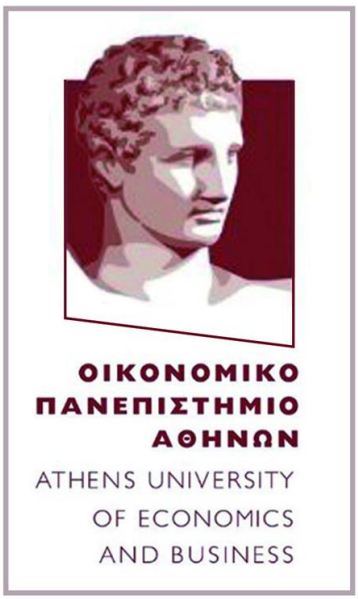


I. Ανδρουτσόπουλος

2022-23

Αφελής ταξινομητής Bayes & ID3

Αναστάσιος Σπυρίδων Μπουρτσουκλής
3190138

Αντώνιος Προμπονάς
3190179

Περιεχόμενα

Τρόπος χρήσης

ID3

1. Ανοίγουμε το αρχείο “Main.ipynb”
2. Τρέχουμε τα κατάλληλα cells, ανάλογα με τον επιθυμητό έλεγχο.

Naive Bayes

1. Ανοίγουμε το Command Prompt
2. Μέσω της εντολής "cd", ορίζουμε το path που βρίσκεται ο φάκελος της εργασίας
   2.1. Πρέπει να είναι της μορφής "cd _:\...\3190138_3190179"

Μέρος Α

3. Τρέχουμε την εντολή: python Bayes.py

Μέρος Β

3. Τρέχουμε την εντολή: python Comparison_of_Bayes.py

Αρχιτεκτονική

ID3

Data

Η κλάση Data είναι υπεύθυνη για την κατασκευή του κατάλληλου λεξιλογίου, σύμφωνα με τις υπερπαραμέτρους και τα train δεδομένα, και την μετατροπή είτε train είτε test δεδομένων σε vectors.

Το function **load** δέχεται ως είσοδο το path προς τα negative και positive train data αντίστοιχα και το πλήθος των αρχείων που θα εξετάσει και κατασκευάζει το αρχικό λεξιλόγιο.

Το function **filter** δέχεται ως είσοδο τις υπερπαραμέτρους m, n, k και το path όπου θέλουμε να αποθηκευτεί το τελικό λεξιλόγιο. Από το αρχικό λεξιλόγιο αφαιρεί τις n πιο συχνές λέξεις και τις k πιο σπάνιες λέξεις, στην συνέχεια, κρατάει τις m πιο συχνές λέξεις και κατασκευάζει αρχείο vocabulary.txt στο δοσμένο path με το τελικό λεξιλόγιο.

Το function **convertTrain** δέχεται ως είσοδο το path από τα δεδομένα, το path όπου θέλουμε να αποθηκευτεί το αποτέλεσμα και . Μετατρέπει όλα τα δεδομένα σε vectors με 1 ή 0, αν περιέχουν ή όχι τις λέξεις του λεξιλογίου και κατασκευάζει αρχείο positive.txt ή negative.txt, ανάλογα με το επιθυμητό αποτέλεσμα, στο δοσμένο path.

Το function **convertTest** δέχεται ως είσοδο το path από τα δεδομένα, το path όπου θέλουμε να αποθηκευτεί το αποτέλεσμα και το πλήθος των αρχείων που θα μετατρέψει. Μετατρέπει όλα τα δεδομένα σε vectors με 1 ή 0, αν περιέχουν ή όχι τις λέξεις του λεξιλογίου και κατασκευάζει αρχείο test.txt στο δοσμένο path.

ID3Node

Η κλάση ID3Node υλοποιεί τα αντικείμενα που αποτελούν τους κόμβους του ID3 δέντρου απόφασης και περιέχει του αντίστοιχους getters και setters.

ID3

Η κλάση ID3 είναι υπεύθυνη για την εκπαίδευση του αλγορίθμου ID3, την κατασκευή του κατάλληλου δέντρου απόφασης και την χρήση του.

Το function **train** δέχεται ως είσοδο τα paths με τα αρνητικά και τα θετικά train δεδομένα και το λεξιλόγιο και τα μετατρέπει σε arrays.

Το function **build** κατασκευάζει το δέντρο απόφασης, σύμφωνα με τα δεδομένα που έγινε trained.

Το function **classify** δέχεται ως είσοδο το path του test αρχείου από vectors και το path όπου θέλουμε να αποθηκευτούν τα αποτελέσματα. Μετατρέπει το test αρχείο σε array, κατηγοριοποιεί κάθε στοιχείο βάση του ID3 δέντρου απόφασης και αποθηκεύει τα αποτελέσματα στο δοσμένο path.

Validator

Η κλάση validator είναι υπεύθυνη για τον υπολογισμό του accuracy, του precision, του recall και του F1 των αποτελεσμάτων του ID3 δέντρου απόφασης.

Το function **check** δέχεται ως είσοδο το path των αποτελεσμάτων του ID3 δέντρου απόφασης και το αναμενόμενο-επιθυμητό αποτέλεσμα και υπολογίζει το πλήθος των true positive, true negative, false positive και false negative.

Το function **validate** υπολογίζει και εκτυπώνει το accuracy, το precision, το recall και το F1 των αποτελεσμάτων που έχουν γίνει ήδη check.

PerformanceChecker

Η κλάση PerformanceChecker είναι υπεύθυνη για την επαναληπτική αξιολόγηση των αποτελεσμάτων του ID3 δέντρου απόφασης, ανάλογα το πλήθος των train δεδομένων, και για την κατασκευή πινάκων και διαγραμμάτων με τα αποτελέσματα.

Το function **checkTrain** δέχεται ως είσοδο το πλήθος των train δεδομένων που θέλουμε να έχει ο 1ος έλεγχος, το πλήθος των train δεδομένων που θέλουμε να έχει ο τελευταίος έλεγχος και το «βήμα» που αυξάνεται το πλήθος των train δεδομένων. Εκτελεί των αλγόριθμο ID3 όσες φορές χρειάζεται για τα train δεδομένα, αξιολογώντας κάθε φορά τα αποτελέσματα και κατασκευάζει κατασκευή πίνακα και διαγράμματα με τα αποτελέσματα.

Το function **checkTest** δέχεται ως είσοδο το πλήθος των train δεδομένων που θέλουμε να έχει ο 1ος έλεγχος, το πλήθος των train δεδομένων που θέλουμε να έχει ο τελευταίος έλεγχος και το «βήμα» που αυξάνεται το πλήθος των train δεδομένων. Εκτελεί των αλγόριθμο ID3 όσες φορές χρειάζεται για τα test δεδομένα, αξιολογώντας κάθε φορά τα αποτελέσματα και κατασκευάζει κατασκευή πίνακα και διαγράμματα με τα αποτελέσματα.

Naive Bayes

Η υλοποίηση του αλγορίθμου του Naive Bayes ξεκινάει με την κλήση της εντολής:

naive_bayes(m,n,k)

Στην εντολή αυτή έχουμε περάσει ως υπερπαραμέτρους τις τιμές m, n και k. Συγκεκριμένα, όπως μας υποδεικνύει η άσκηση, μέσω των τιμών αυτών δείχνουμε στο πρόγραμμα το πλήθος των λέξεων που θα περιέχει το λεξιλόγιο (m=10000), αφού πρώτα έχουμε αφαιρέσει τις n=100 πιο συχνές λέξεις και k=100 πιο σπάνιες λέξεις. Φυσικά για να συμβεί αυτό έγιναν κάποιες συγκεκριμένες ενέργειες.

Ξεκινώντας την υλοποίηση του αλγορίθμου, η πρώτη τάξη που καλείται είναι η train_data(m,n,k), η οποία δημιουργεί το λεξιλόγιο και στη συνέχεια τα διανύσματα που αντιπροσωπεύουν τα δεδομένα εκπαίδευσης και ύστερα καλείται η naive_bayes_test(m,n,k,train_vectors), η οποία δημιουργεί διανύσματα για τα test data.

train_data(m,n,k)

Όπως αναφέραμε ακριβώς από πάνω, οι υπερπαράμετροι της τάξης προσδιορίζουν το μέγεθος του λεξιλογίου ,καθώς και το πλήθος των πιο συχνών και σπάνιων λέξεων που πρέπει να παραλειφθούν. Αφού περάσουμε κατάλληλα το path, με το σημείο που βρίσκονται τα δεδομένα του φακέλου που θέλουμε να διαβάσουμε, καλούμε την εντολή create_vocabulary(), η οποία δημιουργεί και επιστρέφει το λεξιλόγιο που θα χρησιμοποιηθεί στη συνέχεια για την δημιουργία των διανυσμάτων. Έπειτα καλείται η κλάση create_vectors(), η οποία δημιουργεί και επιστρέφει τα διανύσματα τα οποία αντιστοιχούν στις λέξεις των κειμένων. Στη συνέχεια υπολογίζονται τα P(Xi|C=0) και P(Xi|C=1) με την κλήση της τάξης get_probabilities(), όπου αν C=0 υπολογίζουμε τις πιθανότητες για τα δεδομένα από τον φάκελο με τα negative reviews, αλλιώς αν C=1 , υπολογίζουμε τις πιθανότητες για τα δεδομένα από τον φάκελο με τα positive_reviews. Η κλάση αυτή καλείται 2 φορές. Την πρώτη φορά επιστρέφει μία λίστα που περιέχει τα αποτελέσματα του P(Xi|C=0) και την $2^{η}$ φορά μία λίστα που περιέχει τα αποτελέσματα του P(Xi|C=1). Τέλος μετράμε το πλήθος των θετικών κειμένων και των αρνητικών κειμένων καθώς και το συνολικό πλήθος των κειμένων. Για να συμβεί αυτό ελέγχουμε το πρώτο στοιχείο κάθε διανύσματος , αφού όταν δημιουργήσαμε τα vectors, ορίσαμε ότι το πρώτο στοιχείο κάθε διανύσματος θα είναι 0, αν πρόκειται για negative review ή 1 αν πρόκειται για positive review. Τέλος, η κλάση επιστρέφει τα P(Xi|C=0), P(Xi|C=1), pos_prob (τη πιθανότητα το κείμενο να είναι θετικό), pos_neg (τη πιθανότητα το κείμενο να είναι αρνητικό) καθώς και το λεξιλόγιο(vocabulary) που θα χρειαστεί στη συνέχεια.

create_vocabulary(pos_path,neg_path,m,n,k)

Στη συγκεκριμένη κλάση δημιουργούμε το λεξιλόγιο με βάση το οποίο θα κωδικοποιήσουμε στη συνέχεια τις λέξεις των texts από τα positive και negative reviews. Συγκεκριμένα, αρχικά θα δημιουργήσουμε ένα dictionary το οποίο αποσκοπεί στο να μετρά το πλήθος των λέξεων που υπάρχουν τόσο στα positive reviews όσο και στα negative reviews. Στη συνέχεια, δημιουργούμε ένα list το οποίο περιέχει τα μοναδικά διαφορετικά πλήθη των λέξεων μέσα στα texts.

Με βάση λοιπόν, το παραπάνω dictionary και το παραπάνω list αφαιρούμε τις k=100 σπάνιες λέξεις και τις n=100 πιο συχνές λέξεις. Η διαδικασία είναι οι εξής:

Οι πρώτες k=100 λέξεις του dictionary που ισούνται με την μικρότερη τιμή της λίστας και οι πρώτες n=100 λέξεις του dictionary που ισούνται με τις μεγαλύτερες τιμές της λίστας αποτελούν τις πιο σπάνιες και συχνές λέξεις αντίστοιχα. Με την ίδια μεθοδολογία, οι πρώτες m=10000 λέξεις που δεν ανήκουν στην λίστα των πιο σπάνιων ή πιο συχνών αποτελούν το λεξιλόγιο μας.

create_vectors(pos_path,neg_path,vocabulary)

Η κλάση αυτή παίρνει ως υπερπαραμέτρους τα path των φακέλων των positive και negative reviews καθώς και το λεξιλόγιο που έχουμε δημιουργήσει. Αρχικοποιούμε, 2 vectors. Τον “temp” και τον “vectors”. Ο vectors θα περιέχει συγκεντρωτικά τα διανύσματα όλων των texts.

Αρχικά διαβάζει τα positive reviews και προσθέτει ως πρώτο στοιχείο όλων των vectors την τιμή 1. Στη συνέχεια εξετάζει κάθε λέξη του text, και αν η λέξη αυτή υπάρχει στο λεξιλόγιο προσθέτει στο vector τη τιμή 1. Σε διαφορετική περίπτωση προσθέτει την τιμή 0. Η διαδικασία αυτή γίνεται σε έναν προσωρινό vector (που τον έχουμε ονομάσει temp), τον οποίο κάθε φορά που διαβάζουμε όλες τις λέξεις του text, τον προσθέτουμε σε έναν άλλον vector και έπειτα τον μηδενίζουμε. Αυτή η διαδικασία γίνεται για κάθε text των positive reviews.

Η ίδια διαδικασία γίνεται και για τα negative_reviews, με τη μόνη διαφορά ότι προσθέτει ως πρώτο στοιχείο των vectors την τιμή 0.

Στο τέλος η κλάση επιστρέφει το vector, με όλα τα κείμενα κωδικοποιημένα σε δυαδική μορφή.

get_probabilities(vectors)

Στη συγκεκριμένη κλάση υπολογίζουμε τα P(Xi|C=0) και P(Xi|C=1). Για να το κάνουμε αυτό, αρχικά αρχικοποιούμε με 0, 2 dictionaries μεγέθους τόσο όσο το μέγεθος του πιο μεγάλου text. Τα dictionaries αυτά είναι τα P1C0 και P1C1.

Στη συνέχεια, για κάθε text του εκάστοτε review, παίρνουμε την κωδικοποιημένη του μορφή και ελέγχουμε για κάθε στοιχείο του vector, αν η τιμή είναι 1. Αν είναι, τότε αυξάνουμε τον αντίστοιχο «μετρητή» κατά 1 και με αυτόν τον τρόπο έχουμε υπολογίσει πόσες λέξεις του λεξιλογίου υπάρχουν στα texts στη θέση 1, 2,…,n των κειμένων.

Στη συνέχεια, εφαρμόζουμε τη μέθοδο la place. Συγκεκριμένα, εφαρμόζουμε τον τύπο $\frac{P(Xi|C)+1}{C+2}$ για κάθε μία κατηγορία των reviews και το αποτέλεσμα κάθε πράξης για κάθε feature το αποθηκεύουμε σε μία λίστα. Είτε την pos_prob αν πρόκειται για positive reviews είτε την neg_prob αν πρόκειται για negative reviews. Στο τέλος, η κλάση επιστρέφει αυτές τις 2 λίστες.

naive_bayes_test(m,n,k,train_vectors)

Η συγκεκριμένη κλάση δέχεται ως υπερπαραμέτρους τις τιμές m,n,k που έχουμε δώσει και στις παραπάνω κλάσεις καθώς επίσης και τα διανύσματα τα οποία έχουμε δημιουργήσει χρησιμοποιώντας τα δεδομένα εκπαίδευσης. Αυτό που κάνει η συγκεκριμένη συνάρτηση είναι να δημιουργεί διανύσματα για διαφορετικά ωστόσο positive και negative reviews. Τα διανύσματα αυτά διαμορφώνονται με βάση το ίδιο λεξιλόγιο που χρησιμοποιήσαμε για να διαμορφώσουμε τα train vectors. Για αυτό το λόγο καλούμε μόνο τη κλάση create_vectors().

Αφού διαμορφώσουμε διανύσματα για τα test data , μπαίνουμε στο τελευταίο στάδιο υλοποίησης του αλγορίθμου. Συγκρίνουμε τα train_vectors με τα test_vectors ως εξής: Πρώτα υπολογίζουμε τη πιθανότητα κάθε στοιχείου ($1^{ου}$, $2^{ου}$,$3^{ου}$, κλπ) του κειμένου να είναι 1 ή 0 πολλαπλασιάζοντας τα P1C1 και P1C0 με την μεταβλητή priori_pos αν τα test_vectors προέρχονται από positive reviews και τα (1-P1C1) και (1-P1C0) με την μεταβλητή priori_neg αν τα test_vectors προέρχονται από negative reviews . Οι μεταβλητές priori_pos και priori_neg αρχικοποιούνται με 1 κάθε φορά πριν αρχίζουμε να υπολογίζουμε τις πιθανότητες των features κάθε text. Αν το αποτέλεσμα της σύγκρισης priori_pos,priori_neg ταυτίζεται με το test_data και priori_pos>priori_neg, τότε έχουμε ένα true positive στοιχείο, αλλιώς έχουμε ένα true negative στοιχείο.

Αν το αποτέλεσμα της σύγκρισης priori_pos,priori_neg δεν ταυτίζεται με το test_data και priori_pos>priori_neg, τότε έχουμε ένα false_positive στοιχείο, αλλιώς έχουμε ένα false_negative στοιχείο.

Για να υλοποιήσουμε το συγκεκριμένο ερώτημα εφαρμόσαμε την έτοιμη εκδοχή του αλγορίθμου του Bayes μέσω του sklearn και συγκρίναμε τα αποτελέσματα της δικιάς μας υλοποίησης με αυτή του αλγορίθμου. Για τη δημιουργία των train & test vectors ακολουθήσαμε ακριβώς την ίδια διαδικασία με αυτή του α μέρους. Από εκεί πέρα όμως πραγματοποιήσαμε μερικές ενέργειες παραπάνω:

Αρχικά με την εντολή x_train,x_test,y_train,y_test=train_test_split(train,y,test_size=0.33,random_state=17), οπού train τα train data και y τα test data , χωρίσαμε τα δεδομένα σε 67% δεδομένα εκπαίδευσης και 33% δεδομένα ελέγχου.

Έπειτα χρησιμοποιήσαμε την εντολή nb=BernoulliNB(), για να δημιουργήσουμε το αντικείμενο που καλέσει τις έτοιμες συναρτήσεις της πολυμεταβλητής Bernouli. Συγκεκριμένα, κάνουμε fit τα δεδομένα x_train, y_train, μέσω της συνάρτησης nb.fit(x_train,y_train), προβλέπουμε τη y_pred μέσω της εντολής nb.predict(x_test) και στη συνέχεια βρίσκουμε το accuracy,precision,recall & f1 των δεδομένων μας, μέσω των έτοιμων συναρτήσεων της sklearn:

accuracy=accuracy_score(y_expect, y_pred)

precision=precision_score(y_expect,y_pred)

recall=recall_score(y_expect,y_pred, average='macro')

f1= f1_score(y_expect, y_pred, average='macro')

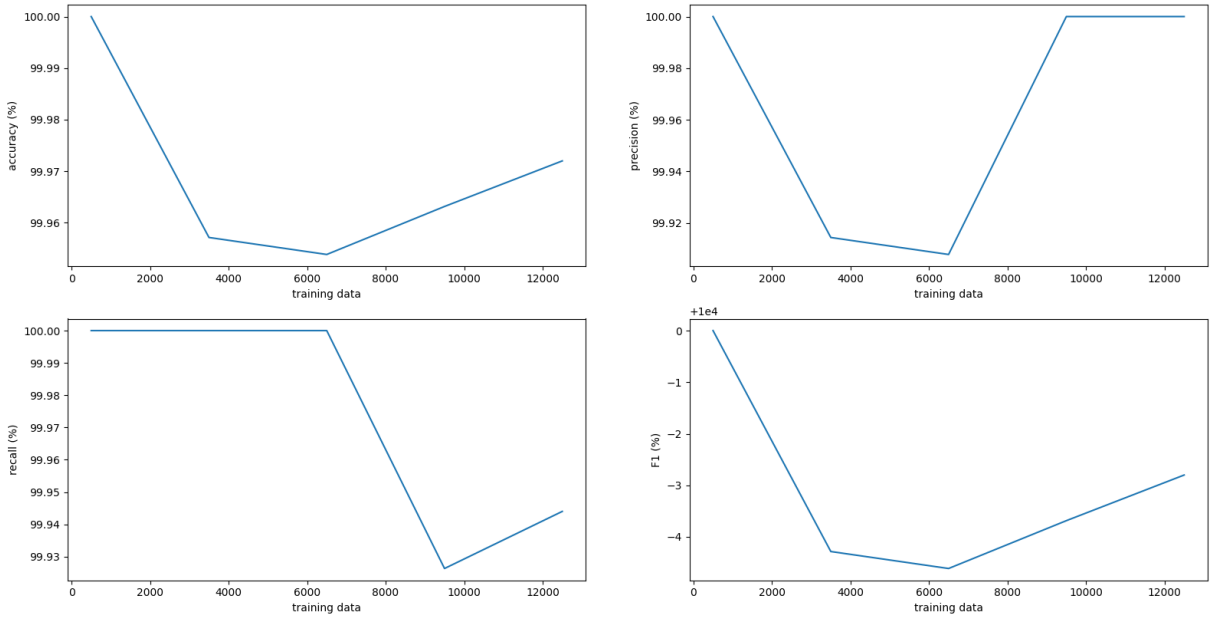
Η παραπάνω διαδικασία γίνεται για κάθε κείμενο των test data.

Παραδείγματα Χρήσης

ID3

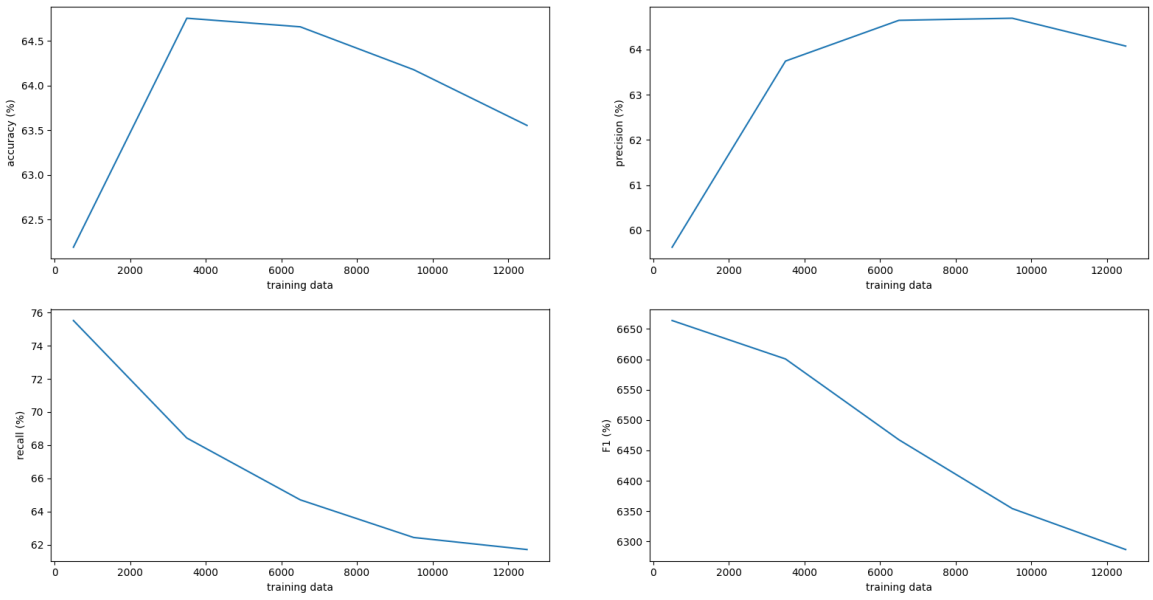
Στα **δεδομένα εκπαίδευσης** με m=500, n=300, k=300:

Data	Accuracy	Precision	Recall	F1
500	100	100	100	10000
3500	99.9571	99.9144	100	9995.72
6500	99.9538	99.9078	100	9995.39
9500	99.9632	100	99.9263	9996.31
12500	99.972	100	99.944	9997.2



Στα **δεδομένα ελέγχου** με m=500, n=300, k=300:

Data	Accuracy	Precision	Recall	F1
500	62.192	59.6261	75.52	6663.84
3500	64.756	63.7454	68.432	6600.56
6500	64.66	64.6471	64.704	6467.55
9500	64.18	64.6937	62.432	6354.27
12500	63.556	64.0774	61.704	6286.83

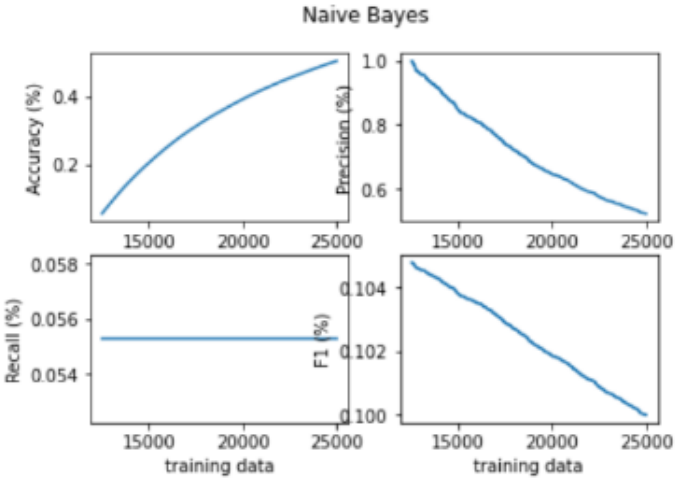


! Οι τιμές των υπερπαραμέτρων επιλέχθηκαν έπειτα από δοκιμές στα δεδομένα του «IMDB dataset», με διαφορετικούς συνδυασμούς και βάσει τόσο του accuracy, όσο και του κόστους της αύξησης του μεγέθους του λεξιλογίου.

Naive Bayes

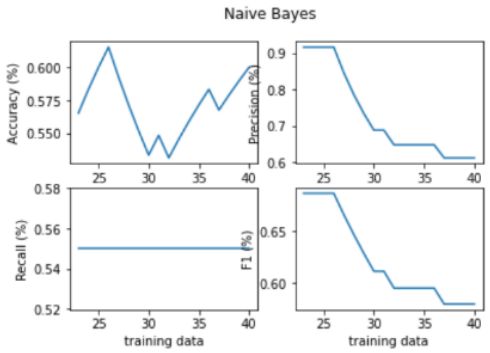
naive_bayes(10000,100,100):

```
Accuracy
0.5024602952354282

Precision:
0.5234848484848484

Recall:
0.055284422753820305

F1:
0.1000072364136334
```



Παρακάτω βρίσκεται η σύγκριση των υλοποιήσεων για την εκτέλεση της εντολής: naive_bayes(10000,40,40).

```
Accuracy
0.6
Precision:
0.6111111111111112
Recall:
0.55
F1:
0.5789473684210527
```




```
Results from BernoulliNB algorithm:
Accuracy
0.7857142857142857
Precision:
0.8
Recall:
0.7444444444444445
F1:
0.7543859649122806
              precision    recall  f1-score   support

           0       0.75      0.60      0.67         5
           1       0.80      0.89      0.84         9

    accuracy                           0.79        14
   macro avg       0.78      0.74      0.75        14
weighted avg       0.78      0.79      0.78        14
```