# DATA ANALYTICS – REPORT

## Requirement 1 : Question 1a :

Σε αυτό το ερώτημα ο σκοπός ήταν να δημιουργήσουμε τέσσερα **WordClouds** για τέσσερις διαφορετικές κατηγορίες (Business, Entertainment, Health, Technology).

**WordCloud** : ονομάζεται μια εικόνα ή αλλιώς ένα σύννεφο που είναι γεμάτο από λέξεις σε διαφορετικά μεγέθη, τα οποία αντιπροσωπεύουν την συχνότητα ή το πόσο σημαντική είναι κάθε λέξη.

Για την υλοποίηση του κώδικα χρησιμοποιήσαμε τις εξής βιβλιοθήκες :

**numpy** : είναι πολύ χρήσιμη βιβλιοθήκη και χρησιμοποιείται για πολυδιάστατους πίνακες.

**pandas** : σε συνδυασμό με την numpy βιβλιοθήκη χρησιμοποιείται για την ανάλυση δεδομένων (Data Analysis).

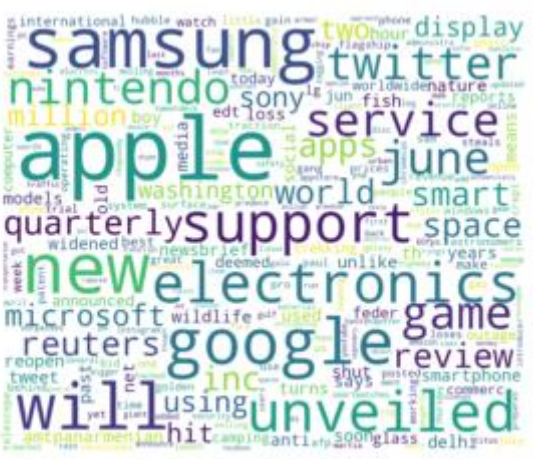
**python os**

**matplotlib** : χρησιμοποιείται για την απεικόνιση, επιτρέποντας σε διάφορες βιβλιοθήκες να τρέχουν και να σχεδιάζουν στη βάση τους συμπεριλαμβανομένου του WordCloud.

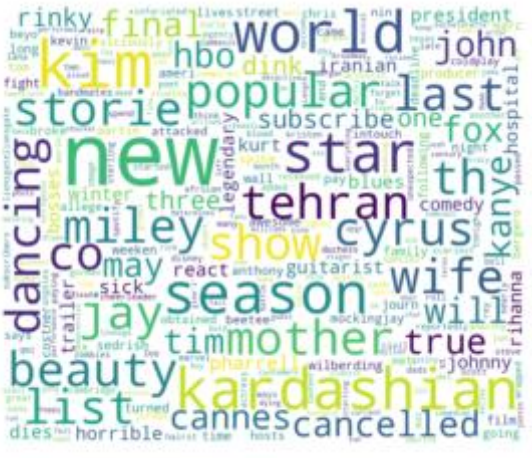
**wordcloud** : χρησιμοποιείται για να εξάγει το WordCloud με τα StopWords και διάφορα χρώματα που χρησιμοποιούνται για να χρωματιστούν οι λέξεις.

Στη συνέχεια, ξεκινώντας τον κώδικα χρησιμοποιήσαμε με την βοήθεια της βιβλιοθήκης pandas την εντολή *df = pd.read_csv("train.csv");* φορτώσαμε τα δεδομένα (111795). Παραθέτουμε στην συνέχεια τα WordClouds για τις τέσσερις κατηγορίες.

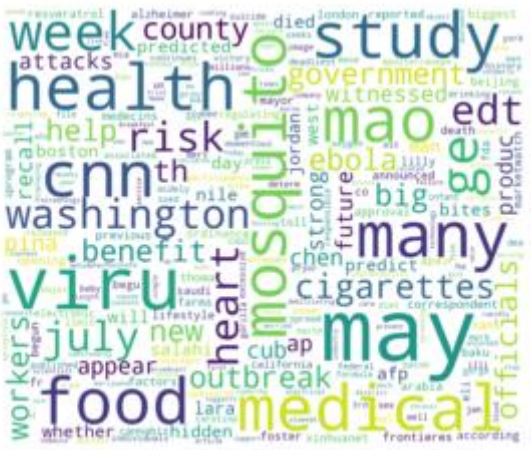
- **Business** : βλέπουμε ότι η λέξη apple είναι αυτή που αναφέρεται πιο συχνά και οι λέξεις samsung και google είναι οι πιο δημοφιλείς. Αυτό συμβαίνει λόγου του γεγονότος ότι η samsung και η google είναι μια παγκοσμίως αναγνωρισμένες εταιρίες.



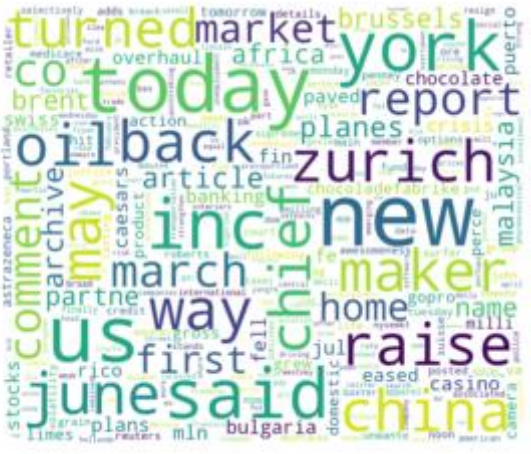

- **Entertainment** : βλέπουμε ότι η λέξη new και η λέξη world είναι αυτές που αναφέρονται πιο συχνά και οι λέξεις kardashian και kim είναι αυτές που είναι οι πιο δημοφιλείς λόγο του γεγονότος ότι είναι παγκοσμίως αναγνωρισμένα ονόματα.




- **Health** : βλέπουμε ότι οι λέξεις health, food και medical είναι κάποιες από τις λέξεις που αναφέρονται πιο συχνά.




- **Technology** : βλέπουμε ότι οι λέξεις new και today είναι αυτές που αναφέρονται πιο συχνά και οι λέξεις zurich και york είναι οι πιο αναγνωρισμένες λόγο του γεγονότος ότι σε αυτές τις πόλεις έχει αναγνωρισμένα τεχνολογικά πανεπιστήμια.




Τέλος, έχουμε χρησιμοποιήσει μια λίστα με **StopWords** για να αποφεύγονται οι πιο κοινές λέξεις όπως and, or, who, an, at κ.ο.κ.

## Requirement 1 : Question 1b

Σε αυτό το ερώτημα χρειάζεται να κάνουμε **classification** (ταξινόμηση) στα δεδομένα του είδους που συναντήσαμε στο πρώτο ερώτημα. Χρησιμοποιούμε δύο τεχνικές :

- **Support Vector Machines (SVM)** και
- **Random Forests**

βάσει των χαρακτηριστικών :

1. **Bag of Words** και
2. **SVD**

Σαν έξοδος δημιουργείται ένα αρχείο κειμένου το οποίο περιέχει το ID των αρχείων και την προβλεπόμενη κατηγορία στην οποία ανήκουν με βάση την ανάλυση που έχει γίνει στα δεδομένα που δίνονται σαν είσοδος.

Μας παρέχονται το αρχείο *train_set* (περίπου 111795 γραμμές) και το αρχείο *test_set* (47912 γραμμές). Το πρώτο αρχείο χρησιμοποιείται από το σύστημα μηχανικής μάθησης για «εκπαίδευση» του αλγορίθμου και το δεύτερο που δεν περιέχει τη στήλη *Label* χρησιμοποιείται για πρόβλεψη και κατηγοριοποίηση των δεδομένων κειμένου (στην περίπτωσή μας των ειδήσεων) ανάλογα με την κατηγορία που ανήκουν.

Στο τέλος παράγουμε κάποιες μετρικές για αξιολόγηση των χρησιμοποιουμένων μοντέλων (*Accuracy*, *Precision*, *Recall*, *F-Measure*). Στο *Beat the Benchmark* γινεται υλοποίηση μιας μεθόδου *"My Method"* και σύγκριση των αποτελεσμάτων της με τις άλλες τέσσερις μεθόδους (με χρήση των *train set* και *5-fold cross validation*).

| Statistic Measure | SVM (SVD) | Random Forest (SVD) | SVM (BoW) | Random Forest (BoW) | My Method |
|---|---|---|---|---|---|
| Accuracy | 0.870186 | 0.850485 | 0.81985 | 0.847851 | 0.897053 |
| Precision | 0.88005 | 0.872741 | 0.863076 | 0.895289 | 0.898527 |
| Recall | 0.852638 | 0.815485 | 0.768368 | 0.796875 | 0.870975 |
| F-Measure | 0.854403 | 0.82949 | 0.790277 | 0.820417 | 0.880758 |

Οπως φαινεται απο τα αποτελέσματα η προσέγγιση αυτή υπερέχει σε σχέση με τις προηγούμενες.

Δημιουργήθηκαν δύο ξεχωριστά αρχεία κώδικα το πρώτο για να παραχθούν οι μετρικές για τις προσεγγίσεις **SVM** και **Random Forest** (*1b_evaluation.py*) και στο δεύτερο αρχείο γράφτηκε ο κώδικας **My Method** για παραγωγή των προβλέψεων (*1b_classification.py*)

Η μέθοδος *“cross validation”* χρησιμοποιείται στη μηχανική μάθηση για να εκτιμηθεί η απόδοση του μοντέλου σε καινούρια δεδομένα. Χρησιμοποιήθηκε η εντολή

*kf = StratifiedKFold (n_splits=5, random_state=123)*

Στη συνέχεια στο αρχείο *evaluation* τραβάμε τα δεδομένα από τις στήλες τους, τα οποία έχουν “φορτωθεί” πριν από το αρχείο *csv*.

```
train_data = pd.read_csv('train.csv',header=0, nrows=400)

# Drop useless columns #
#train_data = train_data.drop(['RowNum', 'Id', 'Title'], axis=1)

# 5-fold #
kf = StratifiedKFold(n_splits=5, random_state=123)

y_train = train_data["Label"]
X_train = train_data["Content"]

# Add labels #
le = preprocessing.LabelEncoder()
X_train_le = le.fit_transform(y_train)
X_train_cat = le.inverse_transform(X_train_le)

# Create matrix of TF-IDF features #
tfidf_vectorizer = TfidfVectorizer(stop_words=ENGLISH_STOP_WORDS)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
```

Μετά γίνεται η επεξεργασία του κειμένου- εισόδου με *Label Encoding*. Μετατρέπουμε τις λέξεις σε αριθμούς που διαβάζονται από τον αλγόριθμο μηχανικής μάθησης.

Στη συνέχεια όπως φαίνεται στον κώδικα, φτιάχνουμε τον *Vectorizer*. Μετατρέπεται με αυτό τον τρόπο μια συλλογή κειμένου σε ένα πίνακα **TF-IDF**, δηλαδή ένα διάνυσμα . Πρακτικά είναι σαν να έχουμε ένα λεξικό που μετατρέπει και αντιστοιχεί κάθε λέξη σε ένα *index* του πίνακα. Γίνεται αφαίρεση των *StopWords* αποθηκεύοντας τα σε ένα διαφορετικό διάνυσμα.

```
# Use LSA for dimensionality reduction #
svd = TruncatedSVD(n_components=100, random_state=123)

# Perform dimensionality reduction #
X_train_reduced = svd.fit_transform(X_train_tfidf)

# Choose classifiers to measure #

clf_RF = RandomForestClassifier(n_estimators=100, criterion="entropy", bootstrap=Fal
clf_SVC = SVC(C=5, kernel="linear", gamma="auto", random_state=123)
        # TruncatedSVD(n_components=2000, n_iter=5, random_state=42)

clfs = [ clf_RF, clf_SVC, clf_RF, clf_SVC ]
clfs_names = ["Random Forest SVD", "SVM SVD","Random Forest BoW", "SVM BoW"]

# Metrics #
scoring = ['precision_macro', 'recall_macro', 'f1_macro', 'accuracy']
```

Για τη μέθοδο **Random Forest** και **SVM** θα πρέπει να εισάγουμε δυο διαφορετικών είδους “*classifiers*” και χρησιμοποιούμε κάθε φορά αυτόν που απαιτείται. Ο αριθμός *“estimators”* που δίνεται σαν παράμετρος στην πρώτη συνάρτηση

*“classsifier”* για **Random Forest** αποτελεί τον αριθμό των δέντρων αποφάσεων που χρησιμοποιούνται στον αλγόριθμο και στο παράδειγμά μας επιλέξαμε να είναι 100.

Ο στόχος του **SVC** *“Classifier*ι” είναι να ταιριάξει τα δεδομένα που του δίνουμε, επιστρέφοντας τον καλύτερο δυνατό διαχωρισμό των δεδομένων με τη χρήση ενός υπερεπιπέδου (*hyperplane*). Δίνοντας κάποια *“features”* στον *“classifier”* μπορούμε να κάνουμε τροποποιήσεις στο μοντέλο πχ χρησιμοποιεί *“kernel=linear”*.

Έπειτα αποθηκεύουμε τους *“classifiers”* σε ένα πίνακα για ευκολότερη προσπέλαση.

Στις περιπτώσεις που απαιτείται να χρησιμοποιήσουμε την τεχνική **Singular Value Decomposition** ( **SVD**), μας χρησιμεύουν οι πρώτες δύο γραμμές κώδικα του παραπάνω παραδείγματος. Ο κώδικας αυτός επιτυγχάνει το να κάνει dimensionality *“reduction”*, δηλαδή εφαρμόζει μια μέθοδο αποσύνθεσης ενός πίνακα στις συνιστώσες του, έτσι ώστε να γίνουν κάποιοι υπολογισμοί απλούστεροι.

Στη συνέχεια του αρχείου γίνονται οι μετρήσεις που χρειαζόμαστε και αποθηκεύονται και εξάγονται στο αρχείο *"EvaluationMetric_5fold.csv"*.

```
7
8   import pandas as pd
9   import numpy as np
10  from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
11  from sklearn.feature_extraction.text import TfidfVectorizer
12  from sklearn import preprocessing
13  from sklearn.preprocessing import Normalizer
14  from sklearn.svm import SVC
15  from sklearn.decomposition import TruncatedSVD
16  from sklearn.model_selection import cross_validate
17  from nltk import PorterStemmer
18  from sklearn.model_selection import StratifiedKFold
19  import time
20
21  #start = time.time()
22  #print("hello")
23
24
25  pd.set_option('precision', 6)
26
27  # Use porter stemmer #
28  #stemmer = PorterStemmer()
29
30  # 5-fold #
31  kf = StratifiedKFold(n_splits=5, random_state=123)
32
33
34
35  train_data = pd.read_csv('train.csv',header=0, nrows=1000)
36  test_data = pd.read_csv('test_without_labels.csv',header=0, nrows=1000)
37
38
39  y_train = train_data["Label"]
40  X_train = train_data["Content"]
41  X_test = test_data['Content']
42  X_title = train_data["Title"]
43  Y_title = test_data["Title"]
```

Στο αρχείο που εκτελεί το *“classification”* κάνουμε ξανά *import* τις βιβλιοθήκες που χρειαζόμαστε. Εδώ θα χειριστούμε διαφορετικά τα δεδομένα *train* και *test*. Κάνουμε την ίδια επεξεργασία στα δεδομένα των λέξεων όπως και πριν.

```
# Add labels #
le_train = preprocessing.LabelEncoder()
X_train_le = le_train.fit_transform(y_train)
X_train_cat = le_train.inverse_transform(X_train_le)


# Create matrix of TF-IDF features #
# Use title efficiently            #
tfidf_vectorizer = TfidfVectorizer(stop_words=ENGLISH_STOP_WORDS)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train + X_title)
X_test_tfidf = tfidf_vectorizer.transform(X_test + Y_title)

# Normalize data #
norm = Normalizer()
X_train_tfidf = norm.fit_transform(X_train_tfidf)
X_test_tfidf = norm.transform(X_test_tfidf)

# Classifier #
clf = SVC(C=1, kernel="rbf", gamma=10)

# Use LSA for dimensionality reduction #
svd = TruncatedSVD(n_components=100, random_state=123)


# Perform dimensionality reduction #
X_train_reduced = svd.fit_transform(X_train_tfidf)
X_test_tfidf = svd.transform(X_test_tfidf)
```

Η διαφορά είναι ότι εδώ θα χρειαστεί να εισάγουμε τα *test* δεδομένα στον *Vectorizer*. Επίσης, στη συνέχεια κάνουμε κανονικοποίηση των δεδομένων με τη χρήση ενός *"normalizer"*. Ουσιαστικά μετατρέπονται τα αριθμητικά δεδομένα μας που είχαμε πρίν σε ακολουθίες από 0 και 1. Ο *"classifier"* που χρησιμοποιήθηκε είναι λίγο διαφορετικός (*Radial basis function kernel*) και επίσης εκτελούμε dimensionality reduction. Υπολογίζονται έπειτα οι μετρικές της *"My method"* και προστίθενται στο αρχείο *CSV*.

```
95   # Create csv #
96   Evaluation_metric_df.to_csv("EvaluationMetric_5fold.csv", encoding='utf-8', index
97
98   # Predict test set #
99
00   # Predict test set #
01
02   # Train model #
03   clf.fit(X_train_reduced, X_train_le)
04
05   # Predict categories #
06   y_test = clf.predict(X_test_tfidf)
07   y_cat = le_train.inverse_transform(y_test)
08
09   # Create csv of predicted categories #
10   cols = ['Id', 'Predicted']
11   lst = []
12
13   # Lst: list of lists #
14   # Every single list  #
15   # contains id and    #
16   # predicted category #
17
18   for i in range(test_data.shape[0]):
19       curr_id = test_data.iloc[i]['Id']
20       lst.append([curr_id, y_cat[i]])
21
22   # Create a dataframe and convert it into csv #
23   pf = pd.DataFrame(lst, columns=cols)
24   pf.to_csv("testSet_categories.csv", encoding="utf-8", index=False)
25
26   #end = time.time()
27   #print(end - start)
```

Η συνάρτηση *"fit"* υπολογίζει την καλύτερη δυνατή λύση για τα δεδομένα που έχουμε ως παραμέτρους κάνοντας το *train*. Αντίστοιχα η συνάρτηση *"predict"*

υπολογίζει το νέο *label* που θα δοθεί στα δεδομένα που μόλις διαβάσαμε από το *test set*.

Πάνω στη μεταβλητή *y_test* αποθηκεύεται το αποτέλεσμα της πρόβλεψης που λαμβάνουμε χρησιμοποιώντας τη συνάρτηση *"predict"*. Στη μεταβλητή *y_cat* έχουμε πλέον το αποτέλεσμα του *classification*, καθώς κωδικοποιείται το *y_test* πίσω στην αρχική τιμή.

Στο τέλος του αρχείου παράγεται και εξάγεται το αρχείο που περιέχει τις προβλέψεις κατηγορίας για κάθε άρθρο *"testSet_categories.csv"*.

## Requirement 2 : Question 2b :

Σε αυτό το ερώτημα ο σκοπός μας είναι να δημιουργήσουμε ένα μοντέλο εάν δύο ερωτήσεις σε ένα site όπως είναι το **Quora** είναι πανομοιότυπες. Παίρνουμε ένα αρχείο *train_set.csv* (περιέχει 283014 ζεύγη) και μας βοηθάει στο να εκπαιδεύσουμε τον αλγόριθμό μας. Επίσης παίρνουμε ένα αρχείο *test_set.csv* (περιέχει 47912 αντικείμενα) και μας βοηθάει να κάνουμε τις προβλέψεις για τα δεδομένα μας.

Θα αναπτύξουμε ένα σύστημα Μηχανικής Μάθησης για να ταξινομήσουμε πότε τα ζευγάρια ερωτήσεων είναι όμοια ή όχι. Ξεκινώντας θα εφαρμόσουμε το μοντέλου του **Bag of Words** ή **TF-IDF** με **Xgboost**. Όταν μιλάμε για ανάκτηση πληροφορίας οι πιο κοινές μέθοδοι είναι οι δύο προηγούμενες. Όσο αναφορά το **Xgboost** είναι μια βιβλιοθήκη που παρέχει ένα gradient boosting framework.

Στην αρχή εισάγαμε τις βιβλιοθήκες που χρειαζόμαστε. Στη συνέχεια προχωράμε στην διαδικασία καθαρισμού του κειμένου. Σε αυτό το κείμενο η διαδικασία καθαρισμού θα είναι διαφορετική από άλλα. Αυτό συμβαίνει για τον λόγο ότι δεν θέλουμε να αφαιρέσουμε τα *StopWords* γιατί λέξεις όπως το what, which είναι σημαντικές για τις ερωτήσεις, όπως και τις λέξεις οι οποίες είναι ρίζες άλλων λέξεων. Αυτό που θέλουμε όμως να αφαιρέσουμε είναι τα σημεία στίξης, τα κόμματα μεταξύ των αριθμών. Θα αλλάξουμε επίσης κάποιους ειδικούς χαρακτήρες (όπως είναι το $) σε γράμματα (όπως dollar). Και τέλος θα αλλάξουμε τις συντομογραφίες με τους αρχικούς του όρους. Χρησιμοποιήσαμε την συνάρτηση *"concat''* από την βιβλιοθήκη **panda** και να ενώσουμε τις δύο ερωτήσεις σε μία (γιατί είναι πιο εύκολο να ταξινομήσουμε ένα κείμενο).

Στη συνέχεια, εφαρμόζουμε το **Bag of Words** μέσω του *"CountVectorizer"*. Στο **Xgboost** χρησιμοποιούμε κάποιες συγκεκριμένες παραμέτρους με βάση τις επιδόσεις από τα δεδομένα επικύρωσης (validation data). Το **TF-IDF** είναι προϊόν δύο στατιστικών στοιχείων (συχνότητα όρων και αντίστροφη συχνότητα εγγράφων).

Προκειμένου να υπολογίσουμε την αναπαράσταση του **TF-IDF** πρέπει να υπολογίσουμε τις συχνότητες των αντίστροφων εγγράφων με βάση τα δεδομένα που χρησιμοποιήσαμε για να εκπαιδεύσουμε τα δεδομένα μας.

Τέλος, παραθέτουμε τον πίνακα με τις μετρήσεις και για τις δύο μεθόδους που χρησιμοποιήσαμε :

| Method | Precision | Recall | F-Measure | Accuracy |
|---|---|---|---|---|
| Method-1 | 0.76 | 0.76 | 0.75 | 0.76 |
| Method-2 | 0.78 | 0.78 | 0.77 | 0.77 |

Από τον παραπάνω πίνακα βλέπουμε ότι η μέθοδος **TF-IDF** είναι αποδοτικότερη.

## Requirement 3 :

Σε αυτό το ερώτημα ο σκοπός είναι να δημιουργήσουμε ένα μοντέλο που θα μπορεί να κρίνει πότε ένας κριτής της ταινίας την βαθμολόγησε θετικά ή αρνητικά. Το σύνολο των δεδομένων μου περιέχει CSV αρχεία. Επίσης περιέχει δύο αρχεία το train_set.csv (το οποίο περιέχει 25000 αντικείμενα) και το test_set.csv (το οποίο περιέχει 25000 αντικείμενα).

Στην αρχή, δημιουργήσαμε ένα κλασικό μοντέλο κλασικής μάθησης χρησιμοποιώντας το **Bag of Words**. Χρησιμοποιήσαμε την βιβλιοθήκη pandas στην οποία φορτώσαμε τα δεδομένα του CSV αρχείου. Ξεκινώντας, εφαρμόσαμε την διαδικασία *''Clean the Data''* και αφαιρέσαμε τους μη-χαρακτήρες (όπως ., #, ! κ.ο.κ.) και τα *StopWords* (στις λίστες που δημιουργήσαμε μετά το *tokenization*). Προχωρώντας, σπάσαμε τις προτάσεις σε λέξεις, λέξεις κλειδιά, φράσεις (μέσω του *tokenization*), έχοντας σαν στόχο να χωρίσουμε τις λέξεις σε κοινές κατηγορίες (μέσω του *Stemming*). Οι κατηγορίες αυτές μπορεί να είναι λέξεις σε διάφορες κλίσεις, ίδια συνθετικά λέξεων και άλλα. Τέλος, αφαιρέσαμε τις λέξεις που εμφανίζονται με την λιγότερη συχνότητα γιατί δεν θα μας επηρέαζαν στα τελικά αποτελέσματα (μέσω της *flat λίστας*). Χρησιμοποιώντας το *"Logistic Regression"* (αλγόριθμος ταξινόμησης Μηχανικής Μάθησης) παίρνουμε τις προβλέψεις μιας εξαρτημένης μεταβλητής (η οποία είναι δυαδική μεταβλητή).

Χρησιμοποιώντας το **Bag of Words** έχουμε σαν στόχο να μετατρέψουμε τις λέξεις σε μορφή αριθμού και πιο συγκεκριμένα σε διανύσματα αριθμών. Στη συνέχεια διαχωρίσαμε το σύνολο των δεδομένων σε δύο σύνολα δεδομένων train και test

(μέσω της συνάρτησης *train_test_split()*). Πηγαίνοντας τώρα στην ερμηνεία των αποτελεσμάτων έχουμε :

- **Precision** : αναφέρεται στον συνολικό αριθμό των προβλέψεων συναισθημάτων που είναι σωστά σε σχέση με το συνολικό προβλεπόμενο συναίσθημα.
- **Recall** : αναφέρεται στον συνολικό αριθμό των προβλέψεων συναισθημάτων που είναι σωστά σε σχέση με το συνολικό σωστό προβλεπόμενο συναίσθημα.
- **F-Measure** : μπορεί να ερμηνευτεί ως ένας μέσος αριθμός του precision και του recall όπου φτάνει την καλύτερη τιμή του στο 1 και την χειρότερη στο 0.
- **Accuracy** : ορίζει τον αριθμό των προβλέψεων που έχουν γίνει σωστές σε σχέση με τον συνολικό αριθμό των προβλέψεων.

Στη συνέχεια του ερωτήματος δημιουργήσαμε ένα μοντέλο από **Deep Learning** χρησιμοποιώντας την βιβλιοθήκη **Keras**. Χρησιμοποιούμε όπως και προηγουμένως την βιβλιοθήκη pandas και σε αυτή δίνουμε την εντολή *read_csv()* η οποία περνάει το CSV αρχείο που περιέχει το σύνολο των δεδομένων μας. Στο κείμενο το οποίο πρόκειται να επεξεργαστούμε περιέχει παρενθέσεις, HTML tags και άλλα. Έπειτα πάμε να δούμε την διανομή των συναισθημάτων σε αρνητικά και θετικά στο σύνολο των δεδομένων μας. Για να γίνει αυτό πρέπει πρώτα να αφαιρέσουμε την περιττή πληροφορία που περιέχουν τα κείμενα με τα οποία θα ασχοληθούμε. Με την συνάρτηση *remove_tags()* αφαιρούμε τα HTML tags, βάζοντας στην θέση αυτών που αφαιρέθηκαν το κενό. Στη συνέχεια, αφαιρούμε τα μεμονωμένα γράμματα που έμειναν από την αφαίρεση των tags και βάζουμε στην θέση τους το κενό. Έπειτα για να ολοκληρωθεί η αφαίρεση βγάζουμε τα διπλά κενά που έχουν δημιουργηθεί. Έτσι μένουμε μόνο με τα γράμματα. Χωρίζοντας τα δεδομένα μας σε train και test (χρησιμοποιώντας την συνάρτηση *train_test_split*). Το train θα χρησιμοποιηθεί για να εκπαιδεύσει τα μοντέλα μας ενώ το test για να αξιολογήσει πόσο καλά τα μοντέλα μας εκτελούνται. Έπειτα χρησιμοποιούμε το *tokenizer* για να δημιουργήσουμε ένα λεξικό. Μετατρέποντας το *X_train* σε λίστες, παρατηρούμε ότι αυτές οι λίστες περιέχουν ακεραίους. Κάθε λίστα απευθύνεται σε κάθε πρόταση του *training set*. Το μέγεθος κάθε λίστας είναι διαφορετικό (επειδή κάθε πρόταση έχει διαφορετικό μέγεθος). Θέτουμε μέγιστο όγκο που έχει μια λίστα το 100 (*maxlen = 100*).

Το μοντέλο **Deep Learning** που θα αναπτύξουμε ονομάζεται νευρωνικό δίκτυο (*model = Sequential()*). Για να μεταγλωττίσουμε το μοντέλο μας θα χρησιμοποιήσουμε για *optimizer=''adams''* και το *binary_crossentropy* ως συνάρτηση απώλειας. Έπειτα ορίζουμε τις μετρήσεις. Στη συνέχεια, χρησιμοποιούμε την μέθοδο *''fit''* για να εκπαιδεύσουμε το νευρωνικό μας δίκτυο.

Το εκπαιδεύουμε μόνο στο *train set*. Για να αξιολογήσουμε την απόδοση του μοντέλου μας μπορούμε απλά να περάσουμε το test set στην μέθοδο αξιολόγησης του μοντέλου μας (*score = model.evaluate(X_test, y_test, verbose=1)*). Στο τέλος, παίρνουμε ένα τυχαίο άρθρο και θα προσπαθήσουμε να προβλέψουμε τα συναισθήματα. Για να το κάνουμε αυτό πρέπει να μετατρέψουμε την κριτική σε αριθμητική μορφή. Για αυτό χρησιμοποιούμε το *''tokanizer''* (όπως και προηγουμένως). Η μέθοδος *text_to_sequences* μετατρέπει τις προτάσεις σε αριθμητική μορφή. Όπως και προηγουμένως θέτουμε 1 αν είναι θετικό και 0 αν είναι αρνητικό. Στην περίπτωση που είναι >=0,5 το θεωρούμε θετικό ενώ στην άλλη περίπτωση το θεωρούμε αρνητικό.

Στο τέλος παραθέτουμε τον πίνακα με τις μετρήσεις και των δύο μεθόδων :

| Method | Precision | Recall | F-Measure | Accuracy |
|---|---|---|---|---|
| Classic Machine Learning | 0.84 | 0.84 | 0.84 | 0.83625 |
| Deep Learning | 0.608835 | 0.788915 | 0.680893 | 0.64625 |