Web Page Phishing Detection

Detect Phishing URLs using Python

Team Members

SN	Student Name	Section Number
1	أحمد حسين بدر رشاد	4
2	أحمد علي فهمي محمد	4
3	عبد الرحمن هلال صالح عبد العزيز	4
4	عبد الله عيد السيد عمار	4
5	محمد حسام الدين عثمان حسن شديد	4

Network Security Course, 4th CSE
Faculty of Engineering, Helwan University

Table of Contents

1. Introduction

1.1 Project Overview

This project focuses on building a machine learning model to detect phishing websites. Phishing attacks are a significant cybersecurity threat, where attackers deceive users into providing sensitive information such as passwords, credit card numbers, or other personal details. This project aims to address this challenge by developing a system that can classify a given URL as either 'legitimate', or 'phishing' based on its features and attributes.

1.2 Objectives

- Develop a robust and accurate machine learning model for phishing detection.
- Compare the performance of multiple algorithms to determine the best approach.
- Evaluate the model's effectiveness using comprehensive metrics.
- Lay the groundwork for potential deployment in real-world scenarios.

1.3 Significance of the Project

Phishing detection is critical in today's digital landscape, where online fraud is increasingly sophisticated. By automating the detection process, this project contributes to enhancing cybersecurity defenses, reducing user vulnerability, and preventing financial losses and data breaches.

2 Dataset general description

2.1 Dataset Overview

- **Number of Records**: 11,430
- **Number of Features**: 89
- **Primary Goal**: Classify URLs as either "phishing" or "legitimate" based on their attributes.
- **Target Variable**: status (categorical: "phishing" or "legitimate").

2.2 Data Structure

- **URL Composition Features**: Metrics describing the structure and components of the URL.
- **Domain and Host Features**: Information related to the domain's behavior, age, and reputation.
- **Content and Behavior Features**: Features describing visual elements, user interaction, and external references.

2.3 Feature Groups

a. URL Composition Features

These features analyze the structure and syntax of the URL.

- **Length and Character Usage:**
  - **length_url**: Total character count in the URL.
  - **length_hostname**: Character count in the hostname portion.
  - **nb_dots, nb_hyphens, nb_at, etc.**: Counts of specific characters like dots, hyphens, and "@" in the URL.
- **Special Indicators:**
  - **ip**: Whether the URL contains an IP address instead of a domain name.
  - **http_in_path**: Presence of "http" or "https" in the URL path (potential obfuscation).
  - **shortening_service**: Indicates the use of URL shorteners like bit.ly.

b. Domain and Subdomain Features

These describe attributes of the domain hosting the URL.

- **Domain Information:**
  - **nb_subdomains**: Number of subdomains present.
  - **random_domain**: Indicates whether the domain appears to be randomly generated.
  - **domain_registration_length**: Duration (in days) of domain registration.
  - **domain_age**: Age of the domain since registration.
- **Behavioral Indicators:**
  - **tld_in_path, tld_in_subdomain**: Presence of the top-level domain (TLD) within paths or subdomains.
  - **suspecious_tld**: Indicates whether the TLD is from a suspicious list.

c. Content and Behavioral Features

These features reflect how the URL behaves and interacts with users.

- **Visual and Interaction Features:**
  - **login_form**: Presence of a login form on the page.
  - **external_favicon**: Usage of a favicon hosted externally.
  - **iframe**: Indicates the presence of iframe elements (potential malicious behavior).
  - **popup_window**: Use of popup windows.
- **Link Analysis:**
  - **nb_hyperlinks**: Total number of hyperlinks on the page.
  - **ratio_intHyperlinks, ratio_extHyperlinks**: Ratios of internal to external hyperlinks.
  - **safe_anchor**: Ratio of "safe" anchor links (e.g., those starting with #).

d. Statistical and Reputation Features

These evaluate the domain's credibility using external data.

- **Statistical Metrics:**
  - **page_rank**: SEO-based ranking of the domain.
  - **web_traffic**: Estimated traffic to the domain.
- **Reputation Indicators:**
  - **google_index**: Indicates if the URL is indexed by Google.
  - **dns_record**: Validity of DNS records for the domain.
  - **domain_with_copyright**: Whether the domain name matches a known copyrighted brand.

2.4 Target Variable

- **Status**: Categorical label indicating the URL classification.
- **Possible values:**
  - **phishing**: URLs designed to deceive users into providing sensitive information.
  - **legitimate**: URLs that are genuine and safe for users.

2.5 Dataset Use Cases

This dataset is designed for tasks such as:

- **Machine Learning Models**: Building classifiers (e.g., logistic regression, decision trees, or deep learning models) to predict phishing URLs.
- **Exploratory Data Analysis**: Identifying key characteristics that differentiate phishing URLs from legitimate ones.
- **Security Analysis**: Assessing potential risks in URLs and strengthening web security protocols.
- **Feature Engineering**: Using statistical and domain-specific indicators to create derived features for enhanced prediction accuracy.

3 Methodology

3.1 Feature Selection

Particle Swarm Optimization (PSO) is used to select the most relevant features for training a machine learning model.

Process:

1. **Fitness Function**:
    - Evaluates the quality of a feature subset.
    - Features are represented by a binary vector (1 for selected, 0 for not selected).
    - Logistic Regression is trained using the selected features, and accuracy is computed on the test set.
    - The fitness value is the negative accuracy to align with PSO's minimization objective.
2. **PSO Optimization**:
    - Defines bounds (lb and ub) for feature selection: 0 (not selected) and 1 (selected).
    - PSO searches for the binary vector that maximizes accuracy by minimizing the fitness function.
3. **Feature Selection**:
    - After optimization, features corresponding to 1 in the binary vector are selected.
    - Training (X_train) and testing (X_test) datasets are reduced to include only the selected features.

3.2 Model Selection

The following machine learning models were implemented and evaluated for their effectiveness in detecting phishing webpages:

1. **Logistic Regression**: A linear baseline model for binary classification.
2. **Support Vector Machines (SVM)**: Effective for high-dimensional data and resistant to overfitting.
3. **K-Nearest Neighbors (KNN)**: A non-parametric algorithm that considers proximity in feature space.
4. **Decision Tree**: A simple and interpretable model for initial insights.
5. **Random Forest**: An ensemble learning approach combining multiple decision trees for improved performance and robustness.
6. **Convolutional Neural Networks (CNNs)**: Designed to extract spatial patterns from features
7. **Recurrent Neural Networks (RNNs)**: Utilized LSTM and GRU layers to capture sequential patterns in the dataset.

3.3 Training and Testing

- **Data Splitting**: The dataset was divided into training (80%) and testing (20%) subsets to ensure reliable evaluation.
- **Cross-Validation**: Applied K-fold cross-validation (k=10) to validate models and reduce the risk of overfitting.
- **Hyperparameter Tuning**: Conducted comprehensive hyperparameter optimization using Grid Search for traditional models and manual tuning for deep learning architectures.
- **Evaluation Metrics**: Performance metrics such as accuracy, precision, recall, F1-score, and AUC-ROC were employed to assess the models.

4 Implementation

4.1 Importing Libraries

The project leveraged an array of robust libraries for data handling, preprocessing, modeling, and evaluation. These include:

- **Data Handling**: pandas, numpy for efficient data manipulation and analysis.
- **Preprocessing**: sklearn.preprocessing tools for feature scaling and label encoding.
- **Modeling**: Implementations of machine learning algorithms such as Logistic Regression, SVM, Random Forest, KNN, Decision Trees, and deep learning architectures (e.g., CNN, RNN) using TensorFlow/Keras.
- **Evaluation**: Performance metrics and visualization tools from sklearn.metrics for comprehensive evaluation.

4.2 Loading the Dataset

The dataset, sourced as dataset_phishing.csv, was loaded using pandas. Initial exploratory analysis ensured the data's integrity and identified any preprocessing requirements.

4.3 Data Preprocessing

To prepare the data for model training, several preprocessing steps were undertaken:

- **Feature Scaling**: Standardization of input features using StandardScaler to maintain uniformity.
- **Label Encoding**: Transformation of target labels into a format compatible with the models.
- **Data Splitting**: Division of the dataset into training and testing sets using train_test_split from sklearn.model_selection to evaluate model performance.

4.4 Transformation Techniques

LSTM/GRU Models:

- **Input reshaping**: The input data is reshaped to fit the LSTM's expected 3D input format (samples, time_steps, features).
- **Recurrent layers**: The LSTM layer is used to capture temporal dependencies, followed by dense layers for nonlinear transformations.
- **Adam optimizer** and **binary crossentropy** loss function are used for efficient training and classification.

CNN Model:

- **1D Convolutional layers**: These layers extract local features by applying filters over the input data.
- **Max pooling**: Reduces the dimensionality of the feature maps to avoid overfitting and improve computational efficiency.
- **Flattening**: Converts the 2D output from the convolutional layers into a 1D vector before passing it to the fully connected layers.
- **Dense layers**: Fully connected layers with ReLU activation for the hidden layers and sigmoid activation for the output.

4.5 Model Development

A variety of models were implemented to explore their effectiveness in phishing detection:

1. Traditional Machine Learning Models:
    - Logistic Regression
    - Support Vector Machines (SVM)
    - Decision Trees
    - Random Forest
    - K-Nearest Neighbors (KNN)

2. Deep Learning Architectures:
    - **Convolutional Neural Networks (CNNs)**: Featuring Conv1D and MaxPooling1D layers for extracting spatial patterns from features.
    - **Recurrent Neural Networks (RNNs)**: Utilizing LSTM and GRU layers for modeling sequential dependencies in data.

5 Evaluation Metrics

This section presents the evaluation of various machine learning models employed for detecting phishing webpages. The models were assessed using a variety of performance metrics to ensure a comprehensive understanding of their effectiveness and efficiency. The metrics include Accuracy, Balanced Accuracy, Area Under the Curve (AUC), F1-Score, Recall, Precision, Matthews Correlation Coefficient (MCC), Log Loss, Youden Index, Brier Loss, Average Precision (AP), and computational time for training and testing.

5.1 Metrics Used

To comprehensively evaluate the models, the following metrics were used:

- **Accuracy**: Overall correctness of predictions.
- **Precision**: Proportion of true positives among predicted positives.
- **Recall (Sensitivity)**: Ability to detect actual phishing URLs.
- **F1-Score**: Harmonic mean of precision and recall, balancing the two.
- **ROC-AUC Curve**: Performance across different thresholds.
- **Log Loss**: Measure of predictive uncertainty.
- **Matthews Correlation Coefficient (MCC)**: Quality of binary classifications, considering true and false positives and negatives.
- **Youden Index**: A metric combining sensitivity and specificity.
- **Brier Loss**: Calibration of predicted probabilities.
- **Average Precision (AP)**: Summarizes the precision-recall curve as a weighted mean.
- **Computational Time**: Time required for training and testing the models.

5.2 Key Observations

1. **Accuracy and AUC**: Random Forest achieved the highest accuracy (95.06%) and AUC (99.06%), highlighting its superior ability to classify phishing webpages correctly.
2. **Efficiency**: Logistic Regression and Decision Tree models demonstrated lower computational overheads with faster training times (0.012 and 0.034 seconds, respectively) compared to other models. KNN also showed quick testing times (0.302 seconds), making these models suitable for scenarios demanding rapid predictions.
3. **Precision and Recall Balance**: Random Forest and CNN exhibited an optimal balance between precision and recall, as reflected in their high F1-scores (0.9498 and 0.9292, respectively). This indicates their reliability in maintaining a good trade-off between false positives and false negatives.
4. **Training Time**: RNN required the longest training time (46.34 seconds), followed by CNN (13.56 seconds), making them less ideal for real-time applications or scenarios demanding rapid updates.

5. **Overall Performance**: Random Forest emerged as the most robust model, with consistently high scores across all metrics, including accuracy, precision, recall, and F1-score. However, Logistic Regression remains a strong contender for applications where computational efficiency is paramount.

6 Results and Discussion

6.1 Findings

- **Random Forest**: The top performer with the highest accuracy (95.06%) and AUC (99.06%), offering a strong balance between precision (95.36%) and recall (94.60%). Despite requiring a longer training time (1.04 seconds), it proved to be the most reliable model overall.
- **Logistic Regression**: Achieved an accuracy of 91.43% and AUC of 96.65%, excelling in speed with the fastest training time (0.012 seconds). This makes it an excellent choice for real-time applications requiring quick computations.
- **Support Vector Machine (SVM)**: Demonstrated high precision (91.19%) and recall (93.53%) with an AUC of 97.55%, but it required the longest training time (4.88 seconds), which limits its use in time-sensitive applications.
- **Neural Network (NN) and Convolutional Neural Network (CNN)**: Showed strong performance with similar accuracy (92.74% and 92.91%, respectively) and AUC (~97.7%). However, they incurred higher computational costs, with training times of 10.55 seconds (NN) and 13.56 seconds (CNN).
- **Decision Tree**: Achieved an accuracy of 92.87% and AUC of 95.37%. While it had slightly lower performance compared to Random Forest and CNN, its low training time (0.034 seconds) made it computationally efficient.
- **K-Nearest Neighbors (KNN)**: Had the lowest accuracy (91.16%) and AUC (96.82%) among the models but was extremely fast in testing, making it suitable for applications requiring quick but less accurate detections.
- **Recurrent Neural Network (RNN)**: Achieved good recall (93.71%) with an AUC of 97.40%, but its slow training time (46.34 seconds) makes it less ideal for scenarios requiring rapid predictions.

6.2 Insights

- Phishing URLs often exhibit certain patterns such as misleading domains, excessive use of special characters, and a lack of SSL certificates. These characteristics make them easier to detect by machine learning models.
- URL Structure: Lengthier URLs with multiple subdomains are more likely to be phishing attempts, as attackers often create longer complex URLs to disguise malicious intent. Such URLs should be flagged for further analysis in any phishing detection system.

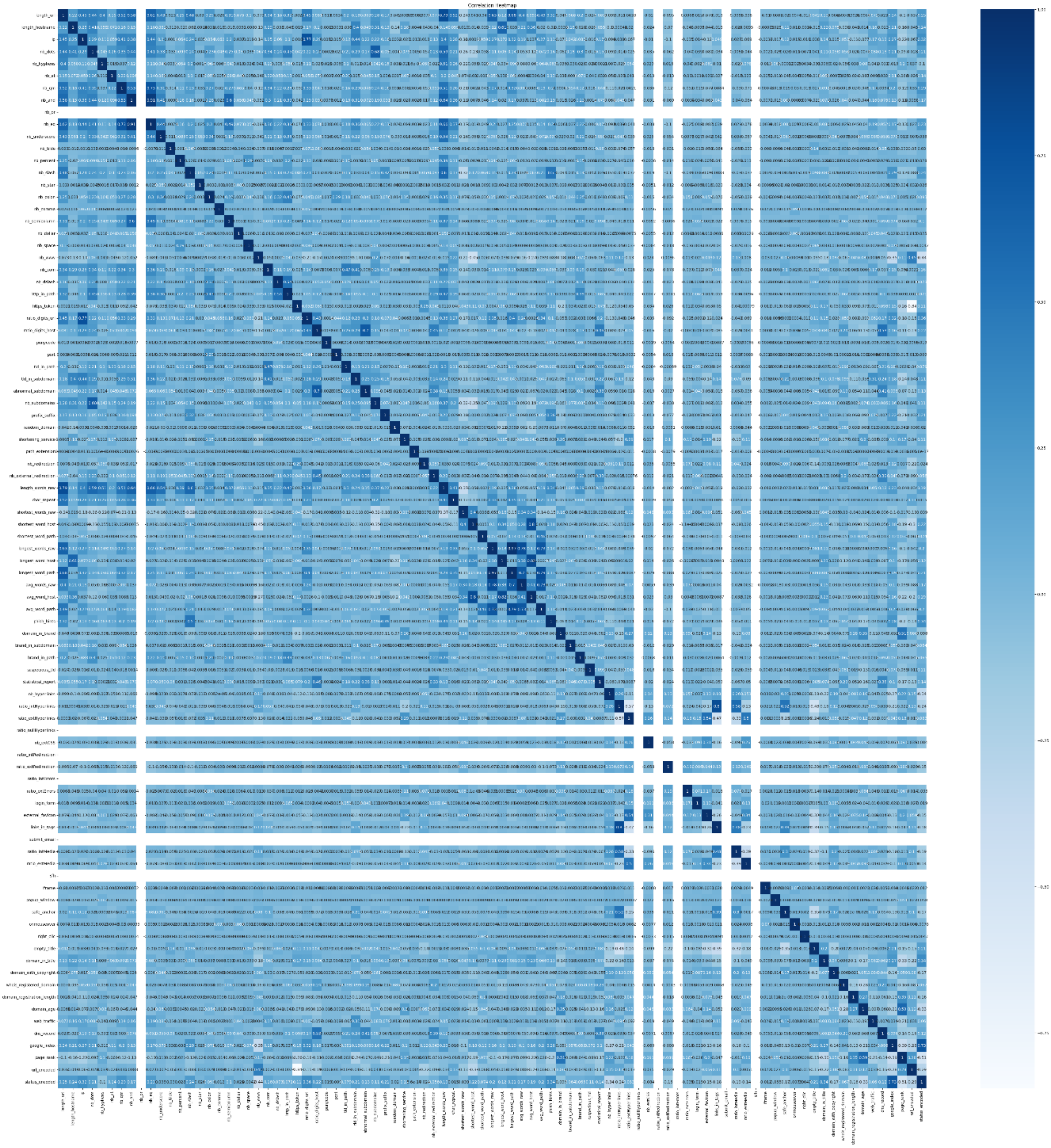
- SSL Certification: The absence of SSL certificates is a key indicator of phishing, as legitimate websites generally employ SSL encryption to secure user data. Models should be trained to prioritize this feature for better detection accuracy.

7 Appendices

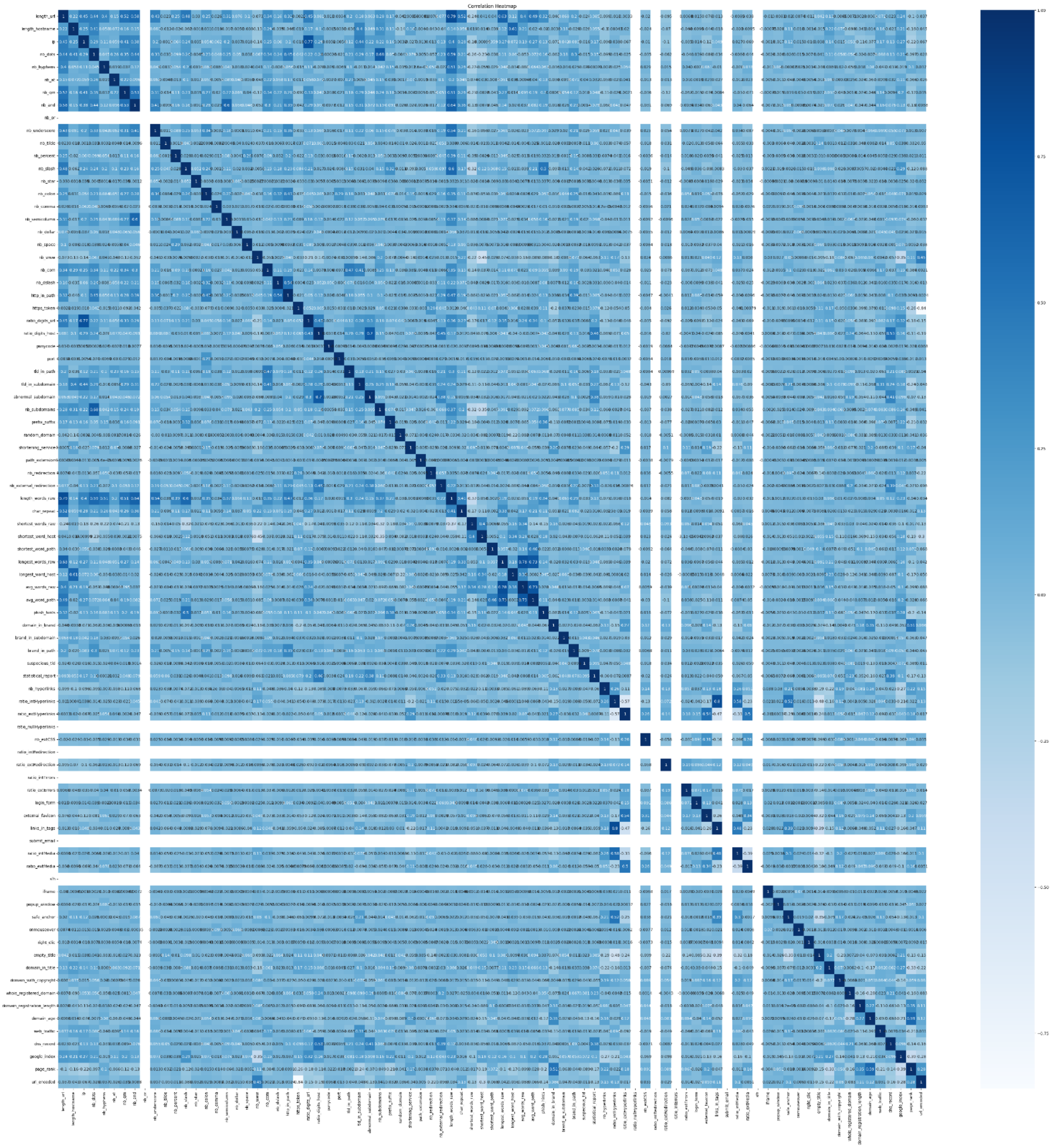
7.1 Table of the machine learning and evaluation metrics

	Accuracy	Balanced Accuracy	AUC	F1-Score	Recall	Precision	MCC	Log Loss	Youden Index	Brier Loss	AP	Training Time	Testing Time
Logistic Regression	0.9143	0.9144	0.9665	0.9141	0.9238	0.9046	0.8287	0.2361	0.8288	0.0666	0.9668	0.012233	0
Decision Tree	0.9287	0.9288	0.9537	0.9287	0.9398	0.9178	0.8577	1.0933	0.8577	0.0605	0.9310	0.034387	0
Random Forest	0.9506	0.9505	0.9906	0.9498	0.9460	0.9536	0.9011	0.1360	0.9010	0.0370	0.9910	1.044327	0.027512
SVM	0.9234	0.9236	0.9755	0.9235	0.9353	0.9119	0.8472	0.2111	0.8472	0.0596	0.9759	4.887992	0.50407
KNN	0.9116	0.9118	0.9682	0.9116	0.9221	0.9013	0.8235	0.2284	0.8235	0.0677	0.9670	0	0.302834
NN	0.9274	0.9274	0.9772	0.9269	0.9327	0.9213	0.8548	0.1911	0.8549	0.0538	0.9773	10.55288	0.239786
CNN	0.9291	0.9293	0.9762	0.9292	0.9415	0.9172	0.8586	0.1955	0.8586	0.0553	0.9775	13.56227	0.208971
RNN	0.9208	0.9210	0.9740	0.9212	0.9371	0.9058	0.8422	0.2083	0.8420	0.0605	0.9741	46.34879	0.562149

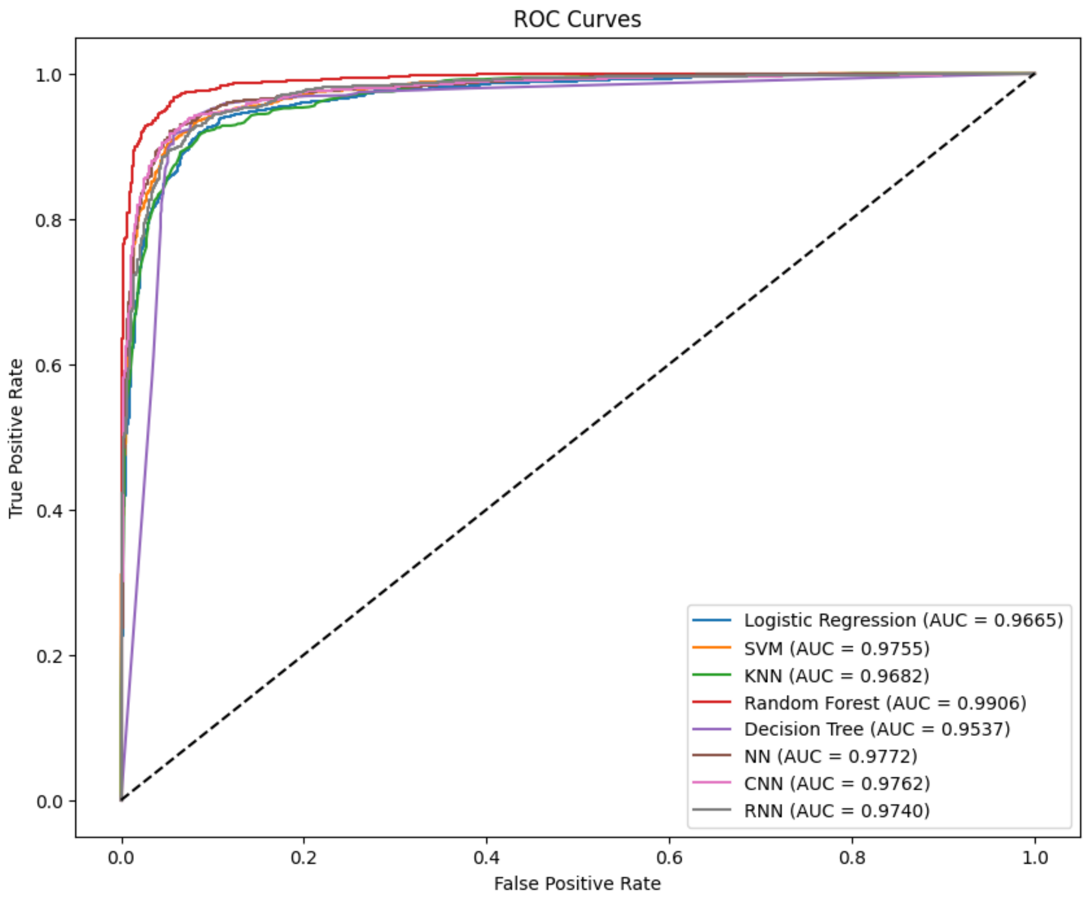
7.2 Correlation Heatmap



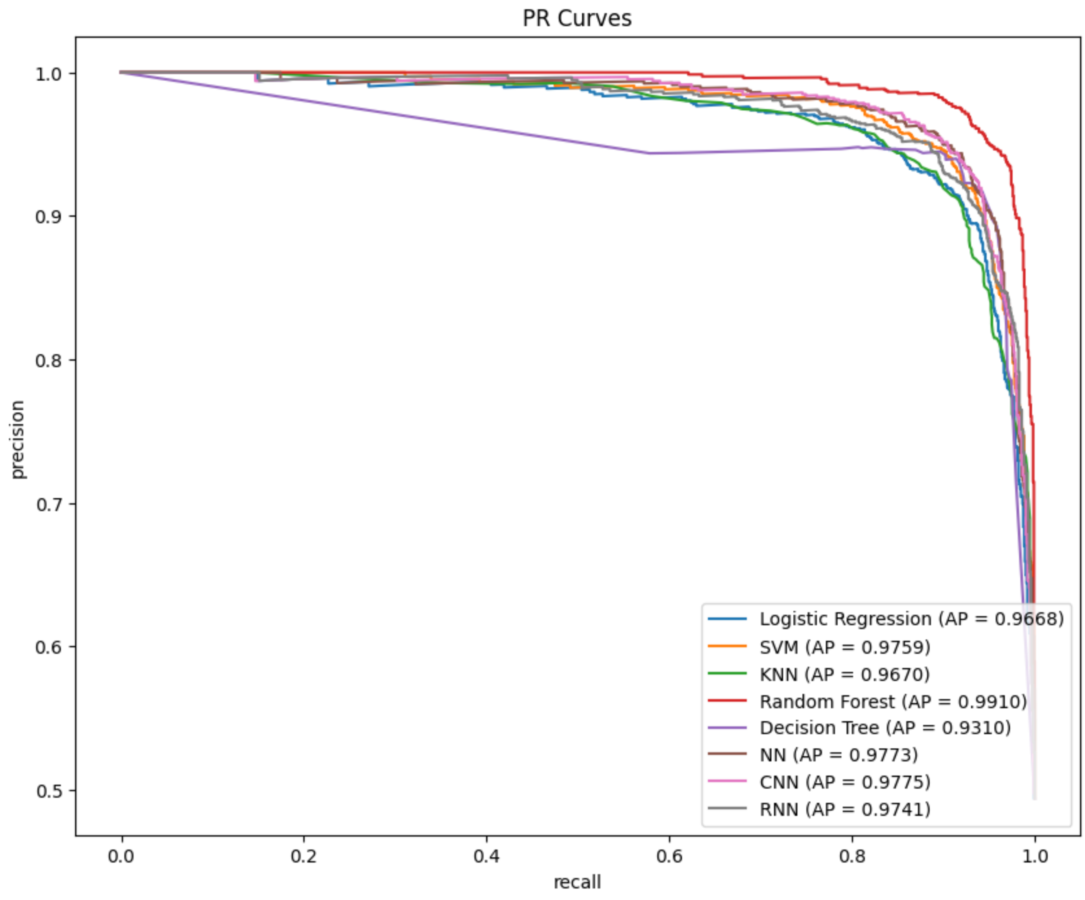
7.3 Correlation Heatmap After Feature Selection



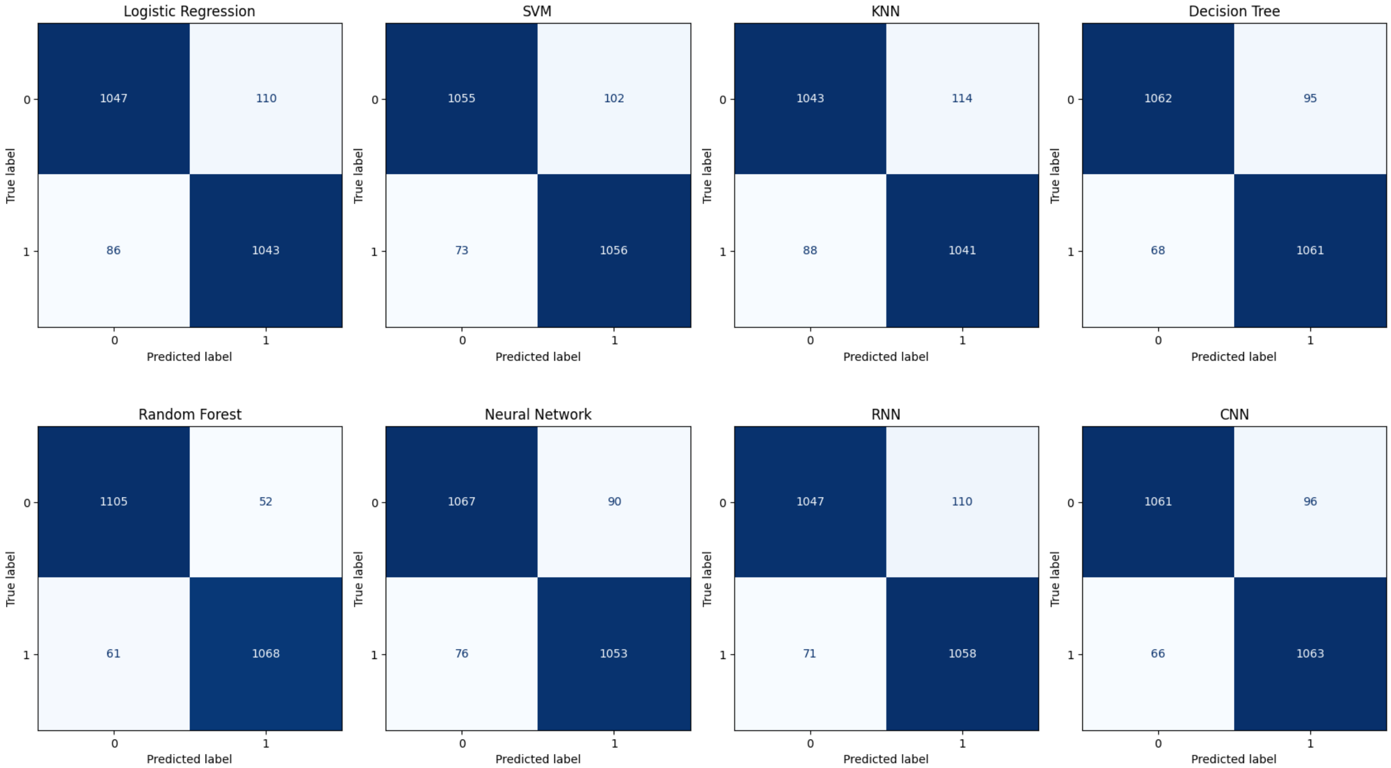
7.4 ROC Curves



7.5 PR Curves



7.6 Confusion Matrix


Logistic Regression
SVM
KNN
Decision Tree
Random Forest
Neural Network
RNN
CNN
True label
Predicted label