**Credit Card Fraud Detection**

**Overview**

This project is a credit card fraud detection system that uses machine learning, specifically a Random Forest Classifier, to distinguish between fraudulent and non-fraudulent credit card transactions. It includes data preprocessing, model training, and evaluation steps. The code is designed to help you build and deploy a fraud detection model.

**Table of Contents**

1. [Prerequisites](#prerequisites)

2. [Installation](#installation)

3. [Usage](#usage)

4. [Data](#data)

5. [Data Preprocessing](#data-preprocessing)

6. [Model Training](#model-training)

7. [Evaluation](#evaluation)

8. [Model Saving](#model-saving)

9. [Contributing](#contributing)

10. [License](#license)

**Prerequisites**

- Python (3.6+)

- Required Python libraries (specified in the code)

- Jupyter Notebook (optional, for running the code interactively)

**Installation**

1. Clone the repository:
2. Navigate to the project directory:2. Navigate to the project directory:
3. Install the required Python libraries:

**Usage**

- Run the Jupyter Notebook or Python script to execute the code. You can customize the model hyperparameters or change the data file path as needed.

**Data**

- The dataset used for this project is named "creditcard.csv." It contains features related to credit card transactions and a binary target variable indicating whether a transaction is fraudulent or non-fraudulent.

- The dataset is not provided with this repository. You can obtain it from [source] and place it in the project directory.

**Data Preprocessing**

- Data preprocessing steps include loading the dataset, exploring it, balancing it to address class imbalance, and splitting it into training and testing sets.

- Additional preprocessing steps (e.g., handling missing values or feature engineering) can be added as needed.

**Model Training**

- The Random Forest Classifier is used for model training. The code specifies the number of trees (estimators) in the model.

- You can experiment with different hyperparameters or choose another machine learning algorithm as needed.

**Evaluation**

- The model's performance is evaluated using several metrics, including accuracy, a classification report, and a confusion matrix. These metrics help assess the model's accuracy and its ability to detect fraudulent transactions.

- Custom evaluation metrics can be added based on specific project requirements.

**Model Saving**

- The trained model can be saved to a binary file for future use or deployment. The code utilizes the `pickle` library for model serialization.

**Contributing**

- Contributions are welcome. If you'd like to contribute to this project, please open an issue or create a pull request.