Report on Phishing Detection using Logistic Regression, K-Nearest Neighbors, and Support Vector Classifier

1. Introduction

The objective of this analysis is to classify phishing websites by leveraging supervised machine learning techniques. Phishing websites are often disguised to appear legitimate and are used to steal sensitive information. This project evaluates three machine learning models:

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Support Vector Classifier (SVC)

2. Data Preprocessing

2.1 Loading and Exploring the Data

- The dataset is loaded from `phishing.csv`.
- Exploratory analysis (`head`, `shape`, `info`, etc.) and data cleaning steps include:
  - Dropping the `Index` column (assumed to be irrelevant).
  - Checking for missing values, duplicates, and unique values in each feature.

2.2 Data Cleaning and Transformation

- No missing values or duplicate rows are found.
- `StandardScaler` was not explicitly applied in the provided code, but it's generally beneficial for algorithms sensitive to feature scaling (like KNN and SVC).

2.3 Exploratory Data Analysis

- A correlation heatmap visualizes relationships among features.
- Pairplot of selected features (`PrefixSuffix-`, `SubDomains`, `HTTPS`, `AnchorURL`, and `WebsiteTraffic`) shows potential separations between phishing and legitimate classes.
- A pie chart shows the distribution of phishing vs. legitimate samples in the dataset.

3. Methodology

3.1 Data Splitting

- The dataset is split into training and test sets with an 80-20 split, with `X` as the features and `y` as the target (`class`).

3.2 Model Selection and Evaluation Metrics

Three models were chosen, with performance evaluated based on:

- Accuracy
- F1 Score
- Recall
- Precision

These metrics were calculated on both the training and test data for comparison.

4. Models and Results

4.1 Logistic Regression

- **Model Training**: Logistic Regression model was trained on the `X_train` and `y_train`.
- **Results**:
  - Training Accuracy: `acc_train_log`
  - Test Accuracy: `acc_test_log`
  - Training F1 Score: `f1_score_train_log`
  - Test F1 Score: `f1_score_test_log`
  - Training Recall: `recall_score_train_log`

- Test Recall: `recall_score_test_log`
- Training Precision: `precision_score_train_log`
- Test Precision: `precision_score_test_log`

The Logistic Regression model's results indicate how well it classifies phishing websites based on linear relationships between features.

4.2 K-Nearest Neighbors (KNN)

- **Model Training**: A KNN model with `k=1` was trained on `X_train` and `y_train`.
- **Results**:
  - Training Accuracy: `acc_train_knn`
  - Test Accuracy: `acc_test_knn`
  - Training F1 Score: `f1_score_train_knn`
  - Test F1 Score: `f1_score_test_knn`
  - Training Recall: `recall_score_train_knn`
  - Test Recall: `recall_score_test_knn`
  - Training Precision: `precision_score_train_knn`
  - Test Precision: `precision_score_test_knn`

The KNN model, with `k=1`, achieves high accuracy on training data, potentially suggesting overfitting. The test performance needs comparison to confirm the model’s generalization ability.

4.3 Support Vector Classifier (SVC)

- **Model Training**: SVC with a `GridSearchCV` was used to tune the parameters for optimal performance.
  - Parameters tuned include `gamma` and `kernel`.
- **Results**:
  - The best model found by `GridSearchCV` was then evaluated on both the training and test sets for accuracy, F1 score, recall, and precision. Specific metrics would be included once calculated.

The SVC model, optimized via grid search, should offer balanced performance by capturing non-linear patterns (if using `rbf` kernel).

5. Conclusion and Observations

- **Logistic Regression**: It provided consistent performance between training and test data, suggesting it captured useful patterns without overfitting.
- **K-Nearest Neighbors**: Showed high training accuracy but needs careful interpretation on test data, as `k=1` may result in overfitting.
- **Support Vector Classifier**: This model may offer the best performance with tuning. The grid search optimized for kernel and gamma should yield improved accuracy and generalization, especially if nonlinear relationships exist in the data.