2η Εργασία

Μέλη Ομάδας:

* Σοφία Βέργη, p3200018
* Παναγιώτα Φυτάλη , p3200215
* Σαραντία Μουρελάτου, p3200106

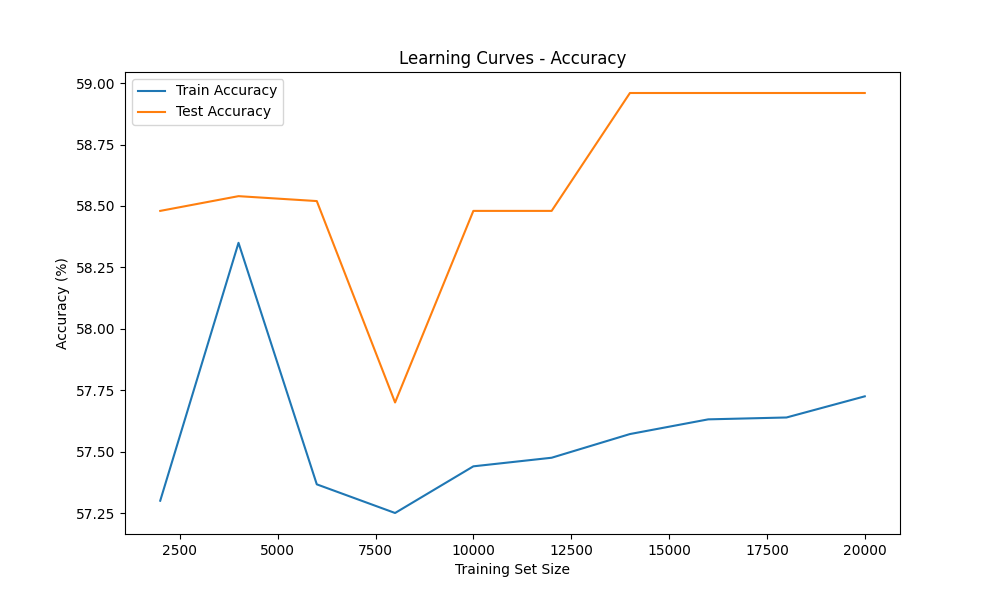
Μέρος Α’

Αρχικά στο αρχείο **Main.py** επεξεργαστήκαμε το αρχείο imdb.vocab, που περιέχει κάθε λέξη που εμφανίζεται στα κείμενα των κριτικών και αποτελεί το αρχικό μας λεξιλόγιο, το imdbEr.txt ,στο οποίο κάθε εγγραφή αντιστοιχεί με την λέξη στην αντίστοιχη θέση του πρώτου αρχείου και αποτελεί την πολικότητα της λέξης ,δηλαδή το πόσο θετική ή αρνητική μπορεί να χαρακτηριστεί και το labeledBow.feat ,το οποίο περιέχει το πλήθος των φορών που εμφανίζεται μια λέξη του κειμένου μέσα στο κείμενο. Από το αρχικό μας λεξιλόγιο φτιάξαμε ένα λεξικό το οποίο περιέχει για κάθε λέξη την ολική συχνότητα εμφάνισης της στα κείμενα και αφαιρέσαμε σε πρώτο στάδιο τις λέξεις με πολικότητα<0.7 ,καθώς οι συγκεκριμένες θεωρούνται αρκετά ουδέτερες και δεν θα χρησίμευαν στην σωστή κατηγοριοποίηση των κειμένων, μπορεί μάλιστα η παρουσία τους να συνέβαλε στο ακριβώς αντίθετο. Τέτοιες λέξεις ,γνωστές και ως stopwords,είναι οι he,she,I,am,there’s κ.α. Ύστερα από τις εναπομείνασες λέξεις ,οι οποίες είναι 26.241 στο σύνολο ,επιλέγουμε ποιες θα κρατήσουμε για να διαμορφώσουμε το τελικό μας λεξιλόγιο μέσω των υπερπαραμέτρων m,n και k. Σε τελικό στάδιο για να μετατρέψουμε το κάθε κείμενο σε ένα διάνυσμα ιδιοτήτων (κάθε λέξη αποτελεί μια ιδιότητα) με τιμές 0 ή 1 «καθαρίσαμε» τις λέξεις του κάθε κειμένου μετατρέποντας τες όλες σε μικρά γράμματα ,αφαιρώντας σημεία στίξης κλπ. και έπειτα ελέγξαμε για κάθε λέξη του κειμένου αν υπάρχει μέσα στο τελικό λεξιλόγιο και αν ναι της αναθέταμε την τιμή 1 ,διαφορετικά την τιμή 0. Τα διανύσματα των κειμένων τα αποθηκεύσαμε σε μορφή dataframe ,όπου κάθε στήλη αντιστοιχεί σε μία λέξη-ιδιότητα και η τελευταία στήλη ονόματι rating αντιστοιχεί στην κατηγορία του κειμένου ( περιέχει την τιμή 0 για την αρνητική κατηγορία και την τιμή 1 για την θετική). Φτιάξαμε 2 dataframes ,ένα για τις θετικές κριτικές και ένα για τις αρνητικές και αποθηκεύσαμε το καθένα σε ένα αρχείο csv ,το οποίο παίρνουμε ως είσοδο σε κάθε αλγόριθμο για να το επεξεργαστούμε.

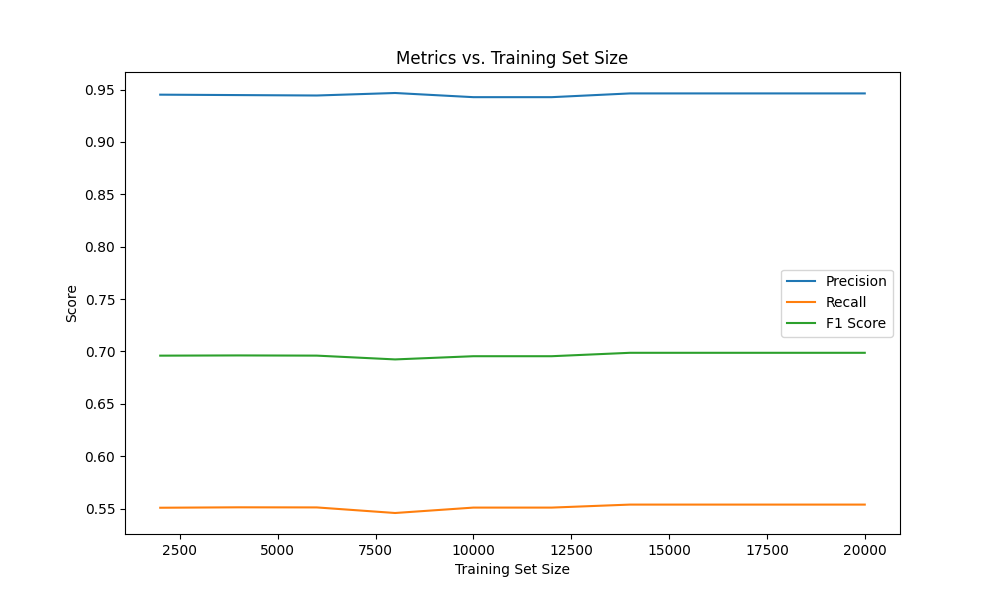
AdaBoost

Όπως ήδη γνωρίζουμε ο AdaBoost είναι ένας από τους πιο γνωστούς αλγόριθμους ενδυνάμωσης, ο οποίος «εκπαιδεύεται» με την αξιολόγηση κάθε φορά ενός μόνο χαρακτηριστικού, αυτή γίνεται με το λεγόμενο stump, το οποίο είναι ένα δέντρο αποφάσεων ύψους 1. Ο στόχος της εργασίας μας, συνοπτικά, είναι από το σύνολο δεδομένων που μας δίνεται να το επεξεργαστούμε με σκοπό να κατανοήσουμε, με βάση ένα «λεξικό» που έχουμε και τον εντοπισμό των λέξεων αυτών σε κάθε κριτική, αν η κριτική είναι θετική ή αρνητική, και με εκπαίδευση του αλγορίθμου με νέες λέξεις και κριτικές κάθε φορά να καταφέρουμε να αξιολογούμε νέες κριτικές ακόμη πιο έγκυρα. Ουσιαστικά, έχουμε κτίσει τον αλγόριθμο AdaBoost, ο οποίος δέχεται ως ορίσματα ένα διάνυσμα (train\_vector ή test\_vector), το οποίο αντιπροσωπεύει ένα 2D πίνακα με κάθε γραμμή του να αντιστοιχεί σε μία κριτική και κάθε στήλη σε μία λέξη από το λεξικό μας(το λεξικό μας αντιστοιχεί στις λέξεις που κρατήσαμε με βάση της παραμέτρους m,k,n από το σύνολο των λέξεων που μας δόθηκε από το imdb\_vocab), αν η τιμή σε κάποια θέση του πίνακα είναι το 0 σημαίνει ότι η λέξη της στήλης δεν υπάρχει στην συγκεκριμένη κριτική και το 1 ότι υπάρχει. Με βάση το σύνολο των λέξεων που υπάρχουν σε μία κριτική και του αν η ίδια η λέξη αξιολογείται ως θετική ή αρνητική, όπου γίνεται με βάση την υπόθεση που κάνει το stump, η εκπαίδευση του αλγορίθμου σταματάει όταν δημιουργήσουμε όλα τα stump που θέσαμε εξαρχής (εμείς έχουμε βάλει να δημιουργούνται 6), συγκρίνουμε αυτή την αξιολόγηση με την συνολική της κριτικής. Πιο απλά, αν έχουμε μία λέξη που υποθέσαμε ότι θεωρείται θετική και την συναντήσαμε σε μία θετική κριτική σημαίνει ότι «καλύφθηκε» από το stump που έκανε την υπόθεση, ενώ αν γίνει το αντίθετο δηλαδή υποθέσουμε ότι μία λέξη έχει αρνητική αξιολόγηση σε μία θετική κριτική, τότε θεωρούμε ότι το stump δεν την κάλυψε και χρειάζεται να την ξανά επεξεργαστούμε στο επόμενο, το οποίο γίνεται αντιληπτό ως αύξηση του βάρους της λέξης. Το βάρος κάθε λέξης είναι, ουσιαστικά, το πόσο απαραίτητη είναι η αξιολόγηση του, για το τελικό συμπέρασμα, αρχικά όλες οι λέξεις είναι ίδιας βαρύτητας, ίση με 1/(πλήθος\_λέξεων) και στην πορεία δημιουργίας των stump αυξάνεται το βάρος των λέξεων που δεν καλύφθηκαν από το stump, ενώ κατά συνέπεια, των άλλων μειώνεται, καθώς κάθε φορά το άθροισμα όλων των βαρών των λέξεων πρέπει να είναι 1 (άρα η μείωση του βάρους των «καλυπτόμενων» λέξεων προέρχεται από την κανονικοποίηση). Όπως οι λέξεις χαρακτηρίζονται με το βάρος τους έτσι και το stump έχει ένα συνολικό βάρος, όπου όσο πιο ψηλό είναι τόσο καλύτερο, πιο «καλυπτικό», θεωρήθηκε. Με βάση αυτό αξιολογούμε τελικά πόσο ακριβής και αξιόπιστος είναι ο αλγόριθμός μας. Για να γίνει αυτό, όμως, πρέπει να κάνουμε πολλές διαφορετικές διασπάσεις των δεδομένων σε εκπαίδευσης και ελέγχου, ώστε να έχουμε μία συνολικότερη εικόνα του πως ο AdaBoost προσαρμόζεται στην νέα πληροφορία, για αυτό έχουμε αναπτύξει την μέθοδο evaluate\_adaboost, που δέχεται ως ορίσματα ολόκληρο το σύνολο των δεδομένων εκπαίδευσης και τα διασπά σε κάθε επανάληψη σε διαφορετικά σημείο και το σύνολο των δεδομένων ελέγχου καθώς και τις αντίστοιχες κατηγορίες (αν είναι θετική ή αρνητική) ανά κριτική κάθε σετ, επίσης θέτουμε και δύο ακόμη παραμέτρους (max\_train\_size, step\_size) οι οποίες είναι για την διάσπαση των δεδομένων, που προαναφέραμε, η πρώτη έχει τεθεί ως το μήκος του train\_vector, που είναι το σύνολο εκπαίδευσης που έχει δημιουργηθεί από τα δεδομένα εκπαίδευσης, ενώ η δεύτερη παράμετρος ορίζει το βήμα αυξομείωσης του μεγέθους του συνόλου εκπαίδευσης. Στον παραπάνω κώδικα, έχει τεθεί ως το 10% του max\_train\_size, καθιστώντας τον αλγόριθμο να αξιολογείται σε συγκεκριμένα επίπεδα μεγέθους του συνόλου εκπαίδευσης. Με αυτούς τους παράγοντες μπορεί να προσδιοριστεί πώς θα αξιολογηθεί η απόδοση του AdaBoost σε διάφορα μεγέθη συνόλων εκπαίδευσης. Η απόδοση του αλγορίθμου μας μπορούμε να παρουσιάσουμε με εμφάνιση των καμπυλών μάθησης και των αντίστοιχων πινάκων που δείχνουν το ποσοστό ορθότητας (accuracy) στα δεδομένα εκπαίδευσης (training data)(όσα έχουν χρησιμοποιηθεί κάθε φορά) και ελέγχου (test data) συναρτήσει του πλήθους των παραδειγμάτων εκπαίδευσης που χρησιμοποιούνται σε κάθε επανάληψη του πειράματος, καθώς και τις αντίστοιχες καμπύλες και πίνακες με αποτελέσματα ακρίβειας (precision), ανάκλησης (recall), F1 για μία από τις δύο κατηγορίες, συναρτήσει του πλήθους των παραδειγμάτων εκπαίδευσης. Αρχικά για να διευκολυνθεί και η μετέπειτα ανάλυση των καμπυλών και πινάκων είναι αρκετά σημαντικό να αναφερθούμε στο τι εκφράζει το κάθε μέγεθος. Ξεκινώντας με την ορθότητα/ακρίβεια εκπαίδευσης (Train Accuracy), όταν αυξάνεται με την αύξηση του μεγέθους του συνόλου εκπαίδευσης, σημαίνει ότι το μοντέλο είναι σε θέση να μάθει καλύτερα τα δεδομένα εκπαίδευσης όσο αυτό το σύνολο γίνεται μεγαλύτερο. Αντίστοιχα η αύξηση της ακρίβειας ελέγχου (Test Accuracy), υποδηλώνει ότι το μοντέλο είναι σε θέση να γενικεύει καλά σε νέα δεδομένα. Η ακρίβεια, η ανάκληση και το f1 το σχολιάζουμε με βάση τις αρνητικές κριτικές, επομένως έχουμε: αύξηση ακρίβειας (Precision) να σημαίνει ότι το μοντέλο είναι καλύτερο στο να αναγνωρίζει τα αρνητικά δείγματα όσο αυξάνεται το μέγεθος του συνόλου εκπαίδευσης. Η ανάκληση (Recall), υποδηλώνει την ικανότητά του μοντέλου να ανιχνεύει τα αρνητικά δείγματα και, τέλος, το F1, συνδυάζει την ακρίβεια και την ανάκληση και είναι μια καλή μετρική για την αξιολόγηση της απόδοσης του μοντέλου. Με μεταβλητές m=500, n=21, k=25720 και τα αποτελέσματα ακρίβειας, ανάκλησης και F1 δόθηκαν σχετικά με την κατηγορία των αρνητικών κριτικών, η χρήση του AdaBoost επέστρεψε τα εξής:

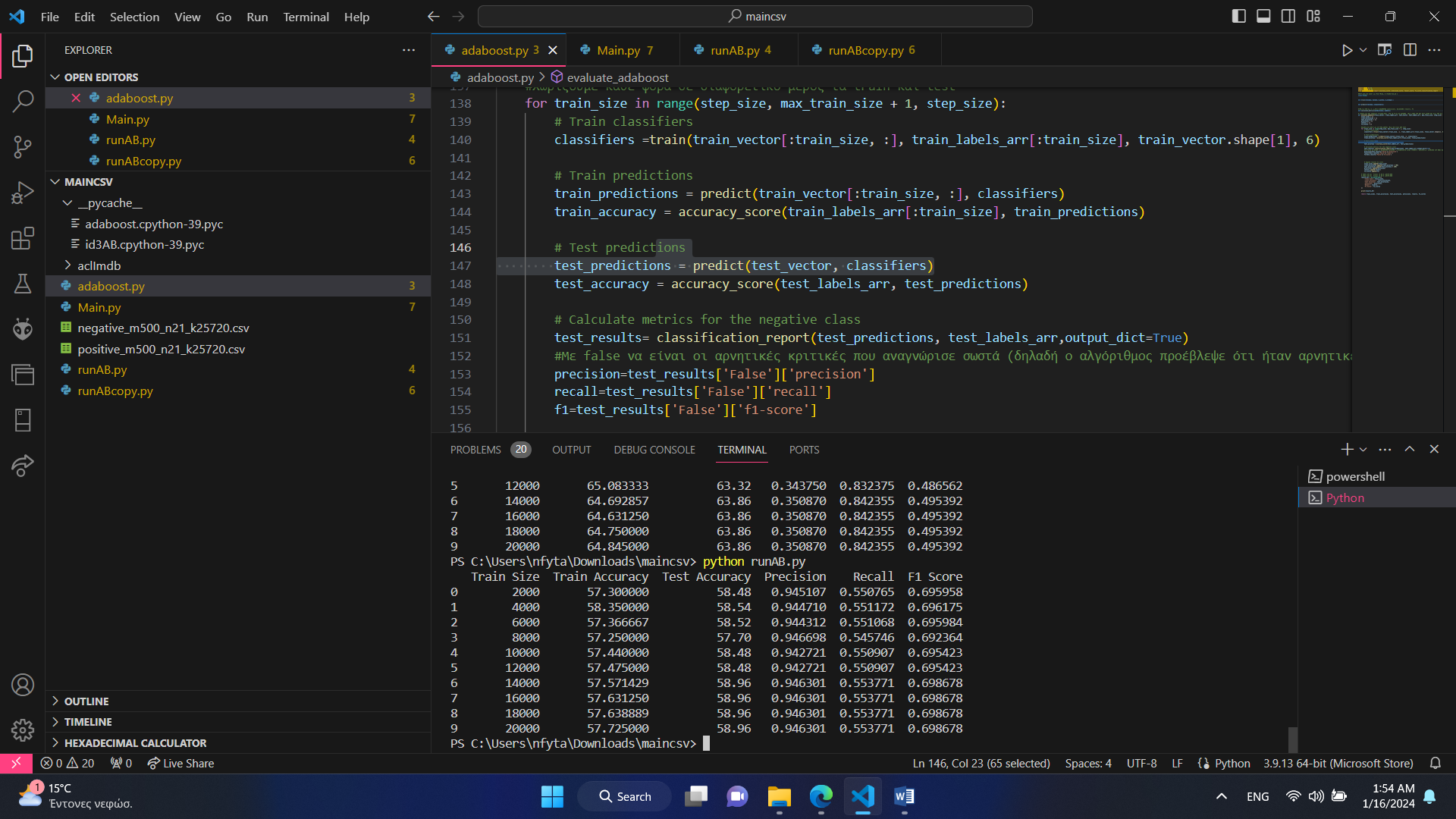
Καμπύλη μάθησης ποσοστών ορθότητας στα δεδομένα εκπαίδευσης και ελέγχου:

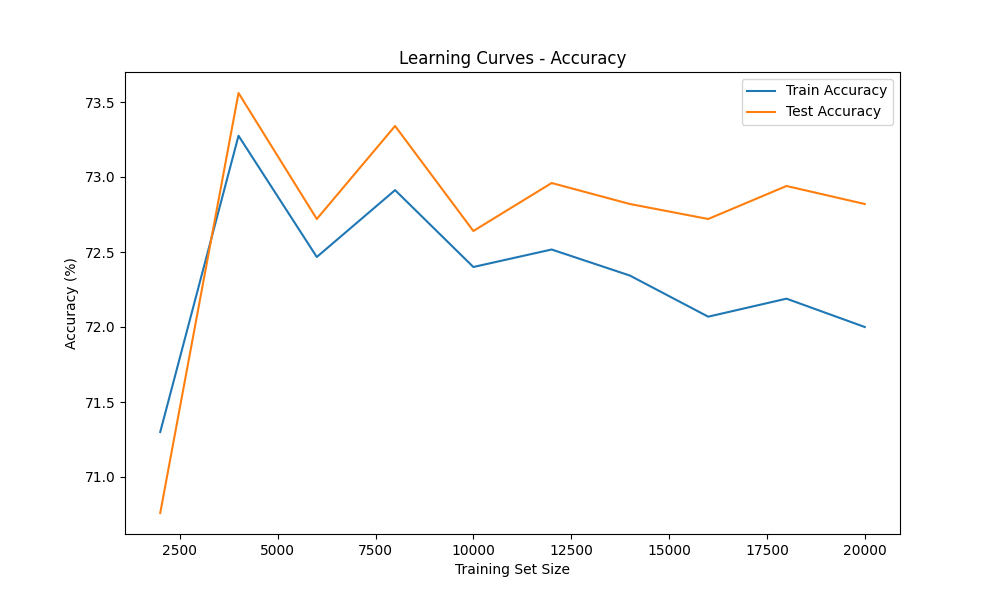
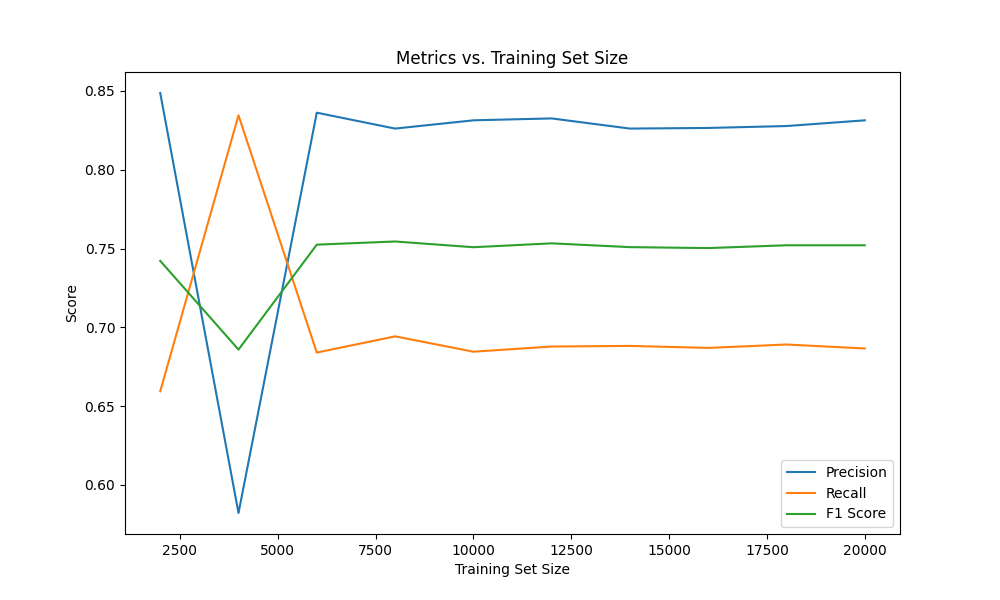


Καμπύλες ακρίβειας, ανάκλησης, F1 για την κατηγορία των αρνητικών κριτικών:

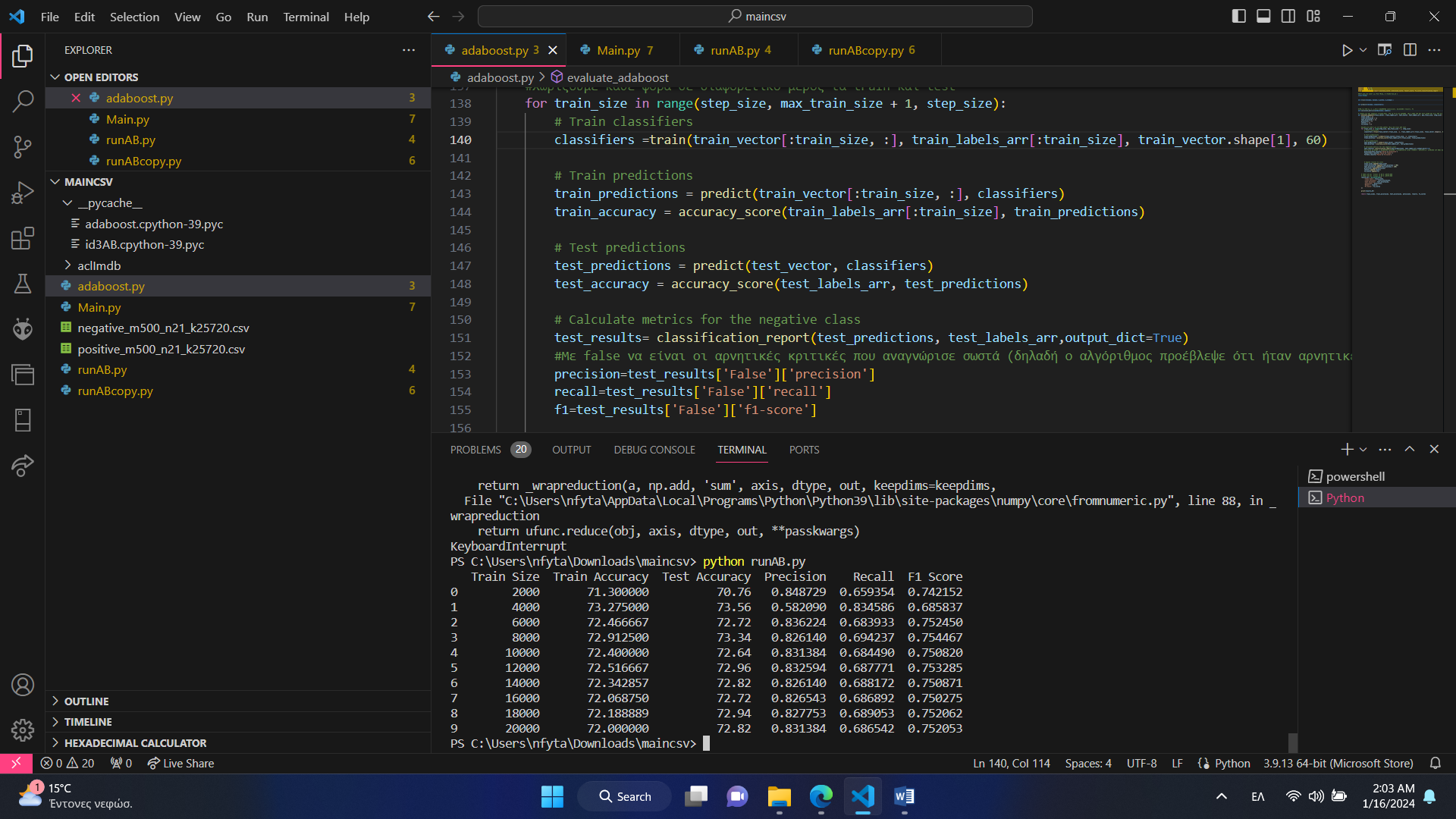


Και οι αντίστοιχες τιμές εκφρασμένες σε πίνακα ανά training size

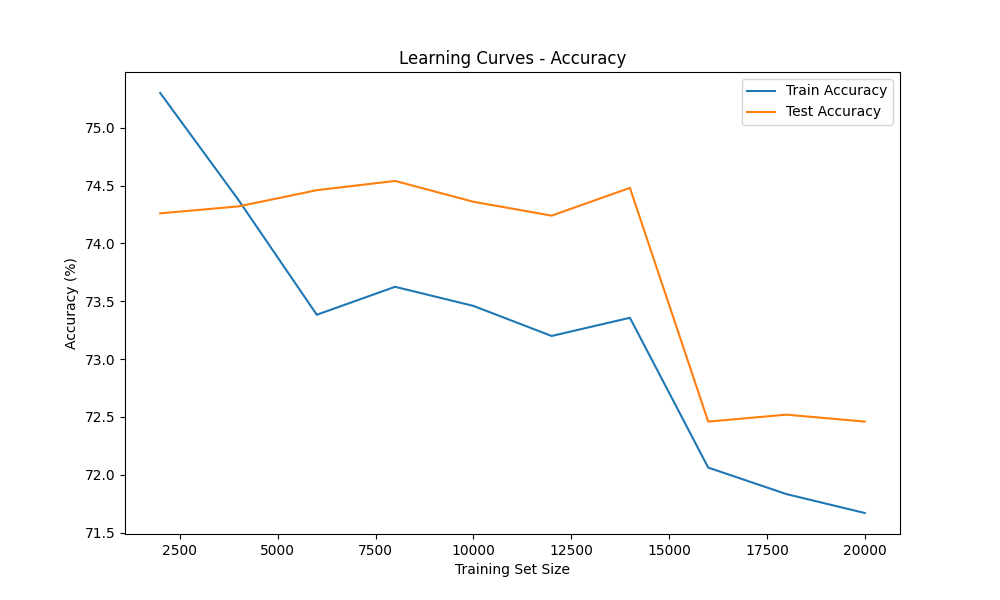
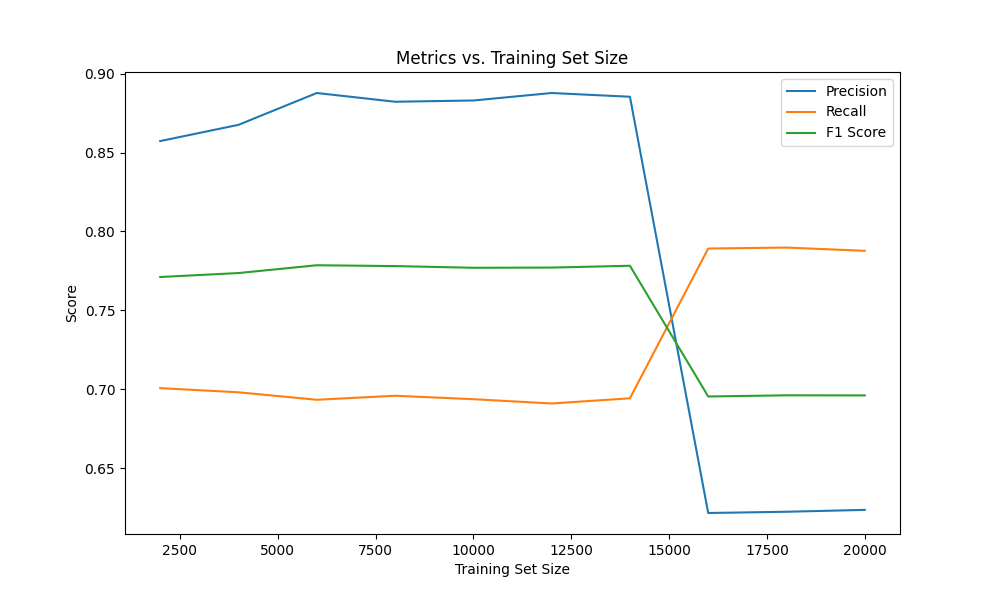


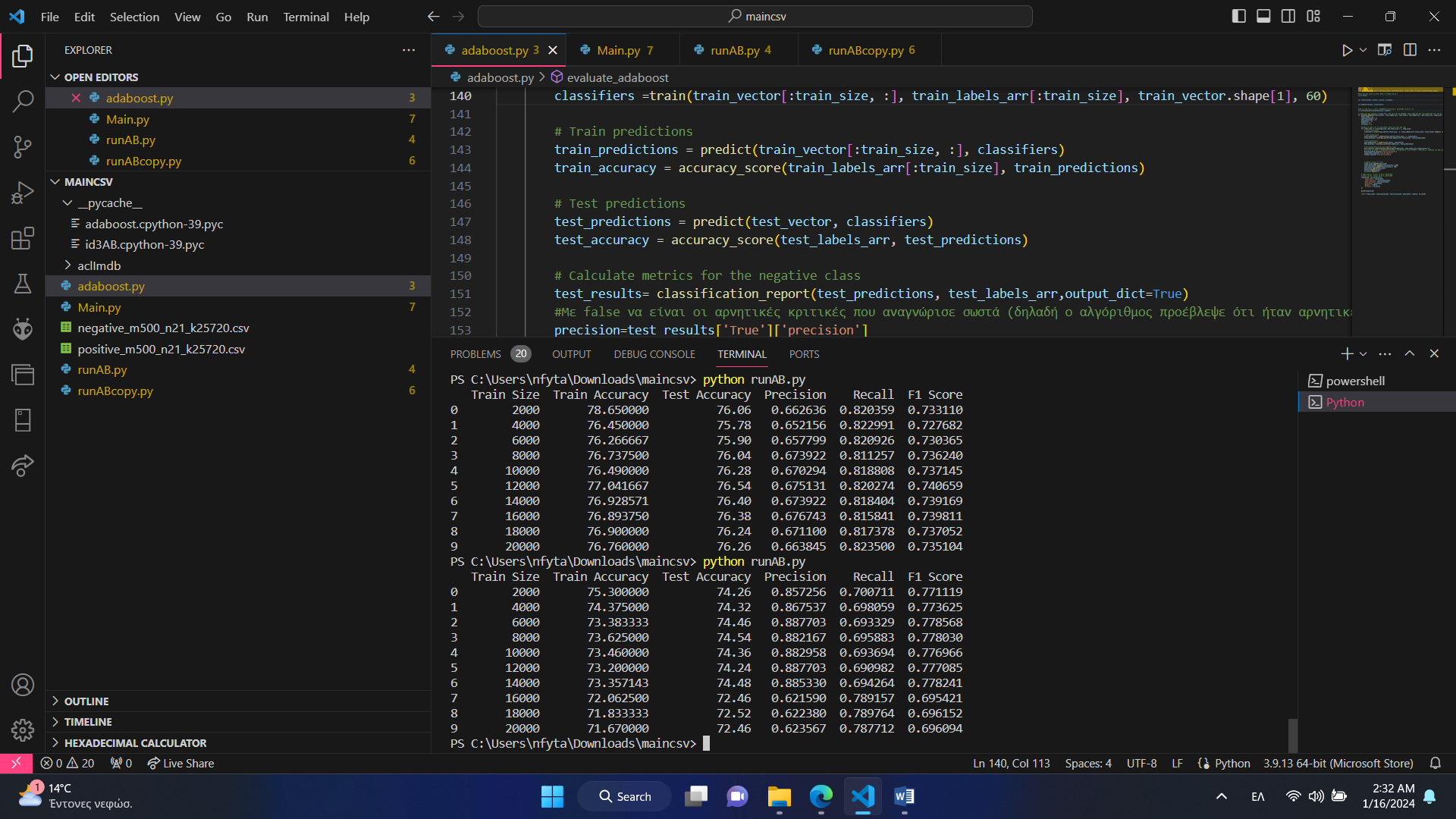
Τα παραπάνω έγιναν με δημιουργία 6 stump, αν τώρα κρατήσουμε όλες τις τιμές ίδιες και αλλάξουμε το πλήθος των stump από 6 σε 60 θα έχουμε, όπως είναι αναμενόμενο, μία σχετική αύξηση ορθότητας. Βέβαια αν παρατηρήσουμε σε σχέση με την προηγούμενη πληροφόρηση αυτή την φορά φαίνεται ότι με την αύξηση του μεγέθους των εκπαιδευτικών δειγμάτων που εξετάζονται η ορθότητα αρχίζει να μειώνεται, το οποίο σημαίνει ότι πρόκειται μετά από ένα σημείο να μην μπορεί να θεωρηθούν αξιόπιστα τα συμπεράσματα μας, αυτό σημαίνει ότι το μοντέλο ενδέχεται να αρχίσει να υπερεκπαιδεύεται όσο αυξάνονται τα δεδομένα εκπαίδευσης.:

Παρατηρούμε εδώ ότι ειδικότερα σε μικρότερα μεγέθη του συνόλου εκπαίδευσης. το μοντέλο δυσκολεύεται να αναγνωρίσει σωστά τα αρνητικά δείγματα.



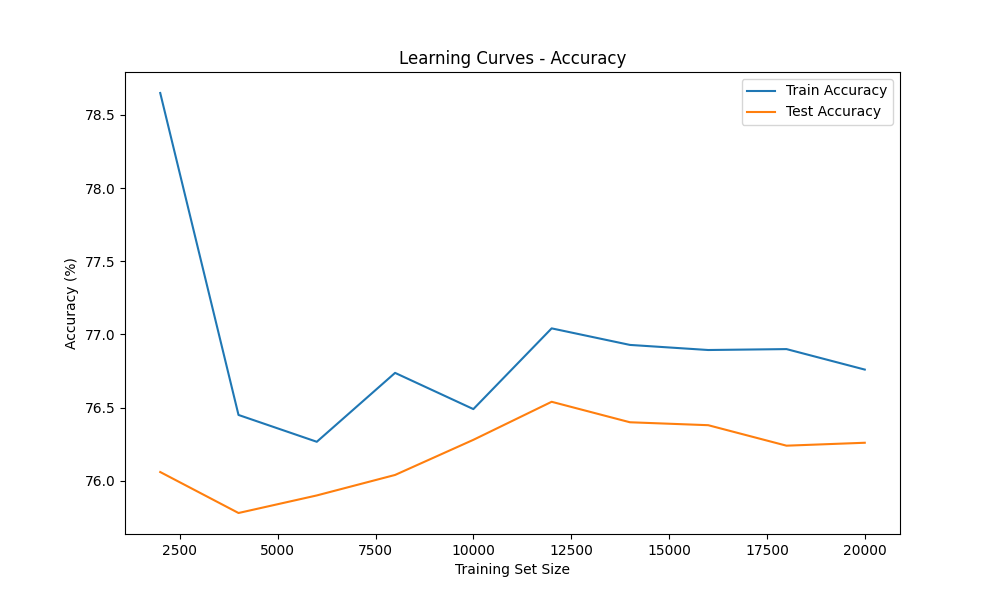
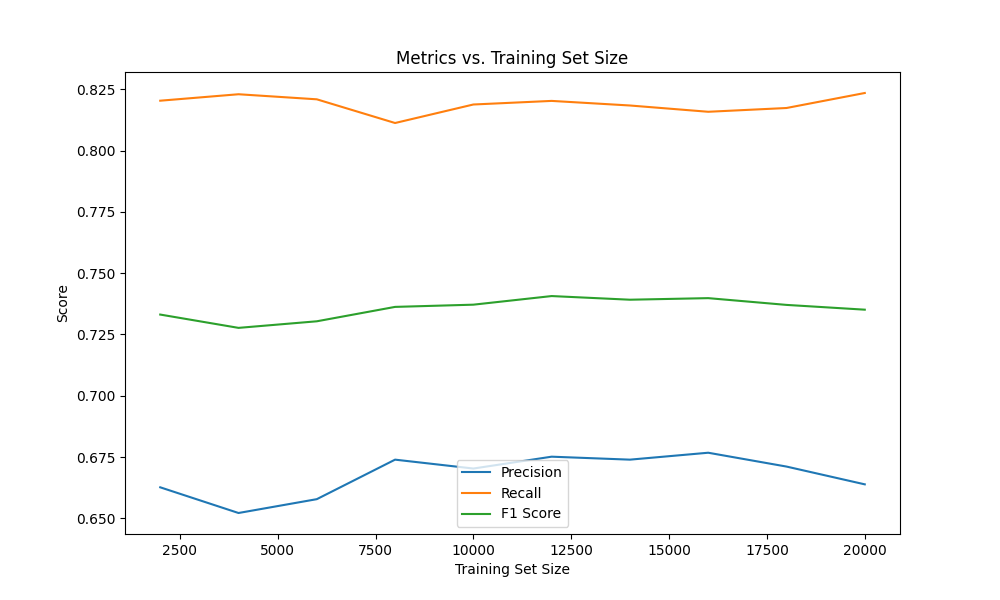
Λόγω της τόσο έντονης δυσκολίας που παρουσίασε ο αλγόριθμος για εντοπισμό αρνητικών κριτικών στα πρώτα παραδείγματα εκπαίδευσης, είναι χρησιμό να δούμε με τα ίδια stump αν θα υπάρχει το αντίθετο αποτέλεσμα με εξαγωγή των καμπυλών ακρίβειας, ανάκλησης και F1 σε βάση τις θετικές κριτικές.

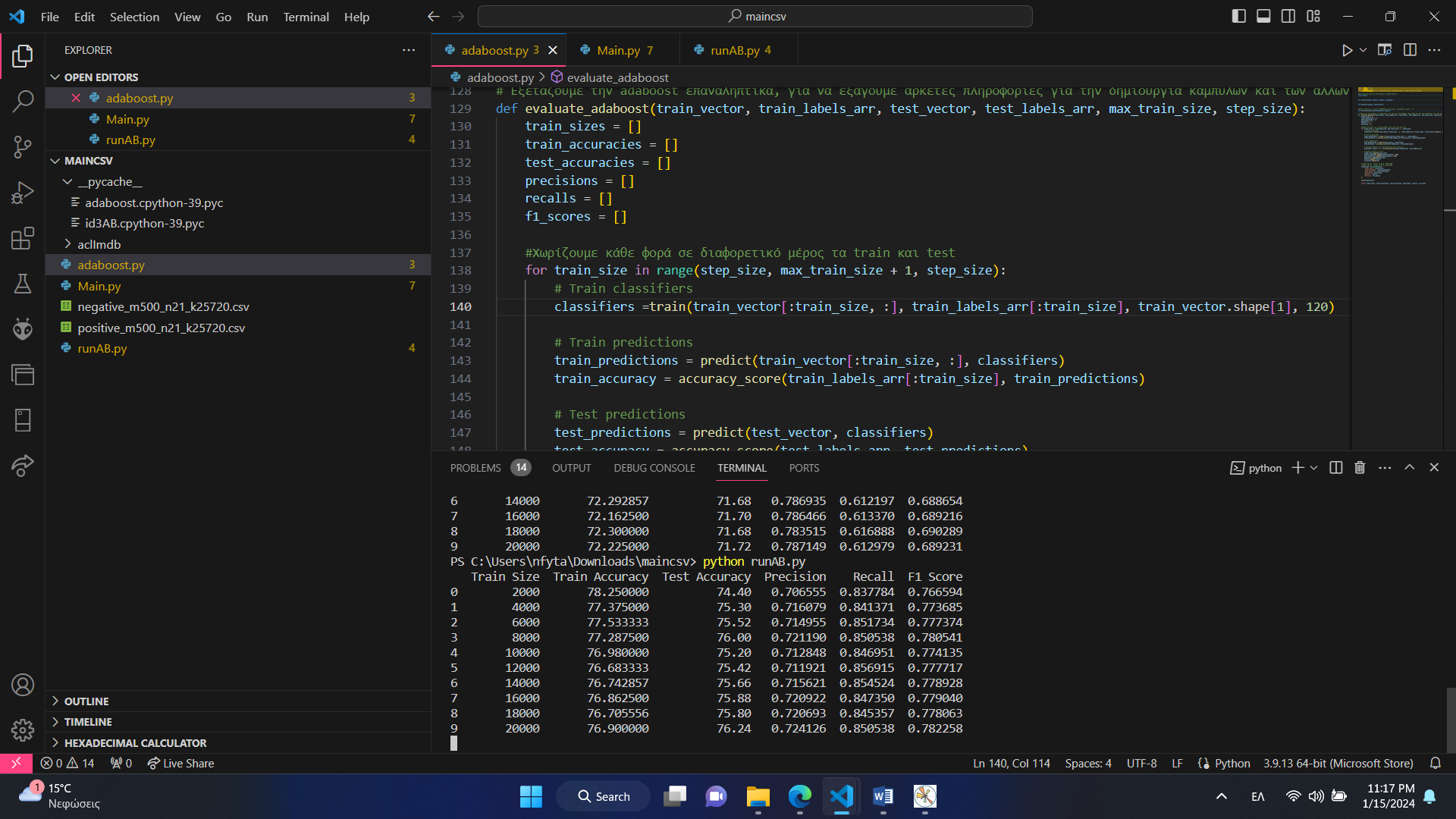




Παρατηρούμε, λοιπόν, αυτό που περιμέναμε ότι ο εντοπισμός στην αρχή των θετικών κριτικών ήταν σχετικά εύκολος (σταθερή/ σχετικά αυξητική ακρίβεια) ενώ εκεί που έχουμε μία σχετικά αυξητική ακρίβεια των αρνητικών κριτικών υπάρχει η αντίστοιχη μείωση των θετικών (πλήθος εκπαιδευτικών παραδειγμάτων 12550-). Επίσης, όσον αφορά την ορθότητα των εκπαιδευτικών δεδομένων και ελέγχου, διαφέρει αρκετά από το προηγούμενο παράδειγμα με βάση την πορεία σε σχέση με την αύξηση των παραδειγμάτων, αλλά η αναλογία μεταξύ των τιμών τους (η απόσταση των δύο καμπυλών) είναι αρκετή, το οποίο σημαίνει ότι έχουμε περιορισμένο overfitting (υπερβολική προσαρμογή στα δεδομένα εκπαίδευσης).

Παραθέτουμε άλλο ένα παράδειγμα εκτέλεσης του αλγορίθμου αυτή την φορά με χρήση 120 stump, παρατηρούμε ότι με την αύξηση των stump, προφανώς χρειάζεται πιο πολύς χρόνος για την εξαγωγή αποτελεσμάτων. Η αύξηση των stump στον συγκεκριμένο αλγόριθμο σημαίνει μεγαλύτερη ορθότητα, καθώς «καλύπτουμε» περισσότερες λέξεις. Στην δική μας περίπτωση βέβαια βλέπουμε ότι ενώ υπάρχει μία αύξηση στην ορθότητα δεν είναι τόσο δραματική όσο κανείς θα περίμενε σε σχέση με το μέγεθος της αύξησης των stump, ίσα ίσα παρατηρείται μελλοντική μείωση της ορθότητας, το οποίο δηλώνει υπερεκπαίδευση και δεν είναι ωφέλιμο για χρήση των αποτελεσμάτων για εξαγωγή συμπερασμάτων για άλλα δεδομένα.

Εδώ επανήλθε ένας αναλογικός εντοπισμός αρνητικών κριτικών.



Γενικότερα τα δεδομένα εκπαίδευσης ορθότητας, δίνουν το ποσοστό των σωστών προβλέψεων στα δεδομένα εκπαίδευσης. Εδώ βλέπουμε ότι η ακρίβεια (Precision) αυξάνεται καθώς αυξάνεται το μέγεθος του συνόλου εκπαίδευσης, προσαρμόζοντας το μοντέλο σε περισσότερα δεδομένα εκπαίδευσης. Όμοια πληροφορία εξάγουμε και από τις τιμές ορθότητας ( Test accuracy) των δεδομένων ελέγχου, στα οποία παρατηρούμε σε όλα τα παραδείγματα ότι η ορθότητα στα δεδομένα ελέγχου είναι κοντά στην ορθότητα στα δεδομένα εκπαίδευσης, προκαλώντας περιορισμένο overfitting (υπερβολική προσαρμογή στα δεδομένα εκπαίδευσης). Συγκρητικά μεταξύ των παραδειγμάτων (διαφορά πλήθους stump) βλέπουμε μία αύξηση στην ακρίβεια όπως και στην ορθότητα το οποίο είναι λογικό διότι η δημιουργία περισσότερων stump, όπως προαναφέραμε, σημαίνει μεγαλύτερη «κάλυψη» δεδομένων. Με τον όρο ανάκληση δίνεται το ποσοστό των αρνητικών κριτικών που κατάφερε να αναγνωρίσει σωστά μεταξύ των παραδειγμάτων, που αντιστοιχούσαν σε αρνητικές κριτικές, το οποίο πάλι σχετικά αυξάνεται με το μέγεθος του συνόλου εκπαίδευσης. Τέλος το F1, είναι ένας συνδυασμός της ακρίβειας και της ανάκλησης και χρησιμοποιείται όταν θέλουμε να εξισορροπήσουμε αυτές τις δύο μετρικές. Βλέπουμε με την αύξηση των stump ότι μας δίνεται μία σχετικά μεγάλη αύξηση στην ορθότητα ενώ στα άλλα στοιχεία που εξετάζουμε η αύξηση δεν είναι πάντα δεδομένη, λόγω του ότι εξετάζουμε συγκεκριμένα για μί ακατηγορία και έχει σχέση με το σετ δεδομένων που κάθε φορά επεξεργαζόμαστε.

Random Forest

Η υλοποίηση του αλγορίθμου γίνεται στο αρχείο RandomForest.py και χρησιμοποιεί και τον κώδικα του φροντιστηρίου που βρίσκεται στο αρχείο id3.py.

Στο αρχείο RandomForest.py περιέχεται η κλάση RandomForest και περιέχει τις συναρτήσεις που βοηθούν στην δημιουργία και στον έλεγχο του μοντέλου και είναι οι εξής:

* \_\_init\_\_(self,n\_trees,features): Κατασκευάζει ένα αντικείμενο της κλάσης αρχικοποιώντας τον αριθμό των δέντρων που θα κατασκευαστούν (n\_trees),την λίστα των ιδιοτήτων που θα χρησιμοποιηθούν στην κατηγοριοποίηση των παραδειγμάτων (features) και την λίστα που θα περιέχει κάθε δέντρο (forest)

*Οι 3 ακόλουθες συναρτήσεις δημιουργήθηκαν για να μπορέσει να κατασκευαστεί η καμπύλη μάθησης.*

* get\_params(self,deep=True):Επιστρέφει ένα λεξικό με τις παραμέτρους της συνάρτησης \_\_init\_\_.
* set\_params(self,\*\*parameters): Θέτει τις παραμέτρους του εκτιμητή (στην περίπτωση μας του random forest αντικειμένου) σύμφωνα με την λίστα παραμέτρων που δέχεται ως όρισμα.
* score(self,y\_true,y\_pred): Αξιολογεί τον αλγόριθμο χρησιμοποιώντας την συνάρτηση f1\_score που υπολογίζει το f1 των δεδομένων ελέγχου.
* fit(self,x\_Train,y\_Train): Kατασκευάζει με την βοήθεια του αλγορίθμου ID3 ένα πλήθος από τυχαιοποιημένα δέντρα απόφασης ,καθένα από αυτά εκπαιδεύεται πάνω σε ένα τυχαίο υποσύνολο των δεδομένων εκπαίδευσης λαμβάνοντας υπόψη ένα τυχαίο υποσύνολο ιδιοτήτων, βάση των οποίων μαθαίνει να κατηγοριοποιεί τα παραδείγματα.
* predict(self,X): Δίνουμε καινούργια παραδείγματα ελέγχου(Χ) στα δέντρα που εκπαιδεύσαμε με την συνάρτηση fit έτσι ώστε να προβλέψουν σε ποια κατηγορία ανήκουν τα παραδείγματα αυτά .Η πρόβλεψη γίνεται με την βοήθεια της αντίστοιχης συνάρτησης του ID3 (predict)και κάθε δέντρο θα έχει διαφορετική απάντηση για κάθε παράδειγμα ,όμως η τελική πρόβλεψη λαμβάνει υπόψη της την απάντηση της πλειοψηφίας των δέντρων ,έτσι αν τα περισσότερα απάντησαν θετικά το παράδειγμα κατατάσσεται στην θετική κατηγορία ,αλλιώς στην αρνητική και σε περίπτωση που δεν υπερισχύει καμία κατηγορία το παράδειγμα κατατάσσεται με 50% πιθανότητα σε οποιαδήποτε από τις δύο κατηγορίες.
* RandomForest\_Train(self,x\_train,y\_train,number\_of\_trees): Είναι η συνάρτηση που καλείται από το ‘κυρίως’ πρόγραμμα για την δημιουργία ενός μοντέλου random forest . Αρχικά διαλέγουμε τυχαία ένα υποσύνολο δεδομένων εκπαίδευσης και ύστερα το χωρίζουμε σε δεδομένα εκπαίδευσης και ελέγχου ,ύστερα δημιουργώντας ένα αντικείμενο της κλάσης καλούμε την συνάρτηση fit για να εκπαιδεύσει το μοντέλο και έπειτα την συνάρτηση predict για να δούμε το ποσοστό ακρίβειας της πρόβλεψης του σε νέα δεδομένα(x\_test) .Τέλος όπως θα δείτε και παρακάτω δημιουργούμε το classification report για την θετική κατηγορία (1) και επιστρέφουμε στο κυρίως πρόγραμμα διάφορες μετρικές για τον σχηματισμό των καμπυλών.

Λόγω χαμηλής υπολογιστικής ισχύος σε κάθε πείραμα χρησιμοποιούμε 10 δέντρα, με λιγότερα από 10 δέντρα μειώνεται η ακρίβεια των προβλέψεων ,σύμφωνα με πολλαπλά πειράματα που εκτελέσαμε. Να σημειωθεί επίσης ότι από το αρχικό πλήθος των ιδιοτήτων που ισούται με 500 χρησιμοποιείται ένα δείγμα του οποίου το μέγεθος κυμαίνεται από 50 εώς 60 για την εκπαίδευση κάθε δέντρου, για να μειωθούν τα σφάλματα εκπαίδευσης και αξιολόγησης. Επίσης αν χρησιμοποιούσαμε λιγότερες ιδιότητες δεν θα γινόταν το ίδιο καλά η κατηγοριοποίηση γιατί θα είχαμε overfitting. Στα παρακάτω screenshots τα recall,precision,f1-score αφορούν την θετική κατηγορία(1). Στο δεύτερο πείραμα που το precision και το recall είναι ίσα με το μηδέν καταλαβαίνουμε ότι το δείγμα είχε κατά συντριπτική πλειοψηφία περισσότερες αρνητικές κριτικές και εξού και το μεγάλο ποσοστό ακρίβειας και των δύο κατηγοριών δεδομένων ,εφόσον κατατάχθηκαν λογικά όλες οι κριτικές στις αρνητικές.

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, γραμματοσειρά, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμματοσειρά, κείμενο, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμματοσειρά, γραμμή, κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα



Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμματοσειρά, κείμενο, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα





Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμματοσειρά, γραμμή, κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα





Εικόνα που περιέχει διάγραμμα, γραμμή, γράφημα, κείμενο

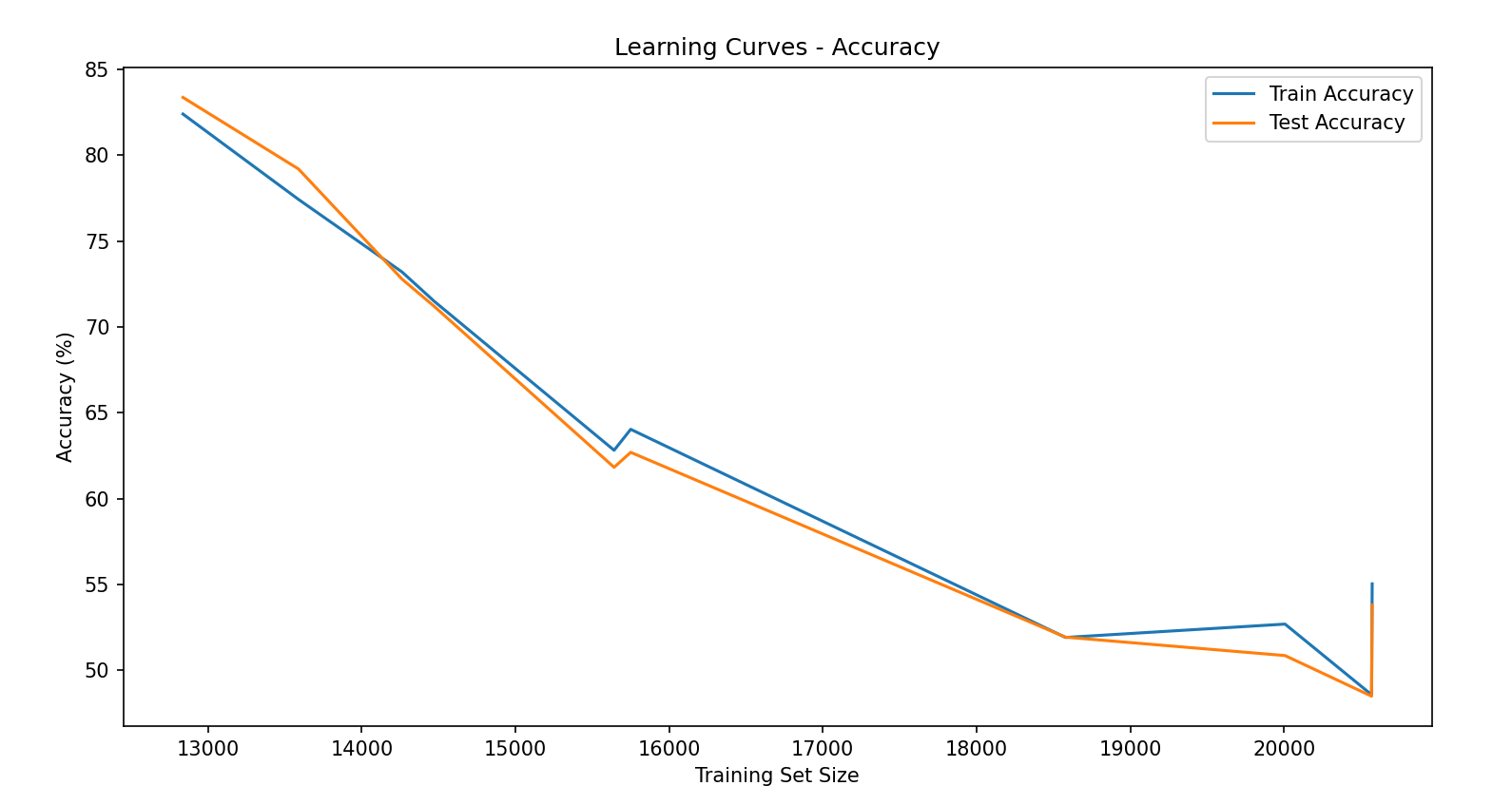
Περιγραφή που δημιουργήθηκε αυτόματαΕδώ τα ίδια δεδομένα φαίνονται σχηματικά:

Αυτή είναι η καμπύλη μάθησης που είχαμε παράξει με διαφορετικά δεδομένα από τα από πάνω. Τα δεδομένα αξιολογήθηκαν με βάση το f1-score:

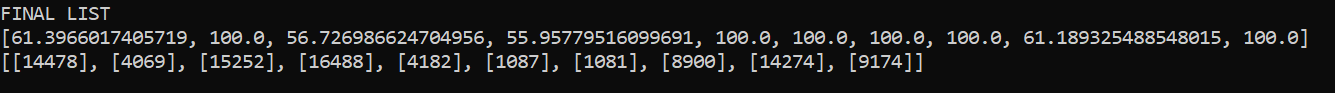
Εικόνα που περιέχει κείμενο, διάγραμμα, στιγμιότυπο οθόνης, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Στο παρακάτω διάγραμμα φαίνεται πως διαμορφώνεται το ποσοστό ακριβείας των προβλέψεων καθώς αυξάνεται το μέγεθος των δεδομένων εκπαίδευσης:



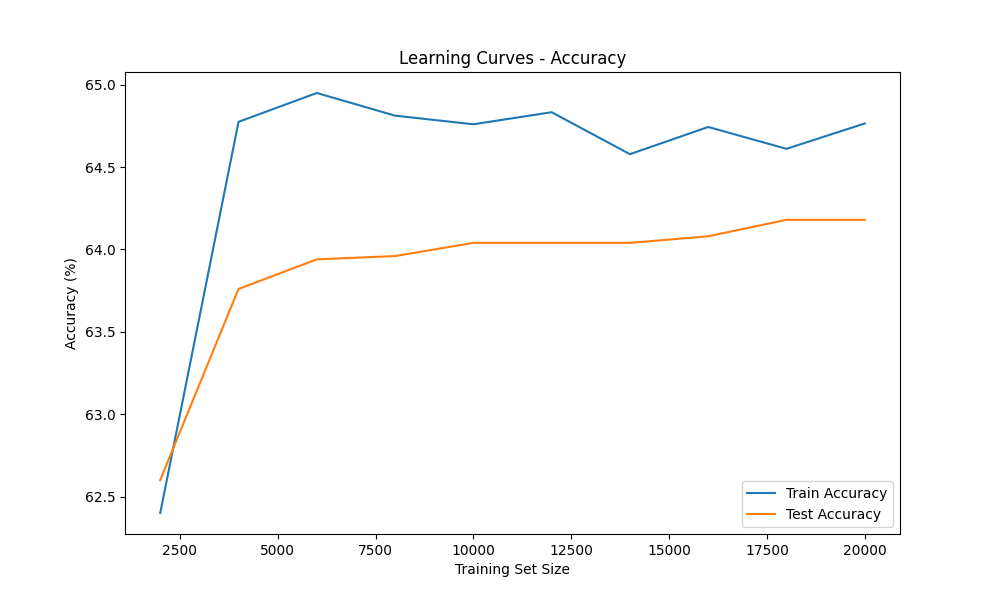
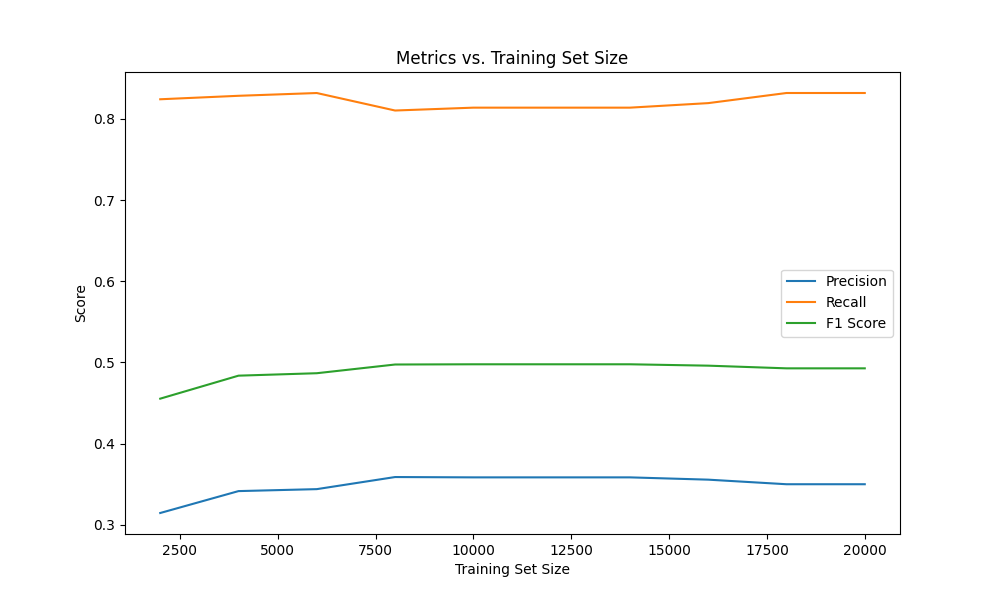
Όπως φαίνεται και στο παρακάτω screenshot η πρώτη λίστα περιλαμβάνει τα ποσοστά ορθότητας κάθε πειράματος για τα test data και η δεύτερη το πλήθος των δεδομένων εκπαίδευσης που χρησιμοποιήθηκαν σε κάθε πείραμα με σταθερό πλήθος δέντρων(=10) (αυτά τα δεδομένα είναι διαφορετικά από αυτά που χρησιμοποιήθηκαν για τον σχεδιασμό των καμπυλών). Σε όλα τα δείγματα που το πλήθος των δεδομένων εκπαίδευσης είναι μικρότερο των 10.000 έχουμε ποσοστό ορθότητας 100% μάλλον λόγω overfitting,οπότε για αυτό ορίσαμε το δείγμα των δεδομένων εκπαίδευσης να είναι πάνω από 10.000 .

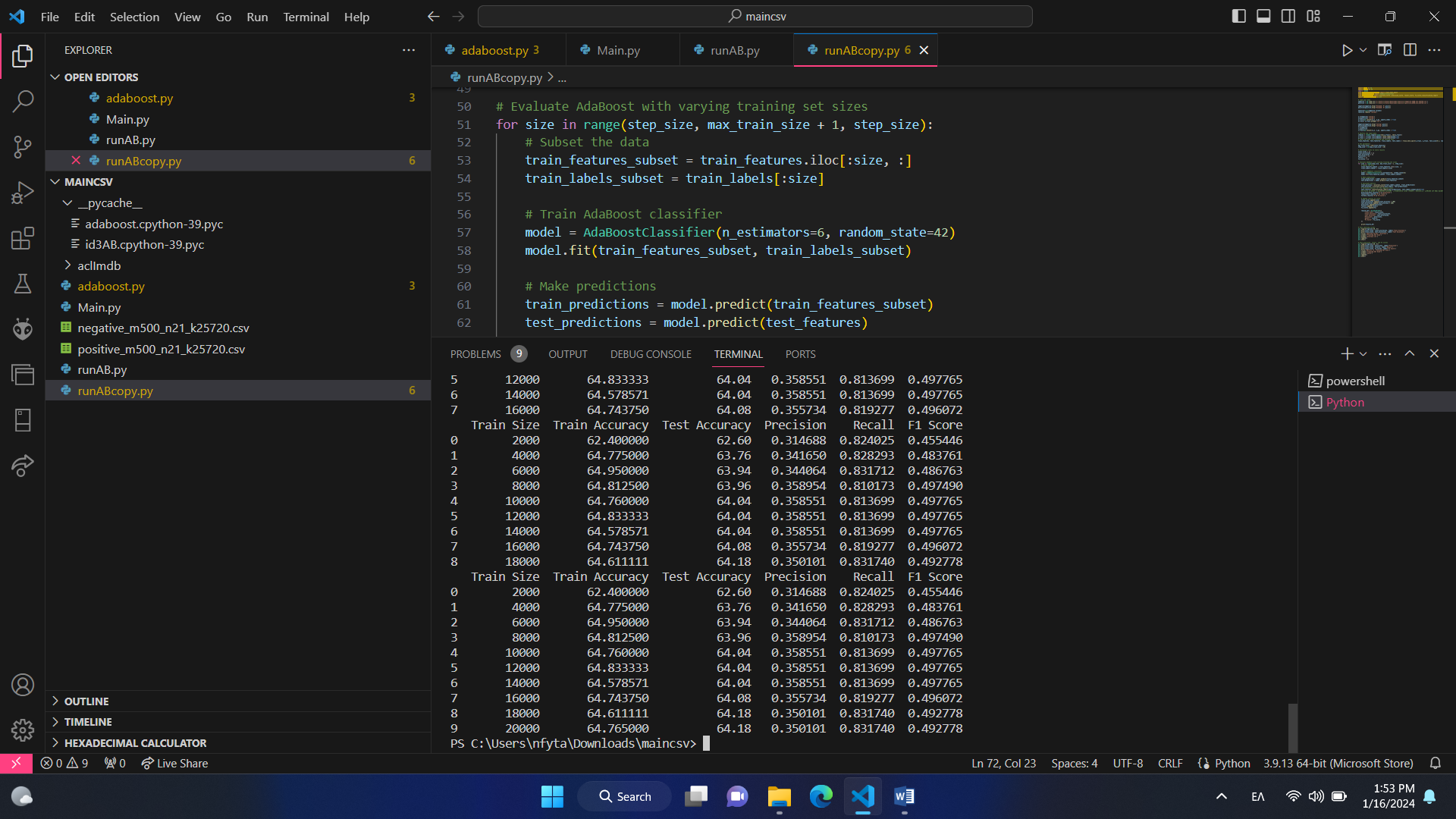


Μέρος Β’

AdaBoost

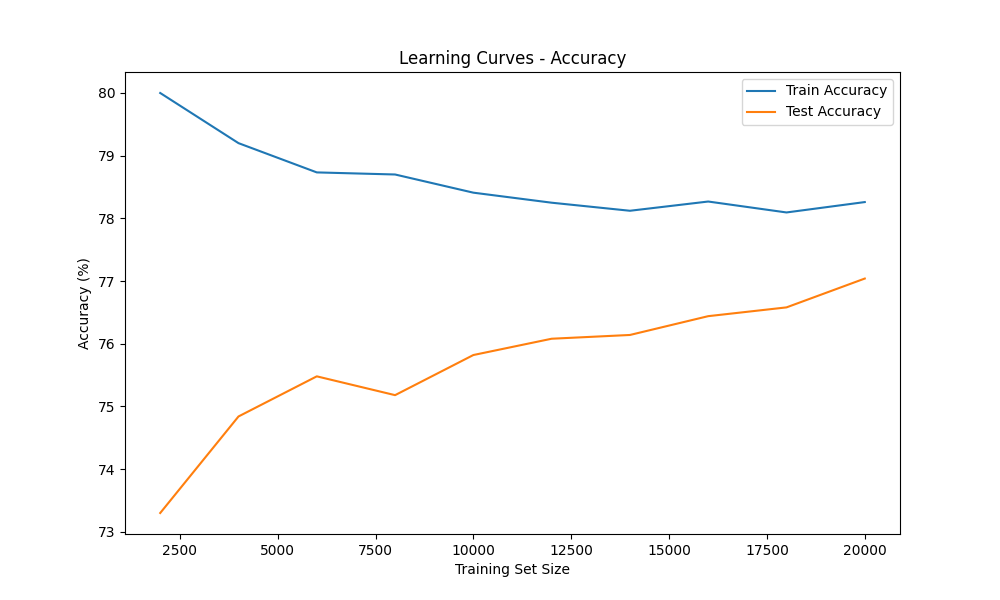
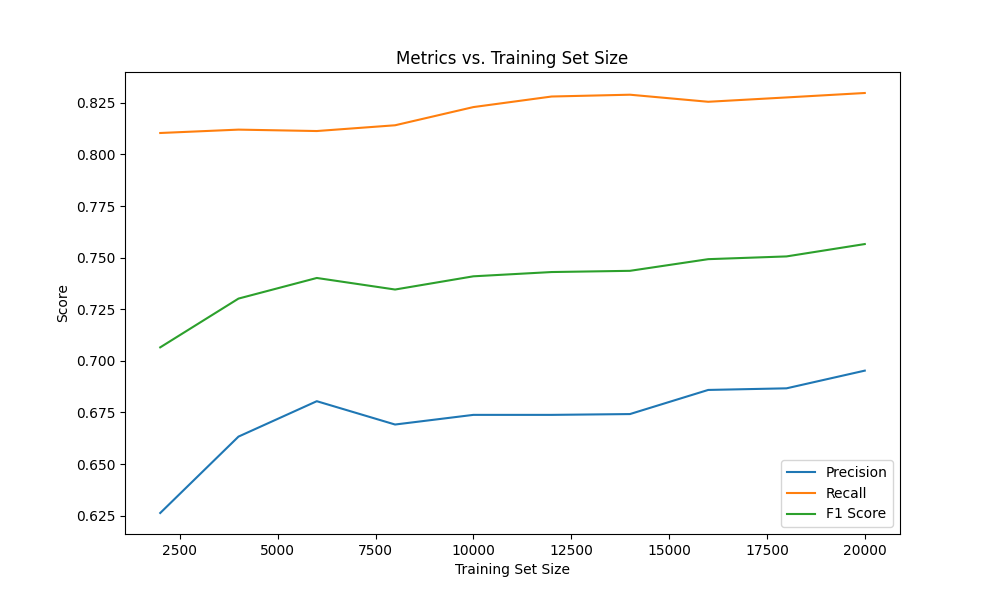
Χρησιμοποιώντας τις ίδιες υπερπαραμέτρους με το Μέρος Α δηλαδή για m=500, n=21, k=25720, θα εκλάβουμε τα ίδια δεδομένα που επεξεργαστήκαμε και προηγουμένως αυτή την φορά όμως θα χρησιμοποιήσουμε τον αλγόριθμο AdaBoost που παρέχεται από το Scikit-learn (from sklearn.ensemble import AdaBoostClassifier) και ουσιαστικά έχουμε το ίδιο κώδικα, δηλαδή και σε αυτή την περίπτωση θα γίνεται η κλήση του αλγορίθμου όσο είναι και το μέγεθος των δεδομένων (δηλαδή κρατάμε τα ίδια μεγέθη max\_train\_size, step\_size), ώστε να μπορούμε να κάνουμε ακριβή σύγκριση με τα πάνω δεδομένα (λόγω του random τρόπου διαχωρισμού των train-test data μπορούμε να εξάγουμε γενικότερα συμπεράσματα σύγκρισης, δηλαδή δεν έχει νόημα να γίνει αριθμητική σύγκριση). Επίσης κρατάμε την ίδια κατηγορία, των αρνητικών κριτικών, στα μεγέθη ακρίβειας, ανάκλησης και F1 για να γίνει η σύγκριση. Ξεκινώντας πάλι με stump = 6 μας δίνονται τα εξής αποτελέσματα:

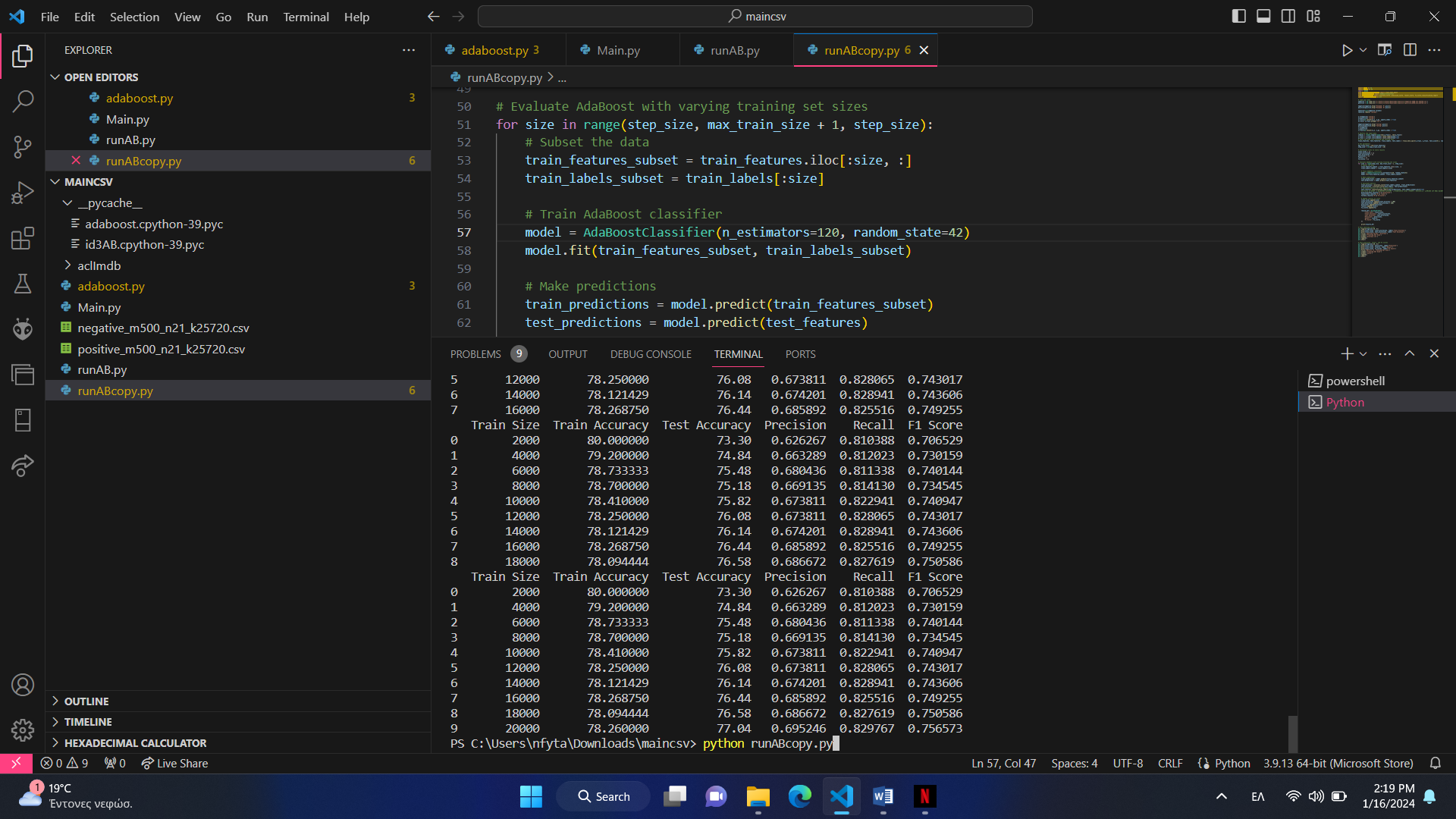




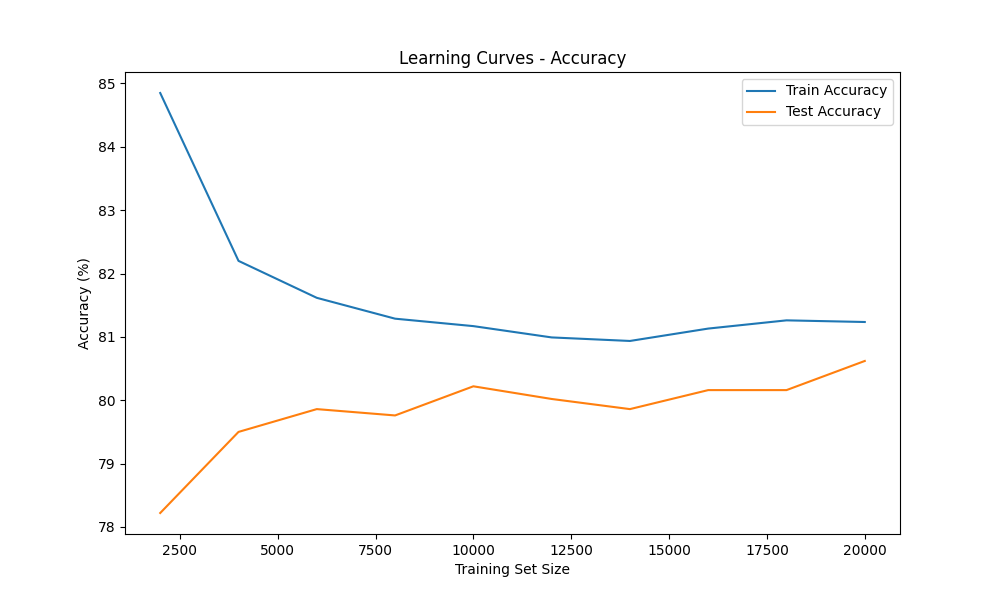
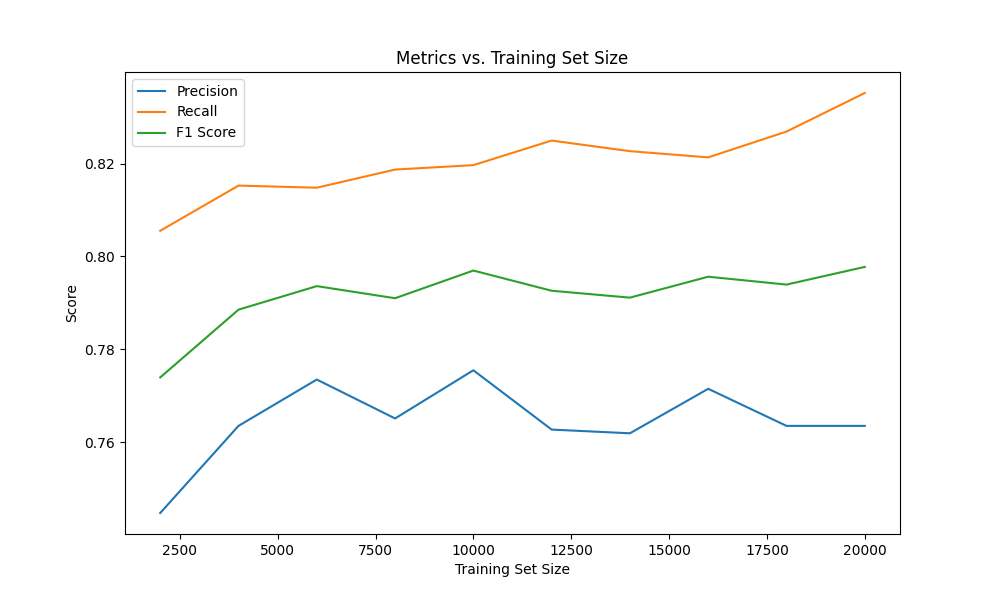
Αρχικά ο χρόνος παραγωγής των αποτελεσμάτων ήταν αρκετά πιο γρήγορος, επίσης, σε σχέση με την δικιά μας υλοποίηση του αλγορίθμου, σε αυτή την περίπτωση έχουμε πολύ πιο υψηλή ορθότητα. Βέβαια βλέπουμε ότι έχουμε μία σχετική ελάττωση ακρίβειας και του F1, αλλά μεγάλη αύξηση των μεγεθών ανάκλησης, αυτό φαίνεται οξύμωρο, λόγω της χαμηλής τιμής της ακρίβειας. Χαμηλή τιμή στην ακρίβεια έχουμε όταν, υπάρχει μεγάλη ανισορροπία στις κατηγορίες, δηλαδή μπορεί να έχουμε εντοπίσει λιγότερες αρνητικές κριτικές, το οποίο δεν είναι αυτό που ευθύνεται, στην δική μας περίπτωση, καθώς εξετάζουμε το ίδιο σετ δεδομένων. Μπορεί, διαφορετικά, να οφείλεται σε σφάλματα που προέκυψαν και πως ο κάθε αλγόριθμος τα διαχειρίζεται, εξ’ορισμού και οι δύο αλγόριθμοι χρησιμοποιούν τον ίδιο τρόπο υπολογισμού σφάλματος, όμως το σφάλμα βασίζεται στην αξιολόγηση των αποτελεσμάτων που προέρχονται από τα τυχαία δένδρα απόφασης, τα οποία, με την σειρά τους, βασίζονται στα βάρη που έχει η κάθε λέξη. Επομένως, καταλήγουμε, ότι η μεγάλη διαφορά στις τιμές των 3 τελευταίων μεγεθών είναι πολύ πιθανό να οφείλεται σε διαφορετική διαχείριση των βαρών των λέξεων.

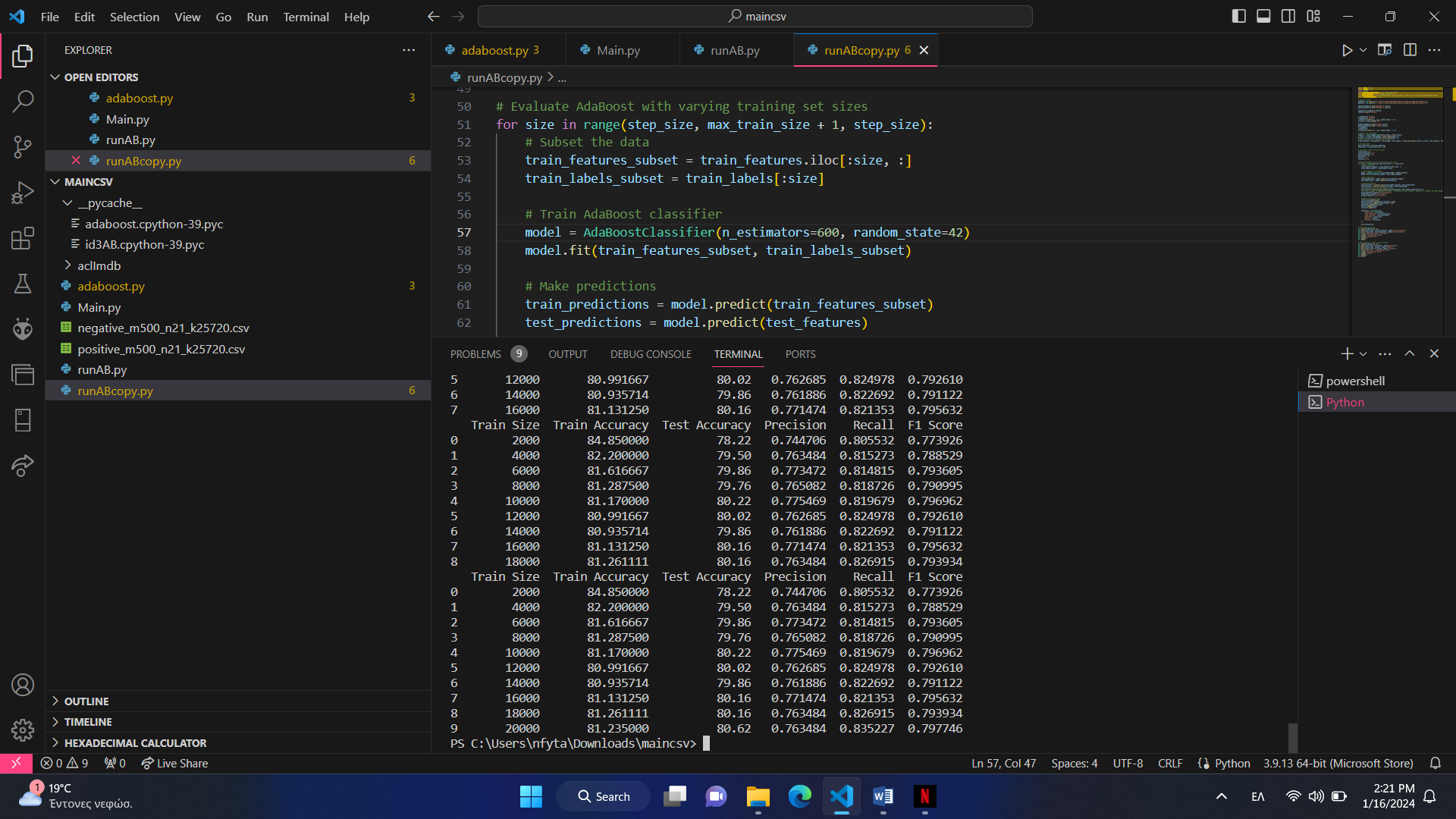
Με την αλλαγή των stump από 6 σε 60 βλέπουμε μία μεγάλη αύξηση στις τιμές όλων των μεγεθών, όχι μόνο συγκριτικά με τον δικό μας αλγόριθμο αλλά και με τον ίδιο (με διαφορετικά stump). Βλέπουμε ότι έχουμε φτάσει την ορθότητα και γενικότερα οι τιμές όλων των μεγεθών είναι πολύ κοντινές με αυτές που είχαμε στον δικό μας αλγόριθμο στα διπλάσια stump (120), το οποίο δείχνει πόσο πιο επιτυχημένη είναι η υλοποίηση του αλγορίθμου από το Scikit-learn σε σχέση με την δική μας. Γενικότερα, παρατηρούμε ότι με την αύξηση του μεγέθους του συνόλου εκπαίδευσης και των stump, και σε αυτήν την υλοποίηση και την δική μας, οι τιμές ορθότητας των δεδομένων εκπαίδευσης και ελέγχου δεν έχουν τόση μεγάλη απόσταση, το οποίο σημαίνει ότι το μοντέλο μας γενικεύει καλά σε νέα, μη-εκπαιδευτικά δεδομένα, δηλαδή έχουμε μία σχετικά καλή γενίκευση, δηλαδή είναι ικανό να αντιμετωπίσει νέα δεδομένα εκτός από αυτά που χρησιμοποιήθηκαν για την εκπαίδευσή του.

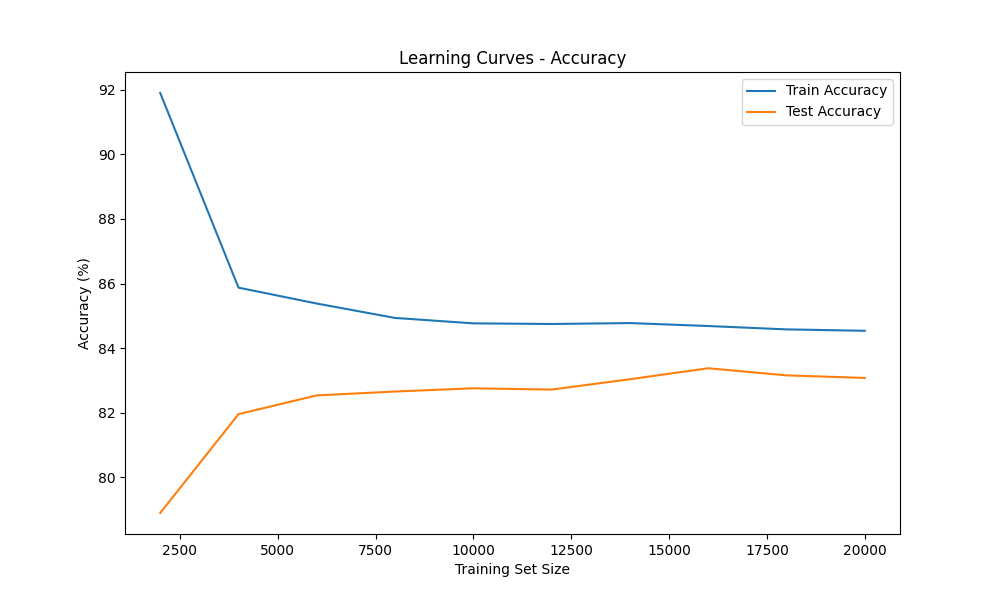
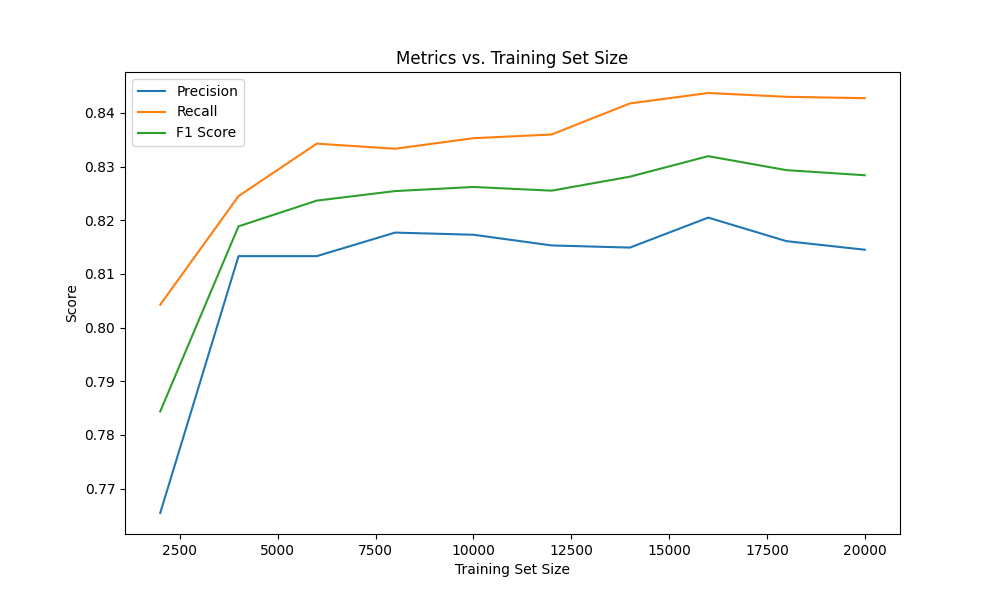


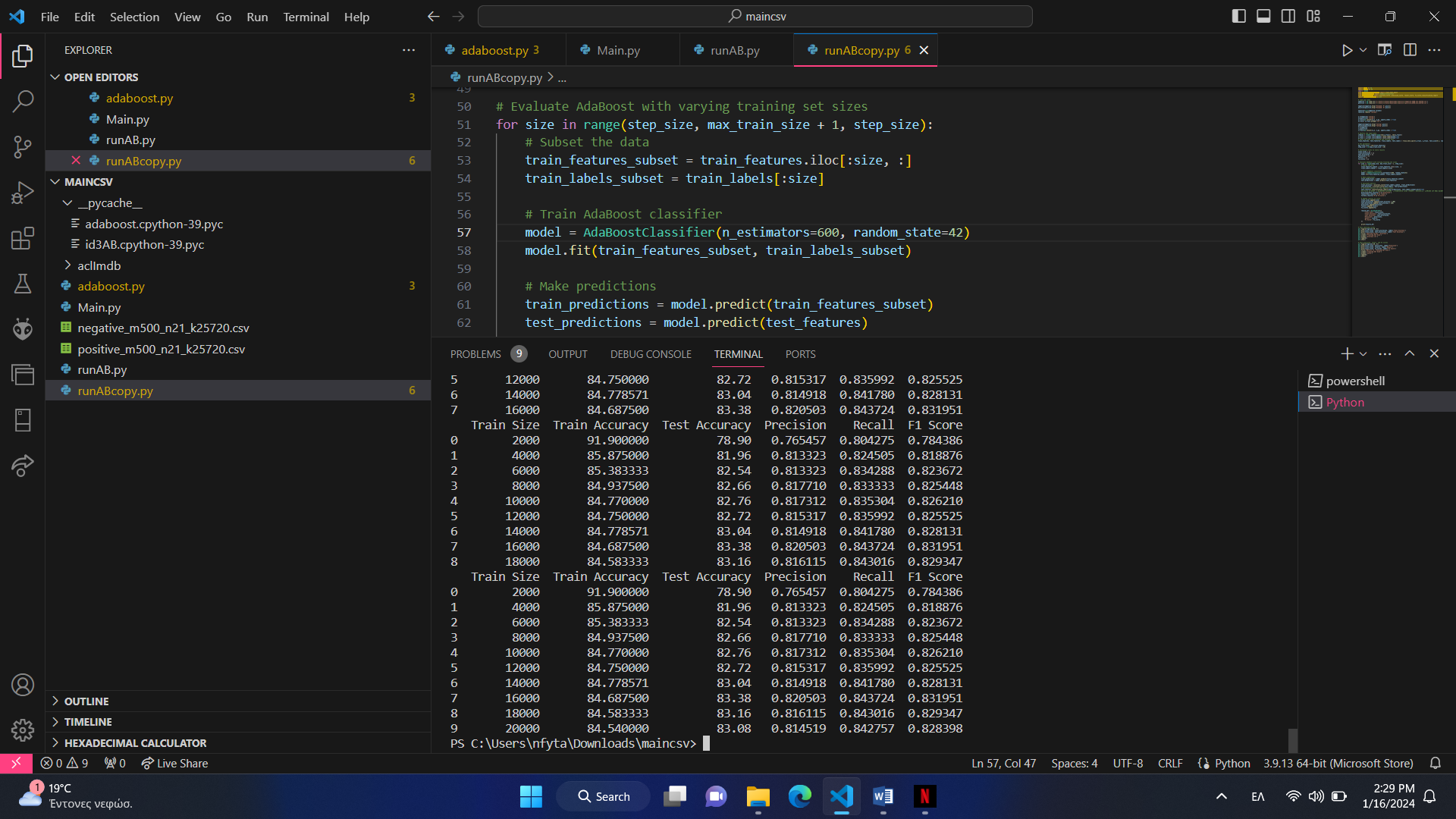


Στην συγκεκριμένη περίπτωση βλέπουμε κιόλας ότι με την αύξηση του μεγέθους των παραδειγμάτων εκπαίδευσης ενώ παρατηρείται μία σχετική μείωση στις τιμές ορθότητας των δεδομένων εκπαίδευση-ελέγχου αρχικά, αν μεγαλώσουμε κι’ άλλο το μέγεθος των παραδειγμάτων προσδοκούμε μία αύξηση των τιμών ορθότητας, ενώ επιπλέον είναι φανερή μία σύγκλιση των δύο μεγεθών στην τιμή ορθότητας 78, το οποίο σημαίνει ότι ο συγκεκριμένος αλγόριθμος δίνει αρκετά έγκυρα αποτελέσματα. Επίσης, βλέπουμε ότι πρόκειται για συνεχής αύξηση των τιμών ακρίβειας, ανάκλησης και F1 το οποίο σημαίνει ότι ο συγκεκριμένος αλγόριθμος δεν είχε κάποια δυσκολία στην ανίχνευση, συγκεκριμένα, αρνητικών κριτικών. Σε αντίθεση, δηλαδή με τον δικό μας αλγόριθμο που στα 60 stump αντιμετωπίστηκε μία δυσκολία ανίχνευσής τους, το οποίο έριξε, όπως είναι λογικό, τις συγκεκριμένες τιμές.

Παραθέτουμε επίσης την εκτέλεση του συγκεκριμένου αλγορίθμου για πλήθος stump=120, παρατηρούμε ότι δεν υπάρχει καμία χρονική σύγκριση, για άλλη μία φορά, έκβασης των αποτελεσμάτων σε σχέση με τον δικό μας αλγόριθμο.



Βλέπουμε ότι όλα τα εξεταζόμενα μεγέθη έχουν μεγάλη αύξηση, το οποίο είναι πολύ λογικό αν λάβουμε υπόψη την λειτουργία του αλγορίθμου όπως την εκφράσαμε θεωρητικά. Για άλλη μία φορά βλέπουμε μία πιο ξεκάθαρη σύγκλιση των τιμών των δεδομένων εκπαίδευσης-ελέγχου κατά την αύξηση των μεγεθών παραδειγμάτων εκπαίδευσης που εξετάζονται, το οποίο σημαίνει ότι ο συγκεκριμένος αλγόριθμος δίνει μία αρκετά καλή γενίκευση του μοντέλου, το οποίο σημαίνει ότι μπορεί να εφαρμοστεί επιτυχώς σε νέα δεδομένα.Τέλος, καθώς βλέπουμε ότι ο χρόνος εκτέλεσης του συγκεκριμένου αλγορίθμου μας επιτρέπει την εκτέλεση του για πολύ μεγάλα stump, παραθέτουμε και τα αποτελέσματα που δίνονται με την εκτέλεση του σε πλήθος stump = 600.



Βλέπουμε, ότι ενώ υπήρχε τεράστια αύξηση στο πλήθος των stump που χρησιμοποιούνται, άρα στο μέγεθος εκπαίδευσης με τα δεδομένα που μας δίνονται, έχουμε μία αύξηση όλων των μεγεθών αλλά όχι όσο δραματική θα περιμέναμε, αυτό οφείλεται και στο ότι μετά από ένα σημείο η πολύ μεγάλη ακρίβεια (ανάκληση, F1) κάνει τον αλγόριθμο να υπερκπαιδεύεται και γίνεται πιο δύσκολο να προσαρμοστεί σε νέα δεδομένα. Αυτό, βέβαια, δεν μπορούμε να πούμε ότι έχει επιτευχθεί ακόμα και με χρήση τέτοιου πλήθους stump, καθώς ακόμη είναι πολύ αντιληπτή η σύγκλιση των δύο τιμών ορθότητας, το οποίο είναι ένδειξη ότι το μοντέλο δεν υπερεκπαιδεύεται ή δεν υποεκπαιδεύεται στα δεδομένα εκπαίδευσης. Τα προβλήματα υπερεκπαίδευσης είναι συνήθως περισσότερο έντονα όταν το μοντέλο έχει περιορισμένο αριθμό δειγμάτων εκπαίδευσης, κάτι που δεν γίνεται στην δική μας περίπτωση. Επομένως, αντιλαμβανόμαστε ότι πρόκειται για ένα πολύ αξιόπιστο αλγόριθμο και συγκριτικά με τον δικό μας πιο ακριβές, «καλυπτικό».

Random Forest

Το αρχείο που περιέχει τον κώδικα με την υλοποίηση του αλγορίθμου με χρήση του Scikit-learn είναι το SKlearn\_RandomForest.py και με αυτό συγκρίναμε την δική μας υλοποίηση. Παρακάτω φαίνονται τα διαγράμματα που παράχθηκαν χρησιμοποιώντας τα ίδια δεδομένα εκπαίδευσης και την ίδια τιμή υπερπαραμέτρου(αριθμός δέντρων=10). Πρώτη και κύρια διαφορά σε σχέση με τον δική μας υλοποίηση είναι ότι ο αλγόριθμος της βιβλιοθήκης τρέχει εκθετικά γρηγορότερα (μέσα σε 2 λεπτά το πολύ ενώ ο δικός μας δυστυχώς απαιτεί 1 ώρα ,μπορεί και παραπάνω ) .Στα παρακάτω screenshots τα recall,precision,f1-score αφορούν την θετική κατηγορία(1). Είναι λογικό να έχουμε μεγαλύτερη ακρίβεια πρόβλεψης στα δεδομένα εκπαίδευσης, η οποία αγγίζει σχεδόν το 100% αφού πάνω σε αυτά εκπαιδεύτηκε το μοντέλο. Σε σύγκριση με τον δικό μας αλγόριθμο βλέπουμε όλες οι μετρικές να είναι πολύ καλύτερες και να επιτυγχάνεται πιο βέλτιστα η κατηγοριοποίηση.

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, γραμματοσειρά, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμματοσειρά, κείμενο, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμματοσειρά, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμματοσειρά, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμματοσειρά, κείμενο, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμματοσειρά, κείμενο, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει στιγμιότυπο οθόνης, γραμματοσειρά, κείμενο, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα



Το επόμενο διάγραμμα παριστάνει γραφικά τα ίδια δεδομένα:

Εικόνα που περιέχει κείμενο, διάγραμμα, γραμμή, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Στο παρακάτω διάγραμμα φαίνεται πως διαμορφώνεται το ποσοστό ακριβείας των προβλέψεων καθώς αυξάνεται το μέγεθος των δεδομένων εκπαίδευσης:

Εικόνα που περιέχει κείμενο, γραμμή, διάγραμμα, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Αυτή είναι η καμπύλη μάθησης που είχαμε παράξει με τα ίδια δεδομένα. Τα δεδομένα αξιολογήθηκαν με βάση το f1-score:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, ορθογώνιο παραλληλόγραμμο, γραμμή

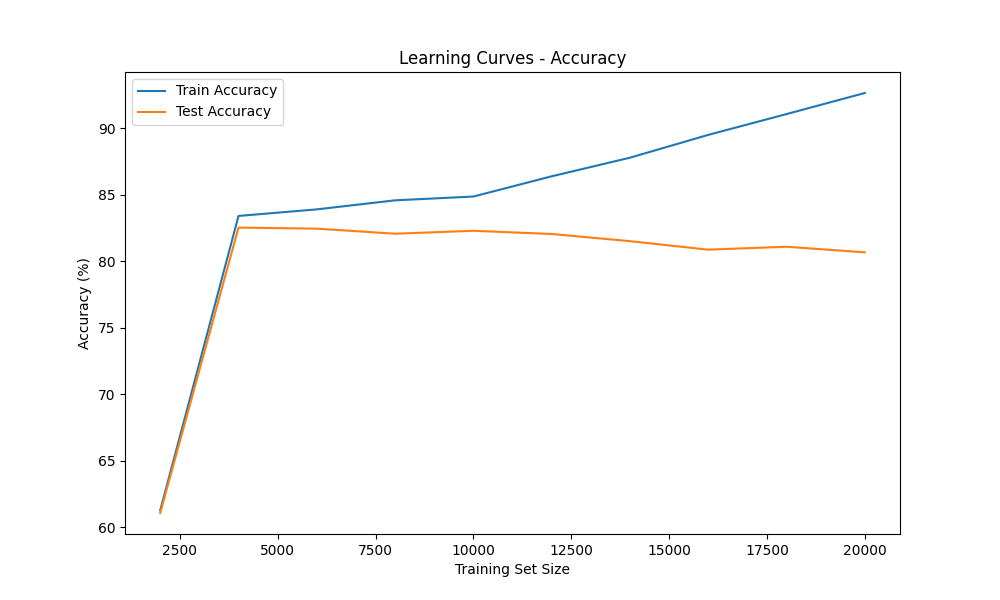
Περιγραφή που δημιουργήθηκε αυτόματα

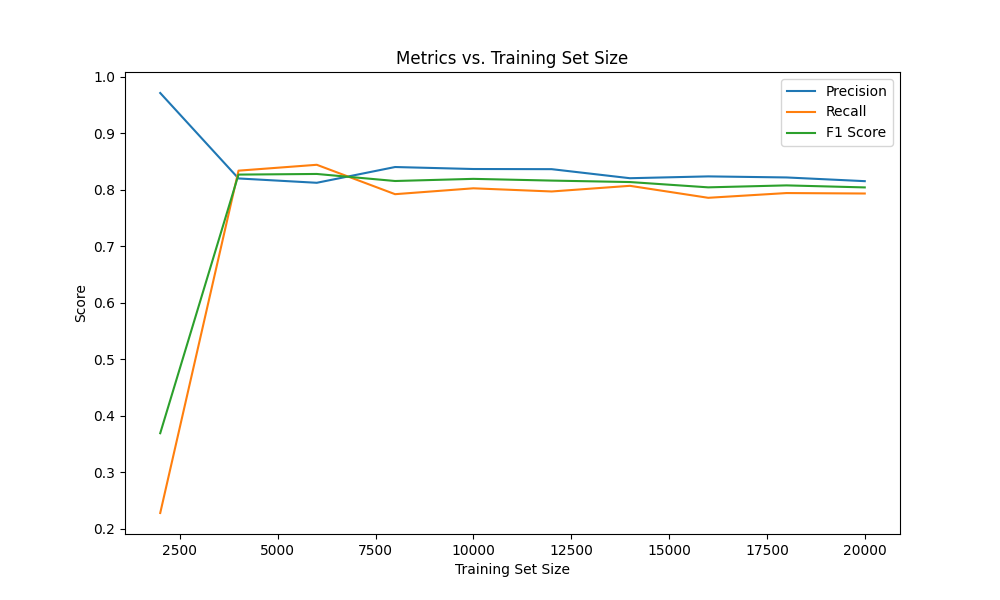
Μέρος Γ’

Σε αυτό το μέρος μας ζητείται με βάση τα δικά μας δεδομένα να εκτελέσουμε τον MLP, ο οποίος υλοποιείται μέσω Tensorflow/Keras, παριστάνοντας κάθε κριτική (κείμενο review) ως το κεντροειδές των ενθέσεων λέξεων (word embeddings). Αρχικά με τον όρο ενθέσεις λέξεων εκφράζουμε διανύσματα που απεικονίζουν τη σημασιολογική και συντακτική ομοιότητα μεταξύ των λέξεων. Οι κεντροειδές ενθέσεις λέξεων του κειμένου, δεν λαμβάνει υπόψη τη σημασία των λέξεων, οπότε μπορούμε να βελτιώσουμε την αναπαράσταση χρησιμοποιώντας τις IDF τιμές ως βάρη για τις ενθέσεις λέξεων. Οι IDF τιμές ενθέσεων λέξεων είναι ένας τρόπος να υπολογίσουμε τη σημασία μιας λέξης, αναφέροντας ως σημασία τη συχνότητα εμφάνισής της σε όλες τις κριτικές. Όσο πιο σπάνια είναι μια λέξη, τόσο πιο υψηλή είναι η IDF τιμή της και τόσο πιο πολύ συνεισφέρει στην αναπαράσταση του κειμένου. Έτσι, οι λέξεις που είναι πιο σπάνιες και πιο χαρακτηριστικές για το κείμενο έχουν μεγαλύτερη επίδραση στο κεντροειδές. Στην δική μας υλοποίηση υπολογίζουμε το βάρος των ενθέσεων σε σχέση με τον υπολογισμό των mostFrequentWords, που είχαμε δημιουργήσει προηγουμένως (λόγω του μεγάλου πλήθους των πιο συχνών λέξεων περιορίζουμε τα δεδομένα μας στην επεξεργασία των 100 πιο συχνών λέξεων, αν έχουμε αρχικά μεγαλύτερο πλήθος). Αρχικοποιούμε ως παράθυρο κειμένου το 5 και με βάση τις πιο συχνά εμφανιζόμενες λέξεις και αν η λέξη που εξετάζουμε κάθε φορά ανήκει σε αυτές δημιουργούμε τις κεντροειδής ενθέσεις, δηλαδή τα διανύσματα των λέξεων. Άρα θα έχουμε τελικά μία λίστα με τις κεντροειδής ενθέσεις λέξεων ανά παράθυρο κειμένου. Αφού έχουμε την λίστα των κεντροειδών ενθέσεων λέξεων, δημιουργούμε ένα Πολυεπίπεδο Νευρωνικό Δίκτυο (MLP) για την κατηγοριοποίηση των κειμένων. Τα word\_embeddings θα έχουν το ρόλο των ιδιοτήτων κατά την εκτέλεση του MLP.

Η υλοποίηση του MLP με χρήση των word embeddings αρχικά χρειάστηκε, να κάνουμε import το tensorflow, από το οποίο πακέτο θα χρησιμοποιήσουμε από τα υπάρχουσα μοντέλα του το Sequential, το οποίο υλοποιεί ένα σειριακό μοντέλο νευρωνικού δικτύου, δηλαδή οι στρώσεις προστίθενται η μία μετά την άλλη, σειριακά. Οι στρώσεις αρχικοποιούνται με την χρήση της Dense(), όσες φορές καλείται τόσες στρώσεις δημιουργούνται, και δέχεται ως ορίσματα το πλήθος των νευρώνων που θα έχει κάθε στρώσει και τον τρόπο, με ποια συνάρτηση, θα ενεργοποιηθεί (Αλλά σχόλια για την δημιουργία της υλοποίησης βρίσκονται στον κώδικα). Για να υλοποιήσουμε τον MLP με τη χρήση λέξεων με κεντροειδή ενθέσεις (word embeddings), χρειαζόμαστε τις λέξεις ως κείμενα και όχι σαν σύνολα συχνοτήτων. Επομένως θα κάνουμε ορισμένες αλλαγές στον τρόπο επεξεργασίας των δεδομένων που μας δίνονται. Από τα αρχεία που έχουμε αποθηκευμένες τις αρνητικές/θετικές κριτικές με την αφαίρεση περιττών σημείων στίξης και τη μετατροπή των λέξεων σε πεζά, όπως ήδη έχουμε κάνει, διαχωρίζουμε τα κείμενα σε λέξεις. Έπειτα επεξεργαζόμαστε τις λέξεις ως κείμενα για τη δημιουργία των word embeddings. Με βάση τα ζητούμενα μας αρχικά θα εκτελέσουμε την συγκεκριμένη υλοποίηση με δύο στρώματα και περισσότερες νευρώσεις, διότι περισσότεροι νευρώνες ανά στρώση, μπορούν να κατανοήσουν και να εκφράσουν πιο σύνθετα χαρακτηριστικά και συνήθως επιτρέπουν στο μοντέλο να μάθει πιο πολύπλοκες αναπαραστάσεις με μειονέκτημα βέβαια την πιθανότητα υπερεκπαίδευσης αν αυξηθεί κατά πολύ το πλήθος τους. Ενώ περισσότερες στρώσεις, ωφελούν σε ιεράρχηση χαρακτηριστικών, πράγμα που στην περίπτωση μας δεν συμφέρει λόγω του περιορισμένου πλήθους χαρακτηριστικών. Ο τρόπος που κάνουμε την εκπαίδευση του μοντέλου γίνεται με χρήση ενός πλήθους που ονομάζεται epochs, το οποίο δηλώνει το πλήθος των περιόδων/εποχών που θα γίνει πλήρη επεξεργασία των δεδομένων εκπαίδευσης, που έχουν οριστεί ως πλήθος από την μεταβλητή batch\_size (Η χρήση των batches βοηθάει στην επιτάχυνση της εκπαίδευσης, καθώς οι υπολογισμοί είναι πιο αποτελεσματικοί για μικρά batches). Για κάθε περίοδο προσαρμόζεται ο αριθμός των δεδομένων εκπαίδευσης καθώς και το βάρος τους. Μία βασική διαφορά, λοιπόν, που έχουμε στην συγκεκριμένη υλοποίηση είναι ότι δεν ορίζουμε εμείς (με βάση την διάσπαση των δεδομένων ποια από αυτά θα χρησιμοποιηθούν ως train και ποια ως test data), αλλά αντ’ αυτού δηλώνουμε το πλήθος epochs και το batch\_size (έννοιες που δηλώσαμε παραπάνω), με τα οποία στοιχεία γίνεται η διάσπαση. Χρησιμοποιούμε και πάλι ως μέγεθος παραδειγμάτων εκπαίδευσης το max\_train\_size=μέγεθος του διανύσματος εκπαίδευσης και step\_size, που είναι το 10% του max\_train\_size. Λόγω της λειτουργίας της συγκεκριμένης υλοποίησης όπως εξηγήσαμε, και λόγω του ότι το testing dataset απορρέει από την εκπαίδευση ανά περίοδο των εκπαιδευτικών δεδομένων με χρήση του πλήθους που αντιστοιχεί κάθε φορά στο batch\_size. Τα word embeddings τα δηλώνουμε με τρόπο τέτοιο που δημιουργούμε δύο επίπεδα στο μοντέλο του νευρωνικού δικτύου. Το ένα είναι το επίπεδο ενσωμάτωσης, το οποίο λειτουργεί ως μια προ-εκπαιδευμένη ενσωμάτωση λέξεων, χρησιμοποιώντας προ-εκπαιδευμένα embeddings ως αρχικά βάρη και απενεργοποιώντας την εκπαίδευση των ενσωματώσεων κατά τη διάρκεια της εκπαίδευσης του μοντέλου (trainable=False) και το άλλο είναι το Flatten Layer, το οποίο χρησιμοποιείται για να επίπεδώσει τα δεδομένα, δηλαδή να τα μετατρέψει από τον τρισδιάστατο χώρο σε έναν δισδιάστατο χώρο. Αυτό είναι απαραίτητο πριν από τα πλήρως συνδεδεμένα επίπεδα του νευρωνικού δικτύου. Το παραπάνω είναι μία ακολουθία επιπέδων, που επιτρέπει στο δίκτυο να εκμεταλλευτεί προ-εκπαιδευμένες αναπαραστάσεις λέξεων και στη συνέχεια επίπεδα πλήρως συνδεδεμένα για την κατανομή τους σε θετικές και αρνητικές (ταξινόμηση). Για την εκτέλεση του παρακάτω παραδείγματος αρχικοποιήσαμε ως output\_dim του μοντέλου, δηλαδή το μέγεθος των παραδειγμάτων εκπαίδευσης, ως 100 (embedding\_dim = 100). Το μοντέλο, που υλοποιούμε, έχει 3 στρώσεις με 64, 32 και 1 νευρώνες. Επίσης, αρχικοποιούμε epochs=2, batch\_size=100, validation\_split=0.2. Για μεγέθη εκπαιδευτικών παραδειγμάτων με πλήθος κάθε φορά όπως αναφέρεται δίνονται τα εξής:

Το μοντέλο, που υλοποιούμε, έχει 3 στρώσεις με 64, 32 και 1 νευρώνες. Επίσης, αρχικοποιούμε epochs=2, batch\_size=100, validation\_split=0.2. Για μεγέθη εκπαιδευτικών παραδειγμάτων με πλήθος κάθε φορά όπως αναφέρεται δίνονται τα εξής:

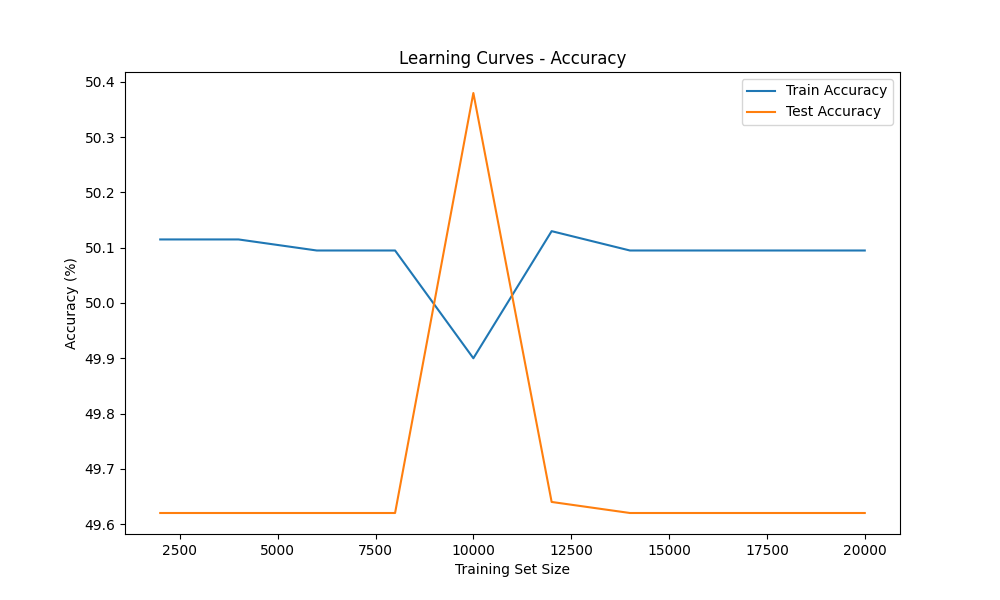


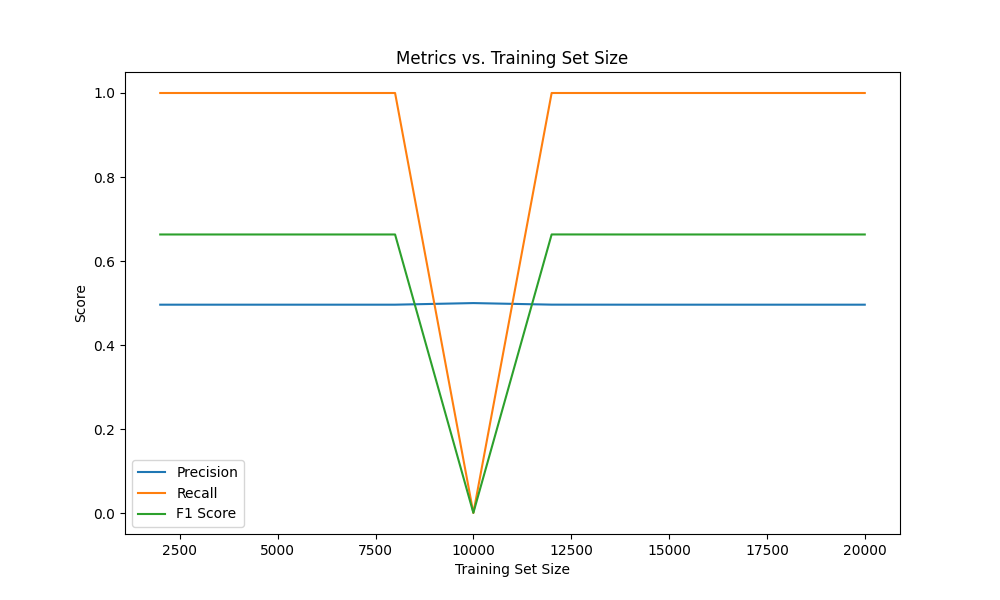


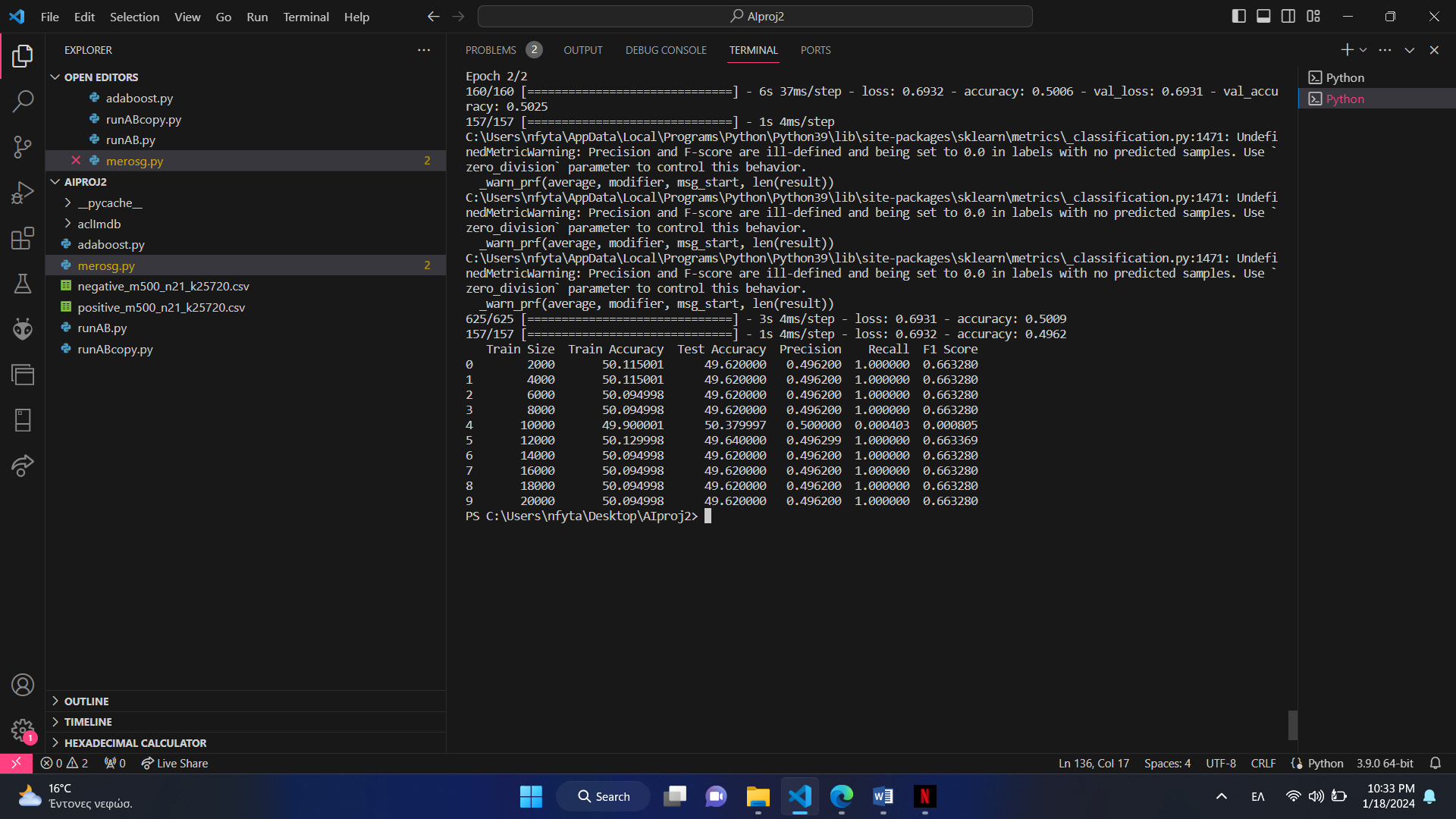


Από τα παραπάνω στοιχεία, παρατηρούμε ότι η εφαρμογή του συγκεκριμένου μοντέλου σε πολύ μικρό αριθμό παραδειγμάτων εκπαίδευσης δεν έχει καλή απόκριση και καθώς μεγαλώνει το μέγεθος η απόδοση όλων των μεγεθών αυξάνεται κατακόρυφα. Αυτό, οφείλεται κυρίως, σε αυτό που προαναφέραμε, ότι δηλαδή σε λίγα εκπαιδευτικά παραδείγματα η απόδοση του νευρωνικού δικτύου, όπως έχουμε ορίσει τις παραμέτρους δεν ήταν η βέλτιστη. Μία γενική παρατήρηση και σε σχέση με τις προηγούμενες υλοποιήσεις εκπαίδευσης, με τους αλγόριθμους που χρησιμοποιήθηκαν στο μέρος Α και Β, είναι ότι έχουμε μία πολύ καλή απόδοση του αποτελέσματος, το οποίο είναι φανερό από την σύγκλιση των τιμών ακρίβειας, ανάκλησης και f1, πράγμα που δεν συναντήσαμε ποτέ με τους παραπάνω αλγορίθμους. Επίσης είναι φανερό ότι έχουμε μία κατακόρυφη βελτίωση όσον αφορά την ορθότητα των αποτελεσμάτων, πράγμα που πετύχαμε στους προηγούμενους αλγόριθμους με την εκτέλεση τους σε όσο το δυνατόν καλύτερες συνθήκες, π.χ. στον adaboost, όπως υλοποιείται από το Scikit-learn, πετύχαμε τέτοια ορθότητα στην δημιουργία 600 stump. Τέλος, παρατηρείται μία συνεχής άνοδο των τιμών της ορθότητας εκπαίδευσης, το οποίο ενώ είναι ένα θετικό σενάριο όσον αφορά την απόδοση του δικτύου που χρησιμοποιούμε, παρατηρείται μία παράλληλη σχετική μείωση της ορθότητας των ελέγχων, το οποίο πολλές φορές δηλώνει περίπτωση υπερεκπαίδευσης.

Αν κρατήσουμε όλα τα δεδομένα, ίδια και απλώς δημιουργήσουμε το μοντέλο με 2 στρώσεις με 64 και 1 νευρώνες. Έχουμε τα εξής:

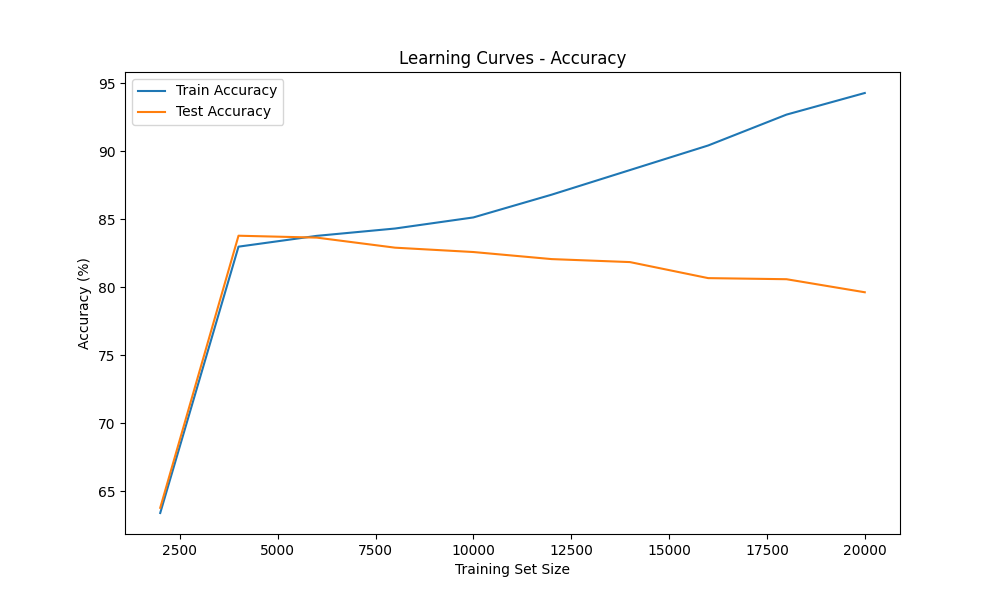


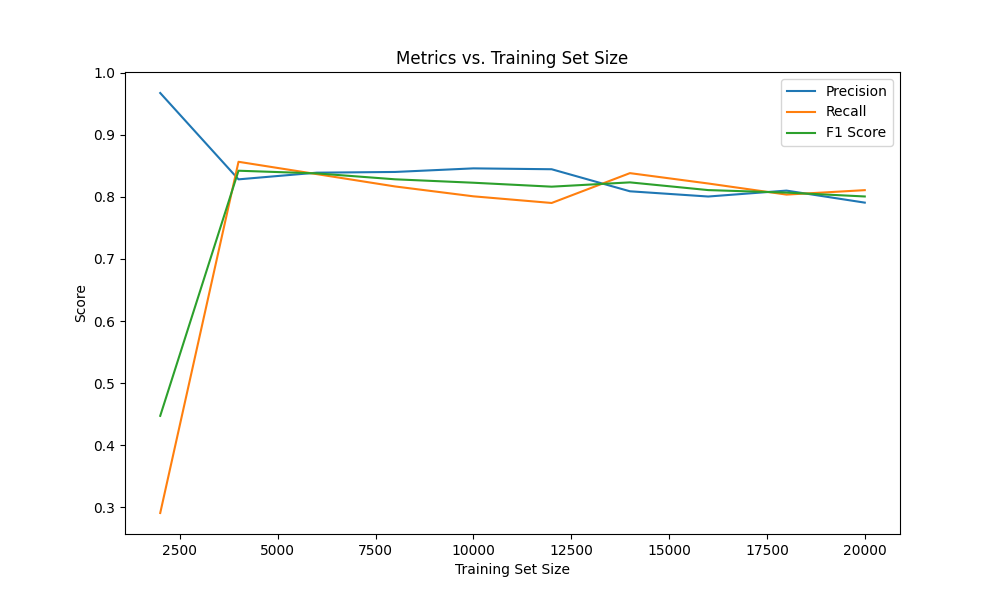


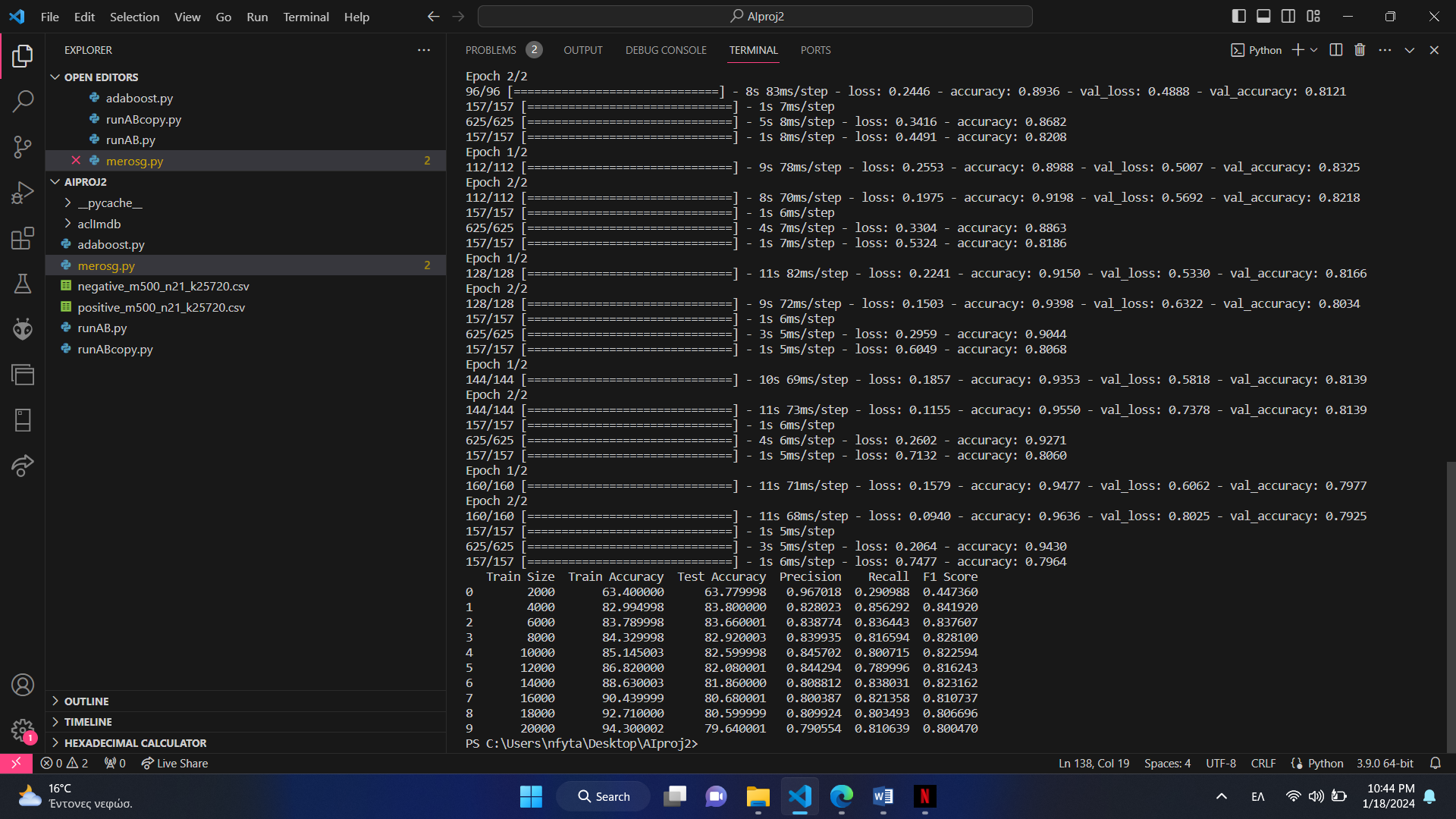


Βλέπουμε ότι στην συγκεκριμένη περίπτωση όλες οι τιμές είναι πολύ κοντινές και δεν έχουμε κάποια τρομερή απόκλιση πράγμα που συμβαίνει εξαιτίας του ότι λιγότερα στρώματα εξυπηρετούν και μικρά μεγέθη παραδειγμάτων, αλλά δυσκολεύονται αρκετά σε μεγαλύτερα μεγέθη. Φαίνεται δηλαδή ότι με τον τρόπο που κτίσαμε τώρα το μοντέλο μας δεν έχει την βέλτιστη απόδοση.

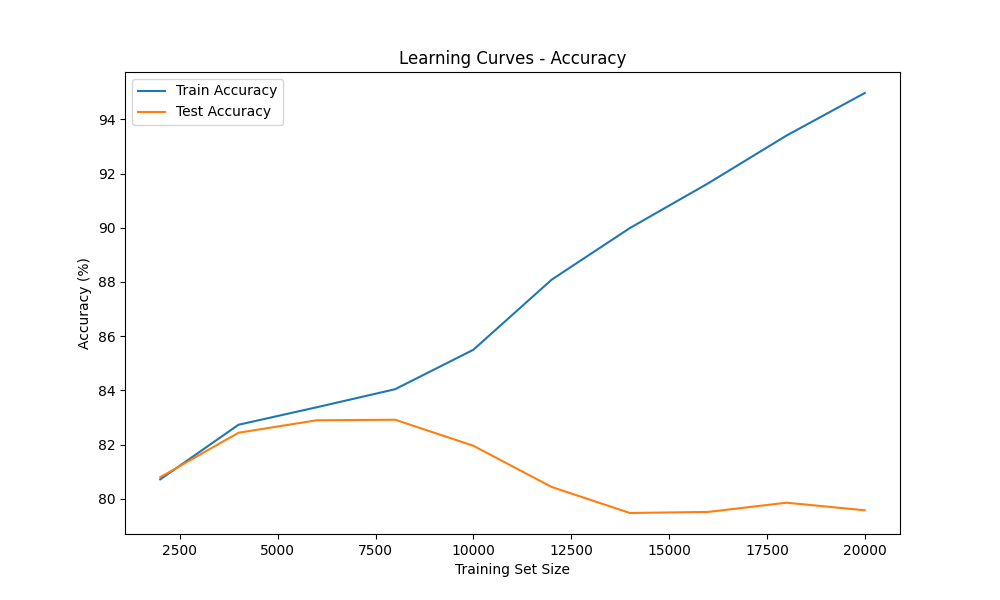
Με τα δύο παραπάνω παραδείγματα, γίνεται αντιληπτή η επιρροή των στρωμάτων του μοντέλου στην απόδοση των αποτελεσμάτων. Αν δοκιμάσουμε με τα πρωταρχικά δεδομένα να αυξήσουμε τους νευρώνες ανά στρώμα, δηλαδή για 3 στρώματα με νευρώνες αυτή την φορά 128, 64 και 1 μας δίνονται τα εξής(προφανώς ο χρόνος εκτέλεσης αυξάνεται σημαντικά):

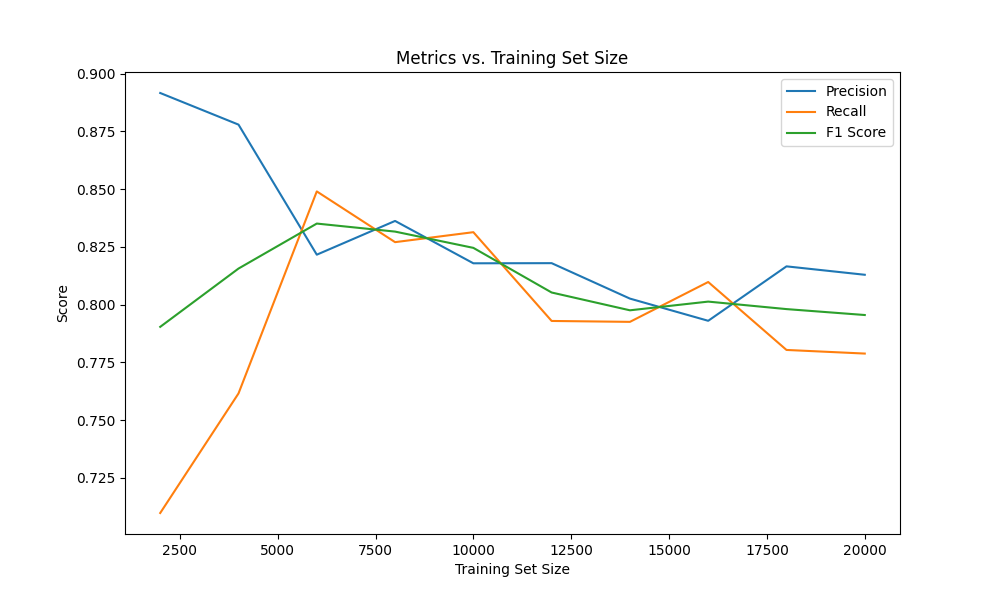


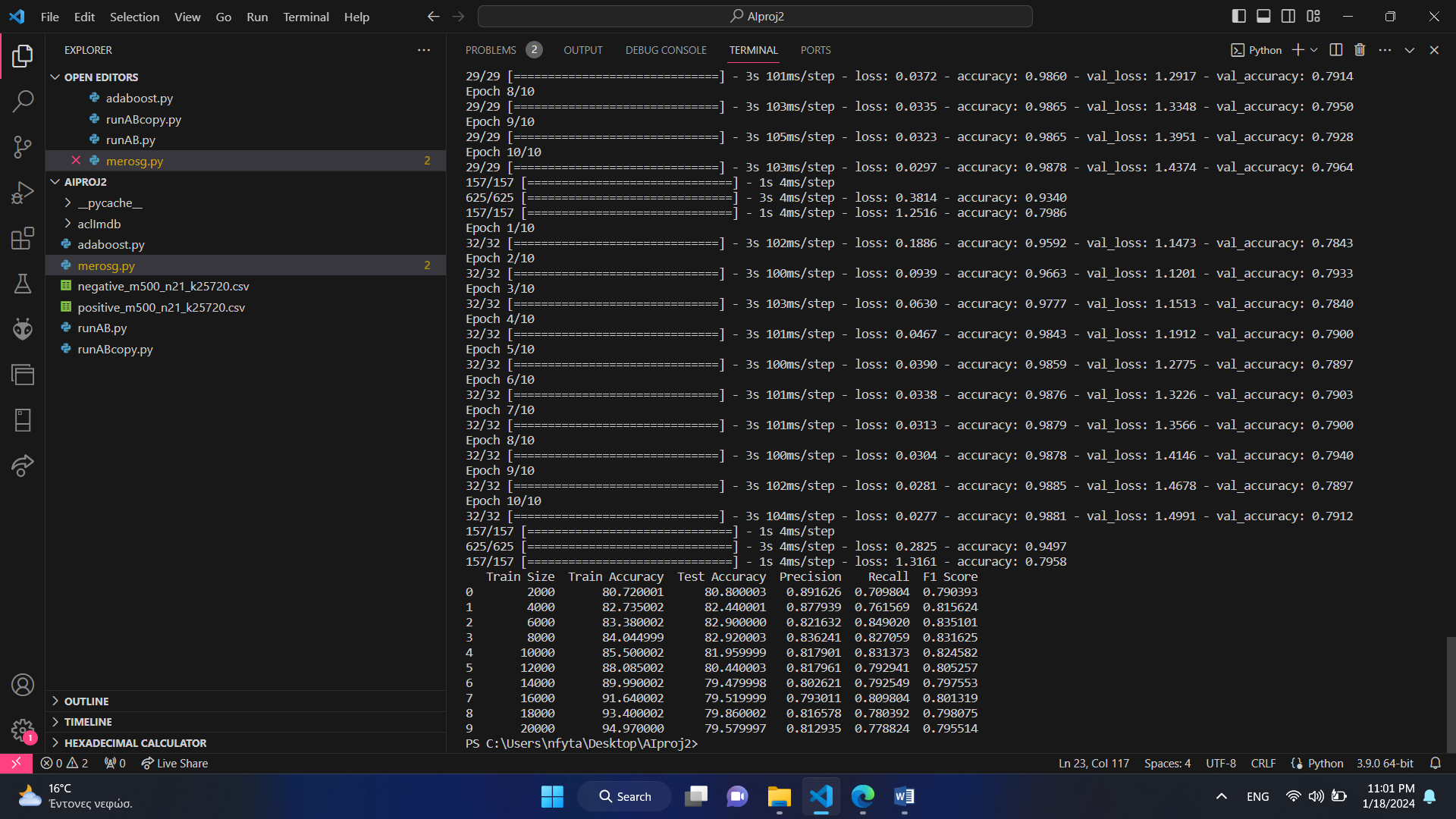




Βλέπουμε ότι πάλι για μικρά μεγέθη εκπαιδευτικών παραδειγμάτων έχουμε μία μέτρια απόδοση του μοντέλου, το οποίο είναι λογικό γιατί όπως προαναφέραμε η αύξηση του αριθμού των στρωμάτων δεν ωφελεί μικρά πλήθη δεδομένων, αλλά όπως αυξανόμαστε υπάρχει μία αντιληπτή αύξηση της απόδοσης σε σχέση και με το μοντέλο 3 στρωμάτων που είχε 64,32 και 1 νευρώνες, που εξετάσαμε παραπάνω. Γενικότερα και σε σύγκριση με την απόδοση των αλγορίθμων που αναπτύξαμε οι ίδιοι στο μέρος Α αλλά και των ήδη κατασκευασμένων από την Scikit-learn που εκτελέσαμε στο μέρος Β, βλέπουμε ότι η κατασκευή ενός MLP με κυλιόμενο παράθυρο, παριστάνοντας τις λέξεις με ενθέσεις, δίνει καλύτερα αποτελέσματα, με ενδεχόμενο την ύπαρξη υπερεκπαίδευσης.

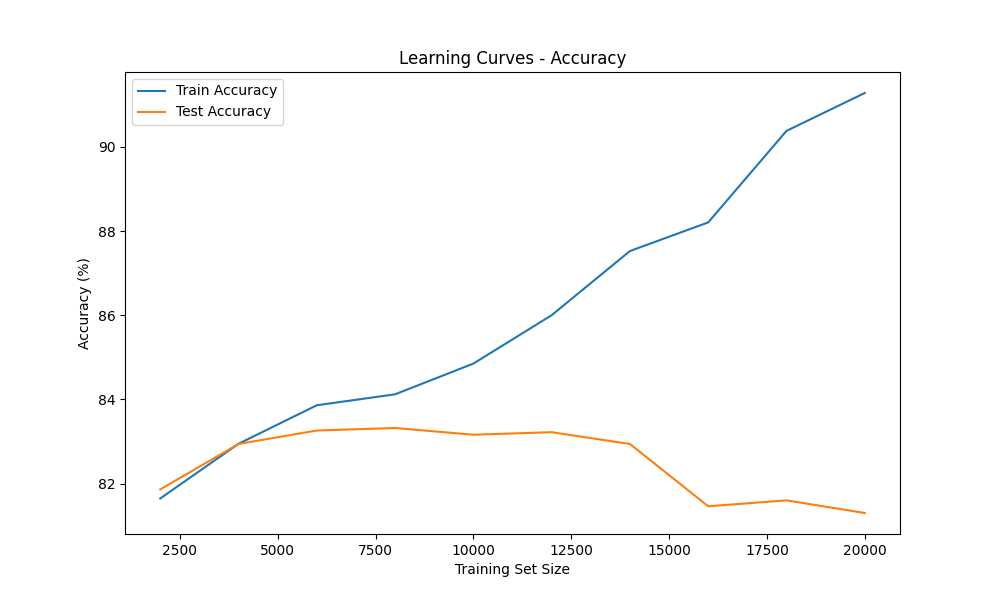
Για να εξετάσουμε και πως οι άλλες τιμές επηρεάζουν την απόδοση του συγκεκριμένου μοντέλου θα χρησιμοποιήσουμε τις ίδιες τιμές, δηλαδή έχουμε 3 στρώσεις με 64, 32 και 1 νευρώνες, αλλά αλλάζουμε τα epochs από 2 σε 10 και το batch\_size από 100 σε 500 (επίτηδες αλλάζουμε και τα δύο για να μην τρέχει πολύ ώρα), validation\_split=0.2 και embedded words=100.



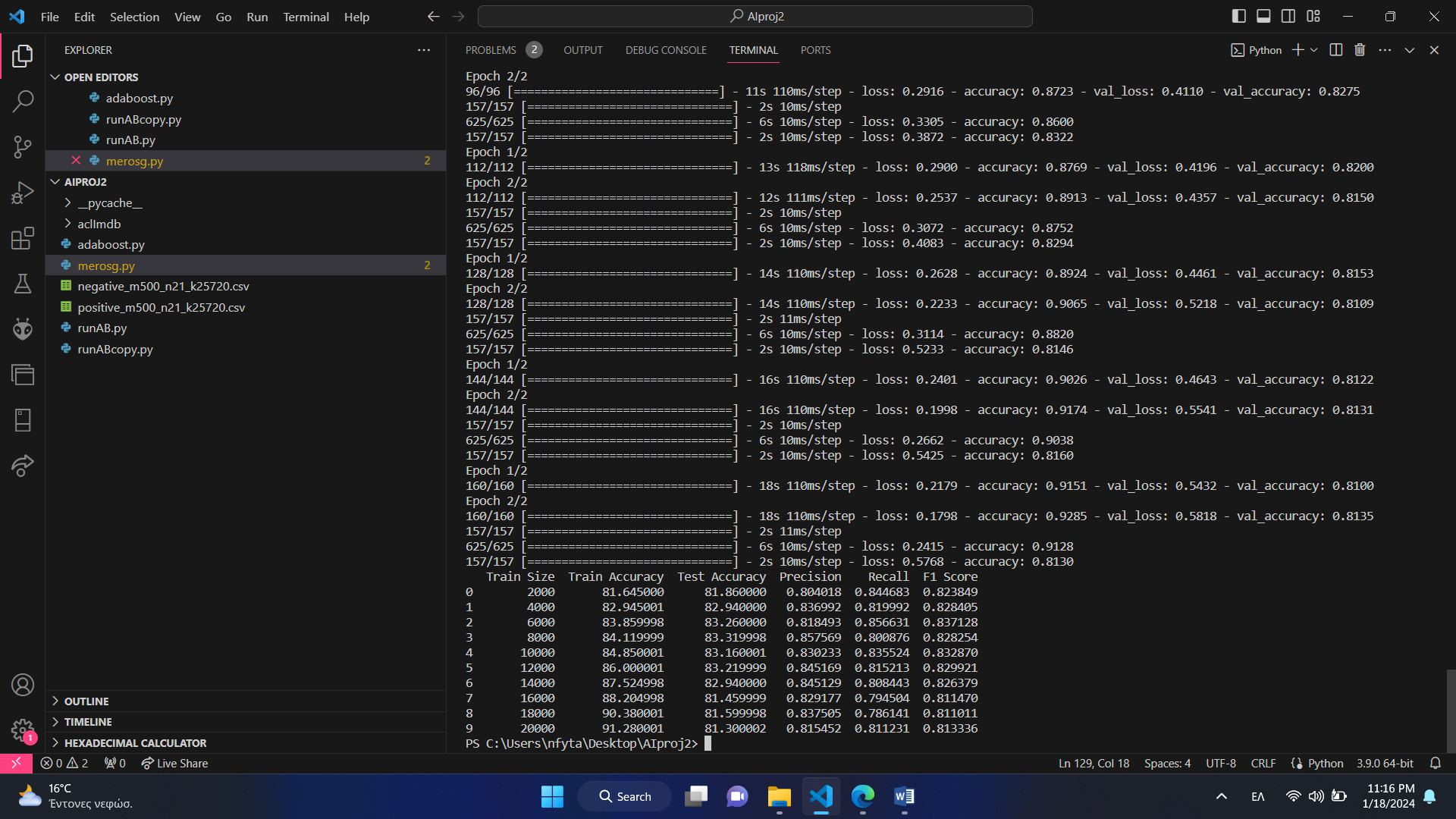


Βλέπουμε ότι η αλλαγή που κάναμε όσον αφορά το πλήθος των φορών που θα επεξεργαστούμε πλήρως τα εκπαιδευτικά δεδομένα που αντιστοιχούν στο batch\_size, όπως έχει οριστεί ενώ δίνει μεγάλη ακρίβεια και απόδοση βλέπουμε ότι η ορθότητα των δεδομένων εκπαίδευσης με αυτής του ελέγχου έχουν μεγάλη απόκλιση κατά την αύξηση του μεγέθους εκπαιδευτικών παραδειγμάτων, κάτι που σημαίνει ότι πρόκειται για υπερεκπαίδευση.

Άλλο ένα παράδειγμα που θα ωφελούσε να μελετήσουμε είναι αυτό της αλλαγής του διανύσματος της ένθεσης λέξεων από 100 σε 300 (που είναι το μέγιστο), το οποίο βλέπουμε ότι δεν επέφερε μεγάλες αλλαγές στην ορθότητα των δεδομένων ελέγχου και εκπαίδευσης αλλά κυρίως στις τιμές ακρίβειας, ανάκλησης και f1 το οποίο συμβαίνει όταν δυσκολεύεται να αναγνωρίσει τις αρνητικές στην δική μας περίπτωση κριτικές, το οποίο λόγω του εξετάσαμε το ίδιο παράδειγμα με μικρότερο αριθμό ένθεσεις λέξεων, φαίνεται ότι ευθύνεται στην αλλαγή της τιμής των embedded words.







Στο μέρος Γ επιπλέον μας ζητήθηκε να δημιουργήσουμε και τις καμπύλες που να δείχνουν τη μεταβολή στου σφάλματος (loss) στα παραδείγματα εκπαίδευσης και ανάπτυξης, συναρτήσει του αριθμού των εποχών. Για να τις χτίσουμε αυτές χρειάστηκε να ορίσουμε την μεταβλητή epoch, και ανεξάρτητα της κλήσης fit του μοντέλου ώστε να δείξουμε το σφάλμα ανά εποχή, για τα εκπαιδευτικά δεδομένα υπολογίζουμε το σφάλμα από το fit των δεδομένων στο μοντέλο (classifiers.history['loss']) ενώ το σφάλμα που προέρχεται από τα δεδομένα ελέγχου γίνεται με βάση την πρόβλεψη και υπολογίζεται από το test\_loss, test\_accuracy = model.evaluate(test\_features, test\_labels). Για τα δεδομένα όπως τα είχαμε παρουσιάσει εξαρχής, δηλαδή με embedding\_dim = 100. Το μοντέλο, που υλοποιούμε, έχει 3 στρώσεις με 64, 32 και 1 νευρώνες. Επίσης, αρχικοποιούμε epochs=2, batch\_size=100, validation\_split=0.2. Για μεγέθη εκπαιδευτικών παραδειγμάτων με πλήθος κάθε φορά όπως αναφέρεται δίνονται τα εξής σφάλματα:



Που σημαίνει ότι με κάθε αύξηση παραδειγμάτων εκπαίδευσης επέρχεται μείωση του σφάλματος στα δεδομένα εκπαίδευσης, το οποίο είναι λογικό διότι όπως παρουσιάσαμε αυξάνει η ορθότητα, ενώ έχουμε αύξηση του σφάλματος στα δεδομένα ελέγχου, το οποίο εξίσου λογικό αν λάβουμε υπόψη τα αποτελέσματα που μας δόθηκαν παραπάνω. Παρατηρούμε, επιπλέον, ότι στην δεύτερη εποχή έχουμε μικρότερο σφάλμα. Για να γίνει πιο ξεκάθαρο ας δοκιμάσουμε την εκτέλεση του συγκεκριμένου νευρωνικού δικτύου με 10 εποχές αντί 2, αυτό που περιμένουμε είναι η συνεχής ελάττωση του σφάλματος των εκπαιδευτικών δεδομένων ανά εποχή.



Βλέπουμε ότι όντως πρόκειται για σύγκλιση στο μηδέν όλων των σφαλμάτων για μεγάλα μεγέθη παραδειγμάτων εκπαίδευσης, ενώ στην αρχή που έχουμε λιγότερα δεδομένα και είναι πιο δύσκολο να εκπαιδευτεί το μοντέλο έχουμε μεγαλύτερο σφάλμα. Επίσης πια είναι φανερό ότι με το ίδιο μέγεθος δεδομένων κάθε επόμενη εποχή είναι καλύτερη από την προηγούμενη.