**Phishing Websites Detection Using Machine Learning**

Table of Contents

[Abstract 5](#_Toc120479705)

[Introduction 6](#_Toc120479706)

[Problem Statement 6](#_Toc120479707)

[Methodology 7](#_Toc120479708)

[Data Wrangling 7](#_Toc120479709)

[Data Analysis 9](#_Toc120479710)

[Data Pre-Processing 13](#_Toc120479711)

[Machine Learning Models 14](#_Toc120479712)

[Decision Tree 14](#_Toc120479713)

[K-nearest Neighbor 16](#_Toc120479714)

[Logistic Regression 17](#_Toc120479715)

[Random Forest 19](#_Toc120479716)

[Results & Evaluation 22](#_Toc120479717)

[Conclusion 24](#_Toc120479718)

[References 25](#_Toc120479719)

**List of Figures**

[Figure 1 Taking random sample of 10000 7](#_Toc120480042)

[Figure 2 Dropping features having maximum values zero 7](#_Toc120480043)

[Figure 3 Dropping unique attribute 7](#_Toc120480044)

[Figure 4 Replacing missing values -1 with Nan values 8](#_Toc120480045)

[Figure 5 Dropping missing values of variables 8](#_Toc120480046)

[Figure 6 Dropping null values 8](#_Toc120480047)

[Figure 7 Reading Dataset & Displaying Five Rows 9](#_Toc120480048)

[Figure 8 Shape of the dataset 9](#_Toc120480049)

[Figure 9 Describe() function of the dataset 9](#_Toc120480050)

[Figure 10 Percentage of missing values in the dataset 10](#_Toc120480051)

[Figure 11 Percentage of missing values 10](#_Toc120480052)

[Figure 12 Phishing website class label total instances 11](#_Toc120480053)

[Figure 13 Distribution: email\_in\_url & server\_client\_domain 11](#_Toc120480054)

[Figure 14 Distribution: url\_google\_index & domain\_google\_index 12](#_Toc120480055)

[Figure 15 Distribution of url\_shortened & qty\_tilde\_url 12](#_Toc120480056)

[Figure 16 Correlation Heatmap Graph 13](#_Toc120480057)

[Figure 17 Defining input and output 13](#_Toc120480058)

[Figure 18 Random Oversampling & Under sampling using Smote Library 13](#_Toc120480059)

[Figure 19 Splitting the dataset into testing & training 14](#_Toc120480060)

[Figure 20 GridSearchCV for Decision Tree 14](#_Toc120480061)

[Figure 21 Decision Tree Implementation 15](#_Toc120480062)

[Figure 22 Confusion Matrix: Accuracy, Precision, & Recall 15](#_Toc120480063)

[Figure 23 KNN - GridSearch CV 16](#_Toc120480064)

[Figure 24 K-nearest Neighbour 16](#_Toc120480065)

[Figure 25 K-nearest Neighbour - Confusion Matrix 17](#_Toc120480066)

[Figure 26 GridSearchCV for Logistic Regression 18](#_Toc120480067)

[Figure 27 Logistic Regression Implementation 18](#_Toc120480068)

[Figure 28 Confusion Matrix: Logistic Regression 19](#_Toc120480069)

[Figure 29 GridSearchCV on Random Forest 20](#_Toc120480070)

[Figure 30 Random Forest Classifier 20](#_Toc120480071)

[Figure 31 Confusion Matrix: Random Forest 21](#_Toc120480072)

[Figure 32 ROC Curve of All Classifiers 22](#_Toc120480073)

# Abstract

Online scams are increasing with the rapid increase in the advancement of technology in the internet world. In cybercrimes, phishing websites are a common concern where criminals or hackers use phishing websites to steal the sensitive information of people. This study highlights the importance of research over phishing website detection, and provides a solution using supervised machine learning technique to enable websites, apps, or any enterprises to have a system that detects phishing websites to protect the users and their privacy from these phishing websites. The machine learning models: Decision Tree, Logistic Regression, Random Forest, and KNN are developed in this project and performance metrics including accuracy, precision, and recall were used to determine the best model for the dataset. So, Random Forest outperformed all other models with an accuracy of 96%.

# Introduction

Online scams are increasing with the rapid increase in the advancement of technology in the internet world. In cybercrimes, phishing websites are a common concern where criminals or hackers use phishing websites to steal the sensitive information of people. These websites are similar to the authentic ones because they use the same visual appearance, icons, logos, etc. to trick people to fall prey. Phishers commonly target financial or payment institutions to steal information like banks or credit card details.

Since the majority of customers access government and financial institution services online, there has

phishing attacks have significantly increased during the past few years. Phishers began to make money and now operate a lucrative company. Phishers attack vulnerable people using a variety of techniques, including SMS, VOIP, faked links, and fake websites. It is quite simple to make fake websites that, in terms of design and content, resemble real websites. These websites would even have the same material as their authentic webpages. Additionally, attackers pose as high-level security measures who provide users with security questions and answers. Users who react to those questions are more susceptible to phishing scams. Many studies have been conducted to stop phishing assaults by various communities worldwide. By identifying the websites and educating people to recognize phishing websites, phishing assaults can be stopped. One of the effective methods for identifying phishing websites is machine learning algorithms. Several techniques for spotting phishing websites have been presented in this study. Anti-Phishing Working Group reports that at least 316,747 phishing attacks happened in December 2021– highest no. ever recorded (Globalnewswire, 2022). Therefore, to protect the user’s sensitive information, we need advance tools to counter the phisher’s plotted traps and make internet a safer place for generations to come.

## Problem Statement

There has been a huge increase in the amount of phishing websites recorded in the year 2021 – highest ever, according to APWG (Anti-Phishing Working Group). So, in order to protect users’ data and privacy over internet from scammers or attackers behind phishing websites, there’s a need to develop a machine learning model that detects phishing website with precision and higher accuracy for the betterment of society.

# Methodology

## Data Wrangling

In fig. 1, we took sample of 10,000 rows from the dataset through pandas sample function.



Figure 1 Taking random sample of 10000

In fig. 2, we removed all those features that have features maximum value equal to zero. This is because these features have no importance and therefore it is important that we neglect those in modelling of our project.

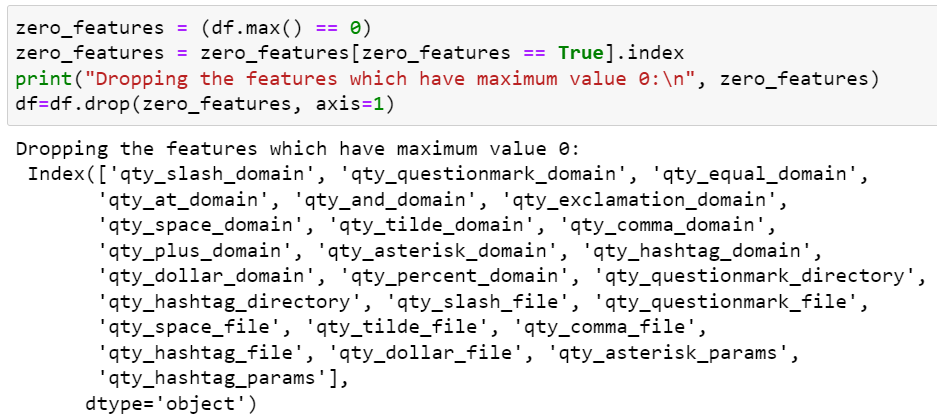


Figure 2 Dropping features having maximum values zero

In fig. 3, we dropped asn\_ip feature because it is a unique feature.



Figure 3 Dropping unique attribute

In the dataset, the missing values was represented by -1. As it doesn’t need it for data analysis; therefore, we replaced -1 values with Nan values as seen in fig. 4.

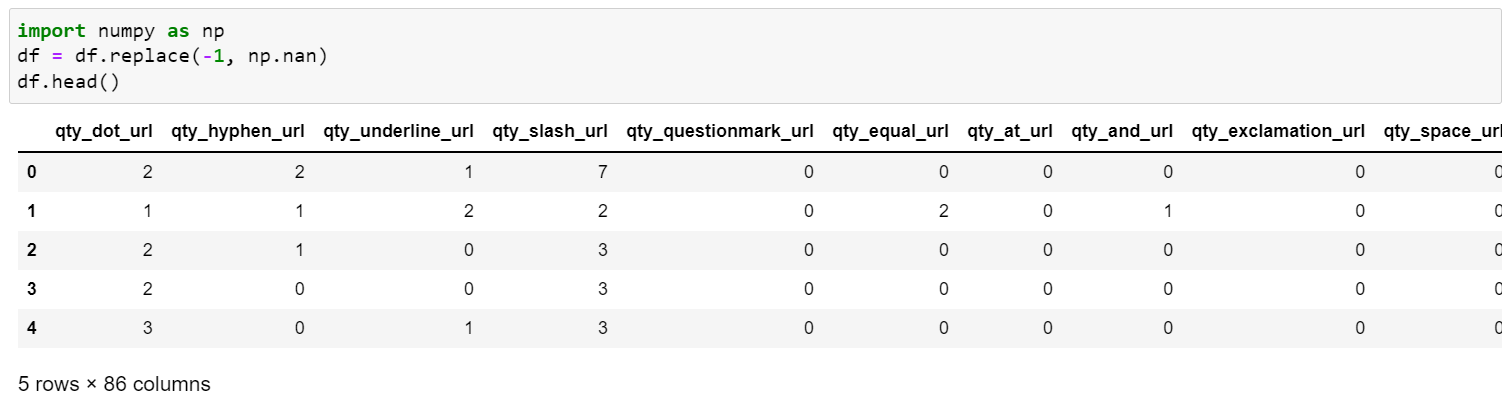


Figure 4 Replacing missing values -1 with Nan values

In fig. 5, we removed all those features that have more than 50% of missing values. This is because it poses problem in modelling and the accuracy of the model is affected.

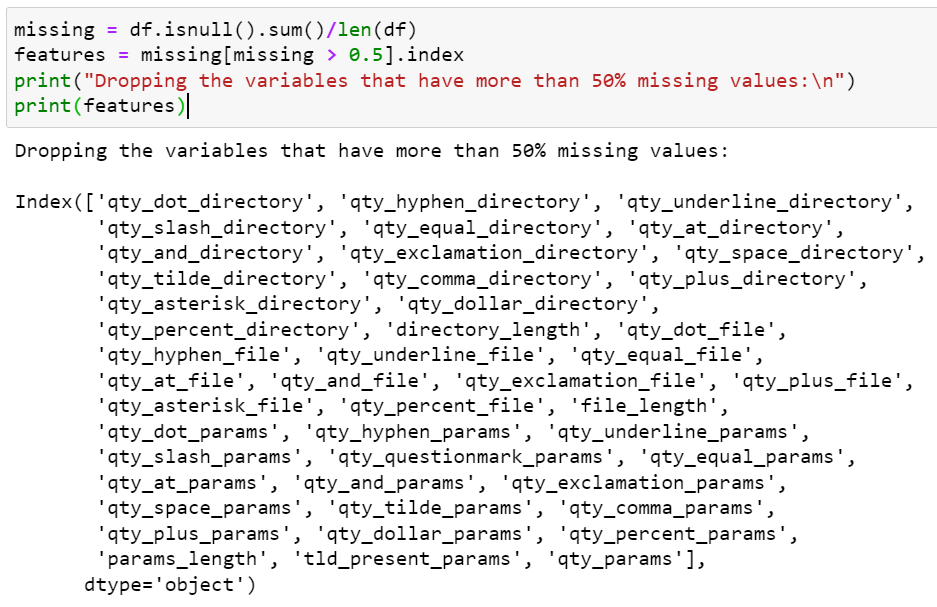


Figure 5 Dropping missing values of variables

In fig. 6, we dropped null values present in the dataset using dropna() function based on rows as axis equal to 0.



Figure 6 Dropping null values

## Data Analysis

In fig. 7, It can be seen the first five rows of the dataset are being displayed by head() function. There are 112 columns in the dataset.

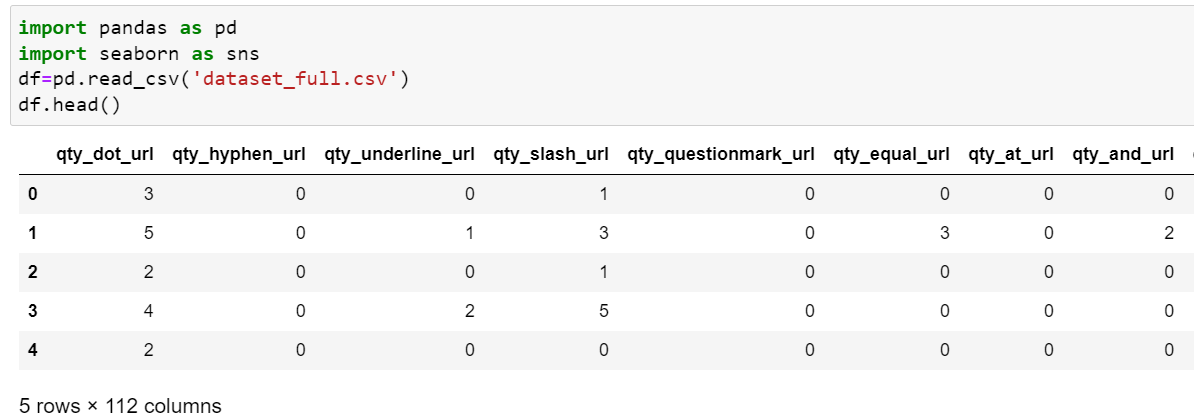


Figure 7 Reading Dataset & Displaying Five Rows

It can be seen that there are 88647 rows in the dataset and 112 columns. This result was achieved through shape variable of panda’s library.

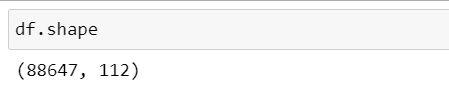


Figure 8 Shape of the dataset

In fig. 9, the describe() function displays some important information of the dataset features such as mean, standard deviation, minimum, percentiles, minimum and maximum rows.

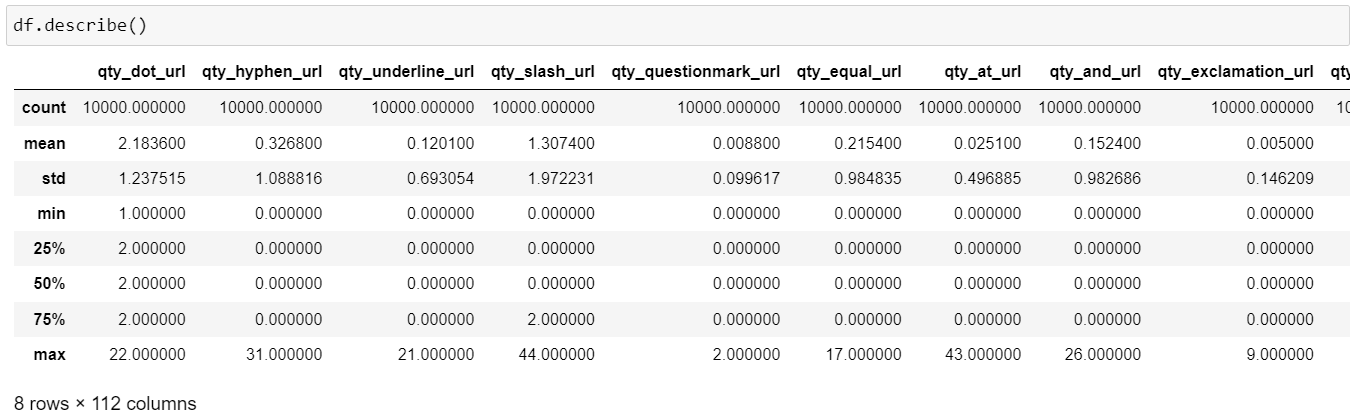


Figure 9 Describe() function of the dataset

It can be seen from fig. 10 and 11 that there are some features that have missing values present in the dataset. For example, feature qty\_exclamation\_params, qty\_plus\_params, qty\_comma\_params have 91.57% of missing values in the dataset. Therefore, we will handle these missing values in data pre-processing.

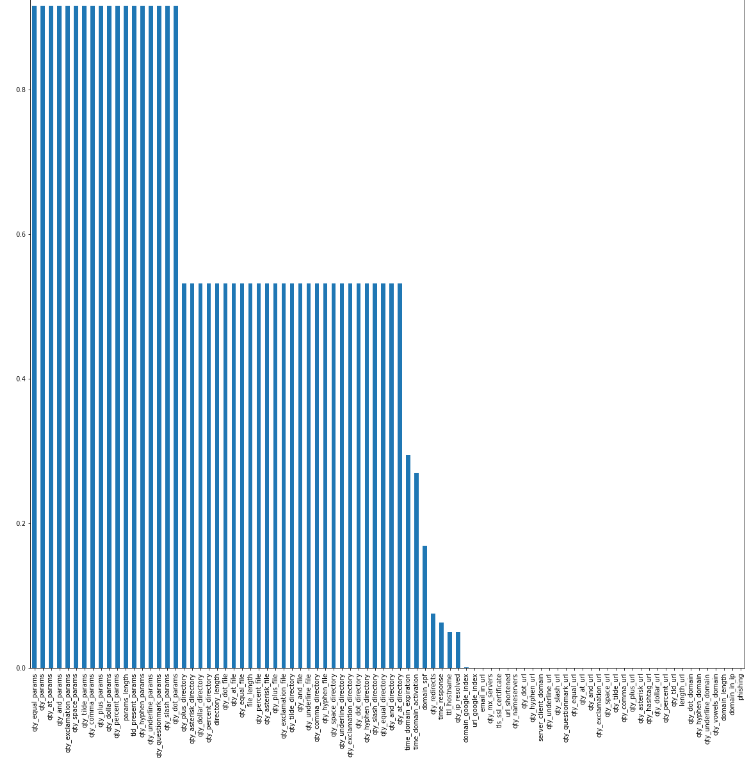


Figure 10 Percentage of missing values in the dataset

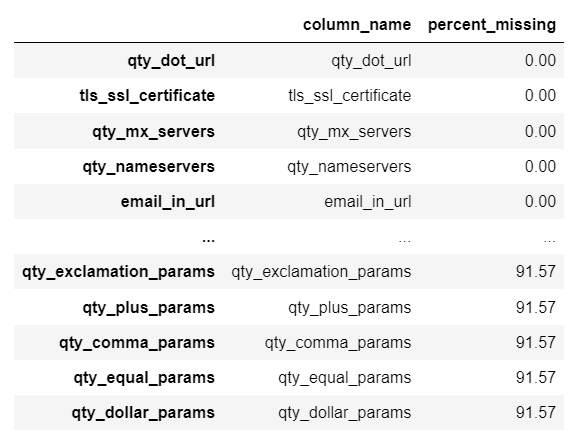


Figure 11 Percentage of missing values

In fig. 12, it can be seen that there are 3722 rows of 0 class label and 1 class label for 1503 for 1. In other words, there are 71% distribution of 0 class label and only 28% of 1 class label. This clearly means that the dataset is highly imbalanced, therefore, the it has to be balanced in data pre-processing phase.

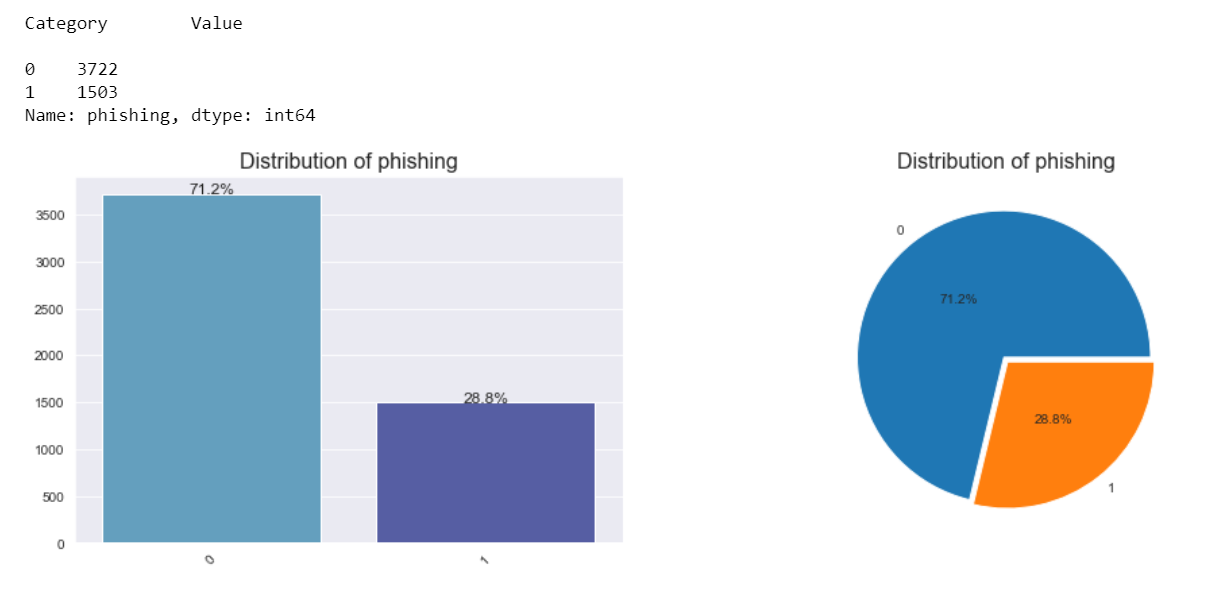


Figure 12 Phishing website class label total instances

An example of a classification problem where the distribution of examples among the recognized classes is biased or unbalanced is an imbalanced classification problem. One case in the minority class for hundreds, thousands, or millions of examples in the majority class or classes can indicate a mild bias all the way up to a serious imbalance. Predictive modeling is challenged by imbalanced classifications because the majority of machine learning methods for classification were built on the premise that there should be an equal number of samples in each class. As a result, models perform poorly in terms of prediction, particularly for the minority class. This is a problem since the minority class is often more significant and the issue is therefore more subject to classifying.

In fig. 13, the distribution of email\_in\_url & server\_client\_domain can be seen. In email\_in\_url, class label 0 has 5163 rows, and class 1 label has 62 rows. Similarly, in server\_client\_domain has 5220 rows of 0 class label, and 5 rows of 1 class label.

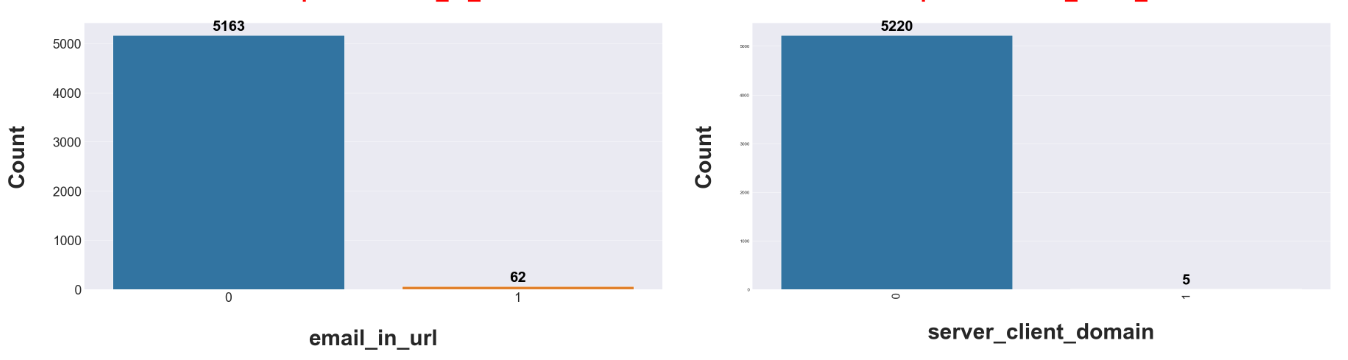


Figure 13 Distribution: email\_in\_url & server\_client\_domain

In fig. 14, distribution of url\_google\_index and domain\_google\_index can be seen where url\_google\_index has 5206 rows 0 class label and 17 for other class label. Similarly, domain\_google\_index has 5207 rows of 0 class label and 18 rows of 1 class label.

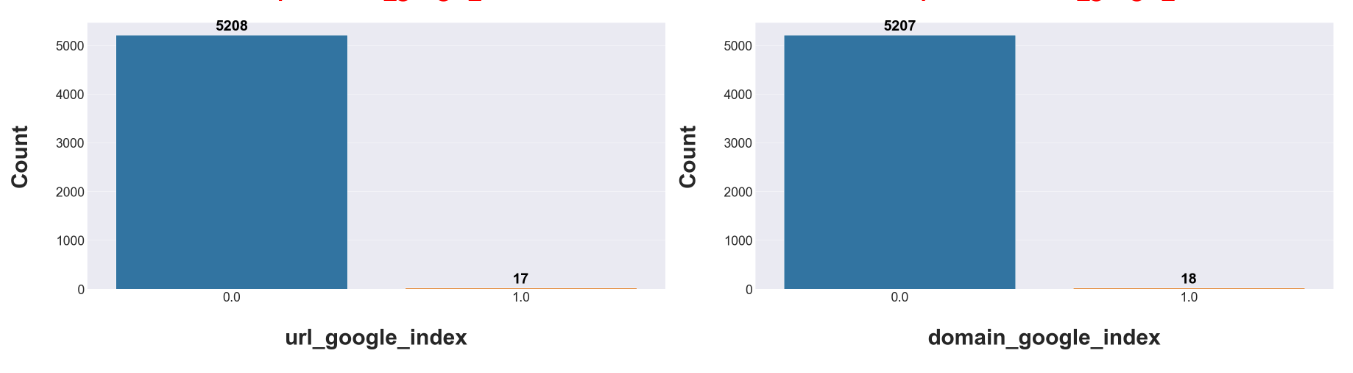


Figure 14 Distribution: url\_google\_index & domain\_google\_index

In fig. 15, the distribution of url\_shortened & qty\_tilde\_url can be seen where url\_shortened has 5197 rows of 0 class label and 28 rows of 1 class label, similarly, qty\_tilde\_url has 5222 rows of 0 class label and 3 rows of 1 class label.

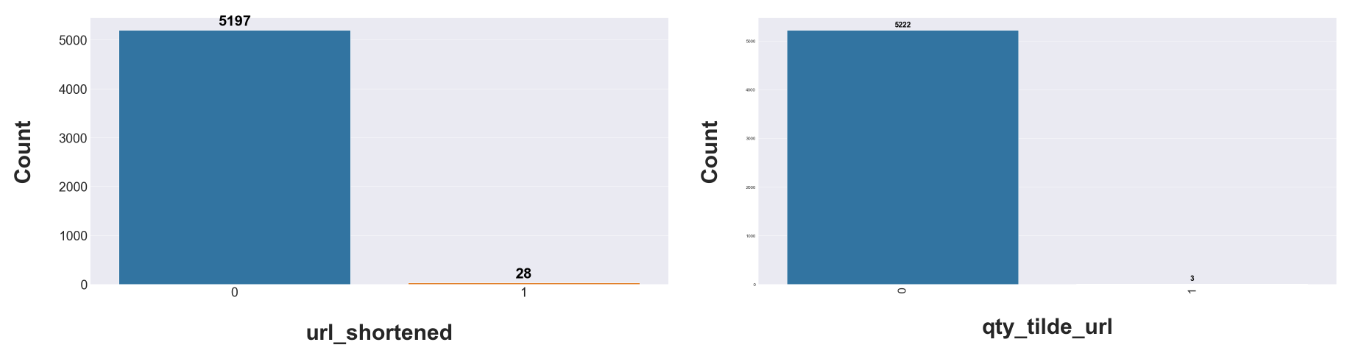


Figure 15 Distribution of url\_shortened & qty\_tilde\_url

In fig. 16, Correlation heatmap graph shows the correlation of features of the dataset. A positive correlation value that is higher means that if one value changes other feature values also change in positive direction. On the other hand, a negative correlation values means that if one value changes other feature also changes but in opposite direction. From figure, it can be seen that qty\_dot\_url has a correlation of 0.62 with qty\_and\_url, qty\_underline\_url has a correlation value of 0.33 with qty\_equal\_url. In this way, other correlation values can be seen through correlation values. We plotted the correlation plot in order to just check which variable has the highest impact to our outcome variable. The variable selection is not based from the below plot.

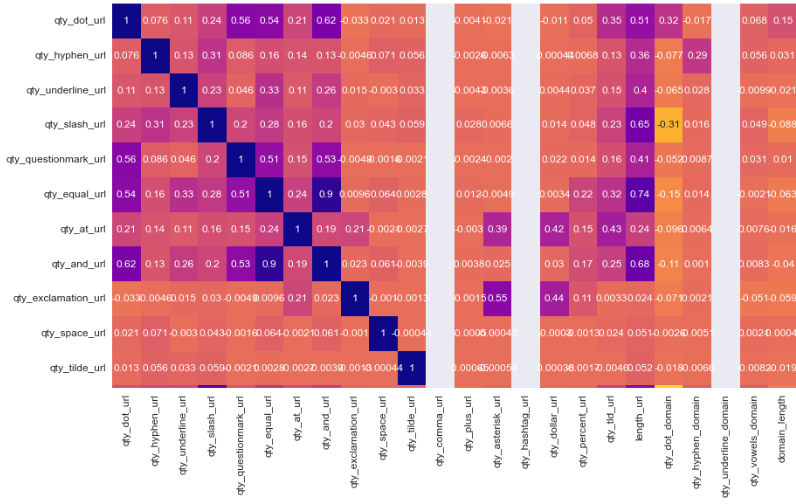


Figure 16 Correlation Heatmap Graph

## Data Pre-Processing

In fig. 17, we dropped defined the input and output variables where X variable is defined all features except ‘phishing’ feature and ‘y’ variable had defined by ‘phishing’ feature.

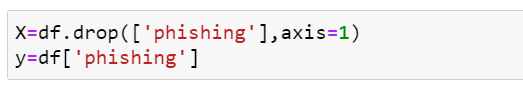


Figure 17 Defining input and output

SMOTE is a statistical method for balancing the adding of more examples to our collection. The component creates new instances from minority situations that you specify as input that already exist. The number of majority cases remains unchanged as a result of this SMOTE implementation.

The new occurrences are distinct from minority cases that already exist. The method instead selects samples from the feature space for each target class and its close neighbors. The system then creates new examples that include features from both the target case and its neighbors. With this method, each class has access to additional features, and the samples are more complete. SMOTE increases the percentage of only the minority cases after using the complete dataset as an input.

We used two types of data pre-processing techniques, Smote and Random under-sampling.

In fig. 18, oversampling and under sampling using SMOTE library can be seen where I used Pipeline with sampling\_strategy with value 0.8 for over sampling and 0.6 with under sampling. It can be seen that after performing oversampling and under sampling with 0 class label equal to 2505 rows and with 1 class label equal to 2004 rows.

Another technique we used in this project is random Under sampling. This method seeks to randomly select and remove samples from the majority class, consequently reducing the number of examples in the majority class in the transformed data. Undersampling results in a transformed data set with fewer examples in the majority class—this process that can be repeated until the number of examples in each class is equal.

Using this approach is effective in situations where the minority class has sufficient examples despite a severe imbalance. On the other hand, it is always important to consider the possible deletion of valuable information, because we accidentally delete them from our dataset because we have no way to discover or retain examples that contain rich information from the majority class.

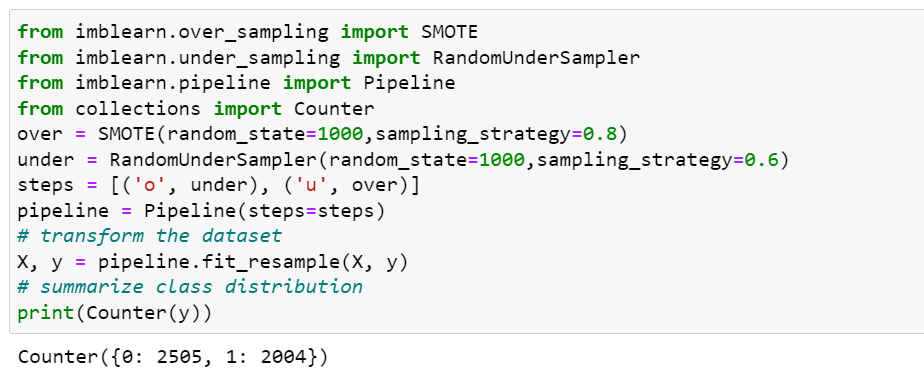


Figure 18 Random Oversampling & Under sampling using Smote Library

In fig. 19, we split the dataset into testing and training using Sklearn library with test\_size equal to 0.33, and random\_state equal to 42.

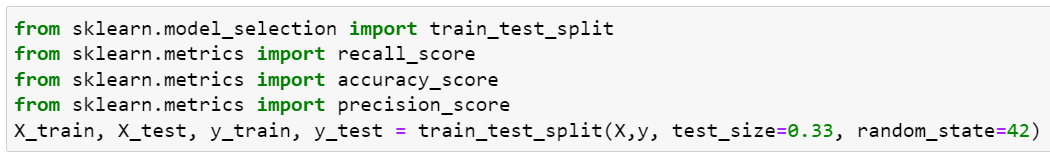


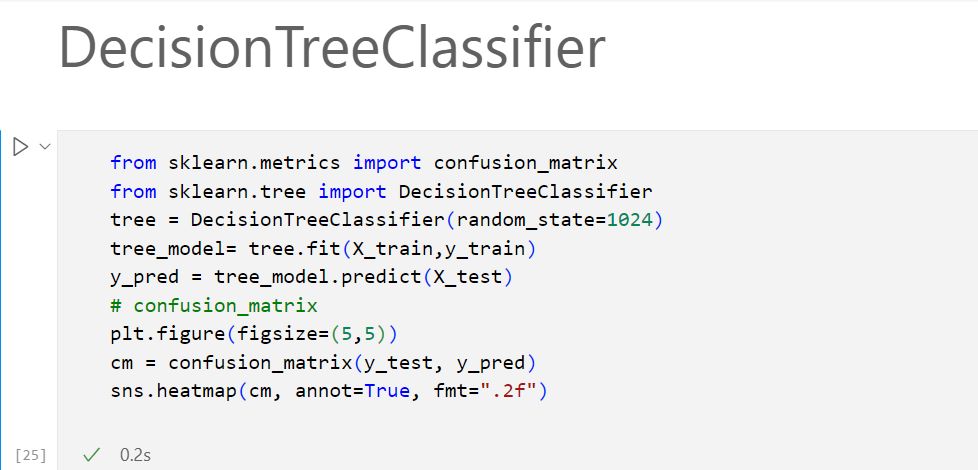
Figure 19 Splitting the dataset into testing & training

## Machine Learning Models

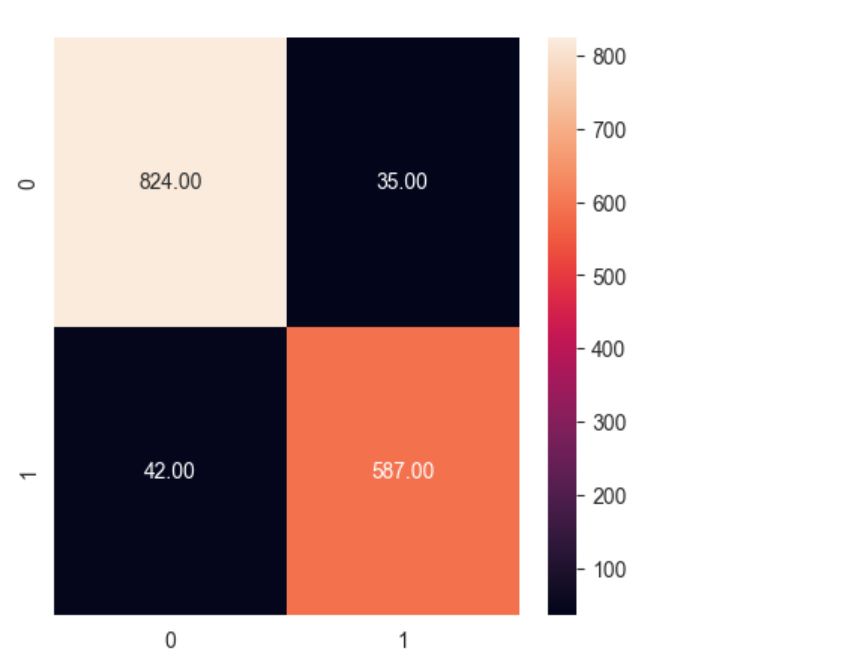
For each of the models that we performed below, we tried to model the system with default parameters and also by using hyper paramter tuning to see whether the model might performs better

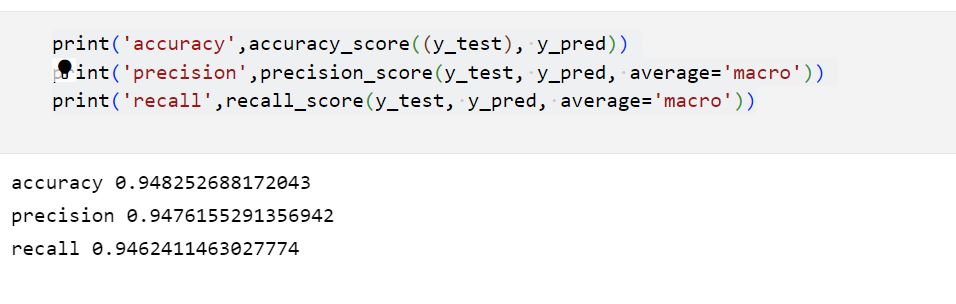
### Decision Tree

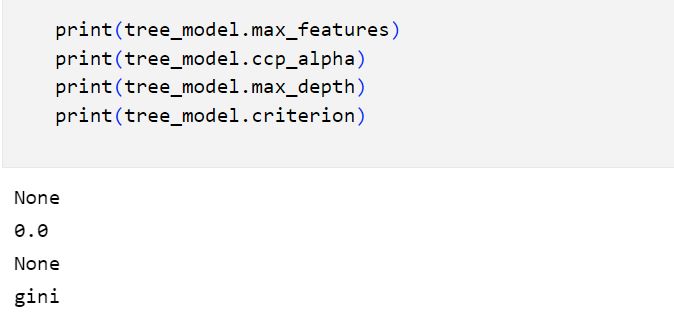
Decision Tree (DT) is a Machine learning algorithm based on supervised learning used for classification and regression problems. As the name suggests, it is based on a tree structured approach (nodes and branches) to solve the problem.



Initially, we constructed a model using decision Tree Classfier, and tried to plot the confusion matrix with the accuracy, precision and recall. As we are using hyper parameter tuning, we wanted to see what parameters are being used default and analyse.



. 



From the above results, we can see the decision tree has an accuracy of 94.8%, precision as 94.7% and recall as 94.6%. As we can see from the above the max features and max depth of the tree are none and criterion as Gini.

In fig. 20, we performed hyper-parameter tuning using GridSearchCV() library to find the optimal hyper-parameters. It provided me with parameters and values that has the highest accuracy with decision tree.

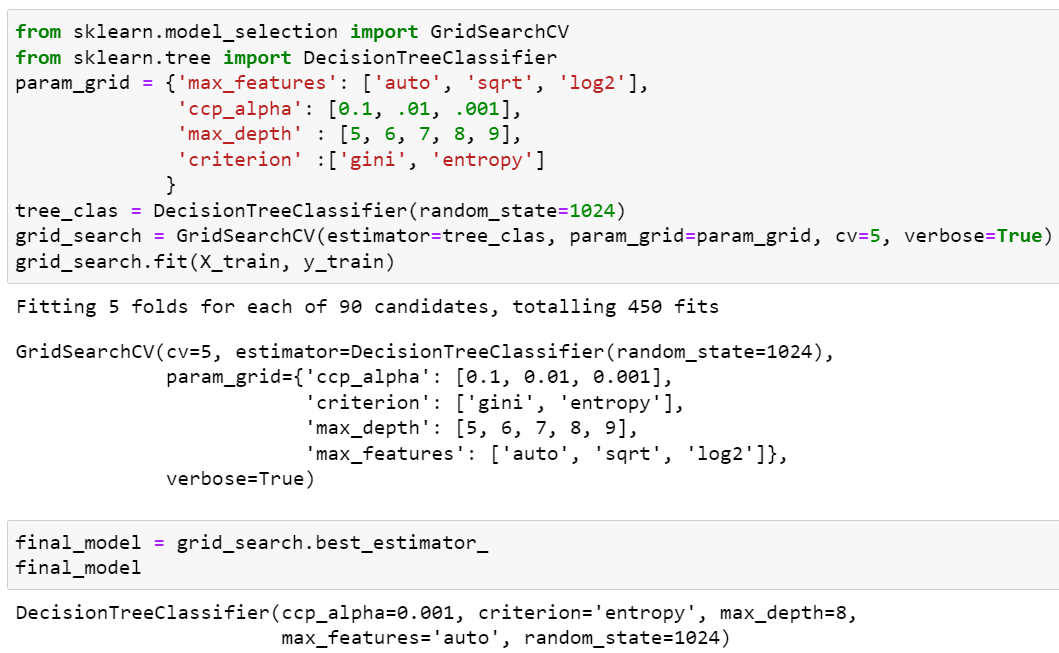


Figure 20 GridSearchCV for Decision Tree

In fig. 21, the implementation of Decision Tree can be seen where we implemented Decision Tree with ccp\_alpha=0.001, criterion=’entropy’, max\_depth=8, max\_features=’auto’, and random\_state=1024.

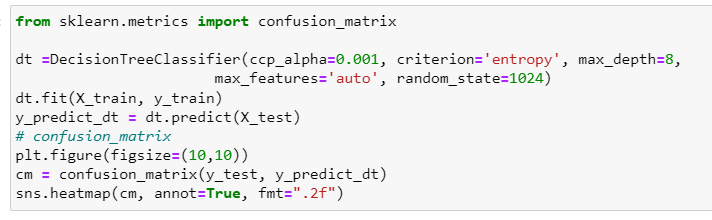


Figure 21 Decision Tree Implementation

In fig. 22, the confusion matrix displays the performance summary of the Decision Tree Model. Decision tree had a accuracy of 92%, precision of 92%, and recall was also of 92%. It predicted 0 class label as 0 797 times rightly, and 62 times wrong. Similarly, it predicted 1 class label as 1 580 times rightly, and 49 times wrong.

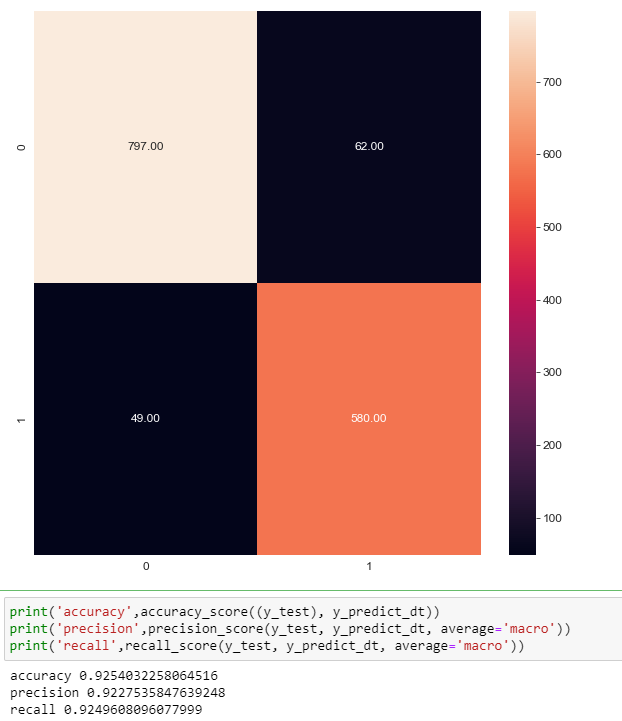
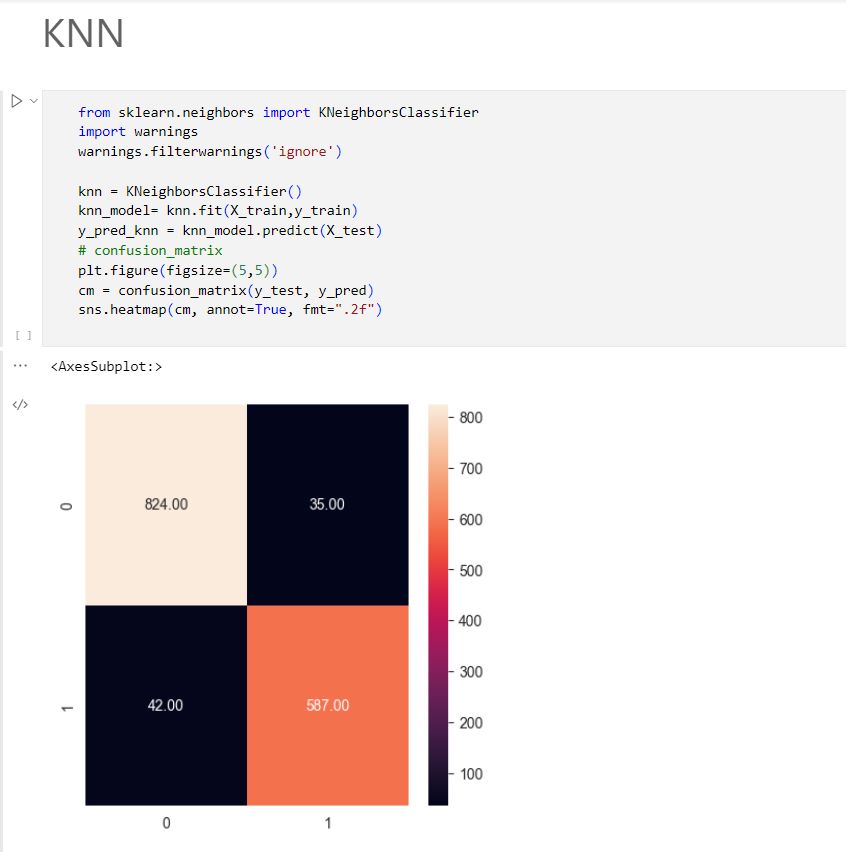


Figure 22 Confusion Matrix: Accuracy, Precision, & Recall

Taking both steps into consideration, we can see that the model without hyper tuning did a better performance than with the hyper tuning. This could be because as we didn’t restrict the depth of the tree as it can take many causalities.

### K-nearest Neighbor

K-Nearest Neighbour (KNN) is a supervised ML algorithm used for classification and regression problems. It doesn’t learn on the data, rather it assumes the similarity between the new case and available cases to categorize the data.



We constructed a model using KNN, and tried to plot the confusion matrix with the accuracy, precision and recall. As we are using hyper parameters, we wanted to see what parameters are being used default and analyse

.

From the above results, we can see the KNN has an accuracy of 82.3%, precision as 81.9% and recall as 82.4%. As we can see from the above the max nearest neighbours are 5, knn metric is minkowski and the algorithm is auto.

In fig. 23, it can be seen that we implemented K-nearest neighbor classifier for hyper-parameter tuning using GridSearchCV(). It provided with optimal hyper-parameters: algorithm='ball\_tree', metric='manhattan', n\_neighbors=10.

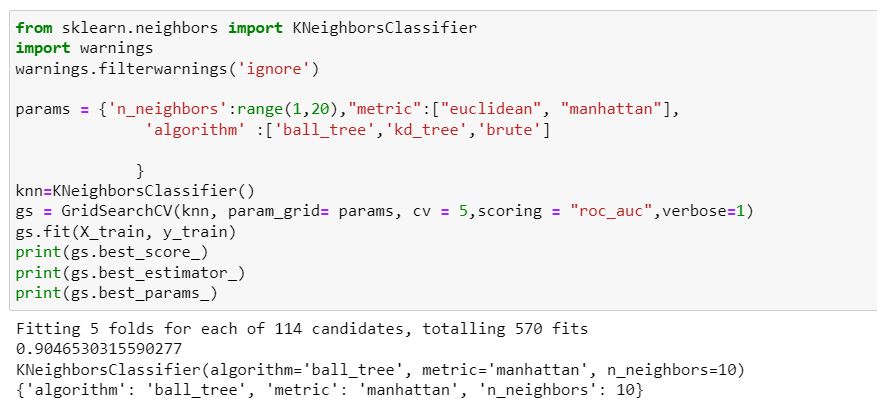


Figure 23 KNN - GridSearch CV

In fig. 24, it can be seen that we implemented K-nearest neighbour classifier with parameters provided by GridSearchCV() library.

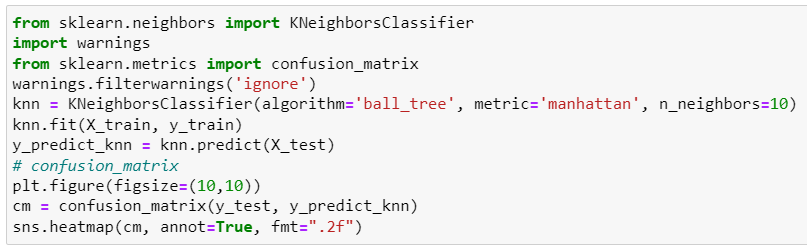


Figure 24 K-nearest Neighbour

In fig. 25, the confusion matrix graph of K-nearest neighbour can be seen that displays the summary of the results. It had an accuracy of 85%, precision of 85%, and recall of 84%. KNN predicted 769 0 class labels as 0 rightly, while 90 were predicted wrong. Similarly, it predicted 497 1 class label as 1 rightly, while 132 were predicted wrong.

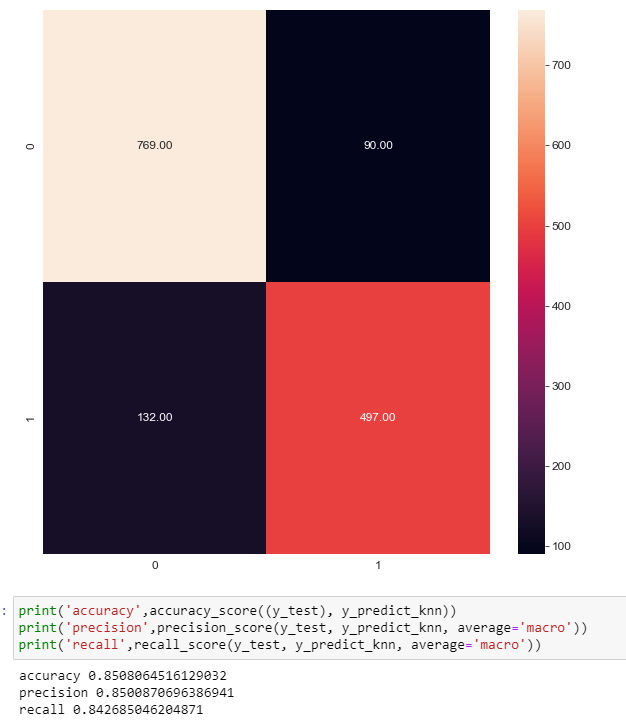


Figure 25 K-nearest Neighbour - Confusion Matrix

Comparing both steps into consideration, by tuning the Knn model it actually helps us in getting better results. This may be happened as we gave a set of knn neighbours for our model to deal with and got better results with 10 nearest neighbours

### Logistic Regression

Logistic regression is another one of the best supervised machine learning techniques in machine learning that is used for classification problems only. It gives a probabilistic value of a categorical dependent variable between 0 and 1 instead of 0 or 1 alone. It is much like similar to Linear regression except that how it is used because Linear regression is used for regression tasks while logistic regression is used for classification tasks.

We constructed a model Logistic Regression, and tried to plot the confusion matrix with the accuracy, precision and recall. As we are using hyper parameters, we wanted to see what parameters are being used default and analyse





From the above results, we can see the Logistic Regression has an accuracy of 93.1%, precision as 93.2% and recall as 92.6%. And we have the parameters as 1.0 and 12 for C and penalty respectively.

In fig. 26, it can be seen that we implemented Logistic Regression classifier for hyper-parameter tuning using GridSearchCV(). It provided with optimal hyper-parameters: C=1.0, and penalty=12.

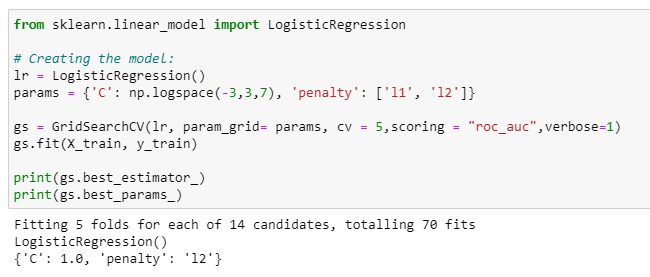


Figure 26 GridSearchCV for Logistic Regression

In fig. 27, we implemented Logistic Regression with the parameters provided by the GridSearchCV library.

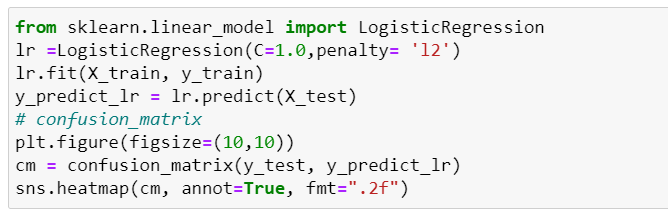


Figure 27 Logistic Regression Implementation

In fig. 28, the confusion matrix graph of Logistic Regression can be seen that displays the summary of the results. It had an accuracy of 93.1%, precision of 93.2%, and recall of 92.6%. Logistic Regression predicted 823 0 class labels as 0 rightly, while 36 were predicted wrong. Similarly, it predicted 563 1 class label as 1 rightly, while 66 were predicted wrong.

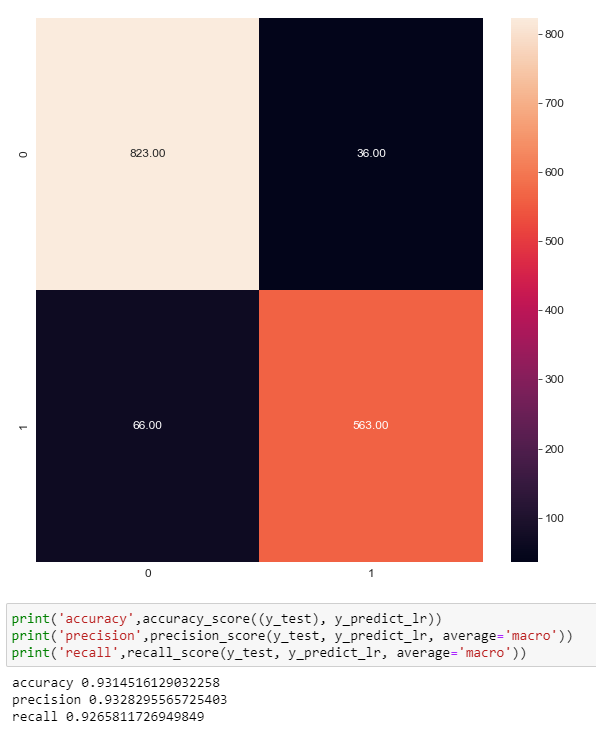


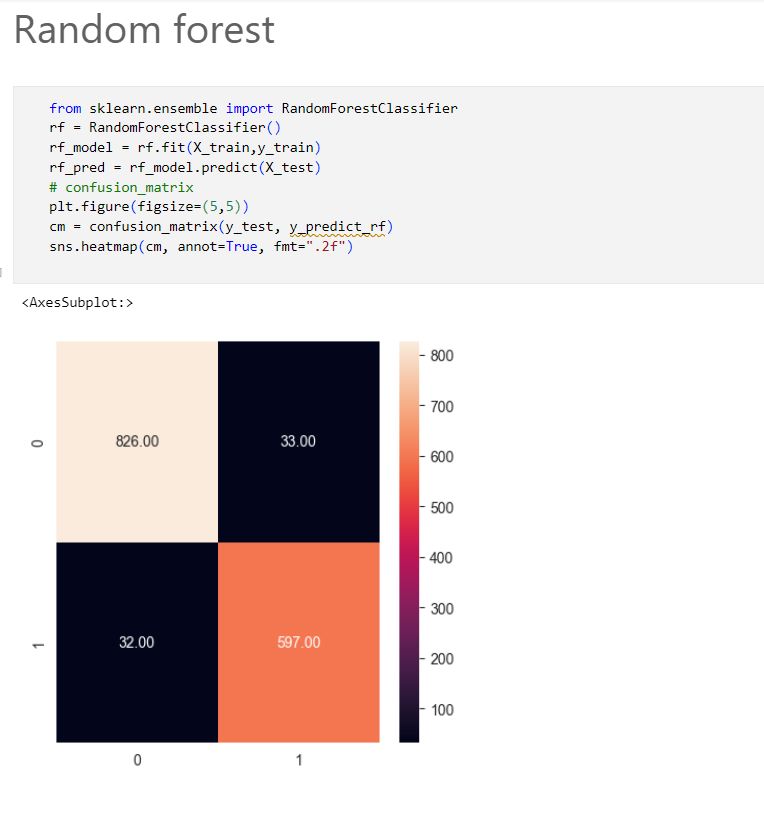
Figure 28 Confusion Matrix: Logistic Regression

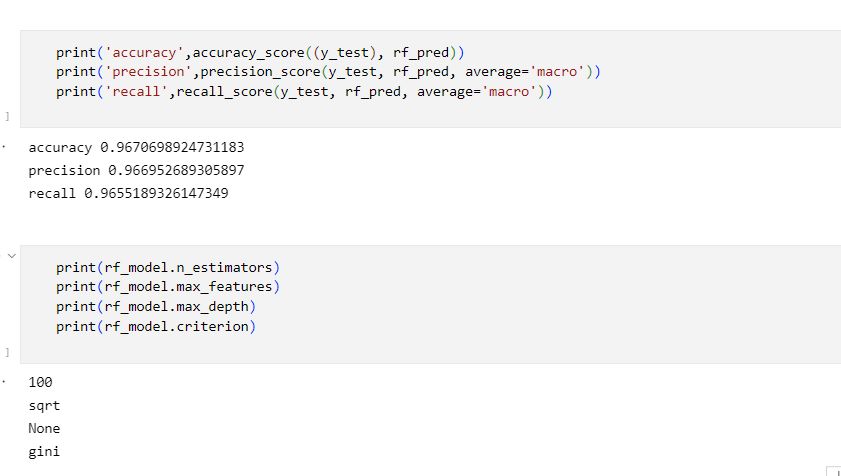
By comparing both the above steps, we can see the there is no change in the results even we tune the model with the hyper-parameter tuning.

### Random Forest

Random Forest is another one of the amazing techniques in machine learning that uses the concept of Ensemble learning. It uses multiple classifiers to compute a complex problem by taking the average to improve the classification performance. The amount of decision trees in Random Forest determines the accuracy of the model: the higher the number of trees, the higher the model.

We constructed a model Random Forest, and tried to plot the confusion matrix with the accuracy, precision and recall. As we are using hyper parameters, we wanted to see what parameters are being used default and analyse.





From the above results, we can see the Random Forest has an accuracy of 96.7%, precision as 96.6% and recall as 96.5%. And the no of estimators are 100, the max feature to find the results are sqrt, max depth is none and the model criterion is Gini.

In fig. 29, it can be seen that we implemented Random Forest classifier for hyper-parameter tuning using GridSearchCV(). It provided with optimal hyper-parameters: max\_depth=8, and n\_estimators=200.



Figure 29 GridSearchCV on Random Forest

In fig. 30, it can be seen that we implemented Random Forest through parameters provided by GridSearchCV() function.

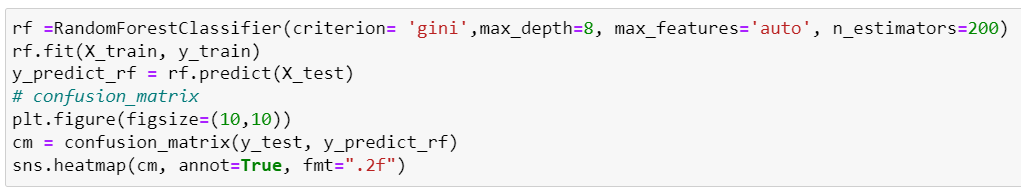


Figure 30 Random Forest Classifier

In fig. 31, the confusion matrix graph of Random Forest can be seen that displays the summary of the results. It had an accuracy of 96.0%, precision of 95.9%, and recall of 95.8%. Random Forest predicted 832 0 class labels as 0 rightly, while 27 were predicted wrong. Similarly, it predicted 597 1 class label as 1 rightly, while 32 were predicted wrong.

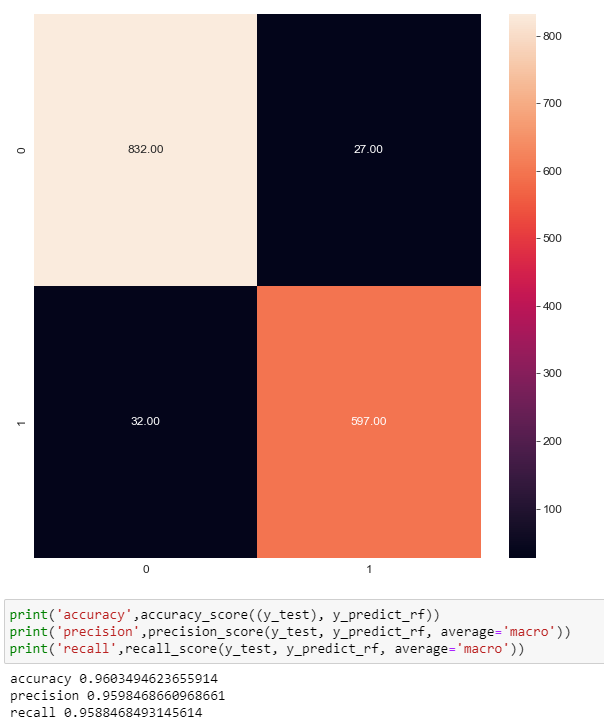


Figure 31 Confusion Matrix: Random Forest

# Results & Evaluation

we implemented four machine learning models: Decision Tree, K-nearest neighbour, Logistic Regression, and Random Forest by performing without and with hyper-parameter tuning using GridSearchCV library of Sklearn. After carefully, analysing the results of the models using performance metrics accuracy, precision, recall, confusion-matrix, and Roc curve. I came to a conclusion that Random Forest outperformed all other models. But to be precise we got better results in Random Forest without using hyper-parameter tuning with accuracy as 96.7%, precision as 96.6% and recall as 96.5%. The dotted line represent the curve for non- tuned models and the solid lines represent tuned models.

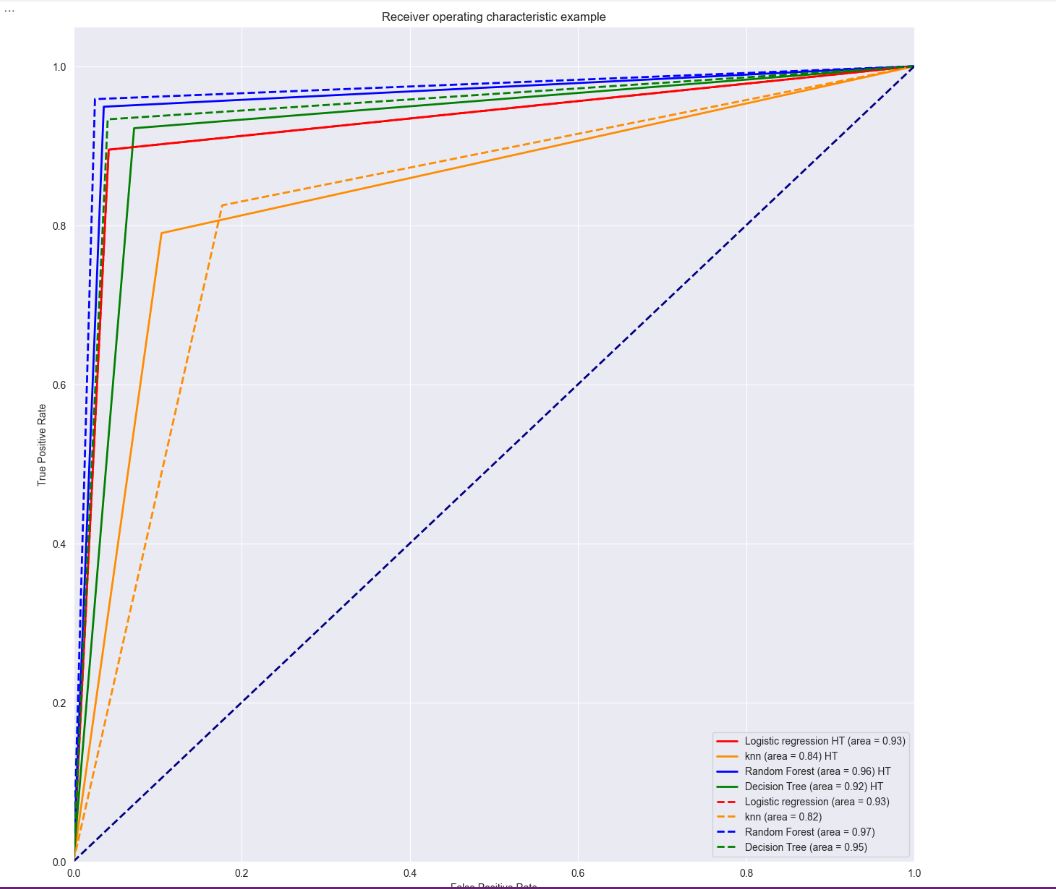


Figure 32 ROC Curve of All Classifiers

Table1 shows the performance metrics of model : Accuracy, Precision and Recall for the models. Random Forest had an accuracy of 96.7%, followed by Decision Tree at 94.8%, Logistic Regression at 93.1%, and K-nearest neighbour at 83.2%.

Table 1 Representation of Data from ML Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. no. | ML Models | Accuracy | Precision | Recall |
| 1 | Decision Tree Classifier | 94.8% | 94.7% | 94.6% |
| 2 | K-Nearest Neighbor | 82.3% | 81.9% | 82.4% |
| 3 | Logistic Regression | 93.1% | 93.2% | 92.6% |
| 4 | **Random Forest** | **96.7%** | **96.6%** | **96.5%** |

Table2 shows the performance metrics of model with tuning: Accuracy, Precision and Recall for the models. Random Forest had an accuracy of 96.0%, followed by Logistic Regression at 93.1%, Decision Tree at 92.5%, and K-nearest neighbour at 85.0%.

Table 2 Representation of Data from ML Models with tuning

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. no. | ML Models | Accuracy | Precision | Recall |
| 1 | Decision Tree Classifier | 92.5% | 92.2% | 92.4% |
| 2 | K-Nearest Neighbor | 85.0% | 85.0% | 84.2% |
| 3 | Logistic Regression | 93.1% | 93.2% | 92.6% |
| 4 | **Random Forest** | **96.0%** | **95.9%** | **95.8%** |

# Conclusion

We studied at a dataset of phishing website detection in this project. There were a ton of null values and missing values in the dataset. Data wrangling was used to clean the dataset and prepare it for data analysis by removing the missing values. In this project, we learnt how to perform multivariate and univariate analysis using a variety of libraries, including Seaborn, Plotly, and Matplotlib. After that, we pre-processed the dataset by defining input and output variables, using Smote to over- and under-sample, and dividing it into training and testing for modeling. There are two main categories to our project. For the first category, we used default parameters while building our models. We plotted confusion matrix to analyse the performance and calculated performance metrics. We decided to take things a step further and employed hyper-parameter tweaking utilizing the GridSearchCV() function to find the best hyper-parameters to select the best features for our models because some of the results were considerably lower than we had anticipated. To identify phishing websites, we developed Decision Trees, Logistic Regression, K-nearest Neighbors, and Random Forest models. In my research, we discovered that the parameter will play a significant role and affect the model's performance. We come to the conclusion that Random Forest without tuning, with an accuracy of 96.7%, outperformed all others. So, for this investigation, I decided that Random Forest was the best model.

# References

*Phishing Website Detection*. (n.d.) https://www.sciencedirect.com/science/article/pii/S2352340920313202

Brownlee, J. (2020, January 16). *SMOTE for Imbalanced Classification with Python*. Machine Learning Mastery. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

‌Altair, 2022. What is Data Wrangling. Altair. Available at: <https://www.altair.com/what-is-data-wrangling>

Afroz S., GreenStatd, R., 2021. Phishzoo: detecting phishing websites by looking at them. Available at: <https://ieeexplore.ieee.org/document/6061361/>

Javatpoint Team. 2022. Random Forest Algorithm, javatpoint, Available at: <https://www.javatpoint.com/machine-learning-random-forest-algorithm>

Javatpoint Team. 2022. Logistic Regression in Machine Learning. Javatpoint, Available at: <https://www.javatpoint.com/logistic-regression-in-machine-learning>

Javatpoint Team. 2022. K-Nearest Neighbor (KNN) Algorithm for Machine Learning. Javatpoint, Available at: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

Javatpoint Team. 2022. Decision Tree Classification Algorithm. Javatpoint, Available at: <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>

Kumar, A., 2022. Correlation Concepts, Matrix & Heatmap using Seaborn. Vitalflux, Available at: <https://vitalflux.com/correlationheatmap-with-seaborn-pandas/>

Gad, A.F, 2022. Evaluating Deep Learning Models: The Confusion Matrix, Accuracy, Precision, and Recall. Paperspace Blog. Available at: <https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/>

Google. 2022. “Classification: ROC Curve and AUC”, developers.google.com, Available at: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc#:~:text=An%20ROC%20curve%20(receiver%20operating,False%20Positive%20Rate>

Zubair, M.D, 2021. Find the Patterns of a Dataset by Visualizing Frequency Distribution. Towards Data Science. Available at: <https://towardsdatascience.com/find-the-patterns-of-a-dataset-by-visualizing-frequency-distribution-c5718ab1f2c2>

APWG, 2022. Phishing Hits All-Time High in December 2021; Attacks Triple Since Early 2020. Globalnewswire. Available at: https://www.globenewswire.com/en/news-release/2022/02/25/2392052/0/en/APWG-Q4-Report-Phishing-Hits-All-Time-High-in-December-2021-Attacks-Triple-Since-Early-2020.html#:~:text=25%2C%202022%20(GLOBE%20NEWSWIRE),has%20tripled%20from%20early%202020.