Twitter Bot Detection

………………………………………………

Prisha Sharma (05901172021) Radhika Bansal (01601172021)

Tejaswi Tyagi (07401172021) Anjali (07301172021)

Karishma Jain (06501172021) Vashitva Bagga (13701032021)

………………………………………………

***Abstract* —** *The main objective of this project is to build a Twitter bot detection model which is capable of identifying malicious Twitter bots. These Twitter bots are always a major concern for social media companies and pose a threat to security. Our aim is to use various classifiers in order to determine which is the best-suited approach for solving our problem statement.*

***Keywords — machine-learning, Data Analysis, Twitter bots, Pandas, Numpy, Feature extraction, Decision Tree Classifier, Multinomial Naive Bayes, Random Forest Classifier.***

# **Introduction**

Today, social media platforms like Twitter have many accounts that are controlled by automated agents called bot or Sybil accounts. Mostly, people aim to have more visitors to their websites, influence the community on a specific topic, recruit people to their organizations that might be an illegal organization, manipulate people for stock market actions, propagate fake news, and blackmail people to spread their private information by the power of these accounts. As a result, the social bot detection framework becomes very crucial to keep people safe from Sybil's accounts.

When these bot accounts are analyzed, it can be seen that there are various types; some of them are very primitive and some of them are very complex that they are hard to diagnose even by humans. In order to avoid detection, they mimic human accounts, develop strategies to a friend or follow human accounts and support each other as a large network to gather trust. Additionally, a large group or network of these accounts can act collaboratively to change trending topics on Twitter for malicious purposes.

The sheer number of these bot accounts and their increasing complexity bestows a challenge for the manual detection of these accounts.

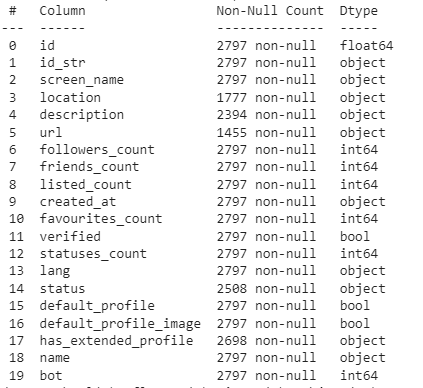
Our aim through this Twitter Bot Detection Project is to find the classifier which fits the best and has the greater accuracy in finding the solution to our problem statement which is - detecting malicious Twitter bots.

# **LIBRARIES USED**

1. Pandas - for data manipulations and analysis
2. Numpy - data analysis done through arrays
3. Matplotlib - plotting visualizations for various classifiers and while data analysis
4. Seaborn - It is used for data visualization and exploratory data analysis. Seaborn works easily with data frames and the Pandas library. The graphs created can also be customized easily.
5. Warning - to display the warning messages
6. Decision Tree Classifier - The algorithm uses training data to create rules that can be represented by a tree structure.
7. train\_test\_split - splits arrays or matrices into random subsets for train and test data
8. sklearn.naive\_bayes - naive bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the “naive” assumption of conditional independence
9. sklearn.ensemble - includes two averaging algorithms based on randomized decision trees: the RandomForest algorithm and the Extra-Trees method.
10. RandomForestClassifier - random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

# **Dataset USED**

The dataset we are working on basically contains the important attributes of a Twitter account that can be useful in determining whether a Twitter account is a bot or not. The basic structure of the dataset attributes includes 3 boolean values attributes, 1 float valued attribute, 6 integer type, and 10 object type attributes. These 20 attributes together form the skeleton of our training dataset.



*The above image shows the attributes of the dataset along with the data type*

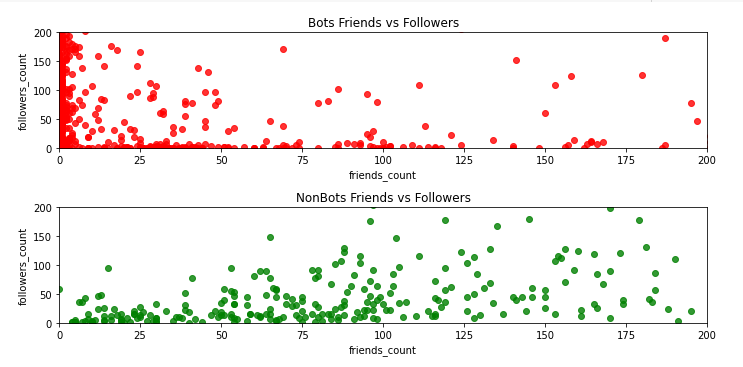
# **Data Analysis**

**IV.I Identifying the missing values**

We began our data analysis by finding the missing values in our dataset, to achieve this we prepared a heatmap that visually represents the NaN fields of the dataset. The results show that attributes - location, description, and URL have the maximum missing values followed by status and has\_extended\_profile attributes. All these attributes have an object type of datatype.

**IV.II Comparison between followers and following counts of bots and non-bots**

The primary filtering criteria for our training dataset will be based on the results after drawing a comparison between the followers and following counts of bots and non-bot accounts. Ideally, a bot account will have more “following” than “followers” and vis-versa would be true for the non-bot accounts. To understand this trend more clearly we plotted the graph with the help of matplotlib.

****

*The above plots show the comparison plots of followers\_count vs friends\_count of bots(red) and non-bots(green)*

**IV.III Further conditions we are using to identify a bot and a non-bot**

After plotting the above graphs, we further started building new conditions to classify the non-bots and bots. If the account is not verified i.e if the verified field of the dataset is false then we can clearly classify that account into the bot category. Further, if the screen name contains the word “bot” it then is also a potential bot account. Another attribute that can be of some help can be the location attribute which is generally null for the bots.

# **V. Feature selection And the correlation**

# Often when we get a dataset, we might find a plethora of features in the dataset. All of the features we find in the dataset might not be useful in building a machine learning model to make the necessary prediction. Using some of the features might even make the predictions worse. So, feature selection plays a huge role in building a machine learning model.

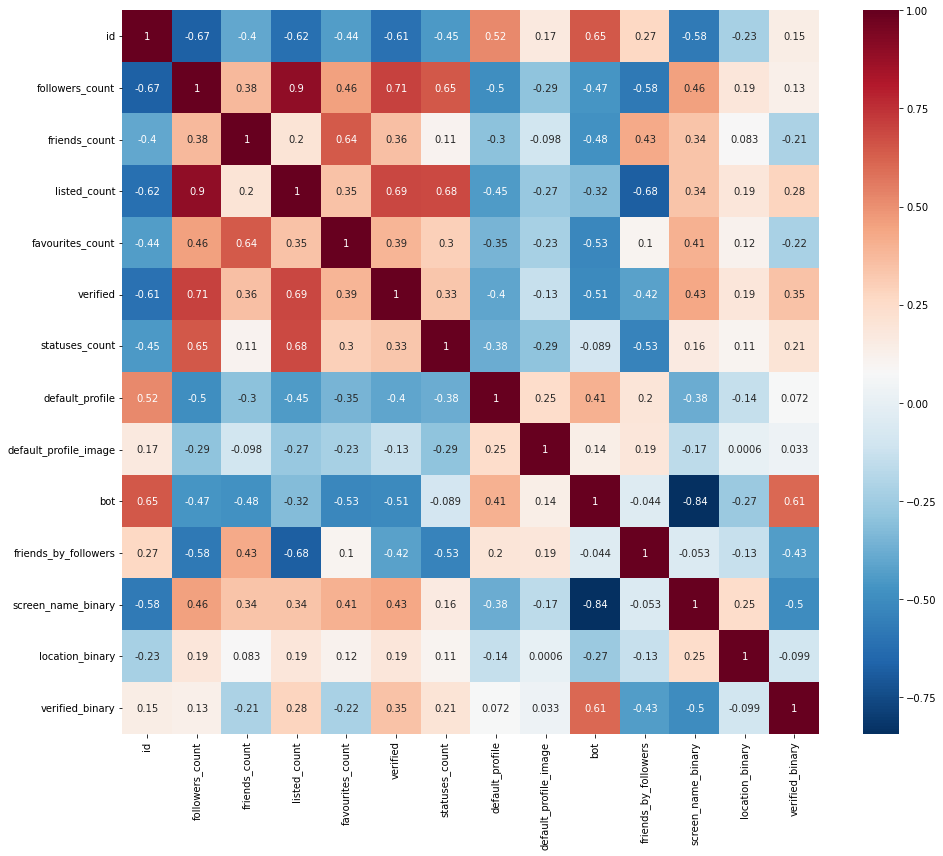
Correlation is a statistical term which in common usage refers to how close two variables are to having a linear relationship with each other. Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable. So, when two features have a high correlation, we can drop one of the two features

We have used the spearman correlation test here because spearman works on both ordinal and continuous data unlike Pearson by using df.corr(method='spearman').

Here the range of the spearman coefficient is -1 to 1.

For negative correlation value is between -1 to 0 and for positive correlation value is between 0 to 1.

**VI. Heatmap:**



*The heatmap created describes the relation between two attributes*

**VII. ResuLT**

There is no correlation between the id, statuses\_count, default\_profile, default\_profile\_image, and the target variable.

There is a strong correlation between verified, listed\_count, friends\_count, followers\_count, and the target variable.

We cannot perform correlation for categorical attributes. So we will take screen\_name, name, description, and status into feature engineering. While using verified, listed\_count for feature extraction.

**VIII. Feature Engineering**

Feature engineering is a machine learning technique that leverages data to create new variables that aren’t in the training set. It can produce new features for both supervised and unsupervised learning, with the goal of simplifying and speeding up data transformations while also enhancing model accuracy.

In this project, we have created a bag of words that are often associated with bot accounts we are searching for these words in the screen name, name, description, and status of the accounts

**IX. Feature extraction**

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data*.* After doing the necessary feature extraction, we created a list of features to be used in the classifiers ahead - ['screen\_name\_binary', 'name\_binary', 'description\_binary', 'status\_binary', 'verified', 'followers\_count', 'friends\_count', 'statuses\_count', 'listed\_count\_binary', 'bot'].

**X. Receiver Operating Characteristic Curve (ROC Curve)**

The receiver operating characteristic curve is a graph used to evaluate the performance of a classification model at all classification thresholds. This curve plots two parameters namely, true positive rate and false positive rate. The False Positive Rate (FPR) is measured by taking the ratio between False Positives and the total number of negative samples. The True Positive Rate (TPR) is measured by taking the ratio between True Positives and the total number of positive examples. The area under the ROC Curve is the measure of the ability of the classifier to differentiate between the classes.

**XI.**  **Decision Tree Classifier**

Decision Treesare a supervised learning method used for [classification](https://scikit-learn.org/stable/modules/tree.html#tree-classification). The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.DecisionTreesare simple to understand and interpret. Trees are easily visualized and understandable.

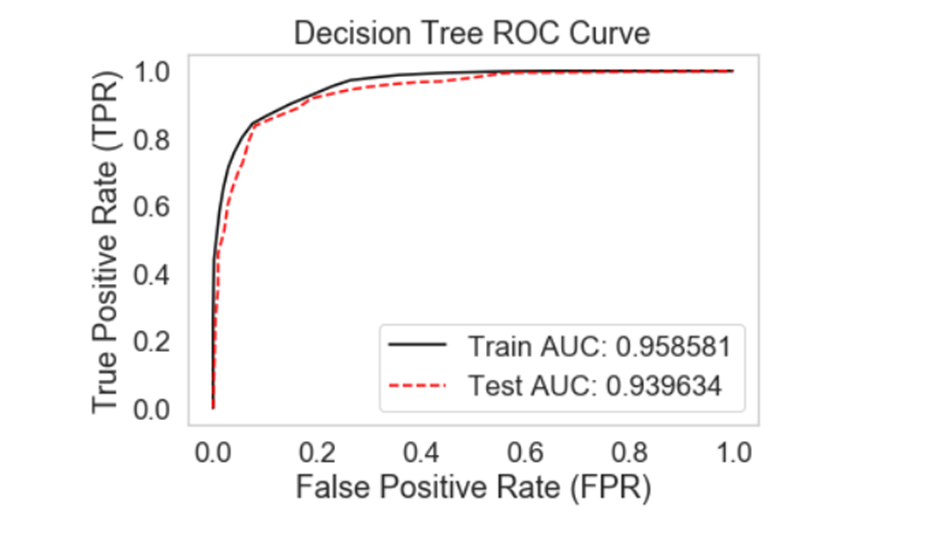
A decision tree is taken because it is one of the intelligent and readable splitting classifiers, as it can take multiple dimensions into account, it does not require any domain knowledge prerequisite to it. With multi-branching, it reduces noise and improves the overall accuracy.

Following are the steps that we need to follow to build a very good Decision Tree:-

The target attribute/feature from the given attributes is selected, information gain of the selected target attribute is calculated and the entropy for all the Categorical variables is calculated.

The attribute having maximum gain value will be our Root node. and recursively make new decision trees using the subsets of the dataset created till the point where we don't have any features to split upon.

.

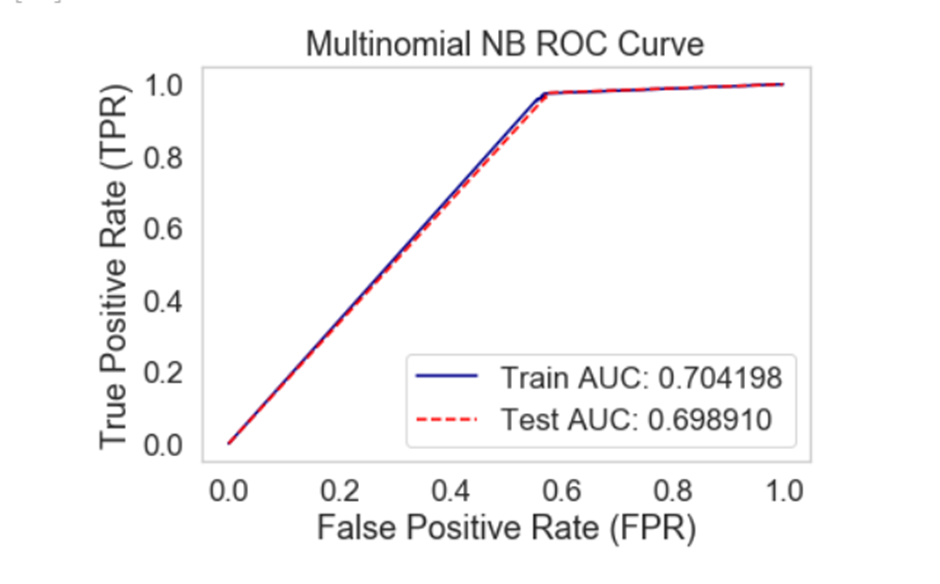


**Test Accuracy: 0.87857 and Training Accuracy: 0.88707**

*Decision Tree gives very good performance and generalizes well. But it may be overfitting which yields inaccurate and poor results.*

#### **XII.Multinomial Naive Bayes Classifier**

* The Naïve Bayes algorithm is a supervised learning algorithm, which is based on the Bayes theorem and is used for solving classification problems. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
* The Naive Bayes classifier works on the principle of conditional probability.



*Multinomial Naive Bayes performs poorly and is not a good choice as the Train* ***AUC is just 0.556 and the Test is 0.555.***

Naïve Bayes doesn’t take into account the special characters when using to find the probability, improper training of a particular class makes it difficult to predict the result. In any category of train-test data split Naïve Bayes should be avoided as overfitting can be very persistent which yields inaccurate and poor results.

Multinomial Naive Bayes performs poorly and its yield is quite low in comparison to decision trees and random forests as the Train is 0.556 and Test is 0.555.

**XIII. Random forest**

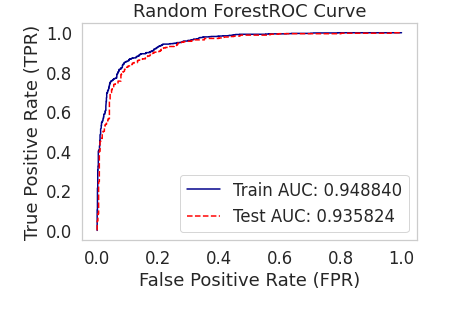
A random forest is a supervised machine learning algorithm that is constructed from decision tree algorithms.

This algorithm is applied in various industries such as banking and e-commerce to predict behavior and outcomes.

Random forest is applied by:

* Randomly select “k” features in given m features
* Among “k” features calculate node d using the best spilt point
* Split the node into daughter nodes using the best-split point. The process is repeated till the “l” number of nodes has reached
* Forest is built and use each tree on the test feature and store outcome
* Calculate votes for each predicted outcome and the highest voted outcome as the final prediction.
* Consider the highest voted outcome as the final prediction

*The random forest algorithm was found to be the best learning model with a* ***test accuracy of 0.87839 and test accuracy of 0.85238.***

****

**XIV. CONCLUSION AND FUTURE SCOPE**

Twitter is one of the most popular social networking sites, it is the fastest means of conveying information. It highly influences people’s perspectives So it is necessary that tweets are sent by genuine users and not by Twitter bots. The project proposes an approach to detect Twitter bots using machine learning algorithms namely Multinomial naive Bayes classifier, Decision Trees, and Random forest. The project began with data analysis to determine the current status of the dataset. The next step was the process of model engineering, which involves the steps of feature selection and optimization. Decision Tree was found to have the best train and test accuracy among all three machine learning algorithms. Hence Decision Tree algorithm was applied to real-time data and the Twitter bots were successfully identified. In the future, we can develop our own classifier to detect bots with even better train and test accuracy.