

CSSE\_CT046-3-M-AML-L-1\_2024-10-10

**Applied Machine Learning**

**Phishing Website Detection**

Student Name: Tabeed Hameed Ahangar

TP Number: TP083006

Intake Code: APUMF2409DSBA(DE)

Programme: Data Science & Business Analytics

Module Lecturer: ASSOC. PROF. DR. Nirase Fatima Abubacker

# Abstract

Cybersecurity experts face a key challenge in detecting phishing websites due to their targeted attacks towards users. The project aims to investigate a wide range of machine learning tools that boost phishing website identification through models Naïve Bayes, Decision Tree, Logistic Regression, SVM, Random Forest, and K-Nearest Neighbours. The project employs a strategic approach which begins with data transformation followed by feature identification before putting models into practice until reaching a final evaluation stage through different validation methods. The results show how ensemble learning boosts classification accuracy performance among various tested models. The model performance gets optimized through the implementation of grid search along with stratified k-fold cross-validation approaches. The research evaluates different classification algorithms for phishing website detection thus helping to design more secure cybersecurity systems.

**Table of Contents**

[Abstract 2](#_Toc190460342)

[Introduction, Aim & Objectives 5](#_Toc190460343)

[1.1 Aim 6](#_Toc190460344)

[1.2 Objectives 6](#_Toc190460345)

[Related Works 7](#_Toc190460346)

[Dataset 9](#_Toc190460347)

[3.1 Overview 9](#_Toc190460348)

[3.2 Features 9](#_Toc190460349)

[3.2.1 URL-Based Features 10](#_Toc190460350)

[3.2.2 Abnormal-based Features 11](#_Toc190460351)

[3.2.3 HTML & JavaScript-based Features 11](#_Toc190460352)

[3.2.4 Domain-based Features 12](#_Toc190460353)

[3.3 Exploratory Data Analysis 12](#_Toc190460354)

[3.3.1 Distribution of Independent Features 13](#_Toc190460355)

[3.3.2 Distribution of Target Variable 17](#_Toc190460356)

[3.3.3 Correlation Heatmap 18](#_Toc190460357)

[3.3.4 Chi-square Tests 19](#_Toc190460358)

[Models 21](#_Toc190460359)

[4.1 Overview 21](#_Toc190460360)

[4.2 Implementation 24](#_Toc190460361)

[4.2.1 Gaussian Naïve Bayes (GNB) 24](#_Toc190460362)

[4.2.2 Decision Tree (DT) 25](#_Toc190460363)

[4.2.3 Logistic Regression (LR) 26](#_Toc190460364)

[4.2.4 Support Vector Machine (SVM) 27](#_Toc190460365)

[4.2.5 Random Forest (RF) 28](#_Toc190460366)

[4.2.6 K-Nearest Neighbours (KNN) 29](#_Toc190460367)

[Model Validation & Fine-Tuning 32](#_Toc190460368)

[5.1 Model Validation Techniques 32](#_Toc190460369)

[5.1.1 K-Fold Cross Validation 32](#_Toc190460370)

[5.1.2 Stratified K-Fold Cross Validation 32](#_Toc190460371)

[5.1.3 Nested Cross Validation 33](#_Toc190460372)

[5.2 Hyperparameter Tuning 33](#_Toc190460373)

[5.2.1 Grid Search 33](#_Toc190460374)

[5.2.2 Random Search 33](#_Toc190460375)

[Analysis & Recommendations 36](#_Toc190460376)

[References 42](#_Toc190460377)

Chapter 01

# Introduction, Aim & Objectives

Phishing attacks have grown into a major cybersecurity threat which continues to target individuals along with organisations. Fake websites made to impersonate trusted entities deceive users into giving their sensitive information like login information or payment information. The Anti-Phishing Working Group (APWG) recorded a record 1,270,883 phishing attacks in the third quarter of 2022, the worst quarter seen to date(Kapan & Sora Gunal, 2023). Traditional rule-based protection mechanisms have been inadequate to keep up with the increasing creation of new phishing methods. The techniques are not able to detect new phishing websites, as malicious actors constantly evolve to tackle security protocols (Kapan & Sora Gunal, 2023).

The ability of Machine Learning (ML) techniques to improve automatic detection of phishing websites has made them the preferred approach in response to these detection challenges. Machine Learning can examine massive datasets in order to discover patterns that separate phishing websites from real ones, and hence increase accuracy in detection (Mandadi et al., 2022). Machine learning-based detection methods surpass traditional methods because they do not need continuous human updates to learn and detect unidentified phishing websites and can generalise well to detect previously unseen phishing websites based on learned features (Mandadi et al., 2022). The detection of phishing websites becomes difficult due to the class imbalance problem in which phishing sites appear fewer than real sites. The class imbalance negatively affects machine learning models by adding bias towards the majority class, thus minimising detection accuracy. Methods like Synthetic Minority Over-sampling Technique (SMOTE) and other resampling techniques have been used to overcome this problem. These methods produce synthetic phishing examples and modify distribution ratios between classes to improve detection models’ performance (Maci et al., 2023).

Phishers are constantly updating their strategies with techniques that involve URL hiding techniques, domain masking and cloaking, which challenge basic classifiers for phishing site discovery. To counter these challenges, researchers have used techniques like ensemble learning to merge several models together. This method utilises multiple algorithms to produce effective anti-phishing systems to counter modern phishing websites (Innab et al., 2024). These advances in phishing detection techniques show the ongoing efforts to combat cyber threats by using machine learning, deep learning and hybrid methodologies. While significant progress has been made, additional enhancements to detection systems is essential since new phishing techniques are being developed and models need better flexibility in practical phishing situations.

## 1.1 Aim

This project aims to train and evaluate various machine learning models, evaluating their effectiveness and improving their accuracy using various validation methods.

## 1.2 Objectives

This project adopts the following targets to accomplish the main aim.

* To gather dataset of phishing and legitimate URLs, handle missing values and feature selection to get the dataset ready for model training.
* To perform Exploratory Data Analysis (EDA) to learn about the most important indicators of phishing websites.
* To train several Machine Learning algorithms and evaluate which are best for detecting phishing websites.
* To use validation methods to optimize model parameters and enhance performance.
* To evaluate models on metrics such as accuracy, precision, recall, F1-score to determine the best method.
* To suggest the implementation of optimal phishing detection strategies as well as investigate prospective research enhancement possibilities.

Chapter 02

# Related Works

Phishing attacks have grown in volume and severity and have become great threats to entities and organizations. As a result, phishing site detection has come to be considered a major subject of research within the cybersecurity study area. Extensive research has been conducted into the application of machine learning and deep learning algorithms in enhancing phishing detection systems in terms of their accuracy and speed (Kapan & Sora Gunal, 2023).

Machine learning has become a vital tool for detecting phishing activity in the last few years. (Patil et al., 2022) their literature demonstrated how Support Vector Machines (SVM), Decision Trees, and Random Forests can be stacked together to create a detection system that is adaptive and evolves to be the future of cyber security. Another group of researchers (Aldaham et al., 2024) analysed machine learning detection of phishing websites by applying Decision Tree along with SVM and Artificial Neural Network (ANN) and the Random Forest model. The study demonstrated that Decision Trees delivered the most accurate results at 96.7% before Random Forest provided 95.75%. The detection of phishing websites using Machine Learning techniques became the preferred solution for dealing with these detection challenges. Through examining extensive data sets Machine Learning searches for distinguishing patterns that differentiate between phishing websites and genuine ones which enhances the detection accuracy (Mandadi et al., 2022). Machine learning detection methods produce superior results than traditional methods because they learn unknown phishing websites and retain learned features which help detect new phishing websites (Mandadi et al., 2022).

To improve the performance of the detector many researchers have used the Ensemble Learning method to combine two or more models in order to produce better predictions (*What Is Ensemble Learning?*, 2024). Multiple methods based on ensemble designs have proven effective for boosting phishing detection precision. The research conducted by (U S et al., 2024) developed an ensemble system integrating ANN and K-Nearest Neighbours and Decision Tree algorithms which used Random Forest as the primary classifier. The stacking ensemble classifier developed by (Newaz et al., 2024) incorporated feature selection with greedy algorithms and deep learning methods along with cross-validation for its design. Their model produced superior results than current phishing detection models because it achieved detection accuracies of over 97% within different datasets.

One of the major problems faced when training models for phishing website detection is the number of features in play. Phishing datasets typically contain numerous features based on URL information, domain features, and HTML structure elements. Features are not equal for detection purposes; some tend to cause ambiguity and decrease the accuracy of the model. To address this issue, researchers have employed methods such as Principal Component Analysis (PCA) and feature selection algorithms in a bid to enhance model performance. By selecting only the required features, these techniques can enhance phishing detection models (Lin & Chen, 2013). Another method proposed by (Çolhak et al., 2024) focusing on HTML content analysis which used a Multi-layer Perceptron (MLP) and Natural Language Processing (NLP) models. This method showed the efficiency of combining deep learning with NLP for the detection of phishing websites based on the content of the webpage and not only the URL features.

Among the numerous features available it becomes very important to filter out those that do not significantly aid in model training. This emphasises the role of feature selection and data preprocessing for training the detection models. The input data we employ to train our machine learning model plays a very significant role in the performance of the model. Dimensionality increase of data can result in enormous problems in supervised and unsupervised learning processes. Training your model on redundant features decreases the overall ability of the model and even decreases the model's accuracy. Moreover, increasing variables in a model increases the overall complexity of the model. Conducting feature selection has various advantages, it decreases overfitting, enhances accuracy, and decreases training time (Hamdard & Lodin, 2023). Another issue is imbalanced class, (U S et al., 2024) demonstrated the importance of handling imbalanced datasets and selecting relevant features. Class imbalance adds skewness and bias towards the majority and thus class balancing become an important process to improve the models (Fairstein et al., 2024).

This project aims to establish a significant emphasis on the use of machine learning methods for the detection of phishing websites

Chapter 03

# Dataset

## 3.1 Overview

The dataset is collected from the UCI Machine Learning Repository which targets phishing website identification. The features from website URLs along with their metadata enable users to distinguish phishing websites from real ones. The dataset is pretty much pre-cleaned and model training phase can be started with minimum steps.

Each record is a website instance and is defined by a broad spectrum of features. These features range across different categories like URL pattern, HTML/JavaScript behaviour and domain based. The target variable is trinary and uses:

* **-1:** *Phishing website*
* **0:** *Suspicious website*
* **1:** *Legitimate website*

## 3.2 Features

A list of all the features in the dataset is given in *Table 1:*

Table 1: Summary of Various Features Present in the Dataset

|  |  |  |
| --- | --- | --- |
| ***Feature*** | ***Non-Null Count*** | ***Dtype*** |
| having\_IP\_Address | 11055 | int64 |
| URL\_length | 11055 | int64 |
| Shortening\_Service | 11055 | int64 |
| having\_At\_Symbol | 11055 | int64 |
| double\_slash\_redirecting | 11055 | int64 |
| Prefix\_Suffix | 11055 | int64 |
| having\_Sub\_Domain | 11055 | int64 |
| SSLfinal\_State | 11055 | int64 |
| Domain\_registration\_length | 11055 | int64 |
| Favicon | 11055 | int64 |
| port | 11055 | int64 |
| HTTPS\_token | 11055 | int64 |
| Request\_URL | 11055 | int64 |
| URL\_of\_Anchor | 11055 | int64 |
| Links\_in\_tags | 11055 | int64 |
| SFH | 11055 | int64 |
| Submitting\_to\_email | 11055 | int64 |
| Abnormal\_URL | 11055 | int64 |
| Redirect | 11055 | int64 |
| on\_mouseover | 11055 | int64 |
| RightClick | 11055 | int64 |
| popUpWindow | 11055 | int64 |
| Iframe | 11055 | int64 |
| age\_of\_domain | 11055 | int64 |
| DNSRecord | 11055 | int64 |
| web\_traffic | 11055 | int64 |
| Page\_Rank | 11055 | int64 |
| Google\_Index | 11055 | int64 |
| Links\_pointing\_to\_page | 11055 | int64 |
| Statistical\_report | 11055 | int64 |
| Result | 11055 | int64 |

### 3.2.1 URL-Based Features

* ***having\_IP\_Address:*** These features determine whether the URL uses an Internet Protocol (IP) address instead of using the standard domain naming system. Phishing sites employ IP addresses to replace domain names for the purpose of avoiding detection methods.
* ***URL\_Length:*** Measures the total number of characters in the URL. Attackers mask malicious features in phish URLs through the addition of extra characters which extends the URL length.
* ***Shortening\_Service:*** URL shortening operation through *‘Shortening\_Service’* examines if shorter URLs such as those created with TinyURL service exist because this behavior makes destinations hard to detect and increases phishing risks.
* ***having\_At\_Symbol:*** This indicator reveals the existence of an “@” symbol in the URL because attackers commonly exploit this symbol to ensure browser-based URL ignoring thereby hiding the phishing content.
* ***double\_slash\_redirecting:*** Evaluates the position of double slashes (“//”) within the URL. Downloading a URL should have its protocol markers positioned at the front but abnormal placement signals possible redirection or encoding activities.
* ***Prefix\_Suffic:*** Detects the presence of hyphens in the domain name. Domains that operate legitimately do not contain hyphens so their appearance indicates a suspicious situation.
* ***having\_Sub\_Domain:*** Monitors subdomain count; an abnormal number or irregular subdomain configuration can fake trusted domains with hidden actual domains.

### 3.2.2 Abnormal-based Features

* ***Request\_URL:*** This analytic checks if any part of the webpage loads from a domain different than the main website since this indicates possible phishing activity.
* ***URL\_of\_Anchor:*** Scans all anchor tags to check if their destination URLs lead to inconsistent or suspicious locations.
* ***Links\_in\_tags:*** Examines links within meta, script, and link tags to ensure consistency in resource domains; anomalies here may signal phishing.

### 3.2.3 HTML & JavaScript-based Features

* ***SFH (Server Form Handler):*** The Server Form Handler derives its name from its ability to analyze the action attribute found within form tags. Any form handler without content or that points to an external address may indicate a phishing scam.
* ***Submitting\_to\_email:*** Checks whether form data is submitted via “mailto:” rather than a secure server-side process—a behavior rarely seen in legitimate websites.
* ***Abnormal\_URL:*** Uses external validation (e.g., WHOIS data) to verify whether the URL conforms to expected norms.
* ***Redirect:*** Redirect checks the number of redirections that phishing sites usually employ to hide their ultimate destination.
* ***on\_mouseover:*** Detects JavaScript events that change the status bar text upon hovering, potentially misleading users about the actual URL.
* ***RightClick:*** The option to right-click is either disabled or enabled through RightClick to prevent users from accessing source code contents.
* ***popUpWidnow:*** Security systems use popUpWindow to detect the misuse of temporary pop-ups that phishers use to ask for information through pop-up dialog boxes.
* ***IFrame:*** Checks for the presence of hidden iframes used to load external content without the user's knowledge.

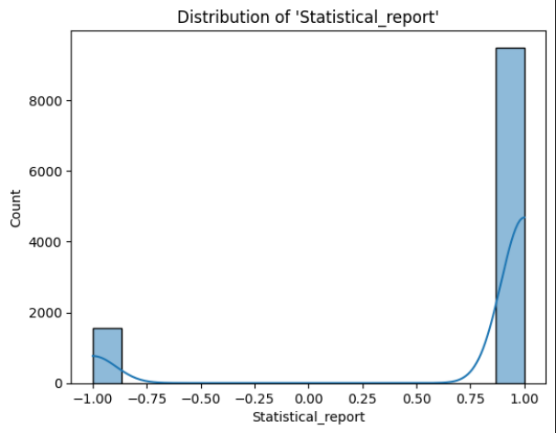
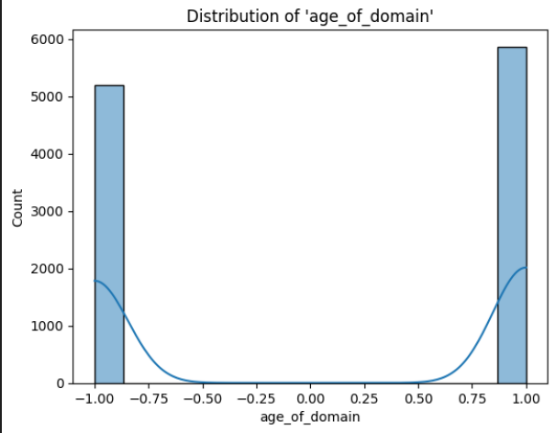
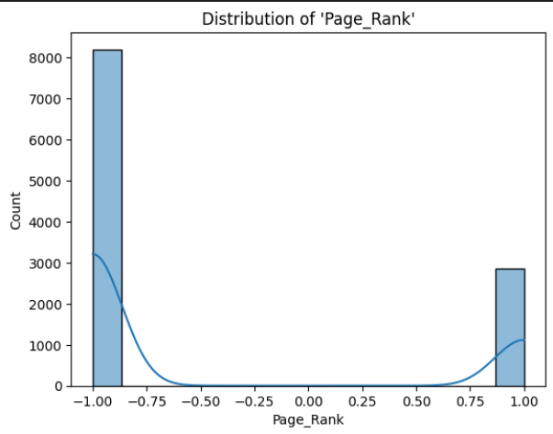
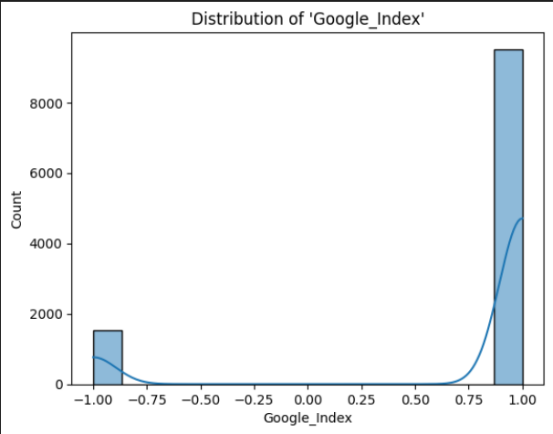
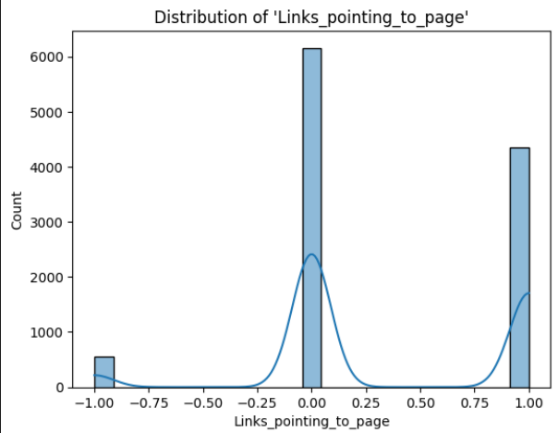
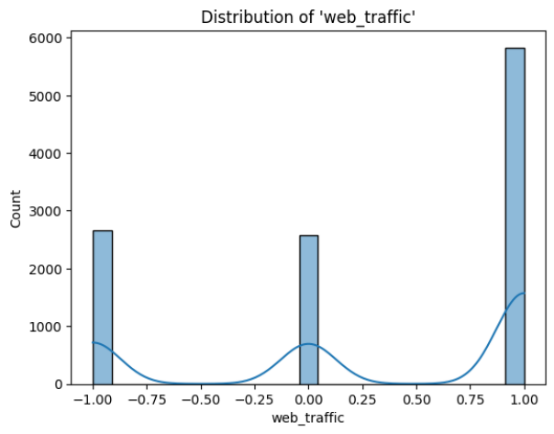
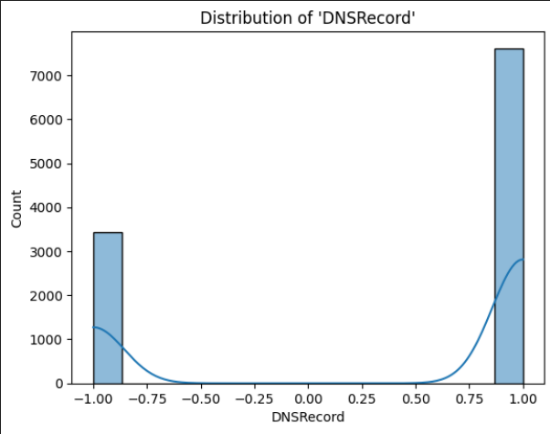
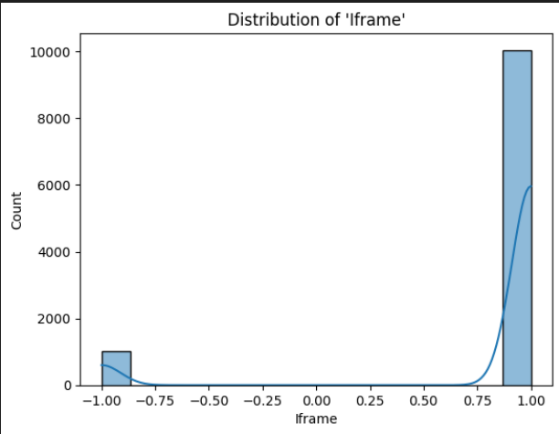
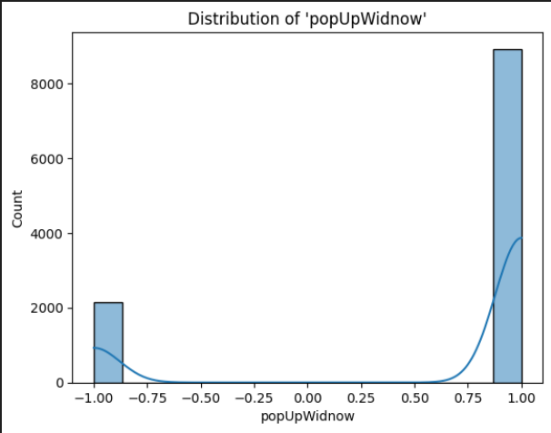
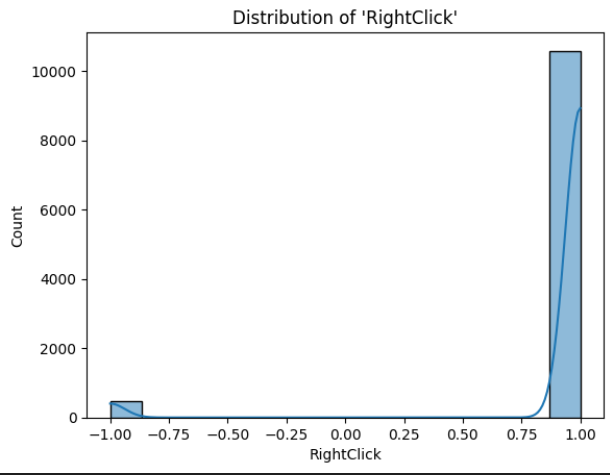
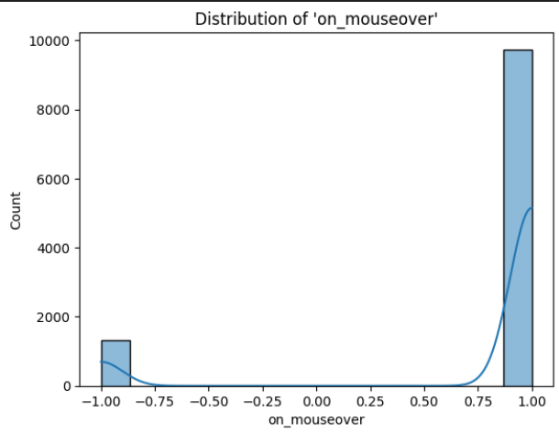
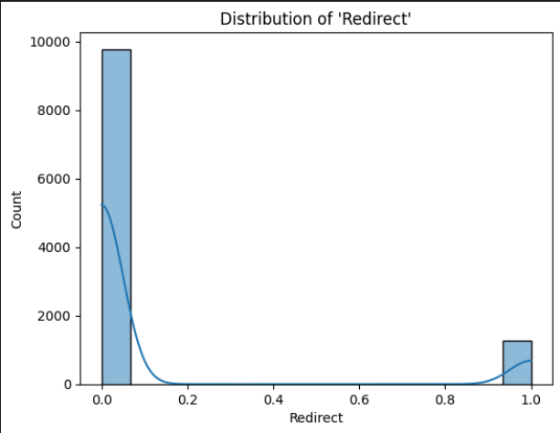
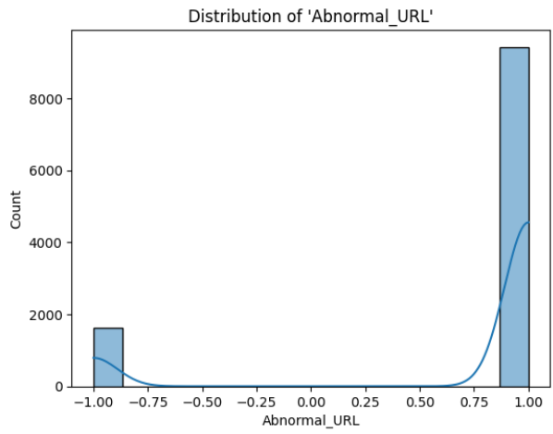
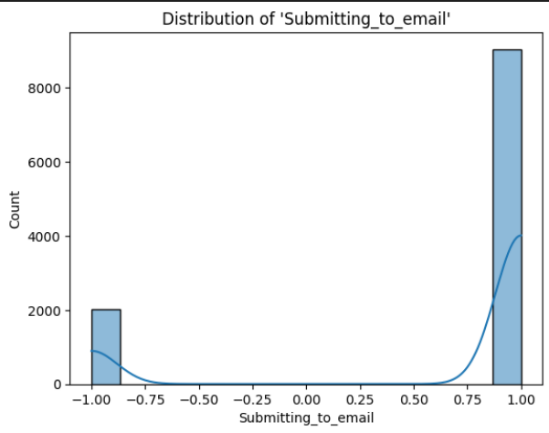
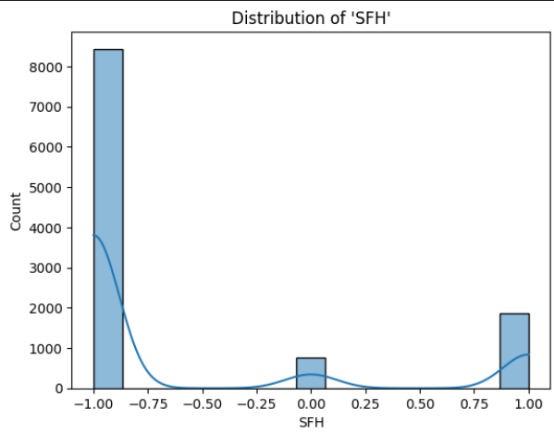
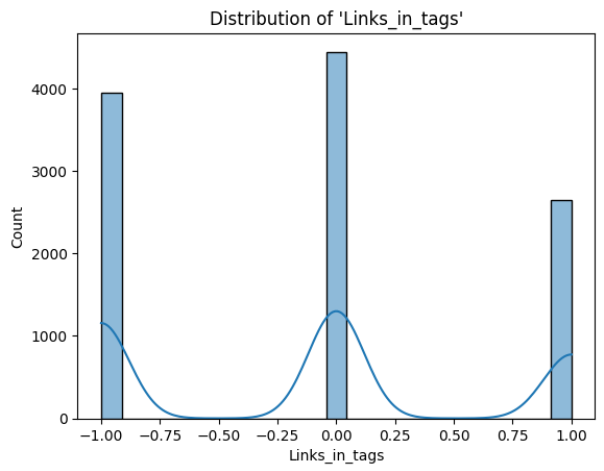
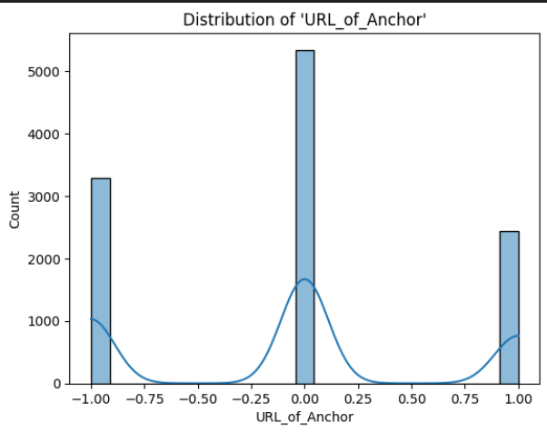
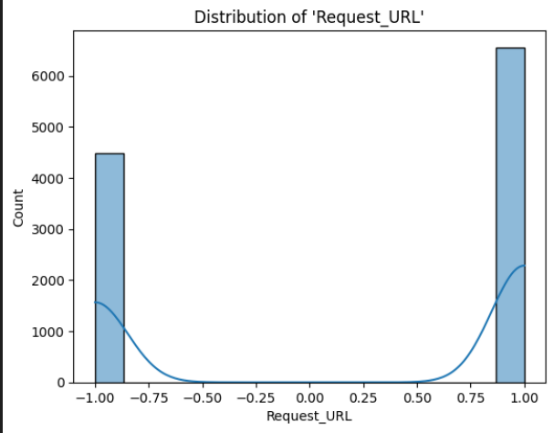
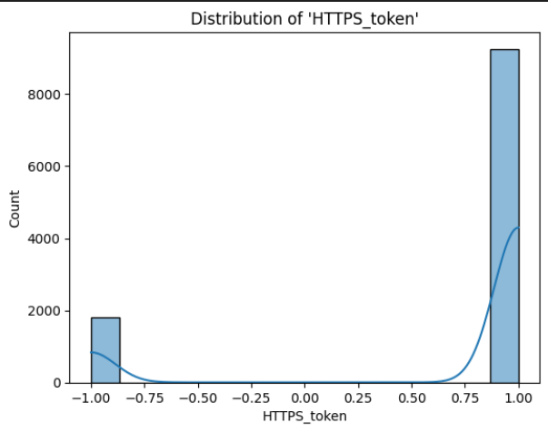
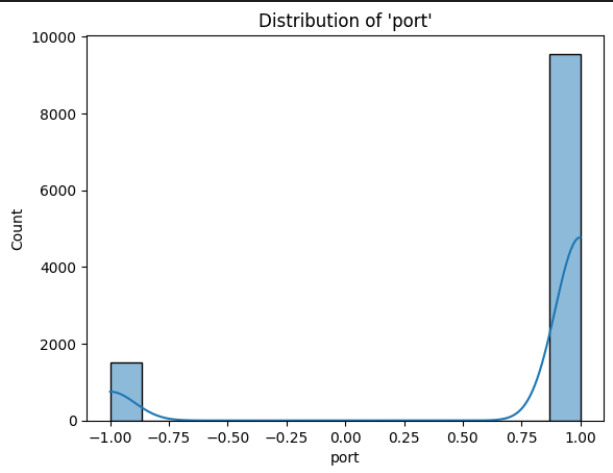
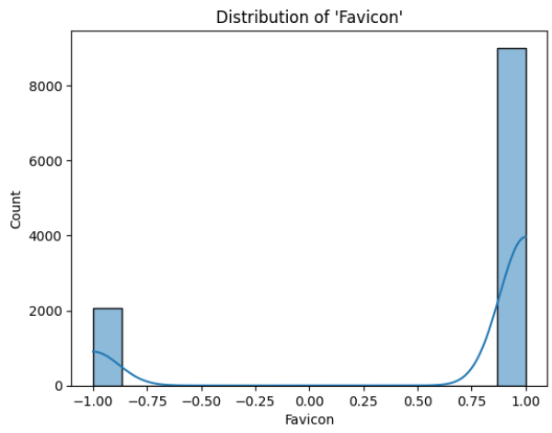
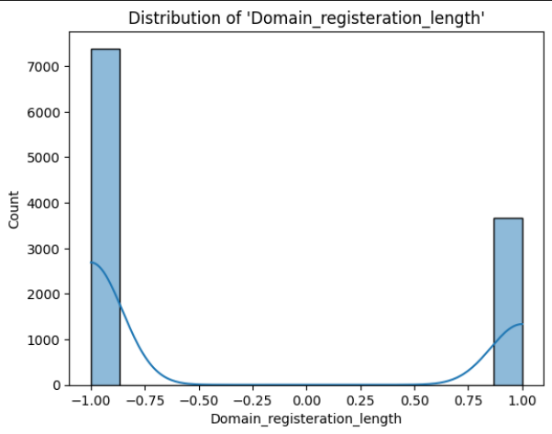
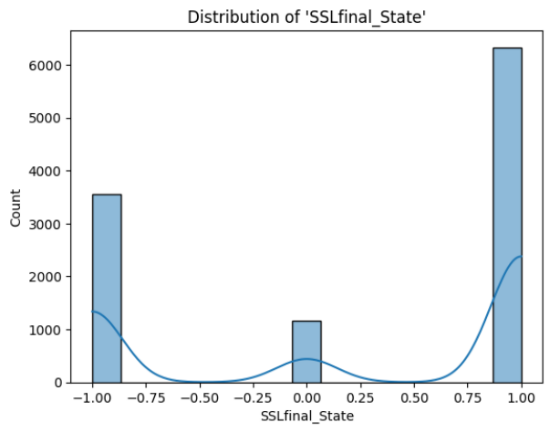
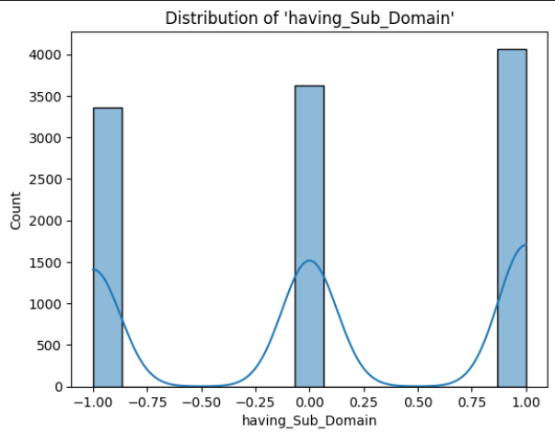
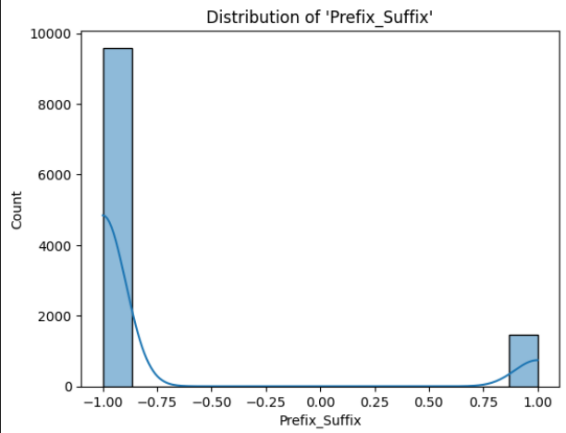
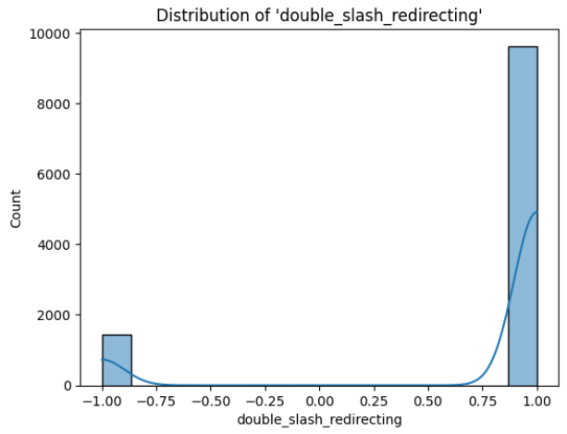
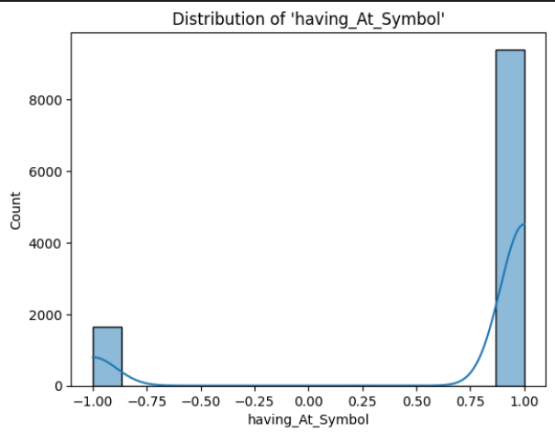
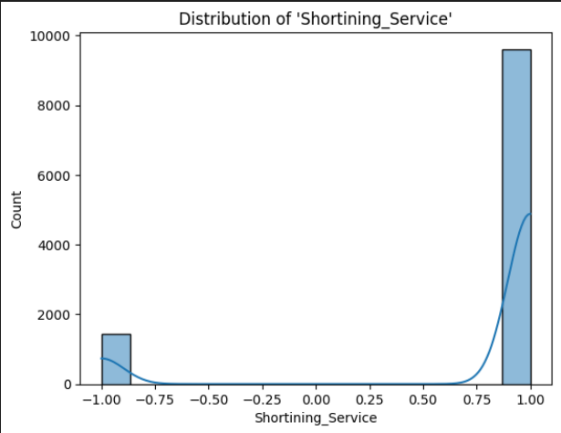
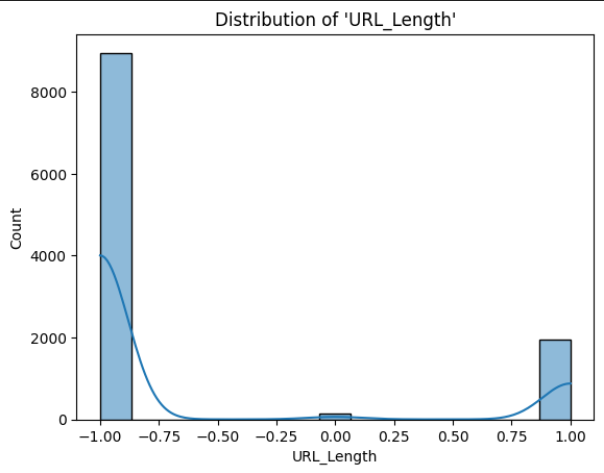
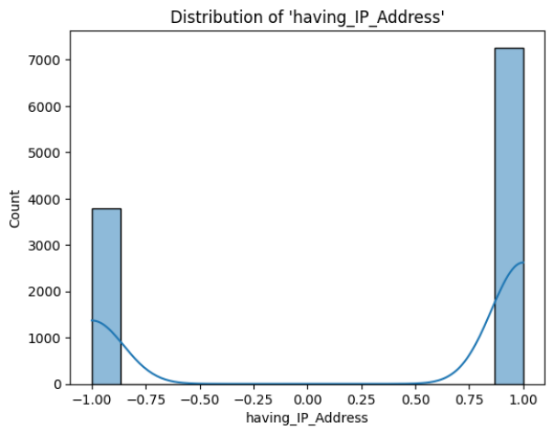
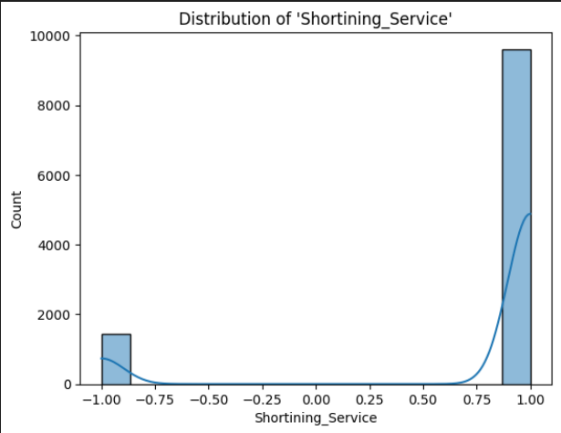
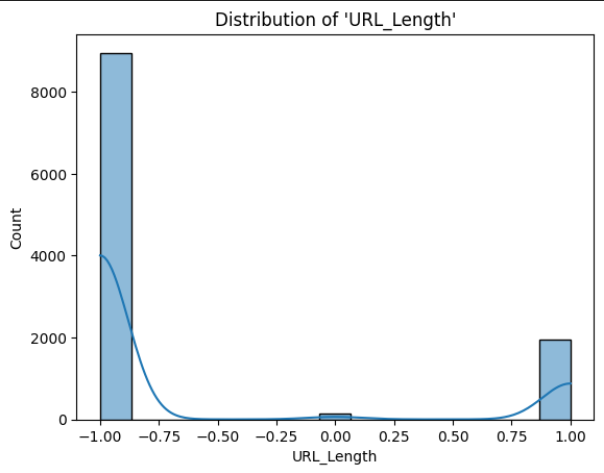
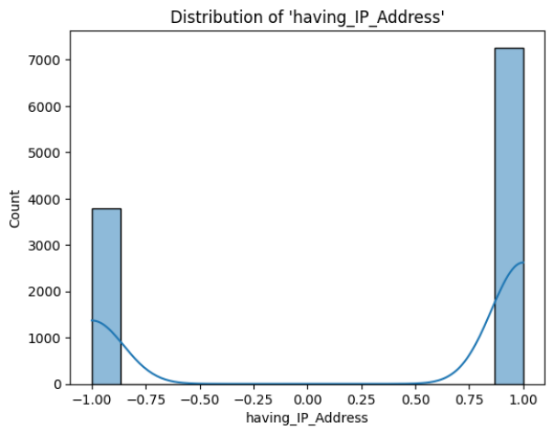
### 3.2.4 Domain-based Features

* ***Age\_of\_Domain:*** The WHOIS tool returns information on the domain’s age through Age\_of\_Domain. Older domains become more trustworthy than new domains while recently registered domains tend towards phishing activities.
* ***Domain\_registration\_length:*** The domain registration length variable identifies the length of time a domain is registered. The typical lifespan of phishing websites amounts to one year when their registration lasts for this brief duration.
* ***DNSRecord:*** DNSRecord proves the authenticity of domain DNS records by checking their presence and correctness which shows clear signs of phishing activity.
* ***Web\_traffic:*** Analyzes website traffic metrics. Sites with legitimate operations receive more online users than phishing sites do.
* ***Page\_Rank:*** A website’s PageRank score assessment shows that phishing sites normally receive poor or non-existent PageRank values.
* ***Google\_Index:*** The Google\_Index test verifies website presence in Google search results because sites without search engine indexing display more warning signs of phishing activity.
* ***Links\_pointing\_to\_page:*** The number of external links directing to the page serves as an indicator of website legitimacy which increases with more links pointing to it.
* ***Statistical\_report:*** The analysis includes statistical reports which retrieve reputation information from external sources such as PhishTank or StopBadware.

## 3.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) refers to the process of summarizing and analyzing data sets in order to know their structure, identify patterns, discover outliers, and identify relationships among variables. EDA involves statistical summaries, visualization, and correlation analysis to glean insights prior to applying machine learning algorithms. EDA is important because it facilitates data cleaning, feature selection, and comprehension of data distributions, thus improving model performance and interpretability.

### 3.3.1 Distribution of Independent Features

****

### 3.3.2 Distribution of Target Variable

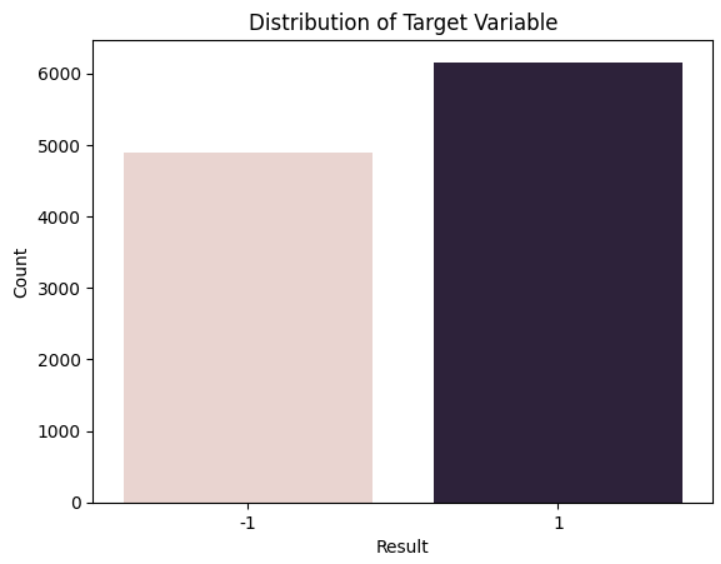
****

Figure 1: Class Distribution of Target Variable

The target variable distribution reveals both groups of websites are present with significant differences with low-class imbalance. The training sample of websites with data points labelled `1` exceeds the phishing websites labelled `-1` although the difference remains small enough to not distort results. Some imbalances remain present even though it does not severely impact the model performance of machine learning algorithms.

### 3.3.3 Correlation Heatmap

A correlation heatmap presents the correlation matrix data through a graphical display to show relationships between various database components. The scale for correlation values extends from -1 to 1 in which a 1 value demonstrates a strong positive correlation and -1 represents a complete negative correlation and 0 indicates no relationship. The colour gradients in the heatmap visualize data relationships through red-hot colours for positive correlations and blues for negative correlations. Visual representations in this format allow users to detect recurring patterns and recurring features as well as possible variable overlaps.

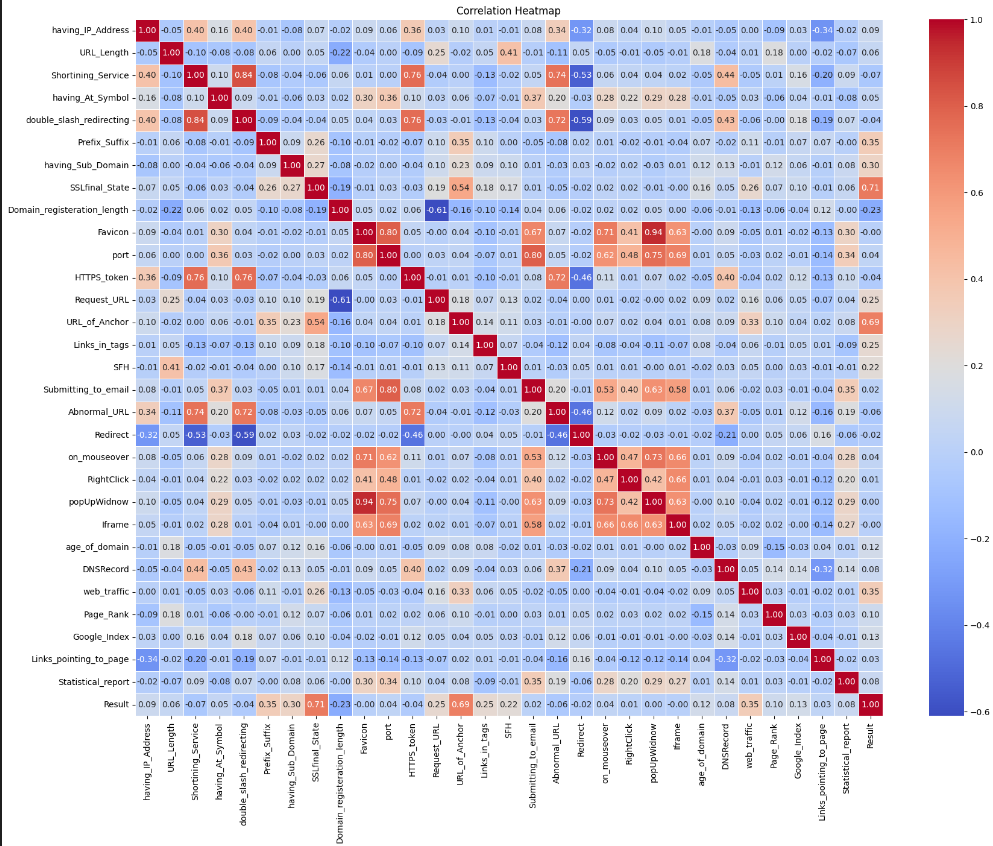
****

Figure 2 Correlation Heatmap of Features

The correlation heatmap for the dataset reveals the connections between different features from the phishing website dataset in terms of their relationship with the target variable *‘Result’*. Three key features named *"Abnormal\_URL"*, *"Statistical\_report"* and *"Submitting\_to\_email"* strongly relate to the target outcome thus proving important in phishing detection. The headmap also shows a high degree of correlation exists between *"Request\_URL"* with *"URL\_of\_Anchor"* which indicates unnecessary duplication of information between these variables. The heatmap serves as a tool for feature selection because it demonstrates which input variables drive classification outcomes while pointing to variables which may be either removed or altered for better model performance.

### 3.3.4 Chi-square Tests

The Chi-Square (χ²) test functions as a statistical technique which measures significant associations between two variables. The Chi-Square test analyses observed frequencies versus expected frequencies when variables are independent to reveal relationships between them. The hypothesis testing method uses this test when studying machine learning feature selection to identify important features which affect the target variable. The Chi-Square test for various features vs the target variable is given in Table 2.

Table 2: Chi-Square Test Result for Features vs Result

|  |  |  |
| --- | --- | --- |
| ***Feature*** | ***Chi-Square Statistic*** | ***P-Value*** |
| having\_IP\_Address | 97.616128 | 5.08 × 10⁻²³ |
| URL\_Length | 57.774681 | 2.85 × 10⁻¹³ |
| Shortining\_Service | 50.661841 | 1.10 × 10⁻¹² |
| having\_At\_Symbol | 30.694313 | 3.02 × 10⁻⁸ |
| double\_slash\_redirecting | 16.247156 | 5.56 × 10⁻⁵ |
| Prefix\_Suffix | 1341.399192 | 1.14 × 10⁻²⁹³ |
| having\_Sub\_Domain | 1595.294123 | 0.00 × 10⁰ |
| SSLfinal\_State | 6686.246056 | 0.00 × 10⁰ |
| Domain\_registration\_length | 562.628588 | 2.25 × 10⁻¹²⁴ |
| Favicon | 0.000023 | 9.96 × 10⁻¹ |
| port | 14.449423 | 1.44 × 10⁻⁴ |
| HTTPS\_token | 17.342195 | 3.12 × 10⁻⁵ |
| Request\_URL | 708.665429 | 3.90 × 10⁻¹⁵⁶ |
| URL\_of\_Anchor | 5966.367231 | 0.00 × 10⁰ |
| Links\_in\_tags | 712.564882 | 1.86 × 10⁻¹⁵⁵ |
| SFH | 542.417223 | 1.64 × 10⁻¹¹⁸ |
| Submitting\_to\_email | 3.587046 | 5.82 × 10⁻² |
| Abnormal\_URL | 40.104733 | 2.41 × 10⁻¹⁰ |
| Redirect | 4.346622 | 3.71 × 10⁻² |
| on\_mouseover | 19.091985 | 1.25 × 10⁻⁵ |
| RightClick | 1.646691 | 1.99 × 10⁻¹ |
| popUpWidnow | 0 | 1.00 × 10⁰ |
| Iframe | 0.104722 | 7.46 × 10⁻¹ |
| DNSRecord | 63.051548 | 2.01 × 10⁻¹⁵ |
| web\_traffic | 1712.181255 | 0.00 × 10⁰ |
| Links\_pointing\_to\_page | 66.522654 | 3.59 × 10⁻¹⁵ |
| Google\_Index | 183.075832 | 1.03 × 10⁻⁴¹ |
| Page\_Rank | 120.577585 | 4.73 × 10⁻²⁸ |
| age\_of\_domain | 162.697327 | 2.91 × 10⁻³⁷ |
| Statistical\_report | 70.036537 | 5.82 × 10⁻¹⁷ |

The Chi-Square test results determine the significance of certain attributes in the classification of phishing sites. Attributes such as *`SSLfinal\_State`, `URL\_of\_Anchor`, `web\_traffic`,* and *`having\_Sub\_Domain*` have extremely high Chi-Square values with p-values of nearly zero, reflecting high correlation with phishing identification. Moderate-value attributes, i.e., *`Google\_Index`, `age\_of\_domain*`, and *`Request\_URL`,* also demonstrate high associations. Conversely, attributes such as *`Favicon`, `popUpWindow`, `Iframe`,* and *`RightClick`* have extremely low Chi-Square values and high p-values, reflecting no or negligible contribution towards classification.

Chapter 04

# Models

## 4.1 Overview

The identification of phishing websites represents a significant challenge in cybersecurity because attackers exploit vulnerabilities in web security together with human trust elements. Machine learning effectively detects phishing attempts because it examines multiple URL and webpage characteristics. For this project, six machine learning models were implemented and tested, including Gaussian Naïve Bayes, Decision Tree, Logistic Regression, Support Vector Machine, Random Forest and K-Nearest Neighbours.

* ***Gaussian Naïve Bayes (GNB)***

Naïve Bayes operates as a probabilistic classifier which uses Bayes' theorem under the condition of independent features. The Gaussian version of Naïve Bayes makes the assumption that input data points match the normal distribution pattern. Naïve Bayes serves as an efficient probabilistic method which runs effectively for text classification tasks as well as probabilistic classification workloads. The binary nature of phishing detection as a legitimate or phishing decision problem fits Naïve Bayes since it works effectively with minimal training data (Talukder et al., 2024).

|  |  |
| --- | --- |
| ***Advantages*** | ***Disadvantages*** |
| * Fast & computationally inexpensive. | * Assumes independence among features, which is not always the case. |
| * Works well with noisy data and missing values. | * Can be outperformed by more complex models. |
| * Performs well on small datasets. |  |

* ***Decision Tree (DT)***

A decision tree is a non-linear classifier or regressor that models decisions using a tree-like structure. It recursively splits the dataset into subsets based on feature values, forming a hierarchy of decision rules. The tree is split on a particular node based on the information gain from the split. The attribute that provides the highest information gain is selected for the split, as it contributes most to making the data subsets more homogenous. Decision trees are able to capture complex relationships among features (*What Is a Decision Tree?*, 2021).

|  |  |
| --- | --- |
| ***Advantages*** | ***Disadvantages*** |
| * Easy to interpret and visualise. | * Can create overly complex models if not pruned properly. |
| * Handles both numerical and categorical data. | * Sensitive to Noisy data. |
| * No need for extensive feature scaling. |  |

* ***Logistic Regression (LR)***

Logistic regression is a widely used statistical method for binary classification problems when the outcome variable is categorical with two classes.  The reason the outcome is a probability is because the bounded dependent variable ranges between 0 and 1. In logistic regression, the odds are logit transformed—that is, the chance of success over the chance of failure. This is also often referred to as the log odds or natural logarithm of odds (*What Is Logistic Regression?*, 2021).

|  |  |
| --- | --- |
| ***Advantages*** | ***Disadvantages*** |
| * Simple & interpretable. | * Struggles with complex, non-linear relationships. |
| * Computationally efficient. | * Assumes independence among predictors. |
| * Works well when features are linearly separable. |  |

* ***Support Vector Machine (SVM)***

The supervised machine learning algorithm support vector machine (SVM) uses an optimal line or hyperplane to separate data points by maximizing the distance between each class in an N-dimensional space. SVMs function as standard methods for classifying data. SVM algorithms define two classes by generating an optimal hyperplane that creates the maximum gap between adjacent points irrespective of their classification. The number of features present in the input data establishes whether the learned hyperplane becomes a line when using 2-D data space or turns into a plane when operating in a higher-dimensional space (*What Is Support Vector Machine?*, 2023).

|  |  |
| --- | --- |
| ***Advantages*** | ***Disadvantages*** |
| * Effective in high-dimensional feature spaces. | * Computationally expensive for large datasets. |
| * Robust against overfitting. | * Requires careful tuning of hyperparameters. |

* ***Random Forest (RF)***

Random Forest combines the results by combining multiple decision trees. It is an ensemble learning method that builds multiple decision trees and averages their predictions. The different trees utilize distinct random portions of the dataset for training purposes before they combine their results through average computation. The solution helps in creating more accurate prediction results (*Random Forest Algorithm in Machine Learning*, 2025).

|  |  |
| --- | --- |
| ***Advantages*** | ***Disadvantages*** |
| * Handles missing data. | * Computationally expensive. |
| * Works well with large datasets. |  |

* ***K-Nearest Neighbours (KNN)***

K-nearest neighbours (KNN) operate as a basic supervised machine learning method by determining predictions through data points in proximity to each other. The algorithm serves as a fundamental approach for classification and regression because people utilize it frequently. The classification process within the KNN algorithm assigns the prevalent class label from among K neighbours to serve as the prediction for an incoming data point. During regression, the KNN algorithm obtains an average or weight-based average of target values from surrounding neighbours to estimate the response value (“Guide to K-Nearest Neighbours Algorithm in Machine Learning,” 2024).

|  |  |
| --- | --- |
| ***Advantages*** | ***Disadvantages*** |
| * Easy to understand and implement. | * Computationally expensive for large datasets. |
| * Works well on smaller datasets. |  |

## 4.2 Implementation

### 4.2.1 Gaussian Naïve Bayes (GNB)

Implemented using *GaussianNB*  function from *sklearn.naive\_bayes.* The model training involves estimating the mean and variance of each feature per class. Predictions are made by computing the posterior probability for each class and selecting the most probable class.

The model achieved an accuracy of 58.30%. The results also show that the model performed well in identifying non-phishing websites (-1) with a recall of 100% but struggled with identifying phishing websites (1) with a recall of only 27%. The f1-score for phishing websites is 0.42 showcasing a poor balance between precision and recall. The full metrics are given in Table 3 the results are also visualised with the confusion matrix in Fig. 03.

Table 3: Statistical Metrics for Gaussian Naive Bayes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***Precision*** | ***Recall*** | ***F1-Score*** | ***Support*** |
| -1 | 0.51 | 1.00 | 0.67 | 956 |
| 1 | 1.00 | 0.27 | 0.42 | 1255 |

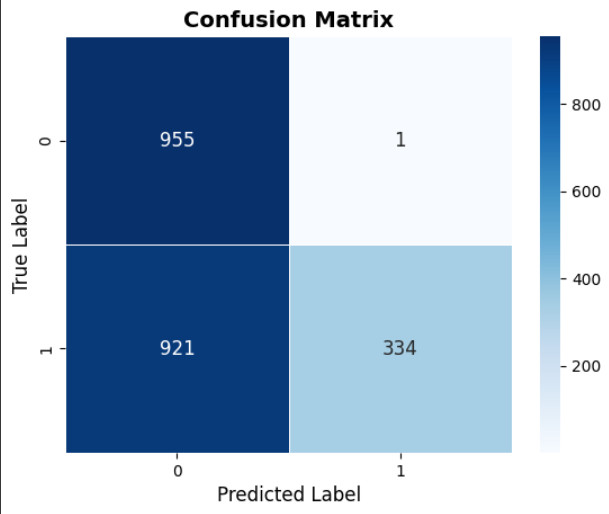


Figure 03: Confusion Metrics for GNB

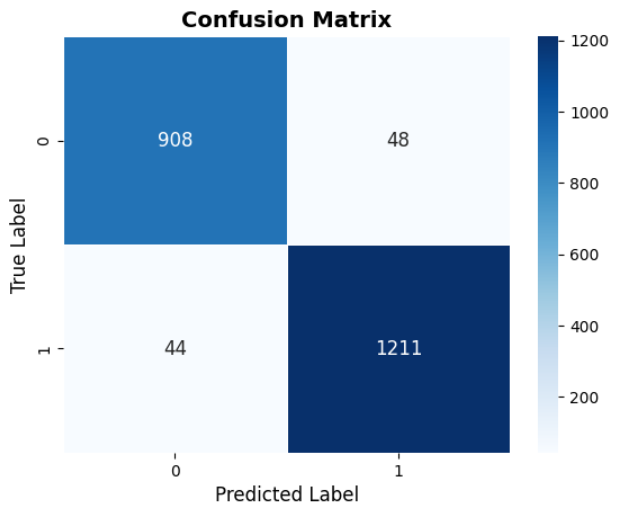
### 4.2.2 Decision Tree (DT)

The *DecisionTreeClassifier* from *sklearn.tree* was used for implementation. The model learns decision rules from the training data and predicts website classifications by following these rules.

The Decision Tree classifier achieved an accuracy of 95.84%. Results show that the model effectively differentiates between phishing and legitimate websites, with both classes achieving high precision and recall scores. The recall for phishing websites (-1) was 95%, while the recall for legitimate websites (1) was 96%, indicating a well-balanced classifier. The f1-score for legitimate websites was 0.96, showing the model's ability to maintain a good balance between precision and recall. The full metrics are given in Table 4 the results are also visualised with the confusion matrix in Fig. 04.

Table 4: Statistical Metrics for Decision Tree

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***Precision*** | ***Recall*** | ***F1-Score*** | ***Support*** |
| -1 | 0.95 | 0.95 | 0.95 | 956 |
| 1 | 0.96 | 0.96 | 0.96 | 1255 |

****

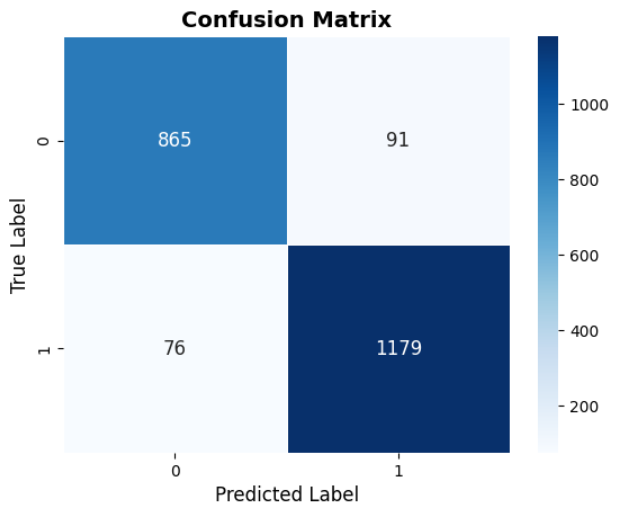
### 4.2.3 Logistic Regression (LR)

Logistic Regression was implemented using *LogisticRegression* from *sklearn.linear\_model*. The model applies a weighted sum of input features and maps it to a probability value between 0 and 1.

The Logistic Regression classifier achieved an accuracy of 92.45%, indicating strong performance in distinguishing phishing from legitimate websites. The classification report shows that the model performed consistently across both classes, with the legitimate class (1) achieving a precision of 0.93 and a recall of 0.94. The f1-score for legitimate websites was 0.93, demonstrating a strong balance between precision and recall. The full metrics are given in Table 5 the results are also visualised with the confusion matrix in Fig. 5.

Table 5: Statistical Metrics for Logistic Regression

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***Precision*** | ***Recall*** | ***F1-Score*** | ***Support*** |
| -1 | 0.92 | 0.90 | 0.91 | 956 |
| 1 | 0.93 | 0.94 | 0.93 | 1255 |



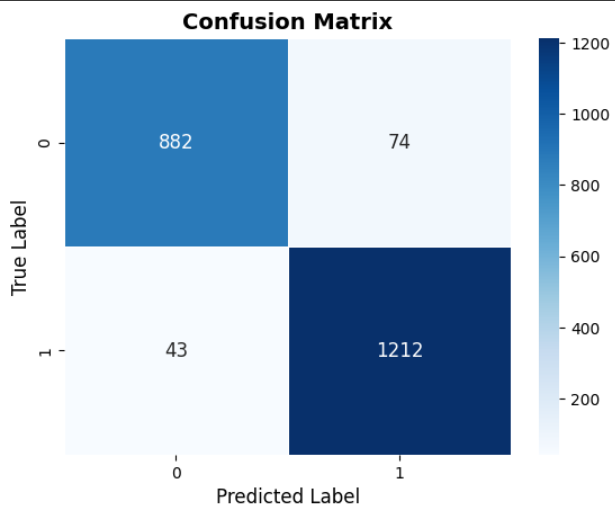
### 4.2.4 Support Vector Machine (SVM)

The model constructs a decision boundary that maximizes the margin between phishing and legitimate websites, improving classification performance.

The SVM classifier achieved an accuracy of 94.71%, indicating high effectiveness in classifying phishing and legitimate websites. The classification report reveals that the model maintained a strong balance between precision and recall for both classes. The recall for phishing websites (-1) was 92%, while the recall for legitimate websites (1) was 97%, demonstrating the model’s ability to correctly classify most instances. The full metrics are given in Table 6 the results are also visualised with the confusion matrix in Fig. 6.

Table 6: Statistical Metrics for SVM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***Precision*** | ***Recall*** | ***F1-Score*** | ***Support*** |
| -1 | 0.95 | 0.92 | 0.94 | 956 |
| 1 | 0.94 | 0.97 | 0.95 | 1255 |



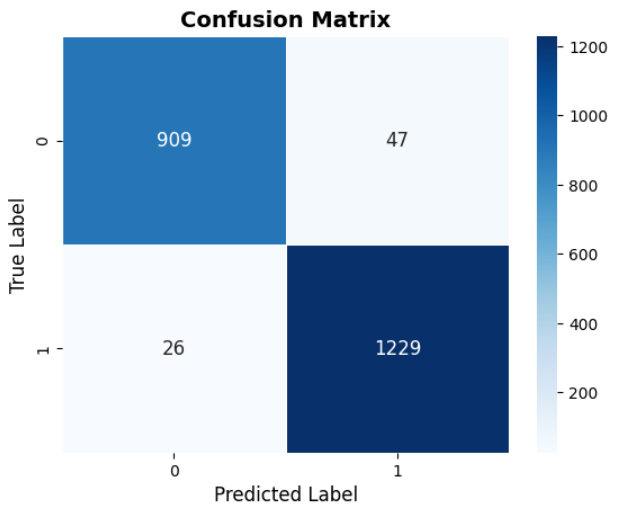
### 4.2.5 Random Forest (RF)

The *RandomForestClassifier* function from *sklearn.ensemble* is used for implementation. The model generates multiple decision trees using different subsets of the training data and combines their outputs to determine the final classification.

The Random Forest classifier achieved an accuracy of 96.70%, showing excellent performance in phishing website detection. The classification report indicates that the model effectively distinguishes between phishing and legitimate websites. The recall for phishing websites (-1) was 95%, while the recall for legitimate websites (1) was 98%, highlighting the model’s ability to capture important patterns. The full metrics are given in Table 7, the results are also visualised with the confusion matrix in Fig. 07.

Table 7: Statistical Metrics for Random Forest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***Precision*** | ***Recall*** | ***F1-Score*** | ***Support*** |
| -1 | 0.97 | 0.95 | 0.96 | 956 |
| 1 | 0.96 | 0.98 | 0.97 | 1255 |



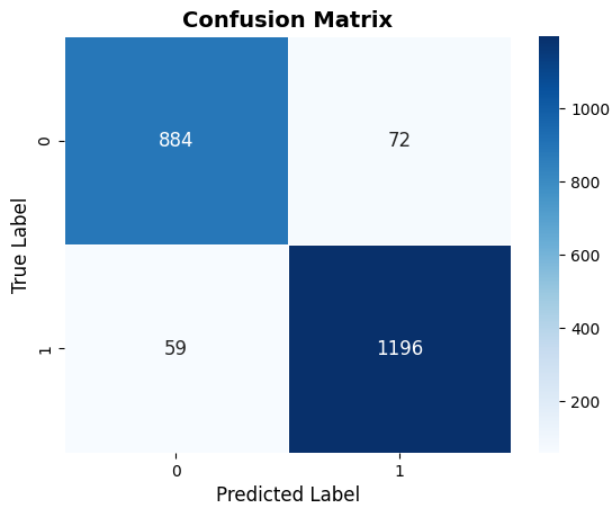
### 4.2.6 K-Nearest Neighbours (KNN)

The *KNeighborsClassifier* function from *sklearn.neighbors* is used in this implementation. The model determines the classification of a new instance by analyzing the k-nearest data points in the training set.

The KNN classifier achieved an accuracy of 94.08%, showing strong performance in phishing detection. The classification report indicates that the model provided balanced precision and recall scores across both classes. The recall for phishing websites (-1) was 92%, while the recall for legitimate websites (1) was 95%, highlighting its ability to generalize effectively. The full metrics are given in Table 8, the results are also visualised with the confusion matrix in Fig. 08

Table 8: Statistical Metrics for KNN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Class*** | ***Precision*** | ***Recall*** | ***F1-Score*** | ***Support*** |
| -1 | 0.94 | 0.92 | 0.93 | 956 |
| 1 | 0.94 | 0.95 | 0.95 | 1255 |



All the models are compared on the basis of accuracy and compared in Table 09.

Table 9: Accuracy Comparison of all Models without Fine Tuning

|  |  |  |
| --- | --- | --- |
|  | ***Model*** | ***Accuracy*** |
| *1.* | *Random Forest* | *96.52%* |
| *2.* | *Decision Tree* | *95.92%* |
| *3.* | *Support Vector Machine* | *94.70%* |
| *4.* | *K-Nearest Neighbours* | *94.07%* |
| *5.* | *Logistic Regression* | *92.44%* |
| *6.* | *Gaussian Naïve Bayes* | *58.29%* |

The random Forest model showed the best accuracy level at 96.52% followed by the Decision Tree at a slightly lower yet still strong accuracy rate of 95.92%. The performance of the Support Vector Machine (SVM) and K-Nearest Neighbours (KNN) produced similar results that reached 94.70% and 94.07% respectively. The simpler linear model Logistic Regression obtained an accuracy level of 92.44%. The strong assumption of feature independence in Gaussian Naïve Bayes produced insufficient performance with a 58.29% accuracy level since this assumption does not work well with phishing detection datasets.

This demonstrates how ensemble techniques like Random Forest and decision-tree models excel at phishing detection tasks. Further, additional refining techniques and validation approaches should be implemented to boost the generalization capacity of these prediction models.

Chapter 05

# Model Validation & Fine-Tuning

## 5.1 Model Validation Techniques

To make the models robust and capable of generalization, different validation methods were used. These included K-Fold Cross Validation, Stratified K-Fold Cross Validation, and Nested Cross Validation to prevent overfitting and verify model stability.

### 5.1.1 K-Fold Cross Validation

K-fold cross-validation divides data into k-subsets (folds), the model is trained on k-1 folds and tested on the remaining fold. This is repeated k-times.

K-Fold Cross Validation was implemented on Logistic Regression, K-Nearest Neighbours and Gaussian Naïve Bayes models. The data set was divided into 15 sections so validation occurred only once in each segment before training on the other parts. By implementing this technique, the model gained the ability to detect generic patterns which reduced the amount of variance in results.

The mean accuracy of Logistic Regression reached 92.78% after K-Fold validation proving the model could work consistently when trained on different subsets. During validation, the K-Nearest Neighbours method improved accuracy to 94.90% while its pre-validation level remained at 94.07%. The accuracy level for Gaussian Naïve Bayes kept fluctuating at 60.42% due to its unstable performance during testing while proving its inadequacy for this particular dataset.

### 5.1.2 Stratified K-Fold Cross Validation

Similar to k-fold, but ensures that each fold maintains the class distribution of the dataset. For class-sensitive classifiers, Stratified K-Fold Cross Validation was employed to preserve a balanced proportion of phishing and non-phishing sites in every fold. This method was very useful for Support Vector Machine (SVM) since the model achieved a uniform accuracy of 94.89%, an increment from its original 94.70%, when training samples were distributed evenly among folds.

### 5.1.3 Nested Cross Validation

In Nested Cross Validation a two-level K-fold validation process is used. Two loops are used where the inner loop is used for hyperparameter tuning and the outer loop is used for evaluating the final model. The Random Forest received Nested Cross Validation treatment through a hyperparameter tuning inner loop that accompanied validation with an outer loop. The validation process showed unbiased performance assessment results since Random Forest maintained its 96.94% accuracy level post-validation.

## 5.2 Hyperparameter Tuning

Hyperparameter tuning was implemented using Grid Search and Random Search to optimize model performance by systematically exploring parameter values.

### 5.2.1 Grid Search

Grid Search tests all the possible combinations of hyperparameters for tuning. The max\_depth parameter served as a key setting in Decision Tree to determine the depth of the tree structure. Increasing the depth improves sophisticated decision boundaries yet raises the probability of overfitting occurring in the model. The best performance occurred when max\_depth was set to None because this value managed to strike a balance between model complexity and generalization. In the parameter tuning process of min\_samples\_split, the model used 2, 5 and 10 as its values. Relocating min\_samples\_split to 2 boosted results with an accuracy reaching 96.37% while the initial pre-tuning showed 95.92%.

### 5.2.2 Random Search

Random Search tests a random selection of hyperparameter values and is faster than grid search and suitable for random forests. The n\_estimators parameter, the size of the set of decision trees in the forest, was tuned at 50, 100, 200, and 300. The best accuracy was achieved with an n\_estimators of 200, as the addition of trees enhanced performance at minimal computational expense. The max\_depth parameter, the maximum tree development to avoid overfitting, was tuned at 10, 20, and None, with max\_depth = None creating the most balanced accuracy. Therefore, the accuracy of Random Forest was improved from 96.56% to 97.08%, confirming the advantage of controlled tree development and ensemble learning.

A summary of model improvement is given in the table

Table 10: Summary of Model Accuracies after Fine Tuning

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***Model*** | ***Method*** | ***Previous Accuracy*** | ***Improved Accuracy*** |
| 1. | Random Forest | Random Search | 96.56% | 97.08% |
| 2. | Random Forest | Nested Cross Validation | 96.56% | 96.94% |
| 3. | Decision Tree | Grid Search | 95.92% | 96.37% |
| 4. | K-Nearest Neighbours | K-Fold Cross Validation | 94.07% | 94.89% |
| 5. | Support Vector Machine | Stratified K-fold | 94.70% | 94.89% |
| 6. | Logistic Regression | K-Fold Cross Validation | 92.44% | 92.78% |
| 7. | Gaussian Naïve Bayes | K-Fold Cross Validation | 58.29% | 60.42% |

The optimization metrics produced enhanced efficiency for machine learning techniques during performance testing sessions. The Random Forest model achieved its maximum accuracy increase by first using a Random Search for an improvement from 96.56% to 97.08% and subsequently using Nested cross-validation to attain a further increase to 96.94%. Decision Tree became more effective when Grid Search was applied as its accuracy rose from 95.92% to 96.37% due to optimizing both tree depth and splitting criteria. The application of K-Fold Cross Validation to K-Nearest Neighbours produced a 0.82% accuracy increase from 94.07% to 94.89% and the Support Vector Machine achieved a similar enhancement by using Stratified K-Fold from 94.70% to 94.89%. Logistic Regression enhanced its generalization capability through K-Fold cross-validation because accuracy improved from 92.44% to 92.78%. Gaussian Naïve Bayes demonstrated the most notable rise in performance metrics after K-Fold Cross Validation since its initial accuracy level was 58.29% but rose to 60.42%. Model performance improved after multiple validation repetitions because this process stabilized its probabilistic prediction capacity. Both Random Forest and Decision Tree benefited the most from parameter optimization and K-Fold Cross Validation and Stratified K-Fold Cross Validation increased model consistency with strong generalization properties.

Chapter 06

# Analysis & Recommendations

After training and validation of all the models, analysis was made on the test vs training accuracy of all the models. Starting with the Gaussian Naïve Bayes algorithms, it was found that the training and testing accuracy of the model demonstrates that the model produces unstable results at first but reaches stability by increasing the training set size. Because of favourable test splits or high variance resulting from small training data the model produces higher testing accuracy than training accuracy at the beginning of the process. As shown in Fig 6.1 the model's accuracy rates during training and testing approach 61% together because the available feature information reaches its learning ability.

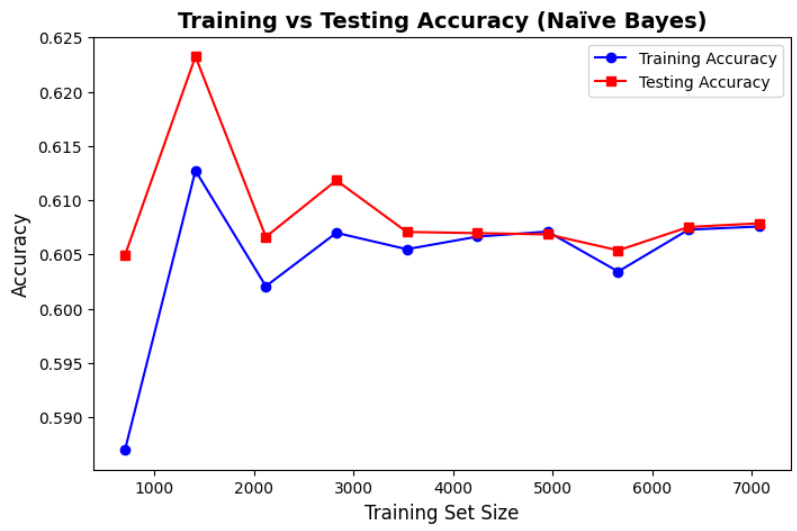


Figure 6.1

Naïve Bayes performance can be enhanced through critical feature engineering which involves eliminating redundant features and features with high correlations to fulfil its independence assumptions. The chosen Naïve Bayes variant should be selected by testing both Gaussian and Multinomial models against the data distribution patterns. The predictive power of Naïve Bayes can be strengthened by implementing both ensemble techniques such as Bagging and hybrid model systems that incorporate this approach. The limited capability of Naïve Bayes hyperparameter tuning allows for performance improvements when optimizing Laplace smoothing and adjusting priors and using SMOTE alongside stratified sampling to handle class imbalances.

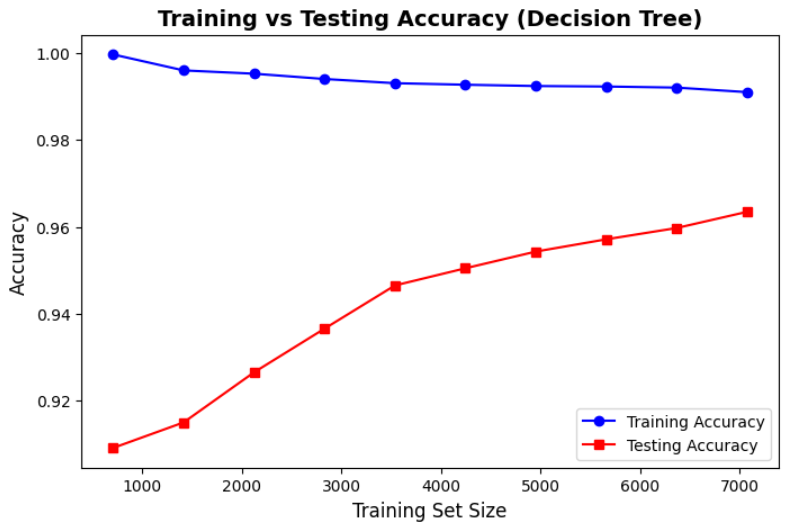


Figure 6.2

Fig. 6.2shows that the Decision Tree model demonstrated overfitting behaviour through its perfect training accuracy but it shows reduced accuracy on testing data. The testing accuracy shows greater improvements along with additional training data yet it sustains a notably lower level which indicates the model uses memorization rather than sound generalization techniques. Deep trees tend to overfit by becoming sensitive to noise which reduces their ability to detect phishing attacks in realistic scenarios. The techniques of pruning and setting depth limits and minimum sample split sizes should be used to improve the model's generalization abilities. Ensemble methods that include Random Forest or AdaBoost enable additional enhancement of stability. Feature selection together with enhanced engineering techniques will decrease noise levels which results in better model performance.

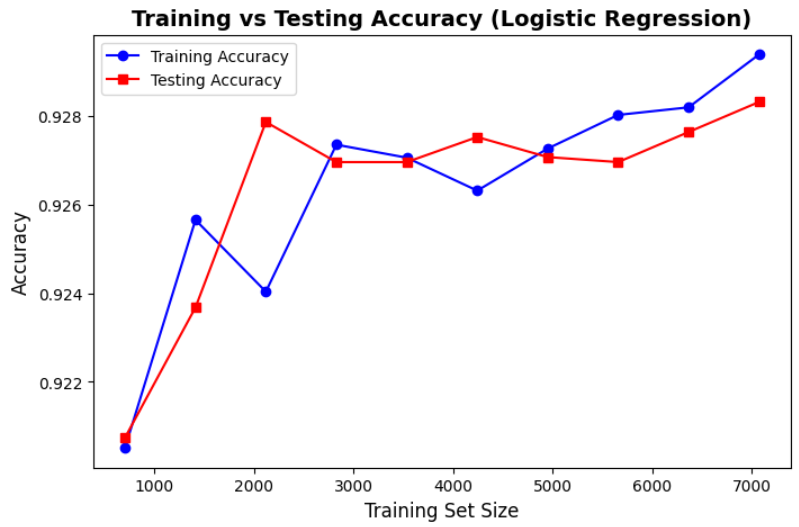


Figure 6.3

Fig. 6.3shows that the training and testing accuracies in Logistic Regression exhibit similar results because the model resists overfitting effects. Additional training data enables a steady improvement of both metrics in the model framework. The minor fluctuations indicate that the model responds to data distribution changes yet these fluctuations indicate dependable generalization performance.

Two types of regularization techniques including L1 (Lasso) and L2 (Ridge) help prevent minor overfitting. Feature scaling along with selection methods help both improve efficiency and convergence performance. Polynomial features together with kernel methods implemented through support vector machines should be considered to create better detection boundaries because non-linear behaviour potentially affects phishing detection.

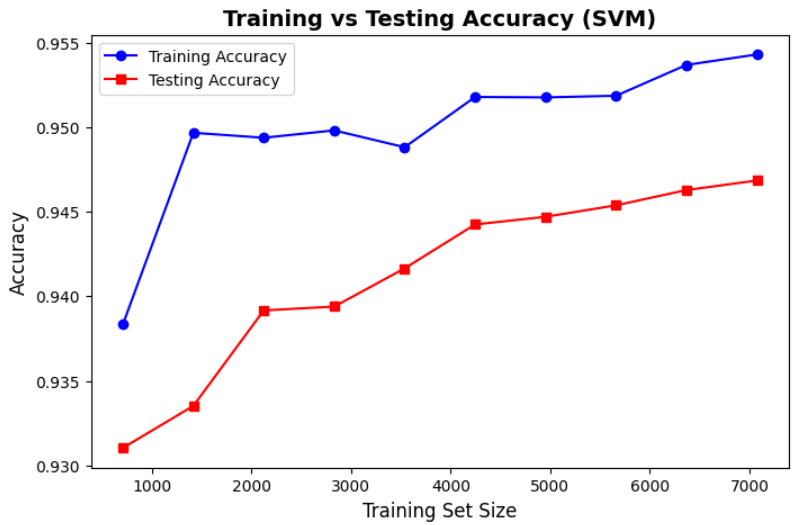


Figure 6.4

The SVM model shows an excellent ability to generalize because its training and testing accuracies keep growing together as the training set expands in size. Fig. 6.4shows that there is minimal overfitting effect shown by the gap between test and training accuracy but the model maintains persistent improvements as training sample numbers increase. The SVM proves its capability to extract meaningful information from the data as it generates precise outcomes.

Reaching better results will become possible through hyperparameter optimization of kernel type alongside C and gamma values selection. Transforming the data using PCA methods can enhance computational performance. Large datasets could be addressed with fast training capabilities through either linear SVM or stochastic gradient descent (SGD)-based SVM.

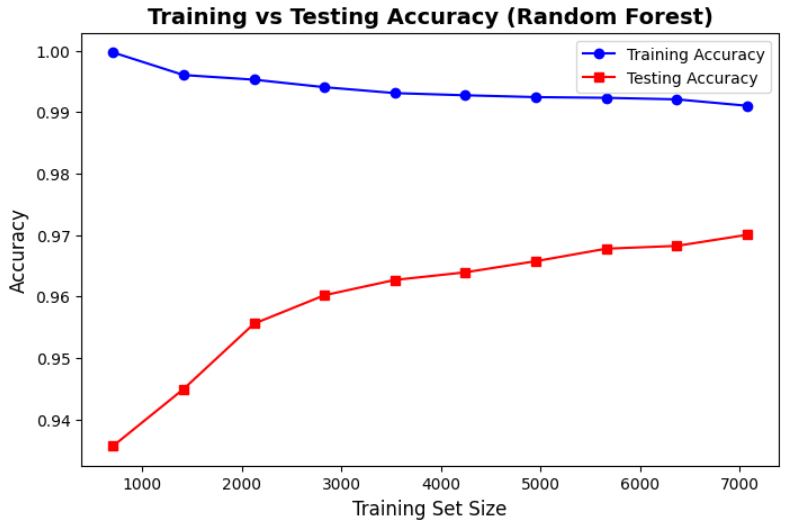


Figure 6.5

Fig 6.5 shows that the Random Forest model demonstrated a training accuracy level near 100% because it proves exceptional in learning training data patterns. The testing accuracy demonstrated growth when the size of the dataset expanded but continued to remain at a lower level than training accuracy thus pointing toward overfitting conditions. Multiple deep trees within Random Forest models cause expected data memorization of training samples because ensemble techniques share this characteristic.

Implementing maximum tree depth reduction and stronger feature split limitations with optimum estimator counts helps prevent model overfitting. Testing accuracy would improve because of feature selection and the reduction of correlated features that help enhance the model's generalization capabilities.

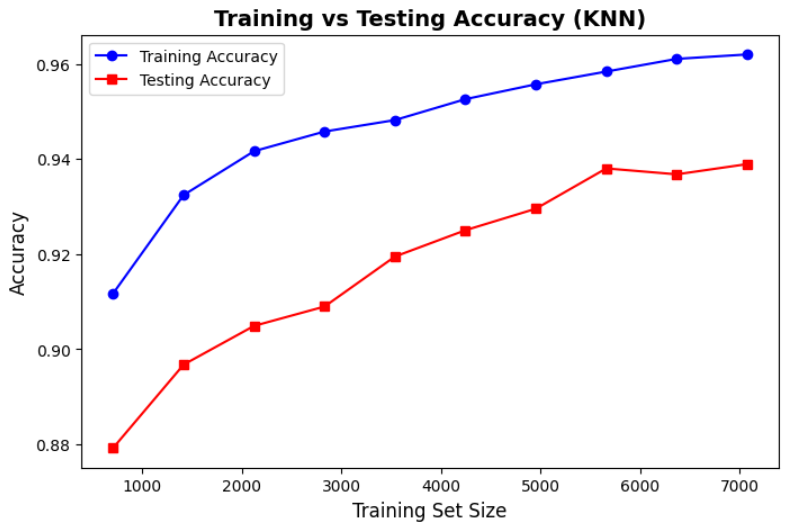


Figure 6.6

Fig. 6.6 shows that the K-Nearest Neighbours (KNN) model demonstrated distinct metrics for training accuracy and testing accuracy because it overfits part of the data. The training accuracy alongside testing accuracy grows together with training set volume although the model showed limited capabilities to achieve perfect generalization performance. KNN generally presents this issue because it becomes susceptible to noise while overfitting occurs when k remains too small.

The decision boundary smoothing for KNN models occurs through k adjustment which usually requires increasing k to achieve better generalization. The model performance can be improved with three techniques: distance weighting, feature scaling and a reduction of dimensionality through PCA methods.

# References

Aldaham, S. S. M., Ouda, O., & El-Aziz, A. A. A. (2024). Improved Detection of Phishing Websites using Machine Learning. *International Journal of Intelligent Systems and Applications in Engineering*, *12*(21s), Article 21s.

Çolhak, F., Ecevit, M. İ., Uçar, B. E., Creutzburg, R., & Dağ, H. (2024). *Phishing Website Detection through Multi-Model Analysis of HTML Content* (No. arXiv:2401.04820). arXiv. https://doi.org/10.48550/arXiv.2401.04820

Fairstein, Y., Kalinsky, O., Karnin, Z., Kushilevitz, G., Libov, A., & Tolmach, S. (2024). Class Balancing for Efficient Active Learning in Imbalanced Datasets. In S. Henning & M. Stede (Eds.), *Proceedings of The 18th Linguistic Annotation Workshop (LAW-XVIII)* (pp. 77–86). Association for Computational Linguistics. https://aclanthology.org/2024.law-1.8/

Guide to K-Nearest Neighbors Algorithm in Machine Learning. (2024). *Analytics Vidhya*. https://www.analyticsvidhya.com/articles/knn-algorithm/

Hamdard, A., & Lodin, H. (2023). Effect of Feature Selection on the Accuracy of Machine Learning Model. *INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH AND ANALYSIS*, *06*. https://doi.org/10.47191/ijmra/v6-i9-66

Innab, N., Osman, A., Ataelfadiel, M., Abu-Zanona, M., Elzaghmouri, B., Zawaideh, F., & Alawneh, M. (2024). Phishing Attacks Detection Using Ensemble Machine Learning Algorithms. *Computers, Materials & Continua*, *80*, 1–10. https://doi.org/10.32604/cmc.2024.051778

Kapan, S., & Sora Gunal, E. (2023). Improved Phishing Attack Detection with Machine Learning: A Comprehensive Evaluation of Classifiers and Features. *Applied Sciences*, *13*(24), Article 24. https://doi.org/10.3390/app132413269

Lin, W.-J., & Chen, J. J. (2013). Class-imbalanced classifiers for high-dimensional data. *Briefings in Bioinformatics*, *14*(1), 13–26. https://doi.org/10.1093/bib/bbs006

Maci, A., Santorsola, A., Coscia, A., & Iannacone, A. (2023). Unbalanced Web Phishing Classification through Deep Reinforcement Learning. *Computers*, *12*(6), Article 6. https://doi.org/10.3390/computers12060118

Mandadi, A., Boppana, S., Ravella, V., & Kavitha, R. (2022). Phishing Website Detection Using Machine Learning. *2022 IEEE 7th International Conference for Convergence in Technology (I2CT)*, 1–4. https://doi.org/10.1109/I2CT54291.2022.9824801

Newaz, A., Haq, F. S., & Ahmed, N. (2024). *A Sophisticated Framework for the Accurate Detection of Phishing Websites* (No. arXiv:2403.09735). arXiv. https://doi.org/10.48550/arXiv.2403.09735

Patil, R., Kaur, G., Jain, H., Tiwari, A., Joshi, S., Rao, K., & Sharma, D. A. (2022). Machine learning approach for phishing website detection: A literature survey. *Journal of Discrete Mathematical Sciences and Cryptography*, *25*, 817–827. https://doi.org/10.1080/09720529.2021.2016224

*Random Forest Algorithm in Machine Learning*. (17:06:15+00:00). GeeksforGeeks. https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

Talukder, A., Alam, F., & Emon, M. (2024). *Detecting Phishing Websites using Naive Bayes Classification* (p. 6). https://doi.org/10.1109/ICAEEE62219.2024.10561829

U S, S. S., Murugesan, P., & Dhandayutham, J. (2024). Phishing Website Detection using Machine Learning. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, *08*, 1–5. https://doi.org/10.55041/IJSREM36212

*What is a Decision Tree? | IBM*. (2021, November 2). https://www.ibm.com/think/topics/decision-trees

*What is ensemble learning? | IBM*. (2024, March 18). https://www.ibm.com/think/topics/ensemble-learning

*What Is Logistic Regression? | IBM*. (2021, August 16). https://www.ibm.com/think/topics/logistic-regression

*What Is Support Vector Machine? | IBM*. (2023, December 12). https://www.ibm.com/think/topics/support-vector-machine