TECHNOCOLABS DATA SCIENCE INTERNSHIP

PROJECT REPORT

TITLE: Bots in the Net-Applying Machine Learning to Identify Social Media Trolls

AIM:

The principle focus of our project is to perform data analysis and train a model using Machine Learning algorithms. By doing so, it helps us to identify if a particular text/tweet is a troll or not by building a classifier.

ABSTRACT:

We have proposed a basic classfier for a tweet which detects if the tweet is by a troll bot or not. In this paper, ML models are used to identify political trolls across social media and compare their accuracies. Using these trained models, we build a classifier for Twitter content that predicts the likelihood a given tweet was created by a troll bot in real time. This tool can aid moderation for social media companies by flagging suspicious tweets in a live feed.

INTRODUCTION:

Following the 2016 elections, the United States and the world at large noticed a disturbing phenomenon of the use of social media for malicious political influence. It was later discovered that the Internet Research Agency, Russia's social media troll farm, spread disinformation to tens of millions of American voters using targeted tweets and Facebook posts (US, 2018).

OVERVIEW:

- Data Segmentation and Data Cleaning
- Exploratory Data Analysis
- Training the model based on the data available

DATASET:

The dataset contains data on nearly 3 million tweets sent from Twitter handles connected to the Internet Research Agency, a Russian "troll factory" and a defendant an indictment filed by the Justice Department in February 2018, as part of special counsel Robert Mueller's Russia investigation. The tweets in this database were sent between February 2012 and May 2018, with the vast majority posted from 2015 through 2017.

Original format of the dataset: XLSX

Converted to : CSV

A brief explanation of every column in the dataset is as follows:

- **External author ID** – This column displays all the external Author ID’s of every troll/ tweet posted during the specified time. It is unique for each tweet.
- **Author** – This column mentions the author of the tweet. However, due to privacy issues, the names are not given but the author is filled with a common name “10_GOP”.
- **Content** – This column mentions the tweet posted by the so called author.
- **Region –** It Refers to the geographic region from where the particular tweet is posted. Due to privacy issues, the region column is filled with “Unknown”.
- **Language -** This column specifies in what language the tweet is posted.
- **Published date –** This indicator refers to the date which the tweet was posted.
- **Harvest date** – This column refers to the date at which the tweet was harvested by IRA.
- **Following –** This refers to the total number of twitter accounts he/she followed at the specified time.
- **Followers –** This refers to the total number of followers the so-called twitter account had during the specified time.
- **Updates** – This refers to the total number of tweets posted by an account since the day the account was created.
- **Post type** – comprises of the type of tweet posted. It mentions whether the tweet was originally posted by the author or retweeted another post.
- **Account type –** This mentions the type of twitter account used.
- **Retweet –** This indicates if a particular tweet is retweeted again or an original tweet. If the tweet is original = 0 and retweet =1.
- **Account category–** It mentions the type of category in which the account falls into. In this case, the category is RightTroll.

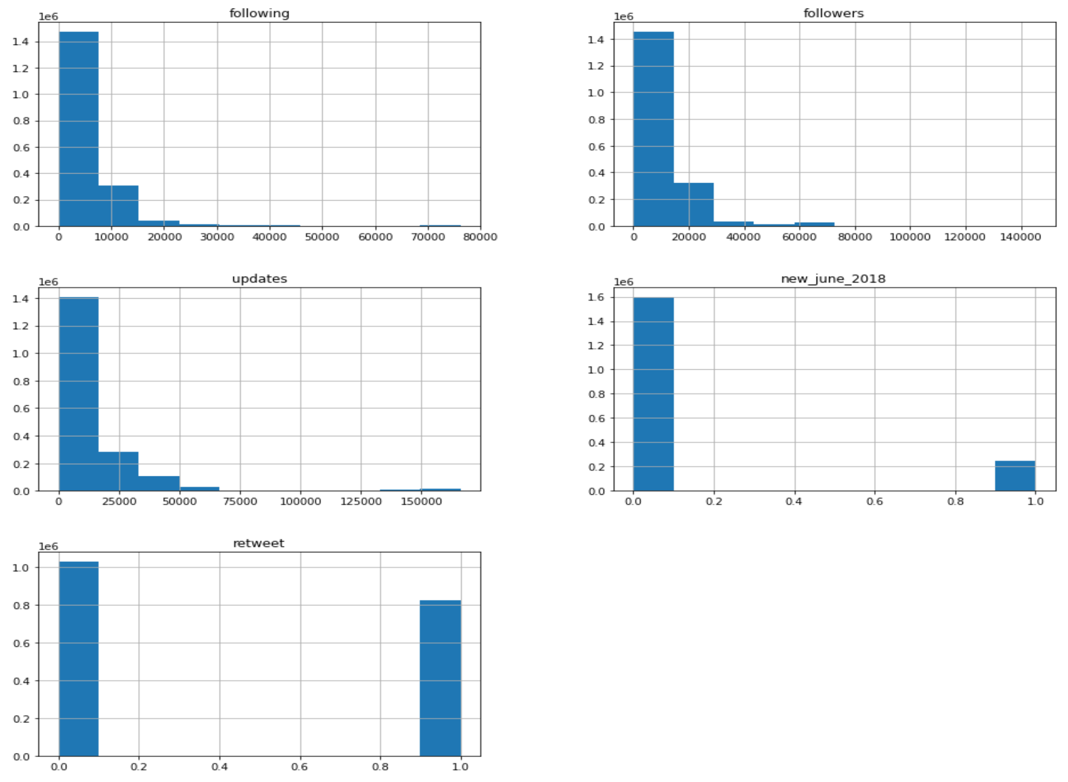
DATA CLEANING:

Data cleaning refers to identifying and correcting errors in the dataset that may negatively impact a predictive model. Data cleaning is used to refer to all kinds of tasks and activities to detect and repair errors in the data. In this, we removed the duplicate data like rename "right" and "Right" to "Right", and removing the NULL values.
And also done the Data Visualization, This is helpful for exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more.

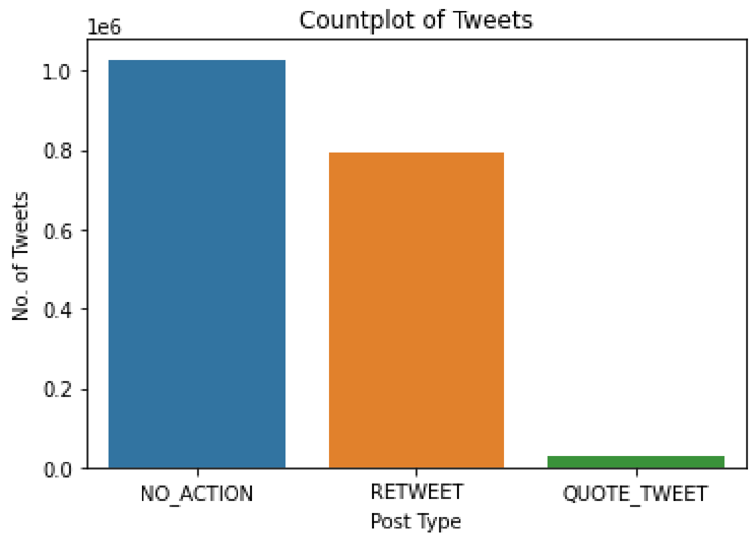
And finally, We try the Random Forest Model to the Dataset. The Random forest is a classification algorithm consisting of many decision trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. While applying I've set the parameter n_estimators=50 and by this, we got an accuracy of less than 80%, which is not quite good.

EXPLORATORY DATA ANALYSIS:

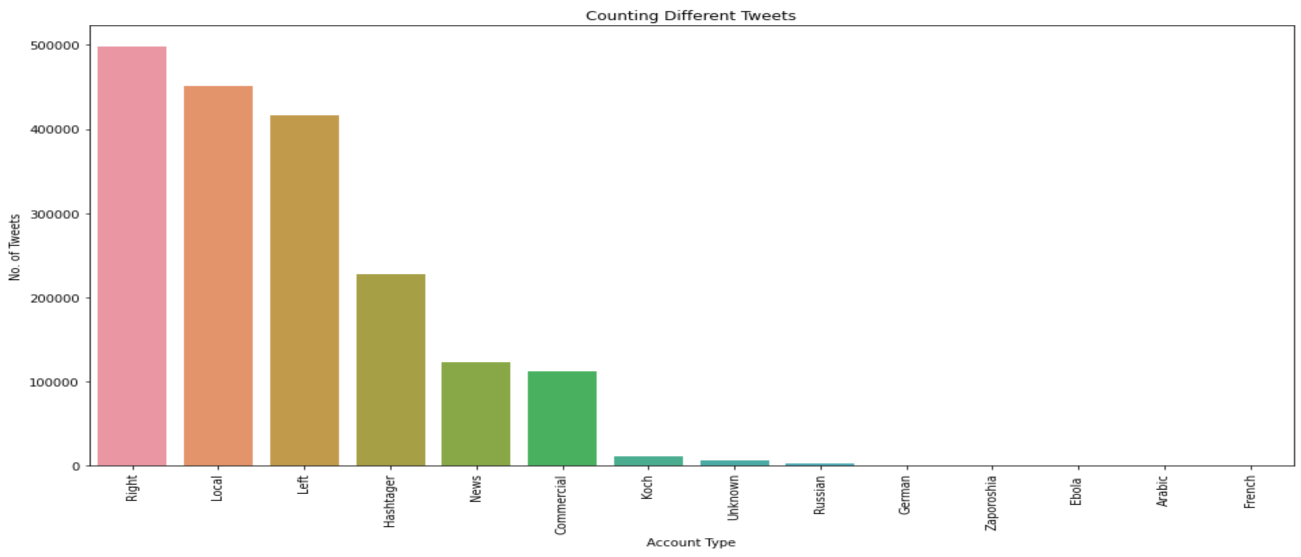
- **Plot 1 –** This plot shows an overall values of the dataset.



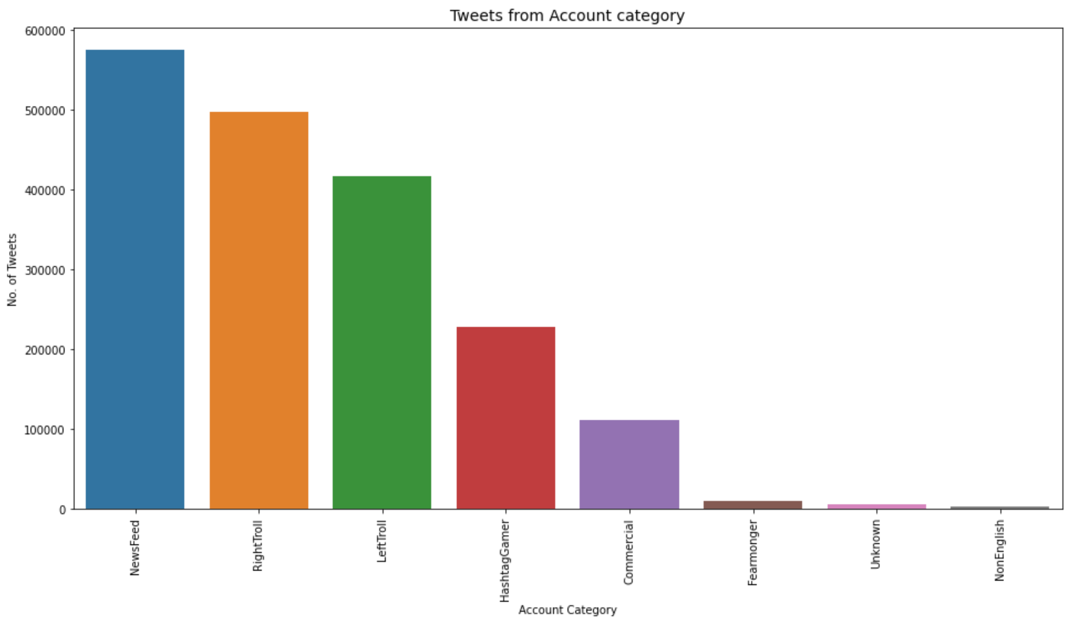
- **Plot 2** – This shows a countplot of all the tweets.



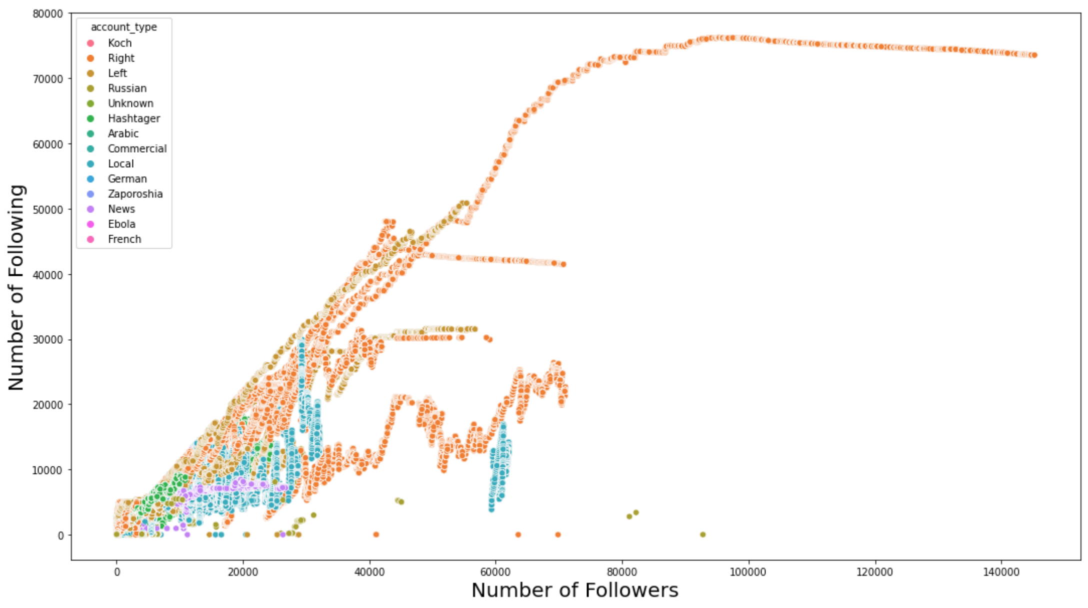
- **Plot 3** – This explains the type of account and the corresponding tweet posted along with their count of tweets.



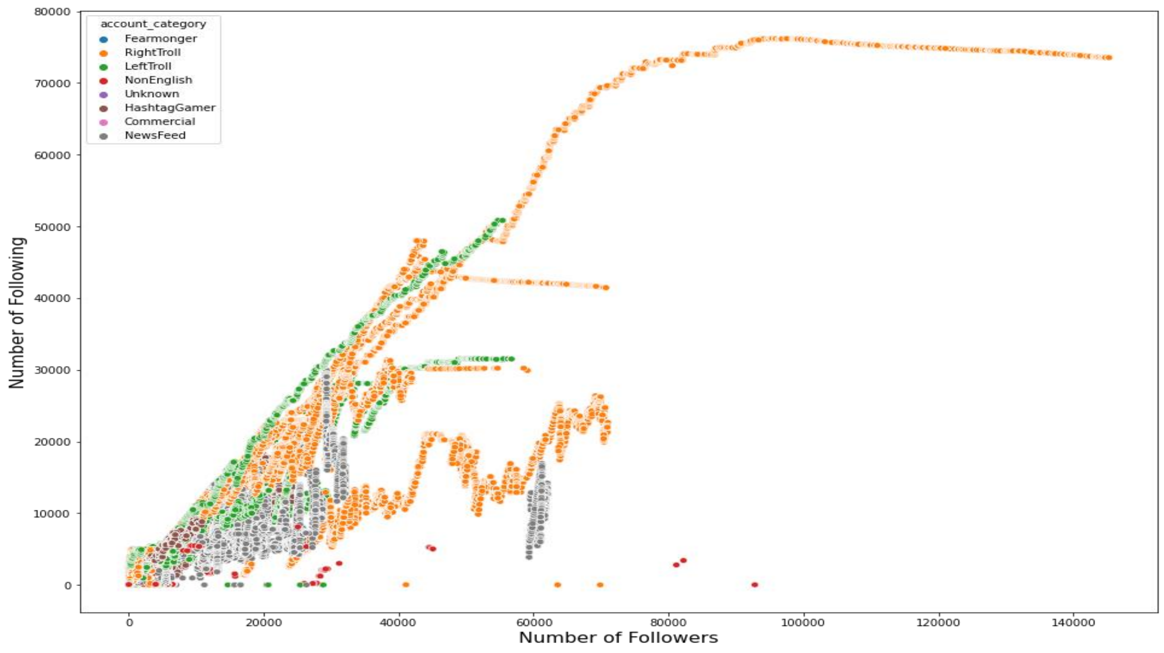
- **Plot 4** – This explains the number of tweets based on their accounts.



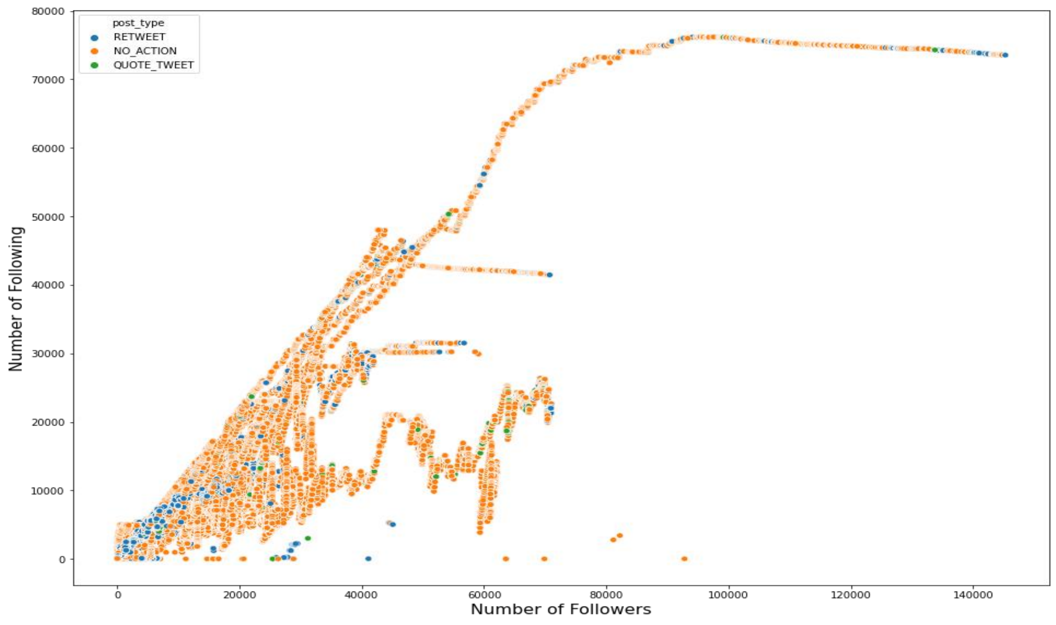
- **Plot 5** – This scatterplot explains the relationship between followers and following of all the twitter account mentioned based on their type of account.



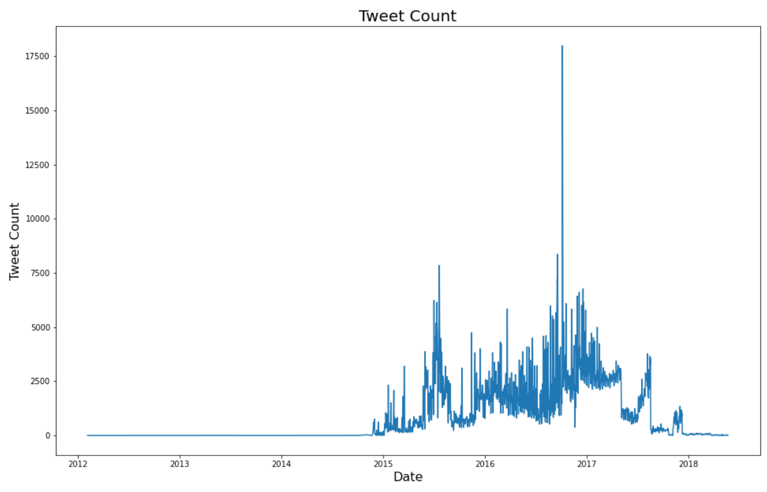
➢ **Plot 6** – This explains the relationship between followers and following of all the accounts based on their account category.



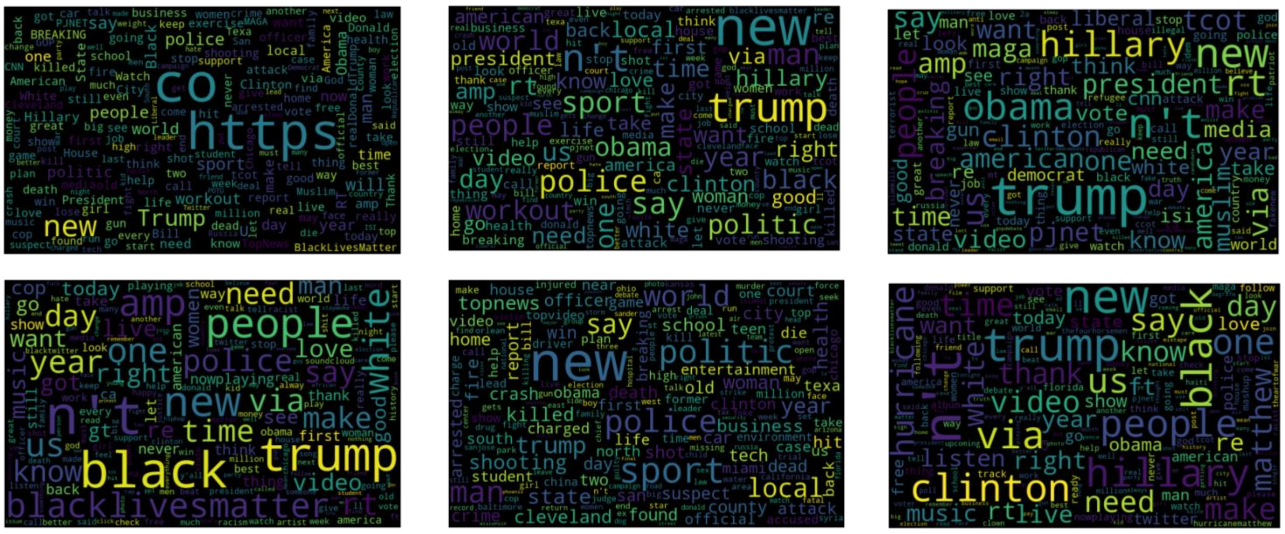
➢ **Plot 7** – This scatterplot explains the relationship between followers and following of all the accounts based on their tweet type.



➢ **Plot 8** – This line plot explains the increasing number of tweets during a specific time period.



- **Plot 9** – Word clouds created during EDA.



TEXT PREPROCESSING :

Text preprocessing is traditionally an important step for natural language processing (NLP) tasks. It transforms text into a more digestible form so that machine learning algorithms can perform better.

Text preprocessing is one of the important step in a NLP problem, there are certain steps to perform text preprocessing. These steps are needed for transferring text from human language to machine-readable format for further processing. We will also discuss text preprocessing tools.

After a text is obtained, we start with text normalization. Text normalization includes:

- converting all letters to lower or upper case.
- converting numbers into words or removing numbers
- removing punctuations, accent marks and other diacritics
- removing white spaces
- expanding abbreviations
- removing stop words, sparse terms, and particular words
- text canonicalization

TRAINING AND TESTING THE MODEL:

The following models are trained and tested for this project.

- Random forest Classifier
- SVM
- Logistic regression

- **Random forest Classifier**

The Random Forest algorithm utilizes multiple decision trees where only a random subset of features are considered. The random forest approach allows us to capture a greater range of correlations between different features for each of our featurizations, wherein the algorithm can capture correlations between words. We utilize the Gini impurity criterion, which is represented by the equation GINI(t) = 1 −∑ji=1 P(i|t)2.

```
Random Classifier Accuracy:  87.30303030303031 % (Bag of words)
Random Classifier Accuracy:  84.15151515151516 % (TF-IDF)
```

- **SVM**

Support Vector Machines attempt to compute a decision boundary that maximizes the margin of data, i.e. data is the most separated possible. We use the RBF kernel, defined by K(x, x′) =exp(−||x−x||22σ2). SVM performs optimization according to the following constraints:

minw,b12||w||2
s.t. y(i)(wT x(i) + b) ≥ 1, i = 1, . . . , n

```
SVM Accuracy:  82.0909090909091 % (Bag of words)
SVM Accuracy:  83.75757575757575 % (TF-IDF)
```

- **Logistic Regression**

Logistic regression is a classification algorithm that maximizes the likelihood of labels given input data and tuned weights. In particular, logistic regression uses the sigmoid function, given by g(z) = 11+e−z , to express p(y = 1|x). Logistic regression then maximizes the following log likelihood, using gradient ascent:

l(θ) = ∑ni=1y(i)log g(θT x(i)) + (1 − y(i))log(1 − g(θT x(i)))

```
Logistic Regression Accuracy:  86.45454545454545 % (Bag of words)
Logistic Regression Accuracy:  83.87878787878788 % (TF-IDF)
```

- **LSTM Neural Network**

  - Long Short Term Memory networks – usually just called "LSTMs"
  - It is a special kind of RNN(recurrent neural network) which is capable of learning long-term dependencies.
  - These are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn

The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A_1 represents "completely keep this" while a_0 represents "completely get rid of this.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Next, we decide what new information we have to store in cell state. For that 'input gate layer' decides which value should be updated and tanh layer create a vector C_t, that could be added to the state.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

After that we work on old cell state for update that state. Which we usually represent by C_{t-1}, into the new cell state C_t. We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add i_t*C_t. This is the new candidate values, scaled by how much we decided to update each state value.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Finally, for output we have to decide what output should be, and This output will be based on our cell state, but will be a filtered version. Firstly, we run sigmoid layer and multiply it by output of sigmoid function so that we get only that output which we have decided.

$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Though after carefully learning everything and implementing model, I got some errors. It had not work as I expected.

However, I took help from internet blogs, you tubes and mentors as well. Finally, after discussing this problem with my internship mentor and my team leader, we concluded that I have to help other team members in deployment part. I helped them and tried to rectify errors which my team members were getting in deployment work.

MODEL EVALUATION :

Model evaluation aims to estimate the generalization accuracy of a model on future data. Methods for evaluating a model's performance are Cross-validation. This methods use a test setCross-validation is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis.The most common cross-validation technique is k-fold cross-validation, where the original dataset is partitioned into k equal size subsamples, called folds. The k is a user-specified number, usually with 5 or 10 as its preferred value. This is repeated k times, such that each time, one of the k subsets is used as the test set/validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get the total effectiveness of our model.

DEPLOYMENT:

The project deployment is done using Flask. Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. The advantage of web applications is that they're platform independent and can be run by anyone who has access to the Internet. Their code is implemented on a back-end server, where the program processes incoming requests and responds through a shared protocol that's understood by all browsers.

Important files inside the program folder:

1. **app.py** contains your Python code wrapped in a minimal implementation of the Flask web framework.
2. **requirements.txt** lists all the dependencies your code needs to work properly.
3. **Procfile**- A **Procfile** is a list of process types in an app.

Local testing:

Flask comes packaged with a development web server. You can use this development server to double-check that your code works as expected. To be able to run the Flask development server locally, you need to complete two steps.

1. Set up a virtual environment.
2. Install the flask package.

```
if __name__ == "__main__":
   app.run(host="127.0.0.1", port=8080, debug=True)
```

These two lines tell Python to start Flask's development server when the script is executed from the command line. It'll be used only when you run the script locally. When Flask starts up the development server, and your terminal will display output similar to the text shown below:

```
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: This is a development server.
   Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
 * Restarting with stat
```

```
* Debugger is active!
* Debugger PIN: 315-059-987
```

Heroku deployment link: https://twitterb-ot.herokuapp.com/

TEAM MEMBERS

- DEEPTHIKA SHIWANI M
- HANNAMANTH GOLLAR
- MOHSIN RAZA
- ROHIT
- SAGAR GYANCHANDANI
- SAIF ALI
- SHIVAM CHOWDHURY
- SOMAY MANGLA