Haiman Wong and James White

Professor Ding

CSC-481/681 Machine Learning for Cybersecurity

21 April 2021

Phishing URL Classification Through Machine Learning

Abstract

Though phishing remains one of the most prevalent cybersecurity threats, the remote landscape brought on by the COVID-19 pandemic exacerbated existing threats and provided an unprecedented opportunity for threat actors to develop and launch thousands of new, sophisticated attacks against unsuspecting users. In this paper, we detail our experiment methodology that applies machine learning techniques to determine whether a URL is malicious or benign. Our work aims to build upon previous research conducted in phishing detection through machine learning and finetune machine learning applications to cybersecurity. To determine whether a URL is malicious or benign, we employ Decision Trees, Support Vector Machines (SVM), Random Forest, and the Logistic Regression model in conjunction with performing 5-fold cross-validation, tuning the hyperparameters manually, and tuning the hyperparameters automatically with the GridSearch function. We also visualize and map the probability a malicious URL occurs within our dataset by plotting the "label" feature within our code to gain a more comprehensive understanding of the probability that a URL is classified as malicious. Finally, we analyze our results and discuss opportunities for future work and research.

1. Introduction

Across all sectors, phishing consistently remains one of the top cybersecurity threats. Phishing refers to the process where malicious actors entice unsuspecting users to visit fraudulent websites and enter potentially sensitive personal information (Sankhwar et. al, 2019). While phishing has always remained a persistent cybersecurity threat, the onslaught of remote working and learning brought on by the COVID-19 pandemic in 2020 exacerbated existing threats and provided an unprecedented opportunity for cybercriminals to launch thousands of new phishing attacks against unsuspecting users. This trend is expected to continue well into 2021 and beyond and is observed by countless cybersecurity researchers, reporters, and companies such as Microsoft, Google, Kaspersky, and Krebs on Security.

The constant development of more sophisticated phishing tools and techniques, such as phishing kits, convincing fraudulent website development, and deliberate reconnaissance work, further complicate attempts to detect and mitigate phishing attacks (McClurg, 2021). These evolving phishing threats to the cybersecurity landscape are exacerbated by our increased reliance on automation and AI techniques. For example, in late January 2021, a novel phishing kit, named LogoKit, was discovered and found to eliminate roadblocks for threat actors by automatically pulling the company logos of targeted victims onto phishing login pages (O'Donnell, 2021). The services that were mimicked by LogoKit included SharePoint, OneDrive, Office 365, Adobe Document Cloud, and several cryptocurrency exchange portals (O'Donnell, 2021). Cybercriminals successfully employed LogoKit to launch phishing attacks on more than 700 unique domains over the course of 30 days, with 300 known phishing attacks launched within the last week of the noted 30 days (O'Donnell, 2021).

Similarly, in February 2021, a new targeted phishing campaign was discovered to include the novel obfuscation technique of using Morse code to hide malicious URLs in an email attachment (Abrams, 2021). This novel phishing campaign proved to be particularly significant because of its unique ability to enable the threat actor to insert logos from the targeted recipient's companies to make the phishing attempt all the more convincing and effectively target victims. Both of these examples demonstrate how quickly threat actors are developing and launching novel phishing techniques that are increasingly more sophisticated and difficult to detect.
With phishing attempts reportedly also increasing by 42% overall between 2019 and 2020, and daily phishing threats topping 25,000 in a given day in 2020, we must develop better practices for successfully detecting and redirecting phishing attacks (O'Reilly, 2021).

In this paper, we present a detailed description of the machine learning techniques that we employ to address the persistent problem of successfully classifying whether a URL is malicious or benign. Section 2 of this paper provides an overview of related work that our project draws inspiration from, while Section 3 describes the problem we aim to solve, the algorithms and models that we used, the dataset that we chose, and the available features. In Section 4, we explain the details behind our machine learning techniques and methodology and analyze the results that we obtained. Finally, we conclude our project in Section 5 by discussing opportunities and recommendations for future work and research.

2. Related Work

Machine learning techniques have often been used to detect and classify phishing and spam attacks in previous related works. At a high level, machine learning is classified as a branch of artificial intelligence where data mining is employed to discover new or existing patterns from a dataset (Akinyelu & Adewumi, 2014). These patterns are then used for classification purposes and can aid in achieving goals of prediction, image processing, learning association, regression, and even medical diagnoses, among other tasks (Kaur, 2019).

One of the most common methods employed for detecting phishing is Bag-of-Words (BOW), which works by analyzing the frequency of which words or a group of words occurs (Gutierrez et. al, 2020). Despite the widespread adoption of the BOW method and its applauded efficiency, many researchers have also been quick to note its limitations in successfully recognizing semantics and word order (Le & Mikolov, 2014; Gutierrez et. al, 2020; Islam & Abawajy, 2012). To address the challenges presented by the BOW method, alternative algorithms and approaches, such as the Paragraph Vector, E-Mail Phishing URL detection (EMUD) model, and multi-tier classification, have been devised and tested in previous studies.

The Paragraph Vector model is an unsupervised framework that trains the vector representation to be useful for accurately predicting words in a given paragraph and builds upon the work of previous researchers who made discoveries in distributed representations of words and phrases (Le & Mikolov, 2014). More simply put, the Paragraph Vector algorithm aimed to address the BOW method's inability to recognize semantics and word order and provide an improved alternative. Results from the Paragraph Vector study demonstrated that the model can successfully capture the semantics of paragraphs, produce significant improvements in algorithmic error rates, and remain competitive with state-of-the-art methods, such as BOW or SVM. (Le & Mikolov, 2014).

In contrast, the E-Mail Phishing URL detection (EMUD) model employs machine learning techniques, such as Support Vector Machine (SVM) and the Naive Bayes (NB), as classifiers to differentiate between legitimate and phishing URLs and websites (Sankhwar et. al, 2019). The EMUD model also selects 14 heuristics to detect whether a URL is malicious or genuine (Sankhwar et. al, 2019). Results from the EMUD study indicated high levels of accurate classification rates from its confusion matrices, with the EMUD model achieving 93.01% accuracy against the EPCMU standard model's 84% accuracy in the same experiment (Sankhwar et. al, 2019).

Another approach, known as the multi-tier classification model, proposes a novel integrated method that begins with creating a set of 21 hand-crafted features from each email message (Islam & Abawajy, 2012). The multi-tier classification model then sequentially classifies email messages by using the first two-tier machine learning algorithms and sending the outputs to the analyzer, which analyzes the outputs and directs them to the appropriate mailbox based on the labeling by the machine learning algorithm (Islam & Abawajy, 2012). When the proposed multi-tier classification model was implemented, the experimental results showed that the multi-tier classification model boasted higher accuracy rates with a 97% accuracy rate in phishing email detection and consistently outperformed the methods that employed only one-tier or one test at a time (Islam & Abawajy, 2012).

Unique from the other three aforementioned studies, a fourth study aimed to enhance the academic cybersecurity community's understanding of the phishing features that contribute to variations of the classifiers without proposing a new machine learning model. To conduct this study, researchers at Texas Tech University crafted a set of phishing and legitimate emails with similar phishing indicators to investigate which indicators are detectable (Gutierrez et. al, 2020). The researchers then fed the detectable indicators to machine learning classifiers with the crafted emails to determine the email embedding performance (Gutierrez et. al, 2020). Machine learning techniques, such as SVM, logistic regression, Naive Bayes, and random forest, were employed as classifiers for this study. At the conclusion of the study, researchers found that the SVM classifier reported the highest classification performance with an accuracy and F1 score of 81.6% and 76.6% in the original feature space (Gutierrez et. al, 2020). This research ultimately paved the way for future work on using evidence theory and fusion in formulating the problem where a set of linguistic features and evidence can be used to decide pignistic probability of whether an email is phishing (Gutierrez et. al, 2020).

While each of the related studies detailed above boasts high accuracy rates, aims to propose new models that remain competitive with conventionally accepted models, and provides an improved alternative to detect phishing, each study clearly demonstrates how the same baseline machine learning techniques can be leveraged in different ways to achieve varying research aims. Our approach builds upon all of the aforementioned work because we employ the Support Vector Machine (SVM) model, as similarly done in the EMUD model, to provide a solution to better classify whether a given domain is malicious or benign. We also employ Decision Trees and the GridSearch function in conjunction with our Logistic Regression models to find the highest accuracy rates and the most effective model. In this way, our approach mirrors the strategy of the multi-tier classification model since we aim to not only test multiple models, but also to understand the underlying causes for why the differences in results may occur. Since one of the core motivations of our work is to practice applying machine learning techniques within a cyber context, we draw inspiration from these related studies to gain a deeper understanding of the work that already exists and the gaps that may continue to persist.

3. Project Methodology and Dataset Description

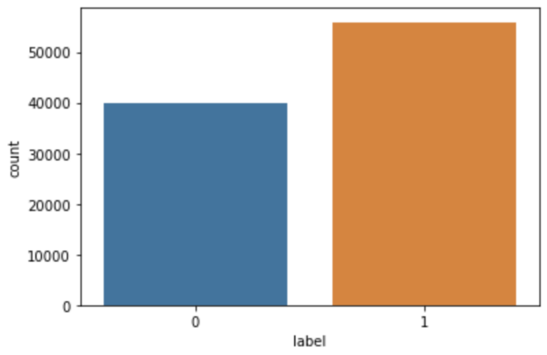
3.1 Defining the Problem, Motivation, and Methodology

Upon researching previous studies that have applied machine learning techniques to improve email and URL phishing classification, we were inspired to use the data set's large sample size and unique features to classify whether a URL is benign or malicious.

Though the aforementioned problem that we set out to solve may not be entirely unique from previous studies, our motivation for addressing this problem stems from the observation that phishing is a persistent vulnerability that continues to be exacerbated by the onslaught of remote working and learning brought on by the COVID-19 pandemic in 2020. With the prevalence of phishing attempts expected to continue to increase in number and sophistication well into 2021 and beyond, we are motivated to identify ways to finetune and apply existing machine learning techniques to the context of phishing and cybersecurity.

To address the problem that we've identified of classifying whether a URL is benign or malicious, we first visualize and map the probability a malicious domain occurs within our dataset by plotting the "label" feature in our code. This plot is shown below in Table 1 and provides a breakdown of how frequently malicious and benign URLs appear in our dataset. Since the data set describes that a "label" feature with a value of 0 assigned to a URL is a legitimate website, while a value of 1 is malicious, Table 1 demonstrates that the majority of URLs in our dataset are malicious.

TABLE 1: FREQUENCY OF MALICIOUS VS BENIGN URLS



We then employ the Support Vector Machines (SVM), the Logistic Regression Model, Random Forest, and Decision Trees to find the best models that detect phishing domains by comparing metrics such as the highest accuracy rates and lowest false-positive rates. To add further depth to our experiment, we also apply the GridSearch function to compare how our manually tuned hyperparameters compare to our automatically tuned hyperparameters. Finally, we perform 5-fold cross-validation techniques to gain a more comprehensive understanding of the model's ability to predict the skill of a machine learning model on unseen data. Section 3.2 and Section 4 of this paper further describes the dataset in detail, highlights the results that we obtained, and analyzes what factors may have contributed to better results in one model's variation over another.

3.2 Dataset Description

For the implementation and testing of our machine learning techniques, we used one publicly available dataset provided by Kumar Kanal, a contributor to the Kaggle platform. For added context, Kaggle is a subsidiary of Google LLC and is recognized as an online community of machine learning practitioners and data scientists.

The dataset we used, dubbed "PhishingAndLegitimateURLs", contains a sample size of 71,677 and 12 unique features. The 12 features include domain, ranking, islp, valid, activeDuration, urlLen, is@, isredirect, haveDash, domainLen, nosOfSubdomain, and label. A brief description of each of the features taken from the host site follows for easier reference:

- Domain: the URL itself
- Ranking: page ranking
- Islp: is there an IP address in the weblink
- Valid: this data is fetched from google's whois API that tells us more about the current status of the URL's registration
- Activeduration: also from whois API - gives the duration of the time since the registration up until now
- urlLen: simply the length of the URL up until now
- is@: if the link has a '@' character, then its value = 1
- Isredirect: if the link has double dashes, there is a chance that it is a redirect
- haveDash: if there are any dashes in the domain name
- domainLen: the length of only the domain name

- nosOfSubdomain: the number of subdomains preset in the URL
- Label: if 0, then it is a legitimate website; if 1, then it is a phishing/spam link

The dataset and its feature descriptions can also be further referenced and downloaded at https://www.coindesk.com/price/bitcoin.

Since the dataset that we chose to work with for this experiment had already undergone standard text pre-processing procedures, such as removing stop words, we did not need to add any further text pre-processing techniques to our data for this study. The data set also primarily contains URL links, numbers, and binary classifications of feature details instead of lengthier text that would have made it more reasonable for us to implement more standard text pre-processing procedures to our data. For example, the data set contains the “label” feature that assigns a label value of 0 to a URL if it is a legitimate website and a label value of 1 if it is a phishing or spam link. This binary classification assignment allows for more streamlined analysis and machine learning and also means that our dataset meets the criterion of falling into the category of labeled data. Because we are working with labeled data, our machine learning techniques would also classify as supervised machine learning.

4. Results Obtained

4.1 Decision Trees Approach, Results, and Analysis

Decision trees contain many hyper-parameters such as the criterion, splitter, max_depth, min_samples_split, min_samples_leaf, min_weight_faction_leaf, max_leaf_nodes, and class_weight, among others. In this study, we tune these hyper-parameters to obtain varying accuracy score results. Once these results are obtained, we can then observe and analyze how different combinations of hyper-parameters positively or negatively impact our accuracy scores for the training and testing splits and why.

To better understand the results we detail here, it’s imperative to first understand what each hyperparameter represents and is capable of doing. In our experiment, we focus on the criterion, max_depth, and max_leaf_nodes hyperparameters. To build our decision tree classifier, we started by splitting our data into training and test sets. Specifically, we use 75% of the dataset for training and 25% of the dataset for testing. Once the data was successfully split for training and testing, we tuned the hyperparameters to create 6 decision tree model variations. All 6 of the decision tree model variations and the subsequent accuracy scores for the training and testing sets are shown in Table 2.

TABLE 2: DECISION TREE MODEL VARIATION RESULTS

Model Variation #:	**1**	**2**	**3**	**4**	**5**	**6**
Accuracy Score (Training):	0.860	0.887	0.900	0.94813	0.98369	0.95647
Accuracy Score (Testing):	0.861	0.885	0.898	0.94216	0.94566	0.94512
Criterion:	Entropy	Entropy	Entropy	Gini	Gini	Gini

Max_Depth:	5	7	15	300	10000	600
Max_Leaf_Nodes:	5	7	15	300	10000	600

As shown in Table 2, half of the decision tree model variations were tuned with the entropy criterion, while the remaining half were tuned with the gini criterion. Overall, the gini criterion yielded higher accuracy scores for both the training and testing sets. Regardless of the criterion assigned to a model variation, the results indicate that the higher the max_depth and max_leaf_nodes are set, the higher the accuracy scores are for both the training and testing sets. Of all 6 of the model variations, model variation 5 performed the best with an accuracy training score of 0.98369 and an accuracy testing score of 0.94566.

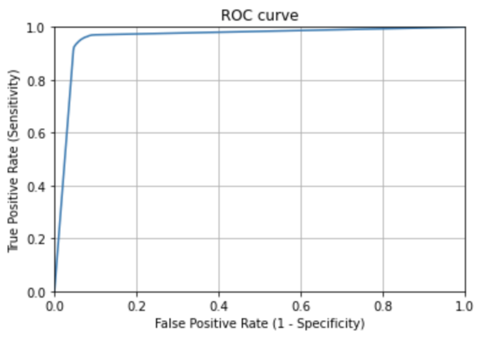
Next, we applied the GridSearch function to our decision tree classifier to identify the best hyper-parameters. The results from the decision tree model with the applied GridSearch function are shown in Table 3.

TABLE 3: DECISION TREE GRID SEARCH RESULTS

Accuracy Score:	0.931
Best Criterion:	Gini
Best Max_Depth:	9
Best Min_Samples_Split:	2
Best Min_Samples_Leaf:	1
AUC Score:	0.959284

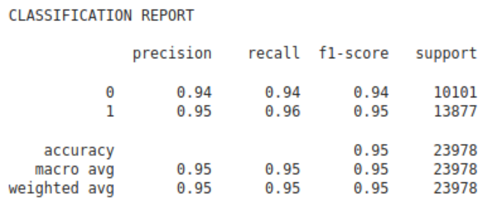
Table 3 displays the best criterion that the GridSearch function identified. Similar to our manually tuned decision tree model variation results in Table 1, the gini criterion proves to be the best criterion because it yields the highest results. However, the GridSearch function identifies a smaller value of 9 as the best tuning for the max_depth hyperparameter and uses the min_samples_split and min_samples_leaf hyperparameters. The AUC Score is also included in Table 3. With a value of 0.959284, the AUC Score for the decision tree model is quite high since it is close to the value of 1, which indicates a good model. The AUC Score stands for the area under the curve and measures the area underneath the entire ROC Curve, which is shown in Table 4.

TABLE 4: DECISION TREE ROC CURVE



The ROC Curve in Table 4 affirms our aforementioned conclusion that the decision tree model is a good model due to the low false positive rate. Table 5 goes onto highlight the confusion matrix for the decision tree model.

TABLE 5: DECISION TREE CONFUSION MATRIX



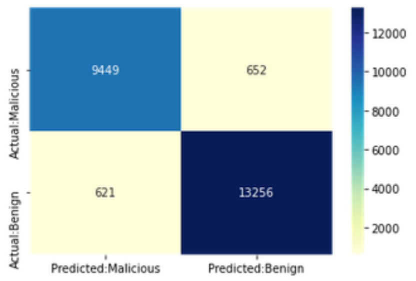
```
CLASSIFICATION REPORT

              precision    recall  f1-score   support

           0       0.94      0.94      0.94     10101
           1       0.95      0.96      0.95     13877

    accuracy                           0.95     23978
   macro avg       0.95      0.95      0.95     23978
weighted avg       0.95      0.95      0.95     23978
```

```
 DECISION TREE CONFUSION MATRIX
<AxesSubplot:>
```



The confusion matrix shown in Table 5 breaks down the information from the Classification Report to clearly see the raw true positive, true negative, false positive, and false negative numbers. The data in the confusion matrix is then used to populate the classification report. The classification report in Table 5 shows the precision, recall, and f1-score for the decision tree model as well as the confusion matrix that describes the performance of the decision tree model on test data. The decision tree model has an weighted average f1-score of 95% which is the true positive rate (recall) and precision. The f1-score can be looked at as the accuracy of the model. In Table 6, the results from the 5-Fold Cross Validation of the decision tree model are presented.

TABLE 6: DECISION TREE 5-FOLD CROSS-VALIDATION RESULTS

Cross-Validated Predictions	[1 0 1 … 1 0 1]				
Cross-Validated Accuracy Scores	0.94792	0.94828	0.94573	0.94907	0.94688
Mean Cross-Validated Accuracy Scores:	0.94757				
Cross-Validated AUC Scores:	0.96216	0.96237	0.95962	0.96323	0.95999
Mean Cross-Validated AUC Scores:	0.96147				

In sum, the decision tree model that produced the best results was decision tree model variation 5 because of its high testing and training accuracy score. This can be attributed to the high max_depth and max_leaf_nodes value assignment of 10,000. Since the value assigned to the hyperparameters max_depth and max_leaf_nodes in decision tree model variation 5 was significantly higher than the value assigned to the same hyperparameters in the decision tree GridSearch model, decision tree model variation 5 yielded the best results because higher tree depths and nodes often correlate with higher accuracy scores. Another point that is noteworthy is identifying "activeDuration", "ranking", "urlLen", "domainLen", and nosOfSubdomain" as the top five most important features for decision tree model variation 5. These five features can be confirmed as the five most important features because each feature has a score of 0.537, 0.180, 0.153, 0.088, and 0.030 respectively when we sort the values by score using the iloc function in Python. This means that "activeDuration", "ranking", "urlLen", "domainLen", and "nosOfSubdomain" are the top features the decision tree model uses to differentiate between classes and are therefore the most significant in impacting the model. Overall, the decision tree model, particularly decision tree model variation 5, prove to be high-performing models when assessing the training and testing accuracy scores, ROC Curve, AUC Score, GridSearch function, 5-Fold Cross-Validation and confusion matrix.

4.2 Random Forest Approach, Results, and Analysis

Similar to the decision tree model, we use 75% of the dataset for training and 25% of the dataset for testing in building our random forest model. The gini criterion was also used when building the random forest classifier. Once the data was successfully split for training and testing, we tuned the hyperparameters to create 4 random forest model variations. While random forests contain many hyper-parameters such as the criterion, max_depth, min_samples_split, min_samples_leaf, min_impurity_split, max_leaf_nodes, and class_weight, among others, we focus on tuning the max_depth and criterion hyperparameters in our approach. All 4 of the random forest model variations and the subsequent accuracy scores for the training and testing sets are shown in Table 7.

TABLE 7: RANDOM FOREST MODEL VARIATION RESULTS

Model Variation #1:	1	2	3	4
Accuracy Score (Training):	0.983	0.983	0.910	0.952
Accuracy Score (Testing):	0.957	0.954	0.906	0.943
Criterion:	Gini	Entropy	Gini	Entropy
Max_Depth:	Default	Default	5	12

As shown in Table 7, half of the random forest model variations were tuned with the entropy criterion, while the remaining half were tuned with the gini criterion. Overall, the gini criterion yielded higher accuracy scores for both the training and testing sets. Of all 4 of the model variations, model variation 1 performed the best with the highest training and testing accuracy scores.

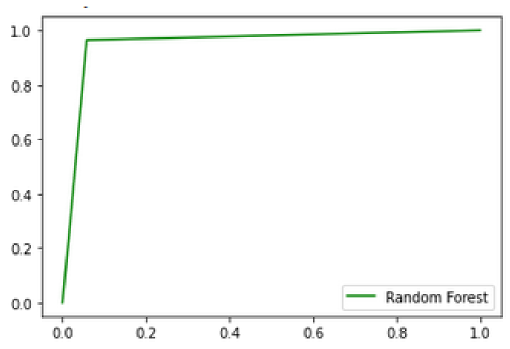
Next, we applied the GridSearch function to our random forest classifier to identify the best hyper-parameters. The results from the random forest model with the applied GridSearch function are shown in Table 8.

TABLE 8: RANDOM FOREST GRID SEARCH RESULTS

Accuracy Score:	0.958
Best Criterion:	Gini
Best Max_Depth:	9
Best Min_Samples_Split:	2
Best Min_Samples_Leaf:	1
AUC Score:	0.9885

Similar to our manually tuned random forest model variation results in Table 7, the gini criterion proves to be the best criterion. However, the GridSearch function identifies a smaller value of 9 as the best tuning for the max_depth hyperparameter and uses the min_samples_split and min_samples_leaf hyperparameters. The AUC Score is also included in Table 8. With a value of 0.9885, the AUC Score for the random forest model is quite high since it is close to the value of 1, which indicates a good model. The AUC Score stands for the area under the curve and measures the area underneath the entire ROC Curve, which is shown in Table 9.

TABLE 9: RANDOM FOREST ROC CURVE



The ROC Curve in Table 9 affirms our aforementioned conclusion that the random forest model is a good model due to the low false positive rate. Table 10 goes onto highlight the confusion matrix for the random forest model.

TABLE 10: RANDOM FOREST CONFUSION MATRIX

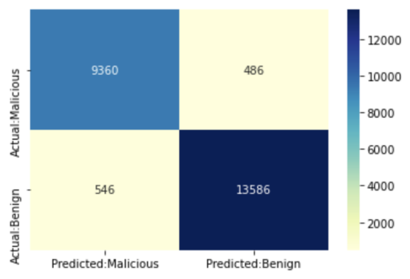
```
CLASSIFICATION REPORT

              precision    recall  f1-score   support

           0       0.94      0.95      0.95      9846
           1       0.97      0.96      0.96     14132

    accuracy                           0.96     23978
   macro avg       0.96      0.96      0.96     23978
weighted avg       0.96      0.96      0.96     23978


 RANDOM FOREST CONFUSION MATRIX
<AxesSubplot:>
```



Based on the confusion matrix and classification report shown in Table 10, the random forest model has a weighted average f1 score of 96%, which is the true positive rate (recall) and precision. The f1-score can also be viewed as the overall accuracy of the model. Table 11 goes onto present the results from the 5-Fold Cross Validation of the random forest model.

TABLE 11: RANDOM FOREST 5-FOLD CROSS VALIDATION RESULTS

Cross-Validated Predictions:	[1 0 1]...[1 0 1]				
Cross-Validated Accuracy Scores:	0.95699	0.95746	0.95587	0.95777	0.95433
Mean Cross-Validated Accuracy Scores:	0.95649				
Cross-Validated AUC Scores:	0.98922	0.98917	0.98951	0.99024	0.98833
Mean Cross-Validated AUC Scores:	0.98929				

Overall, the random forest model that produced the best results was random forest model variation 1 because of its high testing and training accuracy score. This can be attributed to the high max_depth value assignment of "default" and the criterion being set to "gini". Since the value assigned to the hyperparameters max_depth random forest model variation 1 was significantly higher than the value assigned to the same hyperparameters in the random forest GridSearch model, random forest model variation 1 yielded the best results because higher tree depths and nodes often correlate with higher accuracy scores. In this case, since the max_depth hyperparameter's default value in random forest is equal to none, the nodes are able to expand until all leaves are pure or until all leaves contain less than the min_samples_split samples assigned. As a result, random forest model variation 1 proves to be the highest-performing random forest model when assessing the training and testing accuracy scores, ROC Curve, AUC Score, GridSearch function, 5-Fold Cross-Validation and confusion matrix.

4.3 Logistic Regression Model Approach, Results, and Analysis

To build the logistic regression classifier, we started by splitting our data into training and test sets. Specifically, we continue to use 75% of the dataset for training and 25% of the dataset for testing to stay consistent with the aforementioned random forest and decision tree classifiers. Once the data was successfully split for training and testing, we tuned the solver hyperparameters to create 5 logistic regression model variations. All 5 of the logistic regression model variations and the subsequent accuracy scores for the training and testing sets are shown in Table 12.

TABLE 12: LOGISTIC REGRESSION MODEL VARIATION RESULTS

Model Variation #:	**1**	**2**	**3**	**4**	**5**
Accuracy Score (Training):	0.726	0.732	0.839	0.711	0.709
Accuracy Score (Testing):	0.725	0.732	0.837	0.710	0.710
Solver:	liblinear	lbfgs	newton-cg	sag	saga

As shown in Table 12, each of the logistic regression model variations were tuned with a different solver. Overall, the newton-cg solver in logistic regression model variation 3 yielded the highest accuracy scores for both the training and testing sets, which makes this variation the best model.

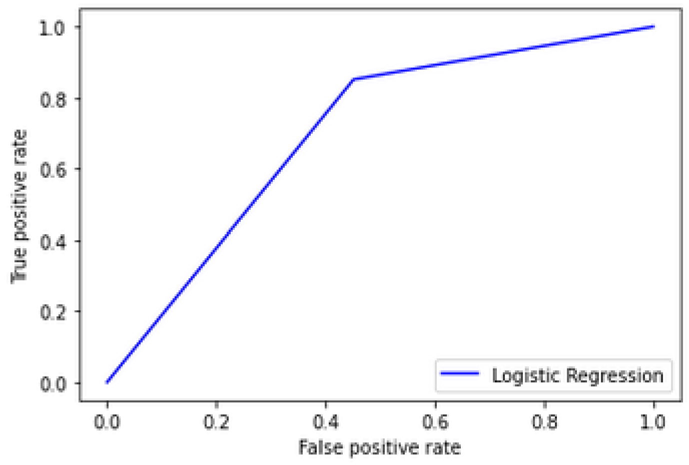
Next, we applied the GridSearch function to our logistic regression classifier to identify the best hyper-parameters. The results from the logistic regression classifier model with the applied GridSearch function are shown in Table 13.

TABLE 13: LOGISTIC REGRESSION GRID SEARCH RESULTS

Accuracy Score (Training):	0.725
Accuracy Score (Testing):	0.727
Best penalty:	l2
Best C:	0.01
AUC:	0.872

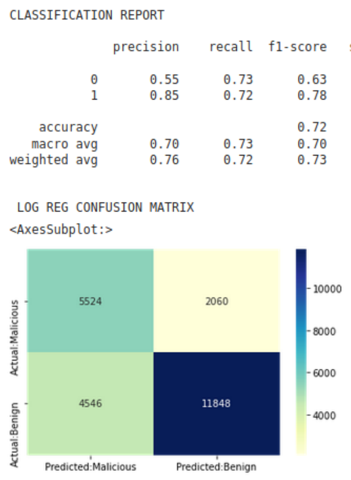
Here, the GridSearch function identifies the best values for the hyperparameters of penalty, C, and min_samples_leaf. However, the GridSearch function produces a training and testing accuracy score that fails to exceed the results produced by our manually tuned logistic regression model variation 3. This can be attributed to the lower value assigned to the max_iter hyperparameter when we ran the GridSearch function. Because the GridSearch function consistently took too long to run and changing other hyperparameters made little difference, one of the remaining options was to scale the data and lower the value assigned to max_iter. While lowering the value assigned to the max_iter hyperparameter resolved the issue of the kernel freezing, it likely also led to a lower training and testing accuracy score as a result. The AUC Score is also included in Table 13. With a value of 0.872, the AUC Score for the logistic regression model is moderately high since it is close to the value of 1, which indicates a fairly good model. The AUC Score stands for the area under the curve and measures the area underneath the entire ROC Curve, which is shown in Table 14.

TABLE 14: LOGISTIC REGRESSION ROC CURVE



The ROC Curve in Table 14 affirms our aforementioned conclusion that the logistic regression model is a fairly good model due to the relatively low false positive rate. Table 15 goes onto highlight the confusion matrix for the logistic regression model.

TABLE 15: LOGISTIC REGRESSION MODEL CONFUSION MATRIX



```
CLASSIFICATION REPORT

              precision    recall  f1-score   support

           0       0.55      0.73      0.63      7584
           1       0.85      0.72      0.78     16394

    accuracy                           0.72     23978
   macro avg       0.70      0.73      0.70     23978
weighted avg       0.76      0.72      0.73     23978

 LOG REG CONFUSION MATRIX
<AxesSubplot:>
```

The confusion matrix and classification report shown in Table 15 indicate that the logistic regression model has a weighted average f1 score of 78%, which is the true positive rate (recall) and precision. The f1-score can also be viewed as the overall accuracy of the model. Table 16 goes onto present the results from the 5-Fold Cross Validation of the logistic regression model.

TABLE 16: LOGISTIC REGRESSION 5-FOLD CROSS VALIDATION RESULTS

Cross-Validated Predictions:	[1 0 1]...[1 0 1]				
Cross-Validated Accuracy Scores:	0.29642	0.72511	0.72902	0.81034	0.72599
Mean Cross-Validated Accuracy Scores:	0.744				
Cross-Validated AUC Scores:	0.86829	0.86806	0.86644	0.90784	0.86750
Mean Cross-Validated AUC Scores:	0.876				

Overall, the logistic regression model that produced the best results was logistic regression model variation 3 because of its ability to produce the highest testing and training accuracy score. This can be attributed to the solver hyperparameter being set to the newton-cg solver. As a result, logistic regression model variation 3 proves to be the highest-performing logistic regression model when assessing the training and testing accuracy scores, ROC Curve, AUC Score, GridSearch function, 5-Fold Cross-Validation and confusion matrix.

4.4 Support Vector Machines (SVM) Approach, Results, and Analysis

To maintain consistency with the aforementioned logistic regression, random forest, and decision tree classifiers, we continue to use 75% of the dataset for training and 25% of the dataset for testing and building our SVM classifier. Once the data was successfully split for training and testing, we tuned the solver hyperparameters to create 5 SVM model variations. All

5 of the SVM model variations and the subsequent accuracy scores for the training and testing sets are shown in Table 17.

TABLE 17: SVM MODEL VARIATION RESULTS

Model Variation #:	**1**	**2**	**3**	**4**	**5**
Accuracy Score (Training):	0.791	0.557	0.583	0.593	0.767
Accuracy Score (Testing):	0.787	0.555	0.582	0.592	0.767
Kernel:	linear	poly	sigmoid	sigmoid	rbf
Max_Iter:	4000	4500	5500	6000	4000

As shown in Table 17, each of the SVM model variations was tuned with a different kernel hyperparameter, and all of the model variations except for two feature the varying max_iter hyperparameter values. Of all the SVM model variations, SVM model variation 1 proves to be the best model because it yields the highest accuracy scores for both the training and testing sets. Still, it's imperative to note that even with SVM model variation 1 proving to be the best model out of all the other SVM model variations, the SVM model is still yielding the poorest accuracy scores compared to the logistic regression, random forest, and decision tree models.

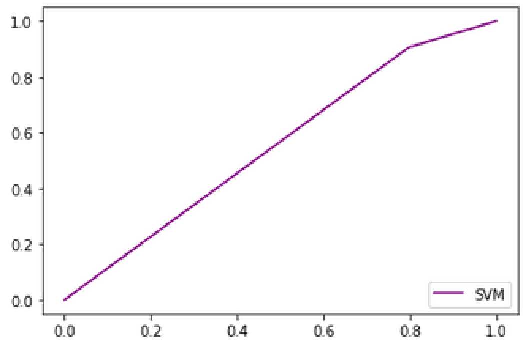
We then applied the GridSearch function to our SVM classifier to identify the best hyperparameters. The results from the SVM classifier model with the applied GridSearch function are shown in Table 18.

TABLE 18 : SVM GRID SEARCH RESULTS

Accuracy Score (Training):	0.735
Accuracy Score (Testing):	0.734
Best Kernel:	linear
Best Gamma	1e-05
Best C:	1
AUC:	0.144

Based on the results shown in Table 18, the GridSearch function identifies the best values for the hyperparameters of kernel, gamma, and C. However, the GridSearch function produces a training and testing accuracy score that fails to exceed the results produced by our manually tuned SVM model variation 1. Similar to the GridSearch function for the aforementioned logistic regression model, the lower training and testing accuracy scores can be attributed to the lower value assigned to the max_iter hyperparameter when we ran the GridSearch function in Python. Since the GridSearch function consistently struggled to run efficiently and changing other hyperparameters made little to no difference, one of the remaining options was to scale the data and lower the value assigned to max_iter. While lowering the value assigned to the max_iter hyperparameter helped the kernel to run more efficiently, it likely also led to a lower training and testing accuracy score as a result. The AUC Score is also included in Table 18. With a value of 0.144, the AUC Score for the logistic regression model is low since it is quite far from the value of 1, which would indicate a good model. The AUC Score stands for the area under the curve and measures the area underneath the entire ROC Curve, which is shown in Table 19.

TABLE 19: SVM ROC CURVE



The ROC Curve in Table 19 affirms our aforementioned conclusion that the SVM model is an average performing model due to the relatively high false positive rate. Table 20 goes onto highlight the confusion matrix for the SVM model.

TABLE 20: SVM CONFUSION MATRIX

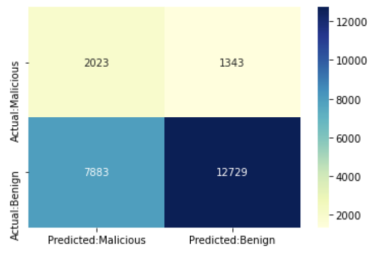
```
CLASSIFICATION REPORT

              precision    recall  f1-score   support

           0       0.20      0.60      0.30      3366
           1       0.90      0.62      0.73     20612

    accuracy                           0.62     23978
   macro avg       0.55      0.61      0.52     23978
weighted avg       0.81      0.62      0.67     23978


 SVM CONFUSION MATRIX
<AxesSubplot:>
```



Based on the confusion matrix and classification report shown in Table 20, the SVM model has a weighted average f1 score of 73%, which is the true positive rate (recall) and precision. The f1-score can also be viewed as the overall accuracy of the model. Table 21 goes onto present the results from the 5-Fold Cross Validation of the random forest model.

TABLE 21: SVM 5-FOLD CROSS VALIDATION RESULTS

Cross-Validated Predictions:	[1 0 1]...[1 1 1]				
Cross-Validated Accuracy Scores:	0.74184	0.73611	0.71385	0.74455	0.60249
Mean Cross-Validated Accuracy Scores:	0.70777				
Cross-Validated AUC Scores:	0.87044	0.87071	0.67606	0.69171	0.85908
Mean Cross-Validated AUC Scores:	0.79356				

Overall, the SVM model that produced the best results was SVM model variation 1 because of its ability to produce the highest testing and training accuracy score. This can be attributed to the kernel hyperparameter being set to linear and the max_iter hyperparameter being set to 4000. As a result, SVM model variation 1 proves to be the highest-performing SVM model when assessing the training and testing accuracy scores, ROC Curve, AUC Score, GridSearch function, 5-Fold Cross-Validation and confusion matrix.

4.5 Comparative Results and Analysis

Here, Tables 22-26 show a collection of the best models and hyperparameter tuning results side-by-side to allow for further comparison and analysis. In Table 22, it's clear that the random forest model proves to be the best because it yields the highest training and testing accuracy scores, ROC, and AUC even when compared to the decision tree model, SVM, and logistic regression.

TABLE 22: COMPARATIVE RESULTS SUMMARY

Type:	Accuracy Score (Training):	Accuracy Score (Testing):	Criterion	Max_Depth:	Max_Leaf_ Nodes	Solver	Kernel	Max_Iter
Decision Tree	0.983	0.946	Gini	10,000	10,000	---	---	---
Random Forest	0.983	0.956	Gini	Default	---	---	---	---
SVM	0.791	0.787	Linear	---	---	---	---	4000
Logistic Regression	0.839	0.837	---	---	---	newton-cg	---	---
ROC	Random Forest							
AUC	0.989 (Random Forest)							

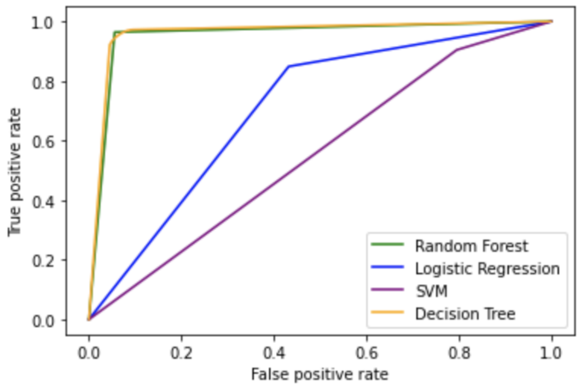
When considering which model is the best after applying the GridSearch function, Table 23 shows that the random forest model proves to be the best model again because it yields the highest testing accuracy score.

TABLE 23: COMPARATIVE GRID SEARCH RESULTS

Model:	Accuracy Score (test)	Criterion	Best penalty	Max_depth	Best Min_Samples_Split	Best Min_Samples_Leaf	C	gamma	Kernel	AUC Score
Decision Tree	0.931	gini	---	9	2	1	---	---	---	0.959
Random Forest	0.958	gini	---	9	2	1	---	---	---	0.989
Logistic Regression	0.727	---	L2	---	---	---	0.01	---	---	0.872
SVM	0.734	---	---	---	---	---	1	1e-05	linear	0.14

Here, Table 24 shows the ROC Curve plots for all four models. While the ROC Curves for the random forest model and the decision tree model appear to overlap with each other, the random forest model still boasts the best ROC curve due to its lower false positive rate.

TABLE 24: COMPARATIVE ROC CURVES



In Table 25, the true positive (TP), true negative (TN), false positive (FP), false negative (FN), and false positive rates (FP rate) for all four models are shown side-by-side. The random forest model has the lowest false-positive rate, which is an indication that the random forest model is the best.

TABLE 25: COMPARATIVE CONFUSION MATRICES

Model	Decision Tree	Random Forest	Logistic Regression	SVM
TP	13215	13370	11866	13732
TN	9473	9554	5576	1297

FP	645	564	4542	8821
FN	645	490	1994	128
FP rate	0.063	0.0557	0.449	0.703

Table 26 shows the results of the best 5-Fold Cross Validation for each of our four models. Here, it is most meaningful to compare the mean cross-validated accuracy score and the mean cross-validated AUC score. In doing so, we maintain that the random forest model is the best because it boasts the highest mean cross-validated accuracy score and the mean cross-validated AUC score.

TABLE 26: BEST 5-FOLD CROSS VALIDATION RESULTS

Model	**Decision Tree**	**Random Forest**	**Logistic Regression**	**SVM**
Cross-validated accuracy scores	0.948, 0.948 0.946, 0.949, 0.947	0.957, 0.957, 0.956, 0.958, 0.954	0.730, 0.725, 0.729, 0.810, 0.726	0.742, 0.736, 0.714, 0.745, 0.602
Mean cross-validated accuracy scores:	0.948	0.956	0.744	0.708
Cross-validated auc scores	0.962, 0.962, 0.960, 0.963, 0.960	0.989, 0.989, 0.990, 0.990, 0.988	0.868, 0.868, 0.866, 0.908, 0.868	0.870, 0.871, 0.676, 0.692, 0.859
Mean cross-validated auc scores	0.961	0.989	0.876	0.794

In this section, we detailed our methodology for running each of the four models and analyzed possible factors that influenced why we obtained the results that we did. Consistently, the random forest model outperformed the decision tree model, the logistic regression model, and the SVM model. Even when the best random forest model, decision tree model, logistic regression model, and the SVM model are compared side-by-side, the random forest model still produced the best training and testing accuracy scores and ROC Curve.

While the decision tree model came the closest to matching the random forest model's results, its inability to compete with the random forest model can be attributed to the decision tree's vulnerability to creating biased trees when certain classes dominate. Because the random forest model is, by definition and by design, a set of multiple decision trees, it does not suffer from issues, such as overfitting, that a single decision tree is more vulnerable towards.

Another trend that is clearly observable throughout all of our results, especially in Section 4.5 when we compare our best models and results side-by-side, is how the GridSearch function did not outperform our model variations with manually tuned hyperparameters. This was an unexpected result because the GridSearch function exists to find the most optimal hyperparameters of a given model that will be most successful in forming accurate predictions. The discrepancy in training and testing accuracy scores between the models with the GridSearch function applied and our manually tuned models can be attributed to the low values assigned to max_iter hyperparameter, especially for the logistic regression and SVM models. While the lowered max_iter hyperparameter values resolved the issue of the kernels freezing in both the logistic regression and the SVM model, it is reasonable to assume that this sacrificed the possibility of achieving higher accuracy scores. This also provides an opportunity for future work and research, where researchers can conduct an independent research study centered on what hyperparameters may consistently cause a kernel to freeze and what can be done to remedy the issue automatically with limited manual intervention.

When evaluating the 5-Fold Cross Validation results, the mean cross-validated accuracy score and the mean cross-validated AUC score indicate that the random forest model is still the best due to the relatively higher mean cross-validated scores. Therefore, the random forest model is the best because it consistently yields the highest training and testing accuracy scores, AUC Scores, ROC Curve, and the lowest false-positive rate based on the results from the confusion matrix, classification report, GridSearch function, and 5-Fold Cross Validation.

5. Discussion

With the trend of phishing attacks growing both in number and in sophistication expected to continue well into 2021 and beyond, we must continue to finetune our ability to apply machine learning techniques within a cyber context and develop better practices for successfully detecting and redirecting phishing attacks. Therefore, in this paper, we have presented a collection of machine learning techniques and demonstrated how tuning hyperparameters in machine learning models produce competing results. These results provide significant insight into how hyperparameters and variables impact a machine learning model's ability to classify whether a URL is malicious or benign successfully.

Opportunities and recommendations for future work and research include conducting more comprehensive research on the strengths and weaknesses of newer machine learning models and approaches could pave the way for the creation of even more sophisticated anti-phishing machine learning models. Furthermore, developing an original dataset by collecting new data or synthesizing it across several existing datasets could allow future work targeted at understanding how to detect and redirect new sophisticated phishing techniques, such as those involving phishing kits, convincing fraudulent website development, and deliberate reconnaissance work. Finally, connecting our best models to web-browser extensions that can monitor potential phishing URLs and safeguard human error. Since human error is responsible for the vast majority of cybersecurity breaches, developing a way to improve a user's accessibility to tools that help detect phishing attempts would be a practical way to protect users from cyber vulnerabilities and build upon the work in this paper.

References

Abrams, L. (2021, February 14). New phishing attack uses morse code to hide malicious URLs. Retrieved from https://www.bleepingcomputer.com/news/security/new-phishing-attack-uses-morse-code-to-hide-malicious-urls/

Andronicus A. Akinyelu, & Aderemi O. Adewumi. (2014). Classification of Phishing Email Using Random Forest Machine Learning Technique. *Journal of Applied Mathematics*, *2014*, 1–6. https://doi.org/10.1155/2014/425731

Brooks, C. (2021, March 03). Alarming cybersecurity STATS: What you need to know for 2021. Retrieved from https://www.forbes.com/sites/chuckbrooks/2021/03/02/alarming-cybersecurity-stats-------what-you-need-to-know-for-2021/?sh=6194c8c758d3

Classification: ROC curve and AUC | machine Learning crash course. (n.d.). Retrieved from https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc#:~:text=AUC%20represents%20the%20probability%20that,has%20an%20AUC%20of%201.0.&text=AUC%20is%20scale%2Dinvariant.

Gutiérrez, L., Abri, F., Armstrong, M., Namin, A., & Jones, K. (2020). *Phishing Detection through Email Embeddings*.

Islam, R., & Abawajy, J. (2013). A multi-tier phishing detection and filtering approach. *Journal of Network and Computer Applications*, *36*(1), 324–335. https://doi.org/10.1016/j.jnca.2012.05.009

Jones, C. (2021, April 08). 50 phishing stats you should know in 2021. (2021, April 08). Retrieved from https://expertinsights.com/insights/50-phishing-stats-you-should-know/

Kaur, A., NT, A., Jeevan, A., Kalafati, A., Raut, A., Nichols, A., & Guest, A. (2020, January 21). Top 10 real-life examples of machine learning. Retrieved from https://bigdata-madesimple.com/top-10-real-life-examples-of-machine-learning/#:~:text=Currently%2C%20machine%20learning%20has%20been,%2C%20learning%20association%2C%20regression%20etc.&text=Machine%20learning%20applications%20provide%20results%20on%20the%20basis%20of%20past%20experience

Le, Q., & Mikolov, T. (2014). *Distributed Representations of Sentences and Documents*.

McClurg, J. (2021, January 04). The art of targeted phishing: How not to get hooked. Retrieved from https://www.securitymagazine.com/articles/94265-the-art-of-targeted-phishing-how-not-to-get-hooked

O'Donnell, A., & O'Donnell, L. (2021, January 28). LogoKit simplifies Office 365, Sharepoint 'Login' phishing pages. Retrieved from https://threatpost.com/logokit-simplifies-office-365-sharepoint-login-phishing-pages/163430/
O'Reilly, L., (2021, February 10). The state of phishing in 2021. Retrieved from https://securityboulevard.com/2021/02/the-state-of-phishing-in-2021/

Sankhwar, S., Pandey, D., & Khan, R. (2019). Email Phishing: An Enhanced Classification Model to Detect Malicious URLs. *EAI Endorsed Transactions on Scalable Information Systems*, *6*(21), 158529–. https://doi.org/10.4108/eai.13-7-2018.158529

Sklearn.ensemble.randomforestclassifier¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Sklearn.linear_model.LogisticRegression¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Sklearn.svm.svc¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Sklearn.tree.decisiontreeclassifier¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html