Project Report: Phishing URL Detection

**Name:** Sushil Saindane  
**NJIT UCID:** sbs8  
**Email Address:** sbs8@njit.edu  
**Date:** 11/24/2024  
**Professor:** Yasser Abduallah  
**Course:** CS 634101 Data Mining

# Introduction

* This project implements multiple machine learning algorithms to detect phishing URLs using the PhiUSIIL Phishing URL Dataset. The models used include Random Forest, Decision Tree, LSTM, and Bernoulli Naive Bayes. The project aims to evaluate the effectiveness of these models in classifying URLs as legitimate or phishing.

# Required Packages

* pandas
* numpy
* scikit-learn
* tensorflow
* matplotlib
* seaborn
* tabulateCore
* Ensure Python 3.x is installed on your system.
* Install required packages using pip:
  + *pip install pandas numpy scikit-learn tensorflow matplotlib seaborn tabulate tqdm*

# Steps to run the project:

1. **Files Required**
   1. **Dataset**: PhiUSIIL\_Phishing\_URL\_Dataset.csv
   2. **Python Script**: saindane\_sushil\_finaltermproj.py
   3. **Jupyter Notebook**: saindane\_sushil\_finaltermproj.ipynb
2. **Open Jupyter Notebook:** 
   1. Launch Jupyter Notebook from your terminal or command prompt by typing jupyter notebook and pressing Enter.
   2. Navigate to the directory containing saindane\_sushil\_finaltermproj.ipynb.
3. **Load the Notebook:** 
   1. Click on saindane\_sushil\_finaltermproj.ipynb to open it in a new tab.
4. **Execute the Notebook:** 
   1. Ensure that the dataset file PhiUSIIL\_Phishing\_URL\_Dataset.csv is in the same directory as the notebook or update the file path in the notebook accordingly.
   2. Run each cell sequentially by clicking on a cell and pressing Shift + Enter, or use the "Run" button in the toolbar.
5. **View Results:** 
   1. The notebook will display tables with metrics for each model, confusion matrices, and a comparison of model performances.
6. **Prerequisites**
   1. Python Installation: Ensure that Python 3.x is installed on your system. You can download it from the official Python website.
   2. Package Installation: Install the necessary Python packages using pip. Open a terminal or command prompt and run the following command:
      1. *pip install pandas numpy scikit-learn tensorflow matplotlib seaborn tabulate tqdm*

# Project Workflow

* This project implements four classification algorithms to analyze the **Phishing URL Dataset**. It takes an average across 10 folds for each algorithm. The chosen algorithms by me are:

1. **Random Forest** - An ensemble learning method that constructs multiple decision trees for improved accuracy.
2. **Decision Tree** - A straightforward model that splits data into branches to make predictions.
3. **LSTM (Long Short-Term Memory)** - A type of recurrent neural network suitable for sequence prediction tasks.
4. **Bernoulli Naive Bayes** - A probabilistic classifier based on Bayes' theorem with strong independence assumptions.

# Implementation Details

1. **Data Preprocessing**:

* Loaded the dataset and limited it to 50,000 rows for efficiency.
* High-cardinality features such as FILENAME, URL, and Domain were processed using **Feature Hashing**.
* Low-cardinality features like TLD and Title were one-hot encoded.
* Numeric features were selected and combined into a feature matrix XX with the target variable yy.

A screenshot of a computer program

Description automatically generated

Figure : Loading the dataset and preprocessing steps

A screenshot of a computer program

Description automatically generated

Figure : Saving pre processed data and scaling features

1. **Splitting the Data**:

* The dataset was split into training (80%) and testing (20%) sets using train\_test\_split.

1. **Feature Scaling**:

* Standardization was applied to the features using StandardScaler.
* Features were binarized for Bernoulli Naive Bayes classification using Binarizer.

A screenshot of a computer program

Description automatically generated

Figure : Binarize the features for Bernoulli NB

1. **Model Training and Evaluation**:

* Implemented K-Fold cross-validation with 10 folds to evaluate model performance.
* Each model was trained on the training set, and predictions were made on validation and test sets.

A screenshot of a computer program

Description automatically generated

Figure : Training and evaluating data

A close-up of a computer code

Description automatically generated

Figure : Training Random Forest and Decision Tree and performing 10 folds

A screenshot of a computer program

Description automatically generated

Figure : Training LSTM and performing 10 folds

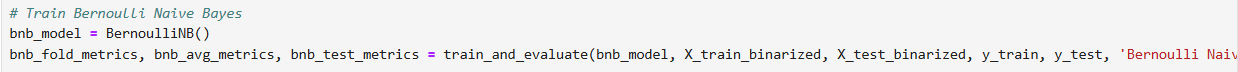


Figure : Training Naive Bayes

1. **Performance Metrics Calculation**:

* Metrics such as accuracy, precision, recall, F1-score, TPR, TNR, FPR, FNR, TSS, and HSS were calculated using a confusion matrix.

A screen shot of a computer code

Description automatically generated

Figure : Function showing a script snippet to display the metrics in tabular format

A screenshot of a computer program

Description automatically generated

Figure : Script to plot confusion matric for all four algorithms

# Results:

* Here is an output example for the metrics I achieved for Random Forest algorithm represented in a tabular format as instructed:
* We can see the metrics for the Random Forest algorithm for 10 folds along with the average metrics and test metrics.

A table of numbers and lines

Description automatically generated with medium confidence

Figure : Random Forest Metrics Output

* Similarly, I have represented metrics for Decision Tree, LSTM and Naive Bayes algorithms, which can be viewed in the jupyter file I that can be found in the zip folder as well as can be accessed on the [***google drive here***](https://drive.google.com/file/d/1bcMJgjs9GT2fgW6r_GkuTlA74Z0mvUkT/view?usp=sharing)**.**
* Below output represents the confusion matric for all the four algorithms:
* Here is how the results were shown in a tabular format as an output for metrics such as accuracy, precision, recall, F1-score, TPR, TNR, FPR, FNR, TSS, and HSS were calculated manually using a confusion matrix.

A close-up of a number

Description automatically generated

Figure : Output showing Comparison of all four Models

# Comparison of Algorithms

* The Random Forest algorithm generally outperformed others in terms of accuracy and robustness against overfitting due to its ensemble nature.
* Decision Trees provided interpretable results but were prone to overfitting without proper pruning.
* LSTM showed promise in capturing sequential dependencies but may require more tuning for optimal performance.
* Bernoulli Naive Bayes performed adequately.
* All models demonstrated high precision and recall, with Decision Tree and Random Forest achieving perfect or near-perfect scores.
* LSTM and Bernoulli Naive Bayes showed a slight trade-off between precision and recall, with recall being consistently higher than precision.
* Decision Tree had zero false positives, making it ideal for applications where minimizing false alarms is crucial.
* Random Forest had a very low FPR (0.0010 on test data), also suitable for low false-alarm scenarios.
* LSTM and Bernoulli Naive Bayes had higher FPRs (0.0471 and 0.0282 respectively), which might be a consideration in certain applications.

# Conclusion

* The Random Forest algorithm was identified as the most effective model for this dataset based on the evaluation metrics. Future work could explore hyperparameter tuning for LSTM and additional feature engineering techniques.
* The comparison of Random Forest, Decision Tree, LSTM, and Bernoulli Naive Bayes classifiers on this dataset provides insights into the effectiveness of various machine learning approaches in addressing the evolving nature of phishing attacks.
* The consistent performance across average and test metrics for all models suggests good generalization and robustness.
* The high performance across all models suggests that the PhiUSIIL Phishing URL Dataset is well-structured and contains highly discriminative features for phishing detection. This underscores the importance of quality data in machine learning applications.

# Future Research:

* Explore ensemble methods combining the strengths of different models to potentially improve overall performance.
* Investigate the few misclassifications, especially for LSTM and Bernoulli Naive Bayes, to understand challenging cases and refine the models.
* Consider feature importance analysis, particularly for Random Forest, to gain insights into the most predictive URL characteristics for phishing detection.

# My GitHub Repository

<https://github.com/sushilsaindane/saindane_sushil_midtermproj>

# References

1. [*PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning:* Elsevier Paper by Arvind Prasad and Shalini Chandra](https://www.sciencedirect.com/science/article/pii/S0167404823004558#se0260)
2. [UC Irvine ML Repository for PhiUSIIL Phishing URL (Website)](https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset)
3. [Scikit-learn documentation](https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html)
4. [TensorFlow documentation](https://www.tensorflow.org/)
5. [Association Rule Mining](https://towardsdatascience.com/apriori-association-rule-mining-explanation-and-python-implementation-290b42afdfc6)
6. [Github example](https://github.com/30lm32/ml-spam-sms-classification)
7. [Perplexity](https://www.perplexity.ai/)