# Machine Learning Engineer Nanodegree

## Capstone Project

Swaraj Dash

October 7th, 2019

## I. Definition

## Project Overview

In the past few years, there has been a huge growth in the use of microblogging platforms, such as Twitter. Spurred by this growth, companies are seeking ways to mine these platforms for information about what people think and feel about their products and services. Sentiment analysis, an application of Natural Language Processing, is the contextual mining of text which identifies and extracts subjective information in the source material, thus helping a business understand the underlying sentiment of their users - whether it is positive, negative or neutral.

Having seen my college senior and mentor work on documents classification on the Reuters dataset, it has always been a personal motivation to work on an NLP application. Understanding how computers process and analyze human language fascinates me. Hence, this capstone project will be about performing sentiment analysis on tweets for each of the six major U.S. airlines.

Research:
This Is How Twitter Sees The World : Sentiment Analysis Part One
Twitter Sentiment Analysis: The Good the Bad and the OMG!
Sentiment Analysis: Concept, Analysis and Applications
Sentiment Analysis of Twitter Data: A Paper from Columbia University

## Problem Statement

The problem is to perform sentiment analysis using Machine Learning techniques. We will attempt to classify the sentiment behind a tweet as one of the three classes: positive, neutral or negative, given the tweet (text). This is a multi-class classification challenge.

This is a Supervised Learning exercise where we will apply Machine Learning algorithms to predict the sentiment of a tweet. We will demonstrate the performance of our models by statistical metrics like test accuracy scores.

## Metrics

The evaluation metric to be used for this problem will be the Accuracy Score. We will try to beat the benchmark model with our model. It is nothing but the ratio of total number of correct predicted outcomes to the total number of outcomes.

Let us try to understand:
Given are two lists, y_pred and y_true, that contain the predicted and actual outcomes respectively. For every position index i, compare the i-th element of y_pred with the i-th element of y_true and perform the following calculation:

1. Count the number of matches
2. Divide it by the number of samples

Consider this example:

```
y_pred = [0, 2, 1, 3, 0], y_true = [0, 1, 2, 3, 0]
```

We see matches on indices 0, 3 and 4. Thus:

```
number of matches = 3 number of samples = 5
```

Finally, the accuracy calculation:

```
accuracy = matches/samples accuracy = 3/5 accuracy = 0.6
```

# II. Analysis

## Data Exploration

For this project, I'll be using the Twitter US Airline Dataset from Kaggle (https://www.kaggle.com/crowdflower/twitter-airline-sentiment/).

Input Dataset: Tweets.csv (14,640 labeled tweets)

- Dataset size for training: 80% of data (11712)
- Dataset size for testing: 20% of data (2928)

The fields of the dataset are as follows:

- **tweet_id**: unique identifier for each tweet in the dataset
- **airline_sentiment**: contains one of ['positive', 'neutral', 'negative'].
- **airline_sentiment_confidence**: a floating point number in the range from 0.0 to 1.0, that specified the confidence of the *airline_sentiment* attribute.
- **negativereason**: specifies the reason for *negative* sentiment. It has 11 unique values, eg: 'Bad Flight', 'Late Flight', 'Lost Luggage', 'Cancelled Flight', etc.
- **negativereason_confidence**: a floating point number in the range from 0.0 to 1.0, that specified the confidence of the *negativereason* attribute.
- **airline**: contains the name of the airline service; one of ['Virgin America', 'United', 'Southwest', 'Delta', 'US Airways', 'American']
- **name**: contains username of the Twitter user
- **retweet_count**: an integer value containing the number of retweets for that particular tweet
- **text**: contains the original tweet itself
- **tweet_created**: contains the timestamp of when the tweet was made

This is how a sample of the dataset looks like:

| | tweet_id | name | text | airline | airline_sentiment | airline_sentiment_confidence |
|---|---|---|---|---|---|---|
| 0 | 570306133677760513 | cairdin | @VirginAmerica What @dhepburn said. | Virgin America | neutral | 1.0000 |
| 1 | 570301130888122368 | jnardino | @VirginAmerica plus you've added commercials t... | Virgin America | positive | 0.3486 |
| 2 | 570301083672813571 | yvonnalynn | @VirginAmerica I didn't today... Must mean I n... | Virgin America | neutral | 0.6837 |
| 3 | 570301031407624196 | jnardino | @VirginAmerica it's really aggressive to blast... | Virgin America | negative | 1.0000 |
| 4 | 570300817074462722 | jnardino | @VirginAmerica and it's a really big bad thing... | Virgin America | negative | 1.0000 |

Fig. 1 Dataset sample

# Exploratory Visualization

First we will try to find the distribution of the three different sentiments of tweets: negative, neutral and positive, in our dataset. From Fig. 2, it is clear that it most tweets carry a negative sentiment towards the airlines, followed by neutral, and then positive tweets.

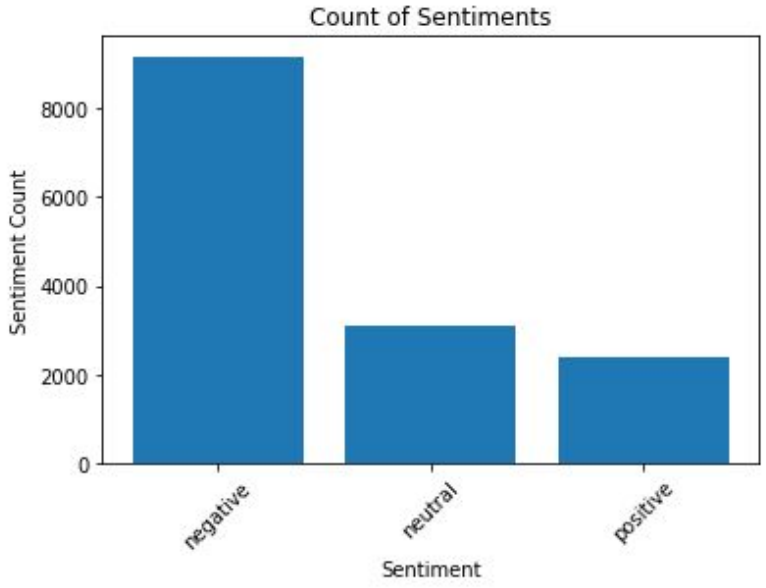



Fig. 2 Count of Sentiments

Next, we will try to glance at the sentiment distribution for the six different US airlines in our dataset. From Fig. 3, we can see that Virgin America has a lot of positive reviews, although the number of tweets made for it is quite small as compared to other airlines. Delta and Southwest seem to have decent number of neutral and positive tweets. While for the remaining three airlines, the number of tweets as well as the negative sentiment count are pretty high.

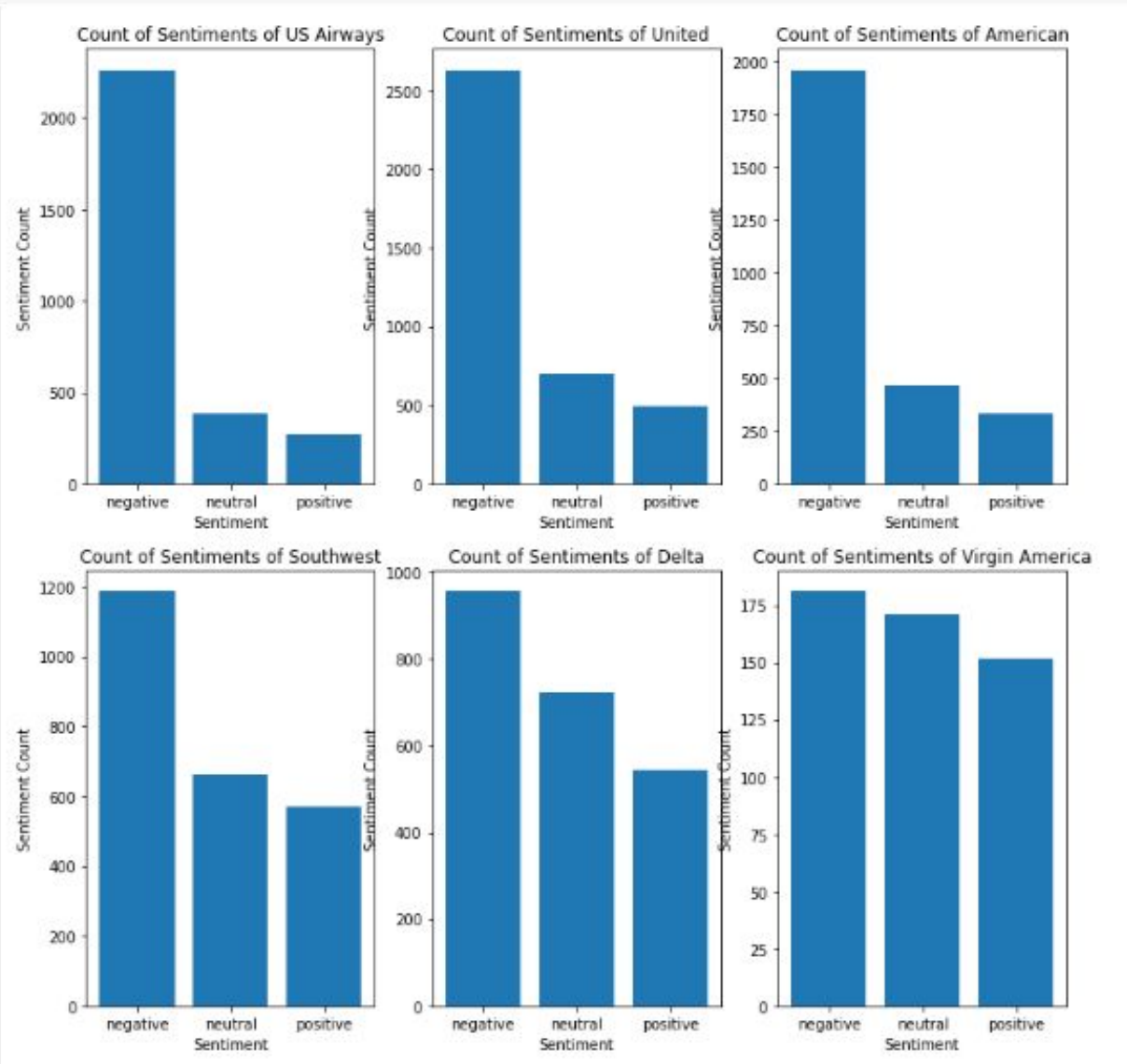



Fig. 3 Sentiment distribution among different airlines

## Algorithms and Techniques

Since we are trying to solve a typical Supervised Learning Classification problem we will use popular classification algorithms such as Decision Trees, Random Forests, SVM which fit the purpose of such problems. Since our dataset has a lot of attributes, tree-type models may tend to perform better by picking out the parameters that have more say in determining a tweet's sentiment.

- **Logistic Regression**:
  Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). This will be used as the benchmark model for this project.

- **Decision Trees**:
  Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

- **Random Forests**:
  The idea behind RF, is to build many small Trees that use a random subset of the features and then combine them into a "Forest" of Trees. Each Tree by itself is a weak predictor, but when combining the many weak predictors, you can often (but not always!) get a stronger model.

- **Support Vector Machines**:
  An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

## Benchmark

During the model optimization exercise, we will be using a Logistic Regression classifier model as the benchmark model. We will try to beat this benchmark model, with proper hyperparameter tuning on our chosen classification model, given they are fitted to the same training and testing set. We will use the models' test accuracy scores as the basis of comparison.

# III. Methodology

## Data Preprocessing

The data preprocessing steps can be summarized as follows:

- First we remove the attributes that have not been defined by the dataset provider. These attributes ['airline_sentiment_gold', 'negativereason_gold', 'tweet_coord'] are dropped from the dataset.

- Then we remove the unneeded attributes from the dataset that we presume not having any correlation with the output sentiment class. These attributes ['tweet_id', 'name', 'retweet_count', 'tweet_created', 'tweet_location', 'user_timezone'] are further dropped as we want to perform our operation on a clean version of the dataset.

- Next, we try to clean our dataset's tweets by refining the contained text. We do this by removing expressions like 'http', '@', 'RT' from our tweets.

- We enhance our tweets furthermore by removing *stopwords* from our tweets. Stop words can be defined as some extremely common words which would appear to be of little value in helping select documents matching a user need, that can be excluded from the vocabulary entirely, when computers process natural language. For example, ['me', 'myself', 'for', 'of', 'which', 'because']. Check out NLTK's list of English stopwords here.

- Since our data has a high imbalance between the three classes of sentiment, we will first perform binary classification on our dataset. That is, we will have two

sentiments: *negative* and *positive* (combining *neutral* and *positive* sentiments). We apply a function that labels *negative* sentiment as 0, otherwise 1.

- We finally split our data into training and testing data into 80-20 ratio. That is, out of the total 14,640 samples, the training set contains 80% of data (11,712 samples) and testing set contains 20% of data (2,928 samples).

## Implementation

- Now, we apply Machine Learning algorithms to build a model that will predict the class of the sentiment of a tweet as positive, neutral, or negative. After appropriate data and text preprocessing, that involved removal of stopwords, we could visualize how the keywords in the tweet help in determining the sentiment of the tweet.

- After the initial preprocessing, we stored our cleaned tweets in *clean_tweet* attribute. Then we separated out it into *train_clean_tweet* and *test_clean_tweet*.

- Next, we perform feature extraction the help of scikit-learn. The fit_transform function calibrates our measurement so that training data and test data have a similar shape. What it does is apply feature extraction objects, CountVectorizer in our case.

- CountVectorizer is one of the most basic ways we can represent text data numerically (that is what our model will learn): one-hot encoding (or count vectorization). The idea is simple. We will be creating vectors that have a dimensionality equal to the size of our vocabulary, and if the text data features that vocab word, we will put a 1 in that dimension. Every time we encounter that word again, we will increase the count, leaving 0s everywhere we did not find the word even once.

- After acquiring our *train_features* and *test_features*, we fit our models that we specified earlier (Logistic Regression, SVM, Decision Trees, Random Forests) to the data and make predictions.

## Refinement

- For finding initial model performances, I used random parameters for the models. Our benchmark Logistic Regression model performs quite poorly giving an accuracy score of 62.1%. Likewise the SVM gives a score of 62.1%.

- However, both the tree type classifiers: Decision Tree and Random Forests perform very well, like we had anticipated. Decision Trees gives an accuracy score of 74.3%, while Random Forests give a score of 81.7%.

- From here, we choose Random Forests to compete with the Logistic Regression model, by performing hyperparameter tuning on both.

- The following is the summary of tuning done on the benchmark Logistic Regression model:

Logistic Regression

| Parameter | Description | Possible values | Best value |
|---|---|---|---|
| C | Inverse of regularization strength | float number | 0.01 (considering lambda = 100) |
| solver | Algorithms to be used for optimization | ‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’ | newton-cg (handles multinomial loss) |
| max_iter | Maximum number of iterations taken for the solvers to converge. | integer | 200 (no significant change above that) |

- This led to an immense improvement on the performance of the Logistic Regression model, in my opinion, particularly due to increasing in the inverse of regularization strength, or, *gamma* value (or decrease in the lambda value) of the initial model. With a large lambda value, the model was clearly underfitting. The figures of the improvement is discussed in the section below.

- The following is the summary of tuning done on the chosen Random Forests model:

Random Forests

| **Parameter** | **Description** | **Possible values** | **Best value** |
|---|---|---|---|
| n_estimators | The number of trees in the forest | integer | 200 (no significant change above that) |
| max_features | The number of features to consider when looking for the best split | int, float, ‘auto’, ‘sqrt’, ‘log2’, none | log2 (handles multinomial loss) |
| criterion | The function to measure the quality of a split | ‘gini’, ‘entropy’ | Entropy (better for exploratory analysis; fits our problem) |

- This tuning exercise led to a slight improvement in the performance of the Random Forests model. I believe it is due to the fact that we specify max_features as ‘log2’, which yields a lower result as compared to ‘sqrt’ or ‘auto’, and the nature of our data is small and simple, for which we do not require to consider a lot of features.

# IV. Results

## Model Evaluation and Validation

- As defined earlier, we have used accuracy scores as the metric to evaluate the performance of our models.
- Our benchmark model does not perform well, scoring around 62.1%.However, both the tree type classifiers: Decision Tree and Random Forests perform very well, like we had anticipated. Decision Trees gives an accuracy score of 74.3%, while Random Forests give a score of 81.7%.
- The initial findings are summarized below:

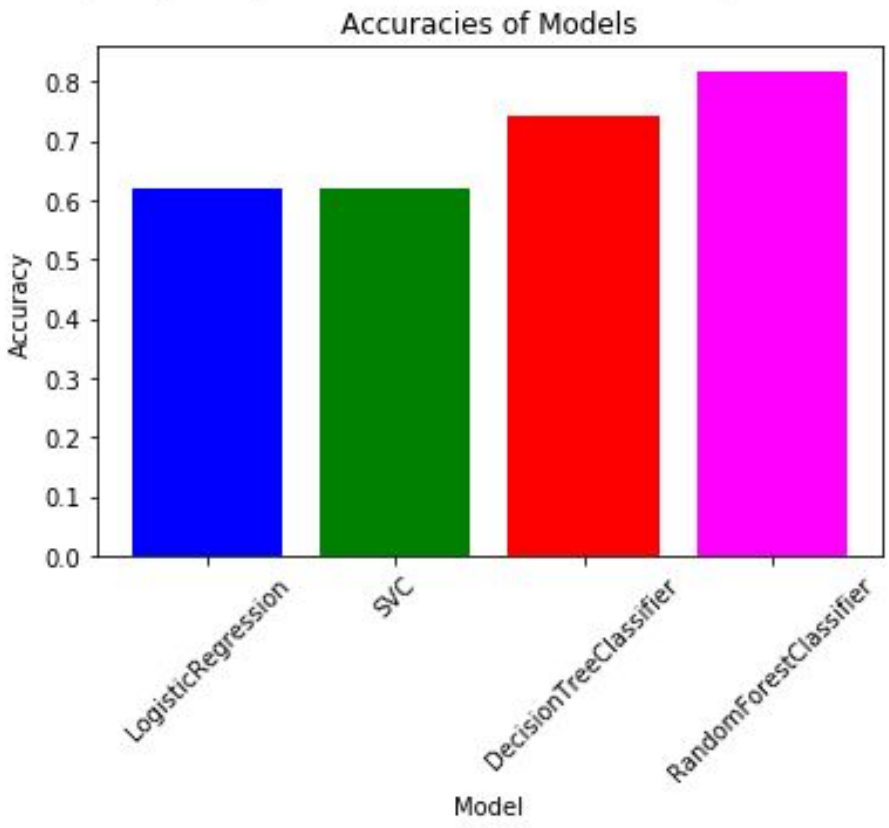



Fig. 4 Initial performance of the models

- After performing the hyperparameter tuning as discussed in the previous section, we noticed a significant improvement in the performance of the Logistic Regression model (new accuracy of 78.89%, beating the previous Decision Tree model). The Random Forests model also showed a slight improvement in its performance, now with an accuracy score of 82.9%.
- The new findings are summarized below:

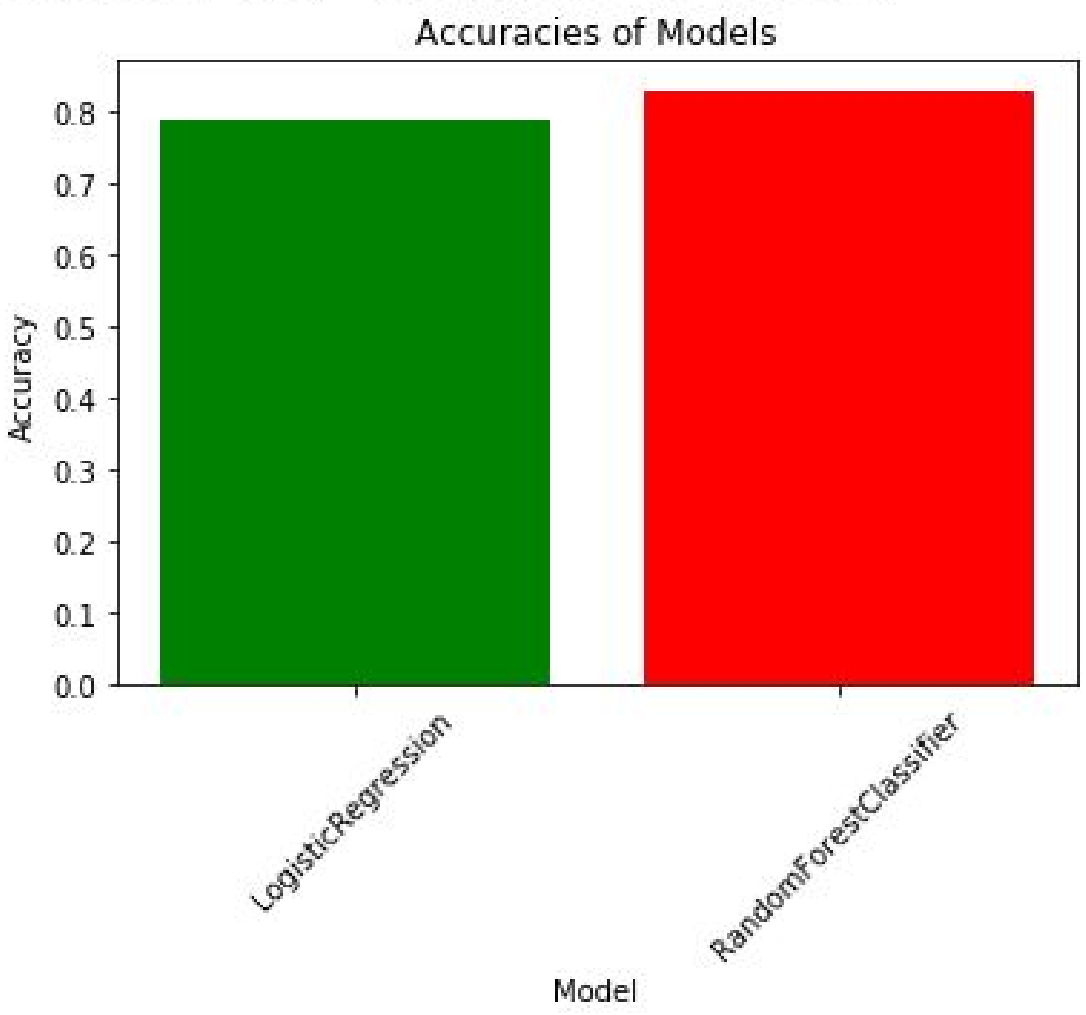



Fig. 5 Performance vs Benchmark

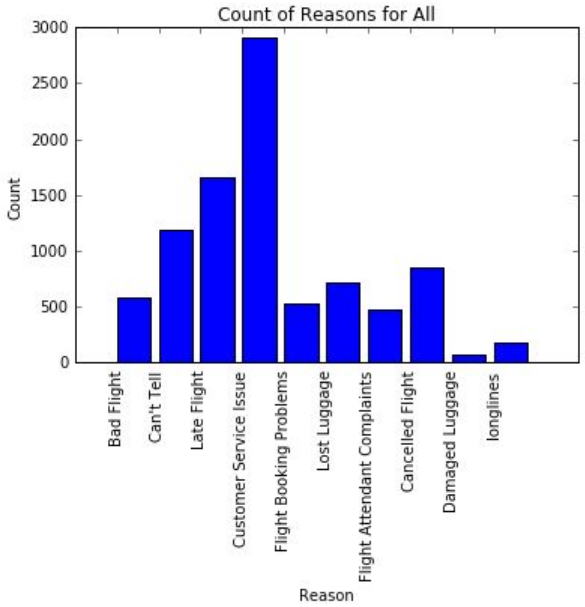
## Justification

- Since my benchmark Logistic Regression model, untuned, didn't give satisfactory results, I tweaked its parameters to get a decent benchmark accuracy score (72.8%) to beat. The Random Forests Classifier model that I preferred to use as the final model for this problem beats the performance of the benchmark model (82.9%).
- I feel the final model is a very decent solution to this sentiment analysis problem, as the Random Forests Classifier is always one of the preferred algorithms to be used for binary classification problems, and the architecture used here can be used for similar use-cases.

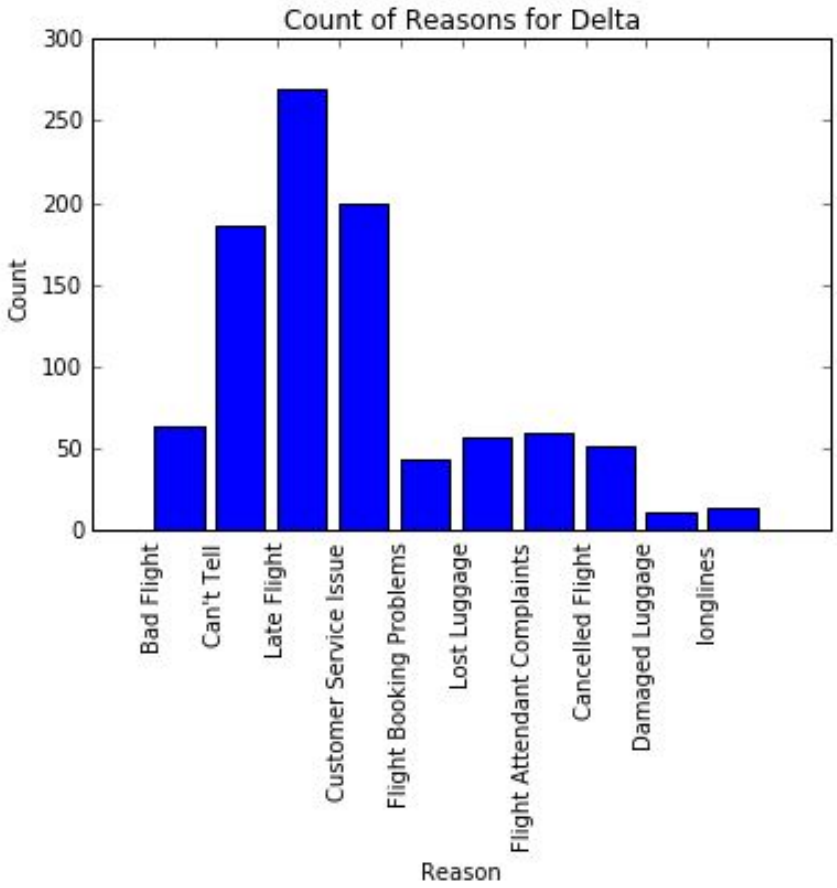
# V. Conclusion

## Free-Form Visualization

- From the given data, I found some interesting cues. From the *negativereason* attribute, I tried to visualize what are the most recurring issues of the airlines that people are dissatisfied about.



- Customer Service seems to be the most common issue affecting most customers, followed by Late Flight by a big margin. The bar graphs for all other airlines were found to be nearly the same, except for Delta which is given below.




- Delta customers seem to get affected most by Late Flight followed by Customer Service Issue. This is something Delta can address and work on, to improve customer satisfaction.

## Reflection

The entire process can be summarized using the following steps:

- Identifying a problem and obtaining the dataset.
- Downloading the data.
- Understanding the data through graphs and plots.
- Performing data preprocessing: removing undefined data, removing unnecessary data.
- Performing text preprocessing: cleaning of tweets.
- Simplifying multi-class classification into a binary classification problem.
- Splitting data into training and testing data.
- Training our models and making predictions.
- Evaluating the models and making improvements with hyperparameter tuning.
- Acquiring the final model.
- I enjoyed working on this sentiment analysis problem: visualizing the data, identifying key features and unimportant features, understanding the encoding method of CountVectorization and implementing it.
- What I found challenging in this project was deciding the workflow of the process, the intermediate steps to be performed for achieving better results, and understanding the nuances of the model parameters, and what fits well to this problem.

## Improvement

- I think one way to achieve better results on this dataset will be by overfitting the data, since this is a pretty small dataset. But that would not do justice to unseen data.

- With the final result as the new benchmark, we can certainly build better solutions. We can use the concept of Priority Polarity Scoring, that is, to assign a score to the pleasantness of a word (a word like ‘delighted’ would have a higher score than a word such as ‘disappointed’). Therefore, it would further enhance the model’s performance at classification. Read more here.