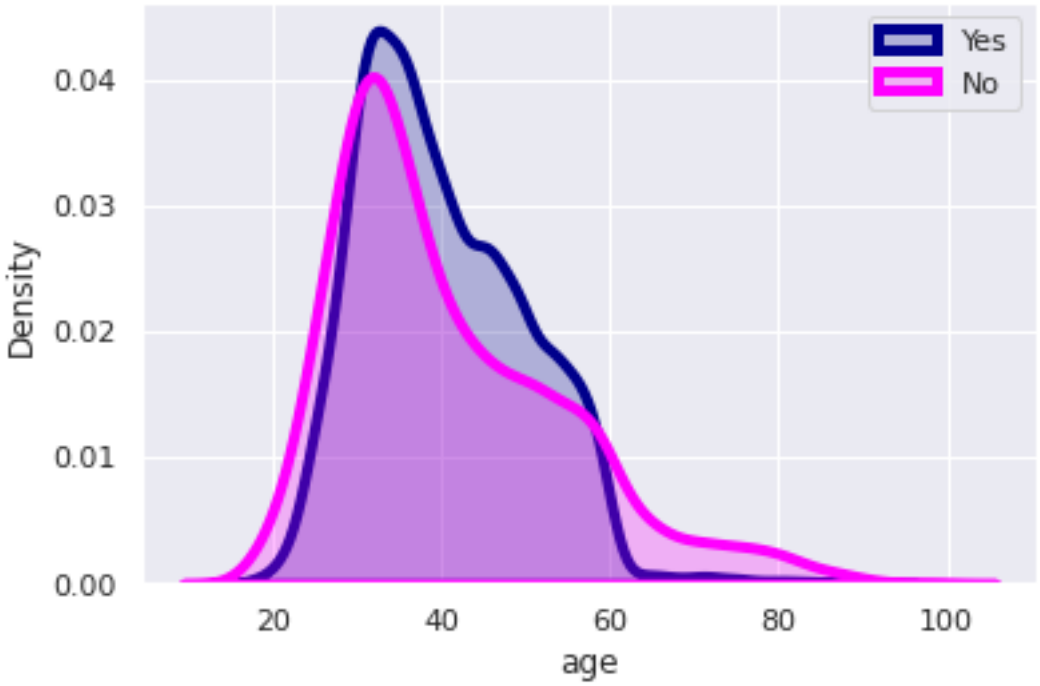
Αφού η «υποδεέστερη» κλάση είχε 1388 rows, επιλέχθηκαν 1300 rows κάθε κλάσης, δηλ.: *sampling_strategy={0: 1300, 1: 1300}* . Οπότε, μετά την εφαρμογή της συνάρτησης, το test set ανακατανέμεται ως εξής:

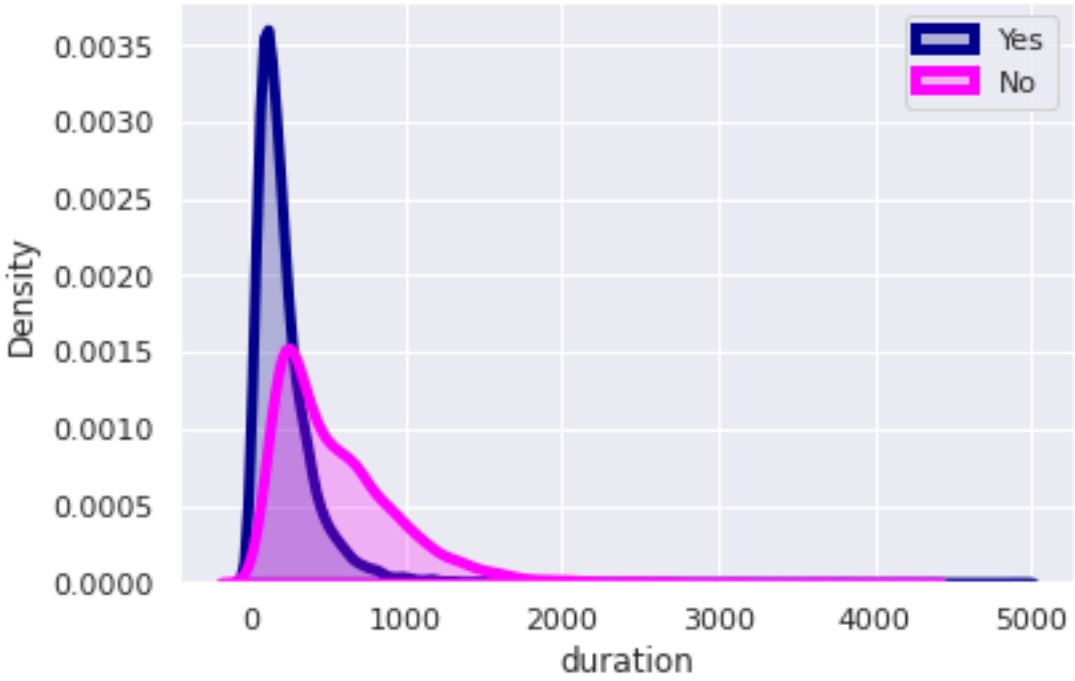
Target	Πλήθος
"Yes"	1300
"No"	1300

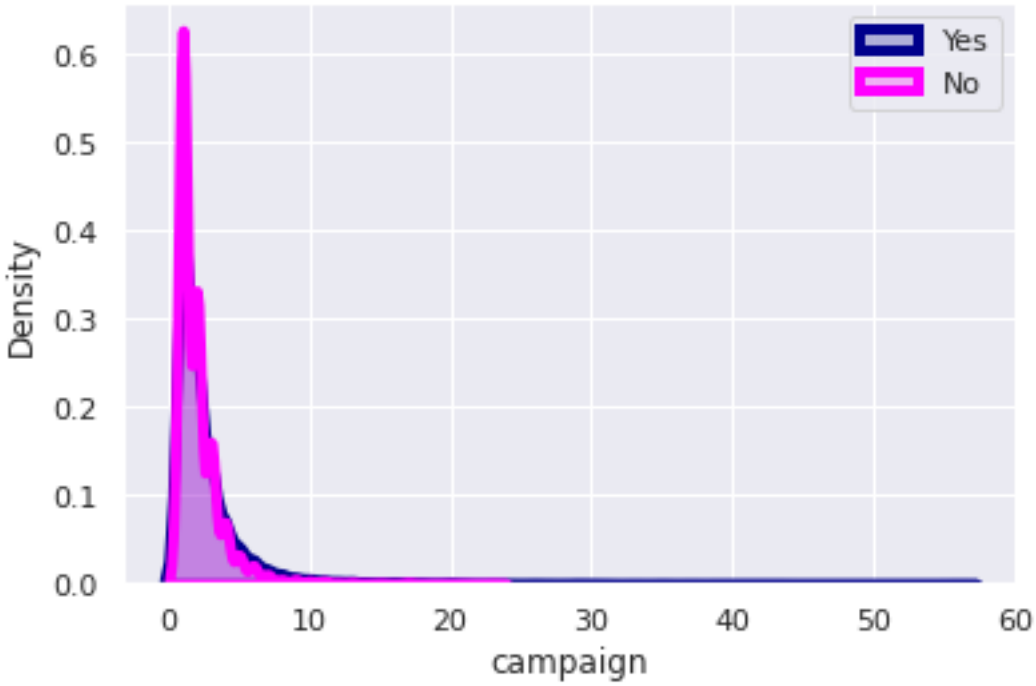
2.5 Διαχωριστική Ικανότητα Μεταβλητών

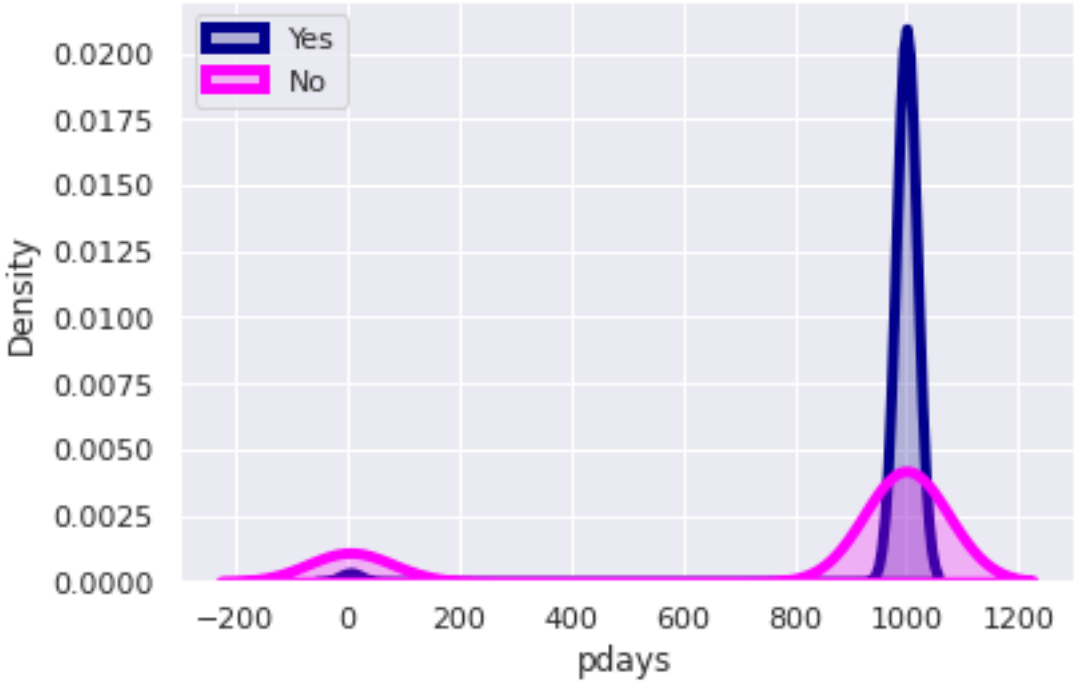
- **Γραφικές Παραστάσεις Πυκνότητας Πιθανότητας συναρτήσει Αριθμητικών Μεταβλητών**

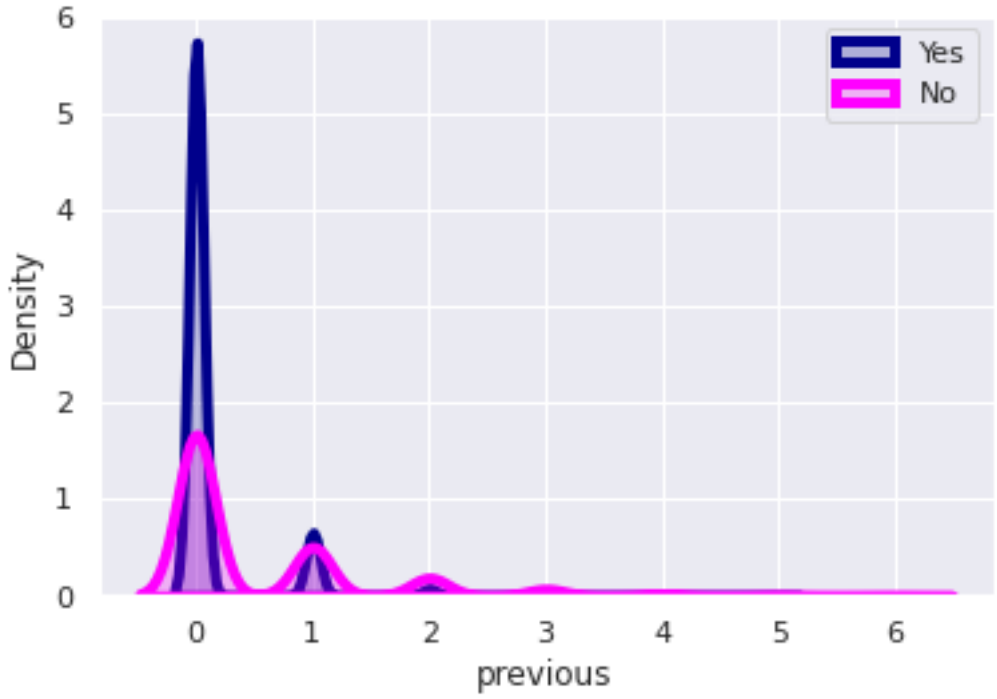
Τα Density Plots δείχνουν την κατανομή των σημείων κατά μήκος ενός αριθμητικού άξονα (ο οποίος αποτελείται από το εύρος τιμών των μεταβλητών πριν την κανονικοποίηση, ώστε να είναι ευκολότερο στον αναγνώστη να έχει αίσθηση πως κυμαίνονται οι πραγματικές τιμές των μεταβλητών). Οι κορυφές των κατανομών δείχνουν που υπάρχει η υψηλότερη συγκέντρωση σημείων ανάλογα την κλάση. Παρακάτω, φαίνονται όλες οι γραφικές παραστάσεις density plots για τις αριθμητικές μεταβλητές του προβλήματος. Φαίνεται πως οι κατανομές όλων των μεταβλητών επικαλύπτονται, δηλαδή καμία μεταβλητή δεν έχει καλή πλήρη διαχωριστική ικανότητα.

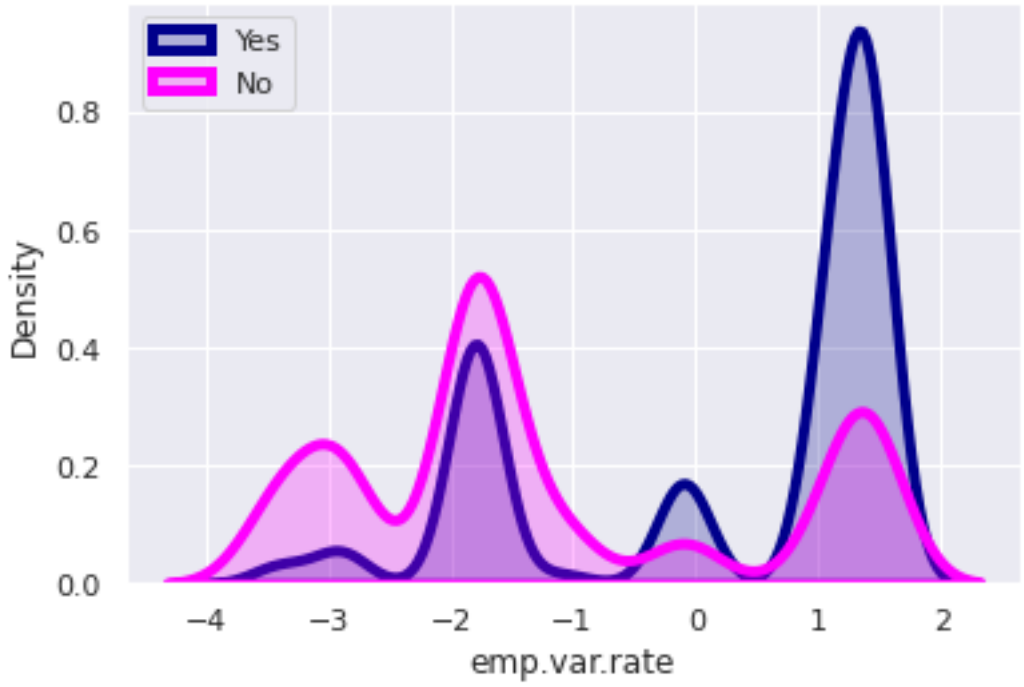

Yes
No
Density
0.04
0.03
0.02
0.01
0.00
20
40
60
80
100
age

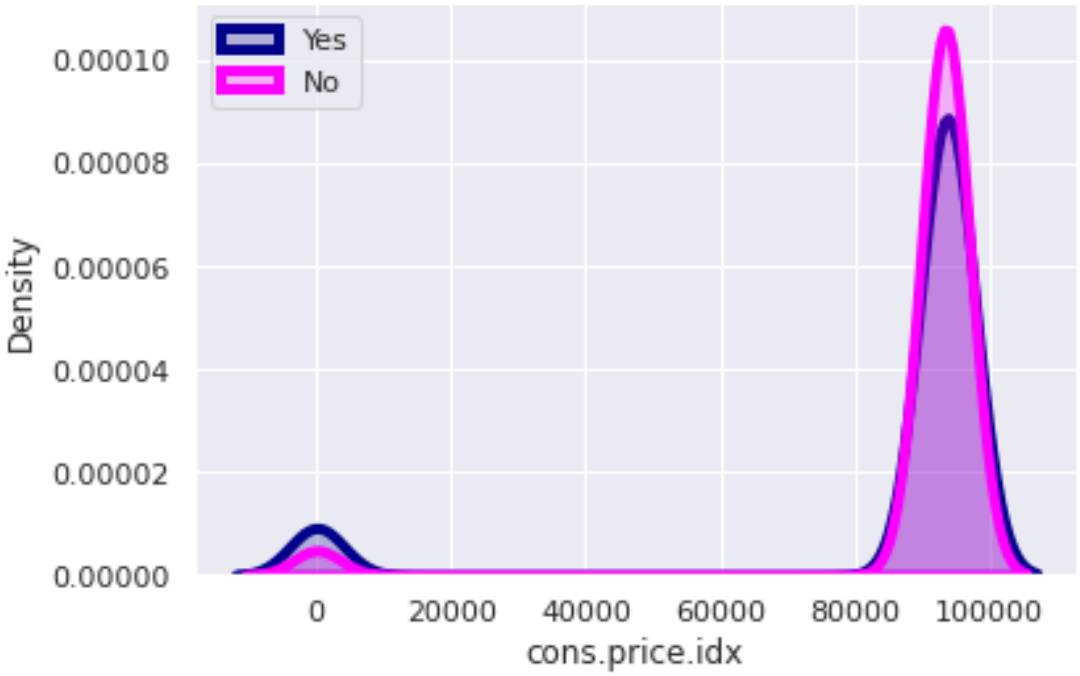

Yes
No
Density
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
1000
2000
3000
4000
5000
duration

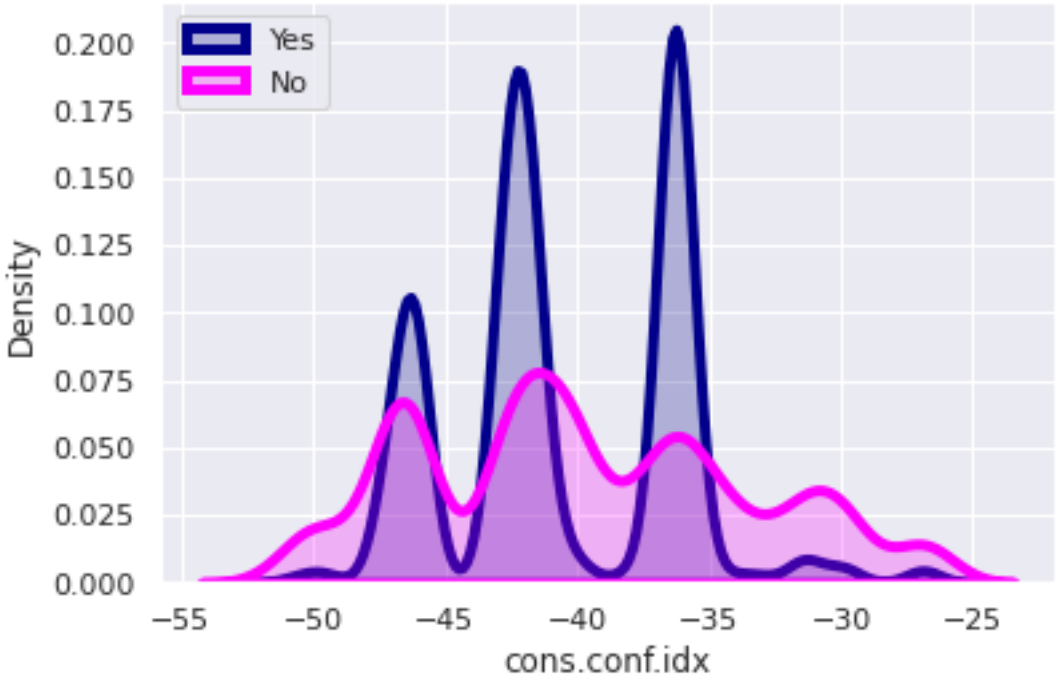

Yes
No
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
10
20
30
40
50
60
campaign

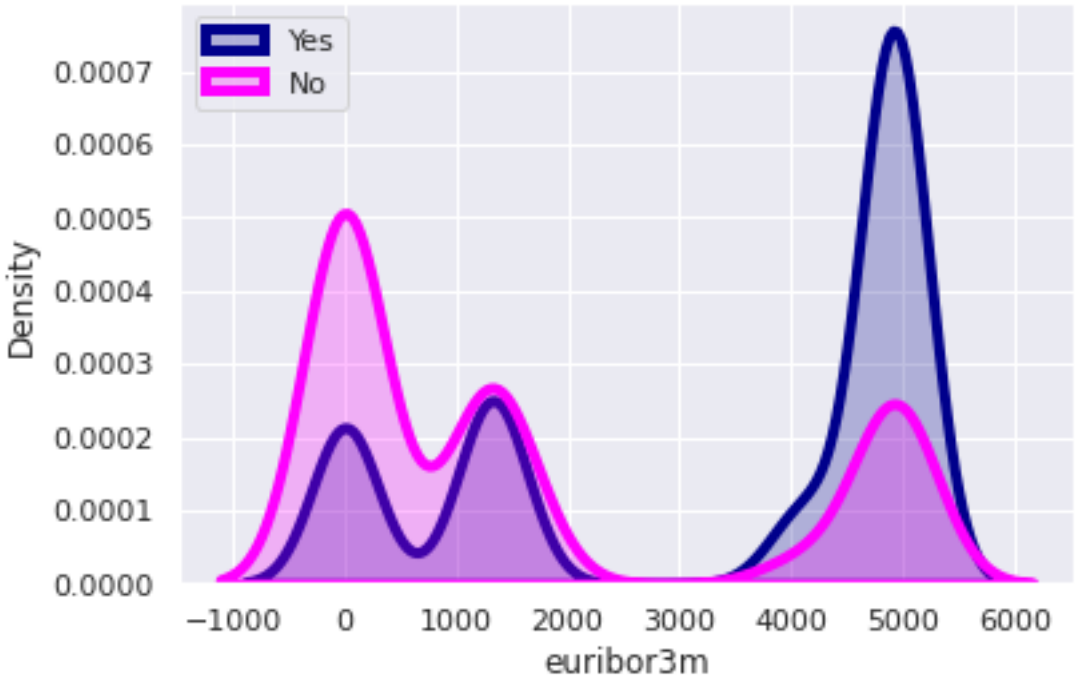

Yes
No
Density
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200
−200
0
200
400
600
800
1000
1200
pdays

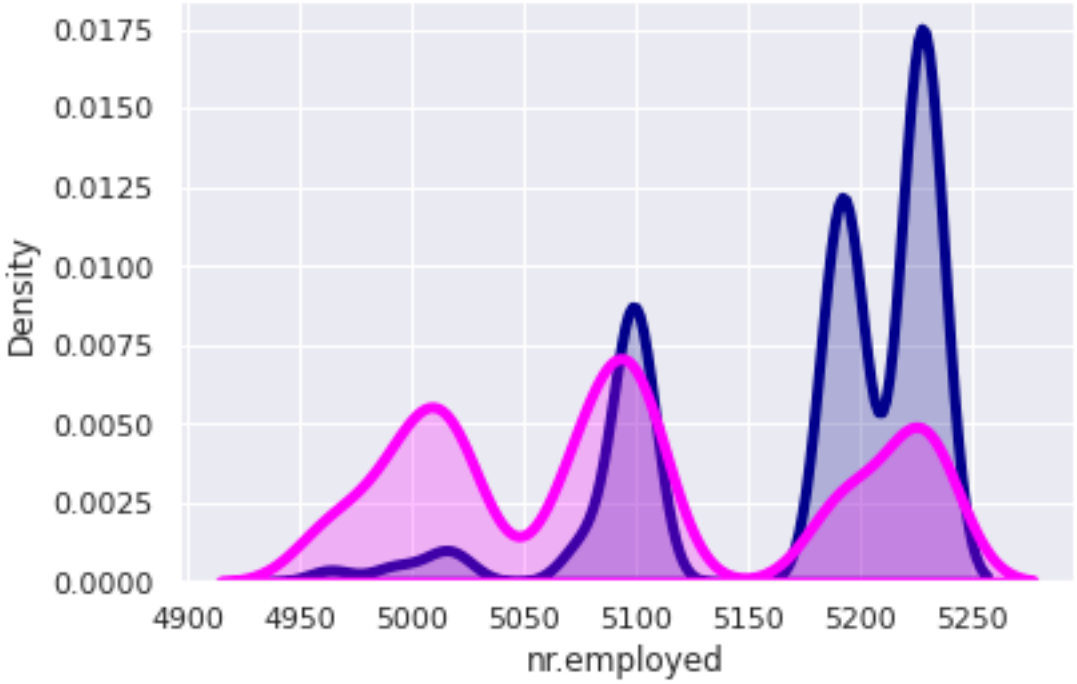

Yes
No
Density
previous
0
1
2
3
4
5
6


Yes
No
Density
emp.var.rate
0.0
0.2
0.4
0.6
0.8
−4
−3
−2
−1
0
1
2

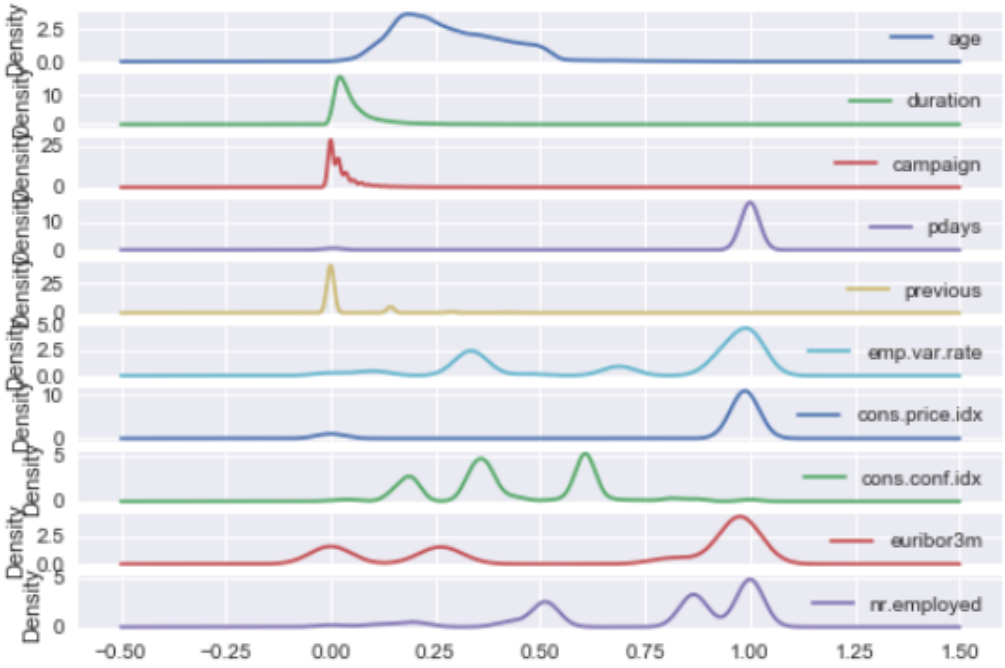

Yes
No
Density
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0
20000
40000
60000
80000
100000
cons.price.idx


Yes
No
Density
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
−55
−50
−45
−40
−35
−30
−25
cons.conf.idx


Yes
No
Density
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
−1000
0
1000
2000
3000
4000
5000
6000
euribor3m


Density
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
4900
4950
5000
5050
5100
5150
5200
5250
nr.employed

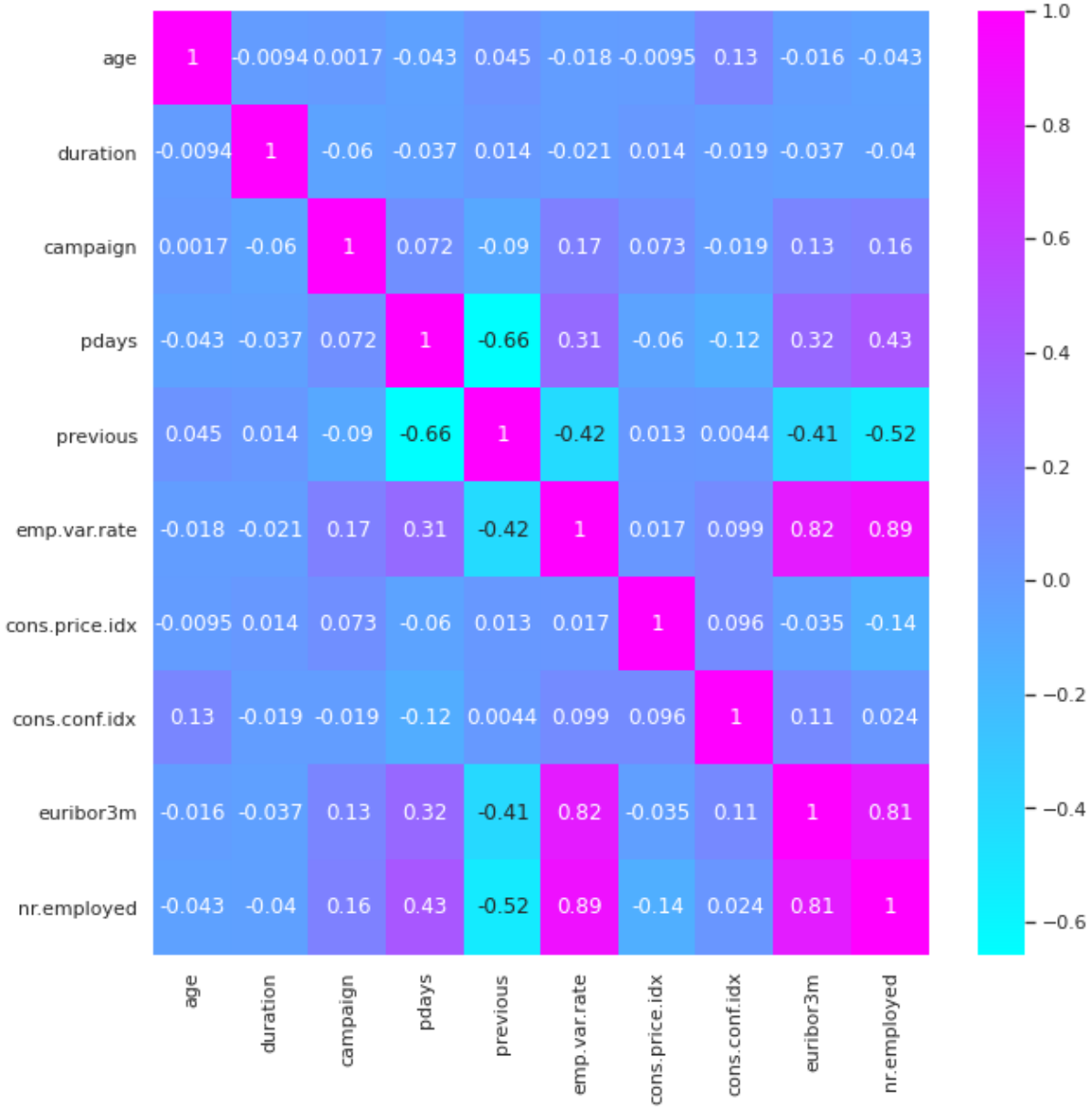
Στη συνέχεια, παρατίθεται μία εικόνα η οποία περιέχει όλες μαζί τις γραφικές παραστάσεις πυκνότητας πιθανότητας συναρτήσει των αριθμητικών μεταβλητών για ολόκληρο το κανονικοποιημένο dataset σε ένα διάγραμμα:



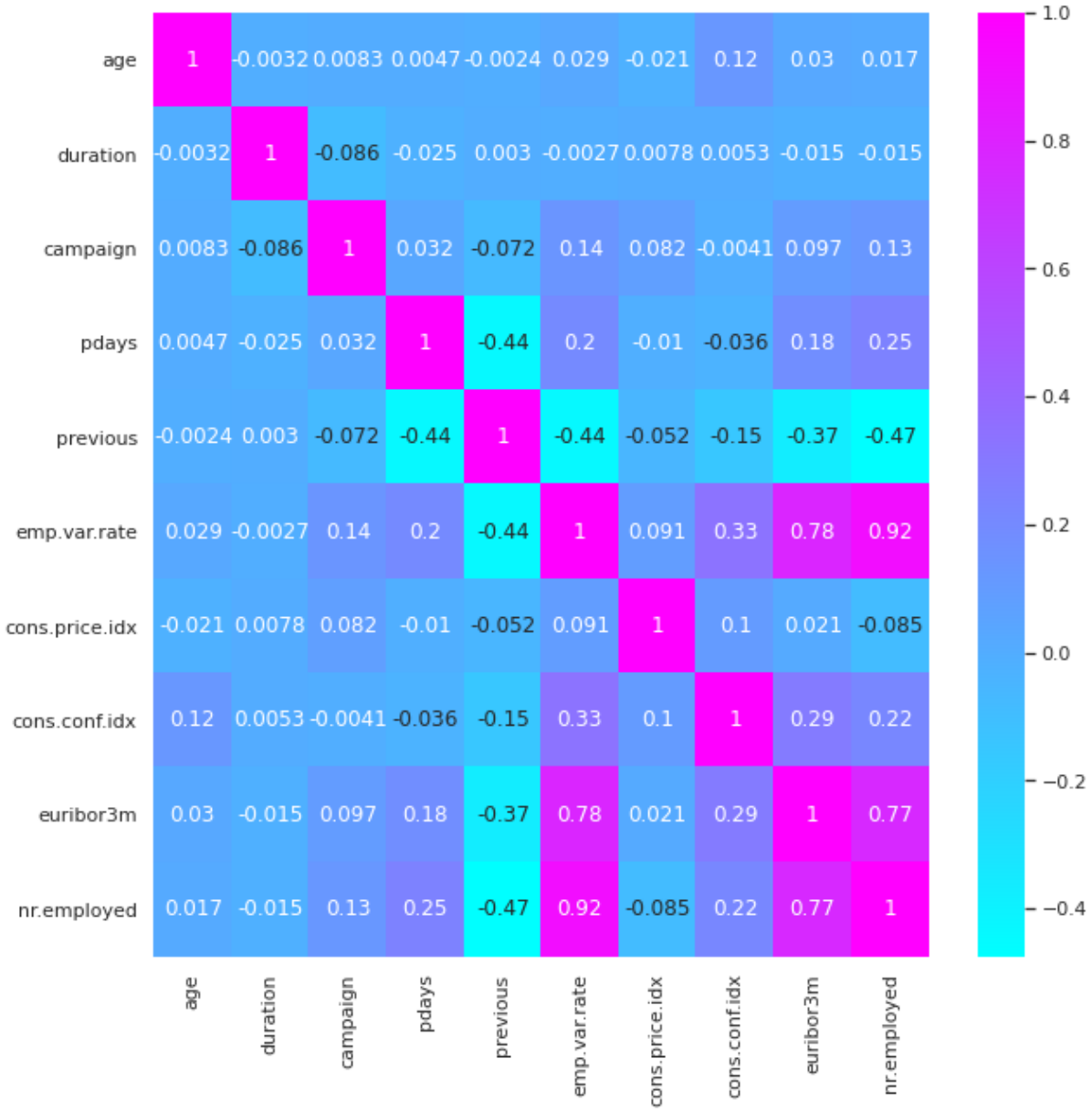
2.6 Συσχετίσεις Μεταβλητών

Πριν προχωρήσουμε στα μοντέλα μηχανικής μάθησης, θα μελετηθεί η συσχέτιση των μεταβλητών. Παρακάτω φαίνονται οι **πίνακες συσχετίσεων** (*correlation matrix*) για κάθε κλάση ξεχωριστά καθώς και για ολόκληρο το δείγμα. Στην κύρια διαγώνιο έχει μονάδες και στα κελιά εκτός της κυρίας διαγωνίου φαίνονται οι συντελεστές γραμμικής συσχέτισης για όλες τις παρατηρήσεις οι οποίοι δείχνουν την συσχέτιση των ζευγών των μεταβλητών.

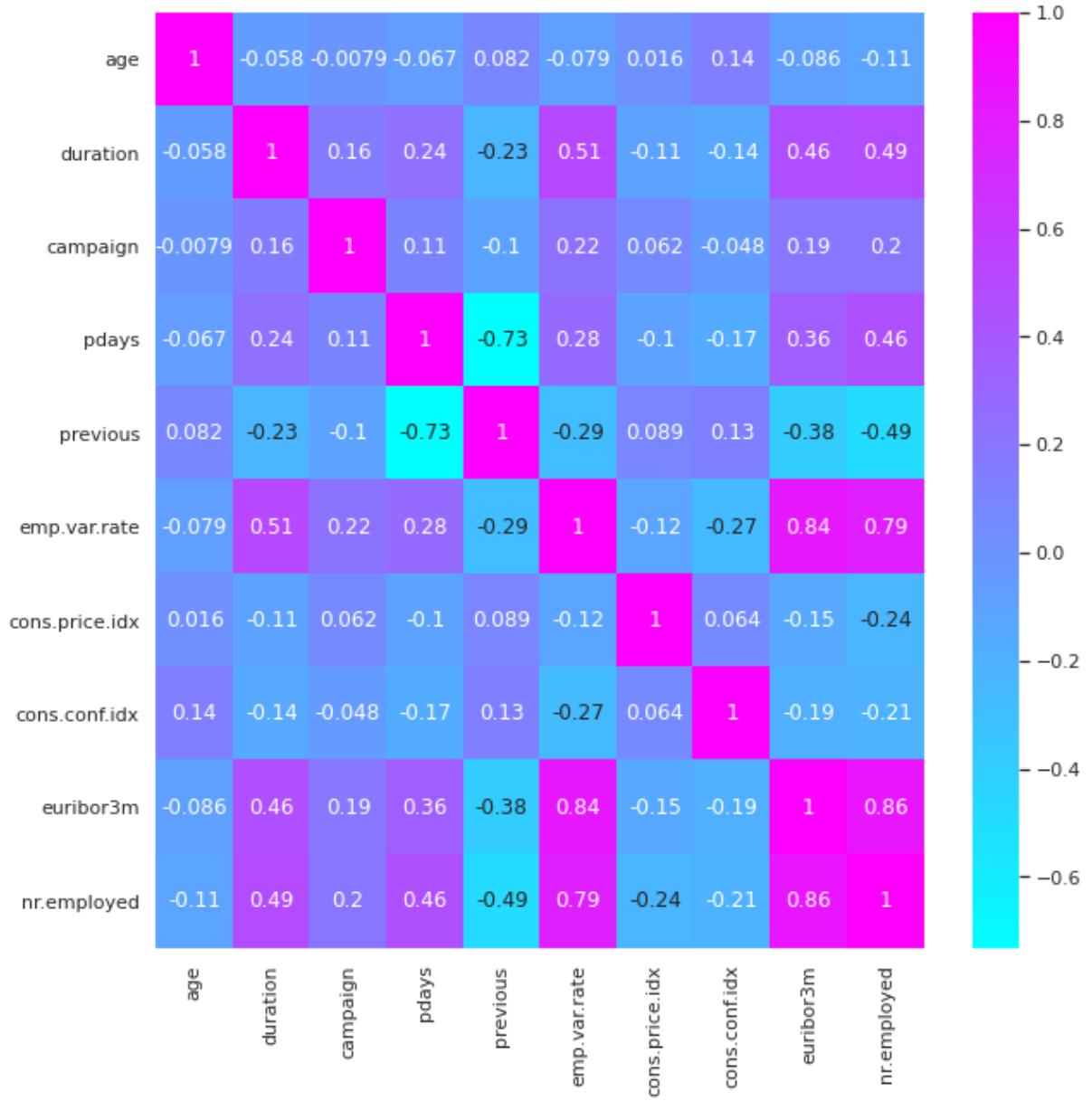
- **Για τα numerical variables ολόκληρου του δείγματος :**



- **Για τα numerical variables της κλάσης "Yes":**

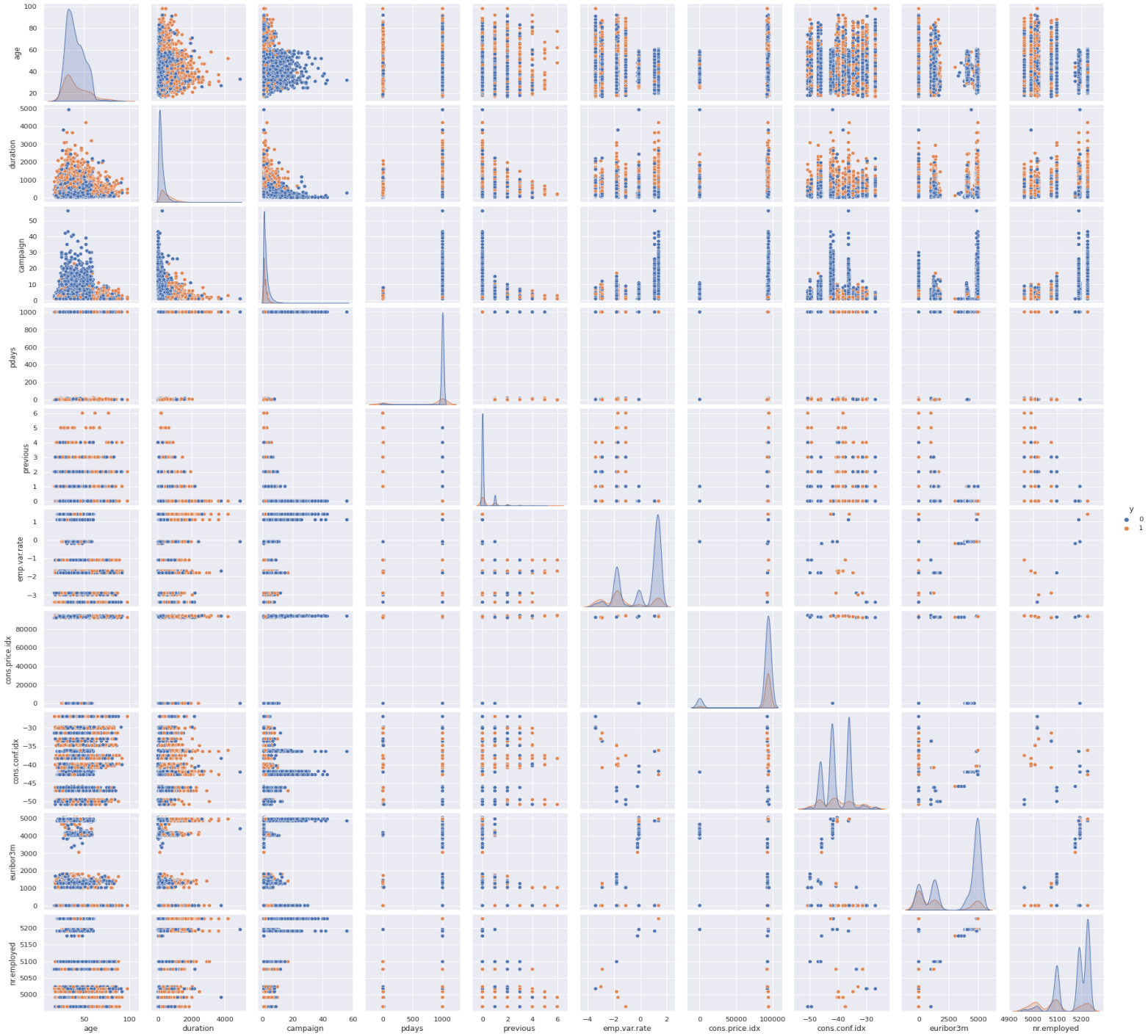


- **Για τα numerical variables της κλάσης "No":**



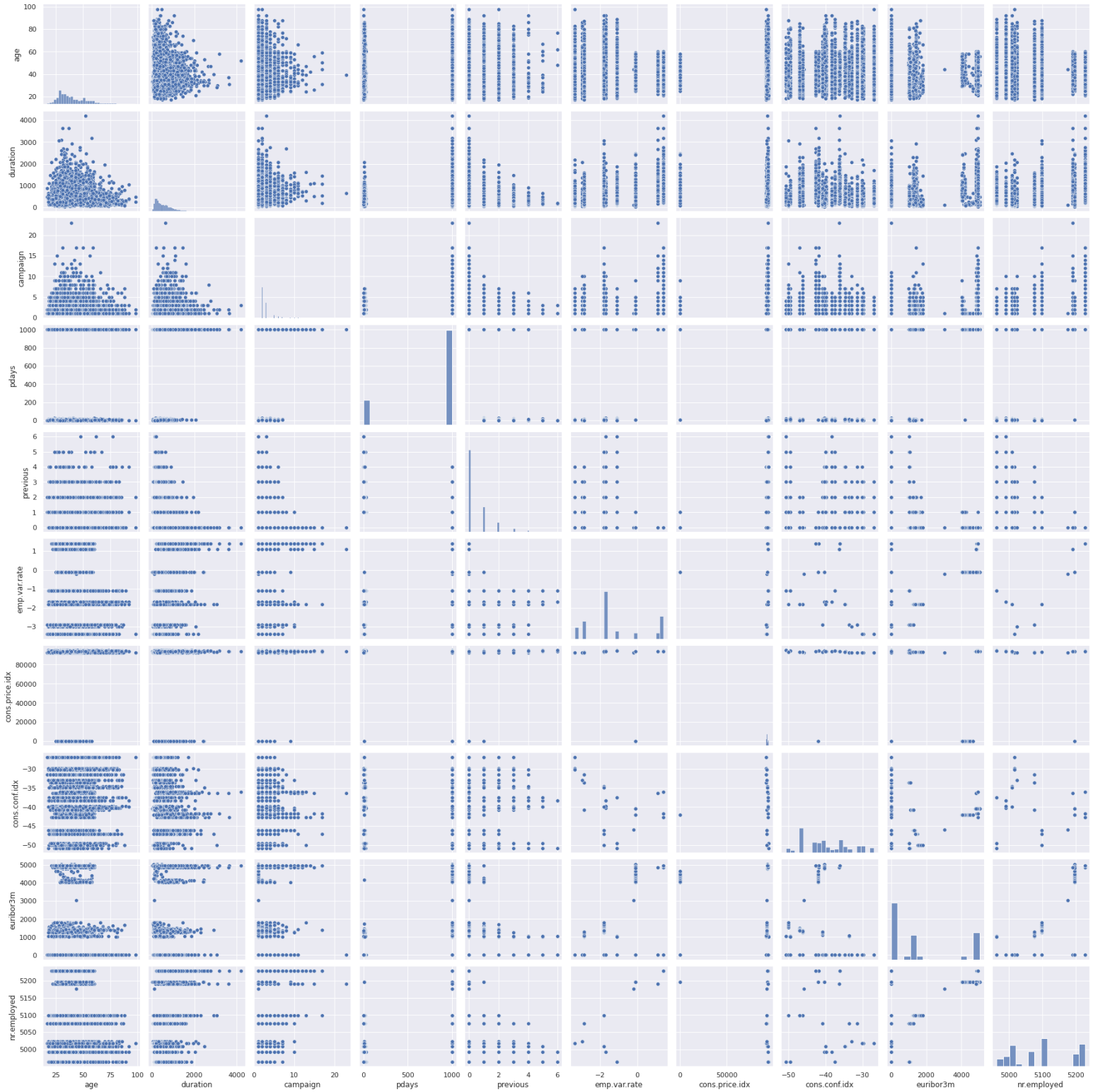
Για περεταίρω κατανόηση της σχέσης μεταξύ όλων των δυνατών συνδυασμών των αριθμητικών μεταβλητών, αλλά και της καλύτερης δυνατής διαχωριστικής τους ικανότητας, παρατίθενται τα Pairplots ολόκληρου του δείγματος, αλλά και της κάθε κλάσης ξεχωριστά.

- **Pairplot ολόκληρου του δείγματος**



- **Pairplot της κλάσης "Yes"**

- **Pairplot της κλάσης "No"**

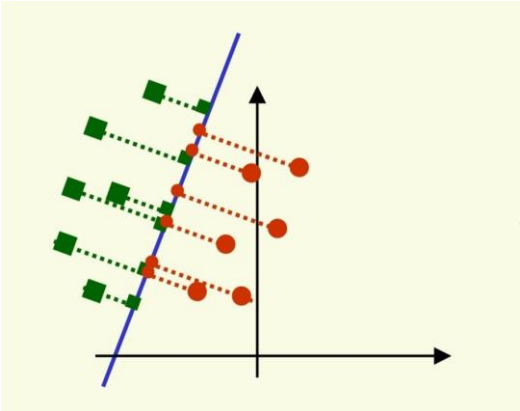


Κεφάλαιο 3 : LDA (Linear Discriminant Analysis)

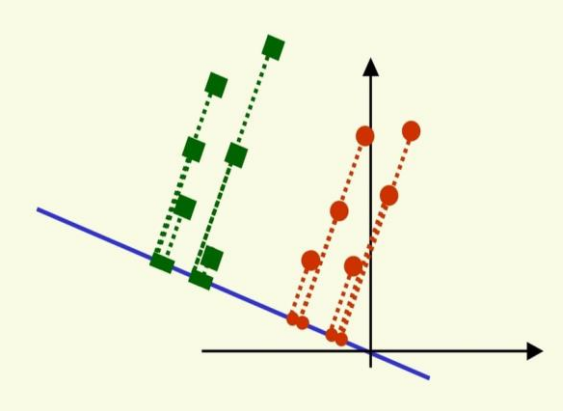
3.1 Εισαγωγή στην LDA & Τεχνικές Μείωσης Διαστάσεων

Η μέθοδος Linear Discriminant Analysis (LDA) είναι μία τεχνική μείωσης διαστάσεων που χρησιμοποιείται ευρέως σε εφαρμογές ταξινόμησης και μηχανικής μάθησης. Συγκεκριμένα, μας παρέχει τη δυνατότητα να μειώσουμε το πλήθος των μεταβλητών σε ένα dataset, διατηρώντας όσο το δυνατόν περισσότερες χρήσιμες πληροφορίες. Φυσικά αυτό όμως δεν είναι απόλυτο, καθώς αυτή η μείωση των διαστάσεων μπορεί να μας οδηγήσει σε μία γενικότερη απώλεια πληροφορίας και συνεπώς όχι τον βέλτιστο διαχωρισμό.

Σκοπός της είναι η μεγιστοποίηση της διαφοράς των μέσων κάθε κλάσης, αλλά και η ελαχιστοποίηση του αθροίσματος των διασπορών γύρω από τις μέσες τιμές της κάθε κλάσης. Έτσι, βρίσκουμε την κατάλληλη προβολή των δεδομένων ώστε να επιτευχθεί ο μέγιστος διαχωρισμός των κλάσεων. Παρακάτω, βλέπουμε ένα παράδειγμα προβολής που διαχωρίζει καλά τα δεδομένα και ένα κακού διαχωρισμού.



Κακό data projection, κακός διαχωρισμός



Καλό data projection, καλός διαχωρισμός

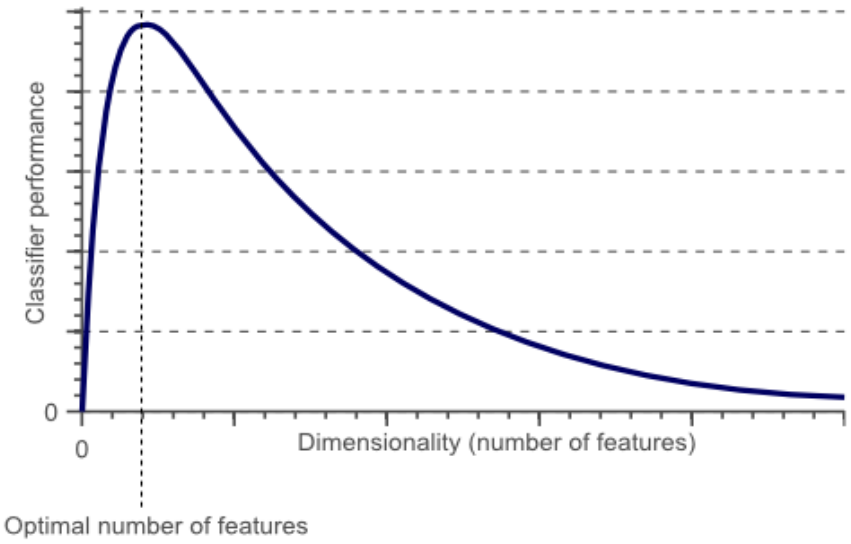
(**Πηγή**: "*https://www.csd.uwo.ca/~oveksler/Courses/CS434a_541a/Lecture8.pdf"*)

Γιατί όμως επιλέγουμε μία τεχνική μείωσης διαστάσεων; Είναι προφανές πως το Machine Learning υπερέχει σε σχέση με τον άνθρωπο στην ανάλυση δεδομένων με πολλές διαστάσεις. Παρόλ'αυτά, όσες περισσότερες διαστάσεις προστίθενται, τόσο αυξάνεται η υπολογιστική ισχύ που χρειάζεται για να αναλυθούν τα δεδομένα και επιπλέον αυξάνεται ο όγκος των δεδομένων και άρα απαιτείται περισσότερη μνήμη. Συνεπώς, είναι πιο εύκολο να διαχειριστούμε ένα πρόβλημα λιγότερων διαστάσεων,

αφού όσο αυξάνεται το πλήθος των μεταβλητών, αυξάνεται και η πολυπλοκότητα του προβλήματος. Το φαινόμενο αυτό, ονομάζεται "The curse of dimensionality" και γίνεται ακόμα πιο κατανοητό μέσω του παρακάτω διαγράμματος.

- **<u>Φαινόμενο Hughes</u>**

Το φαινόμενο Hughes αποτυπώνει ότι καθώς αυξάνεται ο αριθμός των μεταβλητών, η απόδοση του ταξινομητή αρχικά αυξάνεται, μέχρι έναν *βέλτιστο αριθμό χαρακτηριστικών*. Η προσθήκη περισσότερων μεταβλητών ίδιου μεγέθους με το σύνολο εκπαίδευσης, θα κάνει το accuracy του ταξινομητή να φθίνει.

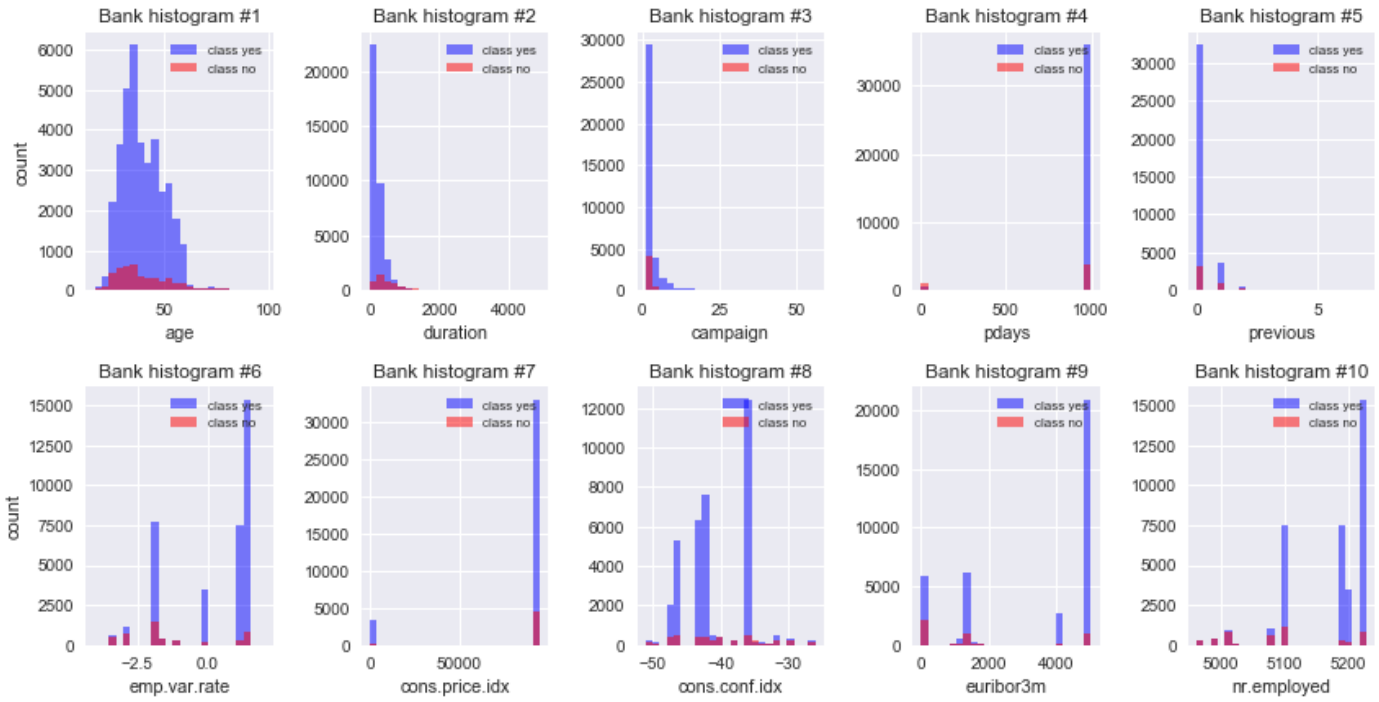


(**Πηγή**: "*https://laptrinhx.com/curse-of-dimensionality-1910417005/"*)

- **Προϋποθέσεις για υλοποίηση της LDA στα δεδομένα**

1. Κάθε μεταβλητή του συνόλου δεδομένων ακολουθεί γκαουσιανή κατανομή.
2. Οι μεταβλητές έχουν ίδια διακύμανση.
3. Δεν πρέπει να υπάρχει πολυσυγγραμικότητα (*multicollinearity*), δηλαδή οι ανεξάρτητες μεταβλητές να είναι ασυσχέτιστες.

Αρχικά, σχετικά με την 1η προϋπόθεση παρατίθεται για κάθε μία από τις δέκα αριθμητικές μεταβλητές ιστόγραμμα, που μας δείχνει πως κατανέμονται σε σχέση με τις δύο κλάσεις. Προφανώς, δεν ακολουθούν κανονική κατανομή.



Επίσης, η 2η προϋπόθεση δεν ικανοποιείται, καθώς οι μεταβλητές δεν έχουν σε καμία περίπτωση κοινές διασπορές.

Σχετικά με την 3η προϋπόθεση, από τους πίνακες συσχέτισης φαίνεται ότι ενώ λίγα ζεύγη μεταβλητών, όπως euribor3m & empvarrate (τρίμηνος δείκτης euribor και employment variation rate) έχουν συσχέτιση της τάξης του 80%, αλλά οι περισσότερες είναι ασυσχέτιστες.
Ωστόσο, παρά τις προϋποθέσεις, θα εφαρμοστεί η μέθοδος και θα αξιολογηθεί.

3.2 Θεωρία Fisher's LDA

Κύρια Ιδέα: Εύρεση της κατάλληλης προβολής των δεδομένων σε μία γραμμή που επιτυγχάνει καλό διαχωρισμό των κλάσεων

Έστω ότι έχουμε *2 κλάσεις* και δείγματα d-διαστάσεων (δηλ. d μεταβλητές) x_1, ….., x_n (n: πλήθος rows) όπου:

- n_1 : δείγματα που ανήκουν στην 1η κλαση
- n_2 : δείγματα που ανήκουν στην 2η κλαση

Έστω επίσης πως γραμμή διαχωρισμού, δίνεται από το μοναδιαίο διάνυσμα w. Η προβολή του δείγματος x_i πάνω στην γραμμή διαχωρισμού w δίνεται άρα από το γινόμενο $w^t \cdot x_i$.

Πως μπορούμε να μετρήσουμε τον διαχωρισμό των κλάσεων; Σύμφωνα με τη συνθήκη Fisher, η μεγιστοποίηση της διαφοράς $|\widetilde{\mu_1} - \widetilde{\mu_2}|$ αποτελεί καλό μέτρο του τελευταίου, όπου $\widetilde{\mu_1}, \widetilde{\mu_2}$ οι μέσοι των προβολών του δείγματος για κάθε κλάση.

Ισχύει:

$$\widetilde{\mu_1} = \frac{1}{n_1} \cdot \sum_{x_i \epsilon C_1}^{n_1} w^t \cdot x_i = w^t \cdot \left(\frac{1}{n_1} \cdot \sum_{x_i \epsilon C_1}^{n_1} x_i \right) = w^t \cdot \mu_1$$

Και όμοια,

$\widetilde{\mu_2} = w^t \cdot \mu_2$, όπου μ_1, μ_2 οι μέσοι των κλάσεων 1 & 2.

Ωστόσο, η διαφορά των μέσων δεν λαμβάνει υπόψη την διακύμανση των κλάσεων. Για τον λόγο αυτό, υπολογίζεται το *scatter* για κάθε κλάση. Έστω ο μετασχηματισμός $y_i = w^t \cdot x_i$, όπου y_i τα μετασχηματισμένα δεδομένα.

Ισχύει :

$$\tilde{s}_1^2 = \sum_{y_i \epsilon \, Class\ 1} (y_i - \tilde{\mu}_1)^2,$$

όπου $\tilde{s}_1$ scatter των μετασχηματισμένων δεδομένων της κλάσης 1.

Και

$$\tilde{s}_2^2 = \sum_{y_i \epsilon \, Class\ 2} (y_i - \tilde{\mu}_2)^2,$$

όπου $\tilde{s}_2$ scatter των μετασχηματισμένων δεδομένων της κλάσης 2.

Παρακάτω φαίνεται η *συνθήκη Fisher*, δηλ. το μέτρο διαχωρισμού των κλάσεων και σκοπός μας είναι η μεγιστοποίηση του. Ουσιαστικά, πρέπει οι μέσοι των projected data να είναι όσο πιο «απομακρυσμένοι», και τα scatters κάθε κλάσης όσο πιο μικρά, δηλαδή τα δεδομένα όσο πιο «μαζεμένα» γύρω από τα μ_1, μ_2.

$$J(w) = \frac{|\widetilde{\mu_1} - \widetilde{\mu_2}|^2}{\tilde{s}_1^2 - \tilde{s}_2^2}$$

Στη συνέχεια ορίζουμε τους εξής πίνακες:

- **Scatter Matrix S**

Ο S μετράει το scatter των αρχικών δεδομένων, δηλαδή πριν την προβολή τους)

$$S_1 = \sum_{x_i \epsilon\, Class\ 1} (x_i - \mu_1)(x_i - \mu_1)^t$$

&

$$S_2 = \sum_{x_i \epsilon\, Class\ 2} (x_i - \mu_2)(x_i - \mu_2)^t$$

- **Διασπορά εντός κλάσεων S_W** (*within class scatter matrix*)

$$\boxed{S_W = S_1 + S_2}$$

Όπως προαναφέρθηκε,

$$\tilde{s}_1^2 = \sum_{y_i \epsilon\, Class\ 1} (y_i - \tilde{\mu}_1)^2$$

Και επίσης :

$$y_i = w^t \cdot x_i\ , \quad \widetilde{\mu_1} = w^t \cdot \mu_1$$

Οπότε έχουμε:

$\tilde{s}_1^2 =$

$$= \sum_{y_i \epsilon\, Class\ 1} (w^t \cdot x_i - w^t \cdot \mu_1)^2 =$$

$$= \sum_{y_i \epsilon\, Class\ 1} \left(w^t \cdot (x_i - \mu_1)\right)^t \cdot \left(w^t \cdot (x_i - \mu_1)\right) =$$

$$= \sum_{y_i \epsilon\, Class\ 1} ((x_i - \mu_1)^t \cdot w)^t \cdot ((x_i - \mu_1)^t \cdot w) =$$

$$= \sum_{y_i \epsilon\, Class\ 1} w^t (x_i - \mu_1)(x_i - \mu_1)^t \cdot w = w^t S_1 w$$

Όμοια:

$\tilde{s}_2^2 = w^t S_2 w$

Οπότε $\tilde{s}_1^2 + \tilde{s}_2^2 = w^t S_1 w + w^t S_2 w = w^t S_W w$

Τελικά: $\boxed{\tilde{s}_1^2 + \tilde{s}_2^2 = w^t S_W w}$

- **Διασπορά μεταξύ των κλάσεων S_B** (*between class scatter matrix*)

Ο S_B μετράει τον διαχωρισμό μεταξύ των μέσων δύο κλάσεων (προ μετασχηματισμού)

$$(\widetilde{\mu_1} - \widetilde{\mu_2})^2 = (w^t \mu_1 - w^t \mu_2)^2 = w^t (\mu_1 - \mu_2)(\mu_1 - \mu_2)^t = w^t S_B w$$

Τελικά: $\boxed{(\widetilde{\mu_1} - \widetilde{\mu_2})^2 = w^t S_B w}$

Συνεπώς, καταλήγουμε :

$$J(w) = \frac{|\widetilde{\mu_1} - \widetilde{\mu_2}|^2}{\tilde{s}_1^2 - \tilde{s}_2^2} = \frac{w^t S_B w}{w^t S_W w}$$

Σκοπός είναι η μεγιστοποίηση της παραπάνω σχέσης , οπότε υπολογίζουμε:

$$\frac{d}{dw} J(w) = 0 \Rightarrow \quad \frac{\left(\frac{d}{dw} w^t S_B w\right) w^t S_W w - \left(\frac{d}{dw} w^t S_W w\right) w^t S_B w}{(w^t S_W w)^2} = 0 \Rightarrow$$

$$\Rightarrow \quad \frac{(2S_B w)w^t S_W w - (2S_W w)w^t S_B w}{(w^t S_W w)^2} = 0 \quad \Rightarrow$$

$$\Rightarrow \quad w^t S_W w(S_B w) - w^t S_B w(S_W w) = 0 \Rightarrow$$

$$\Rightarrow \quad \frac{w^t S_W w(S_B w)}{w^t S_W w} - \frac{w^t S_B w(S_W w)}{w^t S_W w} = 0 \Rightarrow$$

Όμως $J(w) = \frac{w^t S_B w}{w^t S_W w}$

$$\Rightarrow \quad S_B w - \frac{w^t S_B w(S_W w)}{w^t S_W w} = 0 \quad \Rightarrow$$

$$\Rightarrow \underbrace{S_B w = J S_W w}$$

Εξίσωση Ιδιοτιμών

Αν ο S_w είναι αντιστρέψιμος: $S_W^{-1} S_B w = Jw$

Όμως $\forall x$ το $S_B x$ είναι παράλληλο με το μ_1-μ_2, οπότε μπορεί να προσεγγιστεί το w ως εξής:

$$w = S_W^{-1}(\mu_1 - \mu_2)$$

Τελικά η εξίσωση μεγιστοποίησης: $(S_W^{-1} S_B - JI)w = 0$

Συνεπώς, **η λύση w της μεθόδου Fisher** είναι το ιδιοδιάνυσμα που αντιστοιχεί στη μέγιστη ιδιοτιμή του πίνακα $S_W^{-1} S_B$, και τα δεδομένα μετασχηματίζονται σύμφωνα με τη σχέση :

$$Y = \boldsymbol{w} \cdot X$$

- **Ταξινόμηση σύμφωνα με τον κανόνα του Bayes**

Αφού ολοκληρωθεί ο μετασχηματισμός των δεδομένων, τα δεδομένα ταξινομούνται σύμφωνα με τον κανόνα Bayes. Θεωρώντας πως κάθε κλάση έχει συνάρτηση πιθανοφάνειας κανονικής κατανομής, δηλ.

$$P(X = x \mid C = j) = N(\mu_j, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{\frac{1}{2}}} \cdot \exp(-\frac{1}{2}(x - \mu_j)^T \Sigma^{-1}(x - \mu_j)),$$

όπου j=1,2 κλάσεις, d: πλήθος μεταβλητών

Και σύμφωνα με τον κανόνα Bayes: $P(C = j \mid X = x) = \frac{P(X=x \mid C=j) \cdot P(C=j)}{P(X=x)}$

3.3 Υλοποίηση μεθόδου μέσω Python

Παρακάτω φαίνονται αναλυτικά τα βήματα για το πώς υπολογίστηκαν οι προαναφερθέντες πίνακες, οι ιδιοτιμές & τα ιδιοδιανύσματα.

- Αρχικά, υπολογίστηκαν τα mean vectors για κάθε μία κλάση στη μορφή :

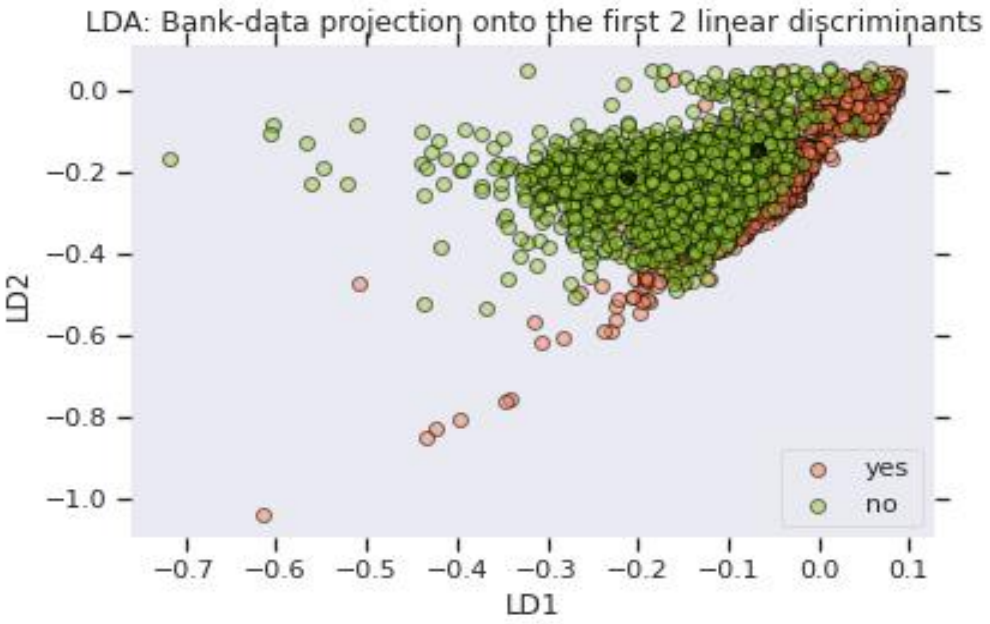
$$\mu_i = \begin{bmatrix} \mu_{w_i(age)} \\ \cdots \\ \mu_{w_i(nr.employed)} \end{bmatrix}, \text{όπου i = 1, 2}$$

- Επιπλέον, υπολογίστηκαν οι 10×10 πίνακες S_w (*διασπορά εντός των κλάσεων*) , S_B (*διασπορά μεταξύ των κλάσεων*) και το γινόμενο $S_W^{-1} \cdot S_B$ ώστε να βρεθούν οι ιδιοτιμές του και τα αντίστοιχα ιδιοδιανύσματα αυτού. Για να πάρουμε τη μέγιστη δυνατή τιμή πληροφορίας παίρνουμε την πρώτη ιδιοτιμή (100% πληροφορία) .

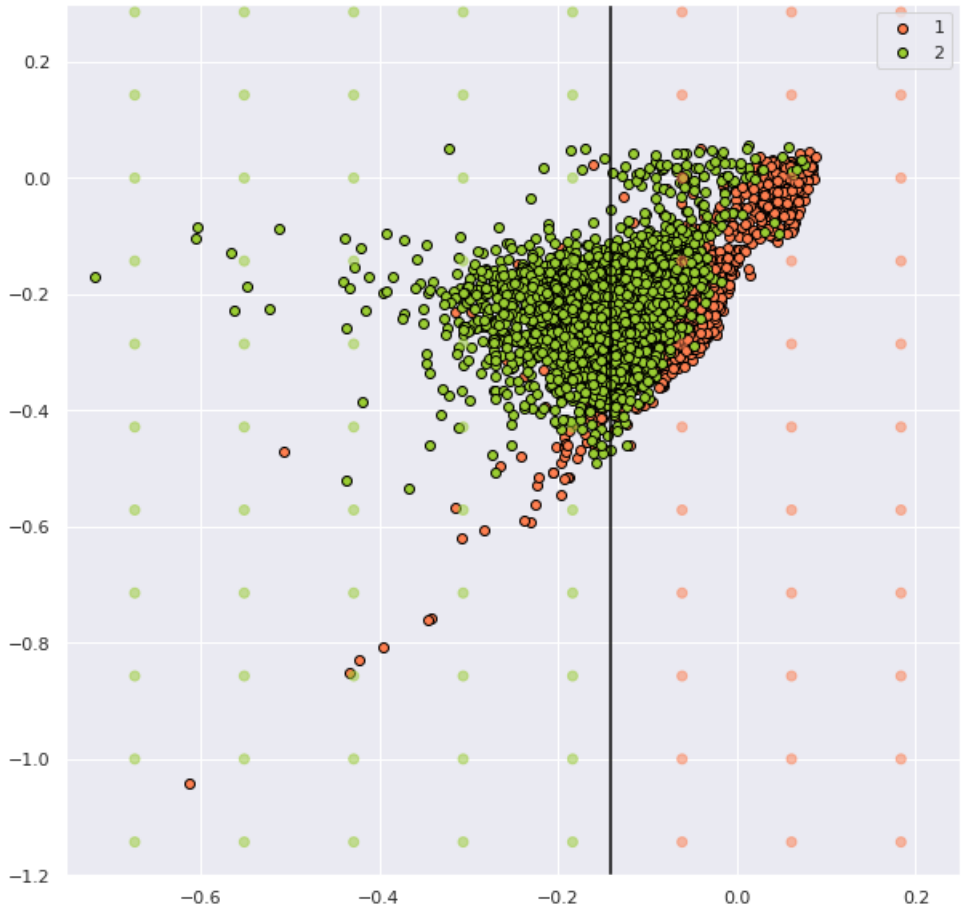
- Ύστερα, βρέθηκε ο πίνακας w, του οποίου οι διαστάσεις είναι 10x2, καθώς κρατήθηκαν και οι τιμές του w που προκύπτουν από τη δεύτερη μεγαλύτερη ιδιοτιμή, για να οπτικοποιηθούν δυσδιάστατα τα δεδομένα και να γίνει πιο εύκολη η κατανόηση των κλάσεων (βλ. παρακάτω σχήμα). Δηλαδή στην 1η στήλη του w, έχουμε το w που αντιστοιχεί στη μέγιστη ιδιοτιμή, ενώ η 2η στήλη τα w που αντιστοιχούν στην 2η ιδιοτιμή. Ο μετασχηματισμός είναι της μορφής :

$$Y = w^t X$$

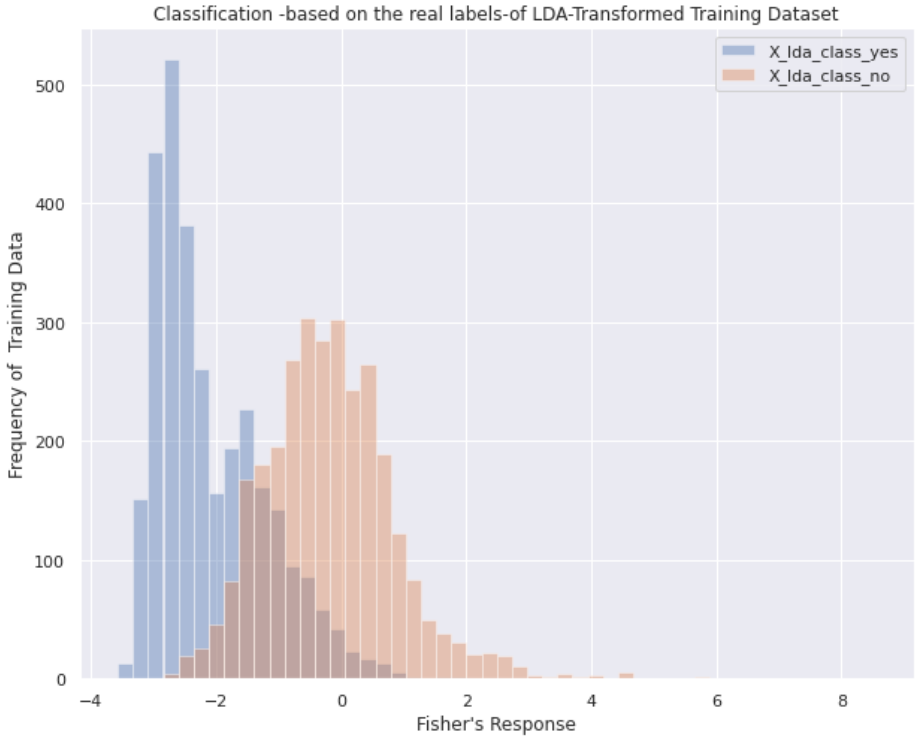
- Στο παρακάτω διάγραμμα βλέπουμε τα **μετασχηματισμένα δυσδιάστατα δεδομένα** και τις **μέσες τιμές κάθε κλάσης**. Ως άξονες έχουμε τις 2 εξής διευθύνσεις:
  - **LD1** : Τη διεύθυνση που προκύπτει από τον πίνακα w της μέγιστης ιδιοτιμής.
  - **LD2** : Τη διεύθυνση που προκύπτει από τον πίνακα w της δεύτερης ιδιοτιμής.

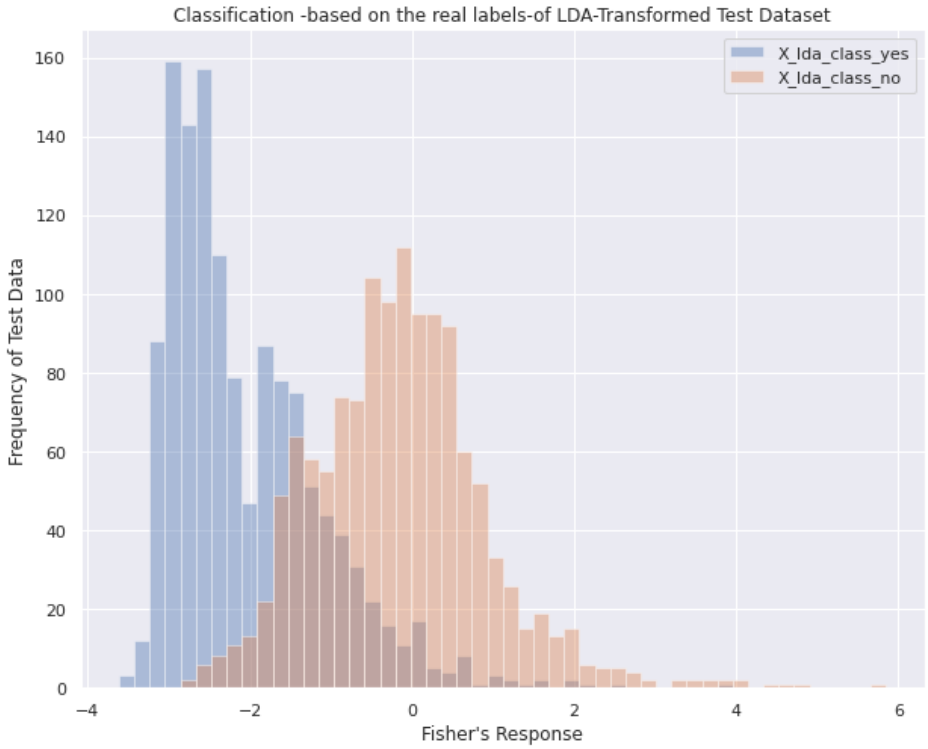


- Ακόμα, παρακάτω φαίνονται τα μετασχηματισμένα **δεδομένα**, **η ευθεία διαχωρισμού** και τα αντίστοιχα **επίπεδα διαχωρισμού**, που είναι ζωγραφισμένα με τα αντίστοιχα χρώματα των δεδομένων κάθε κλάσης.



- Επιπροσθέτως, απεικονίζεται το ιστόγραμμα με τα Frequencies των μετασχηματισμένων data κάθε κλάσης, της LD1 διεύθυνσης, βασισμένα στα πραγματικά labels.





- Επίσης, πέρα από τον αναλυτικό υπολογισμό της μεθόδου, χρησιμοποιήθηκε η βιβλιοθήκη της Python: *sklearn.discriminant_analysis.LinearDiscriminantAnalysis.*

  Τα ορίσματα που της δόθηκαν:

  LinearDiscriminantAnalysis(solver='eigen',n_components=1, tol = 0.0001)

  — *Solver= 'eigen'*, για να εφαρμοστεί η μέθοδος του Fisher
  — *n_components=1*, για μείωση διαστάσεων των δεδομένων σε 1 διάσταση

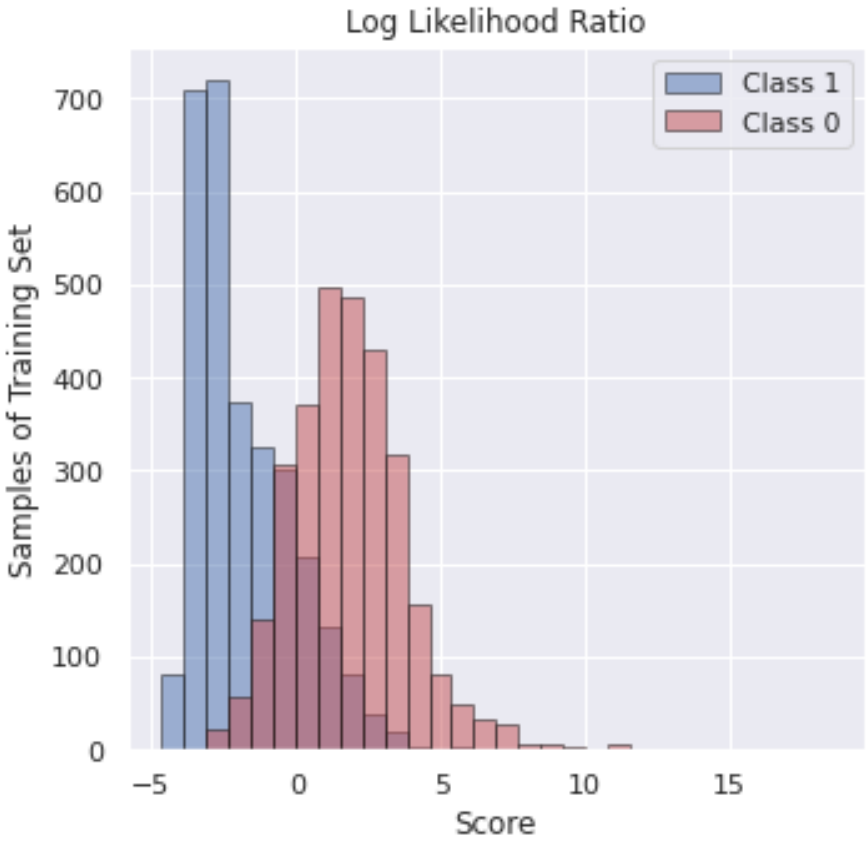
- **Γραφική Αναπαράσταση των Decision Scores για τις δύο κλάσεις**

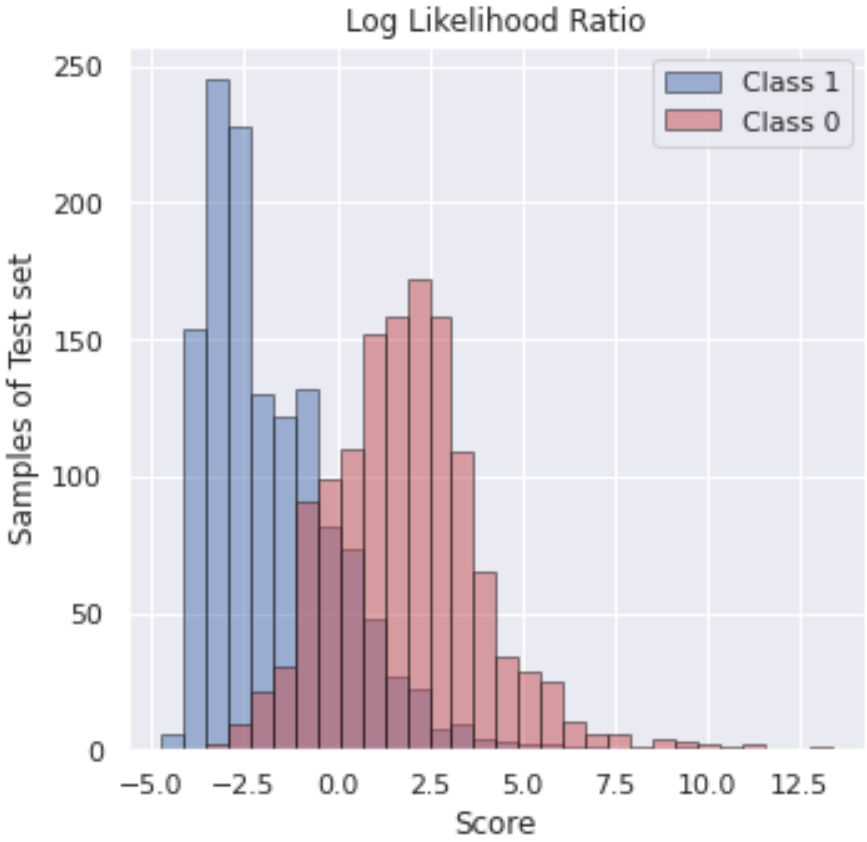
Παρακάτω αναπαριστάται το γράφημα των decision scores αφού εφαρμόστηκε η LDA μέθοδος. Τα scores υπολογίζονται μέσω της *decision_function* της Python. Η συγκεκριμένη συνάρτηση επιστρέφει τη διαφορά $logp(y=1|x) - logp(y=0|x)$, δηλαδή το λογάριθμο του Likelihood ratio: $log\frac{p(y=1|x)}{p(y=0|x)}$.

Στο παρακάτω διάγραμμα φαίνονται οι κατανομές στις δύο κλάσεις ανά πλήθος δειγμάτων σύμφωνα με τα decision scores για το σύνολο εκπαίδευσης αλλά και το σύνολο αξιολόγησης. Χρησιμοποιήθηκαν τα μετασχηματισμένα δεδομένα της Fisher's LDA του sklearn.

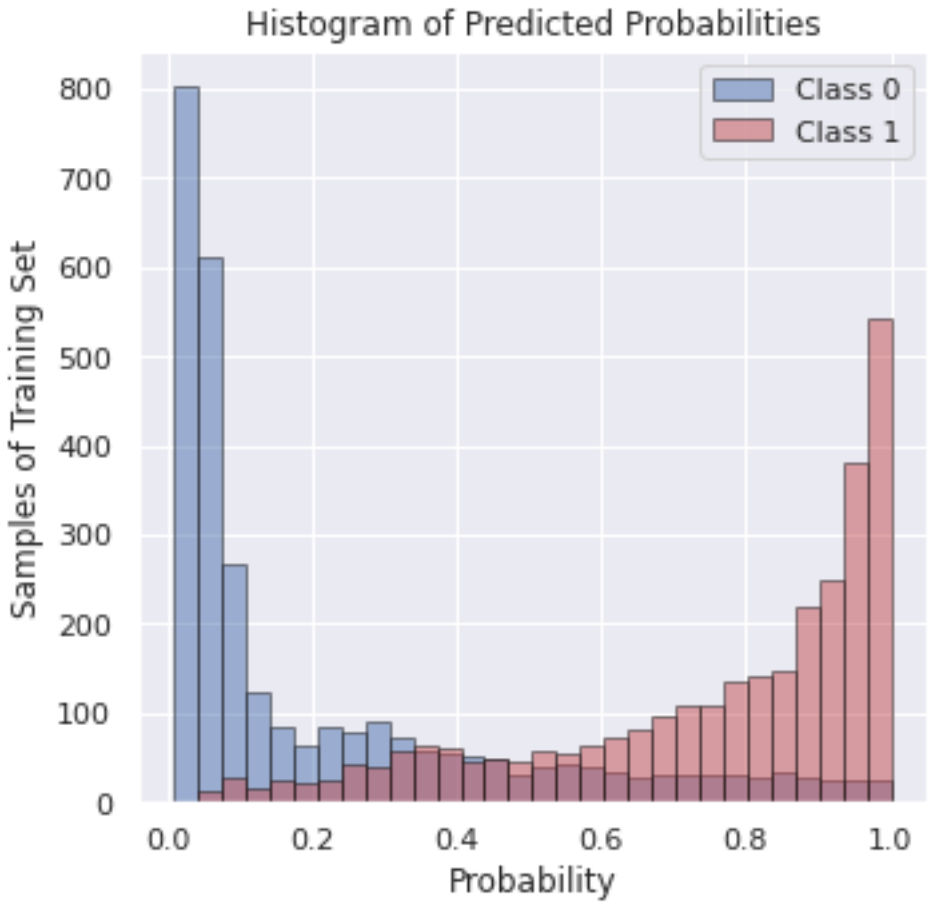
Log Likelihood Ratio:

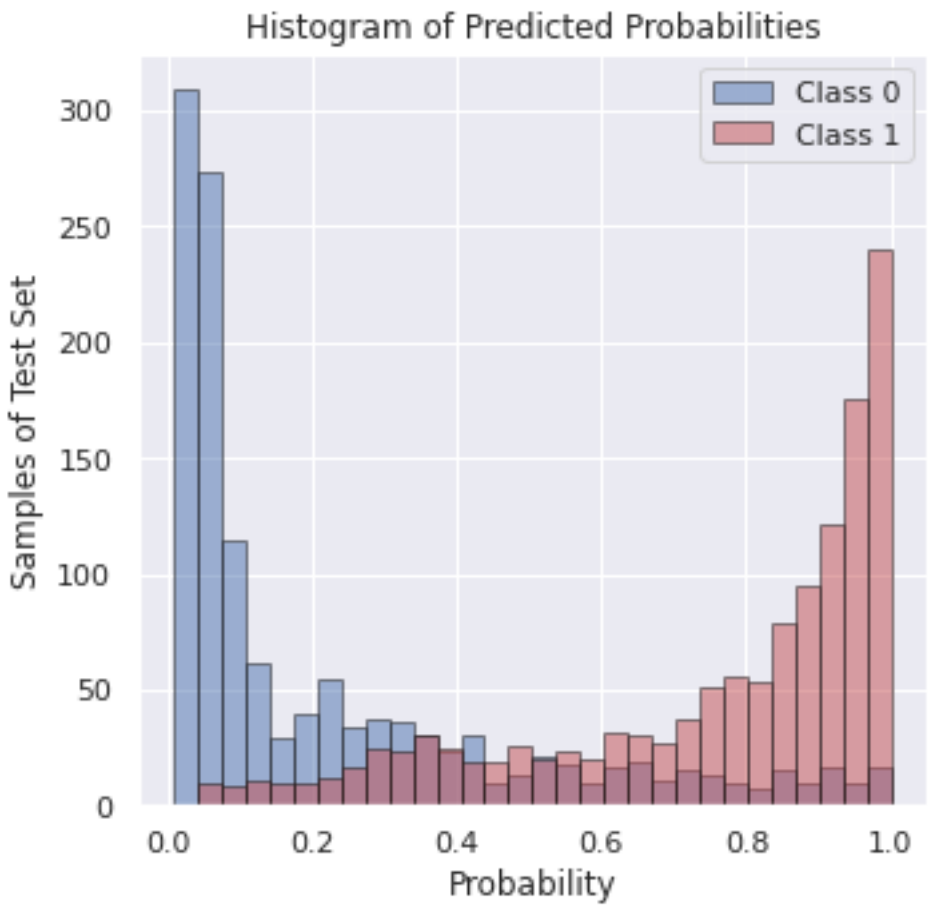
$$\boldsymbol{log\frac{p(y=1|x)}{p(y=0|x)}}$$


Log Likelihood Ratio
Class 1
Class 0
Samples of Training Set
0
100
200
300
400
500
600
700
−5
0
5
10
15
Score


Log Likelihood Ratio
Class 1
Class 0
Samples of Test set
0
50
100
150
200
250
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
Score

- Μέσω της *predict_proba()*, δίνεται η πιθανότητα μια παρατήρησης να ανήκει στην κλάση 0 (κλάση "Yes") και αντίστοιχα στην κλάση 1(απλά αφαιρώντας από το 1 την αντίστοιχη πιθανότητα). Κρατώντας την μία στήλη του output της εντολής, στα παρακάτω ιστογράμματα φαίνονται οι προβλεπόμενες πιθανότητες των δειγμάτων του Training Set & του Test Set να ανήκουν στην κάθε κλάση. Όσο πιο κοντά στο 0, τόσο πιο σίγουρο ότι προβλέπεται η κλάση " Yes", ενώ αντίστοιχα όσο πιο κοντά στο 1, η κλάση "No".

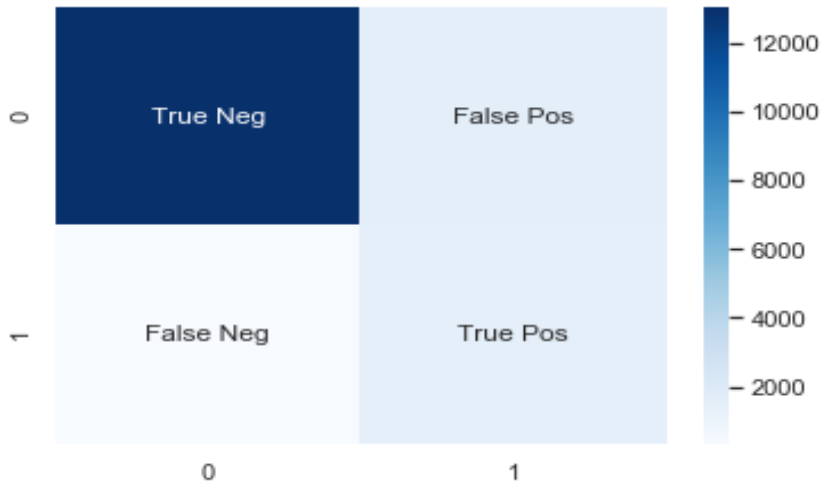




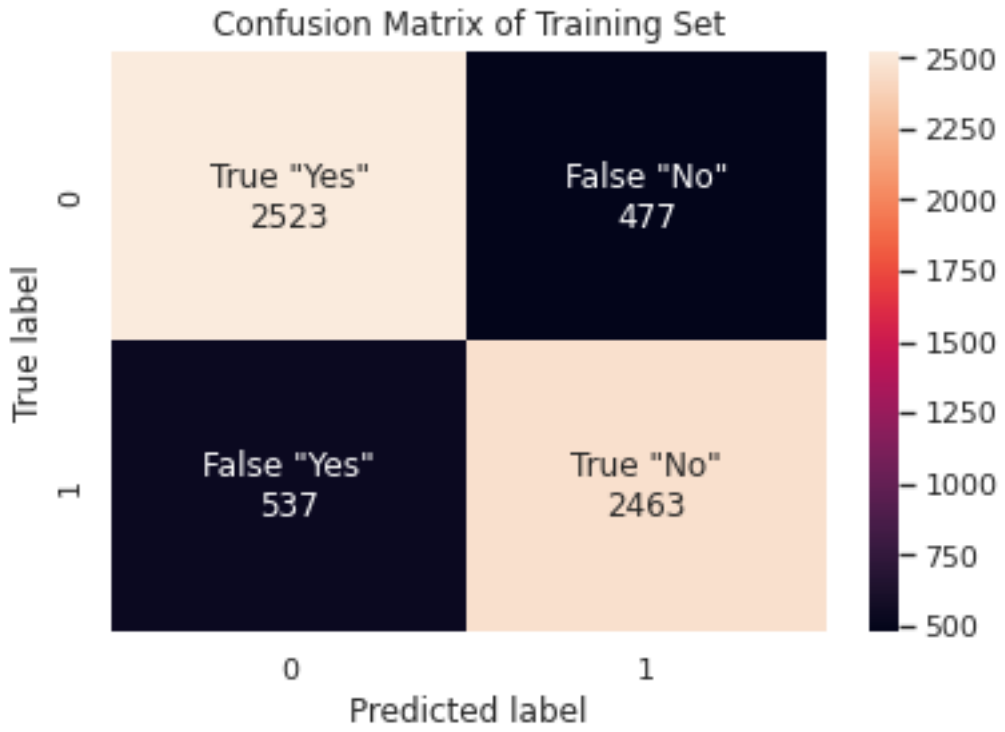
Μετρικές Αξιολόγησης Μεθόδου

1. **Confusion Matrix**

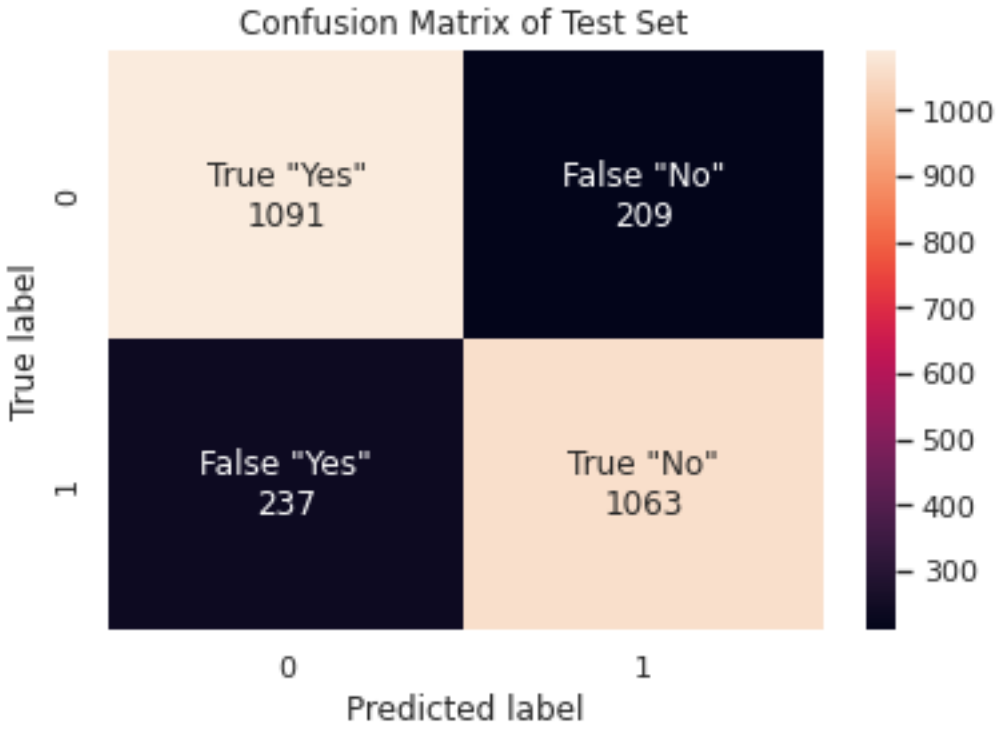
Μέσω του Confusion Matrix, δίνεται η δυνατότητα να πάρουμε μία καλή ένδειξη της αποτελεσματικότητας του μοντέλου μας, καθώς μπορούμε να δούμε το πλήθος των εσφαλμένα & σωστά ταξινομημένων στοιχείων. Κάθε γραμμή του πίνακα αντιπροσωπεύει την πραγματική κλάση, ενώ κάθε στήλη την προβλεπόμενη κλάση. Συγκεκριμένα, ο Confusion Matrix, που παρέχει η Python, είναι της μορφής:



Παρακάτω, παρατίθεται ο Confusion Matrix για το **Training set** .



Στη συνέχεια, παρατίθεται ο Confusion Matrix για το **Test set**.



2. Accuracy Score

Το Accuracy Score είναι από τα πιο γνωστά μέτρα αξιολόγησης ενός machine learning μοντέλου. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Accuracy} = \frac{True\ Positives + True\ Negatives}{Total}$$

$$Acc_{Train} = \frac{2463 + 2523}{6000} = 0{,}83$$

$$Acc_{Test} = \frac{1091 + 1063}{2600} = 0{,}82$$

	Training Set	**Test Set**
Accuracy	0.83	0.82

3. Precision

Το precision ποσοτικοποιεί τον αριθμό των προβλέψεων της κλάσης Positives, που όντως ανήκουν στην κλάση Positives. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Pr_{Train} = \frac{2463}{2463 + 477} = 0{,}83$$

$$Pr_{Test} = \frac{1063}{1063 + 209} = 0{,}83$$

	Training Set	Test Set
Precision	0.83	0.82

4. Recall or Sensitivity

Το recall ποσοτικοποιεί τον αριθμό των προβλέψεων της κλάσης Positives που έγιναν από όλα τα δεδομένα που ανήκουν στην κλάση Positives. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$Re_{Train} = \frac{2463}{2463 + 537} = 0{,}82$$

$$Re_{Test} = \frac{1063}{1063 + 237} = 0{,}82$$

	Training Set	Test Set
Recall	0.82	0.82

5. F1-Score

Το F1-score μετράει την αποδοτικότητα του μοντέλου λαμβάνοντας υπόψη και το recall και το precision. Καθώς τα Precision & Recall από μόνα τους δεν επαρκούν για την αξιολόγηση του μοντέλου, αυτός ο συνδυασμός τους είναι πιο αντικειμενικός. Λαμβάνει τιμές μεταξύ 0 και 1. Όσο μεγαλύτερη η τιμή του, τόσο καλύτερο το μοντέλο.

$$\boldsymbol{F1 - Score} = \frac{2 \cdot Precison \cdot Recall}{Precision + Recall}$$

$$F1_{Train} = \frac{2 \cdot 0{,}82 \cdot 0{,}83}{0{,}82 + 0{,}83} = 0{,}83$$

$$F1_{Test} = \frac{2 \cdot 0{,}82 \cdot 0{,}82}{0{,}82 + 0{,}82} = 0{,}82$$

	Training Set	Test Set
F1- Score	0.83	0.82

<u>Αποτελέσματα για διαφορετικό Test Set</u>

Έγινε μια δοκιμή της μεθόδου χρησιμοποιώντας διαφορετικό σύνολο αξιολόγησης. Συγκεκριμένα, χρησιμοποιήθηκαν οι παρατηρήσεις της υπερέχουσας κλάσης “Yes” που παραλείψαμε λόγω του προβλήματος της ανομοιογένειας. Μέσα από αυτό το σύνολο αξιολόγησης μπορούμε να συμπεράνουμε κατά πόσο όλες οι παρατηρήσεις που ανήκουν στην κλάση “Yes” ταξινομούνται σωστά. Το Accuracy για το νέο σύνολο αξιολόγησης είναι 0,86.

Κεφάλαιο 4 : Ενδυναμωμένα Δένδρα

4.1 Θεωρία Ενδυνάμωσης

Η ενδυνάμωση (boosting) είναι μια αλληλουχία εκπαιδεύσεων ενός βασικού ταξινομητή όπου κάθε φορά τα δεδομένα αποκτούν διαφορετικό βάρος. Το σύνολο εκπαίδευσης επιλέγεται βασιζόμενο στις προηγούμενες επιδόσεις του ταξινομητή. Γενικά, στόχος των αλγορίθμων Boosting είναι να μετατρέψουν "weak learners" σε ισχυρούς.

Ακριβέστερα ένας boosting αλγόριθμος εκπαιδεύει τον «αδύναμο» ταξινομητή, σε κάθε iteration, τοποθετώντας «βάρη» στο training set. Η εκπαίδευση του αρχικού ταξινομητή γίνεται προσδίδοντας ίσα βάρη σε όλα τα δεδομένα του αρχικού συνόλου εκπαίδευσης. Οι υπόλοιποι ταξινομητές εκπαιδεύονται, ανακατανέμοντας τα βάρη στο σύνολο εκπαίδευσης. Πιο συγκεκριμένα, σε κάθε iteration, τα βάρη των λανθασμένα ταξινομημένων παραδειγμάτων αυξάνονται, κάνοντας τον ταξινομητή να επικεντρωθεί κυρίως σε αυτά.

Ο σκοπός του αλγορίθμου είναι να δημιουργήσει έναν τελικό ταξινομητή στη μορφή $H\ (x) : X \rightarrow \{0, 1\}$. Αρχικά, χρησιμοποιεί τα δεδομένα για να παράγει ένα ταξινομητή h^1. Στη συνέχεια, σύμφωνα με τα λάθη του h^1 παράγει τον h^2. Αυτή η λογική συνεχίζεται για τους επόμενους ταξινομητές. Πιο συγκεκριμένα, αν η i-οστή παρατήρηση ταξινομήθηκε λανθασμένα από τον h^t τότε το βάρος της w_i^{t+1} αυξάνεται για τον h^{t+1} ταξινομητή. Συνεπώς, ο επόμενος ταξινομητής επιχειρεί να διορθώσει τα λάθη του προηγούμενου. Η τελική ταξινόμηση γίνεται από τον H(x), ο οποίος αποτελείται από τον γραμμικό συνδυασμό των *h* ταξινομητών.

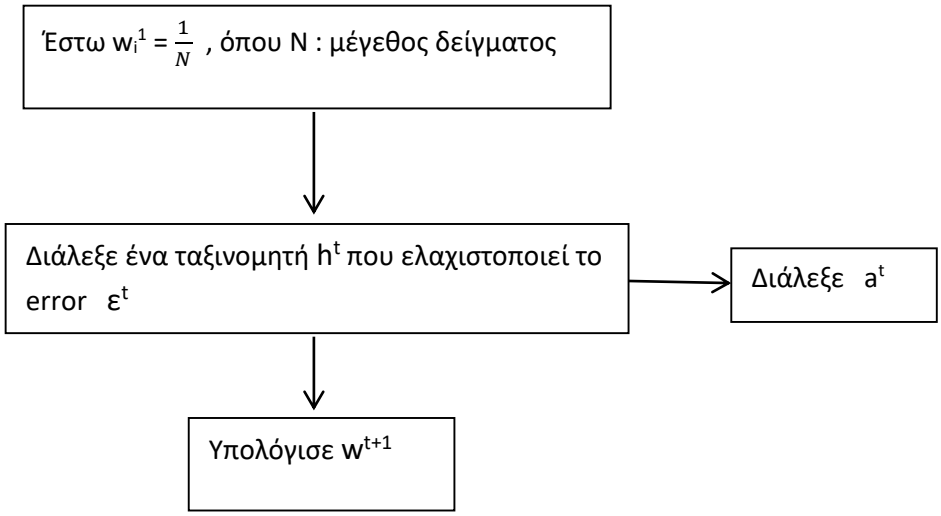
- $H(x) = \text{sign}(a^1h^1(x) + a^2h^2(x) + a^3h^3(x) + \ldots..) = \text{sign} \sum_t (a^t \times h(x)^t)$,

Όπου, $a^t = \frac{1}{2} \times \ln(\frac{1-\varepsilon^t}{\varepsilon^t})$, $\varepsilon = \sum_{Wrong} \frac{1}{N}$

Επιπλέον, σημαντική παρατήρηση είναι πως για τα βάρη w, ισχύει :

- $\sum w_i = 1$
- $w_i^{t+1} = \frac{w_i^t}{z} \times e^{-a^t \times h(x)^t \times y(x)}$, όπου z : Normalizer (λόγω αυτού το $\sum w_i = 1$)

❖ **Μορφή Αλγορίθμου:**



- **Δένδρα Απόφασης με ενδυνάμωση**

Τα Δένδρα Απόφασης (Decision Trees) είναι από τους πιο γνωστούς αλγορίθμους επιβλεπόμενης επαγωγικής μάθησης και έχει εφαρμοστεί με επιτυχία σε πολλούς τομείς όπου απαιτείται ταξινόμηση (π.χ. στην αναγνώριση προσώπων σε εικόνες). Ο αλγόριθμος DT οδηγεί στη δημιουργία μιας δενδροειδούς μορφής που τα φύλλα της αποτελούν κατηγορίες ταξινόμησης (classes). Η δενδροειδής αυτή μορφή μπορεί να αναγνωστεί και ως ένα σύνολο κανόνων που καλούνται κανόνες ταξινόμησης (classification rules). Τα "ενισχυμένα δέντρα" είναι ένας συνδυασμός ενδυνάμωσης και δένδρων αποφάσεων.

4.2 AdaBoost : Θεωρία

Ο AdaBoost αποτελεί έναν από τους πιο γνωστούς boosting αλγόριθμους, ο οπόιος δημιουργήθηκε από τους Schaphire και Freund. Συνοπτικά, η λογική της μεθόδου: "*Reduce general bias using many high-bias models*".

(**Πηγή:** "*https://www.stxnext.com/blog/machine-learning-from-the-woods-exploring-tree-based-ensemble-models-in-python*")

Βασικό εργαλείο της είναι τα decision stumps, δένδρα απόφασης βάθους 1, τα οποία θεωρούνται «αδύναμοι» ταξινομητές. Πιο συγκεκριμένα, ο αλγόριθμος AdaBoost δημιουργεί και καλεί ένα νέο stump. Για κάθε κλήση, μία κατανομή βαρών (που αντιστοιχεί στα δεδομένα) ενημερώνεται έτσι ώστε τα βάρη κάθε στοιχείου εσφαλμένης ταξινόμησης να αυξάνονται, ενώ τα βάρη κάθε σωστής ταξινόμησης να μειώνονται. Έτσι, ο νέος ταξινομητής εστιάζεται σε παραδείγματα που έχουν αποφύγει σωστή ταξινόμηση, δηλαδή δημιουργείται λαμβάνοντας υπόψη τα λάθη του προηγούμενου stump(όπως ακριβώς δηλαδή δουλεύει ένας boosting αλγόριθμος, όπως περιγράφηκε παραπάνω). Συνεπώς, παρόλο που δείχνουν να έχουν μια ουσιαστική τιμή σφάλματος, αφού η απόδοση τους δεν είναι τυχαία, βελτιώνουν το τελικό μοντέλο.

- **<u>Λογική Αλγορίθμου</u>**

Ο αλγόριθμος δέχεται ως input το training set (x1,y1),….(xm,ym), όπου κάθε xi ανήκει σε κάποιο χώρο X και κάθε yi σε κάποιο χώρο Y. Για binary classification problems, ο χώρος Y ορίζεται ως {0, 1}.

Στη συνέχεια, εκπαιδεύεται ο ταξινομητής σε διαδοχικά iterations t = 1,. . . .,T. Ο αλγόριθμος κατανέμει τα βάρη ισόνομα στο σύνολο εκπαίδευσης. Όπως προαναφέρθηκε, τα βάρη των λάθος ταξινομημένων στοιχείων αυξάνονται, έτσι ώστε ο αλγόριθμος να επικεντρώνεται σε αυτά. Σκοπός του αλγορίθμου είναι να ελαχιστοποιήσει το σφάλμα $\varepsilon^t = P_{ri\sim dt}\ [h^t(x_i) \neq y_i\ ]$, όπου D_t η κατανομή των βαρών για το t iteration.

Ο AdaBoost αλγόριθμος επιλέγει μια παράμετρο α^t , η οποία ποσοτικοποιεί την σημαντικότητα του h^t. Για binary classification problems, η παράμετρος αυτή ορίζεται ως: $a^t = \frac{1}{2} \times \ln(\frac{1-\varepsilon^t}{\varepsilon^t})$. Στη συνέχεια η κατανομή Dt ενημερώνεται.

Ο τελικός ταξινομητής ορίζεται ως η πλειοψηφική απόφαση των βασικών ταξινομητών.

$$H(x) = \text{sign} \sum_t (a^t \times h(x)^t),$$

4.3 AdaBoost : Υλοποίηση μέσω Python

Στο συγκεκριμένο πρόβλημα χρησιμοποιήθηκε η κλάση *AdaBoostClassifier()* της βιβλιοθήκης *sklearn* της Python. Για την αποτελεσματικότερη ταξινόμηση, είναι σημαντικό να δοθούν οι κατάλληλες τιμές στις παραμέτρους της AdaBoost, καθώς επιλέγονται εξαρχής. Διαφέρουν μεταξύ των μοντέλων και πρέπει να χρησιμοποιούνται προσεκτικά. Οι προεπιλογές του sklearn της Python είναι λογικές,

αλλά συχνά υπάρχει περιθώριο βελτίωσης. Συνεπώς, για τη μέγιστη αξιοποίηση του μοντέλου, μπορούν να γίνουν δοκιμές και τροποποιήσεις στις παραμέτρους του. Οι πιο σημαντικές είναι :

- **Πλήθος δενδρών**

  Το πλήθος δένδρων αποτελεί αν όχι τη σημαντικότερη, μία από τις πιο βασικές παραμέτρους. Γενικά όσα περισσότερα δένδρα χρησιμοποιούνται, τόσο το καλύτερο. Χωρίς αυτό να αποτελεί γενικό κανόνα, καθώς μπορεί να προκύψει πρόβλημα υπερμοντελοποίησης (overfitting), όταν χρησιμοποιούνται πάρα πολλά δένδρα. Επιπλέον, όσο μεγαλύτερο είναι το πλήθος δένδρων, τόσο αυξάνεται ο χρόνος εκπαίδευσης και πρόβλεψης. (Python: **n_estimators**)

  Παρακάτω βλέπουμε, τα Accuracies Scores των Train & Test Sets που έχουν υπολογιστεί από τον Confusion Matrix για διάφορα πλήθη δέντρων.

Πλήθος Δένδρων	Accuracy of Training Set	Accuracy of Test Set
10	0.87	0.86
50	0.87	0.87
100	0.88	0.87
500	0.89	0.88
700	0.89	0.88
1000	0.89	0.88

- **Ρυθμός Εκμάθησης**

  Ο ρυθμός εκμάθησης ορίζει το μέγεθος των διορθωτικών βημάτων που λαμβάνει το μοντέλο για να προσαρμοστεί για σφάλματα σε κάθε πρόβλεψη. Ένας υψηλός ρυθμός εκμάθησης συντομεύει τον χρόνο εκπαίδευσης, αλλά με χαμηλότερη τελική ακρίβεια, ενώ ένα χαμηλότερο ποσοστό μάθησης διαρκεί περισσότερο, αλλά με τη δυνατότητα μεγαλύτερης ακρίβειας. Συνεπώς, έχει βαθιά συσχέτιση με τα n_estimators, καθώς όσο πιο χαμηλό το learining rate, τόσο μεγαλύτερο πλήθος δένδρων. (Python: **learning_rate**)

  Παρακάτω βλέπουμε, τα Accuracies Scores των Train & Test Sets που έχουν υπολογιστεί από τον Confusion Matrix για διάφορα learning rates.

Learning Rates	Accuracy of Training Set	Accuracy of Test Set
0,01	0.86	0.86
0,1	0.88	0.88
0,5	0.89	0.88
0,8	0.89	0.88

- **<u>Μέγεθος Δένδρου</u>**

  Το βάθος του δένδρου είναι από προεπιλογή 1, καθώς χρησιμοποιούνται stumps. Παρόλ'αυτά μπορούμε να αυξήσουμε το βάθος των δένδρων μέσω του βασικού εκτιμητή DecisionTreeClassifier της Python και για διάφορες τιμές να αξιολογήσουμε το μοντέλο. (Python: **max_depth**)

  Παρακάτω βλέπουμε, τα Accuracies Scores των Train & Test Sets που έχουν υπολογιστεί από τον Confusion Matrix για διάφορα tree depths.

Max Depth	Accuracy of Training Set	Accuracy of Test Set
1	0.89	0.88
2	0.97	0.87
3	1.00	0.86
4	1.00	0.87

- **<u>Συνδυασμός βέλτιστων παραμέτρων</u>**

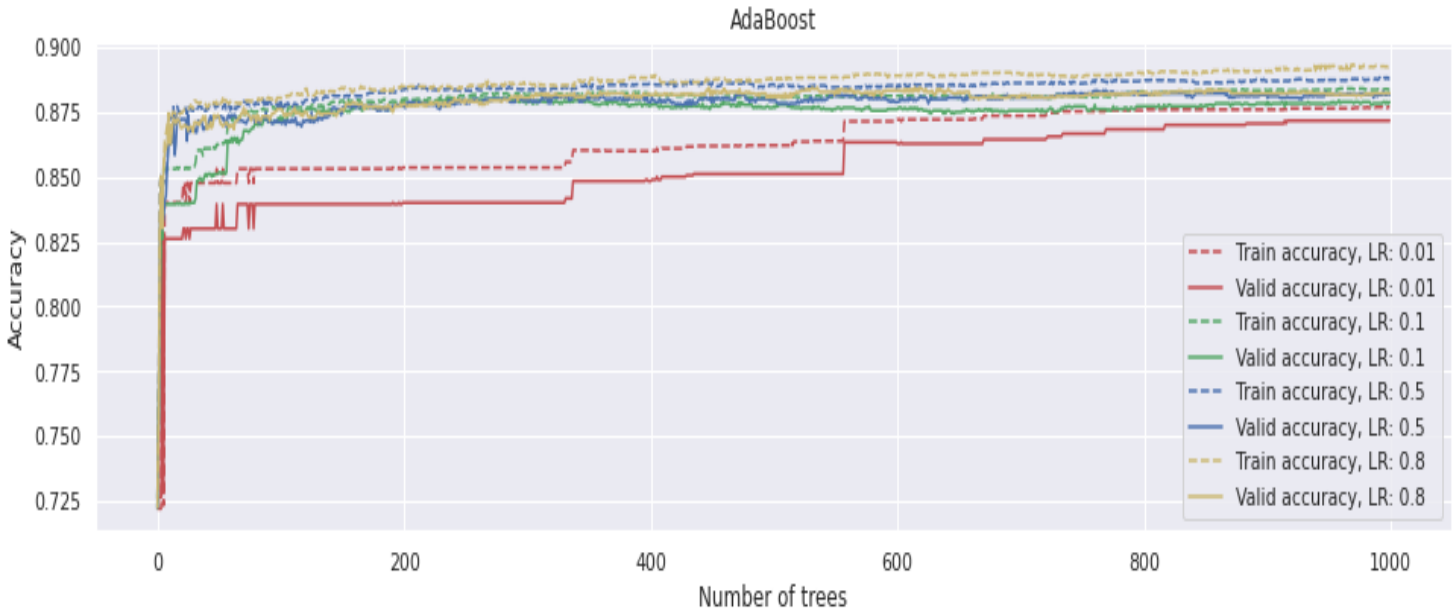
Όπως προαναφέρθηκε, σημαντικό ρόλο στην αποδοτικότητα του μοντέλου παίζει η επιλογή των παραμέτρων του μοντέλου. Καθώς το πλήθος των δένδρων είναι από τους βασικότερους παράγοντες και υπάρχει εξάρτηση από το learning rate, θα βρεθεί η βέλτιστη επιλογή συνδυασμού αυτών των δύο παραμέτρων.

Γι' αυτό το σκοπό παίρνοντας ως παράμετρο όλα τα learning_rates που θα δοκιμαστούν, υπολογίζονται βηματικά τα accuracies του μοντέλου, προσθέτοντας σε κάθε βήμα ένα δένδρο. Για παράδειγμα, στο Ν-οστό βήμα θα υπολογιστεί το Accuracy για Ν-δένδρα και για ένα συγκεκριμένο Learning_rate.

Συνεχίζουν να προστίθενται δένδρα όσο βελτιώνεται και η απόδοση του Test set, όχι μόνο του Training set. Για να αποφευχθεί το πρόβλημα της υπερμοντελοποίησης (overfitting), εξάγεται ένα μέρος των δεδομένων του Training set, το Validation Set,

με σκοπό να παρατηρήσουμε πότε ο αλγόριθμος θα έχει θέμα Overfitting. Γενικά, το πρόβλημα της υπερμοντελοποιήσης ξεκινάει όταν το Accuracy του Training set αυξάνεται, αλλά το Accuracy του Validation set δεν βελτιώνεται.
Επιλέχθηκε εύρος των n_estimators από 0 εώς 1000 δένδρα και Learining rates 0.01, 0.1, 0.5 & 0.8 .

```
AdaBoost: best valid accuracy= 0.88 with 448 trees and learning
rate=0.8
```



- **<u>Τελική Επιλογή Παραμέτρων</u>**

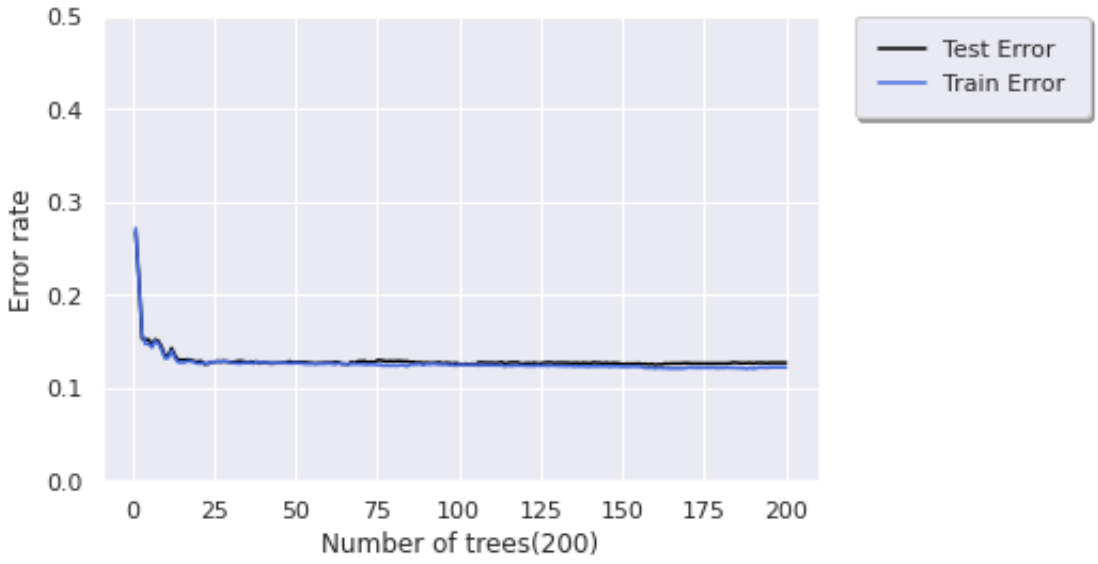
Μετά την αναζήτηση των κατάλληλων παραμέτρων, έγινε η εξής επιλογή :

— n_estimators = 200
— learning_rate = 0.5
— max_depth = 1

- **<u>Διάγραμμα Απώλειας- Σφάλματος</u>**

Παρακάτω παρατίθεται το διάγραμμα απώλειας - σφάλματος συναρτήσει του πλήθους δέντρων του δάσους (δηλαδή αδύναμων ταξινομητών), για πλήθος δέντρων 200. Η μαύρη γραμμή αντιστοιχεί στο σύνολο αξιολόγησης (Test-Set), ενώ η μπλε γραμμή στο σύνολο εκπαίδευσης (Training-Set).

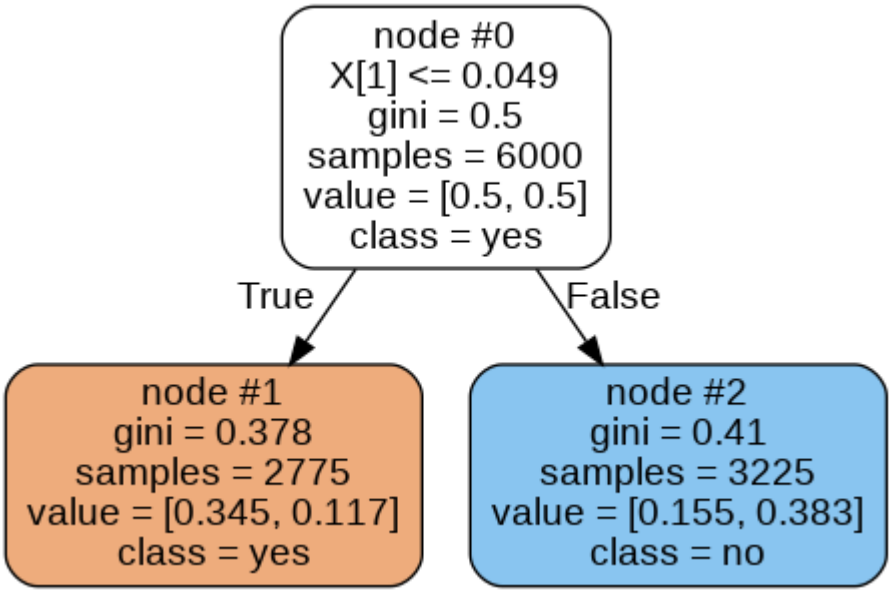
Παρατηρείται ότι τα σφάλματα και του συνόλου εκπαίδευσης αλλά και του συνόλου αξιολόγησης μειώνονται, ενώ μετά από ένα σημείο παραμένουν σχεδόν σταθερά. Όπως είναι λογικό το σφάλμα του συνόλου εκπαίδευσης είναι λίγο μικρότερο από αυτό του συνόλου αξιολόγησης.



Οπτικοποίηση ενός Δένδρου του Δάσους:

Παρακάτω παρατίθεται σε μορφή εικόνας ένα από τα 200 δένδρα της AdaBoost. Όπως έχει προαναφερθεί, είναι βάθους 1. Από τους παρακάτω κόμβους καταλαβαίνουμε τα εξής:

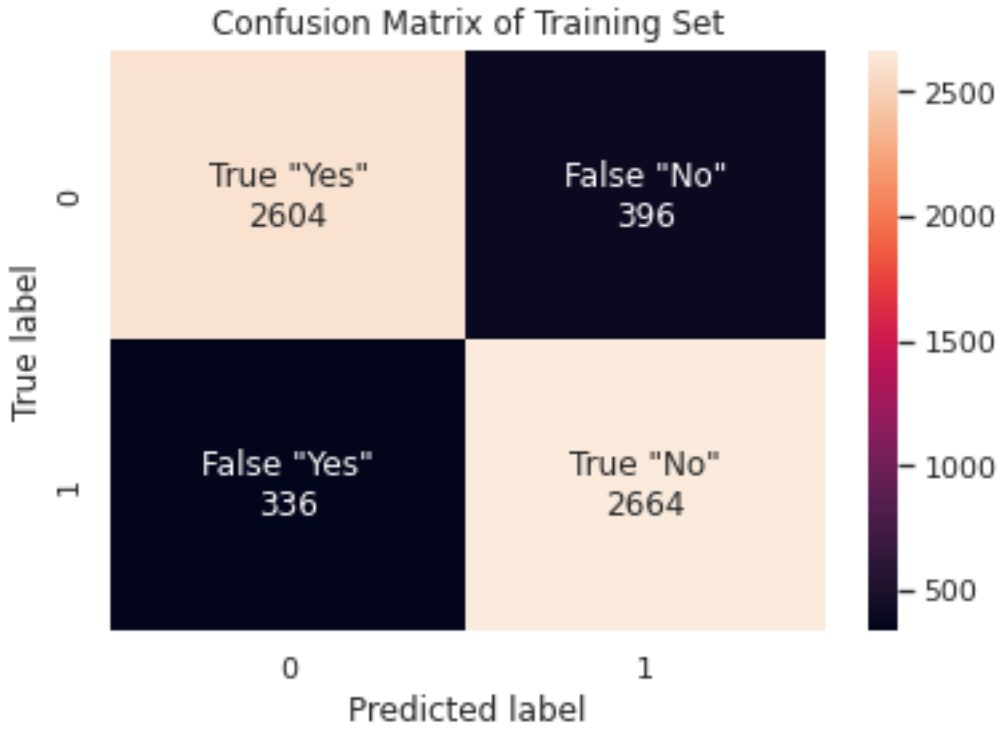
- Το node#0 είναι η ρίζα του δένδρου, η οποία μας δείχνει πως περιέχονται σε αυτή 6000 στοιχεία και υπολογίζονται οι πιθανότητες «value» που μας δείχνει την πιθανότητα που έχει ένα στοιχείο να ταξινομηθεί στην κλάση «Yes» που είναι 0.5 ή στην κλάση «No» που είναι 0.5.
- Στο node#1 που είναι το αριστερό παιδί της ρίζας βλέπουμε πως έχουμε 2775 στοιχεία τα οποία προκύπτουν από την πράξη (0.345 + 0.117) · 6000. Σε αυτό τον κόμβο φαίνεται να υπερισχύει η κλάση «Yes » και παίρνει διαφορετικό χρώμα.
- Όμοια το node#2 που είναι το δεξί παιδί της ρίζας βλέπουμε πως έχουμε 3225 στοιχεία ταξινομημένα στην κλάση «No».



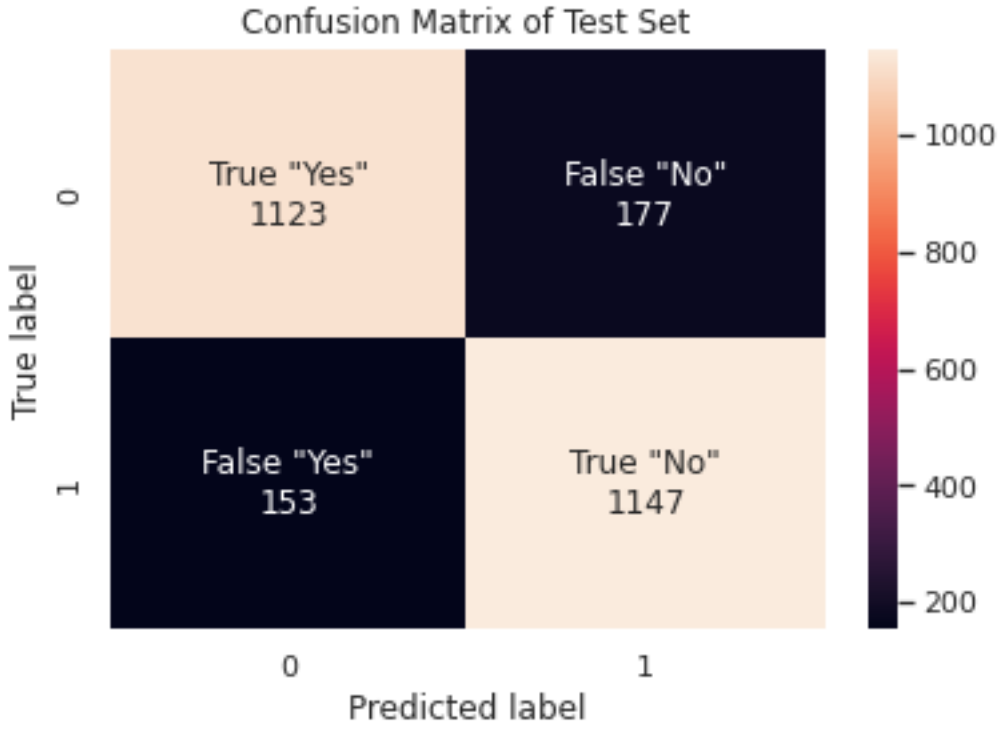
Αξιολόγηση Μεθόδου

1) Confusion Matrix

Παρακάτω, παρατίθεται ο Confusion Matrix για το **Training set**:



Παρακάτω, παρατίθεται ο Confusion Matrix για το **Test set** :



2) Accuracy Score

Το Accuracy Score είναι από τα πιο γνωστά μέτρα αξιολόγησης ενός machine learning μοντέλου. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Accuracy} = \frac{True\ Positives + True\ Negatives}{Total}$$

$$Acc_{Train} = \frac{2604 + 2664}{6000} = 0{,}88$$

$$Acc_{Test} = \frac{1123 + 1147}{2600} = 0{,}87$$

	Training Set	**Test Set**
Accuracy	0.88	0.87

3) Precision

Το precision ποσοτικοποιεί τον αριθμό των προβλέψεων της κλάσης Positives, που όντως ανήκουν στην κλάση Positives. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Pr_{Train} = \frac{2664}{2664 + 396} = 0{,}87$$

$$Pr_{Test} = \frac{1147}{1147 + 177} = 0{,}87$$

	Training Set	Test Set
Precision	0.87	0.87

4) Recall or Sensitivity

Το recall ποσοτικοποιεί τον αριθμό των προβλέψεων της κλάσης Positives που έγιναν από όλα τα δεδομένα που ανήκουν στην κλάση Positives. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$Re_{Train} = \frac{2664}{2664 + 336} = 0{,}89$$

$$Re_{Test} = \frac{1147}{1147 + 153} = 0{,}88$$

	Training Set	Test Set
Recall	0.89	0.88

5) F1-Score

Το F1-score μετράει την αποδοτικότητα του μοντέλου λαμβάνοντας υπόψη και το recall και το precision. Καθώς τα Precision & Recall από μόνα τους δεν επαρκούν για την αξιολόγηση του μοντέλου, αυτός ο συνδυασμός τους είναι πιο αντικειμενικός. Λαμβάνει τιμές μεταξύ 0 και 1. Όσο μεγαλύτερη η τιμή του, τόσο καλύτερο το μοντέλο.

$$\boldsymbol{F1 - Score} = \frac{2 \cdot Precison \cdot Recall}{Precision + Recall}$$

$$F1_{Train} = \frac{2 \cdot 0{,}87 \cdot 0{,}89}{0{,}87 + 0{,}89} = 0{,}88$$

$$F1_{Test} = \frac{2 \cdot 0{,}87 \cdot 0{,}88}{0{,}87 + 0{,}88} = 0{,}87$$

	Training Set	Test Set
F1- Score	0.88	0.87

6) Cross Validation Score

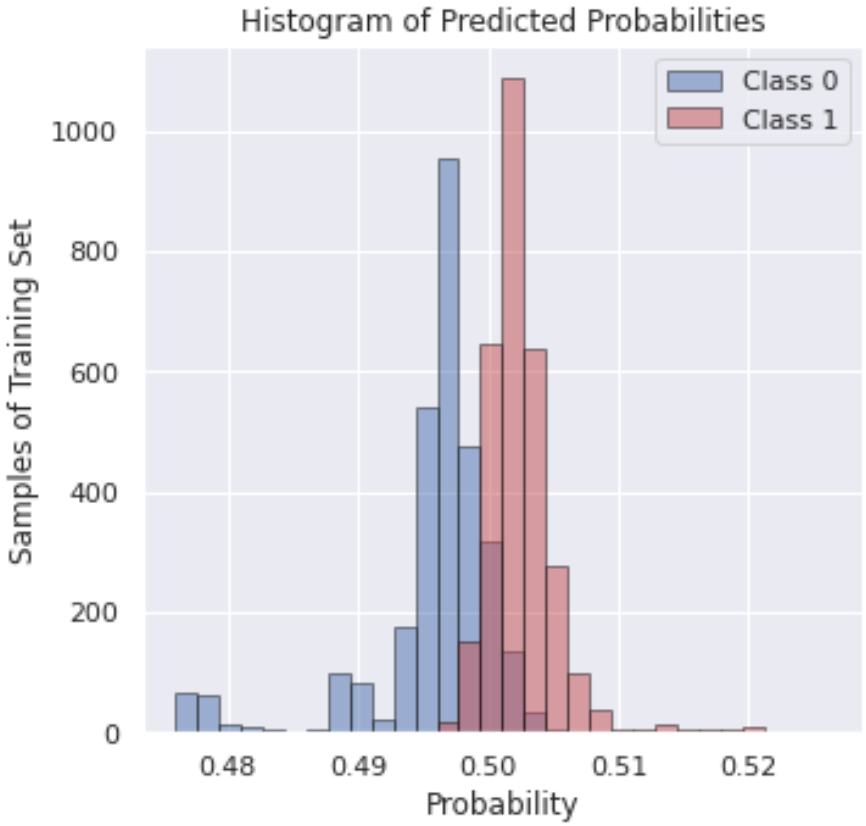
Επιπλέον, μία σημαντική μετρική για να αξιολογήσουμε το μοντέλο μας είναι ο εκτιμητής **Cross Validation**. Πιο συγκεκριμένα, υπολογίστηκε το *Cross_validation_score*. Το σύνολο δεδομένων διαιρείται σε υποσύνολα, κάθε ένα από τα οποία περιέχει διαφορετικές παρατηρήσεις. Η επιλογή των υποσυνόλων είναι τυχαία. Ένα από τα υποσύνολα χρησιμοποιείται ως test set και τα υπόλοιπα συνενώνονται και δημιουργούν το training set. Το μοντέλο εκπαιδεύεται χρησιμοποιώντας το training set και δοκιμάζεται έναντι του test set. Η διαδικασία επαναλαμβάνεται, κάθε φορά χρησιμοποιώντας ένα διαφορετικό σύνολο ως test set και τα υπόλοιπα ως training set. Στο τέλος υπολογίζεται η μέση επίδοση του μοντέλου. Η μέθοδος μπορεί να διαφοροποιηθεί ως προς το πλήθος των τμημάτων. Γενικότερα: Cross Validation k τμημάτων, όπου k συμβολίζει τον αριθμό των δημιουργημένων υποσυνόλων και των επαναλήψεων.

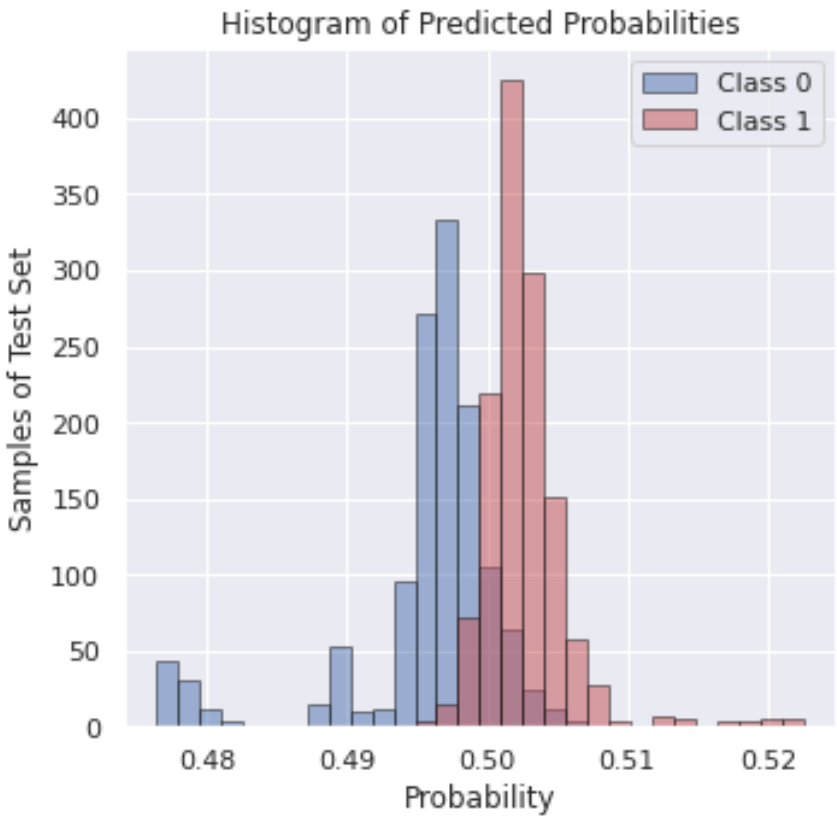
Παρακάτω παρουσιάζεται το Cross Validation Score για το μοντέλο με τα 200 stumps. Στη προαναφερθέν παράμετρο k δόθηκε ο αριθμός 10(είναι και το πιο σύνηθες).

```
0.87 accuracy with a standard deviation of 0.02
```

7)

Μέσω της *predict_proba()*, δίνεται η πιθανότητα μια παρατήρησης να ανήκει στην κλάση 0 (κλάση "Yes") και αντίστοιχα στην κλάση 1(απλά αφαιρώντας από το 1 την αντίστοιχη πιθανότητα). Κρατώντας την μία στήλη του output της εντολής, στα παρακάτω ιστογράμματα φαίνονται οι προβλεπόμενες πιθανότητες των δειγμάτων του Training Set & του Test Set να ανήκουν στην κάθε κλάση. Όσο πιο κοντά στο 0 προβλέπεται η κλάση " Yes", ενώ αντίστοιχα όσο πιο κοντά στο 1, η κλάση "No".





Αποτελέσματα για διαφορετικό Test Set

Έγινε μια δοκιμή της μεθόδου χρησιμοποιώντας διαφορετικό σύνολο αξιολόγησης. Συγκεκριμένα, χρησιμοποιήθηκαν οι παρατηρήσεις της υπερέχουσας κλάσης "Yes" που παραλείψαμε λόγω του προβλήματος της ανομοιογένειας. Μέσα από αυτό το σύνολο αξιολόγησης μπορούμε να συμπεράνουμε κατά πόσο όλες οι παρατηρήσεις που ανήκουν στην κλάση "Yes" ταξινομούνται σωστά. Το Accuracy για το νέο σύνολο αξιολόγησης είναι 0,86.

4.4 GradientBoost: Θεωρία

Η GradientBoost αποτελεί ένα ισχυρό αλγόριθμο μηχανικής μάθησης. Χρησιμοποιείται κυρίως σε μοντελοποιημένα προβλήματα πρόβλεψης, όπως ταξινόμησης και παλινδρόμησης. Η κύρια ιδέα της είναι πως αντί να προσαρμόζεται ένας predictor στα δεδομένα μετά από κάθε επανάληψη, ουσιαστικά προσαρμόζει έναν νέο predictor σύμφωνα με τα residual errors του προηγούμενου predictor.

Μια σύντομη περιγραφή της λογικής του αλγορίθμου :

1. Αρχικά, για να γίνουν οι αρχικές προβλέψεις σύμφωνα με τα δεδομένα, υπολογίζεται το log(odds) του target. Το log(odds) ουσιαστικά αποτελεί τον αριθμό των True Values προς τα False Values. (Σημαντική παρατήρηση : Το log(odds) δεν αποτελεί πιθανότητα, αλλά λόγο πιθανοτήτων).

2. Στη συνέχεια, γίνεται μετατροπή του log(odds) μέσω της *logistic function* σε πιθανότητα, ώστε να γίνουν οι προβλέψεις.

   - *Logistic Function:* $\frac{e^{\log(odds)}}{1+ e^{\log(odds)}}$

3. Συγκρίνοντας την πιθανότητα με το threshold (συνήθως είναι 0.5) η πρώτη –προφανώς λανθασμένη- ταξινόμηση. Μέσω του υπολογισμού των Pseudo-Residuals μπορούμε να δούμε κατά πόσο εσφαλμένη ήταν η αρχική ταξινόμηση.

$$\underline{Residuals = Observed - Predicted}$$

4. Αφού γίνει αυτό, δημιουργείται ένα νέο Decision Tree, που ουσιαστικά προσπαθεί να προβλέψει τα νέα residuals. Σημαντικό να αναφερθεί πως υπάρχει συγκεκριμένος επιτρεπτός αριθμός φύλλων. Αυτό αποτελεί μία παράμετρο που την θέτει ο χρήστης. Οι τιμές της παραμέτρου κυμαίνονται από 8 εώς 32. Συνεπώς, υπάρχουν δύο περιπτώσεις :
   i. Είτε ένα δεδομένο (an instance, a row of the data) «πιάνει» ένα ολόκληρο φύλλο
   ii. Είτε περισσότερα από ένα δεδομένα χρησιμοποιούν κοινό φύλλο.

Για την μετατροπή αυτών, δεν μπορεί να γίνει μία απλή πρόσθεση, αλλά χρησιμοποιείται μία συγκεκριμένη φόρμουλα :

$$Output\ Value\ of\ a\ Leaf = \frac{\sum Residual_i}{\sum[Previous\ Probability_i \times (1 - Previous\ Probability_i)]}$$

5. Για την πρόβλεψη του log(odds), χρησιμοποιείται η παρακάτω φόρμουλα. Ο ρυθμός εκμάθησης επιτρέπει στην αποφυγή του overfitting, αφού ουσιαστικά απαιτεί περισσότερα Decision Trees για να μειώνεται ο ρυθμός που καταλήγουμε στη τελική πρόβλεψη. Μέσω αυτών των μικρών σταδιακών βημάτων, μπορεί να ελέγχεται η διατήρηση μια καλής τιμής bias με συνολικά χαμηλό variance.

Log(odds) Prediction = base log(odds) + (learning_rate × output value of a leaf)

6. Αφού γίνει ο παραπάνω υπολογισμός, μετατρέπεται το Log(odds) σε πιθανότητα χρησιμοποιώντας το προαναφερθέν Logistic Function και ανανεώνονται τα residuals. Η ίδια διαδικασία συνεχίζεται με το επόμενο δένδρο.

7. Τελος, αφού έχουν υπολογιστεί όλα τα output values των δένδρων, το τελικό log(odds) θα είναι της μορφής:

   $Initial\ Prediction + learning\_rate \times predicted\_residual_1 + learning\_rate \times predicted\_residual_2 + learning\_rate \times predicted\_residual_3 + \ldots$

4.5 GradientBoost: Υλοποίηση μέσω Python

Στο πρόβλημα χρησιμοποιήθηκε η κλάση *GradientBoostingClassifier()* της βιβλιοθήκης sklearn της Python. Όπως και στην AdaBoost, για την καλύτερη δυνατή ταξινόμηση, έγινε αναζήτηση των κατάλληλων τιμών των παραμέτρων που μπορούν να δοθούν στο μοντέλο.

Οι πιο σημαντικές είναι :

- **<u>Πλήθος Δένδρων</u>**

Μία σημαντική παράμετρος για τον αλγόριθμο Gradient Boosting είναι το πλήθος των decision trees που χρησιμοποιούνται για το μοντέλο. Ουσιαστικά, οι εκτιμητές ή δένδρα αποφάσης προστίθενται στο μοντέλο διαδοχικά με σκοπό τη διόρθωση και βελτίωση των προβλέψεων των προηγούμενων δένδρων. Ως εκ τούτου, όπως και στην AdaBoost, όσο περισσότερα δένδρα, αναμένονται τόσο καλύτερα αποτελέσματα.

Επιπλέον, για την σωστή επιλογή του αριθμού των δένδρων, πρέπει να λαμβάνεται υπόψη ο ρυθμός εκμάθησης. Για παράδειγμα, όσο περισσότερα δένδρα, τόσο

μικρότερο learning rate απαιτείται και αντίστοιχα λιγότερα δένδρα μπορεί να απαιτούν μεγαλύτερο learning rate.

Όπως και στην AdaBoost, ο αριθμός των δένδρων ορίζεται ως "n_estimators" στην Python και από προεπιλογή είναι 100. Στο συγκεκριμένο πρόβλημα έγινε διερεύνηση στην αποδοτικότητα του αλγορίθμου δίνοντας τιμές στο όρισμα n_estimators μεταξύ 10 και 1.000.

Παρακάτω βλέπουμε, τα Accuracies Scores των Train & Test Sets που έχουν υπολογιστεί από τον Confusion Matrix για διάφορα πλήθη δέντρων.

Πλήθος Δένδρων	**Accuracy of Training Set**	**Accuracy of Test Set**
10	0,87	0,87
50	0,89	0,88
100	0,89	0,88
500	0,93	0,89
700	0,95	0,89
1000	0,96	0,89

- **Subsample**

Το Subsample αφορά τον αριθμό των δειγμάτων που χρησιμοποιεί ένα tree, καθώς δεν είναι ανάγκη να χρησιμοποιείται ολόκληρο το training set. Συνεπώς, για κάθε δένδρο μπορεί να χρησιμοποιηθεί ένα τυχαίο υποσύνολο του συνόλου εκπαίδευσης. Αυτή η τεχνική είναι γνωστή και ως "Stochastic Gradient Boosting" και μειώνει το μεγάλο correlation μεταξύ των δένδρων και εμποδίζει το overfitting. Παρόλο που χρησιμοποιώντας λιγότερα δείγματα μπορεί να αυξηθεί το variance κάθε δένδρου, αυξάνει τη συνολική και τελική αποδοτικότητα του μοντέλου.

Η Python χρησιμοποιεί ως προεπιλογή ολόκληρο το σύνολο αξιολόγησης, οπότε στο όρισμα subsample δίνεται η τιμή 1.0. Παρακάτω βλέπουμε, τα Accuracies Scores των Train & Test Sets που έχουν υπολογιστεί από τον Confusion Matrix για διάφορες τιμές του ορίσματος subsample.

Subsample	**Accuracy of Training Set**	**Accuracy of Test Set**
0,1	0,89	0,88
0,2	0,89	0,88
0,3	0,90	0,88
0,4	0,90	0,88
0,5	0,90	0,88
0,6	0,89	0,89

0,7	0,90	0,88
0,8	0,90	0,88
0,9	0,90	0,88
1,0	0,90	0,88

- **Learning Rate**

Ο ρυθμός εκμάθησης ελέγχει το κατά πόσο θα συνεισφέρει κάθε μοντέλο στην πρόβλεψη. Όπως αναφέρθηκε και παραπάνω, μικρότερα learning rates απαιτούν περισσότερα δένδρα απόφασης και αντίστοιχα μεγαλύτερα learning rates λιγότερα δένδρα.

Το αντίστοιχο όρισμα της Python είναι το learning_rate, και δόθηκαν τιμές από 0.0001 μέχρι και 1.0 για να γίνει η διερεύνηση της καλύτερης αποδοτικότητας.

Learning Rate	Accuracy of Training Set	Accuracy of Test Set
0,0001	0,85	0,85
0,0010	0,85	0,85
0,0100	0,87	0,87
0,1000	0,90	0,88
1,0000	0,95	0,87

- **Βάθος Δένδρου**

Το βάθος του δένδρου αποτελεί άλλη μια σημαντική παράμετρο της Gradient Boost. Ελέγχει πόσο «ειδικεύεται» κάθε δένδρο στο σύνολο εκπαίδευσης, δηλαδή πόσο γενικό ή υπερεκπαιδευμένο μπορεί να είναι. Γενικά προτιμάται να μην είναι πολύ ρηχά, όπως στην AdaBoost, αλλά και ούτε πολύ βαθιά και εξειδικευμένα.

Η Pyhton χρησιμοποιεί το όρισμα max_depth και από προεπιλογή είναι 3. Παρακάτω βλέπουμε, τα Accuracies Scores των Train & Test Sets που έχουν υπολογιστεί από τον Confusion Matrix για διάφορα tree depths.

Max Depth	Accuracy of Training Set	Accuracy of Test Set
1	0.87	0.87
2	0.89	0.88
3	0.90	0.88
4	0.91	0.89

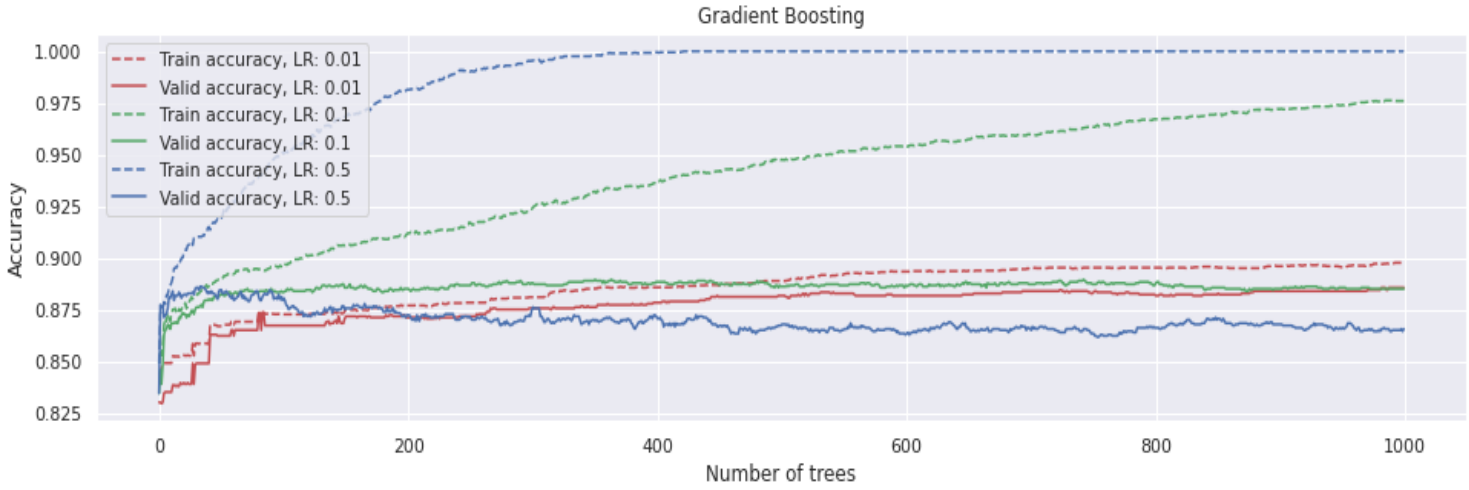
- **<u>Συνδυασμός βέλτιστων παραμέτρων</u>**

Ακριβώς όπως και στην AdaBoost, θα γίνει εξερεύνηση του συνδυασμού δύο σημαντικών παραμέτρων με σκοπό να βρεθεί ο βέλτιστος συνδυασμός τους. Εφαρμόστηκε, λοιπόν, η ίδια συνάρτηση η οποία παίρνοντας ως παράμετρο όλα τα learning_rates που θα δοκιμαστούν και υπολογίζει βηματικά τα accuracies του μοντέλου, προσθέτοντας σε κάθε βήμα ένα δένδρο. Για παράδειγμα, στο N-οστό βήμα θα υπολογιστεί το Accuracy για N-δένδρα και για ένα συγκεκριμένο Learning_rate.

Συνεχίζουν να προστίθενται δένδρα όσο βελτιώνεται και η απόδοση του Test set, όχι μόνο του Training set. Για να αποφευχθεί το πρόβλημα της υπερμοντελοποίησης (overfitting), εξάγεται ένα μέρος των δεδομένων του Training set, το Validation Set, με σκοπό να παρατηρήσουμε πότε ο αλγόριθμος θα έχει θέμα Overfitting. Γενικά, το πρόβλημα της υπερμοντελοποιήσης ξεκινάει όταν το Accuracy του Training set αυξάνεται, αλλά το Accuracy του Validation set δεν βελτιώνεται.

Επιλέχθηκε εύρος των n_estimators από 0 εώς 1000 δένδρα και Learining rates 0.01, 0.1 & 0.5

```
Gradient Boosting: best valid accuracy=0.89 with 351 trees and le
arning rate=0.1
```



- **Τελική Επιλογή Παραμέτρων**

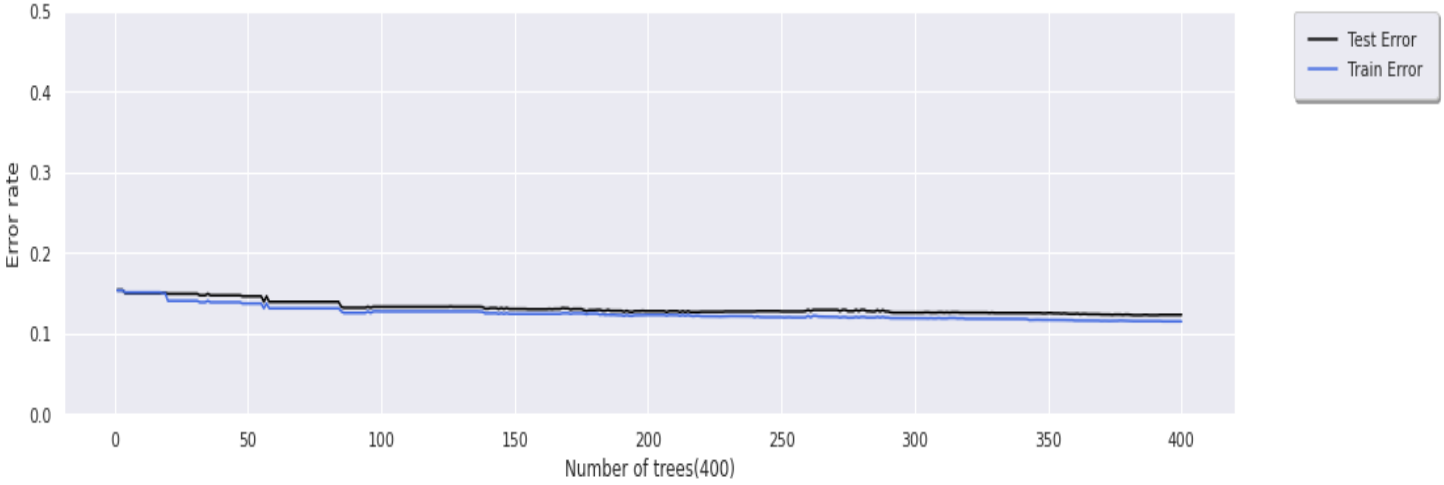
Μετά την αναζήτηση των κατάλληλων παραμέτρων, έγινε η εξής επιλογή :

- n_estimators = 400
- learning_rate = 0.01
- max_depth = 3 (default)
- subsample= 1.0

- **Διάγραμμα Απώλειας- Σφάλματος**

Παρακάτω παρατίθεται το διάγραμμα απώλειας - σφάλματος συναρτήσει του πλήθους δέντρων του δάσους (δηλαδή αδύναμων ταξινομητών), για πλήθος δέντρων 400. Η μαύρη γραμμή αντιστοιχεί στο σύνολο αξιολόγησης (Test-Set), ενώ η μπλε γραμμή στο σύνολο εκπαίδευσης (Training-Set).

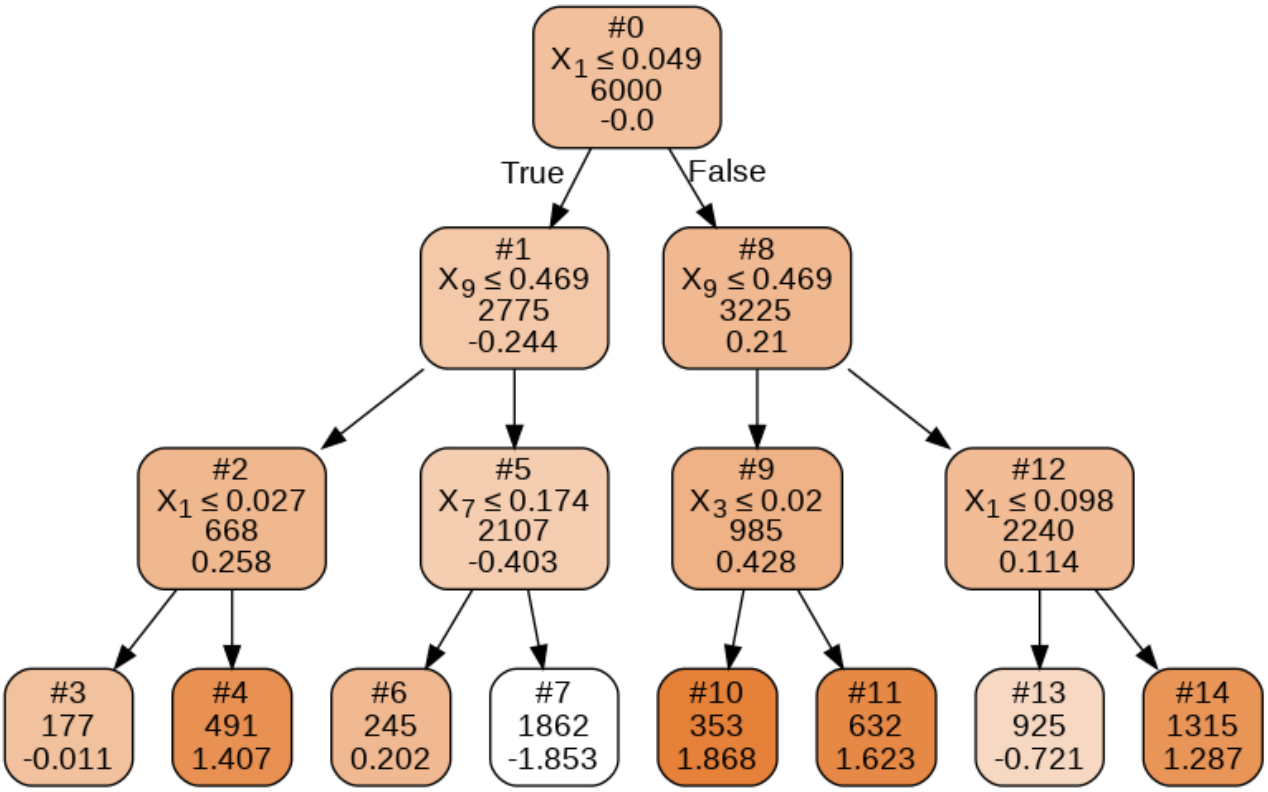
Παρατηρείται ότι το σφάλμα και στις δύο περιπτώσεις μειώνεται. Όπως είναι λογικό το σφάλμα του συνόλου εκπαίδευσης είναι λίγο μικρότερο από αυτό του συνόλου αξιολόγησης, αφού το μοντέλο εργάζεται πάνω στο σύνολο εκπαίδευσης και αναμένεται μικρότερο σφάλμα, σε σχέση με αυτό του συνόλου αξιολόγησης.



Οπτικοποίηση ενός Δένδρου του Δάσους:

Παρακάτω παρατίθεται σε μορφή εικόνας ο 1ος εκτιμητής, από τα 400 δένδρα της GradientBoost. Από τους παρακάτω κόμβους καταλαβαίνουμε τα εξής:

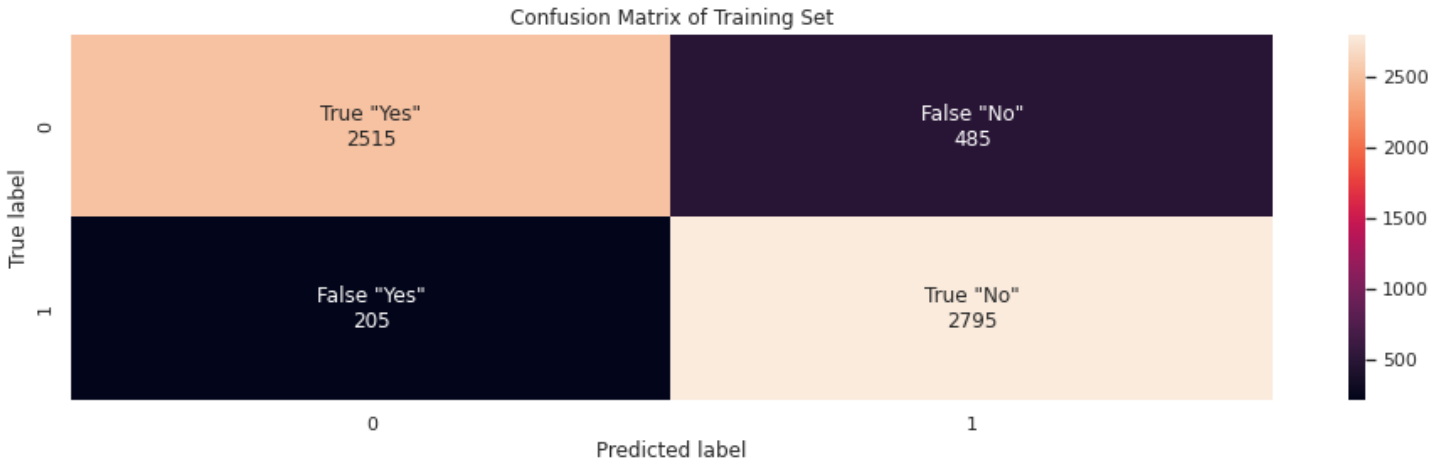
- Το node#0 είναι η ρίζα του δένδρου η οποία μας μας λέει πως περιέχονται σε αυτή 6000 στοιχεία.
- Στο node#1 που είναι το αριστερό παιδί της ρίζας βλέπουμε πως έχουν ταξινομηθεί 2775 στοιχεία, ενώ στο node#8, το δεξί παιδί της ρίζας βλέπουμε πως έχουμε 3225 στοιχεία ταξινομημένα στην κλάση «No».
- Όμοια και για τους υπόλοιπους κόμβους.



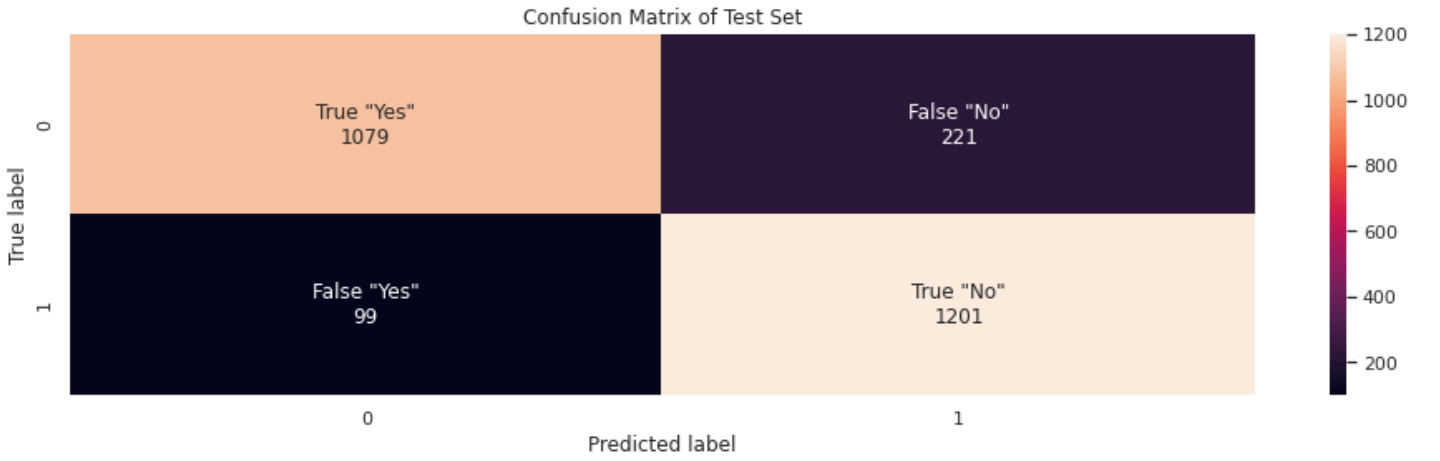
Αξιολόγηση Μεθόδου

1) Confusion Matrix

Παρακάτω, παρατίθεται ο Confusion Matrix για το **Training set**:



Παρακάτω, παρατίθεται ο Confusion Matrix για το **Test set** :



2) Accuracy Score

Το Accuracy Score είναι από τα πιο γνωστά μέτρα αξιολόγησης ενός machine learning μοντέλου. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Accuracy} = \frac{True\ Positives + True\ Negatives}{Total}$$

$$Acc_{Train} = \frac{2515 + 2795}{6000} = 0{,}89$$

$$Acc_{Test} = \frac{1079 + 1201}{2600} = 0{,}88$$

	Training Set	**Test Set**
Accuracy	0.89	0.88

3) Precision

Το precision ποσοτικοποιεί τον αριθμό των προβλέψεων της κλάσης Positives, που όντως ανήκουν στην κλάση Positives. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Pr_{Train} = \frac{2795}{2795 + 485} = 0{,}85$$

$$Pr_{Test} = \frac{1201}{1201 + 221} = 0{,}84$$

	Training Set	**Test Set**
Precision	0.85	0.84

4) Recall or Sensitivity

Το recall ποσοτικοποιεί τον αριθμό των προβλέψεων της κλάσης Positives που έγιναν από όλα τα δεδομένα που ανήκουν στην κλάση Positives. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$Re_{Train} = \frac{2795}{2795 + 205} = 0{,}93$$

$$Re_{Test} = \frac{1201}{1201 + 99} = 0{,}92$$

	Training Set	Test Set
Recall	0.93	0.92

5) F1-Score

Το F1-score μετράει την αποδοτικότητα του μοντέλου λαμβάνοντας υπόψη και το recall και το precision. Καθώς τα Precision & Recall από μόνα τους δεν επαρκούν για την αξιολόγηση του μοντέλου, αυτός ο συνδυασμός τους είναι πιο αντικειμενικός. Λαμβάνει τιμές μεταξύ 0 και 1. ΄Οσο μεγαλύτερη η τιμή του, τόσο καλύτερο το μοντέλο.

$$\boldsymbol{F1 - Score} = \frac{2 \cdot Precison \cdot Recall}{Precision + Recall}$$

$$F1_{Train} = \frac{2 \cdot 0{,}85 \cdot 0{,}93}{0{,}85 + 0{,}93} = 0{,}89$$

$$F1_{Test} = \frac{2 \cdot 0{,}84 \cdot 0{,}92}{0{,}84 + 0{,}92} = 0{,}88$$

	Training Set	Test Set
F1- Score	0.89	0.88

6) Cross Validation Score

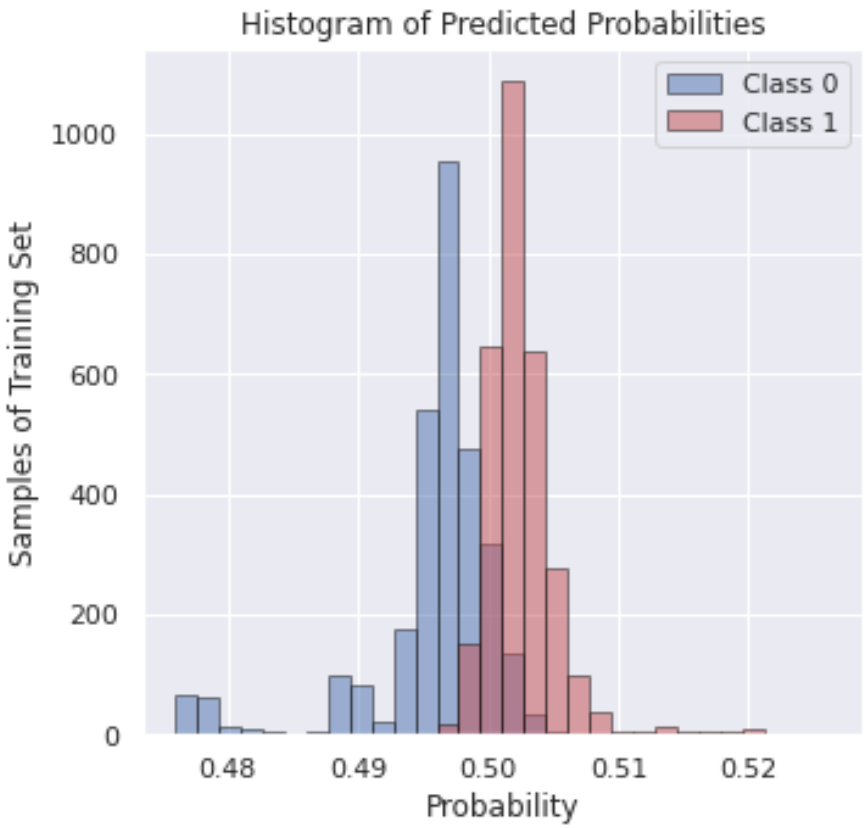
Επιπλέον, μία σημαντική μετρική για να αξιολογήσουμε το μοντέλο μας είναι ο εκτιμητής **Cross Validation**. Πιο συγκεκριμένα, υπολογίστηκε το *Cross_validation_score*. Το σύνολο δεδομένων διαιρείται σε υποσύνολα, κάθε ένα από τα οποία περιέχει διαφορετικές παρατηρήσεις. Η επιλογή των υποσυνόλων είναι τυχαία. Ένα από τα υποσύνολα χρησιμοποιείται ως test set και τα υπόλοιπα συνενώνονται και δημιουργούν το training set. Το μοντέλο εκπαιδεύεται χρησιμοποιώντας το training set και δοκιμάζεται έναντι του test set. Η διαδικασία επαναλαμβάνεται, κάθε φορά χρησιμοποιώντας ένα διαφορετικό σύνολο ως test set και τα υπόλοιπα ως training set. Στο τέλος υπολογίζεται η μέση επίδοση του μοντέλου. Η μέθοδος μπορεί να διαφοροποιηθεί ως προς το πλήθος των τμημάτων. Γενικότερα: Cross Validation k τμημάτων, όπου k συμβολίζει τον αριθμό των δημιουργημένων υποσυνόλων και των επαναλήψεων.

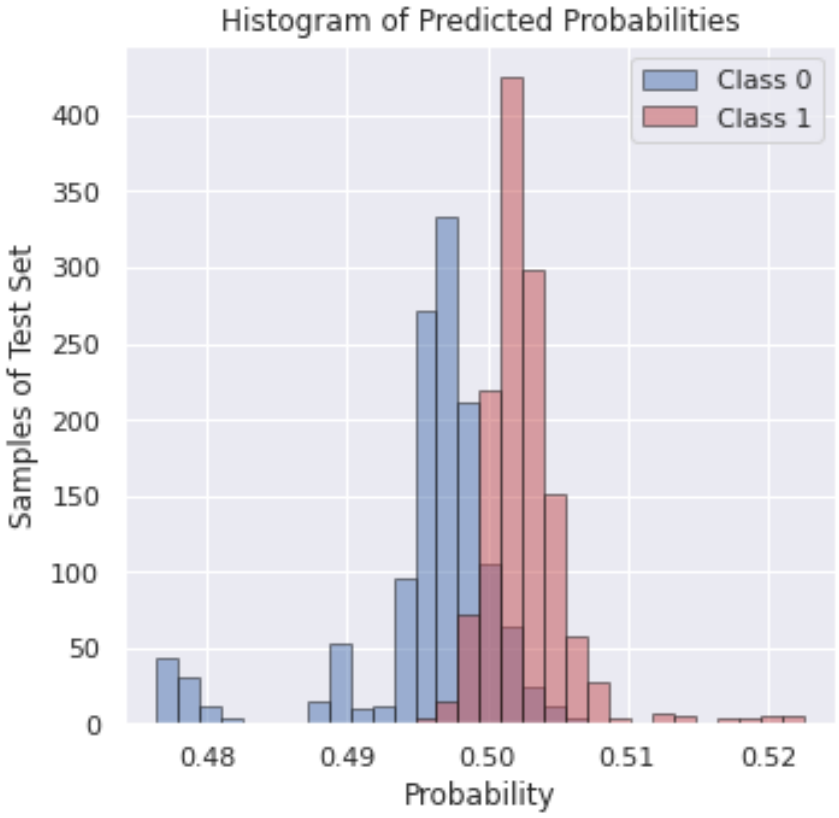
Παρακάτω παρουσιάζεται το Cross Validation Score για το μοντέλο με τα 400 stumps. Στη προαναφερθέν παράμετρο k δόθηκε ο αριθμός 10(είναι και το πιο σύνηθες).

```
0.88 accuracy with a standard deviation of 0.02
```

7)

Μέσω της *predict_proba()*, δίνεται η πιθανότητα μια παρατήρησης να ανήκει στην κλάση 0 (κλάση "Yes") και αντίστοιχα στην κλάση 1(απλά αφαιρώντας από το 1 την αντίστοιχη πιθανότητα). Κρατώντας την μία στήλη του output της εντολής, στα παρακάτω ιστογράμματα φαίνονται οι προβλεπόμενες πιθανότητες των δειγμάτων του Training Set & του Test Set να ανήκουν στην κάθε κλάση. Όσο πιο κοντά στο 0 προβλέπεται η κλάση " Yes", ενώ αντίστοιχα όσο πιο κοντά στο 1, η κλάση "No".


Histogram of Predicted Probabilities
Class 0
Class 1
Samples of Training Set
0
200
400
600
800
1000
0.48
0.49
0.50
0.51
0.52
Probability


Histogram of Predicted Probabilities
Class 0
Class 1
Samples of Test Set
0
50
100
150
200
250
300
350
400
0.48
0.49
0.50
0.51
0.52
Probability

Αποτελέσματα για διαφορετικό Test Set

Έγινε μια δοκιμή της μεθόδου χρησιμοποιώντας διαφορετικό σύνολο αξιολόγησης. Συγκεκριμένα, χρησιμοποιήθηκαν οι παρατηρήσεις της υπερέχουσας κλάσης “Yes” που παραλείψαμε λόγω του προβλήματος της ανομοιογένειας. Μέσα από αυτό το σύνολο αξιολόγησης μπορούμε να συμπεράνουμε κατά πόσο όλες οι παρατηρήσεις που ανήκουν στην κλάση “Yes” ταξινομούνται σωστά. Το Accuracy για το νέο σύνολο αξιολόγησης είναι 0,83.

Κεφάλαιο 5 : Νευρωνικά Δίκτυα

5.1 Θεωρία Νευρωνικών Δικτύων

Τα νευρωνικά δίκτυα χρησιμοποιούνται σε πολλούς τομείς της μηχανικής μάθησης, όπως και σε προβλήματα ταξινόμησης. Συγκεκριμένα, στο πρόβλημα μας χρησιμοποιήθηκαν τα MLP Νευρωνικά Δίκτυα (Multilayer Perceptron) και ανήκουν στη κατηγορία αλγορίθμων μηχανικής μάθησης υπό επίβλεψη (*supervised learning*).

Ορισμός

Ένα νευρωνικό δίκτυο είναι ένα δίκτυο ή κύκλωμα νευρώνων, ή με τη σύγχρονη έννοια, ένα τεχνητό νευρωνικό δίκτυο, αποτελούμενο από τεχνητούς νευρώνες ή κόμβους. Έτσι, ένα νευρωνικό δίκτυο είναι είτε ένα βιολογικό νευρωνικό δίκτυο, που αποτελείται από βιολογικούς νευρώνες, είτε ένα τεχνητό νευρωνικό δίκτυο, για την επίλυση προβλημάτων τεχνητής νοημοσύνης (AI). (**Πηγή**: *Βικιπαίδεια*)

Αρχιτεκτονική Δικτύου

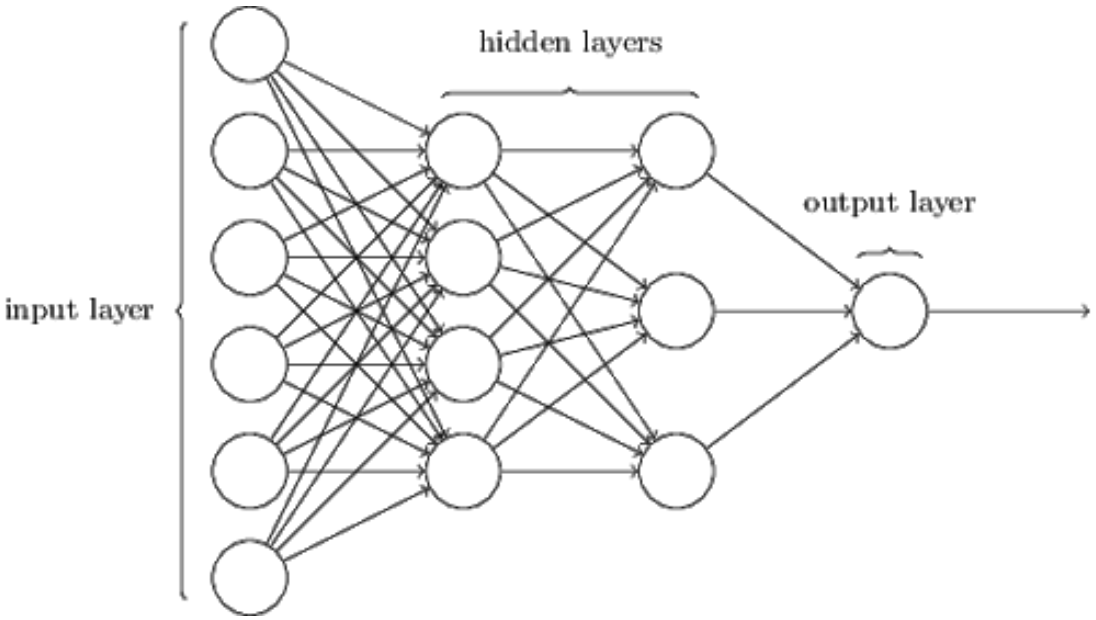
Ένα νευρωνικό δίκτυο αποτελείται:

- από το στρώμα εισόδου (***input layer***), το οποίο δέχεται τα features και περνάει τις απαραίτητες πληροφορίες στα hidden layers, χωρίς στο ίδιο να πραγματοποιείται κάποιος υπολογισμός

- τα κρυφά στρώματα (***hidden layers***) , τα οποία δέχονται την πληροφορία από το στρώμα εισόδου, κάνουν όλους τους απαραίτητους υπολογισμούς και τέλος μεταφέρουν το αποτέλεσμα στο στρώμα εξόδου

- το στρώμα εξόδου (***output layer***) , το οποίο μεταφέρει το αποτέλεσμα του νευρωνικού στον «έξω κόσμο»

Κάθε ένα απ' αυτά εμπεριέχει τους ***νευρώνες*** (ή κόμβους) οι οποίοι είναι μεταξύ τους συνδεδεμένοι.



(**Πηγή**: "*http://neuralnetworksanddeeplearning.com/chap1.html*")

Οι κόμβοι του νευρωνικού δικτύου χρησιμοποιούν μια ***συνάρτηση ενεργοποίησης f(·)*** (***activation function***), της μορφής $f(w \cdot x + b)$, όπου w το διάνυσμα βαρών, b το *κατώφλι* (bias or threshold) και x είναι το διάνυσμα εισόδου, το οποίο είτε είναι ένα διάνυσμα (row) των δεδομένων, αν βρισκόμαστε στο στρώμα εισόδου, είτε αποτελεί το output του προηγούμενου στρώματος.

Τα στρώματα του δικτύου είναι πλήρως συνδεδεμένα, οι παράμετροι κάθε κόμβου είναι ανεξάρτητοι από τους άλλους, και γι'αυτό κάθε κόμβος επιστρέφει ένα μοναδικό σύνολο βαρών. Επιπλέον, σημαντικό είναι να οριστεί και μία ***συνάρτηση κόστους*** (***loss function***) ώστε να μπορεί να γίνει εκτίμηση της απώλειας του μοντέλου και τα βάρη να ενημερώνονται ώστε να μειώνεται αυτή στην επόμενη επανάληψη.

Τέλος, χρειάζεται μία διαδικασία βελτιστοποίησης, και συγκεκριμένα η ***μέθοδος απότομης καθόδου*** (***gradient descent***) μέσω της οποίας υπολογίζονται τα βάρη ώστε να επιτυγχάνεται ελαχιστοποίηση της συνάρτησης κόστους.

- **Συνάρτηση Ενεργοποίησης**

Ορισμός

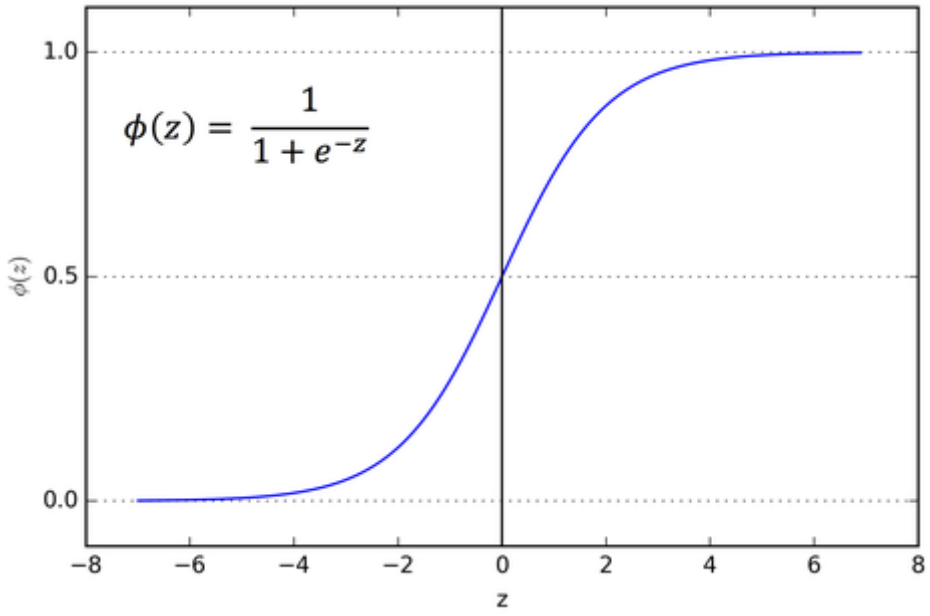
Η συνάρτηση ενεργοποίησης αποφασίζει, εάν ένας νευρώνας πρέπει να «ενεργοποιηθεί» ή όχι, υπολογίζοντας το βεβαρυμμένο άθροισμα και προσθέτοντας επιπλέον το bias σε αυτό. Η συνάρτηση ενεργοποίησης εισάγει τη *μη γραμμικότητα* στην έξοδο ενός νευρώνα.

Ένας νευρώνας λειτουργεί λαμβάνοντας υπόψη τα βάρη, το bias και την συνάρτηση ενεργοποίησης του. Κατά την εκπαίδευση του νευρωνικού δικτύου, ενημερώνονται τα βάρη και τα κατόφλια των νευρώνων με βάση το σφάλμα της εξόδου. Αυτή η διαδικασία είναι γνωστή ως *back propagation*. Οι συναρτήσεις ενεργοποίησης επιτρέπουν στη τελευταία λαμβάνοντας υπόψη το σφάλμα, να ενημερώσει τις παραμέτρους.

Μη γραμμικές Συναρτήσεις:

1) Λογιστική ή Σιγμοειδής

Η Σιγμοειδής συνάρτηση μοιάζει σχηματικά με το γράμμα S. Ο πιο βασικός λόγος που χρησιμοποιείται είναι γιατί οι τιμές της κυμαίνονται μεταξύ του 0 και του 1. Επομένως, χρησιμεύει πολύ στα μοντέλα που ως έξοδο θέλουν να προβλέψουν μια πιθανότητα.



(**Πηγή**:"*https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6*")

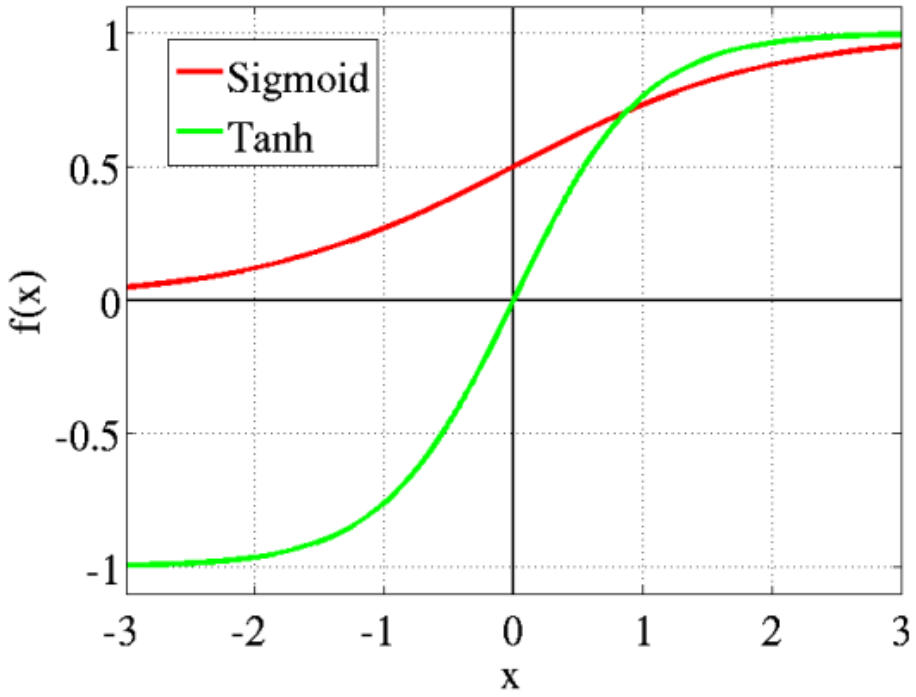
Η συνάρτηση είναι παραγωγίσιμη, πράγμα που σημαίνει ότι μπορούμε να βρούμε την κλίση της σιγμοειδούς καμπύλης σε οποιοδήποτε σημείο. Επίσης, είναι μονότονη, αλλά η παράγωγος της όχι.

Αξίζει να αναφερθεί και η συνάρτηση *softmax*, μια πιο γενικευμένη συνάρτηση της λογιστικής, που συνήθως προτιμάται σε προβλήματα multiclass ταξινόμησης.

2) Tanh ή Υπερβολική Εφαπτομένη

Η tanh είναι μια βελτιωμένη έκδοση της σιγμοειδούς με εύρος τιμών από το -1 εώς το 1. Είναι παραγωγίσιμη και μονότονη, αλλά η παράγωγος της δεν είναι μονότονη.

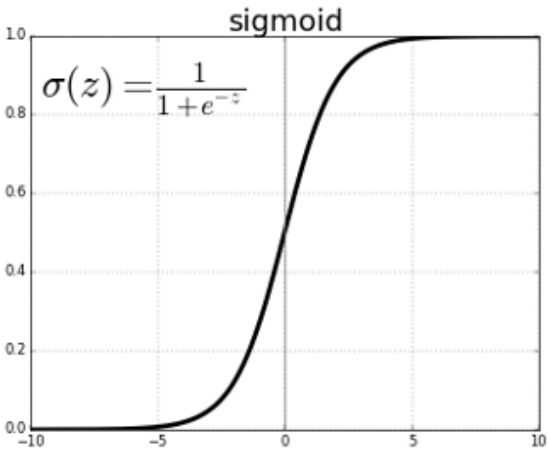
$$f(z) = \frac{2}{1 + e^{-2z}} = 2sigmoid(2x) - 1$$

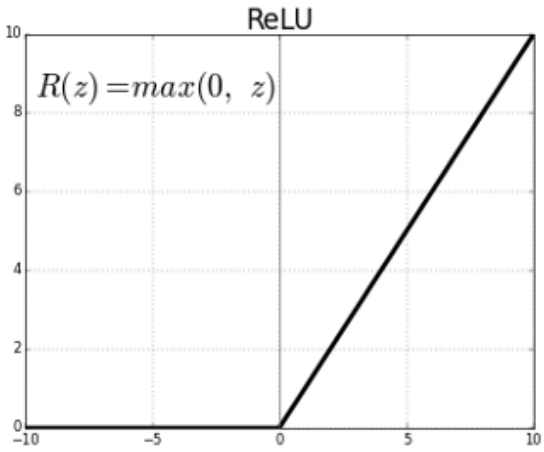


(**Πηγή**:"*https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6*")

3) ReLU

Η ReLU είναι ίσως η πιο συχνή συνάρτηση ενεργοποίησης καθώς χρησιμοποιείται ευρέως στα convolutional neural networks ή στη βαθιά μηχανική μάθηση. Παίρνει τιμές στο [0,∞) και η ίδια αλλά και η παράγωγος της είναι παραγωγίσιμες και μονότονες.



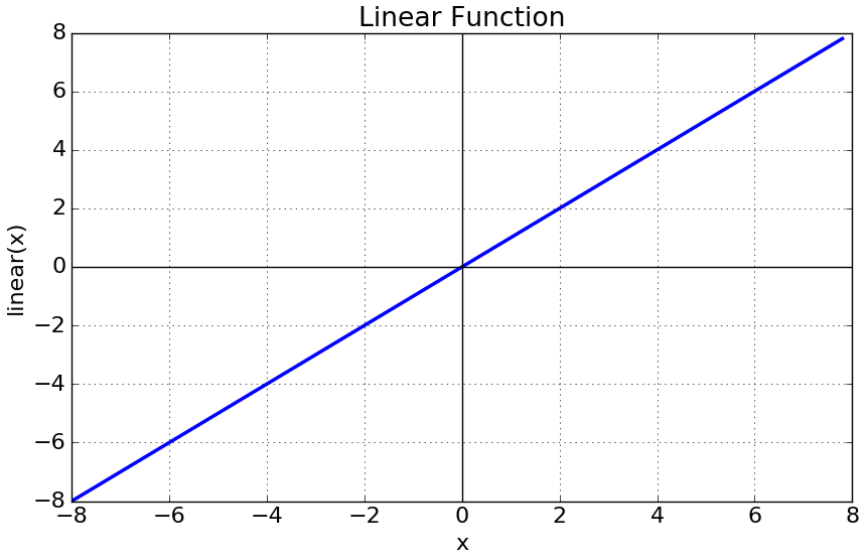


(**Πηγή**:"*https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6*")

Γραμμική Συνάρτήση:

Συνήθως σε προβλήματα παλινδρόμησης χρησιμοποιείται η γραμμική συνάρτηση ενεργοποίησης (στο τελευταίο στρώμα). Προφανώς παίρνει τιμές από $(-\infty, +\infty)$.

$$f(z) = z$$



(**Πηγή**:"*https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6*")

- **Συνάρτηση Κόστους**

Τα νευρωνικά δίκτυα εκπαιδεύονται χρησιμοποιώντας τον αλγόριθμο gradient descent και συνεπώς πρέπει να εκτιμάται επανειλημμένα το σφάλμα για την τρέχουσα κατάσταση του μοντέλου. Αυτό απαιτεί την επιλογή μιας συνάρτησης κόστους, η οποία μπορεί να χρησιμοποιηθεί για την εκτίμηση σφάλματος του μοντέλου, έτσι ώστε τα βάρη να μπορούν να ενημερωθούν κατά την επόμενη αξιολόγηση με σκοπό τη μείωση του σφάλματος.

Ανάλογα με τη φύση του προβλήματος υπάρχουν διάφορες συναρτήσεις κόστους. Για παράδειγμα, σε προβλήματα παλινδρόμησης είναι γνωστά το μέσο τετραγωνικό σφάλματος (Mean Squared Error Loss), το λογαριθμικό μέσο τετραγωνικό σφάλμα (Mean Squared Logarithmic Error Loss) και το Mean Absolute Error Loss (MAE).

Στα προβλήματα δυαδικής ταξινόμησης σκοπός είναι η πρόβλεψη της κλάσης 0 ή 1 και συχνά και η πρόβλεψη της πιθανότητας ένα δείγμα να ανήκει σε κάθε κλάση. Η πιο γνωστή συνάρτηση κόστους για τα προβλήματα δυαδικής ταξινόμησης είναι η *cross entropy*. Η cross entropy μετράει τη διαφορά μεταξύ δύο κατανομών πιθανότητας για μια δεδομένη τυχαία μεταβλητή ή ένα σύνολο συμβάντων. Επίσης, όταν χρησιμοποιείται η binary cross entropy, πρέπει στο στρώμα εξόδου να χρησιμοποιεί τη σιγμοειδή συνάρτηση ενεργοποίησης και το εύρος του output να είναι από 0 εώς 1. Η συνάρτηση κόστους θα επιστρέφει υψηλές τιμές για κακές προβλέψεις, ενώ χαμηλές για καλές προβλέψεις.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\big(p(y_i)\big) + (1 - y_i) \cdot \log(1 - p(y_i)),$$

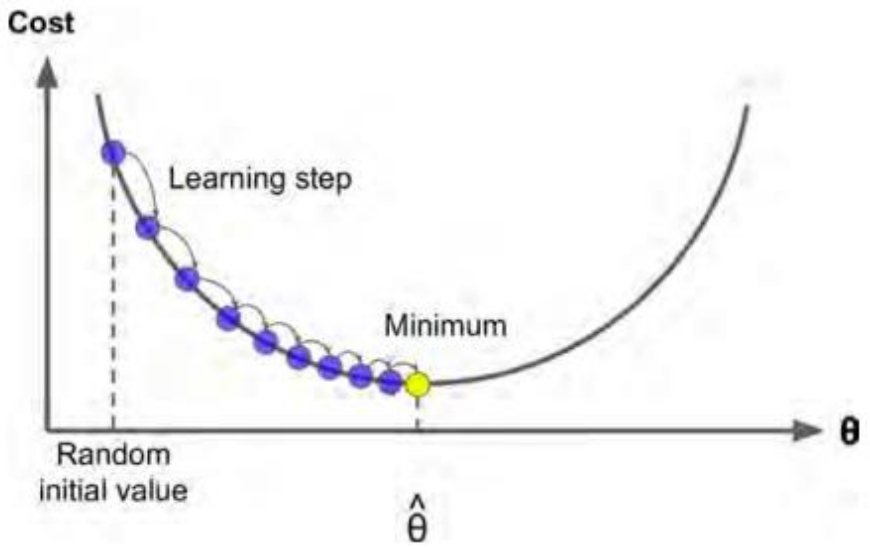
Όπου y η ετικέτα, και p(y) προβλεπόμενη πιθανότητα ενός sample να ταξινομείται στην αντίστοιχη κλάση/ετικέτα.

Υπάρχουν και άλλες συναρτήσεις κόστους που χρησιμοποιούνται σε προβλήματα δυαδικής ταξινόμησης όπως *hinge loss* & *squared hinge loss*.

- **Μέθοδος Απότομης Καθόδου**

Η μέθοδος απότομης καθόδου (Gradient Descent) είναι ένας αλγόριθμος βελτιστοποίησης, ο οποίος βρίσκει την βέλτιστη λύση σε ένα ευρύ φάσμα προβλημάτων. Η γενική ιδέα της Gradient Descent είναι η επαναληπτική τροποποίηση παραμέτρων με σκοπό την ελαχιστοποίηση της συνάρτησης κόστους.

Συγκεκριμένα, γίνεται τυχαία αρχικοποίηση τιμών σε μία παράμετρο θ και στη συνέχεια αυτή βελτιώνεται σταδιακά είτε κάνοντας πολύ μικρά βήματα όταν φτάνει τη βέλτιστη τιμή, είτε μεγάλα όταν είναι μακριά αυτής.



(Πηγή: Βιβλίο *Hands-On Machine Learning with Scikit-Learn & TensorFlow* p.111)

5.2 Υλοποίηση μέσω Python

1. Ορισμός Μοντέλου

Για την κατασκευή των μοντέλων χρησιμοποιήθηκε η βιβλιοθήκη *Keras* της Python. Τα μοντέλα της ορίζονται ως ένα *Sequence* από στρώματα (*layers*). Ουσιαστικά δημιουργούμε ένα *Sequential Model* στο οποίο προστίθενται στρώματα, ένα κάθε φορά, ανάλογα την αρχιτεκτονική του δικτύου.

Αρχικά δόθηκε στο όρισμα input_dim ο αριθμός 53, όσες οι μεταβλητές του προβλήματος (κατηγορικές & αριθμητικές). Αυτό το όρισμα αφορά το στρώμα εισόδου και καθορίζει τη δημιουργία του. Στη συνέχεια δημιουργήθηκε ένα hidden layer με 4 κόμβους και το output layer, χρησιμοποιώντας την κλάση *Dense*. Μέσω αυτής καθορίζεται ο αριθμός νευρώνων ή κόμβων στο εκάστοτε στρώμα, καθώς και η συνάρτηση ενεργοποίησης.

Χρησιμοποιήθηκε η συνάρτηση ενεργοποίησης ReLU για το κρυφό στρώμα, ενώ για το στρώμα εξόδου η Sigmoid, ώστε να διασφαλιστεί ότι η έξοδος του δικτύου κυμαίνεται μεταξύ 0 και 1 και να θεωρηθεί πιθανότητα κλάση με προεπιλεγμένο όριο ταξινόμησης 0,5.

2. Compiling Μοντέλου

Αφού οριστεί το μοντέλο, «μεταγλωττίζεται» με τη βοήθεια βιβλιοθηκών του πακέτου *TensorFlow*, που μας παρέχει η Python. Κατά την εφαρμογή του, καθορίζουμε κάποια ορίσματα που απαιτούνται για την εκπαίδευση του δικτύου. Ουσιαστικά, σκοπός είναι να εκπαιδευτεί ένα δίκτυο βρίσκοντας το καλύτερο σύνολο βαρών για το οποίο η συνάρτηση κόστους μειώνεται.

Σχετικά με την τελευταία, επιλέγεται η Cross Entropy συνάρτηση, η οποία παρέχεται από την Keras, με το όνομα '*binary_crossentropy*', και είναι ιδανική για προβλήματα δυαδικής ταξινόμησης.

Επιπλέον, σχετικά με τη διαδικασία βελτιστοποίησης, θα χρησιμοποιηθεί η '*adam*', μέθοδος stochastic gradient descent. Χρησιμοποιείται συχνά η συγκεκριμένη μέθοδος απότομης καθόδου, καθώς δίνει καλά αποτελέσματα σε ένα ευρύ φάσμα προβλημάτων.

3. Fit Μοντέλου

Αφού έχει οριστεί και μεταγλωττιστεί το μοντέλο, γίνεται ο τελικός υπολογισμός. Ουσιαστικά το μοντέλο εφαρμόζεται πάνω στα δεδομένα και δίνονται τιμές στα epochs & στο batch size. Η διαδικασία εκπαίδευσης επαναλαμβάνεται για όσα epochs (επαναλήψεις) ορίσει ο χρήστης και κάθε epoch χωρίζεται σε batches.

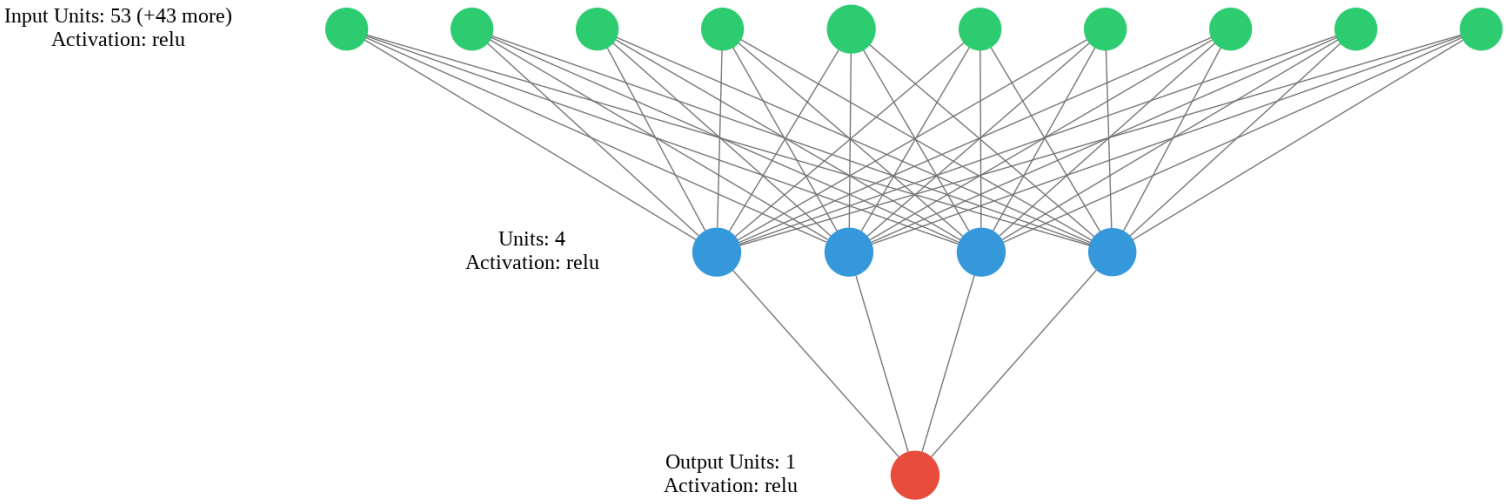
- Batch Size: Αφορά τον αριθμό των δειγμάτων (δηλ. rows) του συνόλου εκπαίδευσης που υποβλήθηκαν σε επεξεργασία πριν την ενημέρωση των παραμέτρων του μοντέλου.
- Epoch: Ο αριθμός των epochs αφορά των αριθμό των επαναλήψεων, που ο αλγόριθμος εκπαίδευσης «θα περάσει» από ολόκληρο το σύνολο εκπαίδευσης.

Σε μία επανάληψη δίνεται η δυνατότητα σε κάθε ένα δείγμα του συνόλου εκπαίδευσης να ανανεώσει τις παραμέτρους του μοντέλου. Μάλιστα, ο αλγόριθμος που σε ένα epoch που έχει μόνο ένα batch, ονομάζεται *batch gradient descent learning algorithm*.

Στο συγκεκριμένο πρόβλημα, δόθηκε στο όρισμα *epochs*=600, και στο *batch_size*=10.

❖ Οπτικοποίηση Νευρωνικού Δικτύου

Παρακάτω φαίνεται μία ποιοτική οπτικοποίηση του νευρωνικού δικτύου που κατασκευάστηκε.



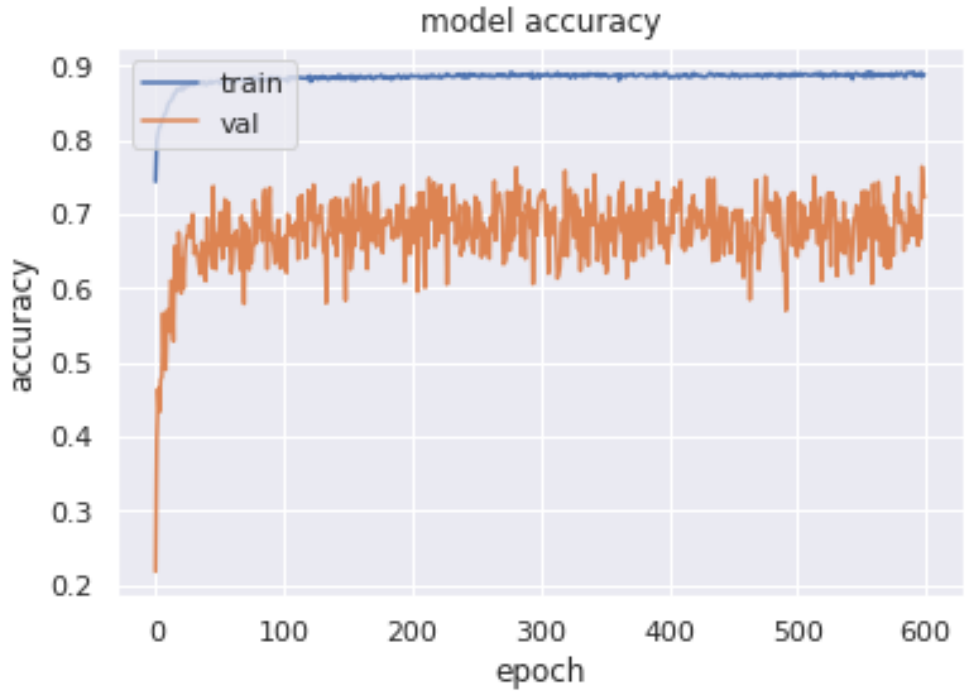
4. Αξιολόγηση Μοντέλου

1) Η Keras μέσω του *History* παρέχει τη δυνατότητα ανάκλησης μετρικών (όπως Accuracy, Loss) του μοντέλου κατά τη διάρκεια της εκπαίδευσης του για κάθε epoch. Επιπλέον, προαιρετικά μπορεί να εμφανίσει και το Accuracy , Loss για το Validation Set, αν το επιλέξει ο χρήστης.

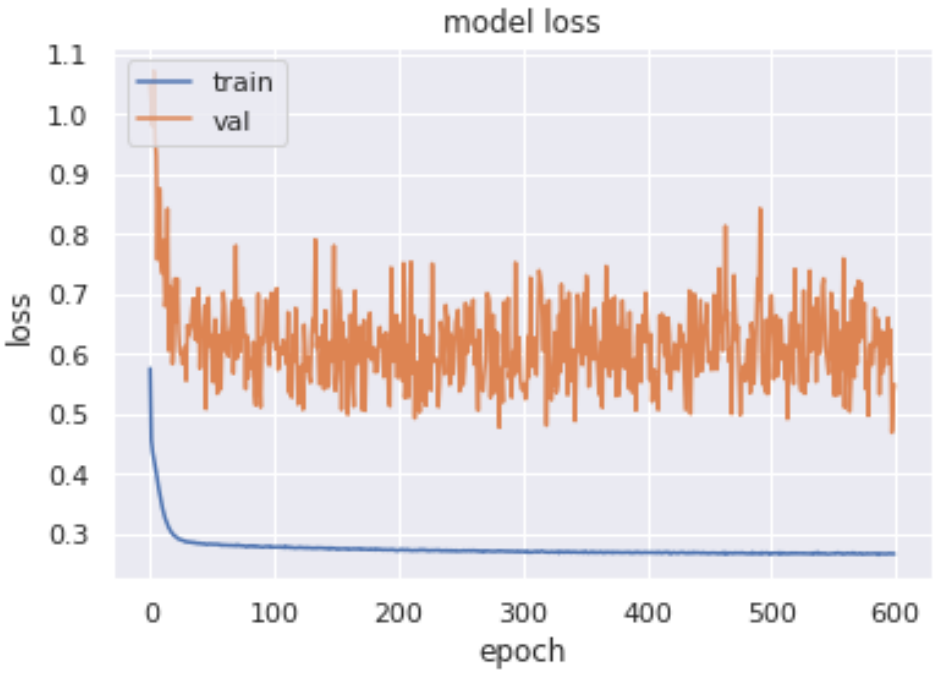
Το *History* δημιουργείται μετά από τις κλήσεις της συνάρτησης *fit()* που χρησιμοποιείται για την εκπαίδευση του μοντέλου. Οι μετρήσεις αποθηκεύονται σε ένα dictionary.

Παρακάτω φαίνονται τα αποτελέσματα του *history* με δύο γραφικές παραστάσεις.

- Accuracy των Training & Validation Datasets σε σχέση με τα epochs

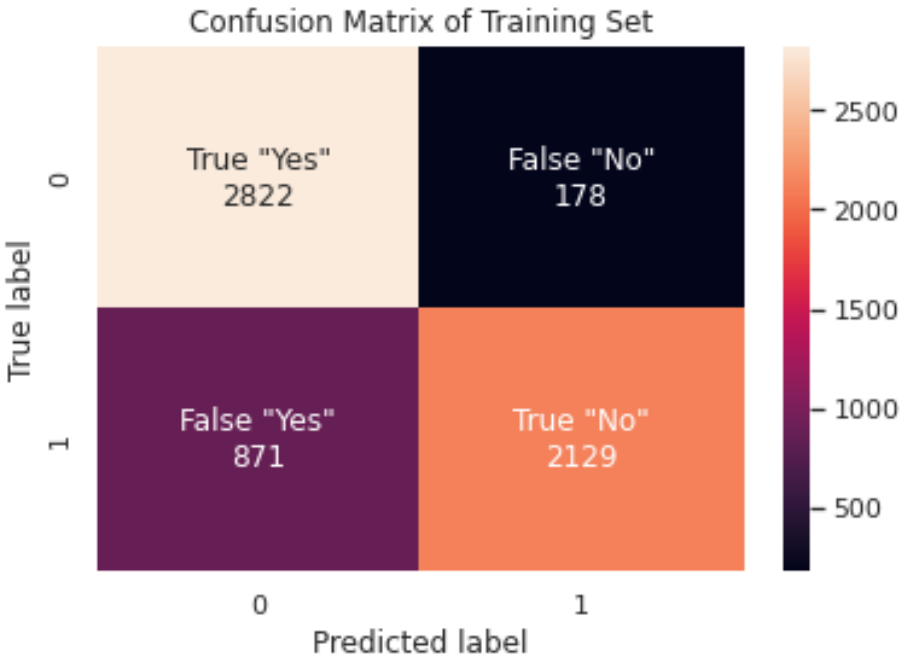


- Loss των Training & Validation Datasets σε σχέση με τα epochs

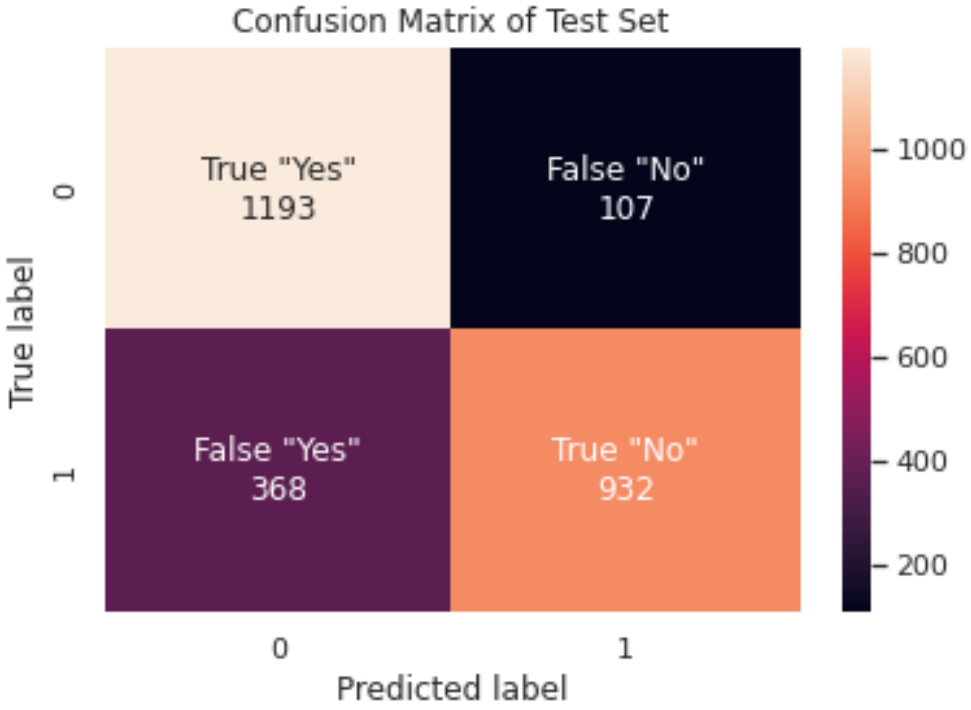


2) Confusion Matrix

Παρακάτω, παρατίθεται ο Confusion Matrix για το **Training set**:



Παρακάτω, παρατίθεται ο Confusion Matrix για το **Test set** :



3) Accuracy Score

Το Accuracy Score είναι από τα πιο γνωστά μέτρα αξιολόγησης ενός machine learning μοντέλου. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Accuracy} = \frac{True\ Positives + True\ Negatives}{Total}$$

$$Acc_{Train} = \frac{2822 + 2129}{6000} = 0{,}83$$

$$Acc_{Test} = \frac{1193 + 932}{2600} = 0{,}82$$

	Training Set	**Test Set**
Accuracy	0.83	0.82

4) Precision

Το precision ποσοτικοποιεί τον αριθμό των προβλέψεων της κλάσης Positives, που όντως ανήκουν στην κλάση Positives. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Pr_{Train} = \frac{2129}{2129 + 178} = 0{,}92$$

$$Pr_{Test} = \frac{932}{932 + 107} = 0{,}90$$

	Training Set	Test Set
Precision	0.92	0.90

5) Recall or Sensitivity

Το recall ποσοτικοποιεί τον αριθμό των προβλέψεων της κλάσης Positives που έγιναν από όλα τα δεδομένα που ανήκουν στην κλάση Positives. Υπολογίζεται από τη σχέση:

$$\boldsymbol{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$Re_{Train} = \frac{2129}{2129 + 871} = 0{,}71$$

$$Re_{Test} = \frac{932}{932 + 368} = 0{,}72$$

	Training Set	Test Set
Recall	0.71	0.72

6) F1-Score

Το F1-score μετράει την αποδοτικότητα του μοντέλου λαμβάνοντας υπόψη και το recall και το precision. Καθώς τα Precision & Recall από μόνα τους δεν επαρκούν για την αξιολόγηση του μοντέλου, αυτός ο συνδυασμός τους είναι πιο αντικειμενικός. Λαμβάνει τιμές μεταξύ 0 και 1. Όσο μεγαλύτερη η τιμή του, τόσο καλύτερο το μοντέλο.

$$\boldsymbol{F1 - Score} = \frac{2 \cdot Precison \cdot Recall}{Precision + Recall}$$

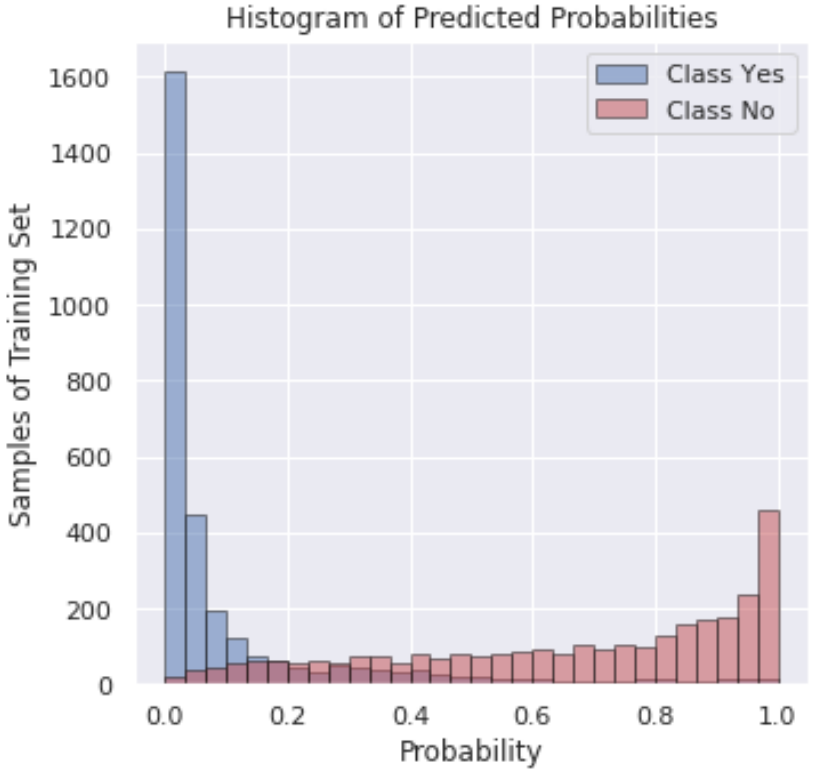
$$F1_{Train} = \frac{2 \cdot 0{,}71 \cdot 0{,}92}{0{,}71 + 0{,}92} = 0{,}80$$

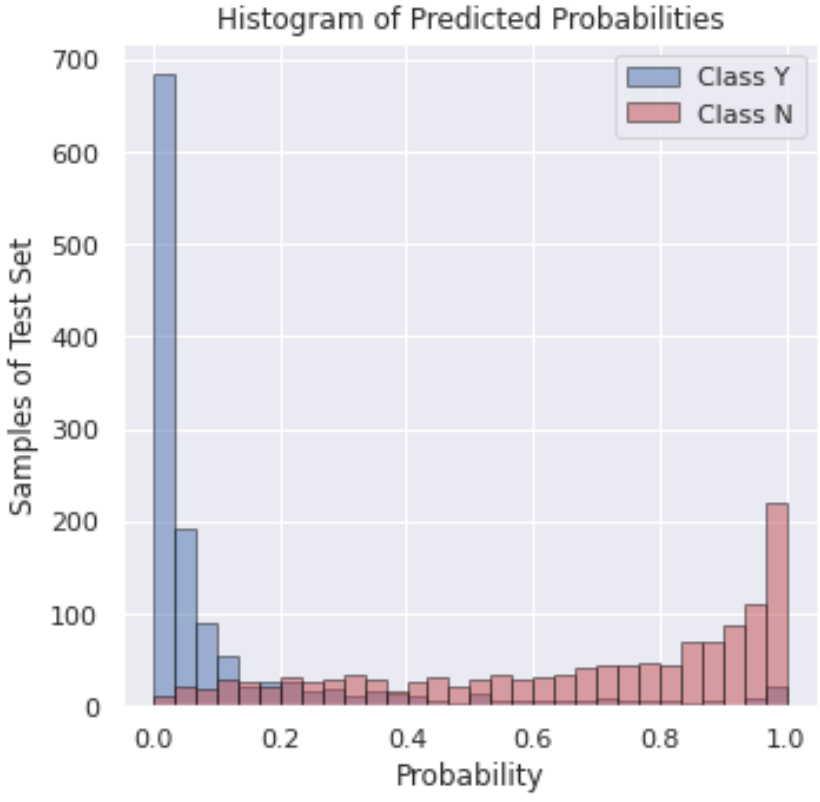
$$F1_{Test} = \frac{2 \cdot 0{,}72 \cdot 0{,}90}{0{,}72 + 0{,}90} = 0{,}80$$

	Training Set	Test Set
F1- Score	0.80	0.80

7)

Μέσω της *predict_proba()*, δίνεται η πιθανότητα μια παρατήρησης να ανήκει στην κλάση 0 (κλάση "Yes") και αντίστοιχα στην κλάση 1(απλά αφαιρώντας από το 1 την αντίστοιχη πιθανότητα). Κρατώντας την μία στήλη του output της εντολής, στα παρακάτω ιστογράμματα φαίνονται οι προβλεπόμενες πιθανότητες των δειγμάτων του Training Set & του Test Set να ανήκουν στην κάθε κλάση. Όσο πιο κοντά στο 0 προβλέπεται η κλάση " Yes", ενώ αντίστοιχα όσο πιο κοντά στο 1, η κλάση "No".


Histogram of Predicted Probabilities
Class Yes
Class No
Samples of Training Set
1600
1400
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0
Probability


Histogram of Predicted Probabilities
Class Y
Class N
Samples of Test Set
700
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
Probability

Αποτελέσματα για διαφορετικό Test Set

Έγινε μια δοκιμή της μεθόδου χρησιμοποιώντας διαφορετικό σύνολο αξιολόγησης. Συγκεκριμένα, χρησιμοποιήθηκαν οι παρατηρήσεις της υπερέχουσας κλάσης “Yes” που παραλείψαμε λόγω του προβλήματος της ανομοιογένειας. Μέσα από αυτό το σύνολο αξιολόγησης μπορούμε να συμπεράνουμε κατά πόσο όλες οι παρατηρήσεις που ανήκουν στην κλάση “Yes” ταξινομούνται σωστά. Το Accuracy για το νέο σύνολο αξιολόγησης είναι 0,58.

Κεφάλαιο 6 : Σύγκριση Μεθόδων

6.1 Θεωρία Νευρωνικών Δικτύων

Αφού ολοκληρώθηκαν οι μέθοδοι, παρουσιάζεται ένα summary table με τις μετρικές αξιολόγησης όλων των μεθόδων. Μέσω αυτού μπορεί να γίνει άμεση σύγκριση των μεθόδων LDA, AdaBoost, GradientBoost & MLP. Συγκεκριμένα, παρατίθενται τα Accuracy, Precision, Recall & F1-Score για το σύνολο εκπαίδευσης αλλά και για το σύνολο αξιολόγησης.

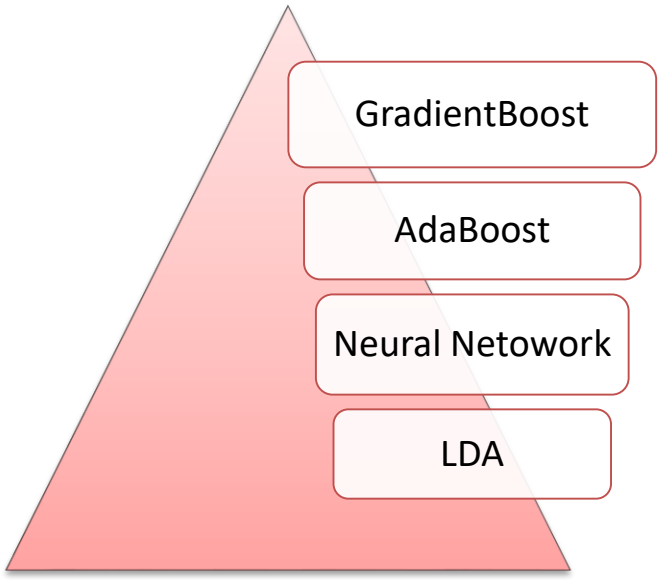
Μέθοδος	**Acc$_{train}$**	**Acc$_{test}$**	**Pr$_{train}$**	**Pr$_{test}$**	**Re$_{train}$**	**Re$_{test}$**	**F1$_{train}$**	**F1$_{test}$**
LDA	0,83	0,82	0,83	0,82	0,82	0,82	0,83	0,82
AdaBoost	0,88	0,87	0,87	0,87	0,89	0,88	0,88	0,87
GradientBoost	0,89	0,88	0,85	0,84	0,93	0,92	0,89	0,88
Neural Network	0,83	0,82	0,92	0,90	0,71	0,72	0,80	0,80

Όπως φαίνεται στον παραπάνω πίνακα, τα Accuracies των Adaboost & GradientBoost δείχνουν ότι είναι οι πιο αποτελεσματικές μέθοδοι καθώς έχουν τις πιο υψηλές τιμές. Αυτό σημαίνει πως ταξινόμησαν σωστά, συνολικά, περισσότερα δείγματα σε σχέση με τις άλλες μεθόδους.

Ωστόσο, όπως φαίνεται, το Precision των Νευρωνικών Δικτύων είναι το μεγαλύτερο, κι αυτό σημαίνει πως η ταξινόμηση στην κλάση "No", έχει μεγαλύτερη επιτυχία σε σχέση με τις άλλες μεθόδους. Ουσιαστικά, με την υλοποίηση ενός νευρωνικού δικτύου, ένα δείγμα (δηλ. ένας πελάτης της τράπεζας) που έχει ταξινομηθεί στη κλάση "No", κατά 92% πιθανότητα θα έχει ταξινομηθεί σωστά.

Επίσης, το Recall της GradientBoost είναι το υψηλότερο, συνεπώς για όσους έχουν ταξινομηθεί στην κλάση "No", το recall μας δείχνει πόσους έχουμε ταξινομήσει σωστά σε αυτή την κλάση στην πραγματικότητα.

Τέλος, από το F1-Score, λαμβάνοντας υπόψη και το precision & το recall, καταλήγουμε πως οι πιο αποδοτικές μέθοδοι είναι οι GradientBoost και AdaBoost.


GradientBoost
AdaBoost
Neural Netowork
LDA

Αναφορές

Κεφάλαιο 1

1) https://el.wikipedia.org/wiki/%CE%9C%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE_%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7

Κεφάλαιο 2

2) https://towardsdatascience.com/machine-learning-target-feature-label-imbalance-problem-and-solutions-98c5ae89ad0
3) https://towardsdatascience.com/histograms-and-density-plots-in-python-f6bda88f5ac0
4) https://repository.kallipos.gr/bitstream/11419/2968/1/02_chapter_03.pdf

Κεφάλαιο 3

5) https://towardsdatascience.com/linear-discriminant-analysis-in-python-76b8b17817c2
6) https://machinelearningmastery.com/linear-discriminant-analysis-for-dimensionality-reduction-in-python/
7) https://www.mygreatlearning.com/blog/linear-discriminant-analysis-or-lda/
8) https://www.csd.uwo.ca/~oveksler/Courses/CS434a_541a/Lecture8.pdf
9) https://machinelearningmastery.com/linear-discriminant-analysis-with-python/
10) https://laptrinhx.com/curse-of-dimensionality-1910417005/
11) https://stats.stackexchange.com/questions/31366/linear-discriminant-analysis-and-bayes-rule-classification/31384#31384

Κεφάλαιο 4

12) https://www.python-course.eu/Boosting.php
13) https://www.youtube.com/watch?v=UHBmv7qCey4&ab_channel=MITOpenCourseWare
https://www.stxnext.com/blog/machine-learning-from-the-woods-exploring-tree-based-ensemble-models-in-python
14) https://www.datasciencecentral.com/profiles/blogs/decision-tree-vs-random-forest-vs-boosted-trees-explained
15) https://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_twoclass.html?fbclid=IwAR3n4fDzh2WmhDtR5YCg5Hv9RqC08qOBDtRl6lMEnhga0IWqIyOu5iHldK8

16) https://towardsdatascience.com/boosting-and-adaboost-clearly-explained-856e21152d3e
17) https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/
18) https://machinelearningmastery.com/gradient-boosting-machine-ensemble-in-python/

Κεφάλαιο 5

19) https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
20) https://machinelearningmastery.com/binary-classification-tutorial-with-the-keras-deep-learning-library/
21) https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8
22) https://machinelearningmastery.com/how-to-make-classification-and-regression-predictions-for-deep-learning-models-in-keras/
23) https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
24) https://machinelearningmastery.com/how-to-make-classification-and-regression-predictions-for-deep-learning-models-in-keras/
25) https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6