ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

1η Εργασία για το μάθημα

Τεχνικές Εξόρυξης Δεδομένων

Χρήστος Κιτσανέλης

Α.Μ. : 1115201200065

Μαρία-Ισαβέλλα Μανωλάκη-Σεμπάγιος

Α.Μ. : 1115201300093

Επιβλέποντες: Γουνόπουλος Δημήτριος, Καθηγητής

ΑΘΗΝΑ
ΜΑΙΟΣ 2017

Wordcloud

Έχουμε δημιουργήσει 2 κλάσεις που χρησιμοποιούνται για την προεπεξεργασία των κειμένων, την article και την preprocessor.

Στην article έχουμε τις εξής συναρτήσεις :

- Τον constructor που αποθηκεύει τον τίτλο, το περιεχόμενο και την κατηγορία ενός κειμένου.
- Την tokenize που αφαιρεί τους ειδικούς χαρακτήρες και τους αριθμούς απο το περιεχόμενο και τον τίτλο ενός κειμένου
- Την removeSW που αφαιρεί τα stopwords από τον τίτλο και το περιεχόμενο ενός κειμένου
- Συναρτήσεις που επιστρέφουν τον τίτλο, το περιεχόμενο, την κατηγορία, και τον τίτλο και το περιεχόμενο χωρίς τα stopwords.

Στην preprocessor έχουμε τις εξής συναρτήσεις :

- Τον constructor ο οποίος αποθηκεύει σε λίστα όλα τα stopwords, διαβάζει το αρχείο και αποθηκεύει σε ξεχωριστές λίστες τους τίτλος, τα περιεχόμενα και τις κατηγορίες και φτιάχνει αντικείμενα article με κάθε τριάδα και δημιουργείται μια λίστα με τις ξεχωριστές κατηγορίες.
- Την tokenize που για κάθε article καλεί την tokenize και αφαιρεί τα stopwords.
- Την getCategoryContent που επιστρέφει κείμενα που περιέχουν όλους τους τίτλους και περιεχόμενα για κάθε κατηγορία(αφού έχουν περαστεί από το tokenize).

Στο κύριο κομμάτι του κώδικα του αρχείου wcloud.py διαβάζουμε αρχικά τα stopwords και όλα τα κείμενα από το train_set.csv, αφαιρούμε τους ειδικούς χαρακτήρες και τα stopwords από κάθε κείμενο και τίτλο, παίρνουμε το πλήθος των κατηγοριών που υπάρχουν και τέλος, για κάθε κατηγορία δημιουργούμε το ειδικό μας κείμενο με όλα τα περιεχόμενα και τους τίτλους, καλούμε την συνάρτηση Wordcloud με τα δικά μας ορίσματα και αποθηκεύουμε το κάθε αποτέλεσμα σε ένα ξεχωριστό αρχείο png.

Πήραμε έτοιμη τη συνάρτηση Wordcloud από τη βιβλιοθήκη wordcloud.

Προτιμήσαμε στο wordcloud που προκύπτει να γίνεται εμφανές εύκολα για ποιας κατηγορίας πρόκειται οπότε χρησιμοποιήσαμε ένα εκτενές κείμενο με stopwords (γύρω στις 500 λέξεις). Φυσικά αυτό κοστίζει στο χρόνο εκτέλεσης του αρχείου.

Illustration 1: Business Wordcloud

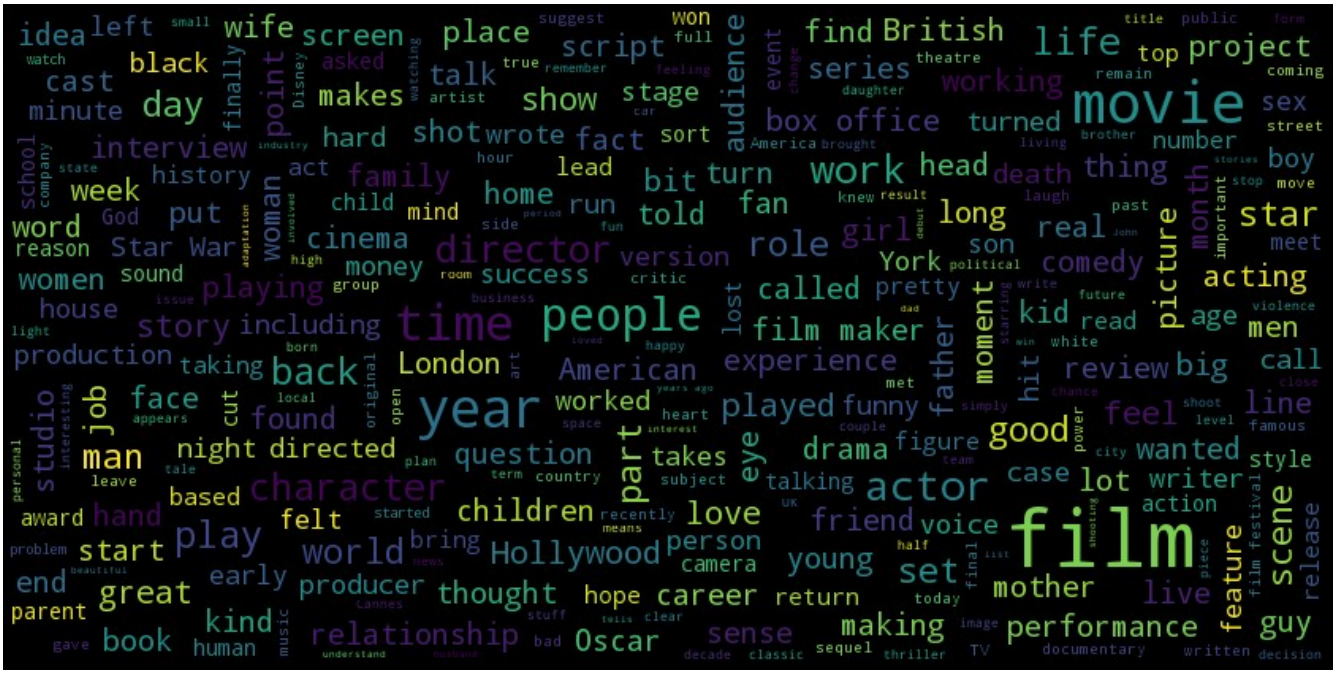


Illustration 2: Film Wordcloud

Illustration 3: Football Wordcloud

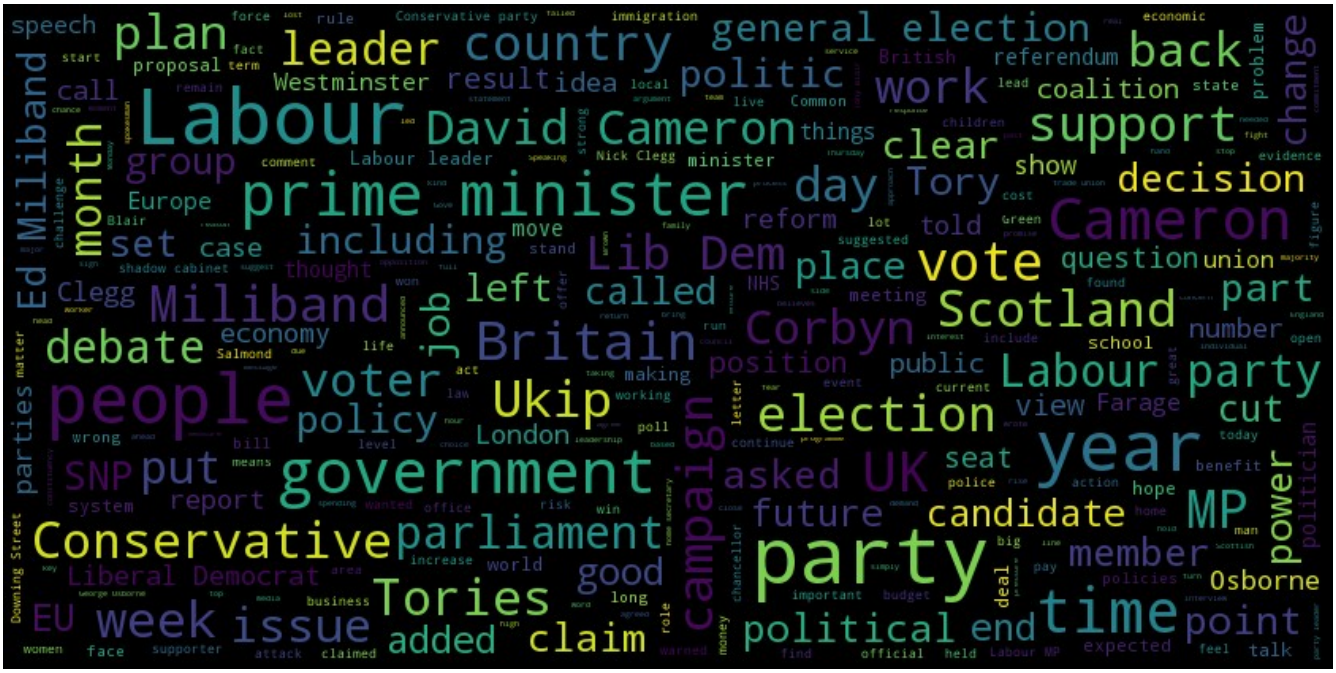


Illustration 4: Politics Wordcloud

Illustration 5: Technology Wordcloud

Συσταδοποίηση

Η συσταδοποίηση με τον αλγόριθμο K-means υλοποιείται στο αρχείο kmeans.py . Αποφασίσαμε να τον υλοποιήσουμε μόνοι μας τον αλγόριθμο. Η διαδικασία που ακολουθείται είναι η εξής :

- Γίνεται τυχαία επιλογή των σημείων όπου θα τοποθετηθούν τα αρχικά κέντρα των clusters.
- Υπολογίζεται για κάθε data point η απόσταση του με όλα τα κέντρα των clusters. Ως μέτρο απόστασης χρησιμοποιείται η Cosine Distance.
- Κάθε data point θεωρούμε πως “ανήκει” στο cluster, το κέντρο του οποίου είναι πιο κοντά στο data point.
- Υπολογισμός της νέας θέσης του κέντρου κάθε cluster έτσι υπολογίζοντας τη μέση τιμή των στοιχείων που ανήκουν στο cluster.
- Υπολογίζουμε ξανά τις αποστάσεις κάθε data point από τα κέντρα και βρίσκουμε σε ποιο cluster ανήκουν.
- Η διαδικασία τελειώνει όταν κανένα data point δεν αλλάξει cluster μετά από υπολογισμό θέσεων.

Στο αρχείο θα βρείτε 4 συναρτήσεις :

1) initCenters : Υπολογίζει K(5) τυχαίους αριθμούς με μέγιστη τιμή το πλήθος των data point και τους τοποθετεί σε μια λίστα.
2) NewCenters : Υπολογίζει τις μέσες τιμές των data point που ανήκουν σε κάθε cluster και τις τοποθετεί σε νέα λίστα. Σε περίπτωση που ένα cluster δεν έχει κανένα data point, υπολογίζεται ένας τυχαίος αριθμός ξανά.
3) Kmeans : Η κύρια συνάρτηση του αρχείου. Δημιουργεί αρχικά τη λίστα με τα πρώτα κέντρα των clusters. Στη συνέχεια μπαίνει σε μια λούπα επανάληψης που θα σταματήσει μόνο όταν ικανοποιηθεί η συνθήκη τερματισμού μας. Μέσα στη λούπα, για κάθε data point υπολογίζονται οι αποστάσεις από τα κέντρα και αποθηκεύεται στο κοντινότερο cluster (με χρήση μίας λίστας για κάθε cluster). Υπολογίζονται τα νέα κέντρα και γίνεται ο έλεγχος για τη περίπτωση που όλα τα data points έμειναν στο ίδιο cluster που βρίσκονταν και πριν.
4) Percentages : Σε κάθε cluster υπολογίζεται ο αριθμός και το ποσοστό των

data points κάθε κατηγορίας και τα αποτελέσματα γράφονται στο αρχείο “clustering_Kmeans.csv”.

Ένα ενδεικτικό αποτέλεσμα εκτέλεσης του προγράμματος :

	Politics	Film	Football	Business	Technology
Cluster 1	0.013	0.009	0.001	0.079	0.898
Cluster 2	0	0.001	0.993	0.003	0.004
Cluster 3	0.087	0.001	0.003	0.901	0.008
Cluster 4	0.972	0.001	0.004	0.019	0.004
Cluster 5	0.012	0.959	0.004	0.006	0.019

ΠΑΡΑΤΗΡΗΣΕΙΣ

Ο TfIdf Vectorizer έδωσε καλύτερα αποτελέσματα από τον CountVectorizer.

Η επεξεργασία με LSI μείωσε σε ιδιαίτερο βαθμό το χρόνο. Λογικό, καθώς μικραίνει κατά πολύ ο αριθμός των λέξεων στις οποίες θα πρέπει να ανατρέξει η Cosine Distance για τον υπολογισμό των αποστάσεων.

Η προ-επεξεργασία γίνεται και στο κείμενο και στον τίτλο, για να δωθεί βαρύτητα και στον τίτλο.

Στις περισσότερες εκτελέσεις, τα αποτελέσματα ήταν πολύ θετικά και κάθε κατηγορία μαζεύοταν σε ένα συγκεκριμένο cluster. Υπήρχαν όμως και περιπτώσεις που μπορει να υπήρχε μεγάλο ποσοστό μιας κατηγορίας σε 2 clusters. Σε αυτές τις περιπτώσεις μπορούμε να συμπεράνουμε ότι μάλλον τα 2 αυτά clusters βρίσκονταν σε πολύ κοντινή απόσταση.

ΠΙΘΑΝΕΣ ΒΕΛΤΙΩΣΕΙΣ

Στην αρχική τοποθέτηση των κέντρων είναι επιθυμητή η όσο γίνεται πιο αραιή τοποθέτηση τους. Θα μπορούσαμε λοιπόν να τοποθετήσουμε το καθένα σε απόσταση Χ από το προηγούμενο.

Θα μπορούσαμε να κάνουμε ακόμη καλύτερη προ-επεξεργασία αφαιρώντας παραπάνω stopwords.

Θα μπορούσαμε να χρησιμοποιήσουμε πολλαπλές φορές τον τίτλο στα δεδομένα μας δίνοντάς του έτσι ακόμα μεγαλύτερο βάρος.

Κατηγοριοποίηση

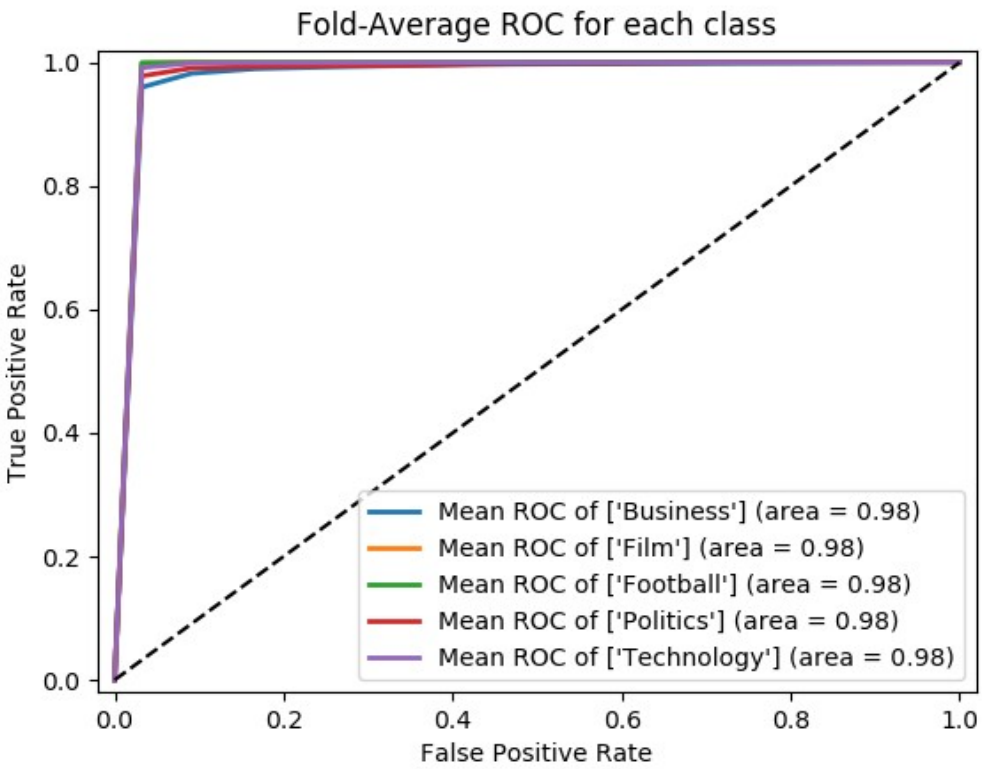
Για την υλοποίηση της κατηγοριοποίησης χρησιμοποιήθηκαν οι εξής classifiers : SVM (Support Vector Machines), Random Forests, Multinomial Native Bayes και δημιουργήσαμε τη δική μας έκδοση του K-Nearest Neighbors. Τους έτοιμους κατηγοριοποιητές τους πήραμε από το εργαλείο Scikit Learn. Καταλήξαμε στις τιμές των ορισμάτων που περνάμε μετά από εκτελέσεις με διαφορετικές επιλογές.

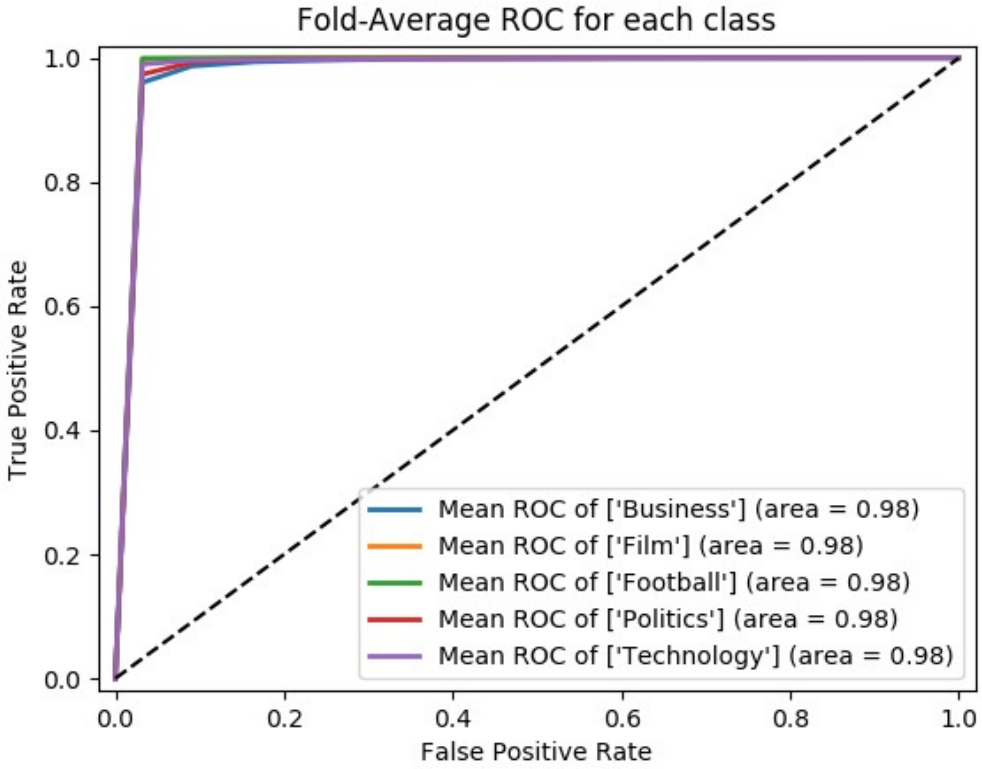
Αρχικά. χρησιμοποιούμε πάλι TfIdf Vectorizer, εκτελούμε προεπεξεργασία αφαιρώντας τα αγγλικά stopwords και επεξεργασία με LSI και πραγματοποιούμε 10-fold cross validation με τη βιβλιοθήκη Kfold από το scikit. Στη συνέχεια εκπαιδεύουμε κάθε κατηγοριοποιητή με τη χρήση του 10-fold και τέλος επιλέγουμε τον κατηγοριοποιητή που επέστρεψε το καλύτερο accuracy για να κατηγοριοποιήσει τα κείμενα του test_set. Τα αποτελέσματα που προκύπτουν γράφονται στο αρχείο “testSet_categories.csv”.

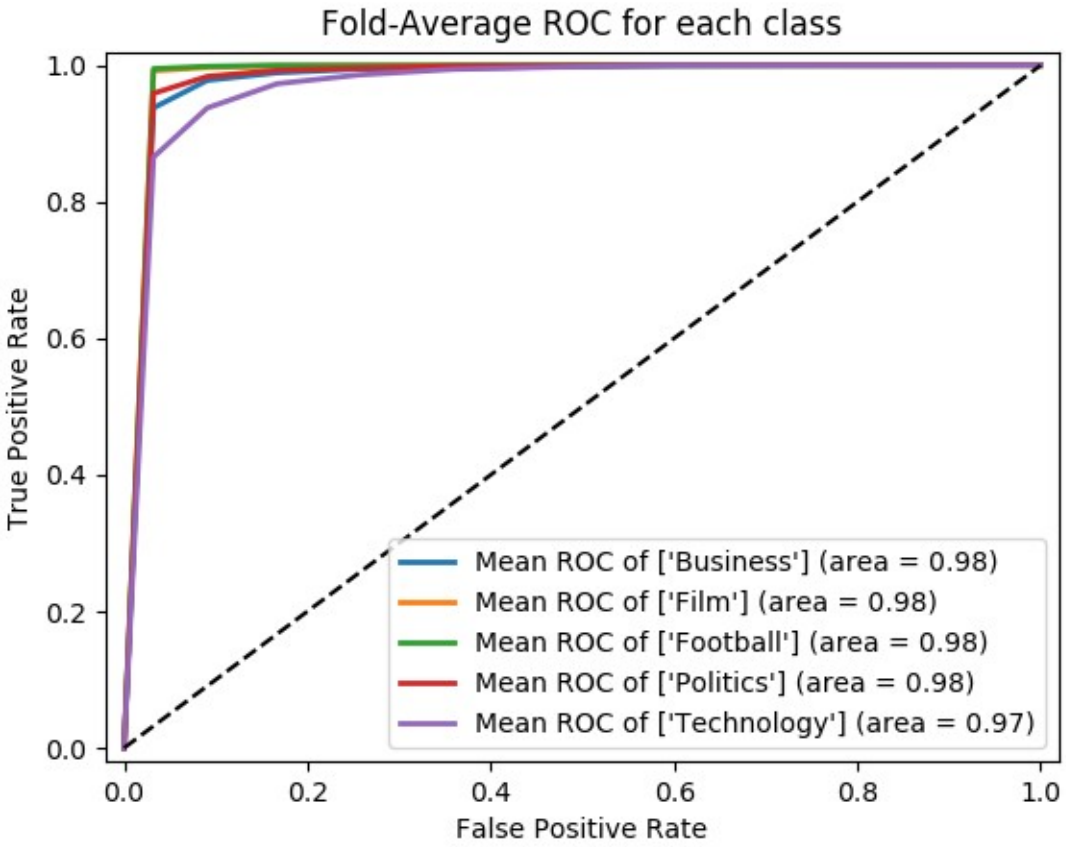
Οι αξιολογήσεις έγιναν χρησιμοποιώντας τη βιβλιοθήκη metrics από το scikit Learn. Ένα παράδειγμα υπολοίησης και αξιολόγησης των κατηγοριοποιητών :

	SVM	Random Forest	Naive Bayes	KNN
Accuracy	0.964	0.964	0.938	0.962
Precision	0.961	0.961	0.944	0.959
Recall	0.962	0.961	0.916	0.961
F-Measure	0.961	0.961	0.924	0.96
AUC	0.491	0.491	0.489	0

Το διάγραμμα ROC είναι πολύ χρήσιμο για την αποτίμηση της απόδοσης ενός ταξινομητή. Από το ROC μπορούμε να υπολογίσουμε και το AUC, από το εμβαδό της καμπύλης που δημιουργείται στο διάγραμμα. Χρησιμοποιώντας πάλι τη βιβλιοθήκη metrics, είμαστε σε θέση να υπολογίσουμε τα σημεία της καμπύλης από τη συνάρτηση roc_curve και το AUC από την auc.
Τα διαγράμματα που προέκυψαν είναι τα εξής :

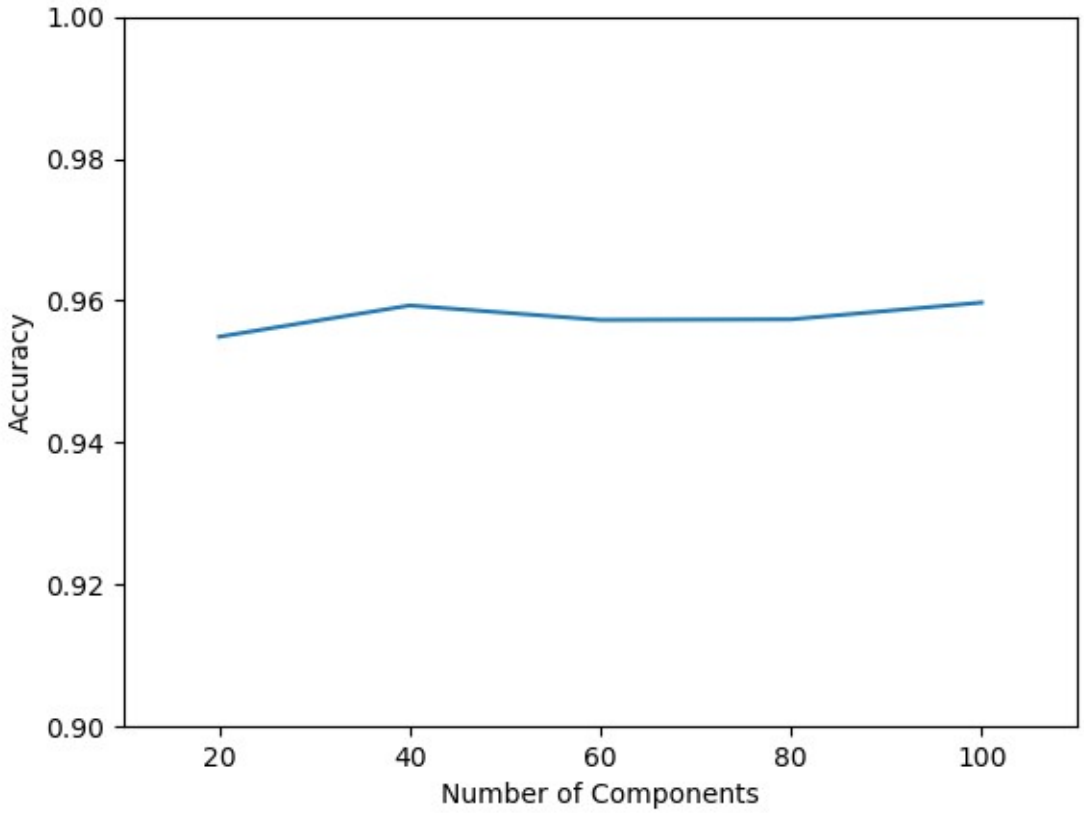

Fold-Average ROC for each class
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Mean ROC of ['Business'] (area = 0.98)
Mean ROC of ['Film'] (area = 0.98)
Mean ROC of ['Football'] (area = 0.98)
Mean ROC of ['Politics'] (area = 0.98)
Mean ROC of ['Technology'] (area = 0.98)


Fold-Average ROC for each class
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Mean ROC of ['Business'] (area = 0.98)
Mean ROC of ['Film'] (area = 0.98)
Mean ROC of ['Football'] (area = 0.98)
Mean ROC of ['Politics'] (area = 0.98)
Mean ROC of ['Technology'] (area = 0.98)



Δε καταφέραμε να φτιάξουμε το διάγραμμα ROC για τον κατηγοριοποιητή KNN, εξού και ο λόγος που το AUC σε εκείνη τη περίπτωση είναι ίσο με 0.

Προφανώς ένας μεγαλύτερος αριθμός από components μπορεί να μας προσφέρει καλύτερο accuracy, αλλά αυξάνει ταυτόχρονα το χρόνο που χρειάζεται για να μας επιστρέψει αποτελέσματα, καθώς απαιτούνται περισσότεροι έλεγχοι. Μπορείτε να εκπαιδεύσετε τον Random Forests κατηγοριοποιητή με ένα πλήθος από components (μεταξύ 20 και 100) στο αρχείο AccuracyChart.py . Προκύπτουν τα εξής αποτελέσματα :




```
Number of components 20 took 36.39 seconds
Number of components 40 took 47.49 seconds
Number of components 60 took 54.23 seconds
Number of components 80 took 61.24 seconds
Number of components 100 took 72.81 seconds
```

ΥΛΟΠΟΙΗΣΗ ΚΝΝ

Παίρνουμε ένα ένα τα data points που θέλουμε να κατηγοριοποιήσουμε και ψάχνουμε τα k κοντινότερά του data point από το train set. Μόλις βρεθούν

υπολογίζουμε ποια κατηγορία έχει μεγαλύτερο αριθμό από data points στους γείτονες και κατηγοριοποιούμε το αρχικό data point σε αυτή την κατηγορία.

Beat the Benchmark

Για βελτιστοποίηση των μεθόδων Classification υλοποιήθηκαν αρχικά οι δομές Preprocessor και Article που αναφέρθηκαν στο documentation του WordCloud, για να χωρίσουμε το κείμενο σε λέξεις και να εξαιρεθούν οι λέξεις που περιέχονται στο αρχείο με τα stop words.

Αρχικά διαβάζεται το DataFrame και δημιουργείται η δομή Preprocessor, έπειτα γίνεται το tokenizing του κειμένου, στο οποίο επιλέξαμε να περιέχεται και ο τίτλος. Ακολουθείται η ίδια διαδικασία όπως και στο classification.py με την διαφορά ότι χρησιμοποιείται η VotingClassifier της sklearn.ensemble, αντί να χρησιμοποιείται κάποιος συγκεκριμένος classifier. Η συνάρτηση αυτή παίρνει ως ορίσματα τους classifiers που θα χρησιμοποιήσουμε (SVM,Random Forests,Naive Bayes). Δίνουμε επίσης mode=hard ώστε η συνάρτηση να λειτουργήσει με Majority Voting. Με τον συγκεκριμένο τρόπο λειτουργίας ένα δείγμα θα ανήκει τελικά στην κατηγορία στην οποία προβλέπουν οι περισσότεροι classifiers του συνόλου.