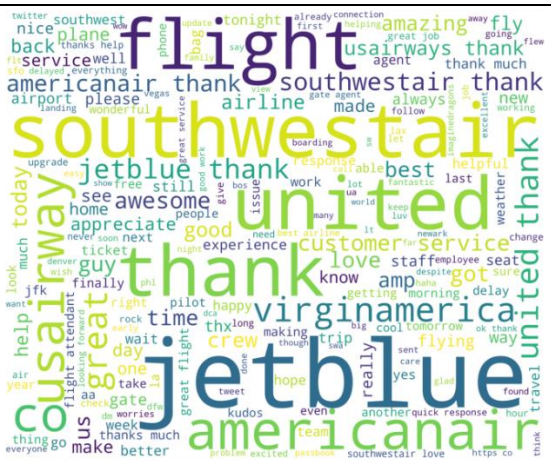
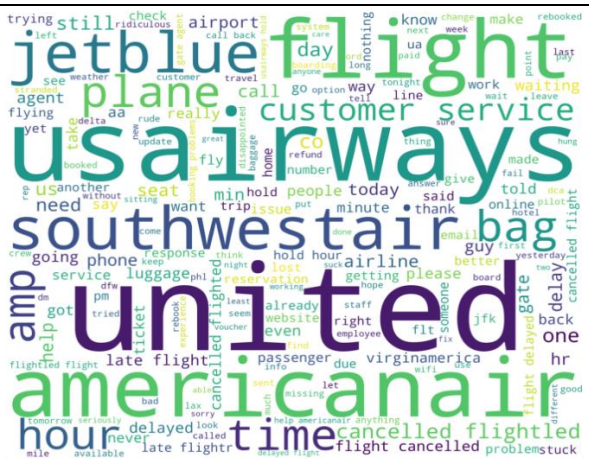
Report

The first step was to analyze the dataset provided. Having a look on a '**text'** column it was evident that it needs to be cleaned from numerous signs such as '@ : ~â€' etc. as they affect the overall sentiment value of a text. Analyzing the '**sentiment**' column I found out that data is imbalanced having more negative sentiments than positive ones.

Number of Negative sentiments	Number of Positive sentiments
9178	2363

Then I decided to find the most frequent words in positive and negative tweets. For this a wordcloud was used.

Positive	Negative
	

Next, I had to split the data with a standard ratio of 80% train and 20% test data. Then the data was sent to a Count Vectorizer to transform the tweet data into the vectors of frequency of occurrence of each word in a tweet. For classification I decided to test out 2 Classifiers: Decision Tree and Random Forest. For model evaluation I used a classification report of scikit-learn library which encompasses of F1-score, Accuracy, Precision and Recall.

Random Forest

Accuracy:0.8999566912083153

	precision	recall	f1-score	support
negative	0.96	0.92	0.94	1947
positive	0.65	0.80	0.71	362
accuracy			0.90	2309
macro avg	0.80	0.86	0.83	2309
weighted avg	0.91	0.90	0.90	2309

Decision Tree

Accuracy:0.8562148116067562

	precision	recall	f1-score	support
negative	0.92	0.91	0.91	1878
positive	0.61	0.63	0.62	431
accuracy			0.86	2309
macro avg	0.76	0.77	0.77	2309
weighted avg	0.86	0.86	0.86	2309

From the table above we can see that a Random Forest Classifier performs better than a Decision Tree in accuracy and precision scores according to the classification report.