**y.saketh**

**21BBS0123**

**CSE(BUSINESS SYSTEMS) (VIT)**

**Online Payment Fraud Detection using Machine Learning in Python**

**As we are approaching modernity, the trend of paying online is increasing tremendously. It is very beneficial for the buyer to pay online as it saves time, and solves the problem of free money. Also, we do not need to carry cash with us. But we all know that Good thing are accompanied by bad things.**

**The online payment method leads to fraud that can happen using any payment app. That is why Online Payment Fraud Detection is very important.**

**Online Payment Fraud Detection using Machine Learning in Python**

**Here we will try to solve this issue with the help of machine learning in**[**Python**](https://www.geeksforgeeks.org/python-programming-language/)**.**

**The dataset we will be using have these columns -**

| **Feature** | **Description** |
| --- | --- |
| **step** | **tells about the unit of time** |
| **type** | **type of transaction done** |
| **amount** | **the total amount of transaction** |
| **nameOrg** | **account that starts the transaction** |
| **oldbalanceOrg** | **Balance of the account of sender before transaction** |
| **newbalanceOrg** | **Balance of the account of sender after transaction** |
| **nameDest** | **account that receives the transaction** |
| **oldbalanceDest** | **Balance of the account of receiver before transaction** |
| **newbalanceDest** | **Balance of the account of receiver after transaction** |
| **isFraud** | **The value to be predicted i.e. 0 or 1** |

**Importing Libraries and Datasets**

**The libraries used are :**

* [**Pandas**](https://www.geeksforgeeks.org/python-pandas-dataframe/)**:  This library helps to load the data frame in a 2D array format and has multiple functions to perform analysis tasks in one go.**
* [**Seaborn**](https://www.geeksforgeeks.org/introduction-to-seaborn-python/)**/**[**Matplotlib**](https://www.geeksforgeeks.org/python-introduction-matplotlib/)**: For data visualization.**
* [**Numpy**](https://www.geeksforgeeks.org/python-numpy/)**: Numpy arrays are very fast and can perform large computations in a very short time.**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**%matplotlib inline**

**The dataset includes the features like type of payment, Old balance , amount paid, name of the destination, etc. You can download dataset from**[**here**](https://drive.google.com/file/d/127JqP3WGjBVihR-ZcUR86T3wwy3_g63v/view?usp=sharing)**.**

**data = pd.read\_csv('new\_data.csv')**

**data.head()**

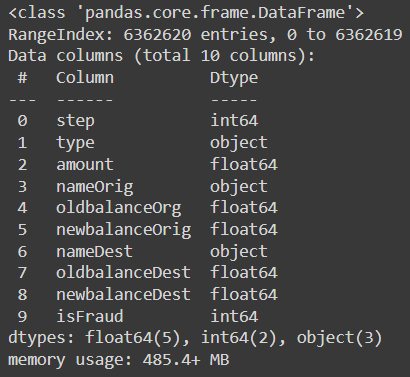
**Output :**

**head**

**To print the information of the data we can use data.info() command.**

**data.info()**

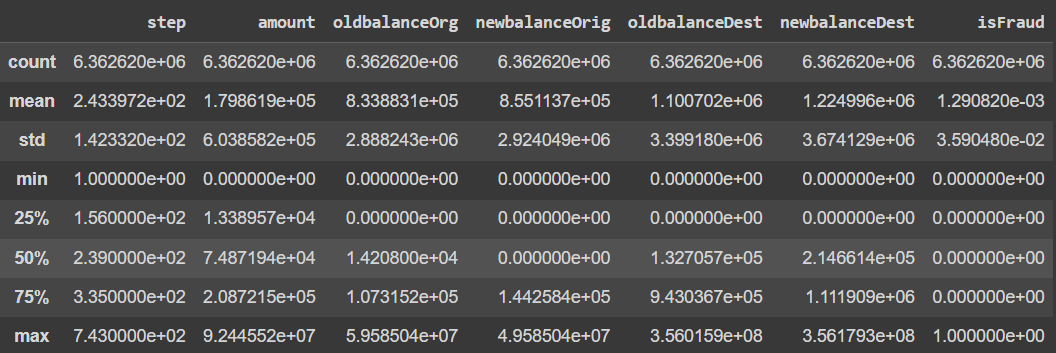
**Output :**

**info**

**Let's see the mean, count , minimum and maximum values of the data.**

**data.describe()**

**Output :**

**describe**

**Data Visualization**

**In this section, we will try to understand and compare all columns.**

**Let's count the columns with different datatypes like Category, Integer, Float.**

**obj = (data.dtypes == 'object')**

**object\_cols = list(obj[obj].index)**

**print("Categorical variables:", len(object\_cols))**

**int\_ = (data.dtypes == 'int')**

**num\_cols = list(int\_[int\_].index)**

**print("Integer variables:", len(num\_cols))**

**fl = (data.dtypes == 'float')**

**fl\_cols = list(fl[fl].index)**

**print("Float variables:", len(fl\_cols))**

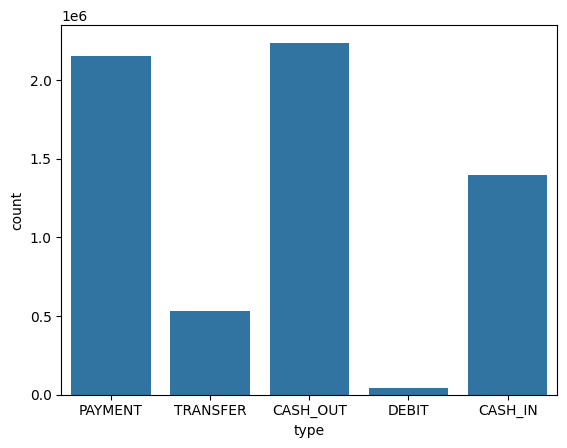
**Output :**

**Categorical variables: 3  
Integer variables: 2  
Float variables: 5**

**Let's see the count plot of the Payment type column using Seaborn library.**

**sns.countplot(x='type', data=data)**

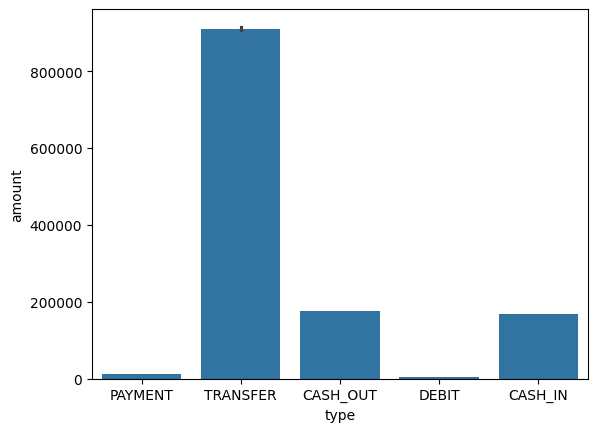
**Output :**

**countplot**

**We can also use the bar plot for analyzing Type and amount column simultaneously.**

**sns.barplot(x='type', y='amount', data=data)**

**Output :**

**barplot**

**Both the graph clearly shows that mostly the type cash\_out and transfer are maximum in count and as well as in amount.**

**Let's check the distribution of data among both the prediction values.**

**data['isFraud'].value\_counts()**

**Output :**

**isFraud count  
0 6354407  
1 8213**

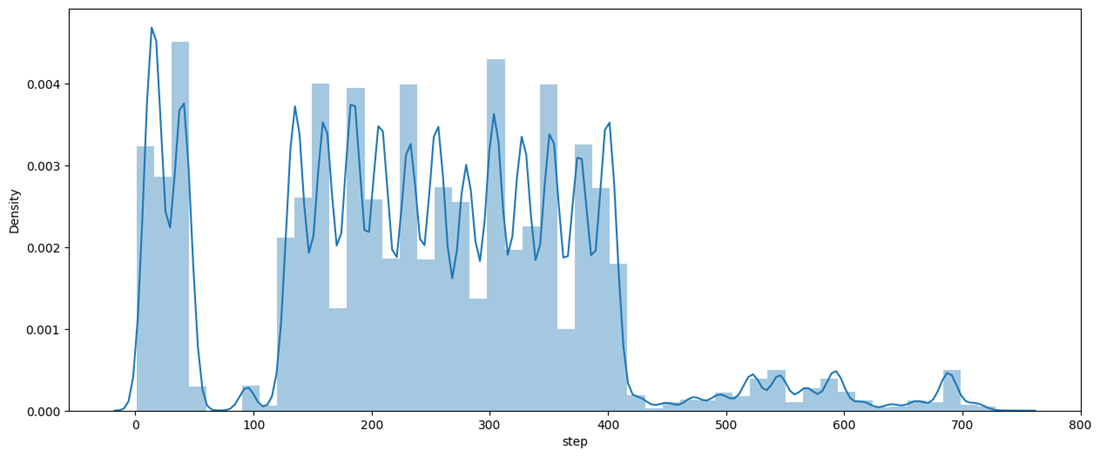
**The dataset is already in same count. So there is no need of sampling.**

**Now let's see the distribution of the step column using distplot.**

**plt.figure(figsize=(15, 6))**

**sns.distplot(data['step'], bins=50)**

**Output :**

**distplot**

**The graph shows the maximum distribution among 200 to 400 of step.**

**Now, Let's find the correlation among different features using**[**Heatmap**](https://www.geeksforgeeks.org/how-to-draw-2d-heatmap-using-matplotlib-in-python/)**.**

**plt.figure(figsize=(12, 6))**

**sns.heatmap(data.apply(lambda x: pd.factorize(x)[0]).corr(),**

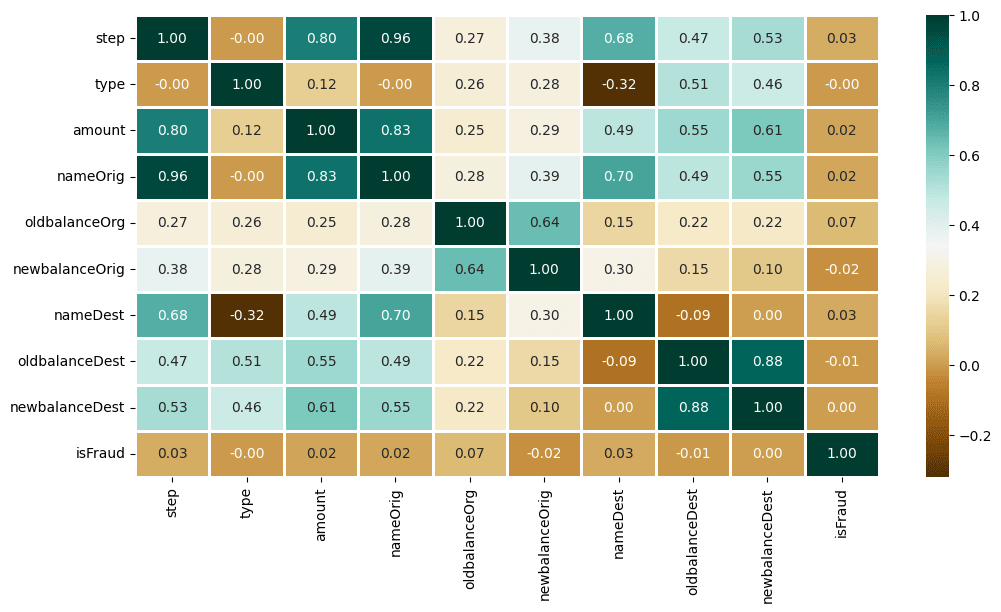
**cmap='BrBG',**

**fmt='.2f',**

**linewidths=2,**

**annot=True)**

**Output :**

**Heatmap**

**Data Preprocessing**

**This step includes the following :**

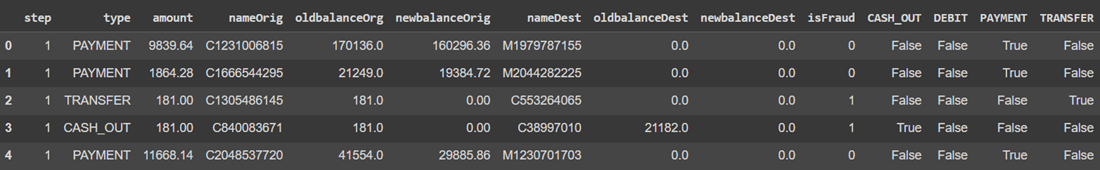
* **Encoding of Type column**
* **Dropping irrelevant columns like nameOrig, nameDest**
* **Data Splitting**

**type\_new = pd.get\_dummies(data['type'], drop\_first=True)**

**data\_new = pd.concat([data, type\_new], axis=1)**

**data\_new.head()**

**Output:**

**Encoding of Type column**

**Once we done with the encoding, now we can drop the irrelevant columns. For that, follow the code given below.**

**X = data\_new.drop(['isFraud', 'type', 'nameOrig', 'nameDest'], axis=1)**

**y = data\_new['isFraud']**

**Let's check the shape of extracted data.**

**X.shape, y.shape**

**Output:**

**((6362620, 10), (6362620,))**

**Now let's split the data into 2 parts : Training and Testing.**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**X, y, test\_size=0.3, random\_state=42)**

**Model Training**

**As the prediction is a classification problem so the models we will be using are :**

* [**LogisticRegression**](https://www.geeksforgeeks.org/understanding-logistic-regression/)**:  It predicts that the probability of a given data belongs to the particular category or not.**
* [**XGBClassifier**](https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/)**: It refers to Gradient Boosted decision trees. In this algorithm, decision trees are created in sequential form and weights are assigned to all the independent variables which are then fed into the decision tree which predicts results.**
* [**SVC**](https://www.geeksforgeeks.org/support-vector-machine-algorithm/)**: SVC is used to find a hyperplane in an N-dimensional space that distinctly classifies the data points. Then it gives the output according the most nearby element.**
* [**RandomForestClassifier**](https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/)**: Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. Then, it collects the votes from different decision trees to decide the final prediction.**

**Let's import the modules of the relevant models.**

**from xgboost import XGBClassifier**

**from sklearn.metrics import roc\_auc\_score as ras**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.svm import SVC**

**from sklearn.ensemble import RandomForestClassifier**

**Once done with the importing, Let's train the model.**

**models = [LogisticRegression(), XGBClassifier(),**

**RandomForestClassifier(n\_estimators=7,**

**criterion='entropy',**

**random\_state=7)]**

**for i in range(len(models)):**

**models[i].fit(X\_train, y\_train)**

**print(f'{models[i]} : ')**

**train\_preds = models[i].predict\_proba(X\_train)[:, 1]**

**print('Training Accuracy : ', ras(y\_train, train\_preds))**

**y\_preds = models[i].predict\_proba(X\_test)[:, 1]**

**print('Validation Accuracy : ', ras(y\_test, y\_preds))**

**print()**

**Output:**

**LogisticRegression() :   
Training Accuracy : 0.8873984626066378  
Validation Accuracy : 0.8849956507155117  
  
XGBClassifier(base\_score=None, booster=None, callbacks=None,  
 colsample\_bylevel=None, colsample\_bynode=None,  
 colsample\_bytree=None, device=None, early\_stopping\_rounds=None,  
 enable\_categorical=False, eval\_metric=None, feature\_types=None,  
 gamma=None, grow\_policy=None, importance\_type=None,  
 interaction\_constraints=None, learning\_rate=None, max\_bin=None,  
 max\_cat\_threshold=None, max\_cat\_to\_onehot=None,  
 max\_delta\_step=None, max\_depth=None, max\_leaves=None,  
 min\_child\_weight=None, missing=nan, monotone\_constraints=None,  
 multi\_strategy=None, n\_estimators=None, n\_jobs=None,  
 num\_parallel\_tree=None, random\_state=None, ...) :   
Training Accuracy : 0.9999774189140321  
Validation Accuracy : 0.999212631773824  
  
RandomForestClassifier(criterion='entropy', n\_estimators=7, random\_state=7) :   
Training Accuracy : 0.9999992716004644  
Validation Accuracy : 0.9650098729693373**

**Model Evaluation**

**The best-performed model is XGBClassifier. Let's plot the**[**Confusion Matrix**](https://www.geeksforgeeks.org/confusion-matrix-machine-learning/)**for the same.**

**from sklearn.metrics import ConfusionMatrixDisplay**

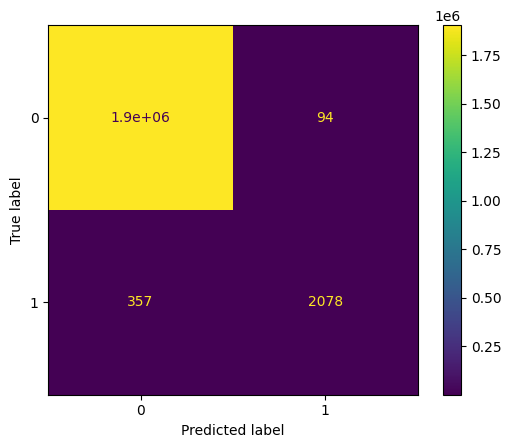
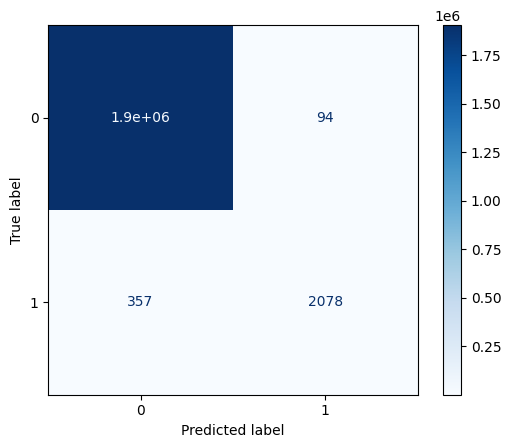
**import matplotlib.pyplot as plt**

**cm = ConfusionMatrixDisplay.from\_estimator(models[1], X\_test, y\_test)**

**cm.plot(cmap='Blues')**

**plt.show()**

**Output:**

**confusion matrixconfusion matrix (blue level)**

**You can download the dataset and source code from here:**

* **Dataset:**[**Online Payment Data**](https://drive.google.com/file/d/133E0TDrfIjnhwRoGTw9OEozwBXUL38D8/view?usp=sharing)
* **Source Code:**[**Online Payment Fraud Detection**](https://media.geeksforgeeks.org/wp-content/uploads/20250117031346547488/online_payment_fraud_detection_using_machine_learning_in_python.py)