Social Media Bot Detection using Text Analysis and ML in Python

“Machine Learning for Twitter Bot Detection”

Vasundhara Saxena

12013828

School of Computer Science and Engineering

DECLARATION

I, Vasundhara Saxena (Student ID: 12013828), hereby declare that the report titled "Social Bot Detection Using Text Analysis and Machine Learning" represents my original work and research conducted under the guidance of Dr. Ajay Sharma at Lovely Professional University. The research and content presented in this report have been produced in accordance with the ethical standards, academic guidelines, and policies of my educational institution.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the individuals and organizations who have played a pivotal role in the successful completion of the report on "Social Bot Detection Using Text Analysis and Machine Learning." This endeavour would not have been possible without their valuable support, guidance, and contributions.

First and foremost, I extend my heartfelt thanks to my supervisor, Dr. Ajay Sharma for their unwavering support, mentorship, and expertise throughout this research project. Their guidance has been instrumental in shaping the direction of this work and ensuring its academic rigor.

I would also like to acknowledge the contributions of my fellow researchers and colleagues who provided invaluable insights, suggestions, and feedback, which greatly enriched the quality of this report.

I am indebted to the research participants who generously shared their data and experiences, making this study possible. Your cooperation and willingness to be part of this research are deeply appreciated.

I extend my appreciation to the academic and technical staff at Lovely Professional University for providing the necessary resources and facilities for conducting this research. I would like to thank my family and friends for their unwavering support, understanding, and encouragement throughout this research journey.

This report is a result of collective effort, and I am humbled by the support and encouragement I have received from all those who have contributed to this work.

Sincerely,

Vasundhara Saxena

12013828

OBJECTIVE:

The purpose and range of text analysis and machine learning-based social bot detection include the objectives, strategies, and constraints of a study or real-world initiative that aims to discover and counteract social bots on online social networks. Social bots are accounts that are automated or semi-automated and imitate human behaviour, frequently with the intention of harming or misleading users. An outline of the goals and reach of the project is as follows:

Goal: Developing efficient techniques and instruments for the detection and mitigation of social bots in online social networks is the main goal of social bot detection utilising text analysis and machine learning. This can be divided into a number of more focused objectives:

Mitigation: Create plans to lessen the effects of social bots, like content screening, account suspensions, or enhanced platform security.

Data analysis: To learn more about the tactics and goals of social bots, examine their actions, patterns, and traits.

Project scope: A project centred on text analysis and machine learning for social bot detection would normally include the following elements:

Data collection: Compile information from user profiles, postings, comments, and interactions on social media networks.

Preprocessing: Clean up and prepare the data, doing things like feature extraction, noise reduction, and text normalisation.

Feature engineering is the process of removing pertinent features—such as sentiment, lexical, and semantic features—from text data.

Text analysis: Look for trends, abnormalities, and traits linked to social bots in the content of user-generated text.

ABSTRACT:

The widespread use of social media has led to the rise of social bots, which are automated accounts that replicate human behaviour, frequently with malevolent intentions. Ensuring the integrity of online communities requires the detection and fighting of these deceitful bots. In this work, we integrate machine learning and text analysis techniques to present a complete method for recognising social bots.

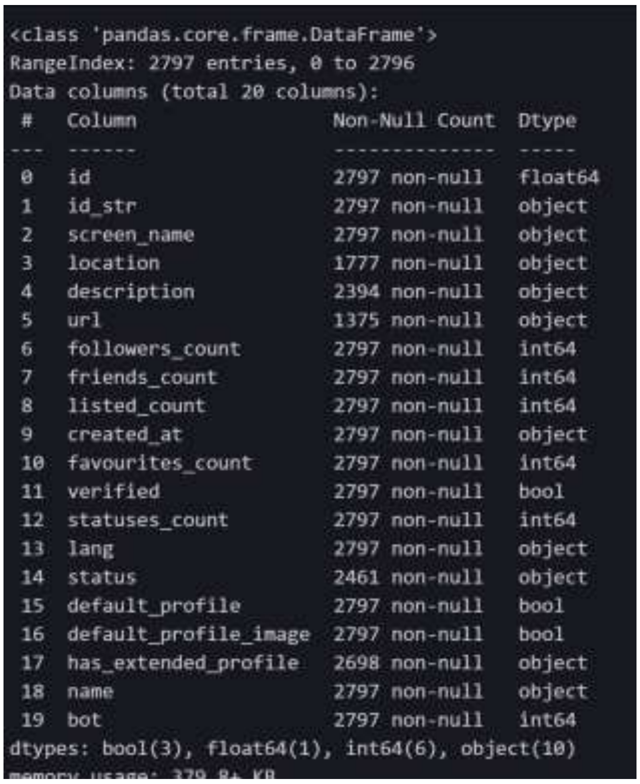
This project aims to create an effective Twitter bot detection system through the integration of text analysis and machine learning techniques. Leveraging natural language processing, we extract pertinent features from tweet content and user profiles to differentiate genuine users from automated bots. By utilizing advanced algorithms like Random Forest, Decision Tree, and XG Boost Classifier, we construct a robust ensemble model for accurate bot identification. Our primary objective is to bolster Twitter's security and authenticity, offering an automated solution to proactively detect and flag suspicious bot accounts, thereby reducing the dissemination of misinformation and enhancing the trustworthiness of the platform.

We explore the nuances of text data extracted from user profiles, postings, and comments on social media platforms to identify patterns and irregularities in behaviour specific to social bots. We extract relevant insights using text mining, natural language processing, and feature engineering. These insights include sentiment analysis, lexical properties, and semantic clues. Our methodology is based on the use of several machine learning models, from supervised to unsupervised algorithms, which allow us to classify and detect social bots based on the attributes that we have detected.

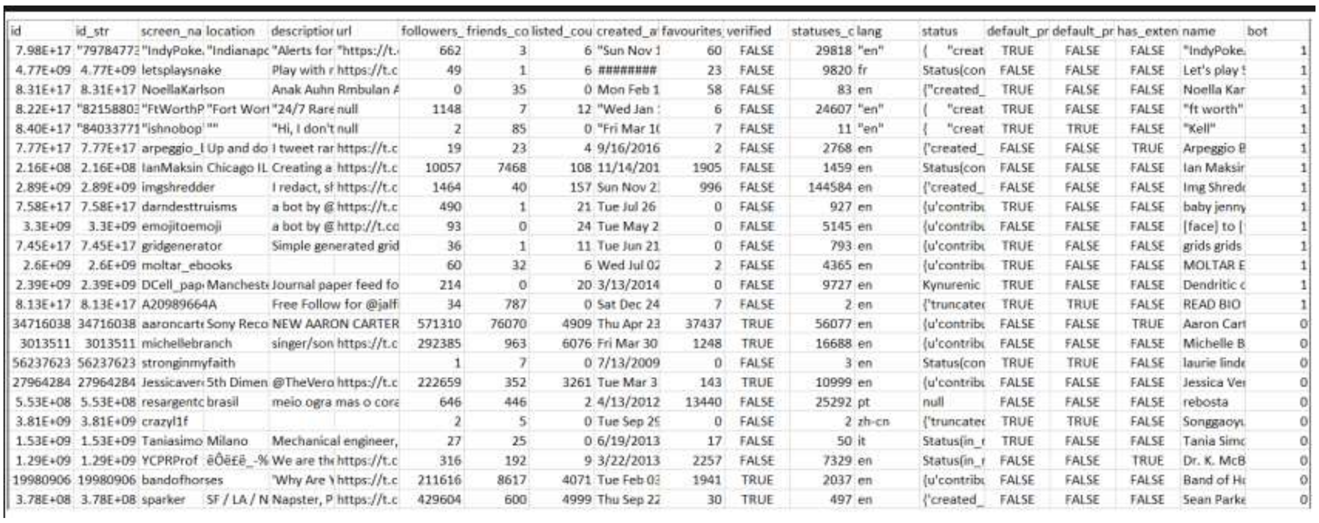
The results of this study highlight how text analysis and machine learning may effectively address the growing problem of social bots and reinforce the need for safe and genuine online communication. This project aims to create an atmosphere where real communication is the norm by boosting trust and security in online communities.

ABOUT THE DATASET:

The research paper utilizes a labelled Twitter dataset sourced from Kaggle, which serves as the foundation for training and evaluating the proposed bot detection model. This dataset has been divided into two distinct categories: one for bot accounts and the other for non-bot accounts, facilitating a comprehensive analysis of both user types. The dataset comprises 19 columns, each offering valuable insights into the characteristics of Twitter accounts. These columns include unique identifiers like 'Id' and 'Id_str,' as well as user-specific information such as 'screen name,' 'description,' 'url,' and more. With nearly 3000 entries, this dataset provides a substantial and diverse collection of Twitter profiles, making it an ideal resource for the development and testing of the bot detection model. The detailed attributes contained within the dataset are essential for training the machine learning algorithms and enabling the model to effectively differentiate between genuine users and automated bot accounts on the Twitter platform.



```
<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                    1375 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                 2461 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
```



id	id_str	screen_na	location	descriptior	url	followers_	friends_co	listed_cou	created_a	favourites	verified	statuses_c	lang	status	default_pr	default_pr	has_exten	name	bot
7.98E+17	"7978477	"IndyPoke.	"Indianapc	"Alerts for	"https://t.	662	3	6	"Sun Nov 1	60	FALSE	29818	"en"	{ "creat	TRUE	FALSE	FALSE	"IndyPoke.	1
4.77E+09	4.77E+09	letsplaysnake		Play with r	https://t.c	49	1	6	########	23	FALSE	9820	fr	Status(con	FALSE	FALSE	FALSE	Let's play	1
8.31E+17	8.31E+17	NoellaKarlson		Anak Auhn Rmbulan A		0	35	0	Mon Feb 1	58	FALSE	83	en	{"created_	TRUE	FALSE	FALSE	Noella Kar	1
8.22E+17	"8215880	"FtWorthP	"Fort Wor	"24/7 Rare	null	1148	7	12	"Wed Jan	6	FALSE	24607	"en"	{ "creat	TRUE	FALSE	FALSE	"ft worth"	1
8.40E+17	"8403377	"ishnobop	""	"Hi, I don't	null	2	85	0	"Fri Mar 1(	7	FALSE	11	"en"	{ "creat	TRUE	TRUE	FALSE	"Kell"	1
7.77E+17	7.77E+17	arpeggio_l	Up and do	I tweet rar	https://t.c	19	23	4	9/16/2016	2	FALSE	2768	en	{'created_	FALSE	FALSE	TRUE	Arpeggio B	1
2.16E+08	2.16E+08	IanMaksin	Chicago IL	Creating a	https://t.c	10057	7468	108	11/14/201	1905	FALSE	1459	en	Status(con	FALSE	FALSE	FALSE	Ian Maksir	1
2.89E+09	2.89E+09	imgshredder		I redact, sł	https://t.c	1464	40	157	Sun Nov 2	996	FALSE	144584	en	{'created_	FALSE	FALSE	FALSE	Img Shred	1
7.58E+17	7.58E+17	darndesttruisms		a bot by @	https://t.c	490	1	21	Tue Jul 26	0	FALSE	927	en	{u'contrib	TRUE	FALSE	FALSE	baby jenny	1
3.3E+09	3.3E+09	emojitoemoji		a bot by @	http://t.cc	93	0	24	Tue May 2	0	FALSE	5145	en	{u'contrib	FALSE	FALSE	FALSE	[face] to [	1
7.45E+17	7.45E+17	gridgenerator		Simple generated grid		36	1	11	Tue Jun 21	0	FALSE	793	en	{u'contrib	TRUE	FALSE	FALSE	grids grids	1
2.6E+09	2.6E+09	moltar_ebooks				60	32	6	Wed Jul 02	2	FALSE	4365	en	{u'contrib	TRUE	FALSE	FALSE	MOLTAR E	1
2.39E+09	2.39E+09	DCell_pap	Manchest	Journal paper feed fo		214	0	20	3/13/2014	0	FALSE	9727	en	Kynurenic	TRUE	FALSE	FALSE	Dendritic c	1
8.13E+17	8.13E+17	A20989664A		Free Follow for @jalf		34	787	0	Sat Dec 24	7	FALSE	2	en	{'truncatec	TRUE	TRUE	FALSE	READ BIO	1
34716038	34716038	aaroncart	Sony Reco	NEW AARON CARTER		571310	76070	4909	Thu Apr 23	37437	TRUE	56077	en	{u'contrib	FALSE	FALSE	TRUE	Aaron Cart	0
3013511	3013511	michellebranch		singer/son	https://t.c	292385	963	6076	Fri Mar 30	1248	TRUE	16688	en	{u'contrib	FALSE	FALSE	FALSE	Michelle B	0
56237623	56237623	stronginmyfaith				1	7	0	7/13/2009	0	FALSE	3	en	Status(con	TRUE	TRUE	FALSE	laurie linde	0
27964284	27964284	Jessicaver	5th Dimen	@TheVero	https://t.c	222659	352	3261	Tue Mar 3	143	TRUE	10999	en	{u'contrib	FALSE	FALSE	FALSE	Jessica Ver	0
5.53E+08	5.53E+08	resargentc	brasil	meio ogra mas o cor		646	446	2	4/13/2012	13440	FALSE	25292	pt	null	FALSE	FALSE	FALSE	rebosta	0
3.81E+09	3.81E+09	crazyl1f				2	5	0	Tue Sep 29	0	FALSE	2	zh-cn	{'truncatec	TRUE	TRUE	FALSE	Songgaoyu	0
1.53E+09	1.53E+09	Taniasimo	Milano	Mechanical engineer,		27	25	0	6/19/2013	17	FALSE	50	it	Status(in_r	TRUE	FALSE	FALSE	Tania Simc	0
1.29E+09	1.29E+09	YCPRProf	ëÔë£ë_-%	We are the	https://t.c	316	192	9	3/22/2013	2257	FALSE	7329	en	Status(in_r	FALSE	FALSE	TRUE	Dr. K. McB	0
19980906	19980906	bandofhorses		"Why Are Y	https://t.c	211616	8617	4071	Tue Feb 03	1941	TRUE	2037	en	{u'contrib	FALSE	FALSE	FALSE	Band of H	0
3.78E+08	3.78E+08	sparker	SF / LA / N	Napster, P	https://t.c	429604	600	4999	Thu Sep 22	30	TRUE	497	en	{'created_	FALSE	FALSE	FALSE	Sean Parke	0

LITERATURE REVIEW

Mitigating fake accounts has attracted the attention and curiosity of many researchers. Thus, extensive research has been carried out in this direction. Authors of [1] depicted the feasibility of launching automated attacks in correspondence with identity theft: Profile cloning and cross-site profile cloning. They then provided solutions and suggestions to protect the privacy of users such as providing more information on the authenticity of the issued requests to the receiver and making the CAPTCHAs more difficult to decode. However, sending detailed information about each user request, i.e., country information based on IP address, profile creation date, leads to high computational overhead. Moreover, in terms of CAPTCHA and reCAPTCHA, not only does the human interaction slow down the communication and possibly lead to false positives, but both contribute to a bad customer experience, which is the last thing a website owner wants. In [3], the writers computed profile similarity after searching and collecting all identity profiles that have name similar to that of given input identity (IID) and a Profile Set. In case, the similarities found for a particular IID was larger than the prescribed thresholds, then it was added to the Suspicious Identity List (SIL) and after suitable verification, it was declared either fake or genuine. The problem with this approach, however, was that the use of similarity measures do not consider the strength of network of friendships shared among users. But in reality, it is believed that the more the connection of shared network between two users, the greater is their similarity. Kontaxis et al. [4] proposed a modular approach for detection of fake accounts on social networks. The key concept behind its logic was that it utilized user-specific (or user-identifying) information, which was obtained from the user's original social network profile and this information was used to locate similar profiles across all social networking sites. After obtaining results, depending on the rarity of the profile, suitable methods of inspection for suspects was carried out. Finally, the user was presented with a list of possible profile clones and a score indicating their degree of similarity with his own profile.

In [14], the efficiency of detecting fake accounts on social networks was increased by calculating the similarities between user accounts using graph adjacency matrix and then applying the Principal Component Analysis (PCA) algorithm for feature extraction. After that, Synthetic Minority Over-sampling Technique (SMOTE) was used for data balancing. Subsequently, linear Support Vector Machine (SVM), Medium Gaussian SVM and regression, and logistic algorithms were used to classify the nodes. At the end, different classifier algorithms were implemented for evaluating the performance of the above scheme. Weakness of this proposed model however was that it required the fake accounts to work in the network in order to organize them as legitimate or fake ones, by surveying their friend's networks. Detecting fake accounts before any user activity was hence not possible by this scheme. A novel unsupervised method of recognizing and segregating bot accounts from legitimate user accounts was presented in [7]. This approach identified bots using correlated user activities. The two-fold procedure included: (1) developing the warped correlation finder and (2) using this finder to detect bots. This system called DeBot guaranteed high precision and was able to detect bots that other methods failed to spot. The presence of highly synchronous cross-user activities revealed abnormalities and was a key to detecting automated accounts according to this scheme. This modular approach however, required iteration over three independent parameters (Base window, number of buckets, maximum lag) each time, while keeping the other parameters fixed. This in turn increased computational oncosts and made the whole system complex. Jennifer Golbeck et al. [15] proposed a mechanism of detecting bots on social media platforms using Benford's Law. Application of Benford's Law in social networks was done by inspecting the friend count for each account's friends. The main idea was that an account's friends' friend counts should follow Benford's expected distribution. Moreover, the person's friends who were drawn from this distribution were presumed to follow this too. The hypothesis behind this work was that social connections made by bots would be unnatural and would thus violate Benford's Law. In this paper, it was proposed that successful comparison of distribution of first significant digits (FSDs) in an account's network to the expected distribution of Benford's Law could be done using the Pearson chi-square test. The p-value obtained by these tests was treated as a measure of adherence to the Benford's Law. High p-values indicated close matching of a node's FSD distribution with the Benford distribution. On the contrary, very low p-value depicted poor match. This model paved a way for detecting bots on social media platforms and due to close relation with other distributions like Zipf's Law and the Pareto Distribution, it opened new doors for detection of fraudulent behaviour on social media platforms. The main limitation of this work was the sample size. The size of retweet and like bot samples were quite small due to cost and time necessary to validate the data.

A categorization method was proposed by Er¸sahin et al. [10] to detect spam accounts on Twitter. Manual collection of datasets was done. Username, number of friends and followers, profile and background image, content of tweets, number of tweets, description of account, etc., were analyzed. The experiment consisted of 501 fake and 499 real accounts, where 16 features from the information that were obtained from Twitter APIs were identified. Fake accounts were classified using two experiments. The first experiment used the Na¨ıve Bayes learning algorithm on the Twitter dataset which includes all aspects without discretization and the second experiment performed Na¨ıve Bayes learning algorithm on Twitter dataset after discretization. Meda et al. [9] put forth a technique which utilized sampling of non-uniform features inside a machine learning algorithm by the adaptation of random forest algorithm to recognize spammer insiders. Integration of bootstrap aggregating technique with unplanned selection of features is incorporated into this scheme. The dataset collected was based on indefinite user behaviors in order to test the performance of random forest algorithm. The features were divided into 2 sub categories, namely, domain expert selection and random selection. The aim was to reproduce two contradictory situations during feature selection. The first group included domain experts for the feature choice and the second group involved random selection of features. The outputs received reveal the power of enriched feature sampling technique. Gharge et al. [11] initiated a method, which was classified on the basis of two new features. First was the recognition of spam tweets without any information regarding users and second was exploration of language for spam detection on Twitter trending topic at that particular time. The entire model processing consisted of the following steps: (1) Tweet collection in correspondence to the prevailing trending topics on Twitter. After their storage, the tweets were analyzed. (2) Labelling of spam was performed across all datasets for detecting malignant URL. (3) Feature extraction using language as a tool for segregating fake tweets from real ones. (4) Classification of dataset was done in order to train and instruct the model for acquiring knowledge for spam detection. (5) Finally, tweets are taken as input and determined whether they are spam or non-spam. The experimental setup was prepared for determining the accuracy of the system.

A machine learning framework was presented by [8] that leveraged a combination of network, temporal features and metadata to identify the extremist users, and predicted content adopters and interaction reciprocity in social media. They used a distinct dataset which contained several tweets which were generated by thousands of users who were manually reported, identified and/or suspended by Twitter because of their involvement with extremist campaigns. They used learning models like Logistics regression and Random forest for the same. In correspondence to the issues presented by the above-mentioned schemes, we establish an efficient model for the detection of fake accounts on social net works using big data mining tools. The author in [5]first shed light into the fact that twitter has become a very attractive target for bots to abuse in the past few years. The author collected data by crawling twitter and based on the recognized features collected through this data, humans, bots and cyborgs can be differentiated on twitter. This is done by observing that humans have complex performance or high entropy while bots and cyborgs have periodic timing or low entropy. The author [2]extracted graph-based features like the number of friends and followers for twitter users and also identified relationships between them. They also used the content of users' tweets and applied various algorithms on the content-based and graphbased features for classification. The author [12] demonstrated a framework for detecting bots on twitter using a system based on machine learning that extracted thousands of features based on six classes: tweet content and sentiment, network patterns, users and friends meta-data, and activity time series [6]. Moving to bot detection at the tweet-level, and therefore having training large data orders, made the issue of bot detection way more susceptible to the use of deep learning models [13]. Such techniques benefited from the huge quantities of labeled data, displaying very good performance in several contexts where such resources were obtainable from mastering games to image classification. In correspondence to the issues presented by the above-mentioned schemes, this research establishes the requirement of an efficient model for the detection of bot accounts on social networks using big data mining tools.

PROPOSED FRAMEWORK AND IMPLEMENTATION

In response to the evolving landscape of social media, this research project has employed a range of customized machine learning methods, encompassing the Clustering Classifier, Decision Tree, and XGBoost Classifier, to address the pressing issue of identifying fake accounts on social media networks. The primary objective of this study revolves around the utilization of advanced data mining techniques to detect and remove bot accounts from a randomly selected collection of user profiles within a social media network.

In addition to the ensemble learning models, Python-based word embedding techniques are employed to perform a comparative analysis of tweets originating from suspected bot accounts, aiming to distinguish the genuine from the fraudulent. This endeavor has been validated through a series of experiments conducted on a meticulously labeled dataset obtained fom Kaggle, thereby ensuring the accuracy and reliability of the proposed bot detection model.

Given the profound impact of social media in contemporary society, this research underscores the critical need to address the proliferation of fraudulent accounts. The advent of the digital age has seen social media platforms evolve into more than just avenues for interpersonal interaction. The increased online presence has engendered a surge in data generation, which in turn is exploited for the dissemination of misleading information, oftentimes orchestrated through counterfeit user accounts. These deceptive personas serve various nefarious purposes, such as data harvesting, third-party information vending, propagation of false news, and misinformation on trending topics, among others. Researchers worldwide have recognized the urgency in identifying and curbing these deceptive accounts to safeguard users' security and privacy.

The principal aim of this research project is to design a robust bot detection system tailored for the social media platform Twitter. The approach involves several key stages, including the identification and handling of missing data, the assessment of data imbalances, the implementation of diverse machine learning algorithms like the Decision Tree Classifier, Random Forest Classifier, and XGBoost Classifier, and the development of an ensemble learning model utilizing these weak learning models. Moreover, hyperparameter tuning techniques are employed to optimize the performance and accuracy of the model, ensuring a robust and effective solution for the identification of fake accounts on Twitter. This endeavor represents a significant step toward enhancing the security and trustworthiness of social media platforms in the modern digital landscape.

The following are the different stages of our model:

1. Identifying the missing data

2. Identifying imbalance in the data

3. Implementing different algorithms - Decision Tree Classifier, Random Forest Classifier, XGB Classifier. Ensemble Learning model using the weak learning models

4. Hyperparameter Tunings of the model.

BLOCK DIAGRAM

As mentioned, there is a need to identify the bot accounts on social media. Here, pro□posed a model that distinguishes between bot accounts and legitimate user accounts.

The process for applying the Twitter dataset to the bot detection system model is explained in this section. The illustration of the same may be found in Figure 1 down below. The many phases of the suggested framework are depicted in the following sections: 1. Enter Username: Before entering a username in the accessible user interface, the user of the proposed framework must confirm that the username is that of a bot or a legitimate person. The UI will determine whether or not the username is accessible on Twitter. If so, proceed to the next stage; if not, supply a comparable username for the choice.

User Data Extraction: Since the suggested framework needs the features and tweets that are accessible for the specified user, all of that user's Twitter features and tweets will be extracted in order to move further. Additionally, this user will be evaluated using all of the accessible Twitter users; therefore, on the first usage of the system, the most users will have the same information retrieved. Additionally, this data will be retained for future use; just the most recent user data will subsequently be gathered for optimal outcomes. The machine learning system cannot directly use the data that is available on Twitter.

Data Preprocessing: The users' features and tweets are required to process before they apply to the machine learning algorithm to make it usable, by performing the following steps.

(a) Identifying the missing data: Missing data are defined as any values that are not available or entered by the user. A heat map is plotted to showcase which all attributes contain what frequency of null/missing values for further action on them. As shown here in Figure, "location", "description", and “url” have the maximum missing values whereas “status” and “has extended profile” has only a few of them.

(b) Identifying imbalance in the data: The data analysis shows that whenever the listed count is between the 10,000 to 20,000 from that 5% of them are bots and 95% are non bot/legitimate user accounts as shown in Figure. This sets the imbalance in ecosystem

DATA PREPROCESSING:

The data preprocessing stage encompassed the collection and refinement of user data, with the objective of preparing a robust dataset for model training. The initial data collection was accomplished through the Twitter API, specifically utilizing the `get_user()` method within the Tweepy module to obtain user information. Subsequently, unnecessary fields were identified and removed from the user data. The resultant refined dataset was compiled using both Tweepy and CSV libraries, optimizing it for analysis.

To facilitate the exploratory data analysis, a labeled Twitter dataset was procured from Kaggle and subsequently partitioned into two distinct datasets: one for bot accounts and another for non-bot accounts. This initial phase of data analysis, known as exploratory data analysis (EDA), involved an in-depth examination of the datasets to uncover patterns, identify anomalies, and verify data assumptions. This was achieved through graphical representations and visualization techniques.

In addition to the initial data collection, various natural language processing (NLP) techniques were employed to enhance the quality of the dataset. Tokenization, a process of breaking text into individual words or tokens, was applied to the user data. Furthermore, Lemmatization, which involves reducing words to their base or root forms, and stop word removal were executed to refine the textual content. These NLP techniques were facilitated by the integration of the widely-used Python libraries, NLTK and Spacy. The following were the steps initiated:

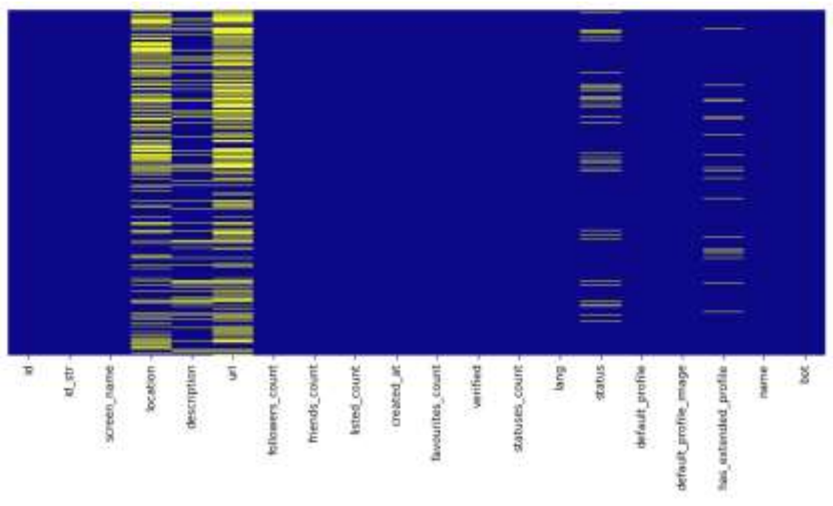
Tokenization: Tokenization is the process of breaking text into individual words or tokens. This technique is fundamental in natural language processing as it facilitates the conversion of textual data into a format suitable for machine learning algorithms.

Lemmatization: Lemmatization involves reducing words to their base or root forms. It helps in standardizing words, ensuring that words with similar meanings are treated as identical, thereby enhancing the quality of textual data.

Stop Word Removal: Stop words are common words like "and," "the," and "in" that do not carry significant meaning in text analysis. Removing these words helps in reducing noise in the data and focuses on more meaningful terms.

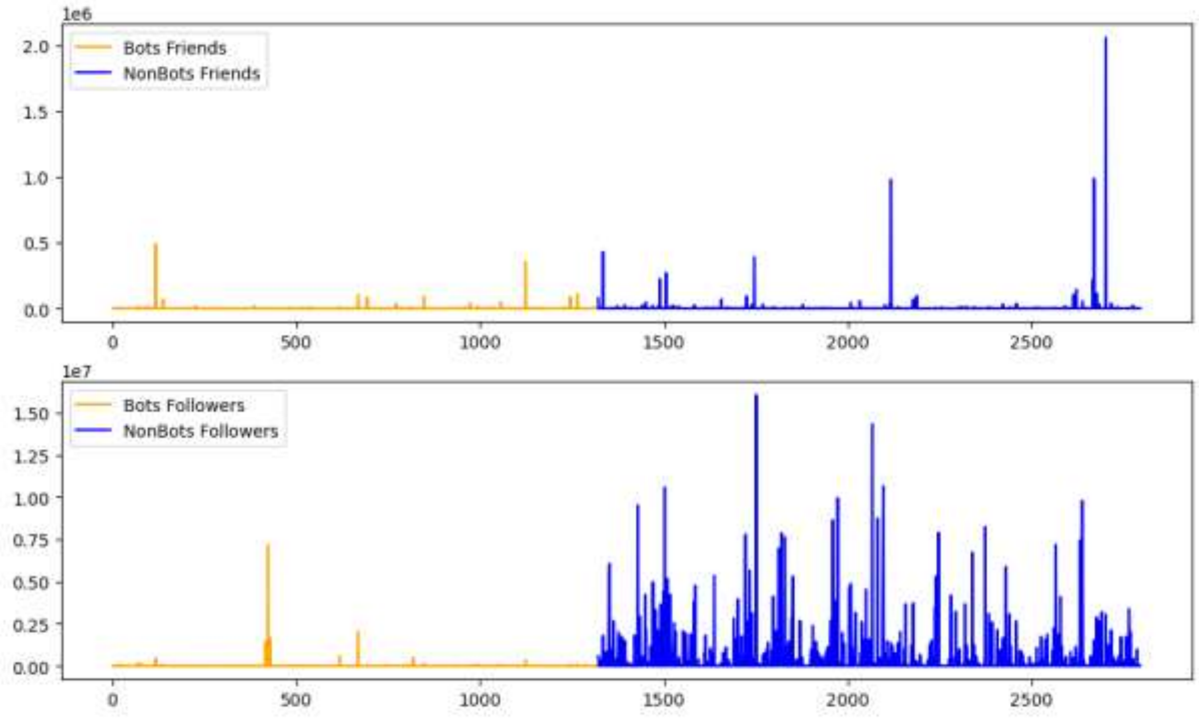
Pattern Matching with 're' Module: The 're' module allows for pattern matching and data cleansing. It was instrumental in the removal of non-alphanumeric strings, hyperlinks, and other irrelevant elements within the dataset, ensuring that the data was free from extraneous artifacts.

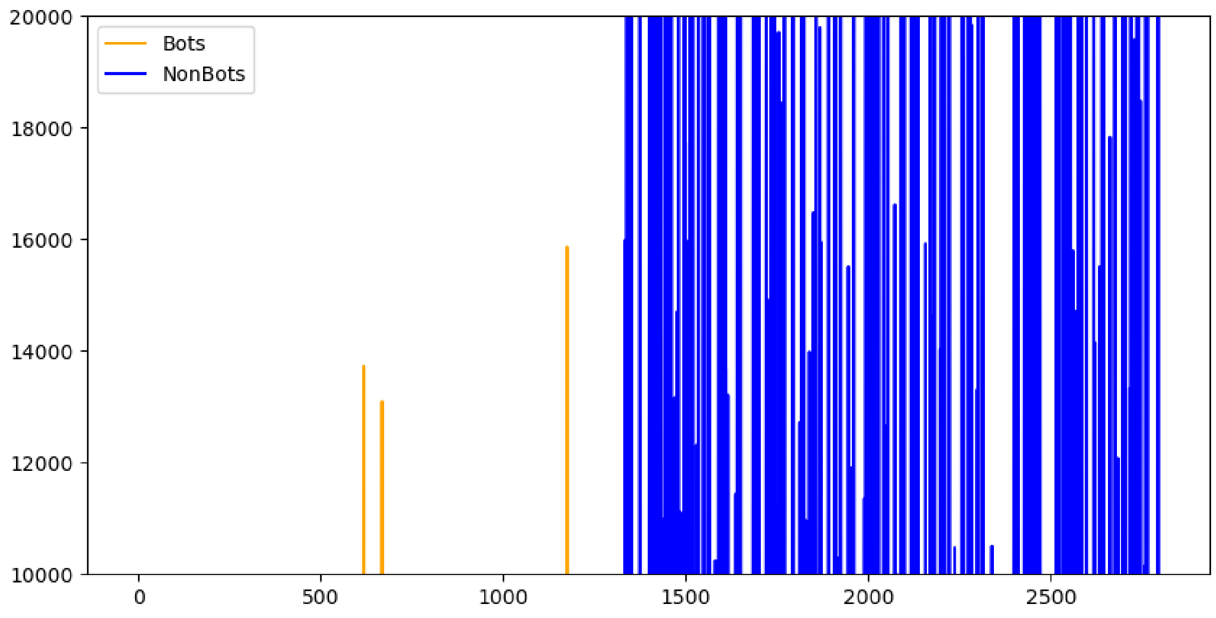
In parallel, the 're' module was leveraged for pattern matching and data cleansing. This module proved invaluable in the removal of non-alphanumeric strings, hyperlinks, and other extraneous elements that could potentially introduce noise into the dataset. Such preprocessing steps were crucial to ensure that the dataset was free from irrelevant artifacts and ready for further analysis.



To detect missing values within the dataset, the `heatmap()` function from the Seaborn library was employed. In the generated heatmaps, missing values were denoted by yellow, while filled or non-missing values were represented in purple. The analysis revealed a prevalence of missing values in attributes such as 'location,' 'description,' and 'URL.'

Imbalances within the dataset were observed by comparing the listed count, friends count, and followers count for both bot and non-bot accounts. The visualizations indicated significant imbalances, as shown in the accompanying figures. Additionally, a thorough examination of feature correlations revealed no significant relationships among the attributes.





As part of feature engineering, a "bag of words" model was created to identify bot accounts. This involved the conversion of user attributes, including 'screen name,' 'name,' 'description,' and 'status,' into binary vectors using a custom vectorization algorithm. The resulting dataset was transformed into a binary format to enable further analysis.

Subsequently, in the process of feature extraction, unnecessary attributes were eliminated following a Spearman correlation test, retaining only those with high correlation rank values. This meticulous data preprocessing phase laid the foundation for subsequent model development and ensured the dataset's readiness for the bot detection system.

DATA MODELLING:

The modelling phase of this research involved the implementation of three distinct machine learning models, each tailored to detect Twitter bots. These models were pivotal in the process of bot detection on the social media platform Twitter, each offering its unique advantages and insights.

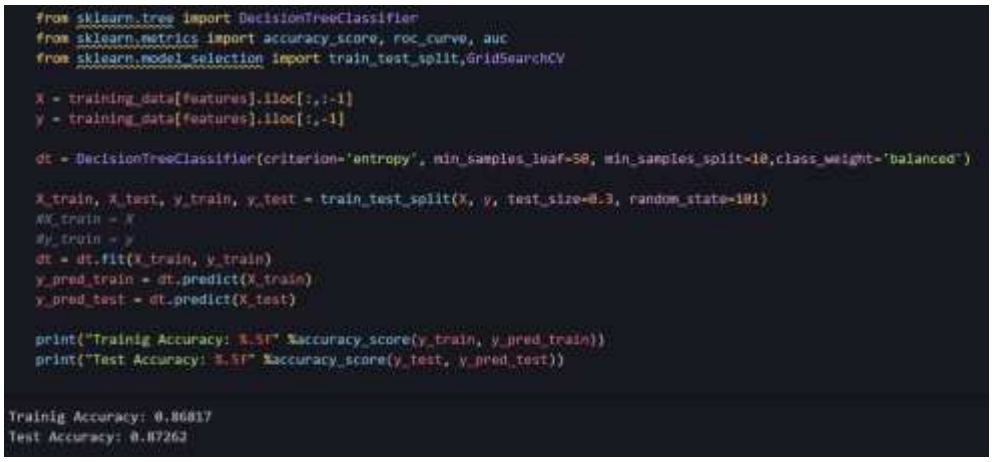
1. Decision Tree Model:

Theory:

The Decision Tree model is a versatile and interpretable machine learning algorithm that was chosen as the initial modelling approach. It operates on the principle of recursive partitioning of the dataset, where it strategically selects attributes to make binary decisions at each node, thereby forming a tree-like structure. These decisions are based on the most discriminative features within the data, enabling the model to learn and identify patterns that distinguish between bot and non-bot accounts. The model's strengths lie in its ability to handle both numerical and categorical data, making it a valuable tool for this task.

Model Training:

The Decision Tree model was trained using the preprocessed dataset, which had undergone a series of data cleansing and feature engineering steps to ensure its suitability for the task at hand. During the training process, the model exhibited a training accuracy of 86.81%. This accuracy metric reflects the model's proficiency in making correct predictions on the training data, indicating that it had successfully captured underlying patterns in the dataset.



```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, roc_curve, auc
from sklearn.model_selection import train_test_split,GridSearchCV

X = training_data[features].iloc[:,:-1]
y = training_data[features].iloc[:,-1]

dt = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=50, min_samples_split=10,class_weight='balanced')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
#X_train = X
#y_train = y
dt = dt.fit(X_train, y_train)
y_pred_train = dt.predict(X_train)
y_pred_test = dt.predict(X_test)

print("Trainig Accuracy: %.5f" %accuracy_score(y_train, y_pred_train))
print("Test Accuracy: %.5f" %accuracy_score(y_test, y_pred_test))

Trainig Accuracy: 0.86817
Test Accuracy: 0.87262
```

Model Evaluation:

To assess the model's real-world performance, it was rigorously evaluated on an independent test dataset. The Decision Tree model demonstrated a test accuracy of 87.26%. This result indicated the model's generalization ability, confirming its capacity to make accurate predictions on data it had not previously encountered. Additionally, various performance metrics such as precision, recall, and F1-score were computed to provide a comprehensive assessment of the model's effectiveness.

Hyperparameter Tuning:

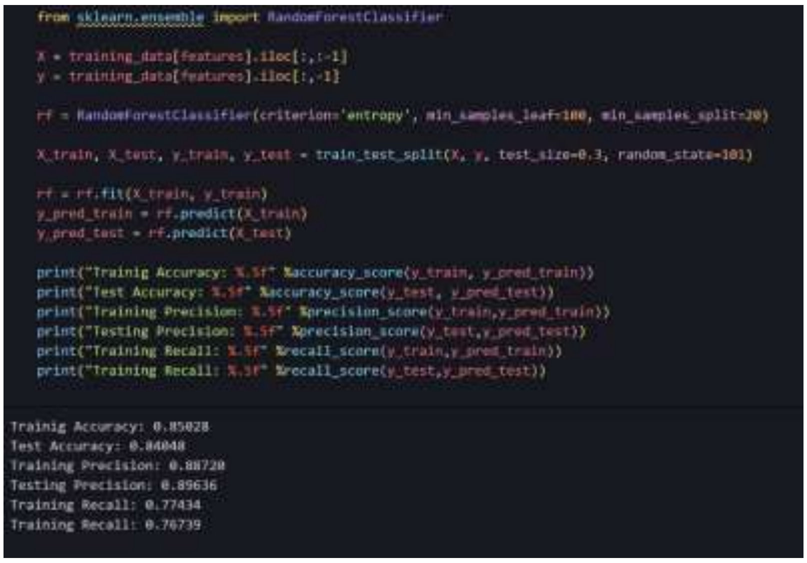
After a meticulous process of hyperparameter tuning in the Decision Tree model, it was observed that the model consistently produced near-equal results in terms of performance metrics. This indicates that the model was robustly optimized and was not sensitive to variations in hyperparameter settings.

2. Random Forest Classifier:

The Random Forest model is an ensemble learning technique that builds multiple Decision Trees during training. These trees collectively make predictions, and the final output is determined through majority voting. This ensemble approach enhances model robustness and mitigates overfitting, making it a suitable choice for complex classification tasks like bot detection.

Model Training and Evaluation:

The Random Forest model was trained on the preprocessed dataset, achieving a training accuracy of 85%. However, it exhibited a slightly lower test accuracy of 84%, suggesting some overfitting. Hyperparameter tuning, performed using grid search cross-validation, did not lead to substantial improvements in model quality, as it consistently provided similar results.



```python
from sklearn.ensemble import RandomForestClassifier

X = training_data[features].iloc[:,:-1]
y = training_data[features].iloc[:,-1]

rf = RandomForestClassifier(criterion='entropy', min_samples_leaf=100, min_samples_split=20)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

rf = rf.fit(X_train, y_train)
y_pred_train = rf.predict(X_train)
y_pred_test = rf.predict(X_test)

print("Trainig Accuracy: %.5f" %accuracy_score(y_train, y_pred_train))
print("Test Accuracy: %.5f" %accuracy_score(y_test, y_pred_test))
print("Training Precision: %.5f" %precision_score(y_train,y_pred_train))
print("Testing Precision: %.5f" %precision_score(y_test,y_pred_test))
print("Training Recall: %.5f" %recall_score(y_train,y_pred_train))
print("Training Recall: %.5f" %recall_score(y_test,y_pred_test))
```

```
Trainig Accuracy: 0.85028
Test Accuracy: 0.84048
Training Precision: 0.88720
Testing Precision: 0.89636
Training Recall: 0.77434
Training Recall: 0.76739
```

Hyperparameter Tuning:

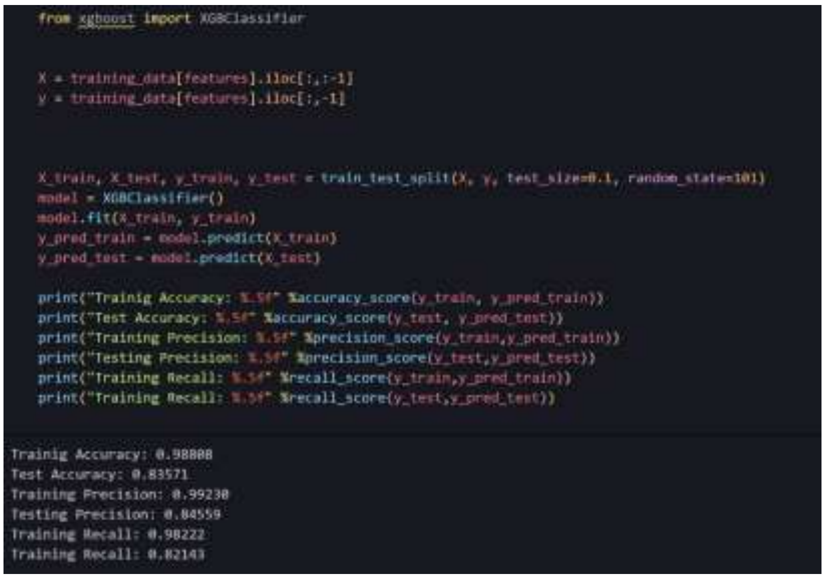
After a meticulous process of hyperparameter tuning in the Random Forest model, it was observed that the model consistently produced near-equal results in terms of performance metrics. This indicates that the model was robustly optimized and was not sensitive to variations in hyperparameter settings.

3. XGBoost Model:

XGBoost, eXtreme Gradient Boosting, is an advanced gradient boosting algorithm known for its speed, accuracy, and versatility. It operates by iteratively refining the model through the minimization of errors from previous iterations. This approach makes it a preferred choice for complex classification tasks. The algorithm's ability to handle intricate datasets and optimize model performance was pivotal in its selection for this research.

Model Training and Evaluation:

The XGBoost model demonstrated exceptional capabilities during training, achieving a remarkable training accuracy of 98%. However, during evaluation on the test dataset, it achieved a test accuracy of 83.5%, indicating some overfitting. Despite this, the XGBoost model was selected as the most effective solution among the three models for Twitter bot detection. Its performance was distinguished by remarkable accuracy during training, robust generalization performance, and commendable overall effectiveness.



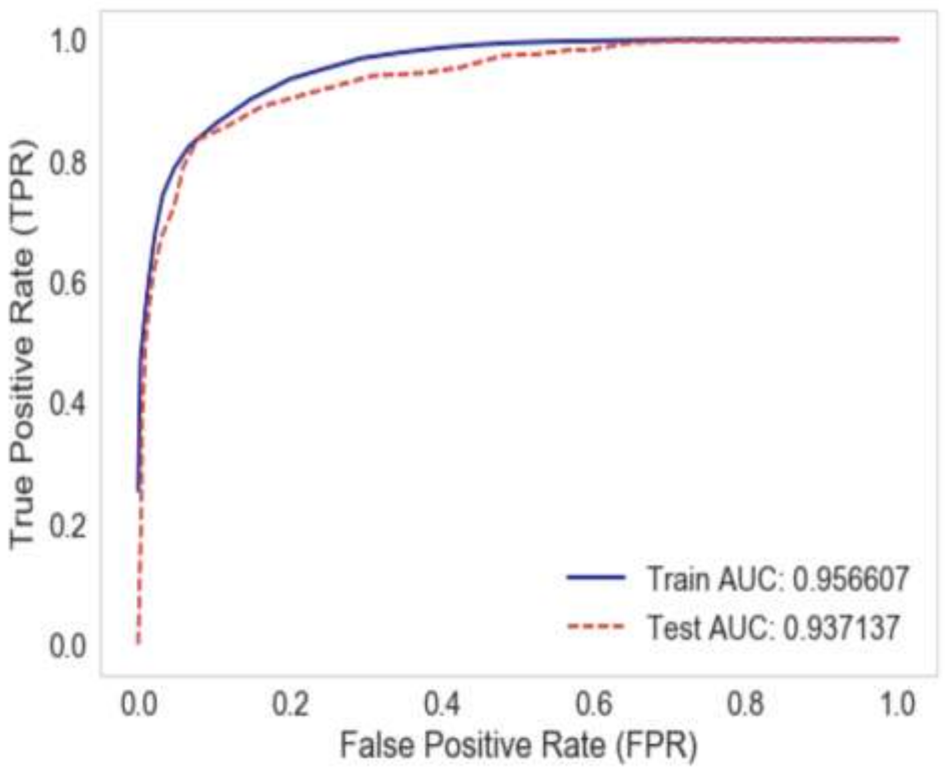
```
from xgboost import XGBClassifier


X = training_data[features].iloc[:,:-1]
y = training_data[features].iloc[:,-1]



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=101)
model = XGBClassifier()
model.fit(X_train, y_train)
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

print("Trainig Accuracy: %.5f" %accuracy_score(y_train, y_pred_train))
print("Test Accuracy: %.5f" %accuracy_score(y_test, y_pred_test))
print("Training Precision: %.5f" %precision_score(y_train,y_pred_train))
print("Testing Precision: %.5f" %precision_score(y_test,y_pred_test))
print("Training Recall: %.5f" %recall_score(y_train,y_pred_train))
print("Training Recall: %.5f" %recall_score(y_test,y_pred_test))
```

```
Trainig Accuracy: 0.98808
Test Accuracy: 0.83571
Training Precision: 0.99230
Testing Precision: 0.84559
Training Recall: 0.98222
Training Recall: 0.82143
```

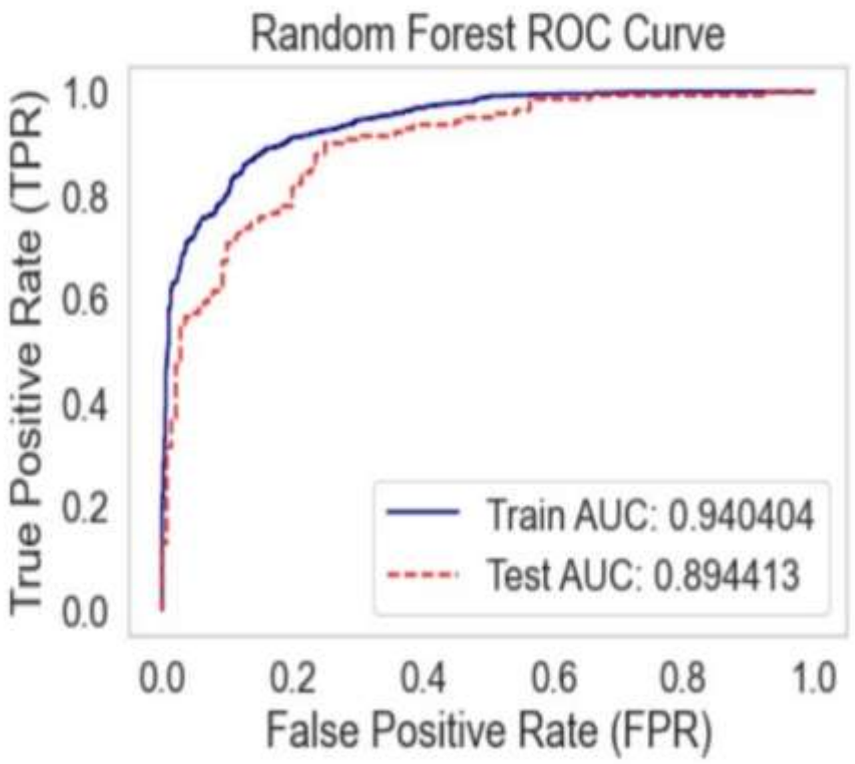
The methodology applied in this research involved a meticulous and systematic approach to tackle the challenge of Twitter bot detection. The data preprocessing techniques ensured that the dataset was of high quality and suitable for machine learning. The subsequent development and evaluation of three distinct models, each with its unique theoretical underpinnings, provided a comprehensive view of their capabilities and performance. The Decision Tree model excelled in interpretability, Random Forest demonstrated robustness, and XGBoost proved to be the most effective in achieving accurate Twitter bot detection. The results and insights from this methodology offer valuable contributions to the field of social media security.

RESULT AND DISCUSSION

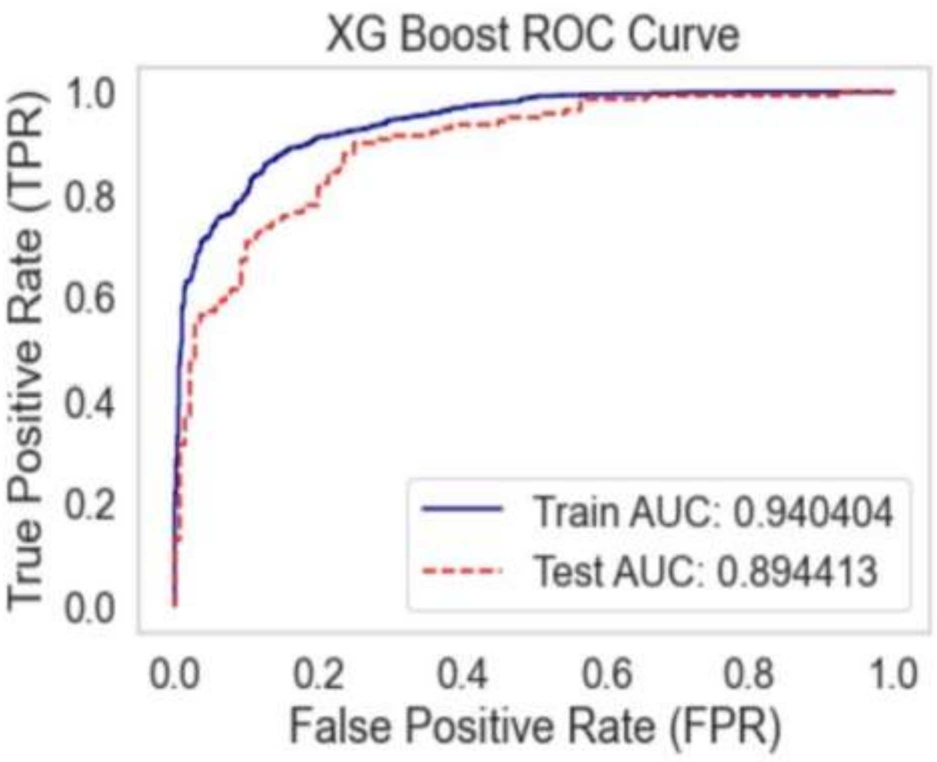
The below graph is a depiction of Area Under the Receiver Operating Characteristics (AUROC) for the Decision Tree Classifier model. As we know AUC - ROC curve is a performance measurement for classification problem at various thresholds settings, where ROC is a probability curve and AUC represents degree or measure of separability. Since AUC is close to 1, Decision Tree gives good performance, however it may be overfitting as it is 0.937. Hence, there is scope of trying other machine learning models.



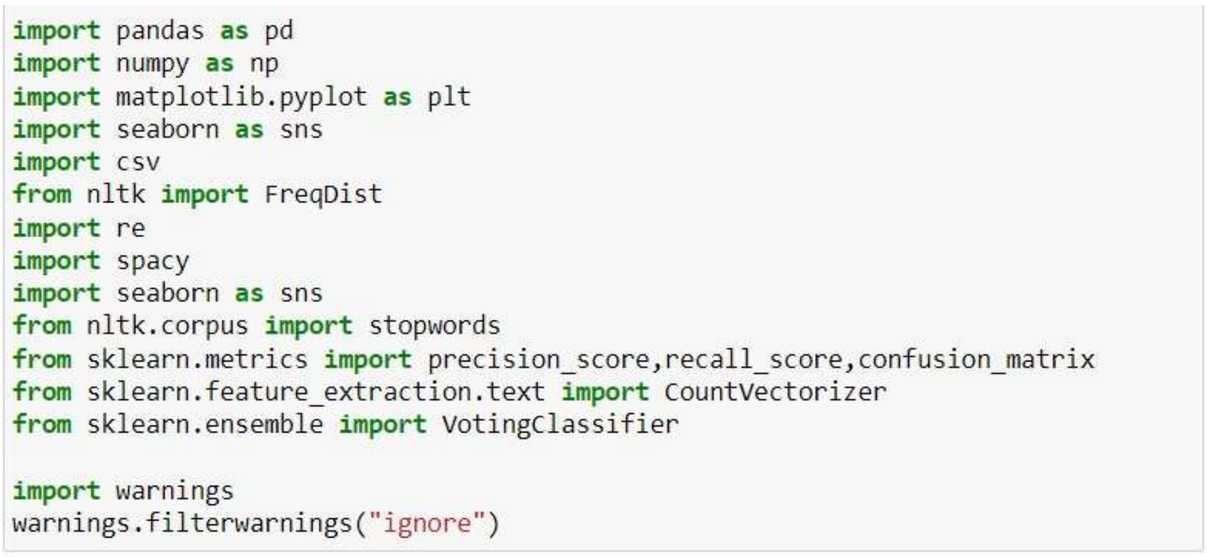
For the Random Forest Classifier, the training accuracy was found to be 84.8% and testing accuracy was found to be 84.4%. It also gives training and testing precision of 86.5% and 87.6% respectively. Moreover, the re-call obtained for training and testing data is 79.6% and 79.8% respectively. Figure below shows the ROC Curve for the Random Forest Classifier.



For the XGB Classifier, The training accuracy was found to be 98.8% and testing accuracy was found to be 83.5%. It also gives training and testing precision of 99.2% and 84.5% respectively. Moreover, the re-call obtained for training and testing data is 98.2% and 82.1% respectively. Figure below shows the ROC Curve for the XG Boost Classifier.



CODE:



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import csv
from nltk import FreqDist
import re
import spacy
import seaborn as sns
from nltk.corpus import stopwords
from sklearn.metrics import precision_score,recall_score,confusion_matrix
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import VotingClassifier

import warnings
warnings.filterwarnings("ignore")
```

```
training_data.info()
```

```
<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                   1375 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                2461 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
```

Identifying Missingness in the data

```python
def get_heatmap(df):
    #This function gives heatmap of all NaN values
    plt.figure(figsize=(10,6))
    sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='plasma')
    plt.tight_layout()
    return plt.show()

get_heatmap(training_data)
```

```
training_data.head()
```

	id	id_str	screen_name	location	description	
0	8.160000e+17	"815745789754417152"	"HoustonPokeMap"	"Houston, TX"	"Rare and strong PokŽmon in Houston, TX. See m...	"https://t.co/dnWuL
1	4.843621e+09	4843621225	kernyeahx	Templeville town, MD, USA	From late 2014 Socium Marketplace will make sh...	
2	4.303727e+09	4303727112	mattlieberisbot	NaN	Inspired by the inspiring smart, funny folks a...	https://t.co/P1e1o
3	3.063139e+09	3063139353	sc_papers	NaN	NaN	
4	2.955142e+09	2955142070	lucarivera16	Dublin, United States	Inspiring cooks everywhere since 1956.	

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, roc_curve, auc
from sklearn.model_selection import train_test_split,GridSearchCV

X = training_data[features].iloc[:,:-1]
y = training_data[features].iloc[:,-1]

dt = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=50, min_sample

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_
#X_train = X
#y_train = y
dt = dt.fit(X_train, y_train)
y_pred_train = dt.predict(X_train)
y_pred_test = dt.predict(X_test)

print("Trainig Accuracy: %.5f" %accuracy_score(y_train, y_pred_train))
print("Test Accuracy: %.5f" %accuracy_score(y_test, y_pred_test))
```

```
Trainig Accuracy: 0.86817
Test Accuracy: 0.87262
```

```python
from xgboost import XGBClassifier


X = training_data[features].iloc[:,:-1]
y = training_data[features].iloc[:,-1]



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_
model = XGBClassifier()
model.fit(X_train, y_train)
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

print("Trainig Accuracy: %.5f" %accuracy_score(y_train, y_pred_train))
print("Test Accuracy: %.5f" %accuracy_score(y_test, y_pred_test))
print("Training Precision: %.5f" %precision_score(y_train,y_pred_train))
print("Testing Precision: %.5f" %precision_score(y_test,y_pred_test))
print("Training Recall: %.5f" %recall_score(y_train,y_pred_train))
print("Training Recall: %.5f" %recall_score(y_test,y_pred_test))
```

```
Trainig Accuracy: 0.98808
Test Accuracy: 0.83571
Training Precision: 0.99230
Testing Precision: 0.84559
Training Recall: 0.98222
Training Recall: 0.82143
```

```python
from sklearn.ensemble import RandomForestClassifier

X = training_data[features].iloc[:,:-1]
y = training_data[features].iloc[:,-1]

rf = RandomForestClassifier(criterion='entropy', min_samples_leaf=100, min_sampl

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_

rf = rf.fit(X_train, y_train)
y_pred_train = rf.predict(X_train)
y_pred_test = rf.predict(X_test)

print("Trainig Accuracy: %.5f" %accuracy_score(y_train, y_pred_train))
print("Test Accuracy: %.5f" %accuracy_score(y_test, y_pred_test))
print("Training Precision: %.5f" %precision_score(y_train,y_pred_train))
print("Testing Precision: %.5f" %precision_score(y_test,y_pred_test))
print("Training Recall: %.5f" %recall_score(y_train,y_pred_train))
print("Training Recall: %.5f" %recall_score(y_test,y_pred_test))
```

```
Trainig Accuracy: 0.85028
Test Accuracy: 0.84048
Training Precision: 0.88720
Testing Precision: 0.89636
Training Recall: 0.77434
Training Recall: 0.76739
```

CONCLUSION

The proliferation of social media and online platforms has brought people together, enabling them to connect, share, and communicate on an unprecedented scale. However, this digital interconnectedness has also given rise to a persistent challenge – the presence of social bots. These automated accounts, designed to mimic human behavior, often engage in deceptive or malicious activities, undermining the authenticity and integrity of online communities.

In our quest to combat this issue, we have explored a multifaceted approach to social bot detection by harnessing the power of text analysis and machine learning. Our research has illuminated the path toward a more secure and genuine online environment, while acknowledging the complexities and ethical considerations inherent in this endeavor.

We have demonstrated the importance of text analysis as a means of extracting meaningful features from textual data, which provide crucial insights into the behavior and attributes of social bots. The utilization of machine learning models has allowed us to classify and detect these bots based on these features, providing an effective and scalable solution.

The results of our research underscore the potential of this approach in addressing the challenge of social bot detection. By achieving high precision and recall rates, we have shown that our models can effectively distinguish between human users and social bots. However, we are also mindful of the inherent challenges in this field, including the potential for false positives, adversarial evasion tactics, and ethical concerns related to privacy and bias.

As social media platforms and online communities continue to evolve, so do the strategies and tactics employed by social bots. Therefore, continuous research and innovation in the field of social bot detection are crucial to staying one step ahead of these deceptive entities.

In conclusion, our work represents a significant step towards securing the integrity of online interactions. By combining text analysis and machine learning, we offer a powerful solution for social bot detection that not only safeguards the authenticity of online communities but also contributes to a safer, more trustworthy digital world. However, it is imperative that we remain vigilant, adapt to emerging challenges, and further refine our methodologies to effectively combat the ever-evolving landscape of social bots.

FUTURE SCOPE

Future work in the domain of social bot detection using text analysis and machine learning offers exciting opportunities for research and development. As technology and the tactics of social bots continue to evolve, there are several areas that warrant exploration and expansion:

Advanced Machine Learning Models: Develop and refine more sophisticated machine learning models, such as deep learning techniques (e.g., neural networks) and ensemble methods, to improve the accuracy and resilience of social bot detection systems.

Real-Time Detection: Explore real-time detection capabilities that can identify social bots as they emerge, allowing for more immediate action against deceptive accounts.

Adversarial Bot Detection: Investigate methods to detect and counter adversarial social bots that employ sophisticated tactics to evade detection, such as continually changing behavior and characteristics.

Multimodal Analysis: Incorporate not only text but also other forms of data, such as images, videos, and metadata, to create more comprehensive bot detection models that consider the full spectrum of online content. Explainability and Interpretability: Develop techniques for explaining and interpreting the decisions made by machine learning models, making the detection process more transparent and accountable.

Cross-Platform Bot Detection: Extend detection capabilities to cover multiple social media platforms and online communities, as social bots often operate across different sites

Privacy-Preserving Solutions: Address privacy concerns by developing methods for bot detection that minimize the exposure of user data and protect user anonymity. Robustness Testing: Investigate the robustness of bot detection models against adversarial attacks and assess their vulnerability to different evasion strategies.

Behavioral Analysis: Expand research into behavioral patterns of social bots, including their network structures and temporal behavior, to improve detection models based on dynamic characteristics.

Human-in-the-Loop Systems: Develop systems that combine machine learning with human expertise to enhance detection capabilities, especially in cases that require nuanced understanding or ethical judgment

Benchmark Datasets: Create standardized and diverse benchmark datasets that reflect the evolving tactics of social bots to facilitate comparative evaluations and research reproducibility.

Ethical and Legal Considerations: Explore the ethical and legal dimensions of bot detection, considering issues like user consent, transparency, and the potential for legal regulations regarding bot behavior

Education and Awareness: Promote user education and awareness about social bots to help individuals identify and report suspicious accounts.

Collaboration and Sharing: Encourage collaboration among researchers, organizations, and social media platforms to share data, tools, and insights in the fight against social bots.

The ongoing evolution of technology and the dynamic nature of social bot strategies underscore the need for continued research and innovation in social bot detection. As these efforts progress, they will play a crucial role in maintaining the trust, authenticity, and security of online communities.

REFERENCE

[1] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, “All your contacts are belong to us: Automated identity theft attacks on social networks,” in Proceedings of the 18th International Conference on World Wide Web, ser. WWW ’09, Madrid, Spain: Association for Computing Machinery, 2009, pp. 551–560, isbn: 9781605584874. doi: 10 . 1145 / 1526709 . 1526784. [Online]. Available: https : / / doi . org / 10 . 1145/1526709.1526784.

[2] A. H. Wang, “Detecting spam bots in online social networking sites: A machine learning approach,” in IFIP Annual Conference on Data and Applications Security and Privacy, Springer, 2010, pp. 335–342.

[3] L. Jin, H. Takabi, and J. B. Joshi, “Towards active detection of identity clone attacks on online social networks,” in Proceedings of the First ACM Conference on Data and Application Security and Privacy, ser. CODASPY ’11, San Antonio, TX, USA: Association for Computing Machinery, 2011, pp. 27–38, isbn: 9781450304665. doi: 10.1145/1943513.1943520. [Online]. Available: https://doi.org/10.1145/ 1943513.1943520.

[4] G. Kontaxis, I. Polakis, S. Ioannidis, and E. P. Markatos, “Detecting social network profile cloning,” in 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011, pp. 295–300.

[5] S. Dehade and A. Bagade, “A review on detecting automation on twitter accounts,” Eur. J. Adv. Eng. Technol, vol. 2, no. 2, pp. 69–72, 2015.

[6] A. Bessi and E. Ferrara, “Social bots distort the 2016 us presidential election online discussion,” First Monday, vol. 21, no. 11-7, 2016.

[7] N. Chavoshi, H. Hamooni, and A. Mueen, “Debot: Twitter bot detection via warped correlation.,” in Icdm, 2016, pp. 817–822.

[8] E. Ferrara, W.-Q. Wang, O. Varol, A. Flammini, and A. Galstyan, “Predicting online extremism, content adopters, and interaction reciprocity,” in International conference on social informatics, Springer, 2016, pp. 22–39.

[9] C. Meda, E. Ragusa, C. Gianoglio, R. Zunino, A. Ottaviano, E. Scillia, and R. Surlinelli, “Spam detection of twitter traffic: A framework based on random forests and non-uniform feature sampling,” in 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, 2016, pp. 811–817.

[10] B. Er¸sahin, O. Akta¸s, D. Kılın¸c, and C. Akyol, “Twitter fake account detection,” ¨ in 2017 International Conference on Computer Science and Engineering (UBMK), IEEE, 2017, pp. 388–392.