# Find Default(Credit faurd detection)

**Problem Statement:**

We are presented with a dataset comprising two days' worth of credit card transactions and tasked with developing a model to accurately detect fraudulent transactions.

**Machine Learning Model:**

To address this challenge, we undertook the following steps:

1. Data Importation: Utilizing the pandas library, we imported the dataset.

2. Data Inspection:

- Checked for missing values and assessed the dataset's shape.

- Fortunately, no missing values were found.

- Identified an imbalanced dataset.

- Balanced the dataset using undersampling techniques.

3. Data Splitting: Segmented the dataset into training and testing subsets using the `train\_test\_split` function.

4. Model Selection:

- Explored various supervised learning models.

- Determined that the Random Forest Classifier yielded optimal results.

- Fitted the model to the training data.

- Predicted outcomes for the testing data.

5. Hyperparameter Tuning:

- Employed grid search to fine-tune model hyperparameters.

6. Model Serialization:

- Created a joblib file containing the trained classifier model.

7. Feature Engineering:

- Engaged in feature engineering to enhance model performance and interpretability.

Purpose:

The outlined procedure serves several key purposes:

- Validates the accuracy and efficacy of the trained machine learning model.

- Identifies potential deficiencies or limitations in the model's predictive capabilities.

- Facilitates the process of debugging and refining the model to enhance its overall performance and reliability.

Implementation:

The testing purpose code can be implemented as a separate script or integrated into the pipeline. It involves loading the saved model weights or parameters and applying them to new data for prediction. Metrics such as accuracy, confusion matrix, or ROC curves may be computed to assess the model's performance.

**Pipeline:**

This Python script demonstrates the utilization of a machine learning pipeline for credit card fraud detection. Leveraging the scikit-learn library, the pipeline integrates preprocessing steps, such as feature scaling, with a trained Random Forest Classifier model to predict fraudulent transactions.

Purpose:

The purpose of this script is to automate the process of loading data, preprocessing features, and making predictions using a pre-trained machine learning model. By encapsulating these steps within a pipeline, it ensures consistency and reproducibility while simplifying the deployment of the fraud detection model.

Implementation:

-*Data Loading:* The script loads data from an Excel file ('predictions.xlsx') containing features and labels (fraudulent or non-fraudulent transactions).

- *Pipeline Definition*: A scikit-learn pipeline is created, consisting of two main steps: feature scaling using StandardScaler and model prediction using a pre-trained Random Forest Classifier loaded from a .pkl file.

- Model Evaluation: The pipeline is fitted to the data, and predictions are made on the same dataset to evaluate model performance. The accuracy of the model is calculated using scikit-learn's accuracy\_score function.

- *Output****:*** The script prints the accuracy of the model on the provided dataset.

Conclusion:

This script demonstrates a streamlined approach to deploy a pre-trained machine learning model for credit card fraud detection. By encapsulating preprocessing and prediction steps within a pipeline, it ensures efficient and consistent model deployment, facilitating real-time fraud detection in financial transactions

**Random Sample generator**

**Introduction:**

This Python script demonstrates the process of generating random sample input data and making predictions using a pre-trained machine learning model for credit card fraud detection. Leveraging the scikit-learn library and pandas for data manipulation, the script generates synthetic data, predicts the likelihood of fraud, and appends the results to an Excel file for further analysis.

**Purpose:**

The purpose of this script is to simulate real-world scenarios by generating random input data and evaluating the model's performance in detecting fraudulent transactions. By repeatedly generating new data and making predictions, it allows for robust testing and validation of the fraud detection model's effectiveness and scalability.

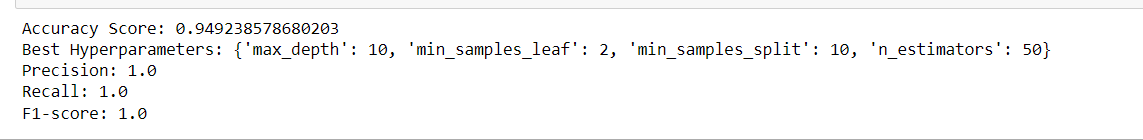
*Implementation:*

- *Model Loading and Prediction*: The script loads a pre-trained machine learning model for credit card fraud detection using joblib. It defines a function to make predictions based on input data and another function to print the prediction result.

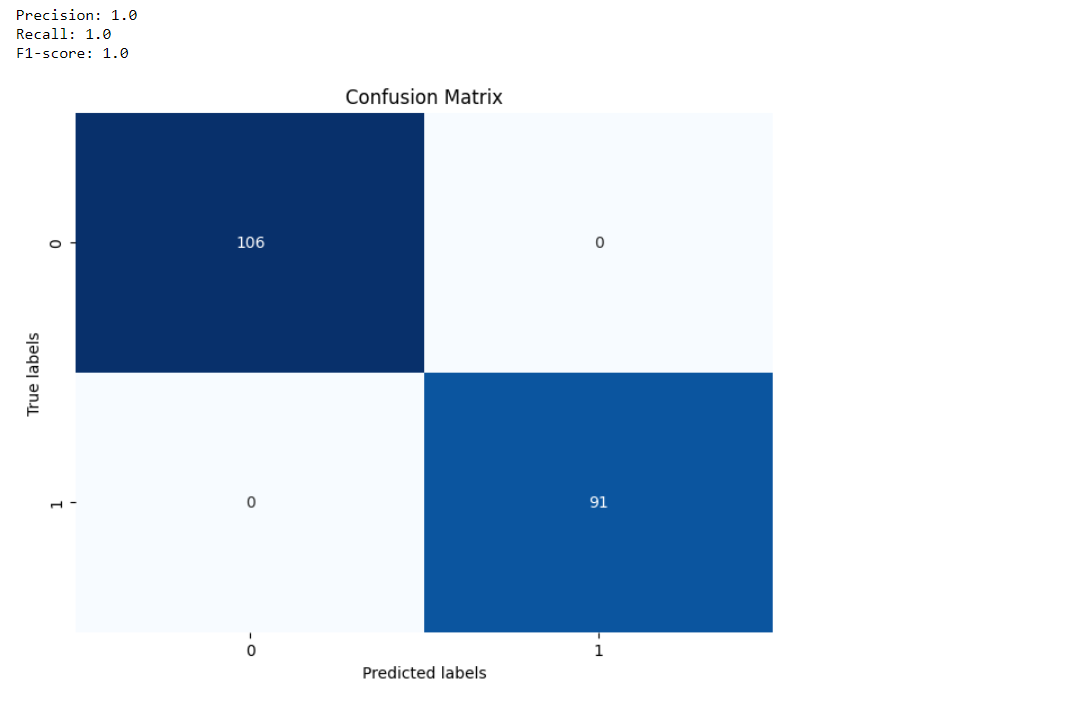
- *Data Generation and Prediction:* For each iteration (100 iterations in this case), the script generates random sample input data resembling credit card transaction details. It then utilizes the deployed model to predict whether the transaction is fraudulent or legitimate. Prediction results are printed, indicating the classification outcome.

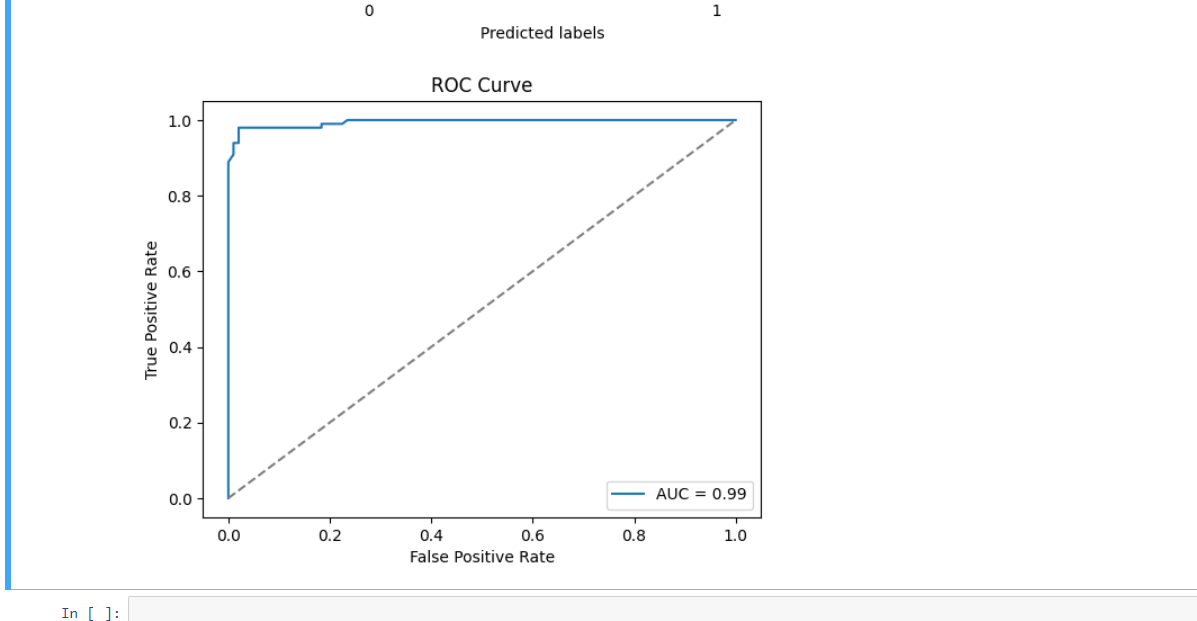
- Result Logging: After each prediction, the script appends the input data along with the prediction result (fraud or legitimate) to an Excel file named 'predictions.xlsx'. If the file exists, it loads the existing predictions and appends the new results. If the file does not exist or encounters an IO error, it creates a new file to store the data.

*Conclusion:*

This script provides a convenient and efficient way to evaluate the performance of a credit card fraud detection model using randomly generated sample data. By iteratively generating data and making predictions, it enables thorough testing and validation of the model's accuracy and robustness. Additionally, logging the prediction results to an Excel file allows for further analysis and comparison of model performance over time.

**Model Validation:**





**Future Work**

In the pursuit of improving our credit card fraud detection system, several potential areas for future exploration and enhancement have been identified:

1**. Feature Engineering**: Experimenting with different feature engineering techniques can offer insights into potentially more informative representations of the data, leading to improved model performance.

2. **Model Selection**: Evaluating alternative machine learning algorithms and ensemble methods beyond the RandomForestClassifier utilized in the current pipeline may uncover models better suited to capturing the complexities of the dataset.

3. **Ensemble Methods:** Exploring ensemble methods such as bagging, boosting, and stacking provides opportunities to combine multiple models effectively, potentially yielding higher predictive accuracy.

4**. Data Augmentation**: Investigating data augmentation techniques, such as synthetic sample generation with methods like SMOTE (Synthetic Minority Over-sampling Technique), may help address class imbalance and improve model generalization.

5**. Advanced Hyperparameter Tuning:** Delving deeper into hyperparameter tuning by leveraging advanced optimization techniques like Bayesian optimization or genetic algorithms can further refine model performance.

6. **Deployment**: Transitioning the developed model into a production environment involves integrating it seamlessly into existing systems, ensuring scalability, reliability, and real-time performance monitoring.

7. **Continuous Improvement:** Establishing a framework for continuous model improvement involves periodic retraining on new data, incorporating user feedback, and adapting to evolving fraud patterns and detection requirements.

By exploring these areas, we aim to enhance the effectiveness and robustness of our credit card fraud detection system, providing better protection for our customers and their financial assets.

**Source Code**

The source code used to create the pipeline can be found in the <pipeline.ipynb> file located in the current directory.