Network Security Project
4th Year Computer Engineering

Spam Mails Detection

Student Info:

Student Name	Section Number	Seat Number
Steven Bahaa Zarif	1	43253031
Ali Azouz Ali	1	43253040
Zeyad Ashraf Mahmoud	1	43253028
Mohamed Ossama Amer	2	43253054

Table of Contents

Introduction

In the current era of digital communication, spam emails continue to be a major concern, causing loss of productivity, data breaches, and compromising security. With the vast amount of data being generated daily, traditional manual methods of detecting spam emails are becoming inefficient. The need for automated, intelligent systems that can classify, and filter spam emails accurately has led to the development of machine learning (ML) models. This project aims to explore various ML algorithms and evaluate their effectiveness in detecting spam emails from legitimate ones. By utilizing several machine learning techniques and evaluating their performance, this study will provide insights into the strengths and weaknesses of each approach.

Project Overview

This project involves the development and evaluation of multiple machine learning models to classify spam emails based on a provided dataset. The dataset includes various features that describe the content of the emails, which are used by the models to learn and predict whether an email is spam or not. The following models have been implemented and evaluated:

Traditional Machine Learning Models: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), k-Nearest Neighbors (KNN).
Deep Learning Models: Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN).
The performance of these models will be evaluated using various metrics, including accuracy, precision, recall, F1 score, ROC AUC, Matthews Correlation Coefficient (MCC), and Precision-Recall AUC. The results will be compared to identify the most effective model for spam email classification. The insights gained from this study could be applied to enhance email filtering systems and reduce the risks associated with spam messages.

Dataset description

The dataset used for this project is the Spam Mails Data Set sourced from Kaggle Platform, it is designed for binary classification tasks to classify emails as either spam or not spam. It contains 56 features and a total of 4,601 entries. The target variable is a binary classifier, where 1 represents spam emails and 0 represents non-spam (ham) emails. The dataset is balanced, with roughly equal proportions of spam and non-spam emails. The features describe various characteristics of emails, likely including frequency-based metrics (e.g., count of specific words or characters), statistical attributes (e.g., average word length or number of capital letters), and domain-specific metrics relevant to spam detection (e.g., presence of keywords often found in spam).

preprocessing steps

the dataset has no missing values, eliminating the need for missing value imputation, and it contains no outliers, so outlier removal is unnecessary.

Categorical Features Encoding

label encoding was applied to transform categorical features into numerical format. Using the LabelEncoder class from the sklearn.preprocessing module, categorical columns in the dataset were identified and encoded. This preprocessing step ensures that machine learning algorithms can interpret these features effectively.

Feature Scaling

Since the features' ranges varied significantly - from thousands to fractions - we standardized all features using sklearn.preprocessing's StandardScaler. This transformed the data so each feature has a mean of 0 and standard deviation of 1. Standardization was essential because many machine learning algorithms, especially those using gradient-based optimization or distance metrics, are sensitive to feature scales. This preprocessing step helps ensure stable and efficient model training.

Feature Selection

In our Project we used the Particle Swarm Optimization (PSO) for feature selection, PSO is a nature-inspired optimization algorithm based on the social behavior of birds or fish. PSO aims to find an optimal solution by having a group of particles (potential solutions) explore the search space. Each particle (solution) updates its position and velocity iteratively, guided by its own best position ("personal best") and the global best position found by the swarm (other particles).
In feature selection, PSO is applied to identify the optimal subset of features that improve the performance of a predictive model while reducing redundancy and noise in the dataset. The implementation was done with the help of pyswarms library.

How PSO Works for Feature Selection

1. **Initialization**:
   - Each particle represents a potential feature subset. Its position in the search space is encoded as a binary vector, where each dimension corresponds to a feature ('1' indicates selection, '0' indicates exclusion).
   - The swarm is initialized with n_particles=10 particles, and their velocities are set randomly.

2. **Objective Function**:
   - The objective function evaluates the performance of a model (e.g., a Random Forest classifier) using the features selected by a particle. In this case, cross-validation accuracy is used as the metric.
   - The objective is to minimize the cost encouraging higher accuracy for the selected features.

3. **Updating Particles**:
   - Each particle updates its velocity and position based on:
     - Its personal best solution.
     - The global best solution found by the swarm.
   - The velocity update is governed by three components:
     - Inertia weight ('w'): Encourages particles to continue moving in their current direction.
     - Cognitive coefficient ('c1'): Pulls particles toward their personal best.
     - Social coefficient ('c2'): Pulls particles toward the global best.

4. **Iterative Search**:
   - The swarm iteratively updates its particles for a specified number of iterations (iter=300), refining the search for the optimal feature subset.

5. **Output**:
   - The algorithm returns the global best solution: a binary vector indicating the selected features.

Feature Selection Results

Applying Particle Swarm Optimization (PSO) as a feature selection algorithm led to a reduction in the number of selected features from 58 to 48. This reduction in features contributed to improved efficiency, as the training and testing times for most of the eight models were noticeably reduced. Additionally, the performance of the models improved across various evaluation metrics. With fewer features, the models were able to focus on the most relevant information, leading to enhanced accuracy, precision, recall, and other performance indicators. Overall, PSO not only streamlined the models by reducing complexity but also boosted their effectiveness, demonstrating the value of feature selection in improving both speed and accuracy. See our results in the excel sheets provided with this document.

Models Used

1. Logistic Regression

- **Description**: Logistic Regression is a statistical method used for binary classification tasks. It predicts the probability that an instance belongs to a particular class. Despite its simplicity, it is effective in situations where the data is linearly separable. It uses the logistic function (sigmoid) to output a probability value between 0 and 1.
- **Use Case**: Effective when the relationship between the features and target is approximately linear, often used for problems like spam email classification.

2. Decision Tree

- **Description:** A Decision Tree model is a supervised learning algorithm that splits the data into subsets based on feature values. It forms a tree-like structure where each node represents a feature, each branch represents a decision rule, and each leaf represents the outcome. Decision Trees are easy to interpret and can handle both numerical and categorical data.
- **Use Case:** Used for classification tasks where interpretability and understanding of the decision-making process are important.

3. Random Forest

- **Description**: Random Forest is an ensemble learning method based on Decision Trees. It creates a collection of decision trees by randomly selecting subsets of data and features for training. The final prediction is made by aggregating the results from all the individual trees. Random Forest is less prone to overfitting compared to a single Decision Tree.
- **Use Case**: Suitable for complex datasets with high dimensionality. It is robust and widely used for classification tasks like spam detection.

4. Support Vector Machine (SVM)

- **Description:** SVM is a powerful classifier that finds the hyperplane that best separates the classes in the feature space. It works well in high-dimensional spaces and is effective for both linear and non-linear classification tasks using the kernel trick. SVM can also provide probabilities for classification by using the Platt scaling method.
- **Use Case:** Useful for text classification tasks like spam detection, especially when the data is non-linear and high-dimensional.

5. k-Nearest Neighbors (KNN)

- **Description:** KNN is a simple, instance-based learning algorithm. It classifies a new data point by considering the k nearest neighbors in the training set. The majority class among the neighbors determines the class of the point. KNN is non-parametric, meaning it doesn't make any assumptions about the underlying data distribution.
- **Use Case:** Effective for problems where the decision boundary is irregular. It is simple and works well for small datasets but may suffer from high computational cost in large datasets.

6. Recurrent Neural Network (RNN)

- **Description**: An RNN is a type of neural network designed for sequential data. Unlike traditional neural networks, RNNs have connections that loop back on themselves, allowing them to retain information from previous steps in the sequence. This makes RNNs well-suited for time-series data or tasks involving sequential dependencies, such as text classification.
- **Use Case**: Used in applications where the order of the data matters, such as speech recognition, sentiment analysis, or spam email classification.

7. Convolutional Neural Network (CNN)

- **Description**: CNNs are deep learning models commonly used for image classification and processing. They consist of convolutional layers that apply filters to detect patterns in the input data. CNNs are especially useful for extracting local features and capturing spatial relationships within the data.
- **Use Case**: While typically used for image-related tasks, CNNs have been adapted for structured data classification, like spam email classification, due to their ability to learn hierarchical feature representations.

Evaluation Metrics Descriptions

1. Accuracy

- **Description**: Accuracy is the ratio of correctly predicted instances to the total instances. It is one of the simplest and most commonly used metrics.
- **Use Case**: It works well when the dataset is balanced, but it may be misleading in the presence of imbalanced classes (e.g., when most emails are not spam).

2. Balanced Accuracy

- **Description**: Balanced accuracy is the average of recall obtained on each class. It is especially useful when dealing with imbalanced datasets, as it adjusts for class distribution.
- **Use Case**: Useful in spam email detection when the number of spam and non-spam emails is imbalanced.

3. Precision

- **Description**: Precision measures the proportion of positive predictions that are actually correct. It answers the question, "Of all the emails classified as spam, how many are actually spam?"
- **Use Case**: Important when the cost of false positives (non-spam classified as spam) is high, e.g., when legitimate emails are wrongly classified as spam.

4. Recall (Sensitivity)

- **Description**: Recall measures the proportion of actual positives that are correctly identified. It answers the question, "Of all the actual spam emails, how many were correctly classified as spam?"
- **Use Case**: Important when missing spam emails (false negatives) is costly, such as when spam emails may contain important information or threats.

5. F1 Score

- **Description**: The F1 Score is the harmonic mean of precision and recall. It is a more balanced metric when there is a trade-off between precision and recall.
- **Use Case**: It is particularly useful when the class distribution is imbalanced and both precision and recall are important.

6. ROC AUC (Receiver Operating Characteristic - Area Under Curve)

- **Description**: The ROC AUC measures the ability of the model to distinguish between classes. It plots the true positive rate against the false positive rate at various thresholds. AUC is the area under the ROC curve, which provides a single value representing model performance across all thresholds.
- **Use Case**: Ideal for evaluating classifiers when you want a model that works well for both class separation and threshold-based decisions.

7. Matthews Correlation Coefficient (MCC)

- **Description**: MCC is a more balanced metric that takes into account all four confusion matrix categories: true positives, true negatives, false positives, and false negatives. It returns a value between -1 (perfectly wrong) and +1 (perfectly correct). A score of 0 indicates random predictions.
- **Use Case**: Useful for imbalanced datasets as it provides a more reliable metric than accuracy, especially when the dataset contains many more non-spam emails than spam emails.

8. Model Complexity

- **Description**: Model complexity refers to the computational resources required by the model during training and inference. It includes factors like training time, memory usage, and the number of parameters in the model.
- **Use Case**: Useful when comparing models to determine which one is more efficient in terms of computational cost, especially for real-time applications like spam detection in large email systems.

9. Adaptability

- **Description**: Adaptability refers to a model's ability to learn from new data and adjust to previously unseen patterns. This is important for tasks like spam detection, where spam patterns evolve over time.
- **Use Case**: Evaluates how well a model can handle new types of attacks or changes in data distribution without needing retraining from scratch.

Additional Metrics Considered:

10. AUC-PR (Area Under Precision-Recall Curve)

- **Description**: AUC-PR is a metric similar to ROC AUC but focuses on precision and recall, which is particularly useful for imbalanced datasets. It measures the trade-off between precision and recall for different threshold values.
- **Use Case**: Useful when you need to evaluate the performance of the model in classifying the minority class (spam emails) in highly imbalanced datasets.

Results & Comparison

Two Excel sheets created by our project to store the results, the sheets have the same structure, they contain evaluation metrics for eight machine learning models: Logistic Regression, Decision Tree, Random Forest, SVM, KNN, Neural Network, RNN, and CNN. The metrics include Accuracy, Balanced Accuracy, Precision, Recall, F1-score, ROC-AUC, Training Time, Testing Time, MCC, PR-AUC, and Cohen Kappa Score. The first sheet, "model_evaluation_metrices_.xlsx" presents the results before feature selection, while the second sheet, "model_evaluation_metrices_after_feature_selection.xlsx" shows the results after feature selection has been applied.

Model performance comparison and key findings

- **Accuracy**: Random Forest achieves the highest accuracy, followed closely by Neural Network.
- **Precision**: Random Forest has the best precision, indicating fewer false positives.
- **Recall**: Both CNN and RNN excel in recall, meaning they perform well in identifying true positives.
- **F1-Score**: : Random Forest achieves the highest F1-Score, balancing precision and recall effectively.
- **ROC-AUC**: Neural Network, CNN and RNN outperform all others in ROC-AUC.
- **MCC and Cohen Kappa Score**: CNN and Neural Network score highest here, reflecting strong overall performance.

Overall Summary:

- Random Forest: Best overall performer across accuracy, F1-score, MCC, and Cohen Kappa Score.
- Neural Network: Excels in ROC-AUC and PR-AUC, with strong overall metrics.
- CNN: Performs well in recall, ROC-AUC, and PR-AUC.
- RNN: Performs well in recall, ROC-AUC, and PR-AUC.
- SVM: Strong precision but the slowest in training and testing.
- Logistic Regression: Fastest testing time but moderate performance overall.
- KNN: Fast training but lower performance compared to neural models.
- Decision Tree: almost small training and testing time.

Conclusion

Summary of results and insights

In this project, we explored and evaluated multiple machine learning models to address the problem of spam email classification. We implemented traditional models such as Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), and k-Nearest Neighbors (KNN), as well as deep learning models like Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN).

We also saw that the application of Particle Swarm Optimization (PSO) for feature selection effectively optimized the machine learning models by reducing the number of features from 58 to 48. This reduction not only shortened the training and testing times but also enhanced the models' performance across key evaluation metrics.

Through comprehensive evaluation using various performance metrics such as accuracy, precision, recall, F1 score, ROC AUC, Matthews Correlation Coefficient (MCC), and Precision-Recall AUC, we were able to assess the strengths and limitations of each model. The results showed that while traditional models provide reliable results, the deep learning models demonstrated superior adaptability and performance in detecting spam emails, especially when dealing with complex patterns in the data.

This study highlights the importance of choosing the right model for specific tasks, as well as the value of evaluating models through multiple metrics to ensure comprehensive performance assessment. The findings can be applied to improve existing spam filtering systems, making them more efficient and capable of handling a diverse range of email content.

Overall, while deep learning models such as RNN and CNN showed promising results, future work could explore further optimizations and the incorporation of more advanced techniques such as ensemble methods or hybrid models for even better performance.