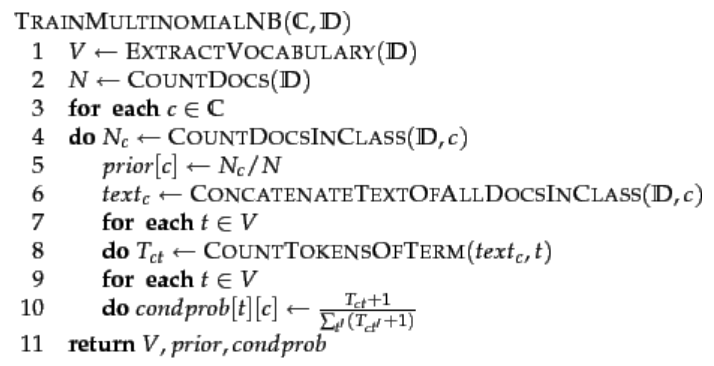
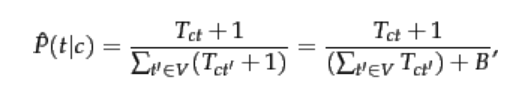
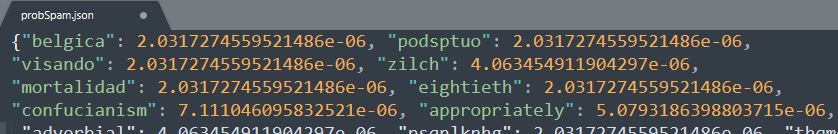
Εργασία 2η *Τεχνητή Νοημοσύνη*

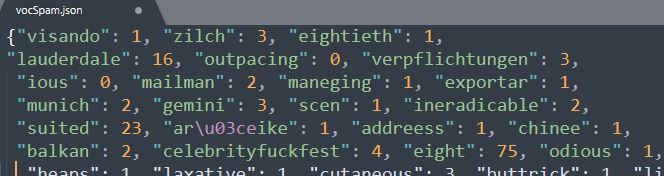
* *Multinomial Naive Bayes*

Η πρώτη μας υλοποίηση είναι ο αλγόριθμος μάθησης :Αφελής ταξινομητής Bayes με πολυωνυμική μορφή (Multinomial Naive Bayes). Το συγκεκριμένο μοντέλο βασίζεται σε μάθηση πιθανοτήτων ,κατά το οποίο βρίσκουμε τη συχνότητα των λέξεων που έχει ένα mail και καταγράφουμε τις πιθανότητες που έχει η συγκεκριμένη λέξη να βρεθεί σε ένα spam ή σε ένα ham mail. Στη συγκεκριμένη υλοποίηση δεν λαμβάνουμε υπόψιν το μέγεθος του κειμένου ή την θέση της λέξης στο κείμενο ,παρά μόνο πόσες φορές έχει εμφανιστεί η λέξη στα δεδομένα εκπαίδευσης. Ο αλγόριθμος στον οποίο βασιστήκαμε και έχουμε υλοποιήσει στο αρχείο κώδικα train.py είναι ο ακόλουθος:

Όπου C είναι οι δύο κλάσεις στις οποίες μπορεί να ανήκει το έγγραφο(είτε ham είτε spam),D είναι τα αρχεία εκπαίδευσης όπου διαθέτουμε, prior[c] είναι ένας πίνακας στον οποίο αποθηκεύουμε τις πιθανότητες P(c=Ham)=(αριθμός αρχείων εκπαίδευσης που είναι ham)/(συνολικός αριθμός αρχείων) και αντιστοίχως για P(c=Spam). Ο αλγόριθμος επιστρέφει το condprob στο οποίο περιέχονται οι πιθανότητες κάθε λέξης που έχει εμφανιστεί στην κλάση Ham και στην κλάση Spam όπου έχει χρησιμοποιηθεί εκτιμήτρια Laplace.

Όπου Tct αριθμός εμφάνισης της λέξης και Β συνολικός αριθμός εμφανίσεων όλων των λέξεων.

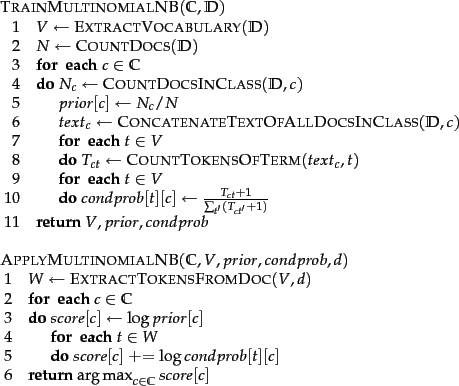
Έτσι, στο πρόγραμμα μας (train.py) έχουμε θέσει πως στην πρώτη θέση κάθε πίνακα που θα έχουμε φτιάξει αντιστοιχεί σε στοιχεία για την κλάση Ham και στη δεύτερη θέση για την κλάση Spam. Ξεκινώντας από την μέθοδο calltrain παίρνει ως είσοδο το path στο οποίο βρίσκονται όλα τα αρχεία εκπαίδευσης ,τα οποία θα τα διαχωρίζουμε μέσω του ονόματος του κάθε αρχείου(αν περιέχει την λέξη spam ή ham). Υπολογίζουμε τον αριθμό των mails και τις πιθανότητες και αποθηκεύουμε σε ένα csv αρχείο την αναλόγια των ham και των spam ,όπου θα μας βοηθήσει για να δημιουργήσουμε τους παρακάτω πίνακες. Στη συνέχεια, υπολογίζουμε τη συχνότητα των λέξεων σε κάθε κλάση με την βοήθεια της μεθόδου counter η οποία προέρχεται από το πακέτο collections της python. Με την μέθοδο remove\_digits δεν αφαιρούμε μόνο τα νούμερα τα οποία θεωρήσαμε ότι δεν βοηθούν στην πρόβλεψη , αλλά και συγκεκριμένες λέξεις-σύμβολα όπως on,in,at,Subject:,-,,,.κ.α. οι οποίες συμπεράναμε πως δεν βοηθούν. Έπειτα , συνενώνουμε τις λέξεις που προέρχονται και από τα spam και τα ham και αποθηκεύουμε το κάθε «λεξικό» όπου έχουμε δημιουργήσει σε ένα αρχείο τύπου json. Τέλος υπολογίζουμε τις πιθανότητες της κάθε λέξης και με παρόμοιο τρόπο αποθηκεύουμε σε αρχεία τύπου json, ώστε να μπορέσουν να διαβαστούν από το test.py. Με αυτόν τον τρόπο προκύπτουν τα παρακάτω ενδεικτικά αρχεία:

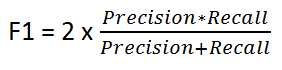
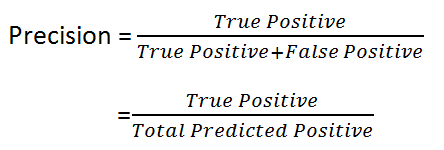


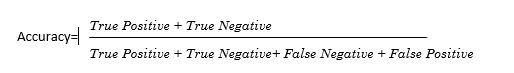
Προχωρώντας στην υλοποίηση του test , η γενική μορφή του μοντέλου αναφέρει πως πολλαπλασιάζουμε τις πιθανότητες της κάθε λέξης που έχει βρεθεί στο καινούριο κείμενο όπου πρέπει να προβλέψουμε αν ανήκει σε ham ή spam.

Η εξίσωση την οποία χρησιμοποιούμε στον ταξινομητής naive Bayes για την επιλογή καλύτερης κλάσης είναι η :

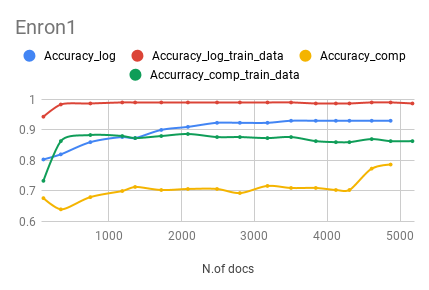
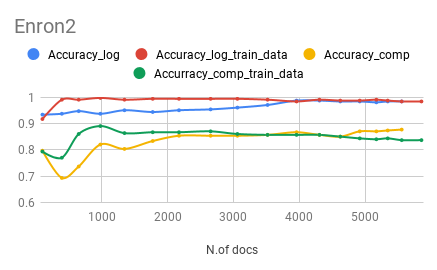
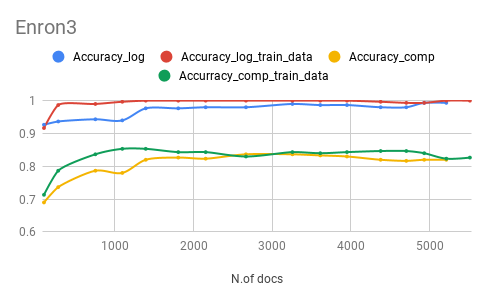
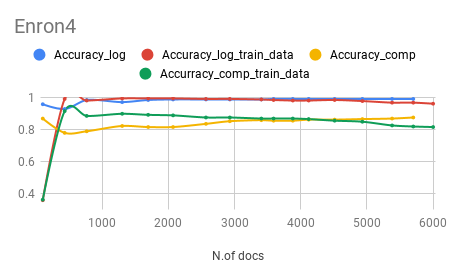
Όμως, λόγω του ότι οι πιθανότητες πολλαπλασιάζονται ,η καθεμία σε κάθε θέση 1<k<nd, αυτό ίσως προκαλέσει υπερχείλιση κινητής υποδιαστολής. Γι’ αυτό έχουμε υλοποιήσει τον αλγόριθμο προσθέτοντας του λογαρίθμους των πιθανοτήτων τους(πολλαπλασιάζοντας στον καθένα πόσες φορές έχει βρεθεί μέσα στο κείμενο η λέξη). Επίσης έχουμε υλοποιήσει και τους δύο τρόπους ώστε να ελέγξουμε ποιος τρόπος είναι αποδοτικότερος.

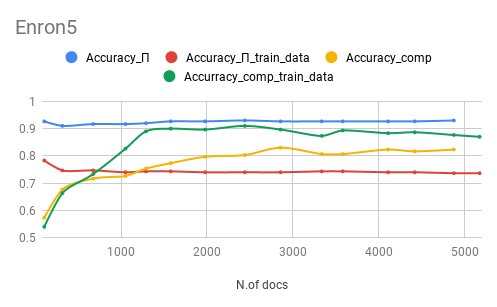
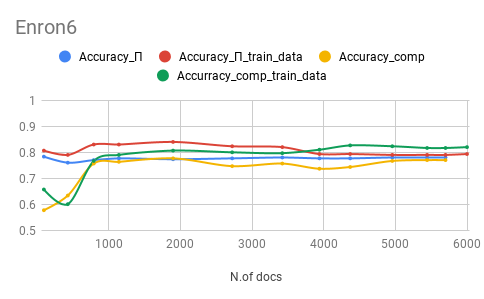
Στο αρχείο κώδικα test.py,έχουμε βασιστεί στον παρακάτω κώδικα:

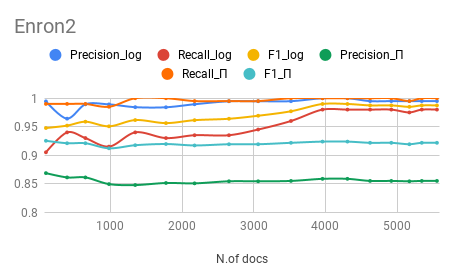
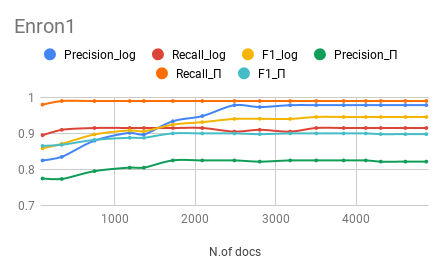
Έτσι, στη μέθοδο calltest λαμβάνουμε τα paths στα οποία έχουμε τα αρχεία για test και διαβάζουμε όλα τα αρχεία στα οποία έχουμε τα «λεξικά» και τις πιθανότητες. Έπειτα για κάθε αρχείο όπου έχουμε για test υπολογίζουμε τις πιθανότητες σε κάθε κλάση Ham και Spam ώστε να τις συγκρίνουμε και να δούμε την πρόβλεψη μας. Αυτό έχει υλοποιηθεί και τον πρώτο τρόπο που αναφέραμε και για τον δεύτερο τρόπο(με λογάριθμο). Τέλος , έχοντας υπολογίσει τα σωστά κατατασσόμενα αρχεία και τα λανθασμένα υπολογίζουμε τα accuracy,precision,recall και f1 τα οποία προέρχονται από τους παρακάτω τύπους:

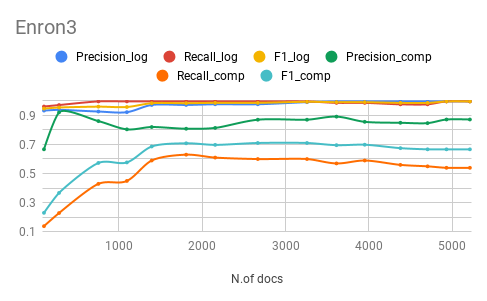
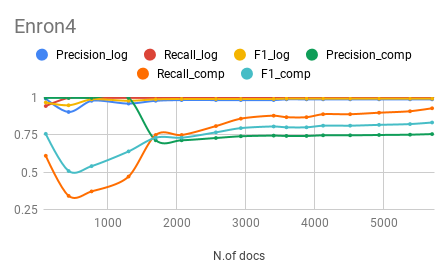


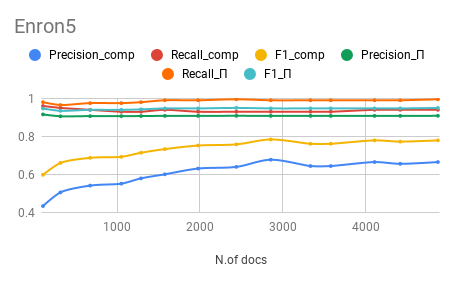
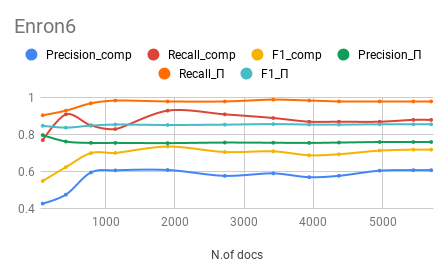
H μέθοδος write\_csv μας βοηθάει στην κατασκευή των καμπυλών που θα ακολουθήσουν.

Στη πράξη, έχουμε ένα αρχείο κώδικα main στο οποίο δημιουργούμε τα αρχεία για τις καμπύλες και συγκρίνουμε την υλοποίηση μας με μία έτοιμη υλοποίηση την οποία βρήκαμε στο διαδίκτυο(<https://github.com/rohitthapliyal2000/Spam-Classification/blob/master/src/naivebayes_classifier.py>) και κάναμε κάποιες μικρές αλλαγές(αρχείο testmain.py) ώστε να διαβάζει τα ίδια αρχεία όπως με τον δικό μας κώδικα και να αποθηκεύει τα δεδομένα του για να συγκριθούν με τα δικά μας δεδομένα. Επίσης εξετάζονται ταυτόχρονα σε δεδομένα εκπαίδευσης(300 mails,200 ham και 100 spam) και ελέγχου(300 mails,200 ham και 100 spam). Έχουμε χρησιμοποιήσει το σύνολο δεδομένων *enron-Spam* στη προεπεξεργασμένη μορφή τους. Παρακάτω παραθέτουμε τις καμπύλες accuracy σε δεδομένα εκπαίδευσης και ελέγχου συναρτήσει του πλήθους των παραδειγμάτων εκπαίδευσης της υλοποίησης με τον *λογάριθμο* και του συγκρινόμενου κώδικα από το διαδίκτυο σε κάθε υποσύνολο δεδομένων του enron-spam.

Παρακάτω παραθέτουμε την σύγκριση του accuracy με τον πρώτο τρόπο(Πολλαπλασιασμός) σε σχέση με τον συγκρινόμενο κώδικα(comp)

Στη συνέχεια ,παρουσιάζουμε τις καμπύλες ακρίβειας ,ανάκλησης και F1(σε δεδομένα ελέγχου) συναρτήσει του πλήθους των παραδειγμάτων εκπαίδευσης συγκρίνοντας την υλοποίηση με λογάριθμο(log) και με πολλαπλασιασμό(Π).

Επίσης ,παρουσιάζουμε τις καμπύλες ακρίβειας ,ανάκλησης και F1(σε δεδομένα ελέγχου) συναρτήσει του πλήθους των παραδειγμάτων εκπαίδευσης συγκρίνοντας την υλοποίηση με λογάριθμο(log) και με του συγκρινόμενου κώδικα(comp).

Tέλος, παραθέτουμε τις καμπύλες τις καμπύλες ακρίβειας ,ανάκλησης και F1(σε δεδομένα ελέγχου) συναρτήσει του πλήθους των παραδειγμάτων εκπαίδευσης συγκρίνοντας την υλοποίηση με πολλαπλασιασμό(Π) και με του συγκρινόμενου κώδικα(comp).

Οι πίνακες για τις παραπάνω καμπύλες βρίσκονται στον φάκελο με όνομα matrix\_charts. Επίσης , όλοι οι πίνακες οι οποίοι δημιουργήθηκαν κατά την διάρκεια την εκτέλεσης του test για τον κώδικα μας βρίσκονται στον φάκελο με όνομα mydata στο οποίο βρίσκεται ένα αρχείο num.csv στο οποίο αναφέρονται οι αναλογίες μεταξύ spam και ham που χρησιμοποιήθηκαν στα δεδομένα εκπαίδευσης. Επίσης, οι πίνακες για τον συγκρινόμενο κώδικα βρίσκονται στον φάκελο data(όπου υπάρχει ο επιπρόσθετος χαρακτήρας ‘’Τ’’ αναφερόμαστε σε δεδομένα εκπαίδευσης που χρησιμοποιήθηκαν και στο test).

Ιστοσελίδες που μας βοήθησαν στην υλοποίηση:

* <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>
* <https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html#laplace>
* <https://web.stanford.edu/~jurafsky/slp3/slides/7_NB.pdf>