Malicious websites classification

Shukai Ni

Data Science Initiative Master's Program

Brown University

https://github.com/9r0x/data1030_final

December 6, 2022

1 Introduction

Kaggle Dataset(Urcuqui, 2017): https://www.kaggle.com/datasets/xwolf12/malicious-and-benign-websites

1.1 Background

Cybersecurity threats have increased at an unprecedented rate. In 2021, Australia alone reported 71,299 incidents of phishing, usually in the form of fraudulent websites designed to trick users into revealing sensitive information, and 4.3 Million AUD lost(Scamwatch, 2022). The ability to distinguish malicious sites from benign sites protects inexperienced Internet users from the information or financial loss, enables organizations to identify viruses/attacks from external networks, and assists website owners in enhancing their site's design to improve their search engine ranking.

1.2 Dataset

This project(Ni, 2022) classifies websites as Malicious(type = 1) and Benign(type = 0) using the "Malicious and Benign Websites" dataset from Kaggle(Urcuqui, 2017). 1781 data points with 20 raw characteristics and one target variable(Type) are contained in this dataset. Features fall into two categories: The Application layer and the Network layer, as defined by the Open Systems Interconnection (OSI) model(Ames, 209): the former addresses what the user actually sees in their browser, whereas the latter handles the underlying data packets that are delivered to your computer. URL_LENGTH, CHARSET, CONTENT_LENGTH, and SERVER are examples of application features, whereas DISTREMOTETCP_PORT, APP_BYTES, and DNSQUERYTIMES are examples of network features.

The dataset was compiled from reputable sources, such as malwaredomainlist.com, and then filtered using another security application, such as VirusTotal. After initial filtering, Python scripts were utilized to confirm site availability and generate application layer and network layer data for each active site.

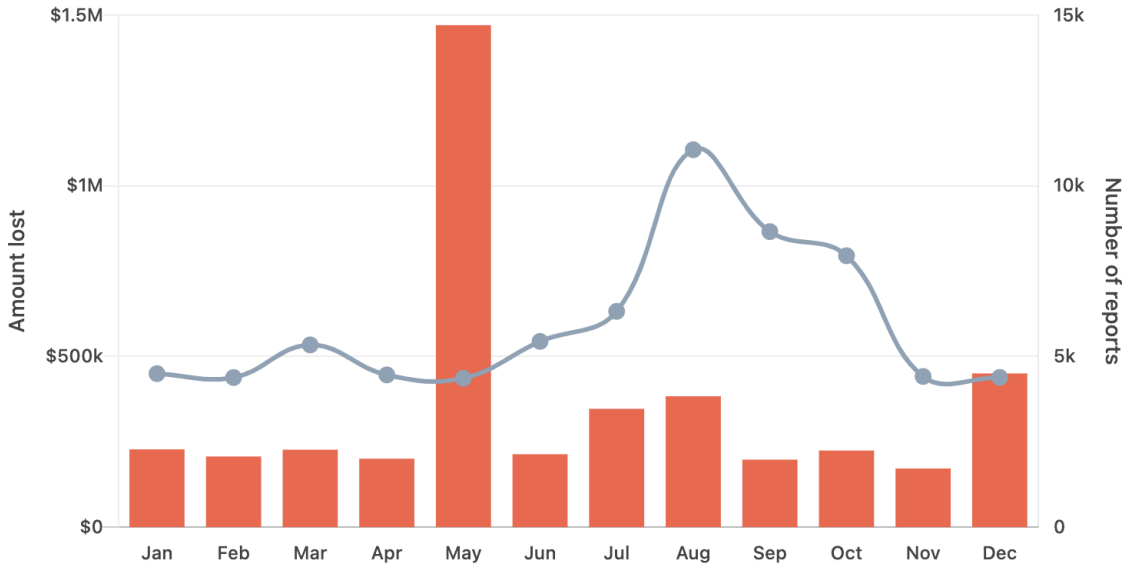

Figure 1: Australia phishing statistics for 2021(Scamwatch, 2022): phishing attacks can exhibit seasonal variations, with the most impactful attacks occurring in May. Nonetheless, even during the "winter" of phishing attacks, the volume is enormous, making site detection an essential task.

1.3 Relevant research

Existing research(Lavreniuk & Novikov, 2020) or analytical projects(Jadiya, 2020, Rickert, 208) on this dataset have demonstrated similar macroscopic trend from different perspectives.

Source	Metric(s)
Jadiya, 2020	accuracy: 95.29%, roc: 0.7953
Lavreniuk and Novikov, 2020	accuracy 96.26%
Rickert, 208	accuracy 95.89%

Table 1: Existing project metrics: Most existing projects use accuracy as their metrics, and have achieved strong out-of-sample predictions

2 Exploratory Data Analysis

Before making any assumptions about the dataset, it is advantageous to examine the characteristics and extract relevant information. This section presents four significant or intriguing discoveries from the dataset. They direct the subsequent preprocessing and modeling.

As the following figures demonstrate, we have found a long tail problem with category characteristics, a skewness problem with numerical features, and an imbalance problem with the outcomes. Each should and could be addressed during the preprocessing phase.

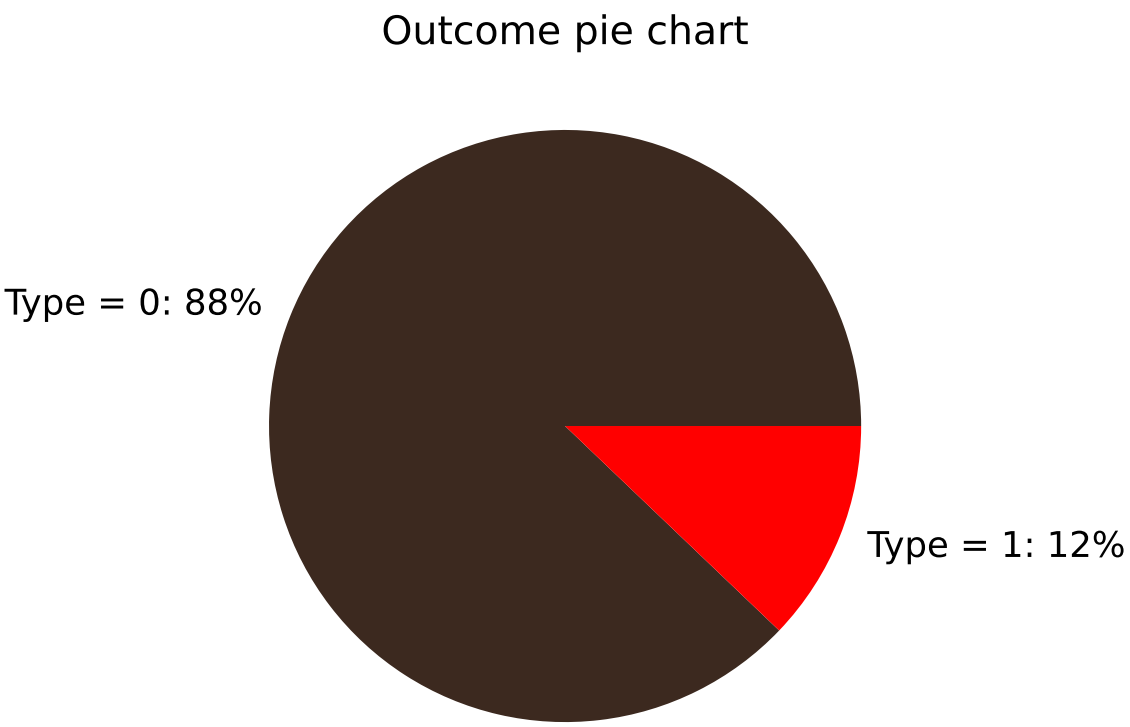

Figure 2: Pie chart of outcome variables: 88% of the 1781 data samples have type 0 outcomes. This skewed data makes sense, given that websites in the real world are predominantly benign and serve meaningful purposes. This plot suggests a need to deal with fewer positive samples in our splitting, training, and analysis process.

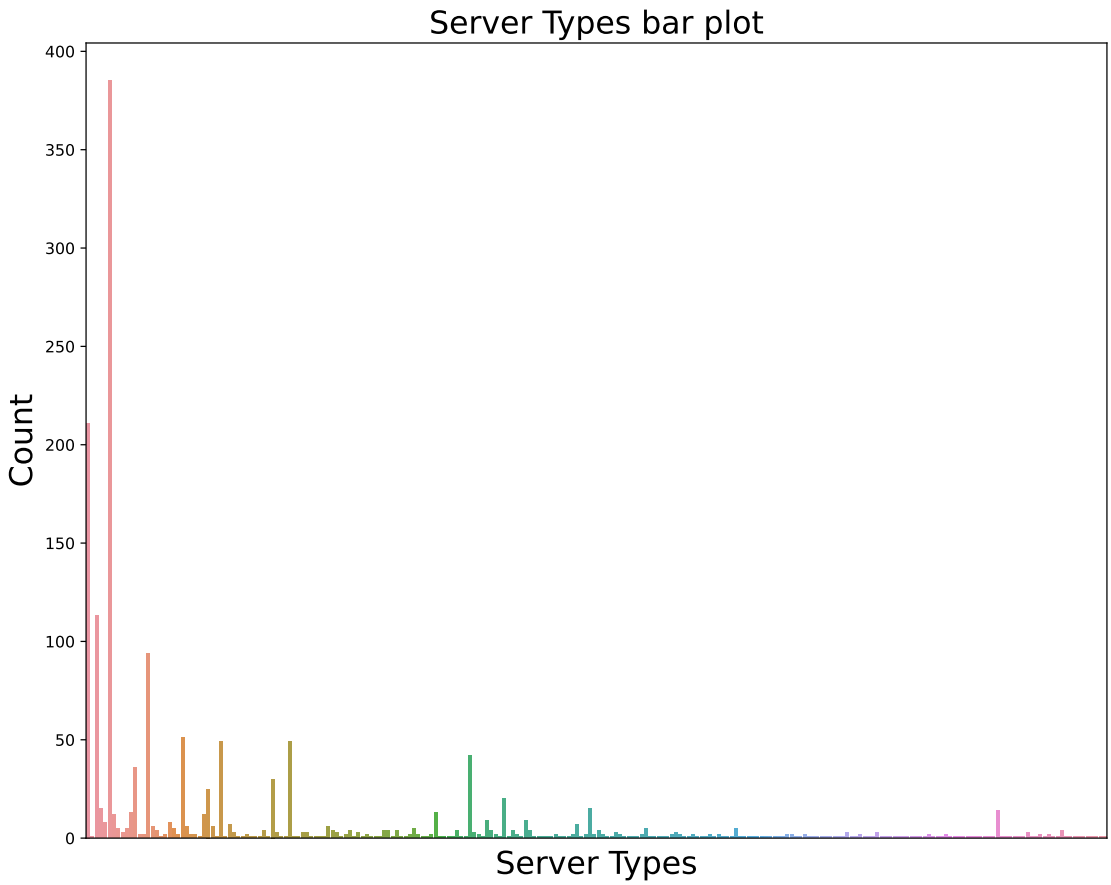

Figure 3: The ineligible bar plot is purposefully left as-is to illustrate the 'long tail' if server types are ordered according to their frequency. Not only does the server name provide little information, but the sheer number of server types (238 in all) makes it difficult to encode this feature or to model when just a few instances of particular types exist.

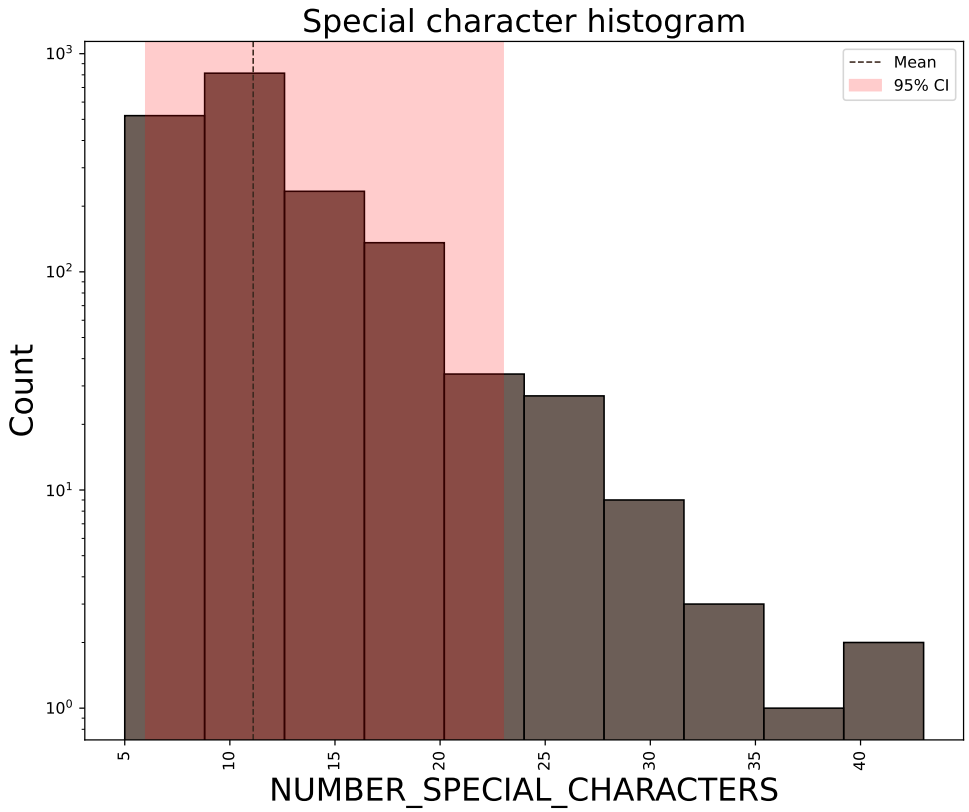

Figure 4: As an illustration for the majority of continuous characteristics in this dataset, the y-log histogram displays exponentially large counts for smaller values and a significant tail. The right-skewed distribution with a mean bigger than the median may result from a log-normal distribution, which is typically represented by the multiplicative product of a large number of independent random variables.

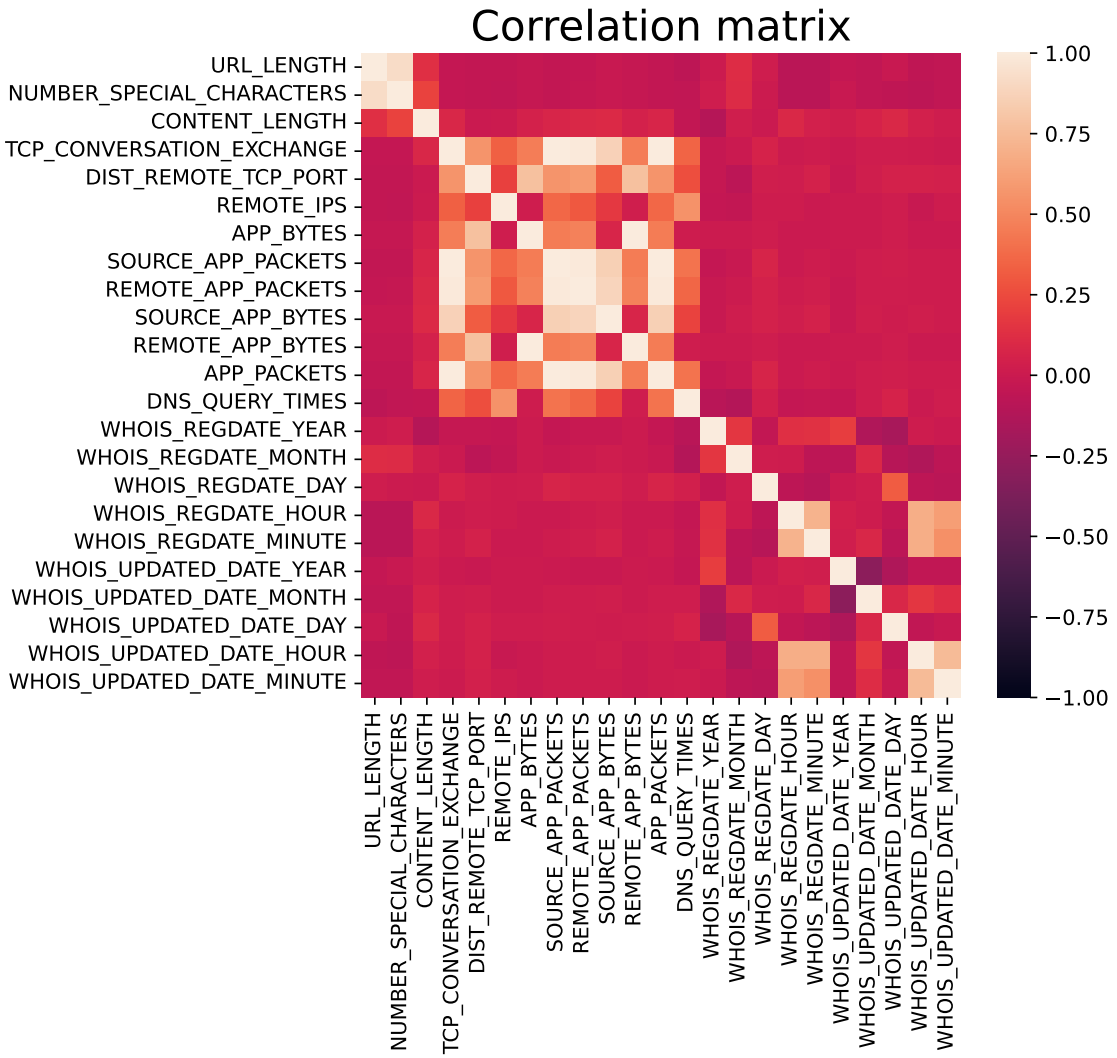

Figure 5: High intra-correlation between network layer features is indicated by the brighter top left corner. This result is reasonable, as certain characteristics are intrinsically associated; for example, REMOTE_APP_BYTES often grow when APP_BYTES increase. This graph indicates the need to reduce feature dimensions through feature engineering or dimensionality reduction approaches such as principal component analysis (PCA).

3 Methods

Now that the EDA has provided us with insight into the dataset's issues and anomalies, it is time to prepare it for modeling. This section elaborates on the processes and outcomes of data splitting, preprocessing including imputing, encoding, and scaling, as well as the actual machine learning pipeline.

3.1 Splitting strategy

It is assumed that all samples are i.i.d., with no group structure, and are not time-dependent. The assumptions can be generally respected because each sample has a distinct domain and has little effective reliance on other sites. We utilized a 80% (train and validation), and 20% (test) split ratio, and within the machine learning pipeline, we conduct a 4-fold cross validation with random shuffle enabled. The purpose of the cross-validation is to estimate the performance of models. It protects against over-fitting on the training dataset and helps maximize the amount

of data used in training. Likewise, shuffling helps prevent information leakage such as intrinsic data order from affecting training.

3.2 Data preprocessing

Before putting the data into training pipeline, one last step is to preprocess the features by their unique nature.

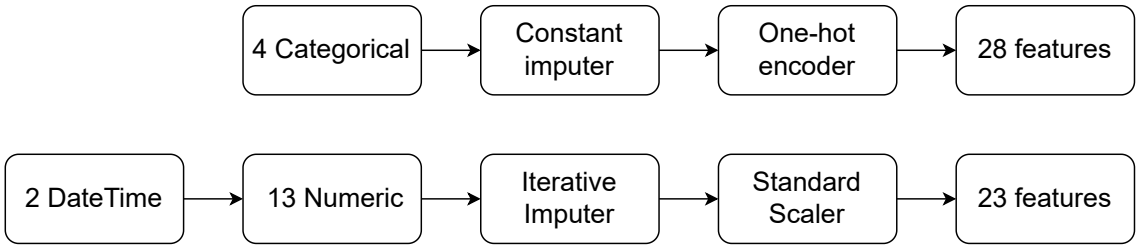

Figure 6: Preprocessing workflow: it shows how the categorical, numerical and date time features were processed in the preprocessor. There are 51 features after preprocessing.

For categorical features, we first use imputation for less frequent categories, labeling them as 'Other,' while we label NAs as 'Missing.' These features are then one-hot encoded to ensure that each column is either numeric or binary. This leaves 28 features for the project.

For numeric features (including year, month, and day), we first utilize iterative multivariate imputer to estimate the missing values, so that the summary statistics of the samples are not excessively affected in comparison to simple imputer. The standard scaler then scales the numerical characteristics, leaving a total of 23 features.

In conclusion, there are 51 features and 1 binary result.

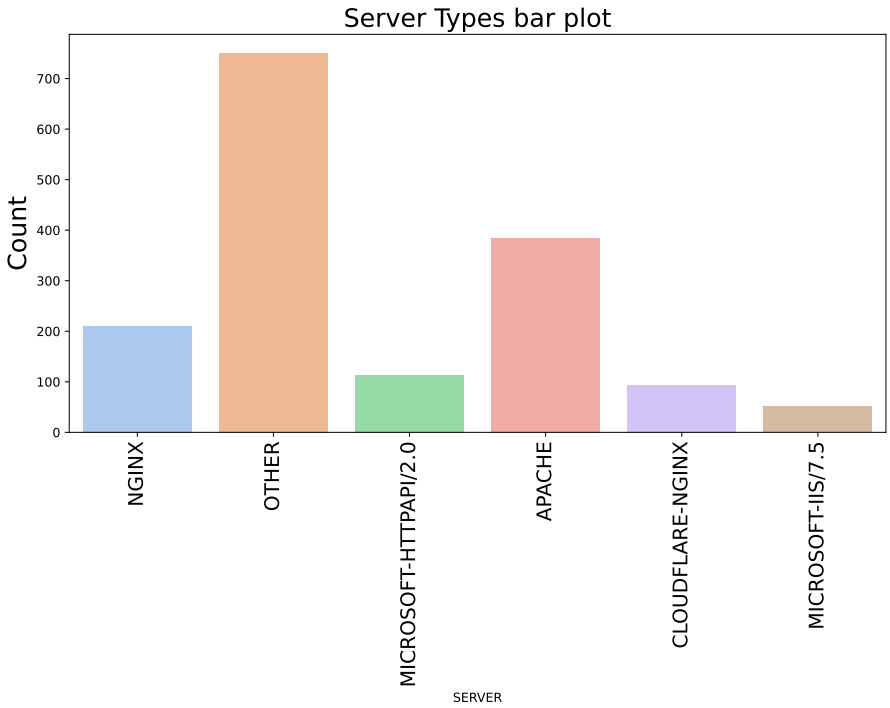

Figure 7: Example of categorical value preprocessing: now there are significantly fewer categories, and the frequency differences are much less significant.

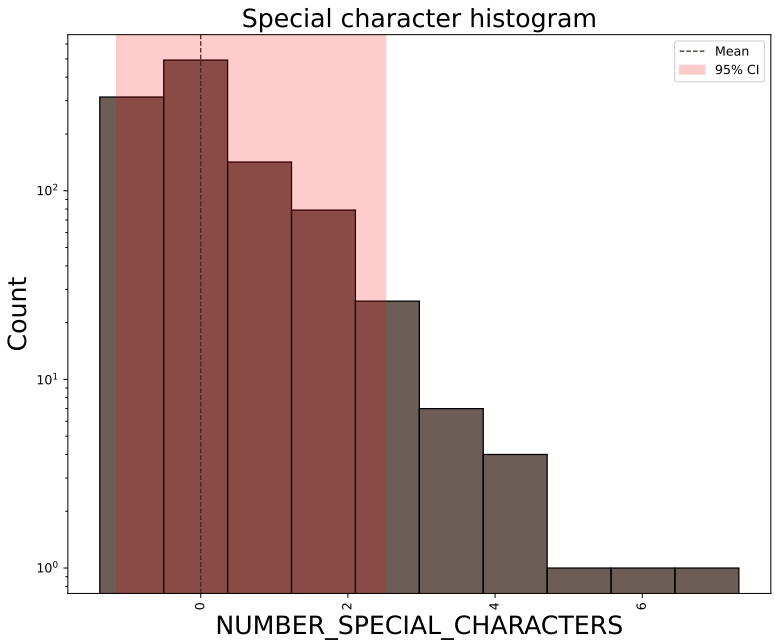

Figure 8: Example of preprocessing of numeric values: while the distribution is still somewhat right-skewed, it is less log-normal than before.

3.3 Machine learning pipeline

This project evaluated a total of five machine learning models: linear classifier with Logistic regression, and non-linear classifiers such as K Nearest Neighbors, Random Forest, and XGBoost. Support Vector Machine can be categorized as either based on the kernel employed. Multiple experiments are conducted for each model to ensure that we can measure the expected performance and stability of the models. To ensure reproducibility, we manually select the initial seed and then generate one seed per experiment. We begin each experiment by splitting the 20% of testing data and then executing GridSearch on the parameter sets. After preprocessing, GridSearch automatically attempts all possible parameter combinations using KFold cross-validation. We select the optimal model based on log loss evaluation on the validation set. After determining the optimal model, we compute evaluation metrics such as log loss, F2, and F0.5 scores on the test set to measure the performance of the ultimate model.

In each experiment, we analyzed a number of metrics to ensure that we could accurately characterize and evaluate the model in the context of the goal problem. Compared to accuracy, log loss is a more effective indicator of our model's decision-making capabilities. For instance, a model that predicts a 71% likelihood of positivity is superior to one that predicts only 51%. Second, F2 assigns a greater weight to recall than to precision, which is useful when missing positive samples is more harmful than a false positive. Similarly, the F0.5 score assigns a greater weight to precision than recall, which provides a more accurate evaluation of the ability to reduce the false positive rate. Finally, the mean and standard deviations of all these metrics are computed to assess the model's average performance and stability, especially when the splitting method or model has a non-deterministic nature.

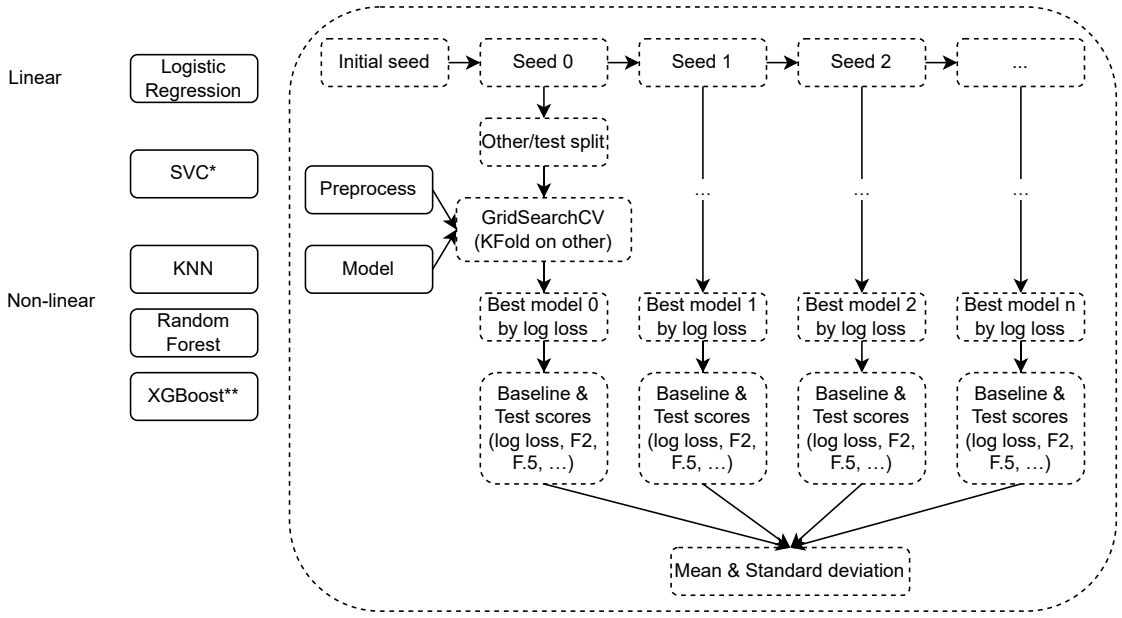

Figure 9: Machine learning pipeline: it illustrates how each experiment is conducted through splitting, preprocessing, and evaluated. It also depicts how multiple experiments support evaluate non-deterministic models.

Logistic regression	• C • L1_ratio	• Adjust regularization strength • Adjust type of penalty
SVC	• Kernel • Degree	• Adjust Kernel type • Adjust degree for poly kernel
KNN	• N_neighbors • Weights	• Adjust no. of neighbours • Adjust ways to compute distance
Random Forest	• N_estimators • Max_depth • Min_samples_split	• Adjust no. of trees • Adjust tree max depth • Adjust node split threshold
XGBoost	• N_estimators • Alpha • Lambda • Max_depth • Subsample	• Adjust no. of trees • Adjust L1 regularization • Adjust L2 regularization • Adjust tree max depth • Adjust ratio of the training instances

Figure 10: Model parameter tuning: for each model, this table describes which parameters are tuned and why each parameter is tuned.

4 Results

4.1 Metrics

After selecting the best model with log loss, we also evaluated the model on test set with three metrics: log loss, F2, and F0.5. Among the five proposed machine learning architectures, XGBoost has been the strongest after careful consideration.

Model	Log loss	F2	F0.5
Log	-10.53	12.74	54.78
SVC	-11.50	9.60	60.12
KNN	-5.99	13.11	46.79
RF	-13.29	15.50	65.38
XGBoost	-5.21	6.64	23.60

Table 2: Differences between test and baseline scores: how many standard deviations apart are the test mean and the baseline mean. The smaller the log loss, the better, while the larger the F score, the better.

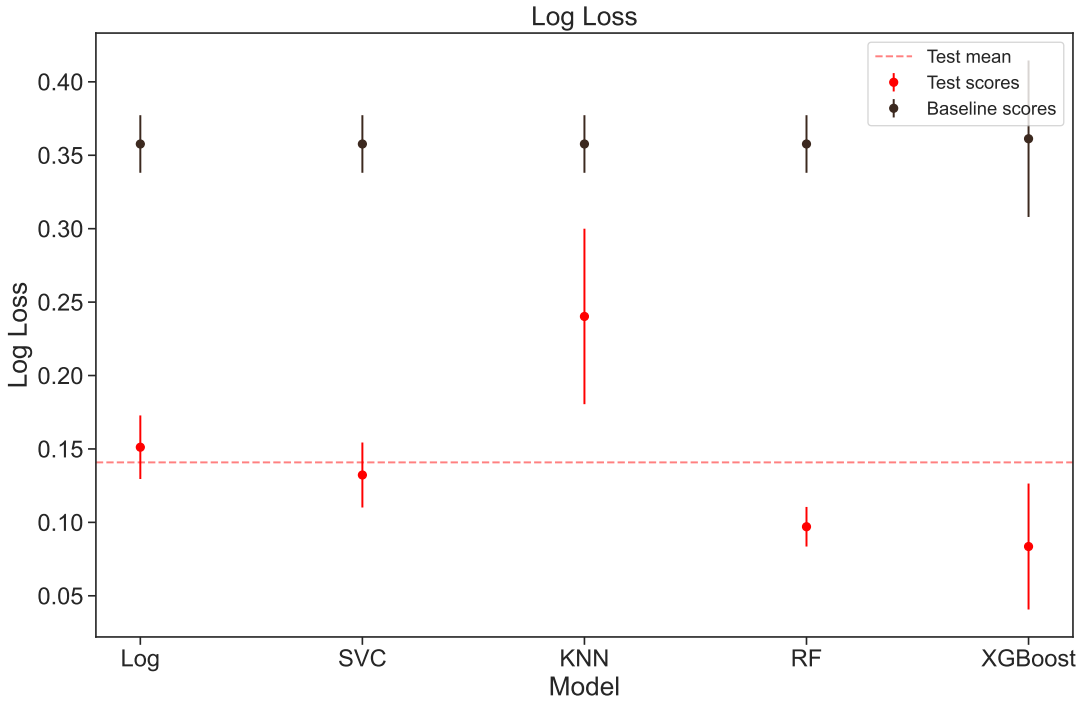

Figure 11: Models log loss scores: XGBoost has the lowest expected log loss among the five models, while KNN has the highest expected log loss. Logistic Regression, SVC, and Random Forest have the lowest standard deviation in terms of stability; XGBoost has a moderately low standard deviation; and KNN has the highest standard deviation of the five models, suggesting that it is less stable. Therefore, in terms of log loss, XGBoost is the best model while KNN is the worst model.

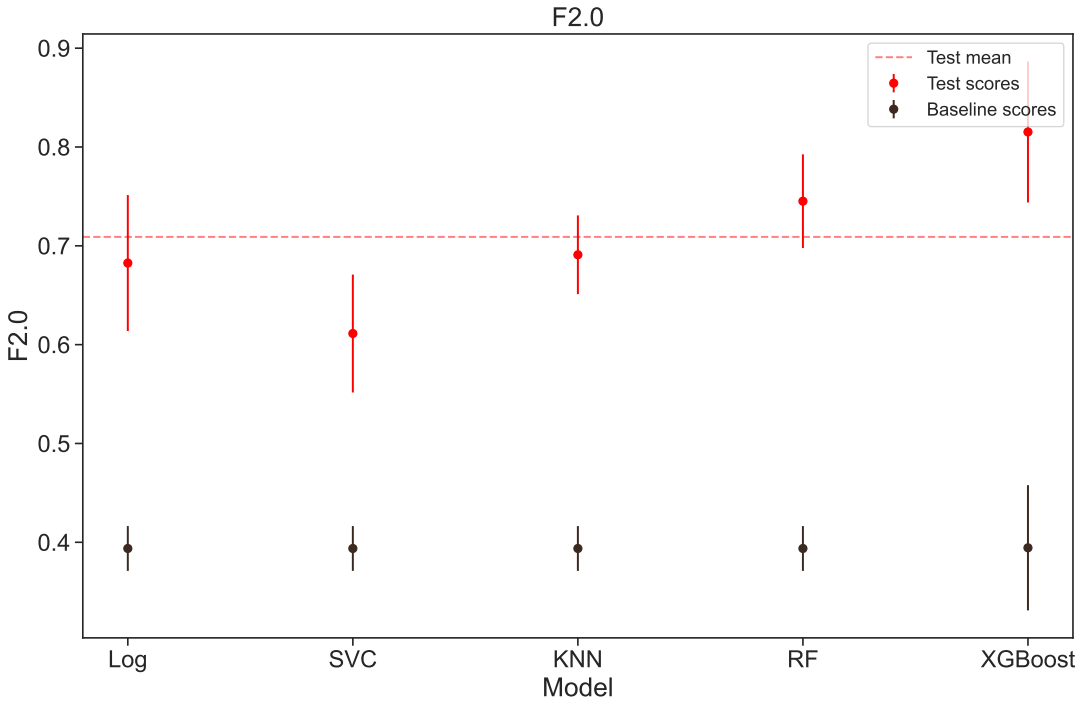

Figure 12: Models F2.0 scores: XGBoost has the highest expected F score among the five models, while SVC has the lowest expected F score. Random Forest, KNN, and XGBoost have the lowest standard deviation in terms of stability; Logistic Regression and SVC have relatively higher standard deviation of the five models, indicating that they are less stable. Therefore, in terms of F2.0 score, XGBoost is the best model, while SVC is the worst model.

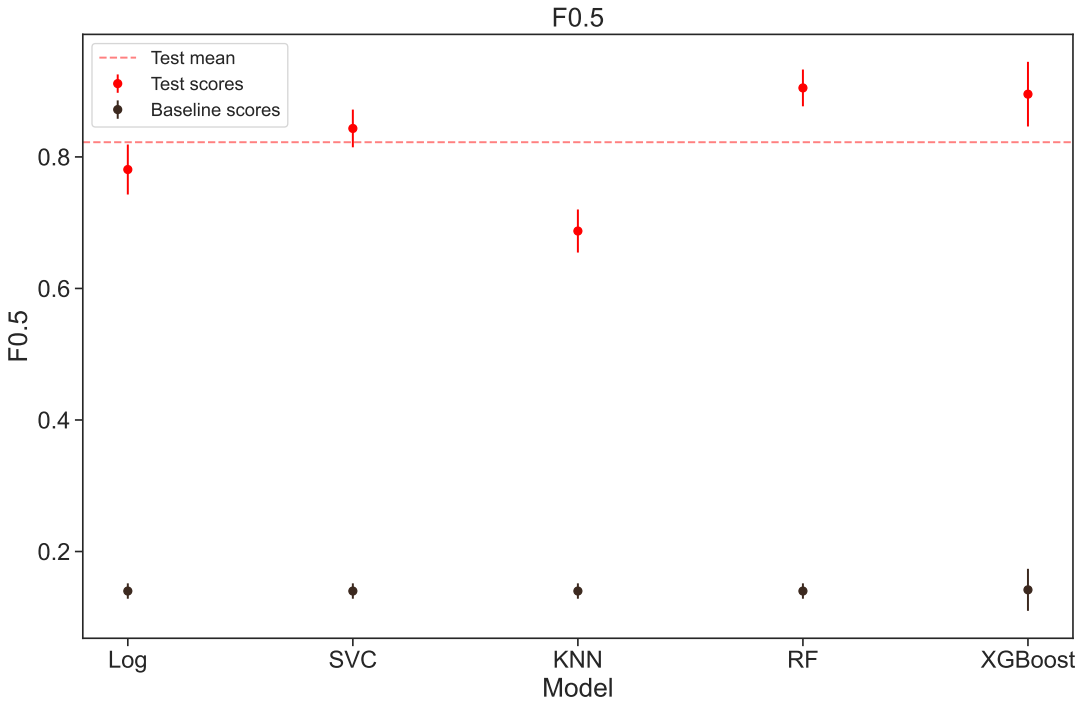

Figure 13: Models F0.5 scores: Random Forest and XGBoost have the highest expected F score among the five models, while KNN has the lowest expected F score. All five models have comparable performance and stability. On the one hand, this suggests that all five models have an equal ability to be precise in their predictions; on the other hand, this could be explained by the data imbalance: with large enough negative samples, the models are inclined to be conservative with their predictions.

4.2 Confusion matrices

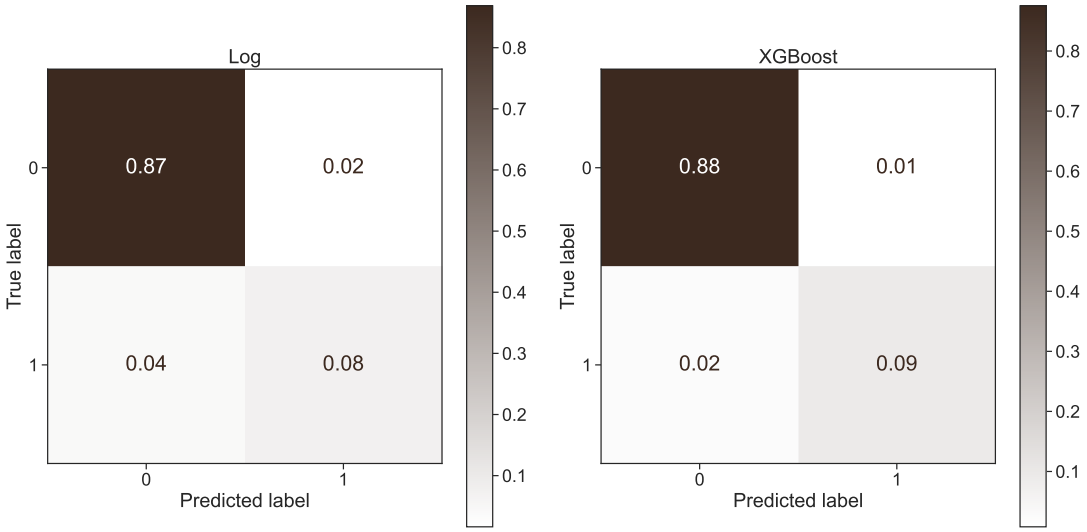

Figure 14: The aggregate confusion matrices of logistic regression and XGBoost are quite similar, but XGBoost is more accurate in both positive and negative predictions. In fact, its false positive and false negative ratios are half of those of logistic regression. This confirms that XGBoost is the superior model, but if speed is a concern, Logistic regression is also acceptable.

4.3 Feature importancee

Last but not least, it's essential to comprehend the models in the context of the problem. The considered approach analyzes the features using three different global importance tools and explains two samples with SHAP local feature importance.

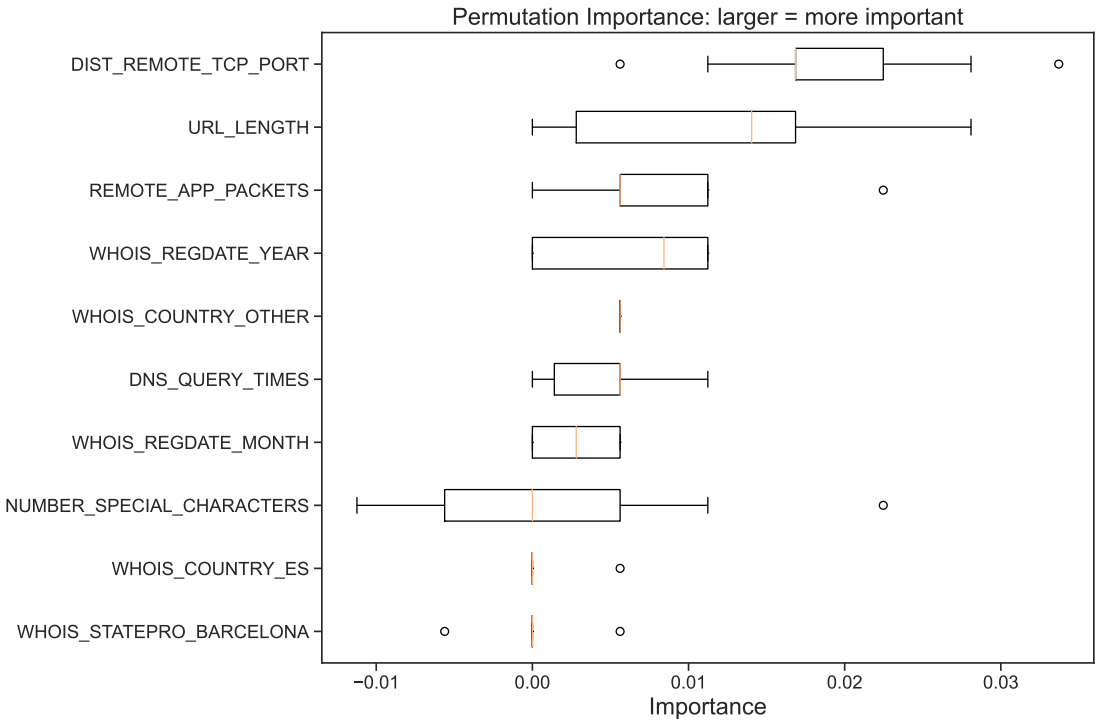

Figure 15: The Permutation Importance is determined by how much the model score degrades when a single feature is shuffled. The most important features are the number of TCP ports and the URL length. The least important feature(not shown) is being in Panama. This makes sense because a greater port number and longer URL indicate that the server performs more operations and may contain more redirects or routing when a client connects.

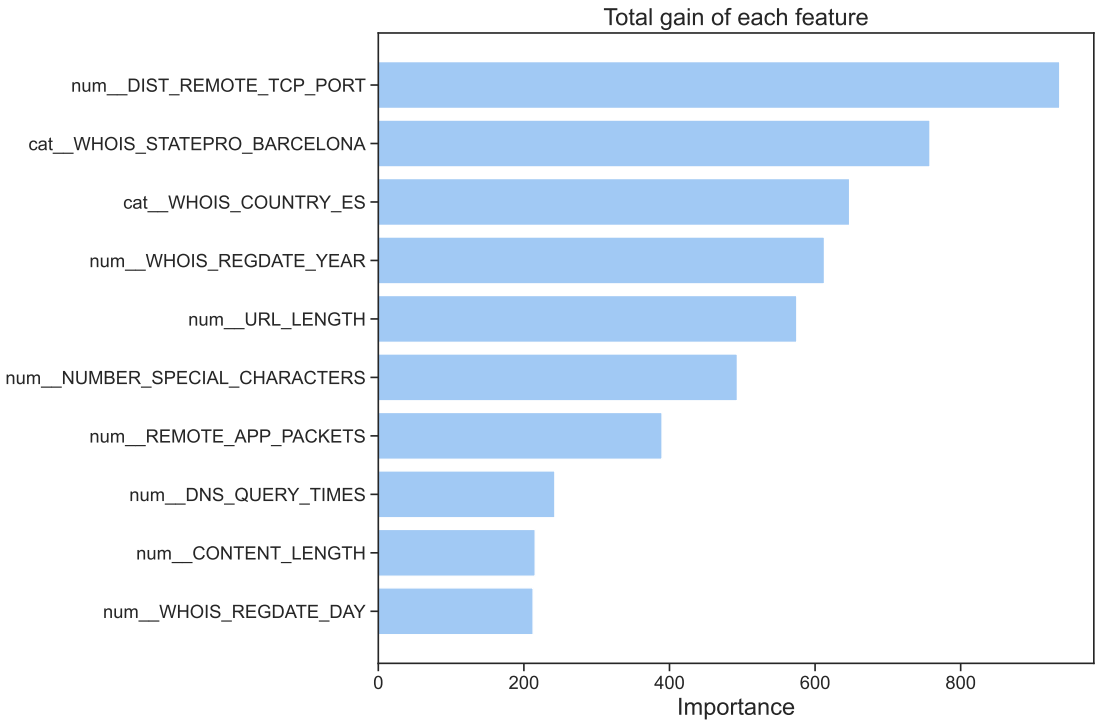

Figure 16: Total gain importance is the increase in score brought about by a model feature. In addition to identifying TCP port count as a significant characteristic, this method also identifies Barcelona as a significant characteristic, which may indicate problems with cyber attack regulations in that state. The least important feature(not shown) is being in Panama.

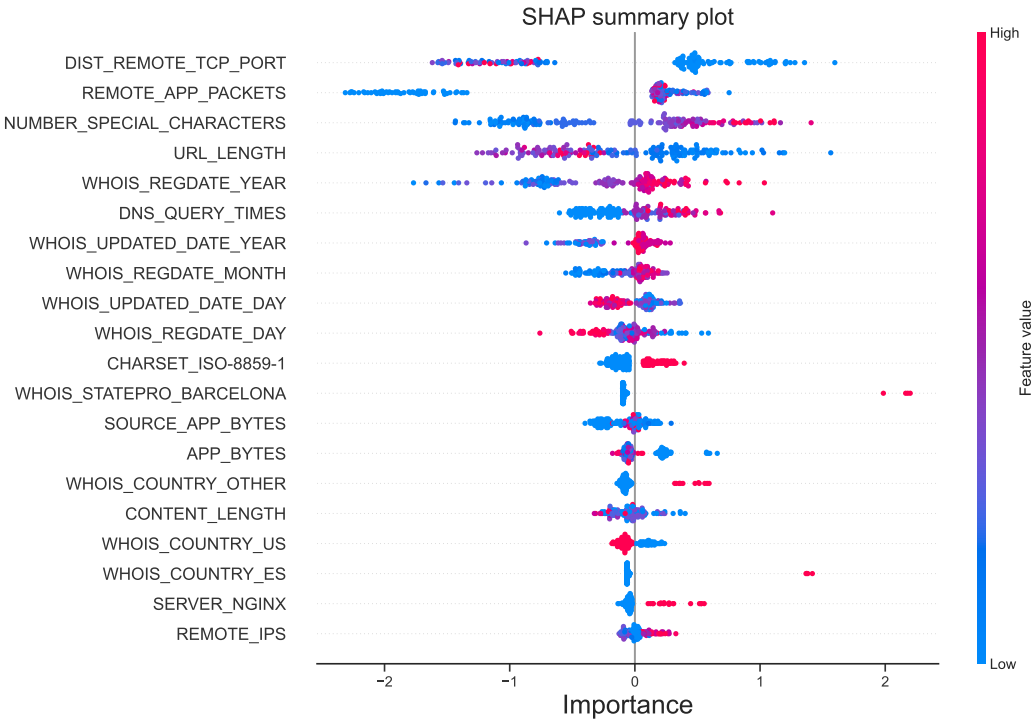

Figure 17: SHAP importance is computed by the difference made from each feature, weighted by the ways to model without that feature. Just like the previous two methods, it lists TCP port count as an essential feature, followed by three application layer features. This indicates that the number and complexity of a server's applications can be used to determine whether or not it is malicious.The least important feature(not shown) is also being in Panama.

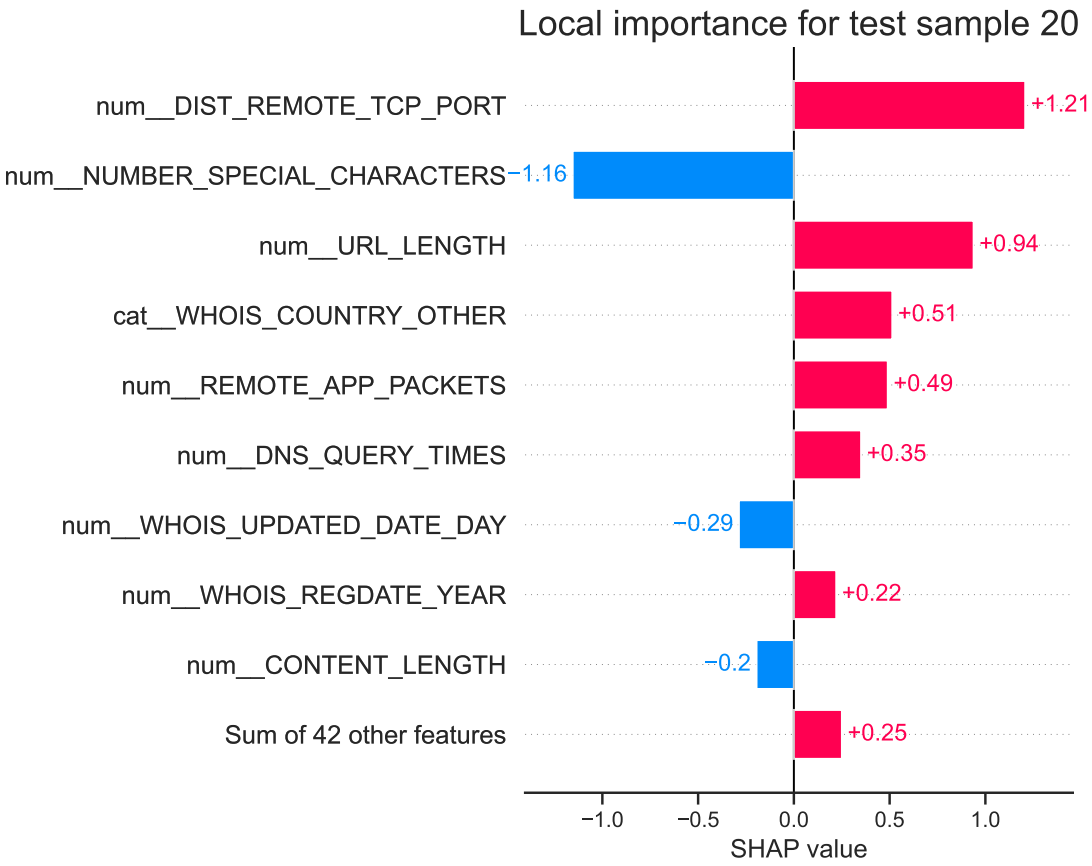

Figure 18: Numerous characteristics of sample 20 make it appear to be a malicious website, including the number of TCP ports, URL length, and other Network Layer characteristics. Despite having a significant amount of positive influence, the number of special characters makes it appear much more benign. Unsurprisingly, it is categorized as positive by a moderate margin.

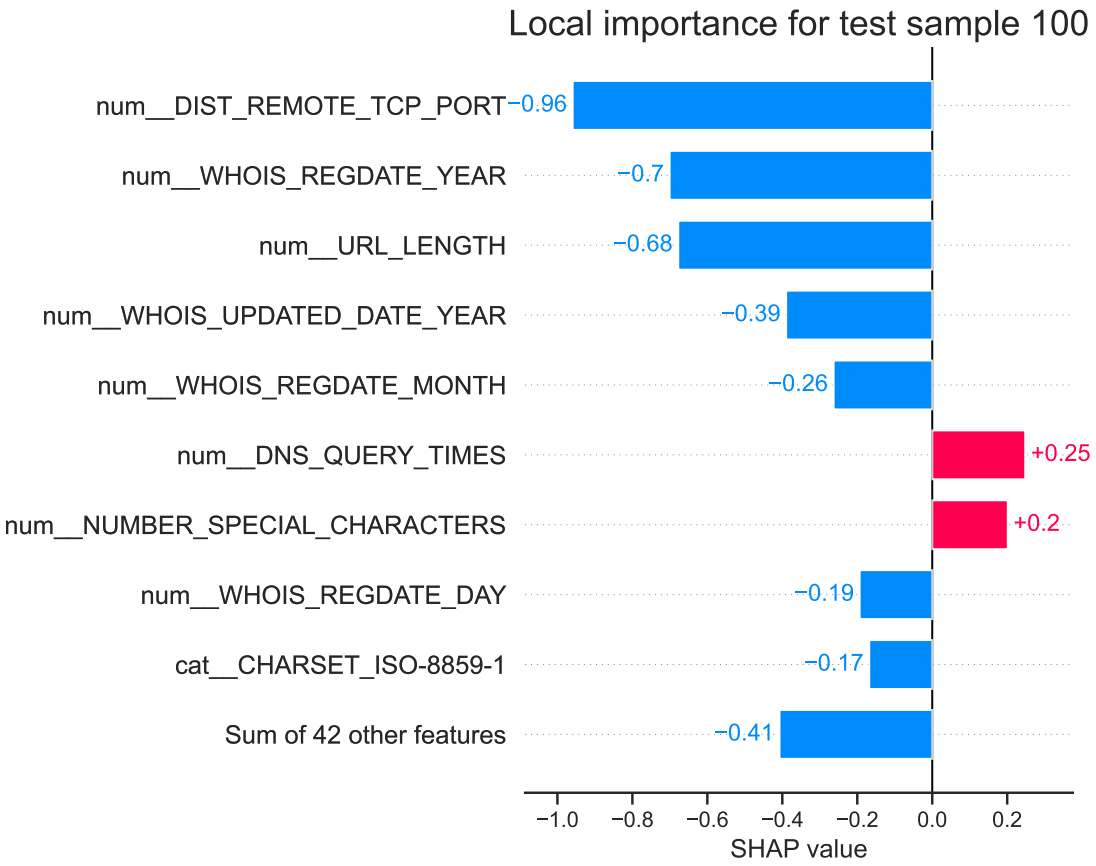

Figure 19: In contrast to sample 20, sample 100 reveals a majority of negative influence features, including TCP port count, URL length, and WHOIS information. Only DNS query times and special characters slightly increase its suspicion level. Consequently, it was predicted to be benign.

5 Outlook

The model can be enhanced in three spectrums. First, we can train on a larger parameter space to confirm that the local parameter optimum is indeed the global optimum. Second, we can implement feature selection and engineering to enhance the fashion in which the model processes information(Shin, 2020). We may also implement an online learning model after deployment(Pagels, 2018). Finally, we can adjust the loss metrics based on the scenario in which the model is used. For example, users may value precision over recall, so we can penalize false positives.

In addition to training a more robust model, we can also increase its prediction speed. Allowing for a trade-off between speed and performance may enable us to come at distinct models under varying constraints. Lastly, it is always beneficial to gather additional data and features.

References

Ames, I. (209). The osi model. https://medium.com/software-engineering-roundup/the-osi-model-87e5adf35e10

Jadiya, P. (2020). Process of classifying malicious and benign websites. https://medium.com/analytics-vidhya/process-of-classifying-malicious-and-benign-websites-815cc2b42435

Lavreniuk, M., & Novikov, O. (2020). Malicious and benign websites classification using machine learning methods. https://pdfs.semanticscholar.org/8864/66398a90c8db74fc966fc74dde9c66a72bdf.pdf

Ni, S. (2022). Malicious websites classification. https://github.com/9r0x/data1030_final

Pagels, M. (2018). What is online machine learning. https://medium.com/value-stream-design/online-machine-learning-515556ff72c5

Rickert, D. (208). Malicious and benign websites learning. https://www.kaggle.com/code/dmrickert3/malicious-and-benign-websites-learning/notebook

Scamwatch. (2022). Phishing stats for 2021. https://www.scamwatch.gov.au/scam-statistics?scamid=31&date=2021

Shin, T. (2020). How i consistently improve my machine learning models from 80% to over 90% accuracy. https://towardsdatascience.com/how-i-consistently-improve-my-machine-learning-models-from-80-to-over-90-accuracy-6097063e1c9a

Urcuqui, C. (2017). Malicious and benign websites. https://www.kaggle.com/datasets/xwolf12/malicious-and-benign-websites