Exploring Machine Learning Techniques for Classification Tasks and Hyperparameter Fine Tuning: A Comparative Analysis of Decision Tree and Multilayer Perceptron Classifiers for Phishing Site Identification

Aaron Gabrielle Dichoso C, Benmar Sim Ramirez G, John Kirsten Espiritu C, Mauries Lopez A

De La Salle University – Laguna Campus
727V+352, LTI Spine Road, Laguna Blvd, Biñan, Laguna

Abstract. The paper investigates the performance of two different machine learning models for the task of classifying web pages as either legitimate sites or phishing sites. Cybercrimes like data phishing involves tricking users into disclosing their sensitive personal information such as passwords or credit card numbers by masquerading as trustworthy entities. The paper uses a dataset sourced from Kaggle that consists of 10,000 web pages, half of which contain legitimate sites, and the other half contain phishing sites, alongside the 48 features to determine the classification. The Decision Tree (DT) and Multilayer Perceptron (MLP) models were used for the research. For the DT model, its hyperparameter of the maximum depth of the decision tree was tested and crossvalidated, and for the MLP mode, its hyperparameters of maximum number of iterations, activation function, and initial learning rate were all tested to find the best configuration for each model. The models were then evaluated based on their accuracy scores, F1 scores, precisions, and recalls. The analysis shows that both models have achieved high levels of accuracy, but the decision tree slightly outperforms the multilayer perceptron in terms of evaluating by the confusion matrix and its explainability.

Keywords: Phishing, Machine learning, Supervised Learning.

1 Introduction

In the 21st century, science and technology play a central role in our daily lives, providing comfort and convenience in performing routine activities. For instance, instead of sending a message through traditional mail, which would require a trip to the post office, technology enables us to send messages instantly with just a tap on our devices. While the benefits of technology are undeniable, it is important to be mindful of the potential risks associated with its use. The messages we send is data that can be stored, and can become personally identifying. As such, it is crucial to ensure that the data privacy and security of a user is maintained. There have been instances where data has been forcefully gathered without the user's permission through methods such as phishing.

Phishing is a type of cybercrime in which personal information is stolen without the user's permission. This can take on many forms, including someone impersonating to know the target, text messages, or scam emails that appear legitimate. The goal of phishing is to gather sensitive data such as personal information, including name, address, age, and height, as well as bank details and passwords. This can pose a significant threat to the individual's security and financial well-being.

Data phishing is a global issue that affects individuals and organizations regardless of their location. The Philippines is not immune to this cybercrime. In fact, according to a report by Kaspersky Security Network (Baclig, 2022), the Philippines had the highest number of data phishing attempts among Southeast Asian countries from February to April 2022. A table in the source stated the phishing vulnerability of countries:

1. Philippines 9.9%
2. Malaysia 8.49%
3. Thailand 7.93%
4. Indonesia 7.70%
5. Vietnam 7.45%
6. Singapore 3.30%.

Therefore, being knowledgeable about technology and data privacy can help individuals take steps to protect themselves from phishing attempts.

Knowing the dangers of phishing, the research aims to create machine learning models that will help with the identification of phishing sites and classification of sites as either legitimate or made for phishing. By using Python as the programming language, decision tree and multi-layer perceptrons are used. Thee hyperparameters of these models are then fine tuned to optimize their performances. After which, the confusion matrix will then be used for comparison. With visualization of each result, the researchers can aid explainability of the dataset to be presented to much concise and clear ideas.

2 Phishing Dataset

The dataset utilized for the research is a Phishing Dataset for Machine Learning, comprising 10,000 web pages, which include 5,000 phishing web pages and 5,000 legitimate web pages (Tiwari, 2021). These web pages were collected from January to May 2015 and from May to June 2017. The dataset consists of 48 features extracted from the total 10,000 web pages using an improved feature extraction technique employing the Selenium WebDriver, with 48 features that are used to describe the nature of the website (See Table 1).

Table 1. Features of the Dataset and their Descriptions.

Feature Name	Descriptions
NumDots	Number of dots in the URL
SubdomainLevel	Number of subdomains in the URL

PathLevel	Number of directories in the URL path
UrlLength	Length of the URL
NumDash	Number of dashes in the URL
NumDashInHostname	Number of dashes in the hostname portion of the URL
AtSymbol	Whether or not the '@' symbol appears in the URL
TildeSymbol	Whether or not the '~' symbol appears in the URL
NumUnderscore	Number of underscores in the URL
NumPercent	Number of percent signs in the URL
NumQueryComponents	Number of query components in the URL
NumAmpersand	Number of ampersands in the URL
NumHash	Number of hash signs in the URL
NumNumericChars	Number of numeric characters in the URL
NoHttps	Whether or not the URL uses HTTPS
RandomString	Whether or not the URL contains a random string
IpAddress	Whether or not the URL contains an IP address
DomainInSubdomains	Whether or not the domain appears as a subdomain
DomainInPaths	Whether or not the domain appears in the URL path
HttpsInHostname	Whether or not the hostname portion uses HTTPS
HostnameLength	Length of the hostname portion of the URL
PathLength	Length of the path portion of the URL
QueryLength	Length of the query string in the URL
DoubleSlashInPath	Whether or not there are double slashes in the path portion
NumSensitiveWords	Number of sensitive words in the URL
EmbeddedBrandName	Whether or not URL contains an embedded brand name
PctExtHyperlinks	Percentage of external hyperlinks in the web page
PctExtResourceUrls	Percentage of external resource URLs in the web page
ExtFavicon	Whether or not the web page uses an external favicon
InsecureForms	Whether or not the web page contains insecure forms
RelativeFormAction	Whether or not the form action is relative
ExtFormAction	Whether or not the form action uses an external URL
AbnormalFormAction	Whether or not the form action is abnormal
PctNullSelfRedirectHyperlinks	Percentage of self-redirect hyperlinks that are null
FrequentDomainNameMismatch	Whether or not there are frequent domain name mismatches
FakeLinkInStatusBar	Whether or not the web page contains a fake link
RightClickDisabled	Whether or not right-clicking is disabled on the web page
PopUpWindow	Whether or not the web page contains a pop-up window
SubmitInfoToEmail	Whether or not the web page submits info to an email address
IframeOrFrame	Whether or not the web page contains an iframe or frame
MissingTitle	Whether or not the web page has a missing title
ImagesOnlyInForm	Indicates whether images are present only in forms or not
SubdomainLevelRT	The number of levels in the subdomain of the URL
UrlLengthRT	The length of the URL, in real-time
PctExtResourceUrlsRT	The percentage of external resource URLs, in real-time
AbnormalExtFormActionR	Indicates whether the external form action is abnormal or not
ExtMetaScriptLinkRT	The number of external meta, script, and link elements
PctExtNullSelfRedirectHyperlinksRT	The percent of external hyperlinks having null self redirects,

The dataset used for predicting whether a website is legitimate or phishing employs a set of features. The distribution of classes in this dataset is even, consisting of 5000 instances each of phishing and legitimate websites. This balanced distribution of classes is optimal for machine learning, as it ensures that both classes receive equal attention during the training process, enabling effective discrimination between the two. Overall, this dataset is well-suited for building a reliable predictive model for web page authenticity.

3 Methodology

For the purposes of this study, two machine learning models were created with Python through Scikit-learn and the Anaconda platform, one using the Decision Tree Classifier, and another with a Multilayer Perceptron (MLP). The researchers used the phishing dataset from Kaggle to train both models. As for the ML Pipeline to be followed, the researchers used a standard ML Pipeline, with steps of preprocessing the input data, training the model, evaluating the model, and generating outputs.

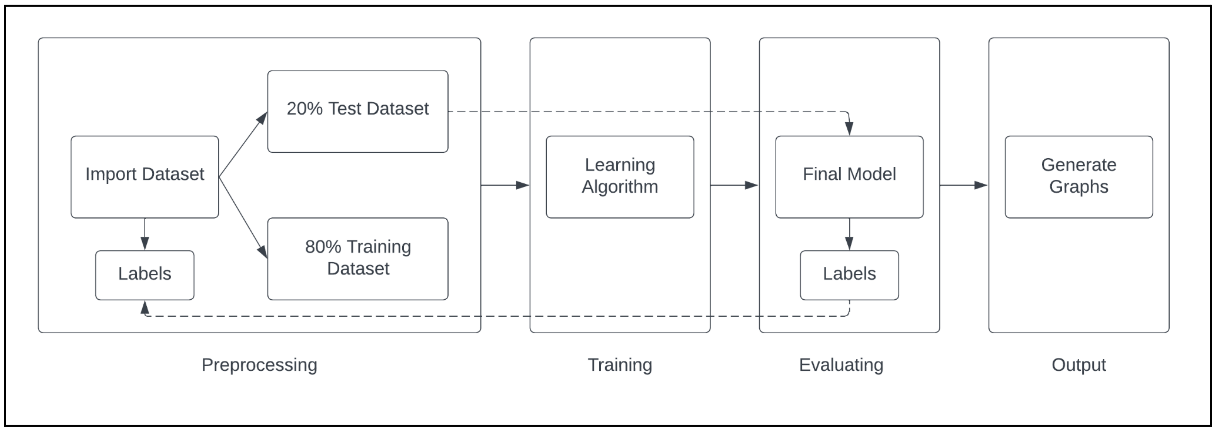
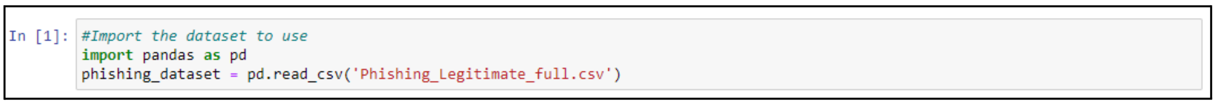


Figure 1. Visual Representation of the machine learning pipeline for both Decision Tree and Multilayer Perceptron

3.1 Preprocessing

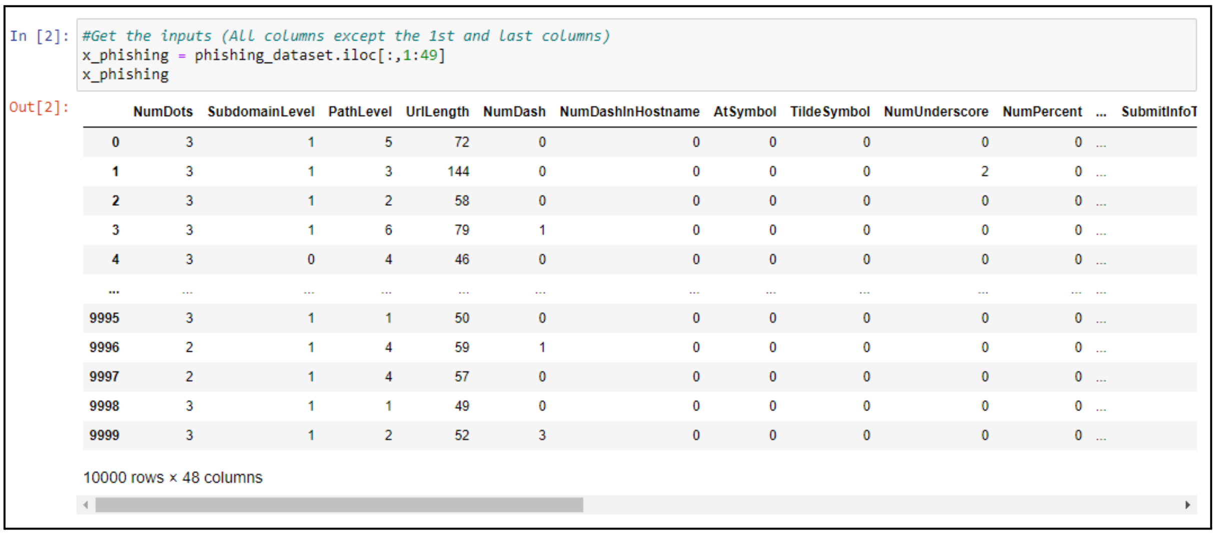
There are 4 main processes done in order to achieve the desired output. First is the preprocessing stage, where the dataset is loaded and analyzed. The labels are saved for later evaluation and the dataset is split with 20% going into testing and with the remaining 80% for training.



```
In [1]: #Import the dataset to use
        import pandas as pd
        phishing_dataset = pd.read_csv('Phishing_Legitimate_full.csv')
```

Figure 2. Importing pandas library for manipulating and analyzing database

Importing the library called pandas is required for the dataset to be manipulated and analyzed. According to (Stackabuse, 2019) using pandas can bring advantages such as it allows the dataset to be presented in a way that it is suitable for data analysis, methods that can be convenient for data filtering, and the most important one is it allows data to be read from a variety of formats such as CSV, TSV, MS Excel, and etc.

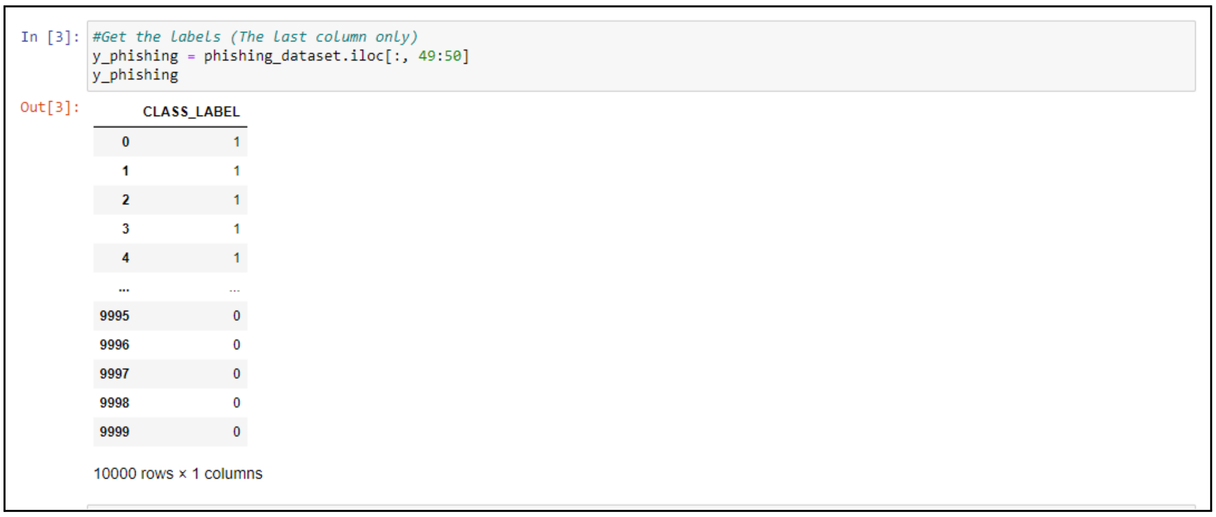

```
In [2]: #Get the inputs (All columns except the 1st and last columns)
        x_phishing = phishing_dataset.iloc[:,1:49]
        x_phishing
```

```
Out[2]:
        NumDots  SubdomainLevel  PathLevel  UrlLength  NumDash  NumDashInHostname  AtSymbol  TildeSymbol  NumUnderscore  NumPercent  ...  SubmitInfoT
   0          3               1          5         72        0                  0         0            0              0           0  ...
   1          3               1          3        144        0                  0         0            0              2           0  ...
   2          3               1          2         58        0                  0         0            0              0           0  ...
   3          3               1          6         79        1                  0         0            0              0           0  ...
   4          3               0          4         46        0                  0         0            0              0           0  ...
 ...        ...             ...        ...        ...      ...                ...       ...          ...            ...         ...  ...
9995          3               1          1         50        0                  0         0            0              0           0  ...
9996          2               1          4         59        1                  0         0            0              0           0  ...
9997          2               1          4         57        0                  0         0            0              0           0  ...
9998          3               1          1         49        0                  0         0            0              0           0  ...
9999          3               1          2         52        3                  0         0            0              0           0  ...

10000 rows × 48 columns
```

Figure 3. Displays the dataset

After the csv file was successfully read by the pandas' method, the dataset is then assigned to a variable called *x_phishing*. The variable contains the columns of the imported dataset which is about 1 to 48 inclusive columns, resulting from the structure of the .csv file in which these columns contain the attributes to be used.


```
In [3]: #Get the labels (The last column only)
        y_phishing = phishing_dataset.iloc[:, 49:50]
        y_phishing
```

```
Out[3]:
        CLASS_LABEL
   0              1
   1              1
   2              1
   3              1
   4              1
 ...            ...
9995              0
9996              0
9997              0
9998              0
9999              0

10000 rows × 1 columns
```

Figure 4. Getting the labels

Here, the labels in the last column are saved into a variable called y_phishing to be used for later. 1 means that the URL is a phishing site and 0 means that it is legitimate.

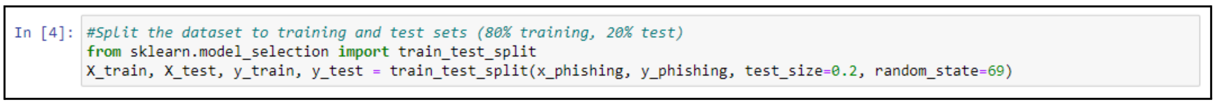

```
In [4]: #Split the dataset to training and test sets (80% training, 20% test)
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(x_phishing, y_phishing, test_size=0.2, random_state=69)
```

Figure 5. Splitting dataset into training and test sets (80% training, 20% test)

As the variables are properly initialized. From the library called sklearn.model_selection it imported train_test_split to be used on splitting the dataset into two sets. Specifically, training and test sets. In the argument of the train_test_split method, test_size is the one representing the data set split into 20% for test sets. Which then assigned to the variables X_train, X_test, y_train, and y_test.

3.2 Training

Then, the training data is sent into the training stage. Here, the learning algorithm uses the data to train the model, slowly fitting the dataset after every iteration. For the hyperparameters, the maximum tree depthwas considered for the decision tree model, and was tested for values 2 to 50. For the MLP model, the maximum number of iterations was tested for values between 2 to 200, making sure that the model does not overfit the training data. Different activation functions were also tested for the MLP model such as Rectified Linear Unit (ReLU), Identity, TanH, and Logistic as well as different learning rates of 1%, 2%, and 3%.

3.3 Evaluating

After the models were trained, they were evaluated by predicting the labels of the test dataset and comparing it with the actual labels. From there, the model's score on accuracy, precision, recall, and F1 metrics can be calculated.

For each hyperparameter, all other hyperparameters were first set to a default value, and different models with different values of said hyperparameter were compared using the aforementioned measures. The hyperparameter that gave the best performance will be picked. In the case of the MLP model, the maximum number of iterations was first evaluated, then the activation functions, then the initial learning rates.

The confusion matrices of the MLP model and DT model were then used as comparison of their performances.

3.4 Output

Through the matplot library, the researchers then generated graphs for the accuracy, precision, recall, and F1 metrics after every iteration or increasing tree depth. For the multilayer perceptron, graphs relating to different activation functions and learning rates were also produced.

4 Results and Analysis

In machine learning, hyperparameters are used to determine the data representation for a dataset before training a model. The depth of a decision tree is an example of a hyperparameter that can be tuned to find the optimal values that result in the best performance of the model. Cross-validation is a technique that can be used to estimate the performance of a model on new data and can be used in conjunction with hyperparameter tuning to achieve the best performance. By comparing the depth of a decision tree with metrics such as accuracy, F-measure, precision, and recall, it is possible to understand how the depth affects the computation of the confusion matrix.

4.1 Decision Tree. Comparison of Accuracy Scores, F- measures, Precisions and Recalls of the models.

Figure 6 provides a visual representation of the relationship between the depth of a decision tree and its accuracy. According to the table, at depths ranging from 2 to 5, the accuracy slowly improves from 91.45% to 94.80%, which could be improved by increasing the depth further. The accuracy gradually increases with the depth of the decision tree and reaches a global maximum of 94.85% at a maximum depth of 8. After which, the accuracy of succeeding depths does not increase any further as it plateaus to a percentage of 94.75% after a maximum depth of 13 (See Figure 6).

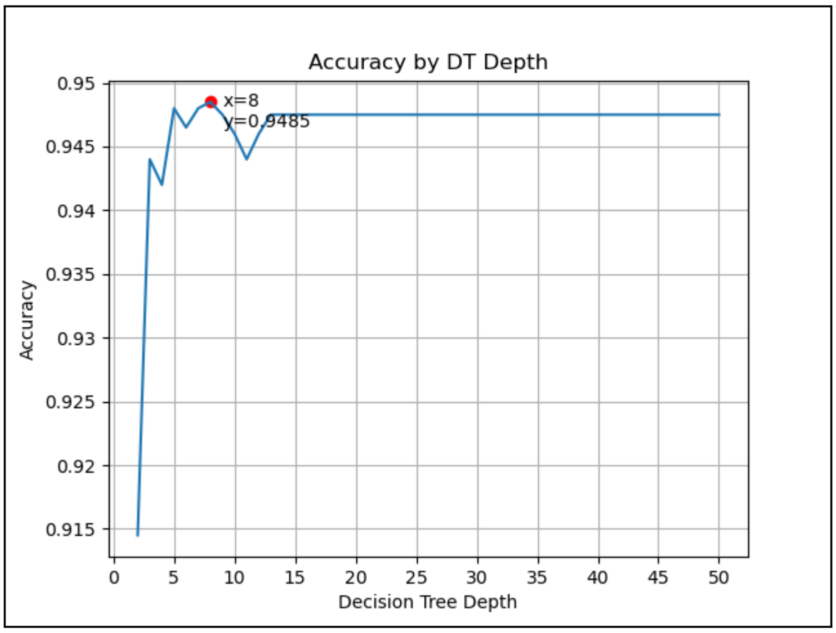


Figure 6. Visual Representation of the Depth of the Tree plotted with its Accuracy.

This suggests that at a depth of 8, the decision tree has found the most effective value for maximum depth to obtain the best accuracy, and increasing the maximum depth any further would negatively affect the accuracy of the decision tree.

Similar findings can also be observed in the precision, recall, and f1 values tested according to incrementing values of maximum depth. The maximum value of precision that can be obtained is also attributed to having a maximum tree depth of 8 as well (See Figure 7).

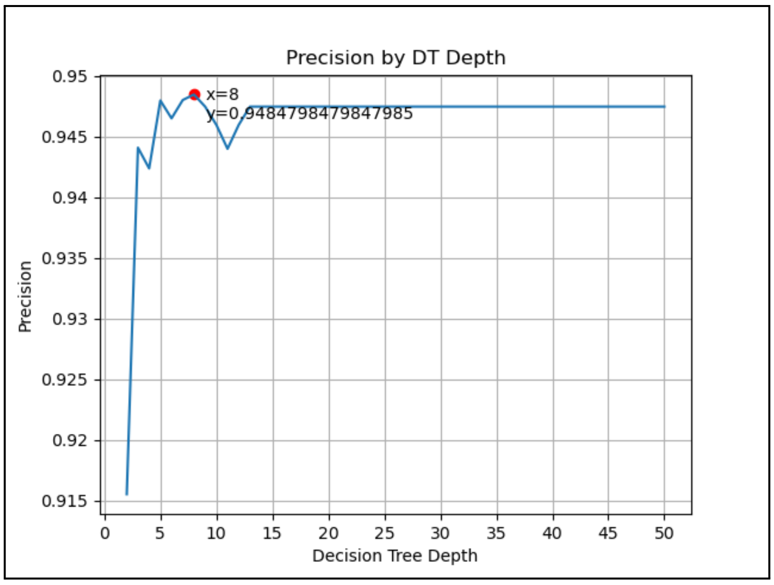


Figure 7. Visual Representation of the Depth of the Tree plotted with its Precision.

According to Scikitlearn (2007), precision is the ability of a data model to not label a negative record as positive. Which means, the data represented shows that when the maximum depth of the decision tree is set at 8, it can correctly identify 94.85% of all phishing websites as not legitimate.

The recall score of the decision tree set at different values of maximum depth can also be observed. In this representation it has the same situation with the previous statistics. However, it is about recall. According to (Scikitlearn, 2007), recall is the ability of a data model to not label a positive example as that of a negative example (See Figure 8).

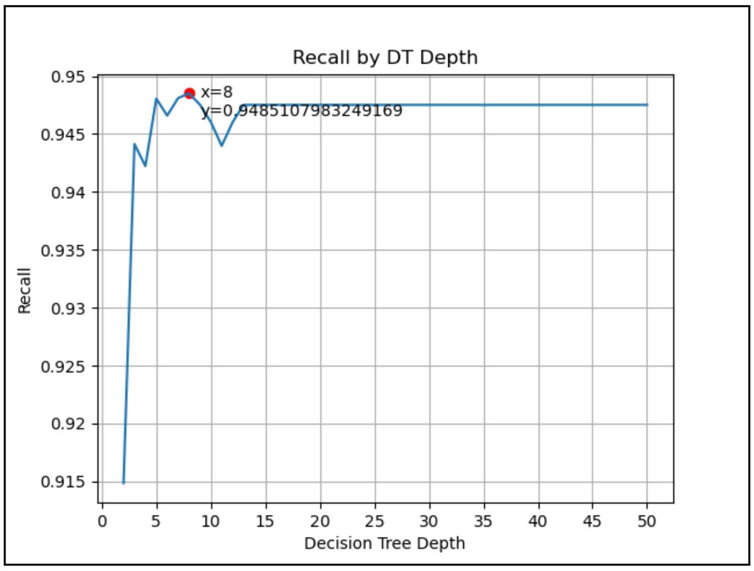


Figure 8. Visual Representation of the Depth of the Tree plotted with its Precision.

From the results, it can be said that the data model is able to effectively not label legitimate websites as phishing websites when the maximum depth of the decision tree is also set at 8, providing a recall score of 94.85%, making it effective at not disallowing users entry from otherwise safe websites.

Finally, the F1 Score of the data model when set at different maximum depths is also observed. The F1 score is described as the harmonic mean between the precision and recall of a data model. As such, from the previous observations, it should be no surprise that the decision tree classifier has its highest F1 score when the maximum depth is also set at 8 (See Figure 9).

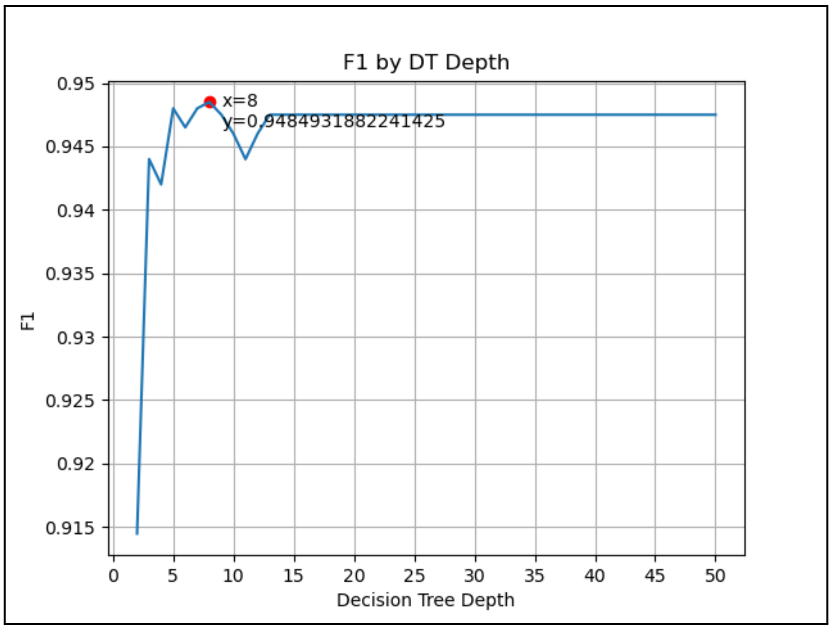


Figure 9. Visual Representation of the Depth of the Tree plotted with its F1 score.

From the results obtained from testing varying maximum depth values for the decision tree, it can be said that the best value for the aforementioned hyperparameter is a maximum depth of 8, giving an accuracy of 94.85%, a precision of 94.85%, a recall of 94.85%, and an f1 of 94.85% (See Figure 10).

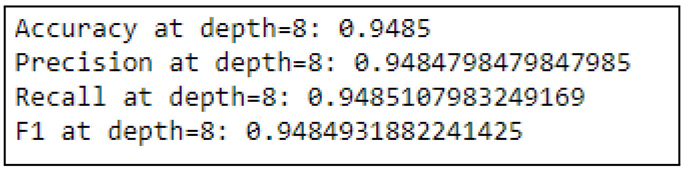
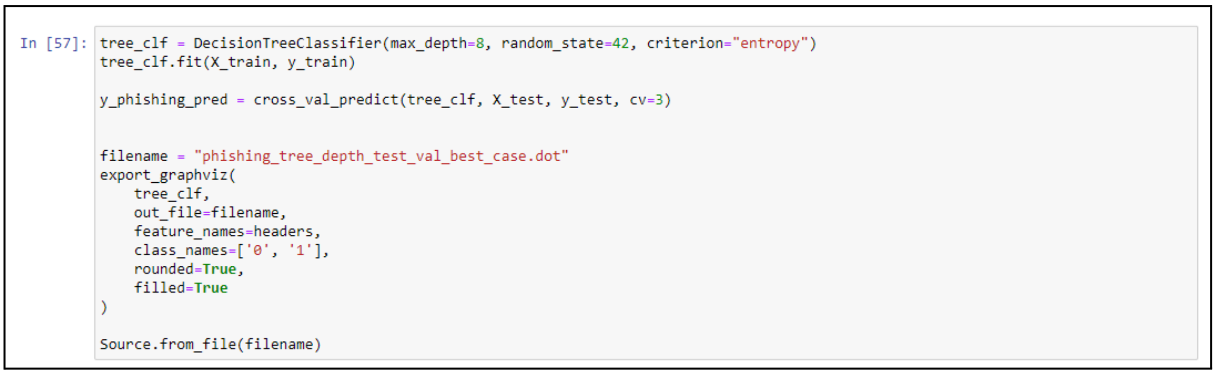

```
Accuracy at depth=8: 0.9485
Precision at depth=8: 0.9484798479847985
Recall at depth=8: 0.9485107983249169
F1 at depth=8: 0.9484931882241425
```

Figure 10. Precise values of Accuracy, Precision, Recall, and F1 at max_depth=8

As for why the optimal maximum depth for the decision tree is at 8, it may be due to the number of attributes given. When the maximum depth is set lower than 8, it may be said that the decision tree is unable to fully utilize all 48 attributes, and when it is higher than 8, the decision tree may start overfitting to the training data instead.

Moving on from the analysis of hyperparameters, the decision tree was made in Jupyter Notebook (See Figure 11), with the resulting decision tree having a root node of PctExtNullSelfRedirectHyperlinksRT, which can be interpreted as the attribute which provides the most information gain. This can be derived from observing the values of its immediate children nodes, with attributes of SubmitInfoToEmail and PctExtHyperlinks, having dramatically lower remaining entropy, with corresponding values of 0.234 and 0.889 (See Figure 11).



```
In [57]: tree_clf = DecisionTreeClassifier(max_depth=8, random_state=42, criterion="entropy")
         tree_clf.fit(X_train, y_train)

         y_phishing_pred = cross_val_predict(tree_clf, X_test, y_test, cv=3)


         filename = "phishing_tree_depth_test_val_best_case.dot"
         export_graphviz(
             tree_clf,
             out_file=filename,
             feature_names=headers,
             class_names=['0', '1'],
             rounded=True,
             filled=True
         )

         Source.from_file(filename)
```

Figure 11. Code of the Creation of the Decision Tree Classifier, with max_depth=8

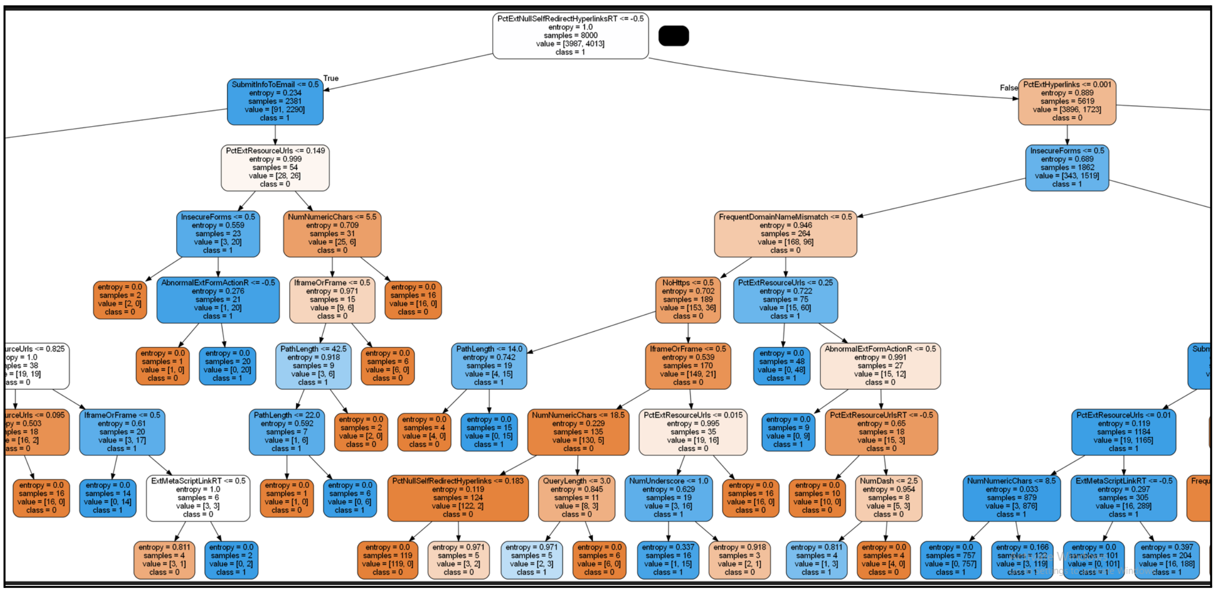


Figure 12. Snapshot of Decision Tree.

The confusion matrix is represented by a figure (See Figure 12), which is for the decision tree at max depth of 8. Each quadrant corresponds to being a positive (legitimate website) or negative (phishing website). With the confusion matrix, it clarifies each of the values gathered from the dataset which will give a better understanding of the results. For the first quadrant (top right), it represents the false positive. For the second quadrant (top left), it represents the true positive or the data model identified as a legitimate website. Third quadrant (bottom left), it represents the false positive which means that the model predicted a legitimate website but it was actually a phishing website. Lastly, the fourth quadrant (bottom right) represents the true negative which means that the model correctly identified a phishing website.

The values of each quadrant represents how the decision tree identifies each website and represents it to its corresponding quadrant. Likeso, there are 960 legitimate websites that are not phishing websites. 53 false positive websites or legitimate websites that are falsely accused of being a phishing website. 50 legitimate websites that are considered to be phishing websites. And lastly, 937 phishing websites that have been correctly identified by the data model.

When cross validated with the testing data, having 2000 example websites it can be seen that the decision tree, correctly identifies 960 legitimate sites as legitimate, 937 phishing sites as phishing sites, 53 legitimate sites as phishing sites, and 50 phishing sites as legitimate sites (See Figure 13).

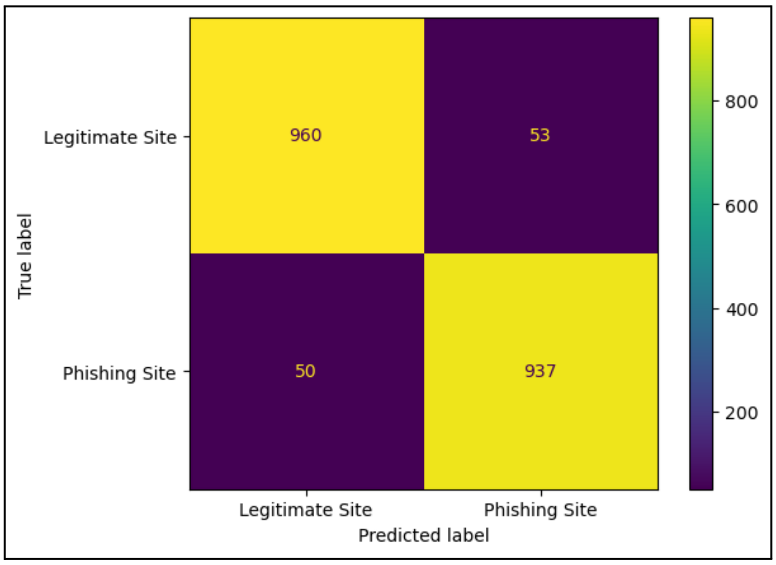


Figure 13. Confusion matrix of the decision tree at max_depth=8

4.2 Multilayer Perceptron (MLP)

Compared to decision trees, MLPs have many different hyperparameters that can be fine tuned in order to maximize its performance. For the purposes of the paper, 3 hyperparameters were given focus. These are:

1. The maximum number of iterations allowed
2. The activation function used; and
3. The initial learning rate used

First is the maximum number of iterations allowed by the MLP for training. This is important as not enough iterations may not allow the MLP to be fully trained, yet having too many iterations may lead to a case of overfitting, where the MLP is not made with generalization in mind, as it focuses too much on the limited training data. To determine the minimum amount of maximum iterations needed by the MLP, we considered 2 cases.

First is the maximum iterations needed to obtain the maximum values of accuracy, precision, recall, and f1 scores when cross validated with the test data. For accuracy, it can be seen that at max_iterations=2, the MLP has an accuracy of 91.45%, which steadily increases, fluctuating around the 95% mark on the graph. When the model reaches 108 iterations, it reaches the maximum attainable accuracy of 95.35%. Furthermore, it can be seen that continuing to iterate past that point, where x = 108, does little to improve the accuracy of the model (See Figure 14).

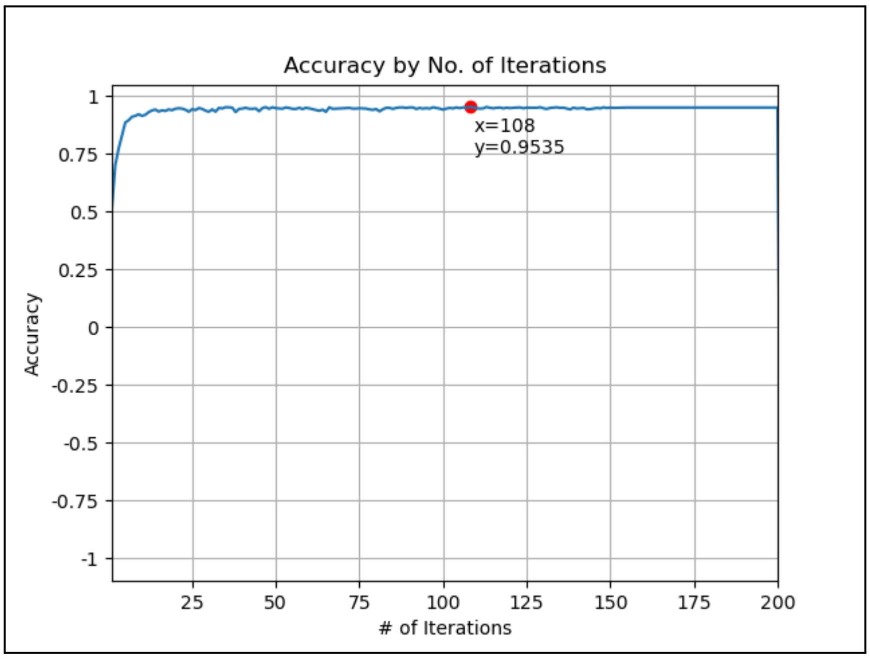


Fig. 14. Visual Representation of the number of iterations in relation with accuracy.

Similar findings can also be observed for the precision, recall, and f1 score of the MLP. At max_iterations=108, the precision score of the MLP is at 95.37% (See Figure 15).

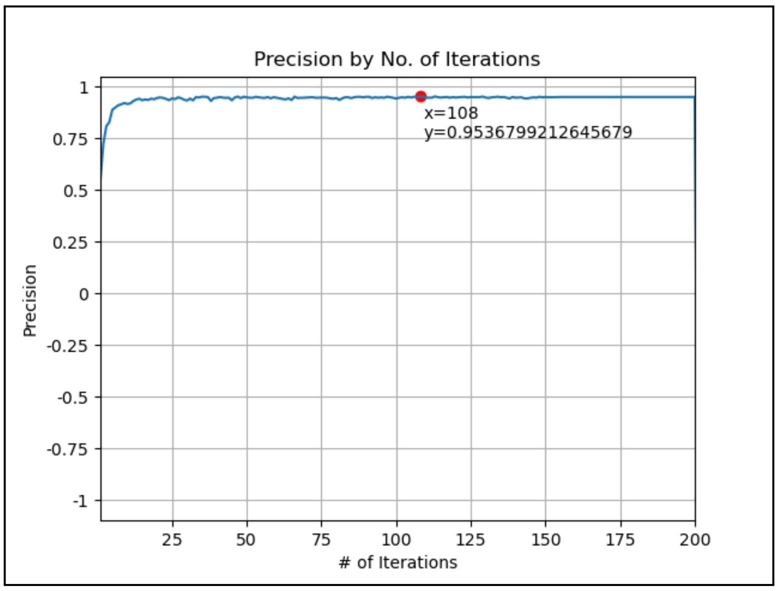


Figure 15. Visual Representation of the number of iterations in relation with precision.

The precision score entails that 95.37% of legitimate websites are classified as legitimate by the MLP.

Similarly, the recall score of the MLP is at a maximum when max_iterations=108, getting a recall score of 95.37% (See Figure 16). Recall is calculated from the total true positives over the total actual positive data points. This means having a high recall score means having fewer false negatives. In terms of phishing detection, having low recall means some malicious phishing URLs are not flagged by the model.

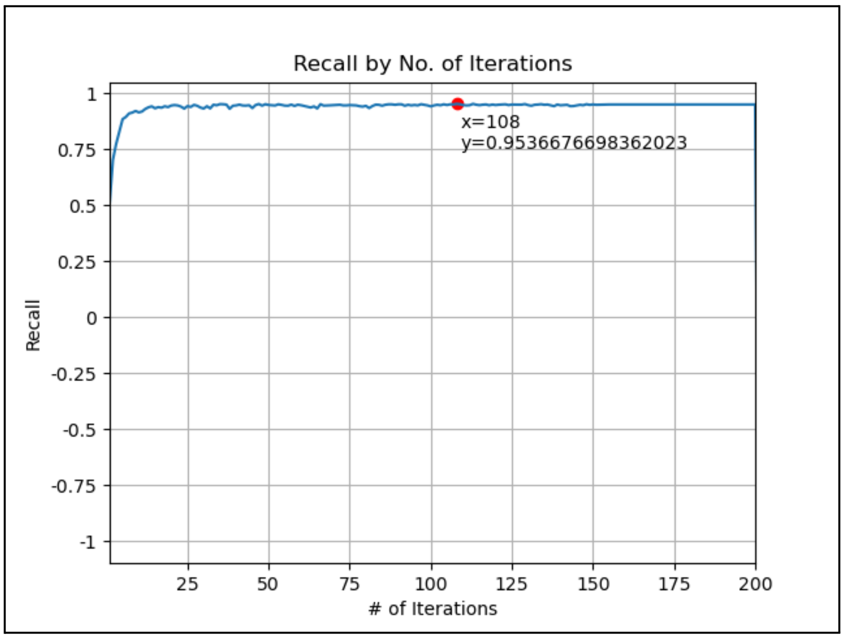


Figure 16. Visual Representation of the number of iterations in relation with recall.

Finally, the F1 score is obtained, and is seen to have a maximum value of 95.35% when max_iterations=108. In terms of phishing detection, having a high F1 score means only a few URLs are wrongly classified by the model (See Figure 17).

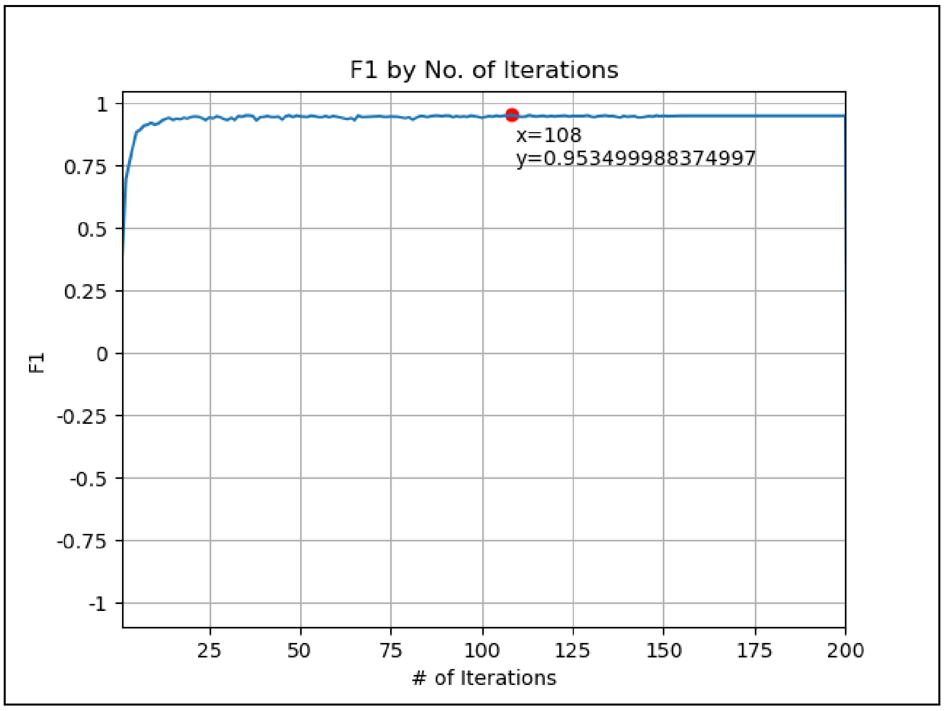
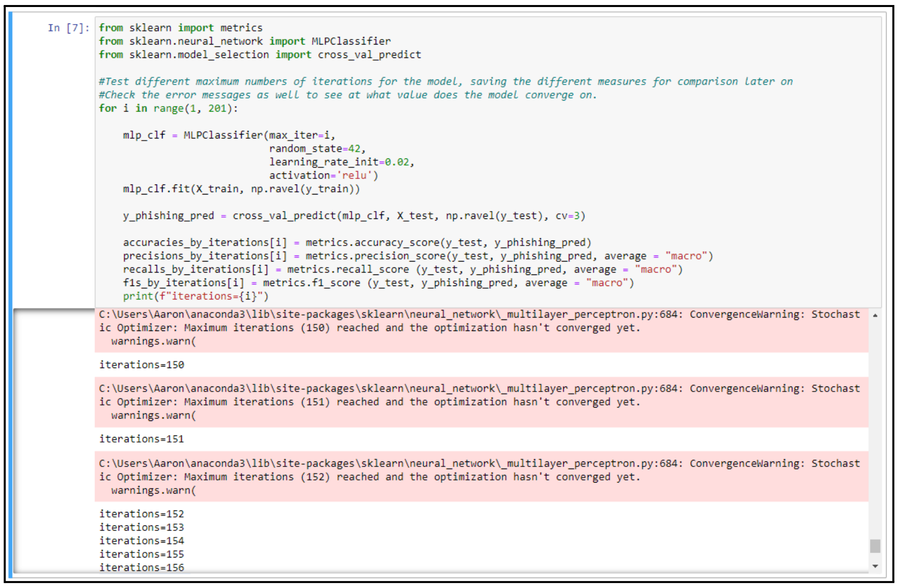


Figure 17. Visual Representation of the number of iterations in relation with F1.

From the results, it can be said that having a maximum iteration value of 108 leads to obtaining the best performance. However, the point of convergence, or the moment when all training data is correctly classified by the MLP, can also be considered and compared. This can be obtained by observing the warnings thrown by the training script, considering differing values of maximum iterations. From the warning log, it can be seen that no warning regarding the MLP not converging yet is seen when the maximum iterations are set at 153 (See Figure 18).



```
In [7]: from sklearn import metrics
        from sklearn.neural_network import MLPClassifier
        from sklearn.model_selection import cross_val_predict

        #Test different maximum numbers of iterations for the model, saving the different measures for comparison later on
        #Check the error messages as well to see at what value does the model converge on.
        for i in range(1, 201):

            mlp_clf = MLPClassifier(max_iter=i,
                                    random_state=42,
                                    learning_rate_init=0.02,
                                    activation='relu')
            mlp_clf.fit(X_train, np.ravel(y_train))

            y_phishing_pred = cross_val_predict(mlp_clf, X_test, np.ravel(y_test), cv=3)

            accuracies_by_iterations[i] = metrics.accuracy_score(y_test, y_phishing_pred)
            precisions_by_iterations[i] = metrics.precision_score(y_test, y_phishing_pred, average = "macro")
            recalls_by_iterations[i] = metrics.recall_score (y_test, y_phishing_pred, average = "macro")
            f1s_by_iterations[i] = metrics.f1_score (y_test, y_phishing_pred, average = "macro")
            print(f"iterations={i}")
```

```
C:\Users\Aaron\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:684: ConvergenceWarning: Stochast
ic Optimizer: Maximum iterations (150) reached and the optimization hasn't converged yet.
  warnings.warn(

iterations=150

C:\Users\Aaron\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:684: ConvergenceWarning: Stochast
ic Optimizer: Maximum iterations (151) reached and the optimization hasn't converged yet.
  warnings.warn(

iterations=151

C:\Users\Aaron\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:684: ConvergenceWarning: Stochast
ic Optimizer: Maximum iterations (152) reached and the optimization hasn't converged yet.
  warnings.warn(

iterations=152
iterations=153
iterations=154
iterations=155
iterations=156
```

Figure 18. Warning Messages of Jupyter Notebook Regarding How the Model has not yet reached convergence in the Current Maximum Amount of Iterations.

To determine which of the two possible values of the maximum number of iterations is to be used, their loss values, validation scores, and confusion matrices are compared. Upon observing these figures, it can be said that there is no difference between the two possible values, hence the lower value of 108 is to be used (See Figure 19).

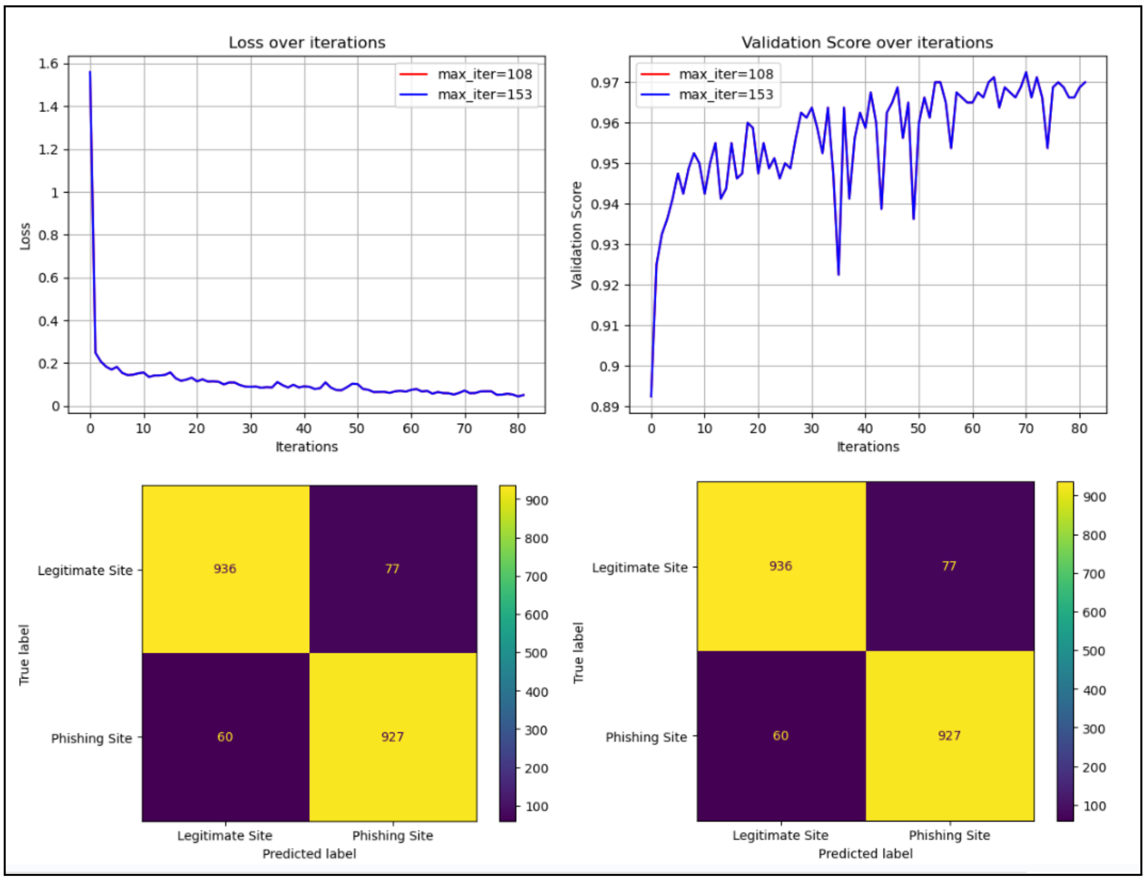


Figure 19. Loss values (Upper Left), Validation scores (Upper Right), and Confusion matrices of max_iterations=108 (Bottom Left) and max_iterations=153 (Bottom Right)

After determining the optimal value for the maximum number of iterations, the activation function to be used is then considered. There are 4 choices provided by sklearn for the activation function of the MLP Classifier, these are:

1. The Identity Function, which returns f(x) = x
2. The Rectified Linear Unit (ReLu) Function; which returns f(x) = max(0, x)
3. The Hyperbolic tangent (tanh) Function, which returns f(x) = tanh(x); and
4. The Logistic Sigmoid Function, which returns f(x) = 1 / (1 + exp(-x))

New models were then created using max_iterations=108, but with differing activation functions to test their performances. The loss values and validation scores of the models are then obtained, where it can be seen that some models stopped iterating earlier than other models, implying that the model may have converged earlier than other models (See Figure 20).

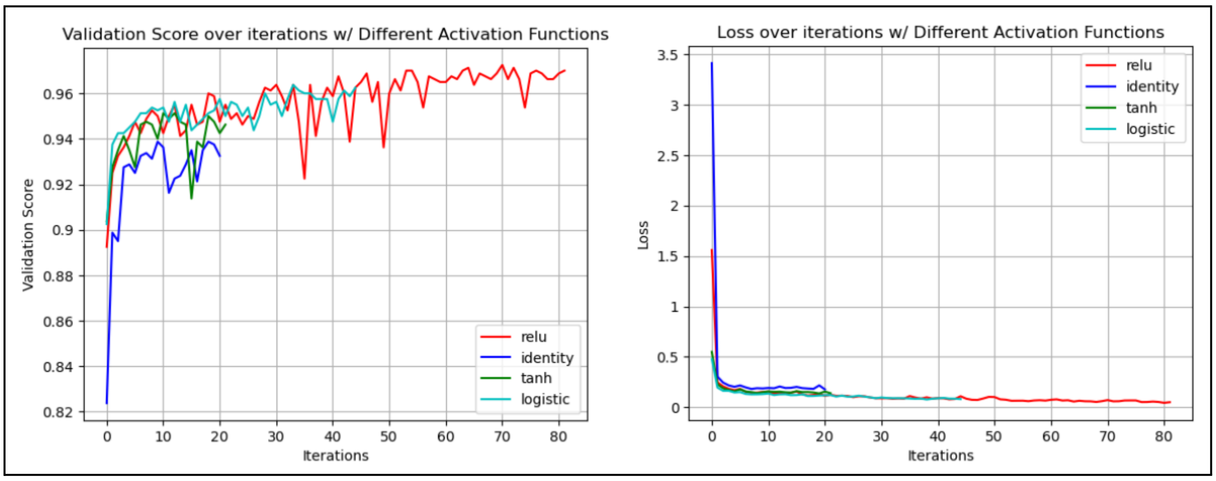


Figure 20. Validation Scores (Left) and Loss Values (Right) of Models with max_iterations=108 and Different Activation Functions.

For the activation function to choose, it can be observed that the logistic sigmoid function gives the best performance for all metrics, as it has correctly determined more true positives and true negatives compared to the other activation functions (See Figure 21).

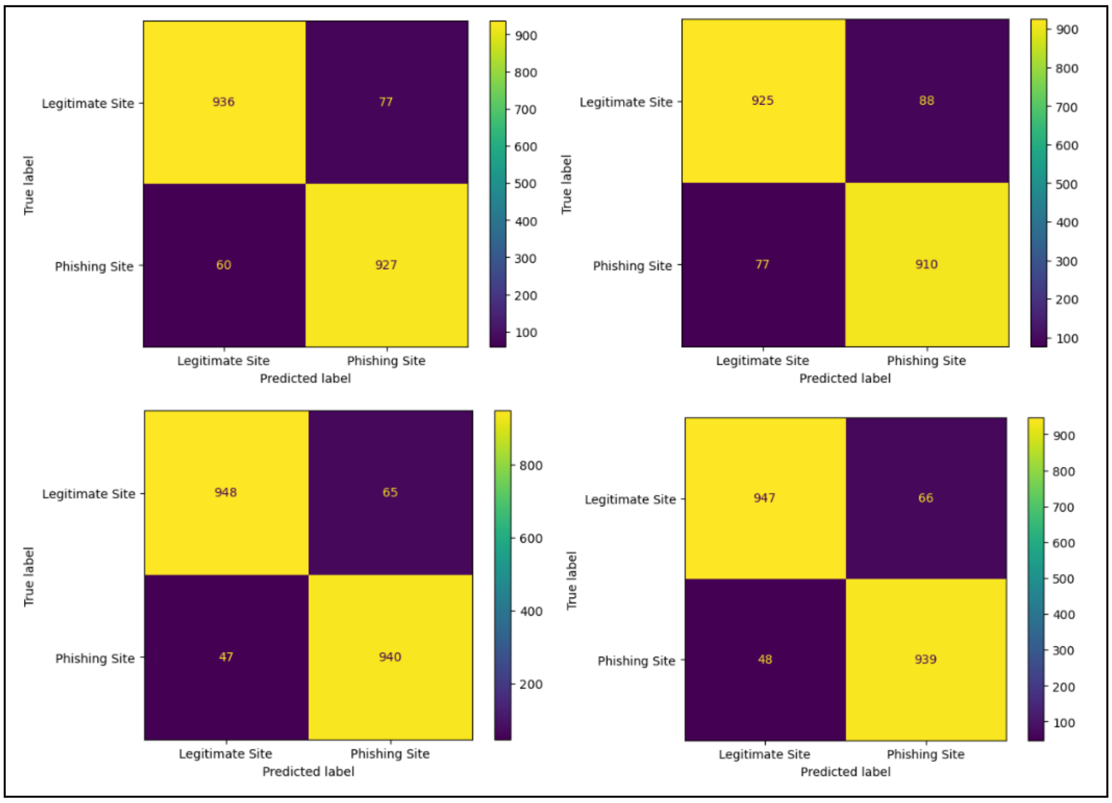


Figure 21. Confusion Matrices of Models with max_iterations=108, and Activation Functions of ReLu (Upper Left), Identity (Upper Right), Logistic Sigmoid (Bottom Left), and tanh (Bottom Right).

Finally, after obtaining the best activation function to use in terms of performance, the initial learning rate should then be fine tuned. Three initial learning rates were considered, 1%, 2%, and 3%, and the initial learning rate which gave the best performance in terms of accuracy was then used. This is done by using models which have had their max_iterations set to 108, its activation function set the logistic sigmoid function, and their initial learning rates were different from one another. From the confusion matrices of these models, it can be observed that having a learning rate of 2% gave the best performance (See Figure 22).

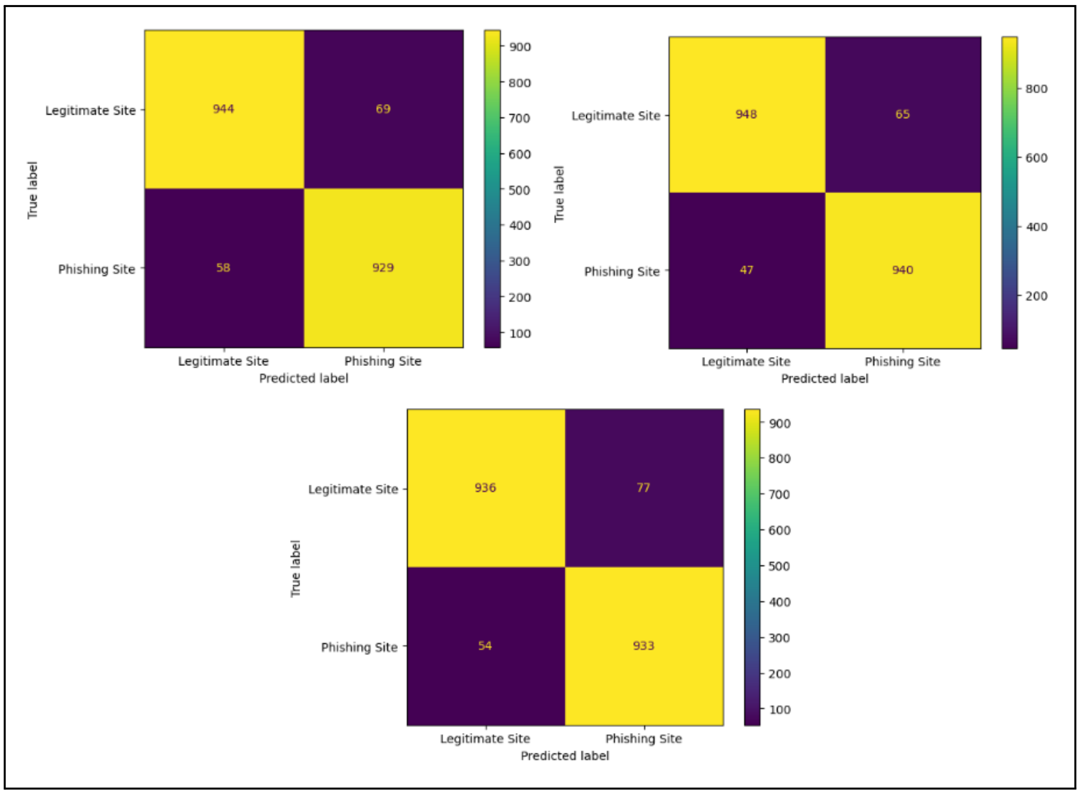


Figure 22. Confusion Matrices of Models with max_iterations=108, activation_function="Logistic Sigmoid", and Initial Learning Rates of 1% (Upper Left), 2% (Upper Right), and 3% (Bottom).

It can be said a learning rate of 3% overcorrects the adjustment of weights, and a learning rate of 1% would undercorrect the adjustment of weights.

All in all, the optimal values for the hyperparameters of the MLP are a maximum number of 108 iterations, a logistic sigmoid activation function, and an initial learning rate of 2%. Such a model would result in an accuracy of 94.40%, a precision of 94.40%, a recall of 94.41%, and an f1 score of 94.40% (See Figure 23).

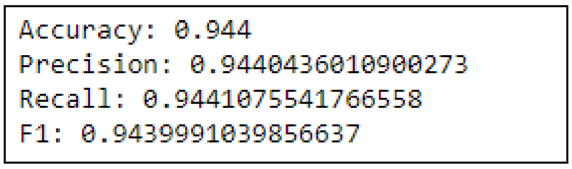

```
Accuracy: 0.944
Precision: 0.9440436010900273
Recall: 0.9441075541766558
F1: 0.9439991039856637
```

Figure 23. Precise values of Accuracy, Precision, Recall, and F1 for MLP at max_iterations=108, activation_function="Logistic Sigmoid", initial_learning_rate=0.02

When cross validated with the testing data, having 2000 example websites, it can be seen that the MLP correctly identifies 948 legitimate sites as legitimate, 940 phishing sites as phishing sites, 65 legitimate sites as phishing sites, and 47 phishing sites as legitimate sites (See Figure 24).

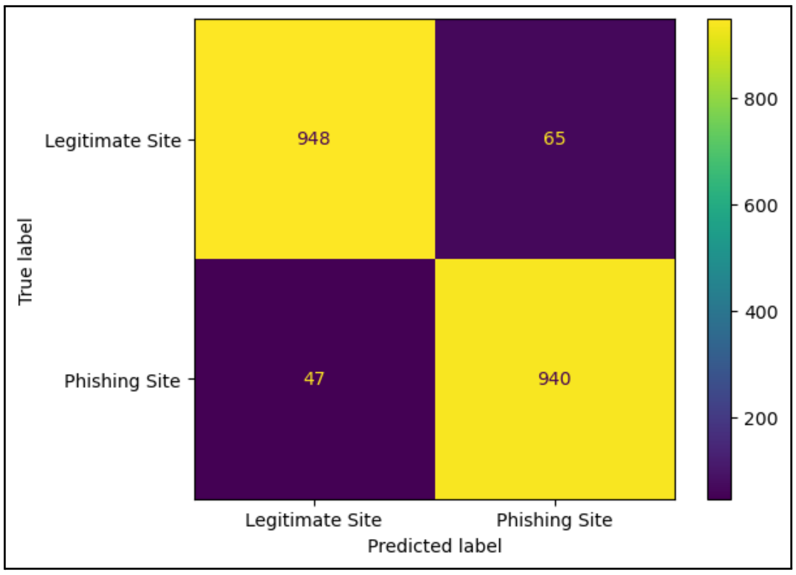


Figure 24. Confusion matrix of the MLP at max_iterations=108, activation_function="Logistic Sigmoid", initial_learning_rate=0.02

4.3 Comparison and Analysis of the Results using the 2 Classification Techniques

The confusion matrices of the DT and MLP Classifiers can be compared. For the true positives of the DT Model it gives an observation of legitimate sites of 960 identified as legitimate. On the other hand, the MLP model only got 948 identified legitimate sites. For the false positives, the decision tree gave 53 legitimate sites falsely accused of being a phishing site, and 65 legitimate sites for the MLP model. For the false negatives, the DT model labeled 50 phishing sites that were actually legitimate sites. 47 phishing websites were incorrectly labeled by the MLP model. Lastly, the DT model obtained 947 true negatives, while the MLP model obtained 940 true negatives (See Figure 25)

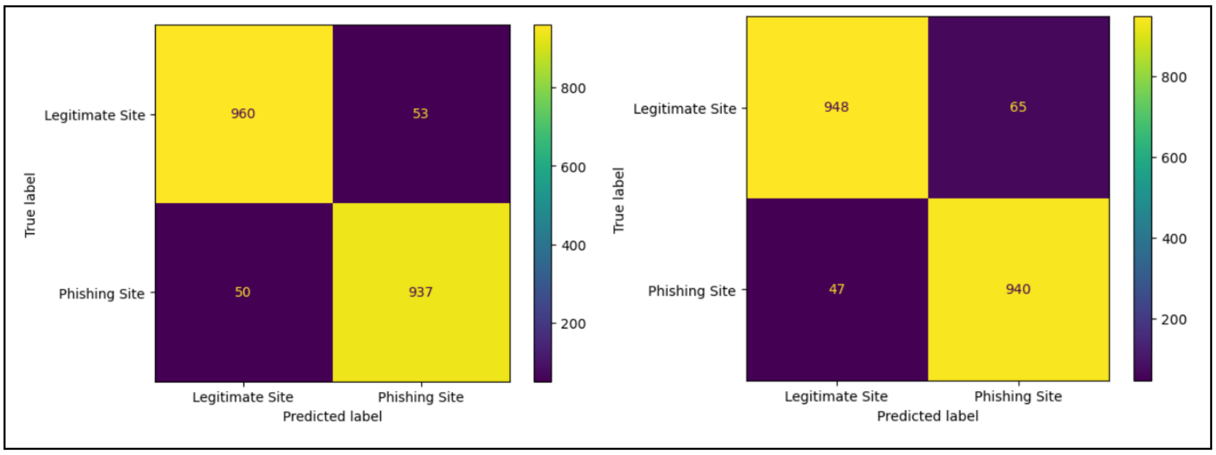


Figure 25. Confusion matrix of the decision tree (left) and the multilayer perceptron (right)

Upon analysis of the different measures, it can be seen that the DT model has values of 94.85% across the board, while the MLP model has values of about 94.40% (See Table 2).

Metric	Decision Tree	Multilayer Perceptron
Accuracy	94.85%	94.40%
Precision	94.85%	94.40%
Recall	94.85%	94.41%
F1 measure	94.85%	94.40%

Table 2. Scores of the decision tree and multilayer perceptron models on performance metrics.

From the results, it can be concluded that the DT model is better at differentiating phishing sites from legitimate sites, only by a small margin of about 0.40% compared to the MLP model.

However, the DT Model can be said to be more valuable due to its explainability, as it also provides a decision tree, in which the process of decision making is readily available and understandable. It is a visual representation of the best model generated using Python with libraries like graphviz. It gives a better understanding of the best case of the decision tree with its depth as the hyperparameter. Each node has these attributes: characteristic of a website, entropy which indicates between 0 for a phishing website and closest to 1 makes it a secured website, number of samples, an array of value indicating the splits of the characteristic, and a class that indicates if it is a phishing website (1) or not (0). With the help of an entropy and cross-validation it greatly helped the decision tree to properly and accurately state which websites are a possible threat in users browsing experience and a website that can be visited without any problems. Due to it having a numerous amount of characteristics for each domain link, it makes the separation of the class labels much more accurate than having a few characteristics, thus aiding to the performance of the DT model (See Figure 26).

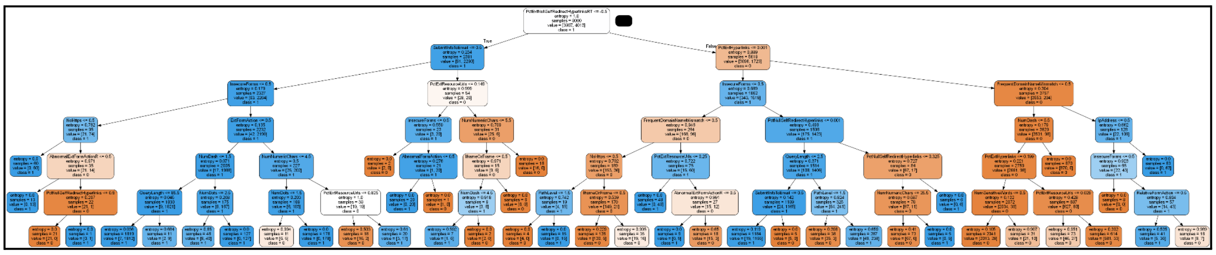


Figure 26. Visual Representation of the depth test of the decision tree in its best case.

5 Conclusion and Recommendations

This project utilized Scikit, a popular open-source machine learning library for Python, to access and analyze a dataset with multiple features. Also including: graphviz, numpy, and other libraries that may help the researchers in visually represent the dataset used. With this dataset, the researchers managed to use different machine learning techniques like decision trees and multilayer perceptrons. With these machine learning techniques, there are few helpful concepts that could help the accuracy of each statistical observation to the results of each machine learning algorithm such as cross-validation.

In conclusion, for the dataset made by Shashwat Titwari called “Phishing Database for Machine Learning”. The best data model to use is the decision tree. With the comparison of each metric for the confusion matrix, the decision tree is above each metric. The decision tree has 94.85% accuracy, while on the other hand for the multilayer perceptron it has 94.40%.

Generally, decision trees perform well for classification problems, on the other hand multilayer perceptrons are better suited for larger datasets with more complex relationships between variables.

To improve the performance of the Decision Tree and MLP models in this scenario with the given dataset, there are several approaches we can consider.

The effectiveness of either models depended on numerous factors, like the size, algorithms and parameters used. For the MLP model, other processes that might increase its performance may be the fine tuning of the number of hidden layers, and the number of nodes in each hidden layer. Additionally, the performance of the DT model may increase even further with the use of different criteria, such as the Gini Index. Further testing of different hyperparameters and the comparison of performance metrics may be needed for these cases.

There are five aspects we can work on, the first one is Feature Engineering wherein the 48 features used to classify the web pages as legitimate or phishing could possibly be lessened, because not all of the features are relevant for the classifications. Having a more thorough understanding and selection of features can improve the accuracy for both models. The second one is Hyperparameter tuning, which is applicable for both models to improve efficiency. In the case of the Decision Tree model, the depth of the tree and the minimum number of samples required to split nodes can be modified. While for the MLP model, the number of hidden layers and neutrons per, can be tuned as well. The three other aspects include Ensemble methods, which is the combining of multiple models, Data augmentation which focuses on increasing the size of the dataset through transformation techniques, and the possibility of using other algorithms well-suited machine learning algorithms that may perform a better job than the current models.

Incorporating these changes in machine learning models can potentially enhance their performance in classification procedures, but it is important to take into account that the effectiveness of these approaches may vary depending on the given datasets and problems. Further experimentation, analysis and research are necessary to determine the most optimal combination of these approaches for any given task.

In the current case, it can be said that both the DT model and MLP model performs well, with the DT model being slightly more favorable due to a slightly higher measure, and better explainability.

6 References

1. Baclig, C. E. (2022, June 24). PH biggest target of phishing in Southeast Asia—cybersecurity report | Inquirer News. INQUIRER.net. https://newsinfo.inquirer.net/1615655/for-posting-edited-ph-biggest-target-of-phishing-in-southeast-asia-cybersecurity-report
2. Scikit-learn. (2007). Precision score. Retrieved April 20, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html
3. Scikit-learn. (2007). Recall score. Retrieved April 20, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html
4. Kuhn, M., & Johnson, K. (2013). Applied predictive modeling. Springer. https://link.springer.com/book/10.1007/978-1-4614-6849-3
5. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. https://link.springer.com/article/10.1023/A:1010933404324
6. Bishop, C. M. (2006). Pattern recognition and machine learning. Springer. https://www.springer.com/gp/book/9780387310732
7. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press. https://www.deeplearningbook.org/
8. Scikit-learn. (2007). Cross Validation. Retrieved April 20, 2023, from https://scikit-learn.org/stable/modules/generated/cross_validation.html
9. Stack Abuse. (n.d.). Beginner's Tutorial on the Pandas Python Library. Retrieved April 20, 2023, from https://stackabuse.com/beginners-tutorial-on-the-pandas-python-library/
10. Tiwari, S. (2021, May 28). Phishing Dataset for Machine Learning. Kaggle. https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning

Appendix A. Contribution of Members

Table 3. Contributions of Members.

Name	Contributions
Ramirez, Benmar	Documentation, Report, Coding
Dichoso, Aaron	Coding, Report, Documentation
Lopez, Mauries	Coding, Analysis Report, Documentation
Espiritu, John	Documentation, Report