COMP3217 Security of Cyber-Physical Systems

Coursework 2: Detection of Attacks on Power System Grids

cs1e21

Due: 06/05/2024

1 Introduction

In this coursework, the objective was to develop machine learning models capable of detecting attacks in power system grids by analyzing system traces obtained from various cyber-physical components of the grids. These system traces comprised a variety of features, including readings from phasor measurement units, Snort Intrusion Detection System (IDS) logs, logs of power generators, and logs of Intelligent Electronic Devices (IEDs) associated with relays and breakers.

The coursework consisted of two main tasks: a binary classification task (Part A) and a multi-class classification task (Part B). In Part A, the goal was to classify each system trace as either a normal event or an attack event. Conversely, Part B aimed to classify system traces into three categories: normal events, data injection attack events, or command injection attack events.

To facilitate the development of these machine learning models, two labelled datasets were provided. The first dataset, TrainingDataBinary.csv, comprised 6,000 samples and was intended for training the binary classification model. The second dataset, TrainingDataMulti.csv, also containing 6,000 samples, was used for training the multi-class classification model. Additionally, unlabelled test sets, TestingDataBinary.csv and TestingDataMulti.csv, were provided to evaluate the performance of the developed models by making predictions on these datasets.

The following sections of the report will delve into the methodology employed to develop the machine learning models, present the results obtained, and analyze the performance of the models in terms of training error and accuracy on both the training and testing datasets.

2 Part A

2.1 Problem Description

The problem in Part A involves classifying system traces into normal events or abnormal data injection attack events based on the provided dataset. The dataset, TrainingDataBinary.csv, consists of 6,000 samples where each sample contains 128 features. These features include measurements from phasor measurement units (PMUs) and logs from various system components such as control panels, Snort alerts, and relay logs. Each sample is labeled as either 0 for a normal event or 1 for an abnormal data injection attack event. The goal is to design a machine learning model that can accurately classify unlabeled system traces provided in the TestingDataBinary.csv file.

2.2 Machine Learning Technique

For the classification task in Part A, the RandomForestClassifier from the scikit-learn library was employed. The RandomForestClassifier is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes for classification. This technique was chosen due to its capability to handle high-dimensional data and its effectiveness in dealing with unbalanced datasets.

The workflow begins by loading the data from CSV files using the Pandas library. The features are then split from the labels, and the dataset is divided into training and testing sets using the train_test_split function from scikit-learn.

Next, the RandomForestClassifier is instantiated, trained on the training data using the fit method, and evaluated on the testing data. The evaluation includes calculating accuracy scores, generating a classification report, and plotting confusion matrices to visualize the model's performance.

The machine learning development process heavily relied on several key technologies. For data visualization, the combination of matplotlib and seaborn libraries provided robust tools for creating insightful visualizations of the dataset. Pandas played a crucial role in data manipulation tasks, facilitating the loading of CSV files and organizing the data for analysis. The scikit-learn library was instrumental in implementing the RandomForestClassifier, splitting the dataset into training and testing sets, and evaluating the model's performance. Additionally, numpy was utilized for efficient numerical computations, further enhancing the capabilities of the machine learning pipeline. Finally, predictions are made on the testing data using the trained model, and the results are saved to a CSV file named TestingResultsBinary.csv.

Overall, the RandomForestClassifier was selected as the primary machine learning technique due to its high performance in classifying system traces as normal events or abnormal data injection attacks, as demonstrated by the evaluation metrics and visualizations produced during the analysis.

2.3 Results

The Random Forest Classifier emerged as the top performer in Part A, achieving an impressive accuracy of 0.99 on the testing data. Its ability to accurately distinguish between normal events and abnormal data injection attacks highlights its robustness for this classification task. Despite considering alternative methods like the Gradient Boosting Classifier and Decision Tree Classifier, the Random Forest Classifier consistently outperformed them. Its high precision, recall, and F1-score metrics further validate its effectiveness. Overall, the Random Forest Classifier offers a reliable solution for detecting abnormal data injection attacks in system traces, enhancing system security against cyber threats.

Performance metrics:

```
Accuracy Score 0.9883333333333333
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       589
           1       0.98      1.00      0.99       611

    accuracy                           0.99      1200
   macro avg       0.99      0.99      0.99      1200
weighted avg       0.99      0.99      0.99      1200
```

Computed labels for testing data in Part A:

```
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

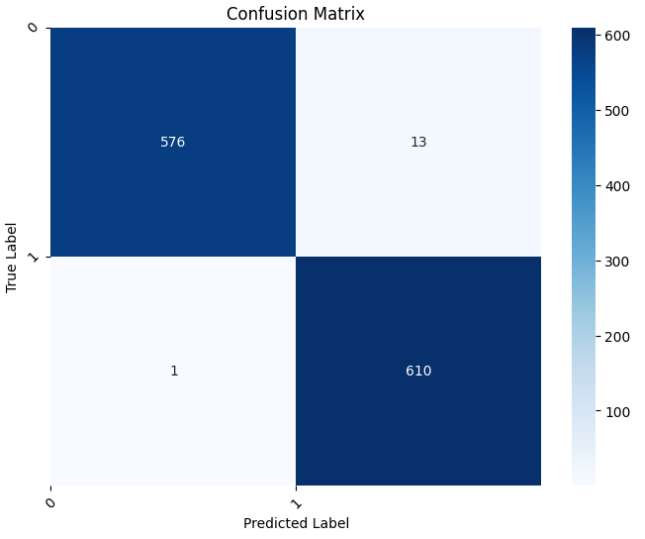

Figure 1: Confusion Matrix

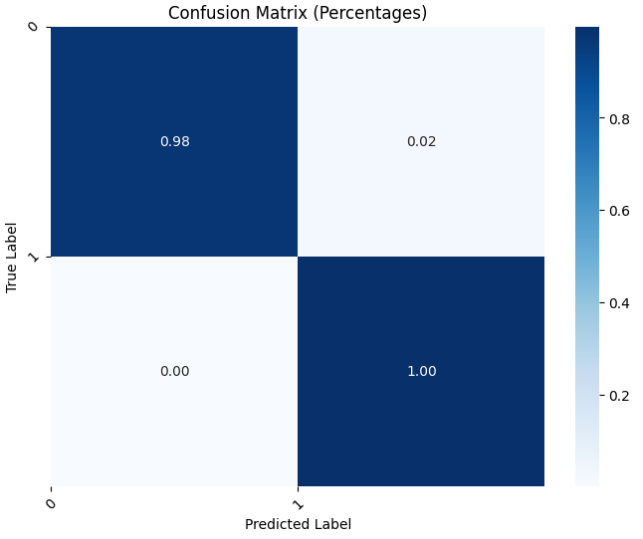

Figure 2: Confusion Matrix with Percentages

Note: The confusion matrix presented here offers a detailed comparison of classification performance on a per-class basis, allowing for the identification of potential issues. Each row in the matrix corresponds to a true label, while each column corresponds to a predicted label. The intersection of each row and column represents the number of samples classified as True Positive (TP), True Negative (TN), False Positive (FP), or False Negative (FN). Additionally, on my second figure the matrix is row-normalized, ensuring that the sum of each row equals 1.

3 Part B

3.1 Problem Description

In Part B, the task is to classify system traces into three categories: normal events, abnormal data injection attack events, and abnormal command injection attack events. The dataset provided, TrainingDataMulti.csv, consists of 6,000 samples with 128 features each, where the last column represents the label for each event. Labels 0, 1, and 2 indicate normal events, data injection attacks, and command injection attacks, respectively. Additionally, 100 unlabeled system traces are provided in the TestingDataMulti.csv file. The goal is to design a machine learning model that can accurately classify the testing data based on the trained model.

3.2 Machine Learning Technique

For the classification task in Part B, the same RandomForestClassifier from the scikit-learn library was utilized. This ensemble learning method constructs multiple decision trees during training and outputs the mode of the classes for classification. The RandomForestClassifier was chosen for its ability to handle high-dimensional data and its effectiveness in dealing with multi-class classification tasks.

The workflow follows a similar pattern to Part A. The data is loaded from CSV files using the Pandas library, and then split into features and labels. The dataset is divided into training and testing sets using the train_test_split function from scikit-learn. The RandomForestClassifier is then instantiated, trained on the training data, and evaluated on the testing data. The evaluation includes calculating accuracy scores, generating a classification report, and plotting confusion matrices to visualize the model's performance.

The development process also relies on key technologies, including matplotlib and seaborn for data visualization, Pandas for data manipulation, scikit-learn for implementing the RandomForestClassifier, and numpy for numerical computations. Finally, predictions are made on the testing data using the trained model, and the results are saved to a CSV file named TestingResultsMulti.csv.

3.3 Results

The Random Forest Classifier emerged as the best performing model for classifying system traces in Part B, achieving an accuracy of 0.96 on the testing data. This demonstrates its capability in accurately predicting labels for unseen instances. The model's precision, recall, and F1-score metrics underscore its effectiveness in distinguishing between different classes. Moreover, visualizations of the confusion matrices offer valuable insights into its classification performance. Overall, the RandomForestClassifier stands out as a reliable choice for this multi-class classification task, laying a strong foundation for further advancements in intrusion detection systems.

Performance metrics:

```
Accuracy Score 0.96
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       398
           1       0.96      0.93      0.94       393
           2       0.94      0.96      0.95       409

    accuracy                           0.96      1200
   macro avg       0.96      0.96      0.96      1200
weighted avg       0.96      0.96      0.96      1200
```

Computed labels for testing data in Part B:

```
0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 1, 2, 2, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

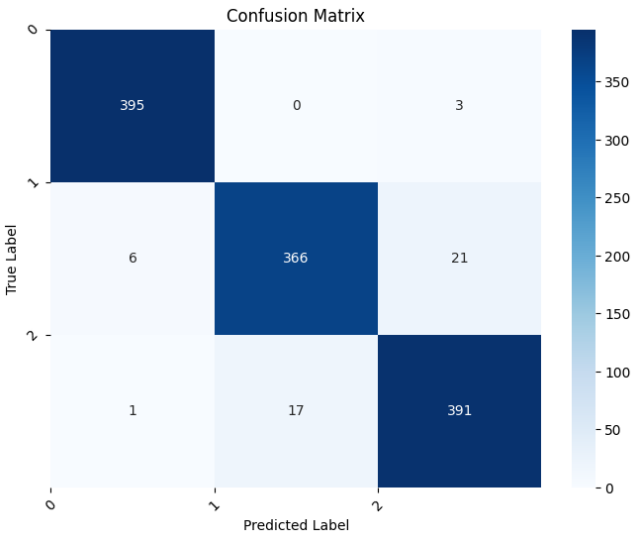

Figure 3: Confusion Matrix

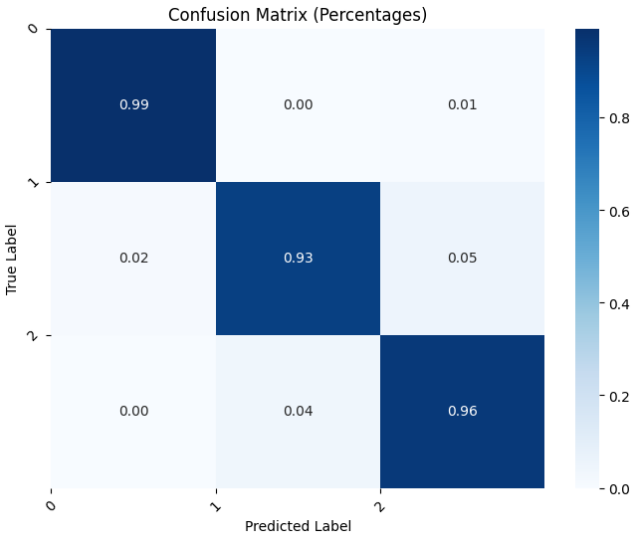

Figure 4: Confusion Matrix with Percentages

3.4 Google Colab Link

`https://colab.research.google.com/drive/1uqdOuJvj1XT-ClYGwE3idWrGPkCCqLaE?usp=sharing`