Hacettepe University

Computer Science and Engineering Department

Name and Surname : Mehmet Alperen Özçelik

Identity Number : 21992995

Course : AIN442

Experiment : Assignment 4

Date Due : 26.11.2021

Advisor : Hayriye Çelikbilek

e-mail : b21992995@cs.hacettepe.edu.tr

1-INTRODUCTION

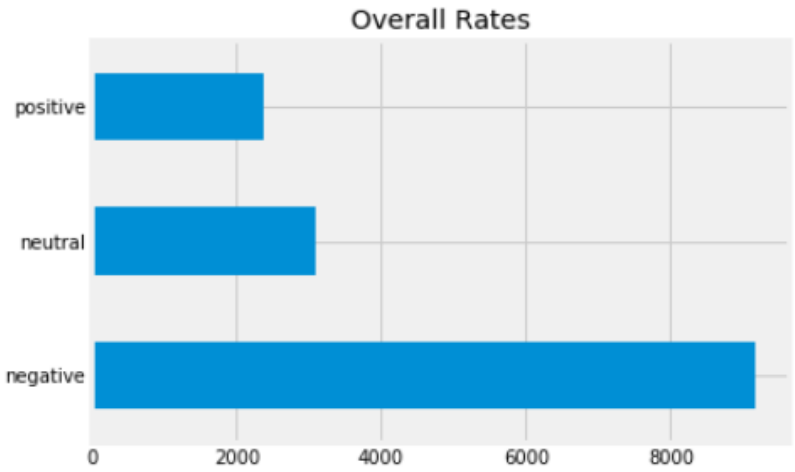
This assignment is a sentiment analysis project about the problems of each major U.S. airline. Twitter data was scraped from February of 2015. Our main goal in our project is to make our dataset meaningful and apply Natural Language Processing and TF-IDF to the given sentences, to understand whether users are making positive, negative or neutral comments about the airlines, and to make it ready for test sets for the future.

2-DATA

Our material is a file named Tweets.csv containing the dataset we used throughout the project. The attributes of the dataset are detailed below:

- tweet_id
- airline_sentiment
- airline_sentiment_confidence
- negativereason
- negativereason_confidence
- airline
- airline_sentiment_gold
- name
- negativereason_gold
- retweet_count
- text
- tweet_coord
- tweet_created
- tweet_location
- user_timezone

Users' tweets for the airlines:



The image after converting the csv file into dataframe:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino

3–METHOD

To summarize, I first tokenized the text data we have and applied the classical NLP preprocess steps, which I will explain in more detail below. At first I tried to vectorize and produce predictions using TF-IDF, but whichever classifier I used, it predicted nearly all of them as "negative" and I had to come up with a solution. As a solution, I used the SMOTE method of the imblearn library. This method is create new object of same type as caller containing n items randomly sampled from the caller object. After using the SMOTE method texts with equal number of each sentiment. After vectorizing my new data and applying TF-IDF again I separated it into train-test sets. There were 4 types of classifiers that I knew before. I tried all of them to see which one would get the best results. These classifiers were: Naive Bayes, Random Forest, KNN and SVC. I chose the Random Forest Classifier because I got the best accuracy score in this classifier.

4-DEVELOPMENT

4.1 Plan:

The general plan of my project was to complete the project by applying basic nlp preprocesses to my data, after tokenizing it, applying TF-IDF and finding the appropriate classifier.

It is not a project that requires very high hardware, but since it took too long to run the spelling correction function, I decided not to use that method. Other than that, evaluate all classifiers accuracy score and throw the stopwords I can say that it took a long time to on my own computer.

4.2 Analysis:

When we looked at the features of data, there were 2 columns that I could use as attributes, the text column and negativereason. As output, we had to get the airline_sentiment part, the remaining attributes were not suitable for use in the project so I only used the text column because I don't know exactly how to proceed if I also get the negativereason column.

4.3 Design:

There are several methods we can use for preprocessing after we tokenize the sentences. As far as I know, these methods are:

- Delete Airline Name
- Remove non-ASCII characters
- Convert all characters to lowercase
- Remove punctuation
- Convert all numbers to textual version
- Remove stop words
- Convert tokens to sentence

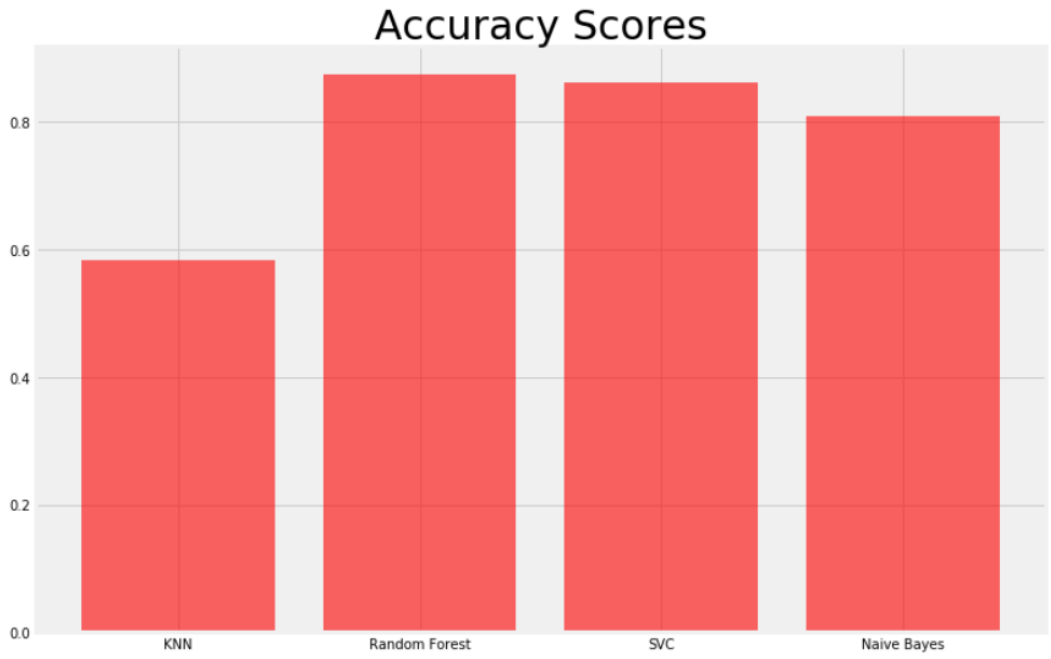
Replace contractions didn't seem to affect much for this dataset, while autocorrect was taking too long, so I didn't use replace contractions and autocorrect from the above methods. I didn't use Stemming and Lemmatization, because I got a higher accuracy score when I wasn't applied these processes. Apart from these methods, I have classifier methods that I will explain in detail below, and I have methods that I use in the word cloud parts.

4.4 Implementation:

When I vectorized and TF-IDF like this number of samples, it predicted nearly all of them as "negative" and I had to come up with a solution. As a solution, I used the SMOTE method of the imlearn library.

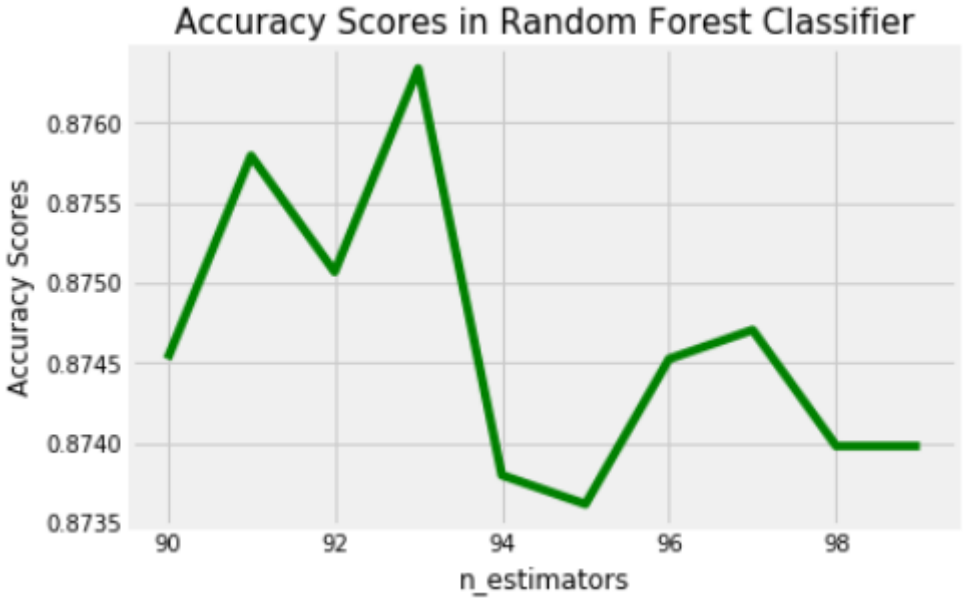
Then i vectorized my data and do the TF-IDF process with sklearn library. After that I divided the data into train and test set.

Then I tried the 4 classifiers I've used so far to find the most suitable classifier. These classifiers were: Naive Bayes, Random Forest, KNN and SVC. I printed the resulting accuracy scores graphically:

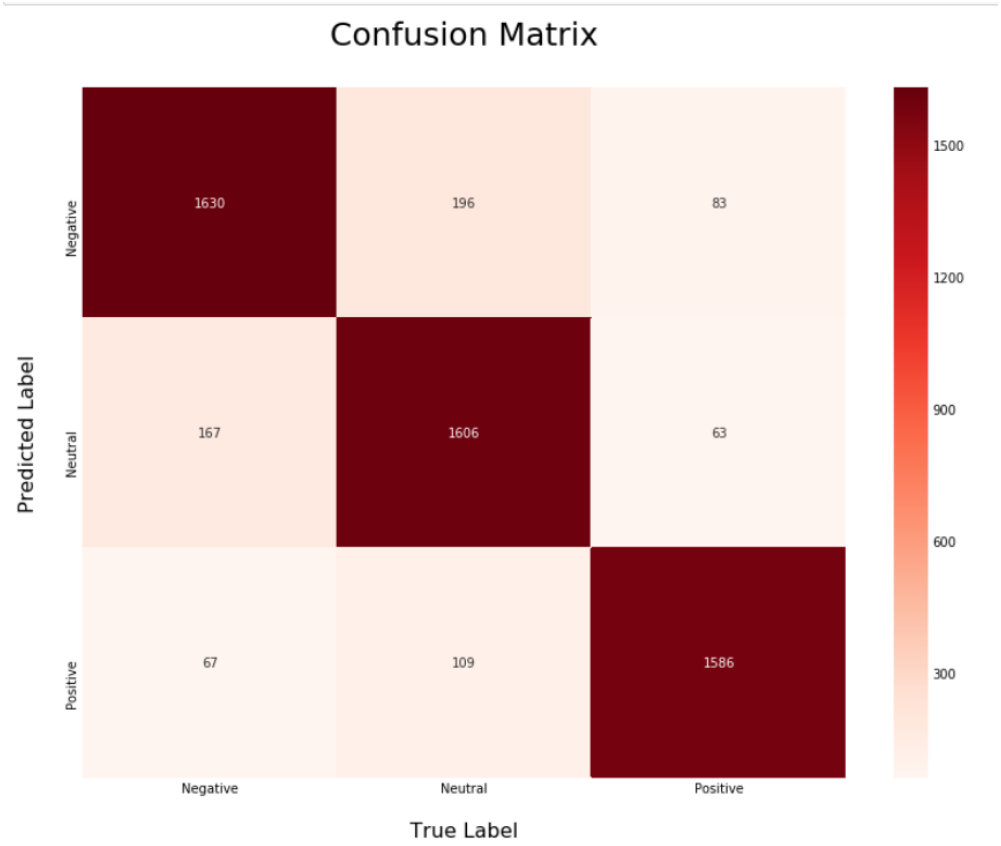


As you can see, Random Forest Classifier gave higher accuracy score, so I decided to use Random Forest Classifier.

The one of the most important parameter of Random Forest classifier is the number of n_estimators. Now I checked which number has the highest accuracy score.



As a result, I found that the most accurate classifier for this dataset is Random Forest and the most accurate number of n_estimators is 93. Finally, I printed my confusion matrix with the heatmap graph the image below.



4.5 Programmer Catalog:

The tutorial shared with us in our AIN442 course and my third assignment in the design part was very instructive and I can say that I spent about 3 hours because I implemented my codes with a very similar design.
I mentioned above that I had difficulties in implementing the TF-IDF part, and finding a different solution from third assignment, I worked around 1 hours here.
The truth is, I didn't have a test set to test my model. I separated 20 percent of it as test data with the classical train-test split method, but it would be more accurate to call this data validation data. it took some time to find the best classifier here and to find the best parameters to give while trying the classifiers, it took about 2-3 hours.
Finally, it took about 3-4 hours to report them with pictures.

My suggestion to other programmers is that Random Forest Classifier gave much better results than other classifiers in my dataset. I think it should be tried in NLP projects. Tokenizing and clearing data is also of great importance for such language processing projects.

4.6 User Catalog:

Any user doing NLP project can use most of my code. Since the source code part I gave above is very simple and clear, I don't think there will be anything confusing about its use.

- ✓ User can use my tokenize and preprocess methods applied to all words as is.
- ✓ Vectorization and TF-IDF part is already a direct library function, so it can be imported directly.
- ✓ The user can test the 4 classifiers I use to find the highest accuracy and select the most suitable classifier.
- ✓ Finally, they can directly use the word cloud processing part by using the tokenized version of their own words and print their own word clouds.

5 - RESULTS, DİSCUSSION AND CONCLUSION

After vectorizing this data set, when I did not apply TF-IDF operation, I got good results without oversampling. I don't think it is very suitable for TF-IDF process without using oversampling because the number of negative tweets samples is too high in the actual data set.

I tried to do cross validation, but because my computer was struggling, I interrupted the process.

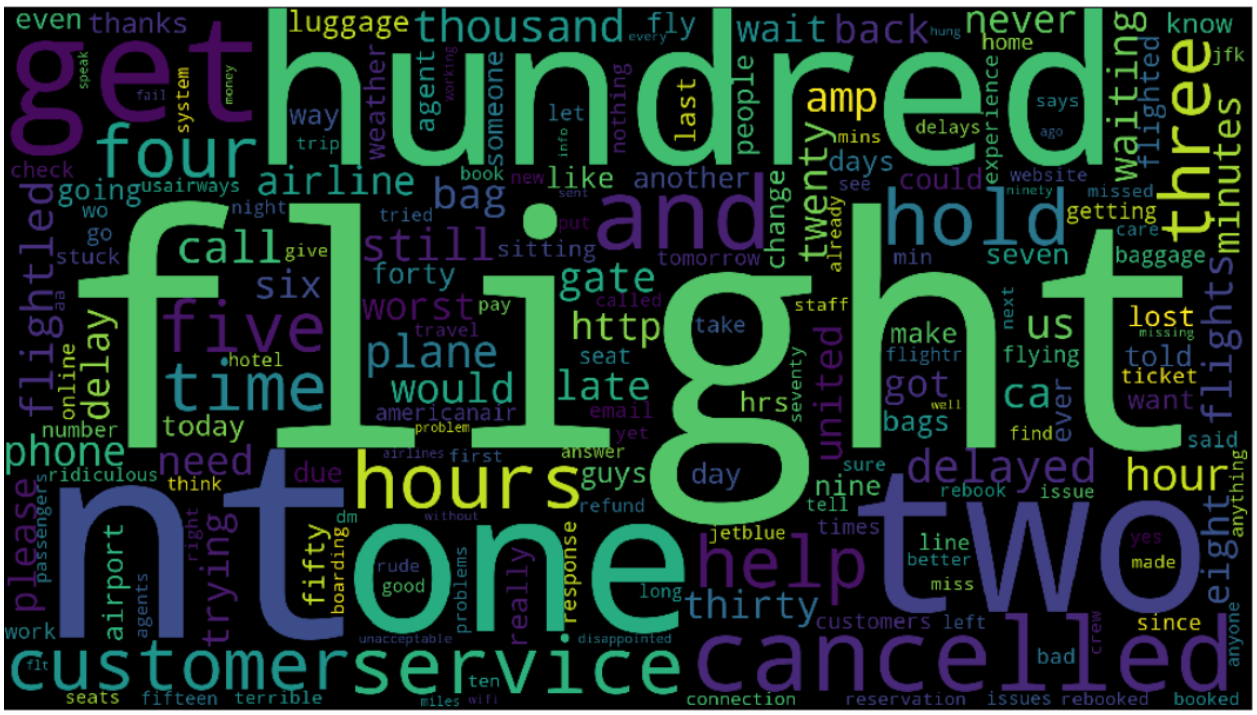
Then I found the most suitable classification method and tried to tune the parameters to get the highest accuracy score.

It was **0.8763** in Random Forest Classifier with n_estimators = 93.

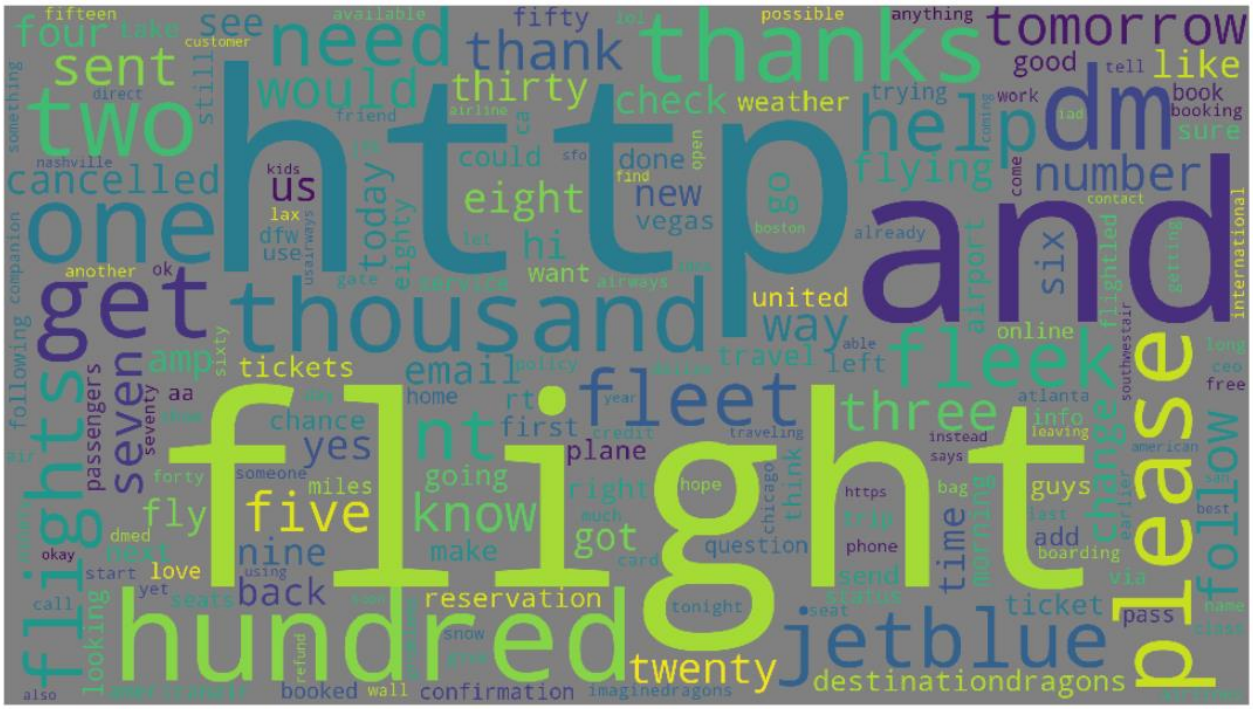
After the TF-IDF process I printed the most effective meaningful words for positive, negative and neutral tweets according to TF-IDF array.

Wordcloud of Positive Tweets:

Wordcloud of Negative Tweets:



Wordcloud of Neutral Tweets:



REFERENCES:

- https://www.kaggle.com/crowdflower/twitter-airline-sentiment
- https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
- https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html
- https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/
- AIN 442 Lecture notes