Phase 6:

Project documentation and demonstration

Team id:593034

Project Documentation

Online Payment Fraud Detection using Machine Learning

Table of Contents

1. Project Overview

1.1 Project Description

This project aims to develop a machine learning model for the detection of online payment fraud. Online payment fraud has become a significant concern for both financial institutions and consumers. Detecting fraudulent transactions is crucial to minimize financial losses and maintain trust in online payment systems.

1.2 Project Team

a. Ashwini Yadav

b. Soumya Singh

c. Harshit Goyal

d. Kritika Sonare

1.3 Project Timeline

Start Date: [18/10/2023]

End Date: [09/11/2023]

2. Objectives

2.1 Problem Statement

Develop a machine learning model to detect online payment fraud accurately and efficiently, reducing false positives and false negatives.

2.2 Goals

Build a fraud detection system using machine learning.

Achieve a high level of accuracy in fraud detection.

Minimize false positives to avoid inconveniencing genuine users.

Minimize false negatives to catch as many fraudulent transactions as possible.

2.3 Scope

The project will focus on using historical transaction data for training and evaluation. Real-time transaction monitoring and integration with payment systems will be considered as future improvements.

3. Methodology

3.1 Data Collection

Data will be collected from various sources, including transaction logs, financial institutions, and payment service providers. The data will include transaction details, user information, and transaction labels (fraud or not fraud).

3.2 Data Preprocessing

Data preprocessing involves handling missing values, encoding categorical variables, and scaling numerical features. It also includes splitting the dataset into training and testing sets.

3.3 Exploratory Data Analysis (EDA)

EDA will be performed to understand the data, visualize the distribution of transaction types, and explore correlations between features and fraud.

3.4 Feature Engineering

Feature engineering includes mapping transaction types to numerical values and creating relevant features to improve model performance.

3.5 Model Selection

Various machine learning models, including decision trees, random forests, and gradient boosting, will be evaluated for fraud detection.

3.6 Model Training

The selected model will be trained using the training dataset.

3.7 Model Evaluation

The model's performance will be evaluated using metrics such as accuracy, precision, recall, and F1-score. A confusion matrix will also be generated.

3.8 Model Deployment

The trained model will be deployed for real-time or batch processing of online payment transactions.

4. Code Implementation

4.1 Data Loading

The project begins with loading the transaction data from a CSV file.

4.2 Data Preprocessing

Data preprocessing steps, such as handling missing values and data encoding, are applied to the dataset.

4.3 Exploratory Data Analysis

The code explores the data's distribution and correlation, creating visualizations to understand the dataset.

4.4 Feature Engineering

Transaction types are mapped to numerical values for machine learning. Additional features may be created.

4.5 Model Training

A machine learning model (e.g., Decision Tree) is trained on the preprocessed data.

4.6 Model Evaluation

The model is evaluated using testing data, and performance metrics are calculated.

4.7 Model Deployment

The trained model can be deployed for real-time or batch processing of payment transactions.

5. Results

5.1 Performance Metrics

Performance metrics, such as accuracy, precision, recall, and F1-score, are calculated.

5.2 Model Accuracy

The accuracy of the trained model is reported.

5.3 Confusion Matrix

A confusion matrix is generated to visualize true positives, true negatives, false positives, and false negatives.

5.4 Prediction Example

An example of using the trained model for making predictions is provided.

6. Conclusion

6.1 Summary

A summary of the project's objectives and results.

6.2 Key Findings

Key findings from the project, including model performance.

6.3 Lessons Learned

Lessons learned during the project and areas for improvement.

7. Future Improvements

7.1 Model Enhancement

Ways to improve the fraud detection model, such as using more advanced algorithms and incorporating additional data sources.

7.2 Real-time Monitoring

Considerations for implementing real-time monitoring of payment transactions.

7.3 Integration with Payment Systems

Exploration of how the model can be integrated with existing online payment systems.

8. References

.

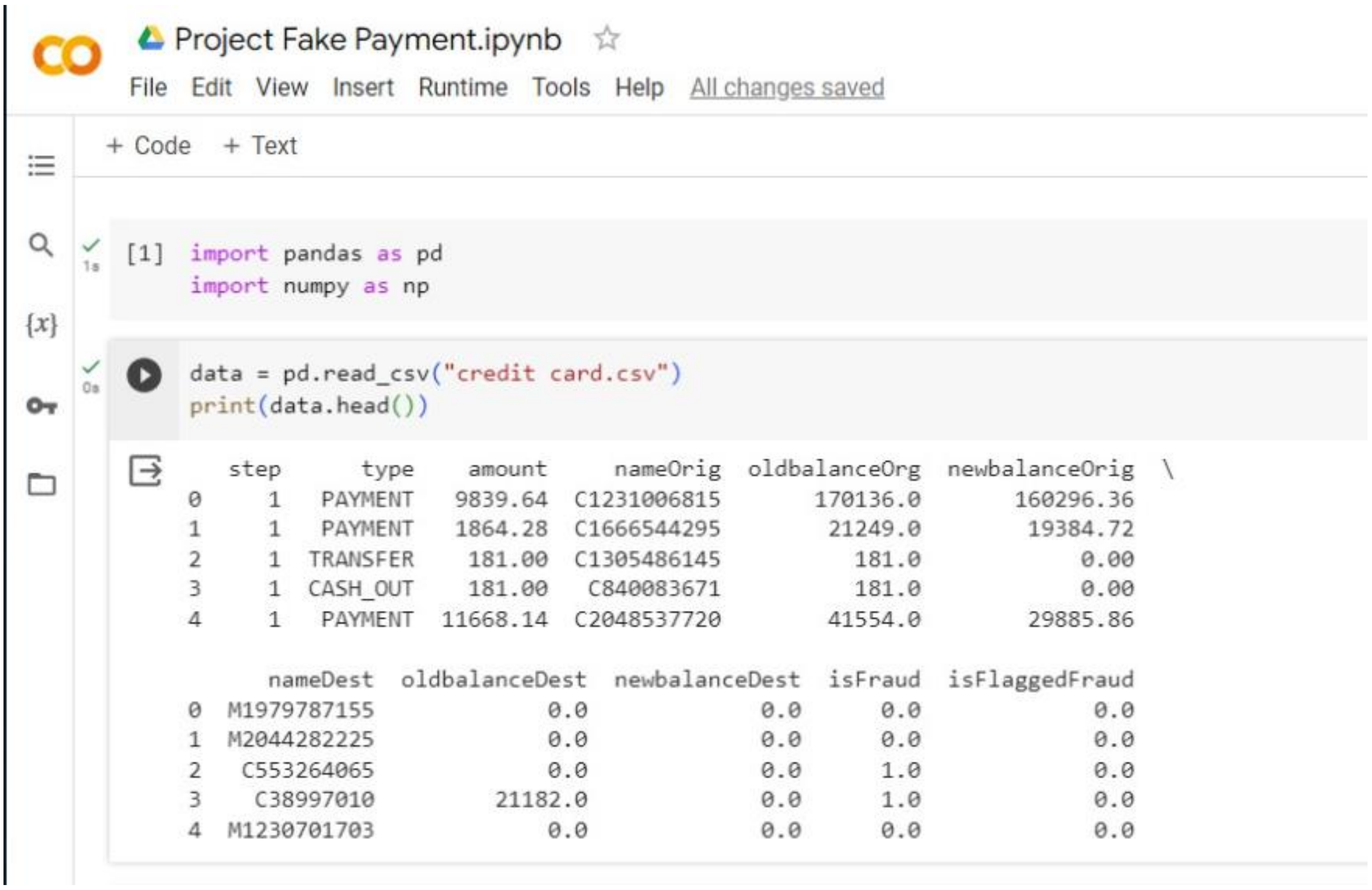
https://stripe.com/en-in/resources/more/how-machine-learning-works-for-payment-fraud-detection-and-prevention

https://www.ravelin.com/insights/machine-learning-for-fraud-detection

https://www.google.com/amp/s/www.tookitaki.com/compliance-hub/how-does-ai-detect-fraud%3fhs_amp=true

https://www.thebanker.com/AI-and-the-new-age-of-fraud-detection-1506931228

Project Demonstation:

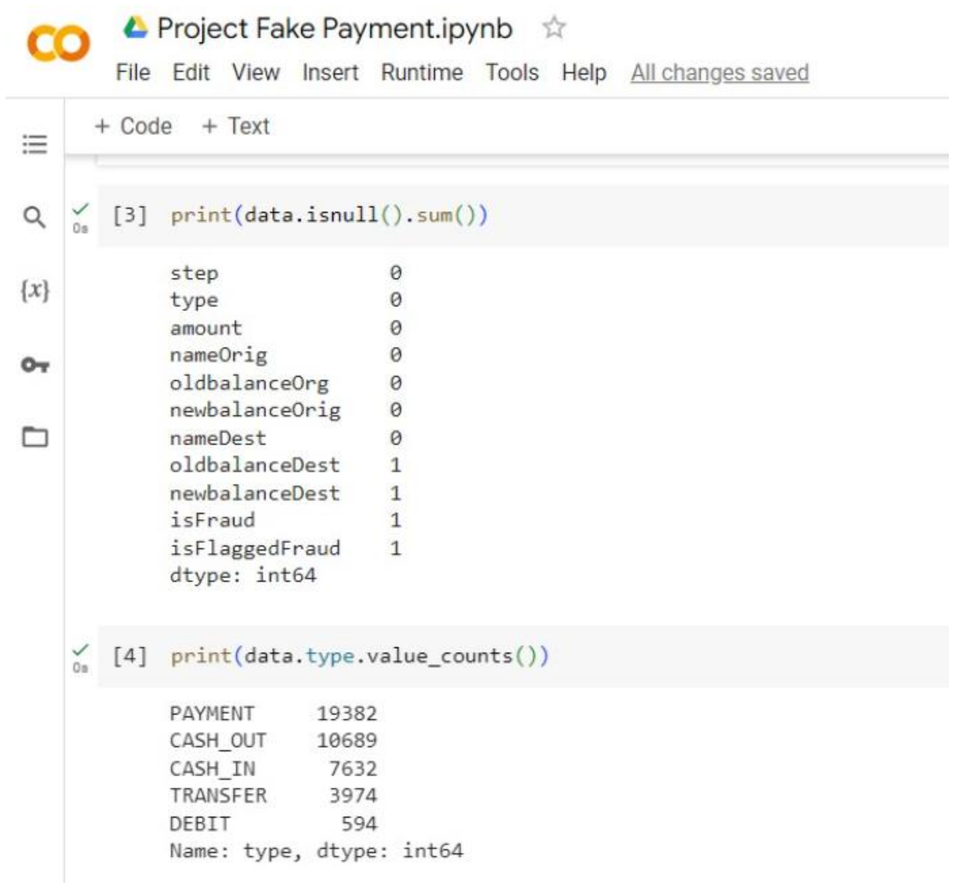

Project Fake Payment.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
[1] import pandas as pd
    import numpy as np
```

```
data = pd.read_csv("credit card.csv")
print(data.head())
```

```
   step      type    amount     nameOrig  oldbalanceOrg  newbalanceOrig  \
0     1   PAYMENT   9839.64  C1231006815       170136.0       160296.36
1     1   PAYMENT   1864.28  C1666544295        21249.0        19384.72
2     1  TRANSFER    181.00  C1305486145          181.0            0.00
3     1  CASH_OUT    181.00   C840083671          181.0            0.00
4     1   PAYMENT  11668.14  C2048537720        41554.0        29885.86

      nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
0  M1979787155             0.0             0.0      0.0             0.0
1  M2044282225             0.0             0.0      0.0             0.0
2   C553264065             0.0             0.0      1.0             0.0
3    C38997010         21182.0             0.0      1.0             0.0
4  M1230701703             0.0             0.0      0.0             0.0
```

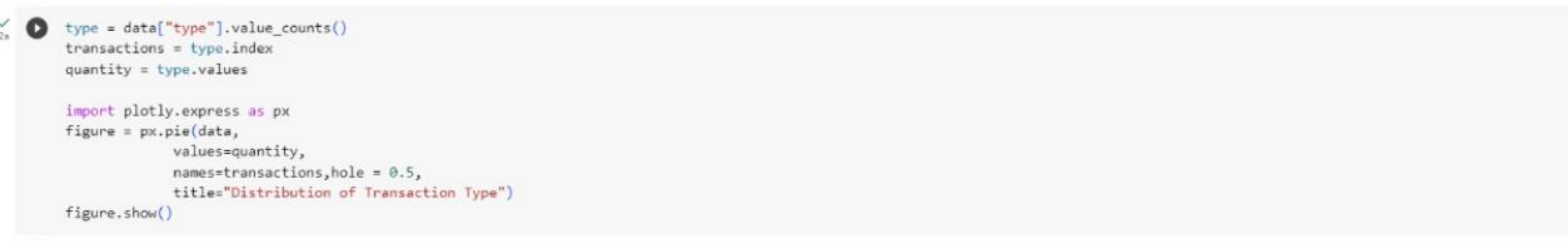

Project Fake Payment.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
[3] print(data.isnull().sum())
```

```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    1
newbalanceDest    1
isFraud           1
isFlaggedFraud    1
dtype: int64
```

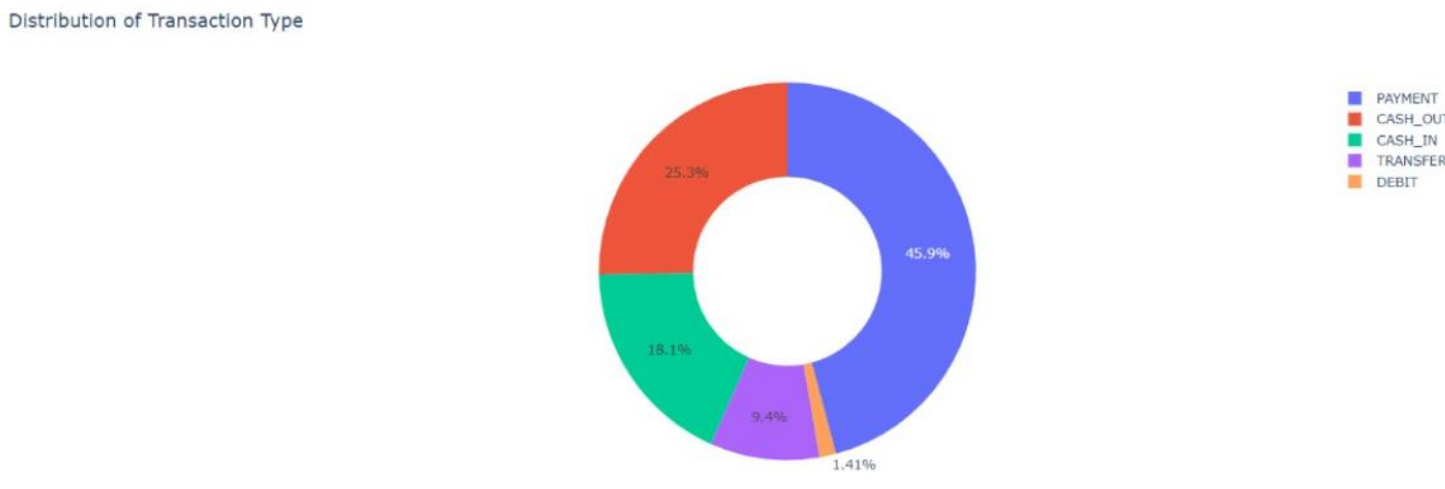
```
[4] print(data.type.value_counts())
```

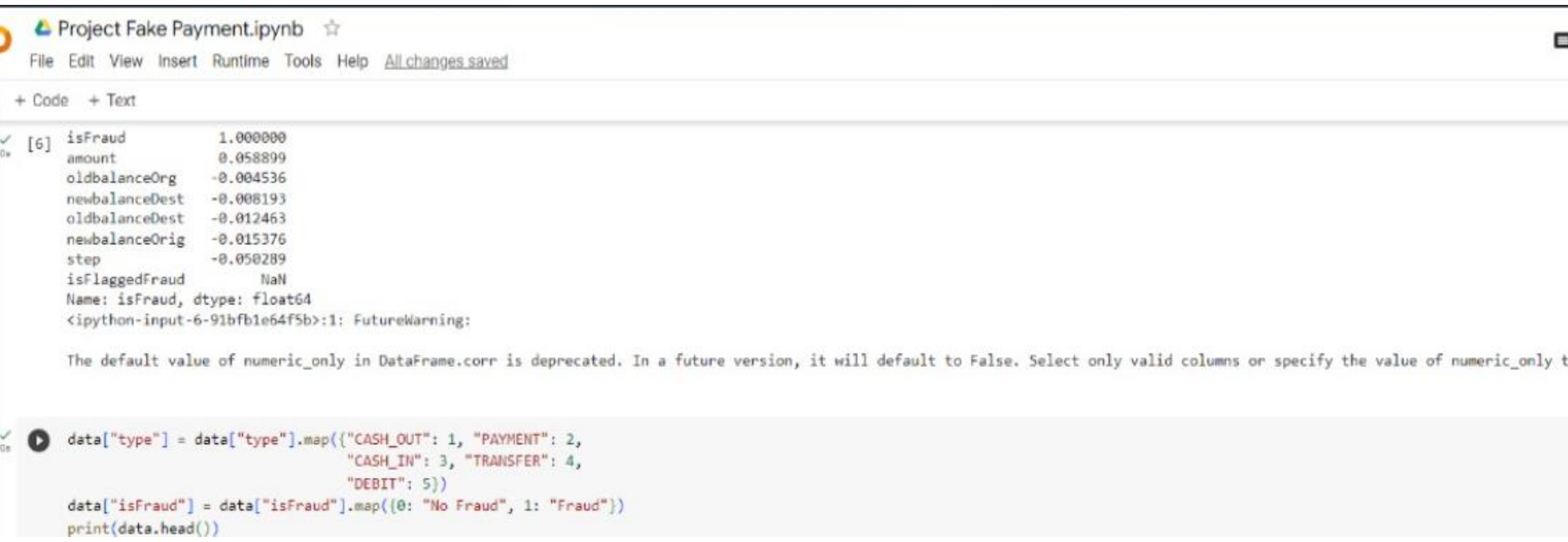
```
PAYMENT     19382
CASH_OUT    10689
CASH_IN      7632
TRANSFER     3974
DEBIT         594
Name: type, dtype: int64
```



```
type = data["type"].value_counts()
transactions = type.index
quantity = type.values

import plotly.express as px
figure = px.pie(data,
             values=quantity,
             names=transactions,hole = 0.5,
             title="Distribution of Transaction Type")
figure.show()
```

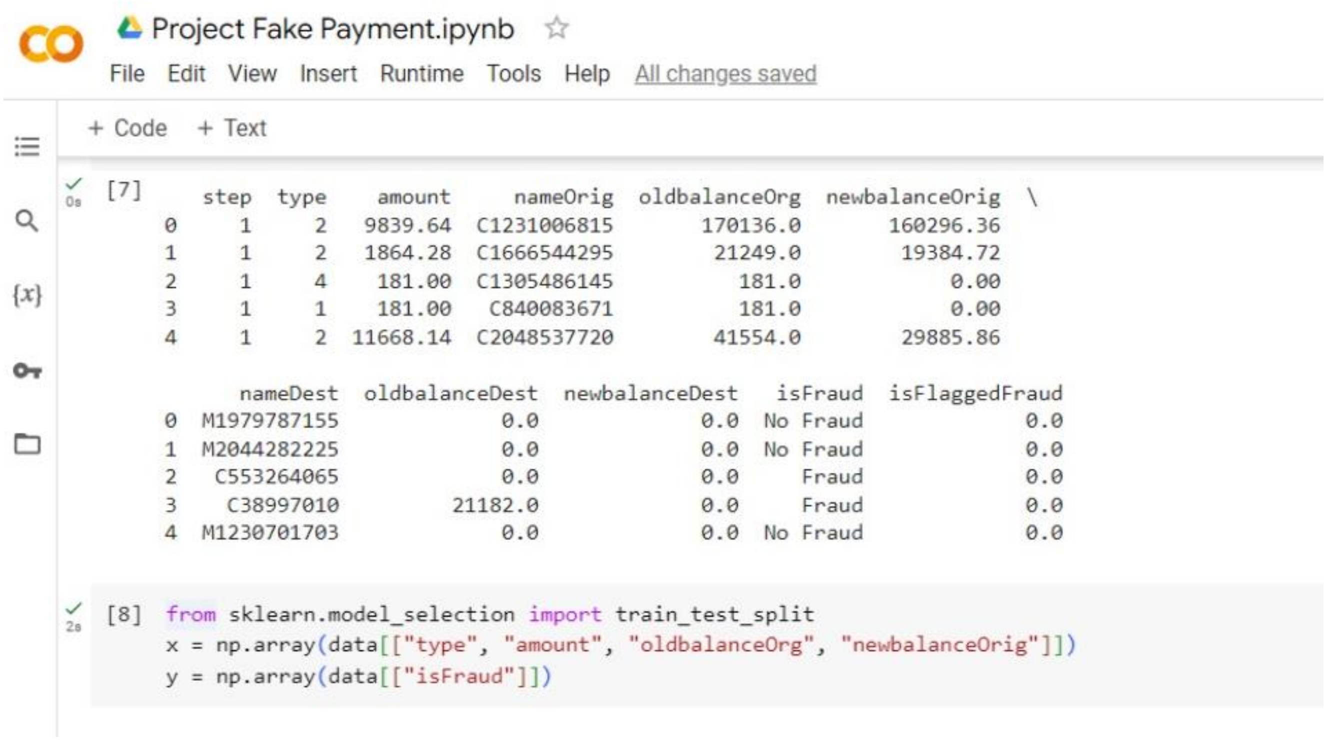



Project Fake Payment.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
[6] isFraud           1.000000
    amount            0.058899
    oldbalanceOrg    -0.004536
    newbalanceDest   -0.008193
    oldbalanceDest   -0.012463
    newbalanceOrig   -0.015376
    step             -0.050289
    isFlaggedFraud         NaN
    Name: isFraud, dtype: float64
    <ipython-input-6-91bfb1e64f5b>:1: FutureWarning:

    The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only t
```

```
data["type"] = data["type"].map({"CASH_OUT": 1, "PAYMENT": 2,
                                 "CASH_IN": 3, "TRANSFER": 4,
                                 "DEBIT": 5})
data["isFraud"] = data["isFraud"].map({0: "No Fraud", 1: "Fraud"})
print(data.head())
```


Project Fake Payment.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
[7]     step  type    amount     nameOrig  oldbalanceOrg  newbalanceOrig  \
     0     1     2   9839.64  C1231006815       170136.0       160296.36
     1     1     2   1864.28  C1666544295        21249.0        19384.72
     2     1     4    181.00  C1305486145          181.0            0.00
     3     1     1    181.00   C840083671          181.0            0.00
     4     1     2  11668.14  C2048537720        41554.0        29885.86

          nameDest  oldbalanceDest  newbalanceDest   isFraud  isFlaggedFraud
     0  M1979787155             0.0             0.0  No Fraud             0.0
     1  M2044282225             0.0             0.0  No Fraud             0.0
     2   C553264065             0.0             0.0     Fraud             0.0
     3    C38997010         21182.0             0.0     Fraud             0.0
     4  M1230701703             0.0             0.0  No Fraud             0.0
```

```
[8] from sklearn.model_selection import train_test_split
    x = np.array(data[["type", "amount", "oldbalanceOrg", "newbalanceOrig"]])
    y = np.array(data[["isFraud"]])
```

demonstration steps:

1.Ensure you have the required libraries installed. We can install them using pip

2.Place the CSV file "credit card.csv" in the same directory where your Python script or Jupyter Notebook is located. The CSV file contains the data we want to analyze.

3.Create a Python script or Jupyter Notebook, and copy and paste the above code into it.

4.Run the script or Notebook. If you're using a Jupyter Notebook, you can run each cell individually. If you're using a Python script, you can execute the entire script.

5.You will see the following outputs and visualizations:

The first few rows of the dataset.

The sum of missing values in each column.

The distribution of transaction types in a pie chart.

The correlation values with the 'isFraud' column.

The accuracy score of the Decision Tree classifier on the test data.

The prediction for the given set of features

Explanation of project code:

Importing Libraries:

python

```
import pandas as pd
import numpy as np
```

These lines import two essential libraries for data manipulation and numerical operations, namely Pandas and NumPy.

Loading Data:

python

```
data = pd.read_csv("credit card.csv")
```

This line reads a CSV file named "credit card.csv" using Pandas' read_csv function and stores the data in a DataFrame called data.

Printing the First Few Rows:

python

```
print(data.head())
```

This line prints the first few rows (default 5 rows) of the DataFrame data to the console, providing a quick look at the dataset.

Checking for Missing Values:

python

```
print(data.isnull().sum())
```

Here, it prints the sum of missing values for each column in the DataFrame, helping to identify if there are any missing data points in the dataset.

Exploring Transaction Types:

python

```
print(data.type.value_counts())
```

This line calculates and prints the count of each unique value in the 'type' column, which presumably represents transaction types. It gives you insight into the distribution of transaction types in the dataset.

Creating a Pie Chart Using Plotly:

python

```
import plotly.express as px
figure = px.pie(data, values=quantity, names=transactions, hole=0.5, title="Distribution of Transaction Type")
figure.show()
```

This code uses the Plotly library to create a pie chart to visualize the distribution of transaction types. The quantity and transactions variables are used to define the values and names for the chart slices. The resulting chart is displayed using figure.show().

Checking Correlation with 'isFraud':

python

```
correlation = data.corr()
print(correlation["isFraud"].sort_values(ascending=False
))
```

It calculates the correlation between all numerical columns in the dataset and the 'isFraud' column. The code then prints the correlation values in descending order, showing how strongly each feature is correlated with the 'isFraud' column.

Mapping Values:

python

```
data["type"] = data["type"].map({"CASH_OUT": 1,
"PAYMENT": 2, "CASH_IN": 3, "TRANSFER": 4,
"DEBIT": 5})
data["isFraud"] = data["isFraud"].map({0: "No Fraud", 1:
"Fraud"})
```

This part of the code maps textual values in the 'type' and 'isFraud' columns to numeric values. This is often done for machine learning models that require numeric inputs.

Data Splitting:

python

```
from sklearn.model_selection import train_test_split
x = np.array(data[["type", "amount", "oldbalanceOrg",
"newbalanceOrig"]])
y = np.array(data[["isFraud"]])
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
test_size=0.10, random_state=42)
```

This part of the code prepares the data for machine learning. It splits the dataset into features (x) and labels (y) and then further splits it into training and testing sets using train_test_split. The feature array 'x' contains specific columns from the DataFrame, and 'y' contains the 'isFraud' column.

Training a Decision Tree Classifier:

python

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(xtrain, ytrain)
```

It imports the DecisionTreeClassifier from Scikit-Learn, initializes a model, and fits the model with the training data (xtrain and ytrain).

Model Evaluation:

python

```
print(model.score(xtest, ytest))
```

This line prints the accuracy score of the trained model on the testing data, which gives an indication of how well the model performs.

Making Predictions:

python

```
features = np.array([[4, 9000.60, 9000.60, 0.0]])
print(model.predict(features))
```

It creates an array 'features' containing values for 'type', 'amount', 'oldbalanceOrg', and 'newbalanceOrig' and then uses the trained model to make predictions on these features, displaying the result. This is an example of using the model for prediction.

The provided code covers data loading, exploration, preprocessing, model training, evaluation, and prediction using a Decision Tree classifier.