Assignment #1

by: Sirinian Aram Emmanouil AM: 2537

Ερώτηση 1η

Στο ερώτημα αυτό έχουν υλοποιηθεί οι εξής classifiers: Logistic Regression, SVM, Decision Trees και Naïve Bayes. Ο στόχος είναι να γίνει ο διαχωρισμός μεταξύ επιχειρήσεων που ανήκουν στις κατηγορίες “Nightlife” ή “Bars”, και τις επιχειρήσεις που ανήκουν στις κατηγορίες “Cafes”, “Coffee & Tea” ή “Breakfast & Brunch”. Οι classifiers δουλεύουν πάνω στο TFIDF των κειμένων.

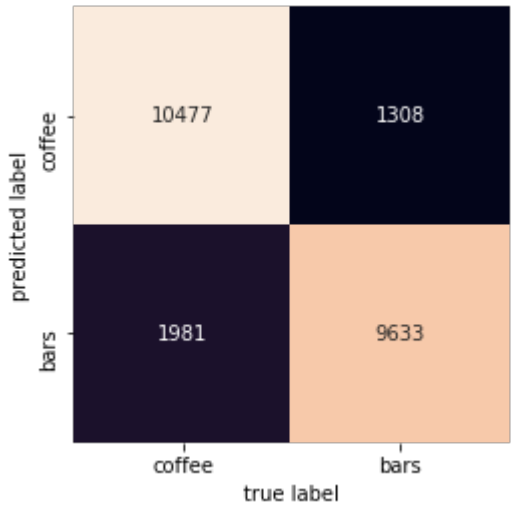
Αρχικά πραγματοποιήστε η εξόρυξη των δεδομένων από τα αρχεία business.json και review.json. Η εξόρυξη αυτή στηρίζεται στις κατηγορίες που έχει κάθε επιχείρηση. Όταν στις κατηγορίες μιας επιχείρησης εμπεριέχονται τα “Cafes” ή “Coffee & Tea” ή “Breakfast & Brunch” μπαίνουν στην μία κατηγορία (Coffee). Όταν στις κατηγορίες μιας επιχείρησης εμπεριέχονται τα “Nightlife” ή “Bars” μπαίνουν στην άλλη (Bars). Έπειτα με τα δεδομένα αυτά επιλέγονται τα δεδομένα που μας ενδιαφέρουνε από το review.json αρχείο και κατασκευάζονται οι λίστες με τα training και testing data που θα χρησιμοποιηθούν στους classifiers.

Τα αποτελέσματα των classifiers είναι τα εξής:

TfidfVectorizer ==> Naïve Bayes Multinomial

accuracy score: 0.85943843754

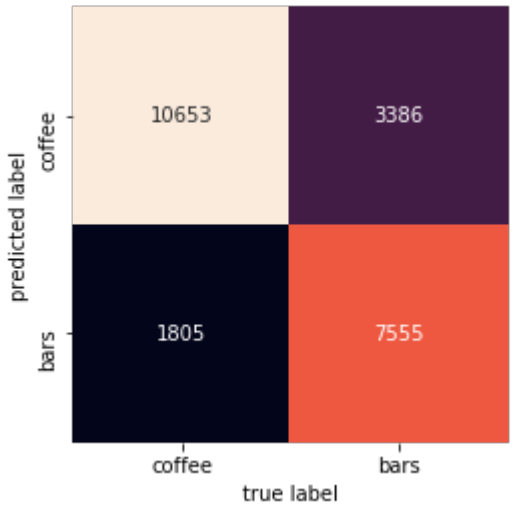
	precision	recall	f1-score	support
0	0.89	0.84	0.86	12458
1	0.83	0.88	0.85	10941
avg / total	0.86	0.86	0.86	23399



TfidfVectorizer ==> Naïve Bayes Bernoulli

accuracy score: 0.778152912518

	precision	recall	f1-score	support
0	0.76	0.86	0.80	12458
1	0.81	0.69	0.74	10941
avg / total	0.78	0.78	0.78	23399



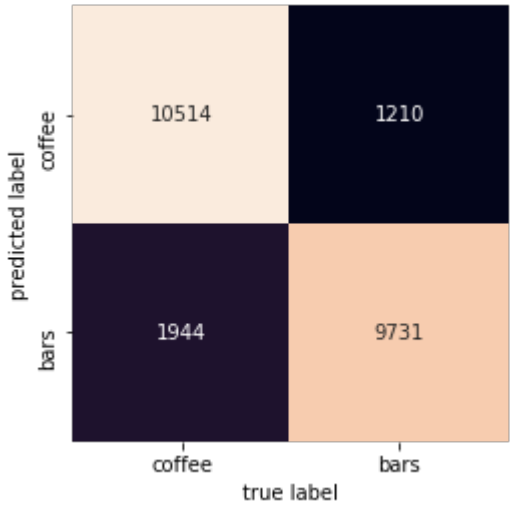
TfidfVectorizer ==> Logistic Regression

Coefficient of the features in the decision function:

[[-0.46393835 -0.00559762 -0.01278525 ..., -0.11904624 0.04989826 -0.08892536]]

```
Intercept (a.k.a. bias) added to the decision function: [-0.12815072]
Actual number of iterations for all classes: [7]
accuracy score:  0.865207914868
```

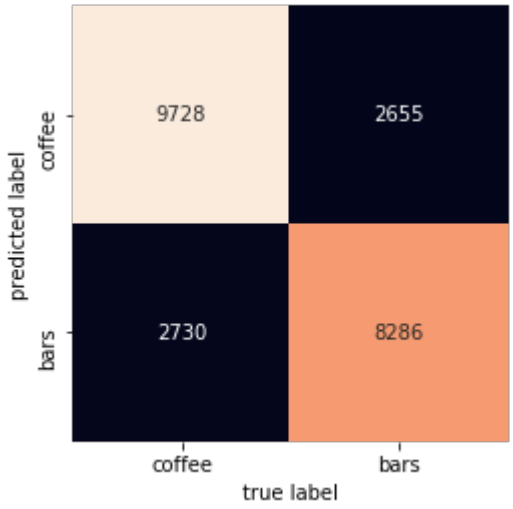
	precision	recall	f1-score	support
0	0.90	0.84	0.87	12458
1	0.83	0.89	0.86	10941
avg / total	0.87	0.87	0.87	23399



TfidfVectorizer ==> Decision Trees

accuracy score: 0.769861959913

	precision	recall	f1-score	support
0	0.79	0.78	0.78	12458
1	0.75	0.76	0.75	10941
avg / total	0.77	0.77	0.77	23399

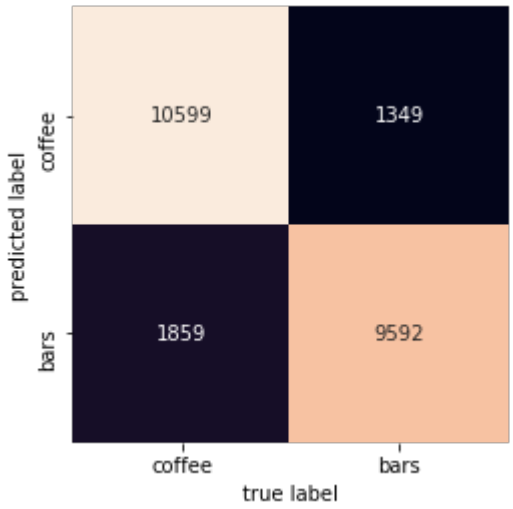


TfidfVectorizer ==> LinearSVC(penalty="l2")

Weights assigned to the features:

[[-0.16446365 -0.12527549 0.   ..., -0.20597106 0.15638964 -0.17507529]]

```
Constants in decision function: [-0.02299581]
accuracy score:  0.862900123937
             precision    recall  f1-score   support

          0       0.89      0.85      0.87     12458
          1       0.84      0.88      0.86     10941

avg / total       0.86 0.86      0.86     23399
```



Μπορούμε να παρατηρήσουμε ότι το καλύτερο accuracy score έχει ο αλγόριθμος (TfidfVectorizer ==> Logistic Regression) 0.865207914868.

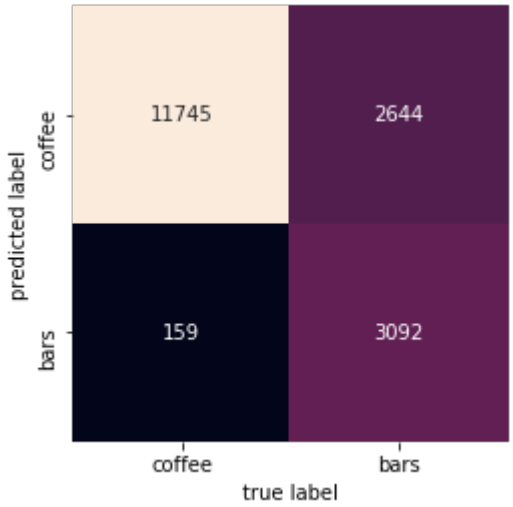
Συνεχίζοντας ακολουθούμε παρόμοια τακτική, το κομμάτι στο οποίο διαφέρει είναι η συλλογή των πληροφοριών. Αρχικά πραγματοποιείτε η εξόρυξη των δεδομένων από τα αρχεία business.json και review.json. Η εξόρυξη αυτή στηρίζεται στις κατηγορίες που έχει κάθε επιχείρηση. Όταν στις κατηγορίες μιας επιχείρησης εμπεριέχονται τα “Cafes” ή “Coffee & Tea” ή “Breakfast & Brunch” ή “Nightlife” ή “Bars” αποθηκεύουμε της υπόλοιπες κατηγορίες που της συνοδεύουν στο σύνολο από κατηγορίες που ανήκουν είτε σε (Coffee) είτε σε (Bars). Έπειτα με τα δεδομένα αυτά (αφού αφαιρέσουμε την τομή τους) επιλέγονται τα δεδομένα που μας ενδιαφέρουνε από το review.json αρχείο και κατασκευάζονται οι λίστες με τα training και testing data που θα χρησιμοποιηθούν στους classifiers.

Τα αποτελέσματα των classifiers είναι τα εξής:

TfidfVectorizer ==> Naïve Bayes Multinomial

accuracy score: 0.841099773243

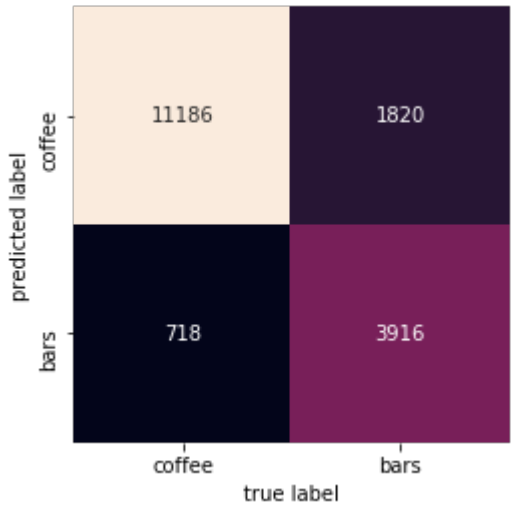
	precision	recall	f1-score	support
0	0.82	0.99	0.89	11904
1	0.95	0.54	0.69	5736
avg / total	0.86	0.84	0.83	17640



TfidfVectorizer ==> Naïve Bayes Bernoulli

accuracy score: 0.85612244898

	precision	recall	f1-score	support
0	0.86	0.94	0.90	11904
1	0.85	0.68	0.76	5736
avg / total	0.86	0.86	0.85	17640

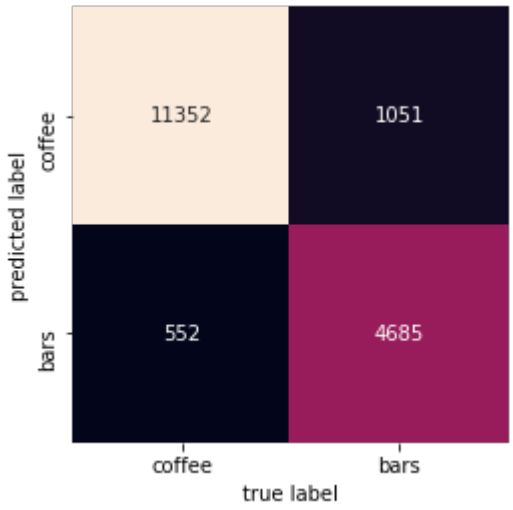


TfidfVectorizer ==> Logistic Regression

Coefficient of the features in the decision function:

[[-0.41106191 0.17889803 -0.0325908 ..., -0.01085912 0.01058974 -0.0570946]]

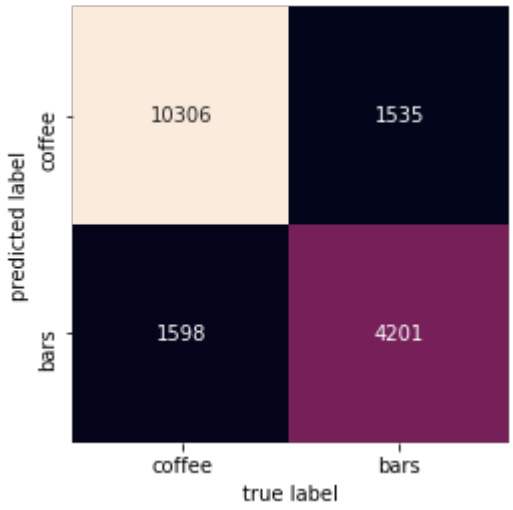
```
Intercept (a.k.a. bias) added to the decision function: [-0.77571593]
Actual number of iterations for all classes: [11]
accuracy score:  0.909126984127
             precision    recall  f1-score   support

          0       0.92      0.95      0.93     11904
          1       0.89      0.82      0.85      5736

avg / total       0.91      0.91      0.91     17640
```



TfidfVectorizer ==> Decision Trees

accuracy score: 0.822392290249

	precision	recall	f1-score	support
0	0.87	0.87	0.87	11904
1	0.72	0.73	0.73	5736
avg / total	0.82	0.82	0.82	17640



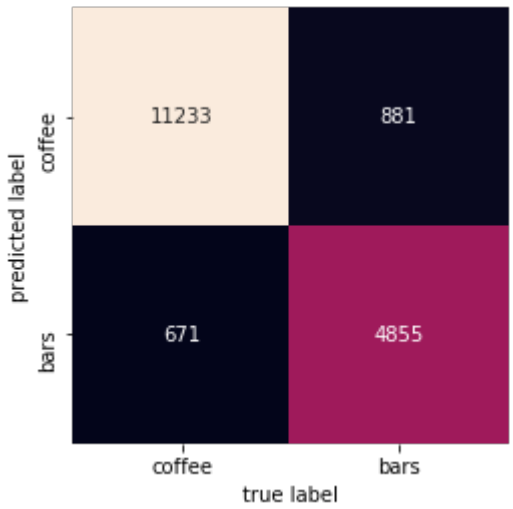
TfidfVectorizer ==> LinearSVC(penalty="l2")

Weights assigned to the features:

[[-0.27874176 0.22571876 -0.04888326 ..., -0.03196314 0.01969902 -0.09743721]]

Constants in decision function: [-0.33023053]
accuracy score: 0.91201814059

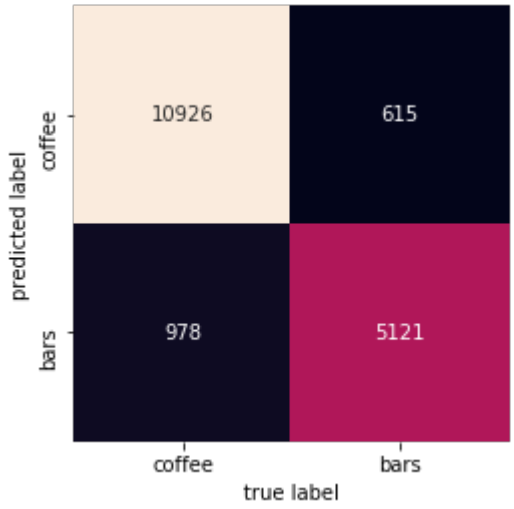
	precision	recall	f1-score	support
0	0.93	0.94	0.94	11904
1	0.88	0.85	0.86	5736
avg / total	0.91	0.91	0.91	17640



TfidfVectorizer ==> LinearSVC(class_weight='balanced')

accuracy score: 0.909693877551

	precision	recall	f1-score	support
0	0.95	0.92	0.93	11904
1	0.84	0.89	0.87	5736
avg / total	0.91	0.91	0.91	17640



Είναι εύκολο εδώ να παρατηρήσουμε ότι στους περισσότερους classifiers το accuracy score βελτιώνεται. Το καλύτερο σκορ έχει ο TfidfVectorizer ==> LinearSVC(penalty="l2") με 0.91201814059.

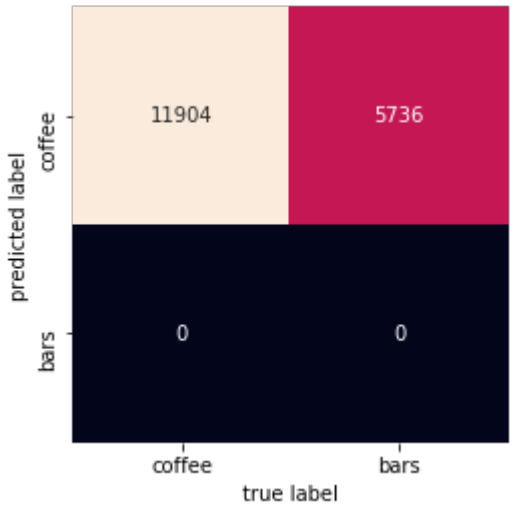
Στην συνέχεια γίνεται μια απόπειρα να παντρέψουμε dimensionality reduction με classification techniques. Πιο συγκεκριμένα χρησιμοποιώντας το TruncatedSVD(n_components=2).

Τα αποτελέσματα των classifiers είναι τα εξής:

TfidfVectorizer ==> TruncatedSVD(n_components=2) ==> Naïve Bayes Bernoulli

accuracy score: 0.674829931973

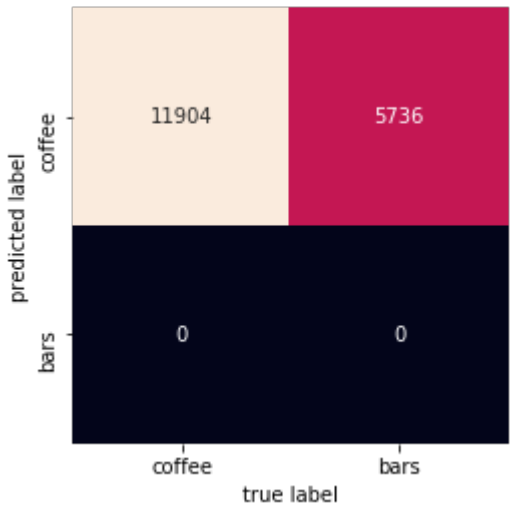
	precision	recall	f1-score	support
0	0.67	1.00	0.81	11904
1	0.00	0.00	0.00	5736
avg / total	0.46	0.67	0.54	17640



TfidfVectorizer ==> TruncatedSVD(n_components=2) ==> Logistic Regression

accuracy score: 0.674829931973

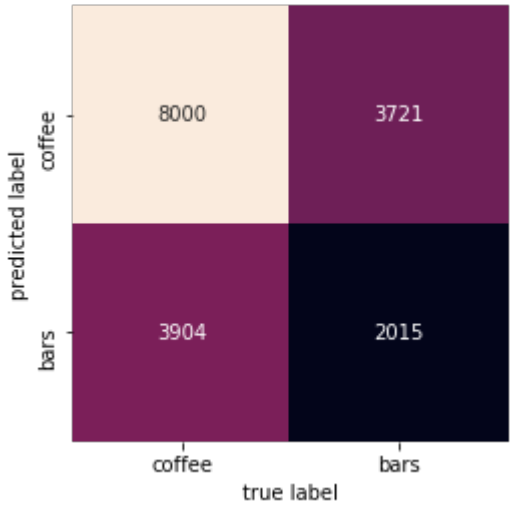
	precision	recall	f1-score	support
0	0.67	1.00	0.81	11904
1	0.00	0.00	0.00	5736
avg / total	0.46	0.67	0.54	17640



TfidfVectorizer ==> TruncatedSVD(n_components=2) ==> Decision Trees

accuracy score: 0.567743764172

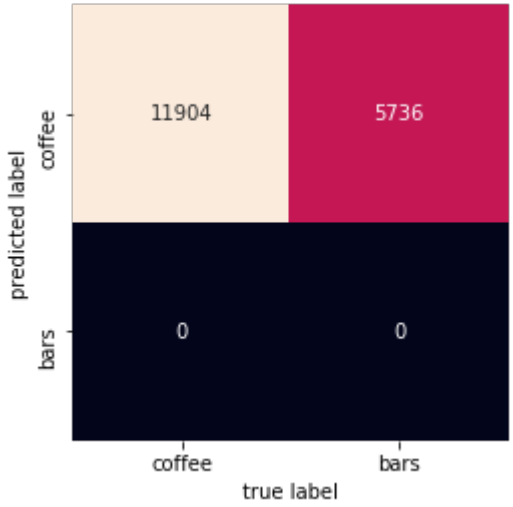
	precision	recall	f1-score	support
0	0.68	0.67	0.68	11904
1	0.34	0.35	0.35	5736
avg / total	0.57	0.57	0.57	17640



TfidfVectorizer ==> TruncatedSVD(n_components=2) ==> LinearSVC(penalty="l2")

accuracy score: 0.674829931973

	precision	recall	f1-score	support
0	0.67	1.00	0.81	11904
1	0.00	0.00	0.00	5736
avg / total	0.46	0.67	0.54	17640



Παρατηρούμε ότι η χρήση dimension reduction techniques επιδρά αρνητικά στους classifiers.

Τέλος μερικά από αυτά που δεν δούλεψαν. Το ποιο σημαντικό πρόβλημα ήταν σχετικό με Decision Trees και ποιο συγκεκριμένα με το graphviz “IOPub data rate exceeded.” κάνοντας έτσι δύσκολη την ανάλυση του τρόπου επιλογής των κόμβων του δέντρου.

Ερώτηση 2η

Για το ερώτημα αυτό πραγματοποιήθηκαν διάφοροι classifiers, το πρόβλημα που προσπαθούν να επιλύσουν όλοι είναι η πρόβλεψη αν ένα review για ένα εστιατόριο (κατηγορίες Food, Restaurants) είναι αρνητικό (έχει 1 ή 2 stars). Ο αλγόριθμος που αποδείχθηκε καλύτερος είναι ο TFIDF with SGD Classifier(loss="modified_huber", penalty="l2") του πειράματος 15 με Score: 0.74172.

Ποιο συγκεκριμένα τα πειράματα που πραγματοποιήθηκαν είναι:

Πείραμα 1: TFIDF with Multinomial Naive Bayes, Score: 0.02572 (file = submission_file.csv)

Πείραμα 2: TFIDF with Logistic Regression, Score: 0.69644 (file = submission_file.csv)

Πείραμα 3: TFIDF with Decision Tree, Score: 0.47521 (file = submission_file.csv)

Πείραμα 4,5: TFIDF with Linear Support Vector Machine, Score: 0.73869 (file = submission_file.csv & submission_file_SVC.csv)

Πείραμα 6: TFIDF with Nearest Centroid, Score: 0.55568 (file = submission_file_NC.csv)

Πείραμα 7: TFIDF with Bernoulli Naive Bayes, Score: 0.53274 (file = submission_file_BNB.csv)

Πείραμα 8: TFIDF with MLP Classifier (alpha = 0.1, hidden_layer_sizes = 2, activation = "logistic"), Score: 0.62324 (file = submission_file_MLP_a0.1.csv)

Πείραμα 9: TFIDF with MLP Classifier (alpha = 0.01, hidden_layer_sizes = 2, activation = "logistic"), Score: 0.73801 (file = submission_file_MLP_a0.01.csv)

Πείραμα 10: TFIDF with MLP Classifier (alpha = 0.1, hidden_layer_sizes = 3, activation = "logistic"), Score: 0.63194 (file = submission_file_MLP_a0.1_lz3.csv)

Πείραμα 11: TFIDF with MLP Classifier (alpha = 0.001, hidden_layer_sizes = 3, activation = "logistic"), Score: 0.70441 (file = submission_file_MLP_a0.001_lz3.csv)

Πείραμα 12: TFIDF with MLP Classifier (alpha = 0.001, hidden_layer_sizes = 2, activation = "logistic"), Score: 0.70062 (file = submission_file_MLP_a0.001_lz2.csv)

Πείραμα 13: TFIDF with MLP Classifier (alpha = 0.001, hidden_layer_sizes = 4, activation = "logistic"), Score: 0.69859 (file = submission_file_MLP_a0.001_lz4.csv)

Πείραμα 14: TFIDF with SGD Classifier(loss="hinge", penalty="l2"), Score: 0.68974 (file = submission_file_SGDClassifier_hinge_l2.csv)

Πείραμα 15: TFIDF with SGD Classifier(loss="modified_huber", penalty="l2"), Score: 0.74172 (file = submission_file_SGDClassifier_modified_huber_l2.csv)

Πείραμα 16: TFIDF with SGDClassifier(loss="log", penalty="l2"), Score: 0.57268 (file = submission_file_SGDClassifier_log_l2.csv)

Πείραμα 17: TFIDF with SGD Classifier(loss="modified_huber", penalty="l2", average=True), Score: 0.72557 (file = submission_file_SGDClassifier_modified_huber_l2_avTrue.csv)

Πείραμα 18: TFIDF with Bagging Classifier(SGD Classifier(loss="modified_huber", penalty="l2")), Score: 0.72503 (file = submission_file_BaggingClassifierxSGDClassifier_modified_huber_l2.csv)

Πείραμα 19: TFIDF with Bagging Classifier(LinearSVC(penalty="l2")), Score: 0.73869 (file = submission_file_BaggingClassifier_LinearSVC_l2.csv)

Πείραμα 20: TFIDF with Linear SVC(penalty="l2", multi_class='crammer_singer'), Score: 0.74021 (file = submission_file_LinearSVC_l2_crammer_singer.csv)

Πείραμα 21: TFIDF with MLP Classifier(hidden_layer_sizes = 5, activation = "logistic", alpha = 0.01), Score: 0.73548 (file = submission_file_MLPClassifier_0.01_5_logistic.csv)

Πείραμα 22: TFIDF with Bagging Classifier(Logistic Regression()), Score: 0.68410 (file = submission_file_BaggingClassifier_LogisticRegression.csv)

Πείραμα 23: TFIDF with MLPClassifier(hidden_layer_sizes = 7, activation = "logistic", alpha = 0.001), Score: 0.70942 (file = submissioen_fil_MLPClassifier_0.001_7_logistic.csv)

Πείραμα 24: TFIDF with Perceptron(alpha=0.001), Score: 0.69917 (file = submission_file_Perceptron_0.001.csv)

Πείραμα 25: TFIDF with Perceptron(alpha=0.01), Score: 0.69917 (file = submission_file_Perceptron_0.01.csv)

Πείραμα 26: TFIDF with Perceptron(alpha=0.01, penalty="l2"), Score: 0.50192 (file = submission_file_Perceptron_0.01_l2.csv)

Πείραμα 27: TFIDF with LogisticRegressionCV(), Score: 0.72738 (file = submission_file_Perceptron_0.01_l2.csv)

Ο τρόπος που γινόταν η επιλογή και βελτιστοποίηση των πειραμάτων άλλαζε κατά την διάρκεια της παραγωγής τους. Αρχικά η επιλογή βασιζόταν κυρίως σε αλγορίθμους που είχαν συζητηθεί εκτενώς στις διαλέξεις όπως οι Naive Bayes, Logistic Regression, Decision Tree, Support Vector Machine. Ένας από αυτούς του αλγορίθμους είναι και ο Kneighbors ο οποίος όμως δεν δούλευε για τα συγκεκριμένα δεδομένα, αλλά ένας άλλος αλγόριθμος που του μοιάζει δούλεψε Nearest Centroid. Από όλους αυτούς ο πιο καλός απεδείχθη να είναι το Linear Support Vector Machine. Στην συνέχεια χρησιμοποιήθηκαν άλλες τεχνικές όπως τα MLP, SVC, Perceptron, SGD και Bagging πολλά από τα οποία είχαν καλύτερα σκορ από τα πρώτα μοντέλα. Με βέλτιστο τον SGD Classifier(loss="modified_huber", penalty="l2").

Το όνομα του λογαριασμού στο kaggle είναι Yutu (αλλά μπορεί να εμφανίζεται και ως Sirinian Aram Emmanouil), θέση 5η και το σκορ 0.74172.

Ερώτηση 3η

Για ένα μη κατευθυνόμενο γράφο η κατανομή σύγκλισης (stationary distribution) ενός τυχαίου περίπατου είναι ανάλογη του βαθμού του κάθε κόμβου.

ΑΠΟΔΕΙΞΗ

P : πίνακας μετάβασης

Π : κατανομή σύγκλισης

d(i) : βαθμός του κόμβου I

Για να δείξουμε ότι δύο μεγέθη είναι ανάλογα αρκεί να δείξουμε ότι η τιμή του ενός είναι πολλαπλάσια του άλλου, δηλαδή οι αντίστοιχες τιμές των δύο μεγεθών έχουν σταθερό λόγο:

$\Pi i = ki \cdot di$ όπου ki είναι μια σταθερά

Ισχύει ότι $\Pi = \Pi \cdot P$

Παρατηρούμε ότι μη κατευθυνόμενο γράφημα ισχύει ότι $Pi, j = \frac{1}{d(i)}$ αν j και i συνδεδεμένα, 0 αλλιώς.

Έστω ότι $\Pi i = ki \cdot d(i)$:

$$(P \times \Pi) j = \sum_{j} (Pi, j) \cdot \Pi i = \sum_{j:(i,j) \in E} d(j) \cdot kj \cdot (\frac{1}{d(j)}) = \sum_{j:(i,j) \in E} kj = \Pi j \quad \text{για} \quad \forall i, j \in N$$

Δηλαδή $\Pi = \Pi \cdot P$ Άρα ισχύει

Εναλλακτικός τρόπος να το δούμε αυτό:

Έστω ότι $\Pi i = ki \cdot d(i)$ δηλαδή: $\Pi = (k1 \cdot d(1), k2 \cdot d(2), \ldots, kn \cdot d(n))$

$$\Pi \cdot P = (k1 \cdot d(1), k2 \cdot d(2), \ldots, kn \cdot d(n)) \times ¿$$

$$¿\left((0, \frac{1}{d(1)}, \frac{1}{d(1)}, \ldots, 0), (\frac{1}{d(2)}, \frac{1}{d(2)}, 0, \ldots, \frac{1}{d(2)}), (\frac{1}{d(3)}, 0, 0, \ldots, \frac{1}{d(3)}), \ldots (0, \frac{1}{d(n)}, \frac{1}{d(n)}, \ldots, \frac{1}{d(n)})\right)$$

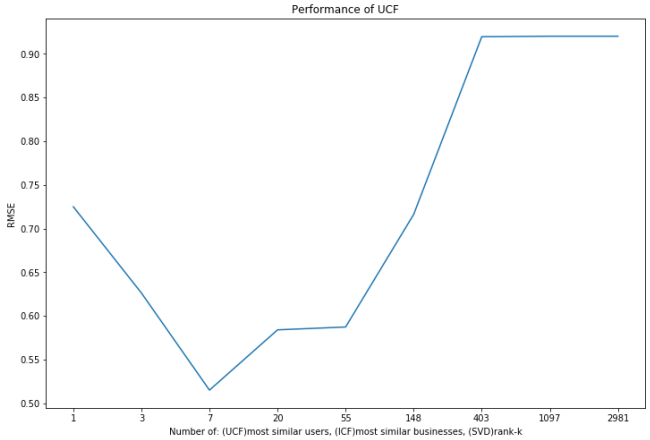
$= \quad (k1 \cdot d(1), k2 \cdot d(2), \ldots, kn \cdot d(n)) = \Pi$ Άρα ισχύει

Ερώτηση 4η

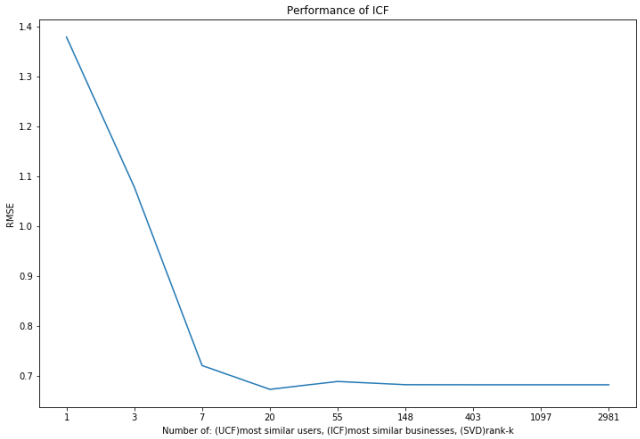
Σε αυτό το ερώτημα υλοποιήθηκε η πρόβλεψη των ratings των χρηστών του YELP για νέες επιχειρήσεις, χρησιμοποιώντας τα δεδομένα που δημιουργήθηκαν στην Δεύτερη Σειρά Ασκήσεων (Ερώτηση 3). Οι προβλέψεις αυτές πραγματοποιούνε με την τεχνική για την διάχυση(propagation) τιμών. Τα δεδομένα αυτά περιλαμβάνουν τις επιχειρήσεις στο Toronto που έχουν τουλάχιστον 10 κριτικές από χρήστες με τουλάχιστον 10 κριτικές. Χρησιμοποιώντας αυτούς τους χρήστες ως κορυφές, δημιουργείτε ένα γράφημα με ακμές τις φιλίες μεταξύ των χρηστών (από το αρχείο user.json). Από το γράφημα αυτό κρατιέται η μεγαλύτερη συνεκτική συνιστώσα (το γράφημα G).

Αφαιρώντας τυχαία το 10% των ratings χρηστών για τις προβλέψεις, υπολογίζετε το Root Mean Square Error (RMSE) για τις μεθόδους: Absorbing Random Walks (ARW), User Average (UA), Business Average (BA), User-based Collaborative Filtering (UCF), Item-based Collaborative Filtering (ICF). Για το Root Mean Square Error (RMSE) του Singular Value Decomposition (SVD) χρησιμοποιούνται όλα τα rating.

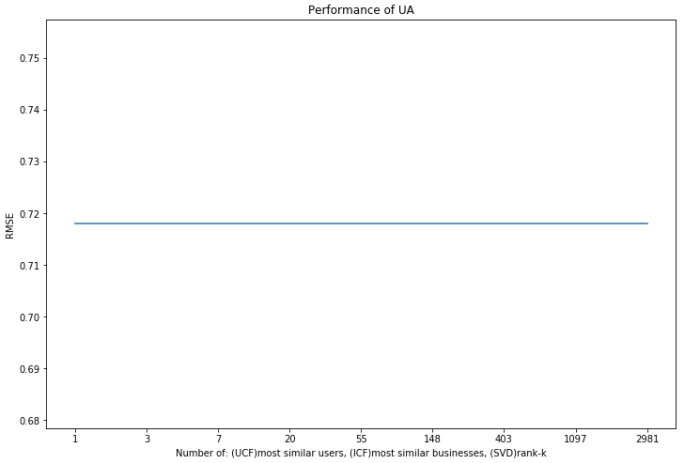
(UCF)



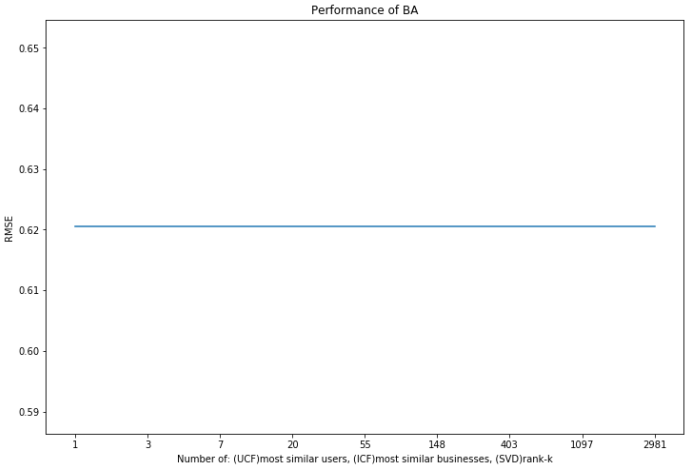
(ICF)



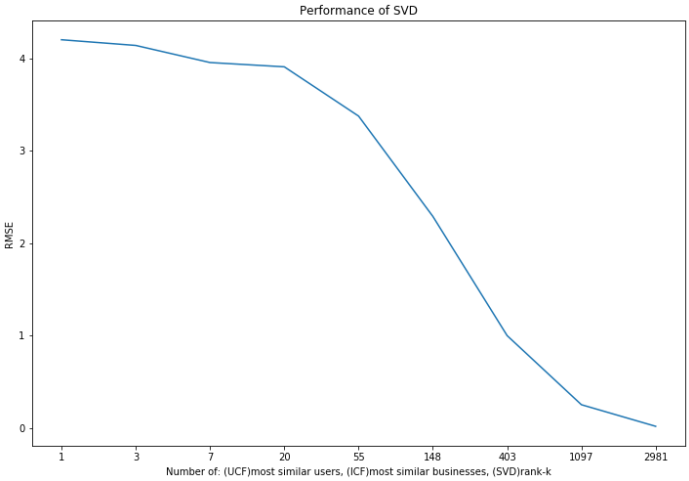
(UA)



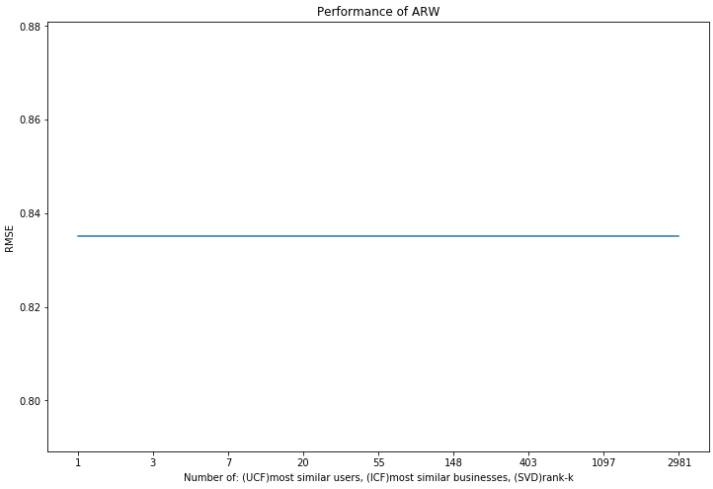
(BA)

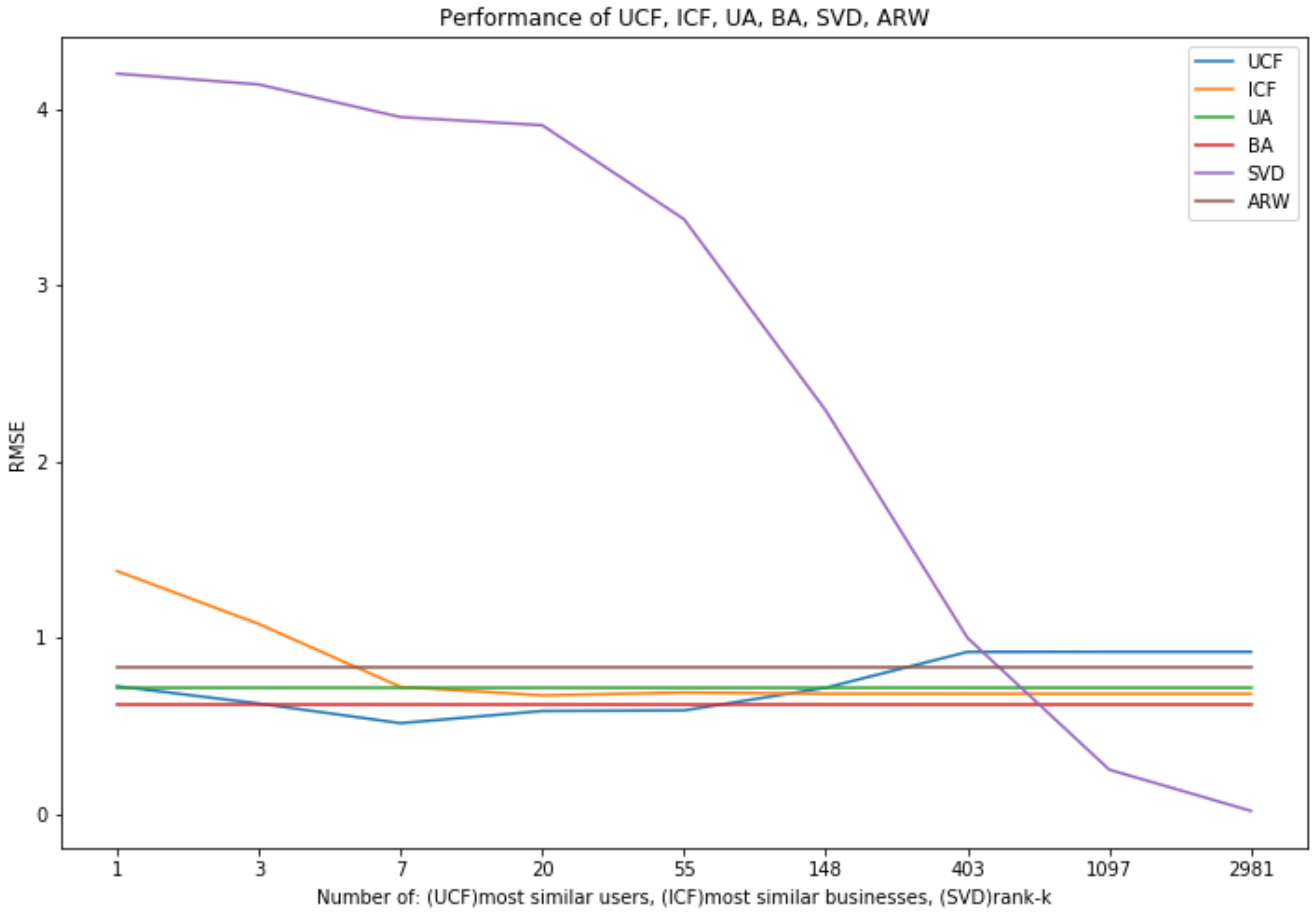


(SVD)



(ARW)





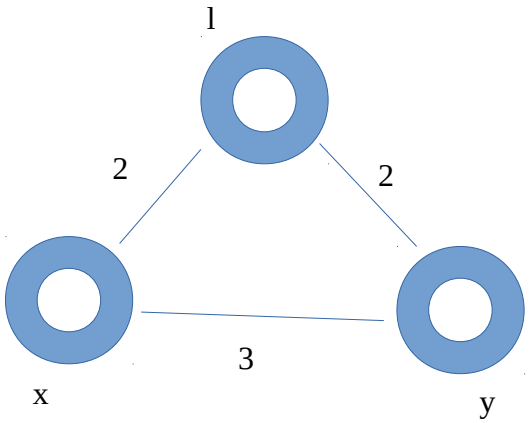
Αυτό που παρατηρούμε είναι ότι η μέθοδος (ARW) δεν επιτυγχάνει πολύ χειρότερα αποτελέσματα από τους υπόλοιπους αλγορίθμους. Αρχικά η μέθοδος (ARW) δεν επηρεάζεται από την τιμή k όπως κάνουν οι αλγόριθμοι (ICF) και (UFC), για αυτό και η γραφική παράσταση της είναι παρόμοια με αυτές των μεθόδων (UA), (BA) και (SVD).

Ερώτημα 5

1) Για κάθε landmark $l \in L$, ισχύει ότι $d(x,y) \leq d(x,l) + d(l,y)$:

ΑΠΟΔΕΙΞΗ
Έστω ότι ισχύει $d(x,y) > d(x,l) + d(l,y)$. Εξορισμού d(x, l) είναι το μήκος του πιο σύντομου μονοπατιού από το x στο l και d(l, y) είναι το μήκος του πιο σύντομου μονοπατιού από το l στο y. Αλλά d(x, y) είναι το μήκος του πιο σύντομου μονοπατιού από το x στο y, επομένως d(x, l) + d(l, y) δεν μπορεί να είναι μικρότερο του d(x, y). Άρα $d(x,y) \leq d(x,l) + d(l,y)$

Αυτό μπορούμε να το διαπιστώσουμε και σχηματικά:



d(x, y) = 3, d(x, l) = 2, d(l, y) = 2

d(x, y) <= d(x, l) + d(l, y) ισχύει

d(x, y) > d(x, l) + d(l, y) άτοπο

Η ισότητα στην σχέση $d(x,y) \leq d(x,l) + d(l,y)$ ισχύει όταν υπάρχει στο ποιο σύντομο μονοπάτι από το x στο y, ο κόμβος l (landmark). Δηλαδή όταν το landmark βρίσκετε στο συντομότερο μονοπάτι x->y.

2) Το πρόβλημα του να βρούμε το μικρότερο σύνολο L από σημεία αναφοράς ώστε να μπορούμε να απαντήσουμε ακριβώς shortest path queries για οποιοδήποτε ζευγάρι από κορυφές (x, y) μπορεί να εκφραστεί ως ένα πρόβλημα ελάχιστης κάλυψης (minimum set cover):

ΑΠΟΔΕΙΞΗ
Παρατηρούμε ότι λόγο της ιδιότητας του να απαντάμε ακριβώς με shortest path queries, μπορούμε να διαιρέσουμε το πρόβλημα αυτό σε δύο υποπροβλήματα. Το πρώτο πρόβλημα είναι η εύρεση του μικρότερου συνόλου από συντομότερες διαδρομές. Το δεύτερο η επιλογή σημείων αναφοράς με βάση τις συντομότερες διαδρομές που βρήκαμε, ολοκληρώνοντας με αυτόν τον τρόπο το μικρότερο σύνολο L.

Το πρόβλημα εύρεσης του μικρότερου συνόλου από συντομότερες διαδρομές μπορεί να εκφραστεί ως πρόβλημα ελάχιστης κάλυψης με τον εξής τρόπο:

1)Το σύνολο από στοιχεία U είναι το σύνολο κόμβων V του γραφήματος.
2)Το σύνολο από υποσύνολα του U, S = {S1, …, Sn}, τέτοιο ώστε η ένωση των στοιχείων του ισούται με το U. Κάθε set του S αντιστοιχεί σε ένα συντομότερο μονοπάτι (shortest path) του γραφήματος.

3)Θέλουμε να βρούμε το μικρότερο υποσύνολο υποσυνόλων $C \subseteq S$ από το S, τέτοιο ώστε η ένωση όλων των στοιχείων του C ισούται με το U

Και με αυτόν τον τρόπο βρίσκουμε το μικρότερο σύνολο από συντομότερες διαδρομές που καλύπτουν όλους τους κόμβους του γραφήματος μας. Η επιλογή των κόμβων landmarks για την δημιουργία του συνόλου L δεν είναι δύσκολη υπόθεση αλλά εξαρτάτε κυρίως από την μορφή του αρχικού μας γραφήματος, δηλαδή αν το γράφημα μας είναι κατευθυνόμενο ή όχι. Αν υποθέσουμε ότι το γράφημα είναι μη κατευθυνόμενο τότε αρχικά φτιάχνουμε υπογραφήματα από το σύνολο C (ενώνουμε μονοπάτια αν έχουν κοινούς κόμβους) και επιλέγουμε από το κάθε ένα γράφημα έναν κόμβο ως landmark. Αν το γράφημα είναι κατευθυνόμενο τότε φτιάχνουμε όπως και ποιο πριν υπογραφήματα από το σύνολο C (ενώνουμε μονοπάτια αν έχουν κοινούς κόμβους) και επιλέγουμε από το κάθε ένα γράφημα κόμβους ως landmark, αν υπάρχει κόμβοι με indegree = 0 διαλέγουμε εκείνους αλλιώς αν είναι συνεκτικό διαλέγουμε έναν οποιοδήποτε κόμβο.

Με αυτόν τον τρόπο εκφράζουμε το πρόβλημα εύρεσης συνόλου L ως πρόβλημα ελάχιστης κάλυψης.

Ερώτηση 6η

Για το ερώτημα αυτό πραγματοποιήθηκαν διάφοροι classifiers, το πρόβλημα που προσπαθούν να επιλύσουν όλοι είναι η πρόβλεψη «τοξικού» περιεχομένου σε συζητήσει. Ο αλγόριθμος που αποδείχθηκε καλύτερος είναι ο TFIDF with LinearSVC(penalty="l2", multi_class='crammer_singer') του πειράματος 4 με Score: 0.7662.

Ποιο συγκεκριμένα τα πειράματα που πραγματοποιήθηκαν είναι:

Πείραμα 1: TFIDF with LinearSVC(penalty="l2"), Score: 0.7541 (file = submission_ex6_LinearSVC_l2.csv)

Πείραμα 2: TFIDF with SGDClassifier(loss="modified_huber", penalty="l2"), Score: 0.6917 (file = submission_ex6_SGDClassifier_modified_huber_l2.csv)

Πείραμα 3: TFIDF with LinearSVC(penalty="l2") with train4.csv, Score: 0.7483 (file = submission_ex6_LinearSVC_l2_modified.csv)

Πείραμα 4: TFIDF with LinearSVC(penalty="l2", multi_class='crammer_singer'), Score: 0.7662 (file = submission_ex6_LinearSVC_l2_crammer_singer.csv)

Πείραμα 5: TFIDF with LinearSVC(penalty="l2", multi_class='crammer_singer') with train4.csv, Score: 0.7577 (file = submission_ex6_LinearSVC_l2_crammer_singer_modified.csv)

Πείραμα 6: TFIDF with DecisionTreeClassifier(random_state=0), Score: 0.7193 (file = submission_ex6_DecisionTreeClassifier.csv)

Πείραμα 7: TFIDF with ExtraTreeClassifier(), Score: 0.6585 (file = submission_ex6_ExtraTreeClassifier.csv)

Πείραμα 8: TFIDF with RandomForestClassifier(), Score: 0.5959 (file = submission_ex6_RandomForestClassifier.csv)

Πείραμα 9: TFIDF with MLPClassifier(), Score: 0.7598 (file = submission_ex6_MLPClassifier.csv)

Πείραμα 10: TFIDF with LogisticRegression(), Score: 0.7131 (file = submission_ex6_LogisticRegression.csv)

Πείραμα 11: TFIDF with LogisticRegressionCV(), Score: 0.7486 (file = submission_ex6_LogisticRegressionCV.csv)

Πείραμα 12: TFIDF with BernoulliNB(), Score: 0.6099 (file = submission_ex6_BernoulliNB.csv)

Πείραμα 13: TFIDF with Perceptron(alpha=0.1), Score: 0.7607 (file = submission_ex6_Perceptron_0.1.csv)

Πείραμα 14: TFIDF with Perceptron(alpha=0.01), Score: 0.7607 (file = submission_ex6_Perceptron_0.01.csv)

Πείραμα 15: TFIDF with Perceptron(alpha=0.001, penalty="l2"), Score: 0.6266 (file = submission_ex6_Perceptron_0.001_l2.csv)

Πείραμα 16: TFIDF with MLPClassifier(hidden_layer_sizes = 2, activation = "logistic", alpha = 0.1), Score: 0.5682 (file = submission_ex6_MLPClassifier_0.1_lz2_logistic.csv)

Πείραμα 17: TFIDF with MLPClassifier(hidden_layer_sizes = 2, activation = "logistic", alpha = 0.01), Score: 0.6975 (file = submission_ex6_MLPClassifier_0.01_lz2_logistic.csv)

Πείραμα 18: TFIDF with MLPClassifier(hidden_layer_sizes = 4, activation = "logistic", alpha = 0.1), Score: 0.5715 (file = submission_ex6_MLPClassifier_0.1_lz4_logistic.csv)

Ο τρόπος που γινόταν η επιλογή και βελτιστοποίηση των πειραμάτων άλλαζε κατά την διάρκεια της παραγωγής τους. Αρχικά η επιλογή βασιζόταν κυρίως σε αλγορίθμους που είχαν συζητηθεί εκτενώς στις διαλέξεις όπως οι Naive Bayes, Logistic Regression, Decision Tree, Support Vector Machine. Παρόλο αυτό δόθηκε μεγάλη προσοχή και σε τεχνικές που είχαν χρησιμοποιηθεί και από παλιότερα προβλήματα παρόμοιο με αυτό εδώ. Ποιο συγκεκριμένα αναφέρομαι στο πρόβλημα που συναντήσαμε στο Assignment3 Exercise2. Επίσης έγινε μια προσπάθεια ανακατασκευής των δεδομένων (training data) χωρίς χαρακτήρες και συμβολοσειρές που γνωρίζουμε ότι δεν θα μας χρειαστούν για να βελτιωθεί η επίδοση των τεχνικών μας (train4.csv). Για όσους αλγορίθμους βασίζονται πολύ στα χαρακτηριστικά που τους δίνουμε (πέρα από τα γενικά χαρακτηριστικά που έχουν όλα τους) έγιναν περισσότερα περάματα αλλάζοντας τα. Από όλους αυτούς ο πιο καλός απεδείχθη να είναι το LinearSVC(penalty="l2", multi_class='crammer_singer'). Στην συνέχεια χρησιμοποιήθηκαν άλλες τεχνικές όπως τα MLP, Perceptron, SGD και Logistic Regression πολλά από τα οποία είχαν καλύτερα σκορ από τα πρώτα μοντέλα. Αξίζει να τονισθεί ότι λόγο των πολλών labels που προσπαθούσαμε να κατατάξουμε τα δεδομένα μας, ο κύριος τρόπος προσέγγισης ήταν να φτιάχνουμε ένα μοντέλο για κάθε label(toxic, severe_toxic, obscene, threat, insult, identity_hate) ξεχωριστά, δηλαδή να έχουμε ένα μοντέλο να εκπαιδεύετε να κατηγοριοποιεί για ένα μόνο label. Παρόλο αυτά σε κάποια μοντέλα που υλοποιήθηκαν, δινόντουσαν όλα τα labels μαζί, τα μοντέλα που υποστηρίζουν αυτήν την δυνατότητα είναι σαφώς λιγότερα. Δύο πράγματα που μπορούμε να παρατηρήσουμε με μεγάλη ευκολία είναι το γεγονός ότι η ανακατασκευής των δεδομένων (training data) χωρίς χαρακτήρες και συμβολοσειρές που γνωρίζουμε ότι δεν θα μας χρειαστούν τελικά χειροτέρευε την απόδοση των αλγορίθμων. Επίσης το μοντέλο του Perceptron φαίνεται να κάνει καλύτερη κατηγοριοποίηση από των MLPClassifier αν και είναι στην ουσία μια πολύ ποιο απλή του εκδοχή. Υπάρχουν πειράματα που δεν μελετήσαμε όσο θα έπρεπε, αρχικά θα μπορούσαμε να χρησιμοποιήσουμε τα μοντέλα μας για περισσότερες δοκιμές με διαφορετικά χαρακτηριστικά (πχ. MLPClassifier(hidden_layer_sizes = 6, activation = "logistic", alpha = 0.01)). Ένα σημαντικό ελάττωμα είναι η έλλειψη εύρεσης κάποιας σχέσης μεταξύ των labels μεταξύ τους ή με το text του training data. Η εύρεση κάποιας σχέσης θα μπορούσε να μας κατευθύνει σε καλύτερα μοντέλα με λιγότερο κόπο. Δεν έγινε πολύ μελέτη σε μοντέλα που δεν μπορούσαν να τρέξουν και ο λόγος που δεν τρέξανε παραμένει άγνωστος. Τέλος δεν έγινε καμία ανάμιξη μεταξύ των μοντέλων, θα μπορούσαμε να έχουμε κάποια μοντέλα να εκπαιδεύονται για κάποια labels και κάποια άλλα για τα υπόλοιπα.

Το όνομα του λογαριασμού στο kaggle είναι Yutu (αλλά μπορεί να εμφανίζεται και ως Sirinian Aram Emmanouil), θέση 2.301η και το σκορ 0.7662.