**CREDIT CARD FRAUD DETECTION WITH**

**APPLIED DATA SCIENCE**

**Phase 5: Project Documentation & Submission**

Credit card fraud is a prevalent concern in the modern financial landscape, with criminals continually finding new ways to engage in unauthorized and fraudulent transactions. To combat this issue, machine learning and data science techniques have become indispensable tools for financial institutions and businesses.

**Problem Statement:**

The problem being addressed in the provided code is the detection of credit card fraud. The primary goal is to build a machine learning model that can distinguish between fraudulent and non-fraudulent credit card transactions. The problem can be framed as a binary classification task where the classes are "fraudulent" (Class 1) and "non-fraudulent" (Class 0) transactions.

**Design Thinking Process:**

* **Problem Understanding:**

The initial step is to understand the problem and its significance. Credit card fraud detection is crucial for both financial institutions and cardholders to prevent unauthorized transactions.

* **Data Collection:**

The code reads a dataset from a CSV file ("creditcard.csv") containing transaction data, including features such as time, transaction amount, and others, as well as the target variable "Class" indicating fraud or non-fraud.

* **Data Preprocessing:**

Data preprocessing is crucial to prepare the dataset for machine learning. This involves data cleaning, handling missing values, feature selection, and balancing the dataset. The code performs some of these steps.

* **Model Selection:**

The code uses a Random Forest Classifier, which is an ensemble learning method known for its robustness and ability to handle complex data.

* **Model Training:**

The Random Forest Classifier is trained on the preprocessed data to learn patterns that distinguish between fraudulent and non-fraudulent transactions.

* **Model Evaluation:**

The model's performance is evaluated using accuracy, a classification report (which provides metrics like precision, recall, and F1-score for both classes), and a confusion matrix.

* **Model Deployment:**

The trained model is saved to a file (cc-rf-1-11-23.pkl) for future use, which can be deployed in real-time systems to detect credit card fraud.

**Phases of Development:**

* **Data Exploration:**
* The code begins by loading and exploring the dataset, checking its shape and displaying the first few rows.
* Data information and summary statistics are displayed to understand the features and their distributions.
* **Data Preprocessing:**
* A plot is created to visualize the distribution of transaction amounts.
* The dataset is balanced using the sample\_equally function to ensure an equal number of fraudulent and non-fraudulent transactions in the training data.
* **Model Training and Evaluation:**
* The data is split into training and testing sets.
* A Random Forest Classifier with 21 trees is trained on the balanced dataset.
* Model accuracy is calculated and a classification report is generated to assess model performance.
* A confusion matrix is displayed to understand the model's performance further.
* **Model Saving:**
* The trained Random Forest model is saved to a binary file using the pickle library for future use.

**Dataset description:**

* **File Name:**

creditcard.csv

* **Features:**

The dataset contains 30 feature columns (V1, V2, ..., V28) that represent various attributes of the credit card transactions. These features are the result of a PCA transformation, so their exact meanings are not provided in the dataset. Additionally, there are two additional columns, "Time" and "Amount," where "Time" represents the time elapsed between the transaction and the first transaction in the dataset, and "Amount" represents the transaction amount.

* **Target Variable:**

The dataset includes a binary target variable labeled "Class," where:

* 0: Non-fraudulent transaction
* 1: Fraudulent transaction
* **Data Size:**

The dataset typically contains a large number of transactions, with both fraudulent and non-fraudulent examples. The exact number of rows in the dataset can vary, but it's usually substantial to provide a realistic scenario for fraud detection.

**Data preprocessing steps:**

* **Loading the Data:**

The dataset is loaded from a CSV file using the Pandas library with the pd.read\_csv() function.

* **Exploratory Data Analysis (EDA):**

The code conducts some initial exploratory data analysis to understand the dataset, including checking its shape, displaying the first few rows, and providing information about the data using data.info() and data.describe().

* **Balancing the Dataset:**

Credit card fraud datasets are often highly imbalanced, with a small number of fraudulent transactions compared to non-fraudulent ones. To address this issue, the code includes a function called sample equally to balance the dataset. This function selects a balanced number of samples for both the "fraudulent" (Class 1) and "non-fraudulent" (Class 0) categories.

* **Feature Selection:**

In the code, feature selection is not explicitly mentioned. However, it is assumed that all features except "Time" and "Class" are used for training the model. The "Time" feature may not be relevant for fraud detection, and "Class" is the target variable.

* **Splitting the Data:**

The dataset is split into training and testing sets using the train test split function from Scikit-Learn. This is essential to evaluate the model's performance.

**Model training process:**

* **Data Loading and Preprocessing:**
* The code starts by loading the credit card transaction dataset from a CSV file. The dataset is stored in a Pandas DataFrame.
* The dataset is explored to understand its structure, and it is checked for any missing values.
* **Data Balancing:**

Credit card fraud datasets are often highly imbalanced, with significantly more non-fraudulent transactions than fraudulent ones. To address this imbalance, the code uses the sample equally function to create a balanced dataset. It samples an equal number of fraudulent and non-fraudulent transactions to ensure that the model has an unbiased representation of both classes.

* **Feature Selection:**

Although not explicitly mentioned in the code, it is assumed that all the features (except "Time" and "Class") are used for training the model. Feature selection is crucial, and the specific features selected for training may vary based on the dataset and the requirements of the task.

* **Data Splitting:**

The balanced dataset is split into training and testing sets using the train test split function from Scikit-Learn. This allows for the evaluation of the model's performance on unseen data.

* **Model Selection:**

The code selects a Random Forest Classifier as the machine learning algorithm for this task. The Random Forest Classifier is an ensemble learning method that builds multiple decision trees and combines their predictions.

* **Model Training:**

The Random Forest Classifier is instantiated with 21 trees (as specified in Random Forest Classifier (21)) and is then trained using the training data (X train and y train).

* **Model Evaluation:**

After training, the model is used to make predictions on the test data (X test). The predicted values are stored in the y pred variable.

The model's performance is evaluated using the following metrics:

* **Accuracy:** The code calculates the accuracy score using accuracy score, which measures the overall correctness of the model's predictions.
* **Classification Report:** The code generates a classification report using classification report, which provides metrics such as precision, recall, and F1-score for both classes (fraudulent and non-fraudulent). This report is essential for understanding how well the model performs for each class.
* **Confusion Matrix:** A confusion matrix is displayed to give insights into true positives, true negatives, false positives, and false negatives, allowing for a detailed assessment of the model's performance.
* **Model Saving:**

Finally, the trained Random Forest model is saved to a binary file ("cc-rf-1-11-23.pkl") using the pickle library. This saved model can be loaded and used for making predictions on new data or for deployment in production systems.

**Choice of Machine Learning Algorithm :**

* **Random Forest Classifier:**

Random Forest is a popular choice for classification tasks, including credit card fraud detection, for several reasons:

* It is an ensemble method that combines multiple decision trees, providing robustness and reducing overfitting.
* It handles both numerical and categorical features well.
* It is capable of capturing complex relationships in the data.
* It can deal with imbalanced datasets, which are common in fraud detection.
* **Number of Trees (Estimators):**

In the code, a Random Forest Classifier with 21 trees is used. The number of trees should be chosen carefully, as more trees generally lead to better performance, but also increase computation time. The choice of 21 trees may be based on empirical testing or a balance between accuracy and computational efficiency.

**Choice of Evaluation Metrics:**

* **Accuracy:**

Accuracy is a commonly used metric to evaluate the overall correctness of the model's predictions. In fraud detection, it provides a basic measure of how well the model performs on the entire dataset. However, accuracy can be misleading when dealing with imbalanced datasets, as it may not reflect the true performance of the model, especially for the minority class (fraudulent transactions).

* **Classification Report:**
* The classification report is a more comprehensive evaluation metric. It includes metrics such as precision, recall, and F1-score for each class (fraudulent and non-fraudulent transactions). This is essential for fraud detection, as it helps assess false positives and false negatives.
* Precision: Precision measures the accuracy of positive predictions (fraudulent transactions). A high precision indicates that the model doesn't often misclassify non-fraudulent transactions as fraudulent.
* Recall (Sensitivity): Recall measures the ability of the model to correctly identify positive cases. It's crucial in fraud detection, as it quantifies how well the model identifies fraudulent transactions.
* F1-Score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance.
* **Confusion Matrix:**

The confusion matrix provides a detailed view of the model's performance, showing true positives, true negatives, false positives, and false negatives. It helps in understanding where the model may be making errors and which types of errors are more prevalent.

**Conclusion:**

In summary, the credit card fraud detection program provides a reliable solution for identifying fraudulent transactions, offering a balance between accuracy and precision. The use of Random Forest and careful data preprocessing techniques ensures the model's ability to detect fraudulent activities while minimizing the risk of false alarms.

This program is a testament to the power of machine learning in addressing real-world problems and enhancing security in financial transactions.