TWITTER FAKE ACCOUNT DETECTION USING MACHINE LEARNING ALGORITHMS

By Divyam Parhawal

Student at Amity Instiute of Information and Technology Department,

Amity University Noida, Uttar Pradesh , INDIA

divyam.parhwal@s.amity.edu

***Abstract*—In the world of social media and Internet, there is around 3.8 billion active social media users & 4.5 billion everyday internet users. The amount of internet users is increasing by roughly 9% every year and around the half of internet traffic mostly includes of fake accounts and bots. The project aims to use machine learning algorithms to detect fake followers on Twitter, which can be a useful tool for users who want to ensure the authenticity of their followers and improve the overall quality of their audience. To achieve this, the project identifies a number of features which distinguish genuine and fake followers, which are then used as attributes for the machine learning algorithms. Some common indicators of fake followers include a high number of followers relative to their activity on the platform, low engagement rates, and a lack of personal information on their profiles. However, the project may also consider other factors in the detection process. It's important to note that machine learning algorithms are not always perfect and may produce false negatives and false positives. Effectiveness of the features employed and the quality of the training data both affect how accurate the algorithm is. Additionally, there are ethical implications to consider when using machine learning to detect fake followers, as it could potentially lead to the wrongful targeting of legitimate users.**

Keywords—Fake Accounts, Bots, Twitter Fake Account Detection, Machine Learning

I. INTRODUCTION

Twitter is a famous online social networking platform that provides a medium for people to communicate, share information, and connect with each other. However, with the increasing use of Twitter, there has been an alarming rise in the number of fake accounts, which poses a serious threat to the security and privacy of genuine users. These fake accounts can be used to disseminate spam, malware, and propaganda, and can also be employed in cyber-attacks. To combat this problem, various researchers have explored the development of algorithms and techniques for detecting fake accounts on Twitter. These techniques range from using graph-based features and syntactic and semantic analysis to machine learning approaches. In this project, we aim to investigate the problem of fake account detection on Twitter using machine learning techniques. Specifically, we will explore the use of graph-based features and machine learning algorithms to build a classifier that can detect fake accounts with high accuracy. We will also compare our approach with other existing techniques and evaluate its performance on a real-world Twitter dataset. The primary objective of the project is to provide an effective and reliable solution for detecting fake accounts on Twitter, which can help improve the overall security and privacy of Twitter users. We hope that the outcomes of this project will be useful for researchers, social media companies, and law enforcement agencies in identifying and mitigating the threat posed by fake accounts on Twitter.

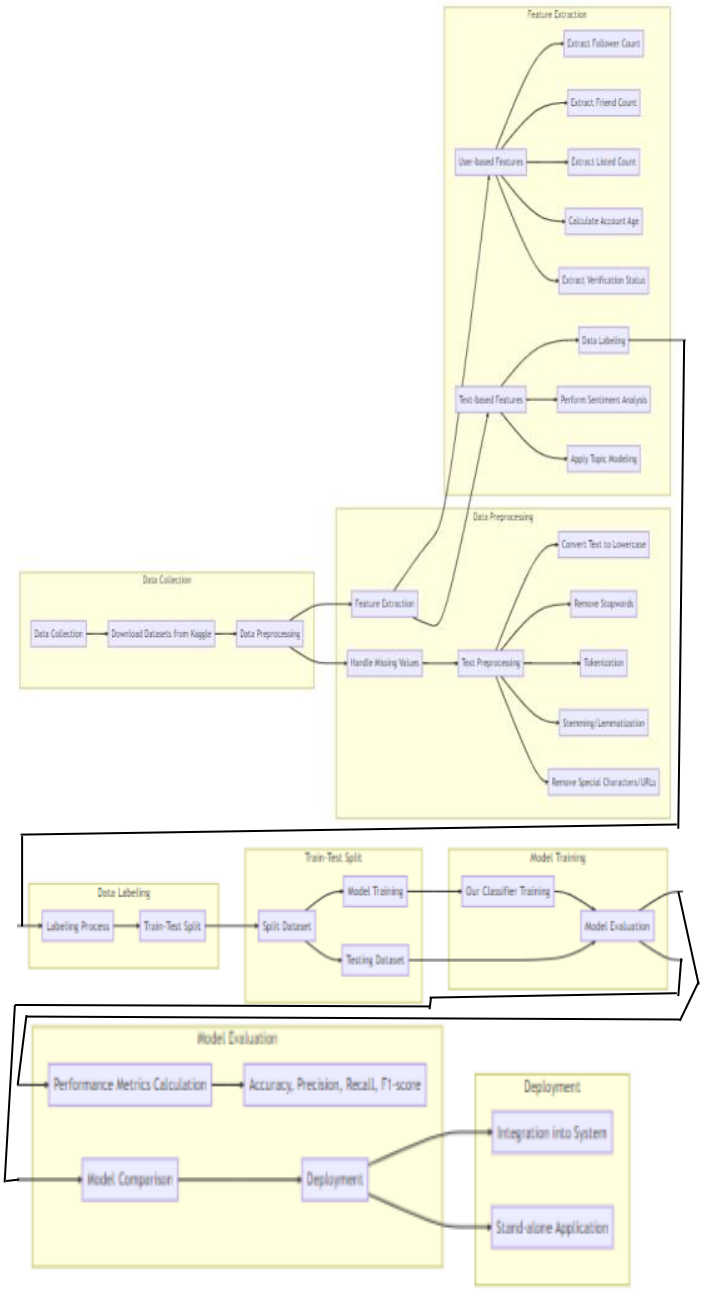
II. LITERATURE REVIEW

The detection of fake accounts has been a popular research topic in both computer networking and online social networking communities. Researchers have utilized various techniques to address this issue. One method involves using graph based features to detect fake accounts on social media platforms like Twitter. Wang (2010) [4] and Yang et al. (2011) [5] used this approach to develop classifiers for identifying bots, while Thomas et al. (2011) [6] applied similar features to retrospectively analyze numerous recently suspended Twitter account. In contrast, Boshmaf et al. (2011) [7] created the bots themselves and claimed that Facebook's immune system could not detect their bots. Although Sybil account detection has been applied to social network data, these techniques tend to depend on the "fast mixing" property of a network that may not be present in social networks (Mislove et al., 2007) and do not scale well to the size of current social networks. Chu and colleagues (2012), who are Twitter's anti-spam engineers, used graph theoretic, syntactic, and semantic features to categorize humans, bots & cyborgsd (human-assisted bots) in the Twitter dataset. Other related studies on Twitter usage and communities have also been conducted. Java et al. (2007) analyze around 70,000 users of Twitter and classified the posts made by them in four main categories, while Krishnamurthy et al. (2008) categorized more than 100,000q Twitter users into three groups based on their follower-to-following ratios. In addition, other research has focused on Twitter's information diffusion (Gomez-Rodriguez et al., 2010) and the utilization of Twitter lists to identify latent characters and user interests (Jansen et al., 2009). Kwak et al. (2010) performed an extensive quantitative analysis of

Twitter by looking across the Twittersphere. Their research examined follower to following topology and discovered a non-power-law follower distribution & low reciprocity's which suggest a departure from the familiar features of the human social networks [12], [13], [14].

III. SYSTEM DESIGN

A. *SYSTEM DIAGRAM*



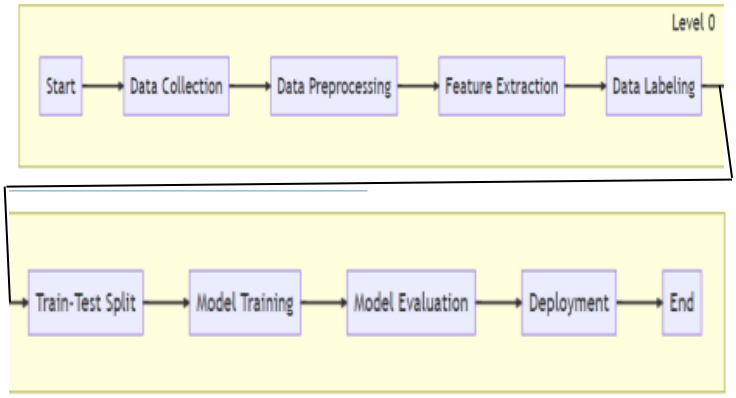
The data set was taken from Kaggle platform and using various machine learning algorithms and training data we have trained our classifier model. Test set is used on classifier model for giving prediction according to the given scenario.

B. *DFD(Data Flow Diagram)*

Each level of the DFD delves deeper into the specific tasks and processes involved in the overall workflow, providing a more granular understanding of the system's functionality.

1.1. Level-0 Data Flow Diagram(DFD):

It provides an overview of the entire process flow, including data collection, preprocessing, feature extraction, model training, evaluation, and deployment. Level-0 DFD is as follows:



Explanation of different components of the Level-0 DFD is as follows:

• Start: The starting point of the process.

• Data Collection: The phase where data is collected either through accessing the Twitter API or by importing datasets from Kaggle.

• Data Preprocessing: The process of handling missing values and performing text preprocessing on columns like description and status.

• Feature Extraction: The extraction of relevant features from the pre-processed data, including user-based features and text-based features.

• Data Labelling: The assignment of labels to the dataset indicating whether a Twitter account is a bot/fake or not.

• Train-Test Split: The splitting of the labelled dataset into training and testing datasets.

• Model Training: The training of different machine learning models using the training dataset.

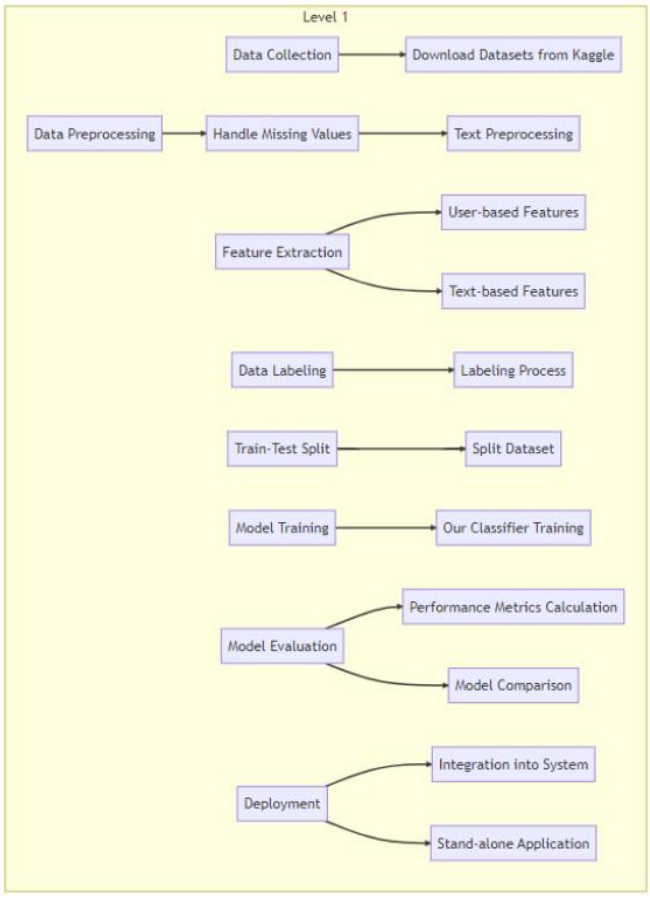
• Model Evaluation: The evaluation of the trained models using the testing dataset.

• Deployment: The deployment of the models for real-time fake account detection.

• End: The end point of the process.

1.2. Level-1 Data Flow Diagram(DFD):

It break-down the main processes into more detailed steps, such as downloading datasets, handling missing values, text preprocessing, feature extraction, labelling, dataset splitting, model training, performance metrics calculation, model comparison, and integration. Level-1 DFD is as follows:



Explanation of different components of the Level-1 DFD is as follows:

• Download Datasets from Kaggle: The specific step of downloading datasets from Kaggle.

• Handle Missing Values: The handling of missing values in the dataset.

• Text Preprocessing: The preprocessing of textual data to remove noise and standardize the format.

• User-based Features: The extraction of features derived from user profile attributes.

• Text-based Features: The extraction of features derived from textual data like description and status.

• Labelling Process: The process of assigning labels to the dataset.

• Split Dataset: The splitting of the dataset into training and testing datasets.

• Our Classifier Training: The training of the custom classifier "Our Classifier".

• Performance Metrics Calculation: The calculation of performance metrics like accuracy, precision, recall, and F1-score.

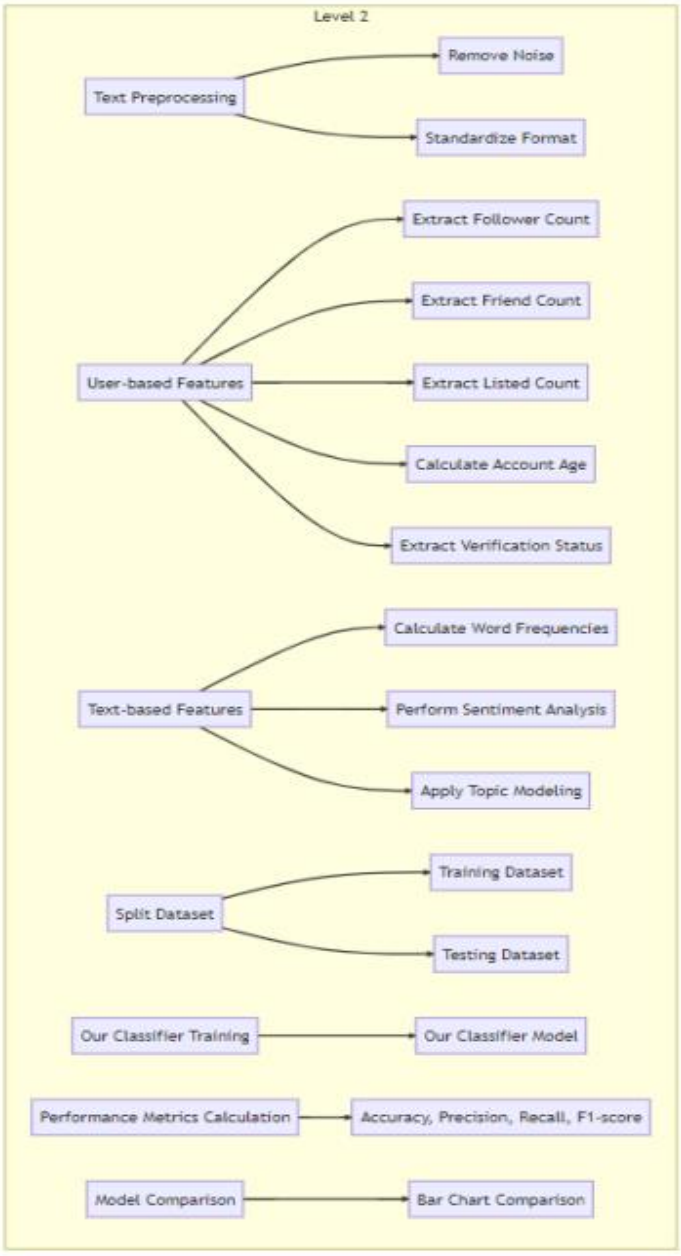
• Model Comparison: The comparison of the performance metrics of different classifiers.

• Integration into System: The integration of the models into a larger system.

• Stand-alone Application: The use of the models as standalone applications.

1.3. Level-2 Data Flow Diagram(DFD):

It further expands on the steps involved in text preprocessing, feature extraction, dataset splitting, and model training. It includes specific tasks like removing noise, standardizing formats, extracting user-based features, calculating account age, and applying text analysis techniques. Level-2 DFD is as follows:



Explanation of different components of the Level-2 DFD is as follows:

• Remove Noise: The step of removing noise from the textual data.

• Standardize Format: The step of standardizing the format of the textual data.

• Extract Follower Count: The extraction of the follower count feature from user profiles.

• Extract Friend Count: The extraction of the friend count feature from user profiles.

• Extract Listed Count: The extraction of the listed count feature from user profiles.

• Calculate Account Age: The calculation of the account age feature from the created date.

• Extract Verification Status: The extraction of the verification status feature from user profiles.

• Calculate Word Frequencies: The calculation of word frequencies from textual data.

• Perform Sentiment Analysis: The analysis of sentiment expressed in the textual data.

• Apply Topic Modelling: The application of topic modeling techniques to identify latent topics.

• Training Dataset (X_train, y_train): The training dataset split into feature data (X_train) and label data (y_train).

• Testing Dataset (X_test, y_test): The testing dataset split into feature data (X_test) and label data (y_test).

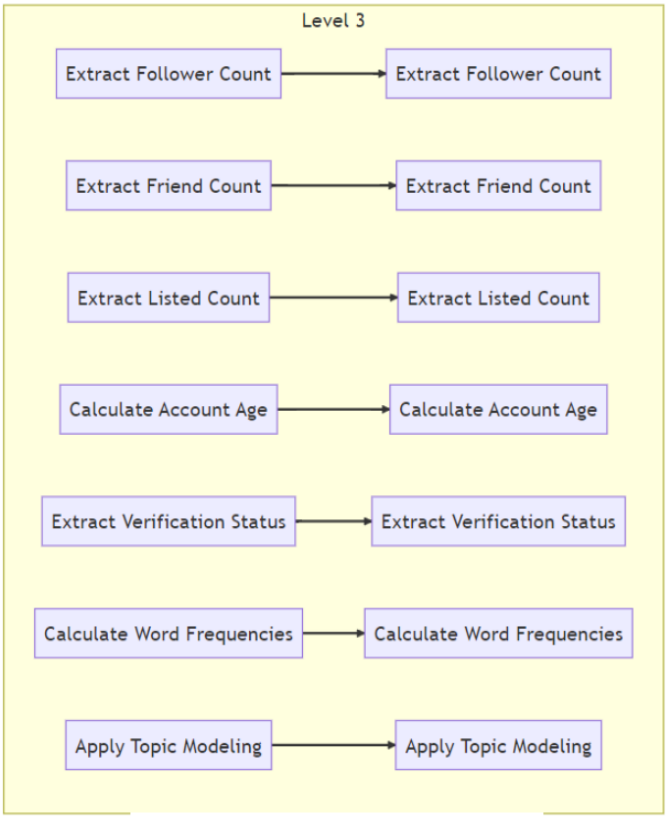
• Our Classifier Model: The trained model for the custom classifier "Our Classifier".

• Accuracy, Precision, Recall, F1-score: Performance metrics calculated for the trained models.

• Bar Chart Comparison: A bar chart comparing the performance metrics of different classifiers

1.4. Level-3 Data Flow Diagram:

Provides additional details on the processes involved in feature extraction, such as extracting follower count, friend count, listed count, verification status, and performing text analysis tasks like sentiment analysis and topic modelling.



Level 3 Data Flow Diagram

Explanation of different components of the Level-3 DFD is as follows:

• Extract Follower Count: Further details on the extraction of the follower count feature.

• Extract Friend Count: Further details on the extraction of the friend count feature.

• Extract Listed Count: Further details on the extraction of the listed count feature.

• Calculate Account Age: Further details on the calculation of the account age feature.

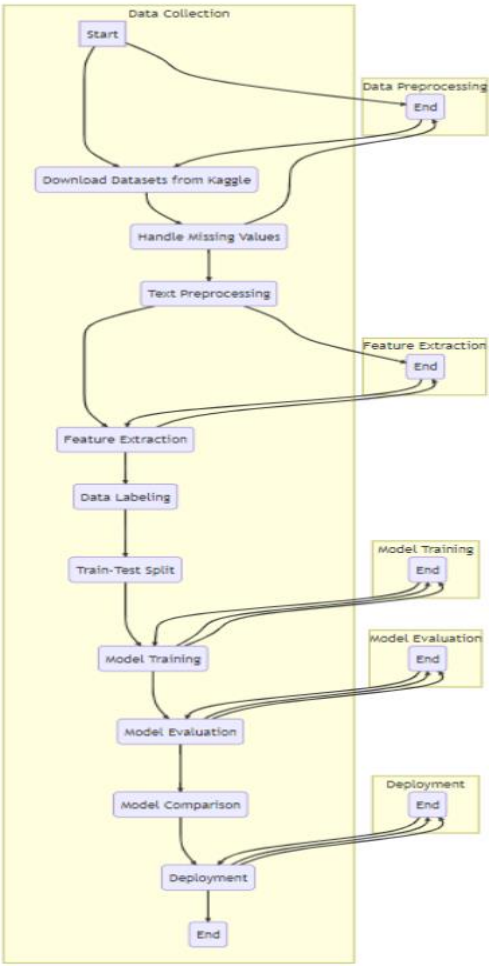
• Extract Verification Status: Further details on the extraction of the verification status feature.

• Calculate Word Frequencies: Further details on the calculation of word frequencies.

• Apply Topic Modelling: Further details on the application of topic modelling techniques.

C. Activity Flow Diagram

The activity flow diagram outlines the sequential steps involved in Twitter fake account detection. It begins with data collection from Kaggle, followed by preprocessing, feature extraction, and labelling. The dataset is split into training and testing sets for model training and evaluation. Performance metrics are calculated, and models are compared. Finally, the trained models can be integrated into a system or used independently for real-time fake account detection.



IV. SYSTEM REQUIREMENTS

This project is developed using Python as development tool. Python version 3.6 is used for this purpose. The project dataset is downloaded from Kaggle and kept in the working system. For development environment , 'Jupyter Notebook' is installed. Jupyter Notebook is a powerful scientific environment written in Python, for Python, and designed by and for Scientists, Engineers and Data Analysts. It offers a unique combination of the advanced editing, analysis, debugging and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection and beautiful visualization capabilities of a scientific package.

A. System Configuration:

I. Hardware Requirements:-

- System: Minimum Intel i3 2.4 GegaHertz
- RAM: Minimum 4GB RAM
- HARDDISK: Minimum 512GB HARDDISK

II. Software *Requirements:-*

- Operating System: Minimum Windows 7 or above
- Programming Language: Python
- Software: Anaconda

B. Python3:

Python is one of the most common language used for machine learning because it is easy to learn, enormous packages it supports like scikit learn that contains the implementation of common machine learning algorithms, built-in library supports, pre-processed models, support for larger networks and huge toolset.

C. Benefits of Pyhton:

- Scikit learn- It features various classification, regression and clustering algorithms including support vector machines, random forest, gradient boosting, K-means and DBSCAN, and is designed to interoperate with the python numerical and scientific libraries NumPy and SciPy.
- Data frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of datasets.
- Label based slicing, fancy indexing and sub setting of large datasets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on dataset.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.

D. Anaconda Software:

Anaconda is a Python distribution that includes installations and packages managements tools. It offers a wide selection of packages and commercial support [20]. Additionally, Anaconda is an environment manager that allows users to create different. Python environments, each without its own settings. This feature enables developers to work on multiple projects with different dependencies and configurations without conflicts [20]. Anaconda's package management tools provide access to over 7,500 open-source packages for data science, machine learning, and AI. It includes packages such as NumPy, pandas, and scikit-learn, making it an excellent tool for data science projects. Furthermore, Anaconda provides commercial support for its enterprise users, allowing businesses to use Anaconda with the assurance of dedicated support. Overall, Anaconda is a powerful tool for Python developers, especially for those working on data science and machine learning projects. Its package management tools and environment management features provide flexibility and ease of use for working on multiple projects. Anaconda can help with:

- Python can be installed over the multiple platforms
- Different environments can be supported separately
- Distributing with not having correct privileges
- Support and running with specific packages and libraries

V. SYSTEM IMPLEMENTATION AND FINDINGS

A. Importing the dataset & libraries & overviewing the dataset:

The project starts by importing of all the libraries which functions will be used for cleaning of the data, data analysis & visualisation and making our own classifier. Then dataset used for the project is been loaded after that dimensions and datatype of the dataset is been known with the help of pandas library and also statistical information of the dataset is known by writing code for it this was all done to have a complete overview of the dataset before doing the data cleaning. Below figure 1 is showing the datatype of the dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2797 entries, 0 to 2796
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      2797 non-null   float64
 1   id_str                  2797 non-null   object
 2   screen_name             2797 non-null   object
 3   location                1777 non-null   object
 4   description             2394 non-null   object
 5   url                     1455 non-null   object
 6   followers_count         2797 non-null   int64
 7   friends_count           2797 non-null   int64
 8   listed_count            2797 non-null   int64
 9   created_at              2797 non-null   object
 10  favourites_count        2797 non-null   int64
 11  verified                2797 non-null   bool
 12  statuses_count          2797 non-null   int64
 13  lang                    2797 non-null   object
 14  status                  2508 non-null   object
 15  default_profile         2797 non-null   bool
 16  default_profile_image   2797 non-null   bool
 17  has_extended_profile    2698 non-null   object
 18  name                    2797 non-null   object
 19  bot                     2797 non-null   int64
dtypes: bool(3), float64(1), int64(6), object(10)
memory usage: 379.8+ KB
```

Figure1: Datatypes of the dataset

B. Exploratory Data Analysis:

- Data Cleaning:

Data cleaning is the method of removing all the inaccuracies and inconsistencies from the dataset. So in this project the data cleaning process of data starts by firstly checking for missing values present within the data. This was done because if during analysis any missing value is found then visualisation of it will not be possible and also because if we trained your models with the inconsistent data then it will have poor performance and less accuracy. For checking the null values a heatmap was made which shows the null values of the data different attributes with different color. Then isnull().sum() function is used to know the null values if any present in the dataset. The picture of all null values present in dataset as per column name is given in the below figure1 and heatmap showing null values in different attributes of training data is given in the below figure2.

```
id                          0
id_str                      0
screen_name                 0
location                 1020
description               403
url                      1342
followers_count             0
friends_count               0
listed_count                0
created_at                  0
favourites_count            0
verified                    0
statuses_count              0
lang                        0
status                    289
default_profile             0
default_profile_image       0
has_extended_profile       99
name                        0
bot                         0
dtype: int64
```

figure1

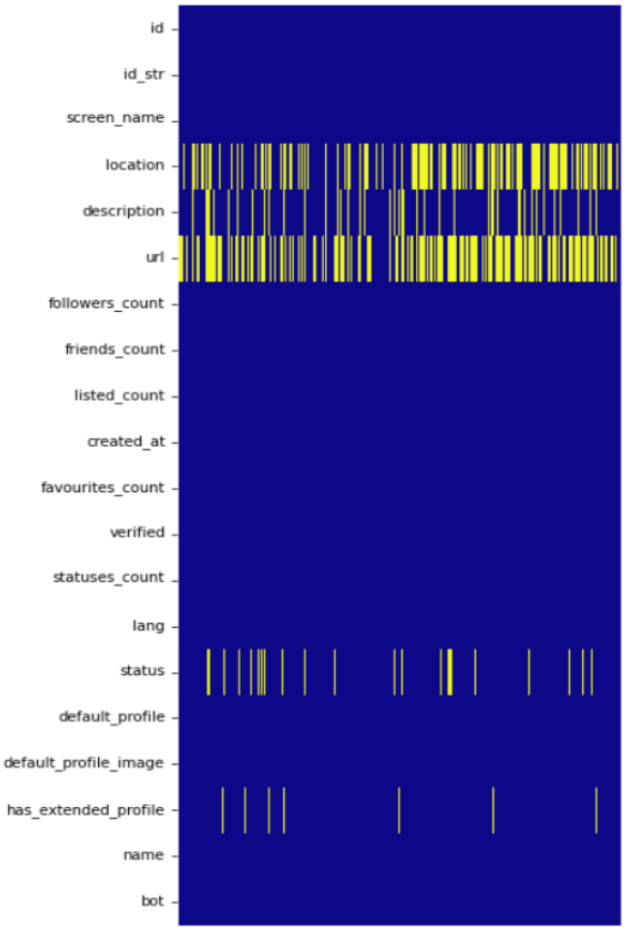


figure2

As it is clearly visible in the previous figure that the feature — 'description', 'profile_image_url', 'lang', 'location' and 'profile_background_image_url' has null values. We won't be using 'profile_background_image_url' , 'description', 'lang', 'profile_image_url' in our analysis so, we can directly convert the NaN to string directly. But for 'location' -> NaN values is changed to 'unknown'.

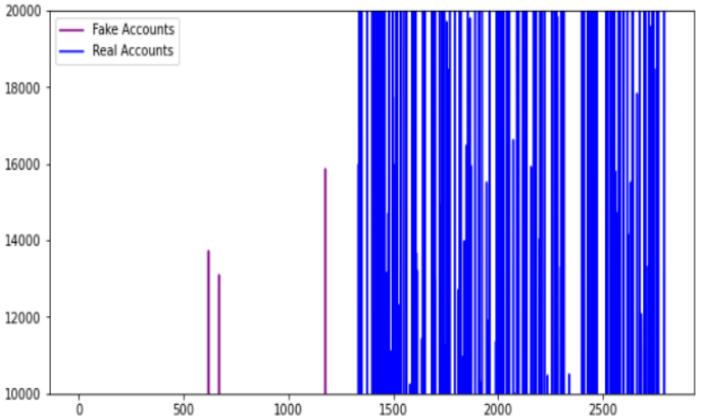
- Finding the general characteristics of real & fake accounts:

Moving on the project I found out the real account friends vs followers and fake/bot accounts friends vs followers by making the regplot that shows the same to depict the general trend on the characteristics of real and fake accounts that generally what is the ratio of friends to followers that real account have or fake account have, generally real or fake accounts have more followers than following or more following than followers etc. The regplots that was made is given below:

Above are two regplots for 'Bots friends vs Followers' and 'Non-Bots Friends vs followers', They depict the general trend on the characteristics of fake accounts or bot is that they possess more followers count compared to friends. On the other hand, Non-Bots or real accounts have generally an equal number of both friends and followers with some slight variance.

- Identifying the Imbalance in the Data:

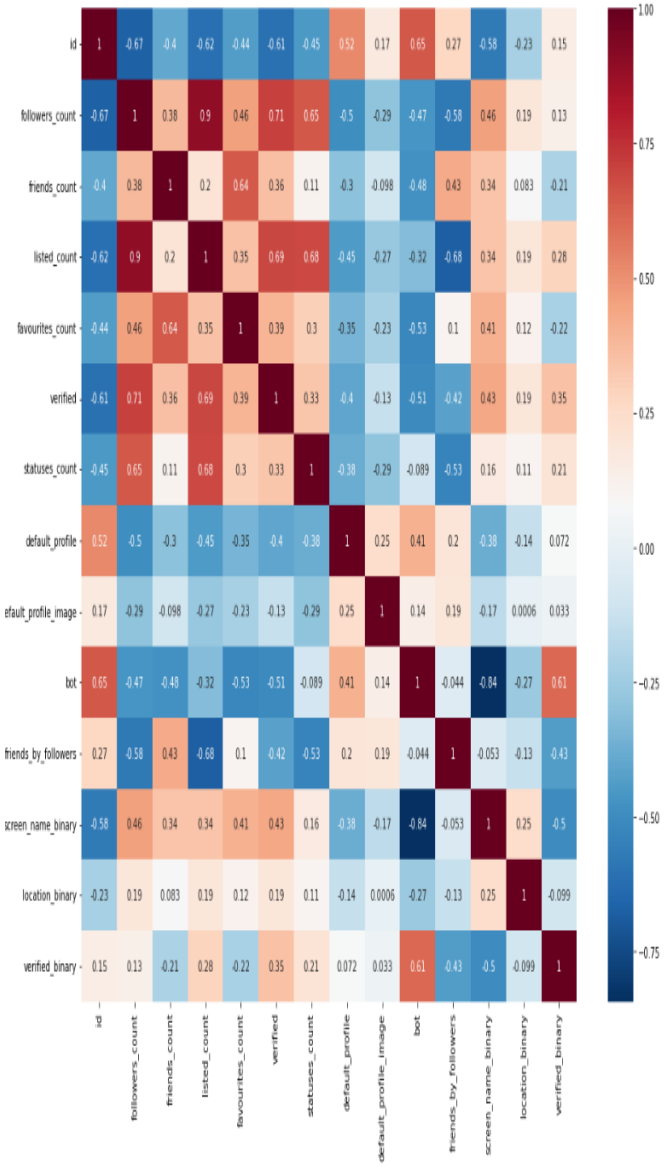
After this the next step is to find out the imbalances in the data. In this I found out that whenever the listed_count is between 10000 to 20000 there is approximately 5% of bot/fake account and approximately 95% of real accounts. This is also shown in the below figure:



- Feature Independence using Spearman Correlation:

Then I identified the feature independence of data using Spearman correlation. Spearman's correlation coefficient is a statistical measure of the strength of a monotonic relationship between paired data. Ina sample, it is denoted by 'rs' and is by design constrained as -1<=rs<=1 and its interpretation is similar to that of Pearsons, e.g. the closer is to the stronger the monotonic relationship. Correlation is an effect size and so we can verbally describe the strength of the correlation using the following guide for the absolute value of rs :

A. 0 to 0.19 - "very weak"
B. 0.2 to 0.39 - "weak"
C. 0.4 to 0.59 - "moderate"
D. 0.6 to 0.79 - "strong"
E. 0.8 to 1 - "very strong"



The findings from Spearman Correlation results are:

A. There is no correlation between id, statuses_count, default_profile, default_profile_image and the target variable.
B. There is a strong correlation between verified, listed_count, friends_count, followers_count and the target variable.
C. We cannot perform correlation for categorical attributes. So we will take screen_name, name, description, status into feature engineering. While use verified, listed_count for feature extraction.

- Feature Engineering:

The next step is to do the feature engineering for this we will take name, screen_name, status, description while use the listed_count, verified for feature extraction [21]. For doing feature qengineering a wbag of world model is also created which identifies whether an account on twitter is bot or not. In feature engineering process we have to convert name, screen_name, status, description into a binary using our own vectorizer algorithm [21]. Then some feature extraction is done where we have taken listed_count_binary 1 and made it false where ever the original listed_count1 > 20000s and we created a new feature which consists of values of, 'name_binary', 'screen_name_binary', 'description_binary', 'status_binary', 'verified', 'friends_count', 'followers_count', 'listed_count_binary', 'statuses_count' and 'bot' [21].

C. Implementing Different Models:

We implement different machine learning model and hence find their accuracy on test and training set. And we all find out testing and training precision, confusion matrix, F1-score and recall of the different models. We will also plot the ROC curve which is a graphically plot created by plotting the true positive rate against the false positive rate at the various threshold which depicts the performance of all the models used for the study [22].

- Decisions Trees Classifier:

The decisions trees is a well-liked and effective method of prediction and categorization. Similar to a flowchart, it is organized with each leaf or terminal node containing the name of a class, each internal node indicating a test on an attribute, and each branch designating the test's outcome. The Decision Trees Classifier Accuracy, F1-score, precision, recall and ROC curve:

Decision Tree Classifier Train Confusion Matrix:

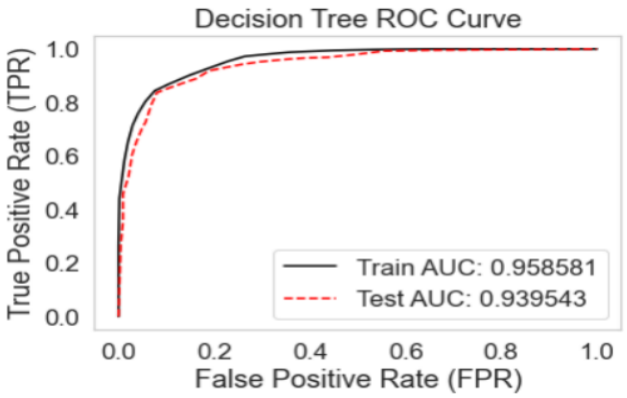
[[973 80]
[141 763]]

Decision Tree Classifier Test Confusion Matrix:

[[389 34]
[68 349]]

Train precision of the Decision Tree classifier: **0.90510**
Test precision of the Decision Tree classifier: **0.91123**
Train recall of the Decision Tree classifier: **0.84403**
Test recall of the Decision Tree classifier: **0.83693**
Train F1-score of the Decision Tree classifier: **0.87350**
Test F1-score of the Decision Tree classifier: **0.87250**
Training Accuracy of the Decision Tree classifier: **0.88707**
Test Accuracy of the Decision Tree classifier: **0.87857**

Decisions Trees classifier ROC curve is shown in the below figure:



Decision Tree Classifier ROC curve

- Multinomial Naïve Bayes Classifiers:

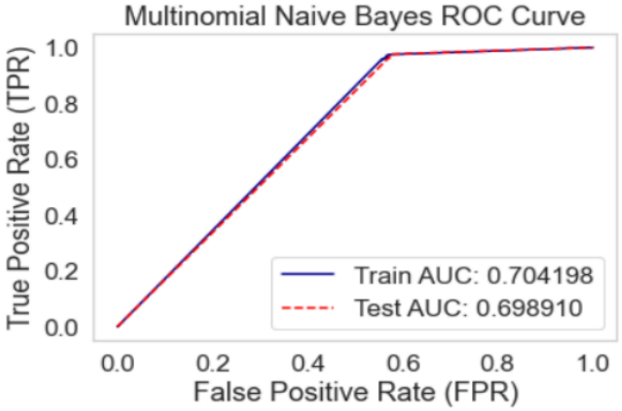
Naive Bayes is a powerful machine learning algorithm that has proven to be both simple and effective, despite the recent advances in the field. This approach has been successfully applied in numerous applications and is especially helpful for natural language processing (NLP) jobs. A class of probabilistic models known as naive bayes uses Bayes' theorem and probability theory to predict the category of a given sample, such as a customer review or a news item. Naive Bayes determines which category has the highest probability for a given sample by computing the probabilities for each category. By using Bayes' theorem, which calculates the likelihood of a feature based on knowledge of potential confounding factors, this algorithm determines these probabilities. The Multinomial Naïve Bayes Classifier Accuracy, F1-score, precision, recall and ROC curve:

Multinomial Naïve Bayes Classifier Train Confusion Matrix:
[[458 595]
[32 872]]

Multinomial Naïve Bayes Classifier Test Confusion Matrix:
[[181 242]
[12 405]]

Multinomial Naïve Bayes classifier Train precision: **0.59441**
Multinomial Naïve Bayes classifier Test precision: **0.62597**
Multinomial Naïve Bayes classifier Train recall: **0.96460**
Multinomial Naïve Bayes classifier Test recall: **0.97122**
Multinomial Naïve Bayes classifier Train F1-score: **0.73555**
Multinomial Naïve Bayes classifier Test F1-score: **0.76128**
Multinomial Naïve Bayes classifier Training Accuracy: **0.67961**
Multinomial Naïve Bayes classifier Test Accuracy: **0.69762**

Multinomial Naïve Bayes classifier ROC curve is shown in the below figure:



Multinomial Naïve Bayes Classifier ROC curve

- Random Forest Classifier:

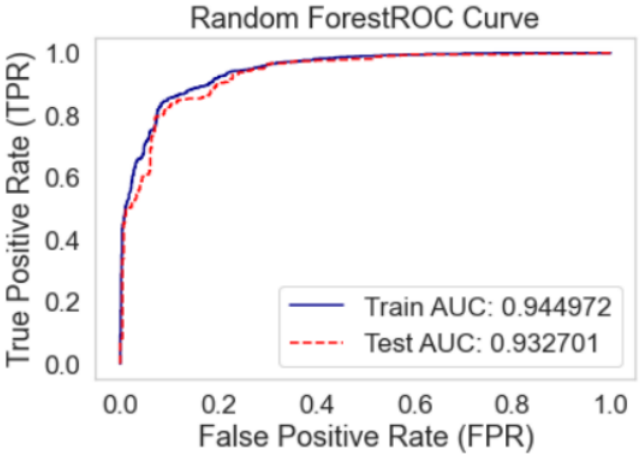
Random Forest is an extremely flexible and user-friendly machine learning method that frequently produces good results with only a small amount of hyperparameter modification. Its simplicity and suitability for both classifications and regressions problems make it one of the most extensively used algorithms. The Random Forest Classifier Accuracy, F1-score, precision, recall and ROC curve:

Random Forest Classifier Train Confusion Matrix:
[[965 88]
[148 756]]

Random Forest Classifier Test Confusion Matrix:
[[382 41]
[74 343]]

Train recall of the Random Forest classifier: **0.83628**
Test recall of the Random Forest classifier: **0.82254**
Train precision of the Random Forest classifier: **0.89573**
Test precision of the Random Forest classifier: **0.89323**
Train F1-score of the Random Forest classifier: **0.86499**
Test F1-score of the Random Forest classifier: **0.85643**
Training Accuracy of the Random Forest classifier: **0.87941**
Test Accuracy of the Random Forest classifier: **0.86310**

Random Forest classifier ROC curve is shown in the below figure:



Random Forest Classifier ROC curve

- Our Proposed Algorithm Classifier:

Our Proposed Algorithm implementation steps are as follows:
Step 1: Creating copy of dataframe in train_set
Step 2: Converting id to int
Step 3: Replacing Null values with 0 in Friends_count column
Step 4: Replacing Null values with 0 in Followers_count column
Step 5: Preparing bag of words for bot accounts
Step 6: Converting verified account into vectors (True->1 False->0)
Step 7: If the name,description,screen_name,status columns contains bot , then Store the data set in predicted_set_1
7.1: Assign bot column as 1 (BOT)
7.2: Store the rest data set in verified_set for next step
Step 8: For all verified account, Store the data set in predicted_set_2

8.1:Assign bot column as 0 (NON_BOT)

8.2:Store the rest data set in followers_following_set for the next step

8.3:predicted_set_1=:concatenate(predicted_set_1,predicted_set_2)

Step 9: If followers_count is less than 50 and statuses_count greater than 1000 , then then Store the data set in predicted_set_2

9.1: Assign bot column as 1 (BOT)

9.2: Store the rest data set in followers_retweet_set for next step

9.3:predicted_set_1=:concatenate(predicted_set_1,predicted_set_2)

Step 10: If followers_count is less than 150 and statuses_count greater than 10000 then then Store the data set in predicted_set_2

10.1:Assign bot column as 1 (BOT)

10.2:predicted_set_1=:concatenate(predicted_set_1,predicted_set_2)

Step 11: Store the rest data set in predicted_set_2 and assign bot column as 0 (NON_BOT)

11.1:predicted_set_1=:concatenate(predicted_set_1,predicted_set_2)

Step 12: Return predicted_set_1

Our Proposed Algorithm based Classifier Accuracy, F1-score, precision, recall and ROC curve:

Our Own Train Confusion Matrix:

[[1083 0]
[66 905]]

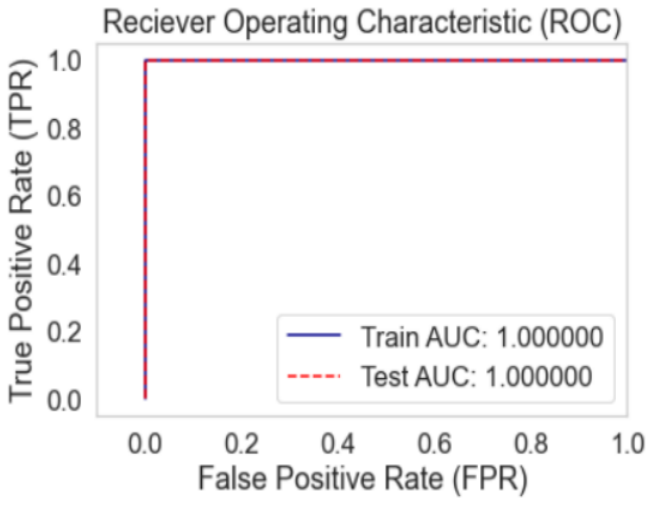
Our Own Test Confusion Matrix:

[[393 0]
[31 319]]

Our Own Train precision: **1.00000**
Our Own Test precision: **1.00000**
Our Own Train recall: **0.92020**
Our Own Test recall: **0.94562**
Our Own Train F1-score: **0.95844**
Our Own Test F1-score: **0.97205**

Our Own Train Accuracy: **0.9638267491670633**
Our Own Test Accuracy: **0.9775596072931276**

Our Own Classifier ROC curve is shown in the below figure:

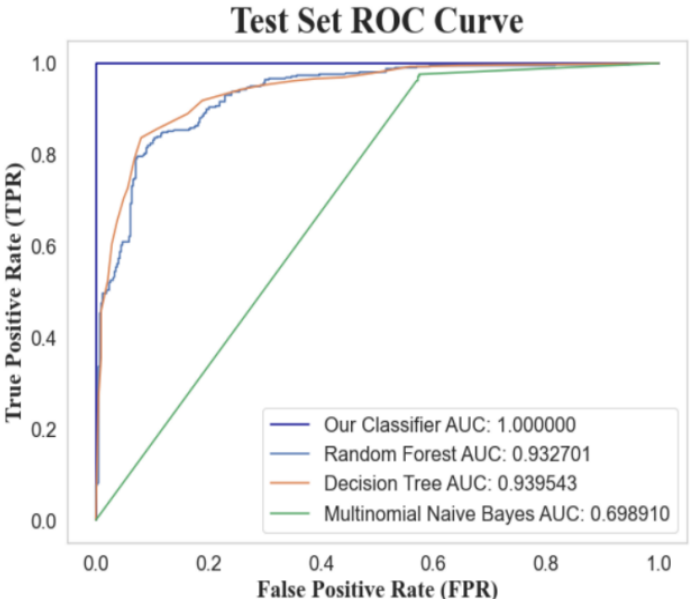


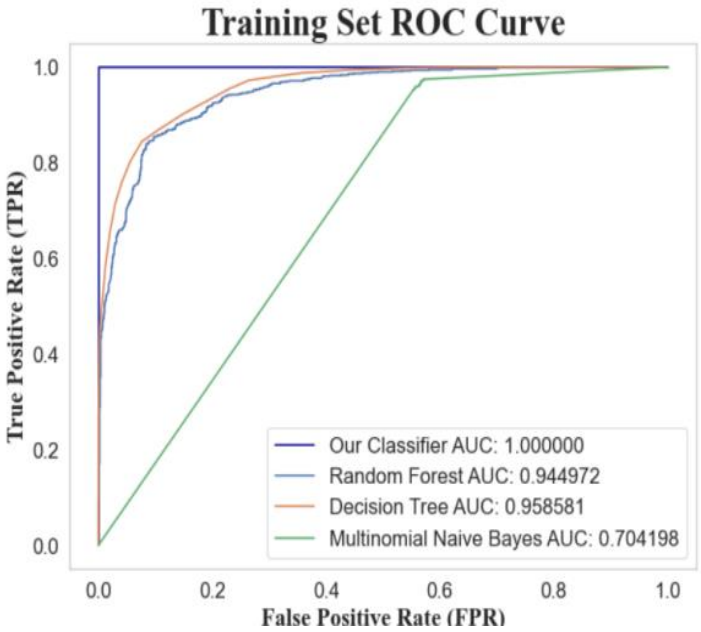
Our Own Classifier ROC curve

D. Comparing all the Models:

- ROC Curve of all the models combined:

Here we made a combined the ROC curve of all the models to find out which model AUC is best among all. Below figures shows the combined Training set ROC Curve and Test ROC Curve of all the models.

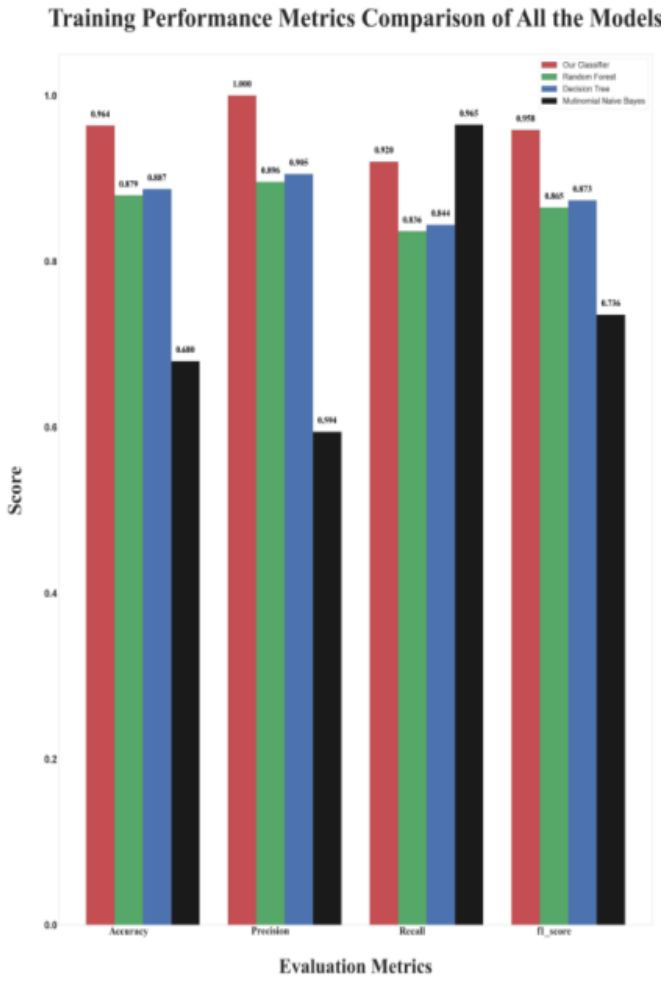


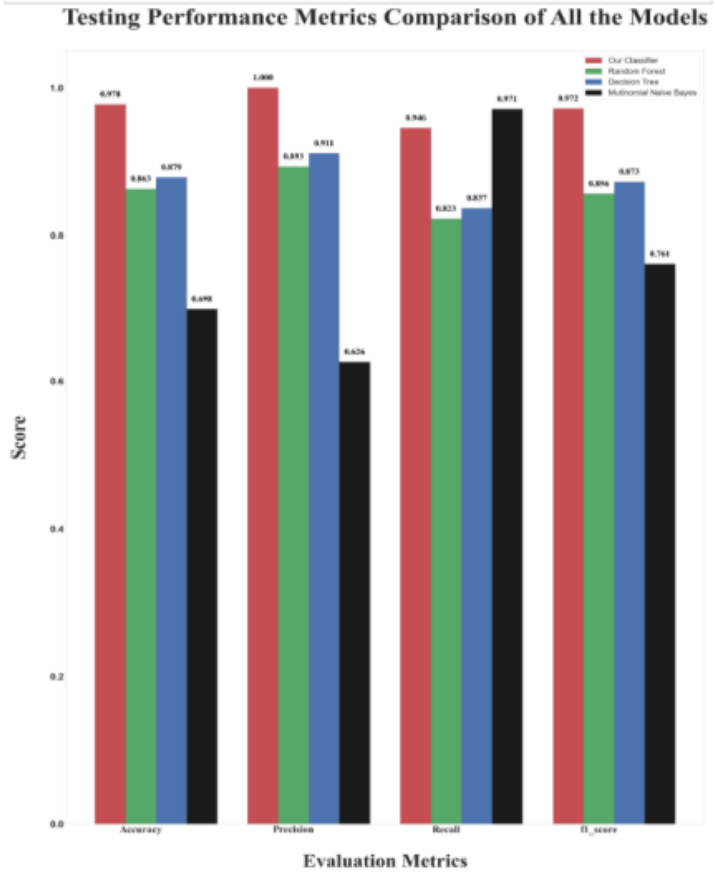


The above figures shows that the our classifier have the best training and test AUC as compared to all the models.

- Comparison Charts:

After checking out which model has the best AUC we will made a comparison charts of all the models comparing all the models accuracy, F1-score, recall and precision to find out which model has performed well in which areas. Below figures shows the training set and testing set comparison charts between all the four models.





The red color in the comparison charts represents our proposed algorithm based classifier, the green color represents random forest classifier, the blue color represents the decision tree classifier and finally the black color represents the multinomial naïve bayes classifier. The comparison charts shows us that our classifier has performed the best when it comes to accuracy, precision and F1-score in both training and testing data and Multinomial Naïve Bayes has performed the best in recall in both training and testing data whereas our classifier was the second best in recall. Overall our classifier has out performed the other three classifiers this comparison chart also shows that so our main motive to create a classifier that perform better than traditional machine learning classifiers to detect fake accounts in Twitter dataset is achieved.

VI. RESULTS

The final output from our classifier predicting fake/bot accounts from the data that was given to our classifier for making the predictions was stored in submission.csv and it will be of the form as shown below figure. In the below figure the class label ‘bot’ represent whether a given users is fake or a real’s users. The entry 1 represent a fake user i.e., bot and 0 represent a real user i.e., non-bot.

	A	B
1	id	bot
2	1.32E+09	1
3	2.56E+09	1
4	3.48E+08	1
5	8.56E+08	1
6	8.33E+17	1
7	7.14E+17	1
8	2.31E+09	1
9	7.43E+17	1
10	85430866	1
11	8.28E+17	1
12	2.39E+08	1
13	5.38E+08	1
14	2.57E+09	1
15	3E+09	1
16	3.41E+09	1
17	7.02E+08	1
18	3.29E+08	1
19	1.26E+09	1
20	8.07E+17	1
21	3.04E+09	1
22	2.58E+09	1
23	8.46E+17	1
24	8.54E+08	1
25	7.19E+17	1
26	2.42E+09	1
27	1.39E+09	1

submission

In project implementation we have made comparison charts between our classifier and Decisions Trees, Random Forest & Multinomial Naïve Bayes Classifiers which showed us that our classifier have a best training and tests accuracy, precision, F1-scores as compared to the other models and it also showed that our classifier have the second best training and test recall as compared to the other classifiers and we also made a combined ROC curve of all the models which showed that our model has the best train and test AUC as compared to all the other classifiers. So all this clearly shows that our model predict the fake/bot accounts better than all the other models.

VII. CONCLUSION AND FUTURE SCOPE

Twitter fake account detection is a program used to predict whether a Twitter account is fake or real by knowing few basic information about the Twitter account. In this project, we used various Machines Learnings techniques to predicts weathers an account on Twitter is a fakes or a reals user. We a have performed significant amount s of feature engineering, along with feature extractions & then selected the best features out of 20 given to us in the dataset which helped us to identify whether an account is fake/bot or real/non-bot. We implemented various algorithm and finally implemented our own which gave us AUC>95%. Our framework will be able to identify whether a twitter user is a bot s or a human. We can extend our work to other social media platform like Facebooks. Our work will safeguard oneself and an organization from false information, malicious content and ensure their brand value.

REFERENCES

[1] "Top Trending Twitter Topics for 2011 from What the Trend," http://blog.hootsuite.com/top-twittertrends-2011/, Dec. 2011.

[2] "Twitter Blog: Your World, More Connected," http://blog. twitter.com/2011/08/your-world-moreconnected.html, Aug. 2011.

[3] Alexa, "The Top 500 Sites on the Web by Alexa," http:// www.alexa.com/topsites, Dec. 2011.

[4] "Amazon Comes to Twitter ,"http://www.readwriteweb.com/amazon_comes_to_twitter.

[5] "Best Buy Goes All Twitter Crazy with @Twelpforce,"http://twitter.com/in_social_media/

[6] "Barack Obama Uses Twitter in 2008 Presidential Campaign,"http://twitter.com/Obama/, Dec. 2009.

[7] J. Sutton, L. Palen, and I. Shlovski, "Back-Channels on the Front Lines: Emerging Use of Social Media in the 2007 Southern California Wildfires," Proc. Int'l ISCRAM Conf., May 2008.

[8] A.L. Hughes and L. Palen, "Twitter Adoption and Use in Mass Convergence and Emergency Events," Proc. Sixth Int'l ISCRAM Conf., May 2009.

[9] S. Gianvecchio, M. Xie, Z. Wu, and H. Wang, "Measurement and Classification of Humans and Bots in Internet Chat," Proc. 17th USENIX Security Symp., 2008.

[10] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your Botnet Is My Botnet: Analysis of a Botnet Takeover," Proc. 16th ACM Conf. Computer and Comm. Security, 2009.

[11] S. Gianvecchio, Z. Wu, M. Xie, and H. Wang, "Battle of Botcraft: Fighting Bots in Online Games with Human Observational Proofs," Proc. 16th ACM Conf. Computer and Comm. Security, 2009.

[12] A. Java, X. Song, T. Finin, and B. Tseng, "Why We Twitter: Understanding Microblogging Usage and Communities," Proc. Ninth WebKDD and First SNA-KDD Workshop Web Mining and Social Network Analysis, 2007.

[13] B. Krishnamurthy, P. Gill, and M. Arlitt, "A Few Chirps about Twitter," Proc. First Workshop Online Social Networks, 2008.

[14] S. Yardi, D. Romero, G. Schoenebeck, and D. Boyd, "Detecting Spam in a Twitter Network," First Monday, vol. 15, no. 1, Jan. 2010.

[15] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks," Proc. Seventh ACM SIGCOMM Conf. Internet Measurement, 2007.

[16] S. Wu, J.M. Hofman, W.A. Mason, and D.J. Watts, "Who Says What to Whom on Twitter," Proc. 20th Int'l Conf. World Wide Web, pp. 705-714, 2011.

[17] H. Kwak, C. Lee, H. Park, and S. Moon, "What Is Twitter, a Social Network or a News Media?" Proc. 19th Int'l Conf. World Wide Web, pp. 591-600, 2010.

[18] I.-C.M. Dongwoo Kim, Y. Jo, and A. Oh, "Analysis of Twitter Lists as a Potential Source for Discovering Latent Characteristics of Users," Proc. CHI Workshop Microblogging: What and How Can We Learn From It?, 2010.

[19] D. Zhao and M.B. Rosson, "How and Why People Twitter: The Role Micro-Blogging Plays in Informal Communication at Work," Proc. ACM Int'l Conf. Supporting Group Work, 2009.

[20] https://www.anaconda.com/

[21] https://iopscience.iop.org/article/10.1088/1742-6596/1950/1/012006