COMP3217 COURSEWORK 2

MUHAMAD ABEED SAHARUDIN (STUDENT ID: 33371806)

May 10, 2024

1 Problem Statement

The problem that were given is to detect physical cyber-attacks on a power system framework configuration using machine learning (ML). There are many different scenarios that can happen on the framework that can indicate whether a physical cyber attack is targeted towards the framework [3], however for this coursework only 3 are described and analysed, which are natural faults, data and remote command injection attacks, and normal events.

This coursework is divided into 2 parts. Part A is concerned with only detecting between normal events and data injection attack events. The second part is concerned with the same events, with the addition of remote command injection attack events.

This report explains the data composition, ML pipeline design, and training evaluation for both parts of the courswork.

2 Data Composition

The training data given to train the ML algorithm are 2 `csv` files, each containing 6,000 system traces, with each of those containing 128 features. The first training file is for part A of this coursework, where the last column is a marker that serves as a label indicating whether an event to the framework is a normal event (labelled as 0) or a data injection attack event (labelled as 1). The second file is for the second part of this coursework, which contains the same data marker, with the addition of the third event, a remote command injection attack (labelled as 2).

Figure 1 shows the label distribution of the dataset for part A. With the distribution being balanced on both class, it is certain that no class will dominate the other and the training performance of the ML model is guaranteed to be effective.

Figure 2 then shows the label distribution of the dataset for part B. Similar to part A, the distribution of labels in part B are even throughout all classes. As a consequence, the ML model will effectively train on this dataset, since there is no class imbalance to skew or bias the ML model into one particular class of labels.

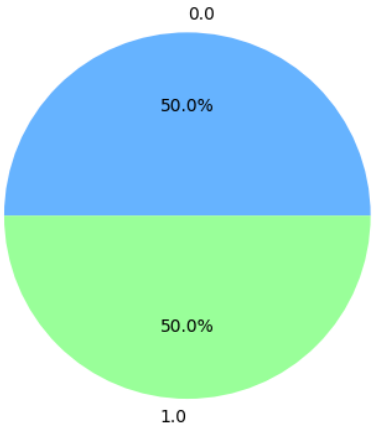

Figure 1: pie chart shows distribution of labels in dataset of Part A

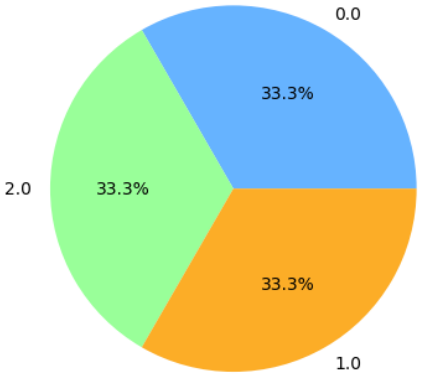

Figure 2: pie chart shows distribution of labels in dataset of Part B

3 ML Pipeline Design

It is required that the design of the ML workflow be done in Google Notebook. The links are provided in the References section [1, 2].

3.1 Pre-Processing

Before training, data pre-processing was done to better prepare the ML models for maximum performance on training and better generalisation on unseen data. One of the pre-processing methods used is standardising the dataset so its data distribution has mean 0 and unit variance. This helps prevent a particular feature from overwhelming other features during training, which might contribute to unfair bias of the ML model towards any feature.

3.2 ML Models

For this coursework, the `RandomForest` classifier from `scikit-learn` was chosen and used to train on the dataset. Other ML models are considered, such as `Logistic` regression and `Support Vector Machines` (`SVM`) classifier, but their performance pales in comparison to `RandomForest` classifier, both in terms of accuracy and f1-score. This remains true for both parts A and B.

3.3 Hyperparameter Tuning

The hyperparameters for `RandomForest` classifier are tuned to give the best performance for it. It is also important to tune the hyperparameters to avoid overfitting.

The hyperparameters are not manually tuned; it is instead searched automatically via script using `GridSearchCV`, which is also provided from `scikit-learn`. This method

of hyperparameter tuning exhaustively searches for the best hyperparameters, by testing each combination of hyperparameters possible and comparing the scores of each combination. The score is user-defined, and in the case of this coursework, the score `f1-score` is chosen over `accuracy` to better reflect the generalisation performance of the ML model. The best hyperparameters is then used to predict the test set.

4 Training Evaluation

Since there is no ground truth in the test set, the predicted dataset cannot be evaluated. However, the given training set does have ground truth, and so is used to split further into a smaller training set and a validation set. The split is done with 80% training set and 20% validation set. This validation set is then used to evaluate the performance of the ML model by measuring its accuracy and f1-score. Figure 3 shows a confusion matrix display for part A, while figure 4 shows a confusion matrix for part B.

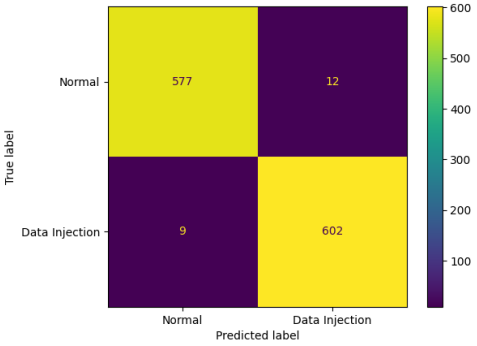

Figure 3: confusion matrix for binary classification

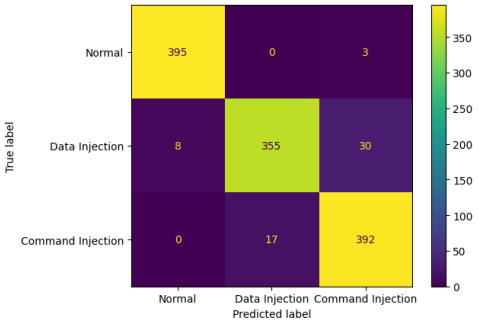

Figure 4: confusion matrix for multi-class classification

From the confusion matrices, the score for both training accuracy and f1-score for part A is 98.3%, while the score for training accuracy and f1-score for part B is 95.2% and 95.1% respectively. This shows that the trained `RandomForest` classifier from part A performs better than the trained `RandomForest` classifier from part B. This can be due to the increase of labels in part B, making it more difficult for the ML model to generalise well on the dataset from part B.

5 Conclusion

In conclusion, the `RandomForest` classifier is the best classification algorithm for this dataset. Its hyperparameters are further tuned to give even more performance to the classifier. Finally, the classifier is evaluated on validation set which is separated early on from the given dataset, since the given test set does not ground truth.

6 References

[1] *Google Colab*. URL: https://colab.research.google.com/drive/1D-d-fE6HDtyurjdwe30B6qVHKBR9
scrollTo=yyC0y3AKpFEh.

[2] *Google Colab*. URL: https://colab.research.google.com/drive/1LSB2kUiGJ3uFcUVLtb6Fy5C2c3EP
scrollTo=yyC0y3AKpFEh.

[3] Mississippi State University and Oak Ridge National Laboratory. *Power system attack datasets*. Apr. 15, 2014. URL: http://www.ece.uah.edu/~thm0009/icsdatasets/PowerSystem_Dataset_README.pdf.