Data Collection and Preprocessing Phase

Date	20 SEP 2024
Team ID	738330
Project Title	Electric Motor Temperature Prediction using Machine Learning
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	**Description**
Data Overview	Basic statistics, dimensions, and structure of the data.
Univariate Analysis	Exploration of individual variables (mean, median, mode, etc.).
Bivariate Analysis	Relationships between two variables (correlation, scatter plots).
Multivariate Analysis	Patterns and relationships involving multiple variables.
Outliers and Anomalies	Identification and treatment of outliers.
Data Preprocessing Code Screenshots	
Loading Data	Code to load the dataset into the preferred environment (e.g., Python, R).

Handling Missing Data	Code for identifying and handling missing values.
Data Transformation	Code for transforming variables (scaling, normalization).
Feature Engineering	Code for creating new features or modifying existing ones.
Save Processed Data	Code to save the cleaned and processed data for future use.

Data Collection:

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset. **Data Set :**

In this project we have used csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link : https://www.kaggle.com/datasets/wkirgsn/electric-motor-temperature

Electric Motor Temperature | Kaggle..

185 hrs recordings from a permanent magnet synchronous motor (PMSM)..

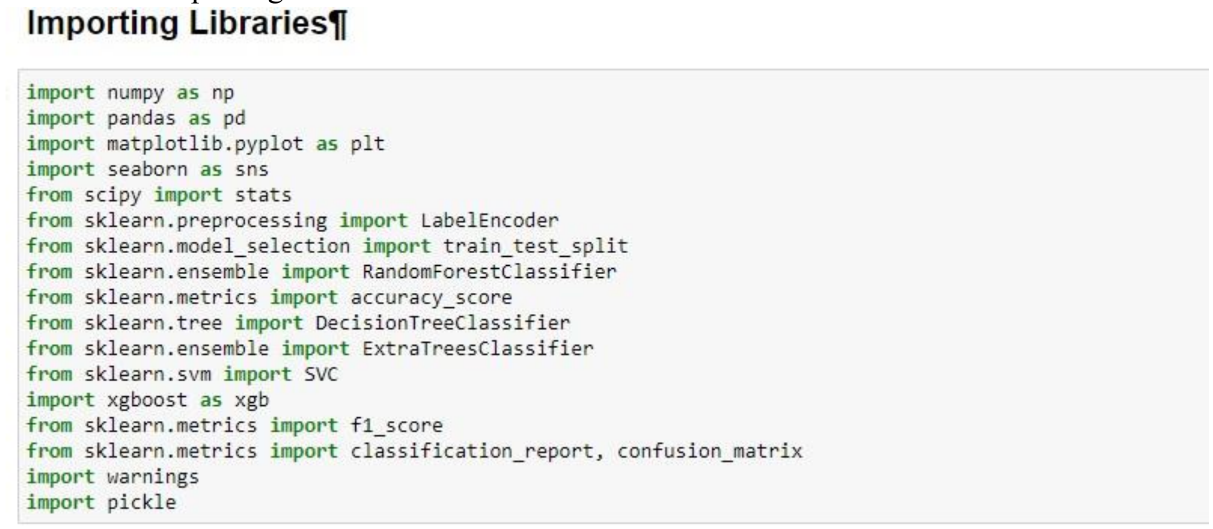
https://www.kaggle.com/wkirgsn/electric-motor-temperature

Visualizing And Analyzing Data:

Predicting electric motor temperature using machine learning involves several steps, including data collection, preprocessing, model selection, training, and evaluation.

Importing The Libraries:

To get started with electric motor temperature prediction using machine learning, you'll need to import the necessary libraries. Below is a comprehensive list of libraries you might use, along with the code for importing them.

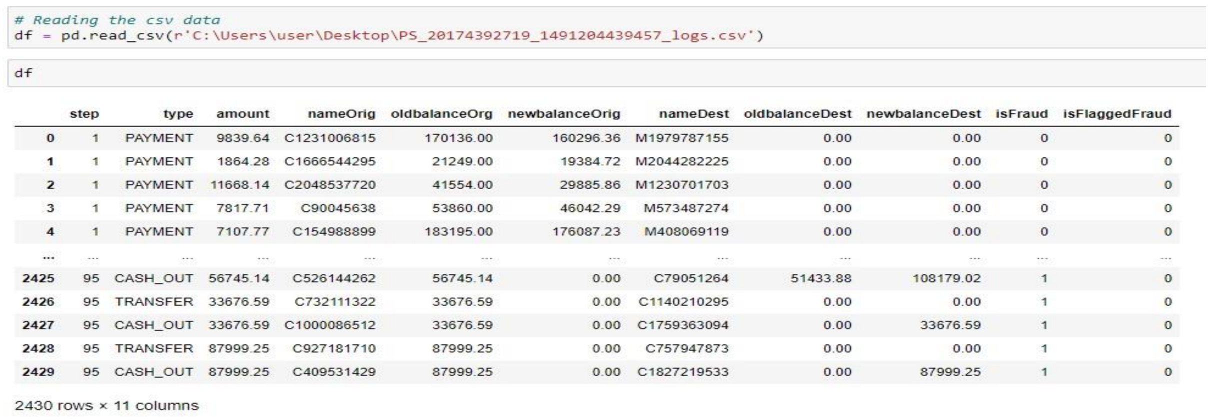


Importing Libraries¶

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.svm import SVC
import xgboost as xgb
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_matrix
import warnings
import pickle
```

Read The Dataset:

To get started with electric motor temperature prediction using machine learning, you'll need to import the necessary libraries. Below is a comprehensive list of libraries you might use, along with the code for importing them..



```
# Reading the csv data
df = pd.read_csv(r'C:\Users\user\Desktop\PS_20174392719_1491204439457_logs.csv')
```

```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.00	0.00	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.00	0.00	0	0
2	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.00	0.00	0	0
3	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274	0.00	0.00	0	0
4	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119	0.00	0.00	0	0
...	...	...	...	...	...	...	...	...	...	...	...
2425	95	CASH_OUT	56745.14	C526144262	56745.14	0.00	C79051264	51433.88	108179.02	1	0
2426	95	TRANSFER	33676.59	C732111322	33676.59	0.00	C1140210295	0.00	0.00	1	0
2427	95	CASH_OUT	33676.59	C1000086512	33676.59	0.00	C1759363094	0.00	33676.59	1	0
2428	95	TRANSFER	87999.25	C927181710	87999.25	0.00	C757947873	0.00	0.00	1	0
2429	95	CASH_OUT	87999.25	C409531429	87999.25	0.00	C1827219533	0.00	87999.25	1	0

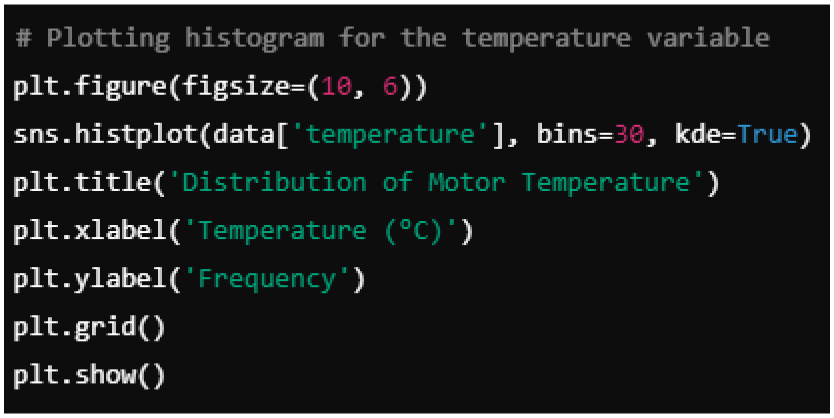
2430 rows × 11 columns

Univariate Analysis :

Univariate analysis focuses on analyzing each variable in your dataset individually to summarize and understand its distribution and characteristics. Here's how you can conduct univariate analysis for your electric motor temperature dataset using Python.

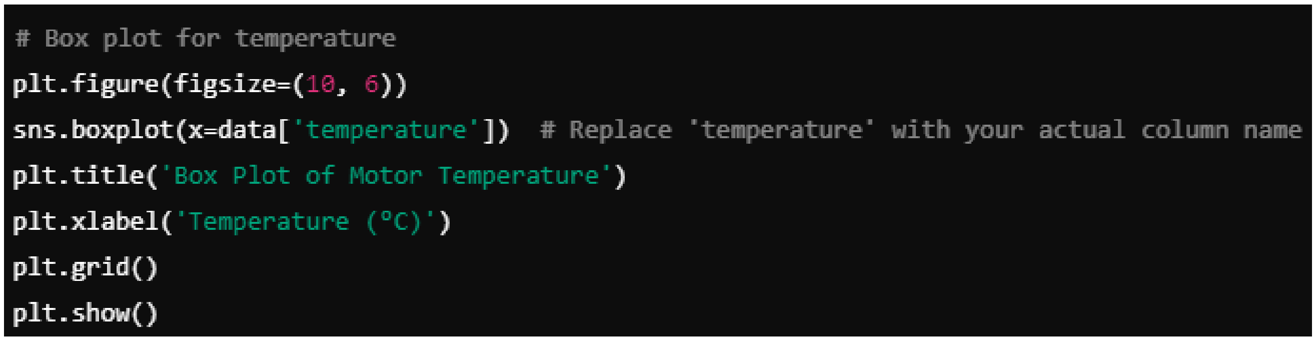
1. Visualizing the Distribution

Histograms: Histograms are useful for visualizing the distribution of a numerical variable.



```python
# Plotting histogram for the temperature variable
plt.figure(figsize=(10, 6))
sns.histplot(data['temperature'], bins=30, kde=True)
plt.title('Distribution of Motor Temperature')
plt.xlabel('Temperature (°C)')
plt.ylabel('Frequency')
plt.grid()
plt.show()
```

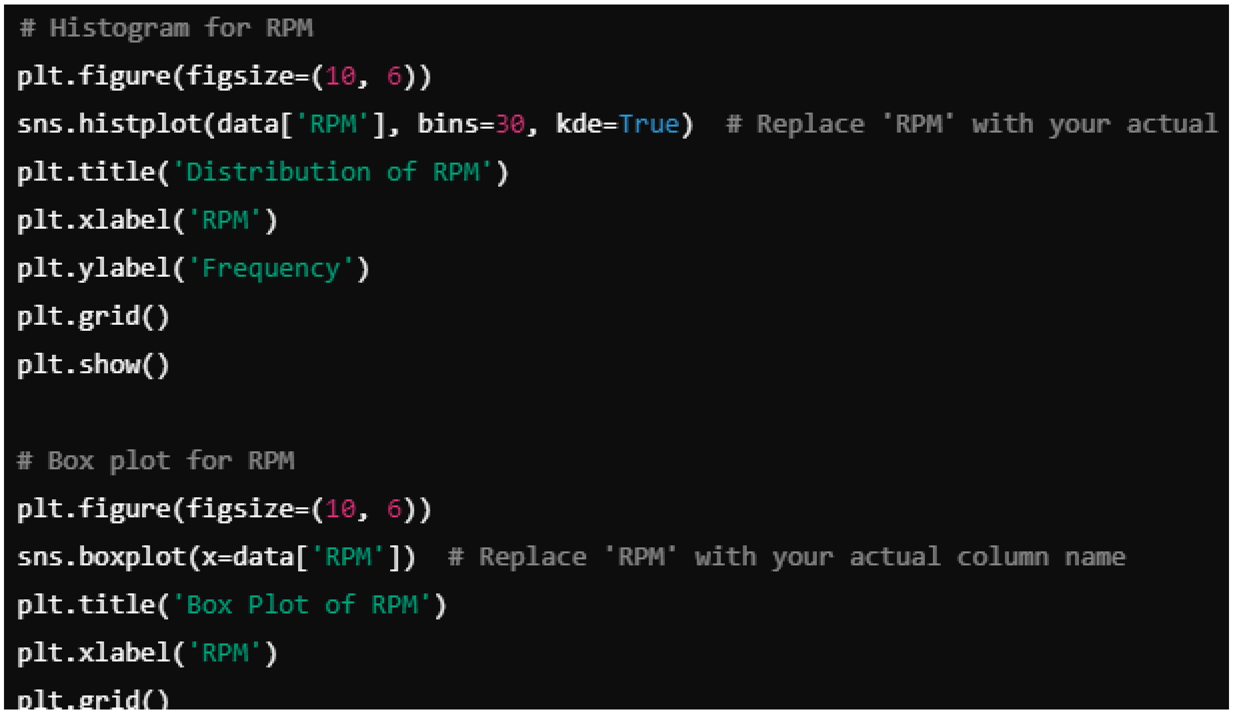
Box Plots: Box plots help identify outliers and visualize the summary statistics (median, quartiles).



```python
# Box plot for temperature
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['temperature'])  # Replace 'temperature' with your actual column name
plt.title('Box Plot of Motor Temperature')
plt.xlabel('Temperature (°C)')
plt.grid()
plt.show()
```

Visualizing Other Features

You can apply similar analysis techniques to other relevant features (e.g., RPM, current, voltage).



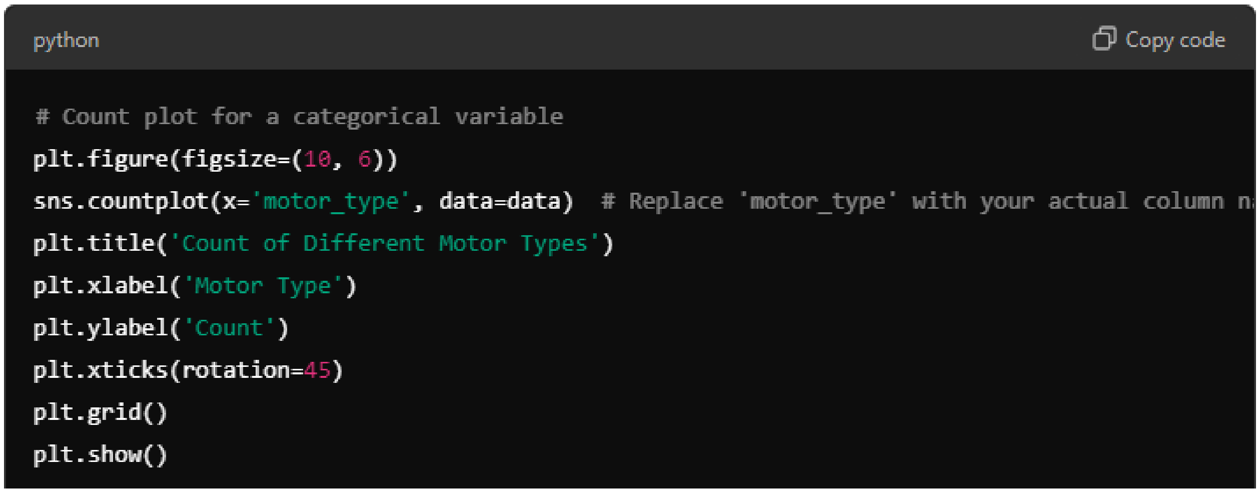
```python
# Histogram for RPM
plt.figure(figsize=(10, 6))
sns.histplot(data['RPM'], bins=30, kde=True)  # Replace 'RPM' with your actual
plt.title('Distribution of RPM')
plt.xlabel('RPM')
plt.ylabel('Frequency')
plt.grid()
plt.show()

# Box plot for RPM
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['RPM'])  # Replace 'RPM' with your actual column name
plt.title('Box Plot of RPM')
plt.xlabel('RPM')
plt.grid()
```

5. Categorical Variables

If you have categorical variables (e.g., motor type, operational status), you can analyze them as well:

Count Plots:



python Copy code

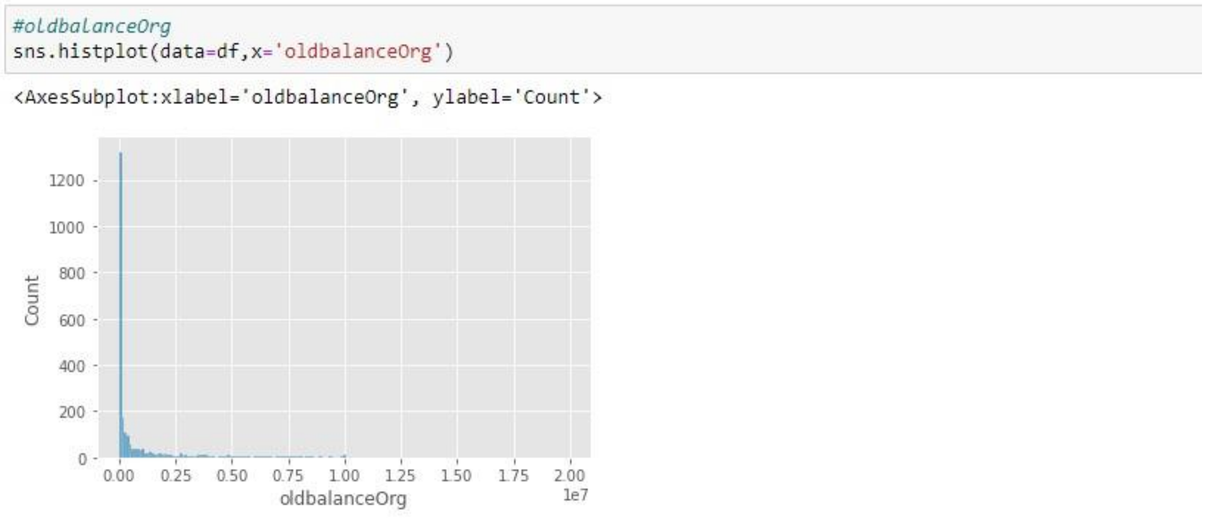
```python
# Count plot for a categorical variable
plt.figure(figsize=(10, 6))
sns.countplot(x='motor_type', data=data)  # Replace 'motor_type' with your actual column n
plt.title('Count of Different Motor Types')
plt.xlabel('Motor Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid()
plt.show()
```

Here, the relationship between the amount attribute and the boxplot is visualised.

- Understand the distribution of each variable.

- Identify potential outliers.
- Gather summary statistics.
- Explore relationships in categorical data.

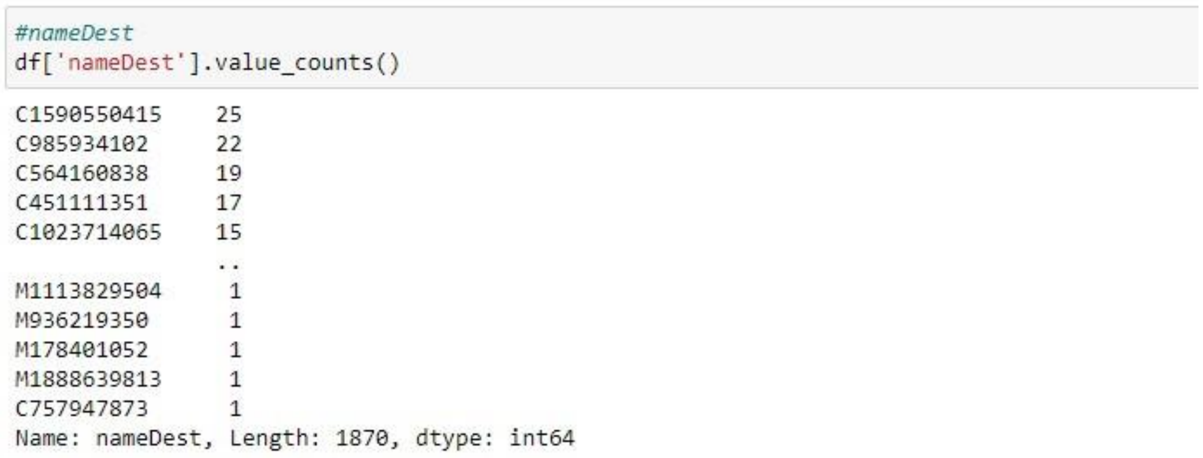
Once you complete univariate analysis, you can move on to bivariate or multivariate analysis to explore relationships between variables. If you have any specific questions or need further assistance, feel free to ask



```
#oldbalanceOrg
sns.histplot(data=df,x='oldbalanceOrg')
```

```
<AxesSubplot:xlabel='oldbalanceOrg', ylabel='Count'>
```

By creating bins along the data's range and then drawing bars to reflect the number of observations that fall within the oldbalanceOrg attribute in the dataset.



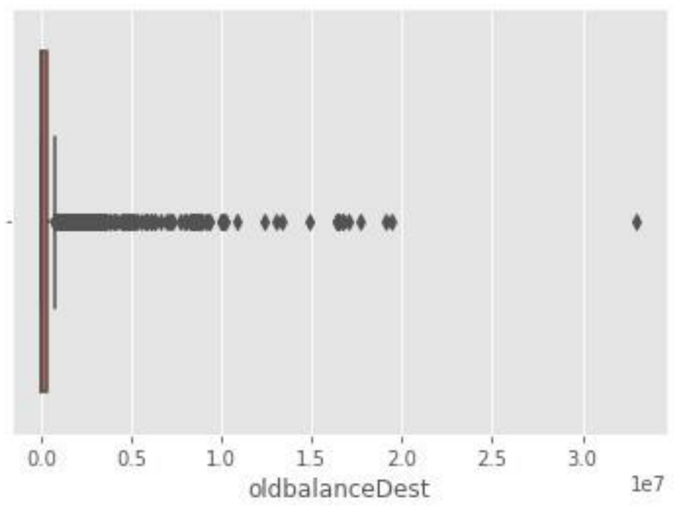
```
#nameDest
df['nameDest'].value_counts()
```

```
C1590550415    25
C985934102     22
C564160838     19
C451111351     17
C1023714065    15
               ..
M1113829504     1
M936219350      1
M178401052      1
M1888639813     1
C757947873      1
Name: nameDest, Length: 1870, dtype: int64
```

utilising the value counts() function here to determine how many times the nameDest column appears.

```
: #oldbalanceDest
  sns.boxplot(data=df,x='oldbalanceDest')
```

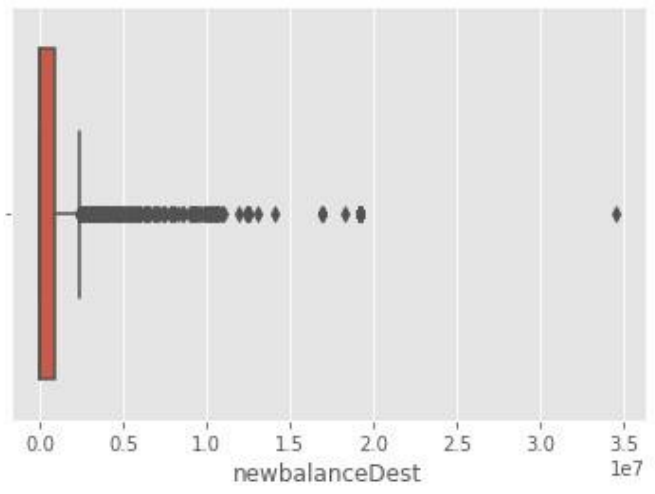
```
: <AxesSubplot:xlabel='oldbalanceDest'>
```



Here, the relationship between the oldbalanceDest attribute and the boxplot is visualised.

```
#newbalanceDest
sns.boxplot(data=df,x='newbalanceDest')
```

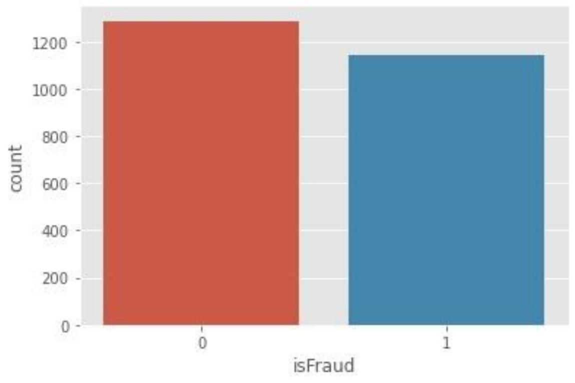
```
<AxesSubplot:xlabel='newbalanceDest'>
```



Here, the relationship between the newbalanceDest attribute and the boxplot is visualised.

```
#isFraud:
sns.countplot(data=df,x='isFraud')
```

```
<AxesSubplot:xlabel='isFraud', ylabel='count'>
```



using the countplot approach here to count the number of instances in the dataset's target isFraud column.

```
df['isFraud'].value_counts()
```

```
0    1288
1    1142
Name: isFraud, dtype: int64
```

Here, we're using the value counts method to figure out how many classes there are