2η Εργασία Τεχνητής Νοημοσύνης

Στογιαννίδης Ηλίας Μάριος 3180178

Διαμαντόπουλος-Πανταλέων Οδυσσεύς 3180049

Παπαθανασίου Αθανασία-Μαρία 3180147

Περίληψη:

Διαλέξαμε να υλοποιήσουμε τους αλγορίθμους:

1. ID3
2. Random Forest
3. Logistic Regression

Main.py:

Η MAIN ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ.

if __name__ == '__main__':

Προσφέρει το μενού στον χρήστη.

accuracy(y_true, y_pred):

Δέχεται τις απαντήσεις που έδωσε ο αλγόριθμος και αυτές που πραγματικά είναι, τις συγκρίνει και βγάζει το επιστρέφει ακριβής ήταν ο αλγόριθμος.

vectorize(sequences, dimension=1000):

Δέχεται όλα τα δεδομένα(x train και x test μαζί)και επιστρέφει ένα πίνακα results στον οποίο οι γραμμές παριστάνουν τις κριτικές και οι στήλες τις λέξεις που έχουμε αποφασίσει να χρησιμοποιήσουμε. Οπότε αν στην γραμμή 2, στήλη 1 υπάρχει άσος σημαίνει ότι η $3^{η}$ κριτική έχει την δεύτερη λέξη. Τέλος επιστρέφει τον results.

plotfunction(train_accs, dev_accs, str1, str2, operation=0):

Δέχεται δύο λίστες που περιέχουν μετρημένα δεδομένα και δύο strings και τα χρησιμοποιεί για να φτιάξει δύο plots.

runID3(numword, skiptop, maxdepth, plot=False):

Είναι η μέθοδος που καλεί την ID3 και δέχεται ως ορίσματα τον αριθμό των λέξεων που θα πάρει, πόσες να παρακάμψει και το max depth, και αναλόγως με το τι έχει επιλέξει ο χρήστης στο μενού, είτε την εκπαιδεύει σιγά σιγά και μετά κατασκευάζει διαγράμματα και παρουσιάζει τα

δεδομένα που προκύψανε, είτε τον εκπαιδεύει σε όλα τα train δεδομένα μαζί και βγάζει τα αποτελέσματα.

runRandomForest(numword, skiptop, maxdepth, trees, plot=False):

Είναι η μέθοδος που καλεί την RandomForest και δέχεται ως ορίσματα τον αριθμό των λέξεων που θα πάρει, πόσες να παρακάμψει, το max depth και τον αριθμό των δέντρων, και αναλόγως με το τι έχει επιλέξει ο χρήστης στο μενού, είτε την εκπαιδεύει σιγά σιγά και μετά κατασκευάζει διαγράμματα και παρουσιάζει τα δεδομένα που προκύψανε, είτε τον εκπαιδεύει σε όλα τα train δεδομένα μαζί και βγάζει τα αποτελέσματα.

runLogisticRegression(numword, skiptop, numiter, learning, plot=False):

Είναι η μέθοδος που καλεί την Logistic Regression και δέχεται ως ορίσματα τον αριθμό των λέξεων που θα πάρει, πόσες να παρακάμψει, τον αριθμό των iterations και τον αριθμό του learning rate, και αναλόγως με το τι έχει επιλέξει ο χρήστης στο μενού, είτε την εκπαιδεύει σιγά σιγά και μετά κατασκευάζει διαγράμματα και παρουσιάζει τα δεδομένα που προκύψανε, είτε τον εκπαιδεύει σε όλα τα train δεδομένα μαζί και βγάζει τα αποτελέσματα.

precision(y_true, y_pred):

Υπολογίζει το precision.

recall(y_true, y_pred):

Υπολογίζει το recall.

f1(y_true, y_pred):

Υπολογίζει το f1

counterFunction(y_true, y_pred, operation):

Υπολογίζει τα True positive, False positive, True negative, False negative.

ID3:

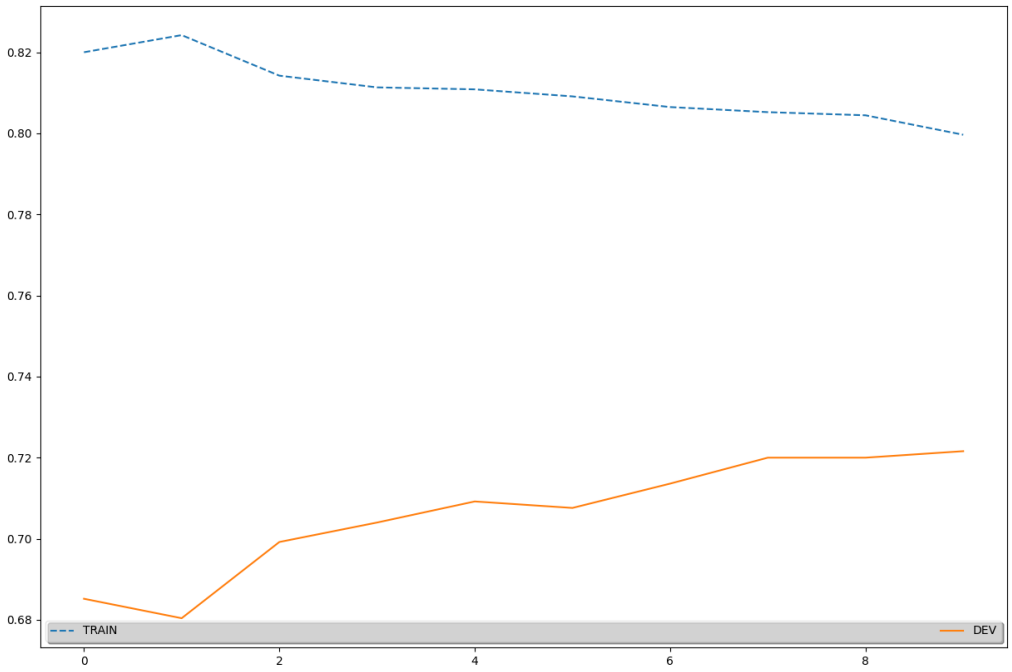
Χωρίσαμε τα δεδομένα στα δύο, πήραμε 25000 test data και 25000 training data. Για να φτιάξουμε τα plots τρέξαμε το αλγόριθμο πολλές φορές και κάθε φορά του δίναμε διαφορετικό ποσοστό training data. Έπειτα αποθηκεύσαμε το accuracy που παίρνουμε τεστάροντας τον εκπαιδευμένο αλγόριθμο και στα data που έχει εκπαιδευτεί και στα dev δεδομένα που δεν έχει ξαναδεί, και χρησιμοποιούμε τα αποθηκευμένα accuracies για να κατασκευάσουμε τα plots. Τέλος υπολογίσαμε το precision και το recall για το καλύτερο μοντέλο μας και για διαφορετικά thresholds και φτιάξαμε τα plots.

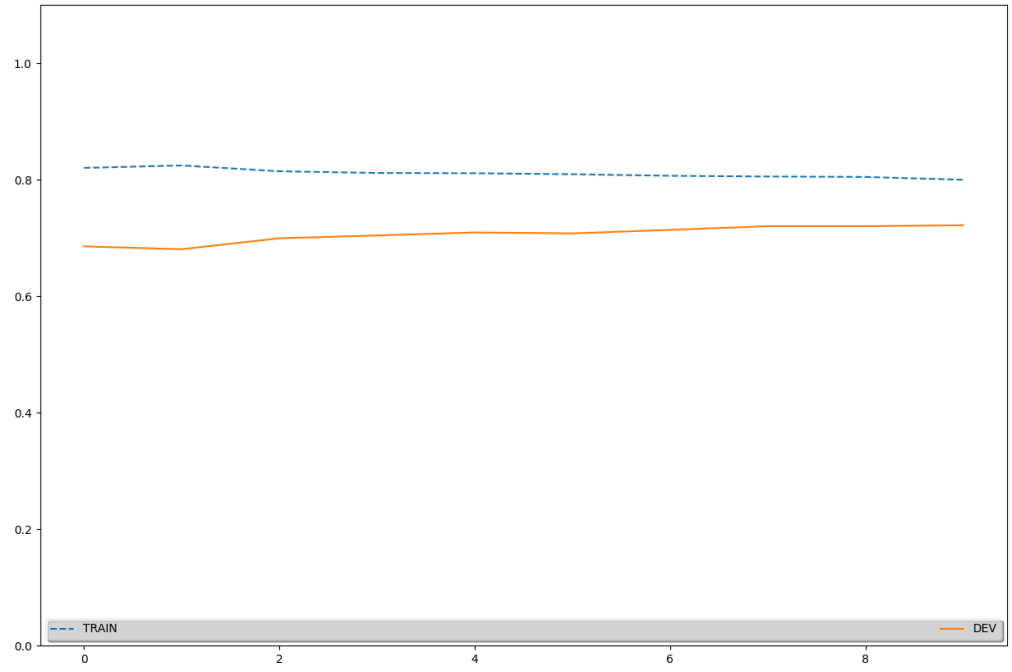
Οι τιμές των υπερπαραμέτρων επιλέχθηκαν μετά από πολλές δοκιμές. Πιο συγκεκριμένα:

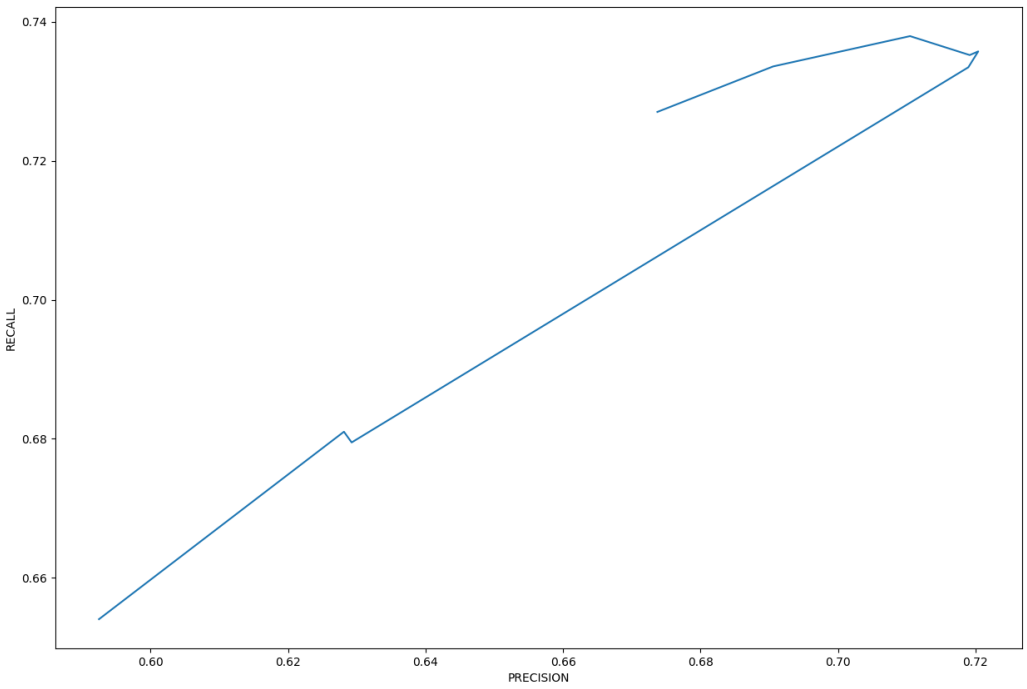
Accuracy	Number of Words	Skip Words	Max Depth
0.642	100	10	10
0.71096	500	100	15
0.72048	1000	200	20
0.67928	4000	400	20

Τα plots που προκύψανε με 1000 words, 200 skip words και 20 max depth είναι τα :

```
Highest Test Accuracy for the best dev point:  0.7198
```




1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
TRAIN
DEV


0.74
0.72
0.70
0.68
0.66
RECALL
0.60
0.62
0.64
0.66
0.68
0.70
0.72
PRECISION

Precision: [0.7270251376199934, 0.733570622782197, 0.73793334471796, 0.7352071084662367, 0.735733371841984, 0.7334479612577576, 0.6794709161398031, 0.6810128929464954, 0.6540373848839713]

Recall: [0.67372, 0.6905600000000001, 0.71048, 0.71916, 0.7203999999999999, 0.71896, 0.62924, 0.62812, 0.59248]

F1: [0.6993583094631771, 0.7114158226284392, 0.7239464958910706, 0.7270950244222376, 0.7279859542048625, 0.72613172096527, 0.6533914770619009, 0.653497930763604, 0.6217387330416176]

Μέθοδοι:

entropy(y):

Δέχεται ένα πίνακα y ο οποίος περιέχει τις πιθανές απαντήσεις στα δεδομένα και υπολογίζει σύμφωνα με τον τύπο της εντροπίας την εντροπία.

class Node:

Αφού θα φτιάξουμε δέντρο χρειαζόμαστε κόμβους.

__init__(self, feature=None, threshold=None, left=None, right=None, *, value=None):

Το feature είναι η λέξη με βάση διαχωρίσαμε τον κόμβο, το threshold είναι η τιμή που λαμβάνει το feature, το left είναι το αριστερό παιδί του κόμβου και το right το δεξί. Τέλος αν έχει value τότε είναι leaf node.

is_leaf_node(self):

Επιστρέφει αν το node έχει value, δηλαδή αν είναι φύλλο

class DecisionTree:

Η κλάση του αλγορίθμου

__init__(self, min_samples_split=2, max_depth=100, n_feats=None, threshold=0.5):

Το min samples split λέει ότι τα minimum δεδομένα που χρειαζόμαστε για να κάνουμε split είναι 2. Το max depth είναι το πόσο θα αφήσουμε το δέντρο να μεγαλώσει. Το threshold είναι 0.5 αλλά μπορεί να αλλάξει για τα precision recall diagrams. Τέλος το n_feats χρησιμοποιείται για την random forest οπότε για τον ID3 δεν χρειάζεται.

fit(self, X, y):

Χρησιμοποιείται για να ενσωματώσει τα δεδομένα μας για να μπορέσει να τα χρησιμοποιήσει ο αλγόριθμος.

predict(self, X):

Διατρέχει το δέντρο και κάνει predict τι κριτική είναι με βάση το κατασκευασμένο δέντρο.

_grow_tree(self, X, y, depth=0):

Δημιουργεί το δέντρο. Το δέντρο δεν μπορεί να μεγαλώσει παραπάνω αν ξεπεραστεί το max depth ή αν είμαστε σε leaf node ή αν τα δεδομένα μας είναι πολύ λίγα για να κάνουμε split. Αν δεν ισχύει κάτι από αυτά βρίσκει ποιο είναι το καλύτερο κριτήριο (εδώ λέξη) για να χωρίσει το δέντρο και αφού το βρει μεγαλώνει το δέντρο με βάση αυτό και επιστρέφει το νέο κόμβο.

_best_criteria(self, X, y, feat_idxs):

Αποφασίζει ποιο είναι το καλύτερο χαρακτηριστικό. Χρησιμοποιεί την information gain και κρατάει το καλύτερο information gain και επιστρέφει τις πληροφορίες του.

_information_gain(self, y, X_column, split_thresh):

Υπολογίζει με βάση τους τύπους το information gain κάποιου χαρακτιρηστικού. Χρησιμοποιεί την entropy για να το υπολογίσει.

_split(self, X_column, split_thresh):

Κάνει split με βάση ένα threshold.

_traverse_tree(self, x, node):

Διατρέχει το δέντρο. Ελέγχει αν ο κόμβος έχει leaf node και τότε επιστρέφει την τιμή του. Αν δεν έχει τότε ελέγχει αν η κριτική περιέχει την λέξη με βάση την οποία έκανε split το δέντρο και αναλόγως ξανακαλεί την traverse για ένα από τα παιδιά του κόμβου αυτού και επιστρέφει ότι βρει από αυτό.

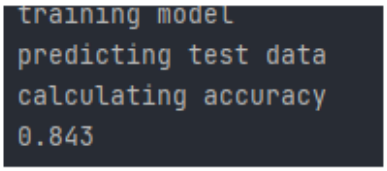
_most_common_label(self, y):

Επιστρέφει 1 αν το ποσοστό των θετικών είναι μεγαλύτερο από δοσμένο threshold.

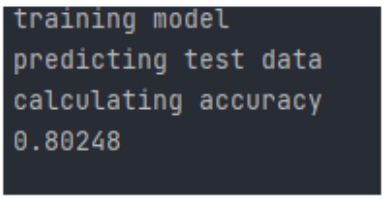
Logistic Regression:

Κι εδώ χωρίσαμε τα δεδομένα στα δύο, πήραμε 25000 test data και 25000 training data.Τρέχουμε τον αλγόριθμο με σταθερό αριθμό Iterations(1000), λέξεων(1000), skip words(100) και με διαφορετικά λ (learning rate) κάθε φορά για να δούμε ποιο είναι το βέλτιστο. Όπως διακρίνουμε από τα παρακάτω στιγμιότυπα τα καλύτερα αποτελέσματα τα παίρνουμε για λ = 0.2

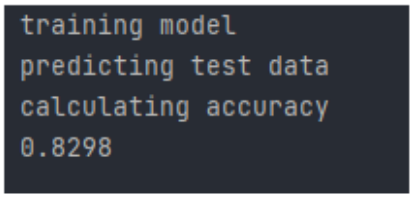
λ = 0.2



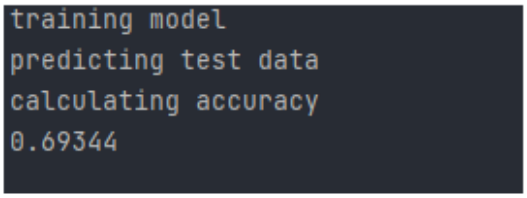
λ = 0.02



λ =0.1



λ = 0.002



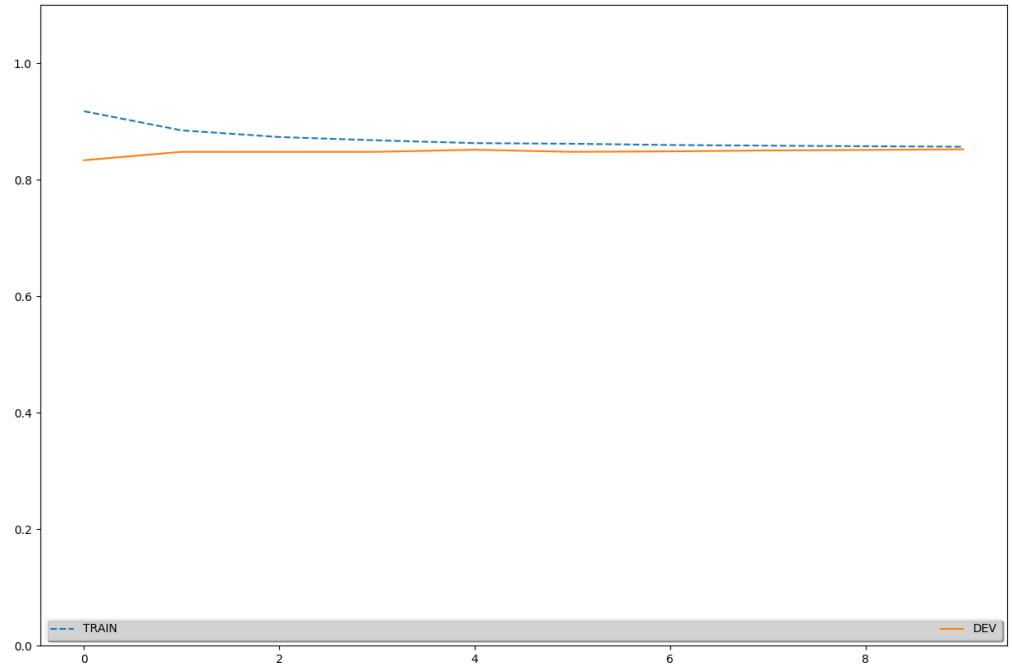
Αφού βρήκαμε το καλύτερο λ για το μοντέλο κάναμε ό,τι κάναμε και στον ID3 για τα Plots.

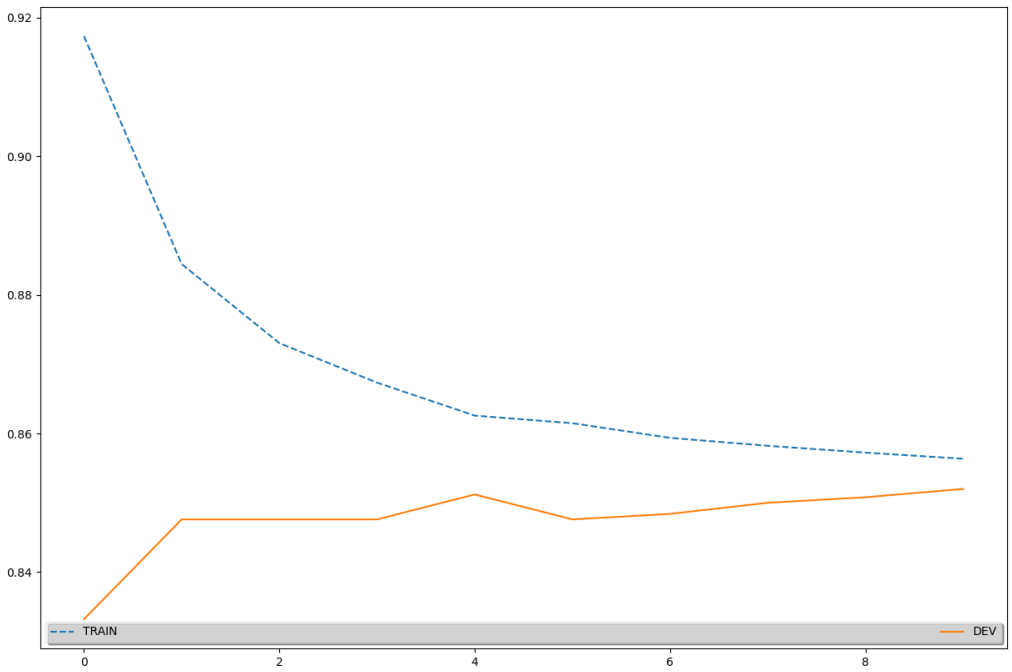
Τα αποτελέσματα ήταν τα εξής:

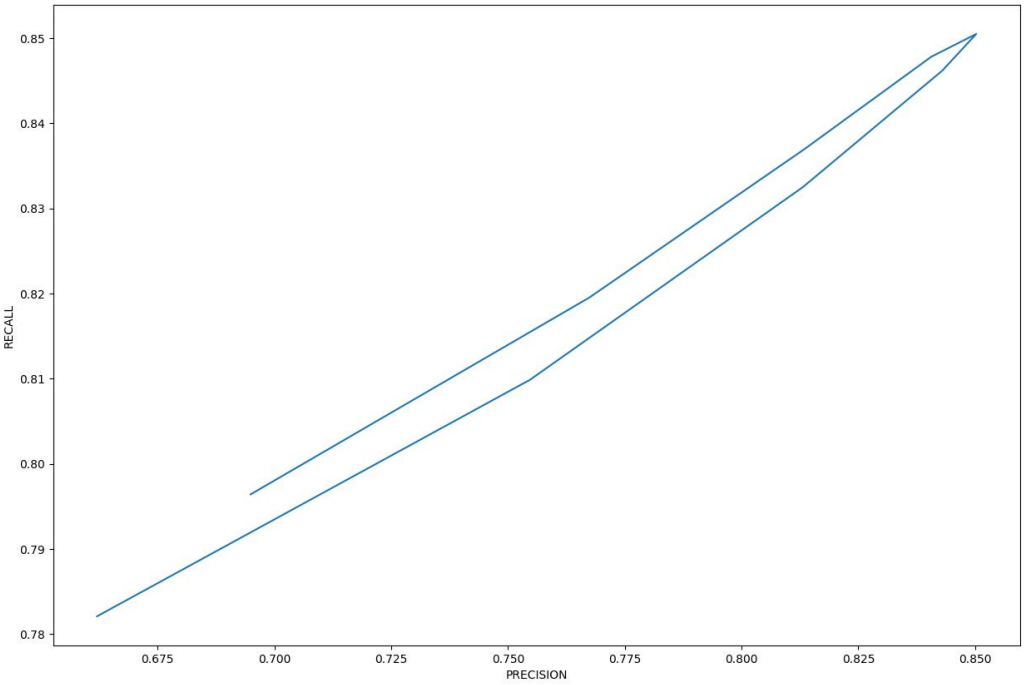
Precision: [0.7964191861421372, 0.8195320984748274, 0.8369280027402078, 0.8478203839722693, 0.850490235447716, 0.8461958608375548, 0.8325026690602599, 0.8098845606822642, 0.7820973451990192]

Recall: [0.6948799999999999, 0.7674000000000001, 0.81336, 0.84064, 0.85024, 0.843, 0.8131200000000001, 0.7547200000000001, 0.662]

F1: [0.742192806391972, 0.7926097568686348, 0.8249757123344205, 0.8442149242562907, 0.8503650993147717, 0.8445949072268759, 0.8226971868986702, 0.781329789006344, 0.7170547667621351]


1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
TRAIN
DEV


0.92
0.90
0.88
0.86
0.84
0
2
4
6
8
TRAIN
DEV



Όσον αφορά τις μεθόδου του μοντέλου της λογιστικής παλινδρόμησης που υλοποιήσαμε έχουμε τις εξής:

- Η μέθοδος sigmoid υλοποιεί την σιγμοειδή συνάρτηση $S(x) = \frac{1}{1+e^{-x}}$:
- Η μέθοδος loss υλοποιεί την συνάρτηση κόστους

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

- Η μέθοδος fit είναι αυτή που εκπαιδεύει το μοντέλο στα δεδομένα εισόδου που δέχεται χρησιμοποιώντας την μέθοδο gradient descent. Στον δοσμένο αριθμό iterations αρχικοποιεί τα θ και στην συνέχεια υπολογίζει το z με την χρήση της σιγμοειδούς. Αφού βρει το z υπολογίζει την κλίση(gradient) και προσαρμόζει τα θ κατάλληλα
- Η μέθοδος predict_prob είναι η μέθοδος που τρέχει το εκπαιδευμένο μοντέλο που έχει δημιουργηθεί στα δεδομένα εισόδου και υπολογίζει την πιθανότητα του.
- Η μέθοδος predict παίρνει την πιθανότητα της παραπάνω συνάρτησης και την στρογγυλοποιεί σε 0 και 1 κατατάσσοντας ουσιαστικά την κριτική στην αντίστοιχη κατηγορία

Random Forest:

Χωρίσαμε τα δεδομένα στα δύο, πήραμε 25000 test data και 25000 training data. Για να φτιάξουμε τα plots τρέξαμε το αλγόριθμο πολλές φορές και κάθε φορά του δίναμε διαφορετικό ποσοστό training data. Έπειτα αποθηκεύσαμε το accuracy που παίρνουμε τεστάροντας τον εκπαιδευμένο αλγόριθμο και στα data που έχει εκπαιδευτεί και στα dev δεδομένα που δεν έχει ξαναδεί, και χρησιμοποιούμε τα αποθηκευμένα accuracies για να κατασκευάσουμε τα plots. Τέλος υπολογίσαμε το precision και το recall για το καλύτερο μοντέλο μας και για διαφορετικά thresholds και φτιάξαμε τα plots.

Οι τιμές των υπερπαραμέτρων επιλέχθηκαν μετά από πολλές δοκιμές.

Accuracy	Number of Words	Skip Words	Max Depth	Number of Trees
0.62328	100	10	10	2
0.71972	500	100	15	3
0.72892	1000	200	20	4
0.73252	1000	200	20	5

Παρατηρούμε ότι με τα ίδια δεδομένα ο random Forest βγάζει λίγο καλύτερα αποτελέσματα από τον ID3.

class RandomForest:

Η κλάση που υλοποιεί τον αλγόριθμο.

__init__(self, n_trees=10, min_samples_split=2,

max_depth=100, n_feats=None):

Αρχικοποιεί την random forest. Τα trees είναι τα δέντρα που θα φτίαξουμε, το min samples split είναι ο λιγότερος αριθμός δεδομένων στον οποίο επιτρέπεται να κάνουμε split, το max depth είναι το μεγαλύτερο μέγεθος του δέντρου και το n_feats είναι ο αριθμός των ιδιοτήτων, δηλαδή οι λέξεις.

fit(self, X, y):

Ορίστηκε στην decision tree. Η διαφορά είναι ότι χρησιμοποιείται για να φτιάξει τα πολλά δέντρα.

predict(self, X):

Ορίστηκε στην decision tree. Η διαφορά είναι ότι

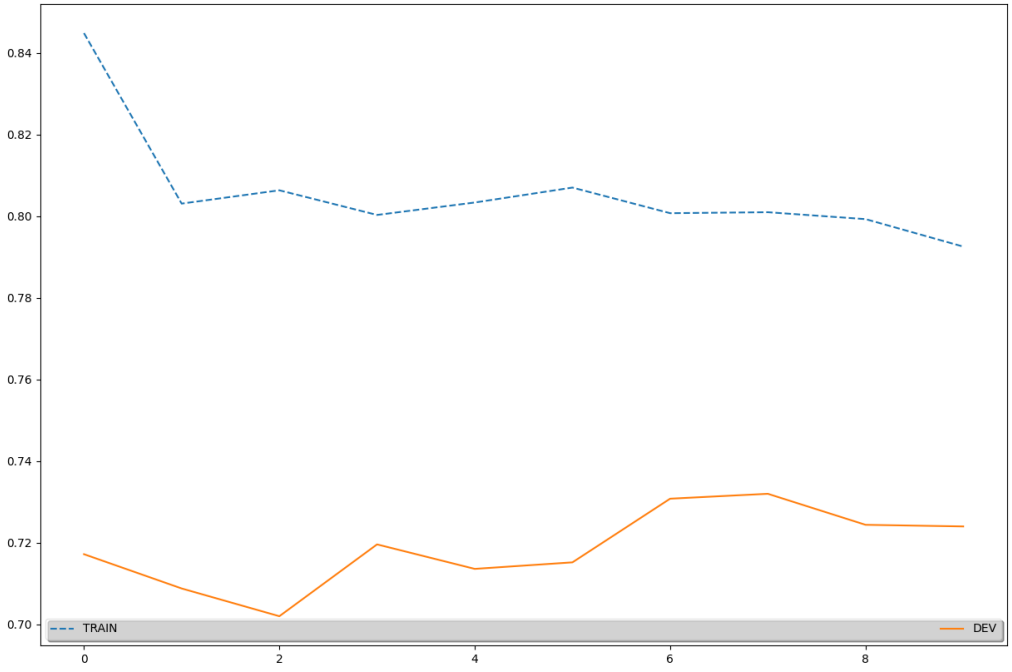
most_common_label(y):

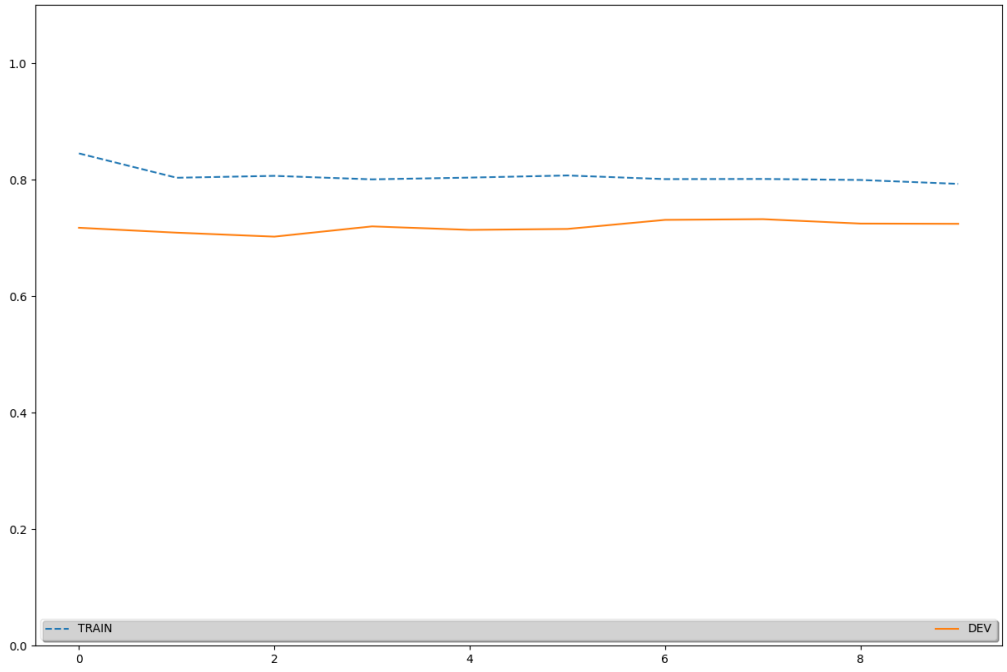
Ορίστηκε στην decision tree.

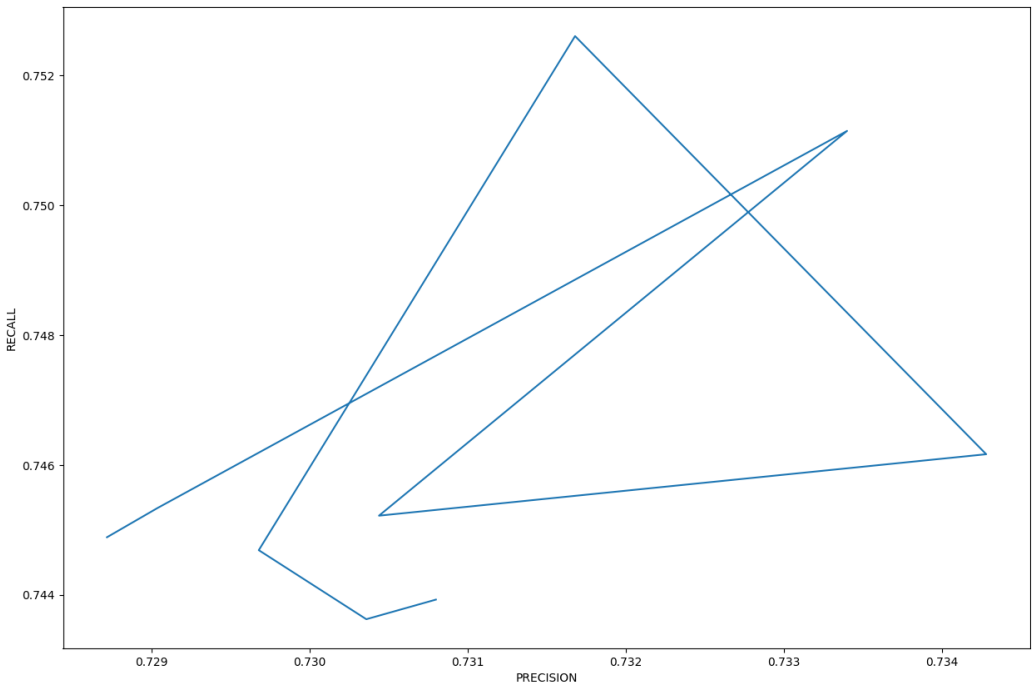
bootstrap_sample(X, y):

Διαλέγει τυχαία ένα υποσύνολο από τις κριτικές για να κατασκευαστεί το κάθε δέντρο.

Τα plots που προκύψανε είναι για 1000 λεξεις, 200 skip words, 20 max depth και 5 δεντρα:




1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
TRAIN
DEV


0.752
0.750
0.748
0.746
0.744
RECALL
0.729
0.730
0.731
0.732
0.733
0.734
PRECISION

Precision: [0.7439292478354979, 0.7436271783552515, 0.7446906409380456, 0.7526091383826077, 0.7461685779502717, 0.7452219993866565, 0.7511489217398164, 0.7453404745127118, 0.7448896842124652]

Recall: [0.7308, 0.73036, 0.72968, 0.73168, 0.73428, 0.73044, 0.7334, 0.72904, 0.72872]

F1: [0.7373061802580131, 0.7369338810525841, 0.737108908427466, 0.7419970141017627, 0.7401765540224383, 0.7377569625811854, 0.7421683598791248, 0.7371001297590265, 0.7367161284222999]