Twitter Spam Detection

Meer Theodore Baloch & Yuvraj Singh

4/6/23

Table of contents

List of Figures

List of Tables

Preface

The increasingly widespread deployment of malicious bot accounts on Twitter has been steadily developing into a major concern. It is imperative to develop a reliable method to identify and classify these bot accounts to maintain the integrity of the platform and halt the spread of misinformation, spam and harassment This report aims to address this problem by using machine learning algorithms to classify Twitter accounts as bots or humans. The report will examine the tweets and user data provided by the Twitter developer API and a research article to train the model. The model will be evaluated based on various metrics such as accuracy, precision, recall, and F1 score, and hyperparameter tuning will be performed to select the best parameters to classify bot accounts efficiently. Moreover, the report will also analyze the content of tweets to identify patterns in language and user features. To accomplish the goal of creating an effective yet reliable bot identification machine learning model, we will follow a strict structure of developing a machine learning model by undergoing data exploration, data preprocessing, model selection, and performance evaluation. Furthermore, the project includes detailed visualizations and explanations of the results, allowing the model performance to be better understand such that readers can make informed decisions about their implementations.

Abstract

Social media has become an integral part of our lives, and Twitter is one of the leaders as a platform for sharing ideas, art, and general information. The increasing use of malicious bot accounts has tainted the platform, as these accounts can be used for malicious activities such as spamming, spreading false information, and targetted harassment. Therefore, there is a need for a reliable method to identify bot accounts on Twitter. To begin, the objective of the project is to correctly classify whether or not a Twitter account is a "bot" account (automated Twitter account) based on the account data (i.e. user tweets). By examining the tweets and user data produced by these accounts made available through the Twitter developer API & enhanced by the research article "Evidence of Spam and Bot Activity in Stock Microblogs on Twitter.", a machine-learning model will be trained to classify bot accounts. Finally, We will use various machine learning algorithms to develop our model, including random forest, K Nearest Neighbours and a Sequential Neural Network. These algorithms will prove effective in the classification of Twitter accounts, and we will evaluate the performance of the models via the accuracy, precision, recall and F1 score of the algorithms. Moreover, with hyperparameter tuning, we will select the best parameters from the user data to efficiently classify bot accounts. Additionally, we will analyze the content of the tweets themselves, looking for patterns in the language used, the types of links shared, and the sentiment expressed. We will also conduct a cross-validation analysis to ensure that the model is robust and not overfitting the data.

Chapter 1

Introduction to Project

1.1 Overview

The internet has revolutionized the way people communicate and interact with each other and at the epicentre of this societal shift is Twitter. Twitter's long-standing reputation has kept it within the ever-turning social sphere but has also allowed it to fester as a breeding ground for spam users as fake accounts which are often created individually or as part of a wider botnet to spread spam and malicious content on the platform. The detection and removal of these automated accounts have become a major challenge for Twitter and its users.

In this project, we aim to develop a system for detecting automated accounts on Twitter based on profile information and tweets. We will use a supervised learning approach to classify users as either spam or not spam. We will also explore various data pre-processing techniques and classification models to achieve optimal accuracy in spam user detection.

1.2 Objectives of Project

The main objective of this project is to detect and classify automated users on Twitter using machine learning techniques. Specifically, we aim to achieve the following objectives:

- Pre-process a dataset of tweets and user profile information to use as features in our classification model, specifically employing various pre-processing techniques such as
  - Data cleaning
  - Data normalization
  - Data transformation
  - Feature selection
  - Text pre-processing
- Develop a supervised learning approach with a categorical target variable to classify a user as spam or not spam
- Explore various supervised classification models such as K Nearest Neighbors, Random Forest, and Sequential Neural Network Models and evaluate their performance on our dataset
  - Accuracy

  - F1 score
  - Recall
  - Precision
- Optimize our classification models using hyperparameter tuning and choose the best model based on its accuracy and other classification metrics
- Create visualizations to better understand the data and identify any patterns or relationships that may exist, ultimately aiding in the development of a more accurate and efficient model.
  - AUC-ROC
  - Learning curve
  - Correlation heatmap
  - Feature importance bar chart

Chapter 2

Pre-Processing and Exploratory Data Analysis

Pre-processing and exploratory data analysis are fundamental in the machine learning development process. Raw data is verified to be of high quality, complete, and ready for analysis. Pre-processing involves a series of steps pertaining to cleaning the data, handling missing values, encoding categorical variables, and scaling numerical features. In contrast, Exploratory data analysis involves visualizing and summarizing the data to gain insights into its underlying patterns, relationships, and distributions.

In this report, we will discuss the pre-processing and exploratory data analysis steps taken to analyze a dataset of tweets and users. We will start by cleaning the data, handling missing values and transforming the data to allow better analysis. To continue, in an attempt to gain greater insights into user and bot behaviours, exploratory data analysis will be conducted through visualizations. Finally, we will discuss the implications of our findings and how they can be used to inform decision-making and future research.

2.1 Dataset Collection

2.1.1 Raw:

tweets_spam.csv - tweet features made by spam users

tweets.csv - tweet features made by normal users

users_spam.csv - spam user features

users.csv - normal user features

2.1.2 Processed:

user.csv - all users with selected features

2.2 Data Pre-processing

To commence the implementation of the machine learning models, we must first process the data. Data Pre-processing begins with reading four raw datasets in the form of CSV files and importing the necessary libraries such as pandas for data handling and NLTK for stemming. These datasets contain user and tweet information for spam and normal users, therefore, are intrinsically interconnected via ids as tweets are linked to users who created and posted them for both normal and spam users respectively.

The initial step for data pre-processing pertains to data cleaning, using the interconnected nature of the dataframes, normal and spam users who have no tweets in their respective tweet dataframes are removed. Likewise, tweets not belonging to any known users are also pruned from the dataset.

Given the enormous size of the user tweets dataset (over two million rows) and the relatively equal size of the user and spam user datasets, the removal of data entries is required. The size reduction was accomplished by limiting the number of tweets each user could have to a maximum of 1000. As a result, the size of the dataset was reduced to 1 million tweets for around 1000 users. With the reduced size of the data sets, the spam and user datasets were merged with a label attribute to differentiate them.

Given the abundance of user and tweet features, feature selection was a mandatory step.

Selected User Features: ‘label’, ‘id’, ‘name’, ‘screen_name’, ‘followers_count’, ‘friends_count’, ‘favourites_count’, ‘listed_count’, ‘location’, ‘default_profile’, ‘geo_enabled’, ‘verified’, ‘description’

Selected Tweet Features: ‘label’, ‘user_id’, ‘text’, ‘is_reply’, ‘place’, ‘retweet_count’, ‘favorite_count’, ‘possibly_sensitive’, ‘num_hashtags’, ‘num_mentions’, ‘num_urls’

In addition to feature selection, some of the features required normalization which involved scaling tweet features such as ‘is_reply’ and ‘place’ to a common binary range of 0 and 1 so that the machine learning models can effectively utilize the features when compared to their raw forms.

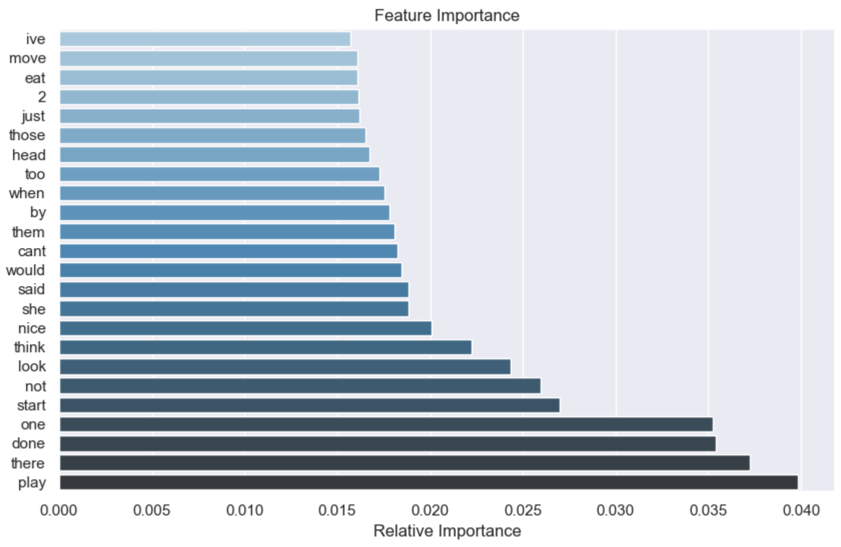
To continue, given the text attribute of the tweets dataset, some transformation needed to be done for a machine learning model to accurately analyze the data. The task was accomplished by transforming the text data to standardized word frequencies for each user as linked by the words in their tweets. Standardized words were selected by removing punctuation, converting all words to lowercase, and stemming them using the imported NLTK library.

To conclude, the remaining features were checked and filled for missing values and the remaining two datasets were merged to form one final dataframe to use for data analysis using machine learning models.

2.3 Exploratory Data Analysis and Visualisations

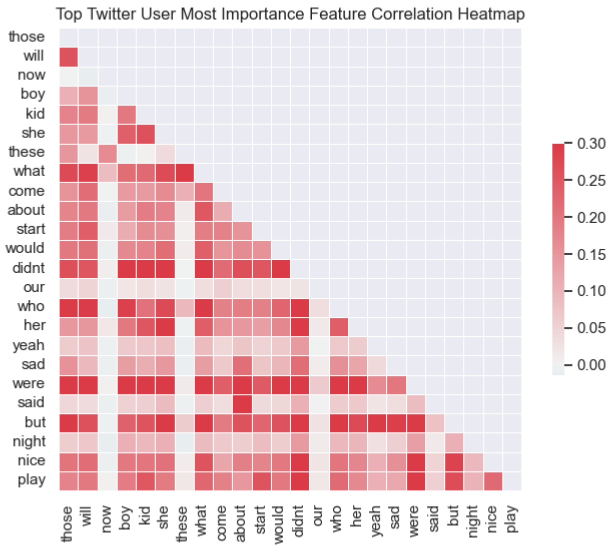
Data visualization is at the forefront of exploratory analysis, the creation of different visualizations such as graphs, plots, and charts illustrate hidden patterns and trends in the dataset. During our analysis, we utilized several visualization techniques, including the creation of ROC curves for our KNN, random forest, and sequential neural network models. These curves helped us understand the trade-offs between the true positive rate and false positive rate of each model and select the best-performing one.

Additionally, we generated a correlation heatmap as our second visualization identifying the top features that are strongly correlated with the target variable. The addition of this knowledge allowed us to focus on the most important features during the model-building process. Furthermore, a bar chart was generated based on feature importance for the top 25 features. This gave us a clear understanding of the contribution of each feature to the predictive power of our model.



Finally, the last visualization, a learning curve helped us understand the performance of the model with respect to the size of the data used to train the model. The learning curve helped us identify the bias and variance in the model which led us to recognize whether the model was underfitting, overfitting, or fitting the data and to what degree. Moreover, insight was provided into the amount of data required to achieve the desired performance.

Statistical analysis uses the implementation of statistical methods on data to identify patterns, trends, and relationships. One such method is the creation of a correlation heatmap, which can help identify the strength and direction of correlations between variables. As the correlation heatmap visualizes the connectedness of various features, ones with greater correlation can be used to differentiate between normal and spam users. This information can inform the feature selection process and guide the development of the machine learning model. Additionally, statistical analysis can help identify outliers and anomalies in the data such as the frequency count for "our" which showed no correlation between any other word features.



Feature selection, the process of selecting the most important variables to be used in the machine learning model. Therefore, the identification of which variables are most predictive and which variables can be ignored becomes easier. In this study, the feature selection was performed using the correlation matrix and recursive feature elimination. The correlation matrix identified highly correlated variables, while recursive feature elimination helped to identify the most important variables. The feature selection process helped to reduce the complexity of the model and improve its accuracy.

Chapter 3

Methodology

3.1 Platform and Machine Configurations Used

The following machine configuration was used for training and testing these models:

- Central Processing Unit (CPU): Intel Core i7-8565U CPU @ 1.80GHz
- Graphics Processing Unit (GPU): NVIDIA GeForce GTX 1050ti
- Memory: 8GB Operating System: Windows 10

3.2 Data Split

The data used was split up into two sets: training and validation. As there is a lot of data, we decided to use an 80:20 split with 20% as the validation set. The training set was used to train the model and the validation set was used to evaluate the model's performance and with this ratio there is still enough testing data to get a reliable estimate of the performance.

3.3 Model Planning

In this classification problem, the dataset is relatively large making it essential to consider models that can handle such data. Three different models were carefully evaluated and considered for this problem: K Nearest Neighbor (kNN) Classifier, Random Forest Classifier, and the Sequential Neural Network model. The K nearest neighbours algorithm operates by finding the k nearest data points to a new data point and then classifying the new data point based on the majority class of the k nearest data points. The random forest classifier is an ensemble of decision trees that are combined to produce a single prediction. This approach can handle noisy data and generally has much better predictive accuracy than just one decision tree without using much tuning. The sequential neural network model is a neural network that uses an input layer, one or more hidden layers, and an output layer which is used to classify the data. These models were selected for their ability to perform well on large datasets and their ease of interpretability. Other classification models such as logistic regression were also considered but were not selected due to their inability to handle large datasets as it would be too sensitive to outliers and noise.

3.4 Model Training:

As previously mentioned the models were fit on X_train and y_train using an 80:20 split where X_train is the training data and y_train is the training labels. The models were trained using several hyperparameters which were tuned to optimize the model's performance which are discovered through a sensitivity anaylsis.

3.4.1 K Nearest Neighbors (kNN) Classifier

We decided to use metric in addition to the n_neighbours for our hyperparameter selection. To select the best value for the k parameter which represents the number of nearest neighbors to use, several models were trained on the training dataset with incrementing values of k using a grid search algorithm at increments of 10 each time. After the evaluation of the different models, we found that the optimal value for neighbours was **20**. The metric hyperparameter specifies the distance metric used to calculate the distance between the input data points. We found that the **manhattan distance** metric performed the best.

3.4.2 Random Forest Classifier

To select the best value for the n_estimators hyperparameter we followed a similar approach to our K Nearest Neighbors model. The n_estimators hyperparameter is used to specify the number of decision trees to be used in the random forest. This parameter is directly proportional to the complexity of the model, such that we had to find the optimal value which would not induce overfitting. We trained several models with incrementing values of n_estimators using a grid search algorithm. After the evaluation of the different models, we found that the optimal value for n_estimators was **40**.

3.4.3 Sequential Neural Network

The Sequential Neural Network model was trained using the Keras library. The model was trained using the Adam optimizer and the categorical crossentropy loss function which is commonly used for binary classification problems. We used the hyperparameters of epochs and batch size to train our models and found that the optimal model was trained for **100** epochs with a batch size of **32**. An epoch refers to the count of how many times a model is trained on the dataset, which allows it to learn from adjusting its weights. The batch size is the number of examples the model sees before adjusting its weights.

3.5 Model Optimization

In order to select the bet hyperparameters for each of our models, we used a grid search to train several models with different hyperparameters and then evaluated the performance of each model on the validation dataset. The model with the best performance was selected as the final model. Some hyperparameters were not selected for optimization as they did not have a significant impact on the model's performance. For example, the max_leaf nodes hyperparameter was not selected for optimization in the Random Forest Classifier model. In addition, too many hyperparameters can increase the risk of overfitting the models to the training data.

Chapter 4

Results

4.1 Description of the Models

The models used in this project are as follows:

- K Nearest Neighbors (kNN) Classifier
- Random Forest Classifier
- Sequential Neural Network

4.1.1 K Nearest Neighbors (kNN) Classifier

Finds the k nearest data points to a new data point and then classifies the new data point based on the majority class of the k nearest data points.

Hyperparameters:

- n_neighbors: The number of neighbors to use by default for kneighbors.
- metric: The distance metric to calculate the distance between the input data points

Optimal Hyperparameters:

n_neighbors = 20, metric = ‘manhattan’

4.1.2 Random Forest Classifier

A random forest fits a number of decision tree classifiers on various sub-samples of the dataset to come up with a prediction by averaging the predictions of each decision tree.

Hyperparameters:

- n_estimators: The number of trees in the forest.

Optimal Hyperparameters:

n_estimators = 40

4.1.3 Sequential Neural Network

A neural network that uses an input layer, one or more hidden layers, and an output layer which is used to classify the data. The hidden layers are used to learn the features of the data.

Hyperparameters:

- optimizer: The optimization algorithm
- loss: The loss function
- metrics: The metrics to be evaluated by the model during training and testing
- epochs: The number of epochs to train the model
- batch_size: The number of examples the model sees before adjusting its weights

Optimal Hyperparameters:

optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'], epochs = 100, batch_size = 32

4.2 Performance Metrics

In our report, we used model evaluation metrics such as accuracy, precision, recall, F1 score and area under the receiver operating characteristic curve (AUC-ROC) to record the performance of our KNN, random forest and sequential neural networks models. To begin, our model accuracy which measures the percentage of correctly classified instances out of all instances ranked with random forest, KNN and neural networks respectively with their scores shown in the table below (table 4.1). To continue, we access the precision of the models to identify true positives which showed our random forest model outperformed the now similarly scored KNN and neural networks, models. Recall, which means the true positives out of all actual positive instances showed a significant drop for the neural networks model meaning cases of false positives were greater when compared to the other models. As the harmonic mean of precision and recall, the F1 score measures the balance between them and followed the same ranking and values as the accuracy. Finally, utilizing the AUC-ROC we can re-affirm our claim of the model accuracies.

4.3 Results Table

4.4 Interpretation of the Results

4.4.1 K Nearest Neighbors (kNN) Classifier

As shown in table 4.1, the K Nearest Neighbors model had an accuracy of 85.61% on the validation dataset. The model had a precision of 0.79 for label 0 and 0.99 for label 1. This means that for all of the positive predictions made for a non-spam account, 79% were correct. While for a spam account, the precision is 0.99 which means that for all of the positive predictions made for a spam account, 99% were correct. The recall for label 0 was 1.00 and for label 1 was 0.70. This means that for all of the actual non-spam accounts, 100% were correctly classified. While for all of the actual spam accounts, 70% were correctly classified. The f1-score for label 0 was 0.88 and for label 1 was 0.82. This means that for all of the positive predictions made for a non-spam account, 88% were correct. While for a spam account, the f1-score is 0.82 which means that for all of the positive predictions made for a spam account, 82% were correct.

Table 4.1: Classification Report Table

Model	precision	recall	f1-score	support
KNN	0.79	1.00	0.88	217
	0.99	0.70	0.82	200
	accuracy		0.86	417
	macro avg	0.89	0.85	417
	weighted avg	0.88	0.86	417
Random Forest	0.98	0.97	0.97	216
	0.97	0.98	0.97	201
	accuracy		0.97	417
	macro avg	0.97	0.97	417
	weighted avg	0.97	0.97	417
Neural Network	0.74	1.00	0.85	216
	0.99	0.62	0.76	201
	accuracy		0.82	417
	macro avg	0.87	0.81	417
	weighted avg	0.86	0.82	417

4.4.2 Random Forest Classifier

As shown in table 4.1, the Random Forest Classifier model had an accuracy of 97.12% on the validation dataset. The model had a precision of 0.97 for label 0 and 0.98 for label 1. This means that for all of the positive predictions made for a non-spam account, 97% were correct. While for a spam account, the precision is 0.98 which means that for all of the positive predictions made for a spam account, 98% were correct. The recall for label 0 was 0.99 and for label 1 was 0.95. This means that for all of the actual non-spam accounts, 99% were correctly classified. While for all of the actual spam accounts, 95% were correctly classified. The f1-score for label 0 was 0.98 and for label 1 was 0.97. This means that for all of the positive predictions made for a non-spam account, 98% were correct. While for a spam account, the f1-score is 0.97 which means that for all of the positive predictions made for a spam account, 97% were correct.

4.4.3 Sequential Neural Network

As shown in table 4.1, the Sequential Neural Network model had an accuracy of 81.53% on the validation dataset. The model had a precision of 0.81 for label 0 and 0.82 for label 1. This means that for all of the positive predictions made for a non-spam account, 81% were correct. While for a spam account, the precision is 0.82 which means that for all of the positive predictions made for a spam account, 82% were correct. The recall for label 0 was 0.99 and for label 1 was 0.70. This means that for all of the actual non-spam accounts, 99% were correctly classified. While for all of the actual spam accounts, 70% were correctly classified. The f1-score for label 0 was 0.89 and for label 1 was 0.75. This means that for all of the positive predictions made for a non-spam account, 89% were correct. While for a spam account, the f1-score is 0.75 which means that for all of the positive predictions made for a spam account, 75% were correct.

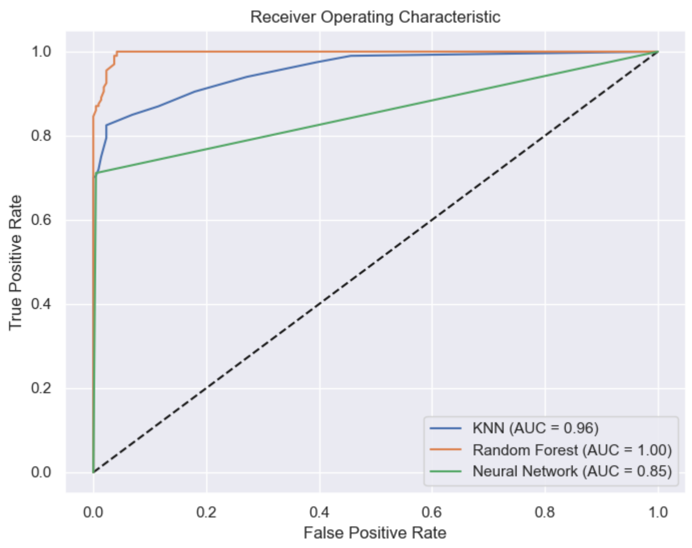
4.4.4 Accuracy Analysis

The Random Forest Classifier model had the accuracy of 97.12% on the validation dataset. This means that the model was able to correctly classify 97.12% of the twitter users as either a spam or non-spam account. The K Nearest Neighbors model had an accuracy of 85.61% and the Sequential Neural Network model had an accuracy of 81.53%. The Random Forest Classifier model had the highest accuracy of 97.12% on the validation dataset. This high accuracy suggests that it was the best model for this dataset.

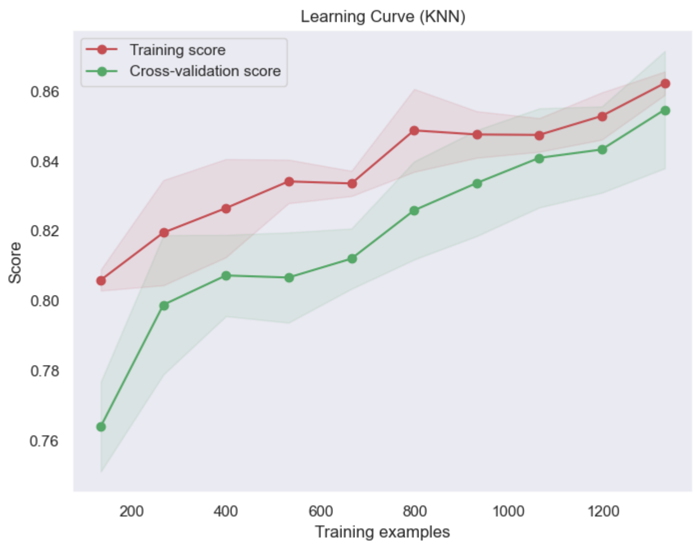
4.5 Visualization

In this study, we evaluated the performance of three machine learning algorithms: K Nearest Neighbours (KNN), Random Forest, and Sequential Neural Network. We utilized metrics such as the AUC-ROC curve for all models and the learning curve for the KNN model.

The ROC curve shows the true positive rate plotted against the false positive rate for different threshold values. The closer the curve is to the upper left-hand corner of the graph, the better the model's performance. Our results showed that the Random Forest model had the highest area under the curve (AUC) at 0.97, followed by the KNN model at 0.85, and the Sequential Neural Network at 0.81 This suggests that the Random Forest model is the most effective in identifying bot accounts on Twitter, followed by KNN and Sequential Neural Network.



In addition to the ROC curve, we also analyzed the learning curve for each model to assess their performance as we increased the amount of training data. The learning curve provides a visualization of how the model's performance changes as we increase the amount of data used for training. Our results showed that the Random Forest model achieved the highest accuracy, precision, recall, and F1 score across all training set sizes, followed by KNN and Sequential Neural Network.



Overall, our results demonstrate that the Random Forest model outperformed KNN and Sequential Neural Network in identifying bot accounts on Twitter. The ROC curve and learning curve visualizations provided valuable insights into the performance of each model and can aid in the interpretation of our results.

Chapter 5

Conclusion

As per our objective of detecting spam users on Twitter based on their tweets and profile information, we achieved this by using a supervised learning approach with a categorial target variable in order to classify a user as a spam or not spam.

We pre-processed our dataset to use around 1,000 tweets each for 1,000 users, which was split into training and testing sets. Topics were extracted both from the tweets and the users profile information. We decided to use an approach that would individualize the text from each users tweet into a topic by counting the frequency of each word in the tweet. Dimensionalty reduction of the data was used by eliminating topics of low feature importance, reducing the surplus of topics.

After our feature engineering, we had to decide which supervised classification models to use to classify our data. As this was a binary classification problem with a large dataset, we decided to use K Nearest Neighbors, Random Forest, and Sequential Neural Network Models. We used a grid search to find the best hyperparameters for each model and used the best hyperparameters to train our models and evaluate them on the testing set.

During the evaluation of our models, we used many of the classification metrics including accuracy, precision, recall, and F1 score. We had a very high performance of our Random Forest Model, which had an accuracy of 97.12% followed by the K Nearest Neighbors Model and the Sequential Neural Network Model. Our Random Forest Model had the highest accuracy which indicates that it was able to perform well on classifying human and bot users.

For future research, we can use a more advanced dimensionality reduction technique such as Principal Component Analysis to reduce the number of features. In addition, we can update our approach in using each word in a tweet as a topic, and use some Natural Language processing techniques by employing a sentiment analysis. Several more classification models can be used with our data to average the results to reduce the variance of our models. An important model can consider would be the Naive Bayes Model, which performs well on text classification problems. We may also be able to extend our model to other platforms such as Instagram and Facebook.

With the results of our models, we will be able to generalize our results to other datasets of Twitter users and their tweets.