2η εργασία Τεχνητής Νοημοσύνης

Φώτιος Πάνος - p3180141

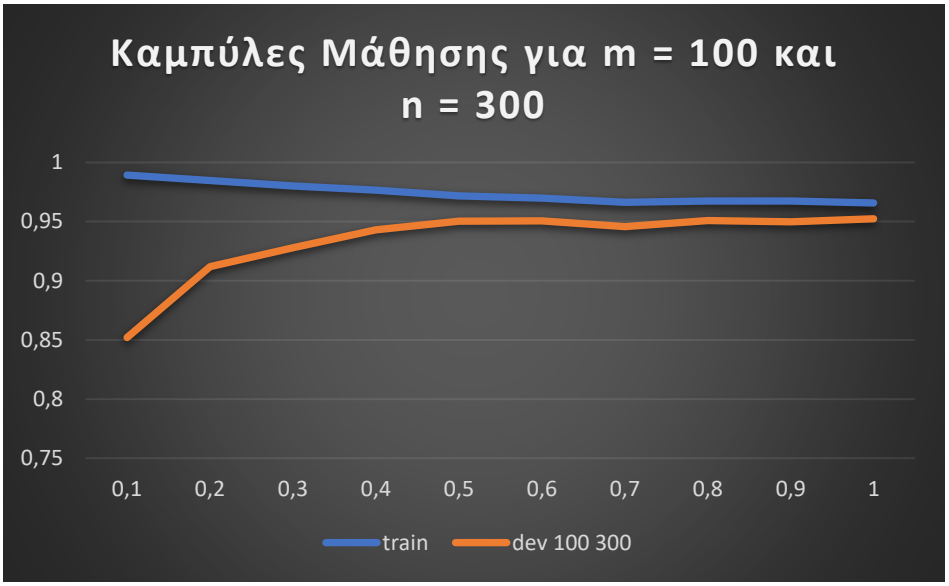
Ιωάννης Παπαχρήστου – p3180150

Αφελής Ταξινομητής Bayes:

Αρχικά υλοποιούμε τον αφελή ταξινομητή Bayes πάνω στα δεδομένα μας. Τα δεδομένα μας ήταν αρχικά χωρισμένα σε train και test, επομένως από τα train δεδομένα αφαιρέσαμε τυχαία το 10% αυτών και τα τοποθετήσαμε στο φάκελο dev, δημιουργώντας έτσι ξεχωριστά δεδομένα για τη διαδικασία του development, όπου εξετάζουμε διάφορες τιμές για τις υπερπαραμέτρους m και n και βρίσκουμε τις βέλτιστες. Ξεκινώντας, ορίζουμε μία μέθοδο text_cleaning για την επεξεργασία κάθε κειμένου που θέλουμε να διαβάσουμε, έτσι ώστε να απομονώσουμε μόνο τις λέξεις από το κείμενο και να μπορούμε να το επεξεργαστούμε. Αφαιρούμε δηλαδή τα σημεία στίξης και μετατρέπουμε όλα τα κεφαλαία γράμματα σε μικρά. Στη συνέχεια ορίζουμε μία μέθοδο ταξινόμησης sort, η οποία μέσω της μεθόδου sorted της python ταξινομεί το λεξικό με φθίνουσα σειρά. Επόμενη είναι η total_freq, η οποία υπολογίζει το άθροισμα των συχνοτήτων των λέξεων ενός λεξικού. Η probability_count είναι μία από τις βασικές μεθόδους για την υλοποίηση του αφελή ταξινομητή Bayes, καθώς υπολογίζει την πιθανότητα μία κριτική να αξιολογηθεί ως καλή ή ως κακή. Οι επόμενες δύο μέθοδοι, testNegatives και testPositives καλούνται κατά τη διαδικασία αξιολόγησης των train, test και dev δεδομένων. Κάθε μία εξ αυτών διαβάζει και αξιολογεί τις κριτικές από τον neg ή τον pos φάκελο αντίστοιχα. Η train είναι η μέθοδος με την οποία γίνεται η αρχική εκπαίδευση των δεδομένων μας. Για συγκεκριμένο ποσοστό των συνολικών δεδομένων επιστρέφει τα δύο λεξικά με τις λέξεις των αρνητικών κριτικών και των θετικών κριτικών ταξινομημένα και στη συνέχεια δίνει το αντίστοιχο ποσοστό των δεδομένων στον αλγόριθμο για να τα ταξινομήσει και να δούμε το ποσοστό επιτυχίας που πετυχαίνει. Η dev για συγκεκριμένο ποσοστό των dev εξετάζει το ποσοστό επιτυχίας νέων κριτικών για τον αλγόριθμο. Η test κάνει ακριβώς το ίδιο πράγμα με την dev αλλά για καινούργια δεδομένα. Η διαφορά μεταξύ τους εμφανίζεται κατά την κλήση τους όπου η dev καλείται για πολλές διαφορετικές τιμές των m και n, συμπεραίνει ποιες είναι οι καλύτερες εξ αυτών και τις δίνει στην test η οποία καλείται μία φορά για τις βέλτιστες m και n. Αφού τελειώσαμε με τους ορισμούς των μεθόδων μας πάμε στη main loop. Αρχικά, για ποσοστά των συνολικών δεδομένων από 10% έως 100% με βήμα 10 καλούμε την train και για το ίδιο ποσοστό καλούμε την dev για διαφορετικούς συνδυασμούς των m, n, ενώ ταυτόχρονα αξιολογούμε και κρατάμε τις βέλτιστες τιμές των m, n για τις οποίες έχουμε επιτύχει τη μέγιστη ορθότητα. Κρατάμε σε αρχείο τα καλύτερα m,n καθώς και το καλύτερο accuracy. Έχοντας βρει τις καλύτερες τιμές m,n επαναλαμβάνουμε την ίδια λούπα, αυτή τη φορά για τα test δεδομένα, έτσι ώστε να υπολογίσουμε τα accuracy, precision, recall και F1.

Τρέχοντας τον αλγόριθμο για διαφορετικές τιμές των υπερπαραμέτρων m και n προκύπτουν διαφορετικά διαγράμματα καμπύλων μάθησης. Ξεχωρίσαμε το παρακάτω ως ενδεικτικό, ενώ μπορείτε να δείτε το σύνολο των διαγραμμάτων στο συνοδευτικό αρχείο bayes_graphs.xlsx. Στα διαγράμματα η πορτοκαλί γραμμή αντικατοπτρίζει την καμπύλη μάθησης για τα dev δεδομένα και η ετικέτα είναι της μορφής «dev m n», πχ για dev 100 300 αναφερόμαστε στον έλεγχό που έγινε για τα dev δεδομένα με m = 100 και n = 300.

	train	dev 100 300
0,1	0,989333333	0,852
0,2	0,984666667	0,912
0,3	0,98	0,928
0,4	0,976555556	0,943
0,5	0,971822222	0,9504
0,6	0,969777778	0,950666667
0,7	0,966222222	0,945714286
0,8	0,967333333	0,951
0,9	0,967506173	0,949777778
1	0,965777778	0,9524

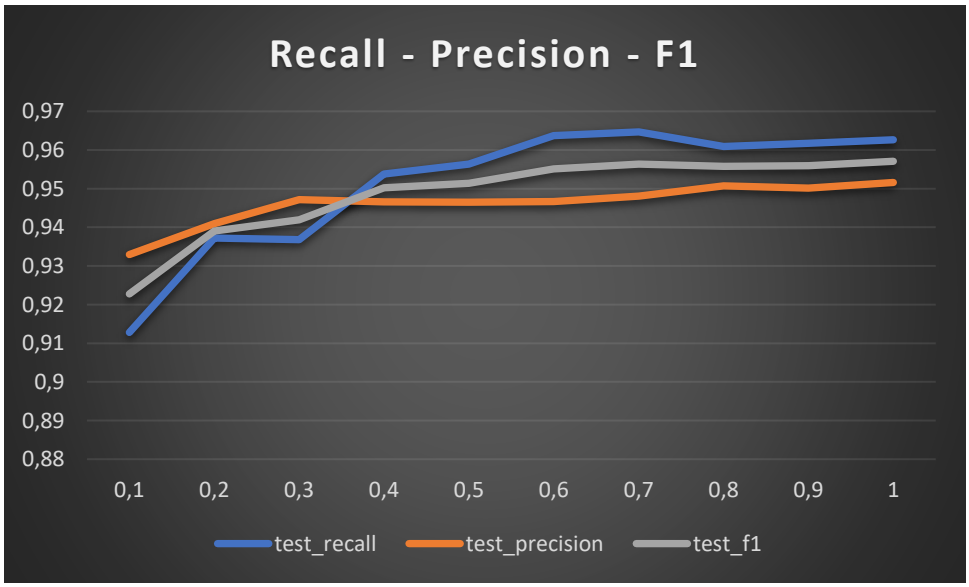


Βλέπουμε πως η συμπεριφορά του αλγορίθμου μας είναι αρκετά καλή, δεδομένου πως η dev καμπύλη υποδεικνύει πως αλγόριθμός μας μαθαίνει και βελτιώνεται σημαντικά μέχρι και το 60% των νέων δεδομένων, ενώ παρατηρείται και μια μικρή αύξηση μέχρι το τέλος. Το γεγονός πως παραμένει σταθερά σε ποσοστό ορθότητας άνω του 85% μας δείχνει πως αξιολογεί και ταξινομεί άριστα, δεδομένου πως πρόκειται για αφελή αλγόριθμο, τις δοθείσες κριτικές.

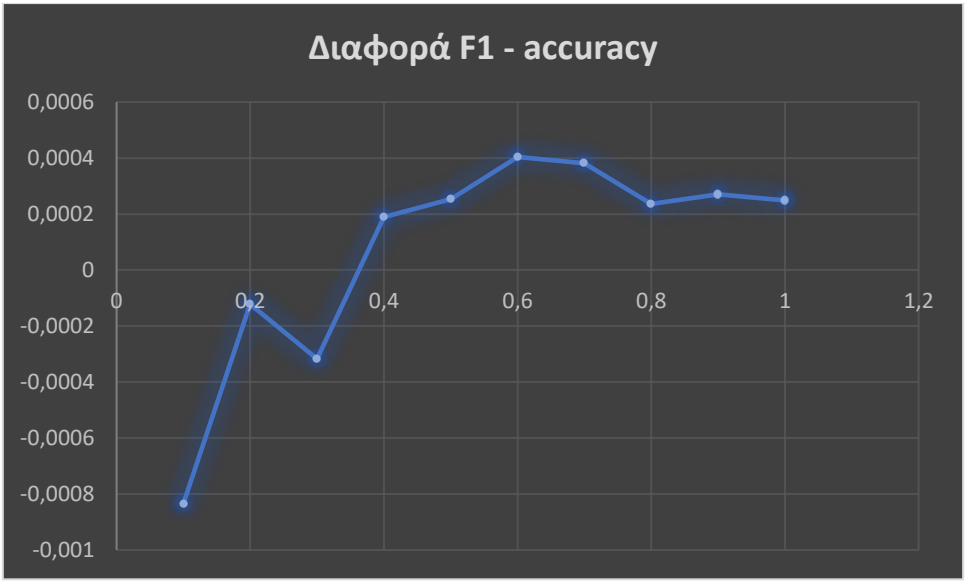
Κατά την διάρκεια του ελέγχου για τα dev δεδομένα προκύπτει πως η βέλτιστη ορθότητα επιτεύχθηκε για τα m,n, που ανήκουν στο διάστημα {100, 200, 300}, με τιμές 200 και 100 αντίστοιχα. Κρατάμε, δηλαδή 200 λέξεις από το εκάστοτε λεξικό, αφού παραλείψουμε τις πρώτες 100.

Κάνοντας τον έλεγχο και αξιολόγηση των test δεδομένων, προέκυψαν οι παρακάτω τιμές για τα precision, recall και F1, οδηγώντας μας στο αντίστοιχο διάγραμμα:

	test_recall	test_precision	test_f1
0,1	0,9128	0,932951758	0,922765871
0,2	0,9372	0,940963855	0,939078156
0,3	0,9368	0,947155568	0,941949323
0,4	0,9538	0,946605796	0,950189281
0,5	0,95632	0,946476643	0,951372861
0,6	0,963733333	0,946692862	0,955137099
0,7	0,964685714	0,94810738	0,956324704
0,8	0,9609	0,950727219	0,955786542
0,9	0,961777778	0,950122936	0,955914833
1	0,96264	0,951601423	0,957088884



Παρατηρήσαμε επίσης πως η F1 αποτελεί μια διαφορετική προσέγγιση για την accuracy του αλγορίθμου μας, αφού όπως φαίνεται και από το παρακάτω διάγραμμα παίρνουν προσεγγιστικά τις ίδιες τιμές με διαφορά μικρότερη της τάξης του 10^{-3}:



Σε σύγκριση που δοκιμάσαμε να κάνουμε με άλλη ομάδα παρατηρήσαμε πως ο αλγόριθμός μας επιτυγχάνει αρκετά μεγαλύτερες τιμές accuracy, precision και recall, καθώς οι δικές τους τιμές κυμαίνονταν στο διάστημα 70-80% και έτσι οι καμπύλες μας δεν ήταν δυνατόν να συγκριθούν και να βγει κάποιο συμπέρασμα αφού και οι δύο φαίνεται πως έχουμε υλοποιήσει τον αλγόριθμο σωστά και οι λύσεις μας διαφέρουν μόνο στη εκτιμήτρια Laplace που υλοποιήσαμε διαφορετικά.

ID3:

Ο ID3 δεν έχει υλοποιηθεί πλήρως λόγω έλλειψης χρόνου, παρόλα αυτά έχουμε ορίσει τις βασικές μεθόδους για τη λειτουργία του. Αρχικά, ορίζουμε μία μέθοδο text_cleaning για την επεξεργασία κάθε κειμένου που θέλουμε να διαβάσουμε, έτσι ώστε να απομονώσουμε μόνο τις λέξεις από το κείμενο και να μπορούμε να το επεξεργαστούμε. Αφαιρούμε δηλαδή τα σημεία στίξης και μετατρέπουμε όλα τα κεφαλαία γράμματα σε μικρά. Στη μέθοδο train γίνεται η αρχική εκπαίδευση των δεδομένων μας. Στη συνέχεια ορίζουμε τις 3 κλάσεις που είναι απαραίτητες για την υλοποίηση του αλγορίθμου. Η data χρησιμοποιείται για την αποθήκευση των δεδομένων μας. Περιέχει μια λίστα από λέξεις καθώς και ένα χαρακτηριστικό type που τις χαρακτηρίζει ως negative ή positive. Η Node αναπαριστά τους κόμβους του δέντρου μας. Η κλάση ID3 είναι η κεντρική κλάση πάνω στην οποία στηρίζεται ο αλγόριθμος και ουσιαστικά σχεδιάζει το δέντρο του ID3. Σε κάθε κόμβο εξετάζει ένα συγκεκριμένο χαρακτηριστικό και στη συνέχεια δημιουργεί αναδρομικά αριστερό και δεξί υποδέντρο. Οι μέθοδοι wordExists και wordDoesntExist χρησιμοποιούνται για τη δημιουργία των δύο υποδέντρων, αριστερό και δεξί. Στη συνέχεια ακολουθούν 3 μέθοδοι που υπολογίζουν εντροπία, information gain και max information gain. Τέλος ακολουθούν οι κλήσεις του αλγορίθμου που δημιουργούν τα δύο datasets μέσω της train για τα positive και τα negative δεδομένα και δημιουργεί το δέντρο του ID3.