Group 105 Project Final Report

Introduction & Background

Malicious URLs can be used to distribute malware and steal private information. Traditionally, manual blacklisting and web scrapers have been implemented to mitigate this issue, but SVM's, Decision trees, and Random Forest have been observed to have a ~90% accuracy rate, with the instance selection method critically impacting the accuracy[2]. Prior research is often trained on external data, such as the content of the URL's webpage, the IP address, or Google Page Rank data, which can be resource intensive or pose security risks[1]. Additionally, research from Cheng Cao showed that behavioral analysis of both the poster and viewer of a URL can achieve high precision and recall[3]. The dataset we have chosen contains 14 features focusing on the lexical qualities of the URL, such as URL length, entropy, digit-letter ratio, and @ count. Our dataset contains 2.5 million data points of URLs labeled malicious or legitimate.

Dataset URL

Problem Definition/Motivation

Pre-existing methods for malicious URL detection include employing blacklists, which is resource intensive. Current ML solutions use a wide-variety of labels, such as IP information and webpage content. Our team's motivation is to apply machine learning strategies to combat this issue in a scalable and automatic way. By only analyzing features found in the URL, our approach eliminates the need for web scraping and quickly classifies malicious vs harmless URLs, effectively addressing this cybersecurity issue.

Methods

We used 3 preprocessing methods on our data

Data Preprocessing

1. Label Encoding: We preprocessed our data using label encoding to transform True/False values into numeric representations (1s and 0s) to prepare it for model training and testing. Additionally, "legitimate" values were encoded as 1, while "non-legitimate" values were encoded as 0. This

approach was essential because our model required numeric inputs, making it necessary to transform non-numeric data.

2. Data Cleaning: We dropped data points with NAN-values and irrelevant features from the dataset. Attributes such as domain-age days contained NAN values. Additionally, we dropped the source and whois_data features, since we are focusing on only the internal attributes of the url.

3. Principle Component Analysis: We also employ PCA to aid with data visualization to make our data plottable. However, PCA has not been used on the data used to train/test the model.

We trained our data on 3 distinct ML models

ML Models

1. K-Means: We utilized Sci-Kit's K-Means. This unsupervised method quickly clusters data based on the distance from each cluster center. We utilized K-means to see if we could classify the data as malicious or benign based on the underlying patterns in the features; K-Means is a simple model to implement compared to our other models. By using k-means we can view how data is grouped in our dataset, which can identify any patterns. Additionally, by looking at that silhouette coefficient, we can identify the ideal number of clusters in the data.

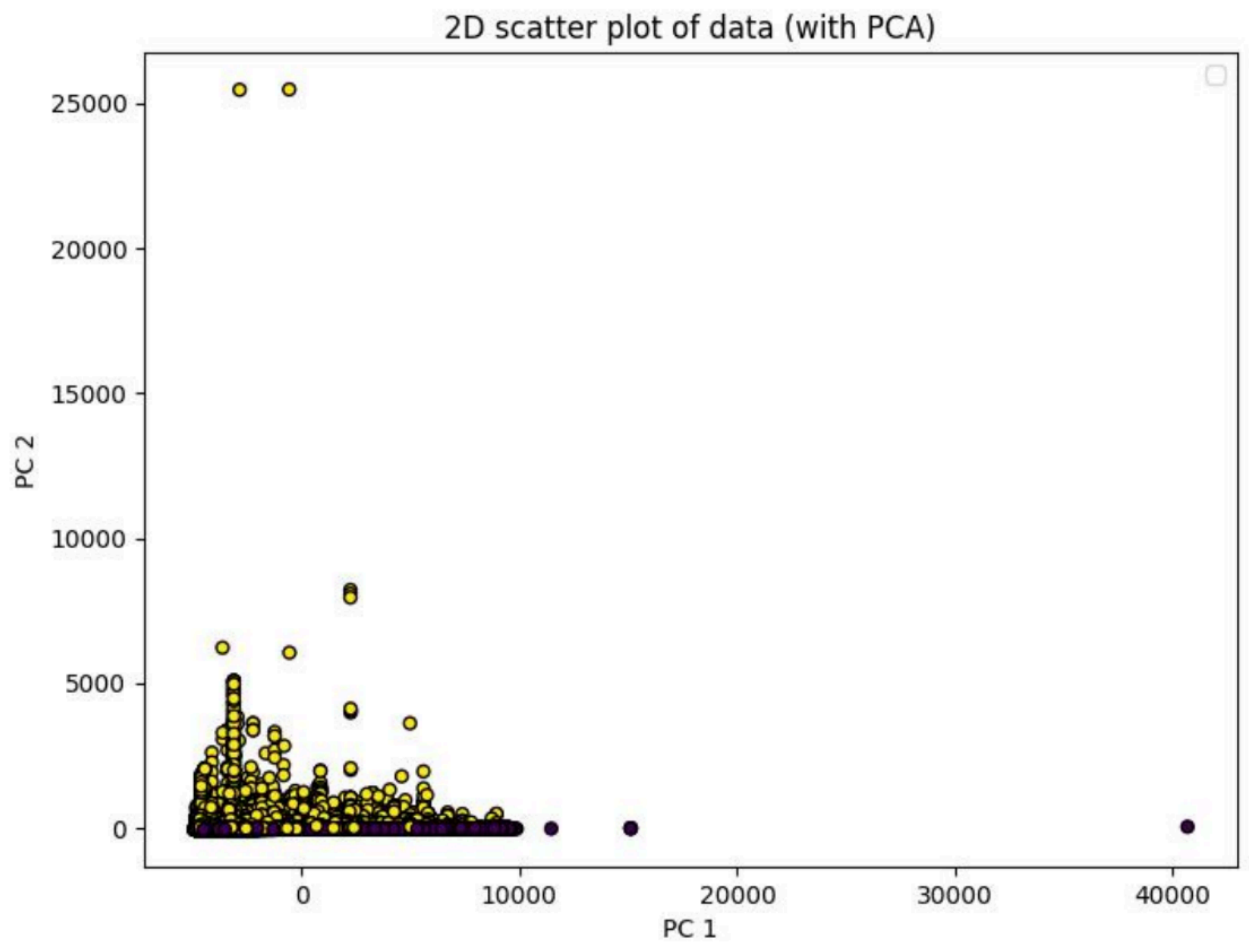
2. Logistic Regression: We used Sci-Kit's Logistic Regression. This supervised learning algorithm predicts a linear equation of the features of our dataset and, then, uses the sigmoid activation function along with a threshold to classify data points. Since logistic regression is a classification algorithm that uses our labeled data to form the model, it is an ideal algorithm for us to classify URLs into legitimate or malicious ones.

3. Random Forest: We used Sci-Kit's Random Forest Classifier. This is also a supervised learning algorithm that creates multiple decision trees given our labeled data using random features selected at each stage. Considering that our data is labeled and our goal is classification of URLs, random forest seems to be appropriate for our purpose.

Results and Discussions

Data Visualization

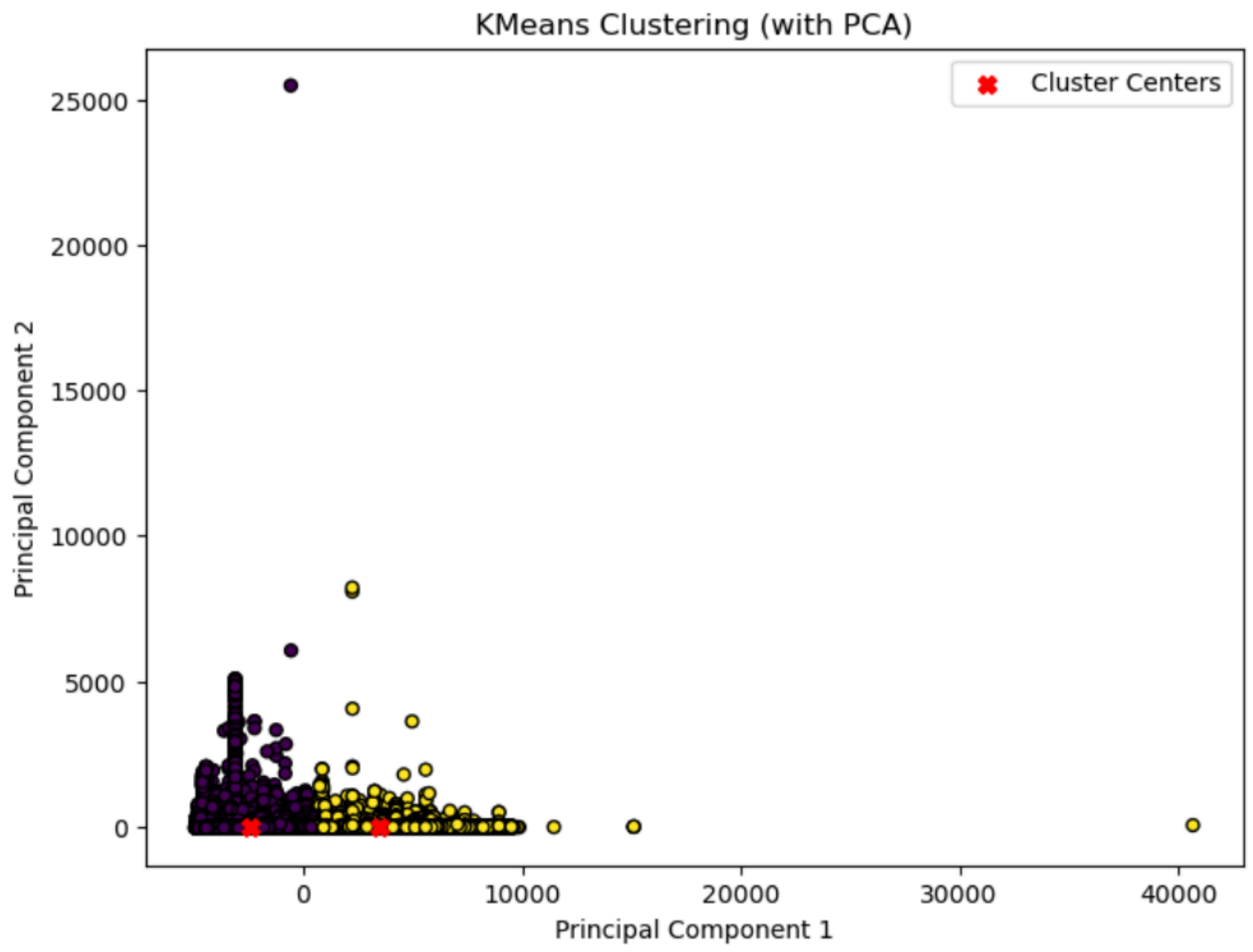
We produced this graph using PCA to reduce our model to 2 features, allowing us to visualize the spread of our data points onto a two-dimensional graph.



Visualizing our data reduced to 2 features by PCA

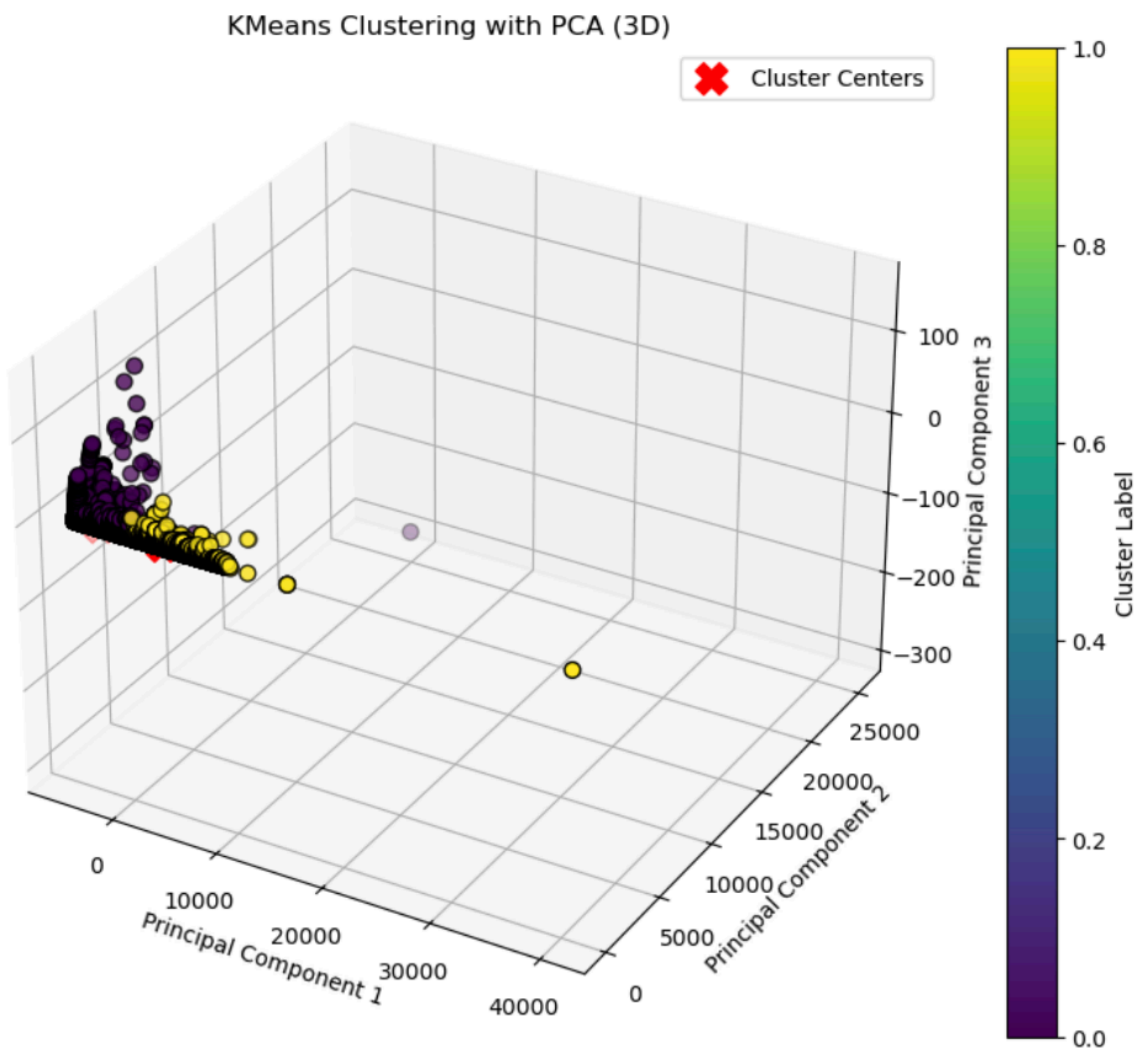
K-Means Visualization

After running K-means for 2 clusters on our model and getting our assigned cluster centers, we plotted the data on a graph to analyze which points were assigned to which clusters and what the cluster centers were. Since we had 14 features in our model, we had to perform dimensionality reduction in order to display it on a graph. We ran PCA on our model to reduce it to 2 features. We then plotted the data points on the graph with the x-axis being one principle component and the y-axis being the other principle component. The two cluster centers detected are marked in red crosses.



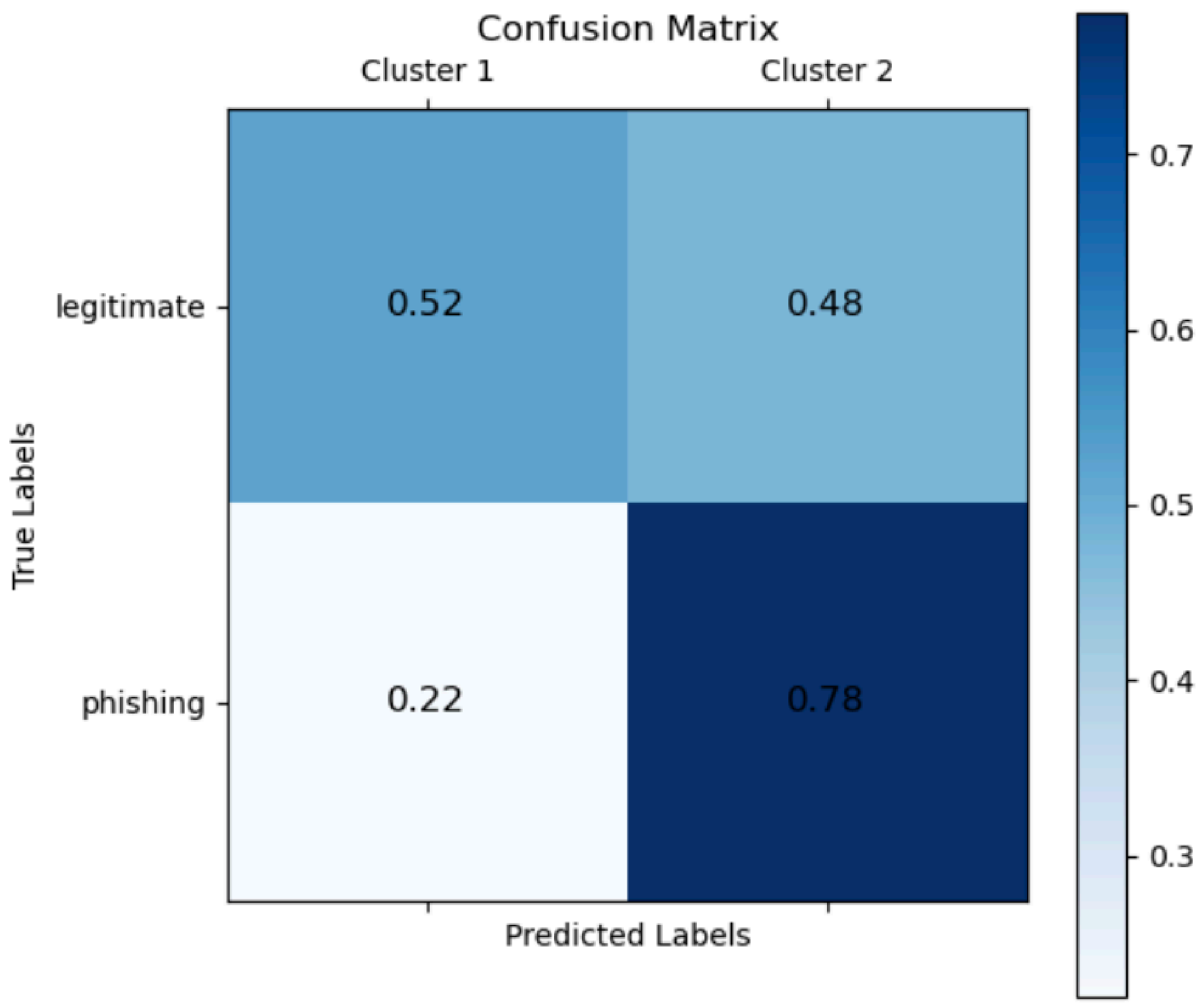
KMeans with 2 Clusters reduced to 2 components

We also ran PCA with 3 features on our model in order to get a 3d graph so that we can better represent the data. The corresponding graph is displayed below.



KMeans with 2 Clusters reduced to 3 components

Following is the confusion matrix obtained from our implementation of KMeans



KMeans Confusion Matrix

The image above is the normalized confusion matrix we produced after our K-means results. The y-axis represents the ground truth labels and the x-axis represents the cluster assignments. From the confusion matrix, we see that out of all the data points with ground truth as 'legitimate', 52% were assigned to cluster 1 and 48% were assigned to cluster 2. This is almost an even split and suggests that the model did not perform well when clustering 'legitimate' urls. For data points with ground truth as 'phishing', 22% were assigned to cluster 1 and 78% were assigned to cluster 2. This is a better result than that of 'legitimate' labels as the majority of the 'phishing' labels were assigned to the same cluster, cluster 2. Furthermore, we can also conclude from the above matrix that cluster 1 is more likely the cluster that holds 'legitimate' urls and cluster 2 more likely holds 'phishing' urls.

K-Means Evaluation

Quantitative Metrics K-means (k=2):

Accuracy: 61.95%

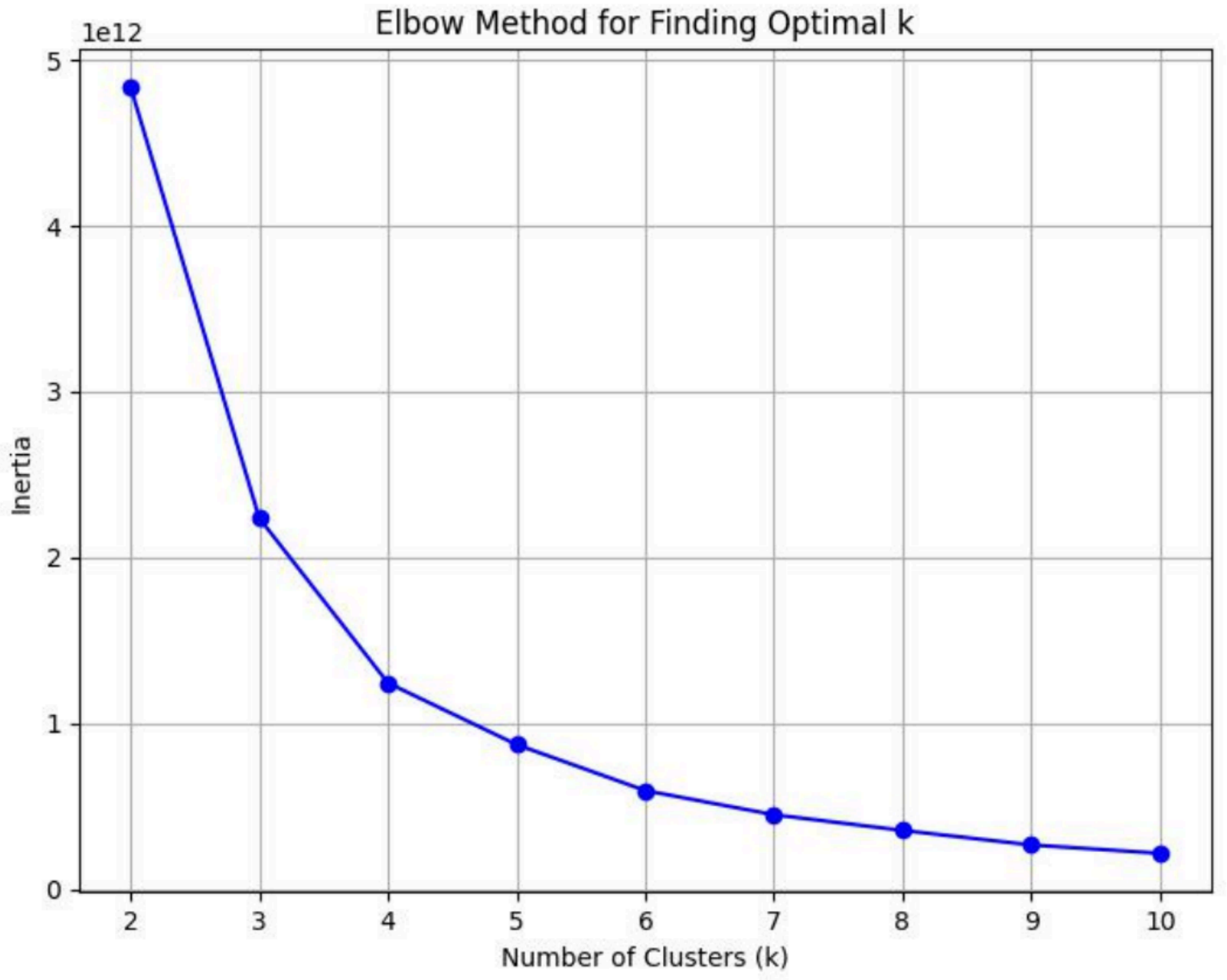
Recall: 78.07%

Precision: 49.38%

These metrics indicate that our model is performing relatively poorly compared to what we want.

Optimal K-Means Analysis:

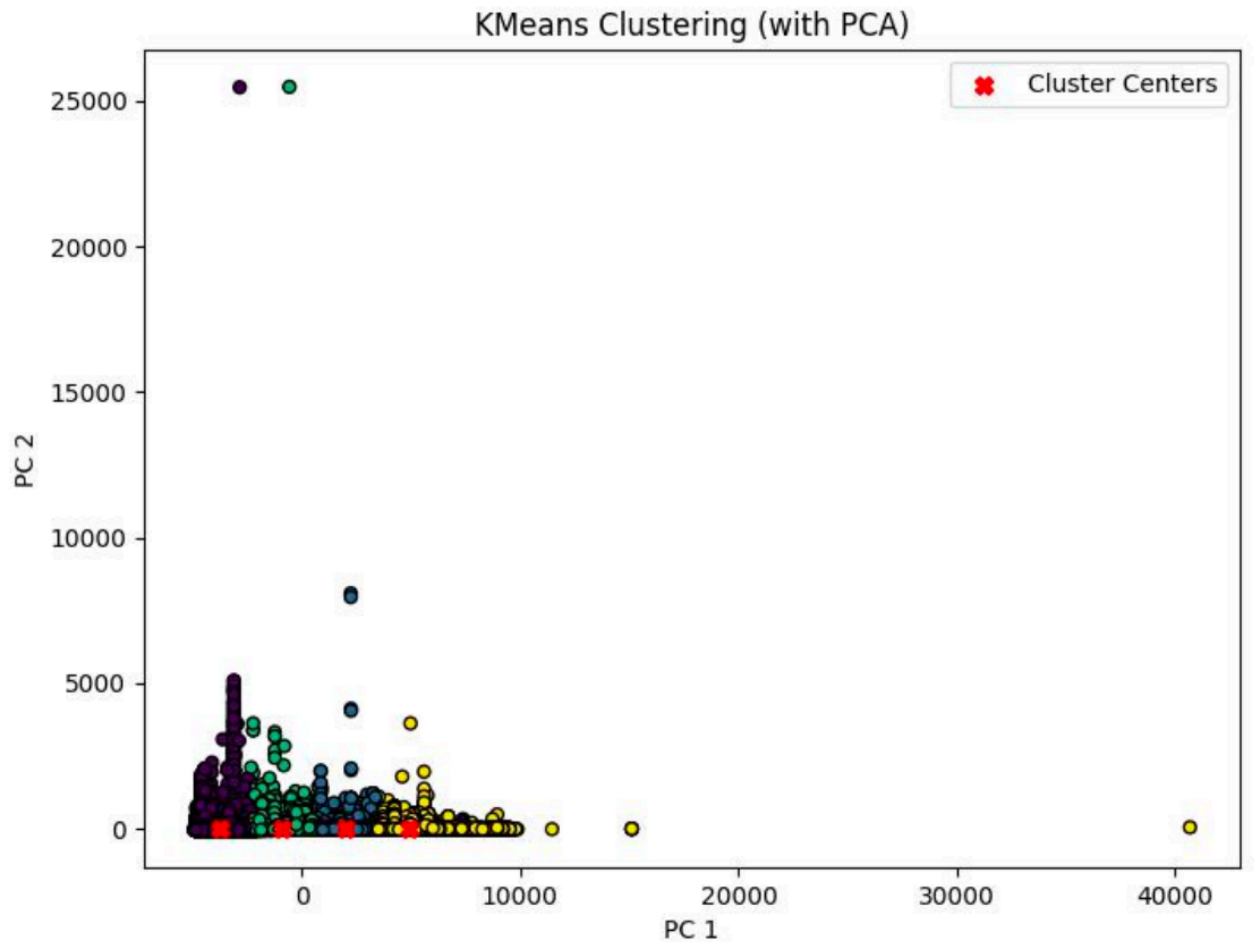
Since we planned to use K-means clustering to classify the given dataset, we use elbow method to find out optimal k



Elbow Method for Optimal K

The plot shows that k=4 is an ideal number of clusters.

This is what our clusters look like after running k-means for 4 clusters instead of 2. The clusters clearly look better for k=4 compared to k=2.



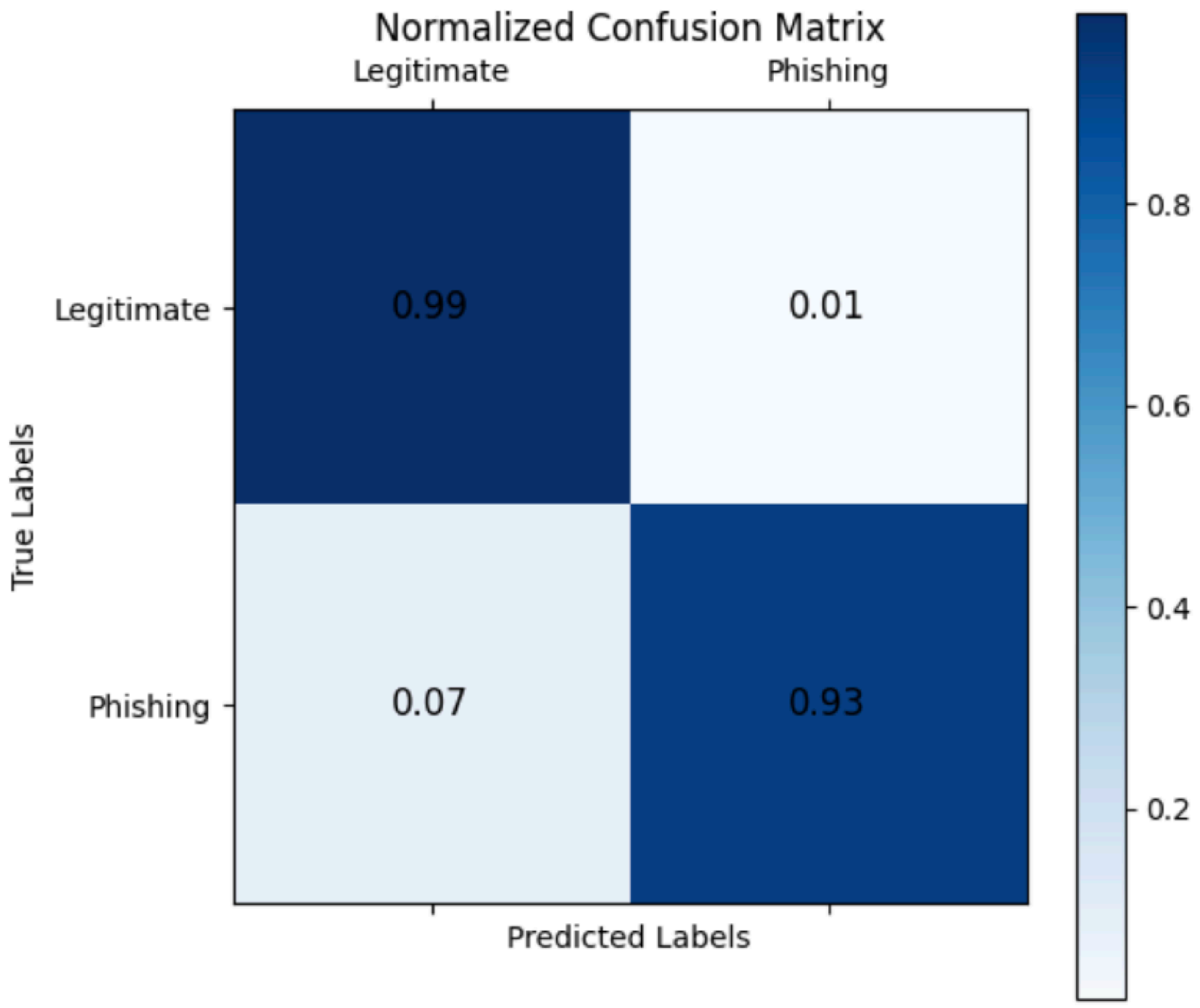
KMeans with 4 Clusters reduced to 2 components

K-Means Output Discussion:

It is clear that our original data didn't really have identifiable clusters, indicating that K-means isn't ideal for our classification purposes. On further analysis, we found that the ideal number of clusters for our data came out to be k=4 instead of 2, which is our known number of ground truth labels. We also get an accuracy of about 61% using K-means. In conclusion, our observations indicate that K-means is not suitable for our goal of classification of our models into legitimate vs. phishing URLs.

Logistic Regression Evaluation

Confusion Matrix



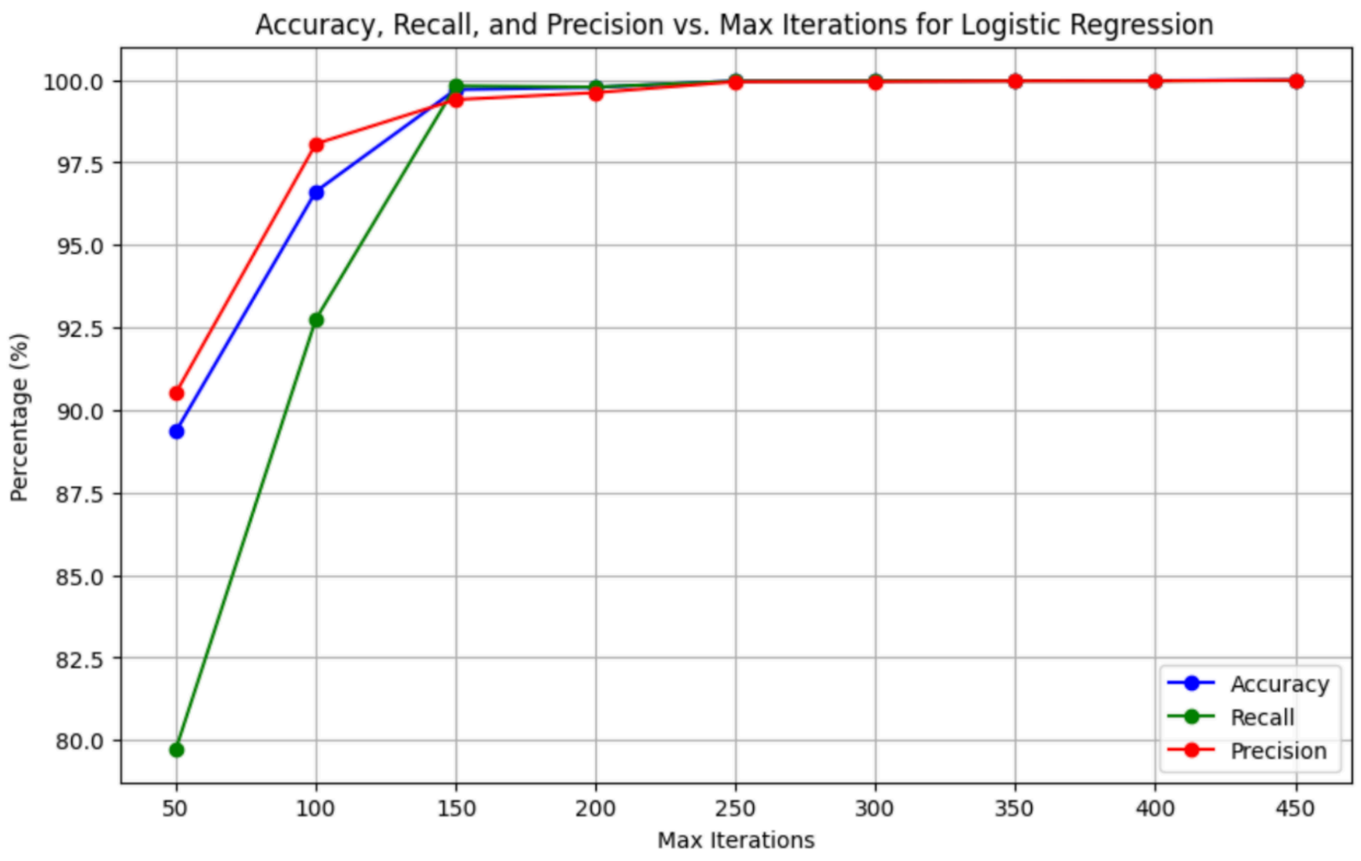
Confusion Matrix for Logistic Regression

First we trained logistic regression on the same unscaled dataset as Random Forest and Logistic Regression. We can see that our model performs quite well: it is able to predict 99% of our legitimate test data as legitimate URLs and 93% of the phishing test data as phishing URLs. Although, we see that 7% of phishing data is predicted as legitimate compared to 1% of legitimate data as phishing: the misclassification of phishing data as legitimate is more than the misclassification of legitimate data, which may be more undesirable as people might trust something that they shouldn't and can be more dangerous. Thus, we can see that our model is not perfect; however, it does perform well-enough for our expectations. We could've seen a perfect classification upon altering the hyperparameters; however, as that may result in overfitting, we opted for a less perfect model rather than overfitting as that may result in worse outputs for classifying URLs not from our dataset, harming our ultimate goal of classifying URLs.

Hyperparameters tuning for Max Iterations

As mentioned above, we tuned hyperparameters, specifically max_iter (representing maximum iterations that our model runs) to avoid overfitting our model. We did this as our model wasn't converging initially with the default hyperparameter values. However, since convergence (which was observed around max_iter=600) happened around a large iteration value, we also saw overfitting.

Thus, even though our model that we chose to go with doesn't achieve convergence, it does give us a good enough accuracy: max_iter=100 gives us good quantitative metrics as can be seen in the graph.



Accuracy, Precision, Recall VS Max Iterations for Log Regression

Quantitative Metrics (max iterations = 100)

Accuracy: 96.62%

Recall: 98.06%

Precision: 92.75%

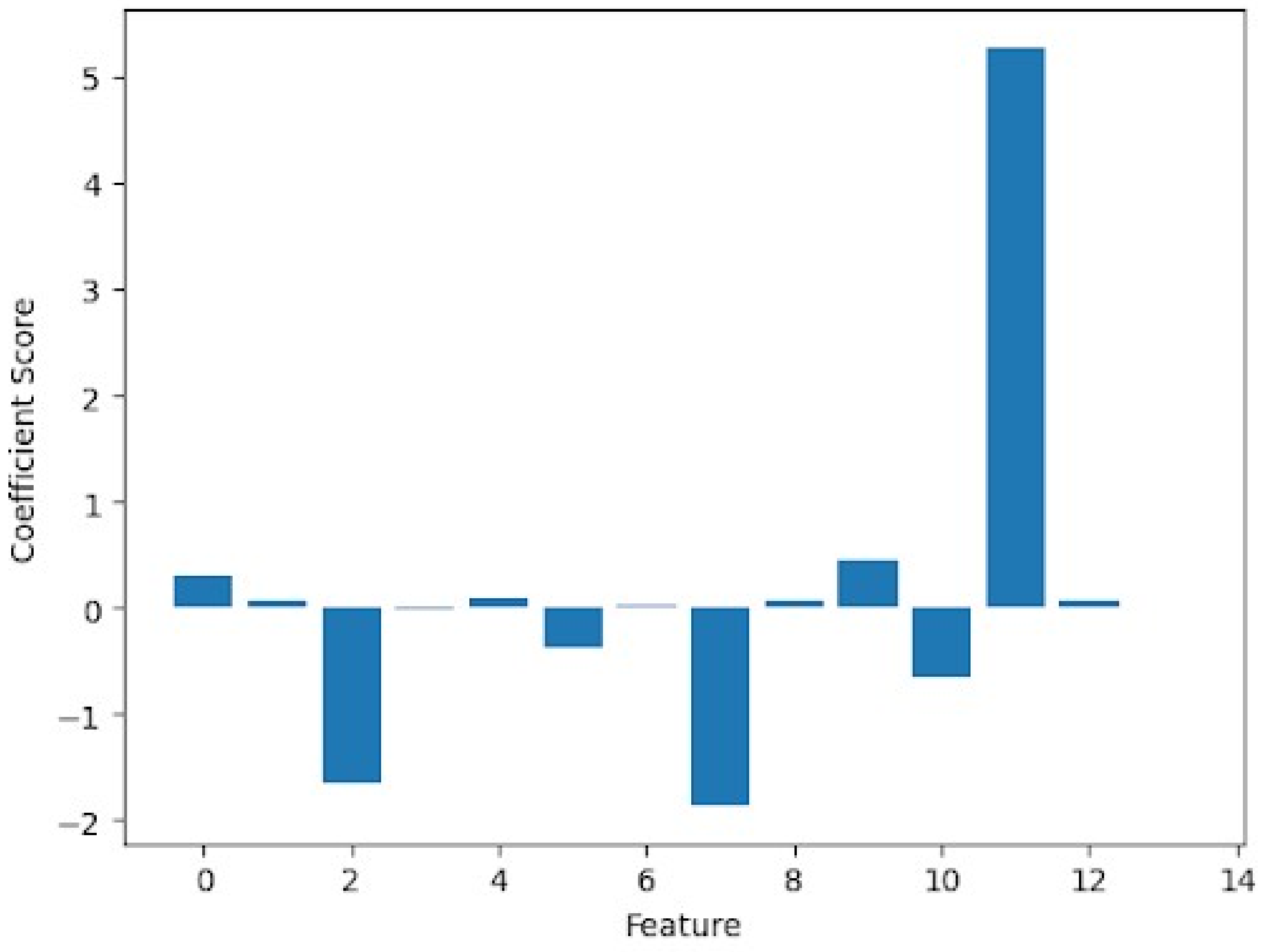
We would like to note that the metrics change a little on model runs done at different times, which could be due to the inner workings of Sci-Kit's Logistic Regression library.

Feature Analysis

	Feature	Score
0	Feature: 0	0.3226
1	Feature: 1	0.0656

	Feature	Score
2	Feature: 2	-1.8535
3	Feature: 3	-0.0183
4	Feature: 4	0.0889
5	Feature: 5	-0.4291
6	Feature: 6	0.0344
7	Feature: 7	-1.9183
8	Feature: 8	0.0729
9	Feature: 9	0.4806
10	Feature: 10	-0.7170
11	Feature: 11	5.6166
12	Feature: 12	0.0709
13	Feature: 13	-0.0002

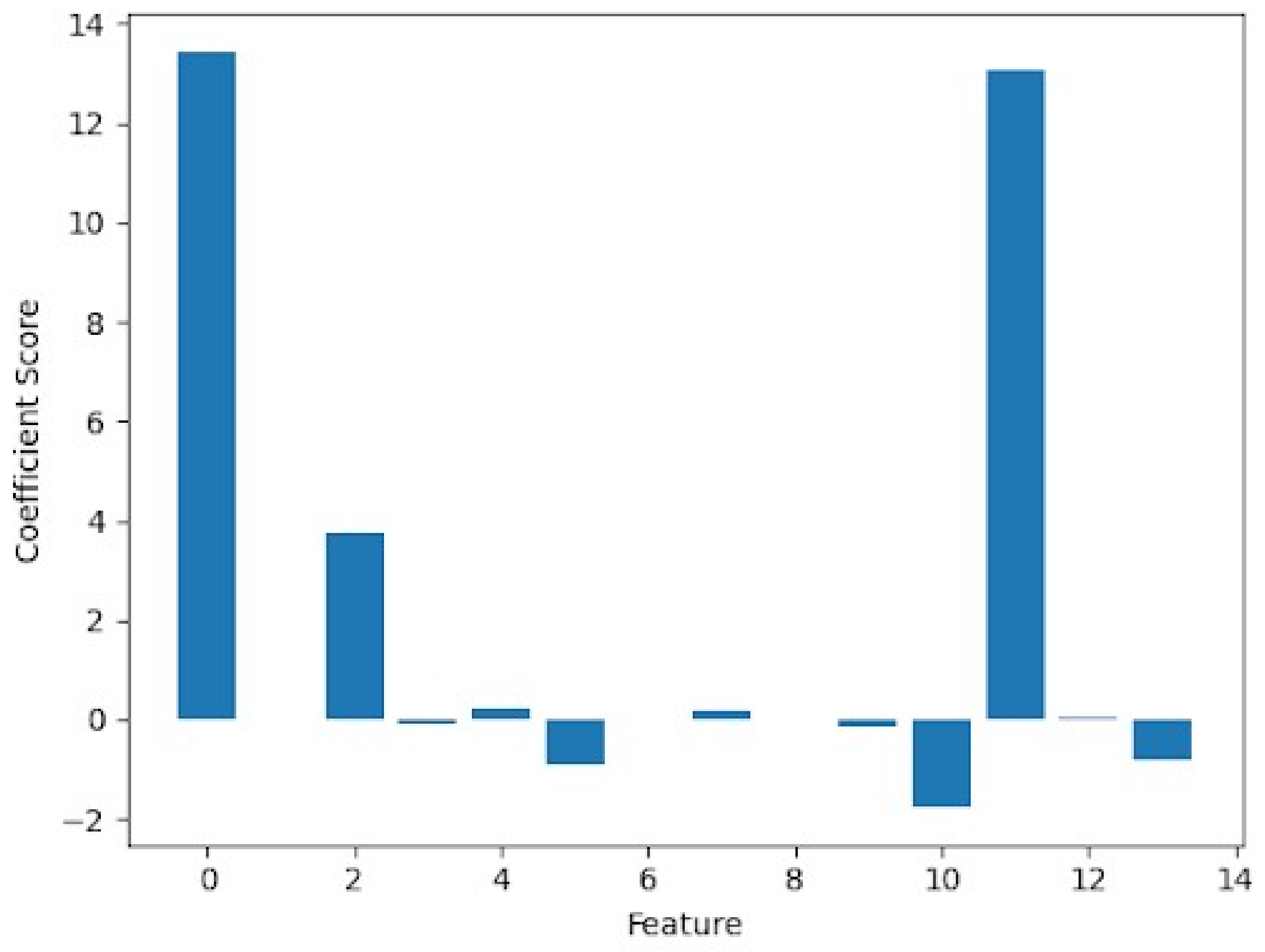
For a max_iter of 100 when using unscaled data, we see that our model relies heavily on one feature, nan_char_entropy (feature 11), with some influence from url entropy(feature 2) and dash-count (feature 7). With one feature primarily having such a large influence, we believe that we must scale the feature.



Feature Reliance

Scaling Data

Another feasible approach to overcome the convergence problem was to scale our data before running Logistic Regression. This was also helpful in our model not relying largely on one feature (feature 11: nan entropy) as our scaled model produced a more varied feature importance.



Feature Reliance on Scaled Data

Feature results:

	Feature	Score
0	Feature: 0	13.8203
1	Feature: 1	0.0000
2	Feature: 2	3.8932
3	Feature: 3	-0.0899
4	Feature: 4	0.2211
5	Feature: 5	-0.8904
6	Feature: 6	0.0006
7	Feature: 7	0.1926
8	Feature: 8	0.0088
9	Feature: 9	-0.1229

	Feature	Score
10	Feature: 10	-1.7600
11	Feature: 11	12.9376
12	Feature: 12	0.0245
13	Feature: 13	-0.7930

We observed that now our most important features were url_length (feature 0), url_entropy (feature 2), and nan_char_entropy (feature 11). Other features like dot_count(feature 5), subdomain_count(feature 10), and domain_age_days (feature 13) also had some influence.

Considering that url_length (feature 0) is often associated with malicious URLs and that it proceeds in the positive direction for the coefficient, malicious URLs would be toward the positive end sigmoid function's output. Similarly, domain_age_days, which proceeds in the negative direction, would indicate how long this URL has been active which is commonly associated with legitimate URLs. Thus, we can conclude that having a higher value on some features might indicate that they are malicious vs. others that indicate a legitimate URL: we are able to classify features. This indicates that higher the url_length (feature 0), url_entropy (feature 2), and nan_char_entropy (feature 11), the higher the likelihood that it is malicious. Similarly, the higher the dot_count(feature 5), subdomain_count(feature 10), and domain_age_days (feature 13), the higher the likelihood that the URL is legit. This makes sense because humans often associate the features just like the model has done, indicating that our model captures the essence of the problem well.

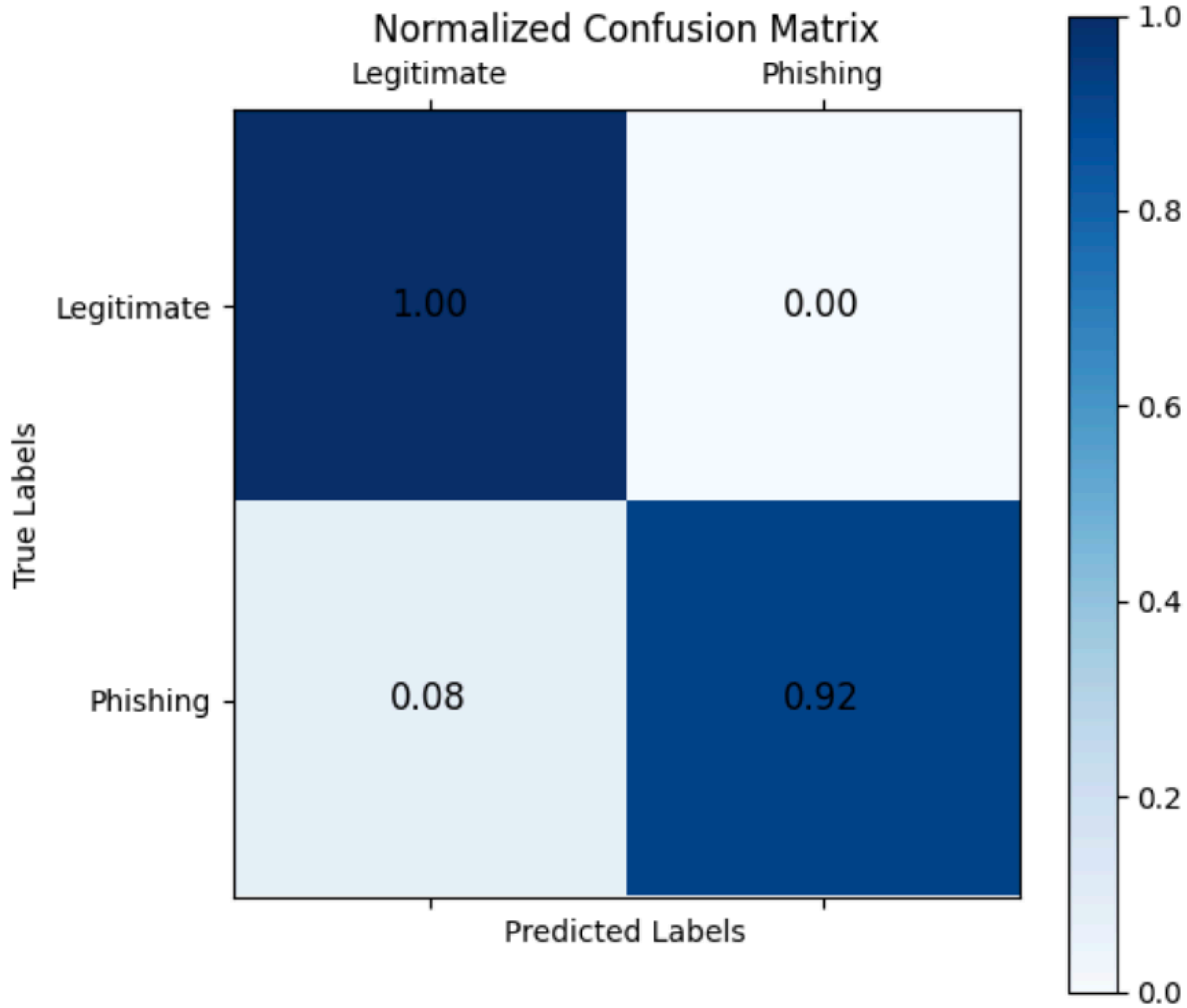
We also saw a slight improvement in accuracy; however, given how it varies on different runs, we can confidently say, the model trained on scaled dataset would give us about the same accuracy as one trained on our original dataset. We also observed that the model ran faster on the scaled dataset.

Quantitative Metrics for Scaled Dataset were observed as follows:

Accuracy: 97.04%

Recall: 100.00%

Precision: 92.05%



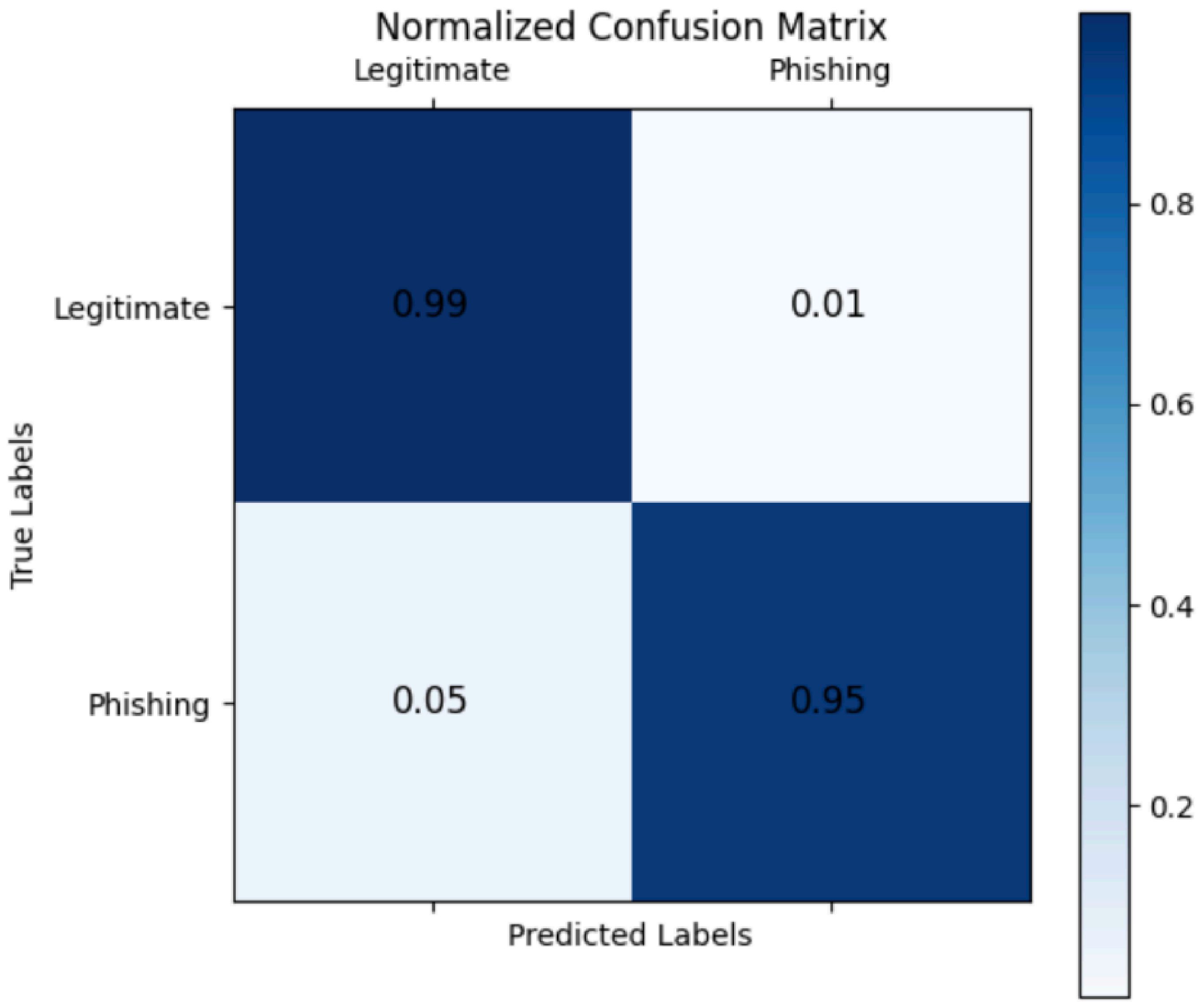
Confusion Matrix for Log Reg on Scaled Data

Discussing Logistic Regression output

The classification algorithm works well for our purpose of classifying our URLs into legitimate vs. phishing as it gives us good quantitative metrics and good output in our confusion matrix. One problem of overfitting that could've been present was avoided by tuning the hyperparameter of maximum iterations. In fact, our Logistic Regression model trained on our Scaled dataset generally gave us similar results to our Unscaled dataset with convergence and faster runtimes. Thus, this model performs quite well, especially the one ran on the scaled dataset, and achieves our expectation of about >95% accuracy and is suitable for achieving our goal of producing a discriminative model to classify URLs. The scaled dataset was great at correctly detecting legitimate labels as indicated by the 100% recall. This could potentially indicate overfitting. Since the model primarily relies on url_length (feature 0), url_entropy (feature 2), and nan_char_entropy (feature 11), it could also indicate that our legitimate urls have a similar structure. The model had a small issue with phishing urls being classified as legitimate. This could potentially be due to malicious urls having a physical appearance similar to legitimate urls. In the context of a malicious URL detection system, we would want a more reactive system, so it is an acceptable result but not ideal.

Random Forest Evaluation

Confusion Matrix

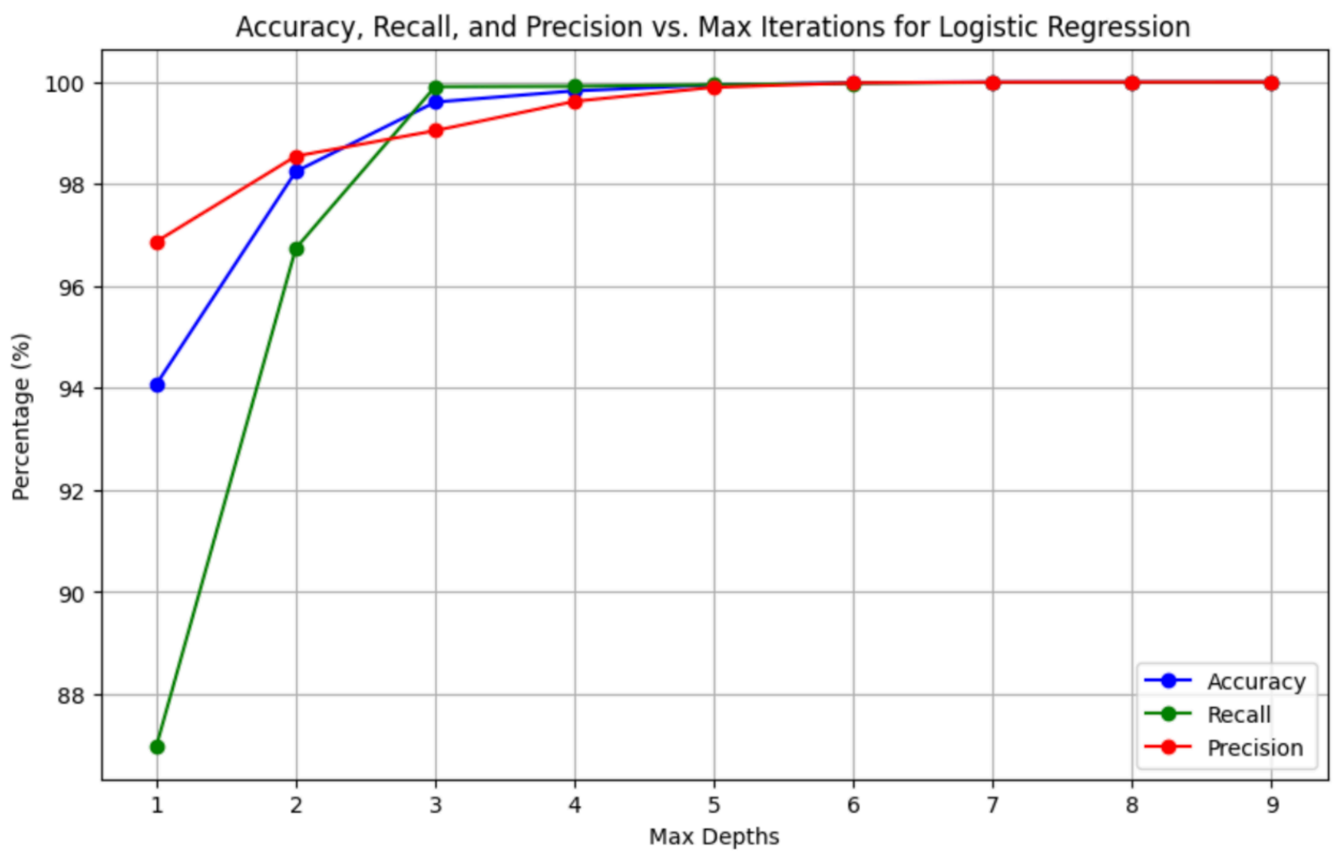


Confusion Matrix for Random Forest

Similar to Logistic Regression, we can see that our model performs quite well: it is able to predict 99% of our legitimate test data as legitimate URLs and 95% of the phishing test data as phishing URLs. Although, we see that 5% of phishing data is predicted as legitimate compared to 1% of legitimate data as phishing: the misclassification of phishing data as legitimate is more than the misclassification of legitimate data, which may be more undesirable as people might trust something that they shouldn't and can be more dangerous. Thus, we can see that our model is not perfect; however, it does perform well-enough for our expectations. In fact, this specific run of Random Forest seems to perform the best out of all our models as we observe the least amount of misclassification in this example.

Hyperparameters tuning for Max Depths

Similar to Logistic Regression, we tuned hyperparameters, specifically max_depth (representing maximum depth that our decision trees can grow to) to avoid overfitting our model. We did this as our model was giving us suspiciously perfect outputs, indicating that it was overfitting. Upon observing the outputs at different maximum depths, we realized that our decision trees only needed to have a height of 2 or 3 to get a good-enough output.



Accuracy, Precision, Recall VS Max Depths for Random Forest

Quantitative Metrics

Accuracy: 97.42%

Recall: 98.35%

Precision: 94.70%

We would like to note that the metrics change a little on model runs done at different times, which could be due to the randomness of feature selection for our different trees.

Discussing Random Forest output

Random Forest works well as a classification algorithm for our purpose of classifying our URLs into legitimate vs. phishing as it gives us good quantitative metrics and good output in our confusion matrix. One problem of overfitting that could've been present was avoided by pruning the tree, which we did by tuning the hyperparameter of maximum depth. Thus, this model performs quite well and achieves our expectation of about >95% accuracy and is suitable for achieving our goal of producing a discriminative model to classify URLs. Similar to Logistic Regression, the Random Forest model had a small issue with phishing urls being classified as legitimate, which could potentially be dangerous. Thus, Random Forest would also be considered acceptable model, though not ideal.

Overall Results

Our results showed that analyzing the structural information of a url is an effective method at detecting whether or not a url is malicious or not. K-Means did not work due to the simplicity of the model, but models such as Random Forest and Logistic Regression provided high accuracy. Features such as url_length (feature 0), url_entropy (feature 2), and nan_char_entropy (feature 11), seem to be the key indicators for whether or not a url is malicious. Both models falsely flagged malicious urls as non-malicious. This indicates that some malicious urls potentially have similar structure to non-malicious urls. There is also some concern for potential overfitting, and these models would have to be tested on real-world data to determine their accuracy.

Next Steps

After implementing and testing our 3 models - KMeans, Random Forest and Logistic Regression - we have determined the strengths of each. While KMeans was deemed unsuitable as we are restricted to having only 2 clusters, Random forest Logistic Regression appears to reflect high accuracy. This is expected as both appropriate hyperparameter tuning and the binary labels of the dataset aid supervised learning models. From here, there are numerous ways we can improve our delivery.

To further make our models robust, we can refine our datasets by including feature reduction and/or better feature selection. Here, we can narrow down on the features that are best for the model, and also explore other features that we could add that improve it. This can serve as a foundation to our feature extraction implementation wherein we can create a dynamic pipeline to parse data from any url gathering features necessary to predict with our model. This pipeline could include implementations in Web Scraping, or more specifically, applications of NLP. Doing so will allow us to implement pipelines that can predict whether any random URL is legit or not by extracting the relevant features and then predicting it against our model.

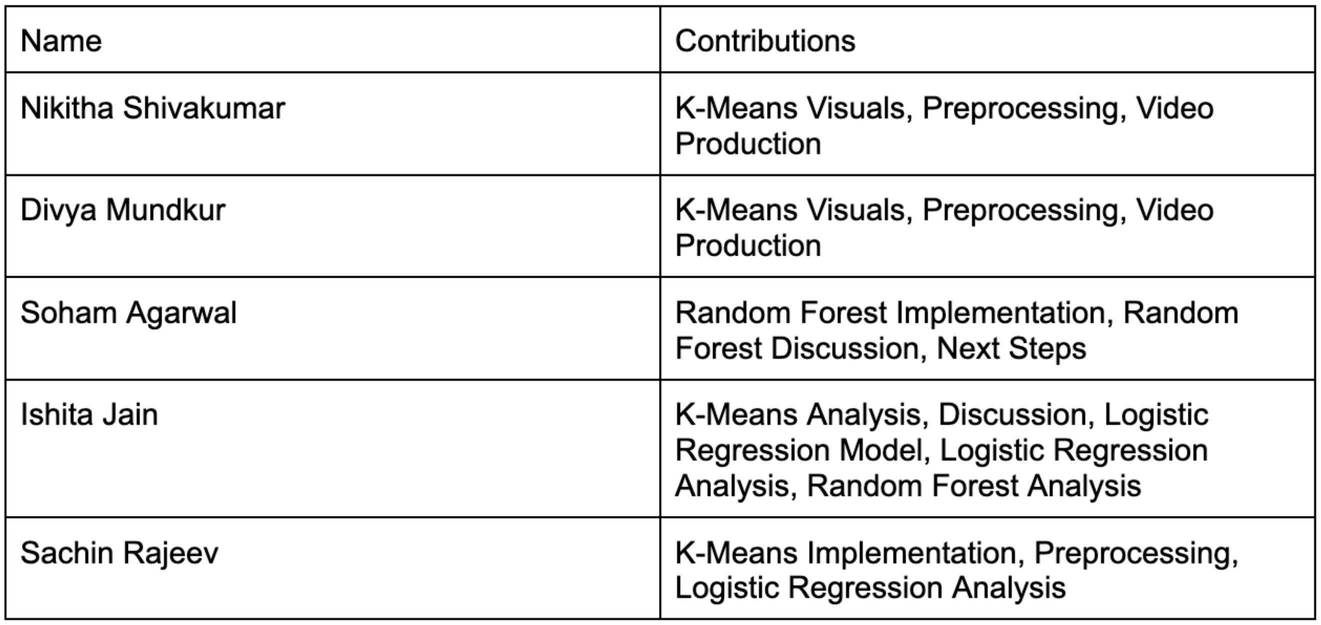
We can also look to stack multiple models, such as Random Forest with logistic regression to provide a holistic output of our work. In doing so, we will be creating a meta-model that synthesizes our predictions of individual models. This could also include training more models such as SVM that stack well with others to be as holistic with our predictions.

We can also broaden the scope of the project by including phishing texts or emails since their underlying characteristics such as linguistic cues, intent, and structure is similar to phishing URLS. Finally, we can make our model more accessible by creating an end-to-end application that hosts the model so that users can input any urls and determine whether or not they are legit.

References

1. C. Cao and J. Caverlee, “Detecting spam URLs in social media via behavioral analysis,” *Lecture Notes in Computer Science*, pp. 703–714, 2015. doi: 10.1007/978-3-319-16354-3_77
2. J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond blacklists: learning to detect malicious websites from suspicious URLs,” in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 2009, pp. 1245–1254.
3. S. Abad, H. Gholamy, and M. Aslani, “Classification of malicious URLs using machine learning,” *Sensors*, vol. 23, no. 18, p. 7760, Sep. 2023. doi: 10.3390/s23187760
4. S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, “A comparison of machine learning techniques for phishing detection,” *Proceedings of the Anti-Phishing Working Group’s 2nd Annual eCrime Researchers Summit*. ACM, Oct. 04, 2007. doi: 10.1145/1299015.1299021.

Contribution Table



Name	Contributions
Nikitha Shivakumar	K-Means Visuals, Preprocessing, Video Production
Divya Mundkur	K-Means Visuals, Preprocessing, Video Production
Soham Agarwal	Random Forest Implementation, Random Forest Discussion, Next Steps
Ishita Jain	K-Means Analysis, Discussion, Logistic Regression Model, Logistic Regression Analysis, Random Forest Analysis
Sachin Rajeev	K-Means Implementation, Preprocessing, Logistic Regression Analysis

Contribution Table

Gantt Chart

Gantt Chart