1^{η} Εργασία μαθήματος Τεχνικές Εξόρυξης Δεδομένων

Απρίλιος 2017

Ομάδα

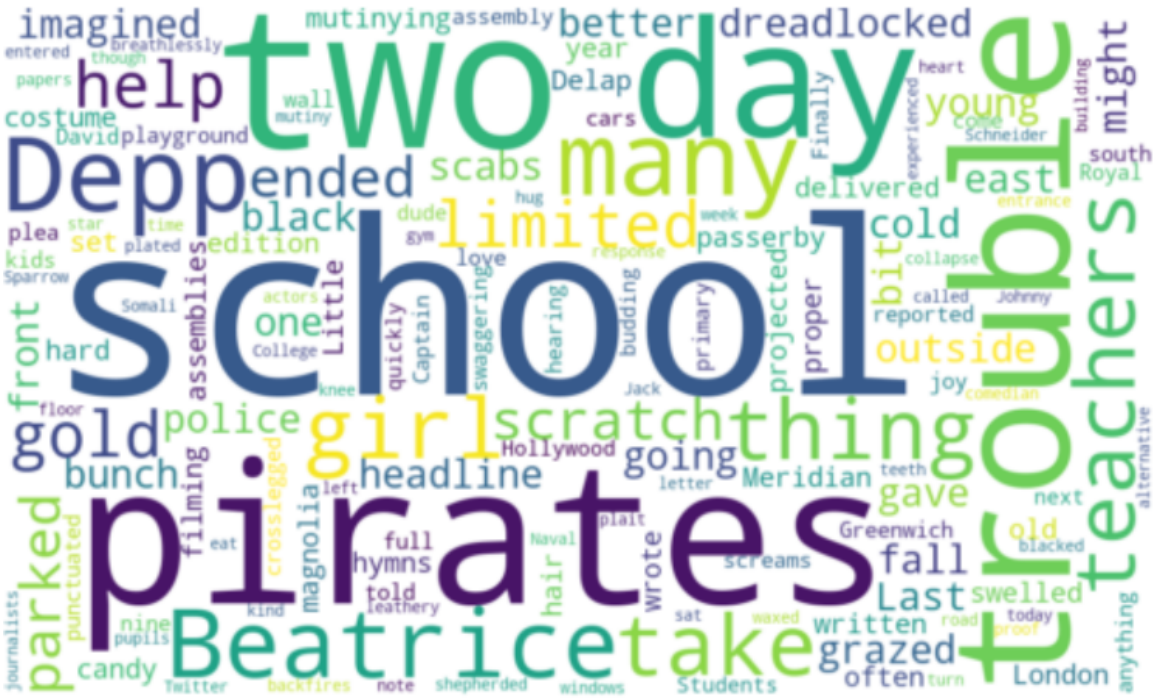
1.Ζέρντεβ Αλέξανδρος ΑΜ: 1115201600283
2.Ανδρινοπούλου Χριστίνα ΑΜ: 1115201500006

1. Wordcloud

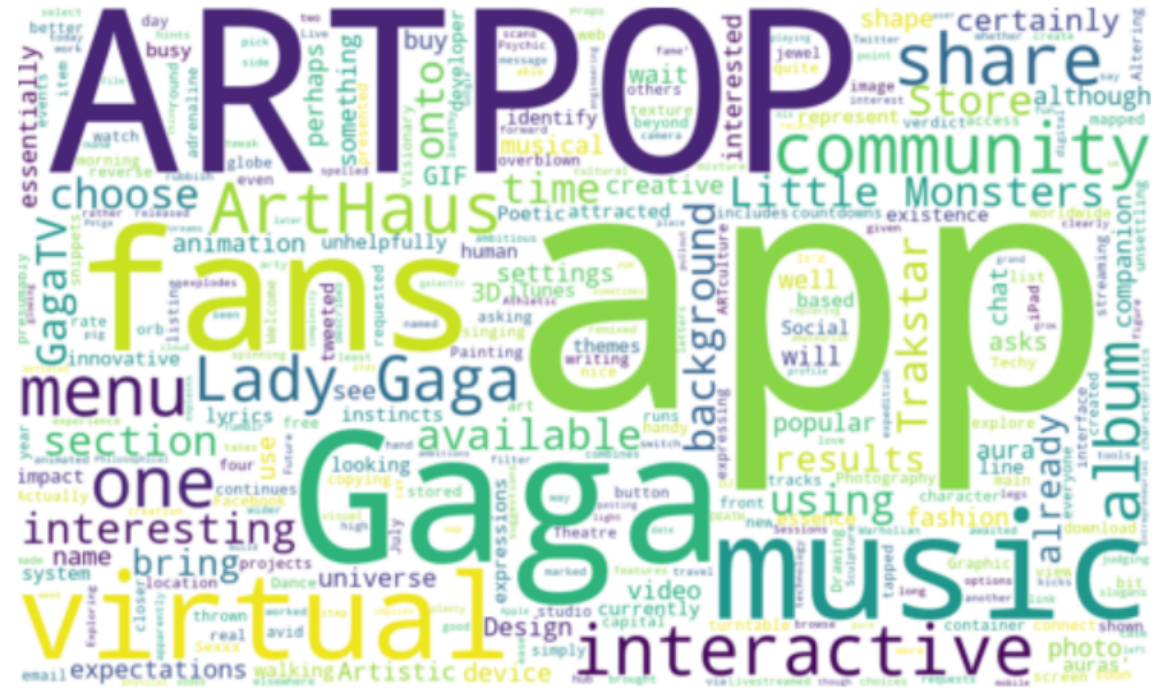
Κατόπιν αφαίρεσης των *stopwords* καθώς και δημιουργίας ενός δικού μας πίνακα με λέξεις που δεν μας δίνουν καμία πληροφορία (π.χ. στα *Politics* τη λέξη *said* (Thank you mr.Obvious ☺)) προκύπτουν τα παρακάτω *wordclouds*

Politics

Film

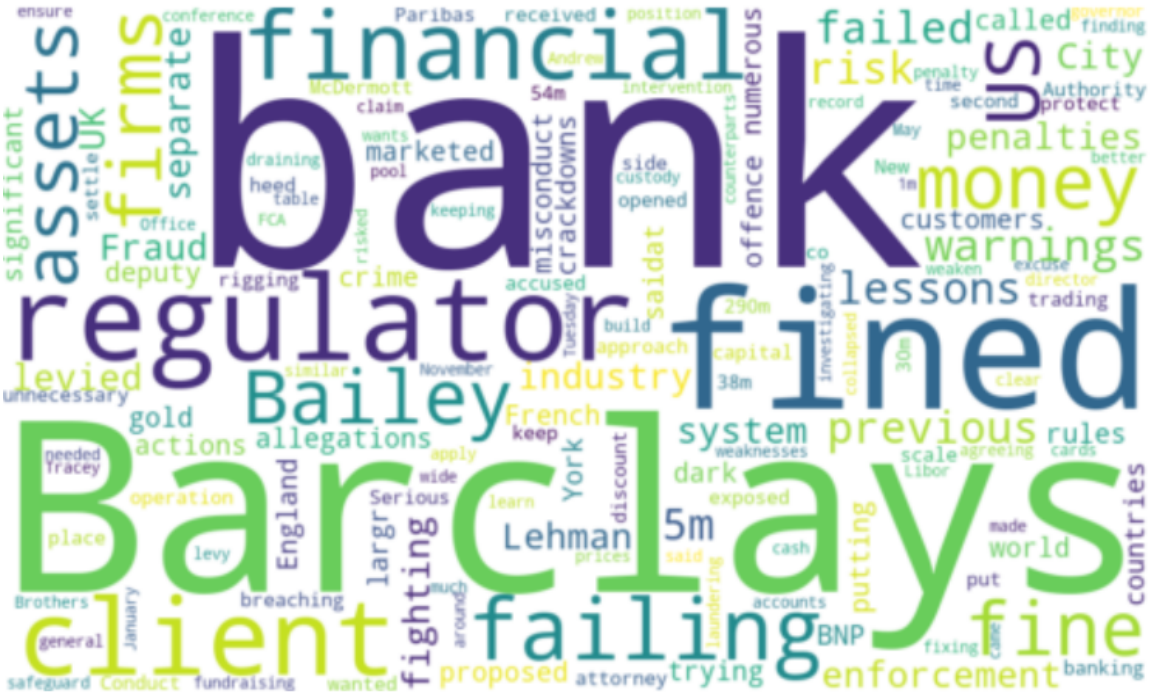


Technology



Football

Buisiness



2. Classification

Εισαγωγή

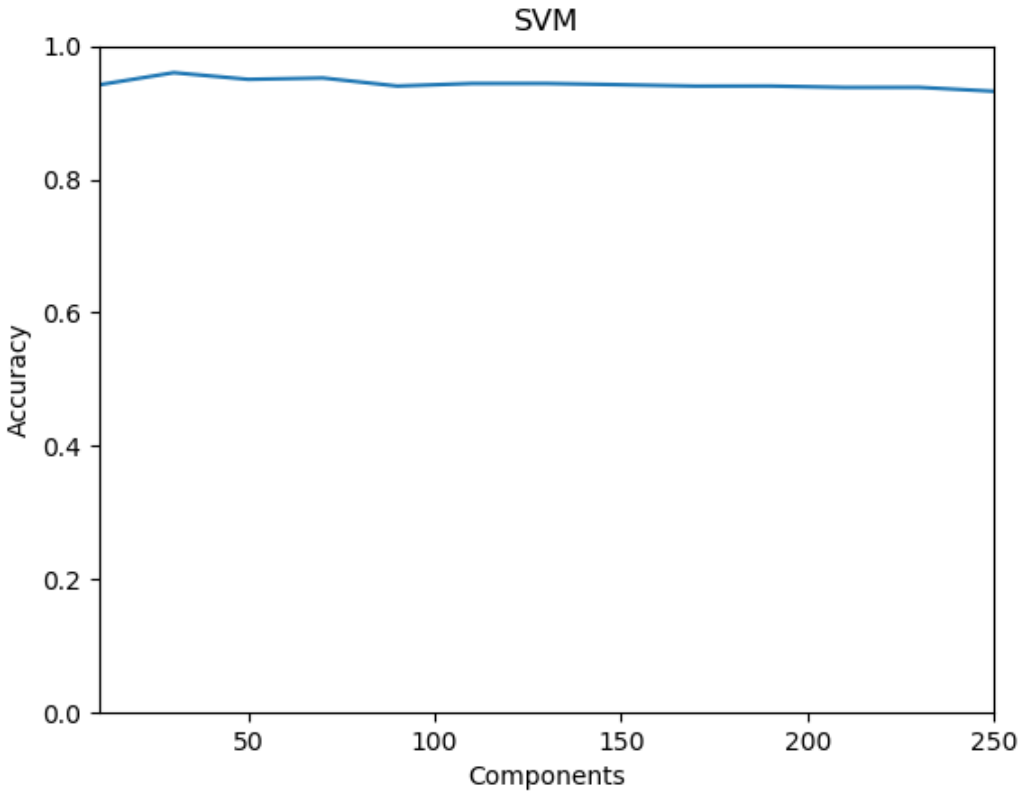
Στην παρούσα εργασία κληθήκαμε να κατηγοριοποιήσουμε μία πληθώρα άρθρων σε πέντε συγκεκριμένες κλάσεις: $Politics$, $Film$, $Football$, $Business$, $Technology$.
Αρχικά με χρήση του $TfidfVectorizer$ μετατρέψαμε τα δεδομένα μας σε διανύσματα. Ο $TfidfVectorizer$ δημιουργεί έναν πίνακα διανυσμάτων με βάση τη συχνότητα εμφάνισης των λέξεων. Επόμενο βήμα ήταν να αφαιρέσουμε όλα τα *stopwords* (*'and'*, *'the'*, *'or'* κτλ.), αφού είναι λέξεις που όχι απλά δεν προσφέρουν καμία πληροφορία σχετικά με το περιεχόμενο του άρθρου, αλλά ταυτόχρονα θα μπορούσαν να μας οδηγήσουν σε στρεβλά αποτελέσματα, καθώς είναι λέξεις που εμφανίζονται με μεγάλη συχνότητα σε όλα τα κείμενα ανεξάρτητα από το θέμα-κατηγορία τους.

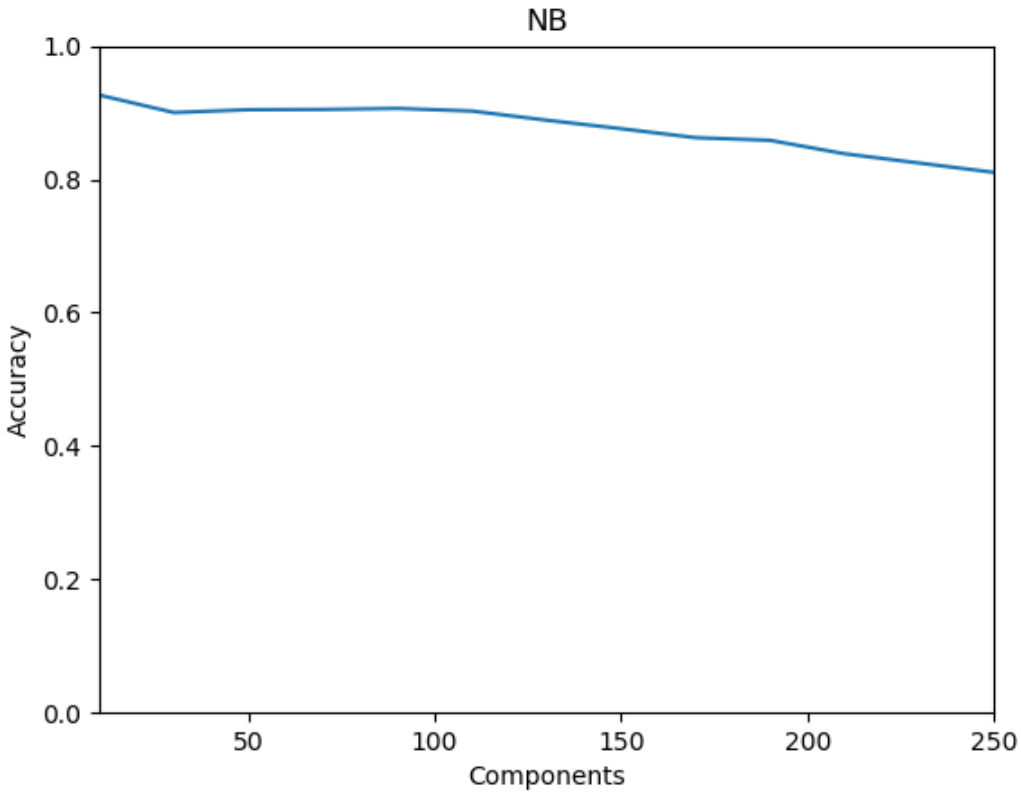
Προεπεξεργασία

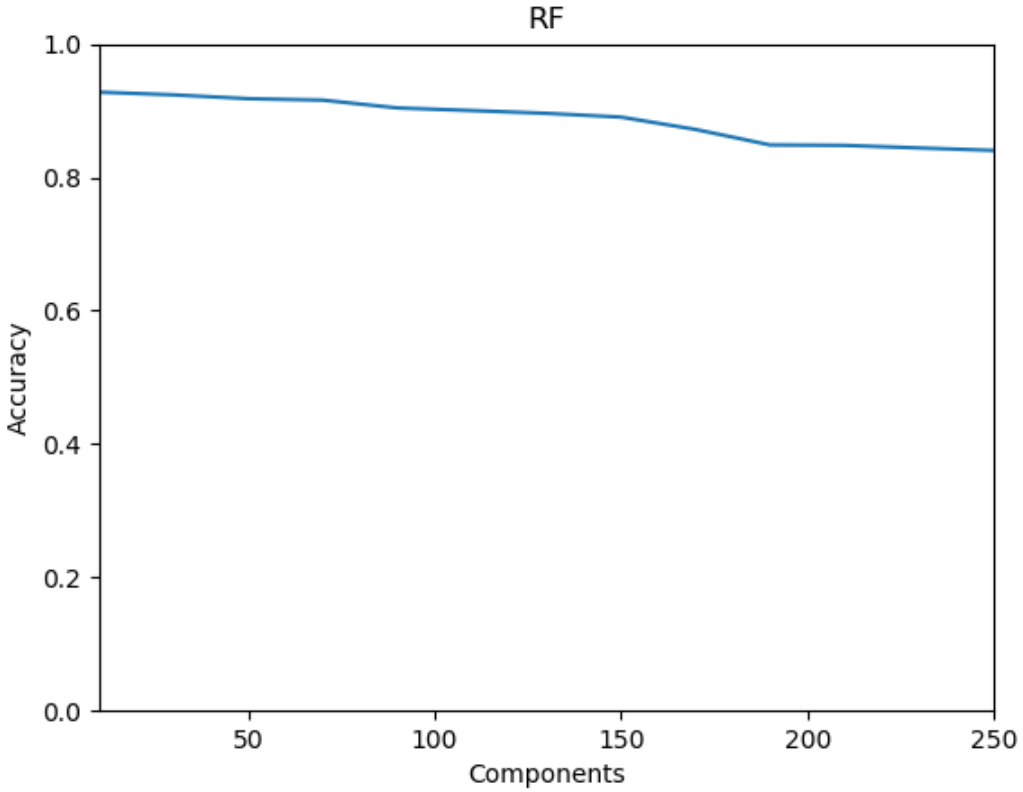
Αρχικά κάναμε *stemming* των λέξεων σε κάθε *dataset* ώστε να ομαδοποιήσουμε τις ομόριζες λέξεις (π.χ το *go* και το *going* ενώ έχουν διαφορετική συμβολοσειρά είναι λέξεις με την ίδια σημασία), ωστόσο δε λάβαμε καλύτερα αποτελέσματα. Έπειτα επιχηρήσαμε να αξιοποιήσουμε τον τίτλο με δύο διαφορετικούς τρόπους: (1) τον προσθέσαμε n φορές στο *vector*, (2) προσθέσαμε στο *vector* μόνο τις λέξεις του τίτλου που απαντώται και στο κείμενο. Τα αποτελέσματα δεν έδειξαν βελτίωση.

LSI

Στη συνέχεια εφαρμόσαμε την τεχνική LSI με την οποία μειώνουμε τη διάσταση του πίνακα διανυσμάτων σε k *components* απαλείφοντας τις διαστάσεις που εμφανίζονται οι λέξεις με τη μικρότερη συχνότητα. Η τεχνική αυτή είναι χρήσιμη για δυο λόγους: (1) ο κώδικας τρέχει πιο γρήγορα, καθώς έχει λιγότερα δεδομένα προς επεξεργασία και (2) αυξάνεται το *Accuracy* (αρκεί οι τιμές των *components* να μην είναι πολύ μεγάλες συγκριτικά με τις συχνότητες εμφάνισης των λέξεων). Επίσης πειραματιστήκαμε με τα *components*. Τα αποτελέσματα φαίνονται στα παρακάτω γραφήματα:


SVM
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
50
100
150
200
250
Components


NB
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
50
100
150
200
250
Components



Παρατηρούμε ότι ανεξάρτητα του εκάστοτε $classifier$ το $Accuracy$ σε σχέση με τον αριθμό των $components$ παραμένει ελαφρώς σταθερό με μία τάση μείωσης.

Αξιοποίηση Τίτλου

Στην πλειονότητα των περιπτώσεων, ο τίτλος περιέχει λέξεις, που είναι άρρηκτα συνδεδεμένες με το περιεχόμενο του άρθρου. Άλλωστε, το γεγονός ότι επιλέγονται από ανθρώπους και όχι από μηχανή, σημαίνει πως επιλέγονται στοχευμένα και συνεπώς θα μπορούσαμε να πούμε οτι έχουν μεγαλύτερη βαρύτητα από το ίδιο το περιεχόμενο. Συνεπώς, για την αξιοποιήση της πληροφορίας αυτής , σε κάθε $Content$ επισυνάπτουμε $weight$ φορές τον τίτλο στο $content$ ώστε να ληφθεί υπόψιν στο $vectorization$ περισσότερο.

K-Nearest Neighbor

Υλοποιήσαμε δικό μας $implementation$ του K-Nearest Neighbor αλγορίθμου. Ο KNN βρίσκει τους k - γείτονες που απέχουν το λιγότερο (Ευκλείδεια απόσταση) από το εξεταζόμενο στιγμιότυπο δεδομένων

στη συνέχεια με το *majority voting* επιλέγει τη κατηγορία της οποίας βγάζουν το μέγιστο άθροισμα οι k γείτονες αυτοί. Για να παρουμε τα *metrics* με το 10-fold Cross Validation της βιβλιοθήκης $Scikit-learn$ χρειάστηκε να κάνουμε δικό μας *implementation* των μεθόδων: predict(), fit(), get_params() οι οποίες είναι απαραίτητες για να γίνει το *evaluation* του *custom* μοντέλου μας.

SVM

Στη μέθοδο κατηγοριοποίησης SVM χρησιμοποιήθηκε η κλάση SVC με κατάλληλες παραμέτρους C, *gamma* και *kernel*.
Η επιλογή τύπου *kernel* έγινε ανάμεσα σε *linear kernel* και rbf *kernel*. Γνωρίζουμε ότι αν ο αριθμός των χαρακτηριστικών είναι μεγάλος η βέλτιστη επιλογή *kernel* είναι αυτή του *linear*, διότι με τον rbf *kernel* ερχόμαστε αντιμέτωποι με μεγαλύτερη πολυπλοκότητα χρόνου.
Η επιλογή των τιμών για τις προαναφερόμενες παραμέτρους αρχικά έγινε πειραματικά, δοκιμάζοντας διαφορετικούς συνδυασμούς αυτών. Ωστόσο, όταν τα δεδομένα είναι αριθμητικά ο rbf είναι πιο αποδοτικός σε σχέση με τον *linear kernel*. Τέλος όταν τα δεδομένα μας είναι γραμμικά διαχωρίσιμα είμαστε σε θέση να γνωρίζουμε ότι ο *linear kernel* είναι σαφώς μία πιο συνετή επιλογή.
Έπειτα από πειραματισμούς οδηγηθήκαμε στο συμπέρασμα ότι η επιλογή rbf *kernel* είναι αποδοτικότερη σε σχέση με τον *linear kernel* (ως προς το *accuracy*). Τα παραπάνω μας προδιαθέτουν ότι τα δεδομένα μας δεν είναι πιθανώς γραμμικά διαχωρίσιμα.
Η επιλογή rbf *kernel* είναι άμεσα συνυφασμένη με την κατάλληλη επιλογή τιμών των παραμέτρους C και *gamma*. Η παράμετρος C αποτρέπει την εσφαλμένη ταξινόμηση των παραδειγμάτων εκπαίδευσης, σε βάρος της απλότητας της επιφάνειας απόφασης. Με άλλα λόγια, ένα μικρό C εξομαλύνει την επιφάνεια απόφασης, ενώ ένα μεγάλο C ταξινομεί όλα τα παραδείγματα με ορθό τρόπο. Από την άλλη το *gamma* ορίζει πόση επιρροή θα έχει ένα παράδειγμα εκπαίδευσης. Καθώς η βέλτιστη ανάθεση τιμών για τις παραμέτρους C και *gamma* ποικίλει ανάλογα με την έκδοση SVM που επιλέγεται κάθε φορά και φυσικά το εκάστοτε πρόβλημα, έγινε χρήση του $GridSearchCV$. Ο $GridSearchCV$ εξαντλητικά αναζητά και βρίσκει τον καλύτερο συνδυασμό παραμέτρων.

10-fold Cross Validation

Η μέθοδος αυτή χρησιμοποιείται για να μπορεί να γίνει το *evaluation* του μοντέλου μας αποφεύγοντας το $overfit$ (να γίνει, δηλαδή, το μοντέλο μας πολύ εξειδικευμένο στο συγκεκριμένο *train dataset*) ώστε να μην οδηγηθούμε σε ψευδή αποτελέσματα της αξιολόγησης του μοντέλου. Η στατιστική μέθοδος αυτή χωρίζει το *trainset* σε k κομμάτια και εκπαιδεύεται πάνω σε αυτά, και στη συνέχεια διασταυρώνει τα αποτελέσματα με τις προβλέψεις μεταξύ τους.

3. Beat the Benchmark

Classifier

Επηρεασμένοι από το *majority voting* του KNN δημιουργήσαμε έναν δικό μας $classifier$ που κάνει *majority vote* μεταξύ των $Classifiers$ που χρησιμοποιήθηκαν στο ερώτημα 2. Ο τίτλος είχε ήδη αξιοποιηθεί καθώς επίσης είχε γίνει και το *stemming*. Τελικά, κι έπειτα από πειραματισμούς, καταλήξαμε να ξεπεράσουμε την αποδοσή μας αποδίδοντας διάφορες τιμές στα ορίσματα των $classifiers$.

Σχολιασμός του κώδικα

Το ερώτημα 1. βρίσκεται στο αρχείο *WordCloud.py*

Το ερώτημα 2. βρίσκεται στο αρχείο *Classification.py* και χρησιμοποιεί τις κλάσεις *KNearestNeighbour* και *BeatThePenchmark.py*.

Στο αρχείο *Helpers.py* βρίσκονται οι βοηθητικές συναρτήσεις που χρησιμοποιούνται (για να είναι πιο ευανάγνωστος ο κώδικας).

Στο αρχείο *Model.py* βρίσκονται κάποια *enum* (δεν έχει να κάνει με τον *classifier*)

Στο αρχείο *Stemmer.py* βρίσκεται ο κώδικας που κάνει *stem* τα *dataset* μας.

Στον φάκελο *datasets* βρίσκονται τα αρχικά *dataset* και τα *stemmed dataset* (*.csv*)