Detecting Automated vs. Non-Automated Instagram Accounts

Emilio Sierra, Leander Herman, Alex Thompson

May 15, 2024

Abstract

This project was created to use multiple machine learning models to help determine if an Instagram account is Automated or not. We wanted to create and find a model that will be able to make these decisions in the most optimal way. We started with and compared four models including, Decision tree, Random Forest, SVM, and a Neural Network. We tested hyper parameters for each model in order to figure out how we could get the most accurate predictions. We then compared the accuracy of each model in order to determine our best model for predicting if an account is automated.

Contents

1 Introduction

With the rise of social media networks, more and more automated accounts are being created to simulate fake engagement and promote user accounts. Platforms like Instagram are at the forefront of shaping public discourse, influencing market trends, and fostering social interactions, so the presence of these ”bots” can manipulate engagement metrics, spread misinformation, and affect the integrity of user interactions.

This report focuses on the detection automated accounts on Instagram, using advanced machine learning techniques to distinguish between non-automated and automated accounts.

Our experiments use a range of different machine learning models. which include Decision Trees, Random Forest, SVMs, and Neural Networks. These were selected based on prior successes tested on classification dataset related to ours.

Our experiments involved a setup using a fixed training percentage and random seed across all models to ensure comparability. Our runs across all models showed similar performance with different hyper-parameters being used to test. This report will detail these findings, offering a comprehensive analysis of each model's strengths and limitations as observed across different datasets.

2 Background and Related Work

To ensure a thorough understanding of this report on detecting automated accounts on Instagram, it's critical to grasp some concepts. We'll being using different types of supervised learning methods to predict class labels, which these methods create models that categorize data into predefined classes. We'll also be splitting the dataset between a training set and a testing set, which then is used to test performance. Finally, there's a key distinction between ”fake” and ”automated” accounts. Fake accounts are accounts that are made by other real people but aren't active or automated in engagement. Automated accounts are accounts that are mass created and used as engagement farms for users to increase engagement.

In some prior, relevant work, there have been varied approaches comparing different algorithms to detect these automated accounts. One recent report entailed testing not only on Instagram, but across multiple social media platforms, with machine learning methods that involve supervised, semi-supervised, and nonsupervised learning. The experiments take into account different types of features like user behavior patterns, posting frequency, and engagement metrics.

In more recent studies, experiments start using CNN algorithms to detect more of the media side of things when it comes to automated accounts with an accuracy score of 92%

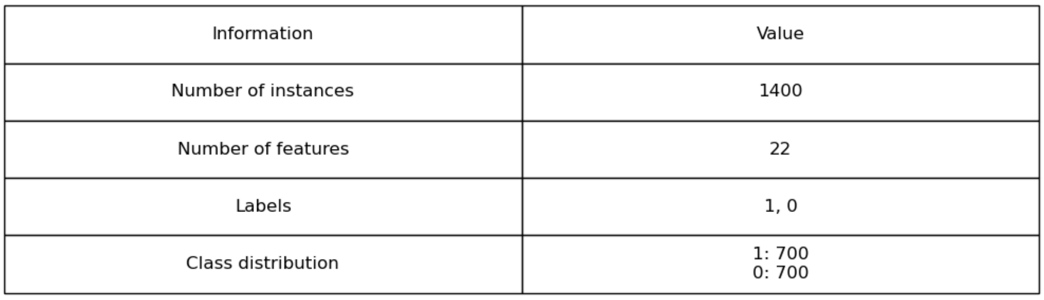
Source Here

3 Problem

Automated users are becoming more and more prominent on social media platforms, and they pose potentially significant threats to the integrity of those platforms. These automated users span from scam bots impersonating other people to automated users created in order to boost other accounts fraudulently. In the case of scam bots, these are sometimes used to manipulate people out of their money or for more nefarious purposes. They may automatically comment on posts and attempt to draw users into clicking links that might let attackers steal their login info or session information among other personal data. Other automated bots may be created and used to boost the legitimacy of these bots so that they seem more similar to real accounts. So identifying either of these is a very important problem.

Machine Learning models have the power to identify complicated patterns which even humans may not always pick up on. If we use the Machine Learning models that we have been learning about throughout the semester, we may be able to identify fraudulent accounts even when some humans may not be able to recognize them. Additionally, automating this process could be an effective way to stop the accounts before they cause any harm by monitoring them in real time and reporting them to Instagram.

Part of the challenge to solving this problem is finding a large enough dataset which contains an adequate set of parameters for our models to train. We sourced our dataset from a github which can be found here. It contains a detailed set of parameters about each account along with labels which show if an account is automated or not.



Information	Value
Number of instances	1400
Number of features	22
Labels	1, 0
Class distribution	1: 700 0: 700

The instances labelled zero are real accounts and datasets labelled one are automated. We have an equal number of each type, which bodes very well for our Machine Learning models. A drawback of this dataset is that it only contains 1400 instances, so there are certainly going to be outliers which our models will not be able to detect in the wild.

Table 1: Description of Column Names

Column Name	**Description**
userMediaCount	Number of media items uploaded by the user
mediaLikeNumbers	Number of likes received on the user's media
mediaCommentNumbers	Number of comments received on the user's media
mediaCommentsAreDisabled	Indicator if comments are disabled on the user's media
mediaHashtagNumbers	Number of hashtags used in the user's media
mediaUploadTimes	Timestamps of when the user's media was uploaded
mediaHasLocationInfo	Indicator if location information is available for the user's media
userFollowerCount	Number of followers of the user
userFollowingCount	Number of accounts the user is following
userHasHighlighReels	Indicator if the user has highlights or reels
userHasExternalUrl	Indicator if the user has an external URL in their profile
userTagsCount	Number of tags used by the user
userBiographyLength	Length of the user's biography
usernameLength	Length of the user's username
usernameDigitCount	Number of digits in the user's username
automatedBehaviour	Indicator of automated behavior detected

In the above table, the parameters of the original dataset are summarized. They are very detailed, which is helpful for our ML algorithms, but they are also a little bit challenging. All of the parameters with the prefix ”media” are actually stored as lists whose indices point to the indices of each post that a user has. Naturally, our ML algorithms will have a hard time taking in lists as parameters, so we had to simplify the data in pre-processing a bit. Despite this, the degree of detail in the dataset is very valuable and gives us a lot to work with for our predictions.

4 Solution

To solve our problem, we decided to apply a set of the supervised learning algorithms that we learned this semester. We chose Support Vector Classifiers, Decision Trees, Random Forests, and Neural Networks. By deploying these four algorithms, we were able to compare the performances across differing methods to find the best approach for the problem at hand. Each model works very differently and has different strengths and weaknesses. It is important to understand how each of these models works before we apply them.

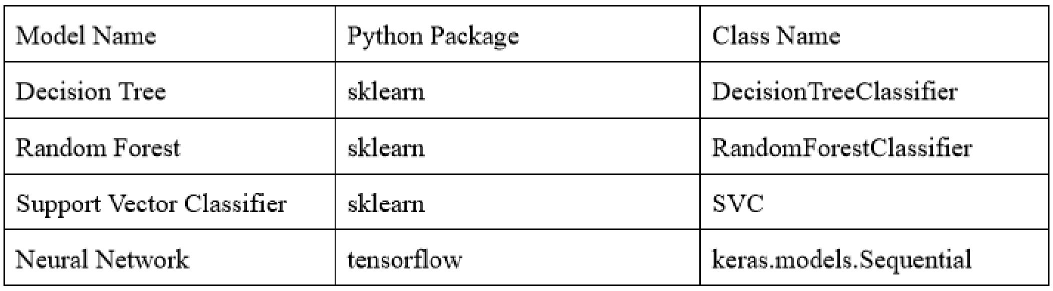
Support Vector Classifiers are powerful classification tools which work by finding the separating hyperplane between classes in the feature space. SVC's attempt to maximize the margins between classes with this hyperplane using the datapoints closest to the decision boundaries. SVC's are pretty effective with an extensive number of parameters and they are quite robust against over-fitting. They are a little bit inefficient, but since our dataset is relatively small, this is not too much of a concern.

Decision Trees are another powerful tool which work by partitioning the feature space into regions based on the feature values. Each internal node of the tree has a rule that splits the data which enters it, and each leaf node has a corresponding class which it predicts. Decision trees are very transparent in their prediction process, which makes them nice to use since we can understand exactly what rules led to the decisions.

Random Forests are simply collections of Decision Trees which work together to make a prediction. They all train on randomly split segments of the training data and they each make a prediction on given test cases. Then they find return the most common result from the set of trees. Random Forests are better at generalizing than Decision Trees since every outlier in the dataset will not necessarily be included in every tree in the forest. This leads to more robust prediction models that often perform extremely well.

Finally, Neural Networks are a supervised deep learning structure designed after networks of neurons in the human brain. They contain layers of interconnected nodes which each hold their own weights for the inputs that they receive from the prior layers. Through back-propagation, they update these weights in order to minimize the error in the predictions during training. They are very powerful tools, but they can be subject to over-fitting if they are too complicated for a given dataset, and they are not very transparent when contrasted with Decision Trees since we don't have a good understanding of why they make the decisions that they make.

We used the sci-kit learn package in python to leverage all of the algorithms except for the Neural Networks, and we used tensorflow to create Neural Networks. In the below table, we have cited the specific implementations that we used for each algorithm:



Model Name	Python Package	Class Name
Decision Tree	sklearn	DecisionTreeClassifier
Random Forest	sklearn	RandomForestClassifier
Support Vector Classifier	sklearn	SVC
Neural Network	tensorflow	keras.models.Sequential

As mentioned earlier, we also needed to do a little pre-processing to make this dataset work for our algorithms. We started by creating new columns for the averages and standard deviations of each of the "media" columns. This seemed to work pretty well, but we also wanted to preserve the connected nature of these values in some way. For instance, we wanted a way to tell our ML algorithms that the post at index 1 has three comments and 200 likes. In order to do this, we combined the values that corresponded to the same posts, and hashed them to get a hash that represented those tuples. Then we combined the hashes for all of the posts into one "combined_posts" parameter for our ML algorithms to read in. Finally, we scaled our dataset attributes using min-max normalization to set all values between zero and one. After performing these pre-processing steps, we were ready to apply our algorithms!

5 Experimental Setup

After our data was ready to use, we set up our experiments to leverage the aforementioned implementations of Decision Trees, Random Forests, Support Vector Classifiers, and Neural Networks to evaluate which models performed the best on our dataset. Since our problem is a classification problem with two potential outputs (automated and non-automated), we used accuracy to evaluate the efficacy of our algorithms. We measured accuracy by counting the number of correct predictions and dividing it by the total number of predictions that we made. We used 75% of the dataset to train for every test, and we used the random seed 1234 each time.

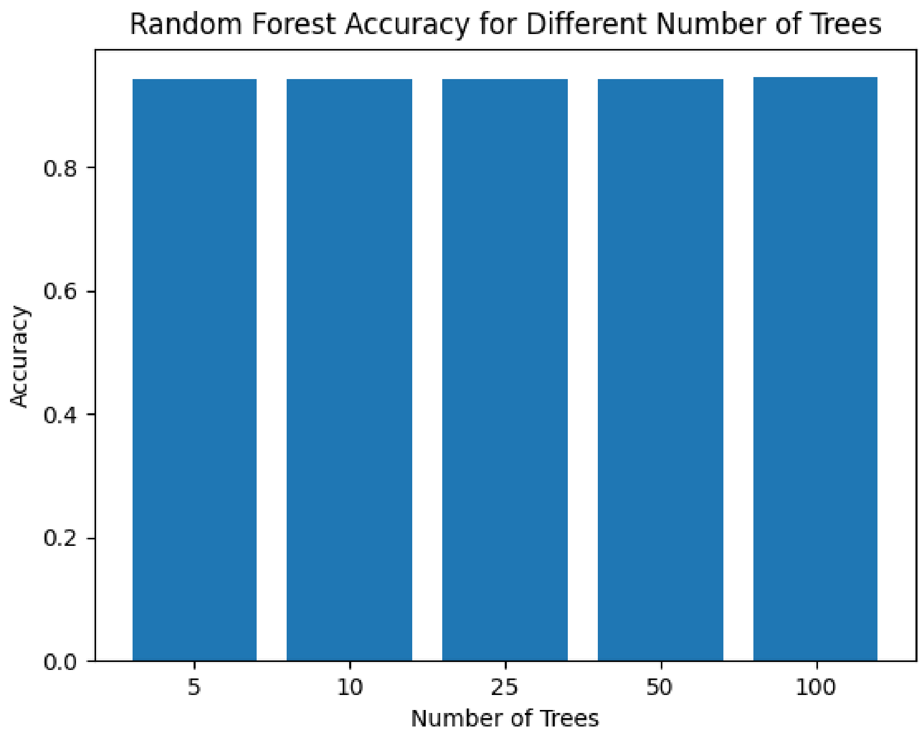
We ran multiple experiments with each type of model to determine the set of hyper-parameters which yielded the best predictions. For Random Forests, we tried 5, 10, 25, 50, and 100 for the number of trees in the forest. For Support Vector Classifiers, we tested with kernels of degree 2, 3, and 4 as well as an rbf kernel. Finally, for our Neural Networks, we tested each of 0.1, 0.01, 0.001, 0.0001 as learning rates with each of 4, 8, 16, 32, and 64 as numbers of hidden neurons. For our neural networks, we also used 400 epochs and a 20% of the training set for validation.

Finally, we tried each of these with and without feature selection prior to training in order to check if we could achieve better results with more concise data.

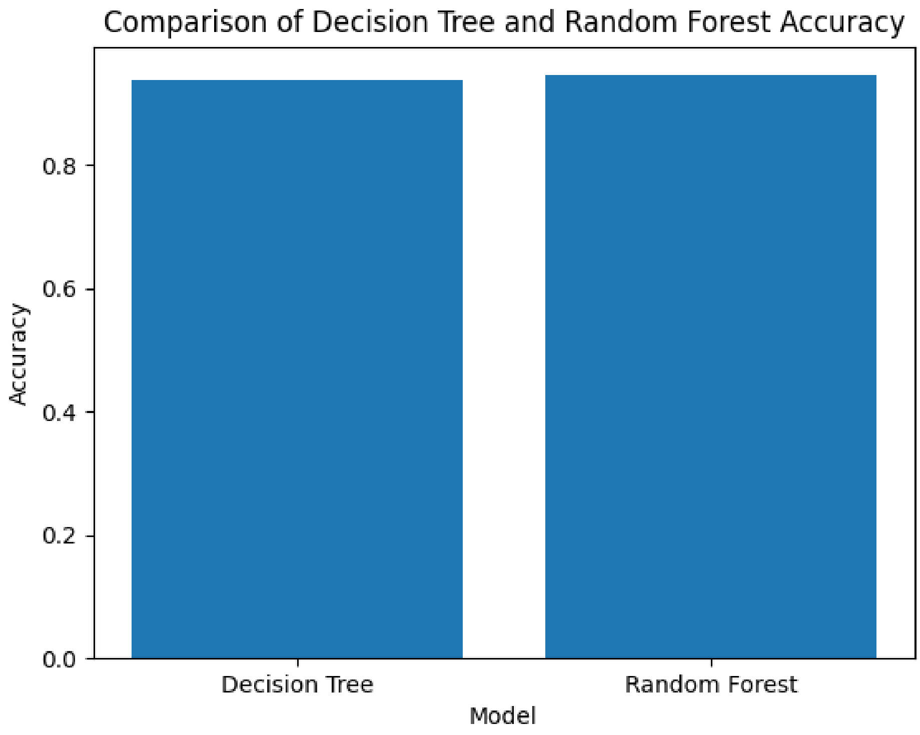
6 Outcome and Evaluations

6.1 Decision Tree and Random Forest

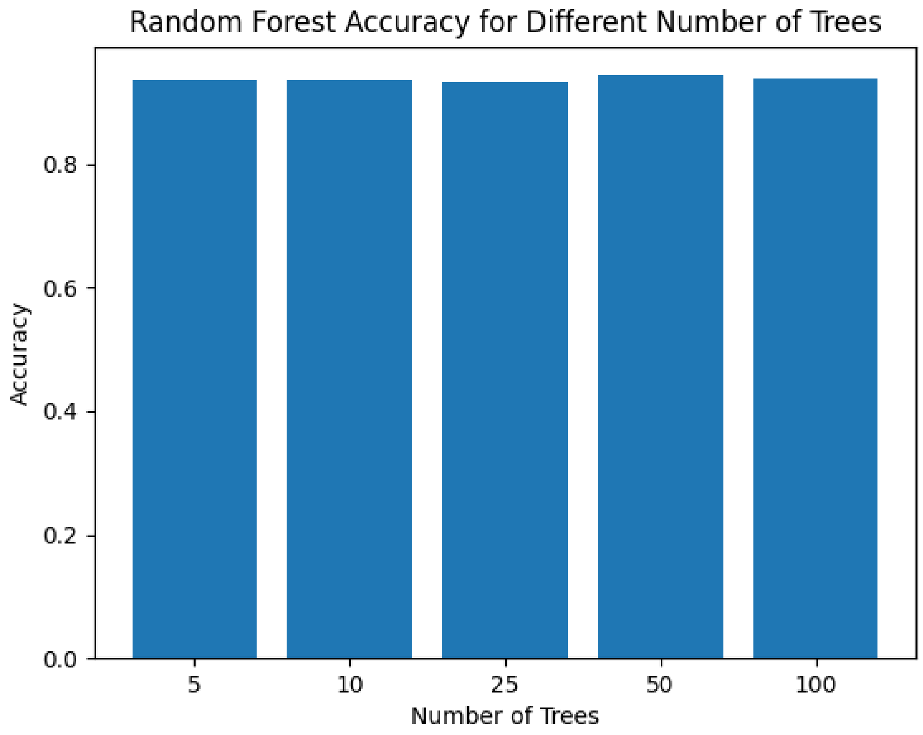
We created and ran our Decision Tree and Random Forest Models together just to compare how they ran in the moment. When first running the models we did not use feature selection. We wanted to find out what the best number of trees to use was so we ran the Random Forest model with each and ended up with these accuracies.



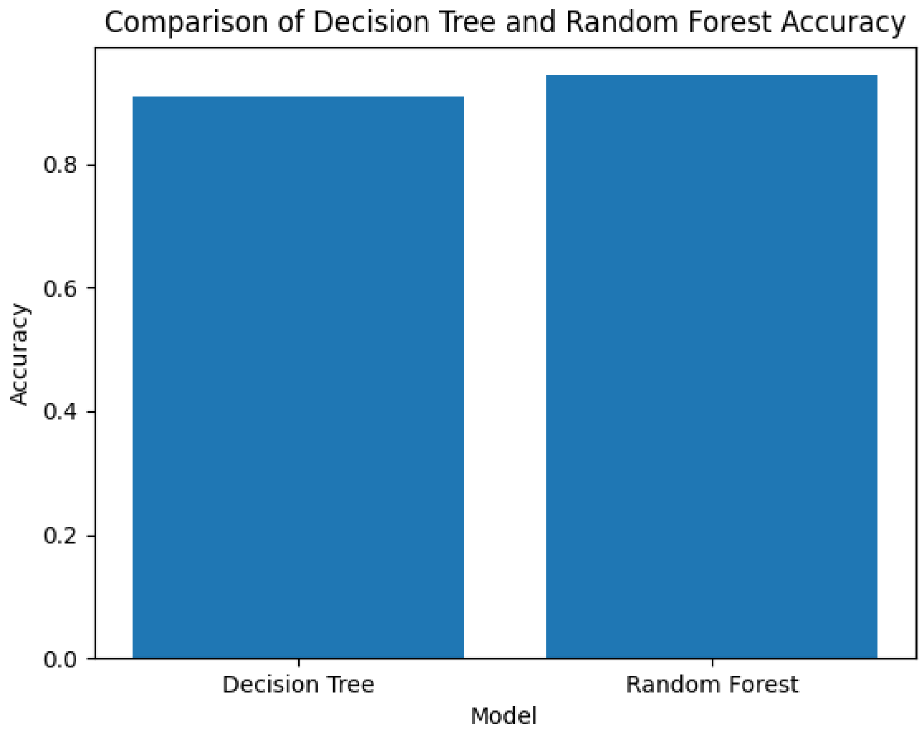
Its hard to see on this bar graph since the accuracies are very close but the Random forest that used 100 trees had the best performance with a 94.6%accuracy. The others: 5 trees - 94.3%, 10 trees - 94.3%, 25 trees - 94.3%, 50 trees - 94.4%. As you can see it was only about 0.002 more accurate then all the rest. We assumed it was the case that they were so close because the data set is not the largest. We then wanted to compare the best Random forest accuracy and the Decision Tree accuracy as seen on this bar graph:



On this graph you can see that the Random Forests accuracy is slightly better at 94.6% when compared to the decision trees accuracy of 93.7%. We concluded that that they are so close in accuracy and very accurate because the model is small and the Random Forest is slightly better because it is a more precise model when making predictions.
We then wanted to see how backward feature selection would affect our results in accuracies, so we ran it on the data set and then went through the same process of finding the best number of trees for the random forest.



There are important elements of the test that may rely on certain perameters that were removed. We saw in the graph that the model using 50 trees performed the best with an accuracy of 94.3%. The rest of the models performed as follows: 5 trees - 93.4%, 10 trees - 93.4%, 25 trees - 93.1%, 100 trees - 93.7%. Then we compared the pest performance to the Decision tree model that used feature selection as well.



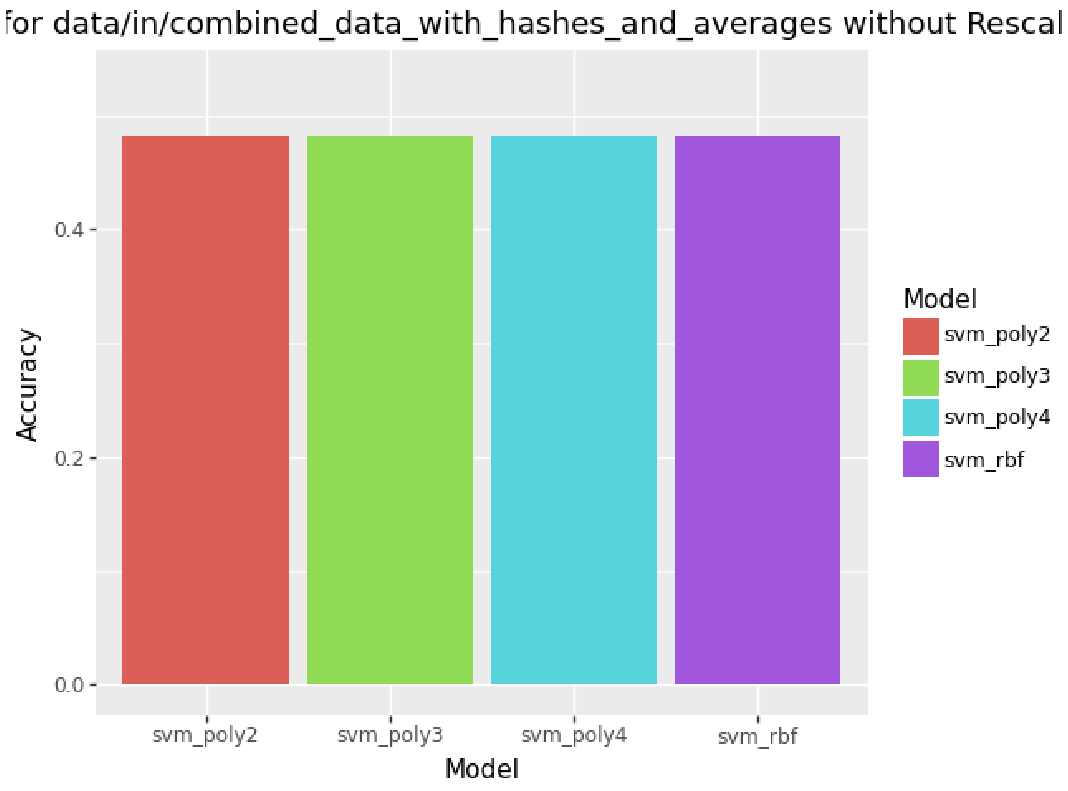
It is noticeable that the Random forest has a better accuracy of 94.3% while the Decision tree had one of 90.9% which is significantly better. It was strange to us that after using feature selection the accuracies went down. One potential reason we came up with was there might be important elements from testing that may have relied on certain parameters that were removed.

All in all the best accuracy for the Random Forest model was 94.6% and was achieved without using feature selection and by using 100 trees. The best Decision tree model was 93.7% and was also not using feature selection.

6.2 SVM's

For SVM's, we testing 4 different models of SVC, with kernels of degree 2, 3, 4, and the RBF kernel. We then tested with scaling and without scaling to evaluate hyper-parameters for best combination. Below is the table of the accuracies and them charted on the different SVM models when it comes to rescaling the dataset.

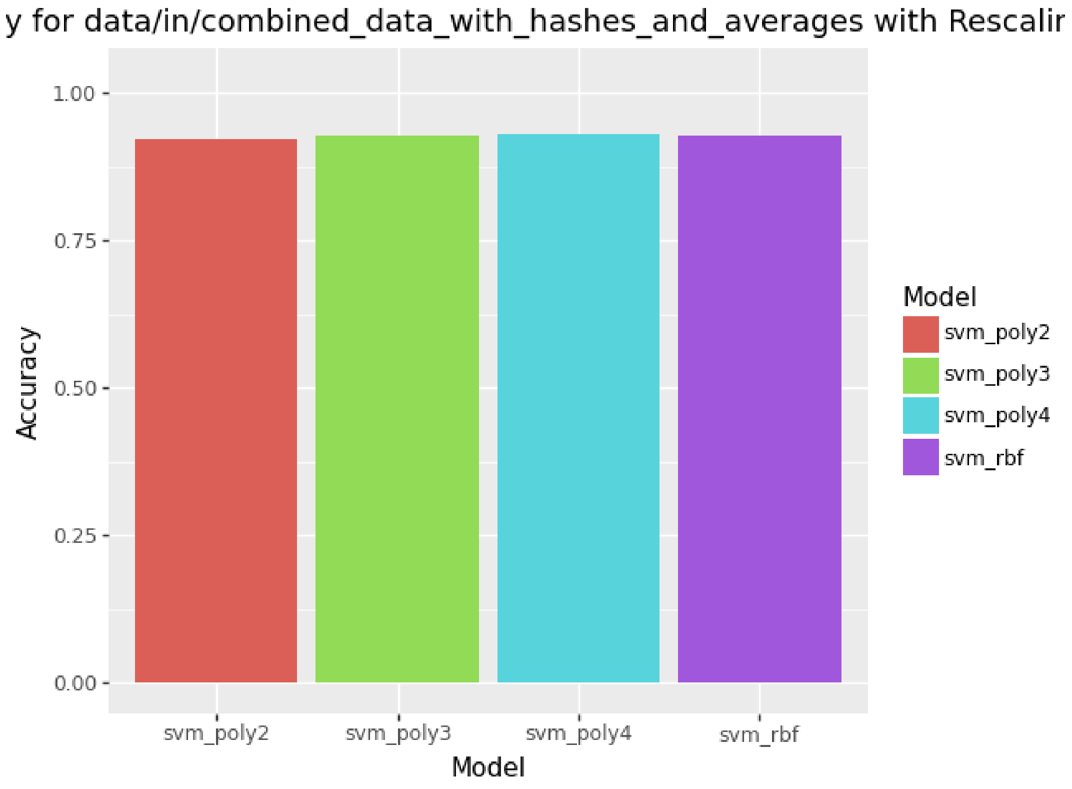
Model	**Accuracy**
svm$_p oly2$	0.4828571428571429
svm$_p oly3$	0.4828571428571429
svm$_p oly4$	0.4828571428571429
svm$_r bf$	0.4828571428571429



As you can see, all models performed the exact same when it came to not scaling. When looking into this more, we found that all the models were predicting that it was a non-automated account consistently. This may be to the model when not scaled, the hyperplane can be distorted due to the disproportionate influence of some features over others.

When rescaling the dataset and running it through our different models, we see a completely different story and an extreme statistical difference in accuracies compared to not scaling, as shown below.

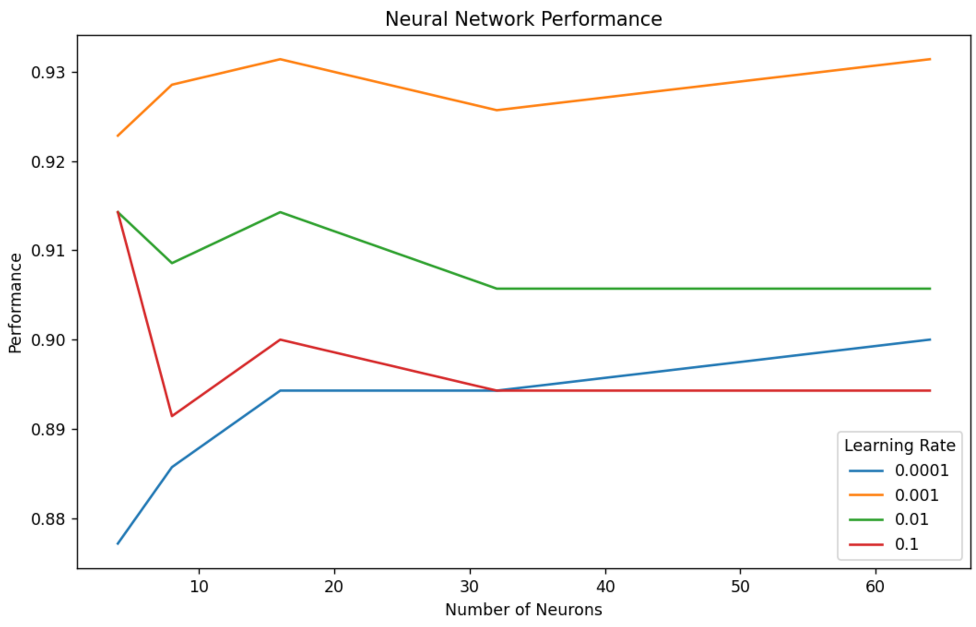
Model	**Accuracy**
svm$_poly2$	0.9228571428571428
svm$_poly3$	0.9285714285714286
svm$_poly4$	0.9314285714285714
svm$_rbf$	0.9285714285714286



The rescaling showed a nearly 50 percentage increase from not scaling, showing the importance of scaling in regards to the hyperplane and can skew the decision boundary which can lead to poor model performance because the SVM might not correctly classify the data points that are less dominant in scale but still significant for making correct predictions.

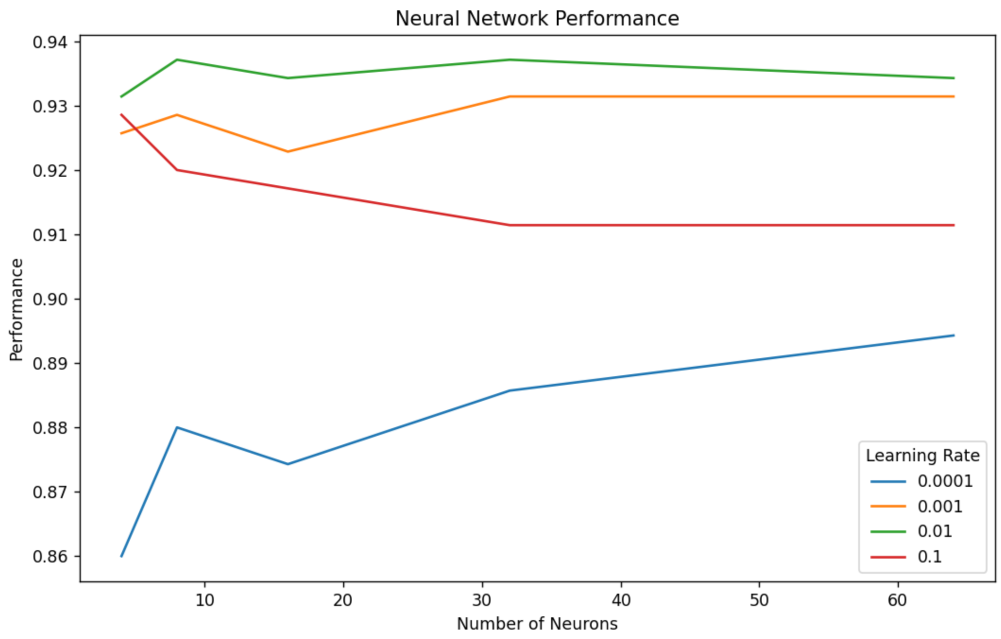
6.3 Neural Network

For our Neural Networks, we created 20 models without feature selection and 20 models with feature selection in order to evaluate the best set of hyper-parameters for our problem. Without feature selection, we created the following line chart which represents our results to display the best performance. Accuracy is shown on the Y-axis, number of hidden neurons is shown on the X-axis, and there is a separate line for each learning rate that we used.



We achieved the best accuracy here with 93.2% using only 16 hidden neurons, and a learning rate of 0.001. The only other model whose confidence interval overlapped with this one had eight neurons and a learning rate of 0.001. After 16 neurons, the drop-off is very significant, which may be because neural networks with more than 16 neurons are a little too complicated for this dataset with only 22 parameters. It also seems like the highest learning rate consistently performed very poorly, which is probably a result of data over-fitting. Finally, the lowest learning rate performs worse than every other model, and it likely got stuck in local minima since the learning rate was too low to escape.

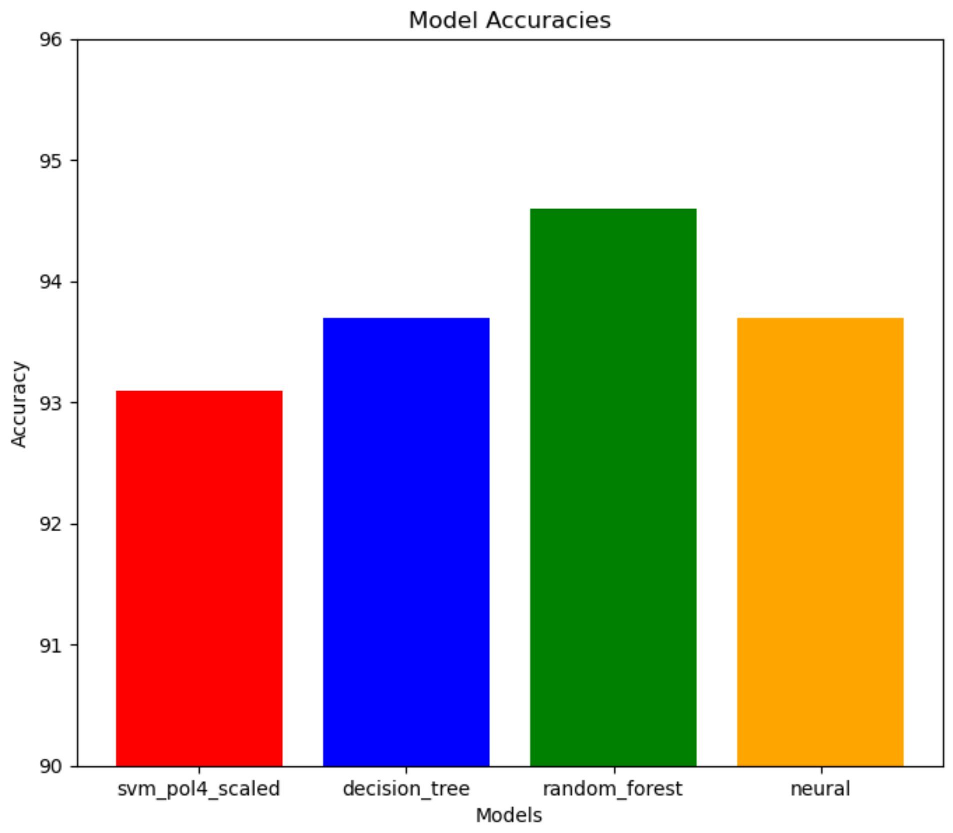
We next explored the benefits of feature selection. We used the same sets of hyper-parameters for these tests, but we ran our feature detection function before training the neural networks. The algorithm elected the following parameters to use in the Neural Networks: userMediaCount, avg_mediaLikeNumbers, std_mediaLikeNumbers, avg_mediaCommentNumbers, std_mediaCommentNumbers, avg_mediaHashtagNumbers, std_mediaUploadTimes, userFollowerCount, userFollowingCount, userHasExternalUrl, userBiographyLength. Specifically, it removed a good deal of the media parameters as well as some information about the usernames that did not seem particularly relevant. After running these models, we generated the following graph using the same representation as the prior, but for our new models:



After feature selection, the learning rate 0.01 performs the best. With a learning rate of 0.01, and 8 hidden neurons, we get slightly better accuracy than any of the models without feature selection at 93.7%. While this performance is not a statistically significant improvement over the models without feature selection, it is very interesting that we have developed models which are just as good with many less parameters. This is extremely useful because our models train much faster with fewer parameters. Each epoch takes about 2/3 the amount of time that it did without the feature selection. This probably happened because some of the extraneous parameters in the dataset did not have a strong relationship with the labels, and were being weighted too heavily in the Neural Networks that used them.

6.4 Comparison

Now that we have evaluated the best hyperparameter values for each of our models, we can compare the best performing models on our dataset. The following bar chart shows the best performances of each of our models:



We can see that the best performing algorithm on this dataset was the random forest, which showed a statistically significant improvement over all of the other algorithms. Neural Networks and decision trees performed similarly, and our best SVM performed the worst.

7 Conclusion and Future Work

In conclusion, this project was able to use four different machine learning techniques in order to identify automated accounts on Instagram, distinguishing them from genuine user profiles. The models we used were Decision Trees, Random Forests, SVMs, and neural Networks. After much testing it was concluded that the Random Forests were the most accurate. The goal of this project is helpful in maintaining the credibility of social media platforms and making sure that user interaction is authentic.

For future work, there are many different improvements that could be made. One of which could be expanding the dataset which would improve many of the models accuracy helping predict bots from real users. Another could be trying to implement a sort of unsupervised learning that might help reveal new patterns that aren't captured by our models to help stay ahead of newer automation methods. If improvements like this are made then the detection process would be more effective and would secure social media environments.