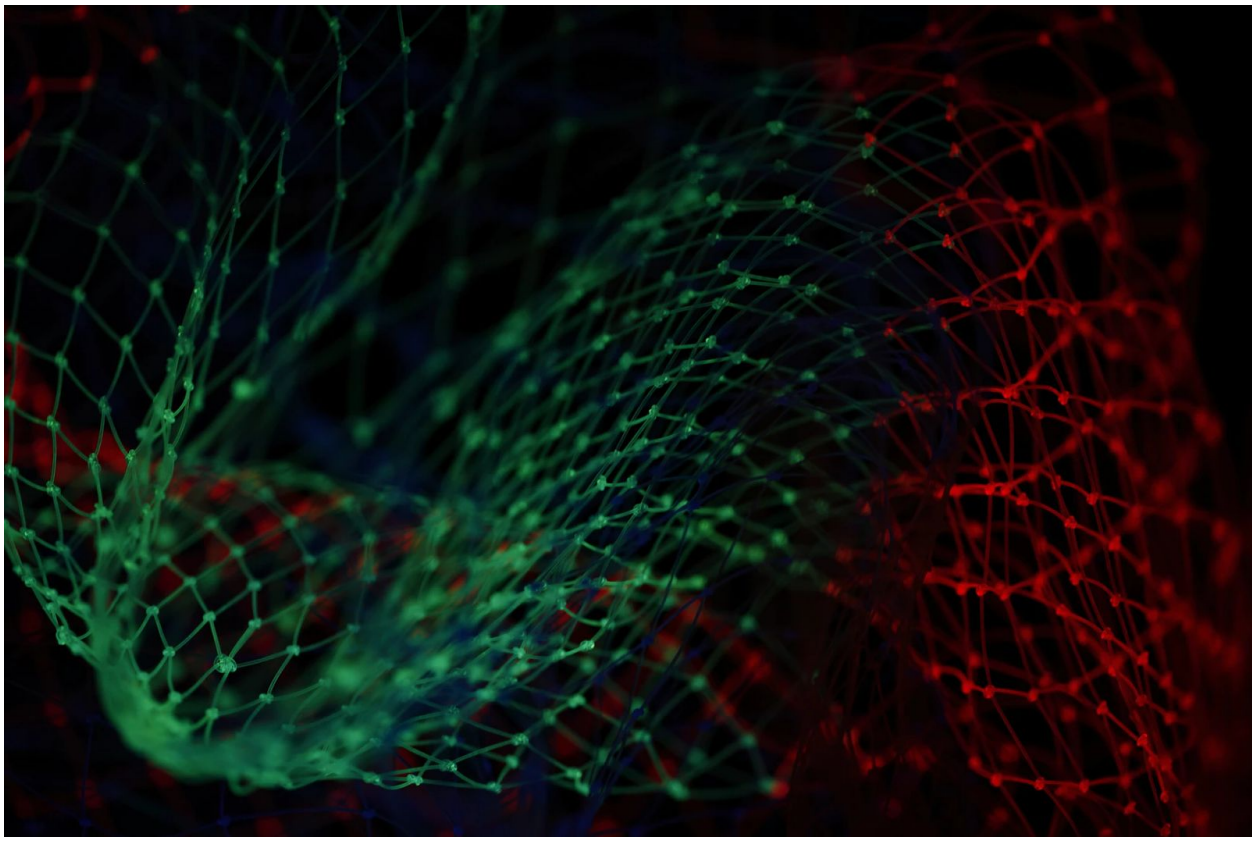
ΤΕΧΝΙΚΕΣ ΕΞΟΡΥΞΗΣ ΔΕΔΟΜΕΝΩΝ

1η Άσκηση



Μελίνα Σίσκου - 1115201200161
Ιωάννης Πελέλης - 1115201200146

Εαρινό Εξάμηνο 2017-2018

Ομαδική Εργασία (2 Ατόμων)

ΕΙΣΑΓΩΓΗ

Σκοπός της εργασίας είναι η κατηγοριοποίηση δεδομένων κειμένου με τη γλώσσα προγραμματισμού python χρησιμοποιώντας standard βιβλιοθήκες για machine learning. Ο τρόπος εργασίας ήταν οτι αρχικά εξοικειωθήκαμε με τις βιβλιοθήκες pandas, numpy, scikit και matplolib και στη συνέχεια εργαστήκαμε παράλληλα στα ίδια ερωτήματα. Έχουμε υλοποιήσει όλα τα ζητούμενα της άσκησης, με εξαίρεση τον classifier KNN. Επίσης έχουμε κάνει submit στο kaggle όπως περιγράφεται.

1. WordCloud

Για την δημιουργία των wordclouds αρκεί η εκτέλεση της παρακάτω εντολής.

```
python main.py w
```

Λεπτομέρειες υλοποίησης

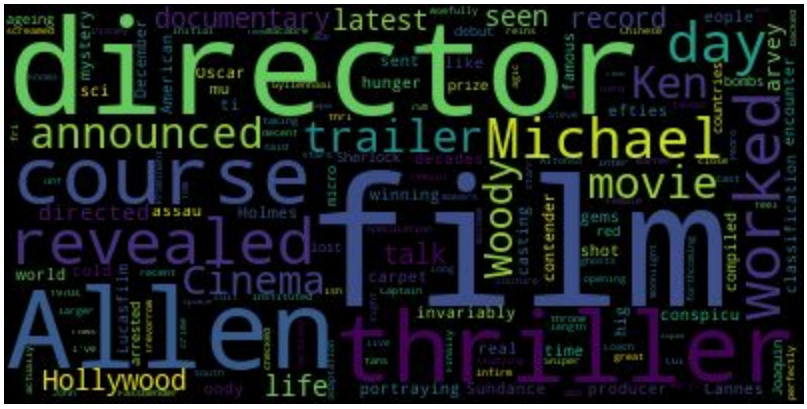
- Η παραπάνω εντολή θα χρησιμοποιήσει το αρχείο που βρίσκεται στο path από το TRAIN_SET_FILE και θα αποθηκεύσει τα αρχεία στο path όπως ορίζεται στη μεταβλητή IMAGE_FOLDER.
- Έχει γίνει αφαίρεση συχνά χρησιμοποιημένων λέξεων με τη χρήση της ENGLISH_STOP_WORDS
- Το wordcloud υπολογίζεται μόνο από το body των άρθρων (χωρίς τον τιτλο)

- Επειδή το word cloud ήταν το πρώτο ζητούμενο χρησιμοποιούμε επίσης κώδικα από το Categories.py. Αυτό το αρχείο χρησιμοποιείται μόνο στο wordcloud.

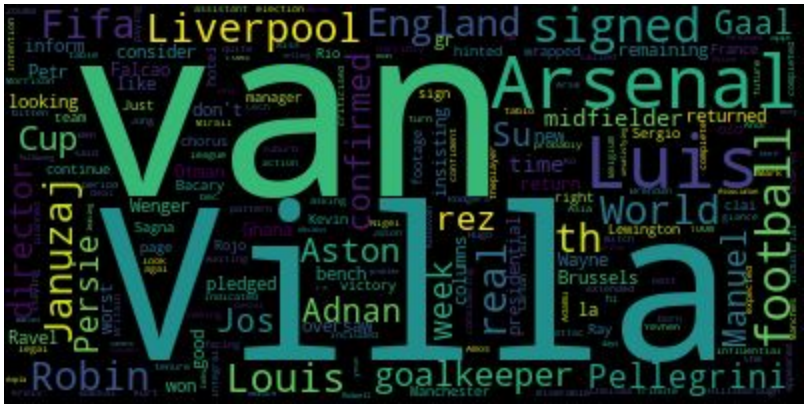
Ακολουθεί παράδειγμα εκτέλεσης:

```
[src $pythonw main.py w
generated the image for Politics. Check it at ../output/images/Politics.png
generated the image for Film. Check it at ../output/images/Film.png
generated the image for Football. Check it at ../output/images/Football.png
generated the image for Business. Check it at ../output/images/Business.png
generated the image for Technology. Check it at ../output/images/Technology.png
```

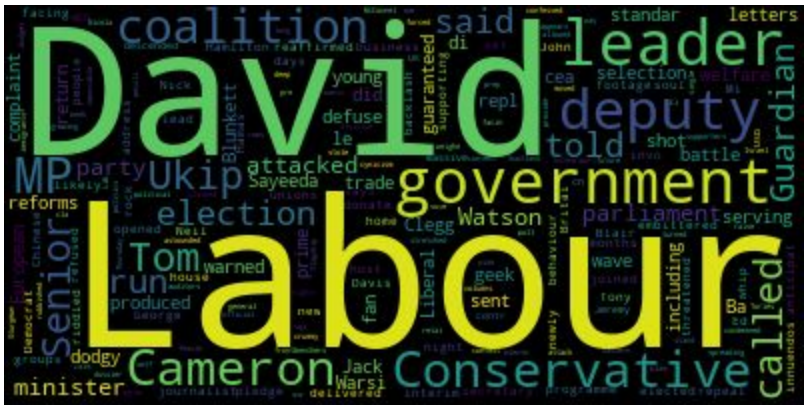
Ακολουθούν οι εικόνες που δημιουργούνται:



Film category

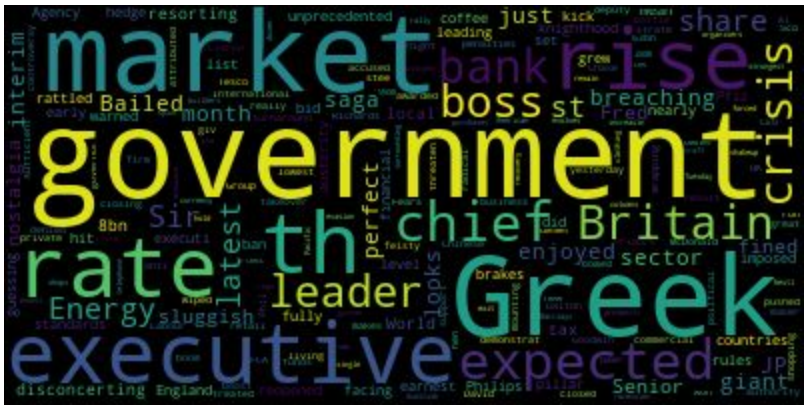


Football category



Politics category

Technology category



Business category

2. Υλοποίηση Κατηγοριοποίησης (Classification)

Το συγκεκριμένο ερώτημα επικεντρώνεται στην εξοικείωση με την διαδικασία κατηγοριοποίησης χρησιμοποιώντας διαφορετικούς classifiers. Για να δοκιμαστούν όλοι οι classifiers και να παράγουμε το αρχείο EvaluationMetric_10fold.csv, εκτελούμε την παρακάτω εντολή:

```
python main.py all
```

```
[src $pythonw main.py all
Starting to generate metrics for bayes
Starting preprocess using 10 features for bayes
Starting k-fold validation for bayes
For features: 10 the accuracy: 0.87787, precision: 0.86493, recall: 0.87821, fmeasure: 0.86579

Starting to generate metrics for random_forests
Starting preprocess using 50 features for random_forests
Starting k-fold validation for random_forests
For features: 50 the accuracy: 0.94856, precision: 0.94555, recall: 0.94217, fmeasure: 0.94376

Starting to generate metrics for svm_clf
Starting preprocess using 350 features for svm_clf
Starting k-fold validation for svm_clf
For features: 350 the accuracy: 0.96804, precision: 0.966, recall: 0.96571, fmeasure: 0.96585

Writing the metrics to EvaluationMetric_10fold.csv
Bye
src $
```

Λεπτομέρειες υλοποίησης

1. Μπορούμε να εκτελέσουμε κάθε αλγόριθμο ξεχωριστά, αλλάζοντας το “all” σε “b” για *Multinomial Naive Bayes*, “rf” για *Random Forests*, “svm” για *Support Vector Machines (*πχ *python main.py svm)*
2. Όταν εκτελούμε έναν συγκεκριμένο αλγόριθμο δοκιμάζουμε επίσης διαφορετικό αριθμό features και δημιουργούμε το plot accuracy/#features. Επίσης δε δημιουργείται το αρχείο EvaluationMetric_10fold.csv. Ωστόσο τα αποτελέσματα των μετρικών εκτυπώνονται στο terminal.
3. Έχει γίνει αφαίρεση συχνά χρησιμοποιημένων λέξεων με τη χρήση της ENGLISH_STOP_WORDS
4. Όλο το training και οι δοκιμές γίνονται στην συνάρτηση generate_test_metrics. Εκεί έχουμε διάφορα conditions (πχ αν βρισκόμαστε σε test) και υπολογίζουμε την αποτελεσματικότητα του κάθε αλγορίθμου.
5. Στον SVM δοκιμάσαμε διάφορες περιπτώσεις και αυτή που εμφάνισε καλύτερα αποτελέσματα είναι με LinearSVC.

Outline για τη διαδικασία του classification

1. Δημιουργούμε έναν πίνακα όπου κάθε στοιχείο είναι ο τίτλος μαζί με το body. Δίνουμε βαρύτητα στον τίτλο γράφοντας τον 10 φορές μπροστά πριν από το body.
2. Δημιουργούμε με τη μορφή TF-IDF τα features που αφορούν το training set (TfidfVectorizer).
3. Μειώνουμε τις διαστάσεις των features χρησιμοποιώντας τον αλγόριθμο SVD (TruncatedSVD), έχοντας optimal αριθμό διαστάσεων ανάλογα τον classifier (περιγράφεται στο 'accuracy graphs').
4. Κάνουμε το prediction.
5. Για να αποφύγουμε overfitting του μοντέλου στα δεδομένα χρησιμοποιούμε kfold validation όπως έχει προταθεί.

Accuracy graphs

Αλλάζοντας των αριθμό των διαστάσεων όταν κάνουμε το dimensionality reduction, παρατηρούμε άμεση επίδραση στην αποδοτικότητα των classifiers. Η μετρική που χρησιμοποιούμε για να μελετήσουμε την αποδοτικότητα, είναι το accuracy το οποίο ορίζεται από τον αριθμό των σωστών προβλέψεων σε σχέση με τον αριθμό των συνολικών προβλέψεων. Επομένως για να παρατηρήσουμε την επίδραση του αριθμού των dimensions σε σχέση με την αποδοτικότητα του μοντέλου, δημιουργήσαμε ορισμένα γραφήματα #dimensions / accuracy.
Για την δημιουργία του κάθε accuracy graph εκτελούμε

Για Multinomial Naive Bayes classifier

```
python main.py b
```

Για Random Forests classifier

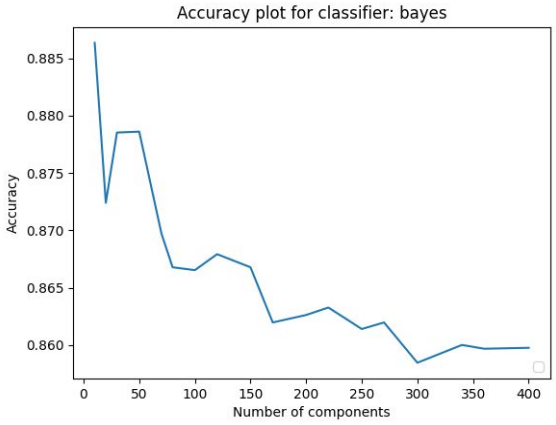
```
python main.py rf
```

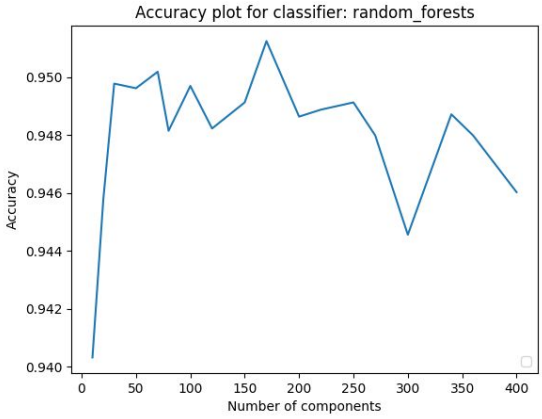
Για Support Vector Machines classifier

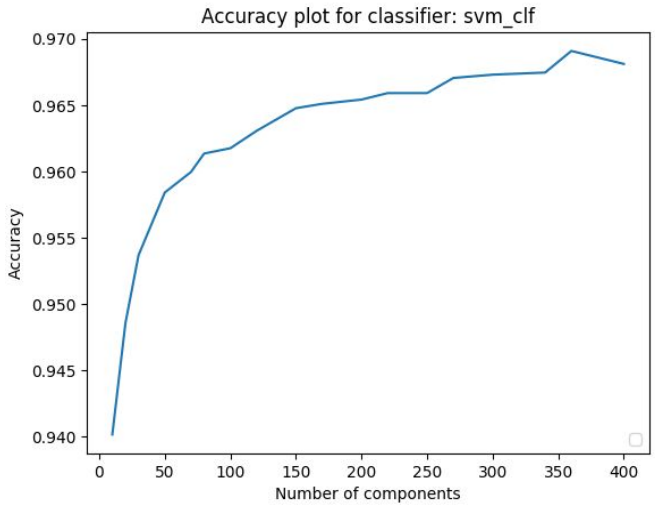
```
python main.py svm
```

Για my method classifier

```
python main.py my
```







Ενδιαφέρον παρουσιάζει η μεγάλη διαφοροποίηση των γραφημάτων ανάλογα με τον classifier που χρησιμοποιούμε. Επίσης το scikit προτείνει οτι 100 είναι ένας καλός αριθμός components να χρησιμοποιούμε, αλλά όπως φαίνεται είναι σημαντικό να επιλέγουμε τον αριθμό βάση του αλγορίθμου που χρησιμοποιούμε.

3. Beat the benchmark

Στο προηγούμενο ερώτημα χρησιμοποιούμε τους classifiers με τις default/προτεινόμενες ρυθμίσεις. Προκειμένου να καταλήξουμε στο αποδοτικότερο αλγόριθμο κάνουμε μια σχετική προεπεξεργασία στο κείμενο και δοκιμάσαμε διαφορετικές ρυθμίσεις στον classifier.

- Αρχικά ο αποδοτικότερος αλγόριθμος υπήρξε ο SVM.
- Για τον svm δοκιμάσαμε διαφορετικές ρυθμίσεις και καταλήξαμε στο linear kernel.
- Κάνοντας τεστ βρήκαμε οτι μια καλή σχέση απόδοσης και dimensions είναι από 200 μέχρι 300 για τον SVM. Ουσιαστικά δεν είναι πολλά components και βρίσκεται στο σημείο που η λογαριθμική καμπύλη αρχίζει να 'σταθεροποιείται'.
- Επαναλαμβάνουμε τον τίτλο αρκετές φορές πριν από το body του κειμένου, με σκοπό να του δώσουμε μεγαλύτερη βαρύτητα. Όπως παρατηρούμε αυτό αυξάνει το accuracy των predictions.

4. Αρχεία εξόδου

EvaluationMetric_10fold.csv

Για την παραγωγή του παρακάτω πίνακα εκτελούμε την παρακάτω εντολή:

```
python main.py all
```

Statistic Measure	Naive Bayes	Random Forest	SVM	my_method
Accuracy	0.87357	0.94861	0.95432	**0.96085**
Precision	0.86006	0.94322	0.94624	**0.95447**
Recall	0.8643	0.94306	0.94824	**0.95929**
F-Measure	0.85908	0.94308	0.94708	**0.9566**

Ο παραπάνω πίνακας χρησιμεύει για να βρούμε τον κατάλληλο classifier σύμφωνα με την απόδοση του. Όπως φαίνεται τις καλύτερες μετρικές τις έχει ο svm τον οποίο και τελικά

χρησιμοποιούμε στο prediction του test αρχείου.

testSet_categories.csv

Για την παραγωγή του αρχείου εκτελούμε την παρακάτω εντολή:

```
python main.py test_categories
```

Αυτό θα εκτελέσει τον κώδικα που αφορά το αλγόριθμο με τις ρυθμίσεις που καταλήξαμε από το beat the benchmark.

Το αρχείο testSet_categories.csv βρίσκεται στον φάκελο csv/

Για την παραγωγή του αρχείου που χρειάζεται για το kaggle εκτελούμε:

```
python main.py test_kaggle
```

Accuracy από kaggle: 73.704%