Predicting Phishing Attacks Based on the URL: A Comprehensive Data Mining Approach

CDS 303-001

Team #2:
<u>Report Coordinator:</u> Rand Alattar
Mohamed Reda Erradi
Ishika Maisha
Ruth Tadesse
Mesfin Mekonnen
Anantha Aksshaj Erigisetty
Junhyung Han

Table of Contents

Abstract

Phishing attacks utilize deceptive tactics to fool people online which can cause incalculable financial and privacy damage to corporations. This paper aims to identify the most common traits of a phishing attack to alleviate their impact. Data mining techniques introduce a model to identify the most common phishing website traits to provide corporations with import tool limit hacking attempts.

Introduction

As technology advances, phishing tactics have become more common which leaves internet users at a risk of data theft. Phishing is a direct attack to steal sensitive information. This study seeks to address this issue of phishing through a machine-learning model that classifies URLs into legitimate and phishing categories.

Finding an effective way to limit the damage of these phishing attacks is crucial. This study implements the Cross Industry Standard Process for Data Mining (CRISP-DM), a scheme taught to us in class. We will begin by explaining our business problem, which provides us with a reason to go through this process. We will then explore various features through our fishbone diagram and quantify the chosen features through our Black Box Diagram. The next step will be to explore our data to learn its characteristics, which will help us determine the best modeling algorithm. We will then explain the steps that were taken to choose and build our model and interpret its results. The last and final step of this process will be evaluation and deployment where we interpret our business problem and explore methods of getting our work into the hands of others.

Business Problem

The business question we will be focusing on is “How can businesses and organizations identify if a URL they have clicked on is part of a phishing attack?”

The question we are studying is crucial since phishing URLs pose a large threat to businesses and organizations which are at a high risk of these phishing attacks due to their handling of sensitive information. This is typically done through sending an email to an employee inbox where they then click on the phishing link.

While it is ideal that individuals avoid clicking on a phishing URL altogether, distinguishing between a dangerous and legitimate website can be challenging. In a study done in 2012 by MIS Quarterly, it was found that, in most cases, anti-phishing software falls below 70% accuracy in detecting false websites. While software has improved in the years since, no model is perfect and false positives and negatives can still occur. While incorrectly labeling a safe site is certainly an ordeal for the website's owner, unflagged phishing websites can have a far worse impact, and prove to be a constant problem. Understanding signs of phishing allows one to mitigate the harm of these phishing attacks by clicking out of a link immediately once the signs of phishing reveal themselves.

Analytic Question

Based on the business problem, our analytic question is: "What are the most frequent features that appear with a phishing website in both its URL and its behavior with the user?" We created a fishbone diagram and a Blackbox diagram to answer this question. These diagrams help break down the problem into specific inputs (features) and outputs (target), guiding the development of our predictive model.

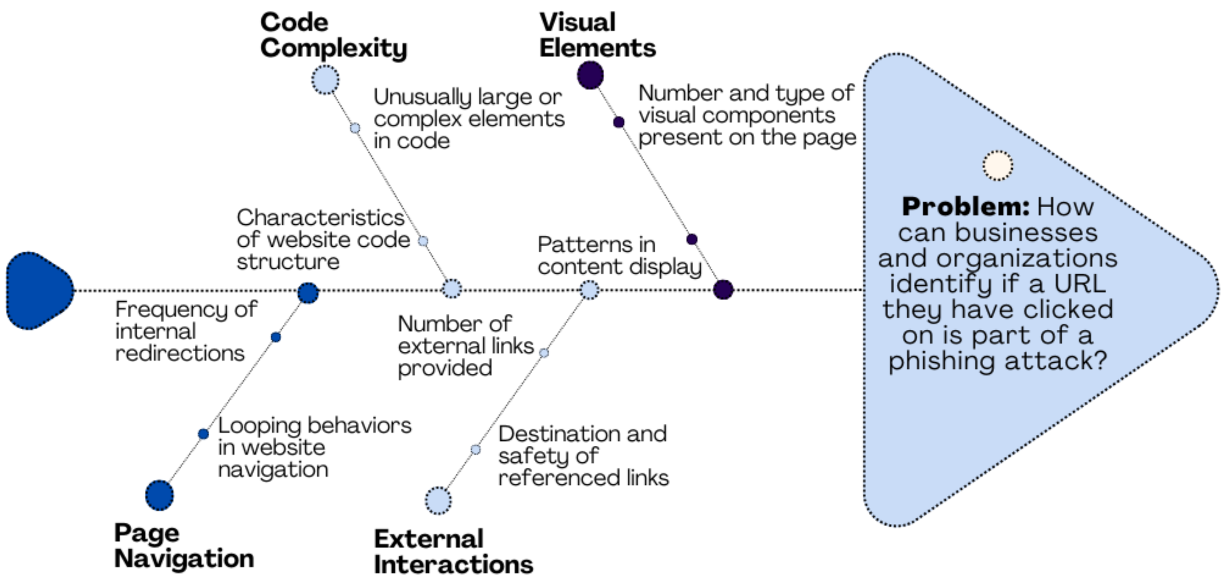

Code Complexity
Unusually large or complex elements in code
Characteristics of website code structure
Visual Elements
Number and type of visual components present on the page
Patterns in content display
Problem: How can businesses and organizations identify if a URL they have clicked on is part of a phishing attack?
Frequency of internal redirections
Looping behaviors in website navigation
Page Navigation
Number of external links provided
Destination and safety of referenced links
External Interactions

Figure 1: Fishbone Diagram

The fishbone diagram identifies four key categories of features that influence whether a website is phishing or legitimate: Code Complexity, Page Navigation, Visual Elements, and External Interactions. Each category highlights specific behaviors or characteristics often seen in phishing websites:

- Code Complexity: Examines unusually large or complex code elements and the structural characteristics of a website's code.
- Page Navigation: Focuses on behaviors such navigation loops, frequent redirects, and unusual content display patterns.
- Visual Elements: Visual elements take into account the number and type of visual components on the page, which may differ significantly in phishing attempts.
- External Interactions: Considers the number of external links and assesses the safety and destination of these links.

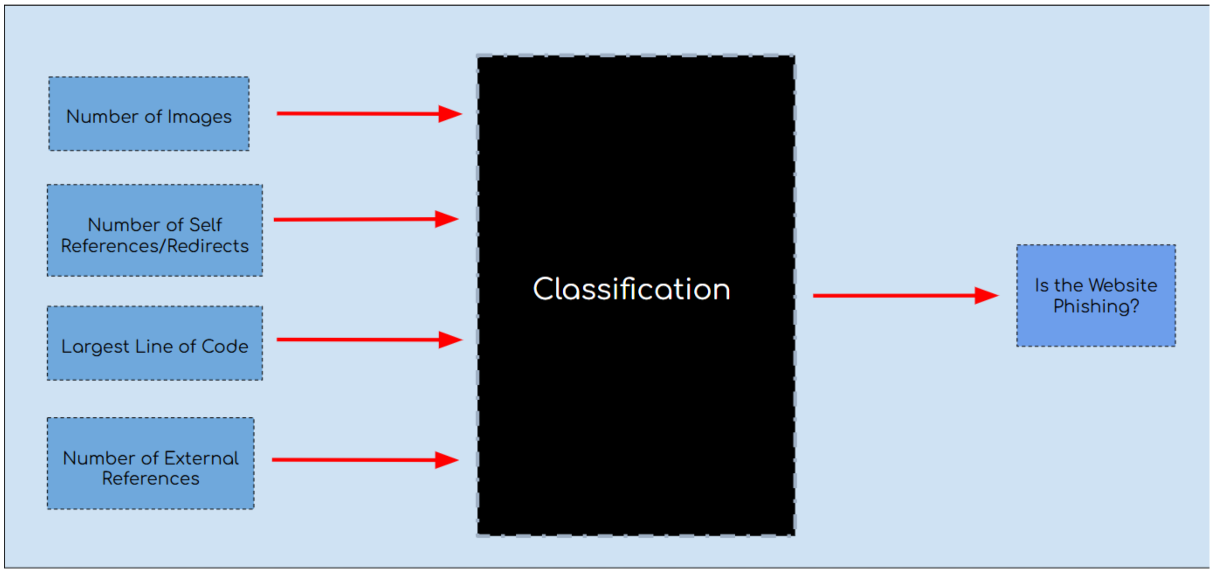


Figure 2: Black Box

The inputs above were chosen from the brainstorm which was done in our fishbone. They were chosen based on which columns were available for us to use in our dataset. We are only identifying whether a URL is phishing or not, which means that there are only two outputs that can happen (As shown in Figure 2). Since what we are identifying happens to be categorical, the approach we must take is classification. We decided on using a model that would utilize binary classification. While we tested other models, we decided on the Support Vector Machine as our main algorithm. While this modeling algorithm may be difficult to interpret, we presented the model in a simple way to make it easier to describe. We will have this model's output be compared to a threshold of 0.5. This threshold will determine how malicious a website input (into the model) happens to be. If a website's likelihood exceeds this threshold, then the website will be classified as phishing. If not, then the website will be classified as not phishing.

Data Background and Feature Description

We chose the dataset called PhiUSIIL_Phishing_URL because we felt that it had many options for features to choose from. We were provided with this dataset by our professor; however its origin lies in the UC Irvine Machine Repository. It was donated by two scholars, Arvind Prasand and Shalini Chandra from Babashaheb Bhimrao Ambedkar University in Lucknow, India. The data and features we used in our analysis were used from the dataset the professor provided us. This data set includes 235,796 rows and 56 columns of data providing us with a wide range for our analysis. The dataset contained data on URL features describing the characteristics of the URL. These features we decided on include the number of images found on the website, the number of self-redirects, the largest line of code, and number of external references. These features will be used to analyze and compare URLs to determine the potential of a phishing attack. Our target variable will be a column called label which identifies whether a website is part of a phishing attack.

*Table 1: C*olumn names and definitions for the 4 features and target variable

Feature and Target Columns we will be using				
Feature and Target	Column Name	Definition	Range	Data Type
Number of Images Found on the Website	NoOfImage	Number of images found on the website	(0, 8956)	Integer
Number of Self-Redirects	NoOfSelfRef	This is a count of the number of times the URL redirects back to itself on it's own page.	(0, 27397)	Integer
Largest Line of Code	LineOfCode	The largest line of code associated with the link	(2, 442666)	Integer
Number of External References	NoOfExternalRef	This is the count of external references. This is an external page which the user is directed to outside the website that is initially clicked on.	(0, 27516)	Integer
Label (Phishing Indicator)	label	This is our target variable, and it shows whether or not the website is phishing. 1 is a legitimate URL and 0 is a phishing URL.	(0, 1)	Boolean

Each of the features chosen, in their own way, contributes to an understanding of how to better identify websites involved in phishing. Table 1 provides a summary of the data types, definitions, column names, and range of data.

The Number of Images on a website often points to the attempts to trick the user present. While normal websites use a varying number of images depending on purpose, phishing websites often use an excess of images to attempt to fool the user. This method is a well-known phishing tactic which is why we chose it as one of our key features to use in our model.

The Number of Self-Redirects refers to how many times a website directs back to itself. This is done to disguise the true nature of a website by showing a different layout before using several redirects to change the website. A higher number of these self-redirects increases the chance of suspicious behavior increases.

While the other two refer to ways to find phishing, the Lines of Code is a general indicator of the amount of optimization going into a website. While proper websites tend to have relatively efficient code done by professionals, phishing sites merely attempt to imitate a professional website. Typically, phishing websites lack the same level of refinement when it comes to their code. A legitimate URL may achieve its objectives using a shorter code which is more efficient while a phishing URL often relies on longer code. We felt that having this variable as a key feature would be important as it looks beyond the surface of a website and goes right into the coding that has built up the site.

The Number of External References refers to the number of unrelated links that are on a phishing site. These links are included as an attempt to trick the user into unrelated websites to steal their information. A higher number of these external pages will generally correlate to a higher chance of being a phishing website. This feature contrasts with the number of self redirects, giving another perspective of how a person can be involved in a phishing attack. Therefore, we felt this variable was necessary to be in our list of chosen features.

Data Cleaning and Exploration

The data cleaning aspect of this project was tedious as there were various aspects to look for. The first step of this process was loading the dataset and investigating the data to better understand it. We explored the columns, data types, shape of data, summary statistics, and other details. We found that there were no missing values in the dataset, therefore there were no imputations or estimations involved for any value in the data. However, one of the main initial concerns for this data was the extreme outliers, which will be discussed in this section.

We started our cleaning process by identifying all the data types, ranges, and definitions for the columns in our dataset. This information can be found in Appendix 1. We identified five categorical columns in our overall dataset which we dropped since we did not think that they would contribute to our analysis. For example, the dataset included a column for the filename and URL which would not be of use to us in both our data exploration and modeling stages. We also decided on dropping the remainder of the columns which are not our either our feature or target to simplify our data frame.

To gain a better understanding of our data, we generated summary statistics of our selected features as displayed in Table 2. Many of the columns have a large range in their values. For example, the NoOfImage max is 8,956 while the min is 0 and the mean is low at 26.07. There are also large standard deviations for our features, indicating wide ranges and extreme outliers. All our feature columns have a much larger mean compared to the median values meaning our data will be positively skewed. Based on the summary statistics alone, we can see that some work needs to be done to scale our data and deal with the extreme outliers.

Table 2: Summary Statistics

	NoOfImage	**NoOfSelfRef**	**LineOfCode**	**NoOfExternalRef**
count	235795	23579	235795	235795
mean	26.07	65.07	1141.9	49.26
std	79.41	176.68	3419.95	161.02
min	0	0	2	0
25%	0	0	18	1
50%	8	12	429	10
75%	29	88	1277	57
max	8956	27397	442666	27516

Another thing we checked for in our data was any class imbalances in our target column as shown in Figure 3. We found that there was a small imbalance of 0.57 to 0.42. This means that there is a larger proportion of legitimate websites compared to phishing sites. Since this imbalance is quite small with a difference of only 0.14379 units, it was not an issue we were too concerned about. However, upon beginning our model building and testing, we had some issues with overfitting within the legitimate URL class (1). To fix this issue, we divided our training and testing data using a Stratified K-fold method. This ensures that each “fold” that is samples has an equal number of classes for our target variable.

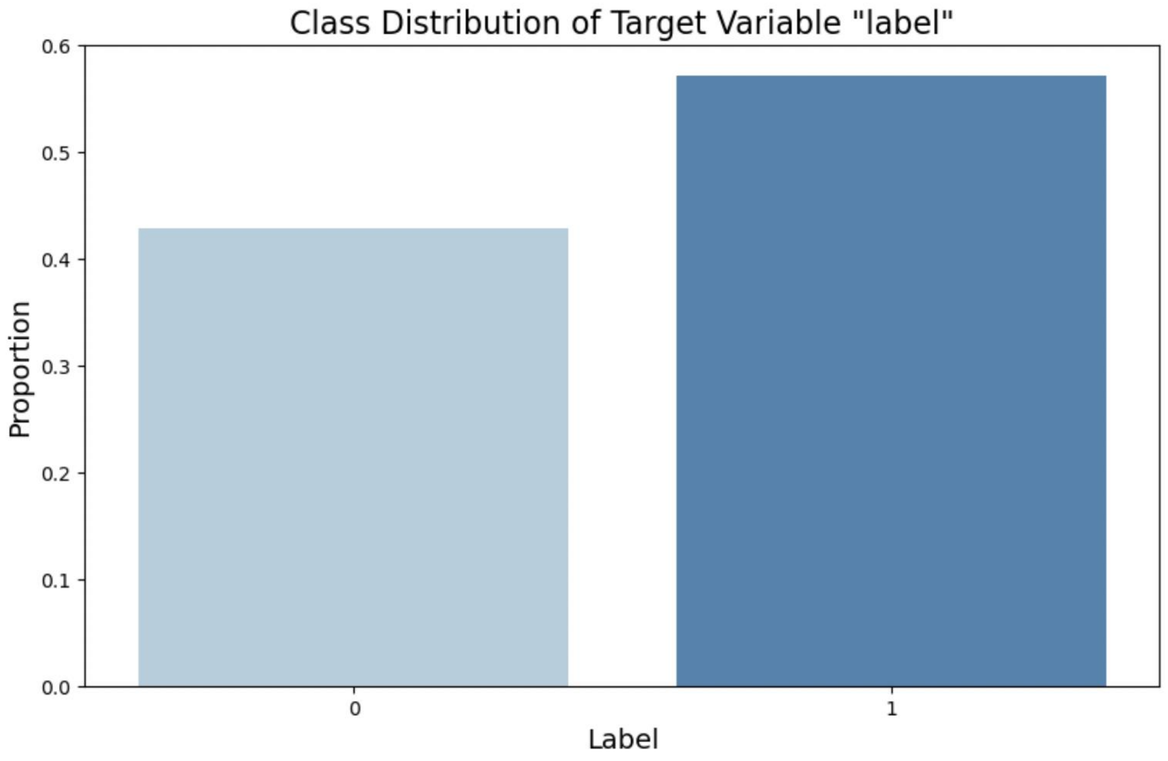


*Figure 3: C*lass Imbalance

As we discussed in our summary statistics, we had large ranges and extreme outliers which may impact our model depending on which one we choose. Figure 4 displays a box plot for each of our features showing the distribution of data. Our first step is to determine which data points were considered outliers. This was done by calculating the IQR (interquartile range) and using that to determine which points are outliers. We did Q1 minus 1.5 times the IQR and Q3 plus 1.5 times the IQR. Any value outside the calculated range would be considered an outlier. We then investigated the outliers to determine whether they would be feasible or not. Through our examination, we discovered that the outliers were feasible. This was done by researching whether the extreme outliers were sensible. For example, the maximum value for the line of code was 442666. Although this value is quite high compared to the remaining points, as long as the code is properly formatted the value is feasible. Though we were initially concerned with keeping in the outliers since we thought it may impact various models that were sensitive to them, we tested these models both ways and found that the outliers did not have an impact on the models.

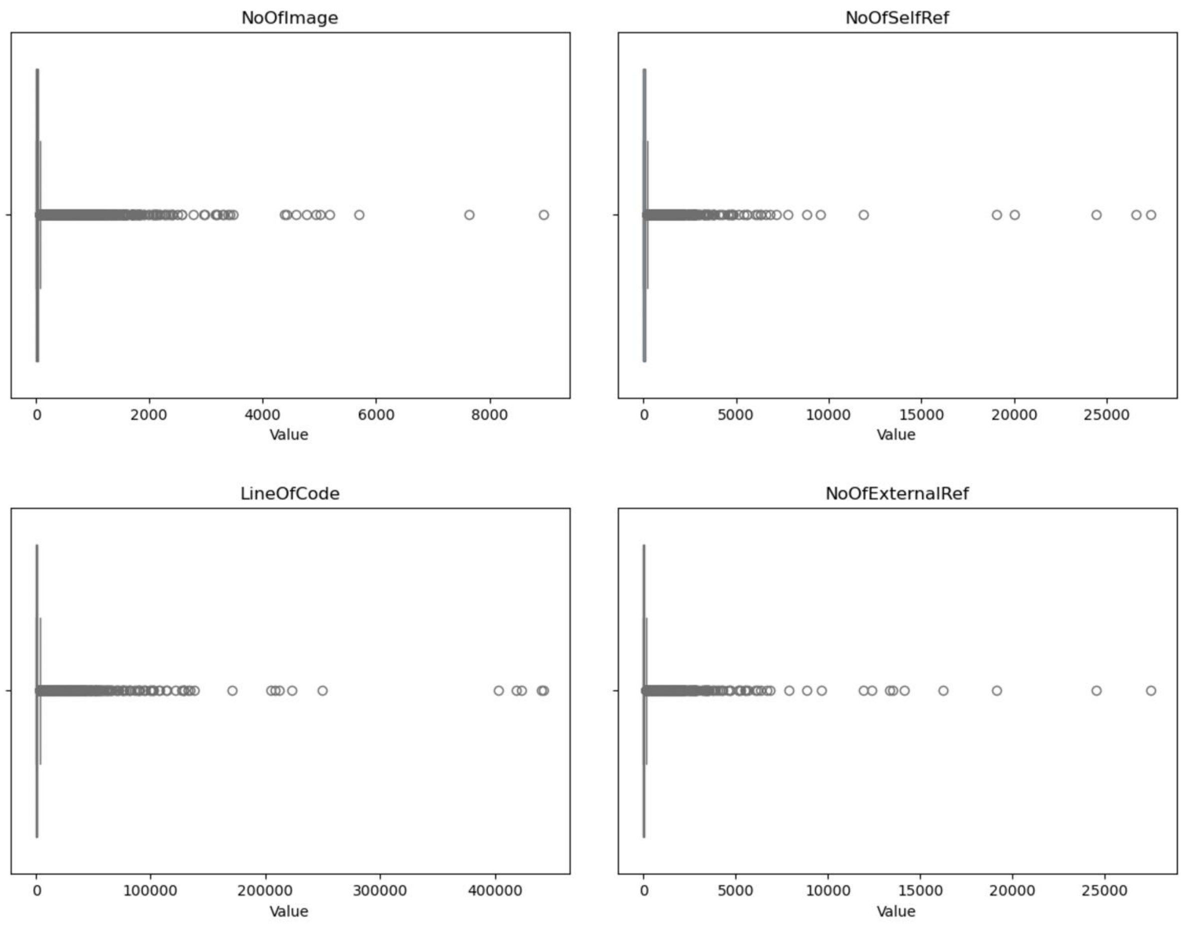


Figure 4: Boxplot of Features

The next thing we did was check for non-normal distributions. We created histogram subplots of each of the features to help us identify patterns in the features as displayed in Figure 5. When creating the initial plot with the data, it was difficult to get a look at the data due to the extreme outliers. Removing these outliers was an essential step that helped us get a better view of the data. Although we removed the outliers to create these visuals, we did not remove the outliers during our modeling process. We included the figures of both the histogram and the ridge plot that contains the outliers in Appendix 2 for reference.

As expected, based on our summary statistics, the histograms were heavily skewed to the right, suggesting our data falls at lower values. We decided to normalize our features using Min-Max Scaling and save it to a new CSV file that can be used on our model. Min-Max Scaling changes the features to fall between 0 and 1 that way each value has an equal opportunity to be considered when modeling. Figure 6 displays a ridge plot which shows the distribution of our features after scaling. Our target feature has bimodal distribution with two peaks displaying the distribution of the two classes. While some features remain skewed, the overall scale of the features are more consistent to what had been prior to normalization. The non-normal data distribution should not be of too much concern since many classification methods do not assume normality.

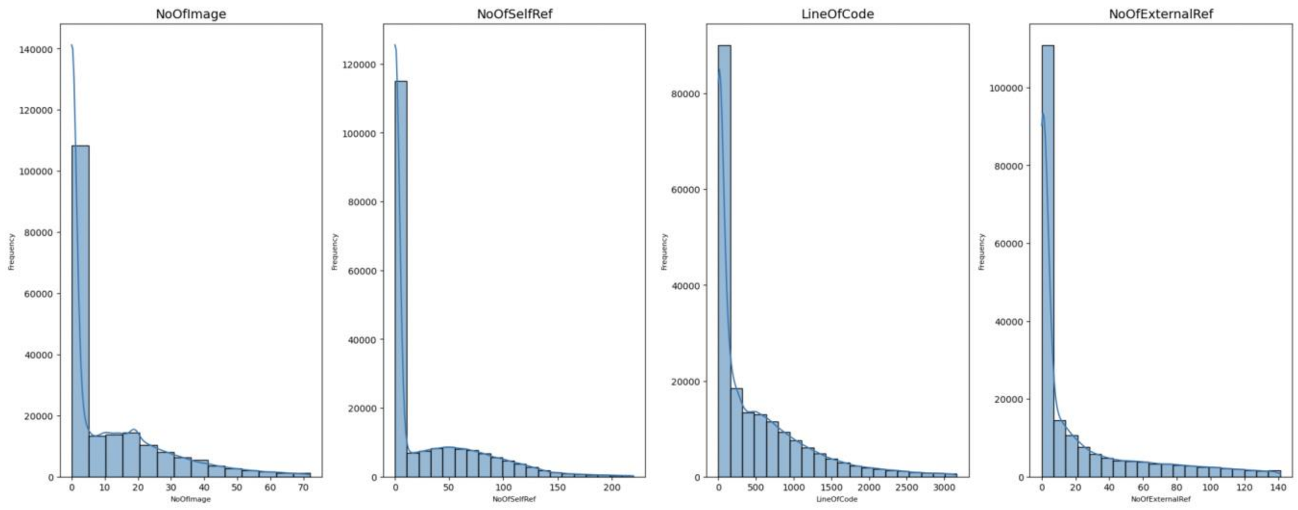


Figure 5: Distribution of Data with the Removal of Outliers

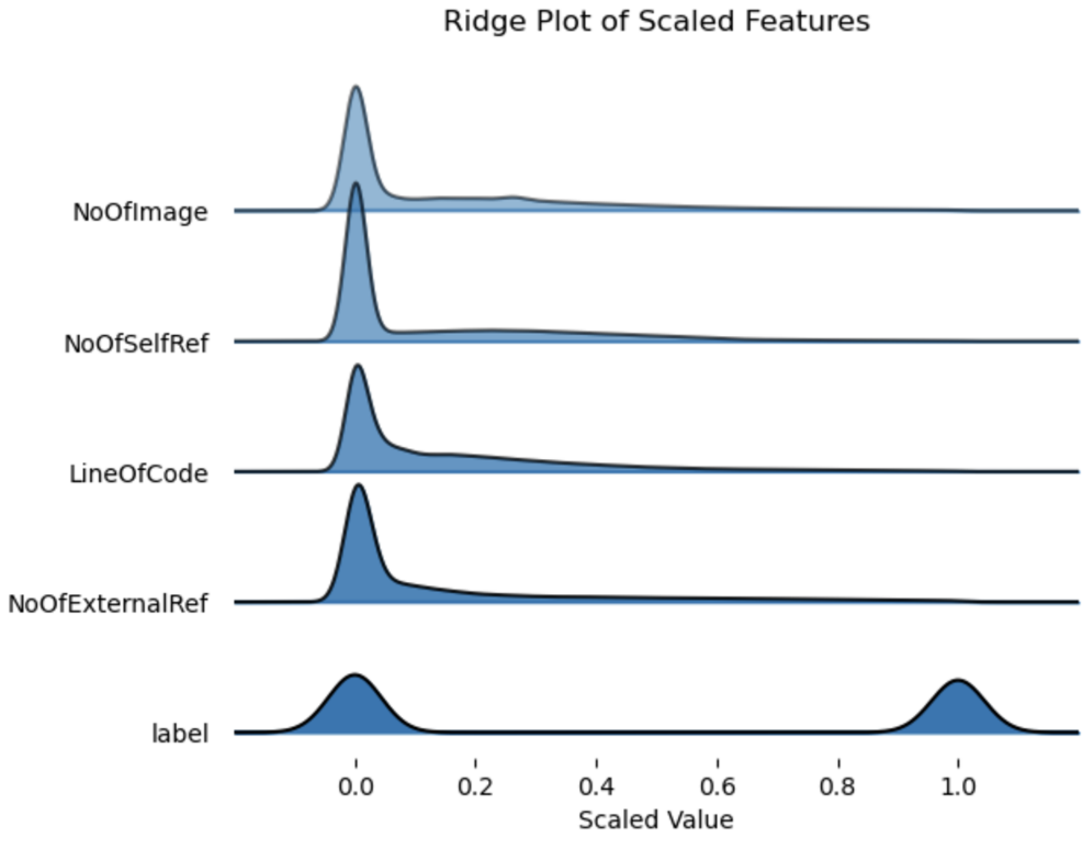


Figure 6: Ridge Plot of Scaled Features with the Removal of Outliers

Another thing we checked for is multicollinearity within our features. We used the correlation matrix to visualize the relationship between selected features and targets in the dataset as seen in Figure 7. It shows the correlation between two features from a minimum of -1 to a maximum of 1. A stronger correlation is indicated by a darker blue and a weaker correlation is indicated by a light yellow. Two features are positively correlated, NoOfSelfRef and NoOfExternalRef. This means that as the number of self-references increases, the number of external references increases. This correlation makes sense since they are both types of references The remaining features are weakly correlated (less than 0.5). We do not have any features that are highly correlated with the target variable. We previously had one feature that was correlated to our target variable, the URL similarity index. We decided to drop that feature due to issues with overfitting later in our modeling process.

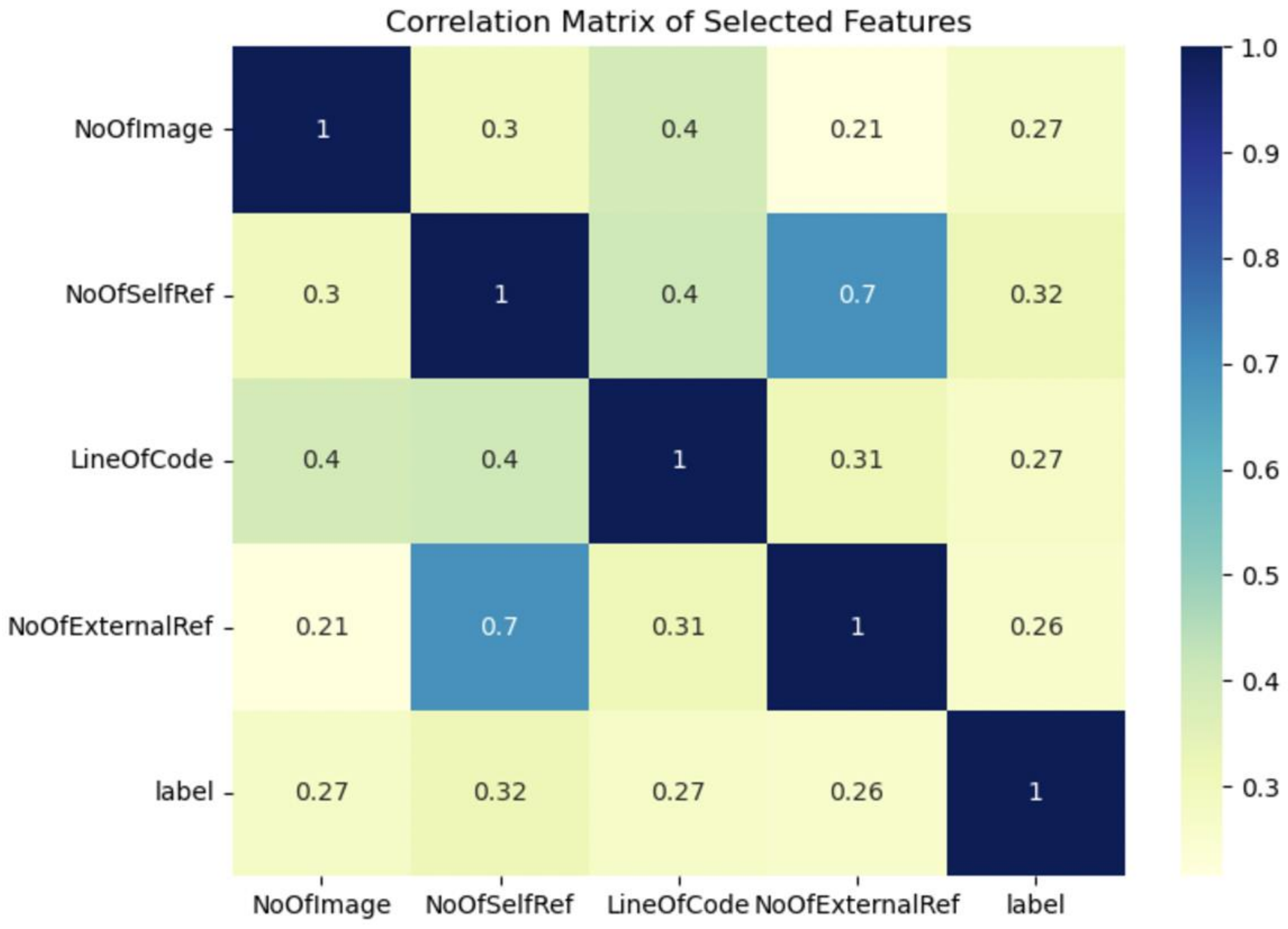


Figure 7: Correlation Matrix

The pair plot in Figure 8 shows the relationships between several features in the dataset. The scatterplots reveal how our features correlate to each other. As we can see, many of the features have no relationship or a non-linear relationship. Two features that appear to have a linear relationship are the number of self-redirects and external redirects. This relationship confirms what we discovered in the correlation matrix earlier. Many of the data points for the features are concentrated at the lower end at zero, with extreme outliers present towards the higher ranges.

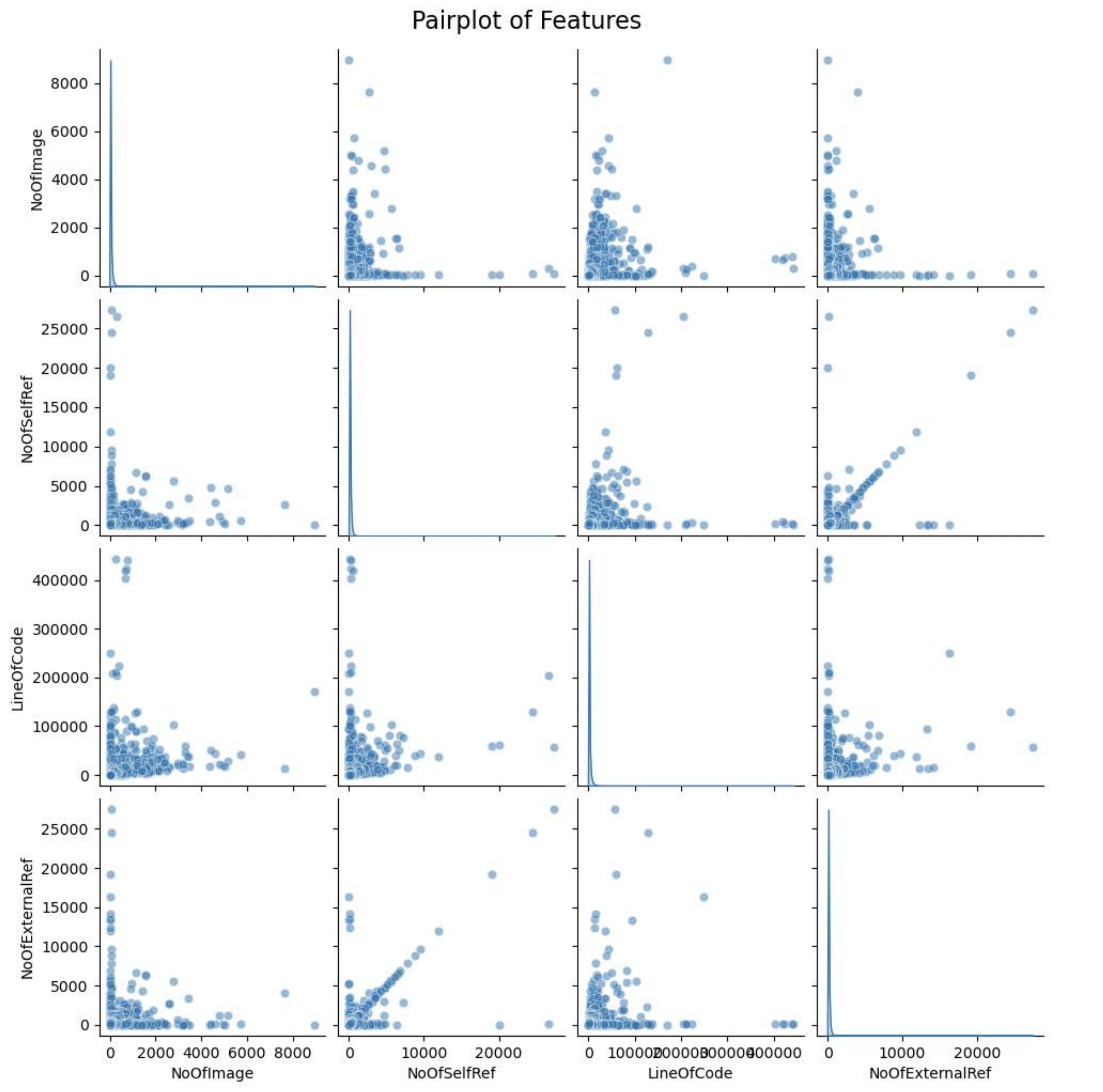


Figure 8: Pairplot

Figure 9 displays a violin plot which shows our data after normalization and the removal of outliers. Violin plots are a cross between a Boxen plot and Kernel Density Estimator. A small boxplot is displayed in the grey area in the middle of each plot. Many of the boxplots are still skewed to the right despite the normalization and the removal of outliers. This should not be too much of an issue since we can choose a modeling algorithm that is not very sensitive to non-normal data. A version of the violin plot which included the outliers is also included in Appendix 2 for reference.

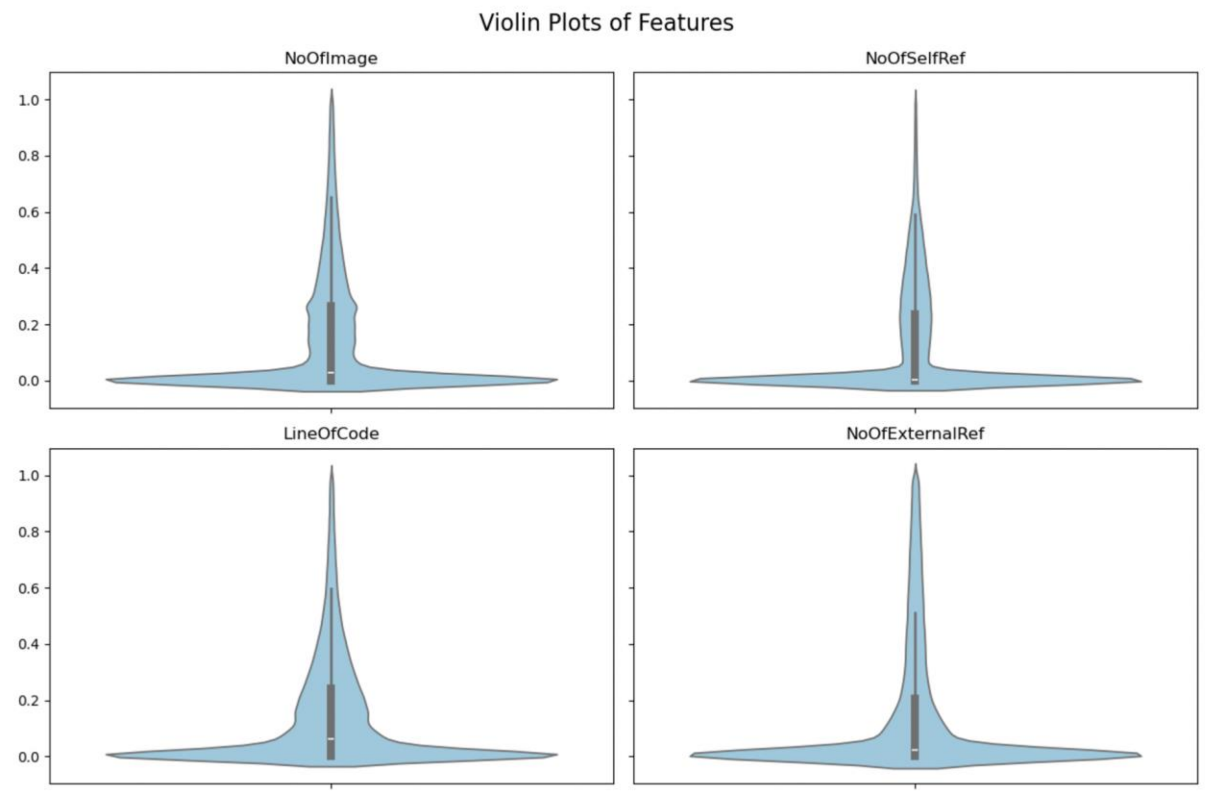


Figure 9: Violin plot with the Removal of Outliers

Now that we have finished cleaning and exploring our data, we can begin our modeling process.

Model Building

We have chosen a classification algorithm and had 6 choices for model approaches. In building our model, we had to determine which algorithms would be best for our data as each has their own advantages and disadvantages. Below we explain the advantages and disadvantages of the various algorithms we tested and considered.

1. Random Forest/Decision Tree: The Random Forest Algorithm was a great option for our data due to its ability to handle extreme outliers and scale to larger datasets. Though Random Forest models are notorious for overfitting which is an issue we ran into during our modeling.
2. Neural Network: Neural Networks were a great algorithm for our non-linear data as it is designed to handle non-linear settings. This algorithm does not require normal distributions, which is an advantage since our data is not normal. This model also has high scalability, which is necessary for our large dataset. However, one disadvantage is that it can be moderately sensitive to outliers which we had in data.
3. Naive Bayes Classifier: Naive Bayes was an overall solid choice for our dataset based on its traits. This model is robust to outliers which is important since we previously established that our data had extreme outliers. This model also has high scalability, which is necessary for our large dataset. However, this model can be sensitive to non-normal

distributions where are present in our data. We can improve the accuracy of this model through normalization which we have done in previous preprocessing steps.

4. Support Vector Machines: The performance of Support Vector Machines is mainly dependent on the kernel and parameters. This algorithm has the ability to handle data regardless of its linearity, providing flexibility. This model is not impacted by non-normal distributions, making it a great option for our data. One disadvantage was the model's sensitivity to outliers which were present in our data.

Model Testing and Results

We tested four different machine learning algorithms to ensure that we got great results. These three models include Naive Bayes, Neutral Network, Random Forest, and SVM. Although we tested a few different algorithms we decided that Support Vector Machine (SVM) would be our main algorithm. This is due to its ability to handle complex, high-dimensional data and its ability to handle binary classification problems. This model type can also handle a substantial amount of noise when the proper kernel is used. Although this algorithm is sensitive to outliers, it surprisingly worked well with our dataset despite keeping them in. When building our model, we decided to run the data with and without outliers to ensure the best results. The results for both the data with outliers and without outliers yielded similar results. Though we had tested four models, we decided to only showcase the results of three models due to overfitting with the Random Forest model. These issues with overfitting were partly due to one feature, the URL similarity index which we decided to drop. Even after scaling, it was highly skewed, with most URLs showing high similarity scores whether they were phishing or not. This made it pretty useless for helping the model distinguish between legitimate and phishing URLs which ultimately lead us to making the decision of dropping it. This feature had caused all our models to overfit and removing them yielded better results. Essentially, these decisions helped us build a simpler, more reliable model that focused on features that genuinely added value. Despite dropping this feature, the metrics of the Random Forest model would not reduce lower than 99% which is why we ultimately decided to omit that model from our analysis.

For each of our models, we isolated the label column containing the target variable for training purposes. We focused on fine-tuning key hyperparameters like the regularization parameter (C) and the kernel type to get the best performance from our SVM model. The C parameter was set to 1, the kernel was set to rbf, and gamma was set to scale. The probability was also set to true to enable predictive probabilities. A more practical approach by iteratively testing parameter ranges based on what we knew about the data and insights from early experiments. We mainly focused on explaining the hyperparameters for SVM since that was our main model. Table 4 displays the hyperparameter settings which were used for each modeling algorithm. For further details on the purpose of these hyperparameters, refer to Appendix 3. Although our class imbalance was quite small, we used a Stratified K-Fold Cross-Validation due to some issues with overfitting in class 1 (legitimate URLs). This kept the class distribution consistent across all splits, ensuring fair evaluation and eliminating overfitting in either class.

Table 4: Model runs and best parameters

Algorithm	Best Parameters
Support Vector Machine	C [1.0], kernel [rbf], gamma [scale], probability [True]
Neural Network	activation: ['relu'], alpha: [0.0001], hidden_layer_sizes: [(50, 30)]
Naive Bayes	var_smoothing [1e-08]

Table 3: Test Scores of the Three Algorithms (Class 0: Phishing)

Model	Mean Accuracy	Precision	Recall	F1-Score
Naive Bayes	96.25%	0.9450	0.9689	0.9568
Neural Network	98.65%	0.9868	0.9816	0.9842
SVM	98.02%	0.97	0.98	0.98

The test metrics are shown for the three algorithms we tested in Table 3. We chose to only include the results for the phishing URL class since our business question focuses on that group. From our results, we can see that the Neutral Network and SVM models performed the best without overfitting.

Since we had decided on a SVM as our main algorithm, we included the test results in further detail as displayed in Table 5. More details on the results for the Neutral Network model and the Naive Bayes can be found in Appendix 4 and 5.

Table 5: Test Set Evaluation SVM

	Precision	Recall	F1-score	Support
0 (Phishing)	0.97	0.99	0.98	20124
1 (Legitimate)	0.99		0.98	27035
Accuracy			0.98	47159
Macro Avg	0.98	0.98	0.98	47159
Weighted Avg	0.98	0.98	0.98	47159

Following these metrics, we further assessed the model's performance by analyzing the Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) as shown in Figure 10. The ROC curve was generated to assess the model’s ability to distinguish between the classes and shows an upward trend toward the top-left corner. The AUC score, which summarizes the performance of the classifier was 0.99 indicating that the model effectively differentiates between positive and negative classes with high accuracy.

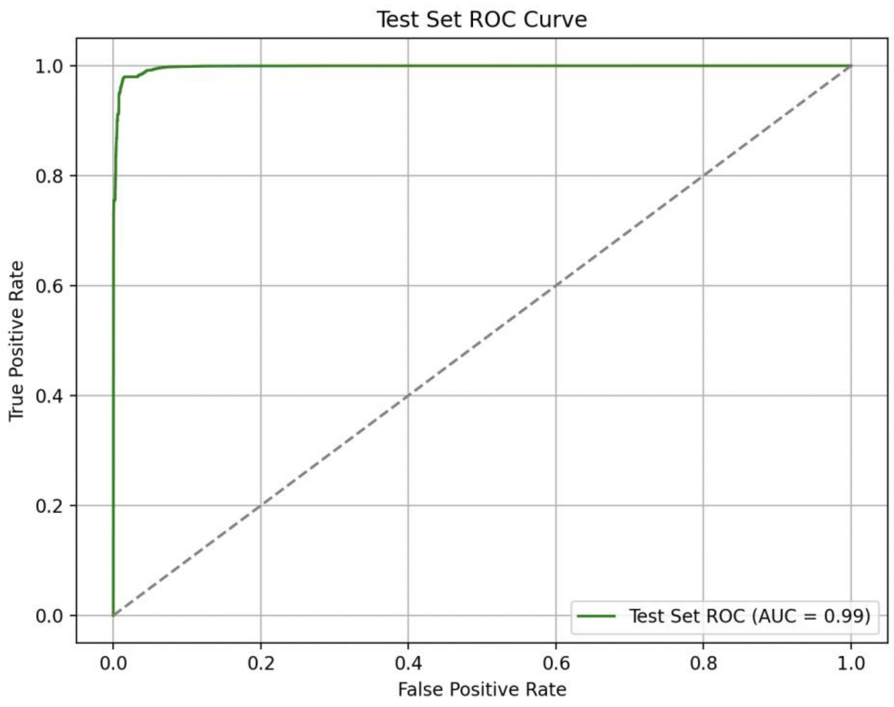


Figure 10: ROC Curve for SVM

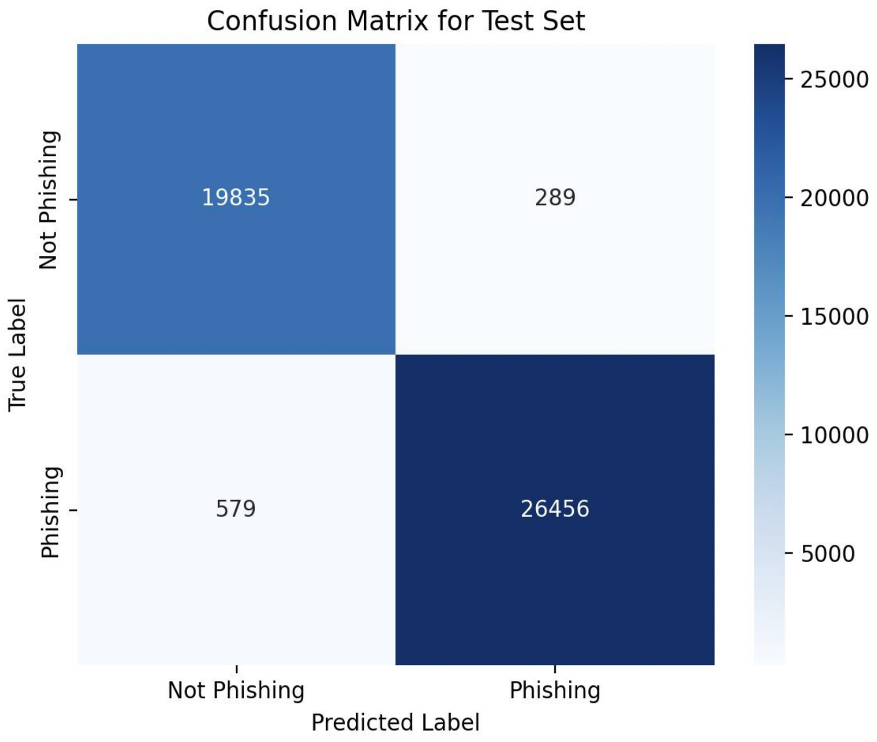


Figure 11: Confusion Matrix for SVM

Above in Figure 11 we have our confusion matrix. We have listed the definitions of the metrics true positive, true negative, false positive, and false negative in context with our study.

- **True Positive:** Both the actual and predicted values are correctly identified as phishing.
- **True Negative:** Both the actual and predicted values are correctly identified as not phishing.
- **False Positive:** The predicted value is phishing, but the actual value is not.
- **False Negative:** The predicted value is not phishing, but the actual value is phishing.

This confusion matrix shown in Figure 11 indicated that the model had 289 False Positives and 579 False Negatives, providing a strong but not perfect classification for both classes. The high performance is evident in the precision, recall, and F1-scores, which are 0.97 for class 0 and 0.99 for class 1, demonstrating the model's effectiveness in distinguishing between the two classes.

The results show a key difference in the importance of detecting each class. Misclassifying a safe URL as not legitimate (False Positive) is less of a concern, but misclassifying a phishing URL as safe (False Negative) poses a serious risk, potentially leaving users exposed to harmful attacks. The model’s high precision of 0.99 for class 1 highlights its ability to minimize these critical errors. Based on the above four measures, the number of correct predictions (TP + TN) made by this model is 46291 and the number of incorrect predictions (FP + FN) is 868.

Table 6: Classification Report

Metric	Formula	Math	Scores
Accuracy	(TP + TN) / (TP + TN + FP + FN)	46291/(46291+868)	0.98159
Precision	TP / (TP +FP)	26456/(26456+289)	0.98919
Recall	TP / (TP +FN)	26456/(26456+579)	0.97858
F-1 Score	2 * {(precision * recall) / (precision + recall)}	2* {(0.97858*0.98919)/(0.97858+0.98919)}	0.9839

The results from the training data are as shown in Table 6. The high accuracy, precision, and recall make this model very efficient from a business perspective. In addition, both false negatives and positives were overwhelmingly fewer than the true negatives and positives, which makes this model a great choice for potential customers in terms of reliability.

Study results

This study successfully answers our business question " “How can businesses and organizations identify if a URL they have clicked on is part of a phishing attack?” Using three machine learning algorithms: Naive Bayes, Neural Network, and Support Vector Machine (SVM).

While we have built several models to answer our question, it can be difficult for any ordinary individual to get access to a model like this. Instead, we can teach individuals to look out for phishing tactics using the four features we have explored in our model. Listed below are the four features that we can use to teach individuals to look out for.

1. Number of Self-Redirects: When users are redirected back to themselves many times on their own page, it can be used to mask the intention of the site from an initial visit. This deception usually works as a way to hide a website’s true intentions from someone previewing it before clicking, and multiple self-redirects works as a clear warning sign.
2. Largest Line of Code: Individuals can "inspect" the URL to make sure it is not too long, which may be a sign of phishing. Large lines of code are usually a sign of unoptimized programming, a result of poor attempts at replicating a legitimate site. If suspecting that someone clicked on a phishing site, sloppy code like this can be a good way to make sure if further action should be taken.
3. Unusual Use of Images: Websites with too many or too few images compared to normal expectations should raise suspicion. These images might be used to hide links or download buttons from the user, and bombard them with viruses without their knowledge. It’s a good, clearly visible red flag, and noticing it can be an early prompt to leave an otherwise dangerous website.
4. Number of External References: Phishing URLs are more likely to have an external page that the user is directed to outside the website that is initially clicked on. If an individual notices an unusual number of links which are not related to the website, it’s likely an attempt to send the user to another, more dangerous site, and clicking out can mitigate the damage done.

These four features can be used as signs for individuals to look for once they have clicked on a website URL. Companies can conduct training to make their employees aware of the major signs of phishing; if they notice any of these signs, they know to click out of the site immediately. Alongside our model, this gives an easier way to present the information and can present countless incidents on the individual level.

Now that we have identified both the most common phishing tactics and built a model to identify these tactics, we must explore some methods of deployment. One method would be to create a web extension for employees to check the website they’re about to click on. If a certain threshold of phishing signs are hit, it can warn, or even forbid entry altogether on company hardware. Beyond simple damage mitigation, this allows for harm to be prevented from occurring in the first place. Over time, the accuracy of the model may decrease due to new phishing tactics. To ensure our model stays up to date, we'll need to update it regularly with fresh data, retrain it to include examples of newer threats, and adapt it to changes in how phishing attacks are carried out. If the model's performance is no longer as expected, we might even have to reevaluate the methods and features to improve it. As the evolution of technology is ultimately difficult to predict, these measures are largely something that will have to be worked on in the moment, but keeping up to date information makes reacting to new phishing techniques in a swift, efficient manner possible.

Conclusion and Next Steps

We have successfully implemented the steps of the CRISP-DM process through adapting the project as needed. As we learned throughout this course, CRISP-DM is not a strictly linear process; it requires us to have flexibility to certain approaches to improve the outcome of our study. Initially we considered a different business question, though we quickly learned that this question does not align with the features we had previously based our model on during the evaluation stage. This understanding had us revisit our business question and understanding section to improve our goals. If we were to conduct a similar study in the future, we might explore different features and test how that can affect our model in the future.

References

Abbasi, Ahmed, et al. "Detecting Fake Websites: The Contribution of Statistical Learning Theory." MIS Quarterly, vol. 34, no. 3, 2010, pp. 435–61. JSTOR, https://doi.org/10.2307/25750686. Accessed 9 Dec. 2024.

George Mason University. (2023, January 26). *Phishing - Information Technology services*. Information Technology Services. https://its.gmu.edu/working-with-its/it-security-office/phishing/

Phishing. (2022, September 13). Federal Trade Commission. https://www.ftc.gov/business-guidance/small-businesses/cybersecurity/phishing

Prasad, A. & Chandra, S. (2024). *PhiUSIIL Phishing URL (Website)* [Dataset]. UCI Machine Learning Repository. https://doi.org/10.1016/j.cose.2023.103545.

Protect yourself from tech support scams. (n.d.). Microsoft Support. https://support.microsoft.com/en-us/windows/protect-yourself-from-tech-support-scams-2ebf91bd-f94c-2a8a-e541-f5c800d18435

1. Supervised learning — scikit-learn 0.22 documentation. (2019). Scikit-Learn.org. https://scikit-learn.org/stable/supervised_learning.html

What is phishing? (n.d.). https://support.microsoft.com/en-us/windows/protect-yourself-from-phishing-0c7ea947-ba98-3bd9-7184-430e1f860a44#:~:text=Phishing%20(pronounced%3A%20fishing)%20is,that%20pretend%20to%20be%20legitimate

Wright, Ryan T., et al. "Research Note: Influence Techniques in Phishing Attacks: An Examination of Vulnerability and Resistance." Information Systems Research, vol. 25, no. 2, 2014, pp. 385–400. JSTOR, http://www.jstor.org/stable/24700179. Accessed 14 Nov. 2024.

Appendix 1: Data Definitions of All Data Columns

Here are some tables for ALL the data columns explaining their definitions, data types, and ranges.

Integer Data Types and Their Ranges

Integer Data Types	
Column Name	**Range**
DomainTitleMatchScore	(0,100)
Domianlength	(4,110)
LargestLineLength	(22, 13975732)
LineOfCode	(2, 442666)
NoOfAmpersandinURL	(0,149)
NoOfCSS	(0,35820)
NoOfDegitsinURL	(0,2011)
NoOfEmptyRef	0,4887)
NoOfEqualsinURL	(0,176)
NoOfExternalRef	(0,27516)
NoOfiFrame	(0,1602)
NoOfImage	(0,8956)
NoOfJS	0,6957)
NoOfLettersinURL	(0,5191)
NoOfMarkinURL	(0,4)
NoOfObfuscatedChar	(0,447)
NoOfOtherSpecialCharsinURL	(0,499)
NoOfPopup	(0,602)
NoOfSelfRef	(0,27397)
NoOfSubDomain	(0,10)
TLDLength	(2,13)
URLLength	(13, 6097)
URLTitleMatchScore	(0,100)

Integer Data Types and their Definitions

Integer Data Types with Definition	
Column Name	**Definition**
DomainTitleMatchScore	A percentage of how similar the domain is to the title page of the website.
Domianlength	The length of the domain based off the amount of characters.
LargestLineLength	The largest line of code associated with the link
LineOfCode	The largest line of code associated with the link
NoOfAmpersandinURL	This is a count of the special character “ampersand” that is found in the link.

NoOfCSS	The amount of lines of CSS files found in the website when inspected.
NoOfDegitsinURL	This is a count of the digits that are in the URL.
NoOfEmptyRef	A count of empty references, which is a type of redirect that leads the user to nowhere when the link is clicked on.
NoOfEqualsinURL	This is a count of the amount of the special character of the equal's sign.
NoOfExternalRef	This is the count of external references, which is an external page outside of the website that the user is lead to when said website is clicked on.
NoOfiFrame	A count of the number of inline frames on the website. Inline frames are files that are embedded onto the website, such as images and videos.
NoOfImage	Number of images found on the website
NoOfJS	The amount of JS (Java Script) files that are used on the website.
NoOfLettersinURL	A count of the number of letters in the URL.
NoOfMarkinURL	This is the count of the amount of marks, or fragments/hashes found in the URL.
NoOfObfuscatedChar	This is the count of obsufucated characters in the URL. Obsufucated characters are characters that are made to be difficult to intepret from the naked eye.
NoOfOtherSpecialCharsinURL	This is a count of the number of special character in the URL. Some examples of special characters include / % *
NoOfPopup	This is count the amount of popups that appear when the URL is clicked on.
NoOfSelfRef	This is a count of the number of times the URL redirects back to itself on it’s own page.
NoOfSubDomain	This is a count of the subdomains that are with the original domain. A subdomain is an extension of the original domain.
TLDLength	This is the count of the amount of characters in the TLD column.
URLLength	This is the count of the amount of characters in the length of the URL.
URLTitleMatchScore	This is a percentage of how similar the URL is to the title page of the website.

Boolean Data Types and Their Ranges.

Boolean Data Types	
Column Name	**Range**
Bank	(0,1)
CharContinuationRate	(0,1)
Crypto	(0,1)
HasCopyrightInfo	(0,1)
HasDescription	(0,1)

HasExternalFromSubmit	(0,1)
HasFavicon	(0,1)
HasHiddenFields	(0,1)
HasObfuscation	(0,1)
HasPasswordField	(0,1)
HasSocialNet	(0,1)
HasSubmitButton	(0,1)
HasTitle	(0,1)
IsDomainIP	(0,1)
IsHTTPS	(0,1)
IsResponsive	(0,1)
label	(0,1)
NoOfSelfRedirect	(0,1)
NoOfURLRedirect	(0,1)
Pay	(0,1)
Roboys	(0,1)

Boolean Data Types and Their Definitions

Boolean Data with Definition	
Column Name	**Definition**
Bank	This is a variable of whether the user had to pay from the bank from the phishing website.
CharContinuationRate	This is the measure of whether the URL has a high amount of Character repeats (or a high character continuation rate).
Crypto	This is a variable of whether the user had to use crypto through the website or not.
HasCopyrightInfo	This is an indication of whether the website has copyright info or not
HasDescription	This is an indication of whether the website has a description or not.
HasExternalFromSubmit	This is an indication of whether the code files associated with the website has an external
HasFavicon	An indication of whether the URL has an icon next to it in the top of the website.
HasHiddenFields	An indication of whether the website has hidden fields or not.
HasObfuscation	An indication of whether the URL has obsfucation or not. Obsfuscation is the act of “encrypting” the words or characters in the URL in order to secure said URL.
HasPasswordField	An indication of whether the website has a password field or not.
HasSocialNet	This is an indication of whether the website has a social network or not.
HasSubmitButton	This is an indication of whether the website has a submit button or not.

HasTitle	This is an indication of whether the website has a title page or not.
IsDomainIP	This is an indication of whether the domain has an ip address associated with it or not.
IsHTTPS	This is an indication of whether there is an https certificate put onto the URL or not.
IsResponsive	An indication of whether the URL is responsive or not.
Label	This is our target variable, and it shows whether or not the website is phishing.
NoOfSelfRedirect	An indication on whether the website has a self direct or not.
NoOfURLRedirect	An indication of whether there are URL redirects or not.
Pay	This is an indication on whether the user must pay the website or not.
Robots	An indication on whether there is a robot attached to the URL or not.

Float Data Types and Their Ranges

Float Data Types	
Column Name	**Range**
DegitRatioInURL	(0,0.684)
LetterRationalURL	(0,0.9260)
TLDLegitimateProb	(0,0.522907)
URLCharProb	(0.001083,0.090824)
URLSimilarityIndex	(0.1556, 100)

Float Data Types and Their Definitions

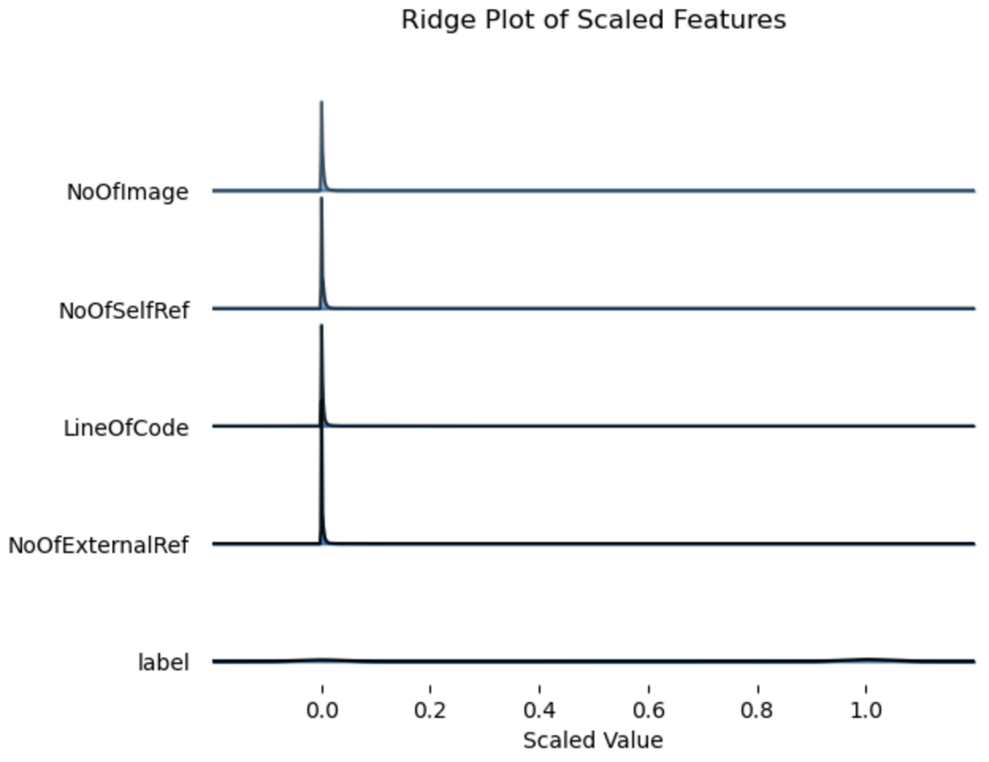
Float Data with Definition	
Column Name	**Definition**
DegitRatioInURL	The ratio of the count of digits to the total count of characters in the URL.
LetterRationalURL	The ratio of the count of letters to the total count of characters in the URL.
TLDLegitimateProb	The probability that the TLD is legitimate.
URLCharProb	The probability that the amount of characters in the URL are legitimate.
URLSimilarityIndex	How similar the phishing URL is to a legitimate URL.

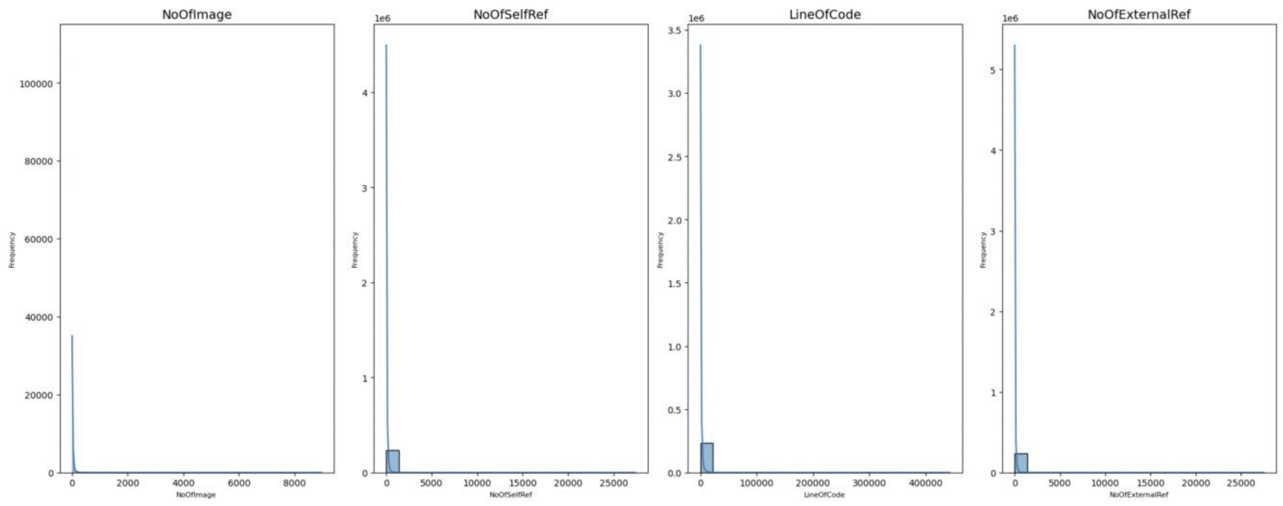
Character Data Types and Their Definitions

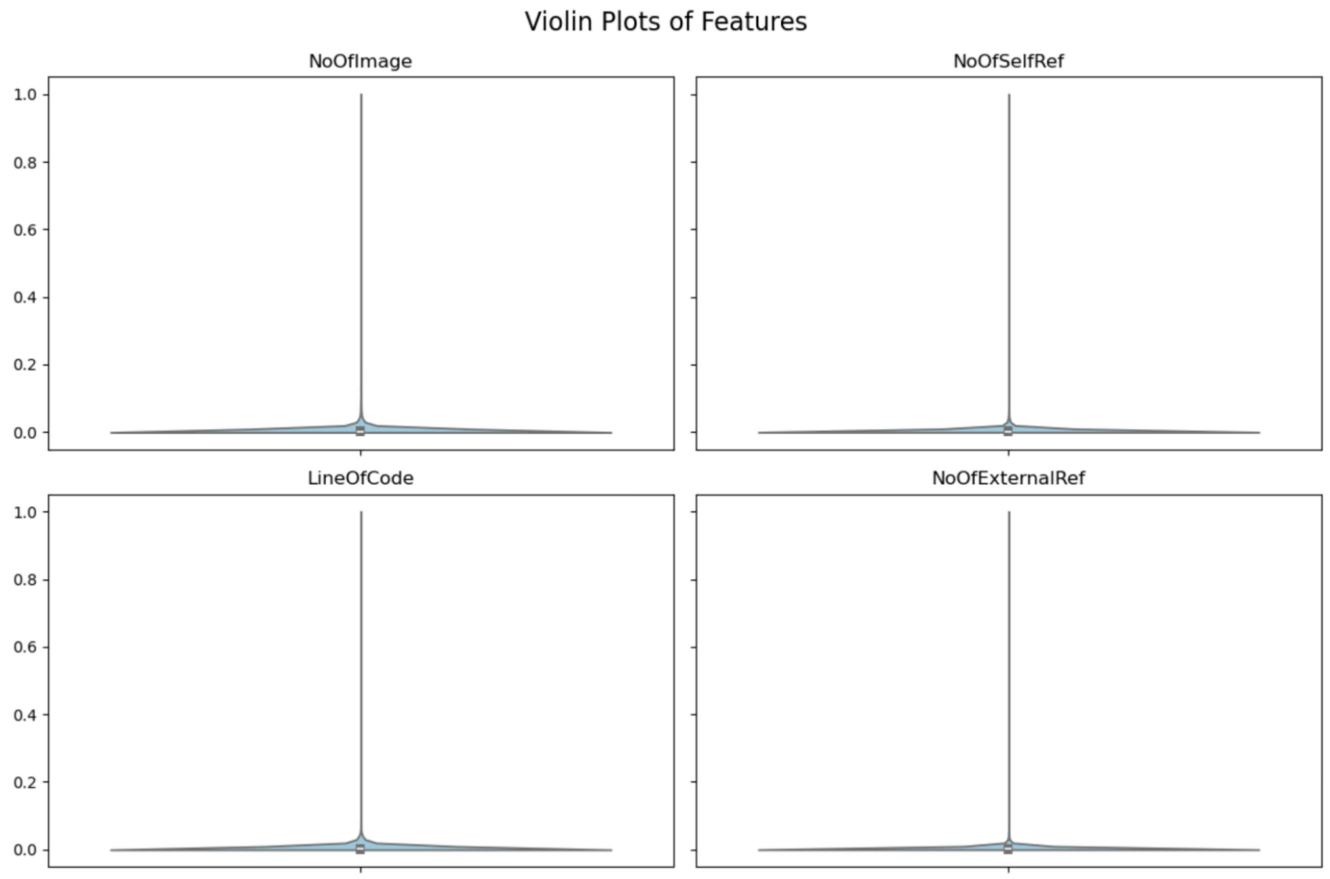
Character Data Definitions

Column Name	Definition
Domain	The domain of the website.
FILENAME	A text file of the link address.
Title	The title of the website.
TLD	The ending of the link address.
URL	The link with the HTTPS certificate.

Appendix 2: Visuals with Outliers






Violin Plots of Features
NoOfImage
NoOfSelfRef
LineOfCode
NoOfExternalRef
1.0
0.8
0.6
0.4
0.2
0.0

Appendix 3: Hyperparameters Definitions

Algorithm	Parameters Uses
Support Vector Machine	C: A regularization parameter C allows SVM to handle overfitting better in high-dimensional spaces kernel: Specifies the kernel type for mapping data into higher dimensions. Gamma: Determines the influence of training examples. Avoids overfitting by having too high of an influence. Probability: Enables predictive probabilities in the model.
Neural Network	hidden_layer_sizes: Number of neurons in hidden layer activation: Activation functions alpha: Regularization strength, the higher the alpha value, the less likely you will overfit
Naive Bayes	var_smoothing: Computes probability with the gaussian distribution.

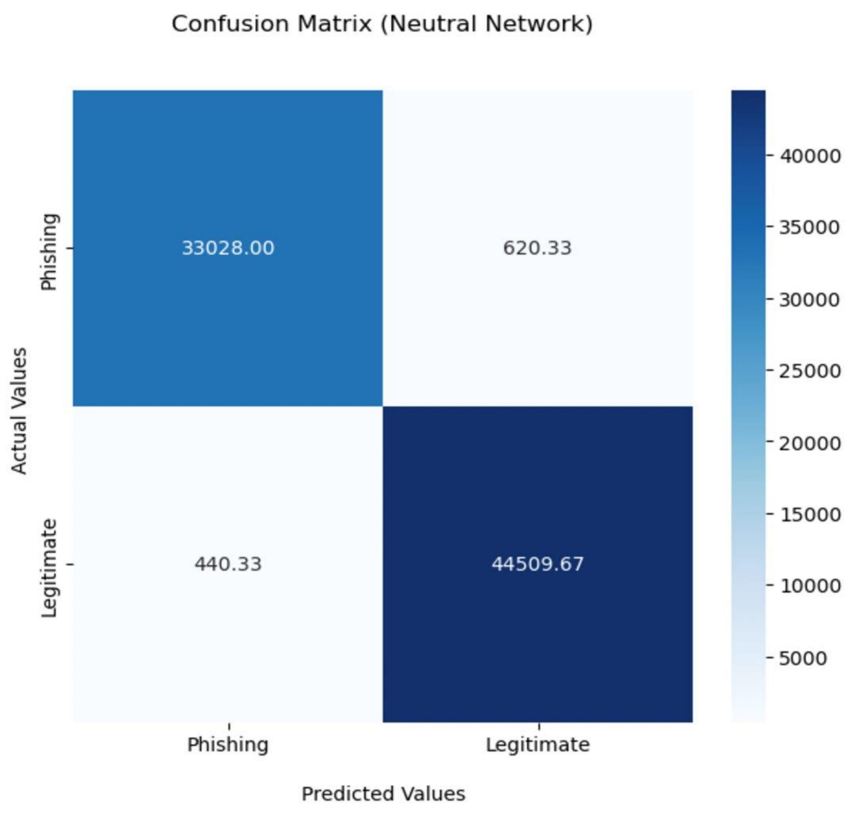
Appendix 4: Neural Network Model

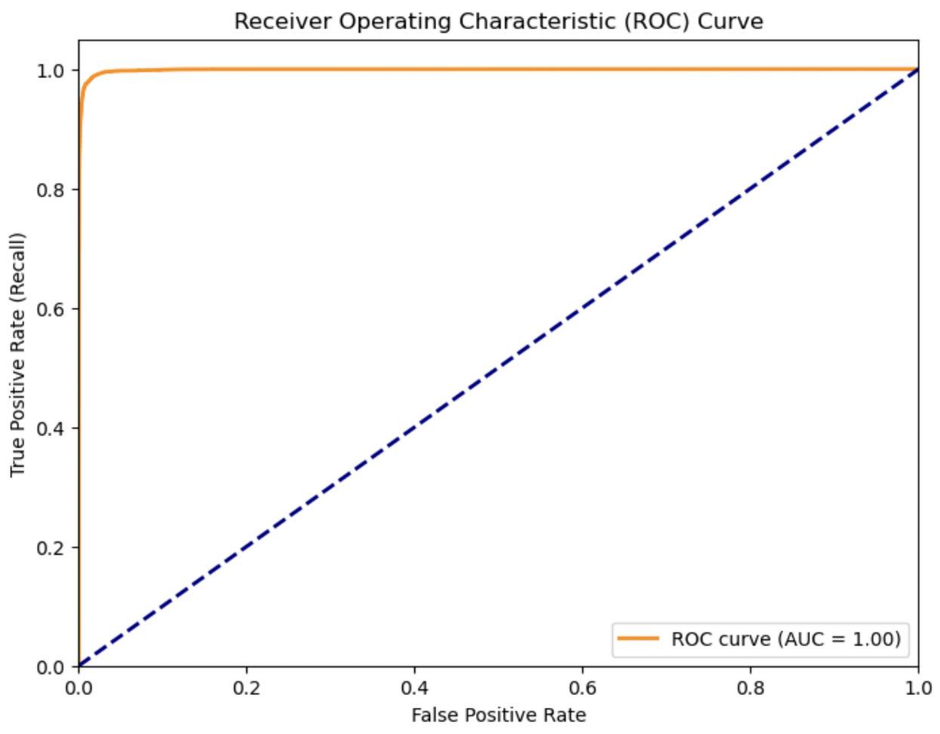
A neural network is a system that happens to mimic the human neural system, but for datasets. As mentioned in the model building section, it does scale well to our dataset because of how flexible it is.

We used the function grid search cross validation and a hyperparameter grid to find the best hyperparameters to use for our dataset. The confusion matrixes shown are all the mean values that indicate whether the neural network predicted a phishing link correctly or not.

Average Classification Report of Phishing Class		
	Testing Data	Training Data
Precision	0.9868	0.9872
Recall	0.9816	0.9816
F-1 Score	0.9842	0.9844

Mean Accuracy	0.9865	0.9867

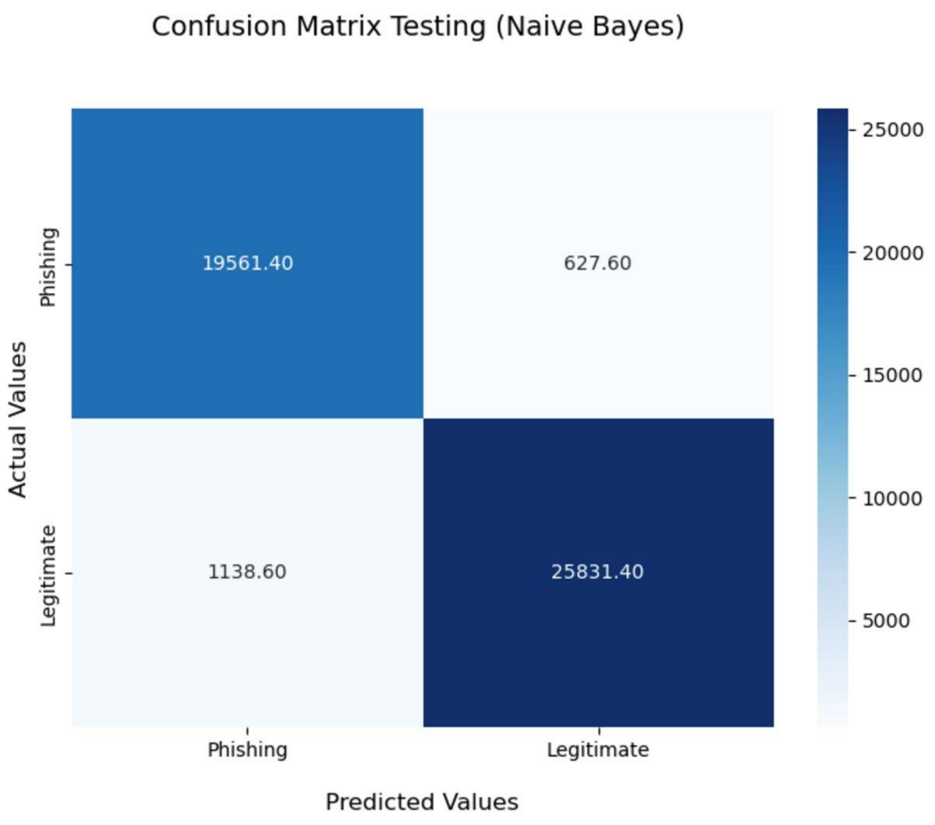




Appendix 5: Naive Bayes Classifier

The Naive Bayes algorithm happens to predict data based on probability. This algorithm happens to use joint probabilities to predict data. This model was decently compatible with our dataset as the algorithm is usually taken in two classes and checks their joint probabilities. Our target variable label has two classes, 0 and 1, indicating safe and phishing (for the website's condition).

Average Classification Report of Phishing Class		
	Testing Data	Training Data
Precision	0.9450	0.9450
Recall	0.9689	0.9689
F-1 Score	0.9568	0.9568
Mean Accuray	0.9625	0.9626



Confusion Matrix of Testing Data

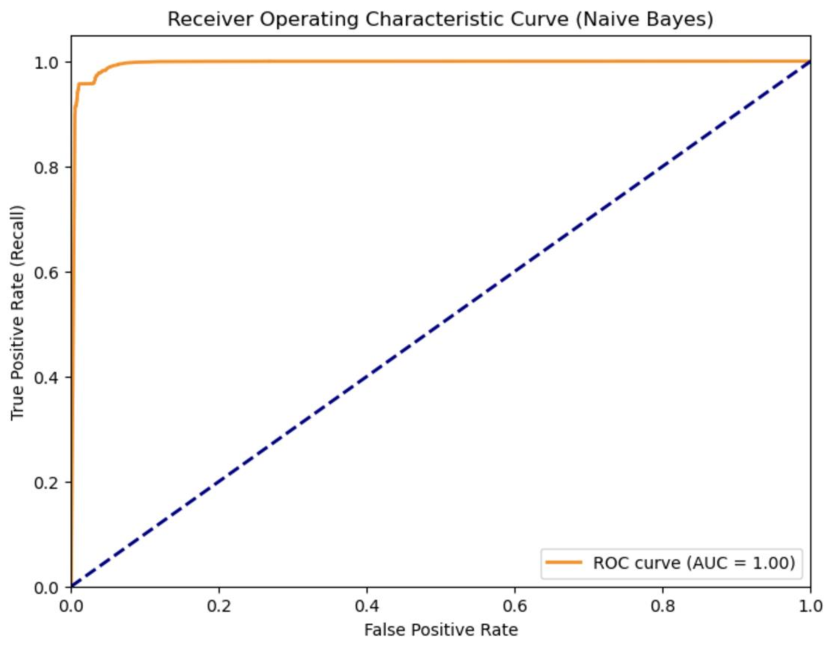


Figure 11: Confusion Matrix of Testing Data