Alexis Navarro, Diogo Meisser, Daniel Mardani

Dr. Sharmistha Guha

Statistical Computing with R and Python

01 May 2025

Detecting Twitter Bots

Introduction

At this current point in time, technological advances have been instrumental in leading the way our world operates. From something as simple as changing how individuals order their coffee from Starbucks to something much more complex, such as a self-driving car. These advances have been consequential in maximizing efficiency in today's modern age. As a result, individuals argue that if these things are possible, then how far is society away from actual robots dominating our world? While this is often seen in fictitious films, this may be the reality today, but not in the way we think.

In this paper, the intention is to detect Twitter bots using four types of classification models: Logistic Regression with Lasso, Linear Discriminant Analysis (LDA), Random Forest, and K-Nearest Neighbors (KNN). This paper explores what bots are, their intentions, and why they are important. Regarding the dataset, we will perform background research, cleaning, preprocessing, and exploration to gain a deeper understanding. The methodology section outlines the process of selecting relevant features, considering potential transformations of predictors, splitting the dataset into training and testing sets, implementing key classification models using R, and evaluating model performance through standard metrics. Further, the results of the four

classification models are presented, followed by a comparison of their performance and a discussion of limitations.

Bots: Definition, Intentions, and Significance

According to the United States Department of Homeland Security (DHS), social media bots are defined as "programs that vary in size depending on their function, capability, and design; and can be used on social media platforms to do various useful and malicious tasks while simulating human behavior" (Department of Homeland Security). Social media bots are used to drive false engagement, phishing, influence elections and manipulate stocks/cryptocurrencies. Social media has grown to become the most important tool available to shape public opinion and influence markets. With social media bots added to the equation, detecting bots is critical in ensuring integrity and safety from mischievous tasks.

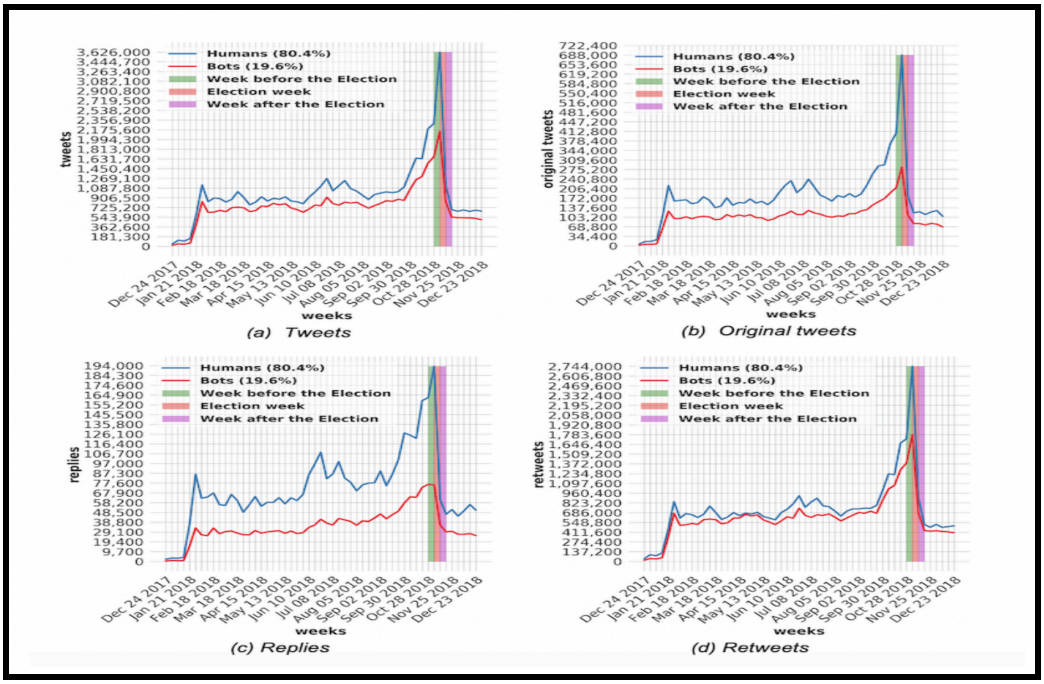
Bots & Social Media Applications

On X, formerly called Twitter, 15% of accounts–roughly 48 million– were likely bots (Rodriguez-Ruiz et al.). More specifically, 66% of links provided regarding news and current events were shared by bots (Pew Research Center). For example, in the 2018 elections in Italy, bot networks amplified right-wing populist content and divisive hashtags (Radicioni et al.). In total, 66% of accounts that tweeted links across a variety of domains — including sports, commercial products, adult content, and even Twitter itself — were identified as automated accounts. These findings underscore the widespread influence of automated accounts across different areas of online discourse, shaping not only political narratives but also everyday information shared on the platform.

Bots: Case Study

Using the 2018 United States Midterm Elections, a determination can be made between human and bot activity dynamics. In Figure 1, the number of tweets among human and bot accounts is shown for each week for a total of 18 weeks. In this study, human accounts represent 80% of accounts while 20% correspond to bot accounts (Luceri, Cardoso, and Giordano). Over this period, bot account activity closely followed human account activity even during the week before the election, when tweets skyrocketed. This parallel trend suggests that bots were strategically designed to mimic human behavior, maintaining consistent activity patterns even during critical moments to spread manipulative narratives and affect beliefs.

Figure 1: Activity During the Midterm Year



Data Background

The data used for this project was provided by Stefano Cresci and is titled *"The Paradigm-Shift of Social Spambots: Evidence, Theories, Tools for the Arms Race"* (2017). The dataset contains multiple folders representing different types of bot activity: social spambots, traditional spambots, and fake followers, with the distinctions between them illustrated in Figure 2 below. Additionally, the dataset includes a folder of genuine (human) accounts. Each folder contains both user-level information and corresponding tweet-level data.

Figure 2: Cresci Dataset information

group name	description	accounts	tweets	year
genuine accounts	verified accounts that are human-operated	3,474	8,377,522	2011
social spambots #1	retweeters of an Italian political candidate	991	1,610,176	2012
social spambots #2	spammers of paid apps for mobile devices	3,457	428,542	2014
social spambots #3	spammers of products on sale at Amazon.com	464	1,418,626	2011
traditional spambots #1	training set of spammers used by C. Yang, R. Harkreader, and G. Gu.	1,000	145,094	2009
traditional spambots #2	spammers of scam URLs	100	74,957	2014
traditional spambots #3	automated accounts spamming job offers	433	5,794,931	2013
traditional spambots #4	another group of automated accounts spamming job offers	1,128	133,311	2009
fake followers	simple accounts that inflate the number of followers of another account	3,351	196,027	2012

For this study, we focused specifically on the Social Spambots #1 folder, which contains data on bot accounts that retweeted an Italian political candidate. This focus was chosen to maintain relevance to political discourse, where manipulation through automated accounts can be particularly detrimental. We also used the Genuine Accounts folder, which included human-operated accounts verified through crowdsensing methods.

Both datasets included 25 columns of tweet data and 42 columns of user data. The Genuine Accounts dataset consisted of 3,474 users and approximately 8,377,522 tweets, while

the Social Spambots #1 dataset contained 991 users and 1,610,176 tweets. However, due to privacy and data protection constraints, only a sample of 2,839,362 genuine tweets was made available for analysis.

User-level data included information such as:

- Basic Account Info (e.g., screen_name),
- Activity Metrics (e.g., followers_count),
- Profile Appearance (e.g., profile_image_url),
- Account Settings (e.g., geo_enabled),
- Privacy & Verification (e.g., verified),
- Timestamps (e.g., created_at),
- Miscellaneous Fields (e.g., test_set_1).

Tweet-level data captured:

- Tweet Content (e.g., text),
- Tweet Metadata (e.g., source),
- User References (e.g., user_id),
- Reply and Retweet Information (e.g., in_reply_to_status_id),
- Engagement Metrics (e.g., retweet_count),
- Content Flags (e.g., possibly_sensitive),
- Hashtag, URL, and Mention Counts (e.g., num_hashtags),
- Timestamps (e.g., created_at).

Together, these two datasets provided the necessary information to analyze bot and human behaviors and build a classification model to predict the likelihood of an account being automated or human-operated based on various behavioral and profile features.

Data Cleaning and Preprocessing

To prepare Cresci's data for analysis, we first assigned binary labels to each account using a dummy variable, where 1 indicated a bot and 0 indicated a human-operated (genuine) account. Next, we cleaned the datasets by removing irrelevant or unusable features that did not contribute meaningful insights into user behavior. For example, columns such as

profile_image_url, background_image_color, and background_image_url were dropped. This pruning ensured the datasets focused on attributes that could inform patterns in bot and human behavior.

We then merged the corresponding user and tweet datasets separately, resulting in one consolidated user dataset and one tweet dataset. To integrate the information from both, we employed feature engineering techniques to create new variables that aggregated tweet-level data by user_id. These included features such as avg_favorite_count, avg_num_urls, and avg_text_length, among others.

Using a left join on user_id from the tweet dataset and id from the user dataset, we created a single merged dataset. However, due to the limited availability of genuine tweet data (only a sample of 2,839,362 tweets), we excluded any users without corresponding aggregated tweet information, resulting in a reduced number of observations.

Finally, we addressed the remaining missing values. For instance, in the default_profile feature, null values were interpreted as 0, assuming a non-default profile where data was absent. The resulting final dataset consisted of 2,074 observations and 24 features. Figure 2 displays a sample of five observations across six features to illustrate the structure of the dataset used in our analysis.

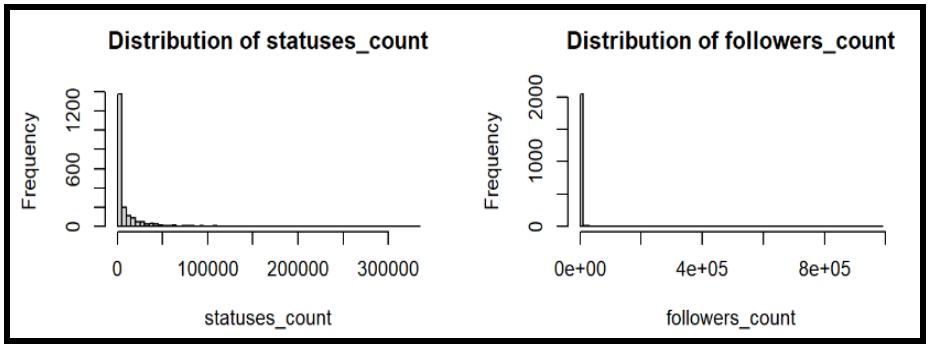
Figure 2: Sample Twitter Data

followers_count	geo_enabled	account_age_days	percent_with_hashtags	avg_text_length	label
2248	1	5323	0.0919255	67.48367	0
641	0	4186	0.1777434	67.22562	0
1042	1	4206	0.2607874	88.21039	0
337	1	4239	0.3714641	94.52751	0
1948	0	5891	0.3582788	109.06739	0

Data Exploration

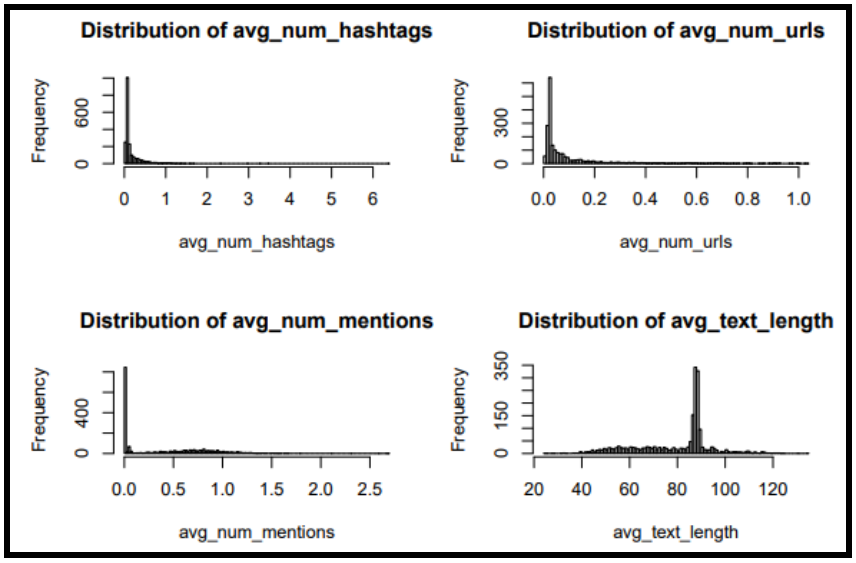
Based on the aggregated data, there is an even class split with 52% of the data consisting of genuine users and 49% comprising social bots. Upon inspection of the number of tweets (statuses_count) and retweets by the user and number of Twitter followers (followers_count) predictors, the distribution and summary output indicates the existence of extreme values. Figure 3 provides a visual and numeric representation of these extreme values. As for the continuous predictors, the normality assumption seems to be violated for the number of tweets (tweet_count), the average number of hashtags (avg_num_hashtags), the average number of URLs (avg_num_urls), the average number of mentions (avg_num_mentions), and average text length (avg_text_length). Figure 4 details this assumption in a plot for these 5 predictors. Class differences in predictions between human and bot means, medians, and standard deviations were explored as shown in Figure 5. Additionally, there is a strong presence of collinearity among some predictors as shown in Figure 6.

Figure 3: Presence of Extreme Values




```
> summary(bot$statuses_count)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      3     138    1208   10950    9553  333504
> summary(bot$followers_count)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
     0.0     52.2    346.0   2038.8   1439.0 986837.0
```

Figure 4: Normality Assumption



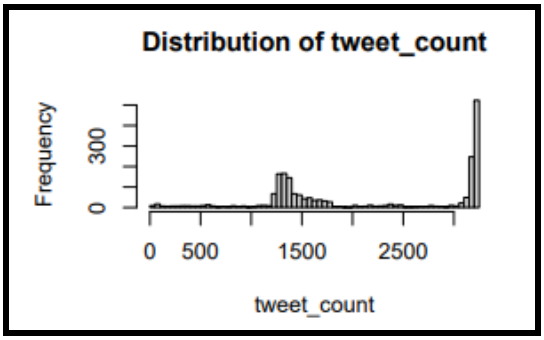
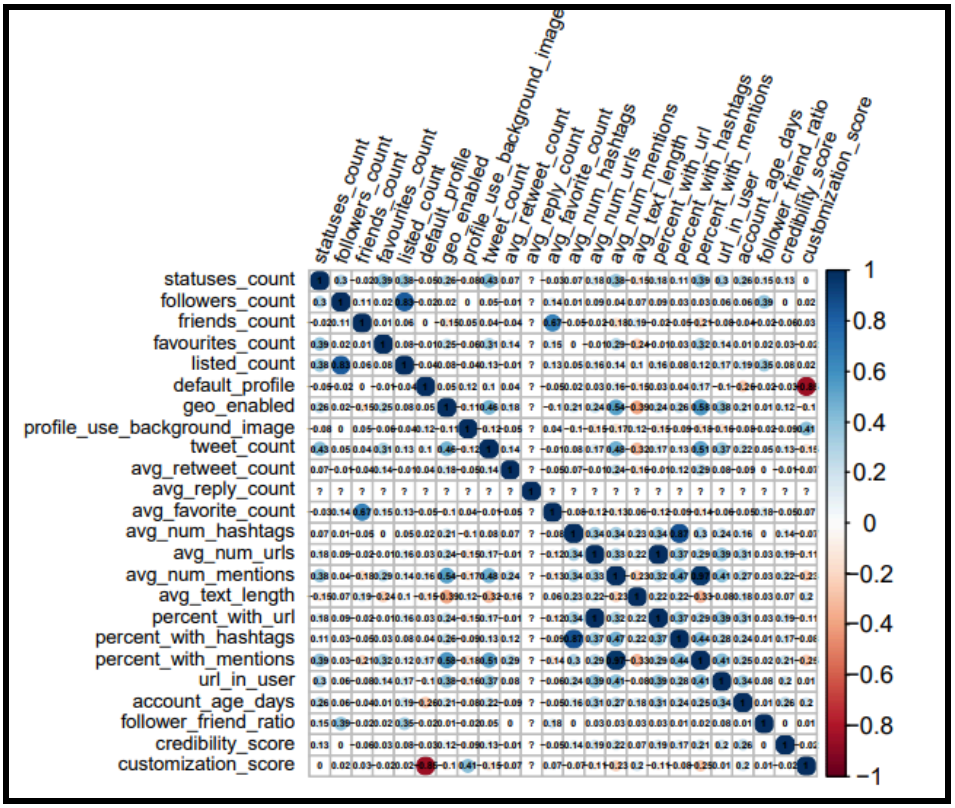


Figure 5: Class Differences

Feature	Mean (Bot)	Mean (Human)	Median (Bot)	Median (Human)	SD (Bot)	SD (Human)
statuses_count	1111.999	19 951.971	138.000	8272.000	5740.507	33 654.668
followers_count	1784.907	2271.072	234.000	407.000	3217.002	30 394.702
friends_count	1853.769	676.940	313.000	366.000	3111.164	1277.074
favourites_count	158.172	4676.110	0.000	1492.000	2160.885	10 223.234
listed_count	3.955	32.500	0.000	3.000	23.658	194.203
default_profile	0.146	0.257	0.000	0.000	0.354	0.437
geo_enabled	0.018	0.643	0.000	1.000	0.134	0.479
profile_use_background_image	0.997	0.895	1.000	1.000	0.055	0.307
tweet_count	1624.656	2621.755	1387.000	3187.000	586.302	1002.075
avg_retweet_count	8.442	829.525	2.035	203.342	99.059	2276.434
avg_reply_count	0.000	0.000	0.000	0.000	0.000	0.000
avg_favorite_count	1.111	0.435	0.438	0.225	2.593	1.295
avg_num_hashtags	0.088	0.268	0.076	0.158	0.105	0.365
avg_num_urls	0.045	0.161	0.024	0.087	0.116	0.184
avg_num_mentions	0.028	0.763	0.008	0.755	0.095	0.320
avg_text_length	87.742	71.989	87.859	69.854	4.734	19.090
percent_with_url	0.045	0.158	0.024	0.086	0.115	0.181
percent_with_hashtags	0.057	0.169	0.051	0.118	0.060	0.160
percent_with_mentions	0.022	0.583	0.006	0.605	0.071	0.192
url_in_user	0.022	0.417	0.000	0.000	0.147	0.493
account_age_days	4755.631	5056.053	4835.000	4818.000	224.054	715.465
follower_friend_ratio	3.424	4.185	0.802	1.044	67.391	32.168
credibility_score	0.003	0.018	0.000	0.009	0.045	0.023
customization_score	0.925	0.819	1.000	1.000	0.178	0.240

Table 1: Comparison of bot and human account metrics (mean, median, and standard deviation).

Figure 6: Collinearity



Methodology

This section will discuss the feature selection, data transformation, and outlier removal considerations, data split and validation sets, classification methods, and performance metrics.

Feature selection was conducted manually based on the data exploration. 18 features in total were selected and predictors that were not informative like avg_reply_count and those with high collinearity(collinearity above 0.8) being percent_with_url, percent_with_mentions, percent_with_hashtags, default_profile, and listed_count were removed. The cutoff for collinearity was chosen to allow for using a large number of features for prediction that do not suffer from an excessive amount of collinearity with other predictors to improve the results of classification methods that are impacted by collinearity(i.e. LDA and Logistic Regression).

Log and Box-cox data transformation were considered to whether approximating the predictors' distribution to normality for particular classification methods was appropriate. Neither log or Box-cox transforms were implemented due to the issue of justifying the value of the arbitrary constant in the case of $\lambda = 0$ or the log transform when it came to the count predictors. Based on the way the data was originally collected, there wasn't a good justification for removing outliers in the data for extreme values due to the author's manual verification of the data.

The data was split into training and test data so that a classification model could be trained using the training data and the test data was used to check if the model is overfit to the training data. For models with hyperparameters like K-NN and LASSO Logistic Regression, 5-fold Cross-validation was implemented to test each of the hyperparameters and determine which

produces the lowest cross-validation estimate. Conducting a 5-fold CV involves the training data being further divided into five equal-sized folds and five different models are fit with a different fold being left out of the training data as a validation set and the model is trained using the other 4 folds. In each of these models, the test error of the validation set is computed and the test errors across all of the models are averaged to compute the CV estimate which is used as a measure of model performance for a particular parameter. The parameter whose CV estimate is the lowest is selected for comparison with the other classification methods. A 5-fold CV was chosen over LOOCV and a CV with a higher number of folds, to use less identical observations resulting in a lower variance regarding the CV estimate despite the increase in bias due to using less of the training data.

The LASSO Logistic regression model was considered for the binary classification of users into either human or bot users. In the multiple logistic regression model we assume that there are two classes(0 =human and 1 = bot) as the binary response, y_i, has entries that are random variables distributed by Bernoulli r.v., uses all of the 18 predictors, X_1,.., X_{18} and the log odds of π_i(the probability of the *i*th user being a bot) is related to a linear combination of the predictors such that $\log(\frac{\pi_i}{1-\pi_i}) = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_{18} x_{18i}$, and the observations are independent.

Parameters in the Logistic Regression model are estimated by maximum likelihood as the Likelihood function of the responses is:

- $f(y_{VEC}; \pi = (\pi_1, .., \pi_n)) = \prod_{i=1}^{n} f(y_i; \pi) = \prod_{i=1}^{n} (\pi_i)^{y_i}(1 - \pi_i)^{1-y_i}$

Using the relationship between the log odds of each π_i we can find a suitable expression for the likelihood function $L(\beta_0,..,\beta_{18})$ with respect to $\beta_0,..,\beta_{18}$. Due to there not being a closed-form expression for the MLEs of $\hat{\beta}_0,..,\hat{\beta}_{18}$ we must consider numerical methods like Newton Raphson or another similar iterative process that will try to find the maximum likelihood of the $L(\beta_0,..,\beta_{18})$ for the set of parameters $\beta_0,..,\beta_{18}$.

The LASSO(Least Absolute Shrinkage and Selection Operator) is considered in the Logistic Regression model for the purposes of performing variable selection and reducing the variance with a small cost of making the model estimates more biased. In the case of logistic regression, the L1 penalty term, $\lambda \sum_{i=1}^{18} |\beta_i|$,(notably with hyperparameter λ) will be added to some logistic loss function, and by minimizing this function estimates of $\beta_0,..,\beta_{18}$ are obtained for the LASSO logistic regression model. Using the cv.glmnet() function(part of GLMNET package) the coefficients in the LASSO LR model for a grid of values for different lambda(hyperparameters) can be found. Further, the function allows for the computation of a 5-fold CV estimate in order to select the desirable hyperparameter lambda. The lambda which produces the smallest 5-fold CV estimate is chosen to represent the LASSO Logistic Regression Model for comparison with the other classification techniques.

Linear Discriminant Analysis or LDA was another classification technique considered for comparison. LDA is a parametric classification method that attempts to estimate the posterior class probabilities by making certain model assumptions regarding the density function, $f_k(x)$,

as for two classes we have that:

$$p_k(x) = P(Y = k|X) = \frac{\pi_k f_k(x)}{\sum_{i=1}^{2} \pi_i f_i(x)} \text{ for } k = 1, 2 ,$$

The observation is then classified to either group 1(human users) or group 2(bot users) based on whether $p_2(x)$ is greater than some threshold value(if it is greater it is a bot user and if not it is a human user). For our purposes, we consider using the default threshold that corresponds with the Bayes decision boundary 0.5. First observations from each class are drawn from a multivariate Gaussian distribution with a class-specific mean vector(μ_k) and a common covariance matrix(Σ). These assumptions lead to the density function of X for an observation(x_{VEC}) that comes from the kth class to be of the matrix form:

$f(x_{VEC}; \mu_k, \Sigma) = \frac{1}{(2\pi)^{18/2}|\Sigma|^{1/2}} e^{(\frac{1}{2}(x_{VEC}-\mu_k)^T \Sigma^{-1}(x_{VEC}-\mu_k))}$. The prior probabilities π_k are generally estimated by using the class proportions of the (training) data. The lda() function in the MASS package provides a means of estimating the class probabilities $p_1(x)$ and $p_2(x)$ by using the assumptions of LDA and estimating the prior probabilities of the classes by using class proportions. Based on the data exploration it may seem unwise to use LDA as we would expect it to perform quite poorly to other methods based on how its key distribution assumptions are violated, however, it may be of interest to see how much worse it performs to other methods with its assumptions violated.

Random Forest is a nonparametric method that was considered for the purposes of binary classification. For the Tree-Based classification method of Random forests, it uses classification

trees that partition the feature space into regions based on measures (like the Gini Index that measures node purity) which indicate whether a particular partition is well suited for the decision tree. Random Forests builds a number of decision trees on bootstrapped(random sampling with replacement) training samples from one random selection of $m < p$ predictors. The randomForest() function part of the randomForest package allows for the creation of random forests that utilize bootstrapping and random feature selection and create decision rules based on the Gini Index to measure the total variance across each class. Arbitrarily we choose $m = floor(p^{0.5})$ in implementing the Random Forest. Arguably Random Forests seems to be the most suitable method based on the data exploration as it is less sensitive to the outliers and less prone to multicollinearity due to the randomness introduced through the bootstrap and random feature selection.

K-NN or K-nearest neighbors is the final classification method considered. K-NN works by using all of the predictors of a particular observation and computing a distance measure involving different observation predictors. Where in our case, we were interested in the K nearest neighbors that are closest to a particular observation based on Euclidean distance. The class of the observation is then assigned based on the K-NN classifier which uses the known classification of the K closest points(represented as set N_0) by using the indicator function:

- K-NN Classifier= $\max_{j=\{0,1\}} \{\frac{1}{K} \sum_{i \in N_0} I(y_i = j)\}$

Due to the presence of extreme values, the predictors were scaled using the scale() function of the class R package to improve the predictive power of the method. The tuning parameter K

was tested for values $K = \{1, 2,..., 100\}$ via 5-fold CV using the train(), trainControl() functions of the R package caret. K-NN was chosen primarily for how it classifies observations based on whether it is similar to the classes of data points that are nearby.

Model performance was evaluated based on general classification performance metrics like accuracy(correct classifications/total classifications), specificity(correct classifications for humans/total number of actual humans), and sensitivity(correct classification for bots/ total number of actual bots). When possible the ROC curve is also displayed, as to consider the AUC.

Results

Upon the implementation of the Logistic Regression with the LASSO model, this model yielded a 97.6% accuracy rate, with 405 observations correctly classified as bot or human accounts and only 10 observations incorrectly classified. The model's sensitivity was 97.8% with a specificity of 97.4%, meaning this model does a great job at catching bots and avoiding misclassification. As for the coefficients of this model, there are two types: a positive coefficient and a negative coefficient. The most positive coefficient is the use of a profile image, indicating the increased log odds of this account being a human. The most negative coefficient is the average number of mentions, indicating the decreased log odds of this account being a bot.

Similar to the Logistic Regression with the LASSO model, the LDA model had a 97.1% accuracy rate, with 403 observations being classified correctly as either bots or humans while only 12 were incorrectly classified. The LDA's sensitivity and specificity were 96.9% and 97.4%, indicating an excellent model. When implementing the Random Forest model, the

accuracy rate was 98.3% with a sensitivity of 99.6% and a specificity of 96.8%, a conclusion very similar to the other two models. This model correctly classified 408 observations while only misclassifying 7 observations, far better than the other models at the margin.

Using a value of K=5 for K–Nearest Neighbors, this model correctly classified 405 observations as either a bot or human account, while only misclassifying 10 observations as a bot or human account. The accuracy rate was 97.6%, the sensitivity rate was 99.4%, and the specificity rate was 96.1%. In Figure 7, the Cross-Validation accuracy rate is portrayed among K values 0-100. As we compare these four models, it is clear that the performance of these models is quite good at achieving a high accuracy rate. However, it is hard to determine which classification model performs better, as it is likely sensitive to the predictors chosen during the feature selection process. Figure 8 compares these four models with respect to accuracy, sensitivity, and specificity.

Figure 7: Cross-Validation Plot

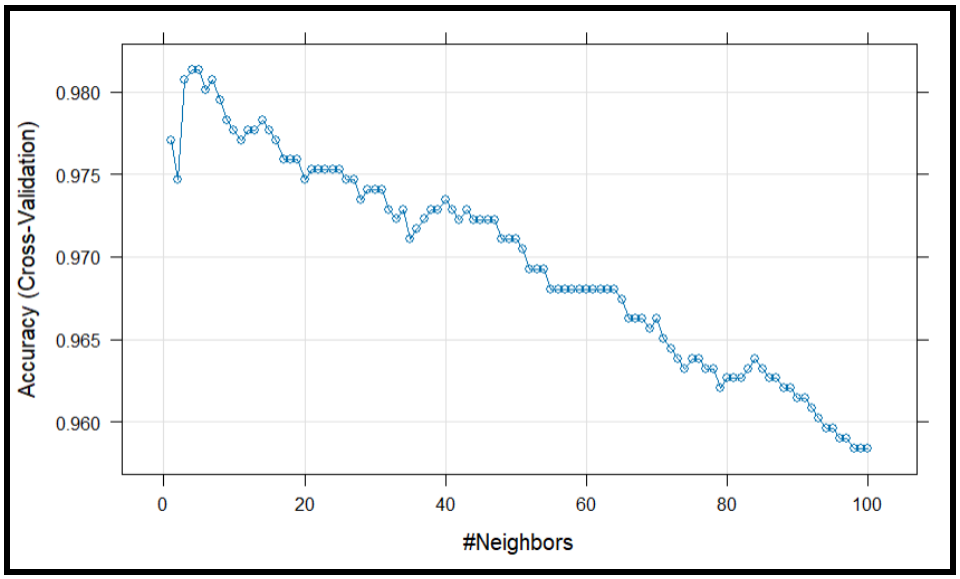


Figure 8: Classification Performance Comparison

Table 2: Classification Performance Comparison

Model	Accuracy	Sensitivity	Specificity
Logistic Regression	0.971	0.969	0.974
LDA	0.971	0.969	0.974
Random Forest	0.983	0.996	0.968
K-Nearest Neighbors	0.9759	0.9945	0.9614

Conclusion

It is evident that these four models work exceptionally well at classifying human accounts and bot accounts as evidenced by the high accuracy rate of those models, respectively. However, these models may prove to be more accurate by modifying our analysis. For one, the original dataset suffered from sampling bias, where it is likely the result of the author choosing to verify genuine users who retweet often. This dataset classifies bot behavior for a specific type of bot, namely one that existed during the 2014 Mayoral Rome elections that aided a particular candidate. This makes it difficult to apply results to other types of social bots based on different behavior patterns. Further, this dataset is heavily skewed and has a large number of outliers. Additionally, it is quite difficult to determine the importance of each feature for each model, which may have improved classification results.

References

Cresci, Stefano, Roberto Di Pietro, Maurizio Petrocchi, Angelo Spognardi, and Maurizio Tesconi. "The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race." *Proceedings of the 26th International Conference on World Wide Web Companion*, ACM, Apr. 2017, pp. 963–972, https://botometer.osome.iu.edu/bot-repository/datasets.html

Cybersecurity and Infrastructure Security Agency. *Social Media Bots Overview*. U.S. Department of Homeland Security, 2017, https://niccs.cisa.gov/sites/default/files/documents/pdf/ncsam_socialmediabotsoverview_508.pdf

Rodríguez-Ruiz, Jesús, et al. *A One-Class Classification Approach for Bot Detection on Twitter*. Computers & Security, Elsevier, 2020, https://www.sciencedirect.com/science/article/abs/pii/S0167404820300031

Pew Research Center. *Bots in the Twittersphere*. Pew Research Center, 2018, https://www.pewresearch.org/internet/2018/04/09/bots-in-the-twittersphere/.

Radicioni, Tommaso, et al. *Analysing Twitter Semantic Networks: The Case of 2018 Italian Elections. Scientific Reports*, 24 June 2021, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8225802/.

Appendix

Diogo Meisser - Developed the code, contributed to the presentation, and co-authored the paper

Daniel Mardani - Developed the code, contributed to the presentation, and co-authored the paper

Alexis Navarro - Contributed to the presentation and co-authored the paper.