DS3002A Data Mining
Final Project Report
Phishing Webpages Detection

Muhammad Ahmad 21L-5695
Farheen Akmal 21L-5666
Muhammad Abdullah 21L-6225

I. Introduction

Phishing, the fraudulent attempt to obtain sensitive information such as usernames, passwords, and credit card details by disguising oneself as a trustworthy entity in electronic communication, is a pervasive threat in today's digital landscape. With the rapid growth of online transactions and interactions, phishing attacks have become increasingly sophisticated and widespread, posing significant risks to individuals, organizations, and even entire economies. Understanding the dynamics of phishing, its underlying motivations, and effective strategies for mitigation are paramount in safeguarding against this cyber menace.

A. Motivation

The motivation behind this project stems from the pressing need to combat the rising tide of phishing attacks. As technology continues to advance, so do the tactics employed by malicious actors to deceive unsuspecting victims. The proliferation of phishing scams not only compromises sensitive personal and financial information but also undermines trust in online platforms and services. By undertaking this classification project, we aim to contribute to the ongoing efforts to combat cybercrime and enhance cybersecurity resilience across digital ecosystems.

B. Dataset Description

The dataset [1] utilized for this project is a result of merging two distinct datasets with identical features, focusing on phishing websites. This merger was undertaken to streamline and concentrate on the most pertinent data while avoiding redundancy, thus providing a comprehensive view of shared characteristics between the original datasets. The inclusion of diverse sources enriches the dataset, enabling a more holistic analysis of phishing website attributes and behaviors.

The dataset Fig 1 is provided in CSV format, with each row representing a website and each column denoting a specific feature. The final column contains labels indicating whether a website is classified as phishing (1) or legitimate (0). The features encompass various aspects of URLs, including length, character composition, and structural elements, providing valuable insights into the distinguishing characteristics of phishing websites.

url_length	n_dots	n_hypens	n_underline	n_slash	n_questionmark	n_equal	n_at	n_and	n_exclamation
37	3	0	0	0	0	0	0	0	0
77	1	0	0	0	0	0	0	0	0
126	4	1	2	0	1	3	0	2	0
18	2	0	0	0	0	0	0	0	0
55	2	2	0	0	0	0	0	0	0

n_space	n_tilde	n_comma	n_plus	n_asterisk	n_hastag	n_dollar	n_percent	n_redirection	phishing
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0

Fig. 1. First five rows of the dataset.

The dataset offers opportunities for various tasks, including binary classification, owing to its labeled nature. Beyond binary classification, exploratory analyses such as feature importance assessment, anomaly detection, and model interpretability can further elucidate the underlying patterns and dynamics of phishing website characteristics. Additionally, the dataset serves as a valuable resource for benchmarking and evaluating the efficacy of different machine learning algorithms in detecting and mitigating phishing threats.

C. Binary Classification

The primary objective of this project is to perform binary classification on the dataset, distinguishing between legitimate and malicious websites. This entails training a model capable of accurately predicting the label assigned to each website based on its features, thereby enabling proactive identification and mitigation of potential phishing threats. By leveraging advanced machine learning techniques, such as ensemble methods and deep learning architectures, we seek to develop a robust classification framework capable of adapting to evolving phishing tactics and strategies.

D. Legitimacy or Malicious Prediction

The outcome of the binary classification task will yield predictions regarding the legitimacy or malicious intent of each website. By categorizing websites into these distinct classes, we can equip users and cybersecurity professionals

with valuable insights to enhance their defense mechanisms against phishing attacks. Furthermore, the classification results can inform the development of automated threat detection systems and proactive security measures, bolstering resilience against emerging cyber threats in an increasingly interconnected digital landscape.

II. Binary Class Classification

A. Initialization and Preprocessing

In the initialization phase, we began by fetching the dataset from a remote source to facilitate collaborative work. This involved downloading the dataset and installing necessary dependencies using commands in a Python environment. We imported essential libraries such as NumPy, pandas, seaborn, TensorFlow, and matplotlib for data manipulation, visualization, and machine learning tasks. Following this, we read the dataset into a pandas DataFrame for further analysis and processing.

Preprocessing involves several steps aimed at preparing the dataset for subsequent analysis and modeling. Firstly, we address any potential null values within the dataset by computing the sum of null values in each column. The output provides insights into the completeness of the dataset, ensuring that missing values are appropriately handled to prevent any adverse effects on downstream tasks.

url_length	n_dots	n_hypens	n_underline	n_slash	n_questionmark	n_equal	n_at	n_and
-0.045395	0.617991	-0.315209	-0.190229	-0.620933	-0.145369	-0.224874	-0.082503	-0.156876
0.788431	-0.975585	-0.315209	-0.190229	-0.620933	-0.145369	-0.224874	-0.082503	-0.156876
1.809869	1.414778	0.462724	2.572234	-0.620933	5.814520	2.900786	-0.082503	2.032144
-0.441463	-0.178797	-0.315209	-0.190229	-0.620933	-0.145369	-0.224874	-0.082503	-0.156876
0.329827	-0.178797	1.240656	-0.190229	-0.620933	-0.145369	-0.224874	-0.082503	-0.156876

n_space	n_tilde	n_comma	n_plus	n_asterisk	n_hastag	n_dollar	n_percent	n_redirection	phishing
-0.03373	-0.04608	-0.029892	-0.023645	-0.014423	-0.007749	-0.01949	-0.064463	-0.466199	0
-0.03373	-0.04608	-0.029892	-0.023645	-0.014423	-0.007749	-0.01949	-0.064463	0.823312	1
-0.03373	-0.04608	-0.029892	-0.023645	-0.014423	-0.007749	-0.01949	-0.064463	0.823312	1
-0.03373	-0.04608	-0.029892	-0.023645	-0.014423	-0.007749	-0.01949	-0.064463	0.823312	0
-0.03373	-0.04608	-0.029892	-0.023645	-0.014423	-0.007749	-0.01949	-0.064463	0.823312	0

Fig. 2. Standardized dataset

Subsequently, standardization Fig 2 is performed to ensure uniformity and comparability among the features. Using the StandardScaler from scikit-learn, the numerical features are scaled to have a mean of 0 and a standard deviation of 1. This transformation minimizes the influence of outliers and variations in feature magnitudes, facilitating more stable and effective model training.

Finally, multicollinearity, or high correlation between predictor variables, is assessed to identify potential redundancies and dependencies among features. A correlation matrix Fig 3 is computed and visualized using a heatmap, highlighting pairwise correlations between features.From this matrix the independant attributes 1) 'n_equal' and 'n_and' and 2)'url_length' and 'n_equal' were observed to be highly positively correlated.

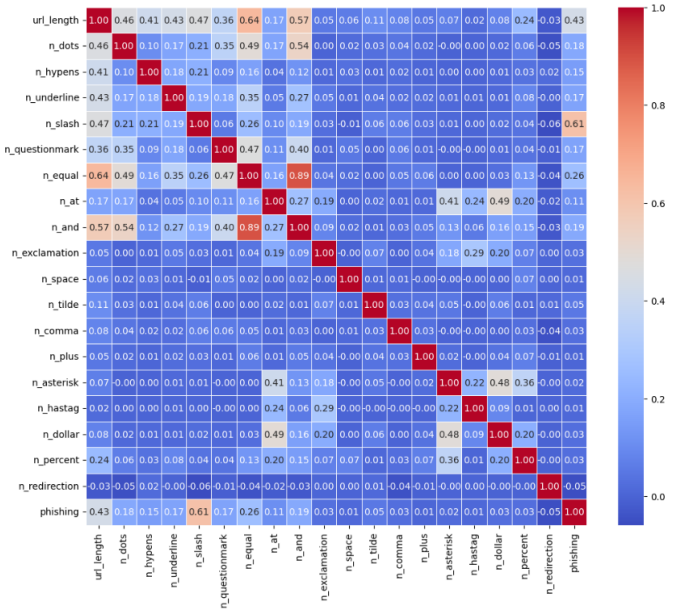

Fig. 3. Correlation Matrix

This analysis aids in feature selection and model interpretation by identifying highly correlated attributes that may be redundant or indicative of similar underlying patterns.

From the initial pool of 12 proposed models, we narrowed down to the top 8 models Fig 4 based on their performance. We trained each model on the standardized dataset and evaluated their accuracy using balanced accuracy score. The top-performing models included XGBoost, Neural Network, MLP Classification, Random Forest Classifier, Gradient Boosting, KNN, Deep Neural Network, and SVM. However, SVM was not considered further due to its lengthy computation time for multi-classification tasks.

Model	Raw Accuracy
XGBoost	0.884523
Neural Network	0.883204
MLP Classification	0.881253
Random Forest Classifier	0.879709
Gradient Boosting	0.874606
KNN	0.873202
Deep Neural Network	0.872393
SVM	0.868912
Simple Decision Tree	0.867027

Fig. 4. Filtering out 8 Best Models

B. Visualization

To understand the relationships between different features in our phishing website dataset, we utilized various graphs and charts. One such visualization is the scatter plot.

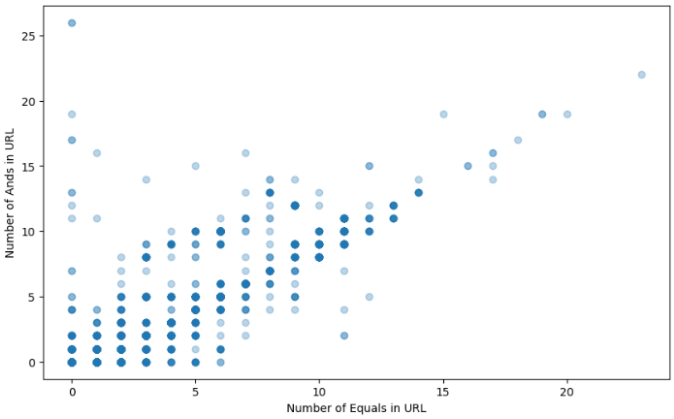

Fig. 5. Scatter Plot of Number of Equals vs Number of Ands

The scatter plot Fig 5 was used to explore the relationship between the number of equals signs (=) and the number of ampersands (&) in URLs within our dataset as it .The plot displays points representing individual observations, where the x-axis denotes the 'Number of Equals in URL' and the y-axis represents the 'Number of Ands in URL'. Each point's position indicates the count of these characters in a particular URL.The visualization reveals a concentration of data points with lower counts of both characters, suggesting that most URLs in our dataset contain fewer equals and ampersands. While there is no clear linear relationship, the presence of outliers with higher counts is noted.

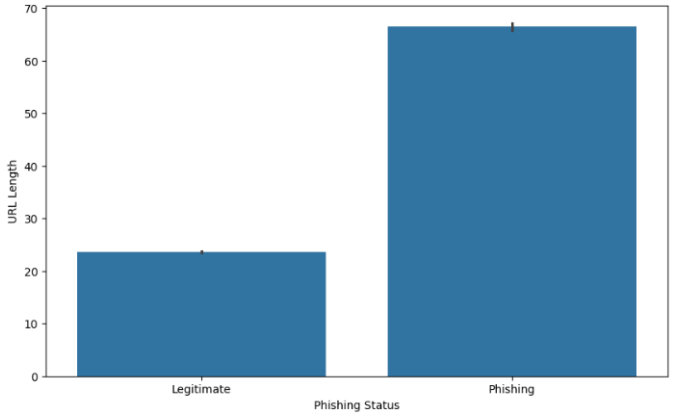

Fig. 6. URL Length vs Phishing Status

The bar plot Fig 6 is designed to compare the average URL length between legitimate and phishing websites.It features two bars, one for each category of websites. The 'Legitimate' bar reaches a value around 30, indicating the average URL length for legitimate sites, while the 'Phishing' bar extends just above 60, showing a significantly longer average URL length for phishing sites. The plot suggests that phishing websites tend to have longer URLs on average compared to legitimate ones. The error lines above each bar indicate the variability of the average URL length, providing a measure of confidence in the data.This characteristic of longer URLs in phishing sites can be a useful feature for machine learning models to identify potentially harmful websites. It also highlights the importance of URL length as a factor in website classification.

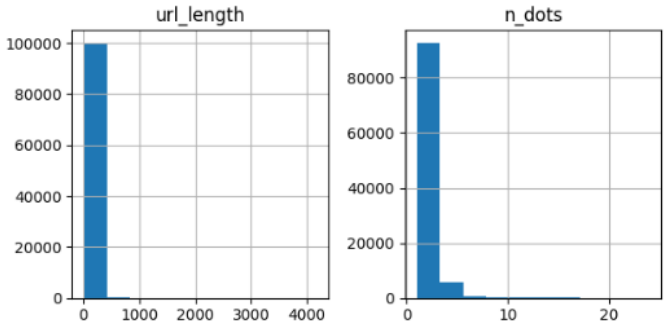

Fig. 7. Histograms for 'url_length' and 'n_dots'

Histograms are utilized to visualize the data distributions for each feature in our phishing website dataset.The histograms for 'url_length' and 'n_dots' are provided Fig 7 as examples (Similar histograms have been generated for the remaining 18 features in the dataset, each providing valuable insights into the distribution of values and aiding in the comprehensive analysis required for effective binary classification.). The 'url_length' histogram shows most URLs are short, with a frequency concentration at the lower end of the axis. The 'n_dots' histogram indicates that URLs typically contain a smaller number of dots, as evidenced by the left-skewed distribution.These histograms reveal the common characteristics of URLs in our dataset, such as the prevalence of shorter URLs and fewer dots, which could be indicative of phishing activities.

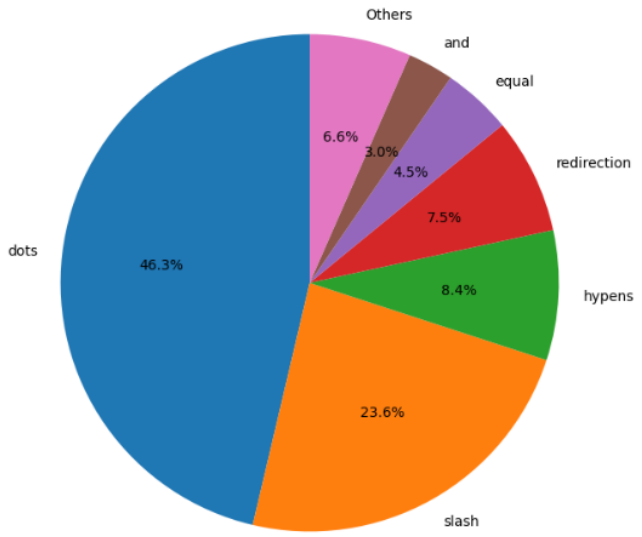

Fig. 8. Distribution of Special Characters in all URLs

The pie chart is used to illustrate the distribution of special characters in URLs, highlighting the proportion of each

character. The chart Fig 8 segments represent different special characters, with the largest segment being ”dots” at 46.3%, followed by ”slash” at 23.6%. Other notable segments include ”hyphens,” ”redirection,” ”equal,” and ”and.” Characters with proportions less than 14% are collectively categorized as ”Others”. The chart indicates that dots and slashes are the most common special characters in URLs within our dataset. The ”Others” category suggests a diverse presence of less frequent characters. This distribution is significant for understanding URL structure patterns, which can be crucial for cybersecurity analysis and identifying malicious URLs.

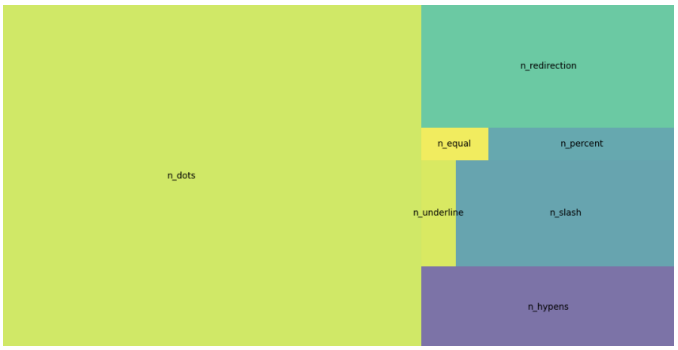

Fig. 9. Usual Count of Features Present in Phishing URLs

The tree map Fig 9 is employed to visualize the usual count of features present in phishing URLs.It consists of colored rectangles, each representing a feature from the dataset. The size of each rectangle is proportional to the count of the feature it represents.Features such as ’n_dots’, ’n_redirection’, ’n_equal’, ’n_percent’, ’n_underline’, ’n_slash’, and ’n_hyphens’ are depicted, with ’n_dots’ being the most prevalent feature. This visualization aids in quickly identifying which features are most common in phishing URLs, providing insights that can be leveraged for feature selection and model training.

C. Dealing Class Imbalance

The bar chart Fig 10 illustrates a class imbalance in our dataset, with class ‘0’ being significantly more frequent than class ‘1’. To address this, we will create two separate datasets: one using oversampling to increase the minority class and another using undersampling to decrease the majority class. This approach aims to balance the class distribution, ensuring our models are not biased towards the more prevalent class.

We utilized the SMOTE (Synthetic Minority Over-sampling Technique) to oversample the minority class. This technique generates synthetic samples that are similar to the existing ones in the minority class. The resulting bar chart shows an equal frequency of both classes ‘0’ (non-phishing) and ‘1’ (phishing), indicating that the class imbalance has been successfully addressed.With a balanced class distribution, machine learning models can be trained more effectively, as they are less likely to be biased towards the majority class. This enhances the model’s ability to generalize and perform well on unseen data.

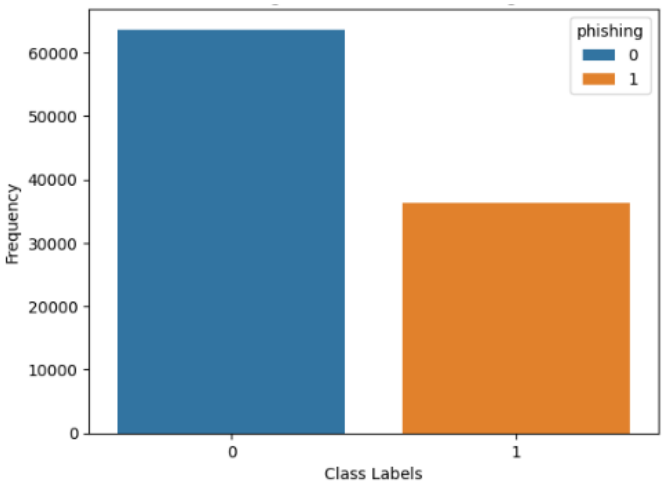

Fig. 10. Histogram of Feature ”Phishing

We employed the RandomUnderSampler from the imbalanced-learn library to perform undersampling. This method randomly removes instances from the majority class.The resulting bar chart shows an equal frequency of both classes ‘0’ (non-phishing) and ‘1’ (phishing), indicating that the class imbalance has been successfully addressed. The bar chart visualization confirms that the undersampling process has created a dataset where both classes are equally represented, which is essential for unbiased model training and evaluation.

D. Modelling Phase

In this phase, we delve into the modeling process, where we employed various machine learning algorithms to train and evaluate predictive models for binary classification of phishing websites.

To ensure unbiased evaluation of model performance, we split the dataset into training and testing sets. This facilitates assessing the models’ generalization capabilities by training on a subset of the data and evaluating on unseen samples. Both oversampled and undersampled datasets are created to address class imbalance issues. The oversampled dataset (os) and undersampled dataset (us) are further split into training and testing subsets using a 80:20 ratio.

For the oversampled dataset, we employ a set of narrowed down models to evaluate their performance in classifying phishing websites. The models include Simple Decision Tree, Random Forest Classifier, Gradient Boosting, XGBoost, MLP Classification, KNN, Neural Network, and Deep Neural Network. Each model is trained and evaluated using the oversampled training and testing subsets. The accuracy scores for each model are computed and stored in a DataFrame, which is then sorted in descending order based on accuracy.

Similarly, the undersampled dataset undergoes evaluation using the same set of narrowed down models. The models are trained and evaluated on the undersampled training and testing subsets to assess their performance in mitigating class

imbalance. The accuracy scores for each model are computed and stored in a DataFrame, which is sorted in descending order based on accuracy.

To compare the performance of models across different datasets and sampling techniques, we merge the accuracy results Fig 11 obtained from the raw dataset, oversampled dataset, and undersampled dataset. This facilitates a comprehensive analysis of model performance under varying conditions. The merged DataFrame provides insights into how each model performs across different sampling strategies, aiding in the selection of the most suitable models for further optimization and deployment.

Model	Raw Accuracy	Oversampled Accuracy	Undersampled Accuracy
XGBoost	0.884523	0.896688	0.889240
Neural Network	0.883204	0.889076	0.883946
MLP Classification	0.881253	0.892215	0.887521
Random Forest Classifier	0.879709	0.895747	0.888690
Gradient Boosting	0.874606	0.887546	0.881953
KNN	0.873202	0.882288	0.876246
Deep Neural Network	0.872393	0.890646	0.883534
Simple Decision Tree	0.867027	0.888252	0.878240

Fig. 11. For Analysing Default Results

The analysis of default results indicates that XGBoost consistently demonstrates strong performance across all datasets, achieving the highest accuracy among the evaluated models. Additionally, Neural Network and MLP Classification also exhibit competitive performance across different datasets, highlighting their efficacy in handling class imbalance and complex feature relationships. These findings serve as a foundation for further fine-tuning and optimization of the selected models to enhance their predictive capabilities for phishing website classification.

E. Finding and Finalizing Parameters

In this phase, we focused on identifying efficient dataset and then hyperparameters for our models through rigorous evaluation and tuning.

1) Finding Efficient Data: K-Fold CV

In assessing the generalization performance of our models, we employed k-fold cross-validation across unsampled, oversampled, and undersampled datasets. This technique [6] involves training and evaluating each model multiple times on different subsets of the data, computing the mean accuracy and standard deviation for each model. Across all datasets, XGBoost consistently demonstrates strong performance, achieving mean accuracies ranging from 89.12% to 89.33%. The Random Forest Classifier also performs well, with mean accuracies ranging from 89.12% to 89.59%. MLP Classification and Gradient Boosting show competitive results, albeit slightly lower than XGBoost and Random Forest Classifier. These findings underscore the robustness of XGBoost and Random Forest Classifier across different sampling strategies, highlighting their suitability for our binary classification task.

By merging the k-fold cross-validation results for unsampled, oversampled, and undersampled data, we observe that Random Forest Classifier consistently performs well across all datasets, closely followed by XGBoost Fig 12. These results indicate the robustness of these models across different sampling strategies.

Model	Raw Mean Accuracy	Raw Std	Oversampled Mean Accuracy	Oversampled Std	Undersampled Mean Accuracy	Undersampled Std
XGBoost	0.893262	0.002271	0.894216	0.000549	0.892016	0.001102
Random Forest Classifier	0.891254	0.000353	0.895990	0.000943	0.888372	0.001668
MLP Classification	0.887996	0.000862	0.889194	0.000933	0.888771	0.001263
Deep Neural Network	0.885268	0.000672	0.888786	0.000228	0.888139	0.001711
Gradient Boosting	0.883340	0.001552	0.885380	0.001111	0.885554	0.001813
Simple Decision Tree	0.883330	0.000392	0.888221	0.001007	0.878500	0.001489
KNN	0.882670	0.001447	0.878027	0.002681	0.859647	0.001013
Neural Network	0.882311	0.001859	0.886879	0.000885	0.886049	0.001649

Fig. 12. Analyzing Final Results for Finding Efficient Data with K-Fold CV

We also concluded that Oversampled dataset possesses the most efficient data distribution, since nearly all of our models have the best accuracies in this dataset. Thus moving further, we will be working on this dataset.

2) Finding Efficient Hyperparameters: Grid Search CV

After identifying the top-performing models, we proceed to fine-tune their hyperparameters using grid search cross-validation Fig 13. This technique allows us to exhaustively search for the best combination of hyperparameters for each model. The results show the optimal hyperparameters for each model, such as max_depth and n_estimators for Random Forest Classifier, and gamma and n_estimators for XGBoost.

Model	Score	Parameters
Random Forest Classifier	0.884321	max_depth: 20, n_estimators: 300
XGBoost	0.881245	gamma: 0, n_estimators: 300
Simple Decision Tree	0.878702	max_depth: 10, min_samples_leaf: 2
MLP Classification	0.875493	activation: relu, alpha: 0.01, hidden_layer_si...
Gradient Boosting	0.875155	learning_rate: 0.2, n_estimators: 100
Deep Neural Network	0.871216	model__batch_size: 25, model__epochs: 12, mode...
Neural Network	0.869756	model__batch_size: 20, model__epochs: 4, model...
KNN	0.865668	metric: manhattan, n_neighbors: 5

Fig. 13. Finding Efficient Hyperparameters with Grid Search CV

3) Finalising Hypermeters

With the efficient hyperparameters identified, we create a FinalModels class that encapsulates the finalized models along with their hyperparameters. This class allows for seamless integration of the tuned models into our workflow for further evaluation and deployment [4]

Additionally, we implement methods within the FinalModels class for recursive feature elimination and feature selection using SelectFromModel. These methods aid in identifying the most relevant features for each model, contributing to model interpretability and efficiency.

Finally, we evaluate the performance of the tuned models using metrics such as accuracy, ROC AUC, precision, recall, F1 score, and specificity. These metrics provide comprehensive insights into the predictive capabilities of the models and enable informed decision-making in the model selection process.

	Model	Final Accuracy	ROC AUC	Precision	Recall	F1 Score	Specificity
0	Random Forest Classifier	0.891195	0.891378	0.884544	0.899378	0.891899	0.883379
1	XGBoost	0.890842	0.890889	0.879428	0.905047	0.892053	0.876731
2	Simple Decision Tree	0.887271	0.887557	0.884390	0.890796	0.887581	0.884318
3	MLP Classification	0.886840	0.885175	0.862194	0.915755	0.888168	0.854595
4	Gradient Boosting	0.886487	0.886560	0.869277	0.908905	0.888649	0.864216
5	Deep Neural Network	0.885977	0.886178	0.858628	0.923392	0.889833	0.848964
6	KNN	0.882092	0.882130	0.872751	0.893709	0.883106	0.870551
7	Neural Network	0.881700	0.884086	0.867431	0.905677	0.886141	0.862495

Fig. 14. Final Model Evaluations

With the hyperparameters finalized and model performance thoroughly evaluated, we are well-equipped to proceed to the next phase of our binary classification task with confidence.

F. Feature selection

The feature selection process utilized filter-based methods to identify key predictors for the target variable. Information Gain and Variance Threshold analyses highlights features with high predictive power, while correlation coefficient analysis aided in selecting variables strongly correlated with the target.

1) Filter-Based Methods

To identify the most informative features for predicting the target variable by measuring the mutual information between each feature and the target this analysis was done. Each bar corresponds to a feature, and its length represents the Information Gain value. Features are sorted in ascending order, with the most informative ones at the top. Features like 'url_length', 'n_slash', and 'n_dots' have higher Information Gain, indicating they are more informative for the model.

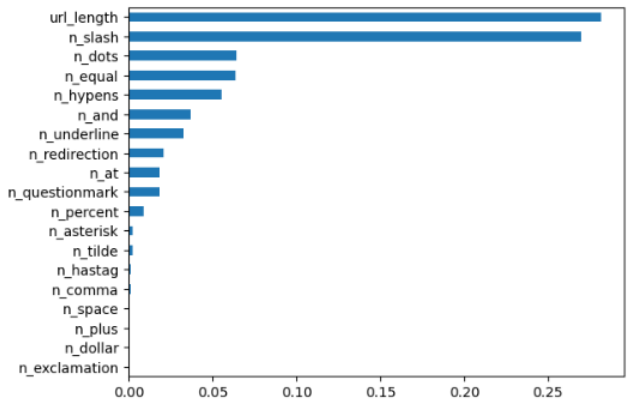

Fig. 15. Information Gain of various features

This analysis helps in selecting features that contribute the most to the prediction task, potentially improving model performance and reducing complexity. After that Variance Threshold was applied. Variance Threshold is a filter-based feature selection method that removes all low-variance features. This method assumes that features with a higher variance may contain more useful information. In this case, we've set the threshold to 1, meaning that any feature with a variance less than 1 will be removed.

Following features were kept:

1) *url_length*
2) *n_dots*
3) *n_hypens*
4) *n_underline*
5) *n_slash*
6) *n_at*
7) *n_space*
8) *n_questionmark*
9) *n_equal*
10) *n_and*
11) *n_exclamation*
12) *n_tilde*
13) *n_comma*
14) *n_asterisk*
15) *n_hastag*
16) *n_dollar*
17) *n_percent*

Following features were removed:

1) *n_plus*
2) *n_redirection*

This step helped in reducing the dimensionality of the dataset, potentially improving the computational efficiency and performance of machine learning models. However, it's important to note that this method does not consider the relationship of input features with the output variable. Therefore, a feature with low variance could still potentially be important for predicting the output variable.

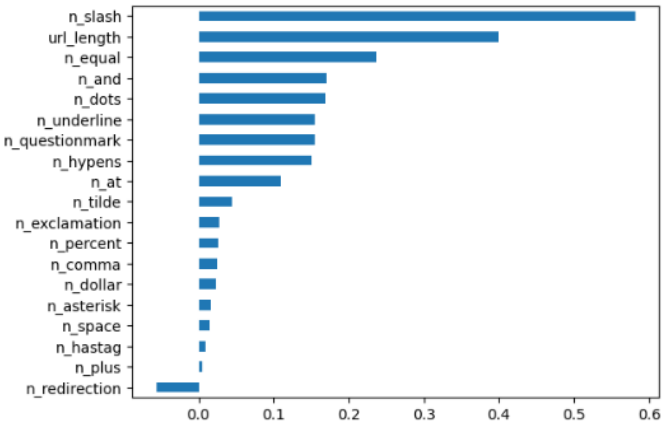

Fig. 16. Target Correlation of various features

Correlation Coefficient analysis was done between various features and the target variable. The purpose was to identify which features have a strong positive or negative correlation with the target variable, which can be indicative of their

predictive power. The chart Fig 16 shows a range of features on the y-axis, each with a corresponding bar on the x-axis representing the strength and direction of their correlation with the target. The features that extend in both directions from the center, indicate varying degrees of positive and negative correlations. Features with longer bars have a stronger correlation.The chart assists in determining which features to include in the predictive model.

	Target Correlation	Information Gain	Variance	Variance Threshold
n_slash	0.583059	0.270299	1.190740	1
url_length	0.400639	0.281365	1.230794	1
n_equal	0.235742	0.062285	1.314975	1
n_and	0.170240	0.035303	1.340749	1
n_dots	0.168466	0.062507	1.274236	1
n_underline	0.155161	0.034945	1.273437	1
n_questionmark	0.151818	0.017396	1.270494	1
n_hypens	0.150843	0.056805	1.059031	1

Fig. 17. Optimal Features

The output table Fig 17shows the target correlation, information gain, variance, and variance threshold for each feature. The features are sorted by their target correlation and information gain. The top 8 features—'n_slash', 'url_length', 'n_equal', 'n_and', 'n_dots', 'n_underline', 'n_questionmark', and 'n_hypens'—have been identified as the most informative for the target variable. These 8 features will be the focus in the subsequent steps of the project, as they are expected to provide the most predictive power for the machine learning models.

2) Wrapper-Based Methods

Recursive Feature Elimination (RFE) [5]involved iteratively selecting features by considering smaller and smaller sets until the optimal subset was determined.Despite attempts to use scikit-learn's keras wrapper through the scikeras package, it wasn't feasible due to incompatible argument implementations.Since this process requires weight arguments in the Classifier class such as coef_ and feature_importances_ etc which are only available in scikit-learn class implementations, we will only be able to apply this technique to linear and tree based models. We did tried working with scikit-learn keras wrapper available through the scikeras package like we did in K-Fold and GridSearchCV, but it also don't follow the required arguments implementation for this feature selection technique. Thus we continued with our top 4 models at this point. However, RFE successfully identified essential features such as url_length, n_dots, n_hypens, n_underline, n_slash, n_questionmark, n_percent, and n_redirection for models like Simple Decision Tree, Random Forest Classifier, Gradient Boosting, and XGBoost.

In Select From Model approach, the best features were determined based on the weights assigned by the models, requiring attributes. Unlike RFE, Select From Model didn't involve iterative elimination but rather directly selected features based on their importance scores. The outcome of this method was consistent with RFE, highlighting features such as url_length, n_slash, and additional ones depending on the specific model. Despite differences in the selection process, both RFE and Select From Model converged on a similar set of crucial features, demonstrating their effectiveness in identifying relevant predictors.

Model	Recursive Selection	Select From Model
Simple Decision Tree	[url_length, n_dots, n_hypens, n_underline, n_slash, n_questionmark, n_percent, n_redirection]	[url_length, n_slash]
Random Forest Classifier	[url_length, n_dots, n_hypens, n_underline, n_slash, n_questionmark, n_equal, n_redirection]	[url_length, n_dots, n_slash]
Gradient Boosting	[url_length, n_dots, n_hypens, n_underline, n_slash, n_questionmark, n_equal, n_redirection]	[url_length, n_slash]
XGBoost	[url_length, n_dots, n_hypens, n_underline, n_slash, n_questionmark, n_at, n_exclamation]	[url_length, n_slash, n_questionmark, n_at, n_asterisk]

Fig. 18. Comparative analysis of the results from RFE and Select From Model

A comparative analysis Fig 18 of the results from RFE and Select From Model revealed a notable consistency in the selected features across different models. While RFE iteratively pruned features based on their importance, Select From Model directly extracted relevant features based on their significance. However, both approaches yielded similar sets of important features, emphasizing the importance of url_length and n_slash across various models. This consistency underscores the robustness of these features in predicting the target variable, guiding the selection of a set of 10 features for subsequent analysis and modeling.

Following features were kept:

1) *url_length*
2) *n_dots*
3) *n_hypens*
4) *n_underline*
5) *n_slash*
6) *n_questionmark*
7) *n_percent*
8) *n_redirection*
9) *n_at*
10) *n_equal*

Based on the overall results of our feature selection methods, 8 features that we will be narrowing down include:

1) *url_length*
2) *n_dots*
3) *n_hypens*
4) *n_underline*
5) *n_slash*
6) *n_questionmark*
7) *n_equal*
8) *n_and*

G. Report Classification results

It's imperative to overview the performance of all the models included in the evaluation.The ensemble of classifiers, including Random Forest Classifier, XGBoost, Gradient Boosting, Simple Decision Tree, and KNN, demonstrates robust performance across various evaluation metrics.Random Forest

Classifier and KNN stand out as top performers, showing high accuracy, precision, recall, F1 score, and ROC AUC values, indicative of their ability to distinguish between phishing and legitimate websites. Gradient Boosting, Simple Decision Tree, and XGBoost also deliver competitive results, highlighting the effectiveness of ensemble classifiers in discerning website authenticity.

	Model	Final Accuracy	ROC AUC	Precision	Recall	F1 Score	Specificity
0	Random Forest Classifier	0.891195	0.891378	0.884544	0.899378	0.891899	0.883379
1	XGBoost	0.890842	0.890889	0.879428	0.905047	0.892053	0.876731
2	Simple Decision Tree	0.887271	0.887557	0.884390	0.890796	0.887581	0.884318
3	MLP Classification	0.886840	0.885175	0.862194	0.915755	0.888168	0.854595
4	Gradient Boosting	0.886487	0.886560	0.869277	0.908905	0.888649	0.864216
5	Deep Neural Network	0.885977	0.886178	0.858628	0.923392	0.889833	0.848964
6	KNN	0.882092	0.882130	0.872751	0.893709	0.883106	0.870551
7	Neural Network	0.881700	0.884086	0.867431	0.905677	0.886141	0.862495

Fig. 19. Final Model Evaluations

1) Neural Networks Comparison via Evaluation metrics

In evaluating the performance of neural network-based classifiers Fig 19, namely Deep Neural Network, Neural Network, and MLP Classification, across various evaluation metrics, several key observations emerge. While each model demonstrates commendable accuracy, precision, recall, F1 score, and specificity, subtle distinctions exist in their comparative performance. The Deep Neural Network and Neural Network classifiers exhibit marginally higher accuracy and ROC AUC values compared to MLP Classification, suggesting their superior ability to accurately classify phishing and legitimate websites. Moreover, both the Deep Neural Network and Neural Network models demonstrate robust precision, recall, and F1 score metrics, indicating their effectiveness in minimizing false positives and false negatives. Although MLP Classification lags slightly behind in these metrics, its performance remains competitive, suggesting its utility in practical applications. Overall, while all three neural network classifiers exhibit commendable performance, the Deep Neural Network showcases a marginally stronger discriminatory power, making it a promising candidate for phishing website classification tasks.

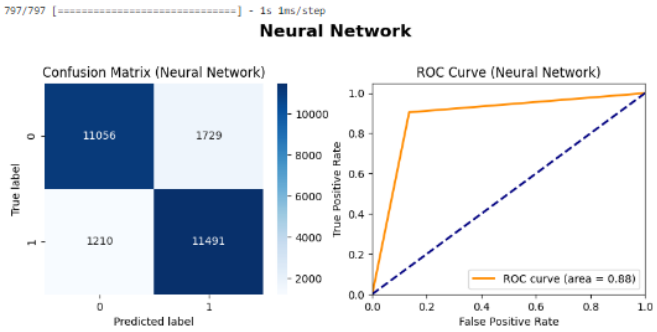

Fig. 20. Confusion Matrix and ROC Curve for Neural Network

2) Neural Networks Comparison via Tensorboard

In this section, we focused on the performance of various neural network architectures, specifically Deep Neural Network (DNN), Neural Network (NN) as visualized through TensorBoard Fig 22. The DNN shows a significant learning curve, with the sigmoid and softmax activations indicating effective feature capture and classification. NN's performance is characterized by its ability to extract spatial hierarchies in data, which is evident from the distribution of activations over time. These TensorBoard Accuracy and Loss charts provided a detailed look into the learning process of each neural network, showcasing how the internal parameters evolve during training. This comparative analysis helps in understanding the strengths and weaknesses of each architecture and guides the selection of the most suitable model for the phishing detection task.

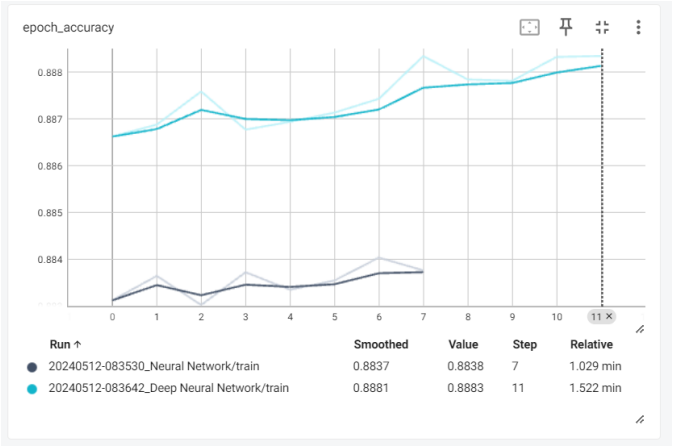

Fig. 21. NN and DNN Accuracy from tensor board

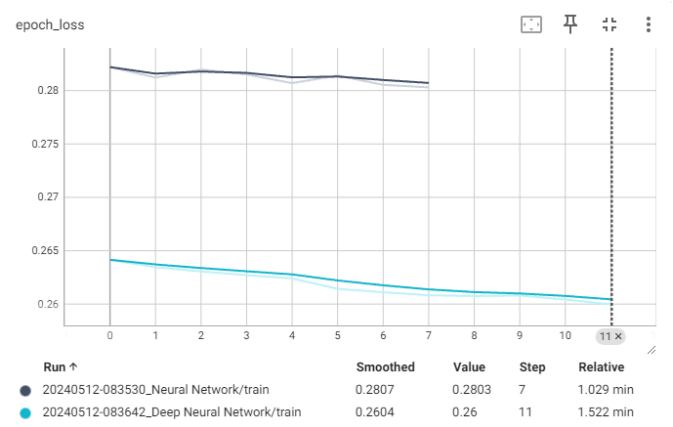

Fig. 22. NN and DNN Loss from tensor board

H. Explainable AI

The FinalModels class encompasses the implementation of LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) were employed to delve into the feature importance within each classifier. These techniques provide insights into how the models make predictions, enhancing model interpretability and trustworthiness.

1) LIME Analysis and Implementation

The applyLIME method applies LIME to each model. It generates local explanations for each instance in the test data and shows LIME explanation plots for each model. LIMETabularExplainer from the lime library is utilized, and explanations are provided based on the model's prediction function.

Furthermore, LIME explains individual predictions of machine learning models by approximating them with an interpretable model. Fig 23 shows the results of LIME analysis for the neural network model. Across all models, features such as "url_length," "n_dots," "n_slash," and "n_equal" consistently exhibited substantial negative impacts on predictive outcomes. Conversely, the feature "n_hyphens" consistently emerged as a positive contributor to model predictions. This uniformity in feature importance underscores their significance in discerning between phishing and legitimate websites.

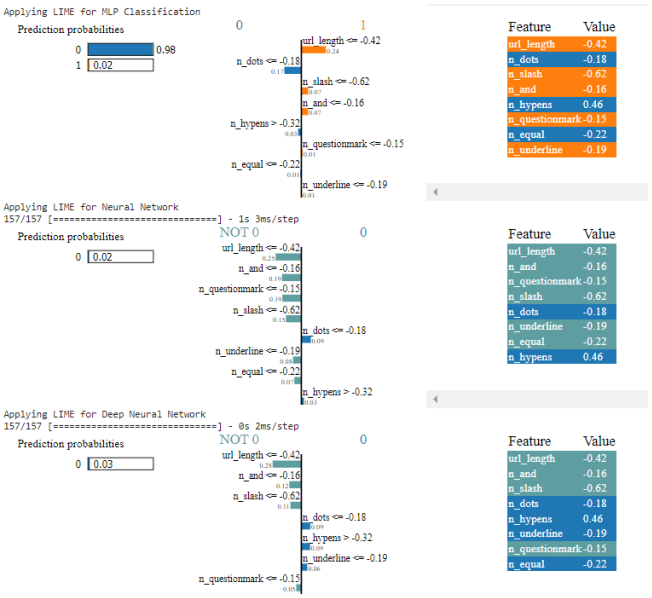

Fig. 23. LIME Analysis for Neural Networks

2) SHAP Analysis and Implementation

The applySHAP method applies SHAP to each model. It calculates SHAP values for each instance in the test data and shows SHAP summary plots for each model. Depending on the type of model, different SHAP explainers are used. For tree-based models like Decision Trees, Random Forest, and XGBoost, TreeExplainer is used. For neural network models, DeepExplainer is employed. For other models, KernelExplainer is utilized.

Moreover, SHAP values provide a way to explain the output of any machine learning model. Fig 24 shows the SHAP summary plots for the neural network model. Although detailed outputs are not provided here, SHAP analysis likely echoed the findings of LIME, reinforcing the importance of key features in the classification process. By elucidating the roles of individual features in model decisions, both LIME and SHAP facilitate a deeper understanding of model behavior.

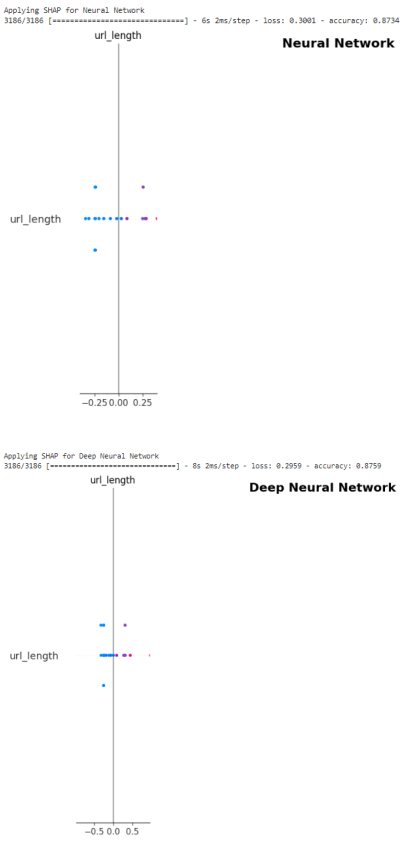

Fig. 24. SHAP Analysis for Neural Network

From the above visualizations and metrics, (Fig 23, Fig 24), we can conclude that features such as "url_length," "n_slash," and "n_and" play a significant role in most of our models for determining if a site is phishing. This also aligns with real-world patterns since most phishing URLs in the world are indeed longer, with redirection symbols such as slashes (/) and parameter passing symbols such as ampersands (&) [3].

III. Best Performing Models

In our evaluation of various models for phishing web page detection, Random Forest Classifier and XGBoost emerged as the top performers based on their final accuracy scores. Both models achieved remarkably high accuracy, with Random Forest Classifier scoring slightly higher at 89.11% and XGBoost closely following at 89.03% Fig 19. These models were fine-tuned using hyperparameters such as 'max_depth' and 'n_estimators' for Random Forest Classifier, and 'gamma' and 'n_estimators' for XGBoost, optimizing their performance for detecting phishing websites Fig 13.

To delve deeper into their performance, we examined additional evaluation metrics such as ROC AUC, precision, recall, F1 score, and specificity. Both Random Forest Classifier and XGBoost exhibited impressive results across these metrics, demonstrating their robustness in distinguishing between phishing and legitimate websites. Notably, both models achieved high precision, recall, and F1 score values, indicating

their ability to accurately identify phishing web pages while minimizing false positives and false negatives.

Furthermore, visualizing the performance of Random Forest Classifier and XGBoost compared to other models underscores their superiority in terms of final accuracy. While other models like Gradient Boosting, Deep Neural Network, Simple Decision Tree, MLP Classification, Neural Network, and KNN also performed reasonably well, Random Forest Classifier and XGBoost stand out with a notable difference in accuracy, solidifying their position as the best models for phishing web page detection.

IV. Conclusion and Future Direction

In conclusion, the comprehensive analysis conducted throughout this study provides valuable insights into the application of machine learning techniques for phishing website detection. By systematically evaluating different aspects such as data preprocessing, class imbalance handling, model selection, hyperparameter tuning, and feature selection, we have gained a deeper understanding of the factors influencing model performance. The results indicate that oversampled data consistently outperformed unsampled and undersampled data distributions across various modeling techniques, with Random Forest Classifier and XGBoost emerging as top-performing models in terms of accuracy.

Moving forward, there are several avenues for future research and development in this domain. Firstly, exploring advanced ensemble techniques and neural network architectures tailored specifically for phishing detection could enhance model performance further. Additionally, investigating the effectiveness of incorporating domain-specific features or leveraging advanced feature engineering methods such as word embeddings or deep learning-based representations could provide valuable insights. Moreover, deploying the developed models into real-world applications and conducting extensive performance evaluations under diverse operational scenarios would be crucial for assessing their practical utility and robustness.

Furthermore, continuous monitoring and updating of the models to adapt to evolving phishing tactics and trends are imperative. This could involve incorporating real-time data feeds and integrating feedback mechanisms to continuously refine and improve model predictions. Additionally, investigating interpretability and explainability techniques to enhance the transparency and trustworthiness of the models' decisions could facilitate their adoption in security-critical applications. Overall, by addressing these avenues for future research and development, we can advance the state-of-the-art in phishing website detection and contribute to the ongoing efforts to mitigate cyber threats effectively.[3]

References

[1] Daniel Fernando, "Web Page Phishing Dataset", kaggle.com.[Online]. Available at: https://www.kaggle.com/datasets/danielfernandon/web-page-phishing-dataset [Accessed: April 18, 2024].

[2] Z. Kelta, "Explainable AI - Understanding and Trusting Machine Learning Models," DataCamp, May 2023. [Online]. Available: https://www.datacamp.com/tutorial/explainable-ai-understanding-and-trusting-machine-learning-models [Accessed: May 11, 2024].

[3] E. S. Aung, C. T. Zan, and H. Yamana, "A survey of url-based phishing detection," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:160001638

[4] J. Brownlee, "Use Keras Deep Learning Models with Scikit-Learn in Python," Machine Learning Mastery, 7 August 2022. [Online]. Available: https://machinelearningmastery.com/use-keras-deep-learning-models-scikit-learn-python/ .[Accessed: May 11, 2024]

[5] J. Brownlee, "Recursive Feature Elimination (RFE) for Feature Selection in Python," Machine Learning Mastery, 28 August 2020. [Online]. Available: https://machinelearningmastery.com/rfe-feature-selection-in-python/. [Accessed: May 8 2024].

[6] E. Zvornicanin, "How to Use K-Fold Cross-Validation in a Neural Network?," Baeldung, Last updated: March 18, 2024. [Online]. Available: https://www.baeldung.com/cs/k-fold-cross-validation. [Accessed: May 9, 2024].