ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Ανεύρεση και ανάλυση αισθήματος κριτικών και λημμάτων με τεχνικές μηχανικής μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επαμεινώνδας Σ. Αλεβίζος
Δήμητρα Γ. Αλεξανδρινάκη

Επιβλέπων : **Ιάκωβος Βενιέρης**
Καθηγητής Ε.Μ.Π.

Συνεπίβλεψη: Δρ. Σ. Καπελλάκη, ΕΔΙΠ Ε.Μ.Π.

Αθήνα, Μάρτιος 2020

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Ανεύρεση και ανάλυση αισθήματος κριτικών και λημμάτων με τεχνικές μηχανικής μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επαμεινώνδας Σ. Αλεβίζος
Δήμητρα Γ. Αλεξανδρινάκη

Επιβλέπων : Ιάκωβος Βενιέρης,
Καθηγητής Ε.Μ.Π.
Συνεπίβλεψη: Σ. Καπελλάκη, ΕΔΙΠ Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 13η Μαρτίου 2020.

.....	.....	.....
Ι. Σ. Βενιέρης	Δ.-Θ. Κακλαμάνη	Γ. Ματσόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2020

.....
Επαμεινώνδας Σ. Αλεβίζος
Δήμητρα Γ. Αλεξανδρινάκη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Οι διαδικτυακές αγορές έχουν αναπτυχθεί ταχύτατα τις τελευταίες δεκαετίες με αποτέλεσμα πολλές εταιρείες ηλεκτρονικού εμπορίου να έχουν δημιουργηθεί για να καλύψουν την αυξανόμενη ζήτηση. Στο πλαίσιο αυτό το Amazon είναι ένας από τους κολλοσούς του ηλεκτρονικού εμπορίου και χρησιμοποιείται καθημερινά από την πλειοψηφία των ανθρώπων για τις αγορές τους. Η εταιρεία αυτή προσφέρει πληθώρα ελεύθερων δεδομένων η ανάλυση των οποίων αξιοποιείται σε πολλούς τομείς, όπως η ενημέρωση των καταναλωτών αλλά και των πάροχων προϊόντων.

Στην παρούσα διπλωματική ασχολούμαστε με τα σχόλια χρηστών που παρέχει το Amazon για κάθε προϊόν με σκοπό τον εντοπισμό ενός πιθανού ελλατώματός του. Σε αρχικό στάδιο χρησιμοποιώντας έναν αλγόριθμο επιβλεπόμενης μηχανικής μάθησης, συγκεκριμένα έναν ταξινομητή, γίνεται η ανάλυση αισθήματος για κάθε σχόλιο των προϊόντων που επιλέχθησαν , με σκοπό τον καθορισμό της πολικότητάς τους. Με στόχο τη βελτίωση της απόδοσης του επιλεγμένου αλγορίθμου χρησιμοποιείται και μία μέθοδος ημι-επιβλεπόμενης μάθησης που ονομάζεται self-training. Στη συνέχεια με τη βοήθεια στατιστικών μεθόδων έγινε η εύρεση των σημαντικότερων λέξεων ,λέξεων-κλειδιών, για το εκάστοτε προϊον ,οι οποίες αναδεικνύουν το ελλάτωμά του και συνεπώς το που χρήζει βελτίωσης.

Τα προϊόντα που επιλέχθησαν από το Amazon για τις ανάγκες της παρούσας διπλωματικής ανήκουν στην κατηγορία electronics.
Τέλος για να καλυφθούν οι υπολογιστικοί πόροοι που απαιτεί η αποθήκευση ενός τέτοιου μεγάλου όγκου δεδομένων χρησιμοποιήσαμε την elasticsearch ως βάση αποθήκευσης και επεξεργασίας δεδομένων.

Λέξεις κλειδιά:

Ηλεκτρονικό εμπόριο, Μηχανική Μάθηση, Σχόλια, Πολικότητα, Λέξεις-κλειδιά, Amazon, Εξόρυξη Δεδομένων, Ταξινόμηση, Ανάλυση Αισθήματος

Abstract

Online shopping has grown rapidly the last decades and as a result many e-commerce companies have been created to meet the growing demand. In this context Amazon is one of one of the e-commerce giant and is used daily by the majority of people for purchase purposes. This company offers a wealth of free data and their analysis can be very informative for consumers and product providers.

In this thesis we deal with user feedback provided by Amazon for each product to identify a potential flaw. Initially, using a supervised machine learning algorithm, namely a classifier, the sentiment analysis is performed for each comment of the selected products, aiming to determine their polarity. In order to improve the performance of the selected algorithm, a semi-supervised learning method called self-training is used. Then, with the help of statistical methods, the most important keywords for each product were identified, which highlight their flaw and thus need improvement.

The products selected by Amazon for the purposes of this thesis belong to the electronics category.
Finally, to obtain the computational resources required to store such a large amount of data, we used elasticsearch as a database for data storage and processing.

Keywords:

E-commerce, Machine Learning, Reviews, Polarity, Keywords, Amazon, Data Mining, Classification, Sentiment Analysis

Ευχαριστίες

Αρχικά θα θέλαμε να ευχαριστήσουμε θερμά τον καθηγητή του Ε.Μ.Π. κύριο Ιάκωβο Βενιέρη ,για την ευκαιρία που μας δώθηκε να ασχοληθούμε με το αντικείμενο της παρούσας εργασίας το οποίο πέρα από το μεγάλο ενδιαφέρον που παρουσιάζει βρίσκεται και στην αιχμή των τεχνολογικών εξελίξεων, καθώς και την κυρία Σοφία Καπελλάκη μέλος ΕΔΙΠ του εργαστηρίου Ευφυών Επικοινωνιών και Δικτύων Ευρείας Ζώνης για την καθοριστική συμβολή της στην επιτυχή διεκπεραίωση αυτού του δύσκολου έργου.

Ακόμα θα θέλαμε να ευχαριστήσουμε θερμά τις οικογένειές μας για την αμέριστη στήριξη και υπομονή όλα αυτά τα χρόνια.

Τέλος ευχαριστούμε τους φίλους μας εκτός και εντός σχολής για τα όμορφα αυτά φοιτητικά χρόνια.

Πίνακας περιεχομένων

Κεφάλαιο 1 Εισαγωγή

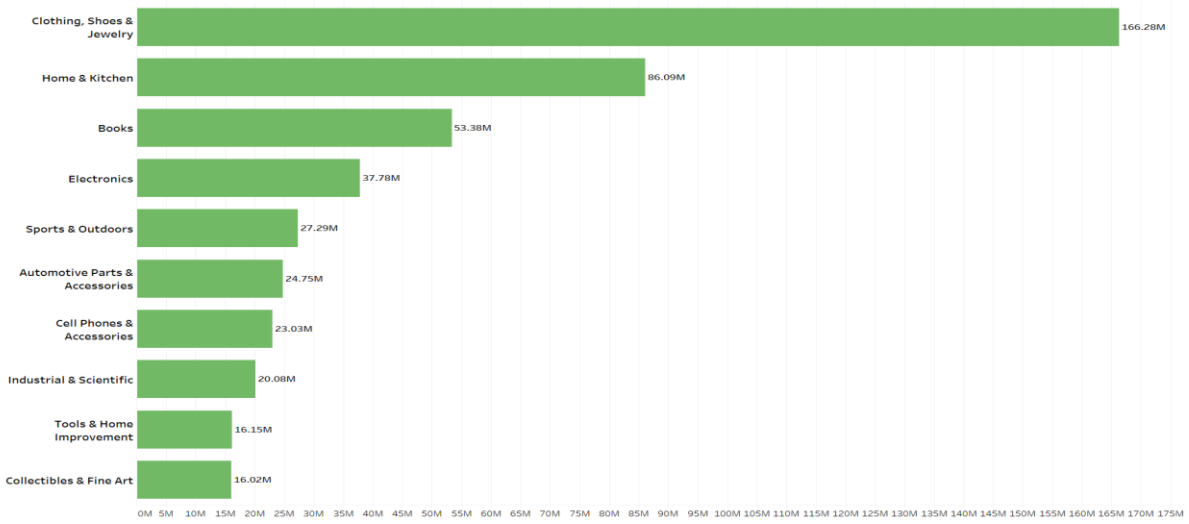
1.1 Γενική περιγραφή

Στη σημερινή εποχή ο παραδοσιακός τρόπος αγοράς προϊόντων διαφόρων ειδών τείνει να αντικατασταθεί από τις αγορές μέσω διαδικτύου ή αλλιως online shopping. Οι αγορές μέσω διαδικτύου δίνουν την δυνατότητα στους καταναλωτές να βρίσκουν πληθώρα προϊόντων και υπηρεσιών που τους ενδιαφέρουν, να συγκρίνουν τιμές σε διάφορους προμηθευτές και να πραγματοποιούν τις αγορές τους εύκολα από την άνεση του χώρου τους χρησιμοποιώντας απλά τον υπολογιστή ή το έξυπνο κινητό τους(smartphone). Αυτοί είναι μερικοί από τους λόγους που το online shopping κατακτά συνεχώς όλο και μεγαλύτερο μερίδιο της παγκόσμιας αγοράς του εμπορίου και οι ιστοσελίδες που παρέχουν υπηρεσίες διαδικτυακού εμπορίου αυξάνουν συνεχώς τόσο τη δημοτικότητα τους όσο και τα έσοδα τους. Χαρακτηριστικό της ραγδαίας αύξησης του ηλεκτρονικού εμπορίου είναι ότι μέχρι το έτος 2040 προβλέπεται πως οι καταναλωτές θα πραγματοποίουν το 95% των αγορών τους μέσω διαδικτύου. [39]

Βέβαια, παρόλα τα πολλά πλεονεκτήματα που προσφέρουν οι ηλεκτρονικές αγορές λόγω της ευκολίας τους αλλα και των πολλών διαθέσιμων επιλογών, υπάρχουν και κάποιοι κίνδυνοι που συνοδεύουν αυτές τις αγορές. Μία κατηγορία τέτοιων κινδύνων είναι αυτοι που αφορούν την ποιότητα και την αξιοπιστία των προιόντων που παραγγέλνουν οι χρήστες μέσω του διαδικτύου. Ένας από τους πιο αποτελεσματικούς τρόπους για να μπορέσει ένας εν δυνάμει αγοραστής να βεβαιώθει για την ποιότητα του προιόντος που σκοπεύει να αγοράσει αλλά και ότι αυτό ικανοποιεί τις δικές του προσωπικές ανάγκες είναι να διαβάσει τις κριτικές, σχόλια και παρατηρήσεις άλλων χρηστών που είτε σκοπεύουν να αγοράσουν το συγκεκριμένο προϊόν είτε το έχουν ήδη στην κατοχή τους. Αυτά τα σχόλια ή αλλιώς reviews βρίσκονται στη σελίδα του κάθε προϊόντος, είναι προσβασιμα από όλους και παρέχουν διαφόρων ειδών χρήσιμες πληροφορίες εκτός από την κριτική αυτού που τα γράφει, όπως πχ βαθμολογία με αστέρια (rating).

Η μεγαλύτερη ιστοσελίδα online shopping αυτήν την στιγμή παγκοσμίως είναι η Amazon [36] , στην οποία ένας καταναλωτής μπορεί να διαλέξει από ένα τεράστιο εύρος προϊόντων, από βιβλία και μουσική μέχρι κινητά τηλέφωνα, λάπτοπ αλλά και ρούχα. Το Amazon.com, όπως μετρήθηκε τον Απρίλιο του 2019, έχει διαθέσιμα για πώληση 119.928.851 προϊόντα [27], τα οποία διαθέτουν από μερικές δεκάδες εώς αρκετές χιλιάδες reviews το καθένα. Όπως φαίνεται λοιπόν και από τον τεράστιο όγκο τους, οι κριτικές αποτελούν

μία από τις χρησιμότερες πηγές πληροφοριών για τα προϊόντα και, αν μπορούσαν να αναλυθούν μαζικά, θα προσέφεραν πλήθος χρήσιμων δεδομένων τόσο για τους καταναλωτές όσο και για τους προμηθευτές των προϊόντων.



1.2 Αντικείμενο της Διπλωματικης

Το αντικείμενο αυτής της διπλωματικής εργασίας είναι η μελέτη των προκλήσεων και η σχεδίαση και ανάπτυξη μίας εφαρμογής, η οποία στοχεύει στην ανάλυση reviews προϊόντων και τελικά στην πρόβλεψη συγκεκριμένων χαρακτηριστικών από αυτά.

Αρχικά, για τον σκοπό αυτό, συλλέχθηλε μεγάλος όγκος reviews από προϊόντα που διατίθενται προς πώληση στο Amazon με σκοπό να δημιουργηθεί ένα σύνολο δεδομένων (dataset) που θα περιέχει χρήσιμες πληροφορίες για τα διάφορα προϊόντα. Στη συνέχεια, χρησιμοποιήθηκε μέρος αυτών, για να εκπαιδευτούν διάφοροι αλγόριθμοι μηχανικής μάθησης (machine learning algorithms) και μετά τη σύγκριση των αποτελεσμάτων των προβλέψεων τους, σε ότι αφορά την πολικότητα (polarity, αρνητικά-θετικά) των κριτικών με διάφορα κριτήρια (metrics), επιλέχθηκε ο αποδοτικότερος για την ανάπτυξη της εφαρμογής.

Έπειτα, ο συγκεκριμένος αλγόριθμος χρησιμοποιήθηκε για την πρόβλεψη αρχικά της πολικότητας (polarity) και στη συνέχεια για την εξαγωγή των λέξεων-κλειδιών (keywords) σε ένα καινούριο dataset από reviews. Έτσι τελικά, δοθέντος ενός συνόλου από reviews για ένα προϊόν να είναι ικανό το σύστημα να προβλέψει την πολικότητα των κριτικών αλλα και να διακρίνει τις σημαντικότερές τους λέξεις. Αυτός ο συνδυασμός πληροφοριών θα μπορούσε να αποδειχθεί πολύ χρησιμος, τόσο για τους καταναλωτές, για την τελική επιλογή προϊόντος, αλλά και για τους κατασκευαστές, ώστε να μπορούν να

έχουν καλύτερη εικόνα των πλεονεκτημάτων αλλά και των μειονεκτημάτων των προϊόντων τους, προκειμένου να τα βελτιώσουν.

1.3 Οργάνωση κειμένου

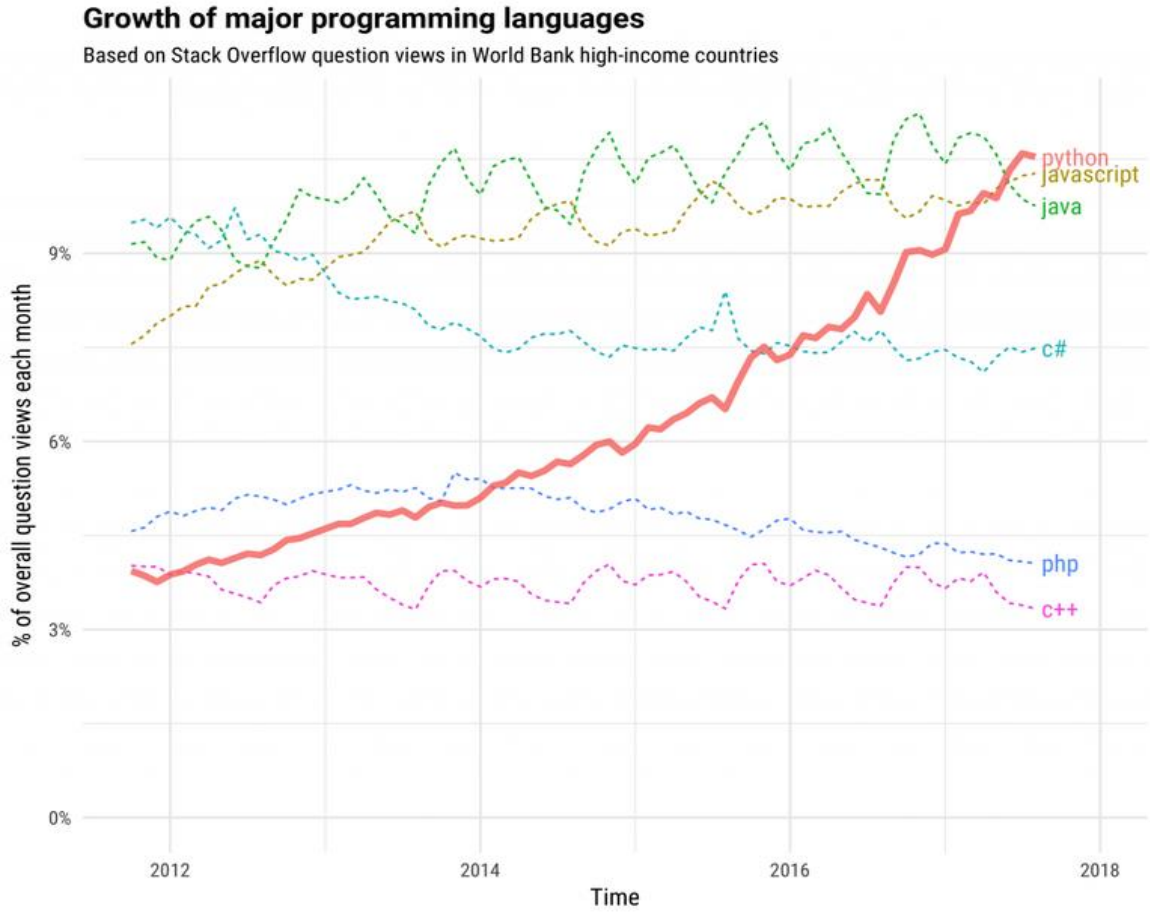
Στο κεφάλαιο 2 παρουσιάζονται όλες τις τεχνολογίες που μελετήθηκαν και χρησιμοποιήθηκαν στο πλαίσιο αυτής της εργασίας. Στο κεφάλαιο 3 αναφέρεται και αναλύεται η θεωρία πίσω από έννοιες όπως μηχανική μάθηση, αλγόριθμοι μηχανικής μάθησης, κατηγοριοποίηση κειμένου (text classification) και ανάλυση αισθημάτων (sentiment analysis). Στη συνέχεια, στο τέταρτο κεφάλαιο παρουσιάζεται η ανάλυση των απαιτήσεων, των λειτουργιών και των σεναρίων χρήσης της εφαρμογής μας, ενώ στο πέμπτο κεφάλαιο περιγράφονται αναλυτικά τμήματα του κώδικα καθώς και το σενάριο χρήσης της εφαρμογής. Τέλος στο έκτο κεφάλαιο παρουσιάζονται αποτελέσματα και συμπεράσματά, καθώς και προτάσεις για πιθανές επεκτάσεις.

Κεφάλαιο 2 Τεχνολογίες

2.1 Η γλώσσα Python

Η python [32] είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού, σχεδιασμένη για να χρησιμοποιηθεί σε ένα τεράστιο φάσμα εφαρμογών λογισμικού. Δημιουργήθηκε από τον Guido van Rossum και κυκλοφόρησε πρώτη φόρα το 1991 με την κεντρική φιλοσοφία της να είναι η αξιοπιστία και η ευκολία στην ανάπτυξη του κώδικα. Είναι μια αντικειμενοστρεφής γλώσσα η οποία βέβαια υποστηρίζει την κλήση μεθόδων (functions) και διαδικασιών (procedures) και χαρακτηρίζεται από τις standard βιβλιοθήκες της και τις δυνατότητες που αυτές προσφέρουν.

Ένα πολύ σημαντικό στοιχείο σχετικά με την Python είναι πως είναι η γλώσσα προγραμματισμού με την μεγαλύτερη ανάπτυξη τα τελευταία χρόνια και μάλιστα προβλέπεται πως θα προσπεράσει σε ενεργούς χρήστες άλλες πολύ σημαντικές και δημοφιλείς γλώσσες. Ο ρυθμός αυτής της ανάπτυξης φαίνεται χαρακτηριστικά στο παρακάτω διάγραμμα που δημοσίευσε το Q&A hub Stack Overflow:



Οι λόγοι αυτής της έκρηξης στη δημοτικότητα αλλα και τη χρήση της Python είναι αρχικά η ευκολία στην ανάπτυξη κώδικα, η πληθώρα βιβλιοθηκών και frameworks που προσφέρει, αλλά και το μεγάλο πλήθος built in βιβλιοθηκών και modules που παρέχει για τομείς όπως artificial intelligence, machine learning, big data και web development και η υποστήριξη από την ευρύτερη κοινότητα της στον προγραμματιστή για την ανάπτυξη του κώδικά του.

Οι προεγκατεστημένες (built-in) βιβλιοθήκες της Python, που χρησιμοποιήθηκαν στην παρούσα διπλωματική εργασία λόγω των πλενεκτημάτων και δυνατοτήτων που παρέχουν στον τομέα της ανάλυσης δεδομένων και της μηχανικής μάθησης (machine learning), είναι οι παρακάτω:

- NumPy

Η NumPy [9][53] είναι μια βιβλιοθήκη της Python που χρησιμοποιείται για την διαχείριση μεγάλων μαθηματικών πινάκων με μία ή περισσότερες διαστάσεις, προσφέροντας πλήθος μαθηματικών συναρτήσεων που μπορούν να εφαρμοστούν σε αυτούς τους πίνακες.
Είναι ένα από το σημαντικότερα και βασικότερα πακέτα που συνοδεύουν την Python και προσφέρει:

- Ένα αντικείμενο τύπου πίνακα N-διαστάσεων με πολλές σημαντικές δυνατότητες
- Αναβαθμισμένες και πολύπλοκες μαθηματικές συναρτήσεις και διαδικασίες που μπορούν, για παράδειγμα, να ενώσουν δυο πίνακες διαφορετικών διαστάσεων, να κάνουν διαφόρων ειδών Join στους πίνακες, να υπολογίσουν στατιστικά metrics για τα δεδομένα του πίνακα (μέση τίμη,επικρατούσα τιμή,ποσοστά κ.α), κρατήσουν μονο ένα συγκεκριμένο κομμάτι του πίνακα που μας ενδιαφέρουν οι τιμές του (slicing) και πολλά ακόμα.
- Συναρτήσεις που πραγματοποιούν πράξεις γραμμικής άλγεβρας, μετασχηματισμού Fourier και παραγωγής τυχαίων αριθμών
- Εργαλεία για διασύνδεση και αλληλεπίδραση με γλώσσες όπως η C, C++ και η Fortran

- Pandas

Το Pandas [9] αποτελεί ένα πακέτο που χρησιμεύει στη γρήγορη, ευέλικτη αλλα και ενισχυμένη με παρα πολλές δυνατότητες ανάλυση και επεξεργασία δεδομένων. Ένα από τα πλεονεκτήματα του είναι πως μπορεί να χρησιμοποιήθει για ένα μεγάλο εύρος δεδομένων που περιλαμβάνει: δεδομένα σε μορφή πίνακα με διαφορετικού είδους τιμές και πληροφορίες σε κάθε στήλη (όπως σε έναν πίνακα μιας SQL βάσης δεδομένων), ταξινομημένα και

μη δεδομένα σειριακού χρόνου (time series data) και αυθαίρετα δεδομένα σε πίνακα με ομοιόμορφες ή ανομοιόμορφες πληροφορίες στις στηλες και τις γραμμές του.
Οι κύριες δομές δεδομένων που χρησιμοποιούνται στο pandas είναι δύο:

- Series, που είναι μια μονοδιάστατη δομή με δεδομένα του ίδιου τύπου
- DataFrame, το οποίο είναι εν γένει ένα είδος πίνακα δυο διαστάσεων και περιλαμβάνει στις στήλες και τις γραμμές του δεδομένα είτε του ίδιου είτε διαφορετικού τύπου

Ορισμένες από τις βασικότερες λειτουργίες και δυνατότητες που προσφέρει η συγκεκριμένη βιβλιοθήκη είναι:

- Πολύ εύκολο χειρισμό του μεγέθους ενός DataFrame καθώς στήλες και γραμμές μπορούν να προστεθούν και να αφαιρεθούν με πολύ απλές εντολές
- "Έξυπνο" διαχωρισμό των δεδομένων με βάση τις επικεφαλίδες τους (labels) αλλά και την ακριβή τους θέση μέσα στον πίνακα δεδομένων (index)
- Πραγματοποιήση ενεργειών όπως concat, merge, join σε διαφορετικά datasets με διαφορετικά μεγέθη με πολλούς τρόπους αναλόγως με τις απαιτήσεις του προβλήματος
- Διαχείριση των "κενών" τιμών στα δεδομένα μας με πολύ απλό και χρήσιμο τρόπο για την επεξεργασία τους
- Δυνατότητα εισαγωγής δεδομένων από διαφορετικές πηγές όπως βάσεις δεδομένων και αρχεία διαφόρων μορφών πχ csv, json, txt.

- Scikit-learn

Η Scikit-learn [9], ή όπως αλλιώς είναι γνωστή sklearn, είναι μια βιβλιοθήκη φτιαγμένη για την γλώσσα Python που χρησιμοποιείται ειδικά σε machine learning αναπτύξεις και εφαρμογές. Περιλαμβάνει έναν μεγάλο αριθμό χαρακτηριστικών και αλγορίθμων, ειδικά για μηχανική μάθηση, και είναι δομημένη έτσι ώστε να μπορεί να αλληλεπιδρά με άλλες βιβλιοθήκες που πραγματοποιούν ανάλυση δεδομένων όπως η NumPy.
Η συγκεκριμένη βιβλιοθήκη είναι απαραίτητη και εξαιρετικά χρήσιμη για να αναπτυχθεί μια εφαρμογή μηχανικής μάθησης καθώς παρέχει ορισμένα από τα σημαντικότερα εργαλεία που χρησιμοποιούνται γενικά, και χρησιμοποιήθηκαν και σε αυτή την εργασία, όπως:

- Feature extraction module

Αυτή η βιβλιοθήκη χρησιμοποιείται σε δεδομένα όπως εικονές και κείμενο για την μετατροπή και εξαγωγή σε μορφή που μπορεί να δωθεί σαν είσοδος σε κάποιον αλγόριθμο μηχανικής μάθησης. Δυο τέτοιες βιβλιοθήκες που χρησιμοποιήθηκαν είναι οι Count Vectorizer και Tf-Idf vectorizer. Η πρώτη μετατρέπει ένα σύνολο προτάσεων κειμένου (corpus) σε ένα πίνακα με την συχνότητα εμφάνισης της κάθε λέξης στις αντίστοιχες προτάσεις, ενώ ο δεύτερος υπολογίζει μια ιδιαίτερη συχνότητα εμφάνισης των λέξεων που προκύπτει από το πόσες φορές εμφανίζονται αυτές στις προτάσεις αλλά και από το πλήθος των προτάσεων και σε πόσες από αυτές συναντάμε τη συγκεκριμένη λέξη. Εκτενέστερη ανάλυσή τους παρουσιάζεται στη συνέχεια.

- Grid-search

Όταν χρησιμοποιούνται οι εκτιμητές των διαφόρων αλγορίθμων δίνονται σε αυτούς κάποιες παράμετροι με προκαθορισμένες τιμές έτσι ώστε να μπορούν να λειτουργήσουν σωστά. Ωστόσο, υπάρχουν τιμές των συγκεκριμένων παραμέτρων, οι οποίες καλούνται υπερπαράμετροι[1], για τις οποίες οι αλγόριθμοι να πετυχαίνουν μεγαλύτερη ακρίβεια στις προβλέψεις τους. Για να προσδιοριστούν αυτές οι τιμές η Grid-search εκτελεί τον αλγόριθμο για όλους τους πιθανούς συνδυασμούς υπερπαραμέτρων που δίνονται και έτσι είναι δυνατή η επιλογή εκείνων που δίνουν τα βέλτιστα αποτελέσματα.

Περιλαμβάνει ακόμα τους σημαντικότερους αλγορίθμους μηχανικής μάθησης που χρησιμοποιούνται τόσο στο πλαίσιο αυτής της εργασίας όσο και σε προβλήματα του πραγματικού κόσμου.

- Support Vector Machines (SVM)
- Naïve Bayes
- Random Forest
- Logistic Regression
- Nearest neighbors

Αυτοί οι αλγόριθμοι εξηγούνται αναλυτικότερα στη συνέχεια του κεφαλαίου.

- Nltk platform

Natural Language Toolkit [41] ή αλλιώς NLTK ονομάζεται μια μεγάλη συλλογή από βιβλιοθήκες και προγράμματα που χρησιμοποιούνται για την επεξεργασία φυσικής γλώσσας (NLP-Natural Language Processing). Πιο

[1] Υπερπαράμετροι είναι εξωτερικοί παράμετροι του αλγορίθμου οι οποίοι ορίζονται είτε από τον προγραμματιστή είτε με χρήση ευρετικών μεθόδων και δεν μπορούν να καθοριστούν από τα δεδομένα εισόδου, δηλαδή μετά την εκπαίδευση του αλγορίθμου, όπως συμβαίνει με τις υπόλοιπες εσωτερικές παραμέτρους του.

συγκεκριμένα, η επεξεργασία φυσικής γλώσσας είναι ένας κλάδος της τεχνητής νοημοσύνης (Artificial Intelligence, AI) που επικεντρώνεται στην αλληλεπίδραση του ανθρώπου και του υπολογιστή με τη χρήση φυσικής γλώσσας. Στόχος λοιπόν είναι να μπορέσει μια μηχανή να καταλάβει και να αποκωδικοποιήσει μια πληροφορία που εκφράζεται σε μια φυσική ανθρώπινη γλώσσα. Οι περισσότερες τεχνικές που εφαρμόνται βασίζονται στη μηχανική μάθηση. Ορισμένες από τις πιο γνωστές περιπτώσεις που εφαρμόζεται η επεξεργασία φυσικής γλώσσας είναι οι εφαρμογές μετάφρασης, όπως το Google translate, οι εφαρμογές ελέγχου ορθογραφίας και γραμματικής, όπως αυτή του Microsoft Word και το Grammarly και φυσικά οι εφαρμογές προσωπικής βοήθειας όπως το OK Google και το Apple Siri.
Ορισμένα από τα χρησιμότερα πακέτα που απαιτούνται για την επεξεργασία κειμένου σε φυσική γλώσσα και που βρίσκουν εφαρμογή και στην παρούσα εργασία είναι :

- Stopwords [41]

Υπάρχει μια λίστα με λέξεις στην αγγλική γλώσσα που είναι κοινές, χρησιμοποιούνται σε όλες τις περιπτώσεις και μάλιστα με μεγάλη συχνότητα όμως δεν προσφέρουν κάποια σημαντική πληροφορία στο νόημα του κειμένου και έτσι δεν οφελούν με κάποιο τρόπο την επεξεργασία και την ανάλυση του νοήματος του. Τέτοιες λέξεις είναι το “the”, “a”, “is”, “and” και άλλες. Αυτές οι λέξεις ονομάζονται stopwords και μέσω του αντίστοιχου πακέτου που προσφέρει το Nltk μπορούμε να αφαιρέσουμε αυτού του είδους τις λέξεις από το κείμενο που επεξεργαζόμαστε, να κρατήσουμε τις σημαντικές και έτσι να μπορούμε να εξάγουμε πολύ πιο χρήσιμα αλλα και ορθότερα συμπεράσματα αναφορικά με το νοήμα του κειμένου.

- Lemmatization [41]

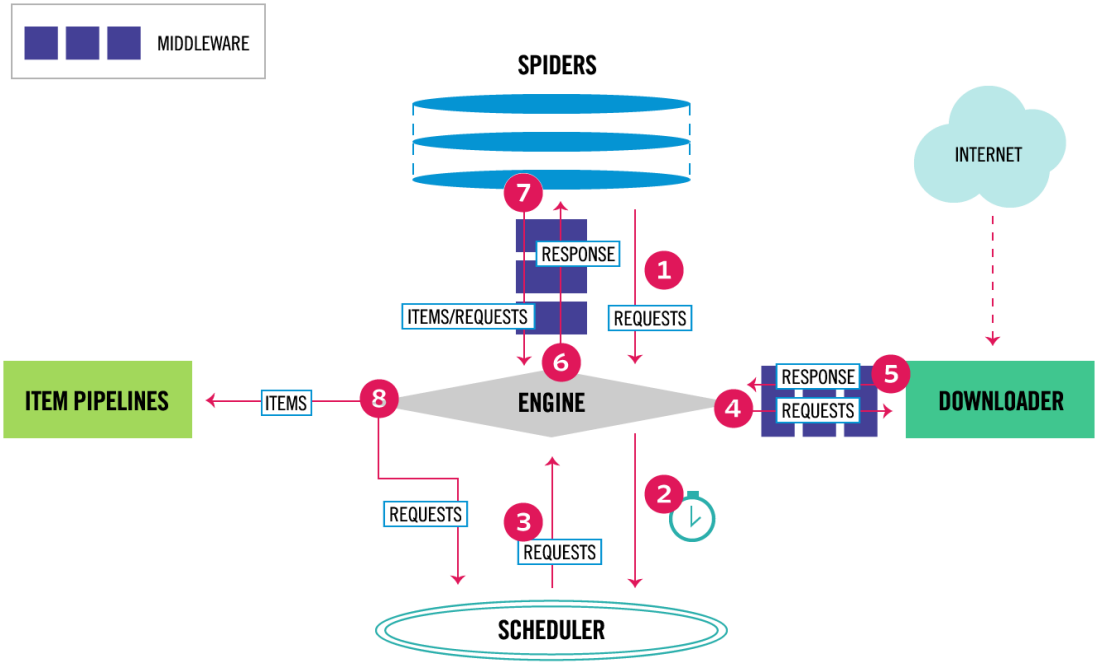
Είναι η διαδικασία ομαδοποίησης διαφορετικών λέξεων που όμως συνδέονται τόσο λεξιλογικά, καθώς περιλαμβάνουν το ίδιο θέμα, όσο και εννοιολογικά. Έτσι, αυτές τις λέξεις τις κατηγοριοποιούμε κάτω από μια ευρύτερη κατηγορία και τις αναλύουμε όλες σαν μία, λόγω του κοινού νοηματικού τους περιεχομένου και των πολύ μικρών διαφορών τους. Για παράδειγμα οι λέξεις run, running και runners θα κατηγοριοποιηθουν κάτω από τη λέξη run που είναι και η ρίζα από την οποία προέρχονται αλλά και συνοψίζει το νόημα τους. Αυτή η λέξη ονομάζεται λήμμα (lemma). Με αυτόν τον τρόπο μπορούμε αφενός να μειώσουμε το πλήθος των λέξεων που θα επεξεργαστούμε και αφετέρουμε να κρατήσουμε την ουσία που θα μας βοηθήσει να εξάγουμε κάποιο νόημα από το κείμενο.

- Stemming [41]

Αυτή η διαδικασία μοιάζει πολύ με την παραπάνω, καθώς πρόκειται και εδώ για μια διαδικασία που ομαδοποιεί διαφορετικές λέξεις σε μια κατηγορία. Η διαφορά τους είναι πως, μέσω του stemming, αφαιρείται από τις λέξεις η κατάληξη ή το πρόθεμα και παραμένει μόνο η ρίζα, χωρίς αυτή να είναι απαραίτητα μια ανεξάρτητη λέξη με κάποιο νόημα. Για παράδειγμα από τις λέξεις trouble, troubled, troubling η κατηγοριοποίηση θα μας δώσει σαν αποτέλεσμα τη ρίζα troubl.

2.2 Scrapy Framework

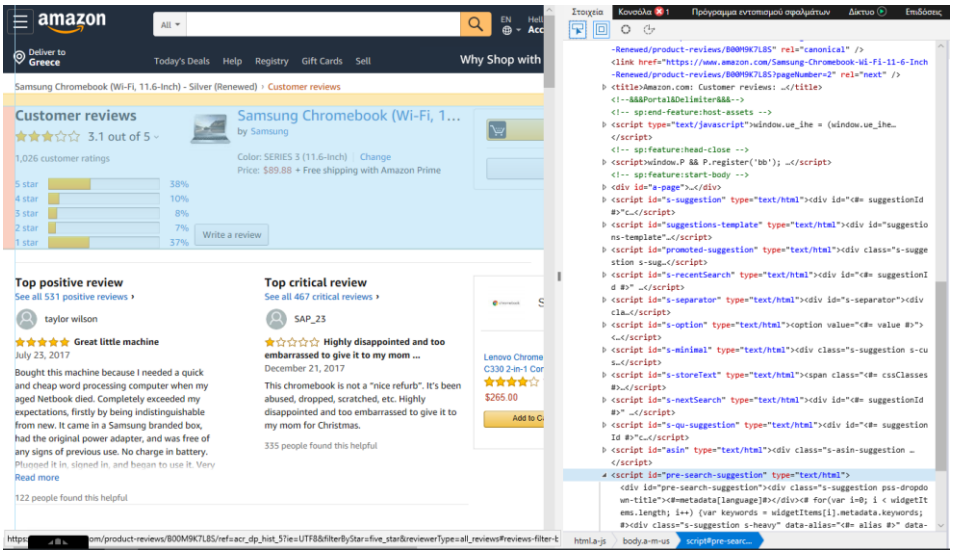
Το scrapy [34] είναι ένα framework της python που χρησιμοποιείται για να προσπελάσει σελίδες στο διαδίκτυο και να εξάγει από αυτές χρήσιμες πληροφορίες. Το σύνολο αυτών των ενεργειών καλείται web scraping και τελευταία έχει αναδειχθεί σε έναν από τους πιο διαδεμένους και αποτελεσματικούς τρόπους συλλογής δεδομένων που προορίζονται για ένα μεγάλο εύρος εφαρμογών.
Πιο συγκεκριμένα, το scrapy, μέσα από τα αυτόνομα κομμάτια κώδικα του, που ονομάζονται spiders, αποτελεί έναν πολύ γρήγορο και αποτελεσματικό τρόπο για να βρεθουν και να εξαχθούν πλήθος πληροφοριών, οι οποίες βρίσκονται στον html κώδικα των ιστοσελίδων στο διαδίκτυο. Αναλυτικότερα, το παρακάτω διάγραμμα δείχνει μια ευρύτερη εικόνα της αρχιτεκτονικής με την οποία λειτουργεί το scrapy, αλλά και των στοιχείων που λαμβάνουν μέρος στη διαδικασία εξόρυξης των δεδομένων.



Η διαδικασία που απεικονίζεται παραπάνω έχει ως εξής: αρχικά τα αυτόνομα τμήματα κώδικα που αναφέραμε (spiders) στέλνουν αιτήματα (requests) στη μηχανή του συστήματος (engine) η οποία, αφού ρυθμίσει την σειρά εκτέλεσης των αιτημάτων στον scheduler, τα προωθεί στον downloader. Στη συνέχεια, ο downloader αφού "κατεβάσει" τη ζητούμενη σελίδα από το internet, δημιουργεί μια απάντηση (response) προς την engine στέλνοντας αυτήν την σελίδα. Με την σειρά της η μηχανή, μεταβιβάζει την απάντηση στο spider για επεξεργασία. Έπειτα από αυτήν την επεξεργασία, έχουμε πια εξάγει την επιθυμητή πληροφορία (scrapped items) και μπορούμε να την προωθήσουμε μέσω της μηχανής στα item pipelines και να προχωρήσουμε στην διεκπαιρέωση νέων αιτημάτων (requests), αν υπάρχουν.

2.3 Amazon

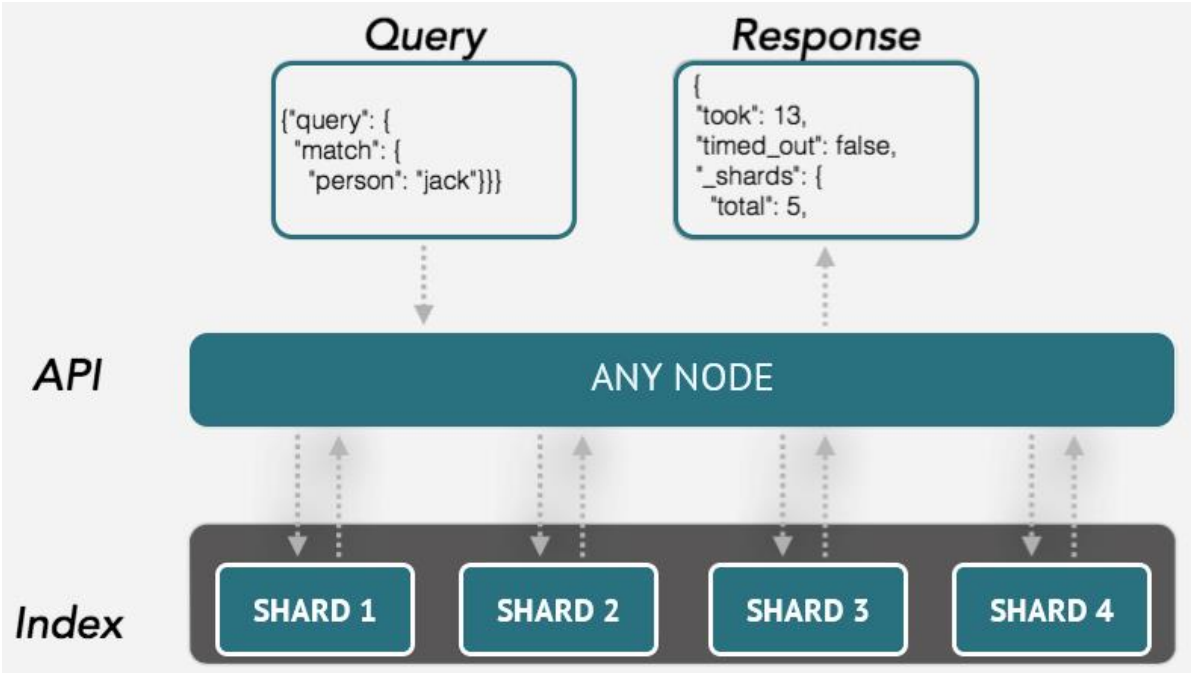
Η Amazon είναι η κορυφαία εταιρία ηλεκτρονικού εμπορίου παγκοσμίως προσφέροντας προς πώληση όπως έχει ήδη αναφερθεί εκατοντάδες εκατομμύρια προϊόντα. Η ιστοσελίδα της (https://www.amazon.com/) αποτελεί την κύρια πήγη του ηλεκτρονικού εμπορίου και περιέχει ένα πολύ ευρύ φάσμα χρήσιμων πληροφοριών από τιμές και πληροφορίες για το εκάστοτε προϊόν, μέχρι κριτικές του προϊόντος, ratings με αστέρια, αριθμό χρηστών που βοηθήθηκαν από την κάθε κριτική και πληροφορίες για τον κατασκευαστή. Έτσι λοιπόν, όλα αυτά που αναφέρθηκαν παραπάνω εφαρμόζονται στον html κώδικα της συγκεκριμένης σελίδας με στόχο ακολουθώντας την προαναφερθείσα διαδικασία να εξαχθούν όλες αυτές οι χρήσιμες πληροφορίες και να διαμορφώθεί το dataset που χρησιμοποιείται και αναλύεται στο πλαίσιο της διπλωματικής.



2.4 Elasticsearch

2.4.1 Περιγραφή

Η elasticsearch [30] είναι μια No-SQL βάση δεδομένων, η οποία προσφέρει περισσότερες δυνατότητες και επιλογές από τις κλασικές βάσεις, κυρίως στους τομείς της αναζήτησης και της ανάλυσης των δεδομένων που αποθηκεύονται σε αυτην. Αν και δεν υπάρχει ένας ιδιαίτερα σαφής και ξεκάθαρος ορισμός του τι είναι μια No-SQL βάση δεδομένων, κάποια από τα κύρια χαρακτηριστικά αυτού του είδους των βάσεων είναι πως δεν υποστηρίζουν τις σχέσεις ανάμεσα στα δεδομένα με την παραδοσιακή μορφή (διάφορα είδη join), είναι κατανεμημένες (distributed) και επεκτάσιμες.
Πιο συγκεκριμένα, η elasticsearch είναι μια βάση δεδομένων που βασίζεται στη δημιουργία documents (document-oriented) και χρησιμοποιώντας την μπορεί κανεις να αποθηκεύσει, να αναλύσει αλλα και να πραγματοποιήσει αναζητήσεις στα δεδομένα του, τα οποία βρίσκονται αποθηκευμένα σε μορφή JSON[2] (JavaScript Object Notation). Οι αναζητήσεις στα δεδομένα που έχουμε αποθηκεύσει ή αλλιώς queries μπορούν να γίνουν εύκολα σε JSON format μέσα από την elasticsearch και λειτουργούν όπως φαίνεται παρακάτω



[2]JSON είναι ένα ανοικτό μορφότυπο το οποίο χρησιμοποιεί κείμενο, που μπορεί να διαβαστεί από τον άνθρωπο, για τη μετάδοση αντικειμένων δεδομένων που αποτελούνται από ζεύγη χαρακτηριστικών-τιμών και τύπου δεδομένων συστοιχιών (ή οποιασδήποτε άλλης σειριοποίησιμης τιμής). Πρόκειται για ένα πολύ κοινό μορφότυπο δεδομένων που χρησιμοποιείται για την ασύγχρονη επικοινωνία περιήγηση-διακομιστή, συμπεριλαμβανομένης της αντικατάστασης για XML σε μερικά συστήματα τύπου AJAX.[42]

Για να γίνει καλύτερα κατανοήτη η δομή και η λειτουργία της συγκεκριμένης βάσης θα δουμε αναλυτικότερα ορισμένα από τα δομικά στοιχεία της [35]:

- **Node**

Node είναι ένας server που είναι μέρος ενός ευρύτερου cluster και σε αυτόν αποθηκεύουμε τα δεδομένα μας αλλά και τον χρησιμοποιούμε για τις αναζητήσεις. Ένας node ορίζεται από το όνομα του, το οποίο εξ ορισμού είναι ένα τυχαίο UUID (Universally Unique Identifier) και του δίνεται εξ αρχής.

- **Cluster**

Cluster είναι ένα σύνολο αποτελούμενο από έναν ή παραπάνω nodes, στους οποίους είναι αποθηκευμένα τα δεδομένα μας. Μπορούν να υπάρχουν N nodes που να αντιστοιχούν στο ίδιο όνομα cluster.

- **Index**

Index είναι ένα σύνολο από documents που έχουν κοινά χαρακτηριστικά. Για παράδειγμα, μπορούμε να δημιουργήσουμε ένα index για ένα συγκεκριμένο προϊόν ή για έναν ιδαίτερο κατασκευαστή και να αναφερόμαστε σε αυτό με το μοναδικό όνομα του όταν θέλουμε να πραγματοποιήσουμε αναζητήσεις, ενημερώσεις ή διαγραφές. Σε ένα cluster μπορούμε να έχουμε πολλαπλά indexes. Το Index είναι αντίστοιχο της βάσης δεδομένων σε RDBMS.

- **Document**

.
Με τον όρο document εννοούμε μια πληροφορία την οποία καταχωρούμε στο index μας. Για παράδειγμα, σε ένα index σχετικά με ένα πρϊόν μπορούμε να έχουμε ένα document για τον κατασκευαστή του. Τα documents καταχωρούνται σε JSON μορφή και αντιστοιχούν σε μια γραμμή ενός πίνακα σε σχεσιακές βάσεις δεδομένων. Σε ένα index δεν υπάρχει περιορισμός στον αριθμό των documents που μπορούμε να αποθηκεύσουμε.

- **Shard**

Η elasticsearch παρέχει την δυνατότητα να χωρίσουμε το index μας σε πολλές υποκατηγορίες που ονομάζονται shards. Κάθε μια από αυτές τις υποκατηγορίες είναι πλήρως αυτόνομη και λειτουργική και μπορεί να βρίσκεται σε οποιονδήποτε node του cluster.

Κάθε στοιχείο της elasticsearch μπορεί να γίνει προσβάσιμο μέσω REST API:

- Index API: χρησιμοποείται για να προσθέσει documents στο index
- Get API: χρησιμοποείται για να λάβουμε το document
- Search API: χρησιμοποείται για να υποβάλουμε requests και να λάβουμε responses

Επίσης, παρόλο που η Elasticsearch είναι κατά κύριο λόγο μια βάση δεδομένων με επιπλέον δυνατότητες αναζήτησης στα δεδομένα που αποθηκεύονται, οι αυξημένες απαιτήσεις των χρηστών για περαιτέρω ανάλυση των δεδομένων και χρήσιμες απεικονίσεις δημιούργησαν το Kibana και το Logstash που μαζί με την Elasticsearch αποτελούν το ELK.

2.4.2 Kibana

Με το kibana [35] ως εργαλείο μπορούμε να πραγματοποιήσουμε γραφήματα και απεικονίσεις για τα δεδομένα που αποθηκεύονται στην Elasticsearch. Έχουμε τη δυνατότητα να επιλέξουμε από μια πληθώρα γραφημάτων όπως histograms, line graphs, pie charts, sunbursts αλλα και να δημιουργήσουμε το δικό μας γράφημα. Ακόμα, μπορουμε να προχωρήσουμε σε περαιτέρω αναλύσεις και να βρουμε συσχετισμούς στα δεδομένα μας, χρησιμοποιώντας ορισμένα χαρακτηριστικά του, όπως advanced time series analysis και machine learning features.

Logstash

Το Logstash [35] είναι ένα ανοιχτό προς όλους pipeline που βρίσκεται στην πλευρά του server και μπορεί να δέχτει δεδομένα από διάφορες πηγές ταυτόχρονα, να τα επεξεργαστεί και να παράγει χρήσιμα αποτελέσματα και συμπεράσματα, όπως για παράδειγμα για το network traffic, τη χρήση της μνήμης, το uptime του συστήματος αλλά και πληροφορίες για κάθε node ξεχωριστά.

2.5 Microsoft Azure

Το Microsoft Azure [43] αποτελεί μια σειρά υπηρεσιών υπολογιστικού νέφους. Οι υπηρεσίες του καλύπτουν ένα ευρύ φάσμα αντικειμένων όπως είναι οι περιοχές των big data, της μηχανικής μάθησης, της ασφάλειας, των ευφυών δικτύων, του διαδικτύου των πραγμάτων (IoT) ,του blockchain, της αποθήκευσης και ανάλυσης δεδομένων καθώς και άλλων πολλών.
Παρέχεται από τη Microsoft και αποτελεί την αντίστοιχη υπηρεσία της εταιρίας των Amazon Web Services και Google Cloud Platform, που προσφέρονται από την Amazon και τη Google αντίστοιχα. Για τη χρήση της υπηρεσίας απαιτείται σύνδεση και εγγραφή με τραπεζικό λογαριασμό ή κάρτα. Για τις χρεώσεις των υπηρεσιών ακολουθείται η πολιτική «pay as you go» δηλαδή καταβάλλουμε ένα ποσό ανάλογα με το χρόνο που κρατάμε δεσμευμένους κάποιους πόρους αλλά και το βαθμό χρήσης που κάνουμε στους πόρους αυτούς.

2.5.1 Microsoft Azure Notebooks

Η υπηρεσία που χρησιμοποιήθηκε κυριώς είναι τα Microsoft Azure Notebooks.Ένα Jupyter Notebook μας δίνει τη δυνατότητα να δημιουργούμε και να επεξεργαζόμαστε αρχεία που περιέχουν κώδικα, γραφήματα και λοιπές απεικονίσεις δεδομένων, εξισώσεις και επεξηγηματικά σχόλια.
Στο Jupyter υποστηρίζονται πολλές γλώσσες προγραμματισμού όπως οι Python 2 and 3, F#, and R. Επίσης, εκτός από το να χρησιμοποιήσει κανείς την διεπαφή του Jupyter notebook, δίνεται η δυνατότητα στον χρήστη να έχει πρόσβαση σε τερματικό και να αλληλεπιδρά άμεσα με τον ενεργό server.
Ορισμένες από τις πιο διαδεδομένες χρήσεις είναι ο καθαρισμός και η μετατροπή δεδομένων, η δημιουργία στατιστικών μοντέλων και η ανάπτυξη machine learning και big data εφαρμογών.

Κεφάλαιο 3 Θεωρητικό Υπόβαθρο

3.1 Μηχανική Μάθηση

Machine Learning (ML) [24] είναι ο επιστημονικός κλάδος που ασχολείται με τη μελέτη αλγορίθμων και στατιστικών μοντέλων που χρησιμοποιούνται από υπολογιστές και υπολογιστικά συστήματα για να λύσουν ένα συγκεκριμένο πρόβλημα χωρίς όμως να ακολουθούν συγκεκριμένες οδηγίες και αυστηρούς κανονές, αλλά βασιζόμενα στη δημιουργία μοτίβων και μοντέλων και την εξαγωγή συμπερασμάτων από αυτά. Συγκεκριμένα, οι αλγόριθμοι που χρησιμοποιούνται στις εφαρμογές μηχανικής μάθησης (machine learning) δημιουργούν, εξάγουν ένα μαθηματικό μοντέλο χρησιμοποιώντας ένα σύνολο δεδομένων πάνω στο οποίο εκπαιδεύονται (training data), έτσι ώστε να μπορούν να προβλέψουν ή να βγάλουν συμπεράσματα για ένα νέο, άγνωστο πλήθος δεδομένων.
Το σημαντικότερο χαρακτηριστικό των machine learning αλγορίθμων, το οποίο μάλιστα αποτελεί και τη βασικότερη διαφορά τους σε σχέση με τον κλασικό προγραμματισμό, είναι ότι μέσω της μηχανικής μάθησης, οι υπολογιστές έχουν την δυνατότητα να "μαθαίνουν" από τις εμπειρίες τους και να βελτιώνονται, χωρίς ουσιαστικά να προγραμματίζονται από κάποιον. Ένα παραδοσιακό πρόγραμμα θα λειτουργούσε ως εξής: θα αναπτύσαμε την λογική του προγράμματος σε κώδικα σε κάποια γλώσσα προγραμματισμού, θα τροφοδοτούσαμε τον κώδικα με δεδομένα ως είσοδο και θα παίρναμε μετά την εκτέλεσή του τα αποτελέσματα. Αντίθετα, σε έναν αλγόριθμο μηχανικής μάθησης του δίνουμε για εκπαίδευση τόσο τα αρχικά δεδομένα όσο και τα αποτελέσματα τους με στόχο να δημιουργήσει αυτός ένα προγραμμα – μοντέλο το οποίο θα μπορούμε μετά να χρησιμοποιήσουμε για να κάνουμε προβλέψεις ή κατηγοριοποιήσεις και να αξιολογήσουμε την ορθότητα τους.

Λόγω των συγκεκριμένων χαρακτηριστικών της είναι σύνηθες η μηχανική μάθηση να μπερδεύεται με την τεχνητή νοημοσύνη (Artificial Intelligence-AI) και μάλιστα αυτοί οι δυο όροι να ταυτίζονται και να χρησιμοποιούνται ο ένας στη θέση του άλλου.
Ωστόσο, κάτι τέτοιο δεν είναι σωστό, καθώς αυτοί οι δύο τομείς έχουν σημαντικές διαφορές. Η τεχνητή νοημοσύνη στοχεύει στο να καλλιεργήσει σε μία μηχανή, όπως είναι ένας υπολογιστής, ευφυΐα έτσι ώστε να μπορεί να προσομοιώσει την ανθρώπινη συμπεριφορά για να αντιμετωπίσει ένα σύνθετο προβλημα. Έτσι λοιπόν, σκοπός μιας εφαρμογής τεχνητής νοημοσύνης είναι να μπορεί μια μηχανή να "φερθεί έξυπνα", μιμούμενη τον άνθρωπο και να έχει τη δυνατότητα να πάρει αποφάσεις δυναμικά ανάλογα με τις συνθήκες του εκάστοτε προβλήματος. Από την άλλη, η μηχανική μάθηση στηρίζεται στην απλή εκπαίδευση μέσω δεδομένων ενός αλγορίθμου με συγκεκριμένα χαρακτηριστικά. Στόχος της λοιπόν είναι, για τη λύση ενός ειδικού προβλήματος ο αλγόριθμος να μάθει από τα δεδομένα που του δίνονται σαν είσοδος και να παράξει όσο το δυνατόν πιο ακριβή αποτελέσματα ή προβλέψεις μόνο για αυτό το συγκεκριμένο πρόβλημα. Για την ακρίβεια, η μηχανική μάθηση είναι ένας υποκλάδος του ευρύτερου επιστημονικού τομέα που ασχολείται με την τεχνητή νοημοσύνη.

Στη σημερινή εποχή λοιπόν, η μηχανική μάθηση χρησιμοποιείται ολοένα και περισσότερο και μάλιστα σε ένα τεράστιο φάσμα εφαρμογών που αφορούν διαφορετικές πτυχές της ανθρώπινης ζωής και καθημερινότητας. Ορισμένα παραδείγματα της χρήσης ML αλγορίθμων είναι:

- οι μηχανές αναζήτησης στο διαδίκτυο, όπου τα αποτελέσματα που θα μας επιστρέψουν αναλόγως την αναζήτηση μας και η σειρά με την οποία θα μας τα παρουσιάσουν σελιδές όπως η google και η bing είναι αποτελέσματα ενός πολύπλοκου αλογορίθμου μηχανικής μάθησης
- τα μέσα κοινωνικής δικτύωσης, που χρησιμοποιούν διάφορους αλγορίθμους που στηρίζονται στη μηχανική μάθηση για να παρέχουν κάποια από τα πιο δημοφιλή χαρακτηριστικά τους όπως η λίστα προτεινόμενων φίλων και η αναγνώριση προσώπων στο facebook. Στην πρώτη περίπτωση για παράδειγμα, οι φίλοι κάποιου στο συγκεκριμένο μέσο κοινωνικής δικτύωσης, οι σέλιδες που του αρέσουν, οι ομάδες στις οποίες είναι μέλος και διάφορα άλλα επεξεργάζονται από έναν αλγόριθμο ο οποίος έχει σαν έξοδο τα προφίλ των ατόμων που ταιριάζουν καλύτερα στον χρήστη.
- οι διάφοροι mail clients για να φιλτράρουν την ανεπιθύμητη αλληλογραφία (spam). Οι κλασικοί κανόνες που εφαρμόζονται αρχικά, δεν είναι αποτελεσματικοί απέναντι σε πιο εξελιγμένες τεχνικές και έτσι το κατάλληλο φιλτράρισμα επιτυγχάνεται μέσω machine learning αλγορίθμων.

Για τον σχεδιασμό και την ανάπτυξη μια τέτοιας machine learning εφαρμογής πρέπει να γίνουν οι εξής ενέργειες:

- Συλλογή δεδομένων που παρέχουν τις κατάλληλες πληροφορίες για το πρόβλημα το οποίο θέλουμε να αντιμετωπίσουμε και είναι σε κατάλληλη ώστε να μπορούμε να τα αποθηκεύσουμε
- Προεπεξεργασία (pre-processing) των δεδομένων ώστε να τα μετατρέψουμε σε αξιοποιήσιμη μορφή για την εφαρμογή μας. Τις περισσότερες φορές τα δεδομένα που έχουμε για να εκπαιδεύσουμε έναν αλγόριθμο είναι σε ακατέργαστη μορφή (raw data), δηλαδή μπορεί να περιέχουν κένες τιμές ή οι τιμές που έχουν να είναι αποθηκευμένες σε διαφορετική μορφή από την επιθυμητή ως είσοδο για τον αλγοριθμό μας (πχ string ή int64). Για παράδειγμα, αν τα δεδομένα μας περιλαμβάνουν κείμενο ή εικόνες θα πρέπει να μετατραπούν σε αριθμητική μορφή, ίσως σε μια λίστα ή σε έναν πίνακα, με τέτοιο τρόπο όμως, ώστε να διατηρήσουν τη συνοχή και τις πληροφορίες τους.
- Διαχωρισμός των δεδομένων σε αυτά που θα χρησιμοποιηθούν για την εκπαίδευση (train set) και σε αυτά τα οποία θα χρησιμοποιηθούν για την πρόβλεψη, την κατηγοριοποίηση ή όποια άλλη ενέργεια απαιτείται (test set).
- Δημιουργία των μοντέλων μέσω της εκπαίδευσης των κατάλληλων αλγορίθμων στο train set.
- Χρήση των παραγόμενων εκπαιδευμένων μοντέλων στα δεδομένα που δεν χρησιμοποιήθηκαν για εκπαίδευση, δηλαδή στο test set και αξιολόγηση των αποτελεσμάτων με διάφορα κριτήρια όπως F1 score, accuracy, precision, recall κτλ

Διάφορα είδη τεχνικών learning [7]:

- Supervised Learning
- Unsupervised
- Semi-supervised learning
- Reinforcement

3.2 Τεχνικές Μηχανικής Μάθησης

Η μηχανική μάθηση είναι ένα υποπεδίο της επιστήμης υπολογιστών που ασχολείται με την κατασκευή μοντέλων ή προτύπων από ένα σύνολο δεδομένων ενός υπολογιστικού συστήματος. Έχουν αναπτυχθεί αρκετές τεχνικές μηχανικής μάθησης που χρησιμοποιούνται ανάλογα με τη φύση του εκάστοτε προβλήματος. Ορισμένες από αυτές τις τεχνικές αναφέρονται στη συνέχεια.

3.2.1 Επιβλεπόμενη Μάθηση

Η επιβλεπόμενη μάθηση (Supervised Learning) [46] αποτελεί κατηγορία μηχανικής μάθησης,στόχος της οποίας είναι ο χαρακτηρισμός/κατηγοριοποίηση δεδομένων εισόδου -με βάση χαρακτηρισμένα δειγματα εκπαίδευσης (labeled data) [ref]. Τα συγκεκριμένα δεδομένα χρησιμοποιούνται ως διάνυσμα εισόδου για την εκπαίδευση των αλγορίθμων της συγκεκριμένης κατηγορίας και το αποτέλεσμα είναι μια τιμή ή κατηγορία εξόδου που προκύπτει από τη 'λογική' εξήγηση της εισόδου. Ένα βέλτιστο σενάριο της παραπάνω διαδικασίας είναι η σωστή κατηγοριοποίηση άγνωστων έως τώρα δεδομένων.

Ένα απλό παράδειγμα είναι η πρόβλεψη της χρονικής διάρκειας μεταφοράς ενός ατόμου από το χώρο εργασίας του στο σπίτι. Σε αυτήν την περίπτωση ο αλγόριθμος θα λάμβανε υπ’ όψιν συγκεκριμένα χαρακτηριστικά, όπως συνθήκες καιρού, ώρα, γιορτές, αργίες και λοιπά, τα οποία συνθέτουν μία συνάρτηση εισόδου-εξόδου. Ένα ακόμη παράδειγμα θα μπορούσε να είναι ο διαχωρισμός μηνυμάτων ηλεκτρονικού ταχυδρομείου ως ανεπιθήμητα και μη. Όπως γίνεται αντιληπτό, αναφερόμαστε σε δύο προβλήματα με παρόμοια χαρακτηριστικά αλλά διαφορετικής φύσεως, καθώς ο στόχος έχει διαφορετική μορφή.

Οι τεχνικές λοιπόν που χρησιμοποιεί η επιβλεπόμενη μηχανική μάθηση είναι η ταξινόμηση και η παλινδρόμηση. Η τεχνική της ταξινόμησης αποσκοπεί στην διακριτή κατηγοριοποίηση των δεδομένων ενώ η παλινδρόμηση αποφέρει συνεχή αποτελέσματα.

Μερικοί δημοφιλείς αλγόριθμοι που εντάσσονται σε αυτές τις κατηγορίες είναι οι Logistic Regression, Naive Bayes -MultiomialNB, GaussianNB, BernoulliNB ,Stochastic Gradient Descent, K-Nearest Neighbours, Decision Tree, Random Forest, Support Vector Machine [46].

Ορισμένες μετρικές αξιολόγησης που είναι ευρέως διαδεδομένες για τις εν λόγω τεχνικές είναι οι accuracy, precision, recall, f1_score, roc-curve, precision-recall curve,confusion matrix για την ταξινόμηση και r2_score,rmse,mape,mae και λοιπές για την παλινδρόμηση αντίστοιχα.[21]

3.2.2 Μη Επιβλεπόμενη Μάθηση

Σε αντίθεση με την επιβλεπόμενη μάθηση (unsupervised learning) που χρησιμοποιεί μόνο χαρακτηρισμένα δεδομένα για την εκπαίδευση των αλγορίθμων, η μη επιβλεπόμενη βασίζεται αποκλειστικά σε μη χαρακτηρισμένα δεδομένα [45].

Η συγκέκριμένη κατηγορία μηχανικής μάθησης έχει ως στόχο την ανακάλυψη κάποιας δομής, και όχι συγκεκριμένης συναρτησιακής σχέσης, που προσομοιώνει καλύτερα τα μη κατηγοριοποιημένα δεδομένα εισόδου. Επιπλέον, θα μπορούσε να εφαρμοστεί ως αρχικό στάδιο προεπεξεργασίας των δεδομένων, τροφοδοτώντας στη συνέχεια αλγορίθμους επιβλεπόμενης μάθησης.

Οι εφαρμογές αυτού του κλάδου μάθησης συναντώται πολύ συχνά σε έργα επιχειρήσεων που προσπαθούν να αποσπάσουν ουσιώδη συμπεράσματα από ακατέργαστες ροές δεδομένων, με σκοπό τη βελτίωση της λειτουργικής τους απόδοσης, καθώς και λοιπών επιχειρησιακών δεικτών. Ένας τέτοιος τομέας εμαρμογής είναι το ψηφιακό μάρκετινγκ και η διαφήμιση.

Υπάρχουν δύο τεχνικές, ανάλογα με τη φύση του εκάστοτε προβλήματος, η συσταδοποίηση (clustering) και η μείωση διαστασιμότητας (dimensionality reduction).

- Η συσταδοποίηση [45] είναι μια επεξηγηματική τεχνική που χρησιμοποιείται για την κατηγοριοποίηση δεδομένων σε ομάδες, συστάδες, χωρίς να υπάρχει κάποια αρχική πληροφορία για το είδος ή το πλήθος των συστάδων που μπορούν να δημιουργηθούν. Βασίζεται αποκλειστικά στα εσωτερικά μοτίβα του συνόλου, κυρίως στις ομοιότητες που χαρακτηρίζουν την κάθε ομάδα και στις διαφορές τους από τις υπόλοιπες.
- Η μείωση της διαστασιμότητας [45] χρησιμοποιείται για την ανεύρεση καλύτερης και απλούστερης αναπαράστασης του αρχικού συνόλου δεδομένων εισόδου μέσω λιγότερων χαρακτηριστικών. Μέσω αυτής της τεχνικής απορρίπτονται οι μη σημαντικές συνιστώσες/χαρακτηριστικά, του συνόλου δεδομένων, τα οποία που δεν παρουσιάζουν ισχυρή συνεκτικότητα με την επιθυμητή έξοδο και συνεπώς, δεν συνεισφέρουν

στην εκπαίδευση του αλγορίθμου. Ένας ακόμη κύριος λόγος προτίμησης της μείωσης διαστασιμότητας είναι η συμπίεση και ως εκ τούτου η μείωση του αποθηκευτικού χώρου που καταναλώνεται από τα δεδομένα. Εν αντιθέσει με την επιβλεπόμενη , η εν λόγω κατηγορία μάθησης δεν έχει δείκτες αξιολόγησης καθώς δεν υπάρχει κάποια αρχική κατηγοριοποιήση των δεδομένων.

Ορισμένοι δημοφιλείς αλγόριθμοι μη επιβλεπόμενης μάθησης είναι οι K-means Clustering,Association Rule, t-SNE,PCA [45].

3.2.3 Ημι-επιβλεπόμενη μάθηση

Η ημι-επιβλεπόμενη μάθηση (semi-supervised learning) [44][1] αποτελεί μια ακόμη μέθοδο της μηχανικής μάθησης, η οποία εκμεταλλεύεται την παρουσία μη κατηγοριοποιημένων δεδομένων σε συνδυασμό με μικρό αριθμό χαρακτηρισμένων δεδομένων. Η τεχνική αυτή βρίσκεται ανάμεσα στα όρια της επιβλεπόμενης και της μη επιβλεπόμενης μάθησης. Αποτέλεσμα πολλών ερευνών αποδεικνύουν ότι η σύνθεση αυτών των ομάδων δεδομένων οδηγούν σε αξιόλογη βελτίωση των αποτελεσμάτων [47].

Στη συγκεκριμένη διπλωματική εργασία χρησιμοποιείται η ημι-επιβλεπόμενη μάθηση καθώς ο μεγαλύτερος όγκος των δειγμάτων εισόδου δεν είναι χαρακτηρισμένος και σε προβλήματα μαζικά εισερχόμενων δεδομένων, η κατηγοριοποίηση αυτών αποτελεί χρονοβόρα και υψηλού κόστους διαδικασία.

Ένα παράδειγμα εφαρμογής της ημι-επιβλεπόμενης μάθησης είναι η ανίχνευση ακατάλληλων μηνυμάτων στα μέσα κοινωνικής δικτύωσης. Σε αυτήν την περίπτωση δεν υπάρχει τρόπος να γίνει επιτυχής κατηγοριοποίηση όλου του όγκου των μηνυμάτων από φυσικό πρόσωπο και ως λύση επιλέγεται η μερική κατηγοριοποίησή τους και στη συνέχεια, στηριζόμενοι στα κατηγοπιοποιημένα ως πρότυπο, η προσπάθεια κατανόησης των υπολοίπων.

Όπως γίνεται αντιληπτό η επιτυχία του αλγορίθμου καθορίζεται σε πολύ μεγάλο βαθμό από τη σωστή σύνθεση του αρχικού σετ κατηγοριοποιημένων δεδομένων. Για τον λόγο αυτό έχουν αναπτυχθεί διάφορες τεχνικές βελτίωσης της απόδοσης του αρχικού αλγορίθμου, αξιοποιώντας τις σωστές εκτιμήσεις που κάνει στην πορεία των πρόβλεψεών του.

Οι δύο δημοφιλέστερες τεχνικές της ημι-επιβλεπόμενης μάθησης, SSL(semi-supervised learning), είναι το self-training και co-training. Όλες αυτές οι μέθοδοι αποσκοπούν στη βέλτιστη αξιοποίηση των κατηγοριοποημένων και μη δεδομένων ώστε η απόδοση του μοντέλου να φτάσει σε επίπεδο που δεν θα

μπορούσε να επιτευχθεί με τις παραπάνω δύο κατηγορίες μάθησης. Δηλαδή χρησιμοποιώντας μόνο τα χαρακτηρισμένα δεδομένα και εφαρμόζοντας επιβλεπόμενη μάθηση ή λαμβάνοντας υπ όψιν μόνο τα άγνωστα δεδομένα και εφαρμόζοντας μη επιβλεπόμενη μάθηση.

Co-training

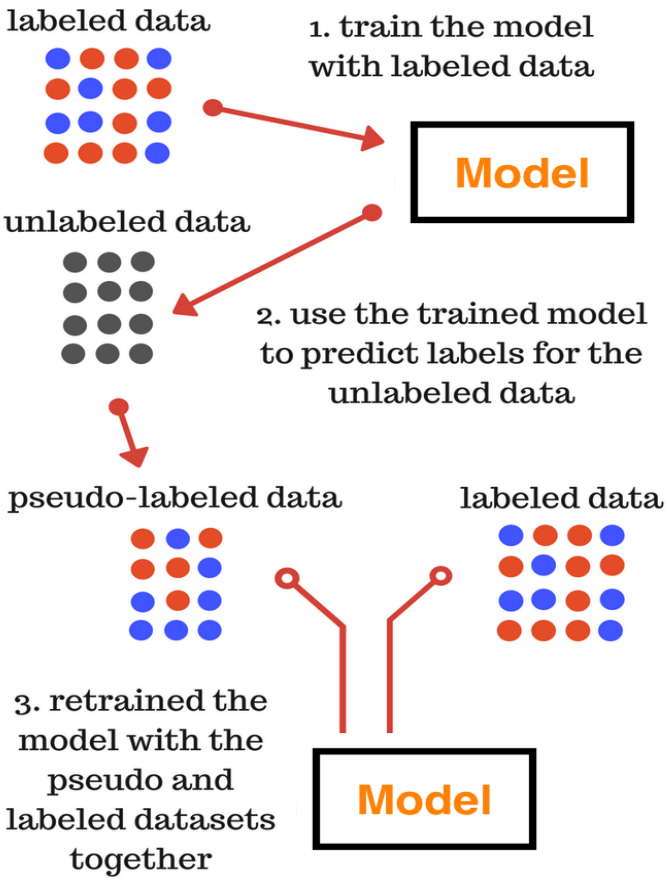
Co-training είναι ένας αλγόριθμος μηχανικής μάθησης που εκμεταλλεύεται μόνο ένα μικρό ποσοστό χαρακτηρισμένων δεδομένων και ένα μεγάλο όγκο απροσδιόριστων. Η αρχική ιδέα, που προτάθηκε από τους Blum and Mitchell [1], είναι ότι κάθε δείγμα μπορεί να χωριστεί σε δύο ανεξάρτητες όψεις, μία από τις οποίες θα είναι επαρκής για την επιτυχή εκμάθηση. Με τον όρο όψη νοείται ο διαχωρισμός των χαρακτηριστικών του αρχικού σετ σε δύο διακριτά σύνολα. Για παράδειγμα στην πρόβλεψη του χρονικού διαστήματος μετακίνησης, ένας διαχωρισμός των χαρακτηριστικών εισόδου θα μπορούσε να είναι οι συνθήκες καιρού στο πρώτο σετ και η ώρα,γιορτές και αργίες στο δεύτερο σετ. Η τεχνική του co-training λοιπόν προτείνει τη χρήση αυτών των όψεων μαζί , γεγονός που θα επιφέρει μια εύλογη αύξηση του σημαντικά μικρού αρχικού σετ δεδομένων. Συγκεκριμένα ακολουθούνται τα εξής βήματα:

1. L: σετ κατηγοριοποιημένων , U: σετ απροσδιόριστων
2. Επίλεξε u τυχαία δείγματα από το U και δημιούργησε ένα καινούργιο σετ U΄.
3. Για k επαναλήψεις

- εκπαίδευσε έναν ταξινομητή h1 με το L σύνολο διατηρώντας μόνο τα χαρακτηριστικά του που ανήκουν στην όψη x1
- εκπαίδευσε έναν ταξινομητή h2 με το L σύνολο διατηρώντας μόνο τα χαρακτηριστικά του που ανήκουν στην όψη x2
- πρόβλεψε με τον h1 τις ετικέτες του U΄ και επίστρεψε σαν έξοδο ένα μέρος, p θετικών και n αρνητικών προβλέψεων
- πρόβλεψε με τον h2 τις ετικέτες του U΄ και επίστρεψε σαν έξοδο ένα μέρος , p θετικών και n αρνητικών προβλέψεων
- σύνθεσε τις παραπάνω εξόδους και τοποθέτησε τες στο L σύνολο
- αναπλήρωσε τα 2p+2n δείγματα του U΄ , που απαλήφθηκαν, λαμβάνοντας καινούρια από το U

Self-training

Η τεχνική του self-training [1] είναι ίσως η πιο δημοφιλής και ευρέως χρησιμοποιούμενη SSL μέθοδος λόγω του συνδυασμού απλότητας και σημαντικής ακρίβειας στις ταξινομήσεις. Αποτελεί ουσιαστικά, όπως και το co-training, ένα περιτύλιγμα ενός βασικού αλγορίθμου που εκπαιδεύεται και χρησιμοποιεί τις προβλέψεις του εκ νέου στην εκπαίδευση. Συγκεκριμένα επιλέγει να εμπλουτίσει το σύνολο εκμάθησης του με τις πιο σίγουρες προβλέψεις του και, ομοίως με την προαναφερθείσα τακτική, επιφέρει μια εύλογη αύξηση του σημαντικά μικρού αρχικού σετ δεδομένων. Στην παρούσα διπλωματική έχει χρησιμοποιηθεί η τεχνική του self-training με αρκετούς κύκλους εκμάθησης ως τρόπος βελτίωσης της απόδοσης του αλγόριθμου της βάσης που επιλέχθηκε.



Αναλυτικότερα τα βήματα που ακολουθούνται είναι τα παρακάτω:

1. L :σετ κατηγοριοποιημένων , U :σετ απροσδιόριστων
2. Για k επαναλήψεις:

- εκπαίδευσε τον ταξινομητή βάσης h με το L σύνολο
- προβλεψε με τον h τις ετικέτες του U και επίστρεψε σαν έξοδο ένα μέρος προβλέψεων που ανταποκρίνονται στα κριτήρια που έχουν τεθεί ώστε να θεωρηθεί αρκετά έμπιστη και σωστή πρόβλεψη
- τοποθέτησε τις παραπάνω εξόδους στο L σύνολο
- αφαίρεσε τις παραπάνω εξόδους από το U σύνολο

Ο βαθμός εμπιστοσύνης είναι μία δημοφιλής μετρική διαλογής, η οποία εξαρτάται από την πιθανότητα που αποδίδει ο αλγόριθμος σε ένα δείγμα να

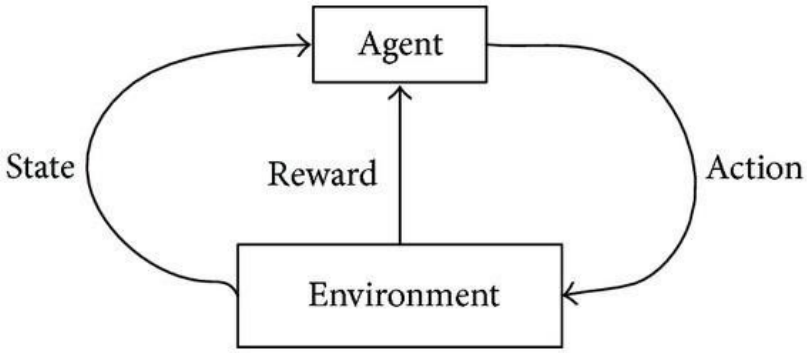
ανήκει σε κάποια κλάση. Συνήθως χρησιμοποιούνται αλγόριθμοι που υπολογίζουν, πέρα από τις τελικές προβλεπόμενες κατηγορίες των δεδομένων, αυτές τις πιθανότητες με σκόπο την παροχή τους και σε κάποια μέθοδο self-training για την ενίσχυση των αποτελεσμάτων.

Εν προκειμένω χρησιμοποιήθηκε ο Multinomial Naive Bayes για το λόγο ότι είναι ένας πιθανοτικός αλγόριθμος, καθώς και για την διακριτική του ικανότητα.

Η αξιολόγηση του τελικού μοντέλου γίνεται σε ένα μοναδικά ορισμένο δείγμα δεδομένων, για το οποίο καλείται να προβλέψει τις εκάστοτε κατηγορίες ο εκπαιδευμένος αλγόριθμος της βάσης. Αρχικά είναι εκπαιδευμένος μόνο με τα ήδη κατηγοριοποιημένα δεδομένα, ενώ στη συνέχεια συμπεριλαμβάνονται και τα υπόλοιπα ψευδο-κατηγοριοποιημένα αποτελέσματα μετά την ολοκλήρωση των κύκλων του self-training.

Ορισμένες μετρικές που είναι ευρέως διαδεδομένες για την εν λόγω τεχνική είναι οι accuracy, precision, recall, f1_score, roc-curve, precision-recall curve,confusion matrix ,όπως ακριβώς και για την ταξινόμηση.

Μερικοί ακόμη αλγόριθμοι της ημι-επιβλεπόμενης μάθησης είναι οι transductive Support Vector Machine, Naive Bayes -GaussianNB, BernoulliNB , Stochastic Gradient Descent ,graph-based methods πχ. Random Forest [44] [48] [49] [6]..



3.2.4 Ενισχυτική μάθηση (reinforcement learning)

Η ενισχυτική μάθηση [46] αποτελεί όρο που έχει δοθεί για την ομάδα των τεχνικών που χρησιμοποιούν την άμεση επαφή και αλληλεπίδραση με το εξωτερικό περιβάλλον για την εκπαίδευση του συστήματος. Αυτή η τεχνική μάθησης βασίζεται στο μοντέλο επιβράβευσης και τιμωρίας (reward - penalty), όπου ο αλγόριθμος μαθαίνει μια στρατηγική ενεργειών για μια δεδομένη παρατήρηση. Αποτελεί ένα αυτόνομο σύστημα, το οποίο παρέχει τη δυνατότητα λύσης πολύπλοκων προβλημάτων χωρίς την προϋπόθεση εξωτερικής επίβλεψης που συναντάται σε άλλες μεθόδους μηχανικής μάθησης. Αυτό είναι σημαντικό γιατί το περιβάλλον, στο οποίο πρόκειται να δραστηριοποιηθεί το εκάστοτε σύστημα, συχνά δεν είναι γνωστό εκ των προτέρων.

Όπως προαναφέρθηκε στόχος αυτής της κατηγορίας είναι η ανάπτυξη ενός αυτοσυντηρούμενου και αυτοδίδακτου αλγόριθμου, ο οποίος μπορεί να βελτιώνεται συνεχώς με την παροχή νέων πληροφοριών. Αυτές οι πληροφορίες λαμβάνονται μέσω δοκιμών που εκτελεί ο αλγορίθμος, δηλαδή μέσω του αποτελέσματος των βημάτων που επιλέγει, καθώς και μέσω ήδη γνωστών ή καινούριων δεδομένων.

Κατά τη διάρκεια της εκπαίδευσης του μοντέλου οι σωστές και οι λάθος δοκιμές επιβραβεύονται ή τιμωρούνται μέσω μίας επιβράβευσης (θετικό σήμα) ή μέσω μίας ποινής (αρνητικό σήμα).

Ένα αρκετά χαρακτηριστικό παράδειγμα είναι η εκμάθηση ενός ρομπότ να εξερευνά άγνωστους για εκείνο χώρους, λαμβάνοντας δεδομένα και επεξεργάζοντάς τα με τη συγκεκριμένη μέθοδο για την επιλογή των επόμενων κινήσεων του. Εκτός αυτού, η ενισχυτική μάθηση ξεκίνησε να γίνεται δημοφιλής μετά από συγκλονιστικές αποδόσεις σε επιτραπέζια και βίντεο παιχνίδια .

3.2.5 Είδη αλγορίθμων

Όπως έχει ήδη αναφερθεί, υπάρχουν διάφορα είδη machine learning αλγορίθμων που όμως χωρίζονται στις εξής τέσσερις βασικές κατηγορίες: supervised learning algorithms, semi-supervised algorithms, unsupervised και reinforcement algorithms.

- Supervised

Τα δύο βασικότερα είδη αυτής της κατηγορίας αλγορίθμων είναι :

- Classification [46] [32]

Είναι η διαδικασία με την οποία αναλύουμε και κατηγοριοποιούμε τα διάφορα δεδομένα με βάση τα χαρακτηριστικά τους σε διακριτές κατηγορίες. Έτσι λοιπόν, χρησιμοποιούμε τα προς εκπαίδευση δεδομένα και τα χαρακτηριστικά τους για να καθορίσουμε τα κριτήρια τα οποία θα χρησιμοποιήσουμε για να χωρίσουμε τα τελικά δεδομένα μας στις διάφορες κατηγορίες. Αφού καθοριστούν αυτά τα κριτήρια διαχωρισμού το επόμενο βήμα είναι να προβλέψουμε για τα δεδομένα στα οποία θα αξιολογηθεί το μοντέλο μας (test data) τις αντίστοιχες κατηγορίες/κλάσεις που ανήκουν. Για παράδειγμα, αν έχουμε ένα αρχικό σύνολο δεδομένων που περιέχει τα φυσικά χαρακτηριστικά διαφόρων φρούτων όπως χρώμα, σχήμα, μήκος, βάρος και την κατηγορία τους (μηλο, πορτοκάλι,μπανάνα κτλ), ο αλγόριθμος κατηγοριοποίησης που χρησιμοποιούμε (classification algorithm) θα εκπαιδευτεί έτσι ώστε να μπορεί με είσοδο τα αντίστοιχα χαρακτηριστικά να αποφασίζει αν ένα φρούτο είναι μήλο, πορτοκάλι ή μπανάνα .

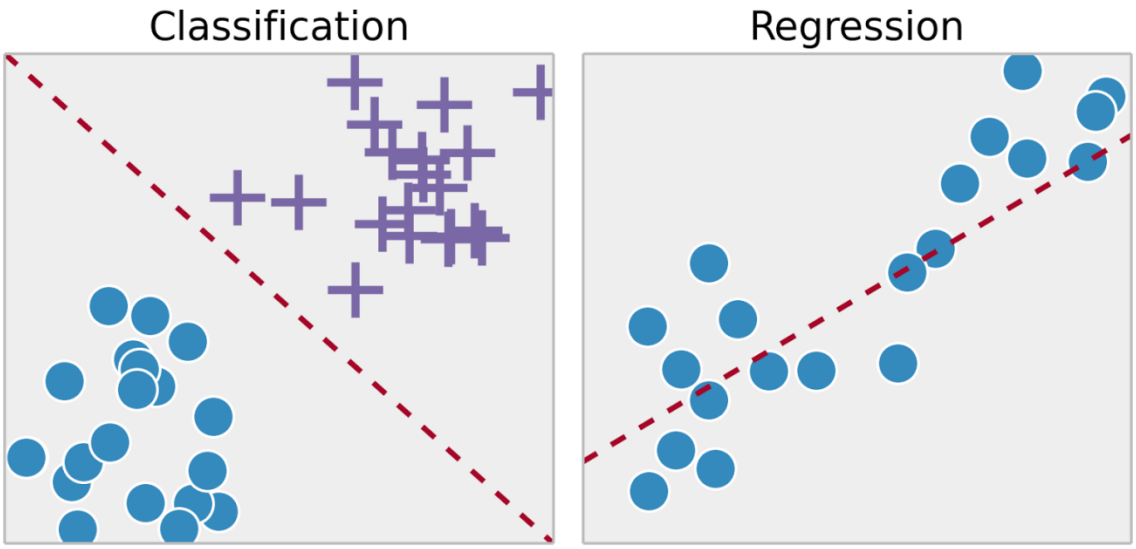
- Regression [46] [15]

Κάθε εξίσωση που περιλαμβάνει διάφορες ανεξάρτητες μεταβλητές, συντελεστές και μια εξαρτημένη μεταβλητη που η τιμή της καθορίζεται από τα προηγούμενα και τις σχέσεις που τα συνδέουν είναι μια συνάρτηση παλινδρόμησης (regression). Στον τομέα της μηχανικής μάθησης οι regression αλγόριθμοι προσπαθούν να καθορίσουν αυτήν ακριβώς την εξίσωση που συνδέει τις μεταβλητές είσοδού (γνωστές μεταβλητές) με την μεταβλητή της οποίας την τιμή θέλουν να προσδιορίσουν. Πιο απλά, με αυτήν τη διαδικασία, που στη στατιστική ονομάζεται regression, προσπαθούμε να καθορίσουμε την σχέση-εξάρτηση που έχουν οι διάφορες μεταβλητές εισόδου (χαρακτηριστικά) με την έξοδο που θέλουμε να προβλέψουμε. Έτσι λοιπόν, δίνοντας στον αλγόριθμο τις τιμές των ανεξάρτητων μεταβλητών αλλά και το αποτέλεσμα της εξαρτημένης μεταβλητής για διάφορες παρατηρήσεις αυτός εκπαιδεύεται

και μπορεί να προσδιορίσει τους συντελεστές – βάρη των εισόδων και τελικά να σχηματίσει μια συνάρτηση που θα κάνει τις προβλεψεις. Η συνάρτηση αυτή θα είναι της μόρφης:

$$y \sim f(x ; w)$$

όπως για παράδειγμα, η: **y = 0.5 x1 + 2.15 x2 + 0.76 x3**



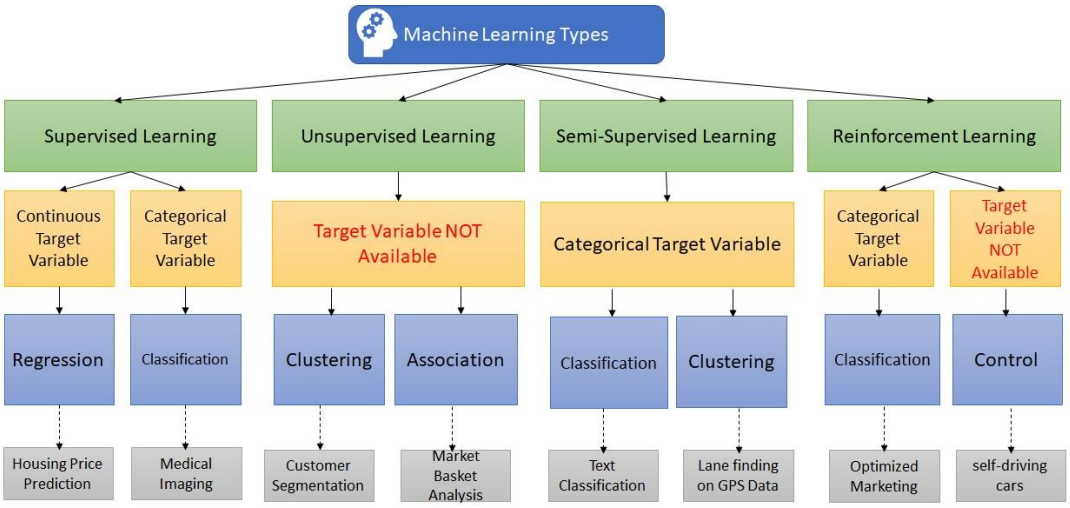
- Unsupervised

Σε αυτή τη κατηγορία υπάρχει ένα κύριο είδος αλγορίθμων:

- Clustering Algorithms [46] [31]

Είναι η διαδικασία ομαδοποίησης παρατηρήσεων που έχουν κοινά χαρακτηριστικά και εμφανίζουν ομοιότητες. Καθώς σε αυτή την κατηγορία δεν έχουμε labels για τα δεδομένα μας ο στόχος είναι να μπορέσουμε να εκπαιδεύσουμε τον αλγόριθμο ώστε να μπορεί να εντοπίσει ομοιότητες στα χαρακτηριστικά των δεδομένων και με βάση αυτές να δημιουργήσει ομάδες-κατηγορίες. Αυτό το είδος αλγορίθμων είναι πολύ σημαντικό, καθώς στον πραγματικό κόσμο δεν υπάρχουν πάντα ιδανικά δεδομένα, δηλαδή δεδομένα που να έχουν ήδη τις κλάσεις που ανήκουν. Έτσι λοιπόν, το να μπορούμε να δημιουργήσουμε κατηγορίες σε τέτοιου είδους δεδομένα εκμεταλλευόμενοι τις ομοιότητες και τις διαφορές τους είναι εξαιρετικά χρήσιμο και εφαρμόζεται σε διάφορες περιπτώσεις, όπως η ανίχνευση ανωμαλιών (anomaly detection) στα συστήματα ασφαλείας των τραπεζών. Μερικοί από τους πιο γνωστούς αλγορίθμους αυτού του είδους είναι οι: K-mean clustering και Hierachical clustering [45].

Μια συνολική απεικόνιση όλων όσων αναφέραμε παραπάνω για τα είδη εκμάθησης των αλγορίθμων αλλά και τις υποκατηγορίες αυτών δίνει η ακόλουθη εικόνα:



3.3 Κατηγοριοποίηση Φυσικού Κειμένου (Text Classification)

Στην ενότητα αυτή παρουσιάζεται και αναλύεται μια πολύ σημαντική έννοια για την συγκεκριμένη διπλωματική εργασία, στην οποία αναφερόμαστε συνήθως ως κατηγοριοποίηση φυσικού κειμένου ή text classification (ακόμα και ως text categorization ή text tagging).
Text classification είναι η διαδικασία μέσω της οποίας μπορούμε να κατηγοριοποιήσουμε κείμενο σε φυσική γλώσσα με κριτήριο το περιεχόμενό του [50]. Είναι ένας από τους κυριότερους τομείς της επεξεργασίας φυσικής γλώσσας και εφαρμόζεται σε διάφορες τομείς όπως sentiment analysis, spam detection, topic labeling και άλλα.
Κείμενα συναντάμε παντού στην καθημερινότητά μας, σε email, άρθρα, ιστοσελίδες, κριτικές προϊόντων και πολλά άλλα και μάλιστα σε αυτά περιέχεται μεγάλος πλήθος πληροφοριών που αν μπορούσαμε να τις εξάγουμε θα ηταν εξαιρετικά χρήσιμες. Γι' αυτό το σκοπό λοιπόν, χρησιμοποιούμε text classification αλγορίθμους ώστε να μπορέσουμε για παράδειγμα να κατηγοριοποιήσουμε άρθρα ανάλογα με το θέμα τους (αθλητικά, πολιτική κ.α), συζητήσεις στα κοινωνικά δίκτυα ανάλογα με τη γλώσσα που χρησιμοποιείται ή κριτικές σε προϊόντα σχετικά με το αν είναι θετικές ή αρνητικές.
Πως λειτουργεί όμως αυτή η διαδικασία;
Η αυτοματοποιημένη κατηγοριοποίηση κειμένου μπορεί να χωριστεί σε τρεις υποκατηγορίες ανάλογα με τους τρόπους που χρησιμοποιούνται για να εκπαιδευτεί ένα σύστημα [10]:

- Συστήματα με χρήση κανόνων (rule-based systems) [10]

Οι προσεγγίσεις που στηρίζονται σε κανόνες προσπαθούν να διαχωρίσουν σε κατηγορίες τα κείμενα χρησιμοποιώντας σύνολα αυστηρών,γλωσσικών κανόνων που τους έχουν λάβει ως είσοδο. Πιο συγκεκριμένα, ορίζοντας λέξεις και φράσεις να αντιστοιχούν στις διάφορες κατηγορίες ανάλογα με το νόημά τους, διαμορφώνονται οι κανόνες που ουσιαστικά καθορίζουν το σύστημα. Για παράδειγμα, αν θέλουμε να κατατάξουμε άρθρα από ηλεκτρονικές εφημερίδες σε δυο κατηγορίες, έστω πολιτικά και αθλητικά, θα πρέπει πρώτα να ορίσουμε μια λιστα από λέξεις και φράσεις που περιλαμβάνουν αθλητική ορολογία και συναντώνται συχνότερα σε αθλητικές περιγραφές και μια αντίστοιχη για τα πολιτικά. Στη συνέχεια, δίνοντας ως είσοδο ένα κείμενο, θα μετρηθεί το πλήθος των λέξεων που υπάρχουν και στο κείμενο και στις λίστες και για όποια λίστα αυτός ο αριθμός είναι μεγαλύτερος στην αντίστοιχη κατηγορία θα ταξινομηθεί το συγκεκριμένο κείμενο.

Ένα βασικό παράδειγμα ανάπτυξης τέτοιου συστήματος είναι:

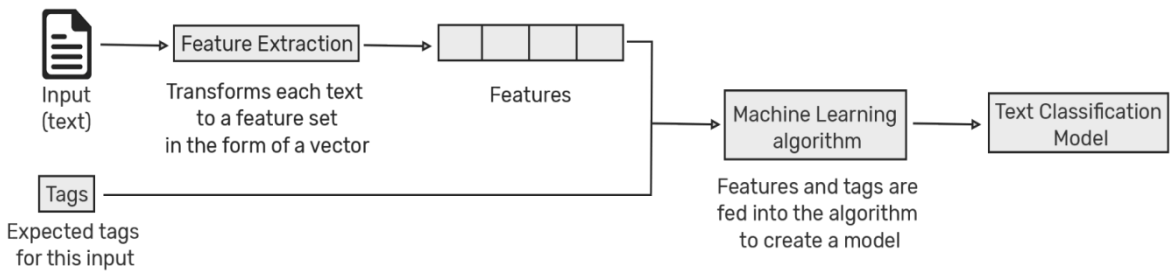
1. Καθόρισε δύο λίστες με πολωμένες λέξεις

2. Δεδομένου ενός κειμένου μέτρησε τις θετικές και αρνητικές λέξεις που εμφανίζονται σε αυτό
3. Αν ο αριθμός λέξεων της θετικής κατηγορίας είναι μεγαλύτερος από αυτές της αρνητικής, επίστρεψε το κείμενο ως θετικό. Σε αντίθετη περίπτωση επίστρεψε αρνητικό, αλλιώς επίστρεψε ουδέτερο.

Αυτή η προσέγγιση λειτουργεί ικανοποιητικά σε απλές εφαρμογές, όμως παρουσιάζει κάποια σοβαρά μειονεκτήματα καθώς δεν λαμβάνει υπ' όψιν τον τρόπο που συνδυάζονται οι παραπάνω λέξεις. Αρχικά, απαιτεί πολύ χρόνο και μεγάλη ανάλυση για τον ορισμό των κανόνων που θα χρησιμοποιηθούν καθώς και βαθιά γνώση του αντικειμένου ώστε οι κανόνες να αποδίδουν σωστά. Επίσης, αυτά τα συστήματα δεν είναι εύκολα επεκτάσιμα αφού η προσθήκη επιπλέον κανόνων ή η χρήση των ήδη υπαρχόντων σε παρόμοιες εφαρμογές δεν μπορούν να εγγυηθούν αντίστοιχα αποτελέσματα. Ως αποτέλεσμα, τα συστήματα αυτά απαιτούν σημαντικές επενδύσεις για το χειροκίνητο συντονισμό και τη διατήρηση των κανόνων.

- Συστήματα βασισμένα στη μηχανική μάθηση (machine-learning systems) [10]

Σε αυτήν την περίπτωση αντί να χρησιμοποιούνται για εκπαίδευση κανόνες που έχουν δημιουργηθεί και ορισθεί από κάποιον, το σύστημα μαθαίνει από προηγούμενες παρατηρήσεις. Δηλαδή, χρησιμοποιώντας ως είσοδο κείμενα τα οποία έχουν ήδη κατηγοριοποιηθεί και περιλαμβάνουν ετικέτες (labels), ο αλγόριθμος μηχανικής μάθησης σταδιακά καταλαβαίνει τη συσχέτιση των κειμένων με τις αντίστοιχες κατηγορίες.
Το πρώτο βήμα για να πετύχουμε κάτι τέτοιο είναι να μετατρέψουμε το κείμενο σε μια μορφή που μπορεί να γίνει κατανοητή και να αξιοποιηθεί από τον εκάστοτε αλγόριθμο. Έτσι, συνήθως μετατρέπουμε τα κείμενα σε διανύσματα, όπου κάθε λέξη εκφράζεται με την συχνότητα εμφάνισής της στο κείμενο. Έπειτα αυτό δίνεται ως είσοδος στον αλγόριθμο για πολλά διαφορετικά κείμενα τα οποία συνοδεύονται από τις κατηγορίες στις οποίες ανήκουν με τελικό σκοπό να παραχθεί ένα μοντέλο κατηγοριοποίησης(classification model) που θα χρησιμοποιηθεί για εκτιμήσεις σε καινουρια δεδομένα.



Μόλις το σύστημα μας λοιπόν δεχθεί ως είσοδο επαρκές πλήθος δεδομένων και ετικετών (labels) μπορεί να πραγματοποιήσει ακριβείς προβλέψεις σε κείμενα που δεν έχει ξανασυναντήσει ποτέ.
Τα αποτελέσματα χρησιμοποιώντας αλγόριθμους μηχανικής μάθησης είναι αρκετά πιο ακριβή από αύτα που προκύπτουν με χρήση κανόνων, γιατί δεν βασίζονται σε περιορισμένες και σταθερές εισόδους προς εκπαίδευση αλλά το μοντέλο διαμορφώνεται δυναμικά από τις προηγούμενες παρατηρήσεις και τα χαρακτηριστικά τους. Επίσης, είναι πολύ πιο εύκολα η διαχείριση αλλά και η επέκτασή τους με την προσθήκη είτε επιπλέον παρατηρήσεων είτε παρατηρήσεων με διαφορετικά χαρακτηριστικά.

- Υβριδικά συστήματα (hybrid systems) [10]

Τα υβριδικά συστήματα συνδυάζουν τους δυο τρόπους που προαναφέρονται για περαιτέρω βελτίωση των αποτελεσμάτων. Χρησιμοποιούν έναν βασικό αλγόριθμο μηχανικής μάθησης για να εκπαιδεύσουν το σύστημα ώστε να πραγματοποιήσει τις πρώτες προβλέψεις και στη συνέχεια σε όσες προβλέψεις δεν πέτυχαν το επιθυμητό αποτέλεσμα εφαρμόζουν ένα σύνολο κανόνων για να προσδιοριστεί σωστά αυτή τη φορά η εκτίμησή τους.
Σύμφωνα με την IBM το 80% όλων των δεδομένων βρίσκονται σε μη δομημένη μορφή και τα κείμενα αποτελούν μια από τις πιο δημοφιλείς μορφές δεδομένων σε αυτή τη μορφή [10]. Λόγω, λοιπόν, της ακανόνιστης μορφής των κειμένων η ανάλυση, η επεξεργασία και η ομαδοποίησή τους αποτελούν δύσκολα και χρονοβόρα εγχειρήματα. Εδώ φαίνεται η αξία της αυτόματης επεξεργασίας και κατηγοριοποίησης των κειμένων ιδίως με μεθόδους μηχανικής μάθησης, καθώς με αυτόν τον τρόπο οι διάφορες επιχειρήσεις και οργανισμοί μπορούν να οργανώσουν και να εξάγουν πληροφορίες από διάφορες μορφές κειμένων όπως emails, νομικά έγγραφα, συνομιλίες στα κοινωνικά δίκτυα και άλλα.
Τα βασικά πλεονεκτήματα αυτής της προσέγγισης είναι:

- η επεκτασιμότητα, καθώς για να μπορέσουν να διαβαστουν, να αναλυθούν και να κατηγοριοποιηθούν εκατομμύρια κείμενα από ανθρώπους απαιτείται μεγάλο χρονικό διάστημα και υψηλό κόστος ενώ με τον αυτοματοποιημένο τρόπο οι απαιτήσεις σε χρόνο και πόρους μειώνονται σημαντικά
- η απόκριση σε πραγματικό χρόνο. Υπάρχουν περιπτώσεις που οι επιχειρήσεις χρειάζονται να πάρουν αποφάσεις και να δράσουν άμεσα ανάλογα με τις αλλαγές στο ευρύτερο περιβάλλον. Η κατηγοριοποίηση κειμένων μέσω μηχανικής μάθησης μπορεί να πραγματοποιήσει αναλύσεις και εκτιμήσεις σε δεδομένα πραγματικού χρόνου με άμεσα αποτελέσματα.

- η ακρίβεια, αφού κάθετι που περιλαμβάνει τον ανθώπινο παράγοντα εμπεριέχει και αυξημένη πιθανότητα λάθους, ενώ μέσω των machine learning αλγορίθμων επιτυγχάνονται μεγάλες επιδόσεις και υψηλά ποσοστά ακρίβειας.

Text classification χρησιμοποιείται σε πολύ μεγάλο βαθμό και σε ένα πολύ μεγάλο εύρος περιοχών, με πιο γνωστες τις ανάλυση αισθήματος (sentiment analysis), εξαγωγή λέξεων-κλειδιών (keyword extraction), επισήμανση θέματος (topic labeling) και αναγνώριση γλώσσας (language detection). Οι έννοιες αυτές εξηγούνται λεπτομερώς στη συνέχεια. Ένα χαρακτηριστικό παράδειγμα που δείχνει την σπουδαιότητα τους, είναι η ανάλυση και η εξαγωγή συμπερασμάτων από δημοσιεύσεις στο κοινωνικό δίκτυο Twitter λίγο πριν το δημοψήφισμα που έγινε στη Μεγάλη Βρετανία σχετικά με το Brexit [10]. Εκατομμύρια δημοσιεύσεις που περιελάμβαν την λέξη Brexit ή #brexit συλλέχθηκαν και αναλύθηκαν για να κατηγοριοποιηθούν ως προς την πολικότητά τους (θετικά/αρνητικά),sentiment analysis, αλλά και ως προς το περιεχόμενο τους και αυτά που ανέφεραν για το συγκεκριμένο θέμα. Τα απότελέσματα ήταν εξαιρετικά σημαντικά και έδειχναν την τάση των πολιτών ως προς την απόφαση εξόδου από την ΕΕ.

3.4 Ανάλυση αισθήματος (sentiment analysis)

Ανάλυση αισθήματος [47], εξόρυξη γνώμης διαφορετικά, ονομάζεται η υπολογιστική ανάλυση της άποψης των χρηστών πάνω σε ένα αντικείμενο με στόχο την κατανόηση αυτής. Αυτή η ανάλυση τεχνικά ονομάζεται sentiment analysis και είναι μία αυτοματοποιημένη διαδικασία που χρησιμοποιεί την τεχνητή νοημοσύνη για την αναγνώριση της θετικής ,αρνητικής ή ουδέτερης πολικότητας των απόψεων διαφορετικών χρηστών. Με αυτόν τον διαχωρισμό διευκολύνεται η απόκτηση γνώσεων από τα σχόλια των κοινωνικών μέσων ενημέρωσης, τις απαντήσεις ερευνών και τις αναθεωρήσεις των προϊόντων και προάγεται η πιο στοχευμένη λήψη αποφάσεων που βασίζονται στις ανάγκες των πελατών.

Σε μια εποχή που παράγεται ένας τεράστιος όγκος δεδομένων ημερησίως αντιλαμβανόμαστε ότι η ανάλυση αισθήματος έχει καταστεί βασικό εργαλείο για την κατανόηση αυτών των δεδομένων.

3.4.1 Επίπεδα ανάλυσης

Υπάρχουν πολλές διαφορετικές προσεγγίσεις όσον αφορά στην οπτική γωνία ανάλυσης αλλά και στην ακρίβεια σχετικά με το επίπεδο της πολικότητας της γνώμης.[28][47]

Στη συνέχεια παρουσιάζονται τρία επίπεδα εφαρμογής της ανάλυσης.

Επίπεδο εγγράφων: Η ανάλυση γίνεται σε ολόκληρο το εύρος ενός εγγράφου ή μίας παραγράφου.

Επίπεδο πρότασης: Η ανάλυση γίνεται σε κάθε μία πρόταση ενός κειμένου ξεχωριστά.

Επίπεδο υπο-πρότασης: Πέρα από την ανάλυση της πρότασης διατηρούνται και οι πολικότητες επιμέρους φράσεών της.

3.4.2 Επίπεδο ακρίβειας πολικότητας

Μερικές φορές ίσως πρέπει να είμαστε πιο ακριβείς σχετικά με το επίπεδο πολικότητας της γνώμης, οπότε αντί να κατηγοριοποιούμε τις απόψεις μόνο σε θετικές, ουδέτερες ή αρνητικές, θα μπορούσαμε να εξετάσουμε και τις ακόλουθες κατηγορίες:

- Πολύ θετική
- Θετική
- Ουδέτερη
- Αρνητική
- Πολύ αρνητική

Αυτό συνήθως αναφέρεται ως λεπτομερής ανάλυση αισθημάτων. Αυτό θα μπορούσε, για παράδειγμα, να αντιστοιχηθεί σε βαθμολογία 5 αστέρων σε ένα σχόλιο χρήστη ως εξής:
Πολύ θετική = 5 αστέρια έως πολύ αρνητική = 1 αστέρι.

Ορισμένα συστήματα μάλιστα παρέχουν ακόμη και διαφορετικές μορφές αντιστοίχισης της απλής πολικότητας, προσδιορίζοντας εάν η θετική ή αρνητική γνώμη συνδέεται με ένα συγκεκριμένο αίσθημα, όπως ο θυμός, η θλίψη ή οι ανησυχίες (δηλαδή τα αρνητικά αισθήματα) ή η ευτυχία, η αγάπη ή ο ενθουσιασμός (δηλαδή τα θετικά αισθήματα).[28]

Για τον λόγο αυτό έχουν αναπτυχθεί και διάφοροι τύποι που δίνουν έμφαση σε διαφορετικές μορφές ανάλυσης.

Αναγνώριση αισθήματος

Η αναγνώριση αισθήματος στοχεύει στην ανίχνευση αισθημάτων όπως χαρά,απογοήτευση,θυμός, λύπη και λοιπα. Πολλά τέτοιου είδους συστήματα βασίζονται σε λεξικά, δηλαδή στο πλήθος λέξεων και στα

αισθήματα που εκφράζουν, ή σε πολύπλοκους αλγόριθμους μηχανικής μάθησης.

Ένα από τα μειονεκτήματα της πρώτης επιλογής είναι το γεγονός ότι ο κάθε άνθρωπος εκφράζει τα αισθήματά του με διαφορετικό τρόπο, συνεπώς θα ήταν λάθος να θεωρούμε ότι δεδομένες φράσεις έχουν το ίδιο βάρος. Επίσης, θα ήταν λάθος να θεωρηθεί ότι δηλώνουν το ίδιο αίσθημα καθώς οι ίδιες λέξεις θα μπορούσαν να είναι διφορούμενες. Ένα παράδειγμα είναι το εξής, *your customer support is killing me, You are killing it.* Το πρώτο σχόλιο δηλώνει θυμό ,ενώ το δεύτερο ικανοποιήση και χαρά.

Ανάλυση βάσει άποψης

Συνήθως αναλύοντας το συνάισθημα σε ενότητες, για παράδειγμα σχόλια προιόντων, ίσως θα ήταν ενδιαφέρουσα όχι μόνο η εύρεση της πολικότητας των σχολίων, αλλά και ο προσδιορισμός των απόψεων ή των χαρακτηριστικών των προιόντων στα οποία στηρίζεται η γνώμη του. Για παράδειγμα στην πρόταση *"The battery life of this camera is too short."* ο χρήστης εκφράζει ένα αρνητικό αίσθημα αλλά και την αίτια που το προκαλεί, δηλαδή την μικρή διάρκεια ζωής της μπαταρίας της καινούργιας του κάμερας.

Ανάλυση βάσει πρόθεσης

Η ανάλυση βάσει πρόθεσης ανιχνεύει τι θα ήθελαν οι χρήστες να γίνει ώστε να είναι ευχαριστημένοι. Ορισμένα παραδείγματα

“Your customer support is a disaster. I’ve been on hold for 20 minutes”.

“I would like to know how to replace the cartridge”.

“Can you help me fill out this form?”

Ένα φυσικό πρόσωπο δεν θα είχε κάποιο πρόβλημα στο να αναγνωρίσει τέτοιου είδους παράπονα ,όπως στο πρώτο σχόλιο, ερωτήσεις,όπως στο δεύτερο,ή αιτήματα ,όπως στο τρίτο.[28]

Παρ’όλα αυτά μία μηχανή θα δυσκολευτεί να τις αντιληφθεί καθώς πρόκειται για εκφράσεις φυσικής γλώσσας που κρύβουν νοήματα στα συμφραζόμενα.

3.4.3 Κατηγορίες αλγορίθμων

Όπως παρουσιάστηκε και σε προηγούμενη ενότητα ενδελεχώς, υπάρχουν πολλές μέθοδοι και αλγόριθμοι ανάπτυξης συστημάτων ανάλυσης αισθήματος, οι οποίοι κατηγοριοποιούνται ως εξής:[28]

- Συστήματα **βάσει κανόνων** που εφαρμόζουν την ανάλυση βασιζόμενοι σε χειροποίητους κανονισμούς
- **Αυτόματα** συστήματα που στηρίζονται σε τεχνικές μηχανικής μάθησης για να μάθουν από τα δεδομένα.
- **Υβριδικά** συστήματα τα οποία συνδυάζουν τις δύο παραπάνω κατηγορίες.

Συστήματα βάσει κανόνων

Συνήθως τέτοιες προσεγγίσεις ορίζουν ένα σύνολο κανόνων σε κάποιο είδος κώδικα που προσδιορίζει την υποκειμενικότητα, την πολικότητα ή το θέμα μίας γνώμης.

Οι κανόνες μπορεί να χρησιμοποιούν ένα πλήθος εισόδων , όπως για παράδειγμα τεχνικές ανάλυσης φυσικής γλώσσας ή λεξικά, δηλαδή λίστες από λέξεις και εκφράσεις.

Αυτόματα συστήματα

Οι αυτόματες μέθοδοι, αντίθετα με τα συστήματα που βασίζονται σε κανόνες, δεν στηρίζουν τη λειτουργία τους σε χειρονακτικά επεξεργασμένους κανόνες, αλλά στις τεχνικές μηχανικής μάθησης. Η τεχνική ανάλυσης αισθημάτων συνήθως διαμορφώνεται ως πρόβλημα ταξινόμησης ,όπου ένας αλγόριθμος τροφοδοτείται με ένα κείμενο και επιστρέφει την αντίστοιχη κατηγορία, π.χ. θετικό, αρνητικό ή ουδέτερο (σε περίπτωση που πραγματοποιείται ανάλυση πολικότητας). Ο εν λόγω ταξινομητής μηχανικής μάθησης μπορεί συνήθως να αναπτυχθεί με τα ακόλουθα βήματα.

Διαδικασία εκμάθησης και πρόβλεψης

Κατά την εκπαίδευση το μοντέλο μαθαίνει να σχετίζει ένα συγκεκριμένο κείμενο εισόδου με μια κατηγορία εξόδου, βασιζόμενο στα επιμέρους χαρακτηριστικά των δειγμάτων στα οποία εκπαιδεύτηκε. Το μέρος του συστήματος που είναι υπεύθυνο για την εξαγωγή των χαρακτηριστικών μετατρέπει το κείμενο σε ένα διάνυσμα εισόδου με τον τρόπο που περιγράφηκε στην ενότητα του text classification.

Κατά τη διαδικασία πρόβλεψης, ο εξαγωγέας χαρακτηριστικών μετατρέπει το κείμενο στην επιθυμητή διανυσματική μορφή και στη συνέχεια δίνεται ως είσοδος στον αλγόριθμο ταξινόμησης.

Εξαγωγέας χαρακτηριστικών κειμένου

Η οντότητα του συστήματος που είναι υπεύθυνη για την εξαγωγή των χαρακτηριστικών/συνιστωσών ενός κειμένου είναι ο εξαγωγέας χαρακτηριστικών. Αυτή η διαδικασία είναι γνωστή και ως παραμετροποίηση κειμένου. Μία κλασσική προσέγγισή της γίνεται με τα bag-of-words ή bag-of-ngrams.[51]

Πρόσφατα , εφαρμόστηκαν νέες τεχνικές τέτοιου σκοπού με βάση τις ενσωματωμένες λέξεις. Αυτό το είδος αναπαραστάσεων επιτρέπει σε λέξεις με παρόμοιο νόημα να έχουν και μία παρόμοια αναπαράσταση, η οποία μπορεί να βελτιώσει την απόδοση των ταξονιμητών.[28] [51]

Υπάρχουν πολλοί τρόποι με τους οποίους μπορούμε να διατηρήσουμε μετρικές απόδοσης για την αξιολόγηση ενός sentiment analysis μοντέλου και την εκτίμηση της ακρίβειας αυτού.

Precision, recall, και accuracy είναι κάποιες συχνά χρημοποιούμενες μετρικές για την εξέταση της ακρίβειας ενός ταξινομητή.

Συγκεκριμένα το precision μετράει πόσα από τα κείμενα που έχουν ταξινομηθεί σε μια συγκεκριμένη κατηγορία, ανήκουν αληθώς σε αυτήν. Το recall μετράει πόσα από όλα τα κείμενα του συνόλου (κατηγοριοποιημένα και μη) έχουν προβλεφθεί σωστά. Από αυτό αντιλαμβανόμαστε ότι όσο περισσότερα δεδομένα τροφοδοτούνται στον αλγόριθμο τόσο βελτιώνεται και η συγκεκριμένη μετρική. Τέλος, το accuracy metric μετράει τη μέση τιμή των σωστών προβλέψεων. Περαιτέρω ανάλυση αυτών γίνεται σε επόμενο κεφάλαιο.

Υβριδικά συστήματα

Αναπτύσσονται με τον συνδυασμό των δύο παραπάνω μεθόδων.

3.4.4 Περίπλοκα σημεία της sentiment ανάλυσης

Υποκειμενικότητα

Η ανίχνευση της υποκειμενικότητας και της αντικειμενικότητας σε ένα σχόλιο χρήστη είναι πολύ σημαντική για την εύρεση του τόνου και συνεπώς της πολικότητας αυτού. Για παράδειγμα οι προτάσεις 'Το πακέτο είναι κόκκινο' , 'Το πακέτο είναι όμορφο' δεν εκφράζουν και οι δύο κάποιο αίσθημα. Η πρώτη είναι μια αντικειμενική δήλωση η οποία δεν εκφράζει κάποιο σαφές αίσθημα.

Συμφραζόμενα

Όλες οι δηλώσεις, σχόλια εκφράζονται σε κάποια δεδομένη χρονική στιγμή, σε κάποιο μέρος, από και σε κάποιους ανθρώπους. Συνεπώς όλες οι δηλώσεις εξαρτώνται από το πλαίσιο μέσα στο οποίο έγιναν. Για παράδειγμα, στην ερώτηση 'Τι σας άρεσε περισσότερο από τη συναυλία;' οι απαντήσεις 'Τίποτα απολύτως!' και 'Όλα' καταχωρούνται σαν αρνητική και θετική αντίστοιχα. Όμως, αν η ερώτηση ήταν 'Τι ήταν αυτό που δεν σας άρεσε από την παρουσίαση;', οι παραπάνω απαντήσεις θα είχαν το ακριβώς αντίθετο αίσθημα.

Για την αποφυγή αυτής της σύγχυσης, χρειάζεται μια επαρκής προεπεξεργασία των δεδομένων για την εξάλειψη της επιρροής των συμφραζόμενων στο τελικό νόημα των σχολίων.

Ειρωνεία και σαρκασμός

Η διαφορά μεταξύ του κυριολεκτικού μηνύματος και της ειρωνείας-σαρκασμού συνήθως μπορούν να αλλάξουν δραματικά το τελικό αίσθημα μίας πρότασης. Ένα αρνητικό μπορεί να γίνει θετικό και αντιστρόφως. Παρ' όλα αυτά η ανίχνευση αυτών των στοιχείων σαρκασμού απαιτεί μία εξειδικευμένη ανάλυση των συμφραζόμενων.

Για παράδειγμα ας εξετάσουμε τα παρακάτω σχόλια για τα χαρακτηριστικά ενός προιόντος.

'Το τηλέφωνο έχει μια καταπληκτική διάρκεια μπαταρίας, μέχρι 38 ώρες.'

'Το τηλέφωνο έχει μια καταπληκτική διάρκεια μπαταρίας, μέχρι 1 ώρα.'

Προφανώς ένα φυσικό πρόσωπο θα μπορούσε να αντιληφθεί ότι η δεύτερη απάντηση είναι σαρκαστική. Το πρόβλημα σε αυτήν την περίπτωση είναι ότι

δεν υπάρχουν συμφραζόμενα από τα οποία θα μπορούσε ένας αλγόριθμος μηχανικής μάθησης να το αντιληφθεί. [28] [47]

Σχόλια με συγκρίσεις

Μία ακόμη πρόκλιση στην ανάλυση αισθήματος είναι ο χειρισμός των συγκρίσεων σε ένα σχόλιο.

'Τουλάχιστον, είναι καλύτερο από το τίποτα.'

Αν γνωρίζουμε τα συμφραζόμενα αυτής της γνώμης τότε πολύ εύκολα θα τη θεωρούσαμε ουδέτερου αισθήματος. Ο αλγόριθμός, εν αντιθέσει, είναι δύσκολο να αντιληφθεί ότι η λέξη καλύτερο μπορεί να λάβει και αρνητικό ή ουδέτερο sentiment. Για αυτό το λόγο παίζει σπουδαίο ρόλο ο χειρισμός ολόκληρων φράσεων και όχι μόνο οι λέξεις σαν μονάδες.[51]

Μη αναγνωρίσιμα σύμβολα

Ένα απλό παράδειγμα είναι τα emojis, σχέδια που συνοδεύουν το κείμενο και συμπληρώνουν το νόημα του. Συνήθως, αυτά τα σύμβολα στο στάδιο της προεπεξεργασίας των δεδομένων απορρίπτονται. Παρ' όλα αυτά, θα ήταν πολύ χρήσιμο σε κάποιες περιπτώσεις, όπως η ανάλυση σχολίων στο tweeter, η αναγνώριση αυτών, καθώς παίζουν σημαντικό ρόλο στην πολικότητα του κειμένου.[47]

Μία τέτοια ανάλυση θα μπορούσε να διευκολυνθεί με ένα 'λεξικό' κατασκευασμένο με τα σημεία στίξης που συνθέτουν ένα τέτοιο σύμβολο και τη σύνδεσή του με το αντίστοιχο αίσθημα.

Εν κατακλείδι κάποια από τα πλεονεκτήματα του sentiment analysis είναι: [28]

Η επεκτασιμότητα

Η επεξεργασία και η ομαδοποίησή μεγάλου όγκου δεδομένων αποτελούν δύσκολα, δαπανηρά και χρονοβόρα εγχειρήματα. Αντίθετα, με το sentiment analysis σε μεγάλης κλίμακας δεδομένα, η ανάλυσή τους επιτυγχάνεται σε πολύ μικρούς χρόνους και χωρίς μεγάλο κόστος συγκριτικά, με την επεξεργασία τους από φυσικό πρόσωπο.

Ανάλυση σε πραγματικό χρόνο

Μπορούμε να χρησιμοποιήσουμε την ανάλυση αισθημάτων για να προσδιορίσουμε τις κρίσιμες πληροφορίες που επιτρέπουν την επίγνωση της κατάστασης σε συγκεκριμένα σενάρια σε πραγματικό χρόνο. Υπάρχει μια

κρίση δημοσίων σχέσεων στα κοινωνικά μέσα που πρόκειται να συμβει; Ένας θυμωμένος πελάτης που πρόκειται να αντιδράσει; Ένα σύστημα ανάλυσης αισθήματος μπορεί να βοηθήσει στον εντοπισμό τέτοιων καταστάσταων και στην έγκαιρη λήψη των κατάλληλων μέτρων.

Σταθερά κριτήρια

Οι άνθρωποι δεν τηρούν σαφή κριτήρια για την αξιολόγηση του αισθήματος ενός κειμένου. Εκτιμάται ότι διαφορετικοί άνθρωποι συμφωνούν μόνο κατά το 60-65% όταν κρίνουν το αίσθημα για ένα συγκεκριμένο κομμάτι κειμένου.[28] Συνεπώς υπεισέρχεται υποκειμενικότητα λόγω προσωπικών εμπειριών, σκέψεων και πεποιθήσεεων. Χρησιμοποιώντας ένα σύστημα ανάλυσης αισθήματος, οι εταιρείες μπορούν να εφαρμόζουν τα ίδια κριτήρια σε όλα τα δεδομένα τους. Αυτό βοηθά στη μείωση των σφαλμάτων και στη βελτίωση της συνοχής των δεδομένων.

3.5 Εξαγωγή λέξεων κλειδιών

Η εξαγωγή λέξεων κλειδιών ή αλλιώς keyword extraction, είναι μία αυτοματοποιημένη διαδικασία για τον εντοπισμό των πιο σχετικών λέξεων, εκφράσεων ή κομματιών κειμένου τα οποία περιγράφουν με τον καλύτερο δυνατό τρόπο το θέμα του κειμένου. [52]

Στη σύγχρονη εποχή, με πάνω από 290 δισεκατομμύρια μηνύματα ηλεκτρονικού ταχυδρομείου να διακινούνται ημερησίως, η χρήση μηχανών για την ανάλυση τεράστιων όγκων δεδομένων και η εξαγωγή των χρήσιμων πληροφοριών τους αποτελεί σίγουρα ένα τεράστιο εγχείρημα.[25]

Για παράδειγμα, μία επιχείρηση ιδανικά θα ήθελε να αναλύσει εκατοντάδες διαδικτυακά σχόλια για τον προιόν της. Το keyword extraction συμβάλλει στη γενική εποπτεία όλων αυτών των δειγμάτων από δεδομένα καθώς και διατηρεί τις λέξεις ή φράσεις που περιγράφουν με λίγα λόγια την κάθε γνώμη του χρήστη μέσα σε δευτερόλεπτα. Με αυτόν τον τρόπο μπορεί να δεί άμεσα τι αναφέρουν πιο συχνά οι πελάτες της γλιτώνοντας ώρες χειρονακτικής ανάλυσης.

Η εξαγωγή λέξεων-κλειδιών απλοποιεί το έργο εύρεσης σχετικών λέξεων και φράσεων μέσα σε αδόμητο κείμενο. Τέτοια παραδείγματα αδόμητων κειμένων μπορεί να είναι οι αναρτήσεις σε μέσα κοινωνικής δικτύωσης, μηνύματα ηλεκτρονικού ταχυδρομείου, συνομιλίες ή οποιοσδήποτε άλλος τύπος δεδομένων που δεν επιδέχεται οργάνωση άμεσα με κάποιο άλλο τρόπο.

Μία ακόμη λειτουργικότητα του keyword extraction είναι η αυτοματοποίηση διάφορων εργασιών, όπως η επισήμανση των εισερχόμενων απαντήσεων σε μία έρευνα ή η ανταπόκριση σε επείγοντα ερωτήματα πελατών, επιτρέποντας έτσι την εξοικονόμηση και διοχέτευση πολύτιμου χρόνου σε λοιπές εργασίες.

Υπάρχουν διάφορες τεχνικές που χρησιμοποιούνται για την αυτοματοποιημένη εξαγωγή λέξεων-κλειδιών. Από στατιστικές προσεγγίσεις που εντοπίζουν τις λέξεις-κλειδιά μετρώντας τη συχνότητα λέξεων μέχρι πιο πολύπλοκες προσεγγίσεις μηχανικής μάθησης που επιτρέπουν τη δημιουργία μοντέλων.[52][25]

Σε αυτή την ενότητα, θα εξετάσουμε τις διάφορες προσεγγίσεις για την εξαγωγή λέξεων-κλειδιών.

Στατιστικές μέθοδοι

Η προσέγγιση του προβλήματος προσδιορισμού λέξεων κλειδιών μέσω της στατιστικής είναι μια αρκετά διαδεδομένη μέθοδος λόγω της απλότητας και της ευκολίας που προσφέρει.
Υπάρχουν διάφοροι τύποι στατιστικών μεθόδων, μεταξύ των οποίων είναι η συχνότητα λεξεων (word frequency), οι συν-εμφανίσεις ή συχνότητα φράσεων (n-grams), η συχνότητα όρου-αντίστροφη συχνότητα εγγράφου (TF-IDF) και η Ραγδαία Αυτόματη εξαγωγή λέξεων-κλειδιών (RAKE- Rapid Automatic Keyword Extraction) .[52]

Αυτές οι προσεγγίσεις δεν απαιτούν δεδομένα εκπαίδευσης προκειμένου να εντοπίσουν τις σημαντικότερες λέξεις-κλειδιά σε ένα κείμενο, γεγονός που τις καθιστά αρκετά εύχρηστες συγκριτικά με άλλες μεθόδους, όπως για παράδειγμα μεθόδους μηχανικής μάθησης. Συγκριτικά με τα μηχανικά μοντέλα μηχανικής μάθησης όμως, υστερούν στο επίπεδο λεπτομερούς ανάλυσης, καθώς μπορεί να παραβλεψουν συναφείς λέξεις που έχουν αναφερθεί μόνο μια φορά αλλά έχουν σημασία. Παρακάτω παρουσιάζονται αναλυτικότερα οι τεχνικές της στατιστικής μεθόδου.

Συχνότητα λέξεων

Η συχνότητα των λέξεων συνίσταται στην καταγραφή των λέξεων και φράσεων που συναντώνται συχνότερα σε ένα κείμενο. Η μέθοδος είναι αποτελεσματική σε μία πληθώρα περιπτώσεων, όπως ο εντοπισμός επαναλαμβανόμενων όρων σε μία σειρά σχολίων από χρήστες ή στην εύρεση των πιο συχνών αιτημάτων στην υποστήριξη πελατών.

Ωστόσο, αυτός ο τρόπος προσέγγισης του ζητήματος αντιμετωπίζει τις λέξεις ως μονάδες χωρίς να δίνει έμφαση στη σύνδεση τους με άλλες λέξεις ή φράσεις, καθώς και με το υπόλοιπο κείμενο, απορρίπτοντας σημαντικές, ίσως, πληροφορίες.[52]

Συχνότητα φράσεων

Σε αντίθεση με τη συχνότητα λέξεων, η συγκεκριμένη μέθοδος, γνωστή και ως n-gram statistics, δίνει έμφαση στον συνδυασμό λέξεων και στην συν-εμφάνιση φράσεων που μπορούν να βοηθήσουν στην κατανόηση της σημασιολογικής δομής του κειμένου. [52]

Οι πιο γνωστές μέθοδοι αυτής της κατηγορίας είναι τα bi-grams, δύο όροι που εμφανίζονται συχνά μαζί, όπως για παράδειγμα «εξυπηρέτηση πελατών» ή «ηλεκτρονικό ταχυδρομείο», καθώς και τα tri-grams, τρείς όροι μαζί, όπως για παράδειγμα «μέσα κοινωνικής δικτύωσης».

Ο όρος συνεμφανίσεις από την άλλη, αναφέρεται σε λέξεις που τείνουν να συνυπάρχουν στο ίδιο κείμενο ,χωρίς αυτό να σημαίνει απαραίτητα ότι συναντώνται παραπλεύρως, αλλά έχουν σημασιολογική εγγύτητα.

TF-IDF

Το TF-IDF αντιπροσωπεύει τη συχνότητα όρου-αντίστροφη συχνότητα εγγράφου. Είναι ένας τύπος που μετράει πόσο σημαντική είναι μία λέξη σε ένα κείμενο μίας ευρύτερης συλλογής εγγράφων.
Αυτή η μέτρηση υπολογίζει τη συχνότητα εμφάνισης μιας λέξης σε ένα κείμενο (συχνότητα όρων) και τη συγκρίνει με την αντίστροφη συχνότητα εμφάνισής της στο σύνολο των κειμένων. Όσο υψηλότερη είναι η βαθμολογία της κάθε λέξης που προκύπτει από την ανωτέρω σύγκριση, τόσο πιο σημαντική είναι η λέξη στο έγγραφο.[52]

Όταν πρόκειται για την εξαγωγή λέξεων-κλειδιών, αυτή η μέτρηση μπορεί να βοηθήσει στον προσδιορισμό συναφών λέξεων, δηλαδή αυτές με την υψηλότερη βαθμολογία, και να τις θεωρήσει λέξεις-κλειδιά.

Ο συγκεκριμένος αλγόριθμος έχει πολλές εφαρμογές ,όπως την επισήμανση των μηνυμάτων που αφορούν την υποστήριξη πελατών, καθώς και την ανάλυση σχολίων χρηστών ή πελατών.
Στην παρούσα διπλωματική, αυτή η μέθοδος χρησιμοποιείται για τον εντοπισμό του προβλήματος/ελαττώματος ενός προϊόντος που πωλείται στο amazon.

Αυτόματη εξαγωγή

Η γρήγορη αυτόματη εξαγωγή, συγκεκριμένα η Rapid Automatic Keyword Extraction (RAKE) τεχνική, είναι μία μέθοδος που χρησιμοποιεί μία λίστα φράσεων - οριοθετών για τον εντοπισμό των λέξεων-κλειδιών σε ένα δείγμα κειμένου.[25]
Το πρώτο πράγμα που εφαρμόζει αυτός ο αλγόριθμος είναι η διάσπαση του κειμένου σε μία λίστα λέξεων και η εξάλειψη των οριοθετών από αυτήν. Οι εναπομείνασες λέξεις είναι γνωστές με τον όρο λέξεις περιεχομένου. Στη συνέχεια ο αλγόριθμος επαναλαμβάνει την παραπάνω διαδικασία διατηρώντας αυτή τη φορά μια λίστα φράσεων. Τέλος από τις δύο παραπάνω λίστες δημιουργείται ένας πίνακας συσχετισμού όλων των λέξεων μεταξύ τους. Πιο απλά, η κάθε λέξη πλέον χαρακτηρίζεται από μια βαθμολογία που μπορεί να υπολογιστεί με το άθροισμα όλων των συσχετίσεων της με τις υπόλοιπες λέξεις στον πίνακα, ή με τη συχνότητα εμφάνισης της στο κείμενο, ή με τη συσχέτιση της προς τη συχνότητα εμφάνισής της. Τέλος, οι λέξεις κλειδιά συλλέγονται με βάση την ανωτέρω βαθμολογία.

Γλωσσικές προσεγγίσεις

Η εξαγωγή λέξεων κλειδιών συχνά χρησιμοποιεί πληροφορίες της γλωσσολογίας των λέξεων και των κειμένων. Συγκεκριμένα, επικεντρώνεται στη μορφολογική ή συντακτική ανάλυση, στη λεξιλογική ανάλυση και στην ανάλυση λόγου. Σε μία γραμματικώς εξαρτώμενη αναπαράσταση των προτάσεων πολλά μέρη του λόγου μπορούν να έχουν μεγαλύτερο βάρος στην επιλογή των τελικών λέξεων κλειδιών, όπως για παράδειγμα τα ουσιαστικά σε βάρος των υπολοίπων. Ορισμένες μέθοδοι πάλι χρησιμοποιούν σημάνσεις λόγου,φράσεις ως επί το πλείστον, που χωρίζουν τον λόγο σε τμήματα, όπως για παράδειγμα εντούτοις ή ακόμη, ή σημασιολογικές πληροφορίες μίας λέξης, όπως για παράδειγμα τις αποχρώσεις της έννοιας αυτής. [25]

Προσεγγίσεις με γράφους

Υπάρχουν διάφοροι τρόποι αναπαράστασης ενός κειμένου σε μορφή γράφου. Ένα πρώτο παράδειγμα είναι οι λέξεις ως μονάδες να θεωρηθούν ακμές και να συνδεθούν μονόδρομα με κατευθυνόμενες ακμές.[25] Άλλες αναπαραστάσεις μπορούν να χρησιμοποιήσουν αμφίδρομες ή μη κατευθυνόμενες συνδέσεις για την αναπαράσταση των λεξικών συσχετίσεων. Η βασική όμως ιδέα σε μια προσέγγιση με γράφους είναι σε κάθε περίπτωση η ίδια, ο καθορισμός του πόσο σημαντική είναι μία ακμή βασισμένη σε κριτήρια που λαμβάνουν υπ' όψιν κάποιες πληροφορίες από τη δομή του γράφου και, εν τέλει, η εξαγωγή των σημαντικότερων λέξεων.

Βαθμός ακμής

Ο βαθμός ακμής ενός κατευθυνόμενου γράφου υπολογίζεται με διάφορες μεθόδους. Μία από αυτές είναι το άθροισμα όλων των κορυφών που ξεκινούν ή καταλήγουν σε μία ακμή προς τον μέγιστο πιθανό βαθμό, δηλαδή το άθροισμα όλων των κορυφών πλήν μίας. Μία ακόμη θα μπορούσε να είναι η γειτονιά των ακμών, δηλαδή το άθροισμα όλων των ακμών που συνδέονται με αυτήν.

Με οποιονδήποτε τρόπο και αν υπολογιστεί η εκάστοτε βαθμολογία των ακμών του γράφου, αυτή είναι που θα καθορίσει και την τελική επιλογή των λέξεων-κλειδιών.

Τεχνικές μηχανικής μάθησης

Τα συστήματα βασισμένα στη μηχανική μάθηση μπορούν να χρησιμοποιηθούν για πολλές εργασίες ανάλυσης κειμένου, συμπεριλαμβανομένης της εξαγωγής λέξεων-κλειδιών. Για να επεξεργαστούν μη δομημένα δεδομένα κειμένου, τα εν λόγω συστήματα πρέπει να τα μετατρέψουν σε κάτι που μπορούν να αντιληφθούν, δηλαδή σε ένα διάνυσμα. Μετατρέποντας λοιπόν τα δεδομένα σε φορείς με διάφορα χαρακτηριστικά, αντιπροσωπευτικά του κειμένου, μπορούν να προχωρήσουν στην περαιτέρω επεξεργασία τους. Υπάρχουν διάφοροι αλγόριθμοι μηχανικής μάθησης που μπορούν να χρησιμοποιηθούν για την εξαγωγή των πιο σχετικών λέξεων-κλειδιών σε ένα κείμενο, όπως τα Support Vector Machines (SVM) και αλγόριθμοι βαθείας μηχανικής μάθησης (deep learning algorithms).[25][52]

Μία δημοφιλής τεχνική αυτής της κατηγορίας είναι τα Υποχρεωτικά Τυχαία Πεδία (conditional random fileds-CRF), μία στατιστική προσέγγιση που μαθαίνει πρότυπα με τη στάθμιση διαφορετικών χαρακτηριστικών σε μία

ακολουθία λέξεων. Με αυτόν τον τρόπο εξετάζει το πλαίσιο του κειμένου και τις σχέσεις μεταξύ διαφόρων μεταβλητών για να κάνει τις τελικές προβλέψεις.

Υβριδικές μέθοδοι

Με σκοπό τη βελτίωση των αποτελεσμάτων έχουν αναπτυχθεί και κάποιες υβριδικές μέθοδοι, οι οποίες συνδυάζουν δύο ή περισσότερες από τις παραπάνω προσεγγίσεις για την εξαγωγή των πιο σχετικών λέξεων.

Η καλύτερη μέθοδος για ένα πρόβλημα εξαρτάται από το χαρακτηριστικά που πρέπει να έχει το μοντέλο, τον τύπο δεδομένων τα οποία διαχειρίζεται και τα αποτελέσματα που επιθυμούμε.

3.6 Αλγόριθμοι μηχανικής μάθησης

Στο σημείο αυτό θα παρουσιάσουμε και θα αναλύσουμε το βασικότερο κομμάτι μιας εφαρμογής μηχανικής μάθησης που δεν είναι άλλο από τον αλγόριθμο που θα εφαρμοστεί για να πραγματοποιήσει εκτιμήσεις. Υπάρχουν πολλοί και διαφορετικοί αλγόριθμοι, που αναλόγα με τα δεδομένα που διαθέτουμε για να δωθούν ως είσοδος, αλλα και τα αποτελέσματα που στοχεύουμε να πάρουμε, ταιριάζουν περισσότερο ή λιγότερο και αποδίδουν με μεγαλύτερη ή μικρότερη ακρίβεια αντίστοιχα. Στη συνέχεια περιγράφονται αναλυτικά ορισμένοι από τους βασικότερους αλγορίθμους που χρησιμοποιούνται στη μηχανική μάθηση και χρησιμοποιήθηκαν και στο πλαίσια αυτής της διπλωματικής.

3.6.1 Linear Regression

Linear regression ή αλλιώς γραμμική παλινδρόμηση [46] στη στατιστική είναι μια προσέγγιση για τη μοντελοποίηση της σχέσης μεταξύ μιας βαθμωτής

εξαρτημένης μεταβλητής Y και μίας ή περισσοτέρων επεξηγηματικών μεταβλητών (ή ανεξάρτητων μεταβλητών) που συμβολίζονται με X_i. Η περίπτωση μιας επεξηγηματικής μεταβλητής ονομάζεται απλή γραμμική παλινδρόμηση. Για περισσότερες από μία επεξηγηματικές μεταβλητές, η διαδικασία ονομάζεται πολλαπλή γραμμική παλινδρόμηση. Η συγκεκριμένη προσέγγιση είναι αρκετά απλή αλλά πολύ σημαντική στην κατανόηση πιο σύνθετων μόντελων. Αρκετά από τα πιο περίπλοκα στατιστικά μοντέλα αποτελούν επεκτάσεις της γραμμικής παλινδρόμησης.
Ας υποθέσουμε την απλούστερη περίπτωση μίας ανεξάρτητης και μίας εξαρτημένης μεταβλητής. Η εξίσωση της γραμμικής παλινδρόμησης έχει ως εξής:

$$Y = \beta_0 + \beta_1 X$$

όπου βο, β1 είναι οι συντελεστές της εξίσωσης.
Στην πραγματικότητα, αυτούς τους συντελεστές προσπαθούμε να προσδιορίσουμε στο μοντέλο μας προκειμένου να μπορούμε να κάνουμε εκτιμήσεις.
Για τον υπολογισμό τους χρησιμοποιούμε τη μέθοδο των ελαχίστων τετραγώνων. Αυτή η προσέγγιση είναι η απλούστερη και επομένως η εκτίμηση είναι η πιο κοινή. Είναι εννοιολογικά και υπολογιστικά απλή. Η μέθοδος των ελαχίστων τετραγώνων ελαχιστοποιεί το άθροισμα των τετραγώνων των σφαλμάτων, και οδηγεί σε μια έκφραση κλειστής μορφής για την εκτιμώμενη αξία της άγνωστης παραμέτρου β. Το σφάλμα υπολογίζεται από την αφαίρεση της εκτιμώμενης τιμής από την πραγματική ως εξής:

$$e_i = y_i - \hat{y}_i$$

Στη συνέχεια, επειδή οι τιμές που προβλέπουμε μπορουν να είναι είτε μεγαλύτερες είτε μικρότερες από τις πραγματικές και έτσι το σφάλμα να έχει θετική ή αρνητική τιμή, το υψώνουμε στο τετράγωνο. Αν δεν το κάναμε αυτό, θα ήταν πιθανό το άθροισμα των σφαλμάτων να μειωνόταν, όμως όχι λόγω της καλύτερης απόδοσης του μοντέλου, αλλά λογω των αρνητικών τιμών των σφαλμάτων. Επίσης με το να υψώνουμε τα σφάλματα στο τετράγωνο δίνουμε μεγαλυτερο βάρος στις μεγάλες αποκλίσεις και έτσι ανάλογα με την τιμή του αθροίσματος (αν θα είναι μικρή ή μεγάλη) μπορούμε να έχουμε μια πολύ καλή εικόνα της πραγματικής απόδοσης του μοντέλου.
Τελικά οι συντελεστές υπολογίζονται από τις σχέσεις:

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

Όπου $\vec{x}$ και $\vec{y}$ είναι οι μέσες τιμές των αντίστοιχων μεταβλητών.

Αφού λοιπόν υπολογίσαμε τις τιμές για τους συντελεστές της εξίσωσης, πρέπει να μπορούμε να αποφανθούμε για την απόδοση του αλγορίθμου. Για να το καταφέρουμε αυτό θα χρειαστεί να υπολογίσουμε ορισμένα στατιστικά, μεγέθη τα οποία παρουσιάζονται παρακάτω:

- Τυπική απόκλιση

Στη στατιστική, η τυπική απόκλιση (standard deviation, SD) είναι ένα μέτρο που χρησιμοποιείται για να υπολογιστεί το ποσό της μεταβολής ή της διασποράς ενός συνόλου τιμών δεδομένων. Η τυπική απόκλιση μιας τυχαίας μεταβλητής, ενός στατιστικού πληθυσμού ή ενός συνόλου δεδομένων είναι η τετραγωνική ρίζα της διακύμανσης της. Μια χαμηλή τυπική απόκλιση υποδηλώνει ότι τα σημεία των δεδομένων τείνουν να είναι κοντά στο μέσο όρο (μέση ή αναμενόμενη τιμή) του συνόλου, ενώ μία υψηλή τυπική απόκλιση υποδεικνύει ότι τα στοιχεία απλώνονται σε ένα ευρύτερο φάσμα των τιμών.
Εκτός από την έκφραση της μεταβλητότητας του πληθυσμού, η τυπική απόκλιση συνήθως χρησιμοποιείται για τη μέτρηση της εμπιστοσύνης στα στατιστικά συμπεράσματα.
Η τυπική απόκλιση υπολογίζεται ως εξής:

$$\sigma_x = \sqrt{\frac{\sum_k (x_k - \bar{x})^2}{N-1}}$$

Η ποσότητα σ^2, δηλαδή το τετράγωνο της τυπικής απόκλισης, ονομάζεται διασπορά ή διακύμανση.

- Τυπικό σφάλμα

Ο ορισμός και η κατανόηση της τυπικής απόκλισης είναι σημαντικά για τον ορισμό του μεγέθους που ονομάζεται τυπικό σφάλμα ή τυπικό σφάλμα μέσου. Επειδή, σε πολλές περιπτώσεις, όταν μετρούμε πολλές φορές στις ίδιες συνθήκες την ίδια ποσότητα, καταλήγουμε σε διαφορετικά αποτελέσματα, βρίσκουμε τη μέση τιμή και το απόλυτο σφάλμα της μέσης τιμής (ή τυπική απόκλιση της μέσης τιμής). Εάν σε ένα πείραμα η μέτρηση του μεγέθους x

επαναληφθεί N φορές, και οι μετρούμενες τιμές είναι x_1, x_2, x_3, …x_N,τότε ως πραγματική θεωρούμε την μέση τιμή

$$\bar{x} = \frac{x_1 + x_2 + \ldots + x_k + \ldots x_N}{N} = \frac{\sum_{k=1}^{N} x_k}{N}$$

Εάν τα σφάλματα των παραπάνω μετρήσεων είναι τυχαία θα διαφέρουν ως προς το πρόσημο και ως προς το μέγεθος. Έτσι, στον υπολογισμό της μέσης τιμής, κάποια από τα τυχαία σφάλματα αλληλοαναιρούνται στο άθροισμα. Έτσι μπορεί να υπολογιστεί η απόκλιση των μετρήσεων από τη μέση τιμή. Το απόλυτο σφάλμα της μέσης τιμής μπορεί να χρησιμοποιηθεί για να εκφράσει τη βεβαιότητα των N μετρήσεων για τη μέση τιμή μιας μεταβλητής έστω y ως εξής:

$$RSE = \sqrt{\frac{1}{n-2} RSS} = \sqrt{\frac{1}{n-2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Όπου RSS (residual sum of squares) ή αλλιώς SSE (error sum of squares) είναι το άθροισμα των τετραγώνων των σφαλμάτων

$$SSE = \sum_{i=1}^{\nu} (y_i - \hat{y}_i)^2 \quad \text{(error sum of squares).}$$

- Συντελεστής προσδιορισμού – R^2

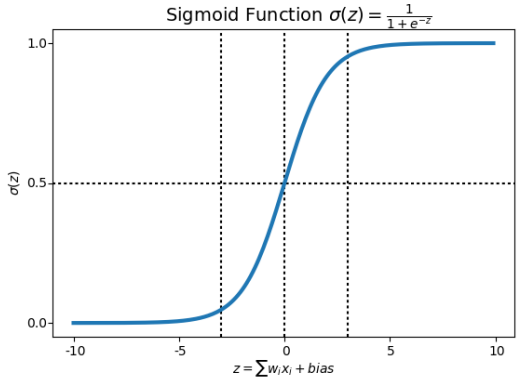
Υπάρχουν πολλοί δείκτες που χρησιμοποιούνται για να εκφράσουν την συσχέτιση δυο τυχαιών μεταβλητών δηλαδή τον βαθμό στον οποίο οι τιμές της μιας καθορίζουν τις τιμές της άλλης. Ένας από τους πιο συνηθισμένους σε τέτοιες εφαρμογές είναι να υπολογίσουμε την ποσότητα $\mathbf{r^2}$ που λέγεται και συντελεστής προσδιορισμού (**coefficient of determination**) (εκφράζεται συνήθως σε ποσοστό %, $100r^2$). Ο συντελεστής προσδιορισμού δίνει το ποσοστό μεταβλητότητας των τιμών της Y που υπολογίζεται από τη X (κι αντίστροφα) κι είναι ένας χρήσιμος τρόπος να συνοψίσουμε τη συσχέτιση δύο τυχαίων μεταβλητών.
Ουσιαστικά μας δείχνει το ποσοστό της συνολικής διασποράς του Y που μπορεί να αποδωθεί στη γνώση του X, ή με απλούστερα λόγια, αν το χαρακτηριστικό X μπορεί να προβλέψει σωστά την τιμή του Y τότε η τιμή του R^2 θα είναι κοντά στην μονάδα αλλιώς θα προσεγγίζει το 0. Η σχέση από την οποία υπολογίζεται η τιμή του είναι:

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}, \quad TSS = \sum (y_i - \hat{y}_i)^2$$

3.6.2 Logistic Regression

Logistic regression είναι ένα είδος αλγορίθμου μηχανικής μάθησης που ενώ περιλαμβάνει τη λέξη regression ή αλλιώς παλινδρόμηση δεν ανήκει σε αυτή την κατηγορία αλγορίθμων (όπως linear regression) που περιγράψαμε παραπάνω αλλά ανήκει στους αλγορίθμους κατηγοριοποίησης (classification). [26][46][32]. Ουσιαστικά, το συγκεκριμένο μοντέλο στηρίζεται στη γραμμική παλινδρόμηση που αναφέραμε, χρησιμοποιώντας όμως μια πιο περίπλοκη συνάρτηση από την απλή γραμμική σχέση δύο ή περισσότερων μεταβλητών. Αυτή είναι η σιγμοειδής συνάρτηση. Με βάση αυτήν, προσπαθεί να αντιστοιχίσει κάθε προβλεπόμενη από την παλινδρόμηση τιμή σε μια πιθανότητα, υπακούοντας έτσι στην "απαίτηση" του logistic regression αλγορίθμου να είναι όλες οι τελικές τιμές στο διάστημα 0 εως 1. Έτσι λοιπόν κάθε πραγματικός αριθμός που προκύπτει ως πρόβλεψη αντιστοιχίζεται σε μια πιθανότητα από τον τύπο της σιγμοειδούς:

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



Σιγμοειδής συνάρτηση

Η βασική λογική του συγκεκριμένου αλγορίθμου είναι ότι, αφου δώσουμε σε μια απλή συνάρτηση όπως αυτή της γραμμικής παλινδρόμησης ως είσοδο διάφορες τιμές του X και λάβουμε ως έξοδο ένα σύνολο πραγματικών

αριθμών για τη μεταβλητή Y, αυτοί θα δωθούν ως είσοδος στη σιγμοειδή και τελικά θα έχουμε σαν αποτέλεσμα ένα σύνολο πιθανοτήτων. Ορίζουμε την τιμή του κατωφλίου απόφασης στον αλγόριθμο με βάση την οποία θα κατηγοριοποιεί τα αποτελέσματα στις αντίστοιχες κλάσεις. Για παράδειγμα αν ορίσουμε ως τιμή κατωφλίου το 0.5, οι τιμές μεγαλύτερες αυτής της τιμής θα αποδίδονται στην κλάση 1 ενώ οι υπόλοιπες στην κλάση 0.
Αφού λοιπόν προσδιορίσουμε τις λεπτομέρειες του μοντέλου μας χρειαζόμαστε όπως και πριν κάποια μεγέθη για να αξιολογήσουμε την απόδοση του. Όπως στην περίπτωση της γραμμικής παλινδρόμησης αναφέραμε ως κριτήρια απόδοσης τις τιμές του τυπικού σφάλματος (RSE) και του συντελεστή προσδιορισμού (R^2), έτσι και εδώ θα παρουσιάσουμε την εξής συνάρτηση:

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

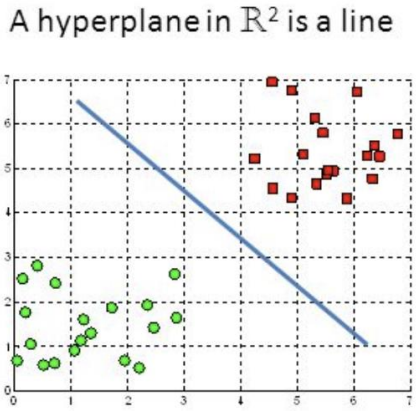
ή συνολικά η συνάρτηση κόστους του logistic regression:

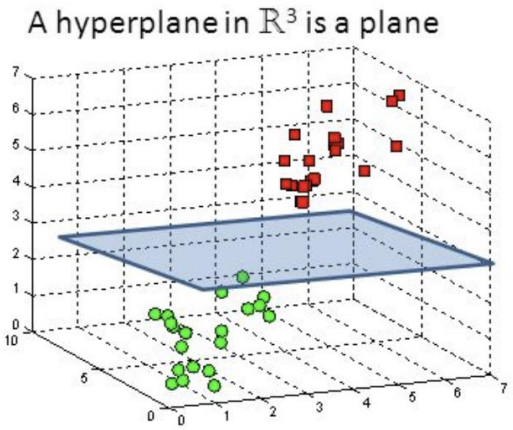
$$J(\theta) = -\frac{1}{m} \sum \left[y^{(i)} \log(h\theta(x(i))) + \left(1 - y^{(i)}\right) \log(1 - h\theta(x(i))) \right]$$

Στόχος για την καλύτερη δυνατή απόδοση του συγκεκριμένου μοντέλου είναι η ελαχιστοποίηση της συγκεκριμένης συνάρτησης. Αυτό επιτυγχάνεται παραγωγίζοντας την παραπάνω σχέση για κάθε μεταβλητή θ, x.

3.6.3 Support Vector Machine

Ένας πολύ σημαντικός και πολύ συχνά χρησιμοποιούμενος αλγόριθμος είναι ο Support Vector Machine (SVM) [11][46][32]. Είναι ένας σχετικά απλός αλγόριθμος, ο οποίος πολλές φορές προτιμάται από άλλους, καθώς προσφέρει καλύτερα αποτελέσματα με λιγότερους πόρους. Επίσης, ένα ακόμα πλεονέκτημά του είναι ότι μπορεί να χρησιμοποιηθεί για την επίλυση τόσο παλινδρομικών (regression) προβλημάτων όσο και προβλημάτων κατηγοριοποίησης (classification).
Στόχος αυτού του αλγορίθμου είναι να καταφέρει να σχεδιάσει σε έναν χώρο N διαστάσεων ένα σύνορο (hyperplane) που θα ξεχωρίζει διακριτά τις παρατηρήσεις στις δύο κλάσεις. Η μορφή που θα έχει το συγκεκριμένο σύνορο εξαρτάται από το πλήθος των μεταβλητών που υπάρχουν στην εξίσωση. Για παράδειγμα σε μια εξίσωση δυο μεταβητών, δηλαδή για N=2, το σύνορο αυτό θα είναι μια γραμμή, ενώ στον τρισδιάστατο χώρο, για N=3, θα είναι ένα επίπεδο.





Αφού λοιπόν, κατανοήσαμε την έννοια του hyperplane και τη σχέση του με τις μεταβλητές της εξίσωσης είναι πολύ σημαντικό να παρουσιάσουμε το πώς γίνεται η επιλογή αυτού. Για να διαχωρίσουμε σημεία (τα οποία αντιστοιχούν σε παρατηρήσεις) σε ένα χώρο N διαστάσεων υπάρχουν άπειρα hyperplanes. Ο συγκεκριμένος αλγόριθμος, για να επιλέξει το σύνορο για τον διαχωρισμό, χρησιμοποιεί ως κριτήριο τη μεγιστοποίηση της απόστασης του συνόρου από τις παρατηρήσεις και των δύο κλάσεων. Αναλυτικότερα, από τις συνολικές παρατηρήσεις ο αλγόριθμος επιλέγει τα σημεία που βρίσκονται πιο κοντά στο σύνορο και προσπαθεί να μεγιστοποιήσει αυτές τις αποστάσεις για να καθορίσει την θέση και τον προσανατολισμό του hyperplane.
Όπως αναφέραμε στην περίπτωση του αλγορίθμου για logistic regression οι τιμές που προκύπτουν από την εξίσωση της γραμμικής παλινδρόμησης δίνονται ως είσοδος στη σιγμοειδή συνάρτηση για να προκύψουν οι πιθανότητες στο διάστημα [0,1]. Αντίστοιχα, στην περίπτωση του SVM αν η τιμή που θα δώσει σαν αποτέλεσμα η γραμμική συνάρτηση είναι μεγαλύτερη του 1 τότε την κατηγοριοποιούμε στη μια κλάση αλλιώς αν αυτή η τιμή είναι μικρότερη του -1 περνάει στη κατηγορία της άλλης κλάσης. Με αυτόν τον τρόπο λοιπόν το περιθώριο απόφασης αυξήθηκε από το [0,1] στο διάστημα [-1,1], αυξάνοντας παράλληλα και την βεβαιότητα με την οποία οι παρατηρήσεις αναθέτονται στις επιμέρους κλάσεις.
Στον SVM αλγόριθμο στοχεύουμε να μεγιστοποιήσουμε την απόσταση των παρατηρήσεων από το διαχωριστικό σύνορο. Για να το πετύχουμε αυτό χρησιμοποιούμε την συνάρτηση απώλειας hinge (hinge loss function) :

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

Η απώλεια που υπολογίζεται από την συνάρτηση είναι ίση με το μηδέν σε περίπτωση που η πραγματική τιμή και η πρόβλεψη έχουν το ίδιο πρόσημο,ενώ σε αντίθετη περίπτωση η τιμή της υπολογίζεται από τον δεύτερο κλάδο της συνάρτησης. Επίσης, προσθέτουμε μια ρυθμιστική παράμετρο , C>0, που ελέγχει την ισορροπία μεταξύ των δεδομένων προσαρμογής και τον όρο $1/2\|w\|^2$ που αντιστοιχεί στην ποινή ενώ ψ_{SVM} είναι η συνάρτηση απώλειας των SVM (hinge loss function) και ετσι τελικά έχουμε:

$$\min_{b,w}\left(\frac{1}{2}\|w\|^2 + c\sum_{i=1}^{n}\psi_{SVM}\left(y_i\left(f(x_i)\right)\right)\right)$$

Εδώ αξίζει να τονίσουμε ότι η ρυθμιστική παράμετρος εξασφαλίζει ότι το μοντέλο έχει μια καλή προσαρμογή στα δεδομένα εκπαίδευσης, ο όρος ποινής αποφεύγει την υπερπροσαρμογή (overfitting) του μοντέλου που προκύπτει και η «hinge loss» συνάρτηση των SVMs χρησιμοποιείται για τη μεγιστοποίηση του περιθωρίου ταξινόμησης.
Έχοντας λοιπόν τη συνάρτηση απώλειας, αυτό που επιδιώκουμε είναι η εύρεση των $(\boldsymbol{w},b)$ που την ελαχιστοποιούν. Παραγωγίζοντας και τους δύο όρους της παραπάνω εξίσωσης υπολογίζουμε τους συντελεστές της ελαχιστοποιημένης εξίσωσης:

$$\frac{\delta}{\delta w_k}\lambda \parallel w \parallel^2 \; = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k}\left(1 - y_i\langle x_i, w\rangle\right)_+ = \begin{cases} 0, & \text{if } y_i\langle x_i, w\rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Στην περίπτωση που η πρόβλεψη για την κλάση είναι σωστή η μεταβλητή **w** υπολογίζεται μόνο από τον όρο της ποινής, ενώ στη περίπτωση λανθασμένης πρόβλεψης θα πρέπει να λάβουμε υπόψη μας για τον υπολογισμό και την συνάρτηση απώλειας. Οι δύο περιπτώσεις αυτές φαίνονται ακολούθως:

$$w = w - \alpha \cdot (2\lambda w)$$

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

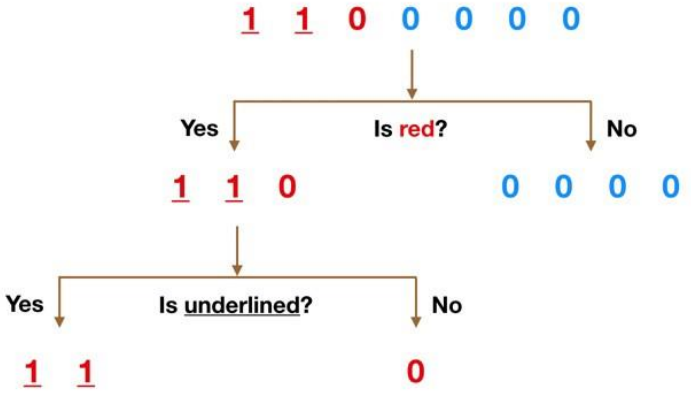
3.6.4 Random Forest

Ενας πολύ σημαντικός αλγόριθμος κατηγοριοποίησης είναι ο Random Forest [6]. Για να γίναι κατανοητός ο τρόπος λειτουργίας του θα πρέπει πρώτα να παρουσιάσουμε τη δομή ενός βασικού δέντρου απόφασης και έπειτα πως πολλά ξεχωριστά δέντρα απόφασης συνδυάζονται για να δημιουργήσουν ένα τυχαίο δάσος (random forest).

Ας υποθέσουμε ότι έχουμε ένα σύνολο δεδομένων που έχει κάποια χαρακτηριστικά και θέλουμε να διαχωρίσουμε το σύνολο αυτό με βάση ένα ή περισσότερα από αυτά τα χαρακτηριστικά. Έτσι λοιπόν, ο πρώτος διαχωρισμός γίνεται δυαδικά στα δεδομένα που έχουν μια τιμή για ένα συγκεκριμένο χαρακτηριστικό και σε αυτά που έχουν μια άλλη τιμή. Αυτή η διαδικασία επαναλαμβάνεται για όσα χαρακτηριστικά θέλουμε μέχρι να μείνουν στο ένα σύνολο τα δεδομένα που θέλουμε. Το παράδειγμα της διπλανής εικόνας θα κάνει ακόμα περισσότερο κατανοητή τη λειτουργία του δέντρου απόφασης. Το αρχικό μας σύνολο περιλαμβάνει 0 και 1 τα όποια διαθέτουν τα χαρακτηριστικά του χρώματος (μπλε, κόκκινο) και της υπογράμμισης (υπογραμμισμένα ή όχι). Σε κάθε βήμα λοιπόν προσπαθούμε να διαχωρίσουμε τα δεδομένα που διαθέτουν τα χαρακτηριστικά που θέλουμε από τα υπόλοιπα. Έστω πως στη συγκεκριμένη περίπτωση επιθυμούμε να κρατήσουμε τα δεδομένα που είναι κόκκινα και υπογραμμισμένα. Τότε τα βήματα που θα ακολουθήσουμε είναι αυτά που φαίνονται δίπλα. Όπως γίνεται κατανοητό λοιπόν ένας κόμβος στο δέντρο είναι το σημείο που τα δεδομένα διαχωρίζονται σε όσα διαθέτουν κάποιο χαρακτηριστικό και όσα όχι. Με αυτή τη διαδικασία επαναληπτικά φτάνουμε στον τελικό διαχωρισμό.

Το μοντέλο random forest λοιπόν αποτελείται από ένα πολύ μεγάλο πλήθος από δέντρα απόφασης όπως περιγράψαμε παραπάνω. Το κάθε ένα από αυτά λειτουργεί αυτόνομα και με την παραπάνω λογική καταλήγει σε μια πρόβλεψη για την κλάση της εκάστοτε παρατήρησης. Στη συνέχεια όλες αυτές οι παρατηρήσεις συγκεντρώνονται συνολικά και η κλάση που έχουν προβλεψει τα περισσότερα αυτόνομα δέντρα είναι η τελική εκτίμηση του μοντέλου μας.

Η θεωρία πάνω στην οποία στηρίζεται, τόσο η λειτουργια, όσο και η πολύ καλή απόδοση του συγκεκριμένου αλγορίθμου, είναι ότι ο συμψηφισμός ενός πολύ μεγάλου αριθμού ανεξάρτητων μεταξύ τους μοντέλων τα οποία λειτουργουν ως ένα είδος επιτροπής απόφασης υπερτερεί σε απόδοση από κάθε ένα ατομικό μοντέλου που



απαρτίζουν το σύνολο. Το κλειδί σε αυτό είναι η πολύ μικρή συσχέτιση που έχουν τα μοντέλα, στην περίπτωση μας τα επιμέρους δέντρα, μεταξύ τους. Έτσι, ενώ κάποια από τα δέντρα σίγουρα θα προβλεψουν την κλάση λανθασμένα, θα υπάρχουν πολλά άλλα που θα πραγματοποιήσουν σωστή πρόβλεψη και με αυτόν τον τρόπο το συνολικό μοντέλο θα κινείται προς την σωστή κατεύθυνση. Ουσιαστικά λοιπόν αυτό που συμβαίνει είναι ότι τα δέντρα προστατεύουν το ένα το άλλο από τα ατομικά τους σφάλματα.
Οι προϋποθέσεις για την καλή απόδοση του random forest μοντέλου είναι:

- Τα δεδομένα να διαθέτουν τουλάχιστον κάποια κοινά χαρακτηριστικά έτσι ώστε να μπορεί ο αλγόριθμος να εκπαιδευτεί και να "μάθει" από αυτά
- Τα αυτόνομα δέντρα να είναι ανεξάρτητα, δηλαδή να έχουν πολύ μικρή συσχέτιση μεταξύ τους ώστε και οι εκτιμήσεις (ή τα πιθανά λάθη) να είναι κι αυτά ανεξάρτητα
- Το πλήθος των ανεξάρτητων δέντρων απόφασης που θα πραγματοποιήσουν τις προβλέψεις να είναι εξαιρετικά μεγάλο καθώς όσο αυξάνεται αυτό αυξάνεται και η πιθανότητα η τελική πρόβλεψη του συνολικού μοντέλου να είναι σωστή.

Υπάρχουν δυο τρόποι που μπορούμε να εξασφαλίσουμε την ανεξαρτησία των δέντρων ώστε να κάνουμε αποδοτικότερο το μοντέλο μας:

- Bagging (Bootstrap Aggregation), μέσω του οποίου ουσιαστικά εκμεταλλευόμαστε την ευαισθησία των δέντρων απόφασης στα δεδομένα πάνω στα οποία εκπαιδεύονται. Έτσι δίνουμε στα μοντέλα μας ένα δείγμα από τα συνολικά δεδομένα χωρίς ωστόσο να μεταβάλλουμε το μέγεθος των δεδομένων.
- Τυχαια επιλογή χαρακτηριστικών (feature randomness). Σε ένα απλό δέντρο απόφασης λαμβάνουμε υπόψη μας όλα τα χαρακτηριστικά πριν αποφασίσουμε ποιο θα χρησιμοποιήσουμε για τον διαχωρισμό. Αντίθετα, με αυτόν τον τρόπο ένα δέντρο του random forest μοντέλου θα μπορεί να επιλέξει μόνο από ένα τυχαίο υποσύνολο των χαρακτηριστικών αυξάνοντας έτσι τη διαφοροποίηση ανάμεσα στα δέντρα.

3.6.5 Naïve Bayes

Το θεώρημα του Bayes είναι ένα πολύ σημαντικό θεώρημα για την θεωρία των πιθανοτήτων και για τη στατιστική. Πάνω σε αυτό βασίζεται ο αλγόριθμος κατηγοριοποίησης Naïve Bayes [48][19][32], ο οποίος είναι ένας από τους πιο συχνά χρησιμοποιούμενος για την επίλυση προβλημάτων αυτής της κατηγορίας. Για να μπορέσουμε λοιπόν να προχωρήσουμε στην ανάλυση αυτού του μοντέλου θα παρουσιάσουμε πρώτα το εν λόγω θεώρημα:

Έστω ότι έχουμε δύο γεγονότα A,B

- P(A) και P(B) είναι οι πιθανότητες των A και B που είναι ανεξάρτητα μεταξύ τους.
  Ανεξάρτητα είναι δυο ενδεχόμενα όταν ισχύει P(AB)=P(A)P(B). Πρακτικά αυτό σημαίνει ότι η πραγματοποίηση του ενός δεν επηρεάζει καθόλου την πιθανότητα εμφάνισης του άλλου.
- P(A | B), η υπό συνθήκη πιθανότητα είναι η πιθανότητα του A δεδομένου ότι ισχύει το B
- P(B | A) είναι η πιθανότητα του B υπό την προϋπόθεση ότι ισχύει το A.

Έτσι λοιπόν το θεώρημα Bayes ορίζεται μαθηματικά από την ακόλουθη σχέση:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

και μας δίνει τη δεσμευμένη ή αλλιώς την υπο συνθήκη πιθανότητα εμφάνισης του ενδεχομένου A υπο την προϋπόθεση ότι έχει ήδη συμβεί το B.
Με χρήση του ανωτέρω θεωρήματος μπορούμε να εκτιμήσουμε την πιθανότητα μια παρατήρηση να ανήκει σε κάθε κλάση.
Ας υποθέσουμε πως έχουμε ένα σύνολο δεδομένων με N χαρακτηριστικά, τα οποία μπορούν να λαβουν μόνο δύο τιμές, και πως η τελική πρόβλεψη είναι δυαδικής μορφής. Θα πρέπει να υπολογίσουμε την πιθανότητα P(A | B) για όλες τις τιμές του παραδείγματος, δηλαδή θα πρέπει να πραγματοποιήσουμε $2*(2^{N-1})$ υπολογισμούς για να εκπαιδεύσουμε το μοντέλο μας. Αυτό σημαίνει πως αν έχουμε για παράδειγμα ένα σύνολο δεδομένων με 30 δυαδικά χαρακτηριστικά θα πρέπει να υπολογίσουμε πάνω από 3 δισεκατομμύρια παραμέτρους, κάτι το οποίο δεν είναι εφικτό σε προβλήματα του πραγματικού κόσμου.
Γι' αυτό η πολυπλοκότητα στον υπολογισμό της πιθανότητας πρέπει να μειωθεί. Αυτό το επιτυγχάνει ο αλγόριθμος Naïve Bayes υποθέτοντας πως τα χαρακτηριστικά που δίνονται ως είσοδος, έστω $X_1,...X_N$, είναι ασυμβίβαστα μεταξύ τους. Ασυμβίβαστα λέγονται δύο ενδεχόμενα για τα οποία ισχύει $A \cap B = Ø$. Τα ασυμβίβαστα ενδεχόμενα λέγονται και ξένα μεταξύ τους ή αμοιβαίως αποκλειόμενα και η πραγματοποίηση του ενός απόκλείει την πραγματοποίηση του άλλου, δηλαδή δεν μπορούν να συμβούν ταυτόχρονα. Επίσης ας υποθέσουμε πως το ενδεχόμενο Y είναι η τελική πρόβλεψη δεδομένων των χαρακτηριστικών που δώσαμε για είσοδο. Αυτό το ενδεχόμενο μπορεί να συμβεί ταυτόχρονα με καθένα από τα ασυμβίβαστα χαρακτηριστικά εισόδου. Έτσι με αυτές τις υποθέσεις ο αλγόριθμος γίνεται "αφελής" (naïve) και η πολυπλοκότητα μειώνεται δραστικά σε 2*N.
Ο παρονομαστής της σχέσης της δεσμευμένης πιθανότητας τωρα δίνεται ως εξής:

$$P(X_1 \ldots X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

Και ονομάζεται τύπος της ολικής πιθανότητας.

Αυτό που μας ενδιαφέρει τελικά είναι να υπολογίσουμε την πιθανότητα P(A | B) για τις διάφορε τιμές του B που συμβολίζει τα χαρακτηριστικά των δεδομένων εισόδου και με δεδομένες τις πιθανότητες P(B | A), P(A) από τα δεδομένα προς εκπαίδευση. Αν υποθέσουμε ακόμα πως η μεταβλητή εξόδου μας, η A στην περίπτωσή μας, είναι ομοιόμορφα κατανεμημένη δηλαδή μπορεί να πάρει οποιαδήποτε τιμή στο διάστημα [0,1] απλοποιούμε κι άλλο το πρόβλημα καθώς αυτή μετατρέπεται σε συνεχή μεταβλητή με τιμή ίση με το αντίστροφο κλάσμα του αριθμού των πιθανών κλάσεων εξόδου. Έτσι ουσιαστικά για να μπορέσουμε να μεγιστοποιήσουμε την δεσμευμένη πιθανότητα πρέπει πια να μεγιστοποιήσουμε μόνο τον παρονομαστή του τύπου Bayes ή αλλιώς τον τύπο της ολικής πιθανότητας.
Ο αλγόριθμος διαθέτει αρκετές παραλλαγές που μπορούν να χρησιμοποιηθούν για την επίλυση διαφορετικών προβλημάτων. Για συνεχείς κατανομές ο **Gaussian naïve bayes** αποτελεί τον καταλληλότερο αλγόριθμο, ενώ για διακριτά χαρακτηριστικά οι **multinomial** και **Bernouli naïve bayes** αποδίδουν καλύτερα. Όλες οι παραλλαγες του συγκεκριμένου αλγορίθμου παρά τις υποθέσεις λειτουργούν αποτελεσματικότερα σε πιο σύνθετα προβλήματα καθώς ένα από τα πλεονεκτήματα τους είναι πως χρειάζονται μικρό μέγεθος δεδομένων εισόδου για να υπολογίσουν τις απαιτούμενες παραμέτρους. Το συγκεκριμένο είδος μοντέλου αποτελεί την ιδανική επιλογή για προβλήματα κατηγοριοποίησης κειμένου (text categorization).

Κεφάλαιο 4 Ανάλυση – σχεδίαση

4.1 Προσδιορισμός στόχου και ανάλυση απαιτήσεων

Οι διαδικτυακές αγορές έχουν αναπτυχθεί ταχύτατα τις τελευταίες δεκαετίες με αποτέλεσμα πολλές εταιρείες ηλεκτρονικού εμπορίου να έχουν δημιουργηθεί για να καλύψουν την αυξανόμενη ζήτηση. Αυτό έχει ως αποτέλεσμα ένα συγκεκριμένο προϊόν να μπορεί να αγοραστεί μέσω πολλών ηλεκτρονικών αγορών και η τιμή του πιθανόν να διαφέρει από ιστότοπο σε ιστότοπο. Εφόσον οι καταναλωτές συνήθως επιθυμούν την καλύτερη δυνατή ποιότητα στη χαμηλότερη τιμή, αλλά συγχρόνως δεν μπορούν χειροκίνητα να κάνουν μία τέτοιου είδους εκτεταμένη αναζήτηση, τα σχόλια των υπόλοιπων χρηστών έχουν αποδειχθεί ένας αρκετά αξιόπιστος τρόπος να αποφασίσουν αν θα ήθελαν να πραγματοποιήσουν την αγορά του προιόντος ή όχι.

Οι διαδικτυακοί πωλητές και έμποροι ζητούν πλέον από τους αγοραστές να μοιραστούν τις απόψεις και τις εμπειρίες τους σχετικά με τα προιόντα που αγόρασαν.
Εκατομμύρια σχόλια στο διαδίκτυο, γνώμες καταναλωτών για κάθε είδος προιόντος υπηρεσίας ή τόπους. Αυτό έχει κάνει το διαδίκτυο την σημαντικότερη πλέον πηγή απόκτησης ιδεών και απόψεων για ένα προϊόν ή μια υπηρεσία.
Το Amazon είναι ένας από τους κολλοσούς του ηλεκτρονικού εμπορίου και χρησιμοποιείται καθημερινά από την πλειοψηφία των ανθρώπων για τις αγορές τους. Για κάθε προϊόν μπορούν να διαβάσουν χιλιάδες κριτικές από άλλους πελάτες για τα προϊόντα που τους ενδιαφέρουν.
Αυτές οι αναθεωρήσεις παρέχουν πολύτιμες εκτιμήσεις για ένα προϊόν όπως η ιδιοκτησία, η ποιότητά του και οι συστάσεις που βοηθούν τους αγοραστές να κατανοούν σχεδόν κάθε λεπτομέρεια ενός προϊόντος.

Ωστόσο, με την αύξηση του αριθμού των διαθέσιμων κριτικών για ένα προϊόν, γίνεται δυσκολότερο το έργο ενός πιθανού αγοραστής. Διαφορετικές απόψεις και αμφίβολες κριτικές δημιουργούν συγκεχυμένη εικόνα για το εκάστοτε προϊόν. Συνεπώς, η αναζήτηση και παρακολούθηση των πηγών πληροφορίας για την εξόρυξη πραγματικά χρήσιμων πληροφοριών που περιέχονται σε αυτές είναι ένα δύσκολο εγχείρημα, κυρίως λόγω της πληθώρας των πηγών. Κάθε site συνήθως εμπεριέχει έναν μεγάλο όγκο κειμένων που εκφράζουν διάφορες απόψεις αλλα δεν είναι εύκολη η αποκρυπτογράφήση τους. Ο μέσος αναγνώστης θα δυσκολευτεί πολύ να εντοπίσει συγεκριμένους ιστοτόπους και να συγκεντρώσει με ακρίβεια τις πληροφορίες και τις απόψεις που περιλαμβάνονται σε αυτό.

Είναι φανερό ότι υπάρχει τεράστια ανάγκη για την αυτοματοποιημένη επεξεργασία και ανάλυση όλου αυτού του όγκου πληροφοριών, τόσο για τους πελάτες, όσο και για τις ίδιες τις επιχειρήσεις ηλεκτρονικού εμπορίου. Στην δεύτερη περίπτωση, τέτοιου είδους αναλύσεις βοηθούν τους κατασκευαστές να κατανοήσουν καλύτερα τους καταναλωτές και τις ανάγκες τους.
Για την αντιμετώπιση αυτού του προβλήματος έχουν χρησιμοποιηθεί διαφορετικές προσεγγίσεις, από την επεξεργασία φυσικής γλώσσας μέχρι την ανάλυση κειμένου και την υπολογιστική γλωσσολογία.

Συγκεκριμένα στην παρούσα διπλωματική προτείνεται και αναπτύσσεται αλγόριθμος ομαδοποίησης σχολίων και τελικών αποτελεσμάτων ανά προϊόν, με σκοπό την ενημέρωση των καταναλωτών, αλλά και των κατασκευαστών, για την ποιότητα και πιθανά ελαττώματα/δυσλειτουργίες των προιόντων τους.
Με την προσθήκη ενός τέτοιου αλγορίθμου, οι ιστότοποι ηλεκτρονικού εμπορίου θα μπορούν να ελέγχουν την ποιότητα των προϊόντων τους και να βελτιώνουν την αξιοπιστίας τους.

4.2 Ανάλυση της εφαρμογής

Αφού λοιπόν παρουσιάστηκε η ανάγκη την οποία εξυπηρετεί η παρούσα εφαρμογή, ακολουθεί αναλυτική περιγραφή της .

4.2.1 Περιγραφή και τρόπος λειτουργίας

Απώτερος στόχος αυτής της εργασίας είναι να δημιουργηθεί το υπόβαθρο για μια εφαρμογή η οποία θα είναι ικανή αρχικά να κατηγοριοποιεί μια κριτική ενός προϊόντος με βάση το αίσθημά της (polarity) σε δύο κλάσεις, μια θετική και μια αρνητική και στη συνέχεια, αφού έχει γίνει ο πρώτος διαχωρισμός να μπορεί να εξάγει από κάθε μια από αυτές τις κριτικές τις σημαντικότερες λέξεις, προκειμένου να διαπιστώνονται τα θετικά ή αρνητικά στοιχεία του προϊόντος.
Η αρχική κατηγοριοποίηση των κριτικών στοχεύει στο να μπορέσουμε να καταλάβουμε σε ποιες από αυτές οι χρήστες μιλούν με θετικά σχόλια και επαινούν το προϊόν, αλλά και σε ποιες αναφέρουν χαρακτηριστικά και χρήσεις του με τα οποία δεν είναι ευχαριστημένοι σαν καταναλωτές. Έπειτα, μπορούμε να επικεντρωθούμε, είτε στις θετικές κριτικές και να τις αναλύσουμε εκτενέστερα με στόχο να αποκτήσουμε σαφή εικόνα για το ποια χαρακτηριστικά του προϊόντος είναι αυτά που θα οδήγησουν κάποιον στο να το αγοράσει, είτε στις αρνητικές και να προσπαθήσουμε να εξάγουμε από αυτές τις σημαντικότερες λέξεις και φράσεις (συμπλέγματα άνω της μίας λέξεις μπορούν να φανούν εξαιρετικά χρήσιμα), ώστε να κατανοήσουμε τους

λόγους για τους οποίους οι καταναλωτές δεν είναι ευχαριστημένοι αλλά και να ληφθούν υπόψιν στις προτάσεις τους με στόχο τη βελτίωση των προϊόντων.
Το κομμάτι που αφορά την εξαγωγή των σπουδαιότερων λέξεων (keyword extraction) είναι ιδιαίτερα σημαντικό και απαιτεί μεγάλη προσοχή. Ο αρχικός διαχωρισμός των κριτικών απλώς σε θετικές και αρνητικές είναι εξαιρετικής σημασίας για την ανάγκη που θέλουμε να καλύψουμε, καθώς αποτελεί αρχική αλλά βασική πληροφορία. Η ανάκτηση λεπτομερέστερης γνώσης, μέσω των λέξεων-κλειδίων ενός κειμένου, εστιάζει σε συγκεκριμένα χαρακτηριστικά και απαιτεί περισσότερη προσοχή. Εξάγωντας ωστόσο τις συχνότερα χρησιμοποιούμενες λέξεις μπορεί να οδηγήθούμε σε λανθασμένα ή ακόμα και σε μη χρήσιμα συμπεράσματα. Για παράδειγμα, οι λέξεις με την μεγαλύτερη συχνότητα εμφάνισης είναι συνήθως άρθρα, σύνδεσμοι, αντωνυμίες και γενικότερα λέξεις οι οποίες δεν προσφέρουν καμία επί της ουσίας πληροφορία. Αυτές λοιπόν πρέπει να αφαιρεθούν πριν την επεξεργασία του κειμένου. Επίσης, το να ληφθεί μία λέξη ως η σημαντικότερή ενός κειμένου μπορεί να μην δίνει όλη την πληροφορία που απαιτείται, με αποτελέσμα κάποιες φορές να είναι μέχρι και παραπλανητική. Γι' αυτό το λόγο, οφείλουμε να εξάγουμε πέρα από μεμονωμένες λέξεις, και φράσεις, εννοιολογικά πλήρεις, οι οποίες επαναλαμβάνονται αυτούσιες, καθώς μπορεί να αποδειχθούν εξαιρετικά χρήσιμες και να περιλαμβάνουν πολλές αξιοποιήσιμες πληροφορίες.

4.2.2 Ανάλυση βημάτων

Για να αναπτυχθούν όλα τα παραπάνω ακολουθήθηκαν τα εξής βήματα:

Εξόρυξη κριτικών χρησιμοποιώντας scrapy tool

Το πρώτο και πολύ σημαντικό βήμα είναι η συλλογή μεγάλου πλήθους δεδομένων, τα οποία είναι σε αξιοποιήσιμη μορφή και περιλαμβάνουν χρήσιμες πληροφορίες για την εξαγωγή των επιθυμητών αποτεεσμάτων. Για το σκοπό αυτό χρησιμοποιήθηκε το framework της python που ονομάζεται scrapy.
Δημιουργήθηκε ένα spider, που ο κώδικάς του έχει τη δυνατότητα να βρίσκει την επιθυμητή πληροφορία στον html κώδικα της ιστοσελίδας του προϊόντος και να επιστρέφει αυτήν την πληροφορία σε διαχειρίσιμη μορφή.
Τα προϊόντα για τα οποία συλλέχθηκαν πληροφορίες βρίσκονται στην ιστοσελίδα του Amazon. Το spider έχει τη δυνατότητα να εντοπίσει τις χρήσιμες πληροφορίες για τα προϊόντα που μας ενδιαφέρουν στην html σελίδα. Οι χρήσιμες πληροφορίες που ανακτώνται από τη σελίδα των προϊόντων, είναι οι εξής:

- Όνομα προϊόντος
- Κατασκευαστής
- Κείμενο κριτικής προϊόντος

- Κωδικός κριτικής (Id)
- Τίτλος κριτικής
- Αστέρια βαθμολόγησης από άλλους χρήστες που διάβασαν την κριτική (rating)
- Ψήφοι από χρήστες που βοηθήθηκαν από την κριτική (found helpful)

Το κύριο μέρος της ανάλυσης γίνεται στο κείμενο της εκάστοτε κριτικής, όμως πληροφορίες όπως το όνομα του προϊόντος, ο κατασκευαστής ή τα αστέρια είναι πολύ σημαντικές για τα τελικά αποτελεσματά. Ειδικά τα αστέρια με τα οποία έχουν αξιολογήσει οι υπόλοιποι χρήστες την κάθε κριτική είναι πολύ σημαντικα σε έναν πρώτο διαχωρισμό των κριτικών σε θετικές και αρνητικές. Τα δεδομένα που εξάγονται από τις ιστοσελίδες των προϊόντων χρησιμοποιήθηκαν, τόσο στην ανάλυση αισθήματος (polarity) όσο και στην εξαγωγή των σημαντικότερων λέξεων.
Τέλος, μέσω ενός pipeline τα δεδομένα αποθηκεύτηκαν σε ένα index σε Elasticsearch ώστε είναι προσπελάσιμα ανά πάσα στιγμή.

Ανάλυση αισθήματος

Από τα δεδομένα της βάσης εισάγονται σε ένα pandas dataframe το κείμενο της κριτικής, τα αστέρια της και ο τίτλος του πρϊόντος.
Οι κριτικές που εξορύχθηκαν από την ιστοσελίδα του Amazon δεν είναι εξαρχής κατηγοριοποιημένες σε θετικές και αρνητικές. Πρόκειται για unlabeled δεδομένα. Αυτό σημαίνει πως για την αρχική εκπαίδευση του αλγορίθμου μας χρησιμοποιείται ένα μικρό πλήθος, ήδη κατηγοριοποιημένων (labeled) δεδομένων προκειμένου να τηρηθεί η αρχή της διαδικασίας του self training που ακολουθεί, και στη συνέχεια τροφοδοτείται με τα δεδομένα από τη βάση.
Αρχικά γίνεται μία προεπεξεργασία των μη κατηγοριοποιημένων δεδομένων που λαμβάνουμε από τη βάση ώστε να τα τροφοδοτήσουμε στη συνέχεια στον αλγόριθμο μάθησης. Χρησιμοποιούμε έναν αλγόριθμο που μετατρέπει τις λέξεις σε διανύσματα, τον CountVectorizer, ο οποίος χρησιμοποιεί το πλήθος εμφάνισης των λέξεων στο σύνολο των reviews, και δίνει ως έξοδο ένα matrix διανυσμάτων όλων των λέξεων που έχει δεί ως προς την προαναφερόμενη συχνότητα τους. Στη συνέχεια, εισάγουμε όλα τα σχόλια με την παραπάνω μορφή στον αλγόριθμο ταξινόμησης και λαμβάνουμε πλέον τις πιθανές κλάσεις στις οποίες ανήκουν. Οι κλάσεις είναι η θετική και η αρνητική, καθώς είναι επιθυμητός ένας ξεκάθαρος διαχωρισμός της πολικότητας των σχολίων και δεν δίνεται έμφαση στις εκφραστικές λεπτομέρειες που συνοδεύουν την ουδέτερη κλάση και μπορεί να περιπλέξουν την ανάλυση.

Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, για τη βελτίωση της απόδοσης του τελικού αλγορίθμου ταξινόμησης εφαρμόζουμε μια τεχνική

self-training, κατά την οποία εξάγουμε το σύνολο των πιο σίγουρων προβλέψεων με στόχο μια σημαντική αύξηση του σημαντικά μικρού αρχικού σετ δεδομένων. Η διαδικασία του self-training ολοκληρώνεται με το πέρας των επιθυμητών κύκλων εφαρμογής του αλγορίθμου και ως αποτέλεσμα διατηρούμε τα δεδομένα που έχουν κατηγοριοποιηθεί σε ξεχωριστό pandas frame για την περαιτέρω ανάλυσή τους.

Εξόρυξη λέξεων-κλειδιών

Με το πέρας λοιπόν sentiment analysis, πραγματοποιείται η τελική επεξεργασία, το keyword extraction.

Τα σχόλια που μας ενδιαφέρουν είναι εκείνα που έχουν αντιστοιχηθεί με την αρνητική κλάση, καθώς αναζητούμε λέξεις-κλειδιά που δηλώνουν τα ελαττώματα και τα στοιχεία των προϊόντων που χρήζουν βελτίωσης. Φιλτράροντας τα σχόλια που ανήκουν μόνο στην επιθυμητή αρνητική κλάση, χρησιμοποιούμε μια σειρά αλγορίθμων για την επεξεργασία του σώματος των μηνυμάτων, όπως την ανακοπή των καταλήξεων και τη λημματοποίησή τους. Στη συνέχεια μετρώντας τις συν-εμφανίσεις ή τη συχνότητα φράσεων ενός, δύο ή τριών λέξεων, καθώς και την αντίστροφη συχνότητα αυτών, tf-idf, καταλήγουμε στις τελικές λέξεις και φράσεις κλειδιά που δίνουν το βασικό νόημα των αρνητικών σχολίων κάθε προιόντος. Το φιλτράρισμα των μηνυμάτων, που αντιστοιχούν σε κάθε προϊόν, γίνεται με βάση το product_model χαρακτηριστικό των προϊόντων που είναι αποθηκευμένο στη βάση δεδομένων μας.

Κεφάλαιο 5 Ανάπτυξη

Σε αυτή την ενότητα θα προσπαθήσουμε μέσα από την ανάλυση κάποιων μεθόδων να δείξουμε τον τρόπο που λειτουργεί η εφαρμογή.

5.1 Scrapy Framework

Αρχικά δημιουργώντας ένα scrapy project έχουμε τη δυνατότητα να στήσουμε όλο το περιβάλλον για την εξόρυξη των σχολίων από το Amazon. Εισάγουμε μία κλάση με είσοδο ένα scrapy spider. Συγκεκριμένα τα spiders είναι αυτόνομοι crawlers που δέχονται μία σειρά από εντολές. Καλώντας ένα spider η συνάρτηση που θα τρέξει πρώτη είναι η parse, η οποία φορτώνει ασύγχρονα, με http requests στο allowed_domains, τις σελίδες που αναφέρονται στο πεδίο start_urls και στη συνέχεια εκτελεί τον κώδικα της. Εν προκειμένω, εκτελεί ένα scrapy request δίνοντας ως είσοδο στην συνάρτηση parse_reviews τις σελίδες με τα customer reviews για κάθε start_url. Κατά την κλήση της parse_reviews γίνεται η αντιστοίχηση των ζητούμενων πεδίων από ένα review με το τοπικό αντικείμενο items, τύπου AmazonreviewsItem, που δημιουργείται για κάθε επιμέρους σχόλιο. Τα πεδία αυτού του αντικειμένου είναι τα εξής: id, title, rating, found_helpfull, review_body, product_model, manufacturer.

```
import scrapy

class AmazonreviewsItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    id = scrapy.Field()
    title = scrapy.Field()
    rating = scrapy.Field()
    found_helpfull = scrapy.Field()
    review_body = scrapy.Field()
    product_model = scrapy.Field()
    manufacturer = scrapy.Field()
    #pass
```

Η διερεύνηση της κάθε σελίδας γίνεται με τα xpaths της. Ένα xpath είναι μία γλώσσα, ένας τρόπος, για να διευθυνσιοδότησουμε κάποια μέρη μίας σελίδας. Αφού γίνει η συγκεκριμένη διερεύνηση της σελίδας, μέσω των xpaths, και η αντιστοίχηση των απαραίτητων πεδίων του item, το αντικείμενο επιστρέφεται μέσω του yield και η συνάρτηση προχωράει εκ νέου στην αίτηση αλλαγής σελίδας καλώντας αναδρομικά τον εαυτό της μέσω πάλι ενός scrapy request. Με αυτόν τον τρόπο επιτυγχάνεται η ασύγχρονη εξόρυξη πληροφοριών από όλα τα προιόντα και για όλες τις σελίδες σχολίων του εκάστοτε προιόντος, γεγονός που μειώνει τον χρόνο εκτέλεσης σε προβλήματα με μεγάλο όγκο σελίδων.

```
import scrapy
from amazonreviews.items import AmazonreviewsItem
request_delay= 5

class ReviewspiderSpider(scrapy.Spider):
    name = "reviewspider"
    allowed_domains = ["amazon.co.uk"]
    start_urls= [  "https://www.amazon.com/Individual-Software-PMM-Q15-Professor-QuickBooks/dp/B00WBJB4AS",
                   "https://www.amazon.com/Final-Fantasy-XI-Ultimate-Collection-PC/dp/B002SQNGX2"
                ]
    def parse(self,response):
        review_page = response.xpath('//a[contains(@data-hook, "see-all-reviews-link-foot")]/@href').extract()
        review_page_url = "https://www.amazon.com" + review_page[0]
        yield scrapy.Request(url=review_page_url, callback=self.parse_reviews, dont_filter=True)
    def parse_reviews(self,response):
        list = response.xpath('//div[contains(@id, "cm_cr-review_list")]/div/@id').extract()
        for id in list:
            item = AmazonreviewsItem()
            item['id']= id
            item['title']= response.xpath('//div[@id=$id]/div/div/div/a/span/text()', id= id).extract()
            item['rating']= response.xpath('//div[@id=$id]/div/div/div/a/i/span/text()', id= id).extract()
            item['found_helpfull']= response.xpath('//div[@id=$id]/div/div/div/div/span/div/span/text()', id=id).extract()
            item['review_body']= response.xpath('//div[@id=$id]/div/div/div/span/span/text()', id= id).extract()
            item['product_model']= response.xpath('//a[contains(@data-hook, "product-link")]/text()').extract_first()
            item['manufacturer']= response.xpath('//a[contains(@class, "a-size-base a-link-normal")]/text()').extract_first()
            yield item
        next_page_url = response.xpath('//li[contains(@class, "a-last")]/a/@href').extract()
        if next_page_url:
            next_page_url = "https://www.amazon.com" + next_page_url[0]
            yield scrapy.Request(url=next_page_url, callback=self.parse_reviews, dont_filter=True)
```

Για να αποθηκευτούν όλα τα αντικείμενα που επιστρέφονται απαιτείται κατά την εκτέλεση να δοθούν εντολές, που αφορούν την μορφή αποθήκευσής τους, ή να δημιουργηθεί ένα pipeline που θα το καθορίζει αυτό. Στην πρώτη περίπτωση θα μπορούσε η μορφή να είναι ένα απλό αρχείο json ή csv. Επιλέχθηκε η δεύτερη και δημιουργήθηκε μια σύνδεση με ένα remote elasticsearch cluster. Οι λεπτομέρειες της σύνδεσης φαίνονται παρακάτω.

```
ITEM_PIPELINES = {
    'scrapyelasticsearch.scrapyelasticsearch.ElasticSearchPipeline': 100
}
ELASTICSEARCH_SERVERS = ['62.74.232.210:9210']
ELASTICSEARCH_INDEX = 'software'
ELASTICSEARCH_TYPE = 'items'
ELASTICSEARCH_UNIQ_KEY = 'id'
```

Πραγματοποιείται η σύνδεση σε έναν remote server σε συγκεκριμένη πόρτα με ορισμένες παραμέτρους που καθορίζουν σε ποιο index, software ή electronic, οι δύο κατηγορίες που επιλέξαμε από τα προιόντα του Amazon, και σε ποια μορφή αποθηκεύονται όλα τα αντικείμενα τύπου items, με πρωτεύον κλειδί το id field του κάθε αντικειμένου. Τέλος, αφού ολοκληρωθεί το crawl όλων των

ζητούμενων προιόντων, η περαιτέρω επεξεργασία τους γίνεται πλέον μέσω της βάσης δεδομένων.

5.2 Σύγκριση αλγορίθμων ταξινόμησης

Έπειτα από την συλλογή και την αποθήκευση των δεδομένων που ακολουθε΄θ η επιλογή αλγορίθμου. Οι υποψήφιοι αλγόριθμοι είναι ο Logistic Regression, ο Support Vector Machine (SVM) και ο Multinomial Naïve Bayes, οι οποίοι ενδείκνυνται για προβλήματα κατηγοριοποίησης κειμένου. Πριν τη σύγκρισή τους, καθορίζενται οι βέλτιστες υπερπαραμέτρους για τον κάθενα.
Η έννοια της υπερπαραμέτρου σε έναν αλγόριθμο είναι διαφορετική από αυτήν της παραμέτρου. Η παράμετρος ενός μοντέλου είναι μια μεταβλητή η οποία είναι εσωτερική, δηλαδή δεν μπορεί να ρυθμιστεί εξωτερικά από κάποιον, αλλά προσαρμόζεται ανάλογα τα δεδομένα. Παραδείγματα ορισμένων παραμέτρων μοντέλων είναι οι συντελεστές της εξίσωσης (coefficients) στην γραμμική παλινδρόμηση ή το πλήθος των support vectors στον SVM. Αντίθετα, μια υπερπαράμετρος ,για παράδειγμα η alpha στον MultinomialNB αλγόριθμο, είναι εξωτερικη μεταβλητή και δεν μπορεί να προσδιοριστεί με την εκπαίδευση στα δεδομένα. Για τον προσδιορισμό των βέλτιστων τιμών των υπερπαραμέτρων εφαρμόζονται διάφορες τεχνικές. Η τεχνική που χρησιμοποιήθηκέ στην παρούσα εργασία ονομάζεται gridsearch. Σε αυτή την τεχνική δίνουμε σε κάθε υπερπαράμετρο ένα σύνολο πιθανών τιμών και στη συνέχεια δοκιμάζονται όλοι οι πιθανοί συνδυασμοί αυτών των τιμών και εξάγονται τα αποτελέσματά τους. Τελικά επιλέγονται οι τιμές εκείνες για κάθε υπερπαράμετρο που μεγιστοποιούν την απόδοση του αλγορίθμου.

Logistic Regression

Logistic Regression

```
In [13]: from sklearn.linear_model import LogisticRegression

classifier = Pipeline([('vect', CountVectorizer(lowercase=False)),
                       ('clf', LogisticRegression())
])
# parameter tuning with grid search
from sklearn.model_selection import GridSearchCV
parameters = {'vect__ngram_range': [(1, 1), (1, 2),(1,3)],
              'vect__max_df': ( 0.7,0.8,0.9,1.0),
              'vect__min_df': (1,2),
              'clf__penalty': ['l1', 'l2'],
              'clf__C':[0.001,.009,0.01,.09,1,5,10,25],
}
gs_clf_LR = GridSearchCV(classifier, parameters,n_jobs=-1, verbose=1,cv=5)
gs_clf_LR.fit(X_train[:5000], y_train[:5000])
best_parameters_LR = gs_clf_LR.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print("\t%s: %r" % (param_name, best_parameters_LR[param_name]))

Fitting 5 folds for each of 384 candidates, totalling 1920 fits

[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  4.3min
[Parallel(n_jobs=-1)]: Done 196 tasks      | elapsed: 16.7min
[Parallel(n_jobs=-1)]: Done 446 tasks      | elapsed: 31.3min
[Parallel(n_jobs=-1)]: Done 796 tasks      | elapsed: 52.2min
[Parallel(n_jobs=-1)]: Done 1246 tasks      | elapsed: 79.9min
[Parallel(n_jobs=-1)]: Done 1796 tasks      | elapsed: 111.6min
[Parallel(n_jobs=-1)]: Done 1920 out of 1920 | elapsed: 119.4min finished

        clf__C: 1
        clf__penalty: 'l2'
        vect__max_df: 0.8
        vect__min_df: 2
        vect__ngram_range: (1, 3)
```

Ο πρώτος αλγόριθμος για τον οποίο αναζητήθηκαν οι βέλτιστες υπερπαράμέτροοι είναι ο Logistic Regression. Αρχικά, όπως φαίνεται στον κώδικα, γίνεται χρήση της κλάσης **Pipeline** της sklearn με την οποία ομαδοποιήούνται οι διάφοροι εκτιμητές (etsimators), κατηγοριοποιητές (classifiers) και μετατροπείς (transformers) που χρειαζόμαστε ώστε να διαχειριστούν τα δεδομένα μας και να εκπαιδεύσουμε το μοντέλο μας. Στη περίπτωσή μας, η ανάλυση κειμένου απαιτεί τη χρήση του **count vectorizer,** ο οποίος θα μετατρέψει κάθε πρόταση σε έναν 1xn πίνακα όπου n το πλήθος των λέξεων που περιλαμβάνει η πρόταση, με πληροφορία την συχνότητα εμφάνισης της κάθε λέξης. Το συνολικό κείμενο μετατρέπεται ουσιαστικά σε έναν Nxm πίνακα (N το πλήθος των προτάσεων και m το συνολικό πλήθος λέξεων). Στη συνέχεια ορίζουμε τον κατηγοριοποιητή (classifier) που θα χρησιμοποιήσουμε για να χτίσουμε το μοντέλο μας ως **clf**. Οι δυο αυτές οντότητες συνδυάζονται στο pipeline για να αναλυθούν παράλληλα. Στη συνέχεια ορίζεται ένα dictionary της python με το όνομα **parameters** όπου καταχωρούνται οι πιθανές τιμές των εξής υπερπαραμέτρων:

Για τον countvectorizer:
-n_gram_range

Ορίζει το εύρος του πλήθους των λέξεων που θα χωρίζουμε το κείμενο μας. για παράδειγμα unigrams (μόνο μία λέξη), bigrams (δύο λέξεις) κτλ

-max_df

Οι επιτρεπτές τιμές βρίσκονται μεταξύ 0 και 1, και αντιστοιχούν στην μέγιστη συχνότητα εμφάνισης μιας λέξης στο κείμενο. Δηλαδή λέξεις με συχνότητα αυστηρά μεγαλύτερη αγνοούνται.

-min_df

Αντιστοιχεί στην ελάχιστη επιτρεπτή τιμή συχνότητας εμφάνισης μιας λέξης στο κείμενο.

Για τον εκτιμητή του μοντέλου Logistic Regression:

-penalty

Αυτή η παράμετρος ουσιαστικά ρυθμίζει (regulizes) το ποσοστό προσαρμογής του μοντέλου μας στα δεδομένα εκπαίδευσης και είναι γνωστή ως **λ**. Στόχος ενός αποδοτικού μοντέλου δεν είναι η υπερβολική γνώση των δεδομένων στα οποία εκπαιδεύτηκε, αλλά η "μάθηση" από αυτά και η απόδοσή του σε καινούρια, η ικανότητα του μοντέλου δηλαδή να γενικεύει. Οι πιθανές τιμές της είναι $\mathbf{l_1}$, $\mathbf{l_2}$.

-C

Είναι και αυτή μια παράμετρος που αντιμετωπίζει την υπερβολική προσαρμογή του μοντέλου στα δεδομένα προς εκπαίδευση (overfittng) και ορίζεται ως το αντίστροφο της παραμέτρου **λ** που αναφέραμε, δηλαδή **1/λ.**

Το dictionary μαζί με το pipeline δίνονται παραμετρικά στην κλάση **GridSearchCV** της sklearn ώστε να δημιουργηθεί ένας εκτιμητής ως **gs_clf_LR** ο οποίος θα εκπαιδευτεί (fit) και μετά από δοκιμές σε όλους τους συδνυασμούς αυτών των παραμέτρων που αναλυσαμε θα μας δώσει τις βέλτιστες τιμές τους.

SVM

SVM

```
In [6]: from sklearn.svm import SVC

classifier = Pipeline([('vect', CountVectorizer(lowercase=False)),
                       ('clf', SVC()),
])
# parameter tuning with grid search
from sklearn.model_selection import GridSearchCV
parameters = {'vect__ngram_range': [(1, 1), (1, 2),(1,3)],
              'vect__max_df': ( 0.7,0.8,0.9,1.0),
              'vect__min_df': (1,2),
              'clf__kernel':('linear', 'rbf'),
              'clf__C':(1,0.25,0.5,0.75),
              'clf__gamma': (1,2,3,'auto'),
}
gs_clf_SVM = GridSearchCV(classifier, parameters,n_jobs=-1, verbose=1,cv=5)
gs_clf_SVM.fit(X_train[:2000], y_train[:2000])
best_parameters_SVM = gs_clf_SVM.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print("\t%s: %r" % (param_name, best_parameters_SVM[param_name]))
```

```
Fitting 5 folds for each of 768 candidates, totalling 3840 fits
```

```
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done 196 tasks      | elapsed: 11.7min
[Parallel(n_jobs=-1)]: Done 446 tasks      | elapsed: 26.2min
[Parallel(n_jobs=-1)]: Done 796 tasks      | elapsed: 45.3min
[Parallel(n_jobs=-1)]: Done 1246 tasks      | elapsed: 72.0min
[Parallel(n_jobs=-1)]: Done 1796 tasks      | elapsed: 106.0min
[Parallel(n_jobs=-1)]: Done 2446 tasks      | elapsed: 143.1min
[Parallel(n_jobs=-1)]: Done 3196 tasks      | elapsed: 188.4min
[Parallel(n_jobs=-1)]: Done 3840 out of 3840 | elapsed: 224.8min finished
```

```
        clf__C: 1
        clf__gamma: 1
        clf__kernel: 'linear'
        vect__max_df: 0.8
        vect__min_df: 1
        vect__ngram_range: (1, 3)
```

Η ίδια διαδικασία χρησιμοποιήθηκε για να προσδιορίσουμε τις βέλτιστες τιμές των υπερπαραμέτρων και για τον αλγόριθμο SVM. Τα πρώτα βήματα είναι αντίστοιχα με αυτά που πραγματοποιήθηκαν στην περίπτωση του Logistic Regression όπου ορίζουμε ως **classifier** ένα **pipeline** που συνδυάζει τον **countvectorizer** και τον εκτιμητή του μοντέλου μας **clf** και στη συνέχεια ένα dictionary με τις υπερπαραμέτρους προς βελτιστοποιήση.
Οι συγκεκριμένες παράμετροι για το object του countvectorizer είναι ακριβώς ίδιες με εκείνες που αναφέρθηκαν παραπάνω, ενώ για τον κατηγοριοποιητή μας είναι οι εξής:

-kernel

Με αυτή την παράμετρο ορίζεται η συνάρτηση που θα χρησιμοποιηθεί για την λειτουργία του αλγορίθμου και γενικά μπορεί να πάρει τις τιμές: ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’. Στη συγκεκριμένη περίπτωση δίνονται μόνο δυο επιλογές την ‘linear’ και την ‘rbf’, ενώ η εξ ορισμού τιμή του σε περίπτωση που δεν την ορίσουμε εμείς είναι η ‘rbf’.

-C

Ρυθμίζει την προσαρμογή του μοντέλου στα δεδομένα που δίνονται για εκπαίδευση και είναι αντιστρόφως ανάλογη της παραμέτρου **λ** όπως την ορίσαμε πριν.

-gamma

Είναι μία παράμετρος που ορίζει το ποσοστό επιρροής ενός μόνο εκπαιδευτικού παραδείγματος. Ο συγκεκριμένος συντελεστής καθορίζεται ανάλογα με την συνάρτηση που θα επιλεχθεί στη παράμετρο kernel και παίρνει τιμές μόνο για μη γραμμικές συναρτήσεις και συγκεκριμένα για τις ‘poly’, ‘rbf’, ‘sigmoid’. Ουσιαστικά ρυθμίζει το μέγεθος της επιρροής που θα έχουν στο μοντέλο οι παρατηρήσεις που διαφέρουν πολύ μεταξύ τους ή αλλιώς “απέχουν” πολύ στο διάγραμμα του χώρου.

MultinimialNB

MultinomialNB

In [37]:
```
classifier = Pipeline([('vect', CountVectorizer(lowercase=False)),
                       ('clf', MultinomialNB()),
])


# parameter tuning with grid search
from sklearn.model_selection import GridSearchCV
parameters = {'vect__ngram_range': [(1, 1), (1, 2),(1,3)],
              'vect__max_df': ( 0.7,0.8,0.9,1.0),
              'vect__min_df': (1,2),
              'clf__alpha': ( 0.022,0.025, 0.028),
}
gs_clf = GridSearchCV(classifier, parameters,n_jobs=-1, verbose=1,cv=5)
gs_clf.fit(X_train, y_train)
best_parameters = gs_clf.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print("\t%s: %r" % (param_name, best_parameters[param_name]))
```

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  9.9min
[Parallel(n_jobs=-1)]: Done 196 tasks      | elapsed: 40.9min
[Parallel(n_jobs=-1)]: Done 360 out of 360 | elapsed: 80.0min finished
```

```
        clf__alpha: 0.028
        vect__max_df: 0.9
        vect__min_df: 2
        vect__ngram_range: (1, 3)
```

Για τις υπερπαραμέτρους του αλγορίθμου Multinomial Naïve Bayes γίνονται οι απαραίτητες ενέργειες για την ομαλή λετουργια του κώδικά, δηλαδή

ορίζεται το pipeline, που θα συνδυάσει το αντικέιμενο vect του countvectorizer και τον κατηγοριοποιητή του αλγορίθμου clf και έπειτα ένα dictionary με λίστες πιθανών τιμών των υπερπαραμέτρων για κάθε κλειδί (key) του.
Οι υπερπαράμετροι για τον countvectorizer έχουν αναλυθεί στην περίπτωση του Logistic Regression οπότε ακολουθεί η περιγραφή της υπερπαραμέτρου που αφορά τον Multinomial Naïve Bayes εκτιμητή:
-alpha
Δέχεται δεκαδικού τύπου τιμές και η προκαθορισμένη τιμή της είναι 1.0. Η υπερπαράμετρος αυτή επηρεάζει το αποτέλεσμα του υπολογισμού της πιθανότητας σε περίπτωση που η προς εκτίμηση πρόταση περιλαμβάνει λέξεις που δεν έχει συναντήσει ο αλγόριθμος στα δεδομένα εκπαίδευσης. Συμβάλλει ουσιαστικά στην εξομάλυνση του αποτελέσματος.

Μετά τον καθορισμό τον υπερπαραμέτρων ακολουθεί η σύγκριση της απόδοσής του. Για τον σκοπό αυτό θα χρησιμοποιήσουμε την τεχνική του **cross validation** η οποία λειτουργει ως εξής: χρησιμοποιώντας ένα δοσμένο σύνολο δεδομένων προχωράει σε τυχαίο διαχωρισμό αυτών σε δεδομένα προς εκπαίδευση και σε δεδομένα για εκτιμήσεις. Αυτή η διαδικασία δεν γίνεται μόνο μια φορά αλλα όσες φορές ορίσουμε εμείς μέσω της παραμέτρου **cv.** Έτσι λοιπόν καταγράφεται η απόδοση του μοντέλου όχι μόνο σε τυχαία διαχωρισμένα δεδομένα αλλα και σε διαφορετικά κάθε φορά, λαμβάνοντας τελικά σαφέστερη εικόνα της αποτελεσματικότητας του.
Στις παρακάτω εικόνες φαίνεται η εφαρμογή της διαδικασίας που περιγράψαμε στα datasets X_train, y_train τα οποία περιλαμβάνουν ως δεδομένα το κείμενο των κριτικών και την κατηγορία τους ανάλογα με την πολικότητα(1ή-1)

Cross Validation

LR cross validation

In [43]:
```
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

vect = CountVectorizer(ngram_range=(1,3),max_df=0.8,min_df=2)
lr = LogisticRegression(C=1, penalty='l2')

clf = make_pipeline(vect, lr)
scores = cross_val_score(clf, X_train, y_train, cv=5)
print  scores,scores.mean()
```

```
[0.81166667 0.795      0.82833333 0.805      0.77666667] 0.8033333333333333
```

SVM cross validation

In [42]:
```
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC

vect = CountVectorizer(ngram_range=(1,3),max_df=0.8,min_df=1)
svm = SVC(C=1, kernel="linear", gamma=1)

clf = make_pipeline(vect, svm)
scores = cross_val_score(clf, X_train, y_train, cv=5)
print  scores,scores.mean()
```

```
[0.79333333 0.79166667 0.83333333 0.795      0.77833333] 0.7983333333333333
```

NB cross validation

In [45]:
```
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

vect = CountVectorizer(ngram_range=(1,3),max_df=0.9,min_df=2)
nb = MultinomialNB(alpha=0.028)

clf = make_pipeline(vect, nb)
scores = cross_val_score(clf, X_train, y_train, cv=5)
print  scores,scores.mean()
```

```
[0.76333333 0.80666667 0.81833333 0.83833333 0.79666667] 0.8046666666666666
```

Όπως είναι φανερό, ο αλγόριθμος, που όχι μόνο έχει την καλύτερη μέση απόδοση για πέντε συνολικά διαχωρισμούς δεδομένων, αλλά και την καλύτερη μεμονωμένη επίδοση ακρίβειας, είναι ο **Multinomial Naïve Bayes**.

5.3 Elasticsearch

Import from ElasticSearch Index

```
In [9]: from elasticsearch import Elasticsearch
import json
# Define config
host = '62.74.232.210'
port = 9210
timeout = 10000
index = "electronics"
size = 10000
body = {}
list_df=[]
# Init Elasticsearch instance
es = Elasticsearch(
    [
        {
            'host': host,
            'port': port
        }
    ],
    timeout=timeout
)
# Process hits here
def process_hits(hits):
    for item in hits:
        list_df.append(item['_source'])
#        print(json.dumps(item, indent=2))
# Check index exists
if not es.indices.exists(index=index):
    print("Index " + index + " not exists")
    exit()
# Init scroll by search
data = es.search(
    index=index,
    scroll='2m',
    size=size,
    body=body)
# Get the scroll ID
sid = data['_scroll_id']
scroll_size = len(data['hits']['hits'])
# Before scroll, process current batch of hits
process_hits(data['hits']['hits'])
while scroll_size > 0:
    "Scrolling..."
    data = es.scroll(scroll_id=sid, scroll='2m')
    # Process current batch of hits
    process_hits(data['hits']['hits'])
    # Update the scroll ID
    sid = data['_scroll_id']
    # Get the number of results that returned in the last scroll
    scroll_size = len(data['hits']['hits'])
    print (scroll_size)
```

Το πρώτο βήμα για την ανάπτυξη της βασικής λειτουργίας της εφαρμογής είναι να εισάχθούν τα δεδομένα από το Elasticsearch index. Για να πραγματοποιηθεί αυτό αρχικά δηλώνονται τα στοιχεία της σύνδεσης στον server της elasticsearch. Τέτοια στοιχεία είναι η IP διεύθυνση (**host**), η πόρτα σύνδεσης (**port**), το όνομα του index (**index**) και το χρονικό όριο μετά το οποίο θα σταματήσει η προσπάθεια σύνδεσης (**timeout**). Αυτά δίνονται ως παράμετροι στην κλήση της βιβλιοθήκης της Python, Elasticsearch με την οποία ουσιαστικά πραγματοποιείται η σύνδεση στη βάση.

Στη συνέχεια, ορίζεται ως **data** η αναζήτηση (query) που θα γίνει στη βάση για να ληφθούν ως απάντηση οι πληροφορίες των προϊόντων που έχουμε εξάγει από το Amazon. Αυτή η αναζήτηση πραγματοποιείται μέσω της μεθόδου **search** που προσφέρει η Elasticsearch και η οποία ουσιαστικά επιστρέφει τα αποτελέσματα του query (**hits).** Ως παραμέτρους σε αυτή την μέθοδο δίνονται το index στο οποίο είναι αποθηκευμένα τα δεδομένα μας, εδώ electronics, το **scroll** που ορίζει τη χρονική διάρκεια που είναι ενεργή και έγκυρη η αναζήτησή, το **size** με το οποίο ρυθμίζεται το πλήθος των documents που θα επιστρέφονται σε κάθε νέα προσπάθεια αναζήτησης και το **body** που καθορίζει την πληροφορία που θα επιστρέφεται, που στην περιπτωσή αυτή είναι όλα τα στοιχεία για την κάθε κριτική.
Σε αυτό το σημείο γίνεται και η πρώτη αναζήτηση στην elasticsearch και λαμβάνονται οι πρώτες εγγραφές, το πλήθος των οποίων είναι ίσο με το size και σε αυτές δίνεται μια νέα τιμή, το **scroll_id**.
Το **scroll_id** είναι μια πολύ σημανική παράμετρος, η οποία είναι ένας κωδικός που δίνεται στα δεδομένα που επιστρέφονται μετά από κάθε αναζήτηση στη βάση. Επειδή το σύνολο των δεδομένων που ανακτώνται είναι πολύ μεγάλο, η αναζήτηση πραγματοποιείται τμηματικά, επιστρέφοντας κάθε φορά πλήθος δεδομένων ίσο ή μικρότερο από το size που έχει τεθεί. Με αυτόν τον τρόπο όμως υπάρχει ο κίνδυνος επιστραφούν ως απάντηση οι ίδιες εγγραφές παραπάνω από μία φορά. Αυτός ακριβώς ο κίνδυνος αντιμετωπίζεται μέσω του **scroll-id**, το οποίο είναι ξεχωριστό για κάθε query, και έτσι τα δεδομένα που μας επιστρέφονται αποκτούν μια μοναδική ταυτότητα και συνεπώς στις επόμενες ανακτήσεις δεν λαμβάνονται documents από το index με **scroll_id** που έχει ήδη χρησιμοποιηθεί.
Έτσι λοιπόν χρησιμοποιούμε μια επανάληψη, η οποία, για όσο λαμβάνουμε απαντήσεις **(hits)** από την αναζήτησή, εκχωρεί τις πληροφορίες κάθε εγγραφής σε μια λίστα μέσω της συνάρτησης **process_hits,** που φαίνεται αναλυτικά στον κώδικά μας, ανανεώνει το scroll_id, βρίσκει και τυπώνει τον αριθμό των εγγραφών που ανακτώνται. Μόλις ολοκληρωθεί η διαδικασία έχουν ανακτηθεί όλες οι πληροφορίες για τις κριτικές από το Amazon σε μια λίστα, η οποία μπορει πολύ εύκολα μέσω μιας μόνο εντολής:

- all_reviews_df=pd.DataFrame(list_df)

να μετρατραπεί σε **pandas dataframe** για τη συνέχεια της ανάλυσή.

5.4 Ανάλυση αισθήματος

Για την ανάλυση πολικότητας χρησιμοποιήθηκαν δύο σετ δεδομένων, το αρχικό μικρό σε μέγεθος κατηγοριοποιημένο σύνολο και αυτό που λαμβάνουμε από τη βάση δεδομένων, που αποτελείται από ένα μεγάλο σε όγκο σύνολο μη κατηγοριοποιημένων σχολίων.
Αρχικά το ταξινομημένο σύνολο χωρίζεται σε train set και test set για την πρώτη εκπαίδευση και την αξιολόγηση του μοντέλου αντίστοιχα. Φιλτράρονται μόνο τα χαρακτηριστικά των σχολίων που μας ενδιαφέρουν, δηλαδη το σώμα τους, review_body , και η πολικότητά τους, polarity. Ακολουθεί η προεπεξεργασία των σετ εφαρμόζοντας τα στον CountVectorizer ώστε να μετατραπούν στην κατάλληλη για τον αλγόριθμο μορφή .
Μέσω του count vectorization ουσιαστικά οι προτάσεις του κειμένου μετατρέπονται σε διανύσματα συχνοτήτων. Πιο συγκεκριμένα, αφού δημιουργείται ένα αντικείμενο της κλάσης countvectorizer, με παραμέτρους που υπολογίστηκαν νωρίτερα για την βέλτιστη απόδοση του μοντέλου, χρησιμοποιούνται οι μέθοδοι **fit** και **transform**.
Με τη μέθοδο **fit** κατασκευάζεται το λεξιλόγιο (vocabulary),το οποίο αποτελεί το σύνολο των λέξεων που υπάρχουν στα κείμενα που έχουν δοθεί και τις οποίες θα αναζητούμε σε κάθε ένα ξεχωριστό κείμενο για να καταγράψουμε το πλήθος εμφάνισής τους. Με την **transform** μετατρέπουμε το κείμενο, που δίνουμε ως είσοδο, σε έναν δισδιάστατο πίνακα, ο οποίος έχει ως στήλες την κάθε λέξη που περιλαμβάνεται στο λεξιλόγιο που δημιουργήθηκε πριν, ως γραμμές το κάθε ένα ξεχωριστό κείμενο (σε περίπτωση που δώσουμε ως είσοδο ένα document με παραπάνω από ένα κείμενα) και περιλαμβάνει κάτω από κάθε λέξη το σύνολο των εμφανίσεων της λέξης σε κάθε κείμενο (γραμμή). Υπάρχει η επιλογή αυτές οι δύο μέθοδοι να εφαρμοστούν με μία μόνο εντολή, την **fit_transform,** κατά την οποία προσδιορίζεται το λεξιλόγιο με βάση το κείμενο εισόδου και δημιουργείται και ο δισδιάστατος πίνακας των συχνοτήτων.
Έτσι λοιπόν το διάνυσμα που δημιουργείται για κάθε μια κριτική του συνολικού dataset X_train, το οποίο δίνουμε ως είσοδο, είναι της μορφής που φαίνεται στο σχήμα που ακολουθεί. Στην πραγματικότητα οι στήλες δεν περιλαμβάνουν τις λέξεις αυτούσιες αλλά αριθμούς που αντιστοιχούν σε αυτές, δηλαδή το index τους.

	about	all	cent	cents	money	new	old	one	two
doc	1	1	3	1	1	1	1	1	1

In theory (a)

Index	0	1	2	3	4	5	6	7	8
doc	1	1	3	1	1	1	1	1	1

In practice (b)

Επόμενο βήμα είναι η εκπαίδευση του μοντέλου με το σετ εκπαίδευσης και η πρόβλεψη πολικότητας των μηνυμάτων στο test set.

Initial train

In [29]:
```
X_train = train['review_body']
y_train= train.polarity
```

In [30]:
```
X_test=accuracy_df.review_body
y_test=accuracy_df.polarity
```

In [31]:
```
cv = CountVectorizer(max_df=0.7,min_df=2,ngram_range=(1,3))
X_train_dtm = cv.fit_transform(X_train)
X_test_dtm = cv.transform(X_test)


nb = MultinomialNB(alpha=0.028)
%time nb.fit(X_train_dtm, y_train)
y_pred = nb.predict(X_test_dtm)
probabilities = nb.predict_proba(X_test_dtm)
```

```
CPU times: user 36.3 ms, sys: 0 ns, total: 36.3 ms
Wall time: 36.4 ms
```

Μια πρώτη εκτίμηση της ακρίβειας του μοντέλου φαίνεται παρακάτω.

No self-train Accuracy measurement

In [98]:
```
# calculate accuracy of class predictions
from sklearn import metrics
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Out[98]:
```
0.9187453692269696
```

5.1.5 Self-training

Εν συνεχεία εφαρμόζεται η διαδικασία του self-training, διατηρώντας το αρχικό σετ αξιολόγησης, μεταβάλλοντας όμως το σετ εκπαίδευσης και πρόβλεψης σε κάθε κύκλο της.
Τα σύνολα δεδομένων που φαίνονται στην παρακάτω δήλωση της συνάρτησης αντιστοιχίζονται ως εξής:
Train: το αρχικό σύνολο εκπαίδευσης
Test: το σύνολο των μη κατηγοριοποιημένων δεδομένων από τη βάση
final_Test: το αρχικό σύνολο στο οποίο γίνεται η αξιολόγηση του μοντέλου σε κάθε κύκλο.

Self-train

```
In [22]: def self_training_NB(Train,Test,final_Test,c):
             cycle = 0
             while True:
                 from sklearn import metrics
                 from sklearn.metrics import accuracy_score
                 from sklearn.feature_extraction.text import CountVectorizer
                 from sklearn.naive_bayes import MultinomialNB
                 import pickle
                 from sklearn.metrics import roc_curve
                 from sklearn.metrics import roc_auc_score
                 from sklearn.metrics import precision_score
                 from sklearn.metrics import recall_score
                 from sklearn.metrics import f1_score
                 from sklearn.metrics import precision_recall_curve
                 from sklearn.metrics import auc
                 from matplotlib import pyplot as plt
```

Με αφετηρία τον κύκλο 0, το μοντέλο εκπαιδεύεται με το Train set και η πρόβλεψη γίνεται στο Test set. Από αυτές τις προβλέψεις στόχος, σε κάθε κύκλο, είναι να εξάγουμε τις πιο σίγουρες, διαγράφοντας τις αντίστοιχες εγγραφές από το Test set και τοποθετώντας τες στο Train set στο οποίο και θα εκπαιδευτεί εκ νέου ο αλγόριθμος.
Για τον προσδιορισμό των πιο σίγουρων προβλέψεων θα χρειαστούμε τις πιθανότητες να ανήκει κάποιο σχόλιο στην κλάση 1, δηλαδή θετική, ή στην κλάση -1, δηλαδή αρνητική, που δίνει ο αλγόριθμος σε κάθε εγγραφή.

```
X_train = Train.review_body.tolist()
y_train = Train.polarity.tolist()
X_test = Test.review_body.tolist()
cv = CountVectorizer(max_df=0.7,min_df=2,ngram_range=(1,3))
X_train_dtm = cv.fit_transform(X_train)
X_test_dtm = cv.transform(X_test)

nb = MultinomialNB(alpha=0.028)
%time nb.fit(X_train_dtm, y_train)
y_pred = nb.predict(X_test_dtm)
posibilities = nb.predict_proba(X_test_dtm)
class_neg = []
class_1 = []

for pred in posibilities:
    class_neg.append(pred[0])
    class_1.append(pred[1])
pos_class_neg = pd.DataFrame({"review_body": X_test,"class_neg" : class_neg})
pos_class_1 = pd.DataFrame({"review_body" : X_test,"class_1" : class_1})
y_pred_class = pd.DataFrame({"review_body" : X_test,"pred_class" : y_pred})

test_pos0 = pd.merge(Test,pos_class_neg,on='review_body',left_index=True)
test_pos1 = pd.merge(test_pos0,pos_class_1,on='review_body',left_index=True)
test_pos = pd.merge(test_pos1,y_pred_class,on='review_body',left_index=True)
```

Τα κριτήρια διαλογής τους είναι η πιθανότητα ενός σχολίου που έχει καταγεγραμένη αξιολόγηση με ένα ή δύο αστέρια, rating =1 ή rating=2 , να ανήκει στην αρνητική κλάση και να είναι μεγαλύτερη ή ίση από έναν αριθμό c, και, αντίστοιχα, για τη θετική κλάση η πιθανότητα ενός σχολίου, που έχει καταγεγραμένη αξιολόγηση με τέσσερα ή πέντε αστέρια, rating =4 ή rating=5, να είναι μεγαλύτερη ή ίση από τον ίδιο αριθμό c.

```
process_data_neg = test_pos[((test_pos['rating'].astype('str').str.get(3) == "1")|(test_pos['rating'].astype('str').str.get(3) == "2"))
                            &(test_pos['pred_class']==-1.0)&(test_pos['class_neg']>=c)]
process_data_1 = test_pos[((test_pos['rating'].astype('str').str.get(3) == "5" )|(test_pos['rating'].astype('str').str.get(3) == "4"))
                          &(test_pos['pred_class']==1.0)&(test_pos['class_1']>=c)]

retrain_temp = pd.concat([process_data_1,process_data_neg],ignore_index=True)
retrain_temp['polarity'] = retrain_temp.pred_class

retrain_temp = retrain_temp.drop(["class_neg","class_1","pred_class"],axis=1)

print ("Reviews for re-train",len(retrain_temp))
```

Ο αριθμός c δηλώνει το πόσο σίγουρα επιθυμούμε να είναι τα σχόλια που θα τροφοδοτηθούν στο Train set για επανεκπαίδευση του μοντέλου και δηλώνεται κατά την κλήση της συνάρτησης.
Το τελικό κριτήριο τερματισμού της συνάρτησης είναι το μέγεθος του Test set και του συνόλου retrain, δηλαδή των δεδομένων που ανατροφοδοτούμε στον αλγόριθμο για επανεκπαίδευση. Αφού ληφθούν διάφορες ενδιάμεσες μετρήσεις και αξιολογήσεις απόδοσης του αλγορίθμου, εφαρμόζεται η διαδικασία τροποποίησης του Train και Test που περιγράφηκε παραπάνω.

```
    if ((len(Test)<=10000)|(len(retrain_temp)<=50)):

        possibilities=probs[:,0]
        fpr, tpr, thresholds = roc_curve(y_test_new, possibilities,pos_label=-1)
        auc_roc = auc(fpr, tpr)
        print('ROC AUC: %.2f' % auc_roc)

        #prec = precision_score(y_test_new,y_pred_new,average='micro')
        #rec = recall_score(y_test_new,y_pred_new,average='micro')
        prec, rec, thres = precision_recall_curve(y_test_new, possibilities,pos_label=-1)
        auc1 = auc(rec, prec)
        print('Precision-Recall AUC: %.2f' % auc1)

        # plot the precision-recall curves
        plt.plot(rec, prec, linestyle='--', label='Recall-Precision Plot')
        # axis labels
        plt.xlabel('Recall')
        plt.ylabel('Precision')
        # show the legend
        plt.legend()
        # show the plot
        plt.show()
        plt.savefig('Recall-Precision.png')

        plt.plot(fpr, tpr, color='orange', label='ROC')
        plt.plot([1, 0], [1, 0], color='darkblue', linestyle='--')
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('Receiver Operating Characteristic (ROC) Curve')
        plt.legend()
        plt.show()
        plt.savefig('ROC.png')
        return cycle
    else:
        cycle = cycle +1
```

```
    retrain_temp_id_todelete = retrain_temp.review_body
    new_test_set = Test
    new_test_set = new_test_set.set_index('review_body')
    new_test_set = new_test_set.drop(retrain_temp_id_todelete)
    new_test_set = new_test_set.reset_index()
    new_train_set = pd.concat([Train,retrain_temp],ignore_index=True,sort=True)

    X_train_new = new_train_set.review_body.tolist()
    y_train_new = new_train_set.polarity.tolist()
    X_test_new = final_Test.review_body.tolist()
    y_test_new = final_Test.polarity.tolist()

    cv = CountVectorizer(max_df=0.7,min_df=2,ngram_range=(1,3))
    X_train_dtm = cv.fit_transform(X_train_new)
    X_test_dtm = cv.transform(X_test_new)

    nb = MultinomialNB(alpha=0.028)
    %time nb.fit(X_train_dtm, y_train_new)
    y_pred_new = nb.predict(X_test_dtm)
    probs= nb.predict_proba(X_test_dtm)

    Train = new_train_set
    Test = new_test_set
```

Κατά την κλήση της συνάρτησης που εκτελεί τη διαδικασία του self-train, βλέπουμε, σε κάθε κύκλο της, το μέγεθος του retrain set που προστίθεται στο σύνολο Train και τέλος σε ποιό κύκλο επανάληψης τερματίστηκε η μέθοδος της αυτοεκπαίδευσης.

```
In [23]: self_training_NB(no_duplicates_train,no_duplicates[:100000],accuracy_df_new,0.9999)

CPU times: user 25.3 ms, sys: 368 µs, total: 25.7 ms
Wall time: 25.7 ms
('Reviews for re-train', 62410)
CPU times: user 219 ms, sys: 0 ns, total: 219 ms
Wall time: 219 ms
CPU times: user 186 ms, sys: 262 µs, total: 186 ms
Wall time: 186 ms
('Reviews for re-train', 17973)
CPU times: user 179 ms, sys: 0 ns, total: 179 ms
Wall time: 179 ms
CPU times: user 238 ms, sys: 385 µs, total: 238 ms
Wall time: 238 ms
('Reviews for re-train', 1874)
CPU times: user 190 ms, sys: 40.9 ms, total: 231 ms
Wall time: 230 ms
CPU times: user 305 ms, sys: 140 µs, total: 305 ms
Wall time: 305 ms
('Reviews for re-train', 527)
CPU times: user 207 ms, sys: 20.5 ms, total: 227 ms
Wall time: 227 ms
CPU times: user 222 ms, sys: 6 µs, total: 222 ms
Wall time: 222 ms
('Reviews for re-train', 176)
CPU times: user 245 ms, sys: 187 µs, total: 246 ms
Wall time: 246 ms
CPU times: user 256 ms, sys: 40 ms, total: 295 ms
Wall time: 296 ms
('Reviews for re-train', 66)
CPU times: user 267 ms, sys: 0 ns, total: 267 ms
Wall time: 267 ms
CPU times: user 300 ms, sys: 409 µs, total: 300 ms
Wall time: 300 ms
('Reviews for re-train', 26)
```

5.1.6 Εξαγωγή λέξεων κλειδιών

Στο τελευταίο στάδιο της εφαρμογής μας πραγματοποιήθηκε η εξόρυξη των πιο σημαντικών λέξεων-φράσεων από τις κριτικές. Για να έχει νόημα αυτή η επεξεργασία και για να είναι δυνατή η εξαγωγή πραγματικά χρήσιμων συμπερασμάτων, θα πρέπει οι λέξεις κλειδιά να αναζητηθούν σε

συγκεκριμένες κριτικές, και όχι σε όλες συνολικά. Ο στόχος είναι οι αρνητικές κριτικές και η επεξεργασία αυτών ανά προϊόν. Εφόσον οι κριτικές έχουν κατηγοριοποιηθεί σε θετικές ή αρνητικές με ικανοποιητικό ποσοστό ακρίβειας, απομονώνονται οι αρνητικά προσανατολισμένες σε ένα ξεχωριστό σύνολο δεδομένων (dataset). Από το ολικό dataset δημιουργείται μία λίστα με τα ονόματα των προϊόντων για τα οποία έχουμε πληροφορίες ως εξής:

product_names=final_with_pred['product_model'].unique().tolist()

Με τη μέθοδο unique() κρατάμε μόνο μια φορά την κάθε εγγραφή της στήλης product_model, δηλαδή το κάθε προϊόν, και όχι όσες φορές εμφανίζεται.
Έτσι, για κάθε ένα προϊόν σε αυτή τη λίστα για το οποίο έχουμε αρνητικές κριτικές, δηλαδή βρίσκουμε εγγραφές στο νέο dataset που δημιουργήσαμε, καλείται η συνάρτηση keyword που ακολουθεί.

In [23]:
```
def keyword(dataset):
    dataset['word_count'] = dataset['review_body'].apply(lambda x: len(str(x).split(" ")))
    dataset[['review_body','word_count']].head()
    dataset.word_count.describe()
    freq = pd.Series(' '.join(dataset['review_body']).split()).value_counts()[:20]
    temp=[]
    if len(freq) < 20:
        print (len(freq))
        return
    for i in range(0,20):
        temp.append(freq.index[i])
        temp
    import re
    import nltk
    nltk.download('stopwords')
    from nltk.corpus import stopwords
    from nltk.stem.porter import PorterStemmer
    from nltk.tokenize import RegexpTokenizer
    nltk.download('wordnet')
    from nltk.stem.wordnet import WordNetLemmatizer
    import matplotlib.pyplot as plt
    stop_words = set(stopwords.words("english"))
    stop_words = stop_words.union(temp)
    corpus = []
    for i in range(0, len(dataset)):
        #Remove punctuations
        text = re.sub('[^a-zA-Z]', ' ', dataset['review_body'][i])
        #Convert to lowercase
        text = text.lower()
        #remove tags
        text=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",text)
        # remove special characters and digits
        text=re.sub("(\\d|\\W)+"," ",text)
        ##Convert to list from string
        text = text.split()
        ##Stemming
        ps=PorterStemmer()
        #Lemmatisation
        lem = WordNetLemmatizer()
        text = [lem.lemmatize(word) for word in text if not word in
                stop_words]
        text = " ".join(text)
        corpus.append(text)
    #print(corpus)
    from sklearn.feature_extraction.text import CountVectorizer
    cv=CountVectorizer(max_df=0.8,stop_words=stop_words, max_features=10000, ngram_range=(1,3))
    X=cv.fit_transform(corpus)
    list(cv.vocabulary_.keys())[:10]
```

Αρχικά, χρησιμοποιείται η μέθοδος **split** με παράμετρο εισόδου το κενό για να πετύχουμε τον διαχωρισμό του κειμένου σε λέξεις. Τα δεδομένα που έχουμε δώσει ως είσοδο στη συνάρτηση είναι το σύνολο των κριτικών για ένα προϊόν και έτσι μπορούμε να το επεξεργαστούμε σαν ένα ενιαίο κείμενο. Αφού λοιπόν χωριστέι το κείμενο σε επιμέρους λέξεις, κρατούνται οι είκοσι πιο συχνά χρησιμοποιούμενες σε όλο το μήκος του κειμένου. Αυτές, εκχωρούνται σε μία λίστα **temp, η οποία συνενώνεται** με τη λίστα **stop_words.** Η συγκεκριμένη λίστα περιέχει τις πιο συνηθισμένες λέξεις της αγγλικής γλώσσας, όπως τα άρθρα (a, the), τις προθέσεις (of, to) ή τους συνδέσμους (and) και σε αυτές προσθέτίθενται οι πιο συνηθισμένες του εκάστοτε κειμένου. Με αυτόν τον τρόπο δημιουργείται ένα σύνολο από λέξεις, οι οποίες πρέπει να αφαιρούνται από την ανάλυση, καθώς από την μια δεν προσφέρουν καμία ουσιώδη πληροφορία για το νόημα του κειμένου και από την άλλη εμφανίζονται με τόσο υψηλή συχνότητα που σε περίπτωση που δεν αφαιρεθούν μπορούν να αλλοιώσουν τα αποτελέσματα, καθώς θα έχουν πάρει τη θέση λέξεων με χαμηλότερη συχνότητα μεν αλλά με πραγματικό νόημα για την πληροφορία του κειμένου δε.
Ακολούθως, δημιουργείται το τελικό σύνολο των λέξεων στο οποίο θα πραγματοποιηθεί η αναζήτηση των λέξεων-κλειδιών. Γι' αυτό, δεν αρκεί να έχει απλώς χωριστεί το κείμενο σε λέξεις, αλλά θα πρέπει να έχει ολοκληρωθεί η διαδικασία του **text preprocessing**, η οποία περιλαμβάνει μία σειρά βημάτων. Τέτοια βήματα είναι η αφαίρεση των σημείων στίξης, των ειδικών χαρακτήρων (πχ @,#), η μετατροπή όλων των γραμμάτων σε πεζά ώστε να μην έχουμε ανούσιες διακρίσεις, αλλά και οι διαδικασίες **stemming** και **lemmatization.** Αυτές οι δυο ενέργειες έχουν αναλυθεί ήδη, όμως για χάρη ευκολίας παρουσιάζονται εν συντομία και εδώ. Το stemming αποτελεί την διαδικασία αφαίρεσης από μια λέξη της κατάληξής της κρατώντας έτσι μόνο την ρίζα της, ενώ το lemmatization είναι μια περισσότερο λεξιλογική διαδικασία που αντικαθιστά τη λέξη με το θέμα της. Με χρήση κατάλληλων βιβλιοθηκών για επεξεργασία φυσικής γλώσσας (nlp), καλείται αρχικά η μέθοδος **sub,** για την αντικατάσταση των σημείων στίξης και των ειδικών χαρακτήρων με το κενό. Στη συνέχεια, εφαρμόζονται τεχνικές μέσω των αντικειμένων **ps** και **lem** και με αυτόν τον τρόπο τελικά δημιουργείται το σύνολο των λέξεων προς επεξεργασία ή αλλιώς **corpus.**

```
def get_top_n_words(corpus, n=10):
    vec = CountVectorizer().fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]
#Convert most freq words to dataframe for plotting bar plot
top_words = get_top_n_words(corpus, n=20)
top_df = pd.DataFrame(top_words)
top_df.columns=["Word", "Freq"]
print(top_df)
#Barplot of most freq words
import seaborn as sns
sns.set(rc={'figure.figsize':(13,8)})
g = sns.barplot(x="Word", y="Freq", data=top_df)
g.set_xticklabels(g.get_xticklabels(), rotation=30)
plt.show()
#Most frequently occuring Bi-grams
def get_top_n2_words(corpus, n=5):
    vec1 = CountVectorizer(ngram_range=(2,2),max_features=2000).fit(corpus)
    bag_of_words = vec1.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec1.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1],reverse=True)
    return words_freq[:n]
top2_words = get_top_n2_words(corpus, n=20)
top2_df = pd.DataFrame(top2_words)
top2_df.columns=["Bi-gram", "Freq"]
print(top2_df)
#Barplot of most freq Bi-grams
sns.set(rc={'figure.figsize':(13,8)})
h=sns.barplot(x="Bi-gram", y="Freq", data=top2_df)
h.set_xticklabels(h.get_xticklabels(), rotation=45)
plt.show()
    #Most frequently occuring Tri-grams
def get_top_n3_words(corpus, n=None):
    vec1 = CountVectorizer(ngram_range=(3,3),max_features=2000).fit(corpus)
    bag_of_words = vec1.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec1.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1],reverse=True)
    return words_freq[:n]
top3_words = get_top_n3_words(corpus, n=20)
top3_df = pd.DataFrame(top3_words)
top3_df.columns=["Tri-gram", "Freq"]
print(top3_df)
#Barplot of most freq Tri-grams
sns.set(rc={'figure.figsize':(13,8)})
j=sns.barplot(x="Tri-gram", y="Freq", data=top3_df)
j.set_xticklabels(j.get_xticklabels(), rotation=45)
```

Οι κύριες ενέργειες για την εύρεση των λέξεων με την μεγαλύτερη συχνότητα εμφάνισης γίνονται στη συνάρτηση **get_top_n_words** που ορίζεται παραπάνω. Αρχικά, ορίζεται ένα αντικείμενο της κλάσης countvectorizer και αφορά την δημιουργία ενός διανύσματος με στήλες το index που αντιστοιχίζεται σε κάθε λέξη και την προσδιορίζει μοναδικά, γραμμές έναν αύξοντα αριθμό για το κάθε ξεχωριστό κείμενο που περιέχεται στο συνολικό corpus και πληροφορία για το πλήθος των φορών που εμφανίζεται η κάθε λέξη. Με τη μέθοδο fit κατασκευάζουμε το λεξιλόγιο και με την transform δημιουργούμε τον πίνακα. Ο πίνακας διαστάσεων 1xN (1 γιατί μπορεί να δίνουμε πολλές κριτικές για ένα προϊόν, τις επεξεργαζόμαστε όμως ως ένα ενιαίο κείμενο και N το πλήθος των ξεχωριστών λέξεων) είναι ο **bag_of_words.** Πάνω στον πίνακα χρησιμοποιείται η μέθοδος **sum()**, η οποία

υπολογίζει άθροισμα σε έναν πίνακα. Ωστόσο, ο τρόπος με τον οποίο το υπολογίζει καθορίζεται από την τιμή της παραμέτρου **axis** η οποία ορίζει αν αυτό το άθροισμα θα υπολογισθεί ως προς τις γραμμές ή ως προς τις στήλες. Επειδή η μορφή του πίνακα είναι 1xN, το άθροισμα πρέπει να υπολογίζεται "κάθετα", δηλαδή ως προς τις στήλες, και γι' αυτό το λόγο η τιμή της axis είναι ίση με το μηδέν. Έτσι, προκύπτει ένας αντίστοιχος 1xN πίνακας που έχει ως index την κάθε λέξη και ως πληροφορία το άθροισμα των φορών εμφάνισής της σε κάθε document. Για να γίνει πλήρως κατανοητό θα δώσουμε ένα παράδειγμα:

Έστω ότι έχουμε το εξής κείμενο που αποτελείται από δύο documents:

texts = ["Hello world", "Python makes a better world"]

εφαρμόζουμε τον countvectorizer

vec = CountVectorizer().fit(texts)

bag_of_words = vec.transform(texts)

και έχουμε το εξής αποτέλεσμα

```
In [32]: print(bag_of_words)

  (0, 1)        1
  (0, 4)        1
  (1, 0)        1
  (1, 2)        1
  (1, 3)        1
  (1, 4)        1

In [33]: bag_of_words.shape

Out[33]: (2, 5)
```

Το οποίο όπως βλέπουμε είναι ένας 5x2 πίνακας, ο οποίος έχει αριστερά την θέση της κάθε λέξης με το πρώτο ψηφίο να δείχνει το document στο οποίο βρίσκεται και το δεύτερο να δείχνει το index της (βλέπουμε δυο φορές το index 4 για την λέξη world) και στα δεξιά τις φορές εμφάνισης της.

Έπειτα ο πίνακας του αθροίσματος προκύπτει ως εξής:

sum_words = bag_of_words.sum(axis=0)

```
In [26]: print(sum_words)

         [[1 1 1 1 2]]
```

Επιβεβαιώνεται λοιπόν αυτό που εξηγήσαμε νωρίτερα ότι τελικά έχουμε έναν πίνακα με τις συνολικές φορές εμφάνισης του κάθε index (λέξης).

Συνεχίζοντας , με τις μεθόδους **vocabulary_.items()** δημιουργείται ένα λεξικό της Python (dictionary), το οποίο έχει ως ζεύγος κλειδί-τιμή τον συνδυασμό λέξη-index. Εκμεταλλευόμαστε αυτή τη μοναδικότητα και με μια επανάληψη για κάθε ζεύγος λέξη – Index εκχωρούμε σε μια λίστα τις τιμές **word**, **sum_words[0, idx]** που αντιστοιχούν στη λέξη και στη συχνότητα εμφάνισης της λέξης όπως αυτή βρίσκεται στον πίνακα sum_words. Η σωστή αντιστοίχιση της λέξης με τη δική της συχνότητα στον πίνακα επιτυγχάνεται πάλι με την μοναδική σχέση λέξης – index. Η λίστα αυτή μετατρέπεται εν συνεχεία σε panda dataframe, και, ουσιαστικά, έχει ολοκληρωθεί ολοκληρωθεί το βασικότερο μέρος της εξόρυξης των σημαντικότερων λέξεων.

Επειδή, όπως έχει ήδη αναφερθεί, σημαντικό ρόλο παίζουν και οι συνδυασμοί δυο και τριών λέξεων, και όχι μόνο οι μεμονωμένες λέξεις, δημιουργήθηκαν δυο ακόμα συναρτήσεις, οι **get_top_n2_words**, **get_top_n3_words.** Λειτουργούν με τον ίδιο ακριβώς τρόπο, μόνο που στον ορισμό του αντικειμένου countvectorizer δίνεται ως παράμετρος **ngram_range=(2,2)** και **ngram_range=(3,3)** αντίστοιχα. Με αυτόν τον τρόπο δημιουργούνται δυο ακόμα dataframes που περιέχουν τα **bigrams** και τα **3grams** με τις υψηλότερες συχνότητες εμφάνισης.

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
tfidf_transformer.fit(X)
# get feature names
feature_names=cv.get_feature_names()
# fetch document for which keywords needs to be extracted
doc=corpus
#generate tf-idf for the given document
tf_idf_vector=tfidf_transformer.transform(cv.transform(doc))
#Function for sorting tf_idf in descending order
from scipy.sparse import coo_matrix
def sort_coo(coo_matrix):
    tuples = zip(coo_matrix.col, coo_matrix.data)
    return sorted(tuples, key=lambda x: (x[1], x[0]), reverse=True)
def extract_topn_from_vector(feature_names, sorted_items, topn=10):
    """get the feature names and tf-idf score of top n items"""
    #use only topn items from vector
    sorted_items = sorted_items[:topn]
    score_vals = []
    feature_vals = []
    # word index and corresponding tf-idf score
    for idx, score in sorted_items:
        #keep track of feature name and its corresponding score
        score_vals.append(round(score, 3))
        feature_vals.append(feature_names[idx])
    #create a tuples of feature,score
    #results = zip(feature_vals,score_vals)
    results= {}
    for idx in range(len(feature_vals)):
        results[feature_vals[idx]]=score_vals[idx]
    return results
    #sort the tf-idf vectors by descending order of scores
sorted_items=sort_coo(tf_idf_vector.tocoo())
    #extract only the top n; n here is 10
keywords=extract_topn_from_vector(feature_names,sorted_items,5)

print("\nKeywords:")
for k in keywords:
    print(k,keywords[k])
```

Τέλος, εντοπίζονται οι σημαντικότερες λέξεις χρησιμοποιώντας έναν διαφορετικό τρόπο υπολογισμού της συχνότητας εμφάνισης μιας λέξης. Πρόκειται για τον **TF-IDF** (term frequency–inverse document frequency) **vectorizer,** ο οποίος δεν μετράει απλά το πλήθος των φορών που εμφανίζεται μια λέξη σε ένα κείμενο, αλλά ουσιαστικά αξιολογεί το πόσο σχετική είναι μια λέξη με το συνολικό νόημα του κειμένου.

Αυτό το πετυχαίνει με τον πολλαπλασιασμό δυο ξεχωριστών μεγεθών, του TF (term frequency) και του IDF (inverse Document Frequency).

- TF

Αυτός ο όρος αντιστοιχεί στην συχνότητα εμφάνισης μιας λέξης στο κείμενο και υπολογίζεται απλά μετρώντας το πόσες φορές εμφανίζεται η συγκεκριμένη λέξη και διαιρώντας αυτόν τον αριθμό με το σύνολο των λέξεων του κειμένου.

- IDF

Είναι η αντίστροφη συχνότητα κειμένου και ορίζεται ως ο λογάριθμος του κλάσματος με αριθμητή το συνολικό πλήθος των κειμένων και παρονομαστή το πλήθος των κειμένων που περιέχουν τη λέξη για την οποία θέλουμε να υπολογίσουμε τη συχνότητα. Ουσιαστικά, αυτός ο όρος υπολογίζει πόσο σπάνια ή κοινή είναι μια λέξη στο σύνολο των κειμένων.
Για να εκφράσουμε με πιο μαθηματικό τρόπο αυτά που περιγράψαμε παραπάνω, η Tf-idf συχνότητα για μια λέξη ***t*** ενός κειμένου ***d*** σε ένα σύνολο κειμένων ***D*** υπολογίζεται ως εξής:

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D)$$

Όπου

$$tf(t,d) = log(1 + freq(t,d))$$

$$idf(t,D) = log(\frac{N}{count(d \in D : t \in d)})$$

Προχωρώντας στην ανάπτυξη, πρέπει πρώτα να τονίστεί ότι, πριν από την επεξεργασία για την εύρεση των πιο σημαντικών λέξεων υπολογίζοντας την TF-IDF συχνότητα, έχει οριστείαντικείμενο countvectorizer ως εξής:

```
cv = CountVectorizer(max_df=0.8,stop_words=stop_words,max_features=
10000, ngram_range=(1,3))
```

με την παράμετρο **stop_words** προσδιορίζονται οι λέξεις που εξαιρούνται από το τελικό λεξιλόγιο (vocabulary) που δημιουργείται. Στη συγκεκριμένη περίπτωση δίνεται η λίστα που δημιουργήθηκε προηγούμενα. Με τη μέθοδο **fit_transform** κατασκευάζεται το διδιάστατο διάνυσμα **X** από το corpus.
Στη συνέχεια, αφού εισάγεται το απαραίτητο πακέτο **TfidfTransformer** της κλάσης **feature_extraction.text,** ορίζεται ένα αντικείμενο tfIdfTransformer και χρησιμοποιείται η μέθοδος fit στο διάνυσμα X. Με αυτόν τον τρόπο υπολογίζεται ο όρος IDF για κάθε ένα index (λέξη) που υπάρχει στο corpus. Έτσι ο **tfidf_transformer,** που δηλώνεται στον κώδικά, είναι ένας 1xN πίνακας με τις IDF συχνότητες της κάθε λέξης, οι οποίες παίρνουν τιμές ίσες και μεγαλύτερες του 1, με τη μονάδα να συμβολίζει ότι η συγκεκριμένη λέξη εμφανίζεται σε όλα τα documents του corpus. Έχοντας πια τις τιμές του όρου IDF μπορεί να γίνει ο υπολογισμός των τελικών συχνοτήτων TF-IDF, αξιοποιώντας τη μέθοδο **transform** του αντικειμένου **tfidf_transformer** για

τον πίνακα με τις φορές εμφάνισης των λέξεων του κειμένου, που προκύπτει από την αντίστοιχη μέθοδο **transform** του countvectorizer. Ουσιαστικά, με αυτό τον τρόπο υπολογίζεται το γινόμενο Tf * Idf και σαν αποτέλεσμα προκύπτει ένας 1xN πίνακας, ο οποίο για κάθε index μας δίνει την Tf-Idf συχνότητά του.
Ο πίνακας μετατρέπεται σε πίνακα με τη μορφή συντεταγμένων (coordinate format) με την μέθοδο **tocoo** και δίνεται σαν παράμετρος στην κλήση της συνάρτησης **sort_coo,** με την οποία ταξινομούνται με φθίνουσα σειρά τα στοιχεία του. Επίσης, με τη μέθοδο **get_feature_names** του αντικειμένου countvectorizer δημιουργείται μία λίστα με τις λέξεις από τις οποίες αποτελείται το λεξιλόγιο που δημιουργεί ο countvectorizer.
Τελικά, καλείται η **extract_topn_from_vector** με τρεις παραμέτρους, τη λίστα με τις λέξεις του vocabulary (feature_names), τον ταξινομημένο πίνακα sorted_items και τον αριθμό που δηλώνει πόσες λέξεις με κορυφαίες συχνότητες θα κρατήσουμε. Εκμεταλλευόμενοι τη μοναδικότητα των ζευγαριών index – συχνότητα και index – λέξη που υπάρχει στους πίνακες sorted_items και feature_names αντίστοιχα, δημιουργούνται δυο λίστες στις οποίες εκχωρούμε τη συχνότητα που αντιστοιχεί στο κάθε index, στην score_vals, και στην λέξη που χαρακτηρίζεται από το αντίστοιχο index, στην feature_vals. Αυτές οι δύο λίστες χρησιμοποιούνται σε μία επανάληψη στην οποία δημιουργείται το λεξικό με τα αποτελέσματα (results), χρησιμοποιώντας ως σημείο αναφοράς το index για να τις παραλληλίσουμε. Από αυτές επιλέγουμε να κρατήσουμε τις πέντε με τις υψηλότερες Tf-Idf συχνότητες, οι οποίες κυμαίνονται από το μηδέν μέχρι τη μονάδα και όσο πιο κοντά σε αυτή βρίσκονται τόσο πιο σχετική είναι η αντίστοιχη λέξη με το κείμενο.

Κεφάλαιο 6 Αποτελέσματα

6.1 Στόχοι και προκλήσεις

Όπως έχει ήδη αναφερθεί, ο αρχικός στόχος ήταν η κατηγοριοποίηση κριτικών που αντλήσαμε από τις διάφορες σελίδες προϊόντων του Amazon με κριτήριο το polarity. Αυτή η κατηγοριοποίηση όμως στη συνέχεια καθορίζει το σύνολο των κριτικών στις οποίες αναζητούνται οι πιο σημαντικές λέξεις και φράσεις. Για αυτό το λόγο, είναι εξαιρετικά σημαντικό να γίνει με μεγάλη ακρίβεια και συνέπεια.

Το ποσοστό ακρίβειας σε εφαρμογές εξόρυξης αισθήματος από κείμενα έχει μεγάλο εύρος καθώς οι επιτρεπτές τιμές του μπορεί, ανάλογα με την εφαρμογή, να ξεκινούν ακόμα και από το 0.5. Αυτό συμβαίνει γιατί κατά την διαδικασία της ανάλυσης του αισθήματος μιας κριτικής υπάρχουν ορισμένοι κίνδυνοι, όπως:

- Σαρκασμός – Ειρωνεία

Η ειρωνεία είναι ένας από τους εξαιρετικά δύσκολα ανιχνεύσιμους τρόπος έκφρασης και συχνά προκαλεί σύγχυση στα μοντέλα ανάλυσης αισθήματος. Αυτό συμβαίνει καθώς για να αποτυπώσουν ειρωνεία οι άνθρωποι χρησιμοποιούν λέξεις εννοιολογικά αντίθετες από το νόημα που στοχεύουν να αποδώσουν. Ορισμένες φορές ο σαρκασμός δυσκολεύει ακόμα και τους ίδιους τους ανθρώπους να τον κατανοήσουν. Έτσι λοιπόν, αν στο υπό μελέτη κείμενουπάρχουν πολλές σαρκαστικές προτάσεις είναι πολύ πιθανό να αποπροσανατολισθεί ο αλγόριθμος και να έχουμε πτώση στο ποσοστό ακρίβειάς του. *Βλέπε παράδειγμα* ειρωνεία και σαρκασμός σελ.51.

- Διαφορετικές μορφές άρνησης

Μια νοηματική άρνηση μπορεί να εκφραστεί με διαφορετικούς τρόπους. Για παράδειγμα στην πρόταση “The show was not interesting” η άρνηση είναι ξεκάθαρη και βρίσκεται αμέσως πριν τη λέξη που αφορά. Αντίθετα, στη περίπτωση “I do not call this film a comedy movie” η άρνηση αντιστοιχεί στην λέξη comedy, η οποία όμως βρίσκεται στο τέλος της πρότασης, πολύ μακριά της και αυτό, ενώ για κάποιον άνθρωπο που το διαβάζει δεν του δημιουργεί έντονο προβληματισμό, μπορεί να οδηγήσει ένα μοντέλο ανάλυσης αισθήματος σε λάθος κατηγοριοποίηση.

- Αμφιλεγόμενες λέξεις

Υπάρχουν λέξεις που το νόημα, το οποίο αποδίδουν, δεν είναι πάντα ξεκάθαρο και μάλιστα δεν είναι πάντοτε το ίδιο, αλλά ποικίλει ανάλογα και με το υπόλοιπο κείμενο που συνδυάζονται. Η δυσκολία σε αυτή την περίπτωση είναι για έναν αλγόριθμο να κατηγοριοποιήσει ένα κείμενο που περιλαμβάνει μια

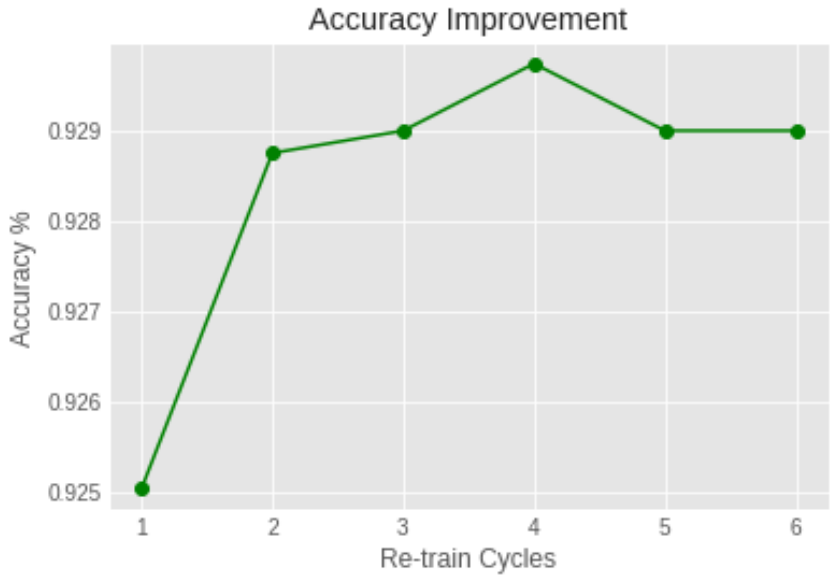
τέτοια λέξη πχ ως θετικό ενώ σε προηγούμενη περίπτωση έχει κατηγοριοποιήσει αντίστοιχο κείμενο ως αρνητικό.

- Συνδυασμός πολικοτήτων

Μια πρόταση μπορεί να περιέχει εναλλαγές αισθήματος όσον αφορά διαφορετικά χαρακτηριστικά. Με δεδομένο ότι απουσιάζουν οι συντελεστές βαρύτητας των χαρακτηριστικών, η προσπάθεια γενικής κατηγοριοποίησής του συνολικού προϊόντος καθίσταται ιδαίτερα δύσκολη. Ένα τέτοιο παράδειγμα είναι το εξής :
“The audio quality of my new laptop is so cool but the display colors are not too good.”

6.2 Αποτελέσματα ακρίβειας

Παρόλα αυτά, στην προκειμένη περίπτωση, λόγω της επιλογής κατάλληλου αλγορίθμου και παραμέτρων βελτιστοποίησης, για την δυική κατηγοριοποίηση σε θετικό – αρνητικό (απουσιάζει δηλαδή η κλάση ουδέτερο) και κυρίως της κατάλληλης επιλογής δεδομένων προς εκπαίδευση, καταφέραμε να πετύχουμε πολύ υψηλό ποσοστό ακρίβειας, της τάξεως του **0.925**.
Αυτό αποτελεί μόνο την αρχική εκτίμηση στα δεδομένα μας, καθώς, όπως έχει ήδη αναλύθει, χρησιμοποιήθηκε η τεχνική αυτοεκπαίδευσης (self training), με την οποία εκπαιδεύτηκε περισσότερες φορές το μοντέλο, τροφοδοτώντας το κάθε φορά με τις κριτικές που στον προηγούμενο κύκλο είχαν προβλεφθεί με ακρίβεια μεγαλύτερη από μία σταθερά c και που έχουν βαθμολογηθεί με 1 ή 2 αστέρια για αρνητικές προβλεψεις και με 4 ή 5 αστέρια για θετικές. Σε αυτή την περίπτωση η σταθερά c είναι ίση με 0.9999. Ουσιαστικά, με αυτόν τον τρόπο διευρύνεται κάθε φορά το μέγεθος των δεδομένων προς εκπαίδευση, καθώς ανατροφοδοτείται ο αλγόριθμος με τις κριτικές που έχουν προβλεφθεί με μεγάλη βεβαιότητα, βελτιώνοντας έτσι ολοένα και περισσότερο την απόδοσή του. Κατί τέτοιο είναι εμφανές στο διάγραμμα που παρουσιάζεται παρακάτω και μας δείχνει τη βελτίωση της απόδοσης του μοντέλου για μέγεθος δεδομένων εκπαίδευσης 3.000 εγγραφές, μέγεθος δεδομένων test 100.000 και μεταβλητή κατωφλίου c=0.9999



Το γράφημα δημιουργήθηκε χρησιμοποιώντας την βιβλιοθήκη **matplotlib** της python.
Ένα ακόμα σημείο που αξίζει να αναφερθεί έιναι η σύγκλιση σε μία τιμή που παρουσιάζει η συγκεκριμένη τεχνική μετά το περάς ορισμένου αριθμού κύκλων επανεκπαίδευσης και μας δείχνει πως έπειτα από αυτό το σημείο ανεξάρτητα από το πλήθος των κριτικών που θα τροφοδοτήσουμε και την ακρίβειά τους δεν μπορούμε να πετύχουμε περαιτέρω βελτίωση.

6.3 Λοιπές μετρικές αξιολόγησης και τα αποτελέσματά τους

Ωστόσο, πέρα από την απλή μέτρηση της μετρικής accuracy των προβλέψεων του μοντέλου μας, που ουσιαστικά συγκρίνει την προβλεπόμενη κλάση με την πραγματική και αυτό το εκφράζει ως ένα συνολικό ποσοστό, υπάρχουν και κάποια ακόμα μεγέθη που μπορούν να δώσουν μια εικόνα απόδοσης του αλγορίθμου. Πριν την ανάλυσή τους, κρίνεται απαραίτητο να αναφερθούν οι παρακάτω έννοιες:

- Αληθώς Θετικό (True Positive,TP)

Υποθέτοντας ότι η μία από τις δύο κλάσεις κατηγοριοποίησης αντιστοιχεί στη θετική τιμή 1, TP είναι όσες έχουν προβλεφθεί ότι ανήκουν στη θετική κλάση και όντως η πραγματική τους τιμή είναι 1.

- Αληθώς Αρνητικό (True Negative,TN)

Ακολουθώντας την παραπάνω υπόθεση, TN είναι οι σωστά προβλεπόμενες για την αρνητική κλάση -1.

- Ψευδώς Θετικό (False Positive, FP)

Σε αυτή την κατηγορία ανήκουν οι προβλέψεις που ανήκουν στη θετική κλάση 1, ενώ στην πραγματικότητα ανήκουν στην κλάση -1.

- Ψευδώς Αρνητικό (False Negative, FN)

Σε πλήρη αντιστοίχιση εδώ ανήκουν οι προβλέψεις που έχουν λανθασμένα προσδιορισθεί ως αρνητικές, ενώ στην πραγματικότητα ανήκουν στην κλάση 1. . *Βλέπε παράδειγμα* συμφραζόμενα σελ.51.

Οι μετρικές απόδοσης πέραν του accuracy είναι:

- Precision

Αντιστοιχεί στον λόγο όσων έχουν προβλεφθεί σωστά στην θετική κλάση (TP) προς όλων όσων έχουν προβλεφθεί στην θετική κλάση (TP + FP), είτε αυτή η πρόβλεψη είναι ορθή είτε είναι λανθασμένη. Ουσιαστικά μας δίνει μια ποσοστιαία έκφραση των σωστών προβλέψεων της κλάσης που μας ενδιαφέρει στο σύνολο των προβλέψεων για την κλάση αυτή.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

- Recall

Είναι το κλάσμα με αριθμητή τις ορθές προβλέψεις της θετικής κλάσης (TP) και παρονομαστή το σύνολο των παρατηρήσεων που είναι στην πραγματικότητα θετικά (TP + FN) (όχι έχουν προβλεφθεί θετικά). Πιο αναλυτικά, μας δείχνει πόσα προβέβλεψε σωστά το μοντέλο στο σύνολο όσων θα έπρεπε να είχε προβλέψει.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

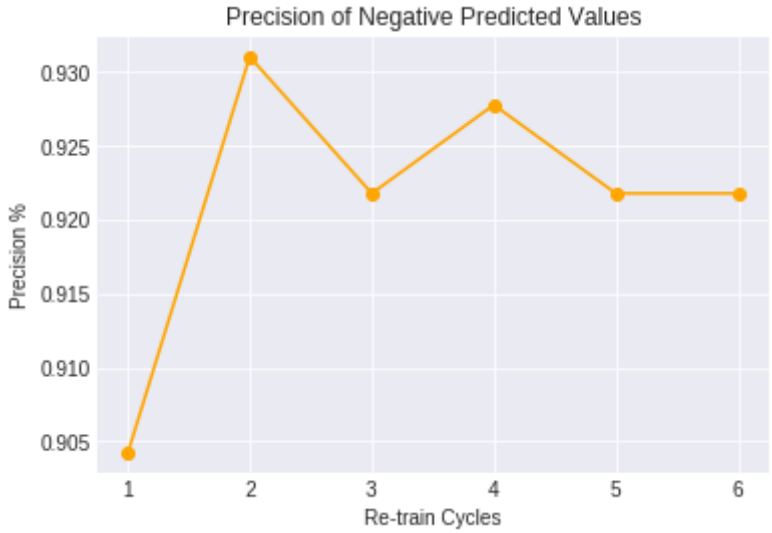
- F1 score

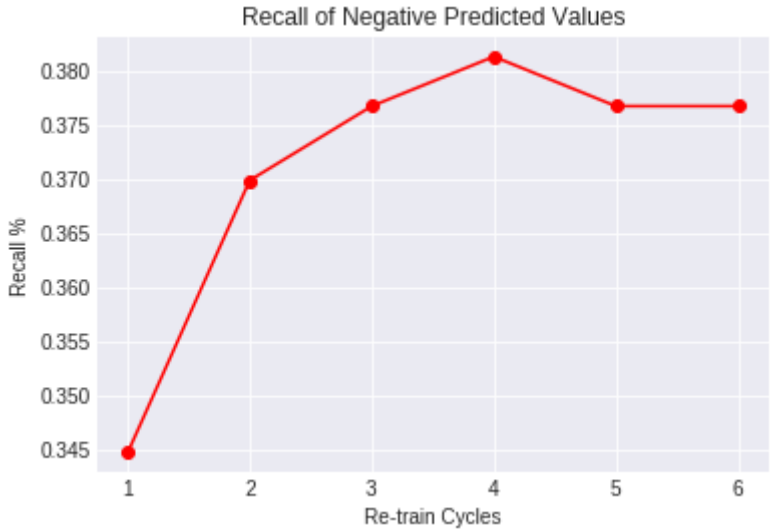
Εδώ λαμβάνεται υπόψη και το Precision και το Recall υπολογίζοντας, όχι απλά τον μέσο όρο τους αλλά τον αρμονικό μέσο τους ως εξής:

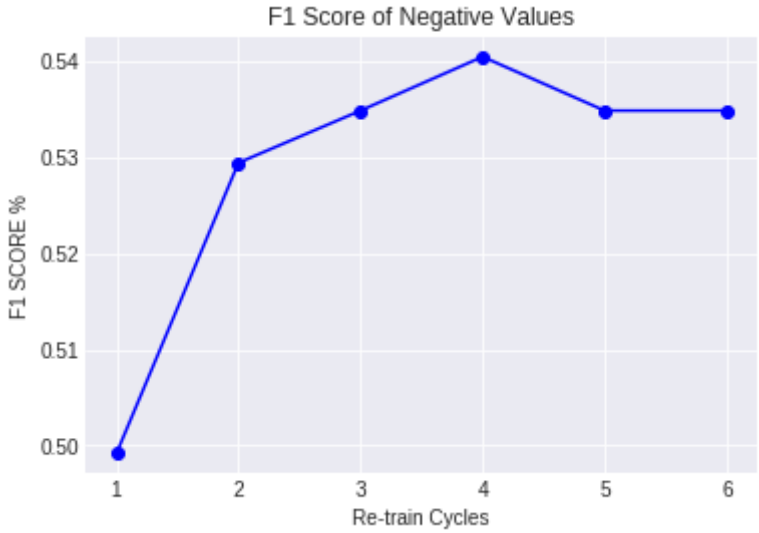
$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Οι τιμές των precision και recall λειτουργούν ως αντιστρόφως ανάλογα ποσά και είναι σχεδόν αδύνατο μια εφαρμογή να πετύχει πολύ υψηλές τιμές και στα δύο αυτά μεγέθη. Αναλόγως το είδος και τους στόχους της κάθε ανάπτυξης πρέπει να είναι υψηλό το αντίστοιχο μέγεθος. Υψηλό precision απαιτείται σε εφαρμογές που το κόστος της λανθασμένης πρόβλεψης στη θετική κλάση ή αλλιως FP είναι πολύ μεγάλο ή με άλλα λόγια ο αλγόριθμος θα πρέπει να είναι πολύ επιλεκτικός ως προς τις προβλέψεις του. Από την άλλη, υψηλή τιμή

recall χρειάζεται σε εφαρμογές που δίνουν πολύ μεγάλη σημασία στο να μην υπάρχουν λανθασμένες προβλεψεις (FN) για την αρνητική κλάση. Παράδειγμα τέτοιας εφαρμογής μπορεί να είναι η ανίχνευση απάτης στα τραπεζικά συστήματα, όπου μας ενδιαφέρει η μεγάλη ακρίβεια στην πρόβλεψη "ανωμαλιών" όπως η προσπάθεια απάτης.
Στη περίπτωση μας είναι πολύ σημαντικό οι προβλέψεις που κάνουμε για τις αρνητικά προσανατολισμένες κριτικές να είναι εξαιρετικά ακριβείς γιατί με βάση αυτές διαμορφώνουμε ένα σύνολο δεδομένων στο οποίο ανιχνεύουμε λέξεις κλειδιά και προσπαθούμε να εξάγουμε ένα νόημα που σχετίζεται με την αρντική κατεύθυνση των κριτικών. Έτσι, απαιτούμε υψηλή τιμή precision και όσο το δυνατόν μεγαλύτερη σιγουριά σε όσα προβλέπουμε ως αρνητικά. Αυτό φαίνεται ξεκάθαρα στα διαγράμματα που ακολουθούν.







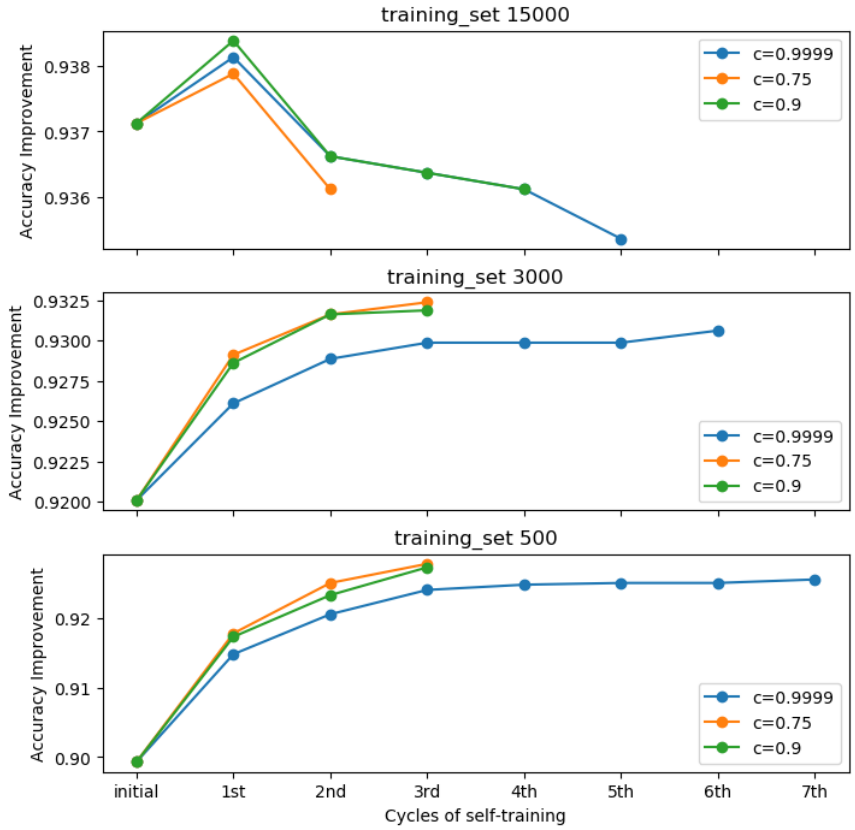
Επίσης εμφανής είναι η πολύ μεγάλη βελτίωση που παρατηρείται σε όλα αυτά τα στατιστικά μεγέθη μέσω της διαδικασία της αυτοεκπαίδευσης και η σύγκλιση σε μία τελική ανώτατη τιμή μετά το πέρας ορισμένου αριθμού κύκλων.

6.4 Συγκρίσεις αποτελεσμάτων

Στο προηγούμενο κεφάλαιο αναφέρθηκε ότι η μέθοδος του self train δέχεται εξωτερική παραμετροποίηση, έναν θετικό πραγματικό αριθμό c, του ποσοστού βεβαιότητας των προβλέψεων, τις οποίες και επιθυμούμε να εξάγουμε από τις συνολικές προβλέψεις για ανατροφοδότηση στο μοντέλο.
Παρουσιάστηκε και αναλύθηκε ο βέλτιστος συνδυασμός με c = 0.99 και με μέγεθος αρχικού train set = 3000. Στη συνέχεια θα δούμε κάποια συγκριτικά αποτελέσματα που προέκυψαν από την εφαρμογή του μοντέλου σε διαφορετικά μεγέθη αρχικών συνόλων εκπαίδευσης 500, 3000 και 15000, και διαφορετικών ποσοστών βεβαιότητας 0.75 , 0.9 και 0.99.

- **Accuracy**

Η πρώτη μετρική που εξετάζουμε είναι το accuracy του μοντέλου συναρτήσει των κύκλων self-training για κάθε μέγεθος training_set ξεχωριστά και συνδυαστικά με όλες τις πιθανές τιμές του c.



Από τις παραπάνω γραφικές γίνεται αντιληπτό ότι η συγκεκριμένη μετρική αυξάνει συνεχώς και συγκλίνει σε μία συγκεκριμένη τιμή με την πάροδο των κύκλων αυτοεκπαίδευσης. Αυτό βέβαια συμβαίνει όταν το μέγεθος του αρχικού συνόλου δεδομένων είναι αρκετά μικρότερο από το μέγεθος του συνόλου μη κατηγοριοποιημένων δεδομένων.

Στην πρώτη περίπτωση, κατά την οποία το training set είναι 15000, ενώ στον πρώτο κύκλο αυτοεκπαίδευσης είναι εμφανής η βελτίωση του μοντέλου , στη συνέχεια πέφτει κατακόρυφα και συνεχίζεται αυτή η συμπεριφορά μέχρι το πέρας της διαδικασίας. Αυτή η συμπεριφορά συναντάται μόνο σε αυτήν την περίπτωση καθώς η αρχική εκπαίδευση του μοντέλου στηρίζεται σε αρκετά μεγάλο μέγεθος ήδη κατηγοριοποιημένων δεδομένων, γεγονός που το εμποδίζει να γενικεύει, ώστε να μπορέσει να κατηγοριοποιήσει με καλή προσέγγιση το υπόλοιπο σύνολο. Εν ολίγοις, το σύστημα από τον πρώτο κύκλο μετεκπαίδευσης επιλέγει ένα πολύ μεγάλο retrain σύνολο , το οποίο έχει πολλά κοινά χαρακτηριστικά με το αρχικό labeled training set, με αποτέλεσμα στους επόμενους κύκλους να αποκτά ασθενή διακριτική ικανότητα για το λοιπό σύνολο που δεν επιλέχθηκε, αφού έχει και λιγότερα κοινά χαρακτηριστικά με αυτό στο οποίο εκπαιδεύεται.

Η πιο σωστή γενίκευση του μοντέλου σε διαφόρων ειδών σχόλια και μηνύματα συναντάται στις επόμενες δύο περιπτώσεις όπου το training set

αποτελείται από 3000 και 500 εγγραφές αντίστοιχα. Σε αυτές τις περιπτώσεις ο αλγόριθμος αυξάνει σταθερά την ακρίβειά του και μάλιστα διακρίνουμε ότι όταν το ποσοστό ακρίβειας είναι αρκετά μεγάλο, 0.99, τότε εξαντλεί την ικανότητα απόκτησης γνώσης από το συγκεκριμένο σύνολο μη χαρακτηρισμένων δεδομένων.

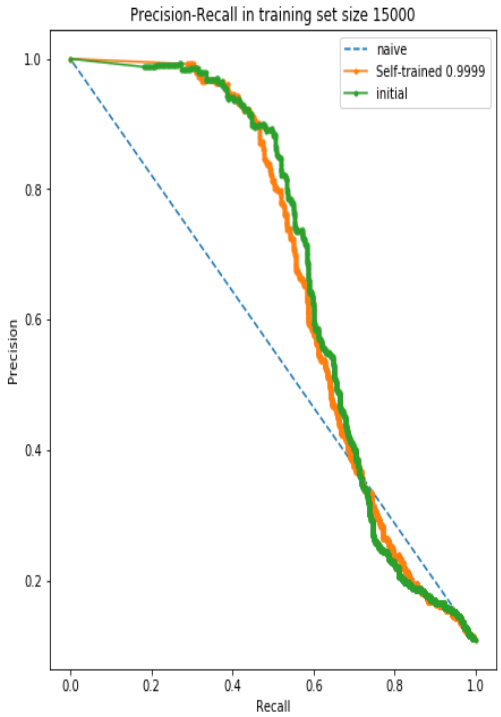
Συμπερασματικά, βλέπουμε ότι ακολουθείται η αρχή του self training, κατά την οποία έχει νόημα να εφαρμοστεί η συγκεκριμένη μέθοδος μόνο στην περίπτωση που έχουμε πολύ λίγα κατηγοριοποιημένα δεδομένα και μεγάλο όγκο αχαρακτήριστων.

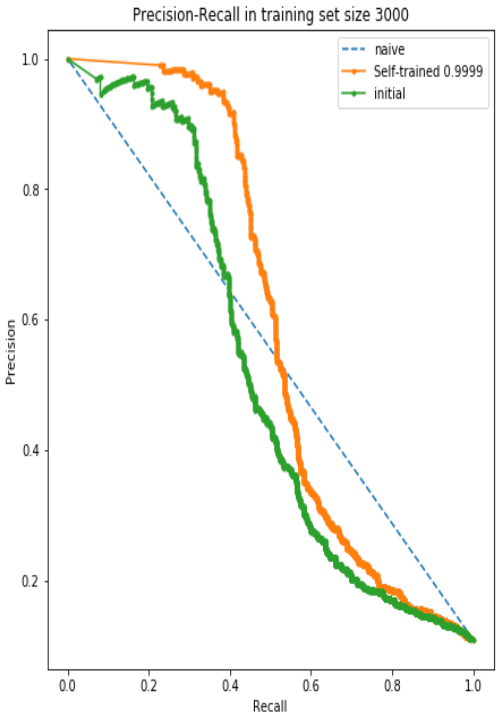
- **Γραφικές μέθοδοι αξιολόγησης**

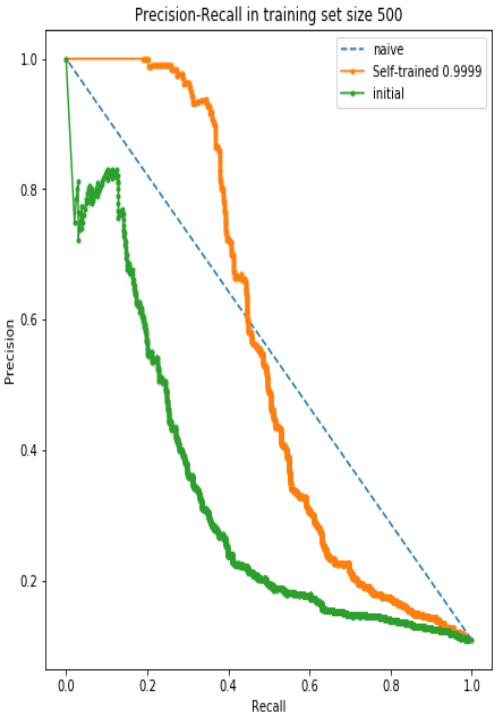
Πέραν των υπολογιστικών μετρικών αξιολόγησης ενός στατιστικού μοντέλου, υπάρχουν και γραφικές μέθοδοι οι οποίες αξιολογούν λαμβάνοντας υπ' όψιν περισσότερες λεπτομέρειες ένός σύνολου προβλέψεων. Δύο από αυτές, η precision-recall και η roc curve παρουσιάζοναι στη συνέχεια.

Precision recall curve

Οι συγκεκριμένες καμπύλες μετράνε και εμφανίζουν τον ρυθμό των σωστών θετικών προβλέψεων προς τις συνολικές θετικές προβλέψεις, ΄precision, καθώς και τον ρυθμό των σωστών προβλέψεων προς τις συνολικές θετικές εγγραφές, recall. Στο δικό μας πρόβλημα στον όρο θετική εγγραφή αντιστοιχούμε την κλάση που στοχεύουμε να διακρίνουμε σωστά , δηλαδή την αρνητική.



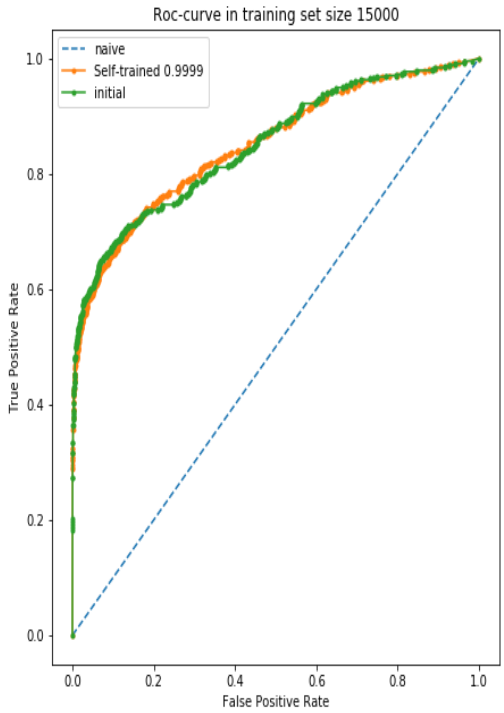


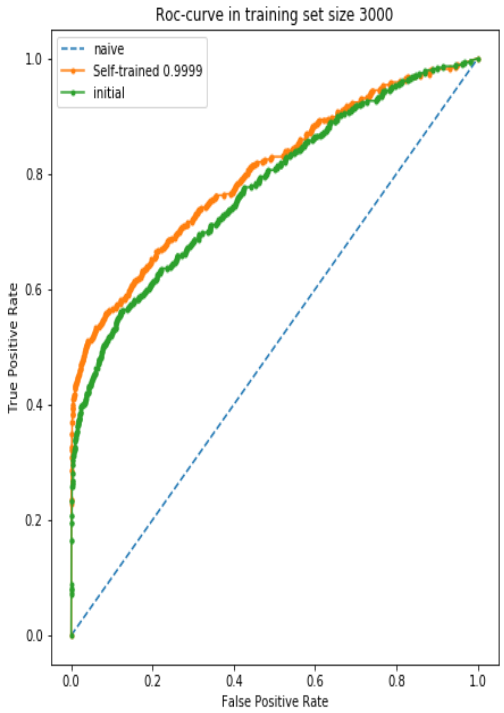


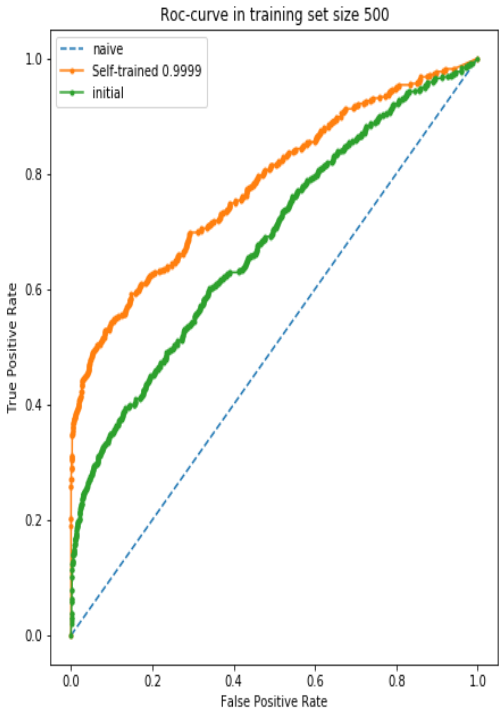
Οι παραπάνω γραφικές απεικονίζουν την precision recall curve που αξιολογεί την απόδοση του μοντέλου σε τρία διαφορετικά αρχικά σύνολα εκπάιδευσης. Η πράσινη καμπύλη είναι η απόδοση του μοντέλου εκπαιδευμένου αποκλειστικά στο αρχικό μικρό σετ κατηγοριοποιημένων δεδομένων, το οποίο μεταβάλλεται σε μέγεθος, και αποτελείται από 15000, 3000 και 500 εγγραφές αντίστοιχα για τις τρείς περιπτώσεις. Η πορτοκαλιά καμπύλη δείχνει την απόδοση του μοντέλου μετά το πέρας της διαδικασίας του self training όταν το ποσοστό ακρίβειας είναι 0.999, αφού διαπιστώσαμε ότι σε αυτό το ποσοστό λαμβάνουμε τη μέγιστη δυνατή ενίσχυση μάθησης από την αυτοεκπαίδευση. Εκτός από την πρώτη περίπτωση που το δείγμα εκπαίδευσης είναι αρκετά μεγάλο, 15000, στις υπόλοιπες περιπτώσεις είναι αισθητή η βελτίωση ικανότητας πρόβλεψης του μοντέλου. Βλέπουμε ότι η καμπύλη τείνει προς την άνω δεξιά γωνία του γραφήματος , όπου θεωρείται ότι βρίσκεται και το σημείο βέλτιστης απόδοσης ενός οποιουδήποτε συστήματος. Στην πρώτη γραφική παράσταση που δεν παρατηρείται αυτό το φαινόμενο το μοντέλο δεν έχει ενισχυθεί από τη διαδικασία του self training για τον ίδιο λόγο που αποτυγχάνει και η βελτίωση του accuracy του, όπως περιγράφηκε και στην προηγούμενη μετρική.

Roc curve

Ένας ακόμη γραφικός τρόπος απεικόνισης της επίδοσης ενός μοντέλου είναι ο σχεδιασμός της καμπύλης roc. Αναλυτικότερα, για τη συγκεκριμένη καμπύλη, απαιτείται η καταμέτρηση του ρυθμού των σωστών και λάθος προβλέψεων της κλάσης στόχου. Ταυτίζοντας την προαναφερόμενη κλάση στόχο με την αρνητική στο δικό μας πρόβλημα προχωρήσαμε στον υπολογισμό των περιγραφόμενων ρυθμών. Στη συνέχεια παρουσιάζονται τα αποτελέσματα σε τρείς διαφορετικές εφαρμογές του μοντέλου μας.







Διαπιστώνουμε ότι και σε αυτή τη μετρική η διαδικασία του self training ενισχυεί αισθητά την απόδοση του μοντέλου στο σύνολο των δειγμάτων αξιολόγησης. Όπως στις precision recall καμπύλες, όταν το σύνολο αρχικής εκπαίδευσης είναι 15000, τότε δεν υπάρχει ευδιάκριτη διαφορά μεταξύ της αρχικής και τελικής, μετά την αυτοεκπαίδευση, απόδοσης αλλά στις υπόλοιπες περιπτώσεις παρατηρείται σημαντική βελτίωση. Μάλιστα η καμπύλη κατευθύνεται προς το άνω αριστερό άκρο του γραφήματος, δηλαδή πλησιάζει τη μέγιστη δυνατή απόδοση ενός συστήματος, ανεξαρτήτως πεδίου εφαρμογής.

6.5 Αποτελέσματα της εξαγωγής σημαντικών λέξεων

Στο τελικό στάδιο της ανάπτυξης μας εξάγονται από τα ταξινομημένα ως αρνητικά σχόλια των χρηστών, οι λέξεις κλειδιά που τα χαρακτηρίζουν. Σε αυτή τη διαδικασία παρατηρούμε ότι δεν υπάρχει κάποιος εύκολος τρόπος αξιολόγησης, όμως μπορούμε εποπτικά να παρατηρήσουμε και να αναλύσουμε κάποιες κατηγορίες περιπτώσεων που το μοντέλο μας είχε διαφορετική συμπεριφορά.

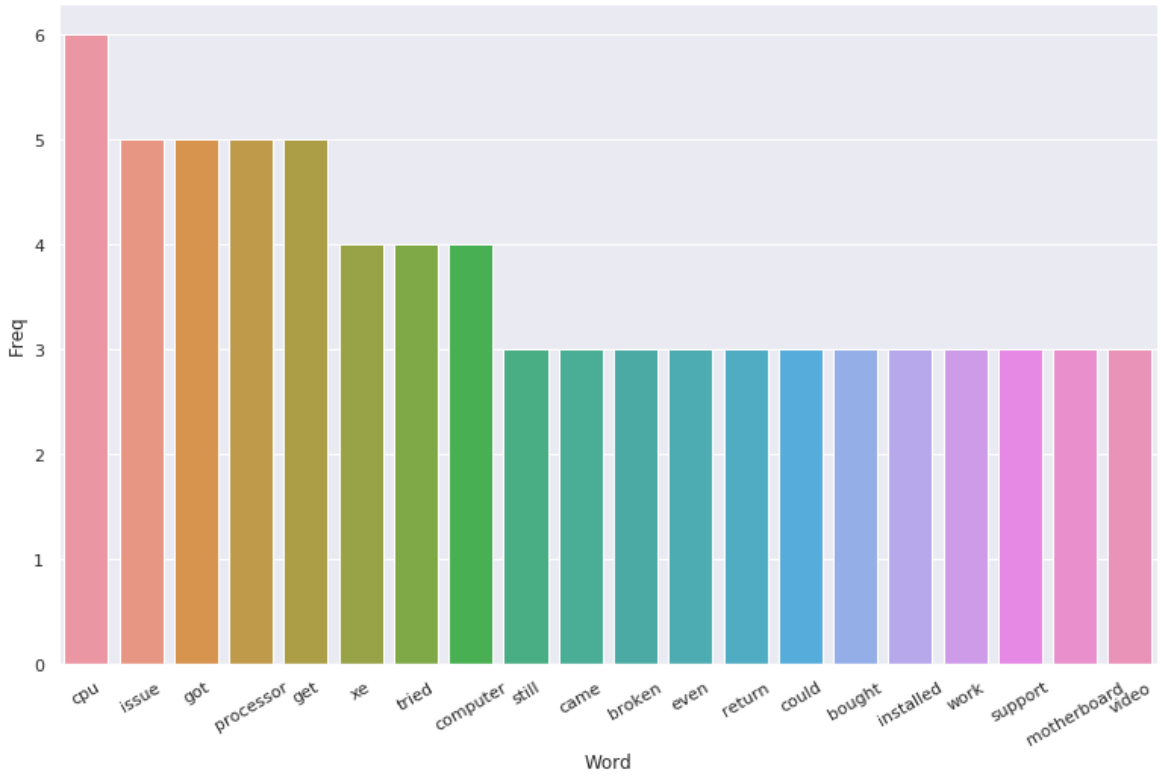
Μερικά από τα προϊόντα, στα οποία είχαν επιτυχή αποτελέσματα και ο countvectorizer, για κάθε n-gram, καθώς και ο td-idf :

1. AMD Ryzen 5 2600 Processor with Wraith Stealth Cooler - YD2600BBAFBOX

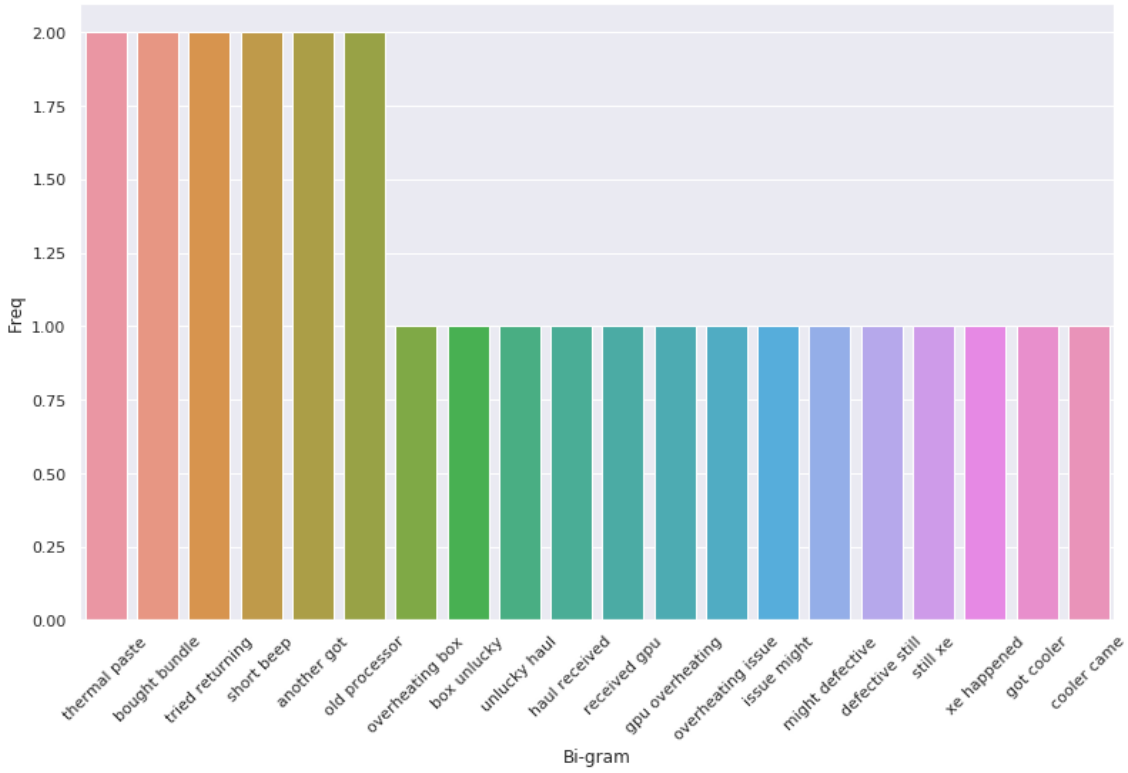
TF-IDF Keywords:

processor 0.379
heatsink came damaged 0.351
heatsink came 0.351
heatsink 0.351
damaged used 0.351

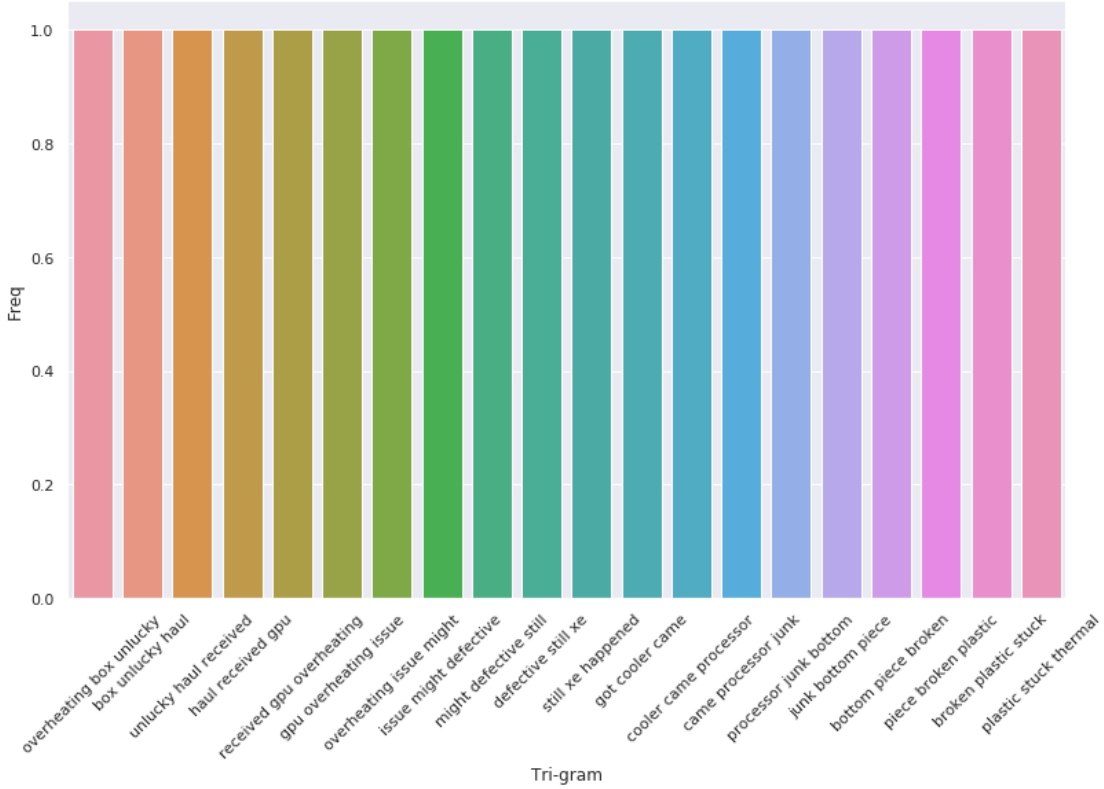
1-gram:


Freq
Word
0
1
2
3
4
5
6
cpu
issue
got
processor
get
xe
tried
computer
still
came
broken
even
return
could
bought
installed
work
support
motherboard
video

2-gram:


Freq
Bi-gram
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
thermal paste
bought bundle
tried returning
short beep
another got
old processor
overheating box
box unlucky
unlucky haul
haul received
received gpu
gpu overheating
overheating issue
issue might
might defective
defective still
still xe
xe happened
got cooler
cooler came

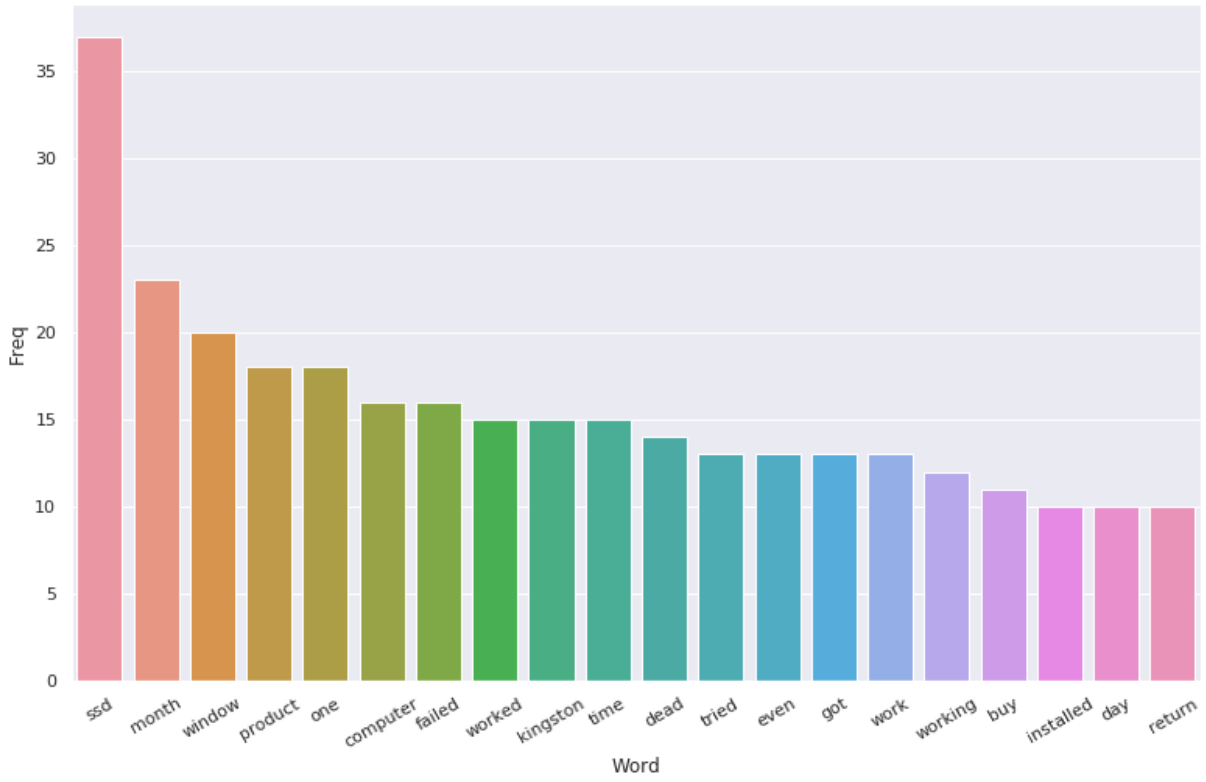
3-gram:



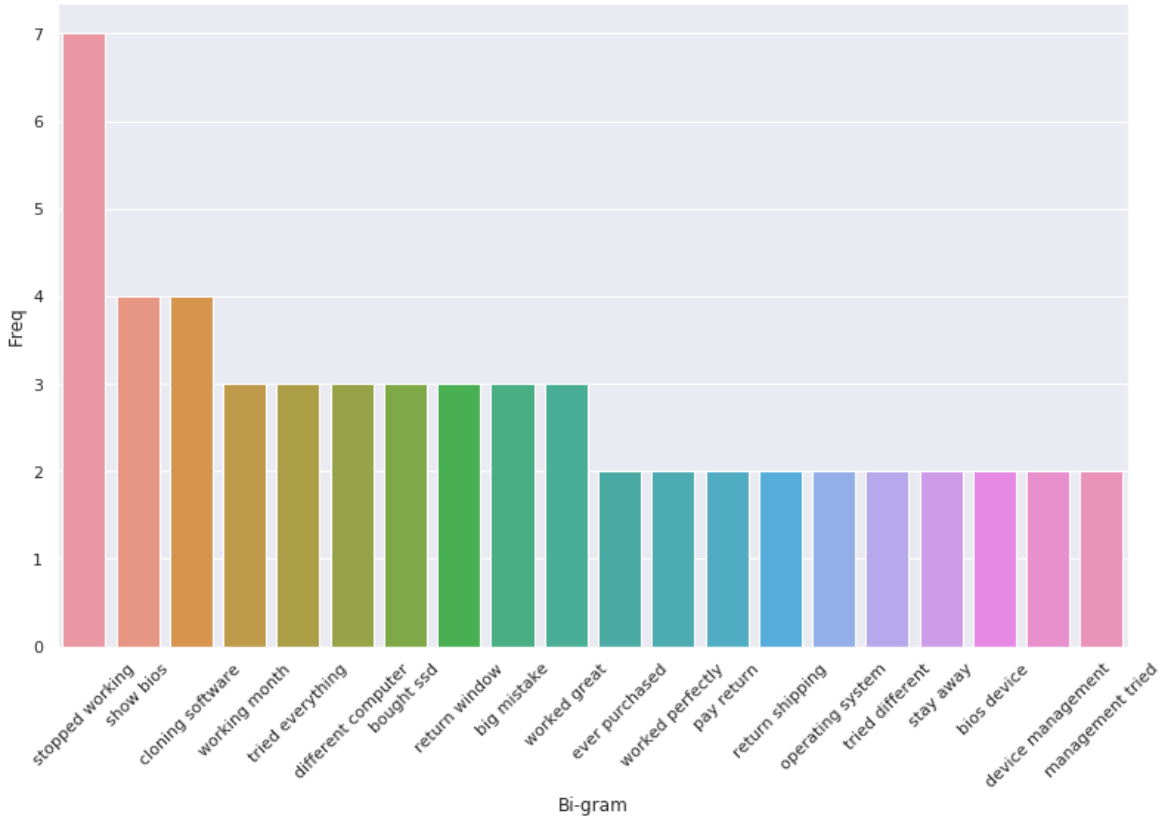
2. Kingston 240GB A400 SATA 3 2.5" Internal SSD SA400S37/240G - HDD Replacement for Increase Performance

TF-IDF Keywords:
doa 1.0
worked issue 0.72
dont buy 0.667
dead arrival 0.637
arrival 0.637

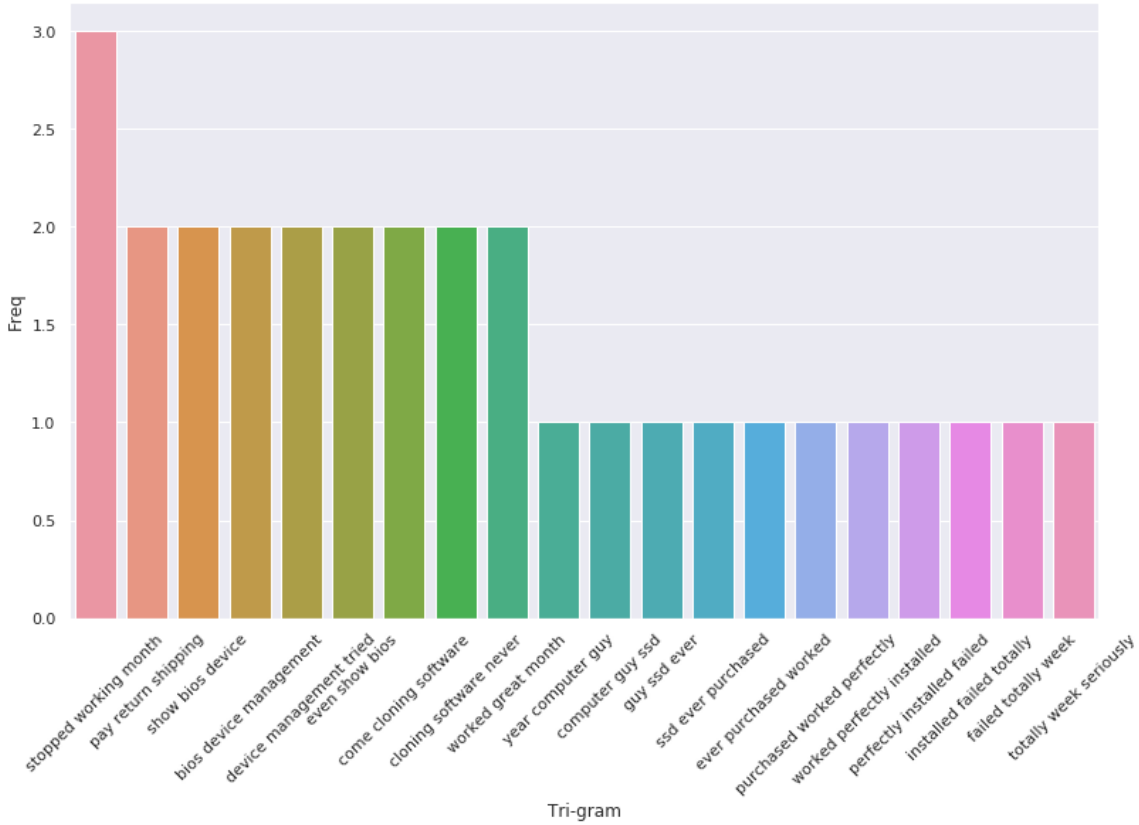
1-gram:


Freq
0
5
10
15
20
25
30
35
ssd
month
window
product
one
computer
failed
worked
kingston
time
dead
tried
even
got
work
working
buy
installed
day
return
Word

2-gram:


Freq
0
1
2
3
4
5
6
7
stopped working
show bios
cloning software
working month
tried everything
different computer
bought ssd
return window
big mistake
worked great
ever purchased
worked perfectly
pay return
return shipping
operating system
tried different
stay away
bios device
device management
management tried
Bi-gram

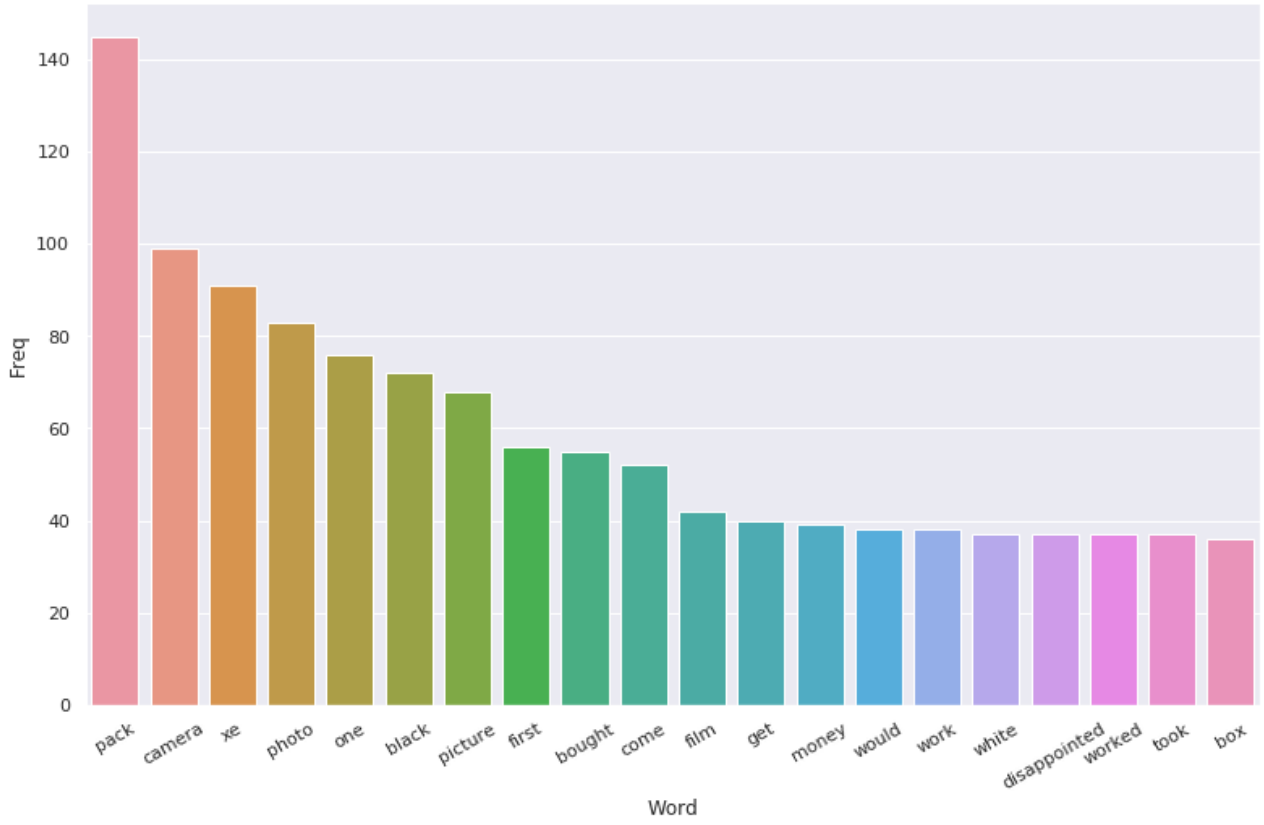
3-gram:



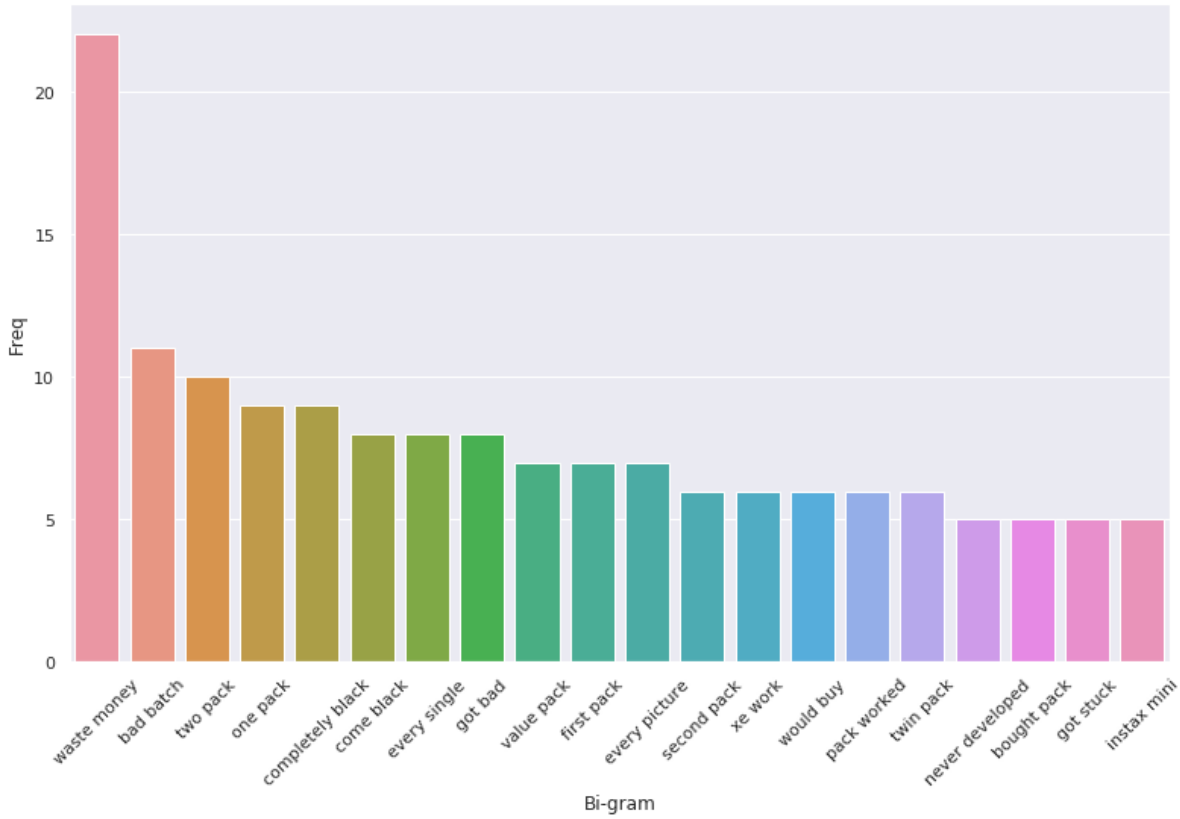
3. Fujifilm Instax Mini Instant Film Value Pack - (60 Total Pictures)(Package may vary)

TF-IDF Keywords:
old 1.0
faulty 1.0
pack bad 0.806
expensive develop 0.733
never problem 0.708

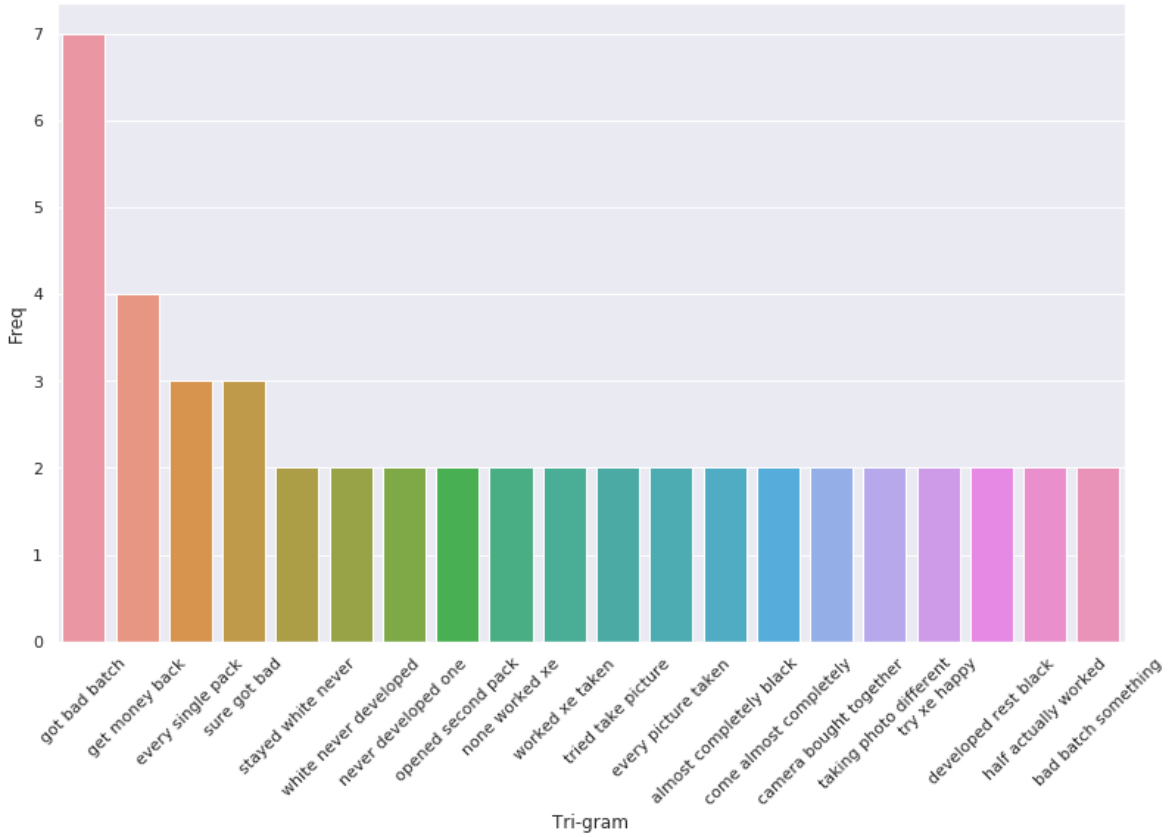
1-gram:


Freq
Word
0
20
40
60
80
100
120
140
pack
camera
xe
photo
one
black
picture
first
bought
come
film
get
money
would
work
white
disappointed
worked
took
box

2-gram:


Freq
Bi-gram
0
5
10
15
20
waste money
bad batch
two pack
one pack
completely black
come black
every single
got bad
value pack
first pack
every picture
second pack
xe work
would buy
pack worked
twin pack
never developed
bought pack
got stuck
instax mini

3-gram:

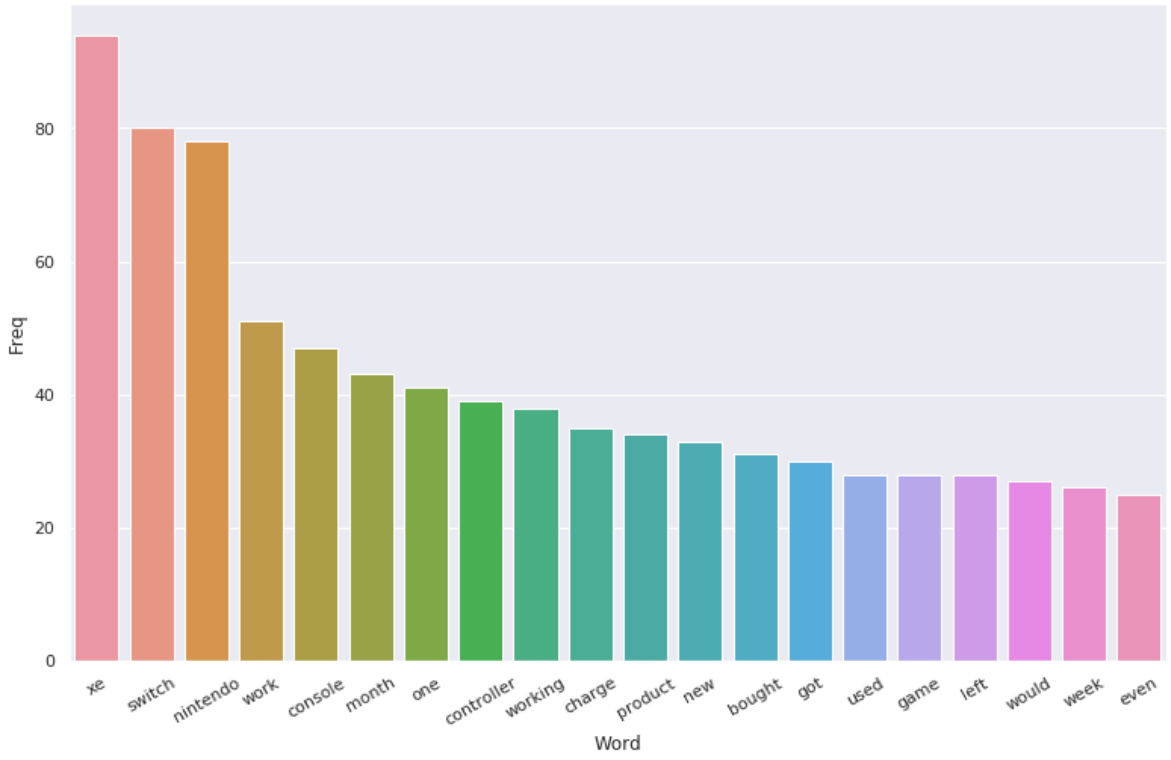


Στη συνέχεια θα δούμε κάποια από τα προϊόντα για τα οποία δεν επιλέχθηκαν ξεκάθαρες λέξεις κλειδιά. Σε αντίθεση με τα προηγούμενα παράδειγματα, πρόκειται για περιπτώσεις που η γενική εικόνα τους είναι ουδέτερη και συνεπώς δεν μπορούμε να διακρίνουμε εμφανή ελαττώματα από τα επιλεγμένα keywords.

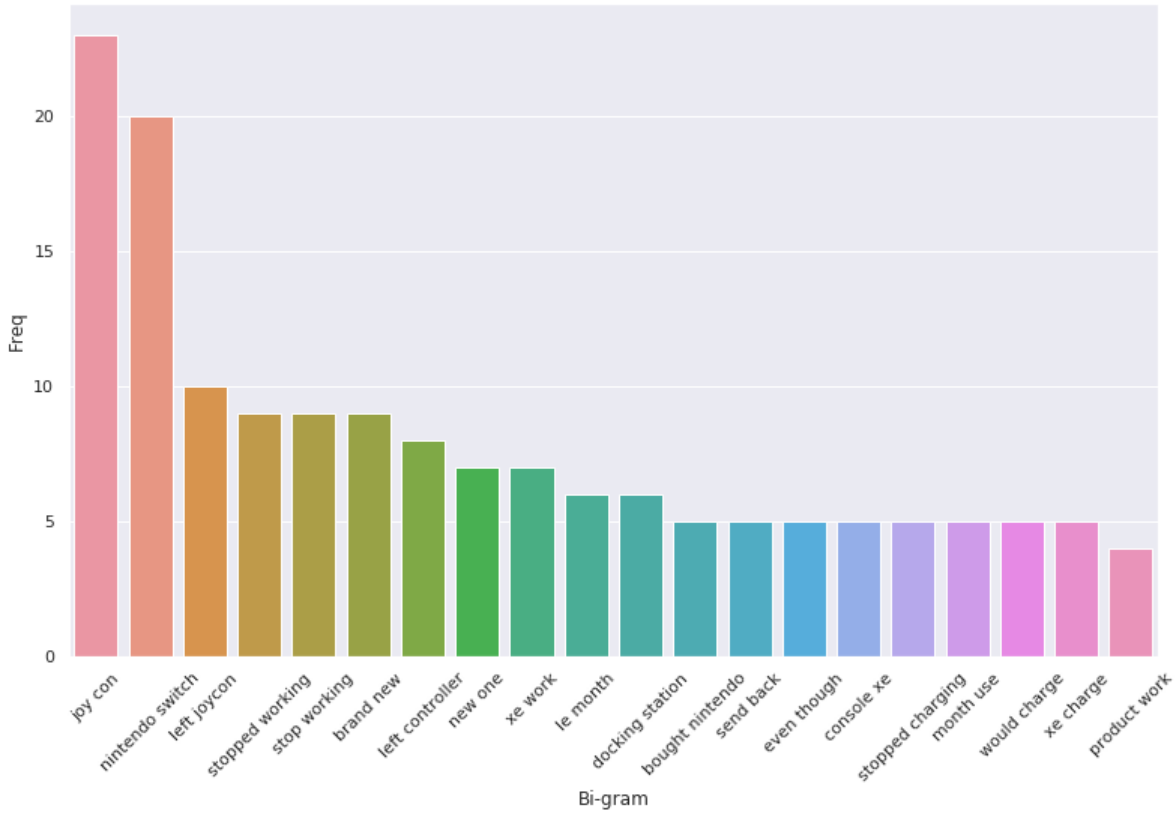
4. Nintendo Switch - Gray Joy-Con - HAC 001 (Discontinued by Manufacturer)

TF-IDF Keywords:
used 1.0
rip 1.0
order 1.0
junk 1.0
fragile 1.0

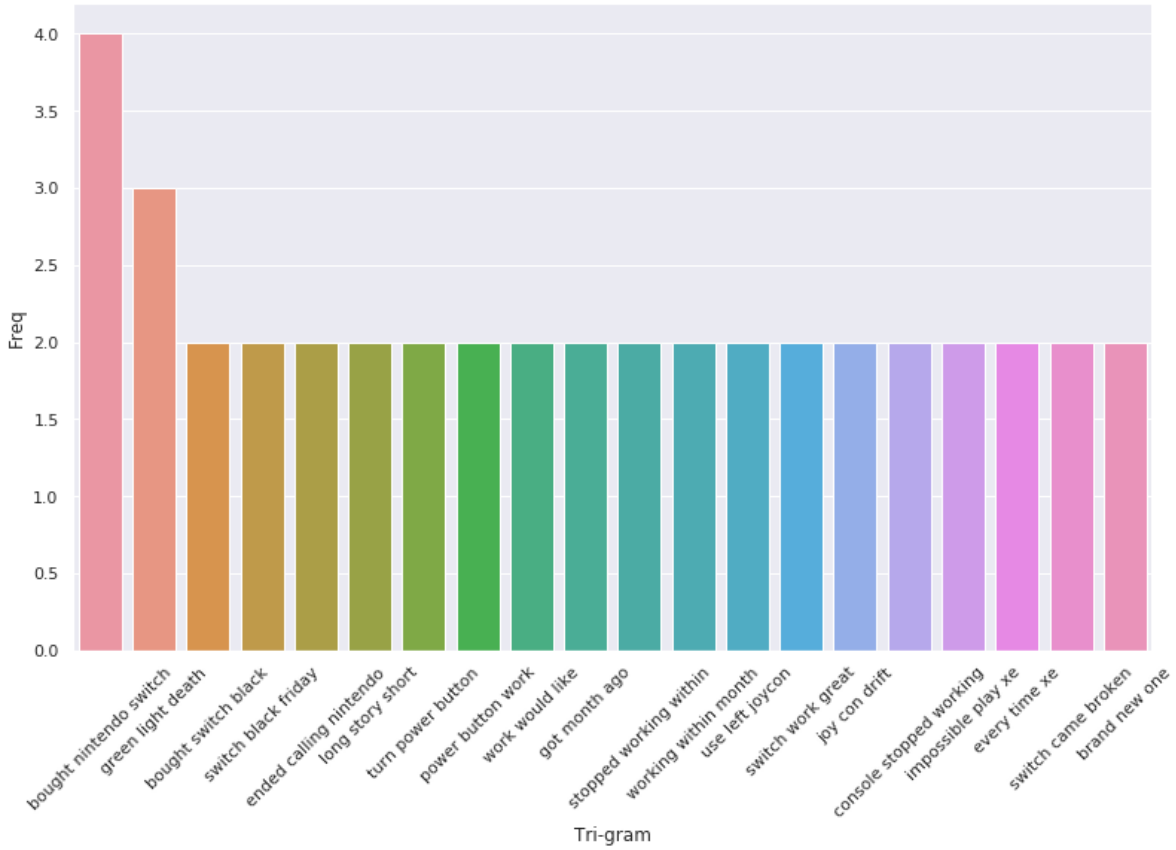
1-gram:


Freq
0
20
40
60
80
xe
switch
nintendo
work
console
month
one
controller
working
charge
product
new
bought
got
used
game
left
would
week
even
Word

2-gram:


Freq
0
5
10
15
20
joy con
nintendo switch
left joycon
stopped working
stop working
brand new
left controller
new one
xe work
le month
docking station
bought nintendo
send back
even though
console xe
stopped charging
month use
would charge
xe charge
product work
Bi-gram

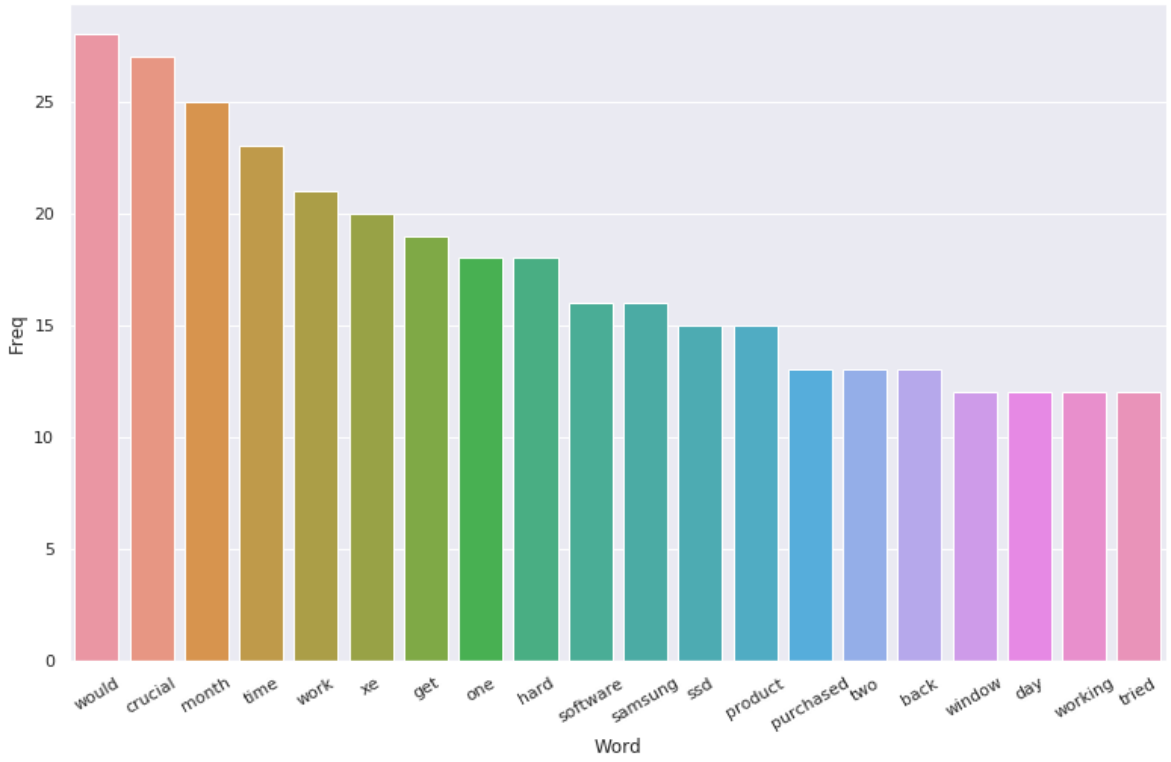
3-gram:



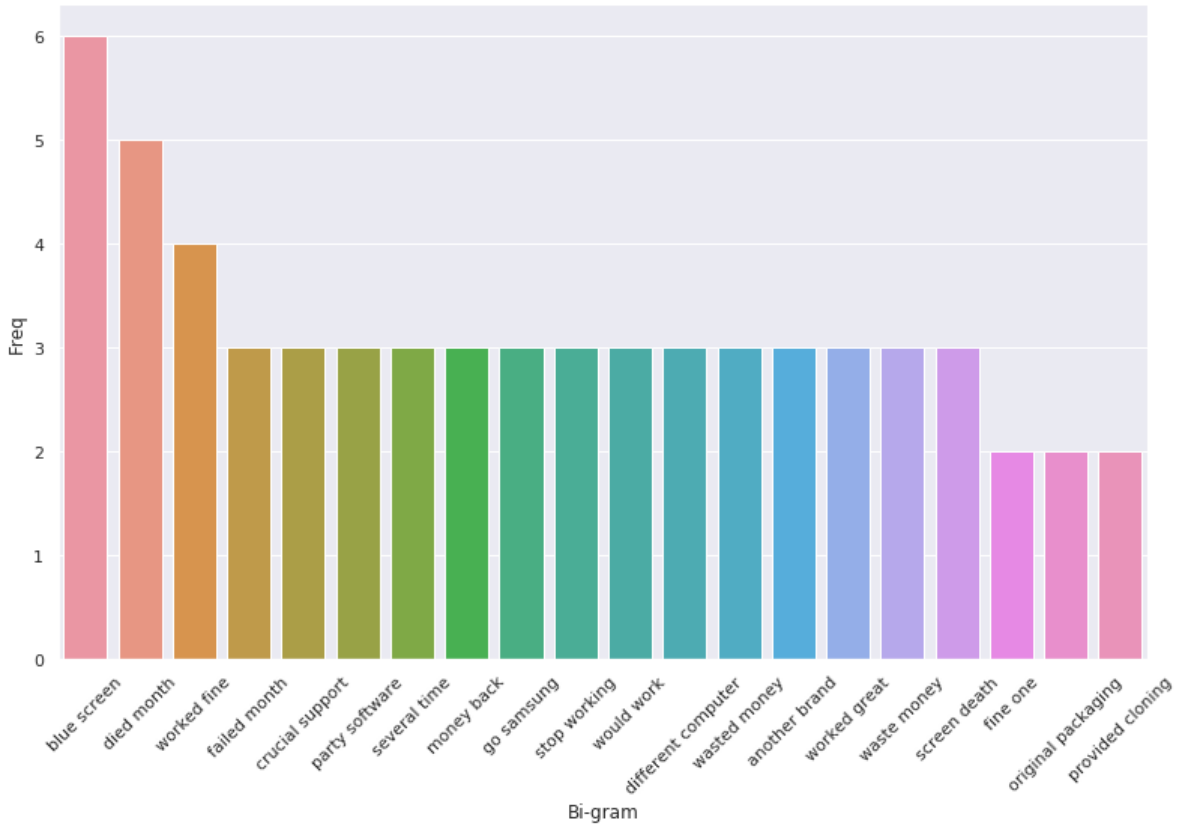
Στο συγκεκριμένο προϊόν αντιλαμβανόμαστε από τον tf-idf ότι υπήρξε πρόβλημα στην μεταφορά και την αξιοπιστία του παρόχου του προϊόντος καθώς μερικές από τις λέξεις κλειδιά στις οποίες κατέληξε είναι τα used rip και fragile. Όμως ο countvectorizer δεν κατάφερε να εξάγει συγκεκριμένα ελαττώματα, φράσεις με αρκετά μεγάλη συχνότητα εμφάνισης, γεγονός που καθιστά τα n-grams αρκετά γενικευμένα και ουδέτερου ύφους.

5. Crucial MX500 500GB 3D NAND SATA 2.5 Inch Internal SSD - CT500MX500SSD1

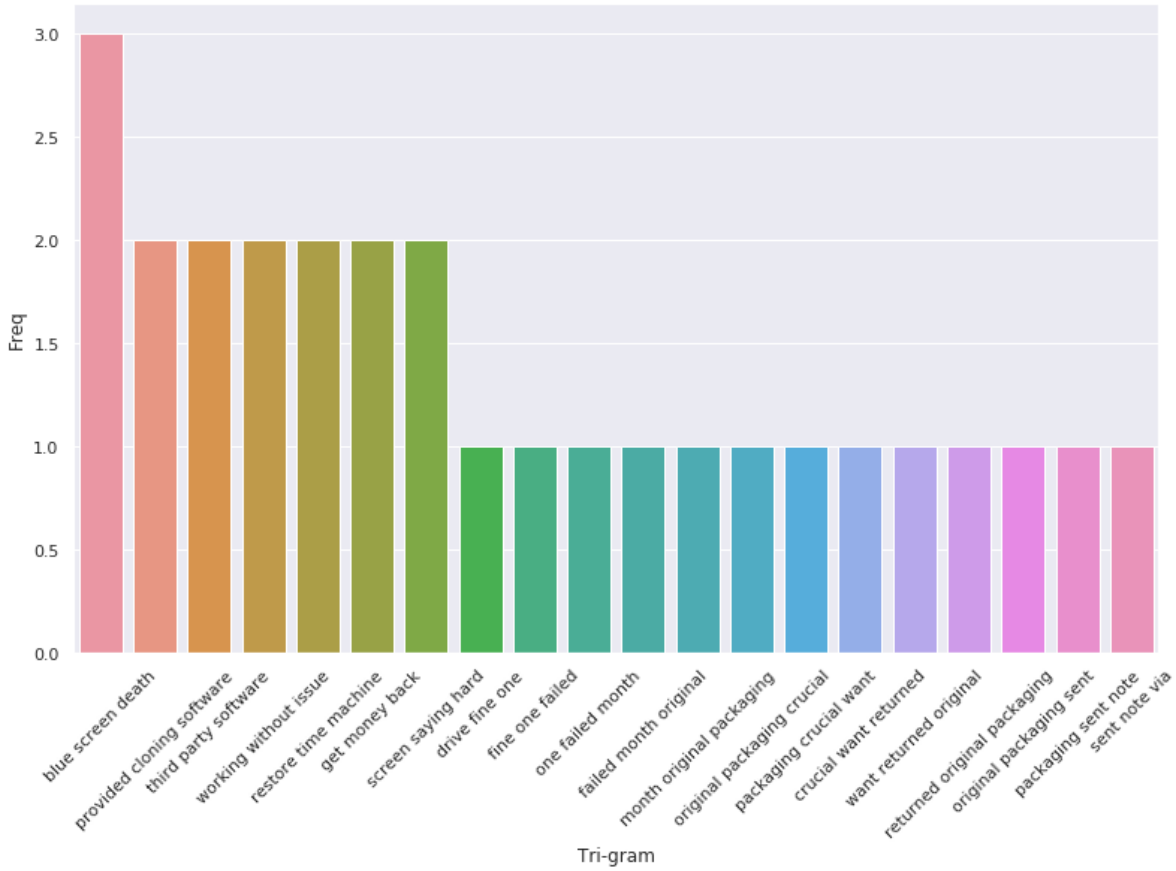
TF-IDF Keywords:
nothing recognize 0.667
xe 0.579
nothing 0.566
recognize 0.484
work dud 0.482
1-gram:


Freq
0
5
10
15
20
25
would
crucial
month
time
work
xe
get
one
hard
software
samsung
ssd
product
purchased
two
back
window
day
working
tried
Word

2-gram:


Freq
0
1
2
3
4
5
6
blue screen
died month
worked fine
failed month
crucial support
party software
several time
money back
go samsung
stop working
would work
different computer
wasted money
another brand
worked great
waste money
screen death
fine one
original packaging
provided cloning
Bi-gram

3-gram:



Αντίστοιχα σε αυτή την περίπτωση βλέπουμε μια αντίστοιχη συμπεριφορά αλλά με ουδέτερα αποτελέσματα στον tdf-idf και κατανοητά keywords από τον countvectorizer, καθώς στα n-grams είναι φανερό ότι υπάρχει ένα blue screen πρόβλημα στο προϊόν.

Βιβλιογραφία

[1] F. Xia, «Semi-supervised learning».

[2] M. Sigdel, I. Dinc, S. Dinc, M. S. Sigdel, M. L. Pusey και R. S. Aygun, «Evaluation of semi-supervised learning for classification of protein crystallization imagery,» σε *Conference Proceedings - IEEE SOUTHEASTCON*, 2014.

[3] I. Livieris, A. Kanavos, V. Tampakas και P. Pintelas, «An Auto-Adjustable Semi-Supervised Self-Training Algorithm,» *Algorithms,* τόμ. 11, αρ. 9, p. 139, 14 9 2018.

[4] D. Iliadis, «DRUG-TARGET INTERACTION PREDICTION USING IMBALANCE AWARE MULTI-LABEL METHODS,» 2019.

[5] «What Is Elasticsearch? (And Why You Need to Be Using It) - DZone Big Data,» [Ηλεκτρονικό]. Available: https://dzone.com/articles/what-is-elasticsearch-and-how-it-can-be-useful.

[6] «Understanding Random Forest - Towards Data Science,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

[7] «Types of Machine Learning Algorithms You Should Know,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861.

[8] «Types of Machine Learning Algorithms with Use Cases Examples | upGrad blog,» [Ηλεκτρονικό]. Available: https://www.upgrad.com/blog/types-of-machine-learning-algorithms/.

[9] «Top 10 Python Libraries You Must Know In 2019 | Edureka,» [Ηλεκτρονικό]. Available: https://www.edureka.co/blog/python-libraries/.

[10] «Text Classification: a comprehensive guide to classifying text with machine learning,» [Ηλεκτρονικό]. Available: https://monkeylearn.com/text-classification/.

[11] «Support Vector Machine — Introduction to Machine Learning Algorithms,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[12] «Supervised vs Unsupervised Learning: Key Differences,» [Ηλεκτρονικό]. Available: https://www.guru99.com/supervised-vs-unsupervised-learning.html.

[13] «Supervised and Unsupervised Machine Learning Algorithms,» [Ηλεκτρονικό]. Available: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/.

[14] «Scrapy 1.8 documentation — Scrapy 1.8.0 documentation,» [Ηλεκτρονικό]. Available: https://docs.scrapy.org/en/latest/.

[15] «Regression — explained in simple terms!! - Towards Data Science,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/regression-explained-in-simple-terms-dccbcad96f61.

[16] «Quotes about Python | Python.org,» [Ηλεκτρονικό]. Available: https://www.python.org/about/quotes/.

[17] «Python (programming language) - Wikipedia,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Python_(programming_language).

[18] «Predicting Polarity of User Reviews - Towards Data Science,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/predicting-polarity-of-user-reviews-8774c2a83dd3.

[19] «Naive Bayes in Machine Learning - Towards Data Science,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/naive-bayes-in-machine-learning-f49cc8f831b4.

[20] «Microsoft Azure Notebooks,» [Ηλεκτρονικό]. Available: https://notebooks.azure.com/help/jupyter-notebooks.

[21] «Metrics to Evaluate your Machine Learning Algorithm,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.

[22] «Machine Learning Types and Algorithms - Towards Data Science,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/machine-learning-types-and-algorithms-d8b79545a6ec.

[23] «Machine Learning over 1M hotel reviews finds interesting insights,» [Ηλεκτρονικό]. Available: https://monkeylearn.com/blog/machine-learning-hotel-reviews-insights/.

[24] «Machine Learning - GeeksforGeeks,» [Ηλεκτρονικό]. Available: https://www.geeksforgeeks.org/machine-learning/.

[25] «Keyword Extraction: a comprehensive guide to extracting keywords from text,» [Ηλεκτρονικό]. Available: https://monkeylearn.com/keyword-extraction/.

[26] «Introduction to Logistic Regression - Towards Data Science,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148.

[27] «How Many Products Does Amazon Sell? – April 2019,» [Ηλεκτρονικό]. Available: https://www.scrapehero.com/number-of-products-on-amazon-april-2019/.

[28] «Everything There Is to Know about Sentiment Analysis,» [Ηλεκτρονικό]. Available: https://monkeylearn.com/sentiment-analysis/.

[29] «Essentials of Machine Learning Algorithms (with Python and R Codes),» [Ηλεκτρονικό]. Available: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/.

[30] «Elasticsearch as a NoSQL Database | Elastic Blog,» [Ηλεκτρονικό]. Available: https://www.elastic.co/blog/found-elasticsearch-as-nosql.

[31] «Clustering — Unsupervised Learning - Towards Data Science,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/clustering-unsupervised-learning-788b215b074b.

[32] «Classification Algorithms | Types of Classification Algorithms | Edureka,» [Ηλεκτρονικό]. Available: https://www.edureka.co/blog/classification-algorithms/.

[33] «BeginnersGuide/Overview - Python Wiki,» [Ηλεκτρονικό]. Available: https://wiki.python.org/moin/BeginnersGuide/Overview.

[34] «Architecture overview — Scrapy 1.8.0 documentation,» [Ηλεκτρονικό]. Available: https://doc.scrapy.org/en/latest/topics/architecture.html.

[35] «An Overview on Elasticsearch and its usage - Towards Data Science,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/an-overview-on-elasticsearch-and-its-usage-e26df1d1d24a.

[36] «Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more,» [Ηλεκτρονικό]. Available: https://www.amazon.com/.

[37] «Amazon (company) - Simple English Wikipedia, the free encyclopedia,» [Ηλεκτρονικό]. Available: https://simple.wikipedia.org/wiki/Amazon_(company).

[38] «7 Types of Classification Algorithms - Analytics India Magazine,» [Ηλεκτρονικό]. Available: https://analyticsindiamag.com/7-types-classification-algorithms/.

[39] «40 Powerful Online Shopping and Ecommerce Statistics - Selz,» [Ηλεκτρονικό]. Available: https://founderu.selz.com/40-online-shopping-ecommerce-statistics-know/.

[40] «10 Machine Learning Methods that Every Data Scientist Should Know,» [Ηλεκτρονικό]. Available: https://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9.

[41] «Stemming and Lemmatization in Python (article) - DataCamp,» [Ηλεκτρονικό]. Available: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python.

[42] «JSON - Βικιπαίδεια,» [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/JSON.

[43] «Get to Know Azure | Microsoft Azure,» [Ηλεκτρονικό]. Available: https://azure.microsoft.com/en-in/overview/.

[44] X. Zhu, «Semi-Supervised Learning Literature Survey».

[45] T. Hastie, R. Tibshirani και J. Friedman, «Unsupervised Learning,» Springer, New York, NY, 2009, pp. 485-585.

[46] M. Mohri, A. Rostamizadeh και A. Talwalkar, «Foundations of Machine Learning».

[47] P. Gonçalves, M. Araújo, F. Benevenuto και M. Cha, «Comparing and combining sentiment analysis methods,» σε *COSN 2013 - Proceedings of the 2013 Conference on Online Social Networks*, New York, New York, USA, 2013.

[48] J. Su, J. Sayyad-Shirabad και S. Matwin, «Large Scale Text Classification using Semi-supervised Multinomial Naive Bayes,» 2011.

[49] L. Bottou, «Large-Scale Machine Learning with Stochastic Gradient Descent,» σε *Proceedings of COMPSTAT'2010*, Physica-Verlag HD, 2010, pp. 177-186.

[50] V. Korde και C. N. Mahender, «TEXT CLASSIFICATION AND CLASSIFIERS: A SURVEY,» *International Journal of Artificial Intelligence & Applications (IJAIA),* τόμ. 3, αρ. 2, 2012.

[51] H. Cui και I. Google, «Comparative Experiments on Sentiment Classification for Online Product Reviews Mayur Datar».

[52] B. Lott, «Survey of Keyword Extraction Techniques Problem Description,» 2012.

[53] «NumPy — NumPy,» [Ηλεκτρονικό]. Available: https://numpy.org/.