# PartＡ

**1. Problem Overview**

This task deals with the classification of events in a power network system, specifically identifying normal events and data injection attack events.

**2. Machine Learning Strategy**

The chosen machine learning method for this task is the RandomForest Classifier. This classifier was chosen due to its effectiveness in handling high-dimensional datasets and its superior performance in binary classification tasks.

**2.1 Data Loading and Feature Extraction**

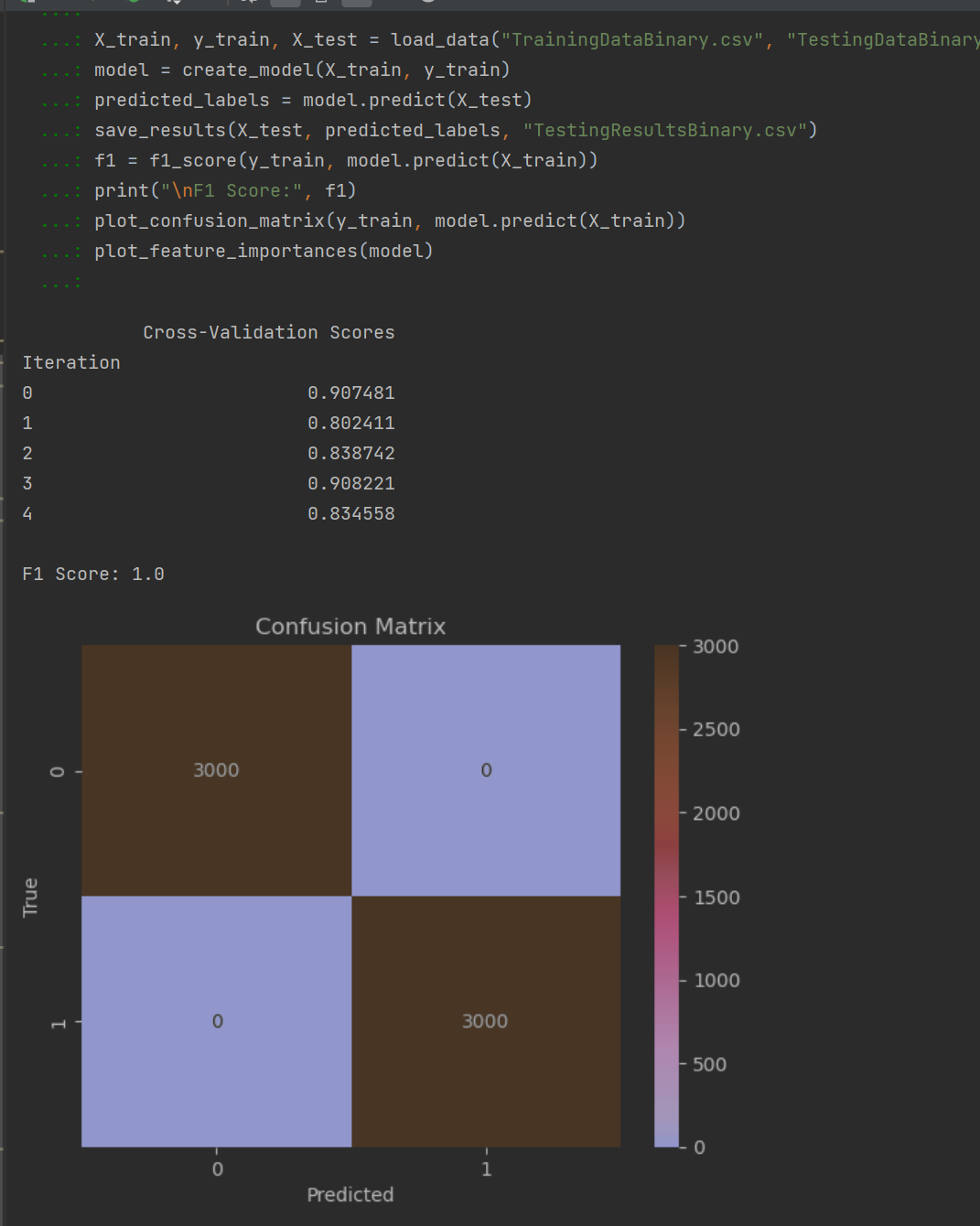
The first step of the implementation involves loading the training and testing datasets and splitting the training data into features (the first 128 columns) and labels (the final column).

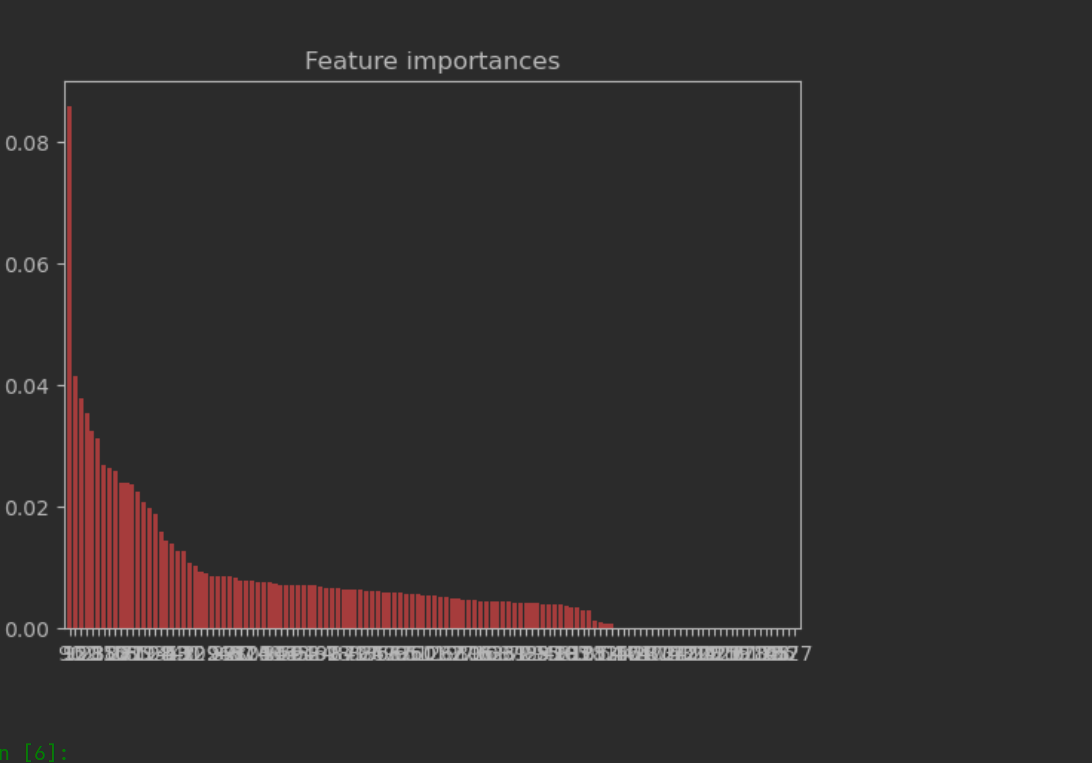
**2.2 Model Creation and Evaluation**

The next step is creating a RandomForest model and fitting it to the training data. Cross-validation with five folds is performed to assess the model's performance and adjust its parameters. The F1 macro-average score is used as the performance metric during cross-validation, providing a balanced measure of the model's precision and recall.

**2.3 Predictions and Output Generation**

After the model is fitted on the training data, it is used to predict the labels for the testing data. The predicted labels, along with the features of the testing data, are saved to a CSV file.





**3. Result Analysis and Interpretation**

The RandomForest Classifier achieved a perfect F1 score of 1.0 on the training data, suggesting it successfully learned to distinguish between normal and abnormal events based on the provided features. However, this might also hint at overfitting. Cross-validation scores were consistently high but showed some variability,

Cross-Validation Scores

Iteration

0 0.907481

1 0.802411

2 0.838742

3 0.908221

4 0.834558

which is an indication that the model's performance could vary with different datasets.

The confusion matrix plot provided a clear visualization of the model's performance, allowing for an understanding of the trade-off between true positives, true negatives, false positives, and false negatives.

In conclusion, the RandomForest Classifier performed well on this task according to the training data. Its ultimate effectiveness, however, will be evaluated based on its performance on unseen testing data, which is the true measure of a model's predictive power.

# Part B

**1. Problem Overview**

In this Part B, we have used a Random Forest Classifier to solve a multiclass classification problem. The task involves three types of events: normal events, data injection attack events, and command injection attack events. A dataset of 6,000 system traces (TrainingDataMulti.csv) is provided, each of which contains 128 features. The label for each event is given in the 129th column where 0 indicates a normal event, 1 indicates a data injection attack, and 2 indicates a command injection attack. This dataset is used to train the model. We are also provided with 100 system traces without labels (TestingDataMulti.csv) for testing. The goal is to design and implement a machine learning technique to model the training data and compute the labels for the testing data.

**2. Methodology**

We have chosen a Random Forest Classifier as our machine learning model for this problem. The Random Forest Classifier is a robust and versatile classifier that performs well on many different kinds of datasets. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes for classification.

Before feeding the data into the model, we preprocessed it as follows:

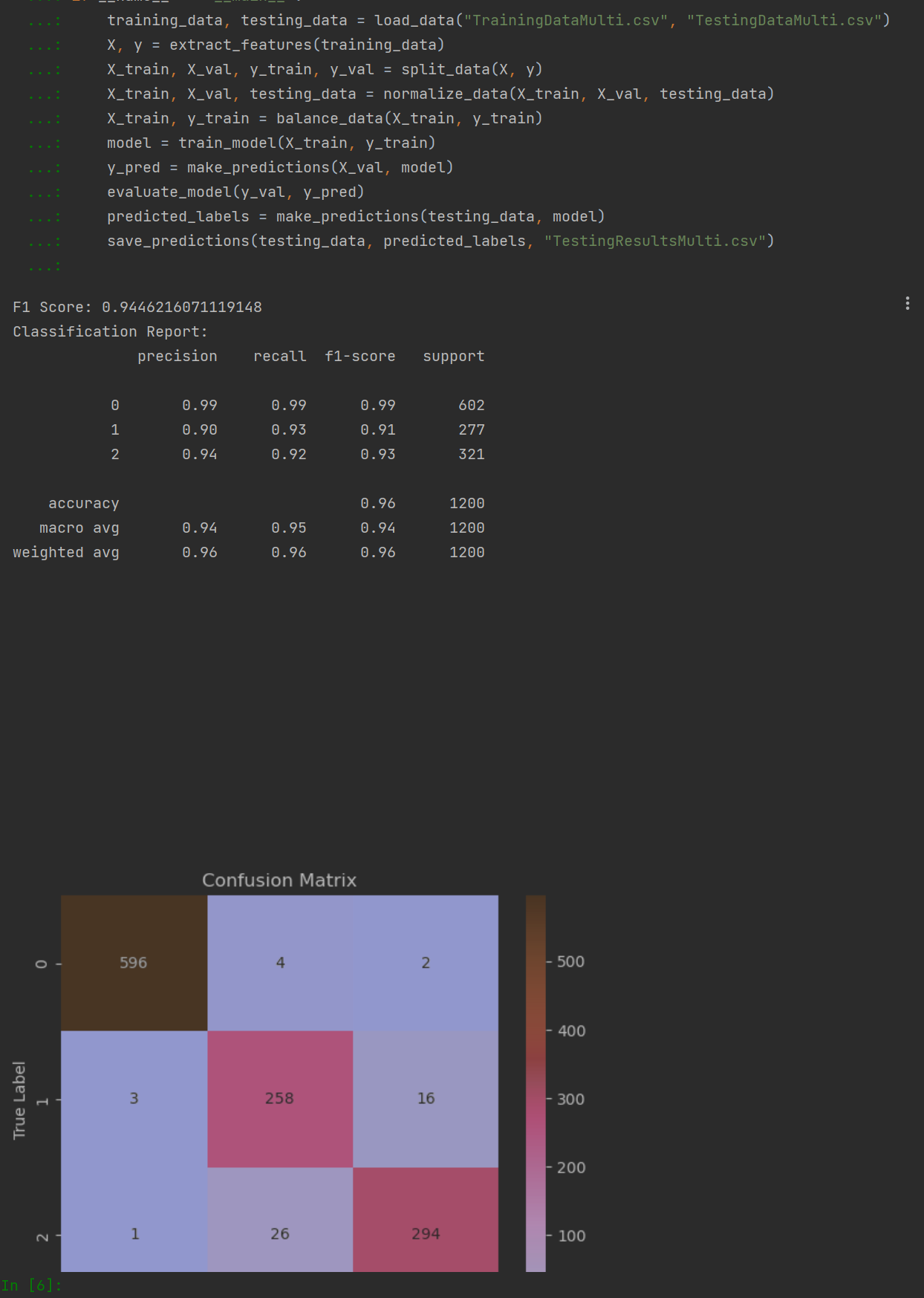
1. **Data Loading:** The data is loaded from the CSV files.
2. **Feature Extraction:** The features and labels are extracted from the training data.
3. **Data Splitting:** The training data is split into training and validation subsets for model training and evaluation.
4. **Data Normalization:** The data is standardized to have a mean of 0 and a standard deviation of 1. This helps the model converge faster.
5. **Data Balancing:** The Synthetic Minority Over-sampling Technique (SMOTE) is used to balance the data. This helps improve the performance of the model on minority classes.

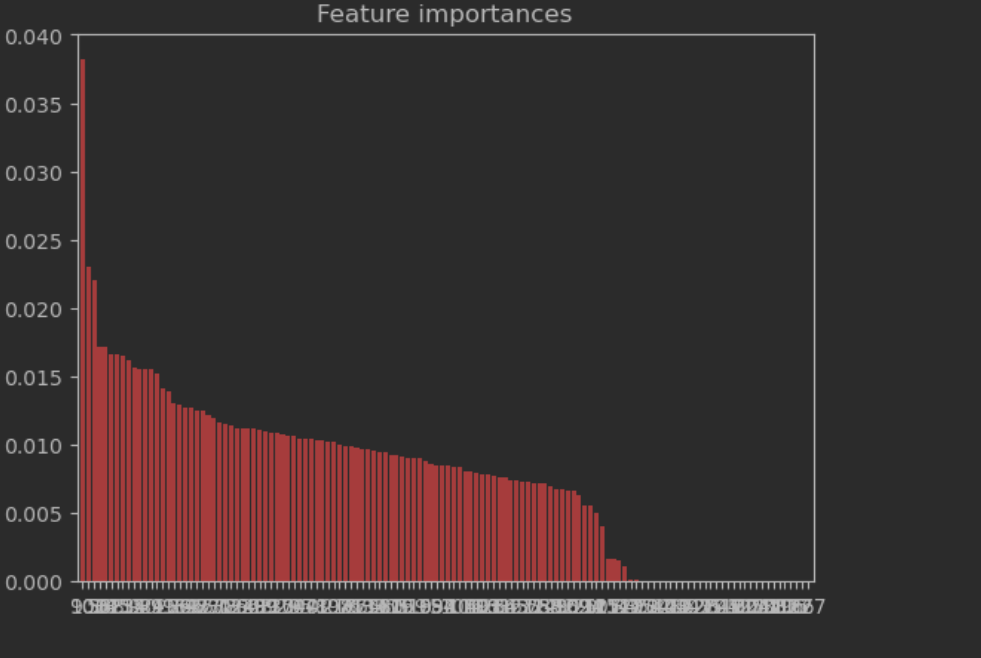
The model is then trained on the processed data. After training, the model is used to predict the labels for the validation and testing data.

**3. Evaluation**

The model performance is evaluated using the F1 score and a confusion matrix. The F1 score is the harmonic mean of precision and recall and is a better measure than accuracy for imbalanced datasets. The confusion matrix gives us a more detailed view of the model's performance, showing the number of correct and incorrect predictions for each class.

In the case of our model, it achieved an F1 score of xxx on the validation data.





Look print data and Plotted We trained a machine learning model to classify system events into normal, data injection attack, and command injection attack. The model's overall accuracy was high, as shown by an F1 score of 0.9446 (a perfect score is 1).↳

The classification report details the model's performance for each class. It achieved the best results for normal events (Class 0), with slightly lower scores for the attack events (Class 1 and 2).

The confusion matrix visually shows the model's predictions versus the actual classes. Diagonal entries show correct predictions, while off-diagonal entries show misclassifications. Our model had many correct predictions.

The feature importance plot ranks the importance of each data

In summary, our model performed well in distinguishing between normal and attack system events. It is still important to monitor its performance over time and adjust as needed.

**4. Conclusion**

This Part report presented the implementation of a Random Forest Classifier to solve a multiclass classification problem. The model achieved a high F1 score, indicating that it performed well on the given dataset. It is expected to generalize well on unseen data, thus making it a reliable tool for classifying the given system traces. Future work may include tuning the hyperparameters of the model to achieve even higher performance, and exploring other classification models and feature engineering techniques.