Στέφανος Κανελλόπουλος 1115201200050

Χανιωτάκης – Ψύχος Χαρίδημος 1115201200194

1η Άσκηση Τεχνικές Εξόρυξης Δεδομένων

Η άσκηση ολοκληρώθηκε με επιτυχία, απατώντας σε όλα τα ζητούμενα ερωτήματα. Η υλοποίηση έγινε με Python 3.6 χρησιμοποιώντας την πλατφόρμα Anaconda (Spyder). Ακολουθεί σύντομη περιγραφή σχετικά με τις τεχνικές υλοποίηση που ακολουθήσαμε.

WordCloud

Για την δημιουργία των word cloud χρησιμοποιήσαμε την βιβλιοθήκη word cloud (https://github.com/amueller/word_cloud).

Κατασκευάσαμε ένα εμπλουτισμένο set από stop words, ενώνοντας τα STOPWORDS από την βιβλιοθήκη wordcloud που αναφέραμε παραπάνω και το text.ENGLISH_STOP_WORDS της βιβλιοθήκης sklearn.

Στην συνέχεια παίρνουμε το Content της κάθε κατηγορίας και κάνουμε generate ένα wordcloud αφού πρώτα έχουμε αφαιρέσει τα stop words.

Τελος, αποθηκευουμε στον τρεχοντα φακελο τις παραγομενες εικονες.

Clustering

Αρχικά κάνουμε fetch απο το train_set.csv το Content και το τοποθετούμε σε μια λίστα. Στην συνέχεια με την χρήση του TfidfVectorizer από την βιβλιοθήκη sklearn την μετατρέπουμε σε διάνυσμα, που αποτελείται από τις 1000 πιο συχνά εμφανιζόμενες λέξεις σε κάθε κείμενο. Το επόμενο βήμα είναι να μειώσουμε τις διαστάσεις του vector σε 500 με την χρήση της TruncatedSVD από την βιβλιοθήκη sklearn. Χρησιμοποιώντας Pipeline, εφαρμόζουμε τα παραπάνω στα δεδομένα που διαβάσαμε από το αρχείο για να παράγουμε τα καινούργια διανύσματα. Για το clustering χρησιμοποιήσαμε τον K – Means της βιβλιοθήκης nltk, με ορίσματα το πλήθος των clusters (5), συνάρτηση απόστασης την cosine distance και κάναμε fit τα διανύσματα που πήραμε από το Pipeline. Τέλος, καταγράψαμε τα στατιστικά του κάθε cluster ξεχωριστά και τα γράψαμε στο output csv αρχείο (clustering_KMeans.csv).

Classification

Υλοποίηση του KNN-Classifier

Αρχικά, η συνάρτηση μας δέχεται σαν ορίσματα τα εξής,

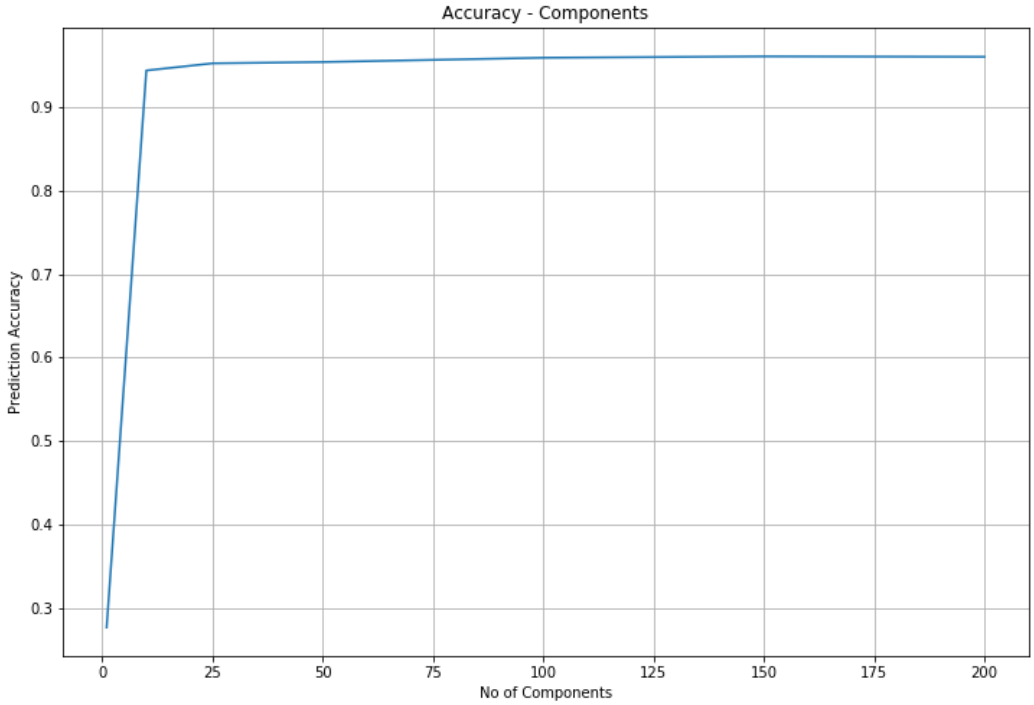
- k : ο αριθμός των neighbors.
- X_train: είναι το vectorized content και ο τίτλος του κειμένου με το οποίο θα γίνει training.
- X_test: είναι το vectorized content και ο τίτλος του κειμένου για το οποίο θα γίνει predict.
- Y_train: είναι οι κατηγορίες στις οποίες ανήκουν τα κείμενα του X_train.

Ο classifier εκτελεί την ακόλουθη διαδικασία. Βρίσκει την απόσταση από το κάθε στοιχείο του X_test (euclidean distance) προς όλα τα στοιχεία του X_train, και δημιουργεί την λίστα distances που περιέχει ταξινομημένες τις αποστάσεις από τη μικρότερη στη μεγαλύτερη. Έπειτα οι κατηγορίες των k κοντινότερων αποστάσεων τοποθετούνται στη λίστα neighbors για να ακολουθήσει η διαδικασία του Majority Voting. Σε ένα dictionary κρατάμε ως key το όνομα της κατηγορίας, και ως value το πλήθος των γειτόνων που ανήκουν σε αυτήν. Η κατηγορία με το μεγαλύτερο πλήθος εμφανίσεων γίνεται append στην λίστα predicted. Αυτή η λίστα περιλαμβάνει την πρόβλεψη για το σε ποια κατηγορία ανήκει το κάθε κείμενο του X_test, η οποία και τελικά επιστρέφεται.

Components - Accuracy

Στην αρχή θα διαβάσουμε τα documents και θα τα μετατρέψουμε σε διανύσματα (με τη χρήση του TFIDF vectorizer). Στη συνέχεια της προ-επεξεργασίας των δεδομένων, χρησιμοποιήσαμε την τεχνική Latent Semantic Indexing (TruncatedSVD) για να μειώσουμε τις διαστάσεις των διανυσμάτων.

Για να βρούμε την καλύτερη δυνατή σχέση components και accuracy, με την χρήση του SVC classifier, κατασκευάσαμε το παρακάτω διάγραμμα δοκιμάζοντας διάφορες τιμές για τα components.

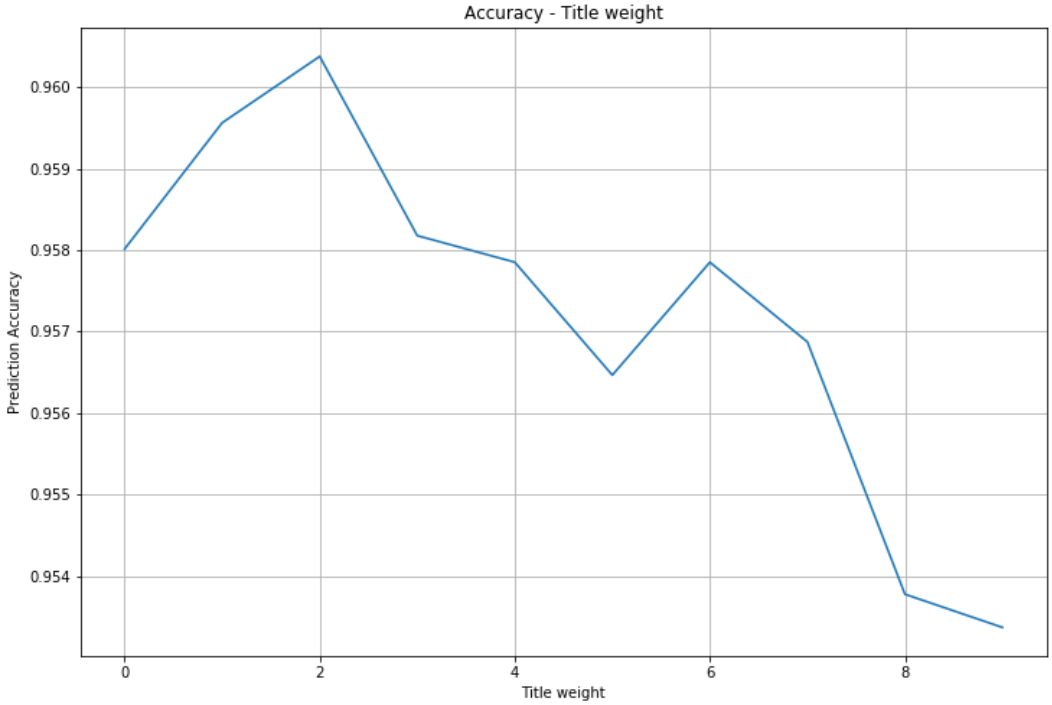


Παραπάνω παρατηρούμε ότι το accuracy αυξάνεται ραγδαία μεταξύ 1-10 components. Μεταξύ 10-100 μεταβάλλεται εξαιρετικά αργά, και για περισσότερα από 100 components τείνει να σταθεροποιηθεί , έχοντας λάβει τη μέγιστη τιμή του. Τέλος, λαμβάνοντας υπόψη και το χρόνο εκτέλεσης του προγράμματος, καταλήγουμε στο συμπέρασμα πως η ιδανική επιλογή components είναι 100 , μιας και όσο αυτά αυξάνονται κάνουν πιο αργό το πρόγραμμα μας.

* Ο πηγαίος κώδικα που δημιουργεί το διάγραμμα βρίσκεται στο αρχείο components.py

Βάρος Τίτλου

Όπως γνωρίζουμε, ο τίτλος σε κάθε κείμενο οφείλει να περιγράφει συνοπτικά και αποτελεσματικά το περιεχόμενο του. Για να αξιοποιήσουμε αυτή τη πληροφορία, εντάξαμε στο content του κάθε κειμένου το τίτλο του με ένα βάρος (δηλαδή προσθέσαμε κάποιες φορές το τίτλο μέσα στο κείμενο). Για να βρούμε πόσες φορές θα έπρεπε να συμπεριληφθεί ο τίτλος στο κείμενο χωρίς να δημιουργεί όμως αλλοιώσεις, σχεδιάσαμε το παρακάτω διάγραμμα



Όπως φαίνεται και στο διάγραμμα, το καλύτερο accuracy το πετυχαίνουμε αν προσθέσουμε 2 φορές το τίτλο του κάθε κειμένου μέσα σ ‘σαυτό.

Προκειμένου να βελτιώσουμε τις προβλέψεις των classifiers, υπήρξε η ιδέα να δώσουμε βάρος τόσο στη θεματική περίοδο όσο και στην κατακλείδα του κάθε κειμένου , όπως κάναμε και με τον τίτλο , με τη λογική ότι κι αυτές περιλαμβάνουν πολύ σημαντικές πληροφορίες για το θέμα του κειμένου (δηλαδή πολλές λέξεις κλειδιά). Η τελευταία ιδέα όμως δεν υλοποιήθηκε.

** Ο πηγαίος κώδικας που δημιουργεί το διάγραμμα βρίσκεται στο αρχείο weight.py

Υπολογισμός Metrics

Για την καλύτερη απόδοση των classifiers δημιουργούμε ένα νέο set από stop words, το οποίο αποτελείται από τα ENGLISH_STOP_WORDS που περιέχονται στη βιβλιοθήκη sklearn.feature_extraction.text , τα stop words της βιβλιοθήκης wordcloud αλλά και ορισμένα δικά μας. Στη συνέχεια προσθέτουμε στο content του κάθε κειμένου τον τίτλο με συντελεστή βάρους 2. Ακολουθεί ο TfidfVectorizer για να δημιουργηθούν τα διανύσματα και μετά ο TruncatedSVD για να μειώσουμε τις διαστάσεις των διανυσμάτων. Για να κρατήσουμε τα στατιστικά των metrics χρησιμοποιούμε ένα πίνακα 5x4 οπού κάθε μια γραμμή αντιπροσωπεύει μια διαφορετική μετρική και κάθε στήλη έναν classifier. Στη συνέχεια ακολουθεί το 10-Fold Cross Validation. Στην αρχή κάθε επανάληψης οι τιμές των μεταβλητών X_train, X_test, Y_train , Y_test αναπροσδιορίζονται μετά την επιλογή των νέων τιμών train_index, test_index. Κάθε classifier εκπαιδεύεται με τα X_train (περιεχόμενο για training), Y_train κατηγορίες κειμένων για training και κάνει predict για το X_test (περιεχόμενο για testing). Υπολογίζουμε τις μετρικές συγκρίνοντας τις κατηγορίες που προέβλεψε (yPred) ο κάθε classifier σε σχέση με τις πραγματικές (Y_test).

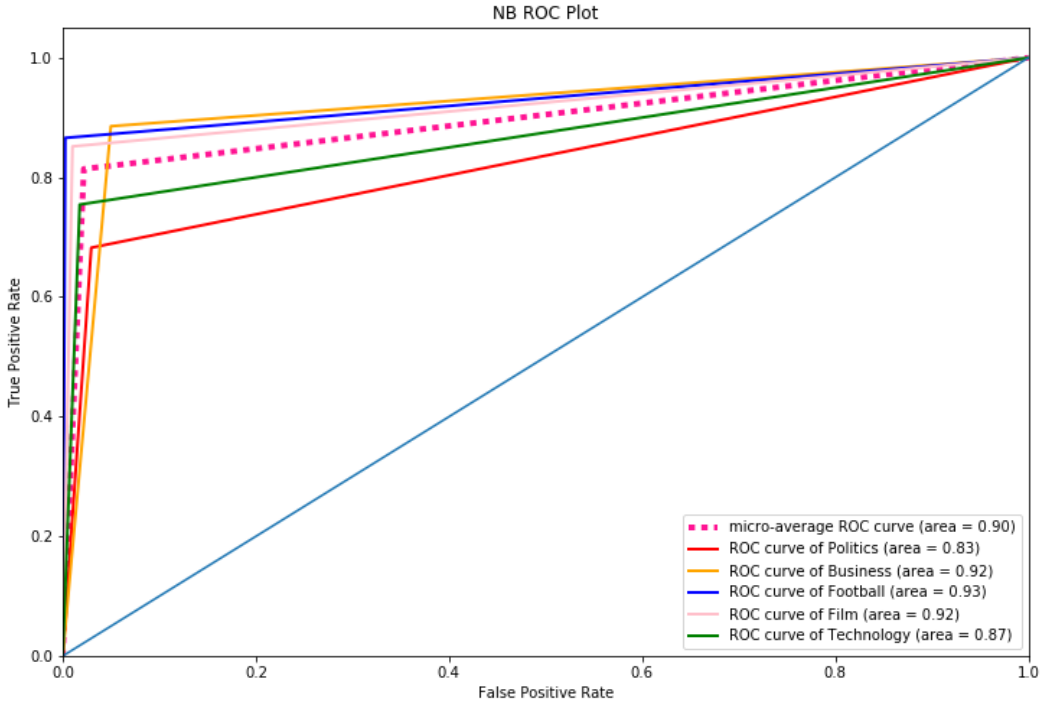
- Για τον υπολογισμό του accuracy χρησιμοποιούμε τη συνάρτηση: accuracy_score()
- Για τον υπολογισμό του precision χρησιμοποιούμε τη συνάρτηση: precision_score()
- Για τον υπολογισμό του recall χρησιμοποιούμε τη συνάρτηση: recall_score()
- Για τον υπολογισμό του F-Measure χρησιμοποιούμε τη συνάρτηση: f1_score()
- Για τον υπολογισμό του AUC χρησιμοποιούμε τη συνάρτηση: roc_auc_score()

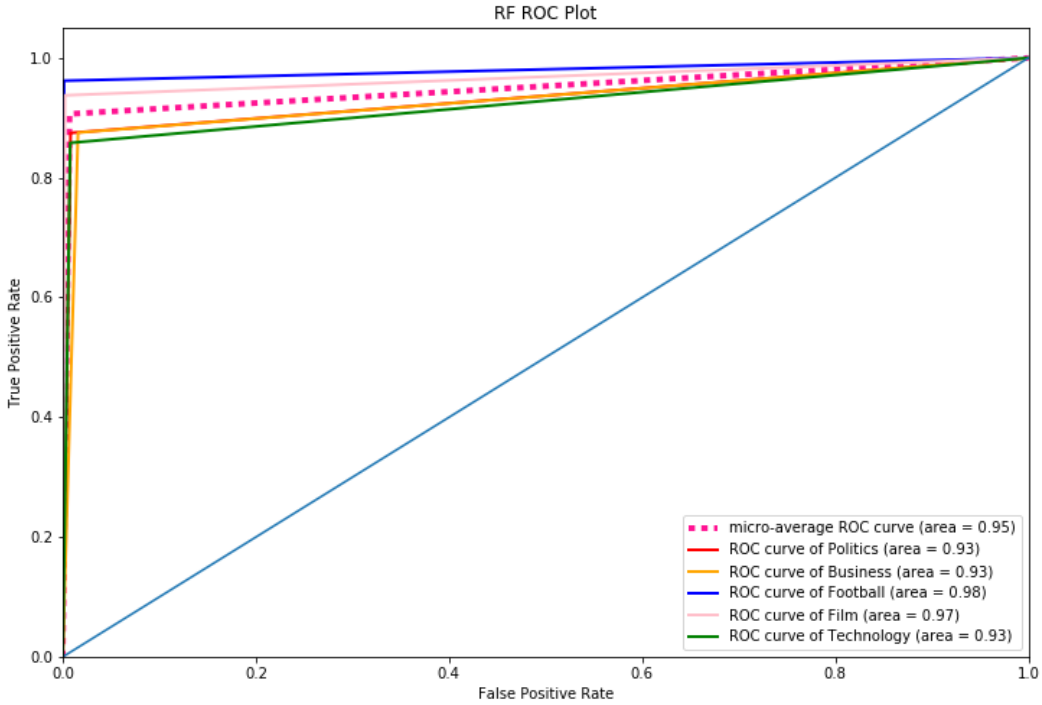
Για την τελευταία προαναφερθείσα συνάρτηση χρειάστηκε να μετατρέψουμε τα δοθέντα ορίσματα της σε binary με τη χρήση της συνάρτησης label_binarize().

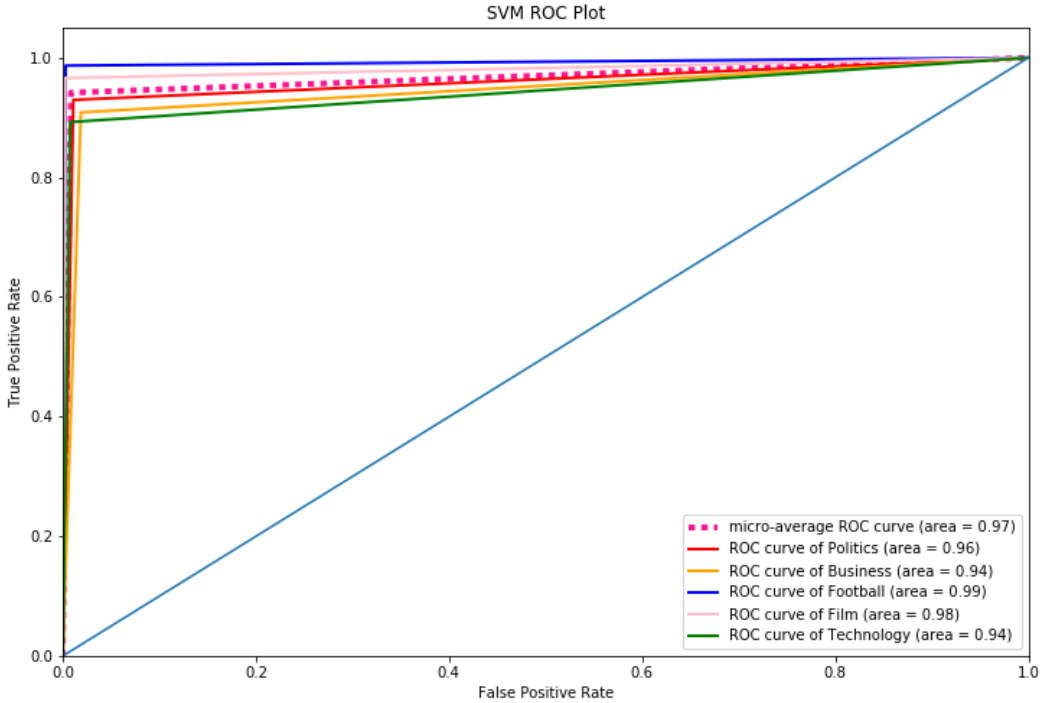
Επίσης να αναφέρουμε ότι σαν παράμετρο του average σε κάθε μετρική συνάρτηση επιλέξαμε την 'weighted' διότι είναι σωστότερο να λάβουμε υπόψη το πλήθος των κειμένων της κάθε κατηγορίας. Δηλαδή δεν πρέπει 100 κείμενα που ανήκουν σε μια κατηγορία να έχουν το ίδιο βάρος, στον υπολογισμό του μέσου ορού, με 1000 κείμενα που ανήκουν σε μια άλλη κατηγορία. Τέλος γράφουμε τα αποτελέσματα στο αρχείο EvaluationMetric_10fold.csv .

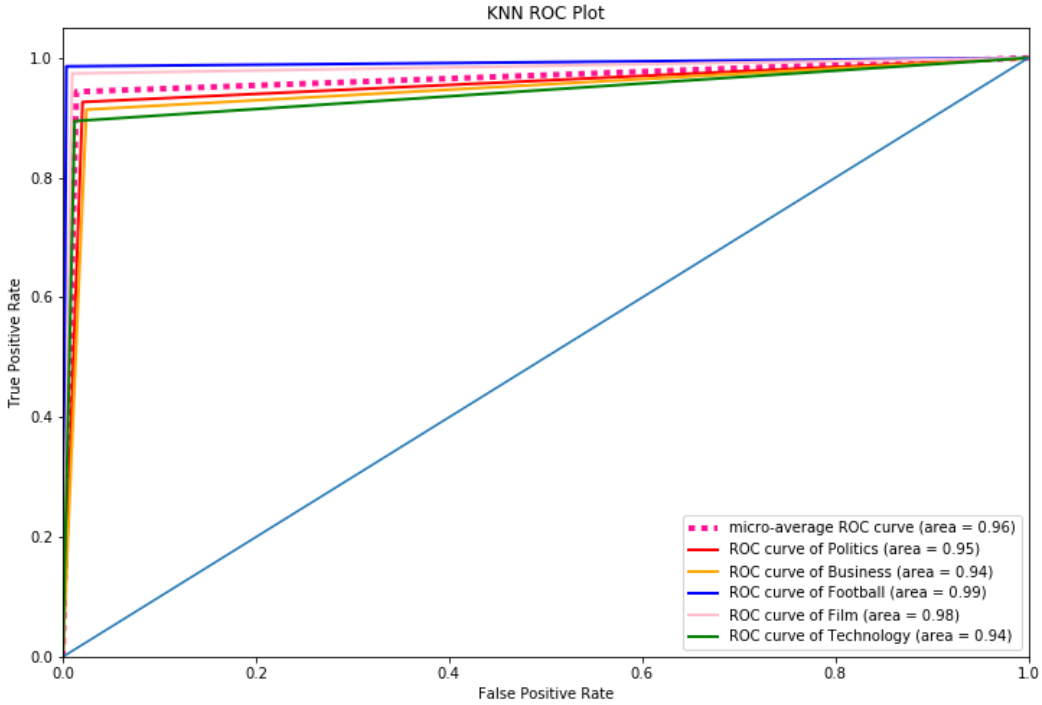
Διαγράμματα Roc Plot

Βασίσαμε τη λογική μας στο παρακάτω link το οποίο προτάθηκε και στο piazza (http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html).Κατασκευάσαμε για κάθε classifier από ένα διάγραμμα Roc Plot που απεικονίζει τα 5 categories καθώς και το μέσο ορό τους (micro-average) με ένα διαφορετικό χρώμα. Τα διαγράμματα περιλαμβάνονται στο παραδοτέο φάκελο της άσκησης με τα ονόματα NB_roc_plot.png, RF_roc_plo.png, SVM_roc_plot.png, KNN_roc_plot.png.


NB ROC Plot
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.90)
ROC curve of Politics (area = 0.83)
ROC curve of Business (area = 0.92)
ROC curve of Football (area = 0.93)
ROC curve of Film (area = 0.92)
ROC curve of Technology (area = 0.87)


RF ROC Plot
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.95)
ROC curve of Politics (area = 0.93)
ROC curve of Business (area = 0.93)
ROC curve of Football (area = 0.98)
ROC curve of Film (area = 0.97)
ROC curve of Technology (area = 0.93)


SVM ROC Plot
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.97)
ROC curve of Politics (area = 0.96)
ROC curve of Business (area = 0.94)
ROC curve of Football (area = 0.99)
ROC curve of Film (area = 0.98)
ROC curve of Technology (area = 0.94)


KNN ROC Plot
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.96)
ROC curve of Politics (area = 0.95)
ROC curve of Business (area = 0.94)
ROC curve of Football (area = 0.99)
ROC curve of Film (area = 0.98)
ROC curve of Technology (area = 0.94)

TestSet Predictions

Μετά από πολλές δοκιμές που μας ζητήθηκαν και από την ιδιά την άσκηση , αλλά και από τα αποτελέσματα των διαγραμμάτων , καταλήξαμε στο συμπέρασμα ότι αποδοτικότερος classifier είναι ο SVC τον οποίο και χρησιμοποιήσαμε για να κάνουμε τις προβλέψεις του test_set.csv