Detecting Phishing URL's

Rohith Pavuluru
ropavu

Prajodh Pragath Sunder
ppsunder

project-ropavu-ppsunder

Abstract

The frequency of fraudulent web addresses, or "phishing URLs," which trick people into disclosing sensitive information, is covered in this abstract. Phishing attacks pose a serious risk, and hackers always devise new strategies to deceive people. The growing significance of identifying and categorizing such dangerous URLs using machine learning techniques is underscored. The ability to recognize and react to phishing scams is greatly improved by machine learning. The utilization of advanced machine learning algorithms and large datasets provides a flexible method for examining URL attributes and actions in real-time, adjusting to new strategies used by online threats.

Keywords

Phishing, Malware, Defacement, Benign, Meta Characters, Phishing detection, Machine learning, URL classification, Gradient Boosting, Logistic regression, Support Vector Machines, Neural Networks, Contextual information, Cybersecurity.

1 Introduction

Many web addresses are URLs for websites that are intentionally created and utilized maliciously to trick and mislead people into disclosing their sensitive and personal data. We refer to these URLs as phishing URLs. Phishing attacks pose a serious risk to both individuals and organizations because hackers are using more advanced strategies to trick users into disclosing private information. The significance of utilizing machine learning algorithms to classify harmful phishing URLs has increased.

To categorize URLs as phishing, defacement, malware, or benign, we explore various machine-learning techniques. Using a large dataset of URLs—including legitimate and known phishing sites, we extract a range of properties, including URL structure, content, and contextual data. Our goal is to construct a classification model namely Logistic regression, Support vector machine, and Gradient Boosting with good performance metrics. We also examine the adaptability of our models in dealing with changing phishing techniques and evaluate their effectiveness in real-world scenarios.

This project contributes to the ongoing effort to enhance cybersecurity and protect users from phishing threats. Ultimately, this work serves as a stepping stone in developing more effective and automated tools to combat the ever-evolving landscape of phishing attacks.

Previous work

Many open-source projects have a large collection of harmful URLs that are continually updated with new phishing or malicious web addresses by the contributors [10]. There are also cloud-based solutions that provide DNS-level security and machine learning models to safeguard enterprises from phishing attacks [3]. A portion of the solutions now in use make use of API's solutions, which use a pre-trained machine learning model to predict the maliciousness of a web URL [7]. Lastly, there are reputation-based systems that use machine learning to detect phishing URLs, which are already part of the operating systems [4].

2 Methods

The analysis of URLs involves several important procedures. First, relevant information such as the path, query parameters, and domain name will have to be extracted from the URLs. To find patterns, connections, and anomalies in the extracted features, exploratory data analysis (EDA) is performed after feature extraction. The next steps involve data preprocessing. The data is scaled to ensure that all features contribute equally to the analysis, and it is then split into training, validation, and test sets to perform model development and evaluation. Feature engineering techniques are applied to enhance the dataset's predictive power, while dimensionality reduction methods help improve the model's efficiency.

To select the most suitable classification model, a grid search with cross-validation (GridSearch CV) is performed, testing various classification estimators(Logistic regression, Support vector machine, and Gradient Boosting) and their hyperparameters. The best-performing estimator is chosen, and its parameters are fine-tuned to optimize model performance. Apart from this, a Tensorflow deep learning model was also trained. Following model training and fine-tuning, the model's performance is rigorously assessed using multiple metrics, including precision, accuracy, recall, and more. This evaluation is conducted on the test set to ensure the model's generalization to unseen data.

Finally, a user-friendly function is developed, allowing users to input a new URL for analysis. The function makes use of the trained model to predict the nature of the URL, whether it is phishing, defacement, or benign, providing valuable insights for online security and threat detection.

Feature Engineering

Through online searches and research papers [1], we were able to learn how metacharacters were important in extracting features from web URLs. In this stage, new features based on the analysis of the web URLs were created. These features include counts of specific characters (such as '.', '-', '_', '/', '?', '=', '@', '&', '!', '', '—', '', ',', '+', '*', '#', '$', '%') in the URLs, the length of the URL, the length of the hostname, the type of domain name, and many more. These features are generated by applying custom count functions to the URL column using the `apply` method in pandas. Now, we have 30 columns/features to train our models.

Exploratory Data Analysis (EDA)

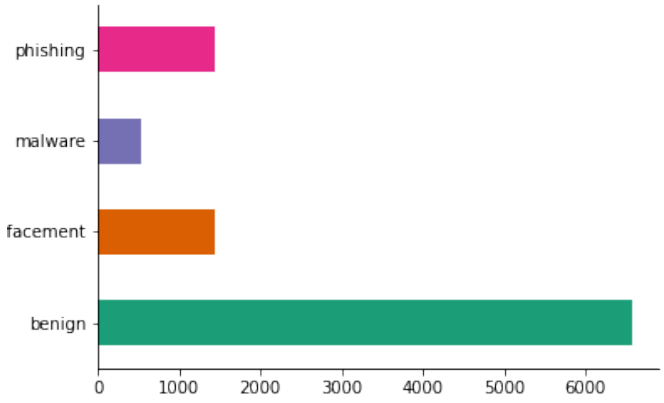


Figure 1: We can see with this bar graph how the dataset given to us is disproportional.

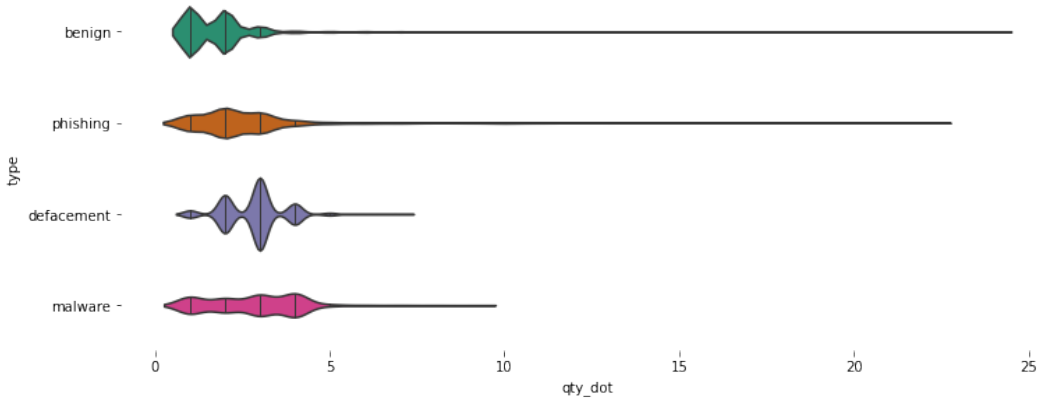


Figure 2: With the above violin plot, we can analyze how the quantity of dots in benign and phishing URLs is much higher than in defacement and malware.

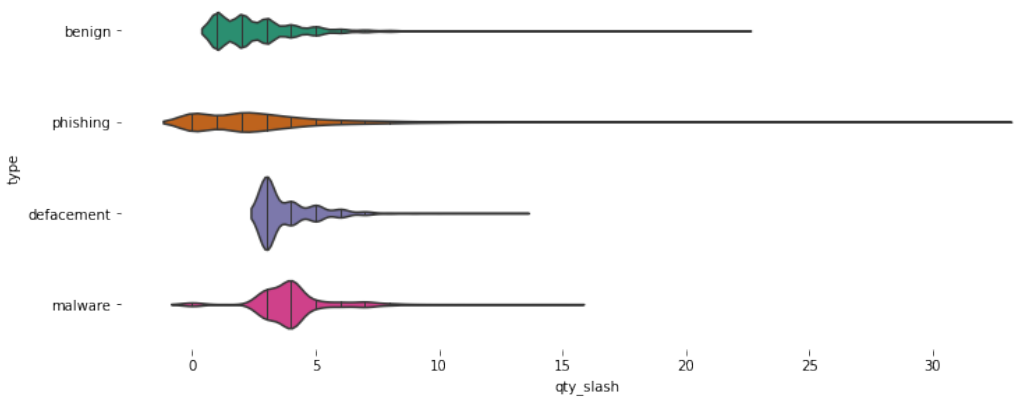


Figure 3: With the above violin plot we can analyse how quantity of slash is not the best feature as even thou the phishing has higher numbers of these they aren't good to distinguish between other classes.

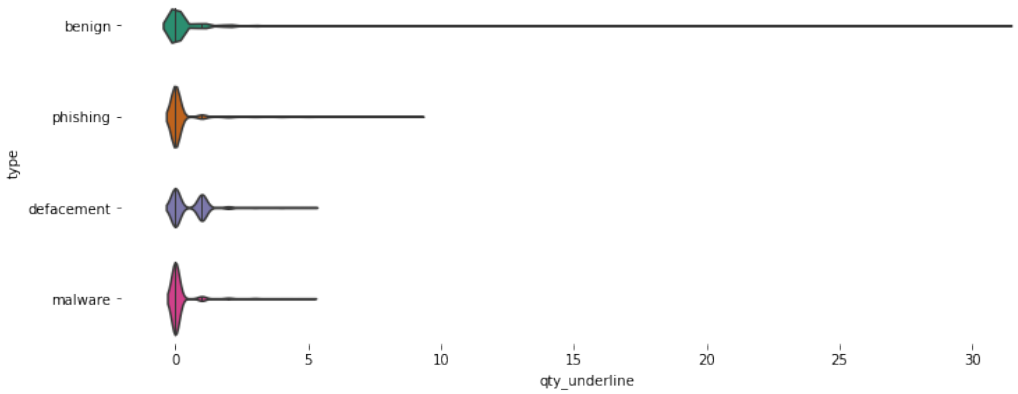


Figure 4: With the above violin plot, we can analyze how the quantity of the underline can be a very good feature to recognize a benign URL.

Types of Sampling

Random sampling is a statistical technique where each URL has an equal chance of being selected to form a sample [8]. Random sampling with replacement is similar to the random sampling technique in which individual URLs are selected randomly from a dataset, and after each selection, the item is placed back into the sampling pool. Weighted sampling is a method of randomly selecting URLs from a population, where each URL has a different probability of being chosen, and the probability is determined by the distribution of the type of URL in the dataset. Stratified sampling is a statistical technique where the dataset is divided into subgroups or strata, and then random samples are taken from each URL type [6]. This method ensures that each subgroup is adequately represented in the final sample. The use of these methods to choose URLs from the dataset was done using the in-built sampling function of Pandas.

Types of Models

For each of the sampling techniques, we ran three models. The first was a pipeline that ran a grid search CV on the three estimators: logistic regression, support vector machines, and gradient boosting with 3-fold cross-fold validation to detect the best estimator [9]. After this, the best estimator is chosen, which was gradient boosting [5] in our case for all the experiments, and we hyperparameterize it with the number of trees, max depth of trees, and learning rate to get the most out of the models. Lastly, we trained a deep learning model with three hidden layers using the TensorFlow API [2].

Inferencing Function

We created a Python function named inference that takes a URL string as input and performs various feature extraction operations on it to prepare the data for prediction using our final gradient boosting model and deep learning model, which were trained on 100,000 data points from the dataset. This function was created to see how these trained models work in real-world scenarios.

3 Results

Random sampling

```
Best Hyperparameters: {'classifier__learning_rate': 0.2, 'classifier__max_depth': 7, 'classifier__n_estimators': 200}
Best Estimator: Pipeline(steps=[('std', MinMaxScaler()),
                ('classifier',
                 GradientBoostingClassifier(learning_rate=0.2, max_depth=7,
                                            n_estimators=200))])
The time taken to select best estimator using this instance of GridsearchcV is 571.5926604270935
Test Accuracy: 0.901
```

Figure 5.1: The best estimator for random sampling was gradient boosting with an accuracy of 0.901

```
classification report

              precision    recall  f1-score   support

      benign       0.92      0.95      0.94      6624
  defacement       0.89      0.87      0.88      1452
     malware       0.94      0.86      0.90       500
    phishing       0.78      0.69      0.73      1424

    accuracy                           0.90     10000
   macro avg       0.88      0.85      0.86     10000
weighted avg       0.90      0.90      0.90     10000
```

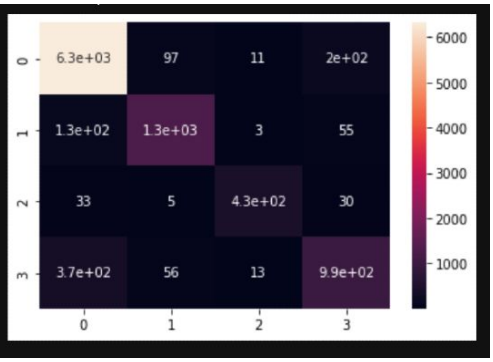
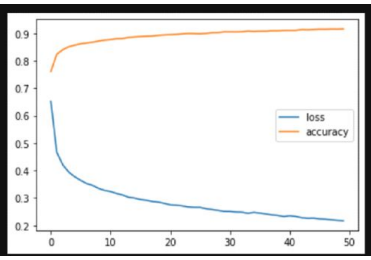


Figure 5.2: Classification report and confusion matrix for hyperparameter-tuned gradient boosting for random sampling which shows that the accuracy metrics for phishing URLs are low

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93      6624
           1       0.90      0.84      0.87      1452
           2       0.76      0.71      0.73      1424
           3       0.94      0.69      0.80       500

    accuracy                           0.89     10000
   macro avg       0.88      0.80      0.83     10000
weighted avg       0.89      0.89      0.89     10000
```



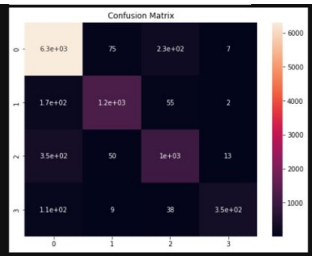


Figure 5.3: Classification report, loss curves, and confusion matrix for a deep learning model for random sampling shows that the accuracy metrics for phishing and malware URLs are low

The optimal hyperparameters found include a learning rate of 0.2, a maximum depth of 7 for the trees, and 200 estimators. The optimal estimator for random sampling is gradient boosting, achieving a high accuracy of 0.901. However, a classification report and confusion matrix reveal low-accuracy metrics for phishing URLs (Figure 5.2). Additionally, a deep learning model for random sampling exhibits low accuracy metrics for both phishing and malware URLs, as illustrated in a classification report, loss curves, and confusion matrix (Figure 5.3). Despite the high accuracy of gradient boosting, the specific analysis indicates challenges in accurately classifying phishing and malware URLs within the random sampling context.

Weighted sampling

```
Best Hyperparameters: {'classifier__learning_rate': 0.01, 'classifier__max_depth': 3, 'classifier__n_estimators': 50}
Best Estimator: Pipeline(steps=[('std', MinMaxScaler()),
                ('classifier',
                 GradientBoostingClassifier(learning_rate=0.01,
                                            n_estimators=50))])
The time taken to select best estimator using this instance of GridsearchcV is 177.8593671321869
Test Accuracy: 1.0
```

Figure 6.1: The best estimator for weighted sampling was gradient boosting with an accuracy of 1.0

```
classification report

              precision    recall  f1-score   support

      benign       1.00      1.00      1.00      9020
  defacement       1.00      1.00      1.00       459
     malware       1.00      1.00      1.00        51
    phishing       1.00      1.00      1.00       470

    accuracy                           1.00     10000
   macro avg       1.00      1.00      1.00     10000
weighted avg       1.00      1.00      1.00     10000
```

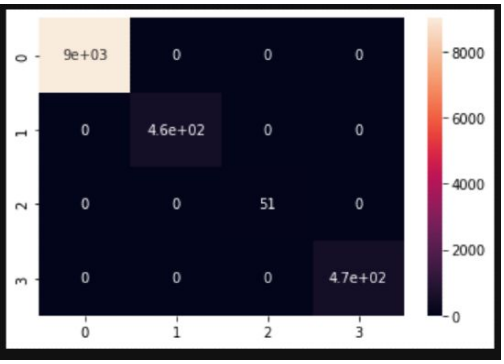
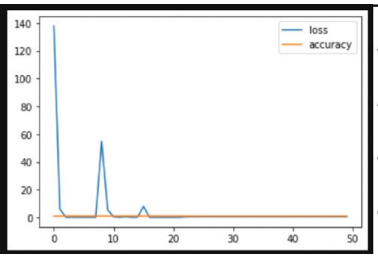


Figure 6.2: Classification report and confusion matrix for hyperparameter-tuned gradient boosting for Weighted Sampling shows that the accuracy metrics for all types perform well

```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      9021
           1       1.00      0.00      0.00       477
           2       1.00      0.00      0.00       450
           3       1.00      0.00      0.00        52

    accuracy                           0.90     10000
   macro avg       0.98      0.25      0.24     10000
weighted avg       0.91      0.90      0.86     10000
```



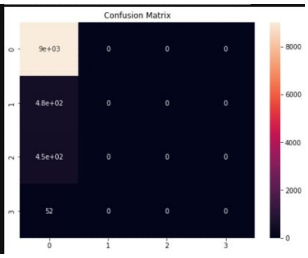


Figure 6.3: Classification report, loss curves, and confusion matrix for a deep learning model for Weighted Sampling shows some anomaly with this sampling technique as recall and f1-score is zero

The most effective estimator for weighted sampling is gradient boosting, achieving perfect accuracy (1.0). The optimal parameters are a learning rate of 0.2, a max depth of 7 for the trees in the model, and 200 estimators (trees). The classification report and confusion matrix in Figure 6.2 demonstrate strong performance across all types. However, Figure 6.3 reveals anomalies in a deep learning model for weighted sampling, where recall and f1-score are zero, suggesting challenges in correctly identifying certain categories. Despite the exceptional accuracy of gradient boosting, the deep learning model exhibits limitations in recall and f1-score for weighted sampling, indicating issues with sampling technique. This points to an imbalanced dataset or a model that predominantly predicts the majority class, disregarding others.

Random sampling with replacement

```
Best Hyperparameters: {'classifier__learning_rate': 0.2, 'classifier__max_depth': 7, 'classifier__n_estimators': 200}
Best Estimator: Pipeline(steps=[('std', MinMaxScaler()),
                ('classifier',
                 GradientBoostingClassifier(learning_rate=0.2, max_depth=7,
                                            n_estimators=200))])
The time taken to select best estimator using this instance of GridsearchcV is 562.1651005744934
Test Accuracy: 0.9059
```

Figure 7.1: The best estimator for Random sampling with replacement was gradient boosting with an accuracy of 0.9059

classification report

	precision	recall	f1-score	support
benign	0.71	0.96	0.82	2510
defacement	0.93	0.87	0.90	2512
malware	0.98	0.88	0.93	2545
phishing	0.87	0.71	0.78	2433
accuracy			0.86	10000
macro avg	0.87	0.86	0.86	10000
weighted avg	0.87	0.86	0.86	10000

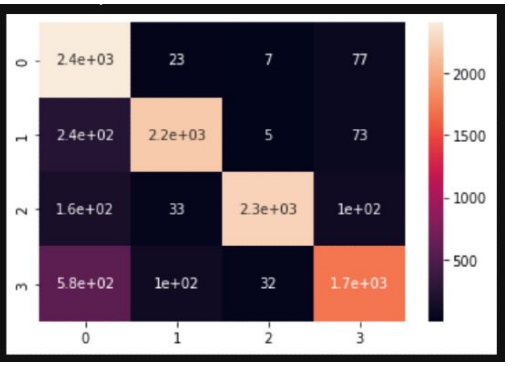
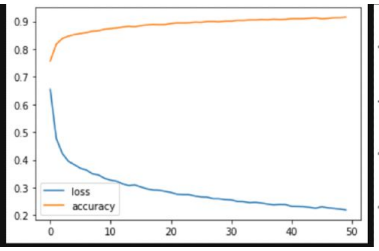


Figure 7.2: Classification report and confusion matrix for hyperparameter-tuned gradient boosting for Random sampling with replacement which shows that the accuracy metrics for phishing URLs are low

	precision	recall	f1-score	support
0	0.92	0.94	0.93	6582
1	0.90	0.84	0.87	1471
2	0.74	0.72	0.73	1438
3	0.86	0.87	0.87	509
accuracy			0.89	10000
macro avg	0.85	0.84	0.85	10000
weighted avg	0.89	0.89	0.89	10000



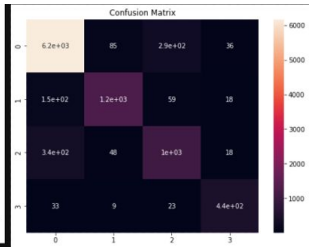


Figure 7.3: Classification report, loss curves, and confusion matrix for a deep learning model for Random sampling with replacement shows that the accuracy metrics for phishing and malware URLs are low

The optimal parameters are a learning rate of 0.2, a maximum depth of 7 for the trees in the model, and 200 estimators (trees). It shows the test accuracy achieved by the model with these hyperparameters, which is 0.9059. However, a detailed analysis in Figure 7.2 reveals low-accuracy metrics for phishing URLs in the classification report and confusion matrix. Similarly, a deep learning model for random sampling with replacement, depicted in Figure 7.3, shows low accuracy metrics for both phishing and malware URLs in the classification report, loss curves, and confusion matrix. Despite the high overall accuracy of gradient boosting, challenges persist in accurately classifying phishing and malware URLs within the context of random sampling with replacement.

Stratified sampling

```
Best Hyperparameters: {'classifier__learning_rate': 0.2, 'classifier__max_depth': 7, 'classifier__n_estimators': 200}
Best Estimator: Pipeline(steps=[('std', MinMaxScaler()),
                ('classifier',
                 GradientBoostingClassifier(learning_rate=0.2, max_depth=7,
                                            n_estimators=200))])
The time taken to select best estimator using this instance of GridsearchcV is 564.8320560455322
Test Accuracy: 0.8989
```

Figure 8.1: The best estimator for Stratified sampling was gradient boosting with an accuracy of 0.8989

	precision	recall	f1-score	support
benign	0.90	0.86	0.88	2510
defacement	0.91	0.94	0.92	2512
malware	0.95	0.94	0.95	2545
phishing	0.85	0.87	0.86	2433
accuracy			0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000

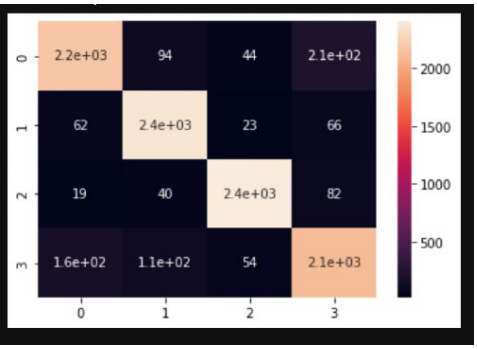
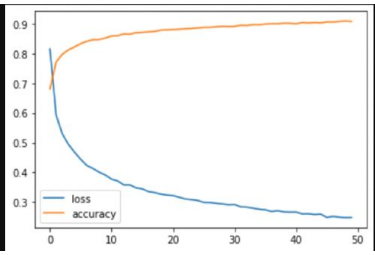


Figure 8.2: Classification report and confusion matrix for hyperparameter-tuned gradient boosting for Stratified sampling which performs equally in all accuracy metrics

	precision	recall	f1-score	support
0	0.89	0.83	0.86	2510
1	0.89	0.92	0.90	2512
2	0.82	0.85	0.83	2433
3	0.93	0.93	0.93	2545
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000



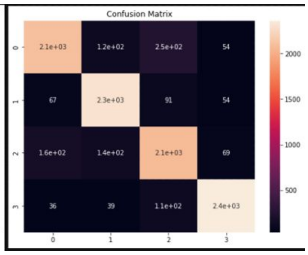


Figure 8.3: Classification report, loss curves, and confusion matrix for a deep learning model for Stratified sampling shows that the accuracy metrics for all URL types are high and equal

The best hyperparameters found for a Gradient Boosting Classifier—learning rate of 0.2, max depth of 7, and 200 estimators. The optimization took around 564.83 seconds, achieving a high test accuracy of 0.8989. A classification report lists precision, recall, and F1-scores for four classes, showing good performance across, particularly for class 3. The report indicates a balanced distribution of instances (support) among classes and overall high accuracy. The confusion matrix visualizes true vs. predicted classifications, with notable correct predictions along the diagonal and some misclassifications evident. The model demonstrates strong predictive capabilities but with some room for reducing errors, as shown by off-diagonal numbers in the confusion matrix.

Inference

```
inference("https://quillbot.com/grammar-check")

('benign', array(['phishing'], dtype=object))
```

Figure 9: Inferring from gradient boosting and deep learning models

4 Discussion

Both random sampling with and without replacement showed how the malware and phishing classes were hard to predict. The malware type as such had very low data points, whereas phishing has doubled the number of data points and still performed worse than malware, as shown in the Figures. 5.2 and 7.2. This represents how hard to detect and sophisticated these phishing URLs have become. The weighted sampling technique had some anomalies or issues as it had an overall accuracy metric of 1.00, as shown in Figure. 6.2, and had 0.00 accuracy for the deep learning model in Figure. 6.3.

Overall, the stratified sampling technique performs well for all the classes in the dataset. The reason is that all classes get equal proportions of training data, therefore making this disproportional shared data proportional. The best estimator was gradient boosting for all the sampling techniques, as it is a decision tree-based algorithm that we tuned on its hyperparameters. The deep learning models performed equally well. Finally, using our above knowledge, we created two final models with a hundred thousand data points and inferred them using a Python function that takes a URL and gives its class or type based on our model predictions. The deep learning model was misclassified, but the gradient boosting model classified the benign URL properly. This can be due to the models not training on more data points or requiring more epochs to converge.

5 Author Contribution

The project was done by a team of two: Rohith (ropavu) performed EDA, model training for weighted sampling and random sampling, and final gradient model training. Prajodh (ppsunder) performed feature engineering, model training from random sampling with replacement, and stratified sampling.and the final deep learning model The final inferencing function was a collaborative effort.

References

[1] A. Aljofey, Q. Jiang, A. Rasool, H. Chen, W. Liu, Q. Qu, and Y. Wang. An effective detection approach for phishing websites using url and html features. *Scientific Reports*, 13:10841, 2023.

[2] Cagatay Catal, Görkem Giray, Bedir Tekinerdogan, Sandeep Kumar, and Suyash Shukla. Applications of deep learning for phishing detection: a systematic literature review. *Knowledge and Information Systems*, 64:1457–1500, 2022.

[3] Inc. Cisco Systems. Cisco umbrella, 2023. https://umbrella.cisco.com/.

[4] Microsoft Corporation. Microsoft defender smartscreen, 2023. https://www.microsoft.com/en-us/wdsi/.

[5] Luca Di Persio and Nicola Fraccarolo. Energy consumption forecasts by gradient boosting regression trees. *Mathematics*, 11(5):1068, 2023.

[6] Roger Hillson, Joel D. Alejandre, Kathryn H. Jacobsen, Rashid Ansumana, Alfred S. Bockarie, Umaru Bangura, Joseph M. Lamin, and David A. Stenger. Stratified sampling of neighborhood sections for population estimation: A case study of bo city, sierra leone. *plos one*, 10(7):e0132850, 2015.

[7] Google LLC. Google safe browsing, 2021. https://developers.google.com/safe-browsing/.

[8] Zhaobin Mo, Xuan Di, and Rongye Shi. Robust data sampling in machine learning: A game-theoretic framework for training and validation data selection. *Games*, 14(1):13, 2023.

[9] Philipp Probst, Bernd Bischl, and Anne-Laure Boulesteix. Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(5):e1484, 2021.

[10] Maltego Technologies. Openphish, 2022. https://openphish.com/index.html.