

# Τεχνικές Εξορυξης Δεδομενων

## Ασκηση 1

ΓΚΑΡΑΓΚΑΝΗΣ ΕΥΑΓΓΕΛΟΣ – 1115201400033

ΚΟΤΡΩΝΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ – 1115201400074

## Δημιουργία Wordcloud

Χρησιμοποιησαμε την βιβλιοθηκη wordcloud για την υλοποιηση του ερωτηματος. Προσθεσαμε στα STOPWORDS της βιβλιοθηκης επιπλεον λεξεις , οι οποιες αλλοιωναν την ορθοτητα του wordcloud.



## Παραδειγματας εκτελεσης του Wordcloud.py

### Αναφορες για το ερωτημα :

- Χρησιμοποιηθηκε η βιβλιοθηκη wordcloud , για την οποια πηραμε πληροφοριες απο [https://github.com/amueller/word\\_cloud](https://github.com/amueller/word_cloud).
    - Εκτελεστηκε επιτυχως σε linux , ubuntu 16.04

## Υλοποιηση Συσταδοποιησης ( Clustering )

Για την υλοποιηση του clustering χρησιμοποιησαμε την βιβλιοθηκης της sklearn , και συγκεκριμενα απο την sklearn.cluster πηραμε τον Kmeans. Η kmeans χρησιμοποιει την ευκλειδεια αποσταση. Χρησιμοποιησαμε τον TfidfVectorizer της sklearn, γιατι κριναμε οτι ειναι πιο αποδοτικος απο τους υπολοιπους.

K-means: Clustering.	
K:	number of clusters
1.	Place K points in the space represented by the objects that are being clustered. These points represent initial group centroids
2.	Assign each object to the group that has the closest centroid.
3.	When all objects have been assigned, recalculate the position of the K centroids
4.	Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the data into distinct groups that have well defined boundaries.

Παραπανω ακολουθεια η επεξηγηση του τροπου λειτουργιας του Kmeans .

Παρακατω παρουσιαζεται το πως ο αλγοριθμος συσταδοποιει τα δεδομενα.

clustering_KMeans.csv - LibreOffice Calc						
	A	B	C	D	E	F
1		Politics	Film	Football	Business	Technology
2	Cluster 1	0.0028248588	0.9745762712	0.0084745763	0	0.0141242938
3	Cluster 2	0.2463235294	0.0147058824	0.0992647059	0.2113970588	0.4283088235
4	Cluster 3		0	0.0020746888	0.9958506224	0
5	Cluster 4	0.0539772727		0	0.0028409091	0.0028409091
6	Cluster 5	0.9925373134		0	0.0074626866	0
7						

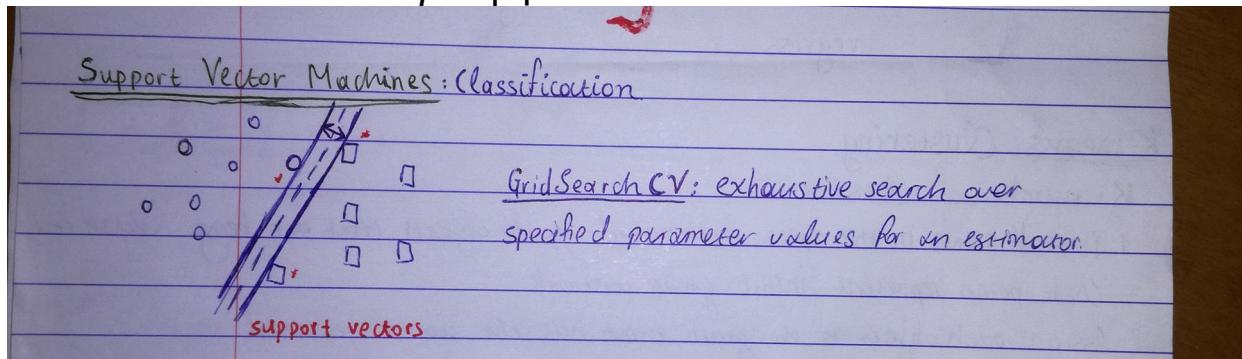
Τα αποτελεσματα του clustering ειναι ικανοποιητικα σαφως , πραγμα που δηλωνει την ακριβης συσταδοποιηση των δεδομενων.

## Υλοποιηση Κατηγοριοποιησης ( Classification )

Σε αυτό το ερωτημα χρησιμοποιησαμε τις εξης μεθοδους Classification:

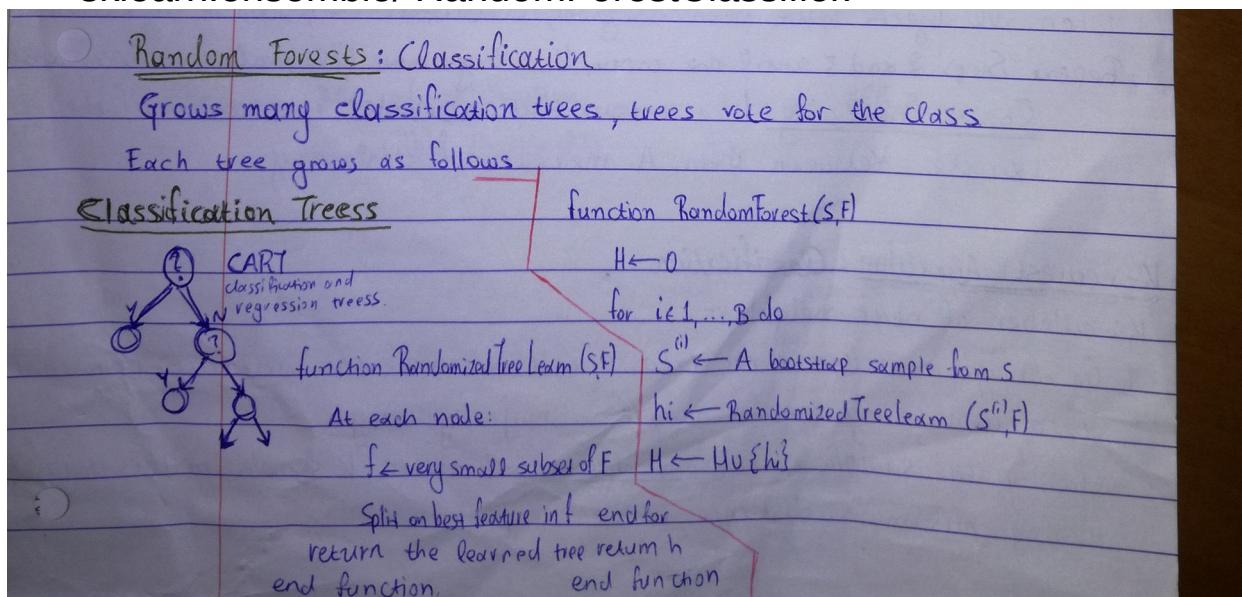
- **SVM :**

Για τον support vector machines χρησιμοποιησαμε την βιβλιοθηκη sklearn.svm και συγκεκριμενα τον LinearSVC.



- **Random Forests**

Στη συγκεκριμενη περιπτωση χρησιμοποιησαμε την sklearn.ensemble/ RandomForestClassifier.



- **Naive Bayes**

Naive Bayes: Classification

$$2 \text{ classes} \left\{ \begin{array}{l} 0 \\ 1 \end{array} \right. \quad \frac{P(1|x)}{P(0|x)} = \frac{P(x|1)P(1)}{P(x|0)P(0)}$$

likelihood      class prior probability

posterior probability

of class

$P(c|x) = P(x|c) \cdot P(c) / P(x)$  predictor prior probability

}

1 Handle data: split training and test set

2 summarize the properties

3 make a prediction

4 make predictions

5. evaluate accuracy

6. tie it together

Για τον naive bayes χρησιμοποιησαμε τον MultinomialNB , απο την sklearn.naive\_bayes

- **K-Nearest-Neighbor**

Δικη μας υλοποιηση. Παρακατω ακολουθει μια αναφορικη επεξηγηση της knn . Η knn μας χρησιμοποιει μια δικια μας συναρτηση για την μετρηση των αποστασεων μεταξυ των δεδομενων. Η εξηγηση της λειτουργιας της υλοποιησης μας εξηγηται με καθοδηγητικα σχολια .

k-nearest Algorithm: Classification

K: number of near neighbors

1. Go through each item in my dataset, and calculate the "distance" from that data item to my specific sample. → majority voting
2. Classify the sample as the majority class between K samples in the dataset having minimum distance to the sample.

Για την αξιολογηση των μεθοδων κατηγοριοποιησης χρησιμοποιηθηκαν οι εξης μετρικες συναρτησεις :

➤ **Recall \ Accuracy \ Precision \ F-Measure :** Χρησιμοποιησαμε την ετοιμη συναρτηση cross\_val\_score απο την βιβλιοθηκη sklearn.model\_selection. Η βιβλιοθηκη αυτη αναλαμαμβανει τοσο

την χρηση των μετρικων οσο και το cross folding για την παραγαωγη ακριβεστερων αποτελεσματων. Μια σχετικη αναλυση/ επεξηγηση των συναρτησεων γινεται απο παρακατω :

### Precision/Recall /F-measure

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

how many returned documents are correct

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

how many positives does the model return

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{Precision} + \text{recall}}$$

Used in document retrieval

### Accuracy

a, d : good predictions

c, b: bad prediction.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total of all cases to be predicted}} = \frac{a+d}{a+b+c+d}$$

		True Positive
a	b	False Negative
c	d	True Negative False Positive

και οι 4 συναρτησεις χρησιμοποιουν μια μαθηματικη αναλογια , αναμεσα στα documents και κατα ποσο ειναι ευστοχη η προβλεψη μας και η κατηγοριοποιηση μας , πραγμα που τις κανει αντιπροσωπευτικες συναρτησεις αξιολογησης των αλγοριθμων μας.

## ROC

It is a plot. True Positive Rate = true positives/all positives

False Positive Rate = false positives/all negatives

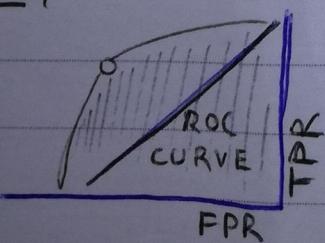


Thresholds.

ROC curve visualizes  
all possible thresholds

minimize FPR  
maximize TPR

## AUC, area under the curve



AUC is a useful metric when your classes  
are highly unbalanced.

## Latent Semantic Indexing

Analyzes relationships between a set of documents and terms they contain by producing a set of concepts related to the documents and terms.

➤ **ROC / AUC Score** : Εχουμε κατασκευασει δικη μας μετρικη συναρτηση , η οποια χρησιμοποιει την ιδια cross\_fold.

Παραπανω ακολουληθει μια σχετικη αναλυση των μετρικων συναρτησεων. Εδω καλυπτουμε την γεωμετρικη αναπαρασταση των αποτελεσματων της αξιολογησης. Η συναρτησεις που εχουμε δημιουργησει δημιουργουν ενδεικτικα αποτελεσματα διαφορετικων ROC curves , που αξιολογουν τις διαφορες μεθοδους classification.

\* Οι παραπανω μετρικες χρησιμοποιηθηκαν για την αξιολογηση ολων των μεθοδων κατηγοριοποιησης πλην της KNN. Για την KNN το cross fold 10 γινεται με δικη μας υλοποιηση , για λογους συμβατοτητας και οι μετρικες που χρησιμοποιηθηκαν ειναι οι ιδες με τις παραπανω.

Ακολουθει η ενδεικτικη αναφορα των μετρικων αποτελεσματων.

	Naive Bayes	Random Forest	SVM	KNN
Accuracy	0.8769583279	0.916	0.97	0.71
Precision	0.9290129443	0.9200943707	0.9700675596	0.6519047619
Recall	0.8769583279	0.8928017256	0.9670562392	0.7004166667
F-Measure	0.8867681392	0.9087304137	0.9680536062	0.6350793651
AUC	0.9270420217	0.9395565617	0.9797463189	0.8265674603

Σε ολες τις μετρικες , εχει εφαρμοστει 10-fold επαληθευση. Χωριζουμε τα κειμενα μας σε train sets/test sets αρκετες φορες , για να εχουμε μια πιο σφαιρικη αναφορα στον βαθμο επιτυχιας της κατηγοριοποιησης πανω σε κειμενα.

Τα αποτελεσματα ειναι ικανοποιητικα , αφου εχουμε ενα ποσοστο επιτυχιας ανου του 80% ανα μεσο ορο , αν εξαιρεσουμε την KNN η οποια ειναι δικια μας υλοποιηση και ειναι λιγο χειροτερη.

*Για την δοκιμή του classification στα test\_sets , εχουμε :*

Χρησιμοποιησαμε Naive Bayes . Ας δουμε , λοιπον , τα αποτελεσματα των predictions μας για τα test sets.

	ID	Predicted_Category
0	15333	Football
1	15332	Football
2	15325	Technology
3	15322	Football
4	15321	Football
5	15315	Politics
6	15305	Football
7	15304	Football
8	15302	Football
9	15299	Football
10	15298	Film
11	15290	Football
12	15284	Football
13	15278	Technology
14	15276	Business
15	15273	Technology
16	15272	Football
17	15269	Football
18	15267	Football
19	15264	Football
20	15260	Football
21	15256	Football
22	15251	Football
23	15249	Football
24	15239	Football
25	15236	Football
26	15234	Football
27	15228	Football
28	15225	Football
29	15224	Football
...	...	...
3037	173	...
3038	171	Politics
3039	170	Technology
3040	167	Business
3041	162	Business
3042	156	Technology
3043	152	Technology
3044	140	Business
3045	139	Business
3046	115	Technology
3047	112	Technology
3048	107	Technology

**Beat the  
benchmark**

- clustering:

-Δοκιμασαμε να κανουμε προεπεξεργασια στα δεδομενα μας τοσο με τον TfidfVectorizer , οσο και με τον Count Vectorizer.

Ο tfidf φανηκε να δινει ακριβεστερα αποτελεσματα οσο αναφορα την σωστη συσταδοποιηση των κειμενων , ενω ο count vectorizer βοηθεια στην ταχυτερη προ επεξεργασια και συσταδοποιηση των δεδομενων.

- Classification:

Οσο αναφορα το classification φροντισαμε :

- Δοκιμασαμε διαφορετικες παραλλαγες των classification methods που μας προτεινατε. Χρησιμοποιησαμε τον LinearSVC εναντι του απλου SVC , το οποιο μας οδηγησε σε πολυ καλυτερη κατηγοριοποιηση των κειμενων.
- Δοκιμασαμε την decision tree , επισης , και τν αξιολογησαμε με 10-fold accuracy.

```
[ 0.6      0.63333333  0.7      0.63333333  0.66666667  0.8
 0.66666667  0.76666667  0.83333333  0.76666667]
socialdeveloper@socialdeveloper-535U4C:~/Desktop/Yliko iis ergasias 2016-2017$ 
socialdeveloper@socialdeveloper-535U4C:~/Desktop/Yliko iis ergasias 2016-2017$ 
socialdeveloper@socialdeveloper-535U4C:~/Desktop/Yliko iis ergasias 2016-2017$ 
socialdeveloper@socialdeveloper-535U4C:~/Desktop/Yliko iis ergasias 2016-2017$ 
socialdeveloper@socialdeveloper-535U4C:~/Desktop/Yliko iis ergasias 2016-2017$ ]
```

```
from sklearn import tree
Tree = Pipeline([
    ('vectorizer', TfidfVectorizer(stop_words='english')),
    ('classifier', tree.DecisionTreeClassifier()) ])

result = model_selection.cross_val_score(Tree, X, Y, cv= kfold, scoring='accuracy')
print(result)
```

Python ▾ Tab Width: 8 ▾ Ln 202, Col 8 ▾ INS

- Βγαλαμε πολλα πιθανα ROC CURVES , ενα για καθε fold , για ενα ικανο αριθμο αλγοριθμων classification