Phishing Detection Using Machine Learning

Karen John
Department of Computer Science
Stevens Institute of Technology
kjohn3@stevens.edu

Joscandy Nunez
Department of Computer Science
Stevens Institute of Technology
jnunez2@stevens.edu

Shakil Mustafiz
Department of Computer Science
Stevens Institute of Technology
smustafiz@stevens.edu

***Abstract*—Phishing is an evident cybersecurity threat. Cybercriminals often pose as reputable organizations and send emails with links to phishing websites to unsuspecting individuals. Individuals who enter phishing websites expose themselves to data breaches and malware. Companies invest considerable time and money in cybersecurity training to protect their employees and customers from phishing attempts. Nevertheless, phishing is still prevalent in today's society. An October 2022 study by security provider SlashNext, found more than 255 million phishing attempts in email, mobile, and browser channels. SlashNext reports that there has been a 61 percent increase in phishing attempts since 2021. To solve this problem, we proposed to create a phishing detection system based on machine learning algorithms. The problem is essentially a classification task of whether a link is legitimate or phishing. We trained a model to detect a phishing website based on a subset of features. Three algorithms used to train the model were KNN classifier, Logistic Regression, and Decisions Trees. We intended to identify the best classifier for phishing detection. KNN was the most accurate classifier, with an accuracy of 79.3 percent.**

I. Introduction

Phishing has become a major concern in the past few years due to the ease of creating websites that look similar to legitimate websites and has become an evident cybersecurity threat. Cybercriminals often pose as reputable organizations and send emails with links to phishing websites to unsuspecting individuals. Individuals who enter phishing websites, expose themselves to data breaches and malware. Companies spend time and money investing in cybersecurity training to protect their employees and customers from phishing attempts. Nevertheless, phishing is still prevalent in today's society. In an October 2022 study, conducted by security provider SlashNext, found more than 255 million phishing attempts in email, mobile, and browser channels. SlashNext reports that there has been a 61 percent increase in phishing attempts since 2021. To solve this problem, we propose to create a phishing detection system based on machine learning algorithms. The problem is essentially a classification task of whether a link is legitimate or phishing.

The three algorithms that we used to implement a phishing detection system are: KNN Classifier, Logistic Regression, and Decision trees. KNN classifier is a supervised learning method that classifies new data based on similarity with existing classified data. Logistic regression is another supervised learning method used to classify data into two classes based on probabilities, determined by a subset of features. A decision tree is a decision-making tool that employs a model of potential outcomes based on certain conditions. These outcomes are determined by previous decisions in the training data. All three algorithms can be used in a classification task such as fraud detection. The purpose of this project is to determine the algorithm that will generate the most accurate phishing detection model.

Our work greatly differs from other projects utilizing this dataset because we use two new algorithms not previously used: KNN classifiers and decision trees. We chose to utilize logistic regression since this is commonly used for detecting spam emails. Though one existing project utilizes logistic regression, the project fails to include any relevant plots that describe the accuracy of the machine learning models. Specifically, our project calculates the area under ROC curve (AUC) to measure the diagnostic ability of the binary classifier. Therefore, our project offers a better comparison of machine learning algorithms for a classification task.

II. Related Work

Under Kaggle, there are 19 different projects utilizing this dataset. Most of these projects utilize different algorithms to train a model that classifies phishing URLs. The algorithms used include the following: neural networks, support vector machines, naive bayes, logistic regression, and random forest.In general, convoluted, recurrent and artificial neural networks have been extensively used in past projects. One project of note is Phishing Detection with Bait by Tyler Sullivan and Matthew Franglen. This project implements three algorithms: naive bayes, support vector machines, and logistic regression. The main advantage of this project is that it uniquely transforms the dataset and produces the machine learning models using vectorization and pipelines. The main disadvantage of this project is that it lacks any pertinent plots to describe accuracy of the machine learning models.

A. Description of Dataset

The first step in building the phishing detection model website is to choose an appropriate dataset that will consist of both phishing and legitimate websites, which will be used for training a model for predicting phishing URLs. The chosen dataset can be found in Kaggle, and is called "Web page Phishing Detection Dataset", by Shashwat Tiwari. The dataset consists of 11430 rows of records containing URLs, and 87 columns that make up the features. This is a good dataset based on the fact that there are a large number of records and

features to allow for a good breakdown of training and testing data. This will also help us avoid either bias or variance while training the model. Initial data analysis reveals that there are no null values in the dataset shown in Figure 1. Also, the dataset rows is evenly distributed between the amount of phishing and legitimate URLs.

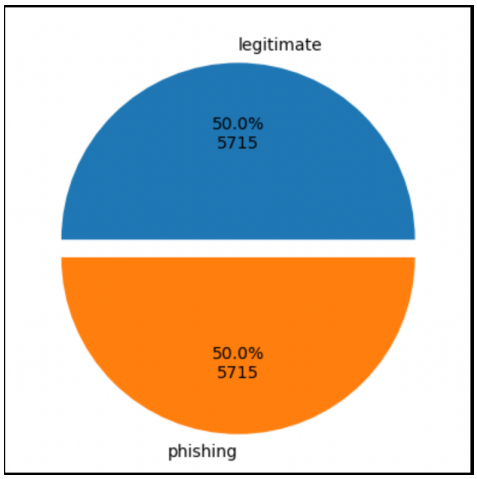

Fig. 1. Distribution of Dataset

Our machine learning models will place an emphasis on the special characters, punctuation and https token of the URLs. In order to achieve this, all of the records will be used. The features that will be extracted to train the model are the following: hyphens, dots, slashes, underscores, ratio of digits, status, and https token. Some of these features are discrete (0 or 1) and some are continuous values. Features that hold discrete values help describe whether something is present or not in the URL (i.e hyphens), while continuous values quantify a feature of the URL (i.e ratio of digits). The values ”phishing” and ”legitimate” will be encoded to 1 and 0, respectively. The dataset will be split into a training size of 75 percent, and testing size of 25 percent.

	https_token	ratio_digits_url	nb_hyphens	nb_dots	nb_underscore	nb_slash	status
0	1	0.000000	0	3	0	3	legitimate
1	1	0.220779	0	1	0	5	phishing
2	0	0.150794	1	4	2	5	phishing
3	1	0.000000	0	2	0	2	legitimate
4	1	0.000000	2	2	0	5	legitimate

Fig. 2. Snippet of features selected.

III. Our Solution

A. Machine Learning Algorithms

Algorithm 1: KNN-classifier is a popular and easy algorithm to understand in machine learning. KNN-classifier is used for both classification and regression, and this problem involves classification in supervised learning. The k in KNN classifier represents the nearest neighbors. In this algorithm, a data point uses the neighbors' classification to predict its own.

	url	length_url	length_hostname	ip	nb_dots
0	http://www.crestonwood.com/router.php	37	19	0	3
1	http://shadetreetechnology.com/V4/validation/a...	77	23	1	1
2	https://support-appleId.com.secureupdate.duila...	126	50	1	4
3	http://rgipt.ac.in	18	11	0	2
4	http://www.iracing.com/tracks/gateway-motorspo...	55	15	0	2

5 rows × 89 columns

Fig. 3. Snippet of entire dataset (5 rows of a total of 11430.

This k can be tuned, allowing for greater accuracy in the model. Something like 3 to 5 is usually a good place to start since odd values are recommended. The best of k value will be determined by comparing the trained model's accuracy values per each k. The model will use X as a group of columns/attributes extracted from an url, and a Y column/array that will be the target value [legitimate, phishing].

Algorithm 2: Logistic Regression is commonly used for classification tasks such as fraud detection. Logistic regression estimates the probability that an instance belongs to a certain binary class. In this project, the logistic regression algorithm will estimate the probability a URL is phishing or legitimate. If the probability that the URL is phishing exceeds 50 percent, the URL will be classified phishing. If this probability is below 50 percent, the URL will be classified as legitimate. In logistic regression, the weighted sum of the input features and bias term is computed and processed through a logistic function to determine the probabilities for a binary classification. The primary reason this algorithm was selected for this task is because it is a common classifier algorithm and our group wanted to determine which algorithm would have the best performance for classifier tasks such as phishing detection.

Algorithm 3: Decision Trees are frequently used in prediction models in domains such as machine learning, statistics, and data mining. A decision tree is based on a tree structure also known as a predictive model, where leaf nodes reflect projected classes and inside nodes represent decisions. The algorithm constructs a decision tree in a top-down fashion. The algorithm begins at the root node and makes the best choice recursively. Remaining instances are divided into child nodes until there are no more features to select. In this project, the decision tree will distinguish phishing urls from legitimate urls so that the receiver is not harmed by the attackers. For the implementation, the following set of features will be used to train models such as hyphens, dots, slashes, underscores, ratio of digits. For example, the slashes in benign URLs are found to be a 5; if the number of slashes in URLs is greater than 5 then the feature is set to 1 in comparison to 0.

A decision tree is a suitable algorithm for URL classification for several reasons:

- Handling categorical and continuous features: URLs have both categorical and continuous features. The categorical features include domain names, paths, and query parameters, while the continuous features include the length of URLs and the number of slashes. The decision tree

algorithm can handle both categorical and continuous features, making it a suitable choice for URL classification.

- Nonlinear relationships between features: URLs can have nonlinear relationships between features. For example, a phishing URL may have a long path and many query parameters, while a legitimate URL may have a short path and no query parameters. A decision tree algorithm can capture such nonlinear relationships and create a tree-like structure that can be easily interpreted and visualized.
- Scalability: A decision tree algorithm can handle large datasets with a high number of features, making it suitable for URL classification tasks that involve a large number of URLs and features.
- Interpretable: Decision trees produce a tree-like structure that can be easily visualized and interpreted. This can be useful for understanding how the algorithm is making predictions and identifying the important features for URL classification.

Overall, the decision tree algorithm is a good fit for URL classification due to its ability to handle both categorical and continuous features, capture nonlinear relationships between features, handle missing values, its scalability, and its interpretability.

B. Implementation Details

Upon completion of pre-processing, implementation of the three algorithms were as follows.

Algorithm 1: For the implementation of KNN classifier, the function usedis KNeighborsClassifier from the library sklearn.neighbors. This method, which creates the KNN classifier, takes in a number of neighbors ”n_neighbors” or ”best_k” in the following fashion knn = KNeighborsClassifier(n_neighbors = best_k). Lastly, to train this model using the training data, we ”fit” it to the training data, such as knn.fit(X_train.values, y_train). The number of neighbors is practically the only parameter used in training the knn classifier model, therefore, this number can be calculated to optimize the prediction model. The process for finding the best value of k is defining a set of k values to test, in this case, number 1 to 61, both numbers inclusive. A for loop was implemented, and in each iteration, a knn classifier model is created using that k, and the accuracy is calculated using python’s cross_val_score function. These iterations of k ranging from 1 to 61 and their respective accuracy for knn classifiers is illustrated in figure 4.

The best k value that yields the highest accuracy is 18, where the line in figure 4 peaks. This best k value used in the prediction step, yields an accuracy of 0.7925122463261022 for the final classifier’s prediction. In addition, the precision is 0.8117998506348021, and the recall is 0.761204481792717.

Lastly, a dynamic function was also developed to be able to pass down any url for this model to make a prediction (legitimate vs phishing). This function extracts features from the given url : extract_features(url_to_predict). Then, these extracted features are passed to our model (either one of the three), using knn as an example: knn.predict([extracted features array]). This allowed us to dynamically test a couple of url addresses, and the result is illustrated in Figure 5:

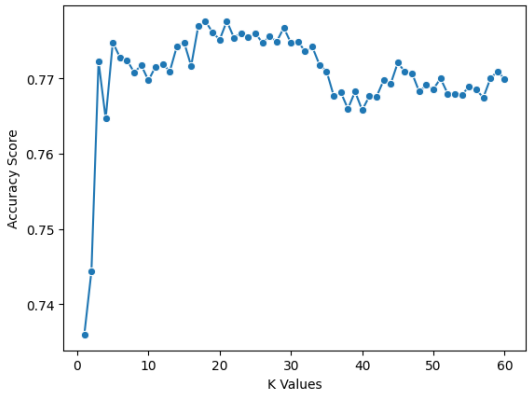

Fig. 4. Accuracy scores using different values of ’k’ neighbors.

URL	Status	Correct Prediction
https://www.coursera.org/learn/python-machine-learning/lecture/MwsUM/k-nearest-neighbors-classification	Legitimate	Yes
https://app.simplenote.com/publish/XhRdVc	Phishing	Yes
https://stackoverflow.com/questions/24878174/how-to-count-digits-letters-spaces-for-a-string-in-python	Legitimate	Yes
https://www.mjwaldman.com/	Phishing	Yes
https://www.simplilearn.com/tutorials/python-tutorial/python-if-else-statement	Legitimate	Yes
https://tubitv.com/category/peliculas_en_espanol	Legitimate	Yes

Fig. 5. Manual run of tests for different urls with KNN classifier.

This aided in our understanding of how well the performance of the model was. The predictions from the manual tests displayed in the chart are very satisfying and outperform expectations considering the accuracy, precision, and recall results.

Algorithm 2: For the implementation of logistic regression, built-in libraries were used from sklearn to train and test the dataset. Initial classification reports showed an accuracy of around 70 percent. Therefore, a scaler was used in the the program to normalize the data features. As a result, accuracy improved to 72.8 percent.

	precision	recall	f1-score	support
0	0.69	0.82	0.75	1407
1	0.79	0.64	0.71	1451
accuracy			0.73	2858
macro avg	0.74	0.73	0.73	2858
weighted avg	0.74	0.73	0.73	2858

0.7288313505948215

Fig. 6. Classification Report of Logistic Regression after Normalization

A heatmap was then produced to display the confusion matrix. In summary, 11555 true negatives and 928 true positives. There were 523 false positives and 252 false negatives.

Furthermore, a R0C was plotted to demonstrated the performance of the classifier. The AUC score of the classifier was determined to be 0.73. Generally, an AUC score over 0.50 is indicative that the classifier can determine more true positives and negatives than false positives and negatives. However, an excellent AUC should be at 0.8 or 0.9. This standard determines that the current logistic regression classifier is acceptable, but not excellent in performance.

In order to improve the performance of the classifier, a GridSearch CV was implemented to find optimal parameters for the logistic regression function. With GridSearch, the performance of the classifier could be slightly improved to 73.2 percent. The parameters that were optimized were random state, maximum number of iterations, and the solver used with a balanced weight. Best parameters for the model were determined to be a random state of 1234, 20 maximum iterations, and a newton-cg solver. Around 300 fits took place to find the optimal parameters that produced that .4 percent increase in accuracy.

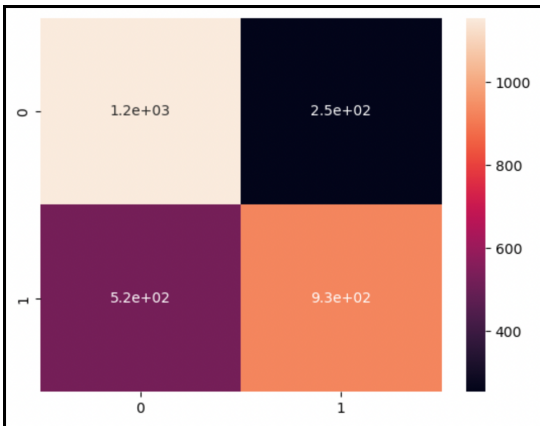

Fig. 7. Heatmap of Logistic Regression after Normalization

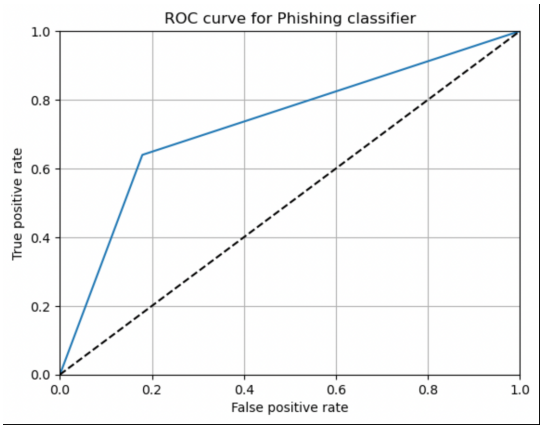

Fig. 8. ROC Curve for Logistic Regression Classifier

Algorithm 3:

To implement the Decision Tree algorithm for URL classification, the built-in libraries from sklearn were utilized for

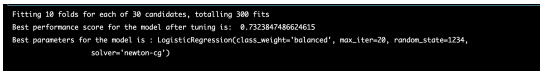

Fig. 9. Accuracy of Logistic Regression after Performance Tuning

training and testing the dataset. The initial classification reports showed an accuracy rate of approximately 76 percent, which indicates good performance of the model. However, in order to further enhance the accuracy of the model, feature selection techniques were applied to the program. Additionally, the Gridsearchcv algorithm was utilized to obtain the best possible set of parameters for the Decision Tree algorithm on the given URL dataset. By incorporating these techniques, the model was able to achieve even higher accuracy rates, thereby improving its effectiveness in accurately classifying URLs. Overall, the integration of feature selection and parameter tuning techniques has resulted in a more optimized and effective Decision Tree model for URL classification.

A classification report is a summary of the performance of a machine learning algorithm for a classification task. It provides a comprehensive evaluation of the precision, recall, F1-score, and support for each class in the target variable.

In the case of the decision tree algorithm applied to a URL classification dataset, the classification report would contain the following metrics:

- Precision: Precision is the ratio of true positives (TP) to the total number of positive predictions (TP + false positives (FP)). It measures the accuracy of the positive predictions made by the algorithm. In the context of URL classification, precision would tell us how many of the URLs predicted to belong to a particular category actually belong to that category.
- Recall: Recall is the ratio of true positives (TP) to the total number of actual positives (TP + false negatives (FN)). It measures the ability of the algorithm to identify all positive instances. In the context of URL classification, recall would tell us how many of the URLs belonging to a particular category were correctly identified by the algorithm.
- F1-score: The F1-score is the harmonic mean of precision and recall. It is a single metric that combines both precision and recall, giving equal weight to both metrics. It provides a measure of the overall performance of the algorithm in terms of its ability to classify URLs accurately.
- The classification report would provide the precision, recall, F1-score, and support for each class in the target variable, allowing us to evaluate the performance of the decision tree algorithm on the URL classification dataset. We can use this information to identify areas where the algorithm may need improvement and adjust the parameters or features used in the model to optimize its performance.

In the case of the decision tree algorithm applied to a URL classification dataset, the confusion matrix would have four

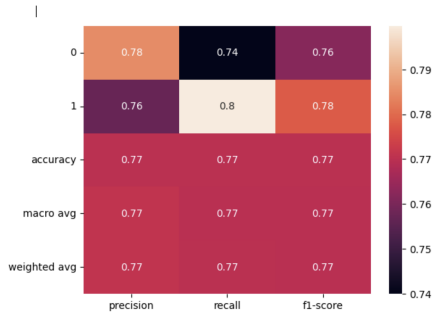

Fig. 10. Classification Report of Decision tree after Normalization

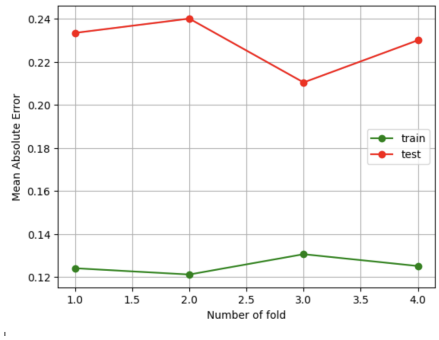

Fig. 11. K-fold validation shows Mean Absolute Error and Number of fold

elements:

- True Positive (TP): The number of samples that were correctly classified as positive, meaning the URL belongs to a specific category.
- False Positive (FP): The number of samples that were incorrectly classified as positive, meaning the URL does not belong to a specific category but was predicted to belong to that category.
- False Negative (FN): The number of samples that were incorrectly classified as negative, meaning the URL belongs to a specific category but was not predicted to belong to that category.
- True Negative (TN): The number of samples that were correctly classified as negative, meaning the URL does not belong to a specific category.

K-Fold cross-validation is a technique used to evaluate the performance of a machine learning model by dividing the dataset into k subsets (or "folds"), using k-1 subsets for training the model and the remaining subset for testing the model. This process is repeated k times, with each fold being used for testing the model once.

Here is how k-fold cross-validation can be applied to training a Decision Tree for URL classification:

1) The dataset is divided into k subsets, each with roughly the same number of samples
2) The Decision Tree algorithm is trained k times, with each subset being used once for testing the model and the remaining k-1 subsets being used for training the model.
3) The performance of the model is evaluated by averaging the performance metrics (such as accuracy, precision, recall, and F1-score) obtained from each of the k iterations.
4) The hyperparameters of the Decision Tree algorithm (such as the maximum depth of the tree) can be tuned using grid search or other optimization techniques based on the average performance across the k iterations.
5) The final model is trained on the entire dataset using the optimized hyperparameters.

The performance of a Decision Tree Classifier on a URL classification task was evaluated using a Receiver Operating

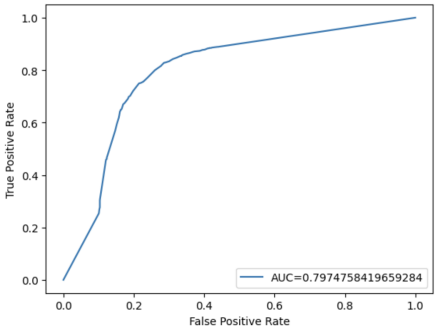

Fig. 12. ROC Curve for Decision Tree Classifier

Characteristic (ROC) curve. The Area Under the Curve (AUC) score of the classifier was found to be 0.79, An excellent AUC score is typically considered to be 0.8 or 0.9. The AUC score of 0.79 achieved by the current Decision Tree Classifier demonstrates that it is close to an excellent AUC score. Overall, the results indicate that the Decision Tree Classifier is a highly effective model for URL classification.

IV. Comparison

In general, KNN-classifiers are the best performing algorithm for phishing detection, with an accuracy of around 79.3 percent. Decisions trees are second in performance, with an accuracy of 76.7 percent. Lastly, logistic regression has an accuracy around 73.2 percent. By comparing the AUC between logistic regression and decision trees, the AUC is around 0.8 for decision trees, while the AUC is around 0.73 for logistic regression. An AUC of 0.8 is considered excellent by scientific standards. Therefore, decisions trees outperform logistic regression. However, accuracy calculations show that KNN-classifier fare better than decisions trees. It is interesting to note that all three algorithms differ by 3 percent in accuracy. It is also a bit shocking to see that even though logistic regression is a commonly-used classifier for fraud detection, it is not the most accurate classifier without substantial performance tuning or cross-validation.

KNN-Classifier	79.3%
Decision Tree	76.7%
Logistic Regression	73.2%

Fig. 13. Accuracy Comparison between Three Algorithms

Decision Trees	0.79
Logistic Regression	0.73

Fig. 14. AUC Comparison between Logistic Regression and Decision Trees

V. Future Directions

If given 3-6 more months to implement this project, our group would compare these algorithms to other algorithms learned in class such as neural networks, naive bayes, and support vector machines. By comparing several algorithms with each other, we can effectively determine which algorithm is the best classifier. Also, our group would spend some time improving the accuracy of our existing models. One way we could do this is by sourcing more data for training and testing. Sourcing more data would take some more time since we must analyze datasets to ensure that are no null values present. We could also clean up our existing dataset further by removing any outliers. Furthermore, we could spend some time working on feature selection. For example, we could use a decision tree to select the features that have the greatest weight on outcomes. These different methods would allow us to identify and perfect a reputable phishing classifier within 3-6 months.

VI. Conclusion

This problem has been solved well because of the different perspectives taken in order to research and offer a solution. This solution contains an in-depth analysis of the dataset. The selected attributes for training the model were carefully chosen, and the solution also implements three different and popular machine learning algorithms. Most importantly, there was significant tuning done to the parameters of the prediction models to ensure the performance is optimized and the final accuracy is as high as possible given the current configuration of the problem. The obtained accuracy results are considerably high numbers, and provide a lot of confidence. In addition, the manual and dynamic testing that was done using different url addresses yielded very accurate results for detecting whether an url is phishing or legitimate.

References

[1] Tiwari, Shashwat. "Web Page Phishing Detection Dataset." Kaggle, 27 June 2021, https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset.

[2] "IEEE Conference Template." Overleaf, Online LaTeX Editor, https://www.overleaf.com/latex/templates/ieee-conference-template/grfzhhncsfqn.

[3] Catak, F. O., Sahinbas, K., Dörtkardeş, V. (2021). Malicious URL Detection Using Machine Learning. In A. Luhach A. Elçi (Eds.), Artificial Intelligence Paradigms for Smart Cyber-Physical Systems (pp. 160-180). IGI Global. https://doi.org/10.4018/978-1-7998-5101-1.ch008

[4] SlashNext. "SlashNext's State of Phishing Report Reveals More than 255 Million Attacks in 2022, Signaling a 61 Percent Increase in Phishing Year-over-Year." PR Newswire: Press Release Distribution, Targeting, Monitoring and Marketing, 26 Oct. 2022, https://www.prnewswire.com/news-releases/slashnexts-state-of-phishing-report-reveals-more-than-255-million-attacks-in-2022-signaling-a-61-increase-in-phishing-year-over-year-301659518.html.

[5] Sklearn.neighbors.kneighborsclassifier. scikit. (n.d.). Retrieved April 1, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html