



ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Ανίχνευση ύπαρξης κρούσματος ασθένειας μέσα από κείμενα του κοινωνικού
ιστού**

ΤΣΩΛΗΣ ΒΑΣΙΛΕΙΟΣ

Π2014025

Επιβλέπουσα Καθηγήτρια: Κάτια - Λήδα Κερμανίδου

ΠΡΟΛΟΓΟΣ:

Στην παρούσα πτυχιική μελέτη τίθεται υπό μελέτη η ανίχνευση κρούσματος κορονοϊού μέσα από κείμενα του κοινωνικού ιστού, και πιο συγκεκριμένα από το Twitter. Αναλυτικότερα, η γρήγορη εξάπλωση του covid-19 άλλαξε ριζικά την καθημερινότητα της σύγχρονης κοινωνίας και κύριο ρόλο σε αυτό διαδραμάτισαν τα μέτρα και οι πολιτικές που εφαρμόστηκαν από τις αρμόδιες αρχές σε κάθε χώρα. Στην Ελλάδα η έννοια της πανδημίας εισάχθηκε στην καθημερινότητα των πολιτών την άνοιξη του 2020 και σε συνδυασμό με τον κοινωνικό περιορισμό, η χρήση των μέσων κοινωνικής δικτύωσης εκτοξεύθηκε. Ο κοινωνικός ιστός, και ιδιαίτερα το Twitter έχει χρησιμοποιηθεί ευρέως για την ανίχνευση κρουσμάτων επιδημίας. Η έγκαιρη ανίχνευση των κρουσμάτων αυτών μπορεί να οδηγήσει στην χαρτογράφηση της διάδοσης της ασθένειας και την έγκαιρη παρέμβαση των αιτίων για την αντιμετώπισή της. Στόχος της εργασίας αυτής είναι η υλοποίηση συστήματος αυτόματης αναγνώρισης κρουσμάτων ασθένειας μέσα από επεξεργασία κειμένου που δημοσιεύουν οι χρήστες του Twitter στο συγκεκριμένο κοινωνικό δίκτυο.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ ΕΙΚΟΝΩΝ:	6
ΠΕΡΙΕΧΟΜΕΝΑ ΠΙΝΑΚΩΝ:	7
1. ΕΙΣΑΓΩΓΙΚΕΣ ΕΝΝΟΙΕΣ	8
2. ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΕΠΙΣΚΟΠΗΣΗ	10
3. ΜΕΘΟΔΟΛΟΓΙΑ	12
3.1. Εργαλεία Υλοποίησης	12
3.2. Απόκτηση Δεδομένων	15
3.3. Λεξικό δεδομένων	18
3.4. Επεξεργασία δεδομένων	19
3.4.1. Stop Words	19
3.4.2. Ειδικοί χαρακτήρες	20
3.4.3. Stemming	20
3.4.4. Lemmatization	21
3.4.5. Τιμή Nan	21
3.4.6. Smote	23
3.5. Προσεγγίσεις	23
3.5.1. Bag of Words	23
3.5.2. Term Frequency Inverse Document Frequency (TFIDF)	24
3.6. Μηχανική μάθηση	25
3.6.1. Αλγόριθμος Bayes	26
3.6.2. Αλγόριθμος Decision Tree	26
3.6.3. Αλγόριθμος Random Forest	27
3.6.4. Αλγόριθμος Logistic Regression	27
3.6.5. Αλγόριθμος Gradient Boosting	28
3.6.6. Αξιολόγηση μοντέλου ταξινόμησης	28
3.6.7. Κατηγοριοποίηση μοντέλου	28

4.	ΤΕΛΙΚΗ ΑΝΑΛΥΣΗ.....	30
4.1.	Αρχιτεκτονική δεδομένων	30
4.1.1.	Δεδομένα εκπαίδευσης	30
4.1.2.	Δεδομένα ελέγχου	33
4.2.	Ανάλυση διαδικασίας.....	33
4.2.1.	Logistic Regression	34
4.2.2.	Naives Bayes.....	34
4.2.3.	Random Forest	35
4.2.4.	Decision Tree	35
4.2.5.	Gradient Boosting	35
4.3.	Ανάλυση Term Frequency Inverse Document Frequency	35
4.3.1.	TF-IDF για συχνότερες λέξεις	37
4.4.	Ανάλυση Bag of Words.....	40
4.4.1.	Ανάλυση Bag of Words για συχνές λέξεις.....	47
4.5.	Ανάλυση κειμένου με bigrams	50
4.6.	Απλό Bag of Words	53
4.7.	Τεκμηρίωση.....	54
4.7.1.	Καμπύλη ROC του TF-IDF	56
4.7.2.	Καμπύλη ROC του TF-IDF με τις συχνές λέξεις	57
4.7.3.	Καμπύλη ROC του Bag of Words.....	58
4.7.4.	Καμπύλη ROC του Bag of Words με τις συχνές λέξεις	59
5.	ΣΥΜΠΕΡΑΣΜΑΤΑ.....	60
5.1.	Βασικές ιδέες.....	60
5.2.	Συμπεράσματα	60
5.3.	Μελλοντικά βήματα	61
	BIBΛΙΟΓΡΑΦΙΑ:	63

ΠΕΡΙΕΧΟΜΕΝΑ ΕΙΚΟΝΩΝ:

Εικόνα 1:Εφαρμογή Anaconda	12
Εικόνα 2:Ιστότοπος API Twitter Πηγή: https://developer.twitter.com/en/docs/twitter-api	16
Εικόνα 3: Δημιουργία κλειδιών Πηγή: https://developer.twitter.com/en/docs/twitter-api	17
Εικόνα 4: Αρχείο δεδομένων cvs	18
Εικόνα 5: Αρχείο δεδομένων cvs	19
Εικόνα 6: Λέξεις που αφαιρούνται.....	20
Εικόνα 7: Ανίχνευση τιμών NAN	22
Εικόνα 8 :Εμφάνιση τιμών NAN	22
Εικόνα 9: Τελικό αρχείο δεδομένων	23
Εικόνα 10:Τελικό αρχείο δεδομένων	23
Εικόνα 11: Πίνακας σύγχυσης	28
Εικόνα 12: Σύνολο από μοναδικές λέξεις	31
Εικόνα 13: Αριθμός tweets.....	31
Εικόνα 14:Δεδομένα ελέγχου και εκπαίδευσης	33
Εικόνα 15:Πίνακας μοναδικών λέξεων για το BOW	41
Εικόνα 16: Δέντρο απόφασης	45
Εικόνα 17:Δείγμα από το δέντρο απόφασης.....	46
Εικόνα 18:Ταξινομητής Extra Trees	47
Εικόνα 19: Λέξεις με bigrams	51
Εικόνα 20: Λέξεις με trigrams.....	52
Εικόνα 21: Πίνακας μοναδικών bigrams	52
Εικόνα 22:Σύνολο λέξεων BOW	53
Εικόνα 23: Καμπύλη ROC για TFIDF	56
Εικόνα 24:Καμπύλη ROC TF-IDF για συχνότερες λέξεις.....	57
Εικόνα 25: Τελική καμπύλη Bag Of Words	58
Εικόνα 26: Καμπύλη ROC με τις συχνότερες λέξεις BOW.....	59

ΠΕΡΙΕΧΟΜΕΝΑ ΠΙΝΑΚΩΝ:

Πίνακας 1: Πίνακας Απαιτήσεων	13
Πίνακας 2: Αρχείο δεδομένων cvs	18
Πίνακας 3:Αποτελέσματα CV	34
Πίνακας 4: Αποτελέσματα tfidf.....	37
Πίνακας 5: Αποτελέσματα tfidf για τις συχνότερες λέξεις	38
Πίνακας 6: Αποτελέσματα Bag of words.....	43
Πίνακας 7: Αποτελέσματα BOW με τις συχνότερες λέξεις	50
Πίνακας 8: Ακρίβεια bigrams	53
Πίνακας 9: Ακρίβεια απλό BOW	54
Πίνακας 10: Αποτελέσματα καμπύλης TFIDF.....	56
Πίνακας 11:Αποτελέσματα ROC για συχνότερες λέξεις	57
Πίνακας 12: Αποτελέσματα καμπύλης για το BOW	58
Πίνακας 13: Αποτελέσματα καμπύλης ROC με τις συχνότερες λέξεις	59

1. ΕΙΣΑΓΩΓΙΚΕΣ ΕΝΝΟΙΕΣ

Τον τελευταίο καιρό, η κοινωνία ταλανίζεται από ένα μείζον θέμα που άπτεται του ιού covid-19. Ειδικότερα, ο ιός αυτός εξελίχθηκε σε παγκόσμια πανδημία έχοντας ως αφετηρία τον Μάρτιο του 2020. Απόρροια της πανδημίας ήταν η δραματική αύξηση των κρουσμάτων και θανάτων σε παγκόσμια κλίμακα. Αναντίρρητα, οι συνέπειες της πανδημίας έχουν επηρεάσει τόσο το κοινωνικό γίνεσθαι όσο και την οικονομία, με άμεσο αντίκτυπο την ψυχολογία του σύγχρονου ανθρώπου. Ένα από τα περιοριστικά μέτρα που τέθηκαν σε εφαρμογή ήταν ο κατ' οίκον περιορισμός και κατ' επέκταση η μείωση του εργατικού δυναμικού και η αύξηση της τηλεργασίας. Παράλληλα, οι εμπορικές συναλλαγές παρουσίασαν πτωτική τάση, ενώ οι ανάγκες σε ιατρικά αναλώσιμα και προσωπικό αυξήθηκαν σημαντικά.

Οι υγειονομικές αρχές κάθε χώρας μέσα από στατιστικές και επιστημονικές έρευνες έχουν παρατηρήσει ότι ο αριθμός των κρουσμάτων παρουσιάζει ανοδική πορεία, ενώ ταυτόχρονα ο ρυθμός μετάδοσης μειώνεται προοδευτικά, μέχρι σταθεροποίησης της κατάστασης. Ας αναφερθεί, ακόμη, ότι οι αρμόδιες αρχές έχουν επικεντρωθεί στην βελτιστοποίηση της ανίχνευσης των κρουσμάτων, προκειμένου η πανδημία να γίνει πιο ελεγχόμενη.

Στην αποτελεσματικότερη ανίχνευση του ιού συμβάλλει ο κοινωνικός ιστός τόσο μέσω της χαρτογράφησης της διάδοσης της πανδημίας όσο και με την έγκαιρη παρέμβαση για την αντιμετώπισή της. Πιο συγκεκριμένα, το Twitter αποτελεί ένα διαδεδομένο κοινωνικό μέσο, το οποίο συμμετέχει στη διάδοση πληροφοριών που αφορούν την επικαιρότητα. Σημαντικό, επίσης, είναι το γεγονός που οι δημοσιεύσεις προέρχονται τόσο από χρήστες όσο και από επίσημες πηγές, οι οποίες είναι άμεσα διαθέσιμες παγκοσμίως. Ιδιαίτερα, οι χρήστες έχουν τη δυνατότητα να μοιράζονται σκέψεις, συναισθήματα και προβληματισμούς που αφορούν την πανδημία. Έτσι, δίνεται η ευκαιρία στην πολιτική ηγεσία και τις υγειονομικές αρχές να εντοπίζουν τα κύρια προβλήματα της επικαιρότητας και να τα διεκπεραιώνουν κατάλληλα, καθιστώντας το Twitter ένα βοηθητικό εργαλείο.

Πράγματι, καθημερινά δημοσιεύεται πληθώρα tweets, τα οποία αφορούν την πανδημία και, κυρίως, τον αυξανόμενο αριθμό κρουσμάτων από covid-19. Ωστόσο, είναι σημαντικό να σημειωθεί ότι συγκεκριμένο ποσοστό tweets δεν είναι έγκυρα, καθώς το συγκεκριμένο κοινωνικό μέσο υπολείπεται ως προς τον έλεγχο αξιοπιστίας. Συνεπώς, κρίνεται αναγκαία η δημιουργία μοντέλων μηχανικής μάθησης, τα οποία θα συμβάλλουν στον αυτόματο εντοπισμό κρουσμάτων.

Στην παρούσα πτυχιική εργασία, αρχικά, πραγματοποιήθηκε ταξινόμηση των tweets σε δύο κατηγορίες: πιθανά κρούσματα και μη πιθανά κρούσματα στην ελληνική γλώσσα.

Συγκεκριμένα, αναλύθηκε ένα αρχείο δεδομένων μέσω της ιστοσελίδας development.twitter που περιείχε 17.601 tweets. Ας αναφερθεί ότι η πλειονότητα των δεδομένων ήταν στρεβλωμένο σε μεγάλο βαθμό, με το 90% των tweets το οποίο αντιστοιχεί σε 15826 tweets, να αναφέρεται σε μη πιθανά κρούσματα. Αντίστοιχα, το 10%, το οποίο αναφέρεται σε 1775 tweets, αφορούσε τις δημοσιεύσεις με πιθανά κρούσματα. Τονίζεται, ότι ο διαχωρισμός των κατηγοριών πραγματοποιήθηκε με χειροκίνητη διαδικασία ένα προς ένα tweet για να μεγαλύτερη αξιοπιστία.

Για τη δημιουργία του μοντέλου μηχανικής μάθησης, αρχικά εξισορροπήθηκε ο αριθμός των πιθανών και μη πιθανών κρουσμάτων και ακολούθησε η ανάλυση των δεδομένων μέσω δύο προσεγγίσεων: του μοντέλου Bag of Words και του μοντέλου Term Frequency Inverse Document Frequency (TFIDF). Η προσέγγιση του Bag of Words είναι μια απλοποιημένη αναπαράσταση που χρησιμοποιείται στην επεξεργασία φυσικής γλώσσας και την ανάκτηση πληροφορίας. Πιο αναλυτικά, σε αυτήν την προσέγγιση, ένα κείμενο, το οποίο μπορεί να είναι στη μορφή μιας πρότασης ή ενός εγγράφου, αντιμετωπίζεται ως πολλαπλό σύνολο των λέξεων, αγνοώντας τη γραμματική, την σύνταξη και τη σειρά των λέξεων. Αντίστοιχα, το Term Frequency Inverse Document Frequency (TFIDF) είναι μια αριθμητική στατιστική που εκφράζει την βαρύτητα μιας λέξης για ένα αρχείο δεδομένων. Ιδιαίτερα, χρησιμοποιείται στην ανάκτηση πληροφοριών, στην εξόρυξη δεδομένων και σε μοντέλα λέξεων.

Σημειώνεται ότι προκειμένου να εισαχθούν τα δεδομένα στους διάφορους αλγόριθμους που χρησιμοποιούνται κάθε φορά, είναι απαραίτητη η επιμέρους επεξεργασία τους, λόγω των αναγραμματισμών, των γραμματικών σφαλμάτων, των άγνωστων συμβόλων και των hashtags. αυτό επιτυγχάνεται μέσω του Lemmatization, του Stemming και της αφαίρεσης λέξεων και συμβόλων. Το Stemming και το Lemmanization αφορά τη μείωση των παραμορφωμένων ή παραγόμενων λέξεων που περιέχουν τα tweets. Παράλληλα, χρησιμοποιείται το stopwords, με το οποίο, αφαιρούνται οι λέξεις οι οποίες δεν επηρεάζουν το νόημα του tweet (άρθρα, συνδετικές λέξεις, αντωνυμίες). Επίσης, παραλείπονται οι τυχόν ξένοι χαρακτήρες, οι οποίοι δεν συνεισφέρουν στο νόημα του εκάστοτε tweet.

Καταληκτικά, στη συγκεκριμένη πτυχιακή μελέτη γίνεται ανάλυση δεδομένων για την ανίχνευση των κρουσμάτων κορονοϊού στο Twitter. Στο κεφάλαιο 3 παρουσιάζεται η μεθοδολογία για την διατύπωση των δεδομένων και η θεωρία στην οποία βασίστηκε, ενώ στο κεφάλαιο 4 πραγματοποιείται η ανάλυση δεδομένων. Στο κεφάλαιο 5 διατυπώνονται τα συμπεράσματα και οι τελικές συστάσεις. Ας σημειωθεί ότι ο κώδικας που χρησιμοποιήθηκε για τη συγκεκριμένη μελέτη δίνεται στο σύνδεσμο: https://github.com/billtsolis/Twitter_Coronavirus_Detection.

2. ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΕΠΙΣΚΟΠΗΣΗ

Το Twitter ανέκαθεν συνιστούσε μια πηγή πληροφοριών σχετικά με την παρακολούθηση συμπεριφοράς της κοινής γνώμης. Ιδιαίτερα στη διάρκεια της πανδημίας λόγω του Covid-19 πολυάριθμες μελέτες πραγματοποιήθηκαν για να διερευνήσουν τις συζητήσεις, τις σκέψεις ή τις εμπειρίες του σύγχρονου ανθρώπου που προκύπταν μέσα από τα tweets. Τα προληπτικά μέτρα, το κοινωνικό στίγμα, τα κρούσματα, οι θάνατοι και η γενικότερη κατάσταση που επικρατούσε στις χώρες λόγω του κορονοϊού συνιστούν κάποιες από τις κατηγορίες που επέλεξαν οι Xue et al. να διερευνήσουν[1].

Οι παρακάτω μελέτες παρουσιάζουν διάφορες ιδέες που περιλαμβάνουν τη χρήση δεδομένων του twitter με διάφορους τρόπους. Χρησιμοποιούν τα δεδομένα από το twitter για να αναλύσουν πληροφορίες μέσω μηχανικής μάθησης, όπως διάδοση πληροφοριών και συμπεριφοράς κατά τη διάρκεια του covid-19[2],[3].

Σε άλλη έρευνα τα tweets αναλύθηκαν χρησιμοποιώντας μοντέλο μηχανικής μάθησης χωρίς επίβλεψη, και πιο συγκεκριμένα το BTM (biterm topic model). Σε αυτό το μοντέλο, τα tweets ομαδοποιήθηκαν σε κατηγορίες ανάλογα το θέμα που αφορούσαν (συμπτώματα, διάγνωση, ανάρρωση) και αξιολογήθηκαν ως προς τα στατιστικά και γεωγραφικά χαρακτηριστικά τους[4].

Οι Cuomo et al. ανέλυσαν ένα σύνολο δεδομένων αποτελούμενο από περίπου 460000 tweets που είχαν άμεση σχέση με τον Covid-19 και αφορούσαν τις Ηνωμένες Πολιτείες. Στη συγκεκριμένη μελέτη χρησιμοποιήθηκε ένας SVM ταξινομητής (support vector machine) και έτσι έγινε μια γεωγραφική ανάλυση όπου παρουσιάζονταν οι διακυμάνσεις μεταξύ των κρουσμάτων στις διάφορες πολιτείες των ΗΠΑ και πως αυτά μεταβαλλόταν με το χρόνο [5], [6].

Επίσης, είναι σημαντικό να αναφερθούν οι μελέτες οι οποίες χρησιμοποιούν το μοντέλο Latent Dirichlet Allocation (LDA) σε θέματα που αφορούν τη διάδοση του covid-19 και στον τομέα της υγειονομικής περίθαλψης. Παράλληλα, γίνεται μοντελοποίηση χρησιμοποιώντας Term Frequency Inverse Document Frequency (TFIDF) με στόχο τον εντοπισμό πιθανών κρουσμάτων [7], [8].

Σύμφωνα με άλλες μελέτες [9], [10], [11], ο αριθμός των ενεργών λοιμώξεων από τη νόσο του κορονοϊού (covid-19) δεν είναι ομοιόμορφα κατανεμημένος, τόσο χωρικά όσο και χρονικά. Έτσι, οι μετρήσεις από μολύνσεις δεν είναι πιθανό να απεικονίσουν το πλήρες εύρος των αρνητικών συνεπειών της πανδημίας για τις κοινωνίες [12]. Σε μια προσπάθεια να γίνουν κατανοητές οι επιπτώσεις του covid-19, αξιολογείται η γεωγραφική και διαχρονική κατανομή των tweets. Συγκρίνονται τα αποτελέσματα με τον αριθμό των επιβεβαιωμένων κρουσμάτων που

αναφέρθηκαν σε διάφορες χώρες. Τα αποτελέσματα βάσει του πληθυσμού υποδεικνύουν ότι οι αγροτικές περιοχές ενδέχεται να μην έχουν εμπλακεί με το θέμα του covid-19 παρά μόνο σε μεταγενέστερα στάδια μιας επιδημίας. [13], [14], [15], [16].

Σημειώνεται ότι γίνεται προσπάθεια να δημιουργηθεί ένα κοινό σύνολο δεδομένων από το twitter. Αυτά τα σύνολα δεδομένων περιέχουν tweets λέξεων από διαφορετικές χώρες τα οποία είναι επεξεργασμένα για μοντέλα μηχανικής μάθησης. Ο βασικός στόχος είναι η άμεση παρακολούθηση πιθανών και μη πιθανών κρουσμάτων παρέχοντας ένα σύνολο δεδομένων που είναι ήδη εκπαιδευμένο [17], [18], [19].

Οι στόχοι πολλών ερευνητών είναι η αλληλεπίδραση των χρηστών μέσω των κοινωνικών δικτύων που θα μπορούσαν να συσχετιστούν με συμπτώματα covid-19, εμπειρίες με κοντινά πρόσωπα και αναφορές εκδηλώσεων ασθενειών χρησιμοποιώντας μια προσέγγιση μηχανικής μάθησης χωρίς επίβλεψη [20], [21], [22].

Παράλληλα, χρησιμοποιείται ένας ταξινομητής που βασίζεται στο BERT [23] για να εντοπίζει αυτόματα tweets που αναφέρουν συμπτώματα σχετικά με τον covid-19. Η απόδοση της βαθμολογίας του ταξινομητή συμπτωμάτων του covid-19 αξιολογήθηκε χρησιμοποιώντας τη μέτρηση βαθμολογίας F σε ένα σύνολο ελέγχου. Ο ταξινομητής πέτυχε βαθμολογία 0,71 και 0,96 για θετικές και αρνητικές κατηγορίες, αντίστοιχα. Μετά την εκπαίδευση και την αξιολόγηση της ταξινόμησης, χρησιμοποιήθηκε σε δεδομένα και αναλύθηκαν τα ταξινομημένα tweets για την αποτελεσματικότητα του ταξινομητή [24],[25].

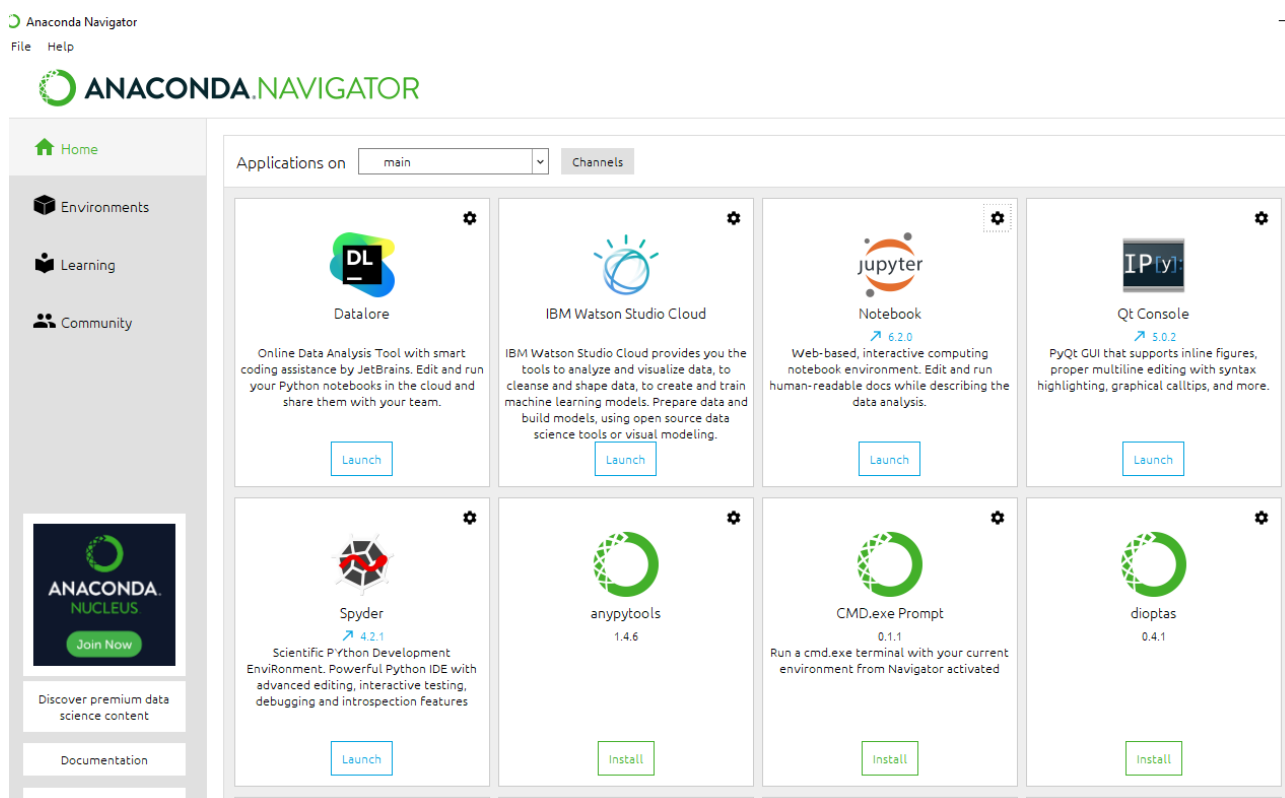
Τέλος, ενώ πολλές μελέτες αναλύουν τις λειτουργίες που μπορούν να εκπληρώσουν τα κείμενα κατά τη διάρκεια ανθρωπιστικών κρίσεων ή καταστροφών, δεν υπάρχει κατανόηση του τρόπου με τον οποίο δομείται το νόημα των κειμένων. Αυτοί οι ερευνητές αναλύουν δεδομένα τα οποία έχουν να κάνουν με τα συμπτώματα ώστε να δημιουργήσουν ένα σύστημα για την αντιμετώπιση τους. Παρουσιάζουν μελέτες περίπτωσης που επικεντρώνεται σε διάφορες επιδημίες που έχουν συμβεί στο παρελθόν και συγκεκριμένα είναι ο ιός του έμπολα και της γρίπη [26], [27],[28].

3. ΜΕΘΟΔΟΛΟΓΙΑ

Στο συγκεκριμένο κεφάλαιο παρουσιάζεται αναλυτικά η μεθοδολογία και η θεωρία στην οποία βασίστηκε η συγκεκριμένη μελέτη. Γίνεται αναφορά στα εργαλεία υλοποίησης, στον τρόπο με τον οποίο αποκτήθηκαν και επεξεργάστηκαν τα δεδομένα, στην προσέγγιση των μοντέλων καθώς επίσης και στην αξιολόγησή τους.

3.1. Εργαλεία Υλοποίησης

Για την υλοποίηση του κώδικα χρησιμοποιήθηκε η εφαρμογή Anaconda Navigator [29]. Η συγκεκριμένη εφαρμογή διαθέτει τη δυνατότητα αναζήτησης και εγκατάστασης βιβλιοθηκών αυτόματα. Ειδικά, ο κώδικας αναπτύχθηκε με γλώσσα Python 3.0 μέσω του Jupyter Notebook και ο τύπος του αρχείου είναι ".ipynb". Σημειώνεται ότι το Jupyter Notebook είναι ένα αρχείο, το οποίο περιέχει κώδικα python, κείμενο, μαθηματικά και γραφικές παραστάσεις.



Εικόνα 1: Εφαρμογή Anaconda

Οι απαιτήσεις που χρειάστηκαν για το συγκεκριμένο κώδικα διατυπώνονται στον Πίνακα 1 και αναφέρονται ακολούθως:

Πίνακας 1: Πίνακας Απαιτήσεων

os	sklearn.tree
re	graphviz
tweepy	sklearn.datasets
pandas	sklearn.naive_bayes
pandas_profiling	sklearn.ensemble
matplotlib.pyplot	dtreeplt
nltk.corpus	nltk.util
seaborn	stopwordsiso
warnings	nltk.stem
unidecode	numpy
sklearn.model_selection	collections
sklearn.linear_model	sklearn.feature_extraction.text
sklearn.metrics	sklearn

- **Os:** χρησιμοποιείται για κανονικές εντολές στη γλώσσα Python.
- **Re:** πρόκειται για μια βιβλιοθήκη που δηλώνει εάν μια συγκεκριμένη συμβολοσειρά ταιριάζει με μια κανονική εντολή.
- **Graphviz:** πραγματοποιείται η σχεδίαση συγκεκριμένου τύπου γραφημάτων.
- **Tweepy:** βιβλιοθήκη Python το οποίο παρέχει πρόσβαση στα δεδομένα του Twitter, καθιστώντας πιο απλή την απόκτηση δεδομένων σε σύντομο χρονικό διάστημα.
- **Pandas:** χρησιμοποιείται για την επεξεργασία και ανάλυση δεδομένων και προσφέρει λειτουργίες για την επεξεργασία αριθμητικών πινάκων.
- **pandas_profiling:** πρόκειται για μια βιβλιοθήκη στην οποία η ανάλυση δεδομένων με λίγες γραμμές κώδικα γίνεται γρήγορα. Επίσης, μπορεί να δημιουργήσει καλύτερη εμφάνιση και παρουσίαση των πινάκων.
- **Matplotlib:** πρόκειται μια βιβλιοθήκη για την εμφάνιση των δεδομένων σε σχεδιάγραμμα.
- **Dtreeplt:** μπορεί να σχεδιάσει ένα δέντρο που δεν χρησιμοποιεί το Graphviz, αλλά με την βοήθεια του matplotlib. Ακόμη, μπορεί να αποθηκεύσει ένα δέντρο απόφασης σε ένα αρχείο.
- **Nltk:** το Natural Language Toolkit, είναι μια βιβλιοθήκη για επεξεργασία σε ένα κείμενο με στατιστικούς όρους. Περιλαμβάνει γραφικές απεικονίσεις και δείγματα δεδομένων.

- **Seaborn:** το Seaborn είναι μια βιβλιοθήκη εμφάνισης δεδομένων που βασίζεται στο matplotlib. Είναι απαραίτητο για τη σχεδίαση σύγχρονων και ενημερωτικών γραφημάτων.
- **Stopwordsiso:** περιέχει μια συλλογή λέξεων σε πολλές γλώσσες με σκοπό την αφαίρεση των λέξεων οι οποίες είναι περιττές στα tweets[30].
- **Unidecode:** Αυτό που παρέχει το Unidecode είναι μια συνάρτηση, που ελέγχει τα δεδομένα και τα μετατρέπει σε μορφή ASCII (δηλαδή, γράμματα τα οποία είναι καθιερωμένα παγκοσμίως σε δυαδική μορφή μεταξύ 0x00 και 0x7F).
- **numpy:** Είναι μια βιβλιοθήκη που δίνει υποστήριξη σε μεγάλους πίνακες, μαζί με μια μεγάλη συλλογή μαθηματικών συναρτήσεων υψηλού επιπέδου για λειτουργία σε αυτούς τους πίνακες.
- **Selection:** Χρησιμοποιείται στη μηχανική μάθηση και επιλέγει ένα υποσύνολο από όλα τα δεδομένα ώστε να κατασκευαστεί ένα μοντέλο.
- **sklearn:** Το Scikit-Learn είναι μια βιβλιοθήκη που διαθέτει διάφορους αλγόριθμους ταξινόμησης όπως είναι logistic regression, random forest κ.α.
- **sklearn.tree:** Αυτή η βιβλιοθήκη διαθέτει τη μέθοδο Decision Tree που χρησιμοποιείται για την ταξινόμηση στην μηχανική μάθηση.
- **sklearn.linear_model:** Το Linear Regression χρησιμοποιείται για τον αλγόριθμο logistic regression.
- **sklearn.feature_extraction.text:** Η sklearn.feature_extraction μπορεί να χρησιμοποιηθεί για την δημιουργία ενός πίνακα λέξεων από το σύνολο δεδομένων, ώστε να εκτελεστεί ένας αλγόριθμος. Δηλαδή, μετατρέπει μια συλλογή κειμένου σε έναν πίνακα TF-IDF.
- **sklearn.metrics:** Αυτή η βιβλιοθήκη χρησιμοποιείται για τον υπολογισμό του πίνακα σύγχυσης ώστε να βρεθεί αυτόματα η ακρίβεια ενός αλγορίθμου.
- **sklearn.datasets:** Εμφανίζει την ταξινόμηση από το σύνολο δεδομένων που εφαρμόστηκε ένας αλγόριθμος.
- **sklearn.naive_bayes:** Αυτή η βιβλιοθήκη περιέχει τον αλγόριθμο Bayes. Ο ταξινομητής Bayes λειτουργεί αρκετά καλά σε πολλές περιπτώσεις.

Το θεώρημα του Bayes δηλώνει την ακόλουθη σχέση:

$$P(y|x_1, \dots, x_n) = \frac{P(y) \cdot P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)}$$

Έστω,

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P((x_i|y)$$

για όλα τα i, και γίνεται απλοποίηση παρακάτω:

$$P(y|x_1, \dots, x_n) = P(y) \cdot \prod_{i=1}^n P(x_i|y) \cdot P(x_1, \dots, x_n)$$

Από $P(y|x_1, \dots, x_n)$ είναι σταθερή τιμή και χρησιμοποιείται ο παρακάτω κανόνας ταξινόμησης:

$$P(y|x_1, \dots, x_n) \propto P(y) \cdot \prod_{i=1}^n P(x_i|y) \Rightarrow \hat{y} = \arg \max_y P(y) \cdot \prod_{i=1}^n P(x_i|y)$$

- **sklearn.ensemble:** Περιέχει τον αλγόριθμο Gradient Boosting για την ανάλυση των μοντέλων που γίνεται η ταξινόμηση μέσω των δέντρων αποφάσεων.

3.2. Απόκτηση Δεδομένων

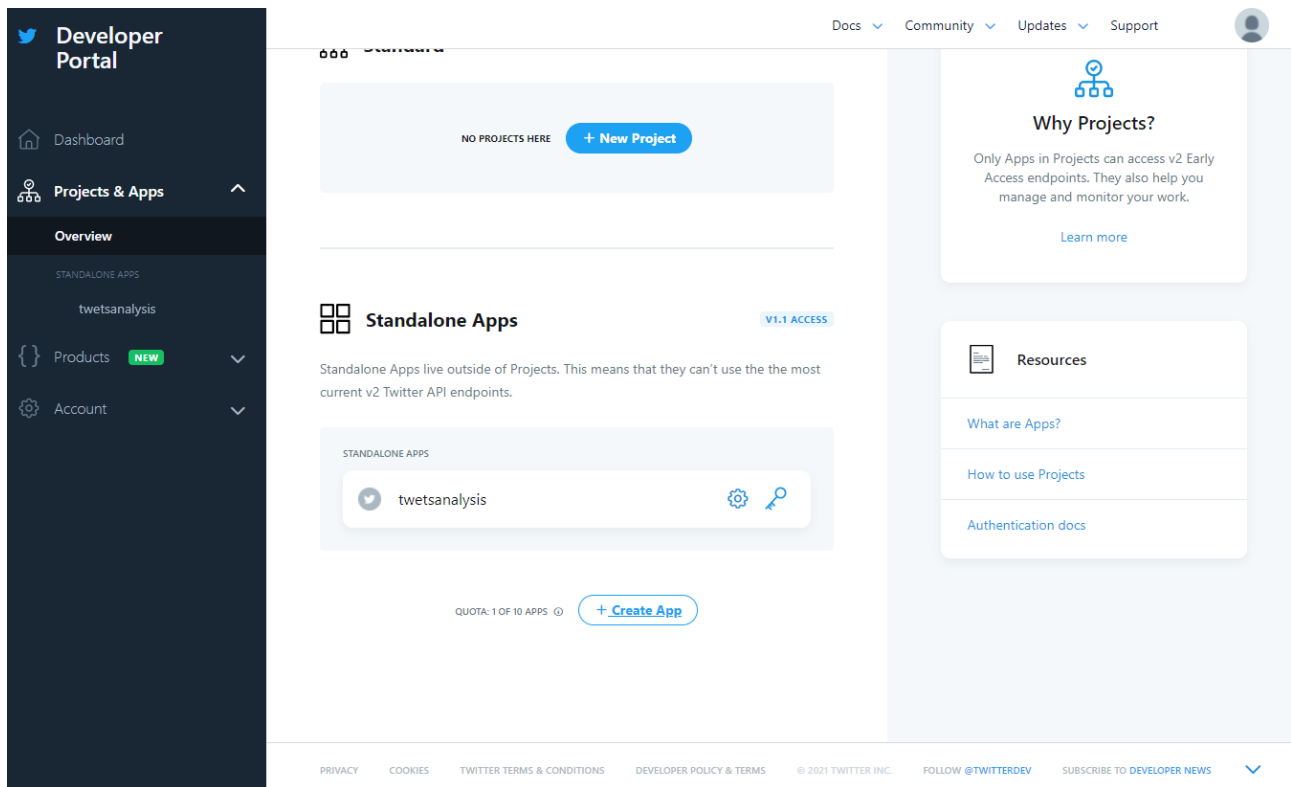
Προκειμένου να συλλεχθούν δεδομένα που αφορούσαν τον κορονοϊό αξιοποιήθηκε το API Twitter σε συνδυασμό με το Tweepy και οι παράμετροι που επιλέχθηκαν για να χαρακτηρίσουν τα δεδομένα ήταν οι εξής:

- Λέξεις αναζήτησης: κρούσμα, συμπτώματα, κορονοϊός, covid, coronavirus
- Γλώσσα: Ελληνικά
- Τοποθεσία: Ελλάδα
- Ημερομηνία: 20-03-2020

Για τα δεδομένα από το Twitter, χρησιμοποιήθηκε το σύνολο δεδομένων που σχετίζονται με το covid-19 από το Μάρτιο του 2020, δηλαδή από την ημερομηνία που επισημοποιήθηκε ο κορονοϊός στην Ελλάδα. Συγκεκριμένα, συλλέχθηκαν tweets στα ελληνικά χρησιμοποιώντας το Twitter API με τις ακόλουθες λέξεις-κλειδιά: Coronavirus, COVID-19, coronavirus, covid-19, covid, κορονοϊός, κρούσμα, συμπτώματα.

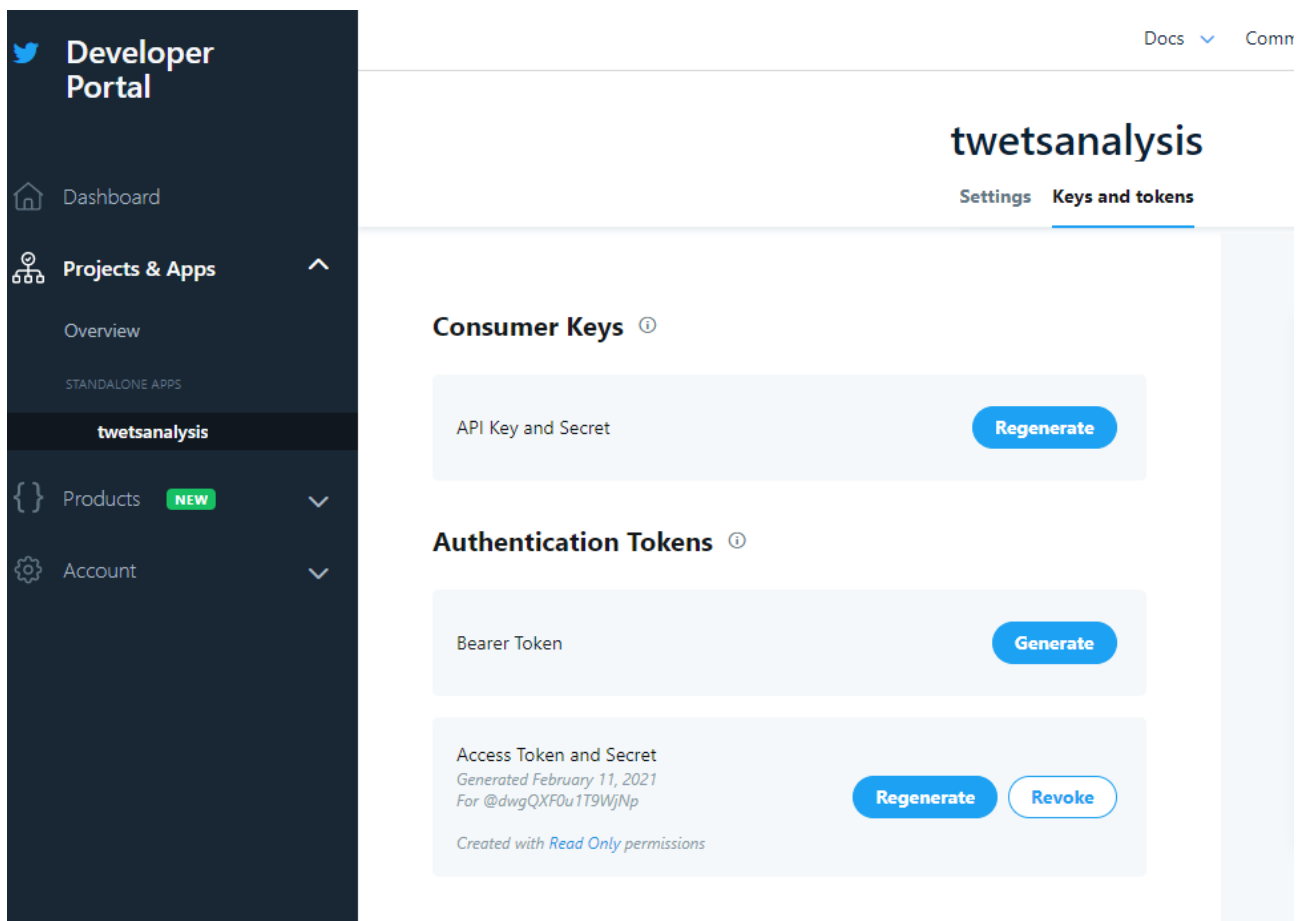
Το Twitter API παρέχει στους προγραμματιστές πρόσβαση στις περισσότερες λειτουργίες του Twitter. Μέσω αυτού γίνεται η απόκτηση πληροφοριών που σχετίζονται με το Twitter, όπως tweets, χρήστες και τοποθεσίες. Αυτό πραγματοποιήθηκε με το Tweepy[31] που είναι ένα πακέτο Python με το οποίο ο καθένας μπορεί να έχει πρόσβαση στα δεδομένα του Twitter με τη γλώσσα Python.

Προκειμένου να δημιουργηθεί η πρόσβαση των δεδομένων, πραγματοποιείται εγγραφή στο Twitter API και στη συνέχεια δημιουργείται η εφαρμογή. Το Twitter δίνει την δυνατότητα πρόσβασης μέσω εφαρμογής. Επομένως, πρέπει να καταχωρηθεί η εφαρμογή για να πραγματοποιηθεί η συλλογή των tweets. Ενδεικτικά μπορούμε να παρατηρήσουμε τις παρακάτω εικόνες:



Εικόνα 2: Ιστότοπος API Twitter Πηγή: <https://developer.twitter.com/en/docs/twitter-api>

Έπειτα, γίνεται η διαδικασία για την παροχή των δεδομένων μέσω της καρτέλας `keys&tokens`, προκειμένου να ληφθούν οι κωδικοί, για να χρησιμοποιηθούν στον κώδικά.



Εικόνα 3: Δημιουργία κλειδιών Πηγή: <https://developer.twitter.com/en/docs/twitter-api>

Αφού δημιουργηθούν οι κωδικοί, χρησιμοποιούνται στον κώδικα και μετά διατυπώνονται οι παράμετροι όπως είναι παρακάτω:

Authentication

```
consumer_key = 'IO8BtoBtTy6FgAOvYkCuzq9wb'
consumer_secret = 'WyQwP2BxKhjfLoUpgJAyxYTOF2lGfqjxrCl3zdknLSTwZMTmtO'
access_token = '1359527942116814849-LKli1NWSyxAnJ2oDs8M17lzY1uuOp9'
access_token_secret = 'rJVfwQ8nJ2rx197NG7Hf0V4PZiOSdmWAEpSSlgonHoH4F'
auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tw.API(auth, wait_on_rate_limit=True)

search_words = "κρούσμα"
noOfTweet = 10000
date_since="2020-03-10"
tweets = tw.Cursor(api.search, q=search_words, lang="el", since=date_since).items(noOfTweet)
tweets
```

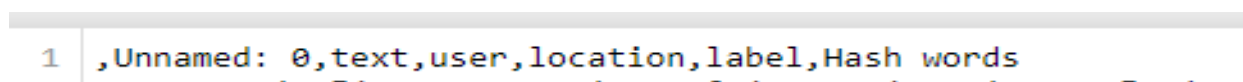
```
tweet_details = [[tweet.text, tweet.user.screen_name, tweet.user.location] for tweet in tweets]
tweet_df = pd.DataFrame(data=tweet_details, columns=['text', 'user', "location"])
pd.set_option('max_colwidth', 800)
tweet_df.head(noOfTweet)
```

3.3. Λεξικό δεδομένων

Μέσω της χρήσης του API Twitter και του Tweepy πραγματοποιήθηκε η συλλογή των δεδομένων στην ελληνική γλώσσα που αφορούσαν το χρονικό διάστημα μετά από το Μάρτιο του 2020 [31]. Πλέον, υπάρχουν περισσότερα από 15000 tweets τα οποία αποθηκεύτηκαν ως αρχεία CVS. Για την καλύτερη αξιοπιστία, διαμοιράστηκαν τα δεδομένα με χειροκίνητο τρόπο σε πιθανά και μη πιθανά κρούσματα κορονοϊού. Το αποτέλεσμα όπως αναφέρθηκε ήταν ανισόρροπο με το 90%(15826) των tweets περιέχουν δεδομένα Twitter χωρίς κρούσματα και το 10%(1.775) των tweets περιέχουν δεδομένα Twitter με πιθανά κρούσματα κορονοϊού. Στην συνέχεια επεξεργαστήκαν αρκετές λέξεις από τα tweets, ώστε να υπάρχει καλύτερη απόδοση στο αποτέλεσμα της ανάλυσης. Παρακάτω αναδεικνύεται ο πίνακας για τα ακριβή στοιχεία των δεδομένων.

Πίνακας 2: Αρχείο δεδομένων cvs

Όνομα	Περιεχόμενο
Label	Class of tweet:1-covid, Class of tweet:0 noncovid
User	Το όνομα του χρήστη που δημοσιοποίησε το tweet
Text	Περιεχόμενο tweet
Location	Η τοποθεσία του χρήστη
Hash word	Λέξεις με Hashtag #



```
1 | ,Unnamed: 0,text,user,location,label,Hash words
```

Εικόνα 4: Αρχείο δεδομένων cvs

0	text	user	location	label	Hash words
0	ουρικό οξύ:ποια συμπτώματα δείχνουν ότι είναι ...	OrganicLife_gr	Greece Ελλάδα	0	No hashtags
1	θέλετε πείτε λόγο ενώ νιώθετε καλά,χωρίς συμπτ...	dDwAgwwjct1p3Tf	NaN	0	#rapidtest?
2	σπυρο λέω καλό σου, όταν αυξάνεται τεστ αυξάνο...	CapitanoZor	NaN	0	No hashtags
3	θέλετε πείτε λόγο ενώ νιώθετε καλά,χωρίς συμπτ...	JimarasG7	NaN	0	#rapidtest?1....
4	Οι μεγάλοι χρειάζονται εμβόλια, ώστε όταν κολλ...	KorinTil	NaN	0	No hashtags

Εικόνα 5: Αρχείο δεδομένων cvs

3.4. Επεξεργασία δεδομένων

Μια διαδικασία επεξεργασίας δεδομένων είναι απαραίτητη για την αποφυγή λέξεων που επηρεάζουν το κείμενο για αυτή την ανάλυση. Αρχικά, γίνεται μια μετατροπή των δεδομένων σε μια συγκεκριμένη μορφή κειμένου. Παράλληλα, γίνεται αφαίρεση συμβόλων, εικόνων και λέξεων που είναι περιττά στα tweets. Για την επεξεργασία των δεδομένων εφαρμόστηκαν οι παρακάτω προσεγγίσεις.

3.4.1. Stop Words

Το Stopwords αποτελεί μία μέθοδο με την οποία λέξεις όπως άρθρα, προθέσεις ή συνδετικές λέξεις αφαιρούνται χωρίς να επηρεάζεται το τελικό μοντέλο της εκπαίδευσης. Συνήθως οι λέξεις που χρησιμοποιείται στην Ελληνική γλώσσα περιλαμβάνουν: "και", "δεν" κ.λπ.[32].

```
from stopwordsiso import stopwords
words = stopwords("el") # Greek stopwords
list(words)[:10] # print first 10 words
data_frame['text'] = data_frame['text'].apply(lambda x : ' '.join([word for word in
str(x).split() if not word in set(stopwords("el"))]))
```

```
In [26]: stopwords("el")

Out[26]: {'ένα',
          'έναν',
          'ένας',
          'αι',
          'ακομα',
          'ακομη',
          'ακριβως',
          'αληθεια',
          'αληθινα',
          'αλλα',
          'αλλαχου',
          'αλλες',
          'αλλη',
          'αλλην',
          'αλλης',
          'αλλιως',
          'αλλιωτικα',
          'αλλο',
          'αλλοι',
          'αλλοι'
```

Εικόνα 6: Λέξεις που αφαιρούνται

3.4.2. Ειδικοί χαρακτήρες

Στη συνέχεια, αφαιρέθηκαν σύμβολα και hashtags. Τα hashtags συνήθως χρησιμοποιούνται για την κατηγοριοποίηση συζητήσεων και δημοσιεύσεων. Αναφορικά με τα σύμβολα, δεν επηρεάζουν το νόημα του κειμένου και συνεπώς είναι απαραίτητη η αφαίρεσή τους [32].

#Remove hash words

```
df['text'] = df['text'].apply(lambda x : ''.join([word for word in str(x).split() if not word.startswith('#')]))
```

```
def clean_tweets(text):
```

```
    text = re.sub("RT @[\\w]*:", "", text)
```

```
    text = re.sub("@[\\w]*", "", text)
```

```
    text = re.sub("https?:/[A-Za-z0-9./]*", "", text)
```

```
    text = re.sub("\\n", "", text)
```

```
    return text
```

3.4.3. Stemming

Με το stemming, οι παραγόμενες λέξεις ή διαφορετικές λέξεις αλλά με το ίδιο νόημα κόβονται και αντικαθίστανται από την βασική λέξη του νοήματος που μπορούν να αλλαχθούν. Μια λέξη εξετάζεται και μειώνεται. Για το σύνολο δεδομένων, χρησιμοποιήθηκε ο αλγόριθμος που ακολουθεί [33].

```
from nltk.stem import PorterStemmer
```

```
#Stemming
```

```
ps = PorterStemmer()
```

```
adwait = data_frame
```

```
#adwait.head()
```

```
data_frame['text'] = data_frame['text'].apply(lambda x : ' '.join([ps.stem(word) for word in x.split()])))
```

3.4.4. Lemmatization

Είναι η διαδικασία ομαδοποίησης των διαφορετικών κλίσεων μιας λέξης, ώστε να μπορούν να αναλυθούν ως μια κοινή λέξη. Είναι παρόμοιο με το stemming. Με αυτόν τον τρόπο, οι λέξεις με όμοια ρίζα θεωρούνται ως ένα στοιχείο [33].

```
from nltk.stem import WordNetLemmatizer
```

```
#Lemmatization
```

```
lemmatizer = WordNetLemmatizer()
```

```
data_frame['text'] = data_frame['text'].apply(lambda x : ' '.join([lemmatizer.lemmatize(word) for word in x.split()])))
```

3.4.5. Τιμή Nan

Το NaN σημαίνει Not A Number και είναι μια ειδική τιμή και δεν μπορεί να μετατραπεί σε άλλον τύπο εκτός από float. Η τιμή NaN είναι ένα από τα σημαντικότερα προβλήματα στην επεξεργασία των δεδομένων και είναι αναγκαίος ο καθαρισμός της για καλύτερα αποτελέσματα του κειμένου.

Η μέθοδος αντιμετώπισης isna () [34] στη βιβλιοθήκη pandas εφαρμόζεται καλύτερα για τον έλεγχο στην περίπτωση που η τιμή είναι null/NaN ή όχι. Θα επιστρέψει True αν η τιμή είναι NaN/null.

```
if pd.isnull(float("NaN")):
```

```
    print("Null Value.")
```

```
In [9]: df.isnull().sum()
```

```
Out[9]: Unnamed: 0      0  
text          54  
user           2  
location     5610  
label         0  
Hash words    0  
dtype: int64
```

```
In [10]: df['text'].isnull()
```

```
1023    False  
1024    False  
1025    False  
1026    False  
1027    False  
1028    False  
1029    False  
1030     True  
1031    False  
1032    False  
1033    False  
1034    False  
1035    False  
1036    False  
1037    False  
1038    False  
1039    False  
1040    False  
1041    False  
1042    False
```

Εικόνα 7: Ανίχνευση τιμών NAN

Ακολουθως γίνεται καθαρισμός από τιμές nan των tweets.

```
df.dropna(subset = ["text"], inplace=True)
```

```
In [22]: df.isnull().sum()
```

```
Out[22]: Unnamed: 0      0  
text          0  
user           1  
location     5571  
label         0  
Hash words    0  
dtype: int64
```

Εικόνα 8 :Εμφάνιση τιμών NAN

Τέλος, αφού έχει ολοκληρωθεί ο καθαρισμός των δεδομένων, για την τελική ανάλυση χρησιμοποιείται μόνο το κείμενο(Text) των tweets και η ετικέτα(label) από τα αρχεία cvs.

	text	label
0	Με κορονοϊό διαγνώστηκε δικηγόρος χαλκίδας – χρήστος θεοφίλης: έχω συμπτώματα γνωρίζω κόλλησα τ...	1
1	συμφωνώ, θέλω ξέρω εφόσον έχω συμπτώματα προφυλάξω ευάλωτες επαφές κυρίως.	1
2	Α' γυμνασίου. κόλλησαν εβδομάδα έκλεισαν σχολεία (πριν χιονιά). ευτυχώς όλα παιδιά περν...	1
3	"είχα ελπίδα ότι επειδή είχα κάνει 1η δόση εμβολίου συμπτώματα ήταν ήπια. και έτσι έγινε.το εμβόλιο ό...	1
4	μπορεί παραπάνω γιατί πολλοί είχαν συμπτώματα έκαναν τε...	1

Εικόνα 9: Τελικό αρχείο δεδομένων

	text	label
15602	Με περιοχές χωρίς θέρμανση νερό, σκοτάδι, 3η (ή 4η;) μέρα, συνεχιζόμενη επέλαση , επα...	0
7909	διαθέσιμη μέσω βεβαίωση εμβολιασμού covid-19	0
10728	Η ρωσία αρχίζει εμβολιάζει αστέγους covid-19 -	0
14222	συμμετοχή διαδικτυακή συζήτηση συριζα - προοδευτική συμμαχία θέμα «οι ηθοποιοί καιρό covid-19	0
12021	έκανε πρεμιέρα ελλάδα μητσotάκηδων πιερακάκηδων...«σε λίγους μήνες κανένας ισραήλ μπορ...	0

Εικόνα 10:Τελικό αρχείο δεδομένων

3.4.6. Smote

Το SMOTE [35] είναι ένας τύπος αύξησης δεδομένων που συνθέτει νέα δείγματα από τα υπάρχον σύνολο δεδομένων. Επιλέγει ένα τυχαίο δείγμα και στη συνέχεια χρησιμοποιεί έναν αλγόριθμο KNN για την επιλογή γειτόνων. Με αυτήν τη διαδικασία, μπορεί να δημιουργήσει όσα σύνθετα δείγματα χρειάζεται. Αυτό καθιστά το SMOTE [35] ιδανικό για το σύνολο δεδομένων το οποία παρουσιάζεται στην συγκεκριμένη μελέτη.

3.5. Προσεγγίσεις

Η παρούσα ενότητα επικεντρώνεται στην εξέταση δύο σημαντικών προσεγγίσεων για τη δημιουργία χαρακτηριστικών: το Bag of Words [36] και το Term Frequency Inverse Document Frequency (TFIDF) [37]. Στη συνέχεια, αναλύονται οι αλγόριθμοι μηχανικής μάθησης που χρησιμοποιούνται και γίνεται λόγος για την αξιολόγησή τους.

3.5.1. Bag of Words

Το μοντέλο Bag of Words [36] είναι ένας τρόπος συλλογής λέξεων από τα tweets. Ονομάζεται Bag of Words, επειδή δεν αναφέρονται οι πληροφορίες σχετικά με τη σειρά ή τη δομή των λέξεων. Το μοντέλο αυτό, μετατρέπει τα δεδομένα σε έναν πίνακα με λέξεις. Ο στόχος είναι να μετατρέψει τα δεδομένα σε ένα πίνακα που μπορεί να χρησιμοποιηθεί για ένα μοντέλο μηχανικής μάθησης. Με λίγα λόγια, εάν το σύνολο δεδομένων έχει 10 λέξεις, βαθμολογεί κάθε

λέξη με μέτρο το 10. Αυτό γίνεται ως δυαδικό σύστημα με τιμή 0 αν λείπει η λέξη και με 1 αν υπάρχει.

Για παράδειγμα, χρησιμοποιώντας την μέθοδο αυτή, παρουσιάζεται το tweet ("Σήμερα πήγα για τεστ και βγήκα θετικός ") και γίνεται η μετατροπή σε ένα δυαδικό σύστημα.

Η διαδικασία θα έχει ως εξής:

- "Σήμερα" = 1
- "πήγα" = 1
- "για" = 1
- "τεστ" = 1
- "και" = 1
- "βγήκα" = 1
- "θετικός" = 1
- "αρνητικός" = 0
- "περιμένω" = 0

Ως δυαδικό διάνυσμα, θα φαίνεται το εξής:

[1, 1, 1, 1, 1, 1, 1, 0, 0]

Τα άλλα tweets θα είχαν το εξής αποτέλεσμα:

1. "Περιμένω τεστ και βγήκα αρνητικός" = [0, 0, 0, 0, 1, 1, 1, 0, 1]
2. "Σήμερα βγήκα θετικός και περιμένω" = [1, 0, 0, 0, 1, 0, 1, 1, 0]

3.5.2. Term Frequency Inverse Document Frequency (TFIDF)

Το TF - IDF είναι μια τεχνική ανάκτησης πληροφοριών που συγκρίνει τη συχνότητα μια λέξης (TF) και την αντίστροφη συχνότητα μιας λέξης (IDF). Κάθε λέξη έχει την αντίστοιχη βαθμολογία TF και IDF:

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

Όπου:

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

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

Με απλά λόγια, όσο υψηλότερη είναι η βαθμολογία TF * IDF (βάρος), τόσο πιο σπάνια είναι η λέξη και αντίστροφα.

Το πιο σημαντικό, ελέγχει πόσο σχετική είναι η λέξη σε όλα τα δεδομένα. Για μια λέξη t σε ένα tweet D, το βάρος W(t, d) ή idf(t, D) της λέξης t στο tweet d δίνεται από:

$$W(t, D) = \log\left(\frac{N}{DF(t)}\right)$$

Όπου:

- $TF(t, D)$ είναι ο αριθμός των εμφανίσεων του t στο tweet D .
- $DF(t)$ είναι ο αριθμός των tweets που διαθέτουν τη λέξη t .
- N είναι ο συνολικός αριθμός tweets στο σύνολο δεδομένων.

Το TF (συχνότητα όρου) μιας λέξης αναφέρει πόσες φορές εμφανίζεται στα δεδομένα και έτσι δείχνει αν χρησιμοποιείτε μια λέξη πάρα πολύ ή πολύ λίγο.

Για παράδειγμα, όταν ένα tweet 100 λέξεων περιέχει τη λέξη «κρούσμα» 18 φορές, το TF για τη λέξη «κρούσμα» είναι:

$$TF_{\text{κρούσμα}} = \frac{18}{100}, \text{ δηλαδή } 0,18$$

Το IDF (αντίστροφη συχνότητα) μιας λέξης είναι το μέτρο με το οποίο δηλώνει πόσο σημαντική είναι αυτή η λέξη σε ολόκληρο το σύνολο.

3.6. Μηχανική μάθηση

Η Μηχανική Μάθηση είναι ένα ερευνητικό πεδίο της Τεχνητής Νοημοσύνης που στοχεύει στη δημιουργία μηχανών ικανών να μαθαίνουν. Συγκεκριμένα αξιοποιώντας ένα σύνολο δεδομένων, συνήθως γνώση που έχει αποκτηθεί από προηγούμενη εκτέλεση μιας εργασίας, κατασκευάζονται μοντέλα ή πρότυπα τα οποία αν ακολουθηθούν βελτιώνουν την απόδοσή των συστημάτων.

Για τη δημιουργία ενός προτύπου αναπτύσσονται αλγόριθμοι, βασισμένοι σε ένα σύνολο εκπαίδευσης, προκειμένου τα δεδομένα να ταξινομηθούν σε κατηγορίες ή να γίνει πρόβλεψη για κάποια τάση ή συμπεριφορά αυτών. Το μεγάλο πλήθος δεδομένων αυξάνεται καθημερινά με αποτέλεσμα ο άνθρωπος να καθίσταται αδύναμος να επεξεργαστεί όλον αυτό τον όγκο. Λύση στο πρόβλημα δίνουν οι παραπάνω τομείς παρέχοντας τεχνικές που εξυπηρετούν στην καλύτερη αποθήκευση και επεξεργασία της πληροφορίας.

Υπάρχουν τρία είδη μάθησης:

- **Επιβλεπόμενη Μάθηση (Supervised learning):** ο αλγόριθμος δημιουργεί μια συνάρτηση λαμβάνοντας ως εισόδους ένα σύνολο στιγμιότυπων εκπαίδευσης με γνωστές εξόδους. Στόχος της είναι η γενίκευση της συνάρτησης ώστε να απεικονίζονται και δεδομένα εισόδου με άγνωστη έξοδο. Εφαρμογή βρίσκει σε προβλήματα ταξινόμησης (classification), πρόγνωσης (prediction) και διερμηνείας (Interpretation).
- **Μη επιβλεπόμενη Μάθηση (Unsupervised Learning):** ο αλγόριθμος προσπαθεί να

ανακαλύψει τυχόν συσχετίσεις μεταξύ των στιγμιότυπων εισόδου με άγνωστη έξοδο προκειμένου να βρεθούν δομικοί σχηματισμοί τους. Εφαρμογή βρίσκει σε προβλήματα ανάλυσης συσχετισμών (association analysis) και ομαδοποίησης (clustering).

- Ενισχυτική Μάθηση (Reinforcement Learning): ο αλγόριθμος μαθαίνει μια στρατηγική ενεργειών μέσα από την άμεση παρατήρηση του τρόπου λειτουργίας ενός δυναμικού περιβάλλοντος, χωρίς κάποιος να του υποδείξει το υλικό εκπαίδευσης. Εφαρμογή βρίσκει σε προβλήματα σχεδιασμού (planning) όπως είναι ένα παιχνίδι σκάκι ή η κίνηση ενός ρομπότ

3.6.1. Αλγόριθμος Bayes

Ο ταξινομητής Bayes [38] βασίζεται στην ιδέα ότι η κατηγοριοποίηση ενός νέου στιγμιότυπου σχετίζεται με την κατανομή των πιθανοτήτων που έχουν τα υπόλοιπα στιγμιότυπα του συνόλου δεδομένων αναφορικά με τη κλάση που ανήκουν. Οι Bayes ταξινομητές βασίζονται στο θεώρημα του Bayes το οποίο διατυπώνεται συνοπτικά ως εξής:

$$P(a|X) = \frac{P(X|a)P(a)}{P(X)}$$

όπου X ένα στιγμιότυπο που θέλουμε να ταξινομήσουμε, a οι δυνατές τιμές της κλάσης, $P(a)$ η πιθανότητα να πραγματοποιηθεί το a , $P(X)$, δηλαδή η πιθανότητα να πραγματοποιηθεί το X ανεξαρτήτως του a και $P(X|a)$, δηλαδή η πιθανότητα να πραγματοποιηθεί το X δεδομένου του a . Για να προβλέψει την κλάση ενός στιγμιότυπου ένας ταξινομητής Bayes, επιλέγει την κλάση που μεγιστοποιεί την $P(X|a)$.

3.6.2. Αλγόριθμος Decision Tree

Ο αλγόριθμος με δέντρο απόφασης [39] χρησιμοποιείται πολύ συχνά σε προβλήματα ταξινόμησης, αφού το μοντέλο εκπαίδευσης αναπαρίσταται με μια δεντρική δομή, το οποίο καθιστά εύκολα κατανοητή την διαδικασία κατηγοριοποίησης. Κάθε δένδρο απόφασης έχει δύο ειδών κόμβους, τους εσωτερικούς στους οποίους αντιστοιχίζονται τα γνωρίσματα των δεδομένων και λειτουργούν σαν έλεγχοι και τους τερματικούς κόμβους που αναθέτει στιγμιότυπα που καταλήγουν σε αυτό την κατηγορία της κλάσης που πιθανός ανήκει. Σε κάθε ακμή που φεύγει από κάποιο κόμβο αναγράφονται οι κατηγορίες των γνωρισμάτων του κόμβου εκτός και αν η ακμή καταλήγει στην τελική κατηγορία της κλάσης του στιγμιότυπου. Ένα δέντρο απόφασης δεδομένου ενός συνόλου δεδομένων εκπαίδευσης δημιουργείται με μια συγκεκριμένη διαδικασία.

Αρχικά, επιλέγεται ως ρίζα του δέντρου εκείνο το γνώρισμα που πετυχαίνει καλύτερο διαχωρισμό των στιγμιότυπων ως προς την κλάση. Έπειτα, χωρίζονται τα δεδομένα ως προς τις κατηγορίες του συγκεκριμένου γνωρίσματος και για κάθε υποσύνολο που περιέχει στιγμιότυπα

διαφορετικών κλάσεων επαναλαμβάνεται η διαδικασία έως ότου χρησιμοποιηθούν όλα τα γνωρίσματα του συνόλου εκπαίδευσης και δεν υπάρχουν υποσύνολα που να περιέχουν περισσότερες από μία κατηγορίες. Έστω ότι έχουμε στα χέρια μας ένα στιγμιότυπο και αναζητούμε την κλάση στην οποία έχοντας φτιάξει ένα δέντρο απόφασης με βάση το σύνολο εκπαίδευσης, τότε η διαδικασία για την ταξινόμηση είναι αρκετά απλή. Αρχικά, ξεκινάμε από τη ρίζα του δέντρου και ακολουθούμε την ακομή που αντιστοιχεί στην τιμή του αντίστοιχου γνωρίσματος του άγνωστου στιγμιότυπου. Έπειτα, πηγαίνουμε στον επόμενο κόμβο και επαναλαμβάνουμε την ίδια διαδικασία μέχρι να φτάσουμε στον τελικό κόμβο που οδηγεί και στην κατηγορία της κλάσης του στιγμιότυπου.

Έχοντας ένα σύνολο απόφασης μπορούμε να δημιουργήσουμε διαφορετικά δέντρα. Το κριτήριο που τα διαφοροποιεί είναι σε ποιους κόμβους θα βάλουμε τα γνωρίσματα. Κάθε φορά επιλέγουμε ένα γνώρισμα ανάλογα με το πόσο καλά διαχωρίζει τα δείγματα του συνόλου με βάση τις κατηγορίες της κλάσης.[10]

3.6.3. Αλγόριθμος Random Forest

Το μοντέλο αυτό [40], είναι ένα σύνολο αλγορίθμων δέντρων αποφάσεων. Είναι μια επέκταση των δέντρων αποφάσεων και μπορεί να χρησιμοποιηθεί για προβλήματα ταξινόμησης. Στην αποθήκευση, δημιουργείται ένας αριθμός δέντρων αποφάσεων όπου κάθε δέντρο δημιουργείται από ένα διαφορετικό δείγμα bootstrap (στατιστική εκτίμηση) του συνόλου δεδομένων εκπαίδευσης. Ένα bootstrap είναι ένα δείγμα του συνόλου δεδομένων εκπαίδευσης όπου ένα δείγμα μπορεί να εμφανιστεί περισσότερες από μία φορές στο δείγμα, που αναφέρεται ως δειγματοληψία με αντικατάσταση. Δηλαδή είναι ένα είδος μάθησης που λειτουργεί με την κατασκευή ενός πλήθους δέντρων απόφασης κατά το χρόνο εκπαίδευσης. Βασικό πλεονέκτημα του ταξινομητή Random Forest είναι η ικανότητά του να διαχειρίζεται αποτελεσματικά μεγάλο αριθμό λέξεων και ο μικρός χρόνος εκτέλεσής του.

3.6.4. Αλγόριθμος Logistic Regression

Ο αλγόριθμος logistic regression [42] είναι μια μέθοδος στατιστικής ανάλυσης για την πρόβλεψη ενός δυαδικού αποτελέσματος, όπως ναι ή όχι, με βάση προηγούμενες παρατηρήσεις ενός συνόλου. Ο αλγόριθμος αυτός, προβλέπει μια μεταβλητή δεδομένων αναλύοντας τη σχέση ανάμεσα σε μία ή περισσότερες υπάρχουσες ανεξάρτητες μεταβλητές.

$$\text{logit}(\hat{\pi}) = \log\left(\frac{\hat{\pi}}{1 - \hat{\pi}}\right)$$

Ο αλγόριθμος της λογιστικής παλινδρόμησης δίνεται από τον παρακάτω τύπο:

$$\text{logit}(\hat{\pi}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_m X_m$$

και η εκτιμώμενη πιθανότητα π υπολογίζεται έως :

$$\hat{\pi} = \frac{\exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_m X_m)}{1 + \exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_m X_m)}$$

3.6.5. Αλγόριθμος Gradient Boosting

Ο αλγόριθμος gradient boosting είναι μια από τις πιο ισχυρές ιδέες μάθησης που εισήχθησαν τα τελευταία είκοσι χρόνια. Συγκεκριμένα, είναι ένας αλγόριθμος μηχανικής μάθησης με επίβλεψη ο οποίος χρησιμοποιείται για προβλήματα ταξινόμησης. Είναι μια τεχνική συνόλου που χρησιμοποιεί πολλά αδύναμα μοντέλα για να δημιουργήσει ένα ισχυρό μοντέλο ταξινόμησης [41].

Το Gradient Boosting βασίζεται στην διαίσθηση ότι το επόμενο είναι το καλύτερο μοντέλο, με συνδυασμό των προηγούμενων μοντέλων, ελαχιστοποιώντας τα συνολικά σφάλματα. Η βασική ιδέα είναι να τεθούν τα αποτελέσματα από τα προηγούμενα μοντέλα στο επόμενο μοντέλο, προκειμένου να ελαχιστοποιηθούν τα σφάλματα.

3.6.6. Αξιολόγηση μοντέλου ταξινόμησης

Μετά την εκπαίδευση του αλγορίθμου σε ένα σύνολο δεδομένων και την επιτυχή δημιουργία ενός μοντέλου ταξινόμησης, είναι σημαντικό να υπάρχει μια ενδεικτική εικόνα για την αξιοπιστία του μοντέλου ή ακόμη και να βελτιωθεί το σύστημα. Έτσι, παρακάτω παρουσιάζονται τρόποι αξιολόγησης του μοντέλου κατηγοριοποίησης.

3.6.7. Κατηγοριοποίηση μοντέλου

Σε αυτήν την εργασία έχουμε δύο κατηγορίες μη πιθανά και πιθανά κρούσματα. Έστω τη θετική και την αρνητική αντίστοιχα. Στην στήλη απόφασης μπορούμε να χωρίσουμε τα αποτελέσματα σε τέσσερις κατηγορίες, όπως φαίνονται στον παρακάτω πίνακα:

Πραγματική Τάξη	Πρόβλεψη Τάξης		
		τάξη=θετική	τάξη=αρνητική
	τάξη=θετική	TP	FN
	τάξη=αρνητική	FP	TN

Εικόνα 11: Πίνακας σύγχυσης

Ο παραπάνω πίνακας ονομάζεται πίνακας σύγχυσης (confusion matrix) και οι μεταβλητές που περιέχει περιγράφουν τα εξής:

- TP (True Positive): το πλήθος των στοιχείων τα οποία ο ταξινομητής προβλέπει σωστά ως

θετικά (μη κρούσματα)

- TN (True Negative): το πλήθος των στοιχείων τα οποία ο ταξινομητής προβλέπει σωστά ως αρνητικά (κρούσματα)

- FP (False Positive): το πλήθος των στοιχείων τα οποία ο ταξινομητής προβλέπει ως θετικά (μη κρούσματα) αλλά στην πραγματικότητα ανήκουν στην αρνητική κατηγορία (κρούσματα).

- FN (False Negative): το πλήθος των στοιχείων τα οποία ο ταξινομητής προβλέπει ως αρνητικά (κρούσματα) αλλά στην πραγματικότητα ανήκουν στην θετική κατηγορία (μη κρούσματα).

Για το μέτρο της αξιοπιστίας του αλγορίθμου χρησιμοποιούμε αρκετές μετρικές κάποιες από τις οποίες αναφέρονται παρακάτω, υποθέτοντας ότι ο αριθμός των δειγμάτων είναι n :

- i. Ακρίβεια (accuracy) είναι το ποσοστό των σωστών προβλέψεων του αλγορίθμου για κάθε κατηγορία και ορίζεται ως:

$$accuracy = \frac{TP + TN}{n}$$

- ii. Ακρίβεια (precision) εξειδικευμένη για κάθε κατηγορία, δηλαδή για τη θετική και την αρνητική κατηγορία αντίστοιχα δίνεται από τις σχέσεις:

$$precision_{positive} = \frac{TP}{TP + FP}$$

$$precision_{negative} = \frac{TN}{TN + FN}$$

- iii. Ποσοστό σφαλμάτων (error rate) είναι το ποσοστό των ψευδών παρατηρήσεων, δηλαδή:

$$error\ rate = \frac{FP + FN}{n}$$

- iv. Ανάκληση (recall) για κάθε κατηγορία είναι το ποσοστό των σωστών προβλέψεων σε σχέση με όλα τα στοιχεία της θετικής και αρνητικής κατηγορίας αντίστοιχα:

$$recall_{positive} = \frac{TP}{TP + FN}$$

$$recall_{negative} = \frac{TN}{TN + FP}$$

- v. F-μέτρο (f-measure). Η μετρική αυτή προσπαθεί να εξισορροπήσει τις τιμές της ακρίβειας και της ανάκλησης υπολογίζοντας τον αρμονικό τους μέσο. Για κάθε κατηγορία έχουμε:

$$F_{positive} = \frac{2 \cdot precision_{positive} \cdot recall_{positive}}{precision_{positive} + recall_{positive}}$$

$$F_{negative} = \frac{2 \cdot precision_{negative} \cdot recall_{negative}}{precision_{negative} + recall_{negative}}$$

4. ΤΕΛΙΚΗ ΑΝΑΛΥΣΗ

4.1. Αρχιτεκτονική δεδομένων

Όπως συζητήθηκε προηγουμένως, τα δεδομένα που έχουν συλλεχθεί είναι ανισόρροπα. Είναι συνολικά 17601 από τα οποία τα πιθανά tweets αποτελούν το 10% των συνολικών δεδομένων. Τονίζεται, ότι η δημιουργία ενός μοντέλου δεν είναι ιδανική σε ένα μη ισορροπημένο σύνολο δεδομένων, διότι το μοντέλο δεν θα είναι σε θέση να ξεχωρίσει τις λέξεις οι οποίες είναι πιθανά κρούσματα. Εάν χρησιμοποιηθεί το συγκεκριμένο μοντέλο με δεδομένα τα οποία δεν έχουν ίδια αναλογία με πιθανά και όχι κρούσματα τότε τα αποτελέσματα δεν θα είναι αξιόπιστα. Σε αυτό το σύνολο δεδομένων το μοντέλο προβλέπει κάθε tweet ως απλό tweet λόγω έλλειψης μάθησης.

Επομένως, χωρίζονται τα δεδομένα σε ένα σύνολο εκπαίδευσης (train data) και ένα σύνολο ελέγχου (test data). Σημειώνεται, ότι τα σύνολα εκπαίδευσης και ελέγχου παραμένουν σε αναλογία 90 προς 10.

4.1.1. Δεδομένα εκπαίδευσης

Σε αυτά τα δεδομένα, εφαρμόζεται η μέθοδος SMOTE (Synthetic Minority Oversampling Technique) [35] στα πιθανά tweets ώστε να εξισορροπηθούν τα δεδομένα. Η διαδικασία γίνεται με την δημιουργία ενός συνόλου από όλες τις λέξεις που υπάρχουν σε πιθανά tweet κρουσμάτων. Μόλις αποκτηθεί ένας πίνακας που περιέχει όλες τις λέξεις με πιθανά κρούσματα, τότε δημιουργείται ένα νέο σύνολο δεδομένων και εισάγονται με νέα πιθανά tweets. Αυτά τα νέα tweets πραγματοποιήθηκαν επιλέγοντας λέξεις τυχαία από τα δεδομένα.

```
from sklearn.model_selection import train_test_split
X_temp, X_test, y_temp, y_test = train_test_split(dff, list(df.label), test_size=0.1)
ds_temp = ds
testdf = X_test
testdf['label'] = y_test
testdf.shape
ds = pd.concat([ds_temp, testdf], axis = 0)
ds.head()
ds.shape
testdf.head()
list(testdf.index)
#νέο σύνολο
```

```
corpus = []
for i in range(ds.shape[0]):
    corpus.append(ds.iloc[i][0])

tdf_covid = tdf[tdf.labelxyz == 1]
tdf_covid.shape

tdf_noncovid = tdf[tdf.labelxyz == 0]
tdf_noncovid.shape
```

	00	000	009	01	02	028	03	030	04	043	...	ύποπτο	ύστερα	ύψη	ύψους	ώπα	ώρα	ώρες
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3357	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.281082
3358	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000
3359	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000
3360	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000
3361	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000

3362 rows × 6178 columns

Εικόνα 12: Σύνολο από μοναδικές λέξεις

Στη συνέχεια, ξαναέγινε αυτή η διαδικασία αυτόματα, ώσπου ο αριθμός των πιθανών tweet ήταν περίπου ίσος με τον αριθμό των tweet που δεν είναι πιθανά κρούσματα στα προσωρινά δεδομένα.

```
In [33]: # pithana (1769)
tdf_covid.shape
```

```
Out[33]: (1769, 6179)
```

```
In [34]: tdf_noncovid = tdf[tdf.labelxyz == 0]
```

```
In [35]: # mi pithana (1593)
tdf_noncovid.shape
```

```
Out[35]: (1593, 6179)
```

Εικόνα 13: Αριθμός tweets

```

X_train_covid = tdf_covid.sample(frac=0.9, random_state=0)
X_test_covid = tdf_covid.drop(X_train_covid.index)
X_train_noncovid = tdf_noncovid.sample(frac=0.406, random_state=0)
X_test_noncovid = tdf_noncovid.drop(X_train_noncovid.index)
X_train_df = pd.concat([X_train_covid, X_train_noncovid], axis = 0)
X_train = X_train_df.drop(['labelxyz'], axis = 1)
y_train = list(X_train_df.labelxyz)
X_test_df = pd.concat([X_test_covid, X_test_noncovid], axis = 0)
X_test = X_test_df.drop(['labelxyz'], axis = 1)
y_test = list(X_test_df.labelxyz)

```

Ωστόσο, στην προσέγγιση Bag of Words [36] για τη δημιουργία συνόλου των λέξεων, γίνεται με δειγματοληψία στα tweets που δεν είναι πιθανά για να γίνουν ομοιόμορφα. Πάρθηκε ένα υποσύνολο μόνο των tweets χωρίς κρούσματα από το σύνολο εκπαίδευσης. Από αυτό το υποσύνολο, περιέχει ένα δείγμα από tweets, όπου ο αριθμός είναι ίσος με τον αριθμό των πιθανών κρουσμάτων tweets στα δεδομένα εκπαίδευσης. Στη συνέχεια, εντάχθηκε με το υποσύνολο των πιθανών κρουσμάτων tweets. Αυτό το σύνολο δεδομένων είναι τα δεδομένα εκπαίδευσης για το Bag of Words που χρησιμοποιείται για τους σκοπούς δημιουργίας και μοντελοποίησης.

```

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
X = cv.fit_transform(corpus)
bow = pd.DataFrame(X.toarray(), columns = cv.get_feature_names())
bow['labelxyz'] = list(ds.label)
bow_covid = bow[bow.labelxyz == 1]
bow_noncovid = bow[bow.labelxyz == 0]
X_train_covid = bow_covid.sample(frac=0.9, random_state=0)
X_test_covid = bow_covid.drop(X_train_covid.index)
X_train_noncovid = bow_noncovid.sample(frac=0.406, random_state=0)
X_test_noncovid = bow_noncovid.drop(X_train_noncovid.index)
X_train_df = pd.concat([X_train_covid, X_train_noncovid], axis = 0)
X_train = X_train_df.drop(['labelxyz'], axis = 1)
y_train = list(X_train_df.labelxyz)
X_test_df = pd.concat([X_test_covid, X_test_noncovid], axis = 0)

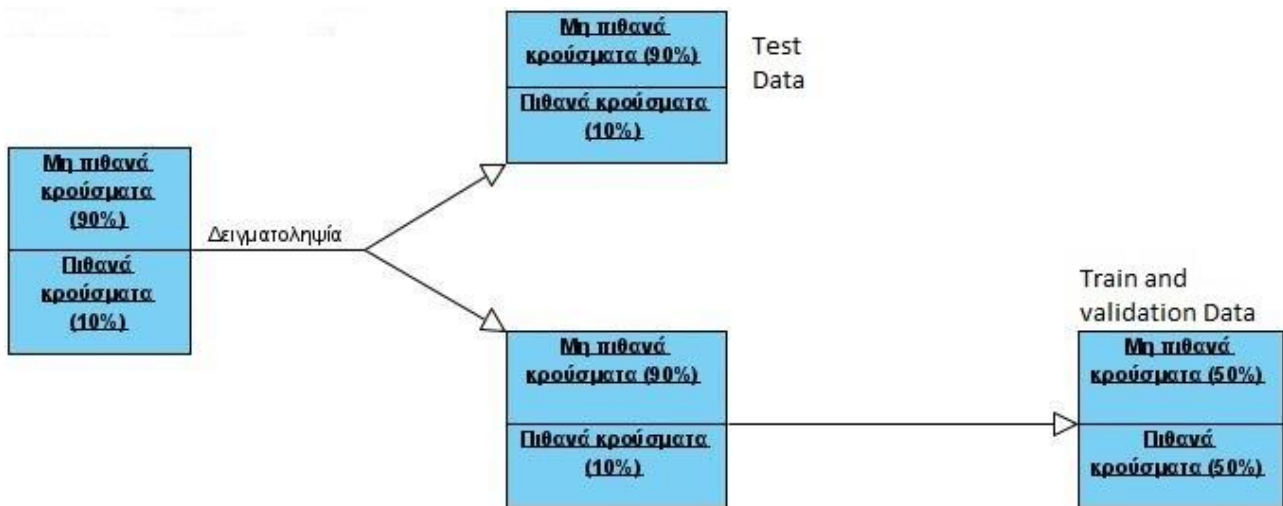
```

```
X_test = X_test_df.drop(['labelxyz'], axis = 1)
```

```
y_test = list(X_test_df.labelxyz)
```

4.1.2. Δεδομένα ελέγχου

Τα δεδομένα ελέγχου εξακολουθούν να είναι σε αναλογία 90 προς 10 (μη πιθανά tweets προς πιθανά tweets αντίστοιχα), καθώς δεν πραγματοποιήθηκε δειγματοληψία. Η δειγματοληψία δεν πραγματοποιήθηκε επειδή τα αρχικά δεδομένα έρχονται σε αυτήν την αναλογία.



Εικόνα 14: Δεδομένα ελέγχου και εκπαίδευσης

4.2. Ανάλυση διαδικασίας

Τόσο για το Bag of Words [36] όσο και για το TFIDF [37], χρησιμοποιείται η μέθοδος CV (cross validation) η οποία βοηθά στην καλύτερη αξιολόγηση των μοντέλων (Logistic Regression και Naive Bayes). Με την μέθοδο αυτή, χωρίζεται το σύνολο δεδομένων σε 3 υποσύνολα δεδομένων, με σύνολα εκπαίδευσης και ελέγχου. Σημειώνεται ότι τα συγκεκριμένα σύνολα είναι διαφορετικά μεταξύ τους. Στην συνέχεια, χρησιμοποιούνται τα 2 σύνολα για την εκπαίδευση του αλγορίθμου και 1 για την αξιολόγηση του μοντέλου. Επαναλαμβάνεται η διαδικασία 3 φορές και υπολογίζονται οι 3 ακρίβειες από την κάθε αξιολόγηση. Η συνολική ακρίβεια του μοντέλου προκύπτει από το μέσο όρο των αξιολογήσεων. Με λίγα λόγια τα σύνολα δεδομένων εκπαιδεύονται χωρισμένα σε 3 υποσύνολα ως δεδομένα εκπαίδευσης και το μοντέλο που προκύπτει εφαρμόζεται στα υπόλοιπα δεδομένα.

```
# Use Cross-validation.
```

```
from sklearn.model_selection import cross_val_score
```

```
# Logistic Regression
```

```
log_reg = LogisticRegression()
```

```
log_scores = cross_val_score(log_reg, X_train, y_train, cv=3)
log_reg_mean = log_scores.mean()
```

```
# Naives Bayes
```

```
nav_clf = GaussianNB()
nav_scores = cross_val_score(nav_clf, X_train, y_train, cv=3)
nav_mean = nav_scores.mean()
```

```
# Create a Dataframe with the results.
```

```
d = {'Classifiers': ['Logistic Reg.', 'Naives Bayes'],
     'Crossval Mean Scores': [log_reg_mean, nav_mean]}
```

```
result_df = pd.DataFrame(data=d)
```

Πίνακας 3:Αποτελέσματα CV

Classifiers	CV Mean Scores
Logistic Regression	0.817326
Naives Bayes	0.831617

Με τα παραπάνω αποτελέσματα διαπιστώνεται ότι τα δεδομένα αποδίδουν με καλές βαθμολογίες. Στην συνέχεια, για τον έλεγχο σε όλα τα δεδομένα εκτελούνται οι 5 αλγόριθμοι ταξινόμησης, δηλαδή οι Logistic Regression, Naive Bayes, Decision Tree, Random Forest και Gradient Boosting. Με την εκτέλεση τους αναφέρονται τα αποτελέσματα σχετικά με την ακρίβεια των αλγορίθμων και την ανάκληση των πιθανών κρουσμάτων. Επιπλέον, εφαρμόστηκε μείωση διαστάσεων στα δεδομένα και στις δύο προσεγγίσεις ώστε να γίνει καλύτερη η κατανόηση των αποτελεσμάτων. Ο βασικός κώδικας των αλγορίθμων αυτών αναδεικνύεται παρακάτω.

4.2.1. Logistic Regression

Το Logistic Regression [42] είναι ένας αλγόριθμος ταξινόμησης μηχανικής μάθησης που χρησιμοποιείται για να προβλέψει την πιθανότητα μιας εξαρτημένης μεταβλητής. Είναι μια δυαδική μεταβλητή που περιέχει δεδομένα ως 1 (ναι) ή 0 (όχι).

```
from sklearn.metrics import accuracy_score
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
predict_log = log_reg.predict(X_test)
accuracy_score(y_test, predict_log)
```

4.2.2. Naives Bayes

```
nav_clf = GaussianNB()
nav_clf.fit(X_train, y_train)
```

```
predict_nav = nav_clf.predict(X_test)
accuracy_score(y_test, predict_nav)
```

4.2.3. Random Forest

Το μοντέλο αυτό [40], είναι ένα σύνολο από δέντρα απόφασης. Είναι μια επέκταση των δέντρων αποφάσεων και χρησιμοποιείται στην ταξινόμηση.

```
from sklearn.ensemble import RandomForestClassifier
rand_clf = RandomForestClassifier()
rand_clf.fit(X_train, y_train)
predict = rand_clf.predict(X_test)
accuracy_score(y_test, predict)
```

4.2.4. Decision Tree

Ο αλγόριθμος decision tree [39] χρησιμοποιείται για την ταξινόμηση αφού αναπαριστά μια δομή από δέντρο και την καθιστά εύκολα κατανοητή στην διαδικασία κατηγοριοποίησης.

```
from sklearn import tree
dt_clf = tree.DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)
predict_dt = dt_clf.predict(X_test)
accuracy_score(y_test, predict_dt)
```

4.2.5. Gradient Boosting

Το Gradient Boosting [41] βασίζεται σε έναν ισχυρό αλγόριθμο μηχανικής μάθησης που χρησιμοποιεί δέντρα αποφάσεων.

```
from sklearn.ensemble import GradientBoostingClassifier
grad_clf = GradientBoostingClassifier()
grad_clf.fit(X_train, y_train)
predict_grad = grad_clf.predict(X_test)
accuracy_score(y_test, predict_grad)
```

4.3. Ανάλυση Term Frequency Inverse Document Frequency

Το μοντέλο TF-IDF περιέχει πληροφορίες για τις πιο σημαντικές λέξεις και διαθέτει χαμηλό βάρος σε ασήμαντες λέξεις. Επομένως, εάν υπάρχει ένα tweet που περιέχει αυτές τις λέξεις, ενδέχεται να μην προορίζεται ως πιθανό tweet. Έτσι, υπολογίζεται μια μέτρηση πόσες φορές αυτή η λέξη εμφανίστηκε στα δεδομένα. Μετά την εκτέλεση των αλγορίθμων, λαμβάνουμε τα ακόλουθα αποτελέσματα.

```
from sklearn.metrics import accuracy_score
```

```
nav_clf = GaussianNB()
```

```
nav_clf.fit(X_train, y_train)
```

```
predict_nav = nav_clf.predict(X_test)
```

```
accuracy_score(y_test, predict_nav)
```

```
0.7479964381121995
```

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix(y_test, predict_nav)
```

```
[671  275]
 [ 8   169]
```

```
log_reg = LogisticRegression()
```

```
log_reg.fit(X_train, y_train)
```

```
predict_log = log_reg.predict(X_test)
```

```
accuracy_score(y_test, predict_log)
```

```
0.5841495992876224
```

```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix(y_test, predict_log)
```

```
[480  466]
 [ 1   176]
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
rand_clf = RandomForestClassifier()
```

```
rand_clf.fit(X_train, y_train)
```

```
predict = rand_clf.predict(X_test)
```

```
accuracy_score(y_test, predict)
```

```
0.7043633125556545
```

```
confusion_matrix(y_test, predict)
```

```
[619  327]
 [ 5   172]
```

```
172/(172+5)
```

```
0.9717514124293786
```

```
from sklearn import tree
```

```
dt_clf = tree.DecisionTreeClassifier()
```

```
dt_clf.fit(X_train, y_train)
```

```

predict_dt = dt_clf.predict(X_test)
accuracy_score(y_test, predict_dt)
0.7631344612644702
confusion_matrix(y_test, predict_dt)
[[693  253]
 [ 13  164]]

```

```

from sklearn.ensemble import GradientBoostingClassifier
grad_clf = GradientBoostingClassifier()
grad_clf.fit(X_train, y_train)
predict_grad = grad_clf.predict(X_test)
accuracy_score(y_test, predict_grad)
0.5138023152270703
confusion_matrix(y_test, predict_grad)
[[403  543]
 [  3  174]]

```

Πίνακας 4: Αποτελέσματα tfidf

Algorithm	Accuracy	Recall(positive)
Logistic Regression Score	0.58	0.75
Naive Bayes Score	0.74	0.95
Random Forest Score	0.70	0.97
Decision Tree Score	0.76	0.92
Gradient Boosting Score	0.51	0.98

Οι αλγόριθμοι Naive Bayes, Decision Tree και Random Forest μας δίνουν τα καλύτερα αποτελέσματα για ακρίβεια, καθώς και ανάκληση.

4.3.1. TF-IDF για συχνότερες λέξεις

Αρχικά, γίνεται μετατροπή του κώδικα για την εύρεση των 1000 συχνότερων λέξεων χρησιμοποιώντας την βιβλιοθήκη collections [43]. Αυτό γίνεται τόσο για το TFIDF όσο και για το Bag of Words. Ο παρακάτω κώδικας, δημιούργησε έναν πίνακα με αυτές τις λέξεις ώστε να ξαναγίνει η εκπαίδευση με στόχο την καλύτερη αξιολόγηση των αποτελεσμάτων.

```

words = []
for k in range(len(corpus)):
    tweet_k = corpus[k].split(" ")

```

```

for m in range(len(tweet_k)):
    words.append(tweet_k[m])
from collections import Counter
counter_obj = Counter(words)
top100 = counter_obj.most_common(1000)
top100words = []
for i in range(len(top100)):
    top100words.append(top100[i][0])

```

Εδώ, χωρίζονται πάλι αυτές οι 1000 λέξεις σε δεδομένα εκπαίδευσης και ελέγχου μέσω δειγματοληψίας που αναφέρθηκε προηγουμένως στην αρχιτεκτονική δεδομένων.

```

tdftop = df2
lst3 = [value for value in list(tdftop.columns) if value in top100words]
tdf100 = tdftop[lst3]
tdf100['labelxyz'] = list(ds.label)
tdf_covid = tdf100[tdf100.labelxyz == 1]
tdf_noncovid = tdf100[tdf100.labelxyz == 0]
X_train_covid = tdf_covid.sample(frac=0.9, random_state=0)
X_test_covid = tdf_covid.drop(X_train_covid.index)
X_train_noncovid = tdf_noncovid.sample(frac=0.406, random_state=0)
X_test_noncovid = tdf_noncovid.drop(X_train_noncovid.index)
X_train_df = pd.concat([X_train_covid, X_train_noncovid], axis = 0)
X_train = X_train_df.drop(['labelxyz'], axis = 1)
y_train = list(X_train_df.labelxyz)
X_test_df = pd.concat([X_test_covid, X_test_noncovid], axis = 0)
X_test = X_test_df.drop(['labelxyz'], axis = 1)
y_test = list(X_test_df.labelxyz)

```

Στην συνέχεια αναδεικνύεται ο πίνακας των αποτελεσμάτων και τους πίνακες σύγχυσης:

Πίνακας 5: Αποτελέσματα tfidf για τις συχνότερες λέξεις

Algorithm	Precision	Recall
Logistic Regression Score	0.53	0.5
Naive Bayes Score	0.82	0.79

Random Forest Score	0.78	0.93
Decision Tree Score	0.75	0.79
Gradient Boosting Score	0.50	0.98

naive bayes

0.8210151380231523

```
[781 165]
[ 36 141]
```

random forest

0.7818343722172751

```
[712 234]
[ 11 166]
```

decision tree

0.7586821015138023

```
[781 165]
[ 36 141]
```

gradient boosting

0.5066785396260017

```
[394 552]
[  2 175]
```

Μαζί με τα αποτελέσματα βλέπουμε και τους πίνακες σύγχυσης για τον κάθε αλγόριθμο . Η αξιολόγηση των δεδομένων όπως παρουσιάζεται στους πίνακες σύγχυσης έχει μεγάλο ποσοστό ταξινόμησης. Αναδεικνύεται ο παρακάτω κώδικας που πραγματοποίησε τα παραπάνω αποτελέσματα:

```
from sklearn.metrics import accuracy_score
nav_clf.fit(X_train, y_train)
predict_nav = nav_clf.predict(X_test)
print('naive bayes')
print(accuracy_score(y_test, predict_nav))
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, predict_nav))
```

```

rand_clf = RandomForestClassifier()
rand_clf.fit(X_train, y_train)
predict = rand_clf.predict(X_test)
print('random forest')
print(accuracy_score(y_test, predict))
print(confusion_matrix(y_test, predict))

dt_clf = tree.DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)
predict_dt = dt_clf.predict(X_test)
print('decision tree')
print(accuracy_score(y_test, predict_dt))
print(confusion_matrix(y_test, predict_dt))

from sklearn.ensemble import GradientBoostingClassifier
grad_clf = GradientBoostingClassifier()
grad_clf.fit(X_train, y_train)
print('gradient boosting')
predict_grad = grad_clf.predict(X_test)
print(accuracy_score(y_test, predict_grad))
print(confusion_matrix(y_test, predict_grad))

```

Τα αποτελέσματα που ελήφθησαν μετά την εκτέλεση των αλγορίθμων με τις συχνότερες λέξεις είναι σχεδόν συνεπή. Ο Naive Bayes και Decision Tree δίνει υψηλή ακρίβεια, το Gradient Boosting πάλι μια καλή ανάκληση και η Logistic Regression δίνει την χαμηλή ακρίβεια και ανάκληση. Το Random Forest δίνει τα καλύτερα αποτελέσματα και για τις δύο αυτές μετρήσεις.

Ωστόσο, δεν υπάρχουν σημαντικές βελτιώσεις για αυτά τα μοντέλα. Οι παραπάνω μετρήσεις βελτιώνονται για ορισμένους αλγόριθμους ενώ μειώνονται για άλλους και η διαφορά με τη οποία βελτιώνεται ή μειώνεται ένας αλγόριθμος είναι δευτερεύουσα.

4.4. Ανάλυση Bag of Words

Σε αυτήν την ενότητα, εφαρμόζεται το Bag of Words στο σύνολο δεδομένων εκπαίδευσης. Πραγματοποιείται ταξινόμηση ότι ένα tweet ανήκει ως πιθανό κρούσμα ή όχι, χωρίς να ξεχωρίζονται οι λέξεις με μεγάλη συχνότητα όπως γίνεται στη προσέγγιση TF-IDF. Τα εκπαιδευτικά δεδομένα είναι συνολικά 3362 tweets τα οποία περιέχουν ένα σύνολο από 6178 μοναδικές λέξεις.

```

from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)
feature_names = vectorizer.get_feature_names()
dense = X.todense()
denselist = dense.tolist()
df2 = pd.DataFrame(denselist, columns=feature_names)
df2
#create label column
tdf = df2
tdf['labelxyz'] = list(ds.label)

```

```
In [31]: tdf.tail()
```

```
Out[31]:
```

	00	000	009	01	02	028	03	030	04	043	...	ύστερα	ύψη	ύψους	ώπα	ώρα	ώρες	Jack	LEADER	the	labelxyz
3357	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.281082	0.0	0.0	0.0	0
3358	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0
3359	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0
3360	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0
3361	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0

5 rows x 6179 columns

Εικόνα 15: Πίνακας μοναδικών λέξεων για το BOW

```

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
X = cv.fit_transform(corpus)
bow = pd.DataFrame(X.toarray(), columns = cv.get_feature_names())

```

Κατά την εκτέλεση των αλγορίθμων ταξινόμησης, ελήφθησαν τα ακόλουθα αποτελέσματα:

```

from sklearn.metrics import accuracy_score
nav_clf.fit(X_train, y_train)
predict_nav = nav_clf.predict(X_test)
print('naive bayes')
print(accuracy_score(y_test, predict_nav))
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, predict_nav))

```

```

0.7471059661620659
[[666  280]
 [ 4   173]]
rand_clf = RandomForestClassifier()
rand_clf.fit(X_train, y_train)
predict = rand_clf.predict(X_test)
print('random forest')
print(accuracy_score(y_test, predict))
print(confusion_matrix(y_test, predict))

0.6981300089047195
[[612  334]
 [ 5   172]]
dt_clf = tree.DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)
predict_dt = dt_clf.predict(X_test)
print('decision tree')
print(accuracy_score(y_test, predict_dt))
print(confusion_matrix(y_test, predict_dt))

0.7230632235084595
[[666  280]
 [ 4   172]]
from sklearn.ensemble import GradientBoostingClassifier
grad_clf = GradientBoostingClassifier()
grad_clf.fit(X_train, y_train)
print('gradient boosting')
predict_grad = grad_clf.predict(X_test)
print(accuracy_score(y_test, predict_grad))
print(confusion_matrix(y_test, predict_grad))

0.4933214603739982
[[380  566]
 [ 3   174]]
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
predict_log = log_reg.predict(X_test)
accuracy_score(y_test, predict_log)

0.7809439002671416

```

$$\begin{bmatrix} 704 & 242 \\ 4 & 173 \end{bmatrix}$$

Με μια πρώτη ματιά, η Logistic Regression είναι το καλύτερο μοντέλο λόγω της υψηλότερης ακρίβειας. Όμως, όπως συζητήθηκε στην αρχιτεκτονική δεδομένων, τα δεδομένα ελέγχου παραμένουν σε αναλογία 90: 10 για πιθανά tweets προς μη πιθανά tweets αντίστοιχα. Έτσι, η υψηλή ακρίβεια μπορεί να μην είναι η καλύτερη μέτρηση.

Συνεπώς, δίνεται μεγάλη βαρύτητα στην ανάκληση, η οποία είναι το ποσοστό των πιθανών tweets που αναγνωρίζονται. Εάν αυτή η μέτρηση είναι υψηλή, θα αντιστοιχούσε σε μεγάλο αριθμό πιθανών tweets που ταξινομήθηκαν σωστά. Σύμφωνα με την ανάκληση, η Logistic Regression φαίνεται να είναι ξανά το καλύτερο μοντέλο.

Επίσης, εάν υπάρχει ένας μεγάλος αριθμός false position τότε είναι πιθανό να ανήκει σε μη πιθανά κρούσματα. Γι' αυτό λοιπόν, δίνεται προσοχή σε μια υψηλή ανάκληση και ταυτόχρονα σε έναν χαμηλό αριθμός false position. Αυτή η προσέγγιση δίνει σε όλα μια υψηλή ανάκληση και επιλέγεται ο καλύτερος αλγόριθμος με βάση την υψηλότερη ακρίβεια του. Έτσι, το Logistic Regression αξιολογείται ως το καλύτερο και αποτελεσματικό μοντέλο αλλά και Naive Bayes και Decision Tree πάλι είναι σχετικά πολύ αποτελεσματικά .

Πίνακας 6: Αποτελέσματα Bag of words

Algorithm	Precision	Recall
Logistic Regression Score	0.78	0.97
Naive Bayes Score	0.74	0.97
Random Forest Score	0.69	0.97
Decision Tree Score	0.72	0.97
Gradient Boosting Score	0.49	0.98

Στην συνέχεια γίνεται αποθήκευση σε γράφημα ο αλγόριθμος Decision Tree [39]:

```
import graphviz
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import tree
from sklearn.datasets import load_wine
from IPython.display import SVG
from graphviz import Source
from IPython.display import display

from six import StringIO
```

```

from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus

dot_data = StringIO()
export_graphviz(dt_clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())

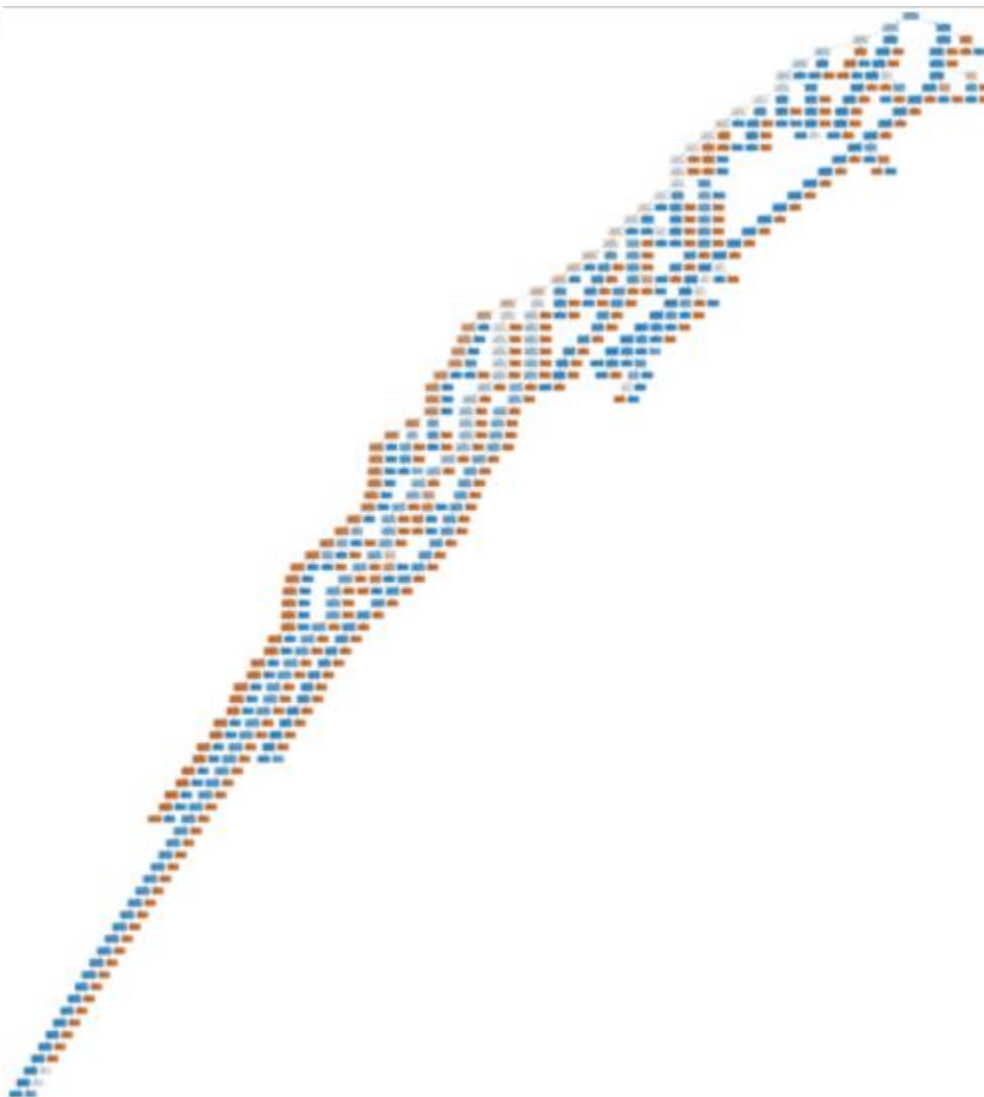
dot_data = tree.export_graphviz(dt_clf,
                                feature_names=list(X_train.columns),
                                out_file=None,
                                filled=True,
                                rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
import collections
colors = ('turquoise', 'orange')
edges = collections.defaultdict(list)

for edge in graph.get_edge_list():
    edges[edge.get_source()].append(int(edge.get_destination()))

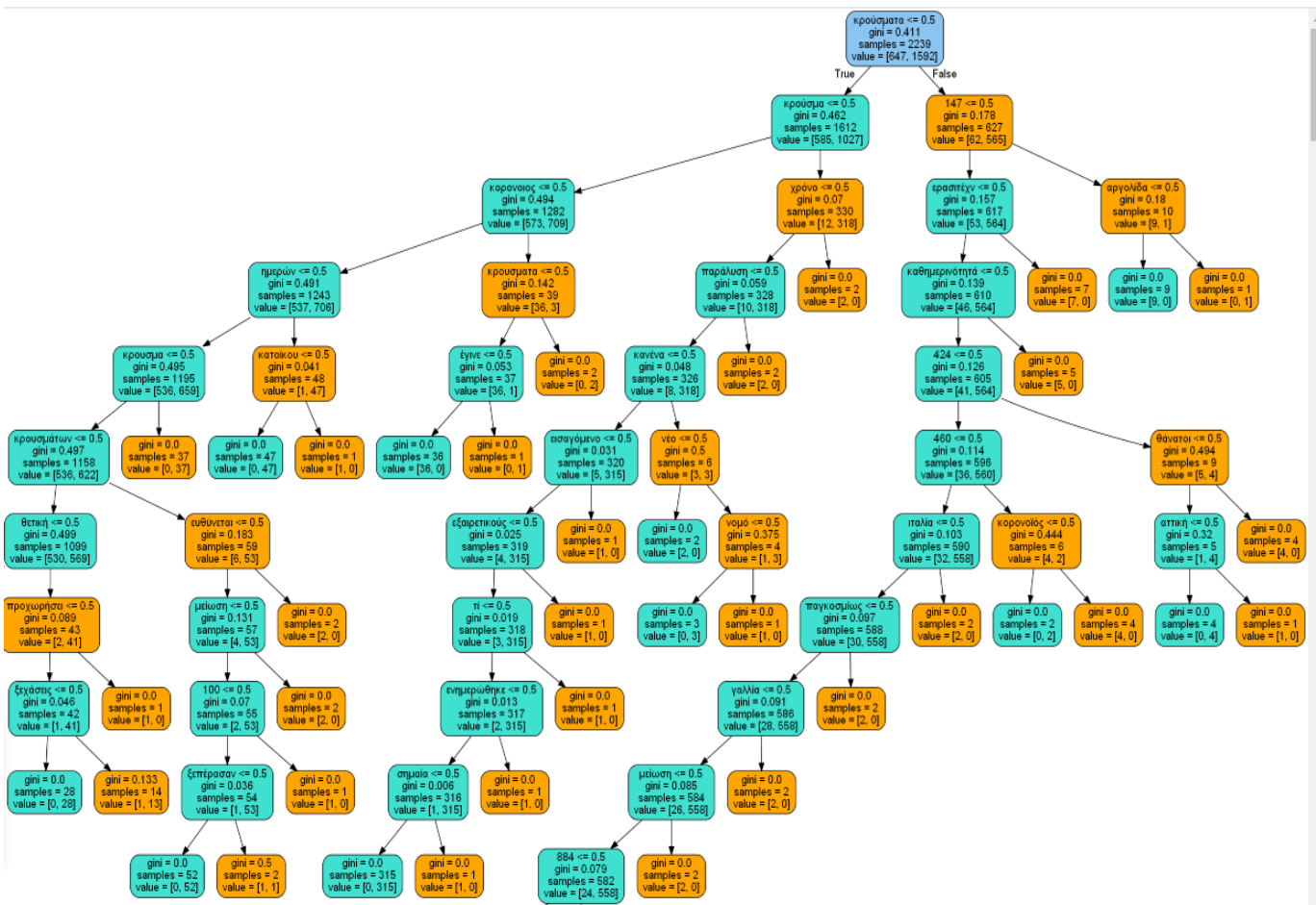
for edge in edges:
    edges[edge].sort()
    for i in range(2):
        dest = graph.get_node(str(edges[edge][i]))[0]
        dest.set_fillcolor(colors[i])
graph.write_png("tree.png")

```

```
out[105]:
```



Εικόνα 16: Δέντρο απόφασης



Εικόνα 17: Δείγμα από το δέντρο απόφασης

Επίσης, χρησιμοποιείται ο αλγόριθμος Extra Trees ώστε να εξεταστεί η σημαντικότητα των δεδομένων εκπαίδευσης και να υπάρχει μια καλύτερη πρόγνωση στο μοντέλο με μείωση διαστάσεων.

```
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
model.fit(X_train, y_train)

print(model.feature_importances_)
importances = rand_clf.feature_importances_

# Sort feature importances in descending order
indices = np.argsort(importances)[::-1]

# Rearrange feature names so they match the sorted feature importances
names = [X_train.columns[i] for i in indices]
```

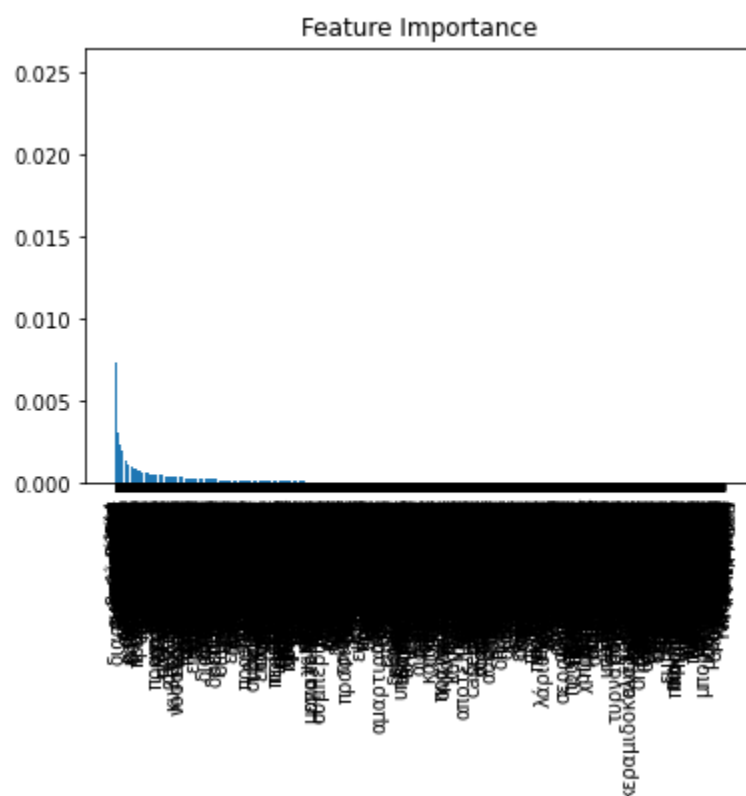
```
# Create plot
plt.figure()

# Create plot title
plt.title("Feature Importance")

# Add bars
plt.bar(range(X_train.shape[1]), importances[indices])

# Add feature names as x-axis labels
plt.xticks(range(X_train.shape[1]), names, rotation=90)

# Show plot
plt.show()
```



Εικόνα 18: Ταξινομητής Extra Trees

4.4.1. Ανάλυση Bag of Words για συχνές λέξεις

Η μείωση των διαστάσεων περιέχει τις 1000 πιο συχνές λέξεις και γίνεται η εκτέλεση των αλγορίθμων ταξινόμησης. Παρακάτω παρουσιάζεται ο σχετικός κώδικας:

```

words = []
for k in range(len(corpus)):
    tweet_k = corpus[k].split(" ")
    for m in range(len(tweet_k)):
        words.append(tweet_k[m])
from collections import Counter
counter_obj = Counter(words)
top100 = counter_obj.most_common(1000)
top100words = []
for i in range(len(top100)):
    top100words.append(top100[i][0])
lst3 = [value for value in list(bow.columns) if value in top100words]
bow100 = bow[lst3]

```

Στην συνέχεια, δημιουργούνται τα δεδομένα ελέγχου και εκπαίδευσης για αυτές τις 1000 λέξεις:

```

bow100['labelxyz'] = list(ds.label)
bow_covid = bow100[bow100.labelxyz == 1]
bow_noncovid = bow100[bow100.labelxyz == 0]
X_train_covid = bow_covid.sample(frac=0.9, random_state=0)
X_test_covid = bow_covid.drop(X_train_covid.index)
X_train_noncovid = bow_noncovid.sample(frac=0.406, random_state=0)
X_test_noncovid = bow_noncovid.drop(X_train_noncovid.index)
X_train_df = pd.concat([X_train_covid, X_train_noncovid], axis = 0)
X_train = X_train_df.drop(['labelxyz'], axis = 1)
y_train = list(X_train_df.labelxyz)
X_test_df = pd.concat([X_test_covid, X_test_noncovid], axis = 0)
X_test = X_test_df.drop(['labelxyz'], axis = 1)
y_test = list(X_test_df.labelxyz)

```

Τα αποτελέσματα που λαμβάνονται καταγράφονται και απεικονίζονται μέσω των αλγορίθμων ως εξής:

```

from sklearn.metrics import accuracy_score
nav_clf.fit(X_train, y_train)
predict_nav = nav_clf.predict(X_test)

```

```

print('naive bayes')
print(accuracy_score(y_test, predict_nav))
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, predict_nav))
0.8130008904719501
[[773  173]
 [ 37  140]]
rand_clf = RandomForestClassifier()
rand_clf.fit(X_train, y_train)
predict = rand_clf.predict(X_test)
print('random forest')
print(accuracy_score(y_test, predict))
print(confusion_matrix(y_test, predict))
0.7934105075690115
[[724  222]
 [ 10  167]]
dt_clf = tree.DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)
predict_dt = dt_clf.predict(X_test)
print('decision tree')
print(accuracy_score(y_test, predict_dt))
print(confusion_matrix(y_test, predict_nav))
0.7747105966162066
[[773  173]
 [ 37  140]]
from sklearn.ensemble import GradientBoostingClassifier
grad_clf = GradientBoostingClassifier()
grad_clf.fit(X_train, y_train)
print('gradient boosting')
predict_grad = grad_clf.predict(X_test)
print(accuracy_score(y_test, predict_grad))
print(confusion_matrix(y_test, predict_grad))
0.5102404274265361
[[398  548]
 [  2  175]]

```

```
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
predict_log = log_reg.predict(X_test)
accuracy_score(y_test, predict_log)
0.7747105966162066
[[698 248]
 [ 5 172]]
```

Πίνακας 7: Αποτελέσματα BOW με τις συχνότερες λέξεις

Algorithm	Precision	Recall
Logistic Regression Score	0.77	0.97
Naive Bayes Score	0.81	0.79
Random Forest Score	0.79	0.94
Decision Tree Score	0.77	0.79
Grad Boosting Score	0.51	0.98

4.5. Ανάλυση κειμένου με bigrams

Με τη μέθοδο Bigrams [1](δηλαδή σύνολο με ζευγάρια των 2 λέξεων) πραγματοποιήθηκε σε ένα νέο σύνολο λέξεων με πιθανά κρούσματα ως bigram σύμφωνα με τον παρακάτω κώδικα:

```
bigramList = []
for i in range(ds.shape[0]):
    s = ds.iloc[i][0]
    s = s.lower()
    tokens = [token for token in s.split(" ") if token != ""]
    output = list(ngrams(tokens, 2))
    bigramList.append(output)

#bigramList[0] - first tweet bigrams
#len(bigramList[0])
actualBigramList = []
for i in range(len(bigramList)):
    #temp = []
    for j in range(len(bigramList[i])):
```

```
actualBigramList.append(" ".join(bigramList[i][j]))
```

```
#actualBigramList.append(temp)
```

```
Out[146]: ['με κορονοϊό',  
          'κορονοϊό διαγνώστηκε',  
          'διαγνώστηκε δικηγόρος',  
          'δικηγόρος χαλκίδας',  
          'χαλκίδας -',  
          '- χρήστος',  
          'χρήστος θεοφίλης:',  
          'θεοφίλης: έχω',  
          'έχω συμπτώματα',  
          'συμπτώματα γνωρίζω',  
          'γνωρίζω κόλλησα',  
          'κόλλησα τ...',  
          'συμφωνώ, θέλω',  
          'θέλω ξέρω',  
          'ξέρω εφόσον',  
          'εφόσον έχω',  
          'έχω συμπτώματα',  
          'συμπτώματα προφυλάξω',  
          'προφυλάξω ευάλωτες',  
          'ευάλωτες -',  
          '-']
```

Εικόνα 19: Λέξεις με bigrams

Έπειτα, δημιουργήθηκαν τα Trigrams σε δεδομένα χωρίς κρούσματα διότι είναι περισσότερα τα tweets και στην συνέχεια συνδέθηκα με τα bigrams με αποτέλεσμα να υπάρχει ένα κοινό σύνολο δεδομένων. Με λίγα λόγια, αφαιρέθηκε η δεύτερη λέξη από τα δεδομένα με trigrams ώστε να γίνουν bigrams.

```
trigramList = []
```

```
for i in range(ds.shape[0]):
```

```
    s = ds.iloc[i][1]
```

```
    output = list(ngrams(tokens, 3))
```

```
    trigramList.append(output)
```

```
actualTrigramList = []
```

```
for i in range(len(trigramList)):
```

```
    #temp = []
```

```
    for j in range(len(trigramList[i])):
```

```
        actualTrigramList.append(" ".join(trigramList[i][j]))
```

```
    #actualBigramList.append(temp)
```

```
uniqueTrigramList = []
```

```
[uniqueTrigramList.append(x) for x in actualTrigramList if x not in uniqueTrigramList]
```

```
uniqueTrigramList
```

```
Out[151]: ['νέα μελέτη nature:',
           'μελέτη nature: η',
           'nature: η ασφάλεια',
           'η ασφάλεια εμβολίων',
           'ασφάλεια εμβολίων έναντι',
           'εμβολίων έναντι sars-cov-2',
           'έναντι sars-cov-2 |',
           'sars-cov-2 | capit',
           '| capit -...']
```

Εικόνα 20: Λέξεις με trigrams

```
uniqueBigramList = []
[uniqueBigramList.append(x) for x in actualBigramList if x not in uniqueBigramList]
uniqueBigramList

gramList12 = uniqueUnigramList + uniqueBigramList

#replace the 2 in the end with 3 for trigrams
tfidf2 = TfidfVectorizer(vocabulary = uniqueBigramList, ngram_range=(1,2))

tfs2 = tfidf2.fit_transform(corpus)
feature_names = tfidf2.get_feature_names()
dense = tfs2.todense()
denselist = dense.tolist()
df_bi = pd.DataFrame(denselist, columns=feature_names)
df_bi
```

	με κορονοϊό	κορονοϊό διαγνώστηκε	διαγνώστηκε δικηγόρος	δικηγόρος χαλκίδας	χαλκίδας - χρήστος	- χρήστος θεοφίλης	θεοφίλης: έχω	έχω συμπτώματα	συμπτώματα γνωρίζω	... εμβολιάστηκαν	άστεγοι εμβολιάστηκαν	εμβολιάστηκαν covid-19	
0	0.382296	0.382296	0.382296	0.382296	0.0	0.0	0.0	0.0	0.350853	0.382296	...	0.000000	0.0
1	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.350853	0.000000	...	0.000000	0.0
2	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.000000	0.0
3	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.000000	0.0
4	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.000000	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
3357	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.000000	0.0
3358	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.56657	0.0
3359	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.000000	0.0
3360	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.000000	0.0
3361	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	...	0.000000	0.0

3362 rows x 14650 columns

Εικόνα 21: Πίνακας μοναδικών bigrams

Αφού, δημιουργήθηκε ο πίνακας που περιέχει όλες τις λέξεις σε bigrams τότε εφαρμόστηκε η μέθοδος CV. Τα δεδομένα είναι αποτελεσματικά και στην συνέχεια εκτελέστηκαν οι παρακάτω αλγόριθμοι:

Πίνακας 8: Ακρίβεια bigrams

Algorithm	Precision
Logistic Regression	0.85
Naives Bayes	0.78

4.6. Απλό Bag of Words

Στην συνέχεια, έγινε εκτέλεση ενός απλού μοντέλου Bag of Words [36] για την δημιουργία ενός απλού πίνακα λέξεων. Αυτός ο πίνακας χρησιμοποιείται για την εκπαίδευση των αλγορίθμων μηχανικής μάθησης. Διαθέτει όλα τα tweets στα δεδομένα εκπαίδευσης στα οποία φτάνουν σε αριθμό 30978 μοναδικών λέξεων. Δεδομένου με τα παραπάνω αποτελέσματα εκτελέστηκαν οι καλύτεροι αλγόριθμοι Logistic Regression και Naive Bayes και εφαρμόστηκε η μέθοδος CV που αναδεικνύεται παρακάτω στον κώδικα.

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
cv = CountVectorizer()
```

$X = cv.fit \ transform(corpus)$

X

cv.get_feature_names()

```
Out[163]: <3362x6178 sparse matrix of type '<class 'numpy.int64''>'
          with 30978 stored elements in Compressed Sparse Row format>
```

```
In [164]: cv.get_feature_names()
```

'έκανε',
 'έκανες',
 'έκδοσή',
 'έκθεση',
 'έκκληση',
 'έκκριση',
 'έκλειψε',
 'έκλεισαν',
 'έκλεισε',
 'έκπτωση',
 'έκρηξη',
 'έκρηξης',
 'έκτακτα',
 'έκτακτη',
 'έκτακτο',
 'έκτακτος',
 'έκτο',
 'έκτοτε',
 'έκτοξευση',
 'έκτοξευσε',

Εικόνα 22:Σύνολο λέξεων BOW

```
bow = pd.DataFrame(X.toarray(), columns = cv.get_feature_names())
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(bow, list(ds.label), test_size=0.2)
```

```
# Use Cross-validation.
```

```
from sklearn.model_selection import cross_val_score
```

```
# Logistic Regression
```

```
log_reg = LogisticRegression()
```

```
log_scores = cross_val_score(log_reg, X_train, y_train, cv=3)
```

```
log_reg_mean = log_scores.mean()
```

```
# Naives Bayes
```

```
nav_clf = GaussianNB()
```

```
nav_scores = cross_val_score(nav_clf, X_train, y_train, cv=3)
```

```
nav_mean = nav_scores.mean()
```

```
# Create a Dataframe with the results.
```

```
d = {'Classifiers': ['Logistic Reg.', 'Naives Bayes'],  
     'Crossval Mean Scores': [log_reg_mean, nav_mean]}
```

```
result_df = pd.DataFrame(data=d)
```

```
result_df
```

Πίνακας 9: Ακρίβεια απλό BOW

Algorithm	Precision
Logistic Regression	0.89
Naive Bayes	0.84

4.7. Τεκμηρίωση

Τέλος, εξετάζονται οι καμπύλες ROC (Receiver Operating Characteristic). Οι καμπύλες ROC τυπικά εμφανίζουν True Positive στον άξονα Y και False Positive στον άξονα X. Αυτό καθιστά την επάνω αριστερή γωνία του γραφήματος το ιδανικό σημείο επειδή το False Positive Rate είναι μηδέν (0) και True Positive Rate είναι ένα (1).

Επίσης, η απότομη κλίση των καμπυλών ROC είναι σημαντική επειδή είναι ιδανικό να μεγιστοποιηθεί το True Positive ελαχιστοποιώντας ταυτόχρονα το False Positive. Οι καμπύλες ROC χρησιμοποιούνται στη δυαδική ταξινόμηση θεωρώντας κάθε στοιχείο ως δυαδική πρόβλεψη (μέσος όρος). Οι παρακάτω μετρήσεις γίνονται με την εύρεση AUC που δηλώνει τον βαθμό απόδοσης των αλγορίθμων:

$$AUC = \frac{FP}{FP + TN}$$

Παρακάτω παρουσιάζεται ο κώδικας για τις καμπύλες ROC:

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

print('Logistic Regression Score: ', roc_auc_score(y_test, p))
print('Naive Bayes Score: ', roc_auc_score(y_test, predict_nav))
print('Random Forest Score: ', roc_auc_score(y_test, predict))
print('Decision Tree Score: ', roc_auc_score(y_test, predict_dt))
print('Grad Boosting Score: ', roc_auc_score(y_test, predict_grad))

from sklearn.metrics import roc_curve

log_fpr, log_tpr, threshold = roc_curve(y_test, p)
nav_fpr, nav_tpr, threshold = roc_curve(y_test, predict_nav)
rand_fpr, rand_tpr, threshold = roc_curve(y_test, predict)
dt_fpr, dt_tpr, threshold = roc_curve(y_test, predict_dt)
grad_fpr, grad_tpr, threshold = roc_curve(y_test, predict_grad)

#nav_fpr, nav_tpr, nav_threshold = roc_curve(y_train, naives_y_scores)
def graph_roc_curve_multiple(rand_fpr, rand_tpr, nav_fpr, nav_tpr, grad_fpr, grad_tpr,
log_fpr, log_tpr, dt_fpr, dt_tpr):
    plt.figure(figsize=(8,6))
    plt.title('ROC Curve \n Classifiers', fontsize=18)
    plt.plot(rand_fpr, rand_tpr, label='random forest')
    plt.plot(nav_fpr, nav_tpr, label='Naive Bayes')
    plt.plot(grad_fpr, grad_tpr, label='Gradient Boosting')
    plt.plot(log_fpr, log_tpr, label='Logistic Regression')
    plt.plot(dt_fpr, dt_tpr, label='decision tree')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.axis([0, 1, 0, 1])
    plt.xlabel('False Positive Rate', fontsize=16)
    plt.ylabel('True Positive Rate', fontsize=16)
    plt.annotate('Minimum ROC Score of 50% \n (This is the minimum score to get)', xy=(0.5,
```

```

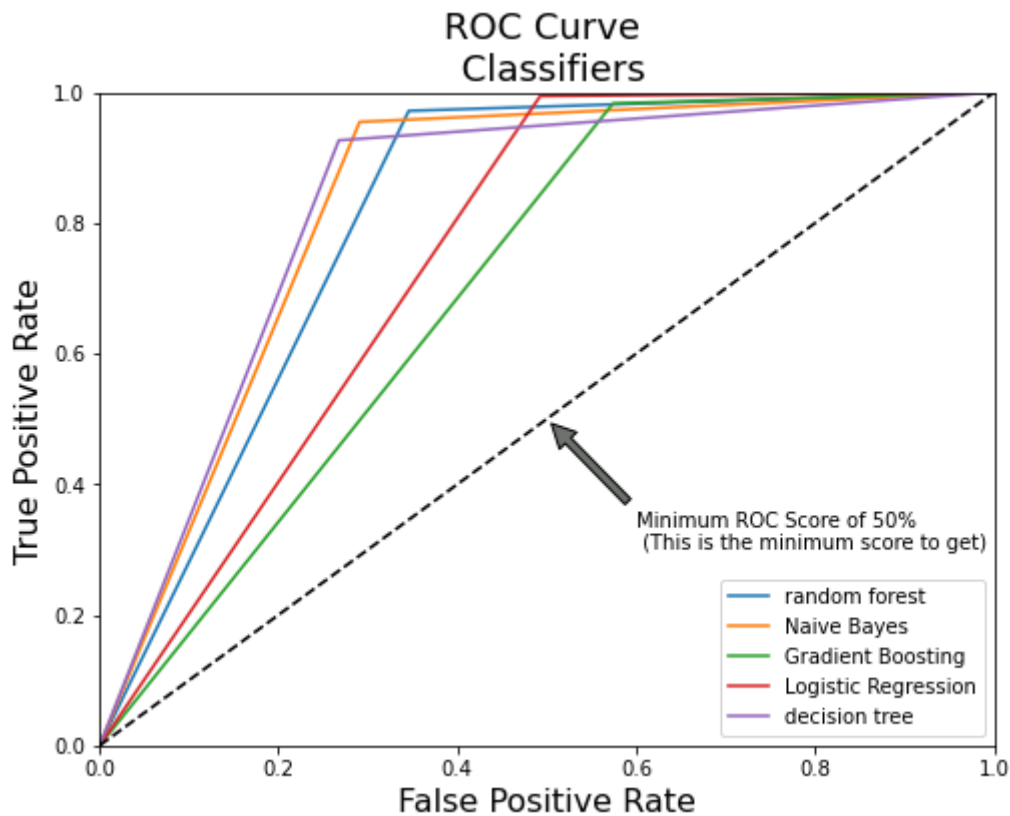
0.5), xytext=(0.6, 0.3),
    arrowprops=dict(facecolor='#6E726D', shrink=0.05),
)
plt.legend()

graph_roc_curve_multiple(rand_fpr, rand_tpr, nav_fpr, nav_tpr, grad_fpr, grad_tpr, log_fpr,
log_tpr, dt_fpr, dt_tpr)
plt.show()

```

4.7.1. Καμπύλη ROC του TF-IDF

Παρακάτω, αναδεικνύεται η καμπύλη ROC και ο πίνακας AUC για την προσέγγιση TFIDF (όλες οι λέξεις) και ο αλγόριθμος Naive Bayes έχει την μεγαλύτερη βαθμολογία.



Εικόνα 23: Καμπύλη ROC για TFIDF

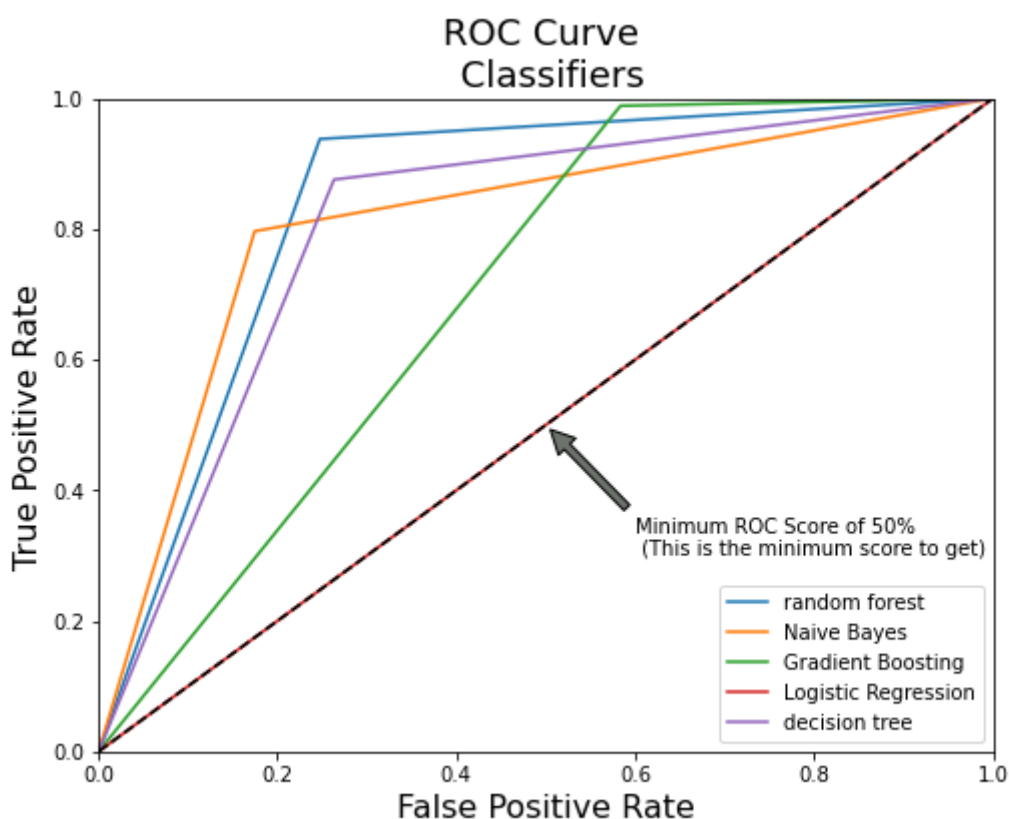
Πίνακας 10: Αποτελέσματα καμπύλης TFIDF

Algorithm	AUC
Logistic Regression Score	0.750
Naive Bayes Score	0.832

Random Forest Score	0.813
Decision Tree Score	0.829
Gradient Boosting Score	0.704

4.7.2. Καμπύλη ROC του TF-IDF με τις συχνές λέξεις

Με μια γρήγορη ματιά στην καμπύλη ROC διαπιστώνεται ότι ο καλύτερος αλγόριθμος είναι το Random Forest όπως βλέπουμε παρακάτω:



Εικόνα 24: Καμπύλη ROC TF-IDF για συχνότερες λέξεις

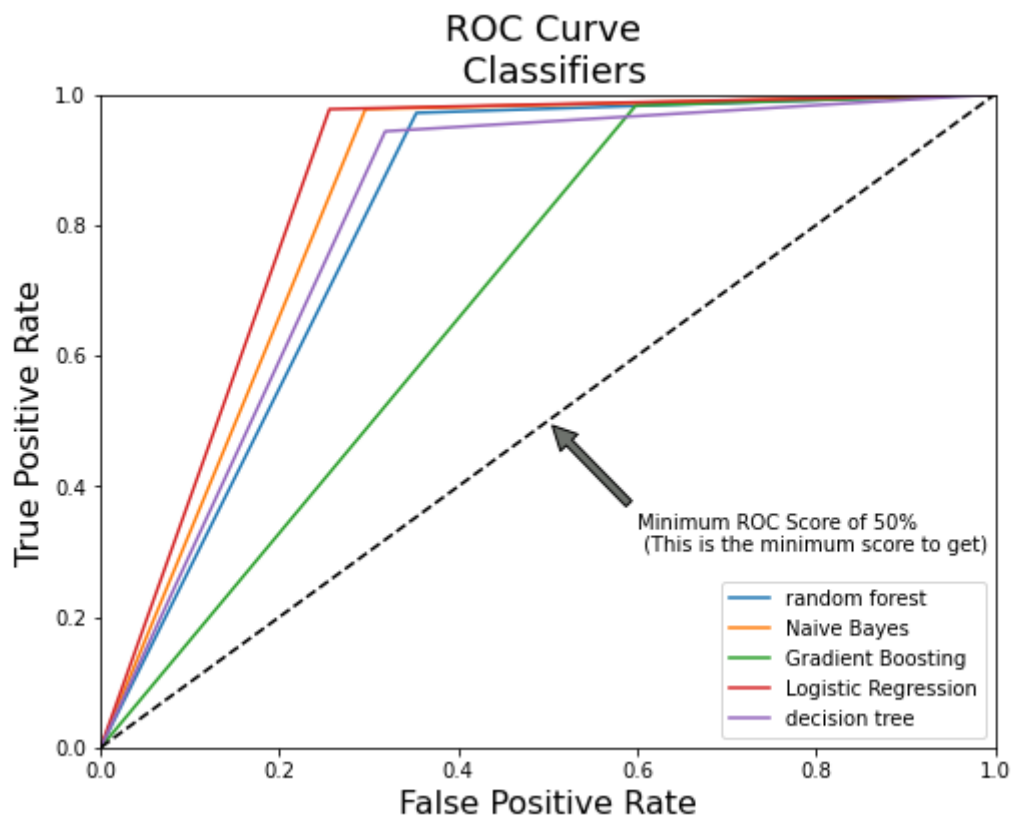
Πίνακας 11: Αποτελέσματα ROC για συχνότερες λέξεις

Algorithm	AUC
Logistic Regression Score	0.5
Naive Bayes Score	0.811
Random Forest Score	0.845

Decision Tree Score	0.806
Gradient Boosting Score	0.702

4.7.3. Καμπύλη ROC του Bag of Words

Ακολουθεί η καμπύλη για την προσέγγιση του Bag of Words (όλες οι λέξεις) και ο αλγόριθμος Logistic Regression έχει την μεγαλύτερη βαθμολογία:



Εικόνα 25: Τελική καμπύλη Bag Of Words

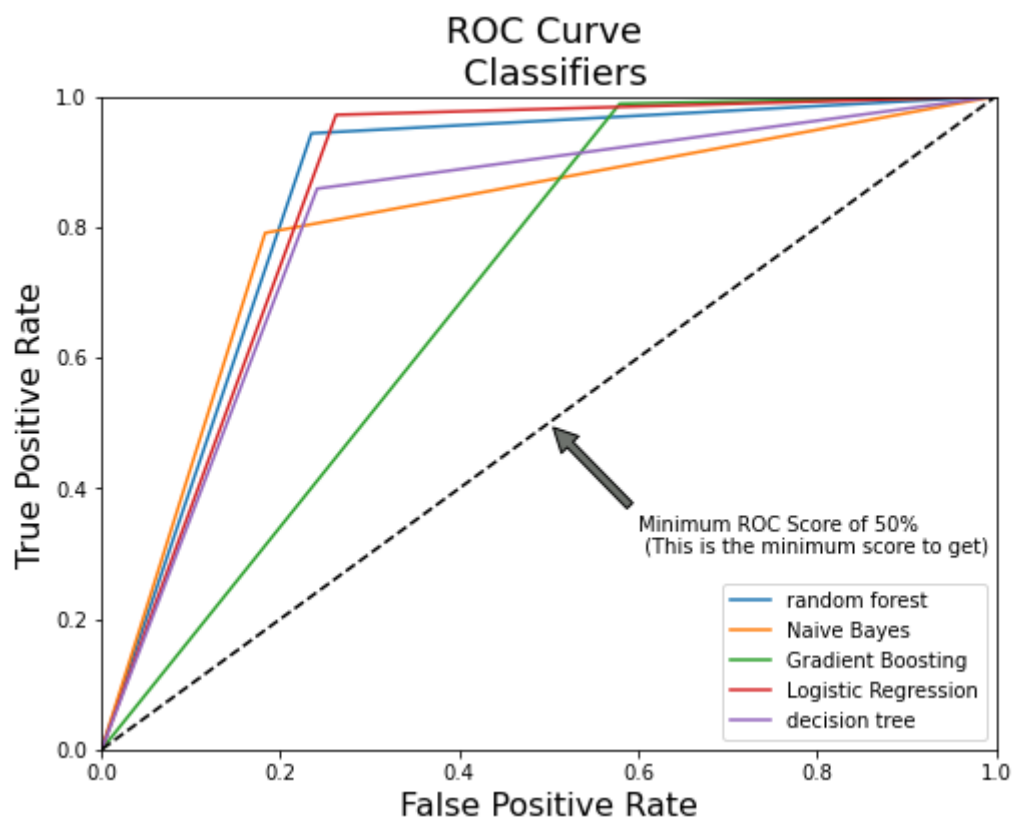
Ο παρακάτω πίνακας δείχνει την βαθμολογία AUC για τους αλγόριθμους:

Πίνακας 12: Αποτελέσματα καμπύλης για το BOW

Algorithm	AUC
Logistic Regression Score	0.860
Naive Bayes Score	0.840
Random Forest Score	0.809

Decision Tree Score	0.812
Grad Boosting Score	0.691

4.7.4. Καμπύλη ROC του Bag of Words με τις συχνές λέξεις



Εικόνα 26: Καμπύλη ROC με τις συχνότερες λέξεις BOW

Πίνακας 13: Αποτελέσματα καμπύλης ROC με τις συχνότερες λέξεις

Algorithm	AUC
Logistic Regression Score	0.854
Naive Bayes Score	0.804
Random Forest Score	0.854
Decision Tree Score	0.808
Gradient Boosting Score	0.704

5. ΣΥΜΠΕΡΑΣΜΑΤΑ

5.1. Βασικές ιδέες

Η εξάπλωση του κορονοϊού και κυρίως στις αναφορές αναφέρουν ότι τα tweets αποτελούν ένα σημαντικό κομμάτι εντοπισμού κρουσμάτων κορονοϊού μέσω μηχανικής μάθησης. Παρόλα αυτά, η έμπνευση για τη μελέτη του θέματος προήλθε από την αύξηση των περιπτώσεων των κρουσμάτων και τη δύσκολη αντιμετώπιση προς την ανίχνευση. Αυτή η άνοδος μπορεί να βοηθηθεί με τη χρήση των κοινωνικών μέσων δικτύωσης που διευκολύνουν τη ανίχνευση των κρουσμάτων.

5.2. Συμπεράσματα

Στην παρούσα πτυχιακή μελέτη έγινε αναφορά για αλγόριθμους μηχανικής μάθησης, αναλύθηκε ο τρόπος λειτουργίας τους και έγινε εφαρμογή σε δεδομένα από το twitter. Τα δεδομένα που χρησιμοποιήθηκαν σχετίζονται με την πανδημία του κορονοϊού και στόχος ήταν να δημιουργηθεί ένα όσο γίνεται πιο ακριβές μοντέλο για την καλύτερη πρόγνωση ταξινόμησης των tweets σε πιθανά ή όχι κρούσματα.

Η ανάγκη για συστήματα αυτόματης ανίχνευσης κρουσμάτων κορονοϊού γίνεται αναγκαία διότι τα κρούσματα συνεχίζουν να αποτελούν κοινωνικό πρόβλημα. Έτσι, υλοποιείται μια προσέγγιση για την ταξινόμηση των tweets σε δύο κατηγορίες: πιθανών κρουσμάτων και μη πιθανών κρουσμάτων. Χρησιμοποιώντας ένα σύνολο δεδομένων από το Twitter, εκτελούνται διαδικασίες ανάλυσης με την χρήση συγκεκριμένων μοντέλων μηχανικής μάθησης.

Συγκεκριμένα, χρησιμοποιούνται οι προσεγγίσεις του Bag of Words και του TF-IDF για τον εντοπισμό κρουσμάτων στο Twitter μέσω της μηχανικής μάθησης. Η ταξινόμηση πραγματοποιείται μέσω των αλγορίθμων Logistic Regression, Naive Bayes, Decision Tree, Random Forest και Gradient Boosting σε διάφορα σύνολα δεδομένων. Η ακρίβεια ενός αλγορίθμου δεν είναι η μόνη μέτρηση που πρέπει να εξεταστεί κατά την αξιολόγηση της απόδοσης του μοντέλου. Τα αποτελέσματα έδειξαν ότι η Logistic Regression αποδίδει καλύτερα για την προσέγγιση Bag of Words.

Αρχικά, ο αλγόριθμος Logistic Regression αναδεικνύει την καλύτερη καμπύλη AUC επειδή τοποθετείται πιο κοντά την περιοχή False Positive Rate (FPR) και True Positive Rate (TPR). Αντίθετα, ο αλγόριθμος που δεν ανταποκρίθηκε καλά σε αυτήν την εργασία είναι ο Gradient Boosting ο οποίος δείχνει μια κακή False Positive.

Αυτό οφείλεται στο γεγονός ότι ο Gradient Boosting δημιουργεί δέντρα απόφασης ένα

κάθε φορά για να διορθώσει τα σφάλματα τα οποία δημιουργήθηκαν από τις προηγούμενες εκπαιδεύσεις του. Για αυτό το λόγο, καθιστά το Gradient Boosting πιο ευαίσθητο στα δεδομένα επειδή χρειάζεται περισσότερο χρόνο για να τα εκπαιδεύσει και είναι πιο δύσκολο να συντονιστεί με αυτά. Γι' αυτό ο συγκεκριμένος αλγόριθμος έχει χαμηλή ακρίβεια καθώς και ανάκληση και στις δυο προσεγγίσεις που αναλύθηκαν και απορρίπτεται για την συγκεκριμένη εργασία.

Βέβαια, ο αλγόριθμος Logistic Regression είναι πιο απλός, γιατί πραγματοποιείται με μια απλής εκθετικής εξίσωσης. Έτσι, λοιπόν, προτείνεται η Logistic Regression ως το καλύτερο μοντέλο για την περίπτωση μας. Αφού, αναφέρθηκαν όλες οι διαδικασίες για αυτήν την εργασία, το μοντέλο επιτυγχάνει εύλογη ακρίβεια (78%) καθώς και ανάκληση (97%). Ωστόσο, κατά την εφαρμογή του μοντέλου Bag of Words όλοι οι αλγόριθμοι είχαν υψηλή ανάκληση και είναι εμφανές ότι έχει μια καλύτερη απόδοση από την προσέγγιση TFIDF.

Συνοψίζοντας, στα πειράματα που εκτελέστηκαν, παρατηρείται ότι η μέθοδος TFIDF ήταν σχεδόν συνεπή με αποτέλεσμα οι αλγόριθμοι Naive Bayes και Decision Tree να παρουσιάζουν υψηλή ακρίβεια, ο Grad Boosting μόνο μια καλή ανάκληση και ο Logistic Regression να δίνει χαμηλή ακρίβεια και ανάκληση. Όλα τα αποτελέσματα του TFIDF δεν ξεπερνάνε την προσέγγιση Bag of Words. Έτσι, συνολικά τις καλύτερες τιμές ακρίβειας επιτυγχάνει ο αλγόριθμος Logistic Regression, και ακολουθούν οι Naive Bayes και Decision Tree οι οποίοι παρουσιάζουν μικρές διαφορές στην επίδοση.

Η διαφορά αυτών των μεθόδων είναι ότι το Bag of Words δημιουργεί απλώς ένα σύνολο λέξεων που περιέχει τον αριθμό των εμφανίσεων στα tweets, ενώ το μοντέλο TF-IDF περιέχει πληροφορίες για τις πιο σημαντικές λέξεις και δεν δίνει βαρύτητα στις ασήμαντες. Το μοντέλο Bag of Words είναι εύκολο να ερμηνευτεί και αποδίδει καλύτερα επειδή τα δεδομένα που διαθέτει αυτή η πτυχιακή μελέτη είναι επαρκή.

Τέλος, έγινε εκτέλεση ενός απλού μοντέλου Bag of Words [36] και μίας μέθοδος bigrams για την δημιουργία δεδομένων εκπαίδευσης. Σύμφωνα με τα αποτελέσματα που εκτελέστηκαν οι καλύτεροι αλγόριθμοι οι οποίοι είναι οι Logistic Regression και Naives Bayes. Τα αποτελέσματα ήταν ικανοποιητικά όπως είχε τεκμηριωθεί με τον αλγόριθμο Logistic Regression να πετυχαίνει την καλύτερη ακρίβεια και να επιβεβαιώνει ότι αποδίδει καλύτερα σε αυτήν την εργασία.

5.3. Μελλοντικά βήματα

Αυτή η εργασία δεν βασίζεται σε πολύγλωσσα δεδομένα, τα οποία θα μπορούσαν να θεωρηθούν ως μια πιθανή μελλοντική κατεύθυνση. Μια απαραίτητη εργασία θα μπορούσε να γίνει με τη συλλογή όλων των διαθέσιμων δεδομένων από όλες τις γλώσσες που σχετίζεται με αυτήν την πανδημία, ώστε να βοηθηθεί η κοινωνία πιο αποτελεσματικά.

Ο κύριος στόχος αυτού του έργου είναι η ανάπτυξη ενός συστήματος που μπορεί να ανιχνεύσει τα κρούσματα από tweets. Ακόμη, υπάρχει δυνατότητα να δημιουργηθεί μία ιστοσελίδα χρησιμοποιώντας HTML και CSS όπου ένας χρήστης μπορεί να εισαγάγει ένα tweet που παρέχεται από την ιστοσελίδα. Για παράδειγμα, ο χρήστης θα μπορούσε να εισάγει ένα πιθανό ή μη πιθανό κρούσμα tweet και στη συνέχεια να κατατάσσεται σωστά.

BIBΛΙΟΓΡΑΦΙΑ:

- [1] J. Xue *et al.*, “Twitter discussions and emotions about the COVID-19 pandemic: Machine learning approach,” *J. Med. Internet Res.*, vol. 22, no. 11, pp. 1–14, 2020, doi: 10.2196/20550.
- [2] C. Wang, Z. Z. Eds, and R. Goebel, *Advanced Data Mining and Applications*. 2020.
- [3] E. Chen, K. Lerman, and E. Ferrara, “Tracking social media discourse about the COVID-19 pandemic: Development of a public coronavirus Twitter data set,” *JMIR Public Heal. Surveill.*, vol. 6, no. 2, 2020, doi: 10.2196/19273.
- [4] T. Mackey *et al.*, “Machine learning to detect self-reporting of symptoms, testing access, and recovery associated with COVID-19 on Twitter: retrospective big data infoveillance study,” *JMIR Public Heal. Surveill.*, vol. 6, no. 2, pp. 1–9, 2020, doi: 10.2196/19509.
- [5] R. E. Cuomo, V. Purushothaman, J. Li, M. Cai, and T. K. Mackey, “A longitudinal and geospatial analysis of COVID-19 tweets during the early outbreak period in the United States,” *BMC Public Health*, vol. 21, no. 1, pp. 1–11, 2021, doi: 10.1186/s12889-021-10827-4.
- [6] H. Zheng, D. H. -L. Goh, C. S. Lee, E. W. J. Lee, and Y. L. Theng, “Uncovering temporal differences in COVID -19 tweets ,” *Proc. Assoc. Inf. Sci. Technol.*, vol. 57, no. 1, pp. 1–5, 2020, doi: 10.1002/pa2.233.
- [7] E. De Santis, A. Martino, and A. Rizzi, “An Infoveillance System for Detecting and Tracking Relevant Topics from Italian Tweets during the COVID-19 Event,” *IEEE Access*, vol. 8, pp. 132527–132538, 2020, doi: 10.1109/ACCESS.2020.3010033.
- [8] C. Ordun, S. Purushotham, and E. Raff, “Exploratory Analysis of Covid-19 Tweets using Topic Modeling, UMAP, and DiGraphs,” no. March, 2020, [Online]. Available: <http://arxiv.org/abs/2005.03082>.
- [9] R. E. Cuomo, V. Purushothaman, J. Li, M. Cai, and T. K. Mackey, “Sub-national longitudinal and geospatial analysis of COVID-19 tweets,” *PLoS One*, vol. 15, no. 10 October, pp. 1–11, 2020, doi: 10.1371/journal.pone.0241330.
- [10] F. Rustam, M. Khalid, W. Aslam, V. Rupapara, A. Mehmood, and G. S. Choi, “A performance comparison of supervised machine learning models for Covid-19 tweets sentiment analysis,” *PLoS One*, vol. 16, no. 2, pp. 1–23, 2021, doi: 10.1371/journal.pone.0245909.
- [11] A. Al-Rawi and V. Shukla, “Bots as active news promoters: A digital analysis of covid-19 tweets,” *Inf.*, vol. 11, no. 10, pp. 1–13, 2020, doi: 10.3390/info11100461.
- [12] J. E. Chire Saire and J. F. Oblitas Cruz, “Study of Coronavirus Impact on Parisian Population

from April to June using Twitter and Text Mining Approach,” *Proc. - 2020 Int. Comput. Symp. ICS 2020*, pp. 242–246, 2020, doi: 10.1109/ICS51289.2020.00056.

- [13] U. Qazi, M. Imran, and F. Ofli, “GeoCoV19: A Dataset of Hundreds of Millions of Multilingual COVID-19 Tweets with Location Information,” pp. 1–3, 2020, [Online]. Available: <https://crisisnlp.qcri.org/covid19>.
- [14] J. M. Box-Steffensmeier and L. Moses, “Meaningful messaging: Sentiment in elite social media communication with the public on the COVID-19 pandemic,” *Sci. Adv.*, vol. 7, no. 29, pp. 1–6, 2021, doi: 10.1126/sciadv.abg2898.
- [15] M. Y. Kabir and S. Madria, “CoronaVis: A Real-time COVID-19 Tweets Data Analyzer and Data Repository,” 2020, [Online]. Available: <http://arxiv.org/abs/2004.13932>.
- [16] V. Antoniou, E. Vassilakis, and M. Hatzaki, “Is crowdsourcing a reliable method for mass data acquisition? The case of COVID-19 spread in greece during spring 2020,” *ISPRS Int. J. Geo-Information*, vol. 9, no. 10, 2020, doi: 10.3390/ijgi9100605.
- [17] I. Engineering, *Performance testing of open-source HTTP web frameworks in an API*. 2017.
- [18] O. Gencoglu, “Large-Scale, Language-Agnostic Discourse Classification of Tweets During COVID-19,” *Mach. Learn. Knowl. Extr.*, vol. 2, no. 4, pp. 603–616, 2020, doi: 10.3390/make2040032.
- [19] S. Sengupta, S. Mugde, and G. Sharma, “An exploration of impact of COVID 19 on mental health - Analysis of tweets using natural language processing techniques,” *medRxiv*, p. 436286, 2020.
- [20] K. Chakraborty, S. Bhatia, S. Bhattacharyya, J. Platos, R. Bag, and A. E. Hassanien, “Sentiment Analysis of COVID-19 tweets by Deep Learning Classifiers—A study to show how popularity is affecting accuracy in social media,” *Appl. Soft Comput. J.*, vol. 97, p. 106754, 2020, doi: 10.1016/j.asoc.2020.106754.
- [21] R. Stevens *et al.*, “Association between HIV-related tweets and HIV incidence in the United States: Infodemiology study,” *J. Med. Internet Res.*, vol. 22, no. 6, pp. 1–10, 2020, doi: 10.2196/17196.
- [22] P. Baby and K. B., *Sentimental Analysis and Deep Learning : A Survey*. 2020.
- [23] A. Giovanni Møller, R. van der Goot, and B. Plank, “NLP North at WNUT-2020 Task 2: Pre-training versus Ensembling for Detection of Informative COVID-19 English Tweets,” pp. 331–336, 2020, doi: 10.18653/v1/2020.wnut-1.44.
- [24] M. A. Al-garadi, Y. Yang, S. Lakamana, and A. Sarker, “A Text Classification Approach for the Automatic Detection of Twitter Posts Containing Self-reported COVID-19 Symptoms,” *Open*

Rev., vol. 003, pp. 1–5, 2020.

- [25] P. Kumar and A. Singh, “NutCracker at WNUT-2020 Task 2: Robustly Identifying Informative COVID-19 Tweets using Ensembling and Adversarial Training,” pp. 404–408, 2020, doi: 10.18653/v1/2020.wnut-1.57.
- [26] C. Morin, A. Mercier, and L. Atlani-Duault, “Text–image relationships in tweets: Shaping the meanings of an epidemic,” *Societies*, vol. 9, no. 1, 2019, doi: 10.3390/soc9010012.
- [27] A. Joshi *et al.*, “Automated monitoring of tweets for early detection of the 2014 Ebola epidemic,” *PLoS One*, vol. 15, no. 3, pp. 1–10, 2020, doi: 10.1371/journal.pone.0230322.
- [28] A. Hassan Zadeh, H. M. Zolbanin, R. Sharda, and D. Delen, “Social Media for Nowcasting Flu Activity: Spatio-Temporal Big Data Analysis,” *Inf. Syst. Front.*, vol. 21, no. 4, pp. 743–760, 2019, doi: 10.1007/s10796-018-9893-0.
- [29] “Anaconda | The World’s Most Popular Data Science Platform.”
<https://www.anaconda.com/> (accessed Jan. 18, 2022).
- [30] “stopwordsiso · PyPI.” <https://pypi.org/project/stopwordsiso/> (accessed Jan. 18, 2022).
- [31] “Tweepy.” <https://www.tweepy.org/> (accessed Jan. 18, 2022).
- [32] “Basic Tweet Preprocessing in Python | by Parthvi Shah | Towards Data Science.”
<https://towardsdatascience.com/basic-tweet-preprocessing-in-python-efd8360d529e>
(accessed Jan. 18, 2022).
- [33] “Stemming and Lemmatization in Python - DataCamp.”
<https://www.datacamp.com/community/tutorials/stemming-lemmatization-python>
(accessed Jan. 18, 2022).
- [34] “pandas.DataFrame.isna — pandas 1.3.5 documentation.”
<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.isna.html> (accessed Jan. 18, 2022).
- [35] “SMOTE for Imbalanced Classification with Python.”
<https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>
(accessed Jan. 18, 2022).
- [36] “An Introduction to Bag of Words in NLP using Python | What is BoW?”
<https://www.mygreatlearning.com/blog/bag-of-words/> (accessed Jan. 18, 2022).
- [37] “TF-IDF from scratch in python on a real-world dataset. | by William Scott | Towards Data Science.” <https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089> (accessed Jan. 18, 2022).
- [38] “Learn Naive Bayes Algorithm | Naive Bayes Classifier Examples.”

<https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/> (accessed Jan. 24, 2022).

- [39] “Decision Trees in Machine Learning | by Prashant Gupta | Towards Data Science.” <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052> (accessed Jan. 24, 2022).
- [40] “Random Forest in Python. A Practical End-to-End Machine Learning... | by Will Koehrsen | Towards Data Science.” <https://towardsdatascience.com/random-forest-in-python-24d0893d51c0> (accessed Jan. 24, 2022).
- [41] “Understanding Gradient Boosting Machines | by Harshdeep Singh | Towards Data Science.” <https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab> (accessed Jan. 24, 2022).
- [42] “Logistic Regression — Detailed Overview | by Saishruthi Swaminathan | Towards Data Science.” <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc> (accessed Jan. 24, 2022).
- [43] “collections — Container datatypes — Python 3.10.2 documentation.” <https://docs.python.org/3/library/collections.html> (accessed Jan. 24, 2022).