PROJECT 2

FIND DEFAULT : PREDICTION OF CREDIT CARD FRAUD

Github link : https://github.com/sdrsonu/Prediction_of_credit_card_fraud

Introduction :

Credit card fraud stands as a significant apprehension for both consumers and financial institutions, fraught with the peril of financial losses and besmirching the reputation of financial entities. Machine learning methodologies have been profoundly employed to discern fraudulent transactions. Within this endeavor, logistic regression serves as the bedrock for categorizing transactions as either bona fide or deceitful contingent upon their intrinsic attributes.

Data :

The dataset employed herein comprises a CSV file housing credit card transactional data, manifesting 31 columns and a voluminous 284,807 rows. The pivotal "Class" column acts as the target variable, delineating the transaction's legitimacy (Class = 0) or fraudulent nature (Class= 1).

Preprocessing:

Prior to model training initiation, a pivotal initial step encompasses the segregation of legitimate and fraudulent transactions. Given the dataset's inherent imbalance, characterized by a preponderance of legitimate transactions vis-à-vis fraudulent ones, a strategic recourse entails the undersampling of the former to ameliorate class disparities. Subsequently, data bifurcation into training and testing subsets transpires leveraging the train_test_split() utility.

Model:

The adoption of logistic regression assumes paramount significance in discerning transactions as either authentic or deceptive predicated upon their distinguishing features. Logistic regression, renowned as a quintessential classification algorithm, delineates the probability of event occurrence contingent upon input features. The logistic regression model is honed via the LogisticRegression() function within the scikit-learn framework, subsequently prognosticating the target variable for the testing data.

Evaluation:

Model efficacy assessment pivots upon the accuracy metric, denoting the proportion of correctly classified transactions. Utilization of the accuracy_score() function from scikit-learn facilitates the computation of accuracy pertaining to both training and testing data.

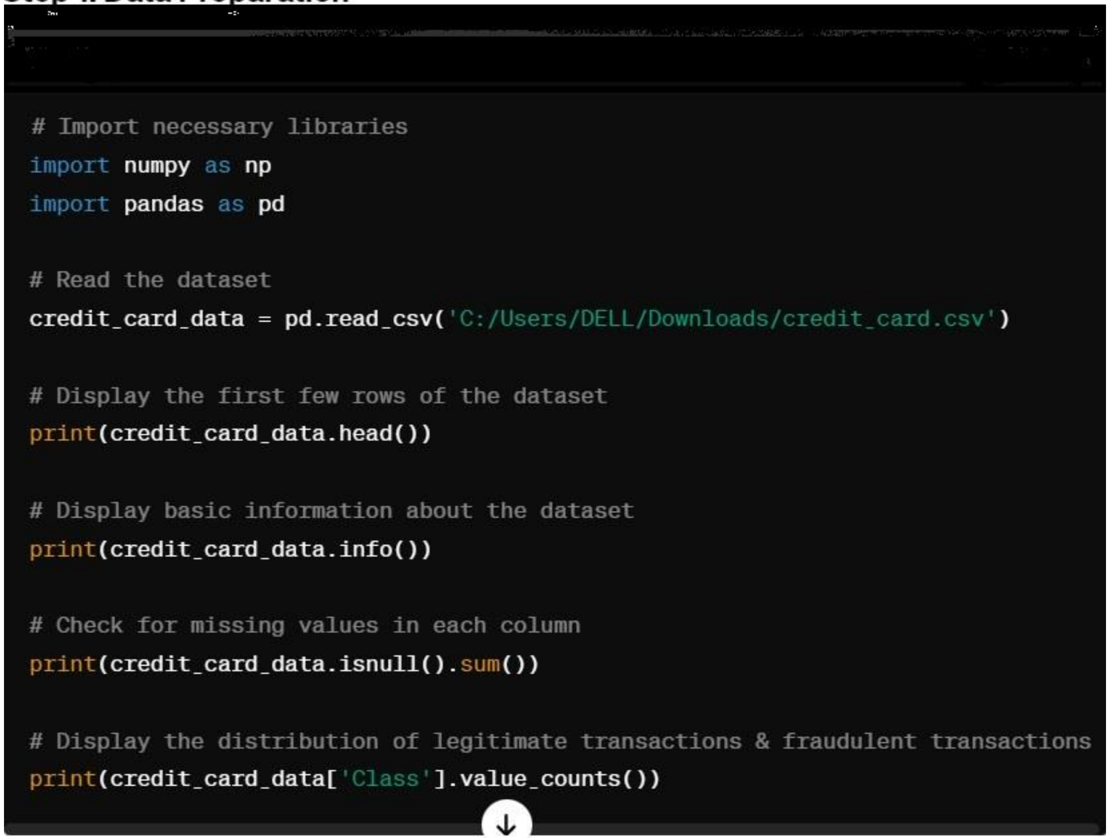
Streamlit Application:

Leveraging Streamlit heralds the creation of a user interface tailored for the credit card fraud detection project. The Streamlit application affords users the capability to upload a CSV file housing credit card transaction data, subsequently utilized for logistic regression model training. Additionally, users are empowered to input transactional attributes, thereby eliciting predictions concerning the transaction's legitimacy or fraudulent nature.

Conclusion:

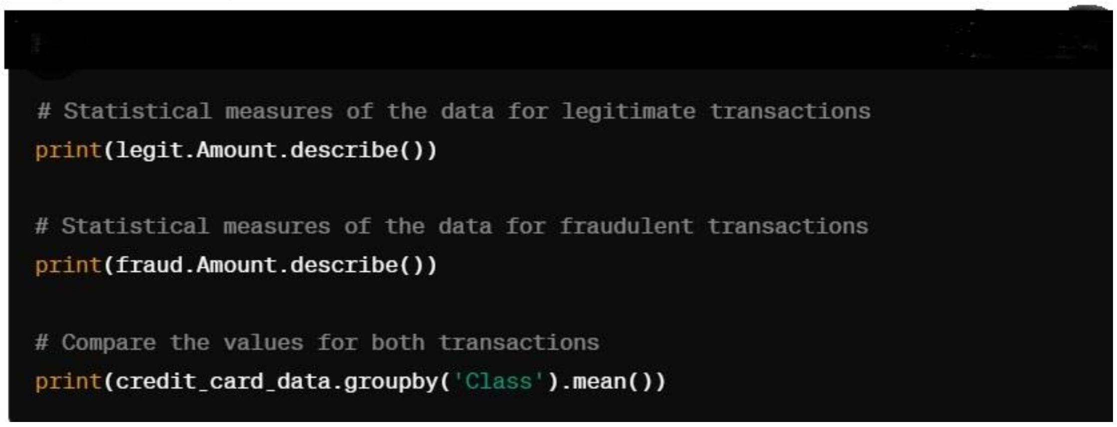
This endeavor underscores logistic regression's efficacy in ferreting out fraudulent credit card transactions. Attainment of commendable accuracy levels across both training and testing datasets underscores the model's efficacy in fraud detection. The Streamlit application furnishes a facile interface facilitating real-time detection of fraudulent transactions.

Step 1: Data Preparation



```python
# Import necessary libraries
import numpy as np
import pandas as pd

# Read the dataset
credit_card_data = pd.read_csv('C:/Users/DELL/Downloads/credit_card.csv')

# Display the first few rows of the dataset
print(credit_card_data.head())

# Display basic information about the dataset
print(credit_card_data.info())

# Check for missing values in each column
print(credit_card_data.isnull().sum())

# Display the distribution of legitimate transactions & fraudulent transactions
print(credit_card_data['Class'].value_counts())
```

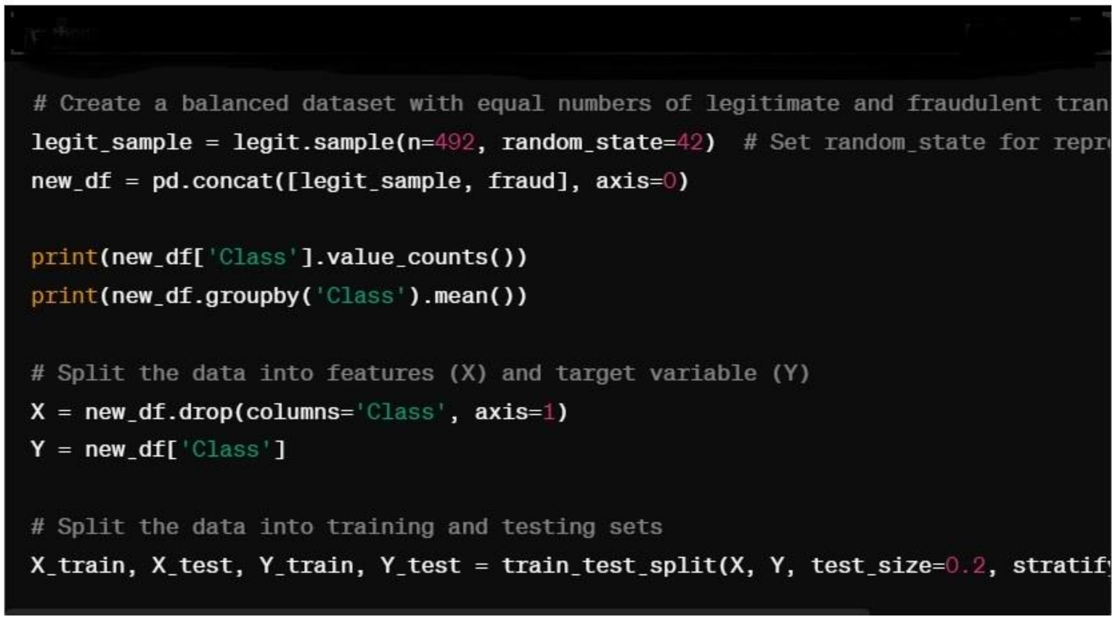
Step 2: Data Exploration



[redacted]

```
# Statistical measures of the data for legitimate transactions
print(legit.Amount.describe())

# Statistical measures of the data for fraudulent transactions
print(fraud.Amount.describe())

# Compare the values for both transactions
print(credit_card_data.groupby('Class').mean())
```

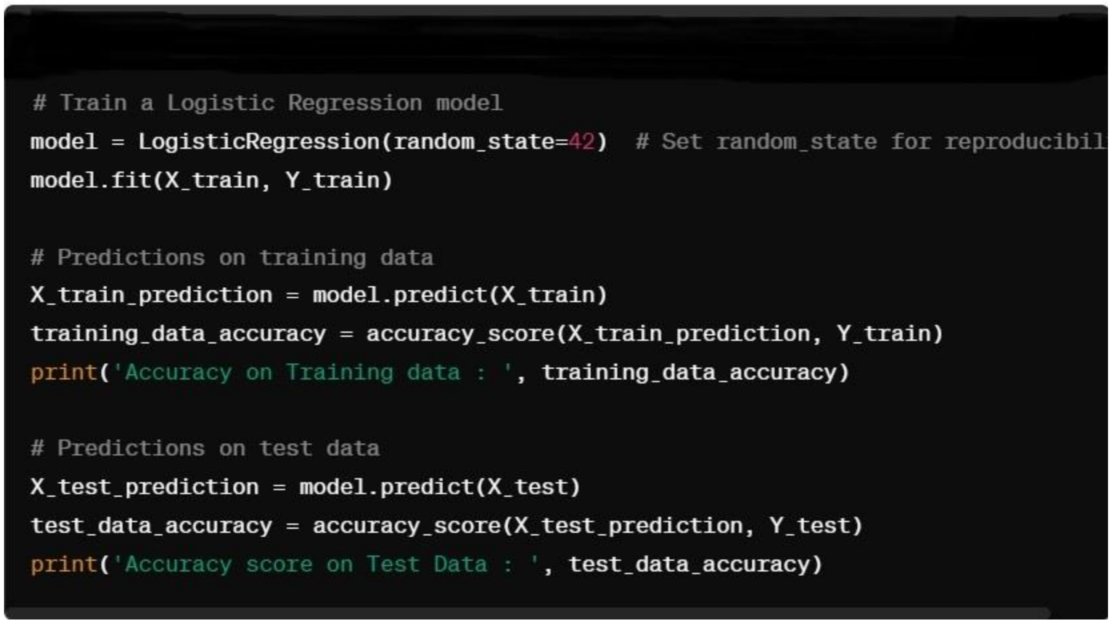
Step 3: Data Balancing



[redacted]

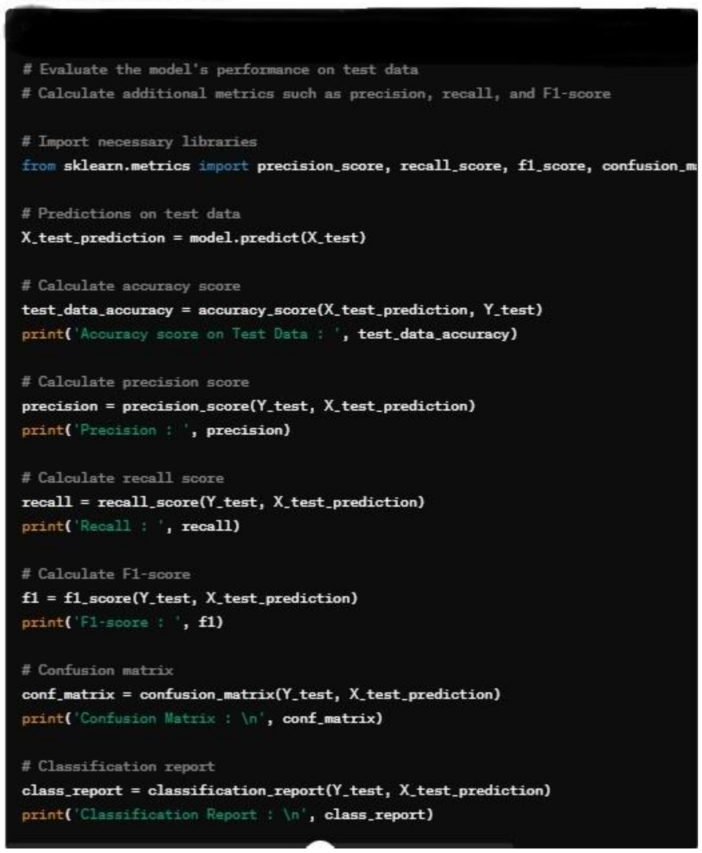
```
# Create a balanced dataset with equal numbers of legitimate and fraudulent tran
legit_sample = legit.sample(n=492, random_state=42)  # Set random_state for repr
new_df = pd.concat([legit_sample, fraud], axis=0)

print(new_df['Class'].value_counts())
print(new_df.groupby('Class').mean())

# Split the data into features (X) and target variable (Y)
X = new_df.drop(columns='Class', axis=1)
Y = new_df['Class']

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratif
```

Step 4: Model Training



```
# Train a Logistic Regression model
model = LogisticRegression(random_state=42)  # Set random_state for reproducibil
model.fit(X_train, Y_train)

# Predictions on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy on Training data : ', training_data_accuracy)

# Predictions on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score on Test Data : ', test_data_accuracy)
```

Step 5: Model Evaluation



```
# Evaluate the model's performance on test data
# Calculate additional metrics such as precision, recall, and F1-score

# Import necessary libraries
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_m

# Predictions on test data
X_test_prediction = model.predict(X_test)

# Calculate accuracy score
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score on Test Data : ', test_data_accuracy)

# Calculate precision score
precision = precision_score(Y_test, X_test_prediction)
print('Precision : ', precision)

# Calculate recall score
recall = recall_score(Y_test, X_test_prediction)
print('Recall : ', recall)

# Calculate F1-score
f1 = f1_score(Y_test, X_test_prediction)
print('F1-score : ', f1)

# Confusion matrix
conf_matrix = confusion_matrix(Y_test, X_test_prediction)
print('Confusion Matrix : \n', conf_matrix)

# Classification report
class_report = classification_report(Y_test, X_test_prediction)
print('Classification Report : \n', class_report)
```