ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

2η Εργασία στην Τεχνητή νοημοσύνη

Παναγιώτης Κάτσος 3180077, Πέτρος Τσότσι 3180193,
Επαμεινώνδας Ιωάννου 3140059

Ιανουάριος 2020

Διαδικασία ετοιμασίας δεδομένων (data preperation) :

Το αρχείο data_prep το χρησιμοποιούμε για να φτιάξουμε 3 αρχεία csv, το train, το validate και το test. Η μορφή του κάθε csv θα ναι ως εξής:

- Οι στήλες θα αποτελούν το λεξιλόγιο, δηλαδή κάθε στήλη θα ναι η αντίστοιχη λέξη του λεξιλογίου.
- Κάθε γραμμή θα αποτελεί ένα review

Στην ουσία κάθε review θα παριστάνεται ως ένα διάνυσμα ιδιοτήτων με τιμές 0 ή 1, οι οποίες θα δείχνουν ποιες λέξεις ενός λεξιλογίου περιέχει το κείμενο, όπως αναφέρεται και στην εκφώνηση της εργασίας. Στο αρχείο αυτό για να δημιουργηθούν τα απαιτούμενα csv χρησιμοποιούνται 2 συναρτήσεις.

1. **create_csv(df,set,tol,label).** Το όρισμα df είναι ένα pandas dataframe το οποίο θα χρησιμοποιηθεί για να αποθηκευτεί το συγκεκριμένο csv. Σε πρώτη φάση θα περιέχει μόνο τις λέξεις του λεξιλογίου, που έχουν παρθεί από το αρχείο imdb_vocab καθώς και τη λέξη label, με την οποία θα διακρίνουμε αν ένα review είναι θετικό ή αρνητικό (pos,neg). Το set αποτελεί το path του folder για το αντίστοιχο csv που θέλουμε να φτιάξουμε. Για παράδειγμα αν θέλουμε να φτιάξουμε το csv για τα train παραδείγματα περνάμε το συγκεκριμένο path. Αυτό το κάνουμε γιατί και στους 2 φακέλους (train,test) υπάρχουν τα αρχεία labeledBow.feat τα οποία αναφέρουν για κάθε review του αντίστοιχου φακέλου σε τι συχνότητα περιέχει τις λέξεις στο imdb_vocab. Το όρισμα tol χρησιμοποιείται για τον αριθμό των reviews που θέλουμε να περιέχει το csv. Αν βάλουμε 1000 για παράδειγμα, το csv θα περιέχει 2000, καθώς καλούμε τη δεύτερη συνάρτηση του προγράμματος 2 φορές, μία για τα αρνητικά reviews και μία για τα θετικά. Πριν καλέσουμε την άλλη συνάρτηση αποθηκεύουμε στη μεταβλητή lines_pos τις συχνότητες των λέξεων που περιέχει κάθε θετικό review που έχουμε επιλέξει, ενώ στη μεταβλητή lines_neg τις συχνότητες που περιέχει κάθε αρνητικό review. Αυτό γίνεται εφικτό προφανώς με χρήση, όπως προαναφέρθηκε, των αρχείων labeledBow. Έπειτα καλείται η άλλη συνάρτηση, η οποία αναλύεται παρακάτω.

2. **add_line_to_df(df,lines,label,tol).** Η συνάρτηση αυτή δέχεται ως όρισμα το df που αναλύθηκε παραπάνω, τα lines που αντιστοιχούν σε lines με τις συχνότητες λέξεων για τον αριθμό reviews που έχουμε επιλέξει, το label που θα ναι είτε pos ή neg, καθώς και το tol , ώστε να γνωρίζουμε το συνολικό αριθμό reviews που θα περιέχει το csv. Στην ουσία σε αυτή τη συνάρτηση ελέγχουμε για κάθε review ποιες λέξεις από το λεξιλόγιο περιέχει που έχουμε στο df και όσες απ' αυτές τις περιέχει, στην αντίστοιχη στήλη με τη συγκεκριμένη λέξη βάζουμε 1, αλλιώς 0. Αυτή η διαδικασία επαναλαμβάνεται για κάθε review (θετικό ή αρνητικό) με ένα απλό for loop, ελέγχοντας στην ουσία κάθε γραμμή από τη μεταβλητή lines. Η επανάληψη σταματάει, όταν ξεπεράσουμε τον αριθμό tol. Κάθε review το κάνουμε append στο dataframe, οπότε στο τέλος θα παραχθεί ένας πίνακας με γραμμές τόσες όσα τα reviews που έχουμε επιλέξει (μισά θετικά, μισά αρνητικά) και κάθε review θα παριστάνεται ως το επιθυμητό διάνυσμα από 0 και

1. Η συνάρτηση επιστρέφει το δημιουργημένο πίνακα.

Στο τέλος της συνάρτησης **create_csv** δημιουργούμε το επιθυμητό csv με την εντολή **df.tocsv().**

Αλγόριθμος Naive Bayes:

Ο 1ος αλγόριθμος που υλοποιήσαμε στα πλαίσια αυτής της εργασίας είναι ο Αφελής ταξινομητής Bayes πολυωνυμικής μορφής. Σε αυτόν τον αλγόριθμο μάθησης προσεγγίζουμε το πρόβλημα με την χρήση πιθανοτήτων. Συγκεκριμένα, με βάση παραδείγματα εκπαίδευσης που έχουμε στη διάθεσή μας μαζί με τις ορθές τους αποκρίσεις, προσπαθούμε να προσδιορίσουμε την πιθανότητα ένα νέο παράδειγμα να ανήκει στην θετική ή την αρνητική κατηγορία. Υπολογίζουμε δηλαδή για κάθε νέο παράδειγμα (έστω ότι παριστάνεται από ένα διάνυσμα ιδιοτήτων X) τις 2 πιθανότητες : P(C =1|X) και P(C =0|X) και ανάλογα με το ποια είναι μεγαλύτερη κατατάσσουμε το παράδειγμα αυτό στην αντίστοιχη κατηγορία. Ο υπολογισμός των πιθανοτήτων αυτών βασίζεται στο θεώρημα του Bayes και σε παραδοχή ανεξαρτησίας μεταξύ των ιδιοτήτων και έτσι προκύπτουν οι τύποι :

$$P(C = 1|X) = P(C = 1) \cdot \Pi_{i=1}^{n} P(X_i = x_i|C = 1)/P(X)$$

$$P(C = 0|X) = P(C = 0) \cdot \Pi_{i=1}^{n} P(X_i = x_i|C = 0)/P(X)$$

Τους οποίους και καλούμαστε να υπολογίσουμε και να συγκρίνουμε για να κάνουμε την κατηγοριοποίηση του παραδείγματος X.

Εκπαίδευση αλγορίθμου (αρχείο naive_bayes.py):

Κατά την εκπαίδευση του naive bayes χρησιμοποιήσαμε 10.000 παραδείγματα εκπαίδευσης (train data) και τις 550-50 συχνότερες λέξεις του λεξιλογίου(στη συνέχεια θα γίνει έλεγχος τους σε ξεχωριστά δεδομένα ανάπτυξης) . Αυτό που καλούμαστε να κάνουμε είναι να υπολογίσουμε βάσει των παραδειγμάτων εκπαίδευσης όλες τις απαιτούμενες πιθανότητες($P(C = 1), P(C = 0), P(X_i = x_i|C = 1), P(X_i = x_i|C = 0)$, όπου Xi η i-οστή ιδιότητα) προκειμένου στη συνέχεια να υπολογίσουμε τις αρχικές πιθανότητες που χρειαζόμαστε για κάνουμε την κατάταξη.

Για το σκοπό αυτό υλοποιούμε την βοηθητική συνάρτηση probabilitytable_indexs η οποία και υπολογίζει τις δεσμευμένες πιθανότητες) δημιουργώντας για κάθε τιμή του C (εδώ 2 κατηγορίες) έναν δισδιάστατο πίνακα για να τις αποθηκεύσει. Ύστερα κατά την χρήση απλώς μπορούμε να ανακτούμε όποια από αυτές μας χρειάζεται. Επιπλέον χρησιμοποιήσαμε εκτιμήτρια Laplace για λόγους εξομάλυνσης των δεσμευμένων πιθανοτήτων που υπολογίζουμε, προσθέτοντας +2 στον παρονομαστή και +1 στον αριθμητή.

Η συνάρτηση εκπαίδευσης του αλγορίθμου μας είναι η fit η οποία παίρνει ως ορίσματα τα παραδείγματα εκπαίδευσης στα οποία θέλουμε να εκπαιδευτεί, και η οποία καλώντας την probabilitytable_indexs θα υπολογίσει τους πίνακες με όλες τις απαιτούμενες πιθανότητες.

*Επιπλέον για προγραμματιστικούς υλοποιήθηκαν οι συναρτήσεις vocabularize, getItemsIterator, getColumns.

Επιλογή υπερπαραμέτρου του αλγορίθμου (αρχείο validation.py):

Στην συνέχεια , προχωρήσαμε στην επιλογή της υπερπαραμέτρου του πλήθους των ιδιοτήτων που θα επιλέξουμε να αναπαραστήσουμε τα παραδείγματα μας. Συγκεκριμένα επιλέξαμε 2000 νέα δεδομένα επικύρωσης , διαφορετικά από τα παραδείγματα εκπαίδευσης, στα οποία θα αξιολογήσουμε τον αλγόριθμο μας, και συγκεκριμένα τι ποσοστά ορθότητας 'βγάζει' αν χρησιμοποιήσουμε τις 100,200,300,400,500 συχνότερες λέξεις του λεξιλογίου μας. Παρακάτω ακολουθούν τα αποτελέσματα του validation που εκτελέσαμε:

```
Accuracy in Naive Bayes Algorithm for a training set of 10k reviews (5k pos, 5k neg), a dictionary of 100 words and a validation set of 2k is: 0.6875
Accuracy in Naive Bayes Algorithm for a training set of 10k reviews (5k pos, 5k neg), a dictionary of 200 words and a validation set of 2k is: 0.735
Accuracy in Naive Bayes Algorithm for a training set of 10k reviews (5k pos, 5k neg), a dictionary of 300 words and a validation set of 2k is: 0.767
Accuracy in Naive Bayes Algorithm for a training set of 10k reviews (5k pos, 5k neg), a dictionary of 400 words and a validation set of 2k is: 0.8005
Accuracy in Naive Bayes Algorithm for a training set of 10k reviews (5k pos, 5k neg), a dictionary of 500 words and a validation set of 2k is: 0.8075
```

Χρήση και αξιολόγηση του αλγορίθμου (αρχείο naive_bayes.py) :

Μετά την εκπαίδευση του αλγορίθμου μας στα train data που του 'δώσαμε', και την επιλογή της υπερπαραμέτρου στα δεδομένα επικύρωσης ακολουθεί η χρήση και αξιολόγηση του αλγορίθμου σε νέα δεδομένα αξιολόγησης (test data) που αποτελούνται από 2000 παραδείγματα κριτικών.

Για τον σκοπό αυτό υλοποιήσαμε την συνάρτηση predict η οποία με δοσμένο το σύνολο των test data κατατάσσει κάθε παράδειγμα των test data σε μία κατηγορία υπολογίζοντας για κάθε ένα τις πιθανότητες P(C =1|X) και P(C =0|X), όπου X το κάθε παράδειγμα. Συγκεκριμένα , με βάση τον τύπο που αναλύσαμε παραπάνω και ανακτώντας τις απαιτούμενες δεσμευμένες πιθανότητες από τους πίνακες που προέκυψαν κατά την εκπαίδευση υπολογίζουμε τις 2 άνω πιθανότητες* . Στο τέλος εξετάζοντας ποια είναι μεγαλύτερη, θα κάνουμε και την ανάλογη ταξινόμηση και με αυτόν τον τρόπο θα προκύψουν οι προβλέψεις του αλγορίθμου για τα test data.

(*Στον υπολογισμό του γινομένου $\Pi_{i=1}^{n} P(X_i = x_i|C)$ του τύπου επιλέξαμε να υπολογίσουμε αντ' αυτού το άθροισμα των λογαρίθμων των τιμών των πιθανοτήτων προκειμένου να αποφύγουμε σφάλματα λόγω underflow.)

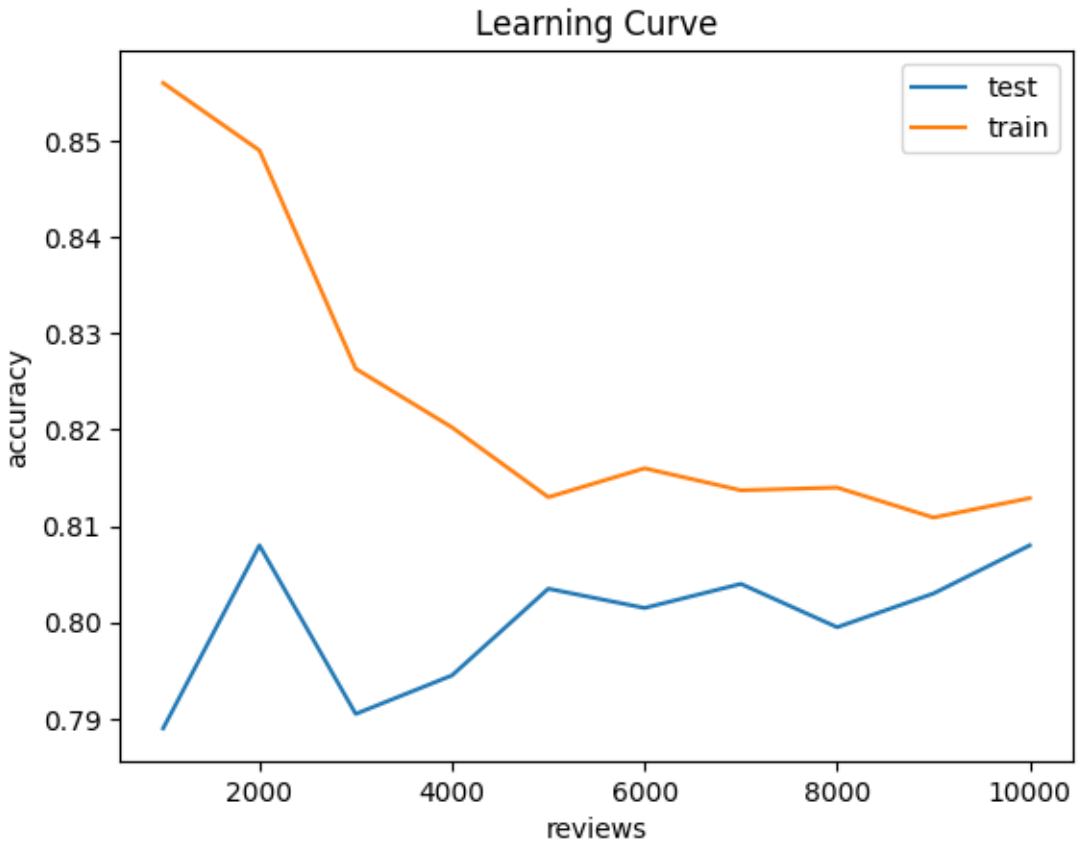
Ακολούθως υλοποιήσαμε μια σειρά από συναρτήσεις με σκοπό να αξιολογή-

σουμε τον αλγόριθμο ως προς διάφορες μετρικές και να κατασκευάσουμε τα απαιτούμενα διαγράμματα και καμπύλες.

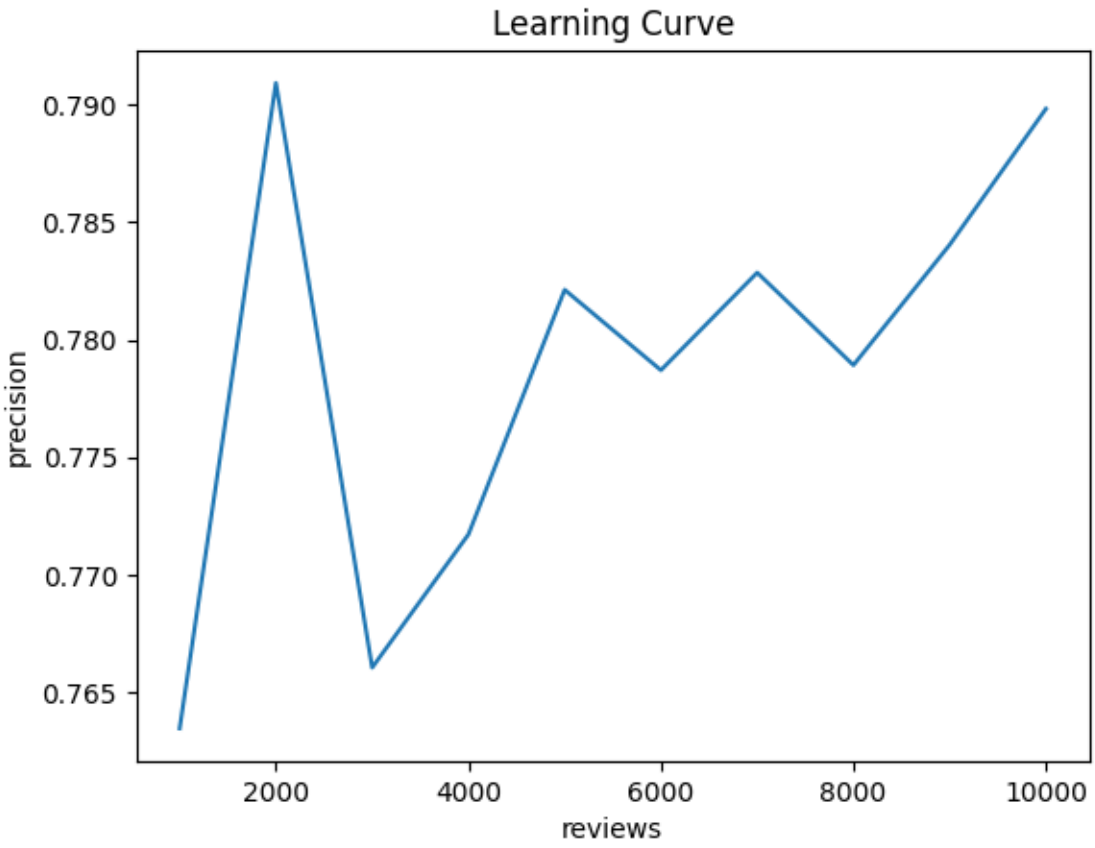
- Η accuracy που υπολογίζει το ποσοστό ακρίβειας στις προβλέψεις του αλγορίθμου.
- Η precision που υπολογίζει την ακρίβεια των προβλέψεων ως προς κάποια κατηγορία.
- Η recall που υπολογίζει την ανάκληση των παραδειγμάτων μιας κατηγορίας.
- Η F1 που υπολογίζει μια μετρική συνδυάζοντας τις precision recall.
- Και συναρτήσεις όπως οι learningCurveTest, learningCurveTrain, showCurves που είναι υπεύθυνες για την κατασκευή των καμπυλών.

Καμπύλες μάθησης και πίνακες:

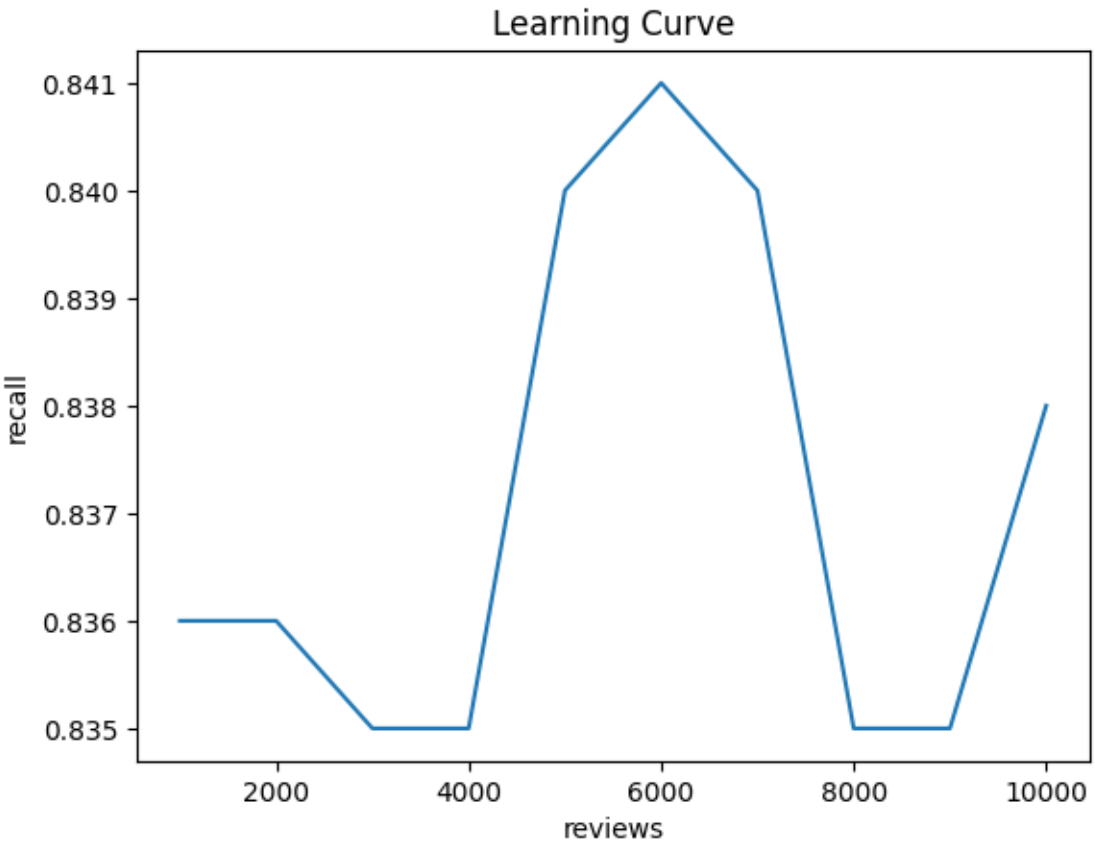
Accuracy :



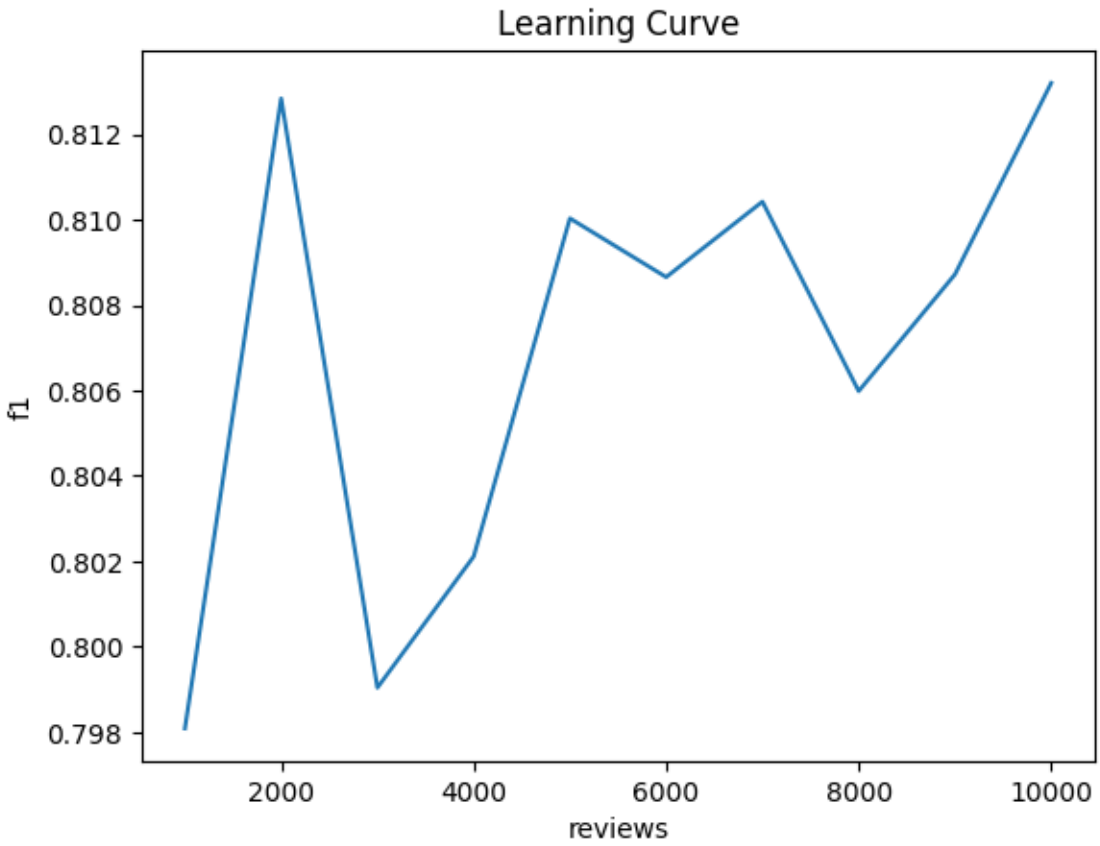
Percision :


Learning Curve
precision
0.790
0.785
0.780
0.775
0.770
0.765
2000
4000
6000
8000
10000
reviews

Recall :


Learning Curve
recall
0.841
0.840
0.839
0.838
0.837
0.836
0.835
2000
4000
6000
8000
10000
reviews

f1 :



Πίνακας:

training size	train accuracy	test accuracy	recall	percision	f1
1000	0.856	0.789	0.836	0.763	0.798
2000	0.849	0.808	0.836	0.790	0.812
3000	0.826	0.7905	0.835	0.766	0.799
4000	0.82025	0.7945	0.835	0.771	0.802
5000	0.813	0.8035	0.84	0.782	0.810
6000	0.816	0.8015	0.841	0.7787	0.8086
7000	0.8137	0.804	0.84	0.782	0.81
8000	0.814	0.7995	0.835	0.7789	0.8059
9000	0.810	0.803	0.835	0.784	0.8087
10000	0.8129	0.808	0.838	0.7898	0.81

αλγόριθμος ID3

Ο 2ος αλγόριθμος που υλοποιήσαμε στα πλαίσια της εργασίας αυτής είναι ο αλγόριθμος μάθησης ID3. Ο αλγόριθμος αυτός κατασκευάζει δέντρα απόφασης από παραδείγματα εκπαίδευσης(train data) στα οποία έχει εκπαιδευτεί προκειμένου να κατατάσσει νέα παραδείγματα αξιολόγησης(test data) στις σωστές κατηγορίες τους.

Η διαδικασία ανάπτυξης του αλγορίθμου περνά από 3 φάσεις:
Α) Την εκπαίδευση του αλγορίθμου στα παραδείγματα εκπαίδευσης(train data).
Β) Την επιλογή τιμών υπερπαραμέτρων του αλγορίθμου μέσω δοκιμών σε παραδείγματα επικύρωσης (validation data) .
Γ) Την χρήση και τελική αξιολόγηση του αλγορίθμου σε νέα παραδείγματα αξιολόγησης(test data) μέσω καμπυλών μάθησης , ακρίβειας , ανάκλησης και F1.

Φάση Α : (αρχείο id3.py)

Σε πρώτη φάση ξεκινάμε με την εκπαίδευση του αλγορίθμου μας . Επιλέγουμε 10.000 παραδείγματα εκπαίδευσης(5000 θετικά και 5000 αρνητικά) από τον φάκελο train του συνόλου δεδομένων imdb μας δόθηκε και τα παριστάνουμε με τις m = 550 συχνότερες λέξεις του συνόλου αυτού αφαιρώντας τις n = 50 πιο συχνές. Στην επόμενη φάση θα γίνει και η τελική επικύρωση των υπερπαραμέτρων (σε ξεχωριστά παραδείγματα ανάπτυξης), που τελικά θα χρησιμοποιηθούν.

Η εκπαίδευση του αλγορίθμου ξεκινά με μια αρχική κλήση της συνάρτησης id3alg η οποία με είσοδο τα δεδομένα εκπαίδευσης , τις επιλεγμένες ιδιότητες και μια προεπιλεγμένη κατηγορία κατασκευάζει το δέντρο απόφασης. Η κατασκευή του θα είναι αναδρομική. Αρχικά θα επιλέγεται η ιδιότητα(λέξη) με το μεγαλύτερο κέρδος πληροφορίας στα παραδείγματα εκπαίδευσης (συναρτήσεις: calculate_entropy,attribute_entropy,info_gain, max_IG) και με βάση αυτήν θα γίνεται έλεγχος στην ρίζα του δέντρου. Τότε θα χωρίζονται τα παραδείγματα εκπαίδευσης σε 2 υποσύνολα (μέσω της συνάρτησης split_data). Σε αυτό με τα παραδείγματα που περιέχουν την λέξη με το μέγιστο information gain και σε αυτό με τα παραδείγματα που δεν την περιέχουν. Σε αυτά τα 2 υποσύνολα παραδειγμάτων θα κληθεί η ίδια συνάρτηση id3alg αναδρομικά μέχρι να φτάσουμε σε κάποια οριακή περίπτωση όπου και θα καταλήξουμε σε φύλλο του δέντρου και θα παίρνουμε απόφαση κατηγοριοποίησης ,π.χ Όταν θα έχουν τελειώσει τα παραδείγματα εκπαίδευσης θα επιστρέφεται η προεπιλεγμένη κατηγορία, όταν θα έχουν τελειώσει οι ιδιότητες θα επιλέγεται η συχνότερη κατηγορία των εναπομεινάντων παραδειγμάτων ,ενώ όταν ένα μεγάλο ποσοστό των train data που έχουν απομείνει ανήκουν σε μία κατηγορία(έλεγχος μέσω της check_purity) θα επιστρέφεται η κατηγορία αυτή . Τελικά θα έχει παραχθεί το ζητούμενο δέντρο απόφασης στα δοσμένα από μας παραδείγματα εκπαίδευσης.
Για προγραμματιστικούς λόγους ορίσαμε και την κλάση SimpleTree που χρησιμοποιείται για τη δημιουργία του decision tree.

Φάση Β : (αρχείο validation.py)

Στην συνέχεια ακολουθεί η διαδικασία της επικύρωσης σε ξεχωριστά δεδομένα (2000 validate data με 1000 θετικά και 1000 αρνητικά παραδείγματα) στα οποία και ελέγχουμε με ποιες ιδιότητες μας συμφέρει να αναπαραστήσουμε τα παραδείγματα ,ελέγχοντας τα ποσοστά εγκυρότητας(accuracy) στα validation data. Συγκεκριμένα ελέγξαμε τη χρήση των 100,200,300,400 και 500 ποιο συχνών λέξεων αφαιρώντας κάθε φορά τις 50 συχνότερες.
Τα αποτελέσματα φαίνονται παρακάτω:

```
Creating the id3 decision tree for a dictionary of 100 words and 10k reviews...
Accuracy for a training set of 10k reviews (5k pos, 5k neg), a dictionary of100 words and a validation set of 2k is: 0.5795
Creating the id3 decision tree for a dictionary of 200 words and 10k reviews...
Accuracy for a training set of 10k reviews (5k pos, 5k neg), a dictionary of200 words and a validation set of 2k is: 0.653
Creating the id3 decision tree for a dictionary of 300 words and 10k reviews...
Accuracy for a training set of 10k reviews (5k pos, 5k neg), a dictionary of300 words and a validation set of 2k is: 0.6785
Creating the id3 decision tree for a dictionary of 400 words and 10k reviews...
Accuracy for a training set of 10k reviews (5k pos, 5k neg), a dictionary of400 words and a validation set of 2k is: 0.683
Creating the id3 decision tree for a dictionary of 500 words and 10k reviews...
Accuracy for a training set of 10k reviews (5k pos, 5k neg), a dictionary of500 words and a validation set of 2k is: 0.688
```

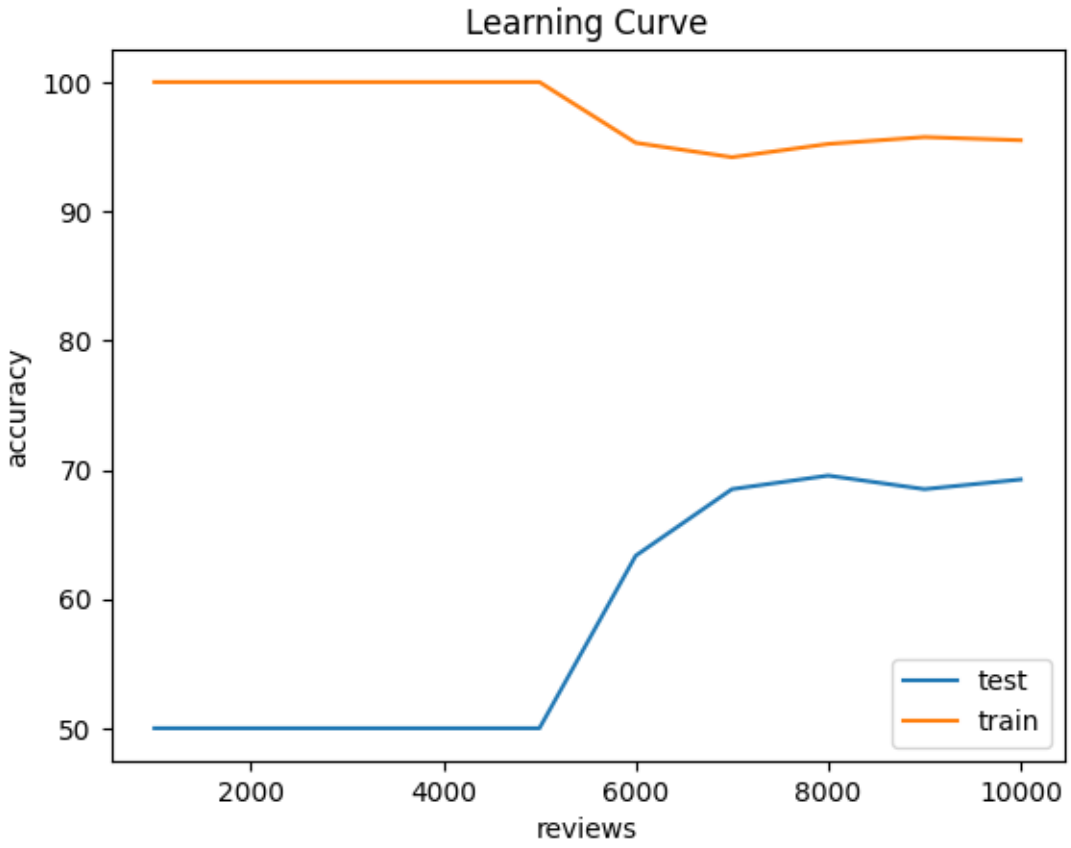
Έτσι επιλέξαμε να αναπαραστήσουμε τα παραδείγματα μας με τις 500 συχνότερες του λεξιλογίου που μας δίνουν και μεγαλύτερο ποσοστό ακρίβειας.

Φάση Γ : (αρχείο id3.py)

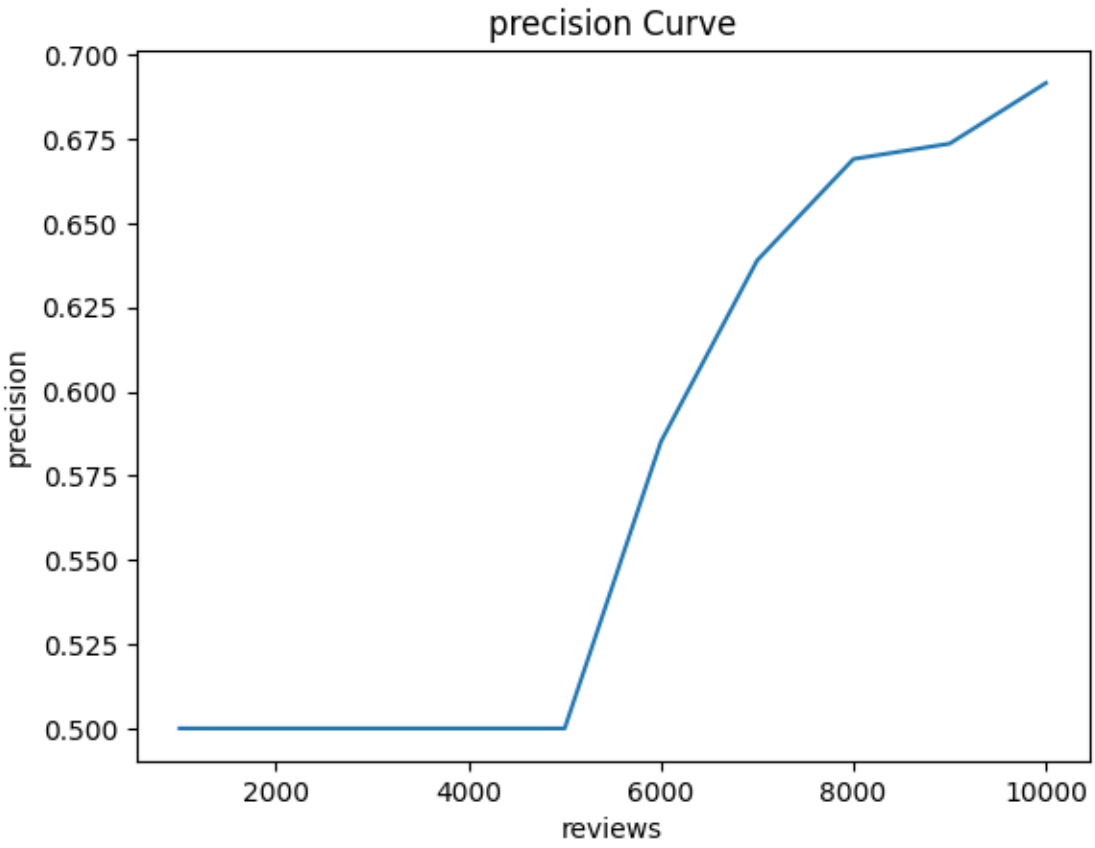
Ακολουθεί η αξιολόγηση του αλγορίθμου σε νέα παραδείγματα αξιολόγησης(2000 test data , 1000 θετικά και 1000 αρνητικά) . Συγκεκριμένα αξιολογούμε το δέντρο απόφασης μας όπως αυτό προέκυψε από την εκπαίδευση της 1ης φάσης με την υπερπαράμετρο που επιλέξαμε στην 2η φάση στα νέα αυτά δεδομένα. Μέσω της μεθόδου classify_review μπορούμε να κατατάξουμε κάθε review του test data με το δέντρο μας κάνοντας 'parse' ουσιαστικά της κριτικής αυτής στο δέντρο ανάλογα με το εάν αυτή περιέχει ή όχι την λέξη κάθε κόμβου. Στο τέλος καταλήγουμε σε ένα φύλλο του δέντρου όπου και παίρνεται από αυτό η απόφαση της κατηγοριοποίησης της κριτικής. Μετά την κατάταξη όλων των test data κριτικών μπορούμε να βγάλουμε συμπεράσματα για την ικανότητα μάθησης του αλγορίθμου μας ελέγχοντας την ακρίβεια κατάταξης(accuracy) του αλγορίθμου , την ακρίβεια(precision) , την ανάκληση(recall) και το F1. Ακολουθούν τα αντίστοιχα διαγράμματα και πίνακες του αλγορίθμου μας:

Καμπύλες μάθησης και πίνακες:

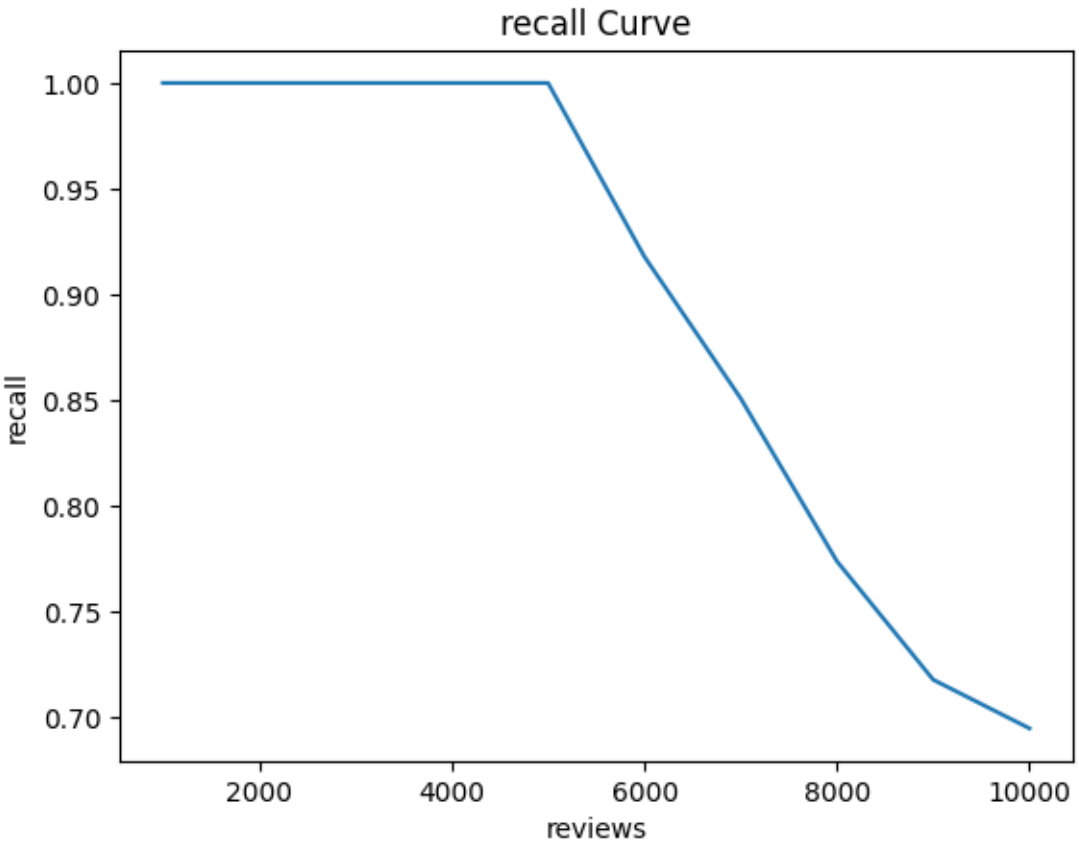
Accuracy :



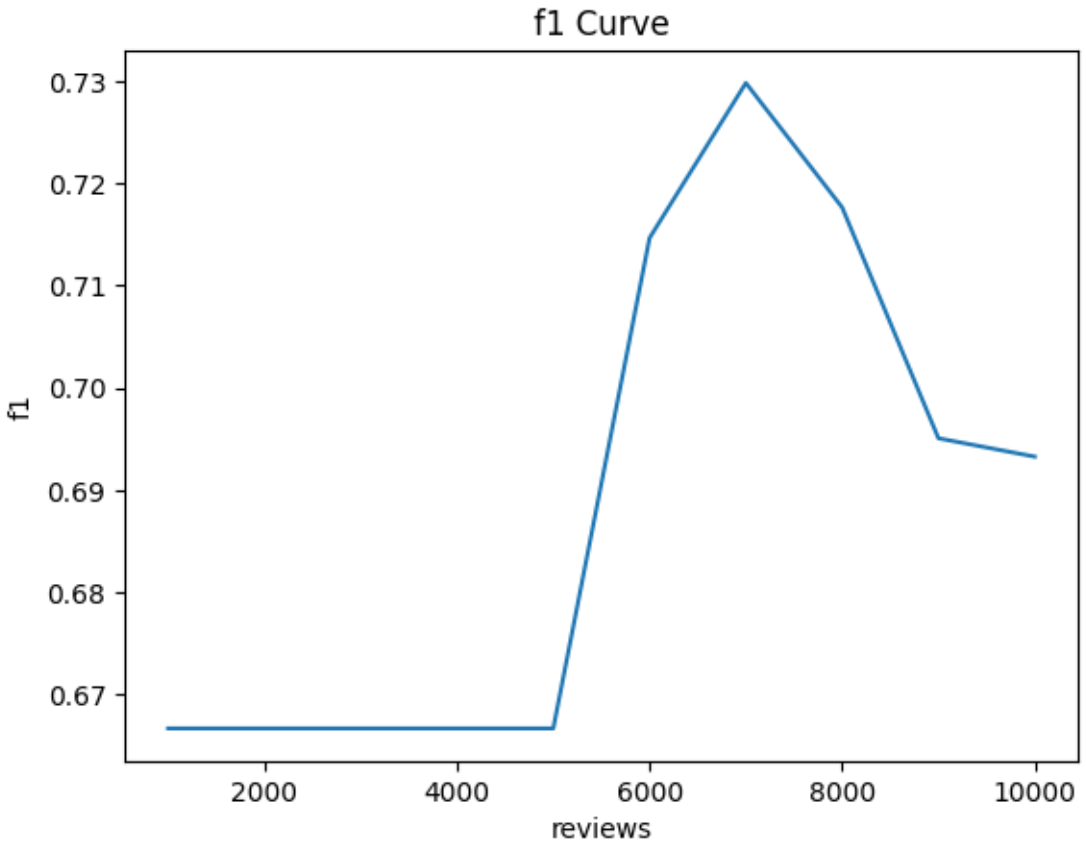
Percision :


precision Curve
precision
reviews
0.700
0.675
0.650
0.625
0.600
0.575
0.550
0.525
0.500
2000
4000
6000
8000
10000

Recall :


recall Curve
1.00
0.95
0.90
0.85
0.80
0.75
0.70
recall
2000
4000
6000
8000
10000
reviews

f1 :



Πίνακας:

training size	train accuracy	test accuracy	recall	percision	f1
1000	100.0	50.0	1.0	0.5	0.66
2000	100.0	50.0	1.0	0.5	0.66
3000	100.0	50.0	1.0	0.5	0.66
4000	100.0	50.0	1.0	0.5	0.66
5000	100.0	50.0	1.0	0.5	0.66
6000	95.3	63.349	0.918	0.585	0.7146
7000	94.1857	68.5	0.851	0.6388	0.7298
8000	95.2125	69.55	0.774	0.6689	0.7176
9000	95.74	68.5	0.718	0.6735	0.695
10000	95.5	69.25	0.695	0.691	0.693

** Για την δημιουργία των καμπυλών υλοποιήσαμε τις συναρτήσεις precision,recall,f1 και accuracy που υπολογίζουν την αντίστοιχη μετρική για κάθε δοσμένο πλήθος test data στον αλγόριθμο μας και την showCurve η οποία κατασκευάζει τις αντίστοιχες καμπύλες κάθε φορά

Αλγόριθμος Random Forest

Ο 3ος και τελευταίος αλγόριθμος μάθησης που υλοποιήσαμε είναι ο αλγόριθμος τυχαίου δάσους(random forest). Ο αλγόριθμος αυτός είναι αλγόριθμος συλλογικής μάθησης(ensemble learning) ο οποίος κατά την εκπαίδευσή του, κατασκευάζει πολλά δέντρα απόφασης(όπως αυτά στον ID3) και κατά την χρήση του, εμπιστευόμαστε την γνώμη της πλειοψηφίας των δέντρων (majority voting) για να προχωρήσουμε στην κατηγοριοποίηση ενός νέου παραδείγματος.

Όπως και στους 2 προηγούμενους αλγορίθμους , έτσι και στον random forest ακολουθήθηκαν 3 διακριτές φάσεις κατά την υλοποίηση του.
Α) Την εκπαίδευση του αλγορίθμου στα παραδείγματα εκπαίδευσης(train data).
Β) Την επιλογή τιμών υπερπαραμέτρων του αλγορίθμου μέσω δοκιμών σε παραδείγματα επικύρωσης (validation data) .
Γ) Την χρήση και τελική αξιολόγηση του αλγορίθμου σε νέα παραδείγματα αξιολόγησης(test data) μέσω καμπυλών μάθησης , ακρίβειας , ανάκλησης και F1.

Φάση Α : (αρχείο random$_forest.py$)

Ξεκινάμε με την εκπαίδευση του αλγορίθμου μας. Για λόγους ταχύτητας των προγραμμάτων μας χρησιμοποιήσαμε 1000 παραδείγματα εκπαίδευσης(train data, 500 θετικά και 500 αρνητικά). Αρχικά υλοποιήσαμε την συνάρτηση bootstrapping βάσει της οποίας θα επιλέγεται κάθε φορά για κάθε δέντρο το σύνολο των παραδειγμάτων στο οποίο αυτό θα εκπαιδευτεί. Από τα αρχικά 1000 train data το κάθε δέντρο θα εκπαιδεύεται σε μία διαφορετική παραλλαγή τους, επιλέγοντας 1000 παραδείγματα εξ' αυτών τυχαία με επανατοποθέτηση(bagging). Ακολούθως υλοποιήσαμε την random_attributes η οποία χρησιμοποιείται προκειμένου να επιλέγει από το σύνολο των (550-50) συχνότερων ιδιοτήτων που χρησιμοποιήθηκαν στον Id3 , ένα τυχαίο υποσύνολο 300 εξ΄ αυτών για την εκπαίδευση κάθε δέντρου.

Στην συνέχεια υλοποιήσαμε την train_forest η οποία εκπαιδεύει ένα τυχαίο δάσος με βάση τον αριθμό των δέντρων που θα επιλέξουμε(κατά tovalidation) και τα δεδομένα εκπαίδευσης που θα του δώσουμε. Συγκεκριμένα, κατασκευάζουμε ένα-ένα τα δέντρα του δάσους εκπαιδεύοντας τα μέσω της συνάρτησης id3alg που υλοποιήθηκε στον ID3 ,σε διαφορετική παραλλαγή του συνόλου εκπαίδευσης το καθένα(μέσω της bootstrapping) και χρησιμοποιώντας διαφορετικό υποσύνολο των διαθέσιμων ιδιοτήτων(μέσω της random_attributes) για κάθε ένα. Έτσι προκύπτει το τυχαίο δάσος που εκπαιδεύεται στα train data που επιλέξαμε.

Φάση Β : (αρχείο validation.py)

Ύστερα ακολούθησε η διαδικασία επιλογής του αριθμού των δέντρων που εν τέλει θα αποτελεί το τυχαίο δάσος. Για να επιλέξουμε αυτήν την σημαντική υπερπαράμετρο του αλγορίθμου μας , εξετάσαμε σε ξεχωριστά δεδομένα επικύρωσης validation data(500 reviews , 250 positive-250 negative για λόγους ταχύτητας) την ακρίβεια accuracy* που επιτυγχάνει ο αλγόριθμος εάν εκπαιδευτεί με διαφορετικό αριθμό δέντρων(πχ 5,11,21,31,51,...).
Παρακάτω φαίνονται τα αποτελέσματα:

```
Accuracy for validation data of 500 reviews (250 pos, 250 neg) and a random forest of 5 id3 decision trees is: 69.6%
Accuracy for validation data of 500 reviews (250 pos, 250 neg) and a random forest of 11 id3 decision trees is: 71.8%
Accuracy for validation data of 500 reviews (250 pos, 250 neg) and a random forest of 21 id3 decision trees is: 73.6%
Accuracy for validation data of 500 reviews (250 pos, 250 neg) and a random forest of 31 id3 decision trees is: 73.0%
Accuracy for validation data of 500 reviews (250 pos, 250 neg) and a random forest of 51 id3 decision trees is: 76.0%
```

Έτσι σύμφωνα με τα παραπάνω επιλέξαμε να χρησιμοποιήσουμε τυχαίο δάσος 51 δέντρων , αφού με αυτήν την υπερπαράμετρο πετυχαίνουμε την υψηλότερη ακρίβεια. *Έχουμε υλοποιήσει την συνάρτηση random_forest_accuracy, η οποία υπολογίζει την ακρίβεια των προβλέψεων ενός τυχαίου δάσους σε ένα σύνολο δεδομένων.

Φάση Γ : (αρχείο random_forest.py)

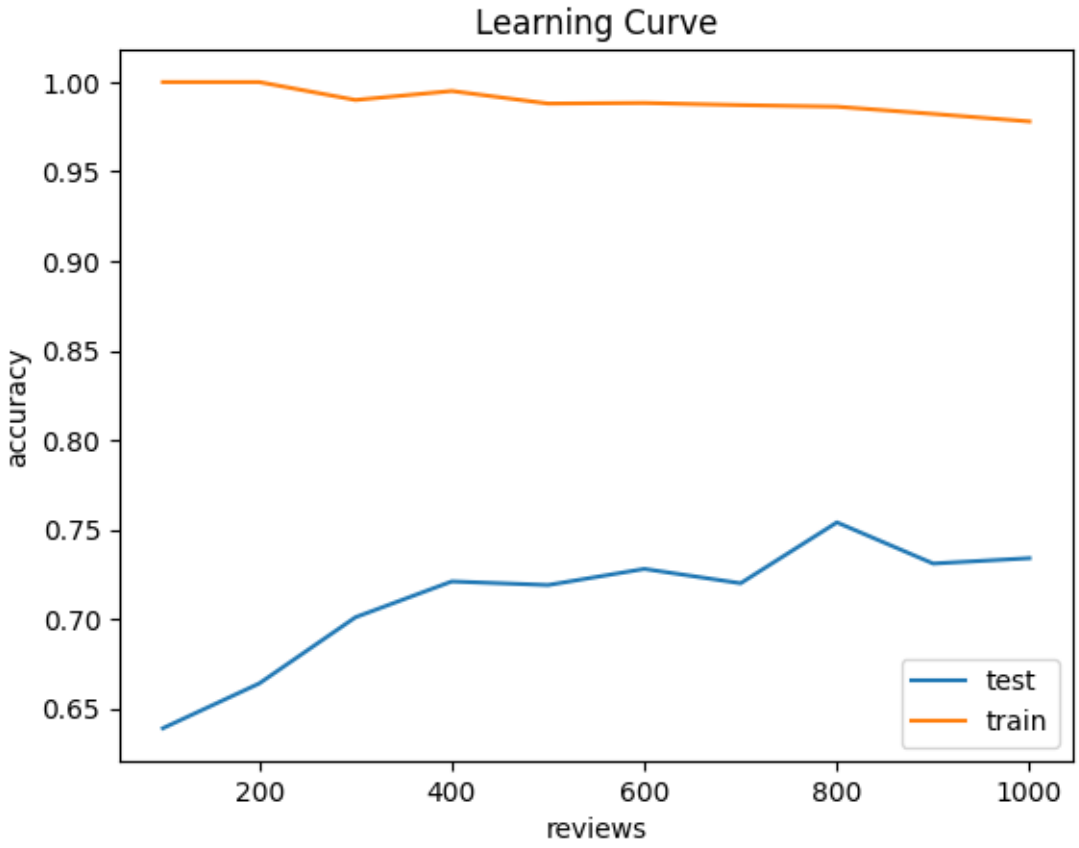
Συνεχίζουμε με την χρήση και αξιολόγηση του τυχαίου δάσους που εκπαιδεύσαμε παραπάνω με 51 δέντρα υπερπαράμετρο σε 1000 νέα παραδείγματα αξιολόγησης test data (500 θετικά, 500 αρνητικά). Για τον σκοπό αυτό χρειάστηκε να υλοποιήσουμε μια σειρά από βοηθητικές συναρτήσεις :

- Την predictions_for_specific_tree η οποία επιστρέφει τις προβλέψεις ενός συγκεκριμένου δέντρου σε δοσμένα test data.
- Την random_forest_predictions η οποία επιστρέφει τις προβλέψεις κάθε δέντρου ενός τυχαίου δάσους σε δοσμένα test data.
- Και την random_forest_decisions που δοσμένων των αποφάσεων των δέντρων του δάσους όπως προέκυψαν από κλήση της random_forest_predictions , κάνει την κατηγοριοποίηση των test data λαμβάνοντας υπόψιν την γνώμη της πλειοψηφίας των δέντρων(majority voting).

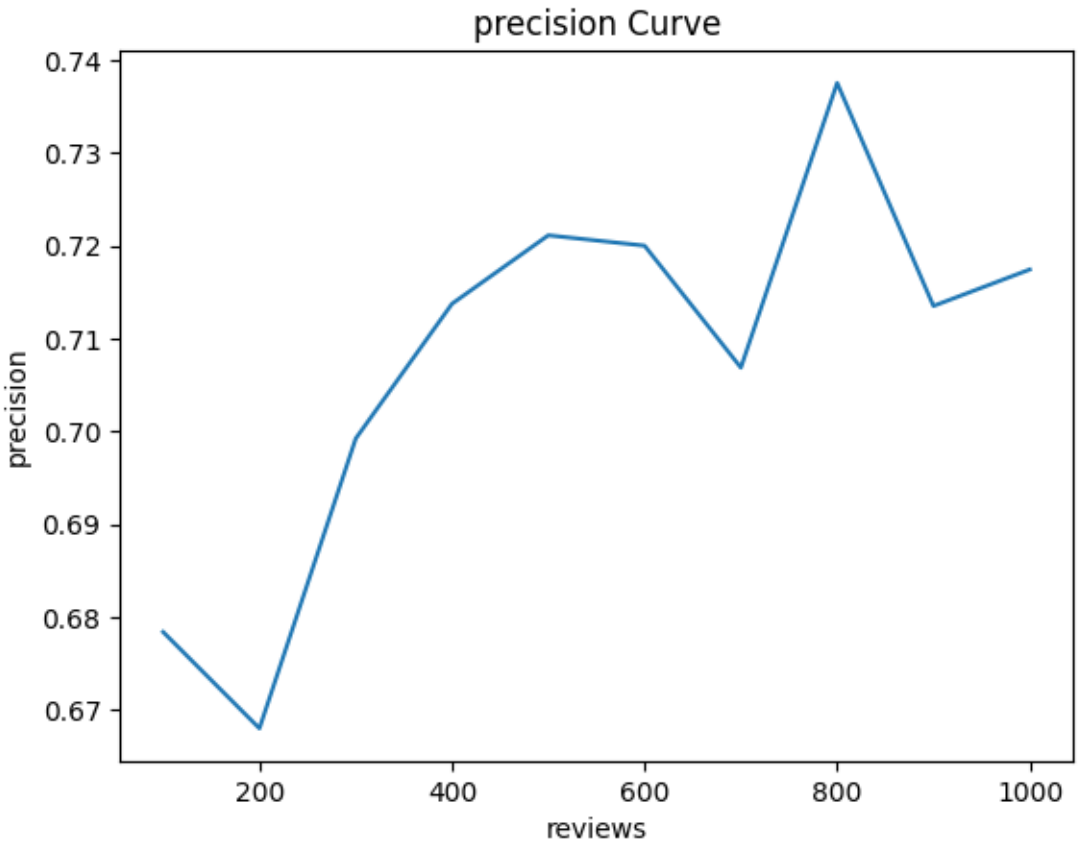
Χρησιμοποιώντας ,λοιπόν , τις παραπάνω συναρτήσεις αξιολογήσαμε το τυχαίο δάσος που προέκυψε από τις 2 πρώτες φάσεις σε 1000 test data όπως αναφέραμε και προηγουμένως. Ακολουθούν οι πίνακες και τα διαγράμματα ακρίβειας (accuracy), ανάκλησης(recall), ακρίβειας(precision) και F1 που δείχνουν οπτικά τις δυνατότητες μάθησης του αλγορίθμου random forest που υλοποιήσαμε:

Καμπύλες μάθησης και πίνακες:

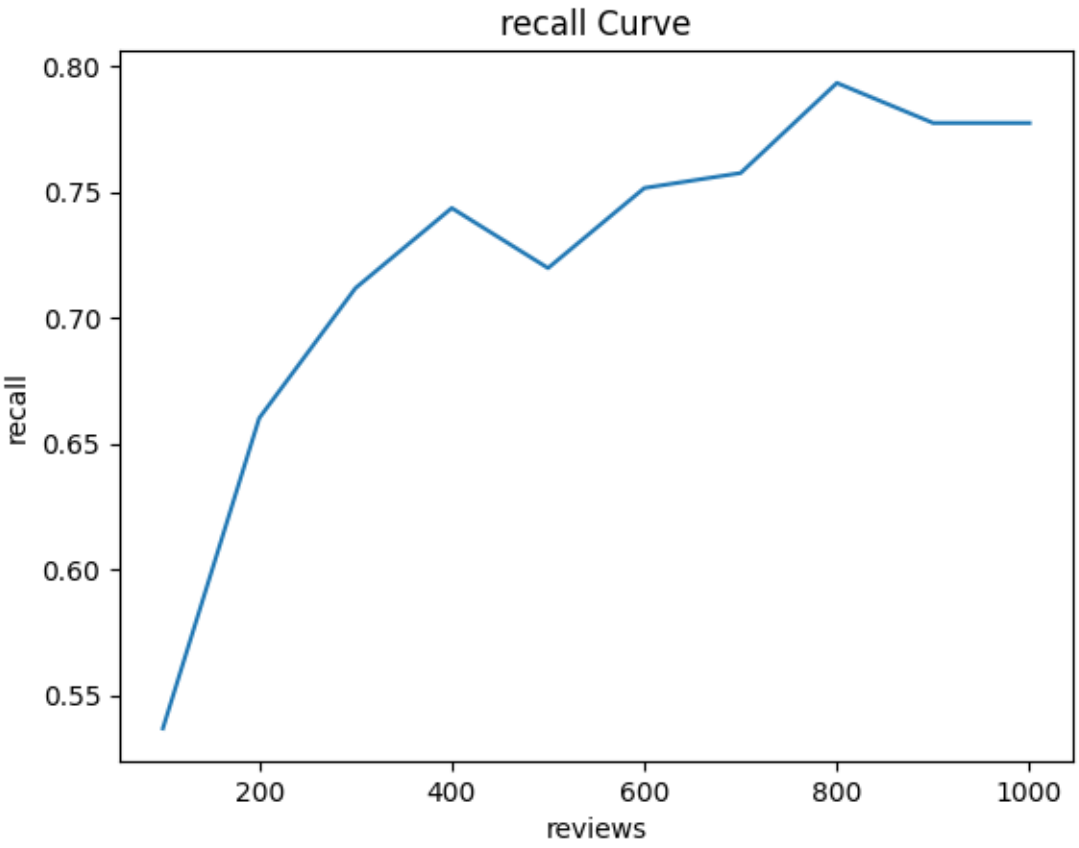
Accuracy :



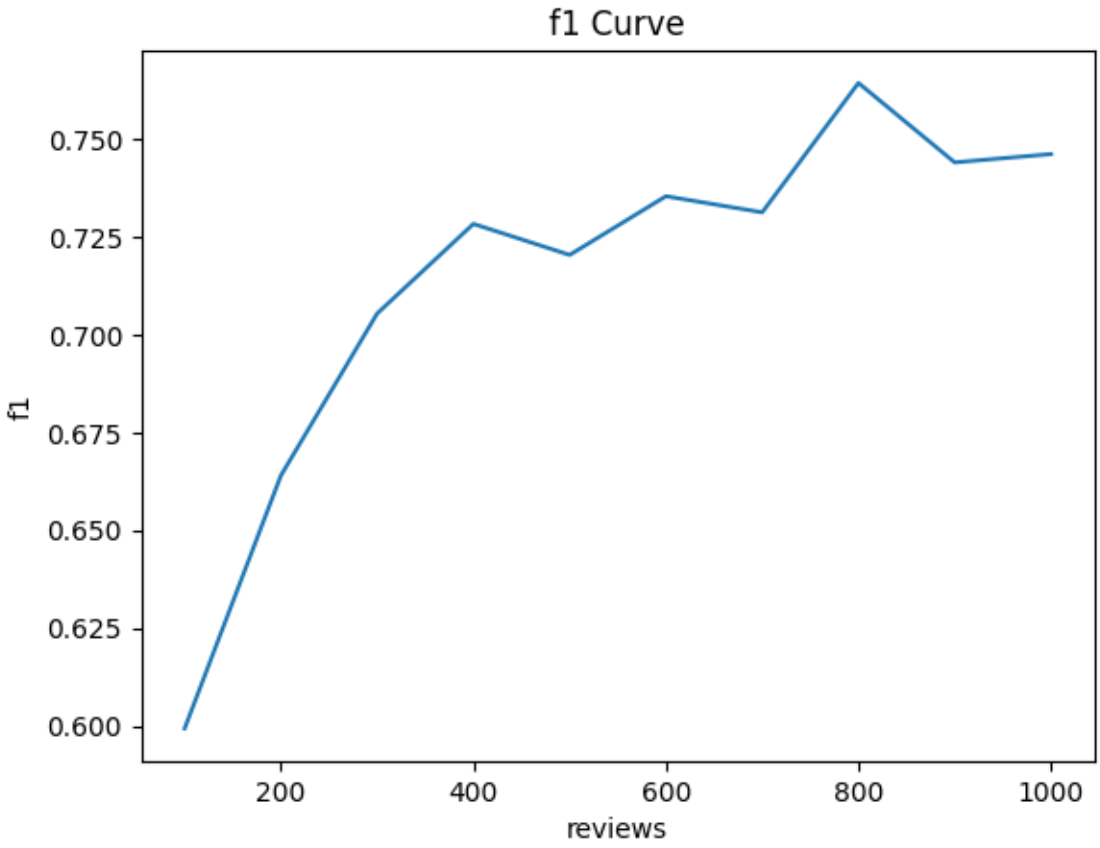
Percision :


precision Curve
precision
reviews
0.74
0.73
0.72
0.71
0.70
0.69
0.68
0.67
200
400
600
800
1000

Recall :


recall Curve
0.80
0.75
0.70
0.65
0.60
0.55
recall
200
400
600
800
1000
reviews

f1 :



Πίνακας:

training size	train accuracy	test accuracy	recall	percision	f1
100	1.0	0.639	0.5367	0.678	0.599
200	1.0	0.664	0.66	0.668	0.664
300	0.99	0.701	0.7117	0.699	0.705
400	0.995	0.721	0.7435	0.7137	0.728
500	0.988	0.719	0.7196	0.721	0.720
600	0.988	0.728	0.75	0.72	0.735
700	0.987	0.72	0.757	0.70	0.73
800	0.98625	0.754	0.79	0.737	0.76
900	0.982	0.731	0.777	0.7135	0.744
1000	0.978	0.734	0.777	0.717	0.746

* Για την δημιουργία των καμπυλών υλοποιήσαμε τις συναρτήσεις precision,recall,f1 και random_forest_accuracy που υπολογίζουν την αντίστοιχη μετρική για κάθε δοσμένο πλήθος test data στον αλγόριθμο μας και την showCurve η οποία κατασκευάζει τις αντίστοιχες καμπύλες κάθε φορά.