**Twitter Sentiment Analysis**

**Abstract:**

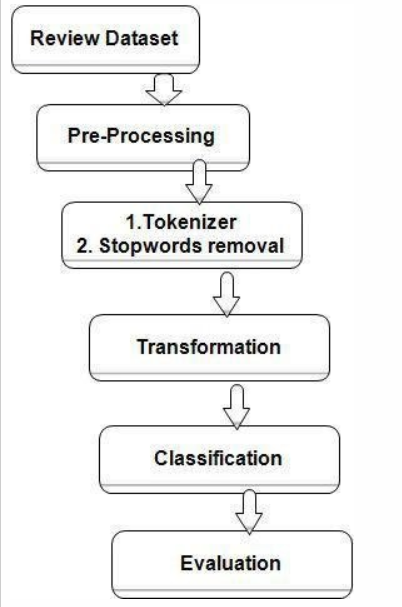
Social media has engaged and attracted good amount of audience in the past recent years. These platforms are used to express views and opinions on various subject areas. Twitter is one of the platforms widely used by people to express their opinions and showcase sentiments on various occasions. Sentiment analysis is an approach to analyze data and retrieve sentiment that it embodies. Twitter sentiment analysis is an application of sentiment analysis on tweets, to analyze sentiments emoted by the user. In the past decades, the research in this field has consistently grown. The reason behind this is the challenging format of the tweets which makes the processing difficult. The tweet format is very small which generates a whole new dimension of problems like use of slang, abbreviations etc. This paper reports on the exploration and preprocessing of data, transforming data into a proper input format and classify user’s perspective via tweets into positive(non-racist/sexist) and negative (racist/sexist).

The objective of this project is to detect hate speech in tweets. For the sake of simplicity, we say a tweet contains hate speech if it has a racist or sexist sentiment associated with it. So, the task is to classify racist or sexist tweets from other tweets.

Given a training sample of tweets and labels, where label ‘1’ denotes the tweet is racist/sexist and label ‘0’ denotes the tweet is not racist/sexist, your objective is to predict the labels on the given test dataset.

**METHODOLOGY:**

In order to perform sentiment analysis, we are required to collect data from the desired source (here Twitter). This data undergoes various steps of pre-processing which makes it more machine sensible than its previous form.



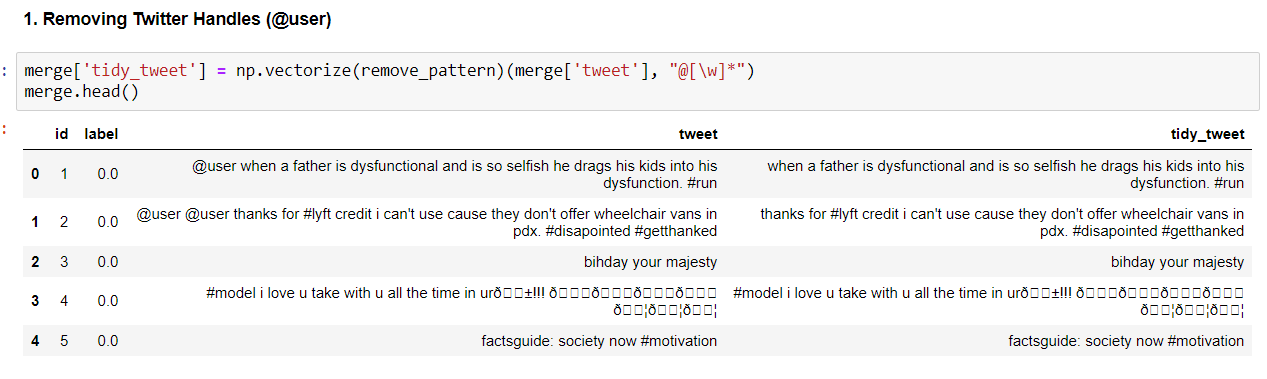
**Tweets Preprocessing and Cleaning**

The preprocessing of the text data is an essential step as it makes the raw text ready for mining, i.e., it becomes easier to extract information from the text and apply machine learning algorithms to it. The objective of this step is to clean noise those are less relevant to find the sentiment of tweets such as punctuation, special characters, numbers, and terms which don’t carry much weightage in context to the text.

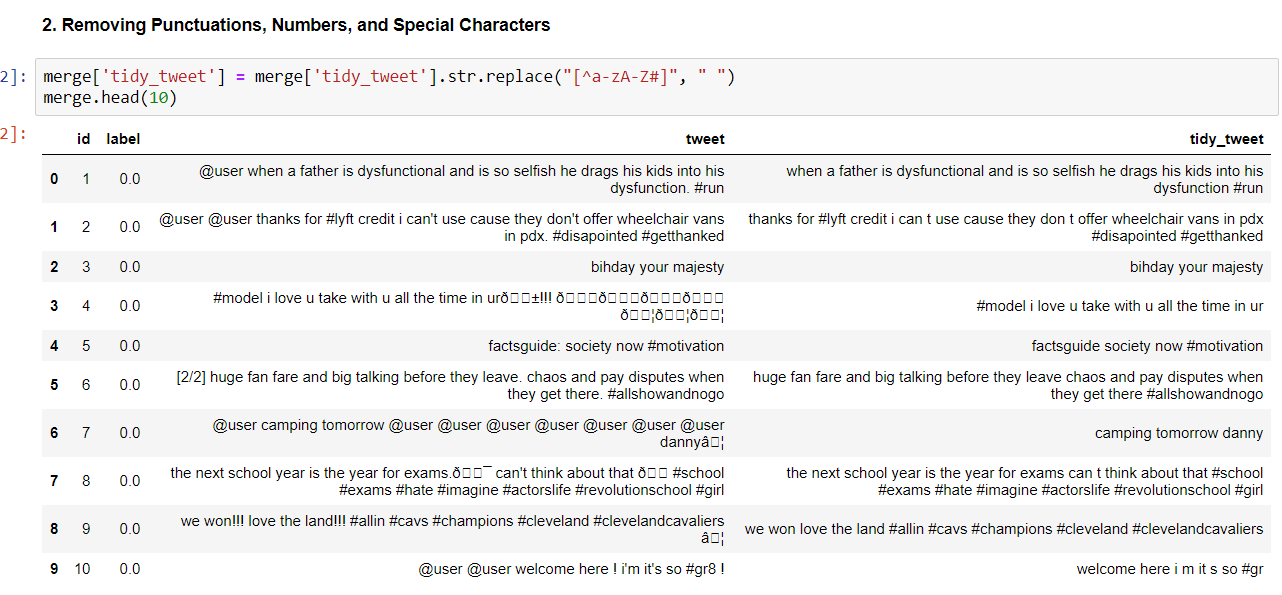
Below steps were involved as a part of preprocessing:



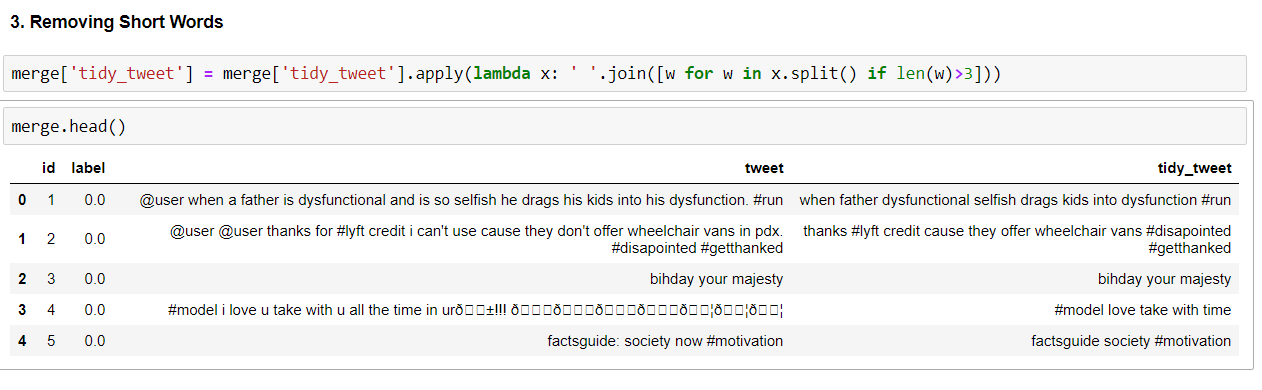
1. Removing Twitter Handles (@user)



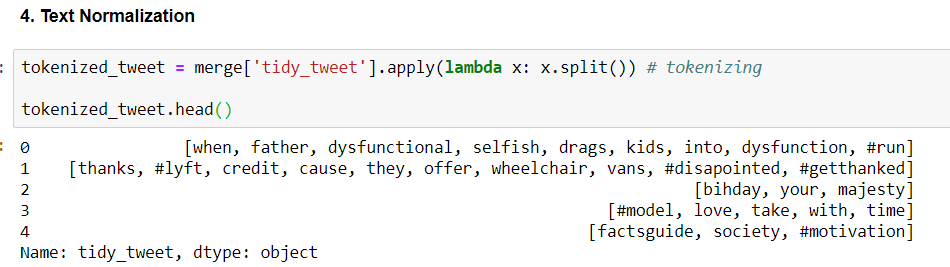
1. Removing Punctuations, Numbers, and Special Characters:

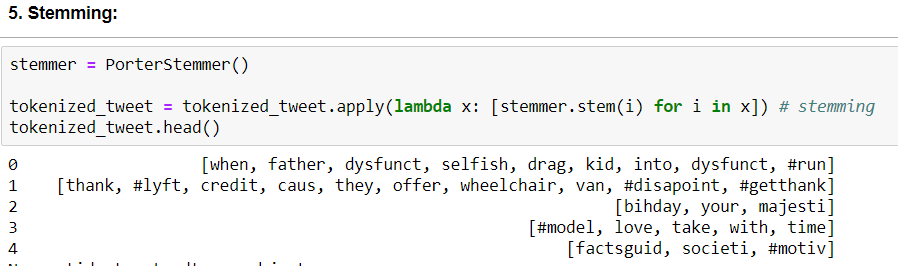


1. Removing Short Words:



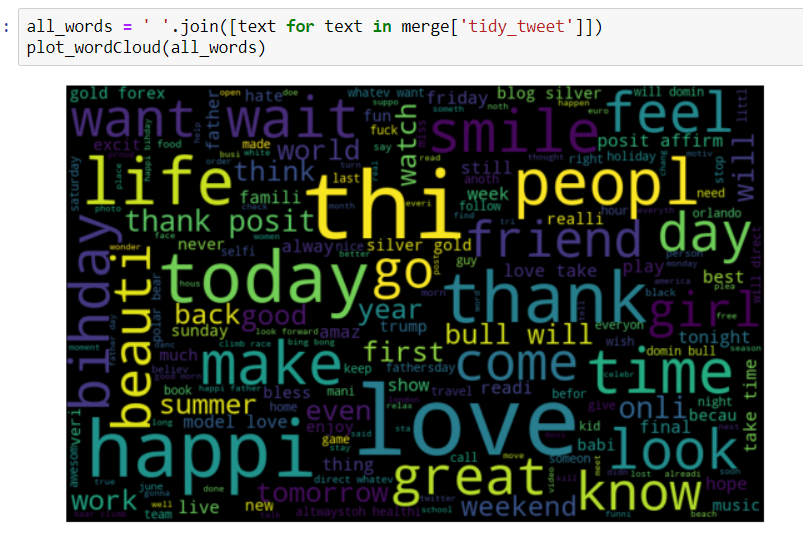
1. Tokenization:



1. Stemming: Stemmers remove morphological affixes from words, leaving only the word stem

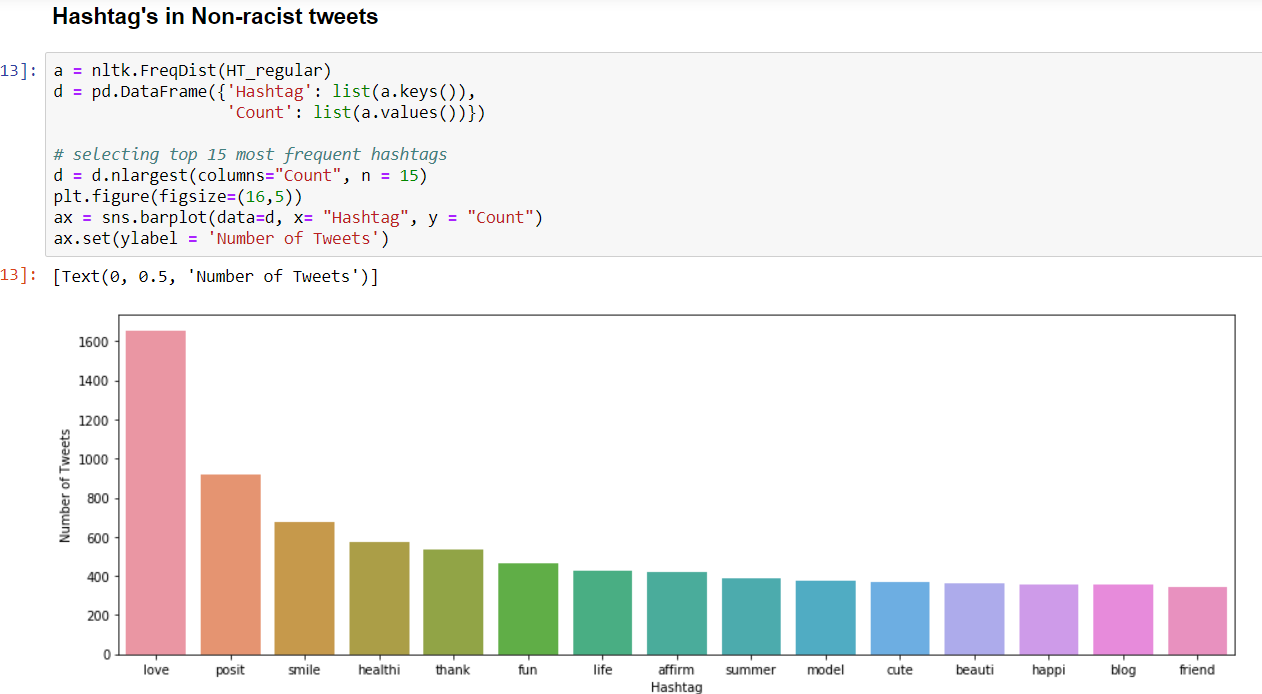
**Visualization:**

### **WordCloud**: *A wordcloud is a visualization wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes.*



**Hashtag Analysis:**

We have extracted the hashtags from the tweets to analyze the hashtags associated with the different categories of tweets.





**Extracting Features from Cleaned Tweets**

**Bag-of-Words Features:** Bag-of-Words is a method to represent text into numerical features.

**TF-IDF Features:** TF-IDF works by penalizing the common words by assigning them lower weights while giving importance to words which are rare in the entire corpus but appear in good numbers in few documents.

Let’s have a look at the important terms related to TF-IDF:

TF = (Number of times term t appears in a document)/(Number of terms in the document)

IDF = log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in.

TF-IDF = TF\*IDF

**Word2Vec Embeddings:**



**Doc2Vec Embeddings:**



**Evaluation metrics:**

**F1 Score:**

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). High precision but lower recall, gives an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model. Mathematically, it can be expressed as :

https://cdn-images-1.medium.com/max/1000/1*_pYttqYh8w-EpLxMi84H8A.gif

F1 Score tries to find the balance between precision and recall.

* **Precision :**It is the number of correct positive results divided by the number of positive results predicted by the classifier.

https://cdn-images-1.medium.com/max/1000/1*wMIDfGwT9bA6HezvYhdbpQ.gif

Precision

* **Recall :**It is the number of correct positive results divided by the number of ***all***relevant samples (all samples that should have been identified as positive).

https://cdn-images-1.medium.com/max/1000/1*gIlQMZBPjtUHWkwOhRG3KA.gif

**Confusion matrix**

It is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values. gives us a matrix as output and describes the complete performance of the model.

**Accuracy**: Overall, how often is the classifier correct?

(TP+TN)/total

**Area Under Curve:**

Area Under Curve(AUC) is one of the most widely used metrics for evaluation. It is used for binary classification problem. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example.

False Positive Rate and True Positive Rate both have values in the range [0, 1].. AUC is the area under the curve of plot False Positive Rate vs True Positive Rate at different points in [0, 1].

Area under ROC curve is often used as a measure of quality of the classification models. A random classifier has an area under the curve of 0.5, while AUC for a perfect classifier is equal to 1. In practice, most of the classification models have an AUC between 0.5 and 1

As evident, AUC has a range of [0, 1]. The greater the value, the better is the performance of our model.

When a classifier cannot distinguish between the two groups, the area will be equal to 0.5 (the ROC curve will coincide with the diagonal). When there is a perfect separation of the two groups, i.e., no overlapping of the distributions, the area under the ROC curve reaches to 1 (the ROC curve will reach the upper left corner)

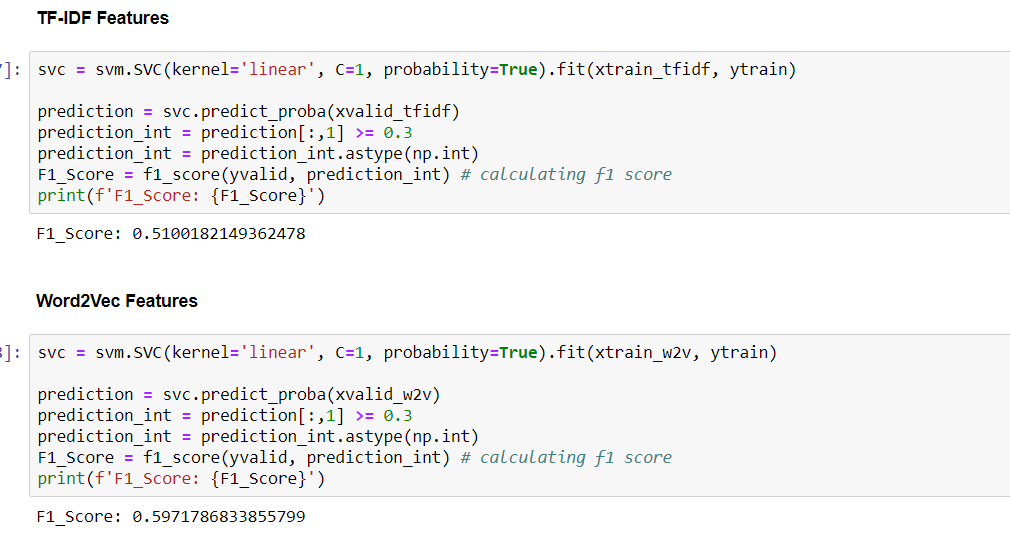
**Model Building:**

We are now done with all the pre-modeling stages required to get the data in the proper form and shape. We will be building models on the datasets with different feature sets prepared in the earlier sections — Bag-of-Words, TF-IDF, word2vec vectors, and doc2vec vectors. We will use the following algorithms to build models:

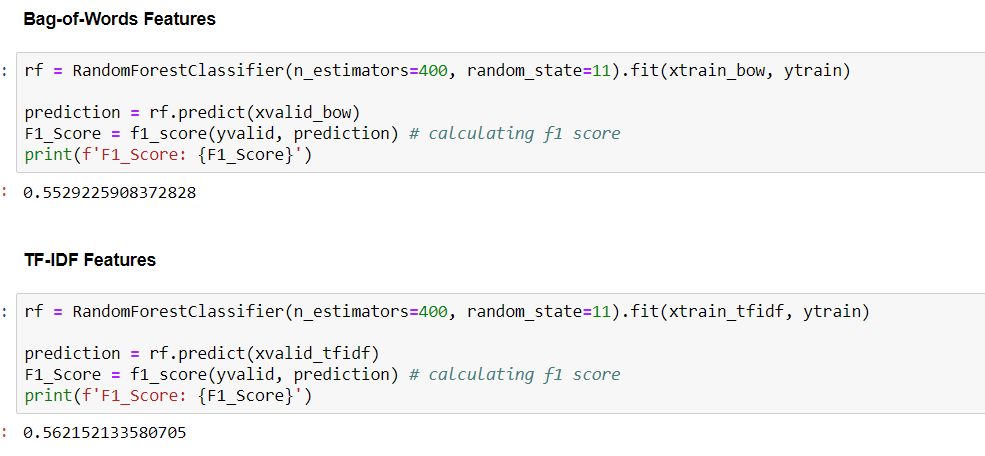
1. Logistic Regression
2. Support Vector Machine
3. RandomForest
4. XGBoost
5. **Logistic Regression:**



### **B. Support Vector Machine**



### **3. RandomForest**



**D. XGBooost:**

**HyperParameter Tuning for XGBoost:**

We have tuned the below hyperarameters:

* min\_child\_weight:
  + Defines the minimum sum of weights of all observations required in a child.
  + This is similar to min\_child\_leaf in GBM but not exactly. This refers to min “sum of weights” of observations while GBM has min “number of observations”.
  + Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
  + Too high values can lead to under-fitting hence, it should be tuned using CV.
* max\_depth:
* The maximum depth of a tree, same as GBM.
* Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
* Should be tuned using CV.
* gamma [default=0]
* A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split.
* Makes the algorithm conservative. The values can vary depending on the loss function and should be tuned.
* subsample [default=1]
* Same as the subsample of GBM. Denotes the fraction of observations to be randomly samples for each tree.
* Lower values make the algorithm more conservative and prevents overfitting but too small values might lead to under-fitting.
* colsample [default=1]
* Similar to max\_features in GBM. Denotes the fraction of columns to be randomly samples for each tree.

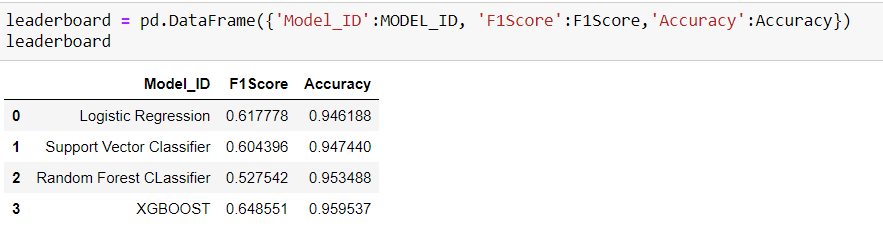
**Distinguishing features implemented by us:**

* Extraction and consideration of different feature sets being BoW, TF-IDF, Word2Vec and Doc2Vec for model testing
* Performing 4 different machine learning algorithms on the data provided.
* Hyperparameter tuning of XGBoost algorithm using grid search
* Model performance evaluation considering the F1 score and Accuracy

**Conclusion:**

In this project, we used multiple training models to make prediction and tried to find the most appropriate model on our dataset. During the process, we learned the analyzing steps and training models in NLTK and scikit-learn library which are helpful and easy to use. With different vectorizer, TFIDF and Count vectorizer, it will generate different results. For example, we can improve the model by removing words in very low or high frequency by TFIDF. It will make the prediction model to filter outliers and improve the accuracy of model.

Below is the comparison of different models considering the evaluation metrics of F1 score and Accuracy:



On evaluation of the models like Logistic Regression, SVM, Random Forest and XGBoost on various features extracted which are Bag of words, Word2Vec, Doc2Vec and TF\_IDF. Considering the evaluation metric of F1 score, our best performing model is XGBoost with tuned params applied on the Word2Vec features with an F1 score of 0.64.

## **Future Scope:**

1. Scaling the project to accomodate live tweets using 'tweepy' API
2. Articulating domains like socio-ecomonic,political views using tweets
3. Ensembling the different features like TF-IDF,Bag of words, Word to Vectors and Documents to Vectors as a single feature set

**Citations**

* <https://github.com/nikbearbrown/CSYE_7245/blob/master/Projects/Research_Papers/Amazon_Fine_Food_Reviews_Analysis.pdf>
* To hide the warning messages: <https://stackoverflow.com/questions/9031783/hide-all-warnings-in-ipython>
* To resolve Matplotlib version issue.: <https://github.com/mwaskom/seaborn/pull/1380>
* To resolve singular matrix issues: <https://stackoverflow.com/questions/10326015/singular-matrix-issue-with-numpy>
* <https://www.datacamp.com/courses/natural-language-processing-fundamentals-in-python>
* TF-IDF logic: <https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e>
* Understanding Data : <https://public.tableau.com/views/BNPParibasCardifClaimsManagement-ExploratoryAnalysis/Sheet1?%3Aembed=y&%3AshowVizHome=no&%3Adisplay_count=y&%3Adisplay_static_image=y>

**License**:

Copyright 2019 Anand Kumar, Varsha Premani

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.