COMP3217 Coursework 2: Detection of Attacks on Power System Grids

Ying Ch’ng [ID: 32492715]
yzc1c20@soton.ac.uk

1 Problem Description

The security of cyber-physical systems (CPSs) in critical infrastructures such as power system grids is important, as attacks on these CPSs would cause power outages that can potentially lead to significant financial losses and disrupt the daily lives of the public. Therefore, preventative mechanisms should be implemented to mitigate attacks on such systems. The main task at hand for this coursework is to apply machine learning techniques to detect attacks on power system grid CPS by classifying system traces into normal or abnormal events.

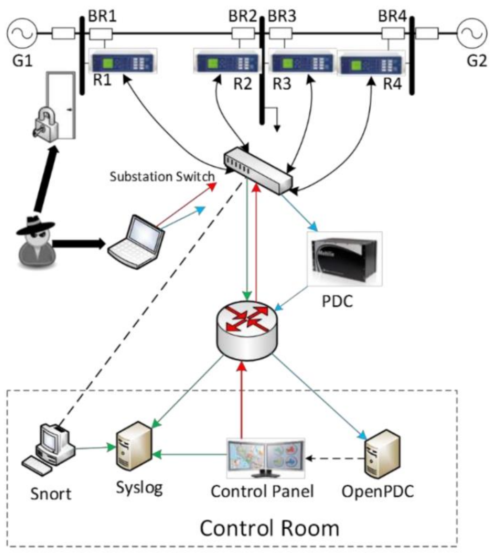


Figure 1: Power system grid CPS considered for the task.

An example of a power system grid CPS is shown in Fig. 1. In Fig. 1, G1 and G2 are power generators, while R1 to R4 are Intelligent Electronic Devices (IEDs) that can switch corresponding breakers (BR1 to BR4) on or off. The IEDs use a distance protection scheme which trips the breakers on detected faults. However, detected faults are not verified to be valid or fake, therefore it is possible for bad actors to compromise the power grid by triggering fake faults through data injection into system parameters such as voltage, current and sequence components.

The breakers can also be manually tripped by issuing commands to the IEDs. This manual override is used when operators need to trip the breakers to perform system maintenance. However, malicious actors who have penetrated outside defenses can also exploit the manual override to trip the breakers by remotely injecting commands to these IEDs.

Thus, three different scenarios can occur in such systems:

1. Normal events during normal operation
2. Disabled system operations due to natural faults and service events
3. Faults due to remote cyber-attacks (data injection/remote tripping command injection)

The power grid monitors its electrical parameters by using Phasor Measurement Units (PMUs) or synchrophasors. The system has 4 PMUs, with each PMU collecting measurements of 29 signals. This brings the total PMU measurements to 116 signals. Apart from using the PMUs' measurements, each of the 4 PMUs provide 3 system logs for system monitoring. The 3 system logs are namely the snort alert, the control panel log, and the relay log. Hence, in total, 128 distinct parameters can be collected to monitor the system.

The coursework is divided into two parts. In both parts, machine learning algorithms will be deployed to train classification models which can detect when the power system grid is compromised by cyber-attacks. The models will be trained by using a set of labelled system traces. The trained models will then be used to classify a provided set of unlabelled system traces to compute labels for all system traces. The machine learning techniques used and the classification results for each part will be further discussed in the following sections.

2 Part A – Binary Classification

2.1 Task Description

To prevent exploitation of the vulnerabilities described in the previous section, a machine learning binary classifier will be deployed to classify system traces between two events, namely whether the CPS is under normal operation, or under a data injection attack.

In terms of data availability, 6000 labelled system traces are provided in the form of a .csv file named TrainingDataBinary.csv. This dataset has a format of 6000 rows and 129 columns. Each row corresponds to a system trace, which has the 128 distinct system parameters mentioned in the previous section, and an event label. All 128 system parameters are given as numerical data, while an event label of ‘0’ indicates normal operation and ‘1’ indicates a data injection attack. The distribution of the event labels is balanced, with 50% of the traces being normal and the rest of the traces being abnormal. Apart from this dataset, another 100 unlabelled system traces are provided in TestingDataBinary.csv. This dataset has a format of 100 rows and 128 columns. The training data will be used to train and evaluate a binary classifier, which will be used to compute the labels for the testing data.

Since the provided training data is labelled and there is no requirement for real-time classification for the task, a supervised offline learning approach will be taken to implement a suitable model for the classifier.

2.2 Algorithm Design and Implementation Techniques

A workplace was created using a conda environment with Python, Jupyter Notebook, scikit-learn, pandas, numpy and matplotlib installed. The code is maintained in a GitHub repository that can be found at https://github.com/ianchngyz/cyphysecCW2. For Part A, the file named “Final Part A.ipynb” contains the best implementation of the machine learning binary classifier.

2.2.1 Data Exploration

Firstly, the training dataset in the .csv file was explored to ensure the data matches the size and format specifications stated in the coursework document. Next, the dataset was checked for invalid data. The dataset was found to not contain any missing values or invalid data. The balance of the event label distribution was verified and visualize in a bar chart shown in Fig. 2. Since there are 128 distinct features, it is not reasonable to perform elaborate feature analysis on each feature. However, from preliminary data exploration, it is found that the 128 features were neither normalized nor scaled. By normalizing or scaling these features, computations can be sped up to make algorithm development faster and more discriminating features can be extracted.

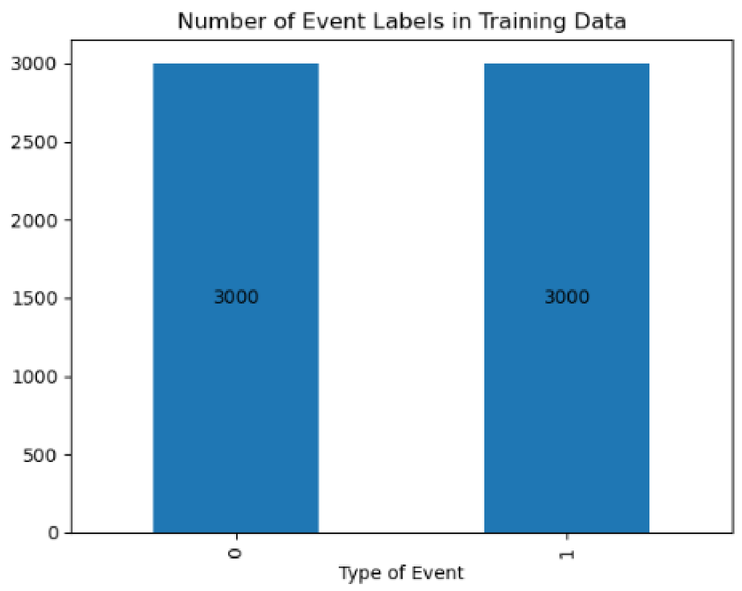


Figure 2: The 50-50 split of event labels in the training data were verified by plotting a bar chart.

2.2.2 Data Preprocessing

Before proceeding further into data preprocessing, the provided training dataset was split into a training set and a test set in an 80-20 manner. This holdout validation strategy was done to prevent data snooping so that completely unseen data can be used to evaluate the model performance after fitting the model. Since there is sufficient data, an 80-20 split was chosen over the conventional 70-30 split in favor of allocating more data to train the model to increase accuracy. To ensure the test set has the same balanced event label distribution, the train-test split is done in a stratified fashion.

No data cleaning was performed since there were no missing values or invalid data found during the data exploration stage. Outliers in features were also kept in the dataset to ensure it can be used to discriminate between the two classes. Feature scaling was done to optimize computation and to avoid data bias by ensuring features are scaled down to the same range. The StandardScaler transformer in scikit-learn was used to standardize all features as this transformation led to the best results when compared to MinMaxScaler and RobustScaler during iterative improvement across different model implementations. The holdout test set and the provided testing data was scaled with the mean and variance of the training set.

2.2.3 Model Selection and Hyperparameter Tuning

Initially, the scaled training dataset was used to train several different models such as logistic regression, k-nearest neighbours (kNN) and support vector classifiers. These different models were implemented with default parameters to compare their baseline performances so that the best model can be shortlisted to be used in the final implementation. The trained models were then evaluated with the 20% holdout test set to determine each model's accuracy and F1 score. Table 1 shows the results of the base implementations of different machine learning models on the allocated test set.

Table 1: Comparison of base model performance on test set for Part A

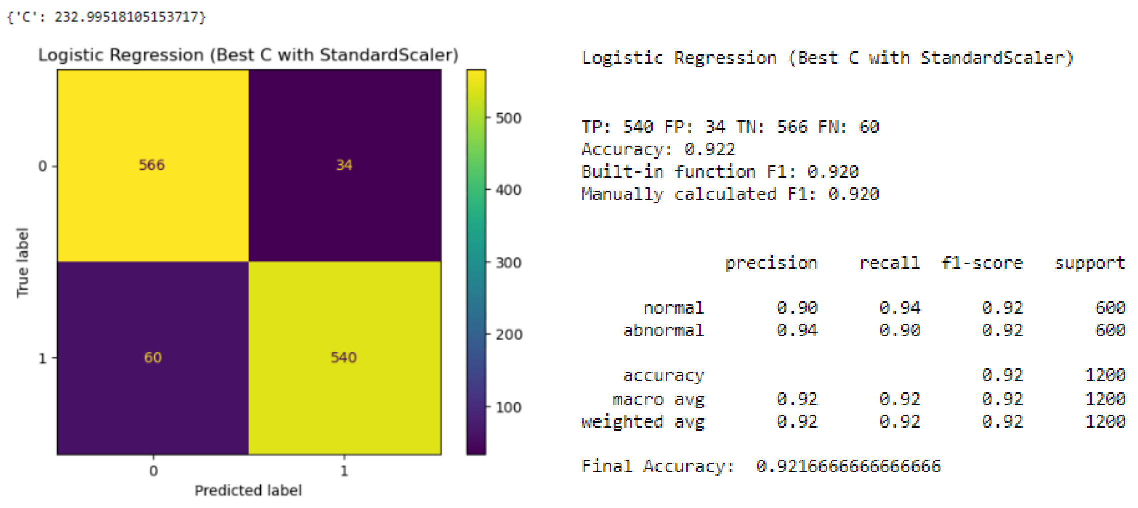
Model	Accuracy	F1 score
Logistic Regression	0.899	0.896
kNN	0.943	0.942
SVC	0.874	0.869
LinearSVC	0.897	0.894

The best model found for the binary classification task was the kNN algorithm, but to prevent overfitting the model, a 5-fold cross validation with hyperparameter tuning was done to further evaluate the models. The hyperparameter tuning was done by using GridSearchCV to find the best parameters to use for each model. Table 2 compares the performance of the tuned models after cross-validation.

Table 2: Comparison of the best-tuned model performance on test set for Part A

Model	Hyperparameter	Accuracy	F1 score
Logistic Regression	C = 234	0.922	0.920
kNN	k = 1	0.952	0.952
SVC	Default	0.874	0.869
LinearSVC	Default	0.897	0.894

As the best hyperparameter for the kNN algorithm is k=1, which is the lower limit of the hyperparameter value, the model may overfit the training data by capturing uncorrelated patterns found in the data features. Therefore, by following Occam's razor, which advises to use a simpler model than a complicated one to prevent overfitting, the second-best algorithm, which is the logistic regression model, was chosen to be implemented as the final classifier design. Fig. 3 shows the final evaluation of the model.



```
{'C': 232.99518105153717}
```

```
Logistic Regression (Best C with StandardScaler)

TP: 540 FP: 34 TN: 566 FN: 60
Accuracy: 0.922
Built-in function F1: 0.920
Manually calculated F1: 0.920

              precision    recall  f1-score   support

      normal       0.90      0.94      0.92       600
    abnormal       0.94      0.90      0.92       600

    accuracy                           0.92      1200
   macro avg       0.92      0.92      0.92      1200
weighted avg       0.92      0.92      0.92      1200

Final Accuracy:  0.9216666666666666
```

Figure 3: Confusion matrix and the classification report of the chosen final algorithm (Logistic Regression, C=234) on the 20% holdout test set.

2.3 Model Evaluation and Results

The final logistic regression model was used to predict the provided testing data in TestingDataBinary.csv. 51 system traces were classified as data injection attacks, while 49 were classified as normal events. Table 3 shows the computed labels for the testing data. According to Fig. 3, the model is more likely to produce false negatives than false positives. This means that it is more prone to misclassify data injection attacks as normal events, rather than misclassifying normal events as data injection attacks. Additionally, since the precision and recall scores are higher than 90%, the model can be considered as neither overfitting nor underfitting.

Table 3: Computed labels for testing data in Part A

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0	0	0
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3 Part B – Multi Classification

3.1 Task Description

Similar from Part A, a machine learning classifier will be deployed, but the classifier for Part B is required to classify system traces between three events, namely whether the CPS is under normal operation, or under a data injection attack, or under a command injection attack.

For this multiclass classification task, 6000 labelled system traces are provided in a .csv file named TrainingDataMulti.csv. Same as Part A, this dataset has a format of 6000 rows and 129 columns. Again, each row corresponds to a system trace, which has 128 distinct system parameters, and an event label. An event label of '0' indicates normal operation, a '1' indicates a data injection attack, while a '2' indicates a command injection attack. The distribution of the event labels is not balanced, with 50% of the traces being normal and the remaining traces being split into data injection attacks and command injection attacks. Another 100 unlabelled system traces are provided in TestingDataMulti.csv. This dataset has a format of 100 rows and 128 columns. The training data will be used to train and evaluate a multiclass classifier, which will be used to compute the labels for the testing data.

As this is a multiclass classification problem, the deployment of ensemble machine learning methods will greatly enhance the performance of the classifier implementation.

3.2 Algorithm Design and Implementation Techniques

The same conda environment used in Part A was used for this task. For Part B, the file named "Final Part B.ipynb" contains the best implementation of the machine learning multiclass classifier.

3.2.1 Data Exploration

The dataset was initially loaded in and checked for invalid data. The dataset was found to not contain any missing values or invalid data. The event label distribution was visualized in a bar chart shown in Fig. 4.

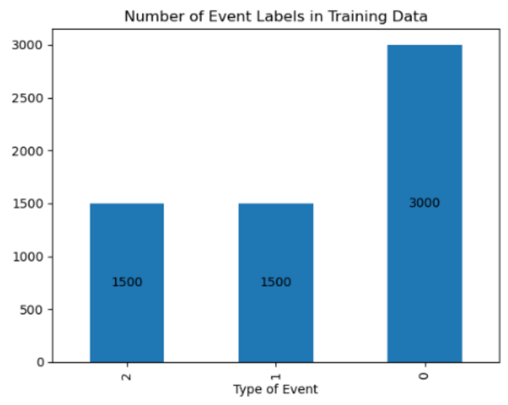
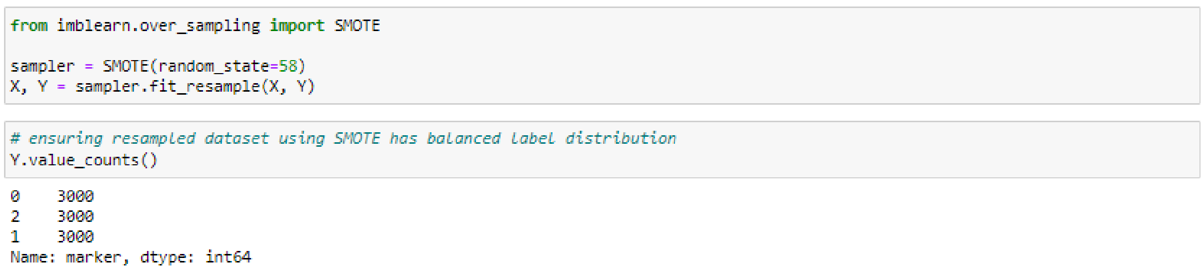


Figure 4: The event label distribution in the training data for Part B.

3.2.2 Data Preprocessing

As shown in Fig. 4, the event label distribution of the training data is skewed towards normal events. To balance the event label distribution, the imblearn library was used to perform oversampling using the Synthetic Minority Over-sampling Technique (SMOTE), as shown in Fig. 5. After balancing out the label distribution with SMOTE, the same holdout validation strategy used in Part A was deployed by splitting the provided training dataset into 80% training set and 20% test set. To ensure the test set has the same balanced event label distribution, the train-test split is done in a stratified fashion.



```python
from imblearn.over_sampling import SMOTE

sampler = SMOTE(random_state=58)
X, Y = sampler.fit_resample(X, Y)
```

```python
# ensuring resampled dataset using SMOTE has balanced label distribution
Y.value_counts()
```

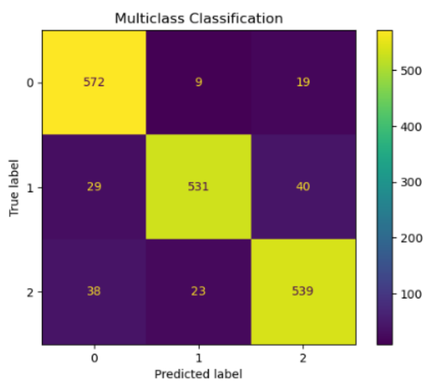
```
0    3000
2    3000
1    3000
Name: marker, dtype: int64
```

Figure 5: SMOTE was used to balance the event label distribution to reduce the effects of biased data during model training.

No data cleaning was performed since there were no missing values or invalid data found during the data exploration stage. The StandardScaler was used to perform feature scaling as in Part A. The holdout test set and the provided testing data was scaled with the mean and variance of the training set.

3.2.3 Model Selection and Hyperparameter Tuning

The best model found for Part B is the Random Forest Classifier. For hyperparameter tuning, a 5-fold GridSearchCV was done to find the best parameter for tree-depth and minimum samples before split to use for the model. The best cross-validation score of 0.913 was obtained with a maximum tree depth of 10 and a minimum number of samples before split of 3. The best tuned model was then evaluated on the holdout test set. An accuracy score of 0.91 and an average F1 score of 0.91 was obtained. Fig. 5 shows the confusion matrix and the classification report of the model.



```
                   precision    recall  f1-score   support

           normal       0.90      0.95      0.92       600
   data_injection       0.94      0.89      0.91       600
command_injection       0.90      0.90      0.90       600

         accuracy                           0.91      1800
        macro avg       0.91      0.91      0.91      1800
     weighted avg       0.91      0.91      0.91      1800

Final Accuracy:  0.9122222222222223
```

Figure 5: Confusion matrix and the classification report of the chosen final algorithm (Random Forest, depth=10, min_samples_split=3) on the 20% holdout test set.

3.3 Model Evaluation and Results

The final random forest model was used to predict the provided testing data in TestingDataMulti.csv. 33 system traces were classified as normal events, 35 were classified as data injection attacks, while 32 were classified as command injection events. Table 4 shows the computed labels for the testing data. According to Fig. 5, the model is most likely to misclassify data injection attacks, as data injection attack classification has the lowest recall score of 0.89 and that the model is more likely to misclassify both types of attacks as normal events, as it misclassified 67 instances of attacks as normal events. Nevertheless, since the precision and recall scores are approximately 90%, the model can be considered as neither overfitting nor underfitting. Thus, the model can be considered reliable as it has a accuracy of over 90%.

Table 4: Computed labels for testing data in Part B

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	2	1	1	1

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	1	2	2	2	2	2	0	0	0	0	0	0	1	1	1	1	1	2	1

41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	2	2	2	2	2	2	2	2	1	2	2	2	1	2	2	2	1	1	1

61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0

4 Conclusion

In both tasks, machine learning classifiers were designed and implemented to classify system traces in a power grid between normal events and cyber-attacks. In Part A, a binary classifier was built while in Part B, a multi class classifier was built to distinguish attacks into data injections or command injections. For both parts, classifiers with over 90% accuracy were achieved. The accuracy of these classifiers will allow power grid system CPS to correctly detect system events approximately 9 times out of 10. Therefore, with accurate detection of cyber-attacks, these CPSs can launch attack mitigation strategies earlier to prevent further compromission and damages to the power system infrastructure. In this application, a model with a higher false positive rate is more reliable than a model with a higher false negative rate. This is because misclassifying cyber-attacks as normal events has greater risk than misclassifying normal events as cyber-attacks. Moreover, to further improve these classifiers, misclassification strategies such as using deep learning models such as neural networks or using better feature selection approaches can be deployed. Through the work conducted with this coursework, the significance of modern technologies such as machine learning is understood in terms of their application in protecting and enhancing the cyber-physical system security of important infrastructures such as power grids.