Coding Guidelines and improvements for “Random Forest for Credit Card Fraud Detection” research paper code implementation

1. Author did not provide the code for the research paper “Random Forest for Credit Card Fraud Detection”. So, I have taken the reference from online to complete this task
2. Here are the references for the author proposed code:
   https://www.kaggle.com/code/hassanamin/credit-card-fraud-detection-using-random-forest/notebook

   https://github.com/Nirjoy/CSE445-Credit-Card-Fraud-Detection-using-Random-Forest-SMOTE

   https://github.com/julian0316/Credit-Card-Fraud-Detection-with-Random-Forest

   https://github.com/Prajwal10031999/Credit-Card-Fraud-Detection-using-Random-Forest

3. I have implemented the code using the dataset available in the Kaggle.com . The Dataset available in the Kaggle website is perfectly fits to our project and did not require any changes in the data.
   Here the reference link to download the dataset:
   https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

4. I have implemented and compiled the reference code. Here the steps to proceed with compiling:
   a. open and run the “ReferenceProjectCode_CreditCardFraudDetection.ipynb” file with Jupyter
   b. Please open the file ReferenceProjectCode_CreditCardFraudDetection.ipynb
   c. Please run these comments, before starting the compilation to avoid errors.

   pip install numpy
   pip install pandas
   pip install matplotlib
   pip install seaborn
   pip install sklearn

   d. Download the dataset file (creditcard.csv) from the Kaggle website and copy the location of the file. Replace the path in the code.

   Example : C:/Users/sivak/OneDrive/Desktop/ML/Code/ExampleCode/creditcard.csv

e. Run each cell one by one in order to achieve the code

5. In my implementation, I have improved the code by adding more methods. In this project I have compared various classification models to check which model gives the best result.

6. Models were built on the imbalanced data and hyperparameters were tuned. Then SMOTE and ADASYN techniques were used to balance the data. Models was tried on both SMOTE and ADASYN data to see which one is producing better result.

   Used the following classification models.

   - Logistic Regression
   - KNN
   - Decision Tree
   - Random forest

7. For model performance parameters, we have used the ROC curve and find the AUC score as performance matrix. ROC curves measure the performance of the model at the different thresholds ich will helps us to find the optimal threshold for the model.
8. In Logistic Regression, we have tuned the few parameters and cross validation few methods used.
9. In KNN model we have used the KNeighborsClassifier and we have observed that, with imbalanced data the model gives us AUC about 0.94 and the recall is 0.77.
10. For Decision tree model we have used the “tree.DecionTreeClassifier” classifier. Using this model we got the AUC is 0.88, which is not satisfactory.
11. In Random Forest model, we have used the RandomForestClassifiers and GridSearchCV. We observed that we got Very good precision of about 0.97
12. In Logistic Regression, we have used the LogisticRegressionClassifier.
13. In Overall we observed that “Random Forest” has more AUC than KNN. KNN with SMOTE oversampling giving us the final results.