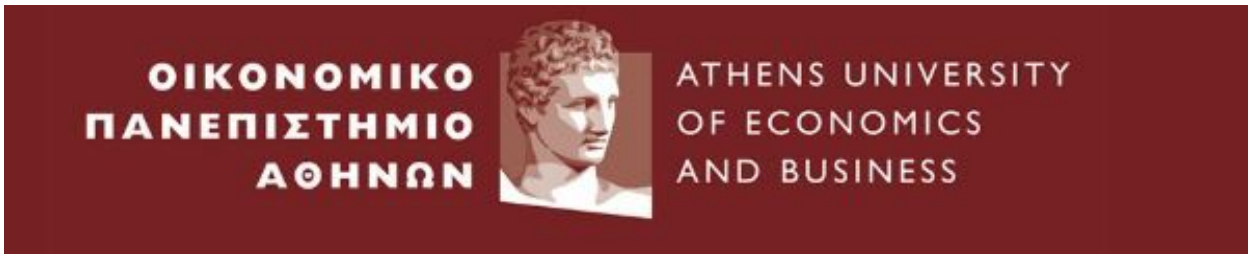

AUEB M.S.c in Data Science (part-time)

Text Analytics: Homework 2: Report

Team members:

Kaplanis Alexandros (P3351802)

Politis Spiros (P3351814)

Proimakis Manos (P3351815)

1. SETUP & EXECUTION

The assignment is implemented in Jupyter Notebook format. In order to setup your environment and execute the code, the following steps should be performed:

1. Setup a Conda virtual environment:

```
conda create -n msc-ds-ta-homework-2 python=3.6
```

2. Activate the virtual environment:

```
conda activate msc-ds-ta-homework-2
```

3. Install required Python packages (from within the assignment folder):

```
pip install -r requirements.txt
```

4. Execute Jupyter Notebook (from within the assignment folder):

```
jupyter notebook
```

2. INTRODUCTION

For the second assignment we decided to perform a sentiment analysis on tweets. The dataset was downloaded from Kaggle for the Sentiment140 project and the bundle contained a training dataset with 1.6 million Tweets and a test dataset.

The training dataset has 2 classes; 0 for negative and 4 for positive. While the test dataset has an additional class for neutral denoted by 2.

DATA CLEANSING & PREPROCESSING

A lot of emphasis was given on the data preprocessing phase during which we created the **Preprocess** class with which we removed all usernames (words beginning with @) and all the hyperlinks from the dataset.

Furthermore, we added some more functions to remove usernames that wouldn't be removed because of an in-between underscore. (i.e. @DJ_RIN) and all special characters.

Concluding the preprocess phase, we dropped all duplicate instances per user and kept only the useful columns (polarity, text). An example of duplicate tweets can be seen below:

user	text	polarity
lost_dog	i am lost please help me find a good home	549

TRAIN/DEVTEST SPLIT

As a next step we split the Training dataset to train and test using sklearn's *train_test_split* with a ratio ¾.

VECTORIZATION

This allowed us to keep the other test dataset that came with the bundle as unseen, in order to able to use it in the final stage of the assignment.

We proceeded with examining a couple of approaches on vectorization:

a) Perform feature extraction using the TF-IDF vectorizer which was also used for feature selection using its *max_features* parameter.

   This first dimensions reduction was mostly done to reduce the running times.

b) Use a Twitter-corpus, pre-trained Word2Vec model ("glove-twitter-200"). We acquired the model using the built-in downloading functionality of the GenSim package, however it can also be downloaded from the official Stanford repo: https://nlp.stanford.edu/projects/glove/

   This model was selected based on the notion that, having been trained on tweets, it would perform good on our dataset. We opted for the 200-dim vectors model, assuming that it would provide more information, hence lead to better classifier performance.

To our surprise, the Word2Vec model did not provide significant gains in classification tasks, being only to provide a marginal - in the order of decimal places - improvement in classification accuracy. We hypothesize that, a better performance could be achieved by adding weights to the word embeddings of the

most popular words or by using the drop-out method, however, we did not proceed with experimentation.

3. CLASSIFIERS

Model Reusability

Due to the long execution times of the classifiers we have created a function that saves the trained models and also loads them when they need to be used.

DUMMY

As a first step we use SKLearn's dummy classifier to use as a base line. We know beforehand that the probability of one class occurring is 50%.

Consequently, we expected that predicted classes using the dummy classifier will be uniformly distributed.

NAIVE BAYES

The Naïve Bayes classifier assumes independency between features given each class and more specifically the multinomialNB which is used explicitly for text classification.

As per the documentation, multinomialNB doesn't necessarily require integers as input and can work with TF-IDF.

Given the below F1-scores and the confusion matrix we can see that Naïve Bayes scores surprisingly high despite its "naïve" assumptions.

```
train f1-score: 0.7828025
test f1-score: 0.77238
```

Predicted	0	1
True		
0	154910	44985
1	46063	154042

LOGISTIC REGRESSION

The Logistic Regression classifier scores slightly higher than the NB classifier as seen below.

```
train f1-score: 0.8082604943371685
test f1-score: 0.7934975471008179
```

Predicted	0	1
True		
0	154189	45706
1	38439	161666

Compared to NB we observe a significant increase in the True-Negative values, a slight increase in the False-Negative and a decrease on the False-Positive values. However based on its F1-Score it is a better classifier.

The higher score is a strong indicator that there is no multicollinearity between the features.

SUPPORT VECTOR MACHINES

Note: Given that SVM would never execute, we used max_features = 5000 in the TF-IDF vectorizer ***just for this classifier****.*

This classifier was trained in two different ways:

We first used the base X_train dataset we also used for all the others and then we applied (truncated) Singular Value Decomposition (SVD) on the training data.

SVM (with base X_train) scores higher on the training data than the other two classifiers but less on the test data.

```
train f1-score: 0.837138508371385
test f1-score: 0.7179689037591024
```

Predicted	0	1
True		
0	1743	727
1	706	1824

After the dimensionality reduction we see a further reduction in both the train and test F1 scores.

```
train f1-score: 0.6702016325961069
test f1-score: 0.6484228117819093
```

Predicted	0	1
True		
0	1765	705
1	978	1552

The cause for such a reduction could be that we reduced the dimensions too much making the data too crowded, resulting to the SVM not being able to classify the data with the same success.