# PhishScan: Project Milestone

**Group 10**

**Jack Hsieh** **Siddharth Ranjan** **Ian Oxley** **Naveen Kumar** **Willem Grier**

## 1 Introduction

### 1.1 Motivation

With the growing use of internet, scams have been more prominent than ever. The FBI's Internet Crime Complaint Center published in their latest report [6] that victims suffered over $4.2 billion in losses. While the common assumption is that elderly citizens are the most vulnerable group, the report shows that people over 60 only account for 23% of victims. In fact, business email account compromise, the most prevalent form of phishing attack, affects younger employees five times more often than older employees [9]. Notably, APWG's Phishing Activity Trends Report for 2023 [1] shows that the majority of attacks are perpetrated through systems provided by reputable corporations such as Google and Microsoft. Additionally, financial institutions are the most targeted sector, accounting for 23% of all attacks [1].

Human error appears to be the root cause of successful phishing attacks. Distraction, fatigue, and pressure to work quickly are all factors that contribute to humans making mistakes and becoming victim to phishing attacks while on the job [9]. While educating people on how to identify phishing attacks is important, the improvement of phishing tactics is rapid and may outpace for the rate at which humans can be trained. Therefore, a robust automated solution is required.

Just as AI has become an effective tool for writing emails and documents for personal and business use, cybercriminals have also begun utilizing AI *en force*. *WormGPT*, *FraudGPT*, *DarkBART*, and *DarkBERT* are chatbots with custom LLMs designed to aid cybercriminals develop phishing attacks [8]. People face challenges in identifying AI-generated versus human-generated content. These challenges have severe social and economic consequences when applied to cybersecurity. Consequently, an AI-driven approach is prudent in tackling this rapidly growing use of LLMs in cybercrime.

### 1.2 Problem Statement

The evident efficacy of certain machine learning models for classification tasks lends itself to addressing the problem of automated phishing identification. Phishing websites remain the most effective and common method of phishing [1]. We propose that machine learning models can be used to classify websites as phishing or legitimate.

The *Web Page Phishing Detection Dataset* [4] contains a balanced set of 11430 websites, exactly half of which are verified phishing sites. The 88 extracted features contain metadata about the URL structure and syntax, content of the web pages, and web traffic.

We first obtain baseline results from a variety of classifiers. Additionally, the BERT transformer [2] is used to compare a multi-modal approach that tokenizes the web page URL. A comparative analysis highlights the need for improvements in methodology and the choice of classification algorithm for this task and dataset. From this insight, an integrated classifier combining

the top two performing models is proposed, designed, and implemented.

### 1.3 Related Work

[7] discusses a deep learning approach to detect phishing web pages. The authors propose a framework using a multilayer perceptron (MLP), a type of feed-forward neural network, to classify web pages as phishing, suspicious, or legitimate. The dataset includes ten features for each web page. The model achieved 95% training accuracy and 93% test accuracy. The paper also suggests adding more layers to the neural network and using more advanced techniques like back-propagation for improvement. We add to this paper by exploring a wider range of classification methods and using a richer dataset of 88 features.

### 1.4 Technical Background

The baseline results are collected for the following classifiers:

- Logistic Regression
- Support Vector Machines
- Decision Tree
- Recurrent Neural Network
- Graph Neural Network
- Random Forest
- XGBoost
- KNN

#### 1.4.1 Logistic Regression

Logistic regression is a statistical model used for binary classification tasks. It's commonly used to predict the probability of a binary outcome (e.g., success/failure, yes/no, 0/1) based on one or more independent variables (features). Logistic regression is particularly useful when the dependent variable is categorical. In our project it is used to find that whether a legitimate or not.

#### 1.4.2 Support Vector Machines

Support Vector Machine (SVM) is a supervised machine learning algorithm commonly used for classification and regression tasks. SVM works by finding the hyperplane that best separates the data points into two classes, maximizing the margin between the closest points of each class, known as support vectors. The decision boundary is determined by these support vectors, making SVM robust to outliers.

#### 1.4.3 Decision Tree

A Decision Tree is a flowchart-like tree structure where each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (decision). It is a simple yet effective tool for classification and regression tasks. Decision trees split the data into subsets based on the values of input features, recursively partitioning the dataset until each leaf node corresponds to a class label.

#### 1.4.4 Recurrent Neural Network

Recurrent Neural Network (RNN) is derived from a feed-forward approach which can also retain information from previous inputs by including a memory component. RNN is commonly used with handling sequential data, meaning the order of the data matters and can affect later entries. This approach suits our project because phishing emails have correlated elements. For example, a URL string feature is sequential data because each character contributes to its structure and semantics. In this milestone, we have implemented a basic RNN model and have evaluated it enough to propose methods of improvement.

#### 1.4.5 Graph Neural Network

Graph Neural Network is a deep learning approach that takes graphs as input. This method captures relation between each connected nodes in the graphs. This model is very useful for geometric data such as the friends network system used on social media platforms. We decided to try this approach on our subject because it would be interesting to see how different graph formulation of the extracted features affects the result. [5] proposed a hierarchical GNN for text classification. In this milestone, we will try using a simpler GNN model as a baseline, and we can draw some conclusion or propose an improvement.

#### 1.4.6 Random Forest

Random Forest is an ensemble learning method used for classification and regression tasks. It operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (in the case of classification) or the mean prediction (in the case of regression) of the individual trees. Random Forest combines the simplicity of decision trees with the flexibility of ensemble methods, resulting in a model that is both accurate and robust to overfitting.

#### 1.4.7 XGBoost

XGBoost is an advanced implementation of gradient boosting algorithm. It is a highly efficient and scalable algorithm that is widely used in machine learning competitions and real-world applications. XGBoost works by sequentially building an ensemble of weak decision tree models, where each new tree attempts to correct the errors made by the previous ones. It employs sophisticated regularization techniques to control overfitting, making it robust and accurate.

#### 1.4.8 KNN

K-Nearest Neighbors (KNN) is a simple, non-parametric, and lazy learning algorithm used for classification and regression tasks. It is based on the principle of proximity, where the class of a data point is determined by the majority class among its k-nearest neighbors. The value of k is a user-defined parameter, and the distance between data points is typically measured using metrics such as Euclidean, Manhattan, or Minkowski distance.

## 2 Progress

### 2.1 Logistic Regression

We started first with data cleaning. The urls present in the url column were embedded using BERT Tokenizer. The embeddings obtained were concatenated with the rest of the columns. A 75:25 train-test split was utilised. Evaluation of the model was done using F1 score.

### 2.2 Recurrent Neural Network

First, the features and target variables are extracted from the dataset. We are currently dropping the url text data, but the model works with all other features included in the dataset. This is then split into training and testing data, shuffled, and then sorted into batches for proper use. Once the data is loaded, the RNN model can be defined. For the best results, it utilizes two convolution layers of size 512 and a learning rate of 0.001. It is trained for a total of 10 Epochs, resulting in an average MSE-Loss of 0.035 per iteration on the 10th Epoch. We found that these values provided us a enough iterations to avoid under fitting, but also not too many to introduce over fitting. We are confident the model is well implemented, however it still needs the following before considered fully complete:

- Include URL, a text feature will allow RNN to better leverage its memory capabilities. See Future Plans section
- Model tuning. Include more than basic ReLU and Linear methods
- Plotting. Include diagrams which can showcase the training process

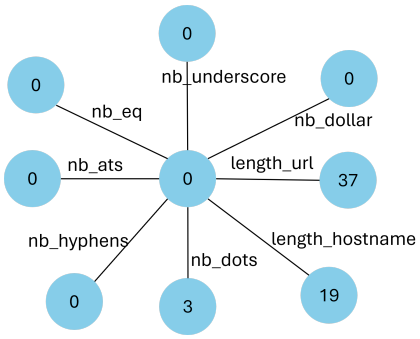


Figure 1: a truncated graph of the first training data showing links between a supernode and its features

### 2.3 Graph Neural Network

We start by creating a graph using our dataset. We first drop the text column which holds the url text data, then we create node for each feature and connect them to an arbitrary super node that is just an ID given for each record. Figure 1 shows a truncated graph of the first record as example.
Next, we defined the GNN model, which consists of two convolutional layers of size 128 and a linear output layer with sigmoid as activation function since this is a binary classification model. We also applied batch normalization to normalize data and 20 percent dropout to prevent from overfitting.
After splitting the dataset into 80% training data and 20% testing data, constructing the graph and training the GNN model with MSELoss, at 1e-4 learning rate for 10 epochs, we got f1 score of 0.6471, which is not ideal. However, some improvements can be made:

- different graph structure: changing relations between each features might make them more useful
- reduce number of nodes
- different aggregation method: current model uses information of the supernode, we can try using sum mean or other aggregation methods
- hyperparameters tuning
- using an embedding of text data: this will be discussed in the Future Plans section

### 2.4 Random Forest

We use Boruta, a feature selector, to obtain 37 significant features. For the hyperparameter tuning process, we apply 5-fold cross validation for 10 randomly sampled tree depths and number of trees, totalling 50 fits. This yields a maximum depth of 18 and 298 estimators (trees). A 80:20 train-test split is utilized.

### 2.5 Other Methods

For XGBoost, Support Vector Machines, K-Nearest Neighbors, and Decision Trees. We used multiple feature selection methods like sklearn.feature selection and BorutaShap out of which BorutaShap which selected top 29 features was selected because of better precision. We apply GridSearchCV with 5 fold cross validation to select the best model. A 80:20 train-test split is utilized.

## 3 Future Plans

### 3.1 Uniform Methodology

Feature extraction, loss evaluation, and data processing methods are not perfectly uniform across all approaches. This makes the results less robust. For this reason, we will share amongst the team the code for formatting the dataset, extracting significant features, and computing loss so that the inputs

Table 1: F1 Scores for each Approach

| Model | F1 |
|---|---|
| Logistic Regression | 0.96 |
| SVM | 0.95 |
| Decision Tree | 0.93 |
| RNN | 0.96 |
| GNN | 0.64 |
| Random Forest | 0.96 |
| XGBoost | 0.98 |
| KNN | 0.95 |

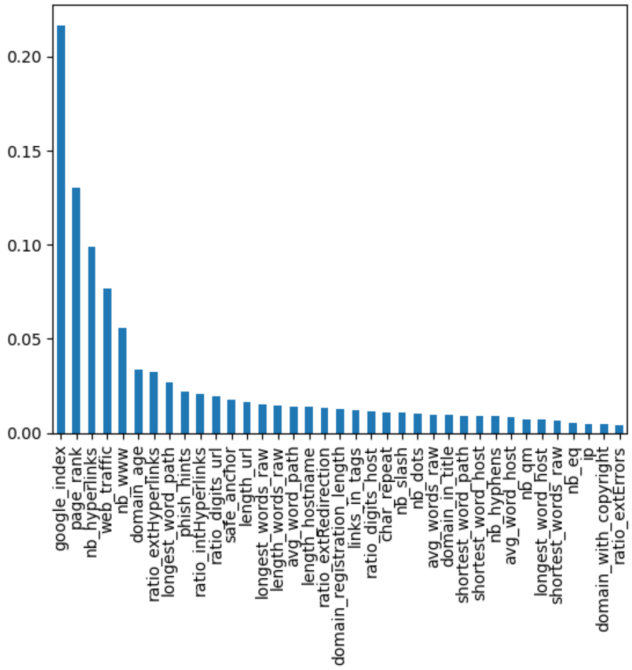


Figure 2: Boruta selection of 37 significant features. 51 features are discarded.

to all models are similar. We will standardize the use of Boruta for feature selection and MSE loss as our loss function. Additionally, we will require hyperparameter tuning for all approaches.
Another improvement regards using the URL string data as inputs. Spoofing is a type of phishing where "someone disguises a [...] website URL - often just by changing one letter, symbol, or number - to convince you that you are interacting with a trusted source" [3]. We hypothesize that encoding the URL using a BERT transformer [2] will improve phishing web site classification. BERT will therefore be implemented for all approaches to set a secondary baseline.

### 3.2 Novel Design

The baseline results demonstrate how certain approaches yield higher accuracy for the phishing detection task. As future work, we will construct a theory for why some models are more accurate. From this, we hypothesize that an integrated approach - a combination of two models - may outperform the top-performing baseline results. We will design and attempt to implement an integrated approach for the final report.

## References

[1] APWG (2023). Phishing activity trends report, 2nd quarter 2023.

[2] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[3] FBI (2023). Spoofing and phishing.

[4] Hannousse, A. (2021). Web page phishing detection.

[5] Hua, S., Li, X., Jing, Y., and Liu, Q. (2022). A semantic hierarchical graph neural network for text classification.

[6] IC3, F. (2021). 2020 internet crime report.

[7] Saha, I., Sarma, D., Chakma, R. J., Alam, M. N., Sultana, A., and Hossain, S. (2020). Phishing attacks detection using deep learning approach. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 1180–1185.

[8] SlashNext (2023). The state of phishing 2023.

[9] Tessian (2023). Psychology of human error 2022: Research report.