Authors: Tim Lu, Sara Mohandes Samani, Francisco Sevilla, Ryan Barnfield

CSCI5523 - Data Mining Final Report - Group 24 - Spring 2024
Determining whether users on social media are spammers/fake apart from genuine users
https://github.com/1LUTIMLU1/DataMiningProject

INTRODUCTION

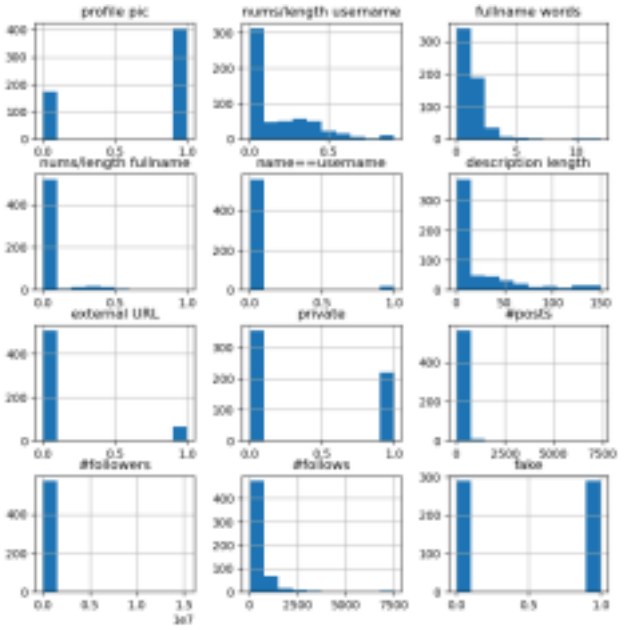
Fake accounts on social media are so widespread and prevalent now that the majority of profiles that appear on most social media platforms are likely fake. In many ways, the number of fake social media accounts has gotten so out of hand that some users may feel as if their social media platforms have been overtaken by fake accounts and given the rising sophistication of "bot" accounts it is no longer easy to distinguish fake accounts from genuine accounts. Moreover, the problems with fake accounts on social media do not end here as unlike "real" people with "real" accounts many of these "fake" accounts have little issue with engaging in behavior such as soliciting merchandise that would normally earn "real" people scorn. Due to how significant fake social media accounts have become, the social media experience is considerably degraded for many. Motivated by the mounting problems posed by fake social media accounts, our paper examines several popular data mining techniques on the tasks of detecting fake social media accounts as well as uncovering hidden patterns relating to fake and genuine accounts

PROJECT SETUP

For this project, our group decided we would work on detecting fake accounts on the popular social media platform Instagram. Fake accounts on social media are of particular interest to us as a group because there are countless variations and ways in which people post fake accounts and try to pass them off as genuine. Although it may be hard to comprehend how exactly people are able to keep coming up with new ways of generating fake accounts on social media, through the direct application of data mining techniques, we can more precisely determine what kinds of common patterns and relationships exist in fake accounts and leave out conjecture. In our project, we examined a dataset on fake versus genuine social media accounts and built classification models off of that and compared model performances. Tim tested two types of algorithms on the fake/genuine social media account dataset, namely **random forests and logistic regression**. Additionally, Tim also analyzed the attributes present in the dataset such as "profile pic, fullname words, private, #posts, #followers" in order to find the most relevant attributes to discover patterns relating to genuine/fake accounts using apriori rule generation and binning. Lastly, Tim tested the predictive abilities of the algorithms on a few real-life Instagram accounts with the aid of a web scraper extension for Instagram and collected some interesting results. Francisco evaluated a **K-NN classifier** using a stratified method and tested different parameters for the classifier, including the distance metric from a default of minkowski, cosine and finally manhattan, Francisco also explored using different numbers of neighbors from 5 to 9. Ryan focused on the **Decision Tree classifier**, selected for its ability to provide clear and interpretable decision-making processes crucial for distinguishing between genuine and fake accounts. The Decision Tree was particularly valuable for its straightforward visualization of decision paths and its effectiveness in dealing with both categorical and numerical data. I conducted extensive experiments to fine-tune the model through hyperparameter optimization using GridSearchCV, which helped improve the accuracy and robustness of the classifier in identifying fake profiles based on key features derived from the dataset. For the **Naive Bayes** section of the project, Sara conducted a thorough comparison of model performance across various scenarios. These scenarios included standardizing the data and reducing the size of the training subset while maintaining class balance. Also, leveraging mutual information for feature extraction proved to be beneficial, yielding comparable results in the analysis. The results revealed that Naive Bayes exhibited moderate performance in distinguishing between fake and genuine accounts using this dataset.

DATA EXPLORATION

The dataset is from Kaggle: https://www.kaggle.com/datasets/free4ever1/instagram-fake-spammer-genuine-accounts It consists of two files, a train and test dataset with 526 and 120 samples respectively. There are 11 features and a binary classification whether the Instagram account is fake. The features are profile picture, length of username, fullname worlds, length fullname, name equal to username, description length, external URL, private account, number of posts, number of followers, number of follows. The following histogram visualization allows us to see the **dataset is balanced** between fake and genuine accounts, along with the distribution for each of the features.




```
#   Column
---  ------
0   profile pic
1   nums/length username
2   fullname words
3   nums/length fullname
4   name==username
5   description length
6   external URL
7   private
8   #posts
9   #followers
10  #follows
11  fake
```

PROPOSED SOLUTIONS

K-NN Classifier

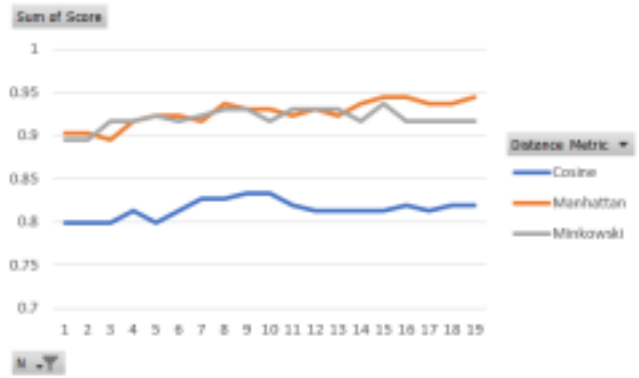
A K-NN classifier does not construct a model, rather it keeps a record of training data, it is a type of Instance-Based learning. The classification is then based on a majority vote. K-NN is a very simple method for classification where the key measure is the “distance” between the test record and the training records, how many training records are stored is referred to as “neighbors” where the classifier will choose *k* of the nearest neighbors.

The proximity measure is thus a critical component of K-NN, this can take the form of Euclidean distance, Manhattan, Cosine, Minkowski, practically any distance metric can be created and used. Each distance metric has its advantages and disadvantages based on the dataset. For example, Cosine measure works better for documents, which is not the case for our project. In the evaluation of the proposed solution the Cosine metric is included to show the results of k-nn with an improper distance metric.

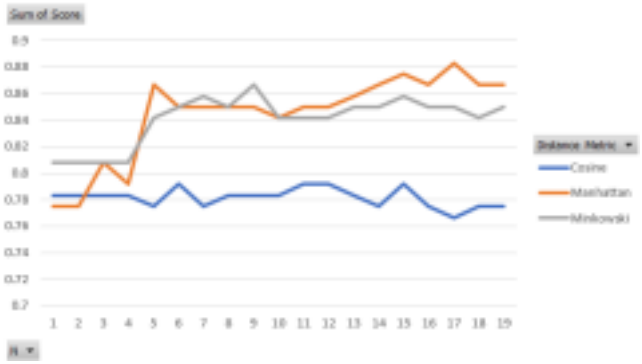
One disadvantage of KNN is that it has difficulty handling missing values, fortunately for our project the dataset selected was not missing any record.

The following charts show the accuracy on the training and validation set (data that the classifier has never seen) by having different number of neighbors for three different metrics: minkowski, cosine and manhattan:

Training Set



Validation Set



It can be observed that none of the three distance metrics yield an accuracy higher than 90% on the validation set, furthermore the Cosine metric is the worst for this particular dataset which further confirms that Cosine measures are better suited for document type of data.

In addition, it can be observed that increasing the number of neighbors beyond 5 does not increase the accuracy significantly on the validation set, however on the training set a decent accuracy can be achieved with $n < 5$.

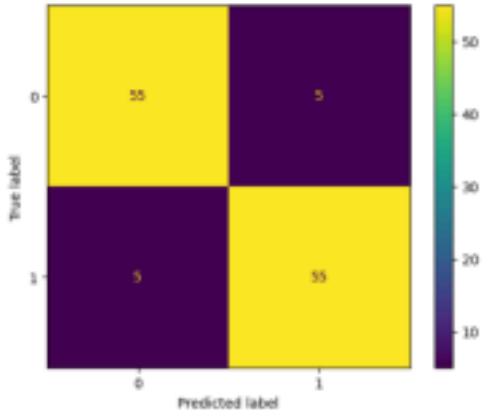
In summary KNN is not a good candidate for our project.. The simplicity of KNN can only get us so far. If we aim to obtain higher accuracy, the model complexity will increase. One could argue though that KNN can be better suited as a generalized model given its simplicity and fairly decent accuracy between 80-90%.

Logistic Regression and Random Forest.

In this project, the logistic regression and random forest algorithms were tested on the social media dataset and compared against each other. The reason these algorithms were selected in particular is because logistic regression and random forest represent two entirely different ways of interpreting the data and prediction making, though it should be noted that despite their differences, these two algorithms were both perfectly capable of achieving 100% accuracy in their own right at least once during K-folds cross validation.

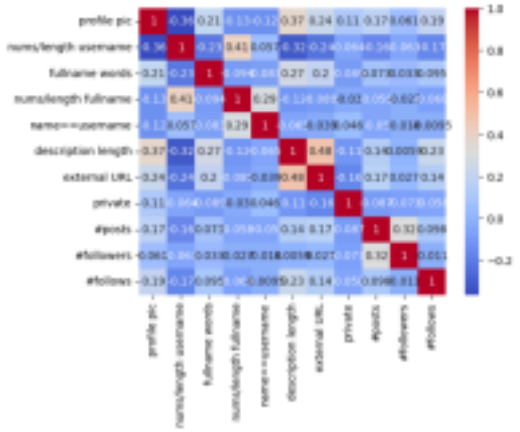
Of the two algorithms, logistic regression is probably the most intuitive to understand when envisioning how a classification algorithm should perform as it produces linear decision boundaries to separate classes similar to support vector machines. As one may guess, logistic regression is more or less a completely deterministic process as it is conceptually related to maximum likelihood estimation from statistical theory. In contrast, random forests, is a stochastic process and look at things from an entirely different angle. Unlike most traditional classification methods, random forests is an ensemble learner with random initialization that aggregates the results of several decision trees

Ultimately, of the two algorithms tested, random forests proved to be superior to logistic regression at the task of social media account classification with a 95% average accuracy compared to an 80% average accuracy although it should be noted again that both algorithms have the same performance ceiling of perfect classification rate. From the confusion matrix of random forests, one can see that random forests has higher accuracy and lower error rate. Moreover, since the odds of random forests making a false positive versus a false negative are the same, we do not have to worry about random forests being biased towards making either type 1 or type 2 errors.



A-priori rule association analysis:

From the correlation heat map shown below, one can see that there are a fair number of attributes in the original dataset though it is unclear just how meaningful each one these attributes are and some of these attributes may in fact be redundant. Another point that needs to be raised about the number of attributes in our datasets is that there are more attributes present than can be realistically extracted from a given Instagram profile page due to the way private profiles and the like work. Taking all of this into consideration, reducing the number of attributes in our dataset down to a set of only the most meaningful attributes might be a good idea especially if we plan on applying our models to the real world. Therefore, to accomplish this task apriori association analysis was applied to uncover the most meaningful attributes and hidden patterns in the dataset.



Before apriori,binning was applied which converted the mixture of binary and continuous data in the original dataset to binary categorical data. Here, a normal distribution was assumed and as such, the bins were divided into various grades with very low/very high representing two standard deviations above or below the mean, low/high representing one standard deviation above or below the mean, and normal representing anything within one standard deviation of the mean. After data conversion, apriori association analysis was performed with minsupport of 80% which generated the rules shown below. According to the rules generated by apriori association analysis, the most important attributes are the number of followers, length of fullname, number of posts, and number of follows which lines up with common sense. For reference distribution plots of the most important attributes are also shown below.

	support	itemsets	length
0	0.812500	(low nums/length username)	1
1	0.815972	(low fullname words)	1
2	0.909722	(low nums/length fullname)	1
3	0.861111	(low description length)	1
4	0.951389	(low #posts)	1
5	0.991319	(low #followers)	1
6	0.934028	(low #follows)	1
12	0.803819	(low #followers, low nums/length username)	2
18	0.810764	(low fullname words, low #followers)	2
21	0.862847	(low nums/length fullname, low #posts)	2
22	0.901042	(low #followers, low nums/length fullname)	2

fullname words	fake	proportion
0	0	52.631579
	1	47.368421
1	1	73.498233
	0	26.501767
2	0	78.074866
	1	21.925134
3	0	73.529412
	1	26.470588
4	0	71.428571
	1	28.571429
5	0	75.000000
	1	25.000000
6	0	100.000000
10	0	100.000000
12	0	100.000000

#followers	fake	proportion
0	1	100.0
1	1	100.0
2	1	100.0
3	1	100.0
4	1	100.0
...	...	...
3896490	0	100.0
5315651	0	100.0
6741307	0	100.0
12397719	0	100.0
15338538	0	100.0

#posts	fake	proportion
0	1	97.452229
	0	2.547771
1	1	85.714286
	0	14.285714
2	1	77.272727
...	...	...
1164	0	100.000000
1232	0	100.000000
1570	0	100.000000
4494	0	100.000000
7389	0	100.000000

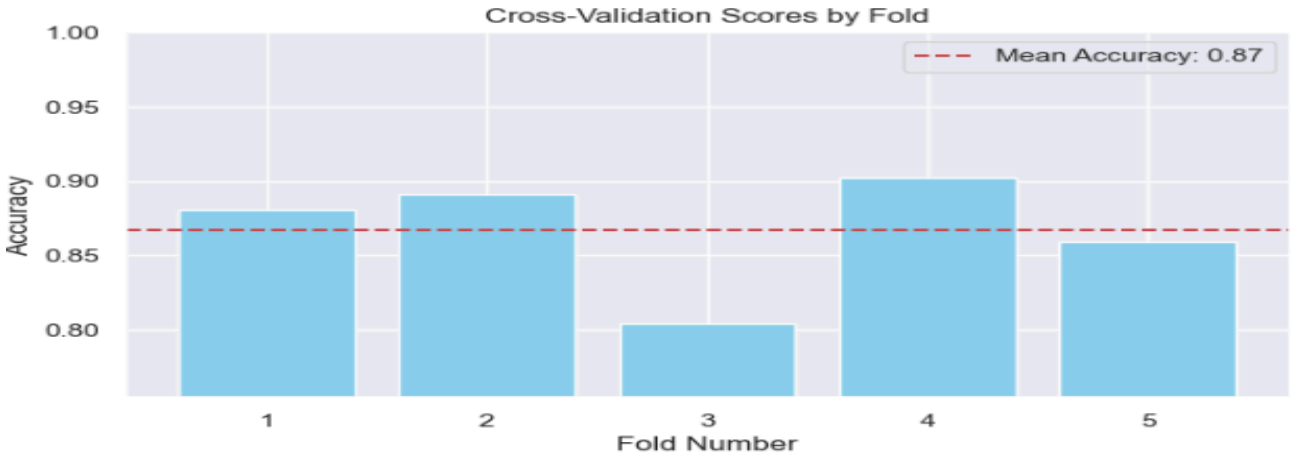
Web scraper extension results

After apriori, the random forest model was tweaked to work with only the attributes with the highest support count rather than all of the attributes in the original dataset. These changes were made to make the random forest model more applicable to real world cases in which not all of the attributes from the original dataset may be present. Logically, the next step that was carried out was testing the predictive abilities of the refined random forest model on actual fake instagram accounts with the help of the Instaloader web scraper extension. In practice, it was found that the random forest classifier can easily identify fake accounts that are obviously fake such as accounts that have no followers and posts like in the case of bot accounts. However, the tweaked random forest model failed to identify imposter accounts as fake because such accounts are often cleverly disguised. Given the already high performance of random forests, it is probably safe to assume that other algorithms would not fare much better at detecting fake imposter accounts.

Decision Tree Classifier

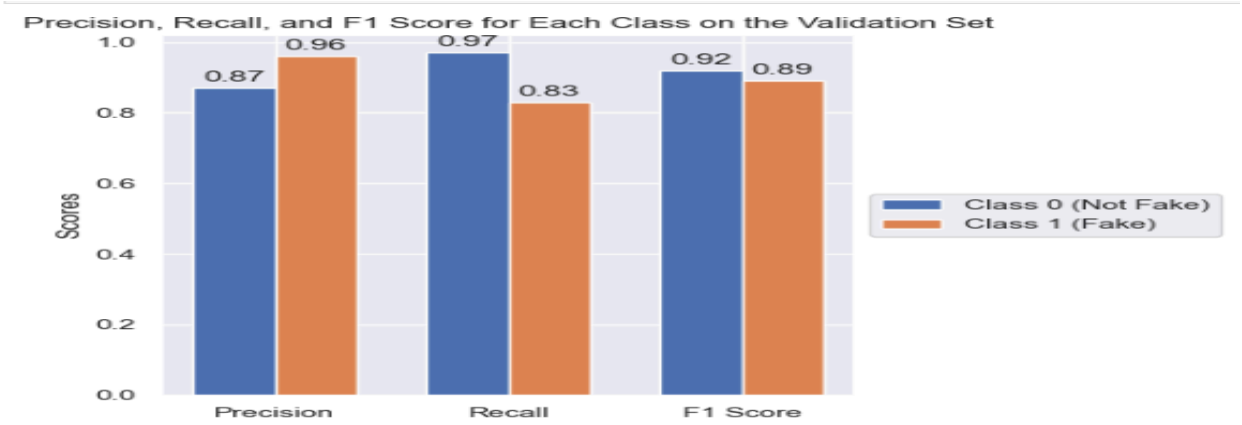
In our study, the Decision Tree classifier was employed to discern the authenticity of social media profiles on Instagram, aiming to distinguish between genuine and fake accounts. Decision Trees are a non-linear classification technique that recursively partitions the data space into subsets based on the feature that provides the maximum information gain at each step. This approach was particularly suited for our dataset due to its interpretability and the ability to handle both numerical and categorical data effectively.

The initial Decision Tree model utilized the training data to fit the model parameters. We also conducted a GridSearchCV to optimize the hyperparameters of the model, focusing on max_depth and min_samples_split to prevent overfitting and ensure the model generalizes well to unseen data.

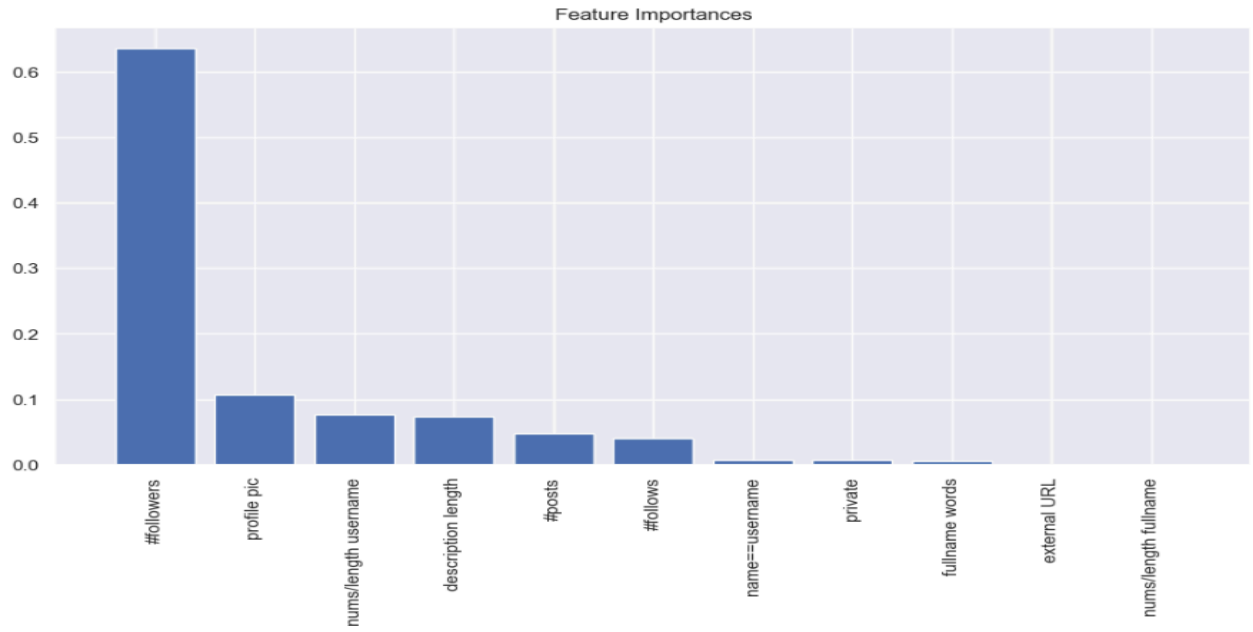


The Decision Tree classifier demonstrated strong performance metrics, as follows:
Training Data: Accuracy of 87%, with a precision of 88% for fake profiles, recall of 83%, and an F1-score of 85%.
Validation Data: Achieved an impressive accuracy of 91%, underscoring its efficacy in generalizing beyond the training dataset.
Test Data: The optimized model excelled with a higher accuracy of 93.33%, precision, recall, and F1-score all aligning at 93%.

These results highlight the Decision Tree's robustness and reliability in identifying fake profiles, with significant improvement observed in the test dataset.



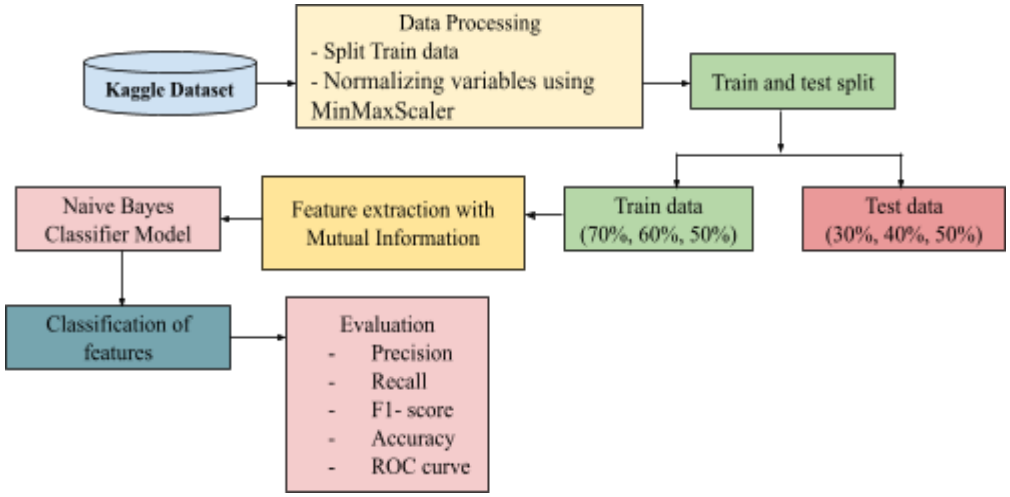
Analysis of feature importance revealed that the number of followers (#followers) held the most significant weight in predictions, accounting for approximately 64% of the decision-making process. Other notable features included the profile picture presence and description length, which were also influential but to a lesser extent. This insight is crucial for future data collection and preprocessing efforts, as focusing on these key areas could further enhance model accuracy and reliability.



Compared to other classifiers reviewed in our project, such as K-NN, Logistic Regression, and Random Forests, the Decision Tree provided a commendable balance between accuracy and interpretability. While Random Forests showed a slightly higher accuracy, the Decision Tree model offered greater transparency in how decisions were made, which is valuable for understanding the underlying patterns in fake profile detection.

The Decision Tree classifier has proven to be a valuable tool in our arsenal against fake social media accounts. Its ability to provide clear decision rules and handle diverse data types makes it a robust choice for this application. Future efforts could explore combining Decision Trees with other ensemble methods like Random Forests to harness both accuracy and interpretability in our ongoing battle against online deception.

The Naive Bayes modeling process has been illustrated schematically in the following figure. Various splits of the training data were created to ensure a balanced distribution of classes. Out of the 576 samples in the training data, subsets of sizes 280, 180, and 120 were selected. Following normalization with MinMaxScaler(), the Gaussian Naive Bayes algorithm was applied to each set of training and test data.

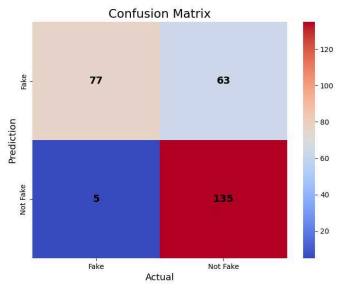


The table below showcases the evaluation results for each scenario. It reveals that reducing the proportion of training data from 70% to 60% enhances all metrics except for training subset recall. While test results show no significant improvement when increasing the test size from 40% to 50%, the overall performance metrics on training data are better in the second scenario (60% train and 40% test). Thus, the train-to-test ratio of 1.5 demonstrates superior performance.

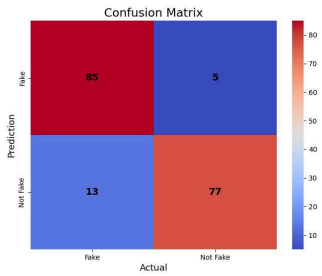
	Train 70% Test 30%		Train 60% Test 40%		Train 50% Test 50%	
	Training	Test	Training	Test	Training	Test
Accuracy	0.76	0.78	0.9	0.9	0.87	0.92
Precision	0.68	0.72	0.94	0.93	0.89	0.96
Recall	0.96	0.92	0.85	0.87	0.85	0.87
F1 score	0.798	0.81	0.89	0.89	0.87	0.91

The confusion matrix for the training set indicates that the model trained with 70% of the data has a reduced ability to differentiate the fake class compared to the scenario where the proportion of training data is increased to 60%. Moreover, the model trained with 60% of the data exhibits improved capability in distinguishing both the fake and genuine classes compared to the other two cases.

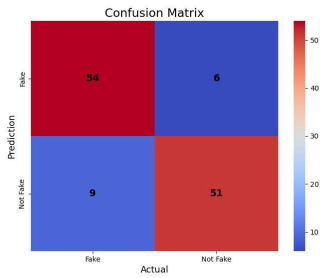
Train 70%



Train 60%

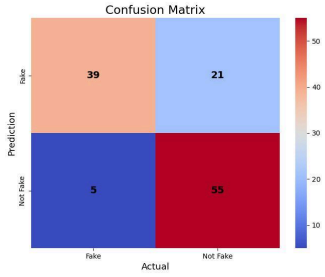


Train 50%

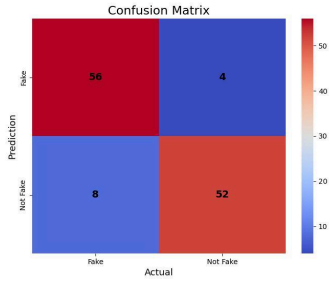


Confusion matrix on test set shows that decreasing the training size from 60% to 50% is not affecting the performance of the model on test test and that again confirms the proportion of train to test equal to 1.5.

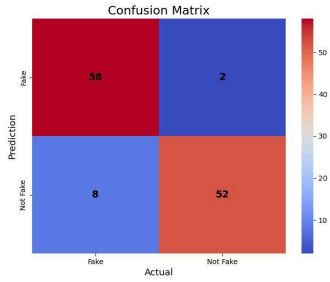
Test 30%



Test 40%



Test 50%



The features selected using mutual information show that when 70% of the training data is subsetted, the selected features remain the same as when 50% of the data is subsetted. Comparing this with the list of informative features in the case with 60% training reveals that '#posts', '#followers', 'profile picture', and 'nums/length username' are consistently the most informative features. However, '#follows' and 'description length' may vary based on the percentage of the training set used.

Train 70%
Selected features: ['#followers', '#posts', 'profile pic', 'description length', 'nums/length username']

Train 60%
Selected features: ['#posts', '#followers', 'profile pic', 'nums/length username', '#follows']

Train 50%
Selected features: ['#followers', '#posts', 'profile pic', 'description length', 'nums/length username']