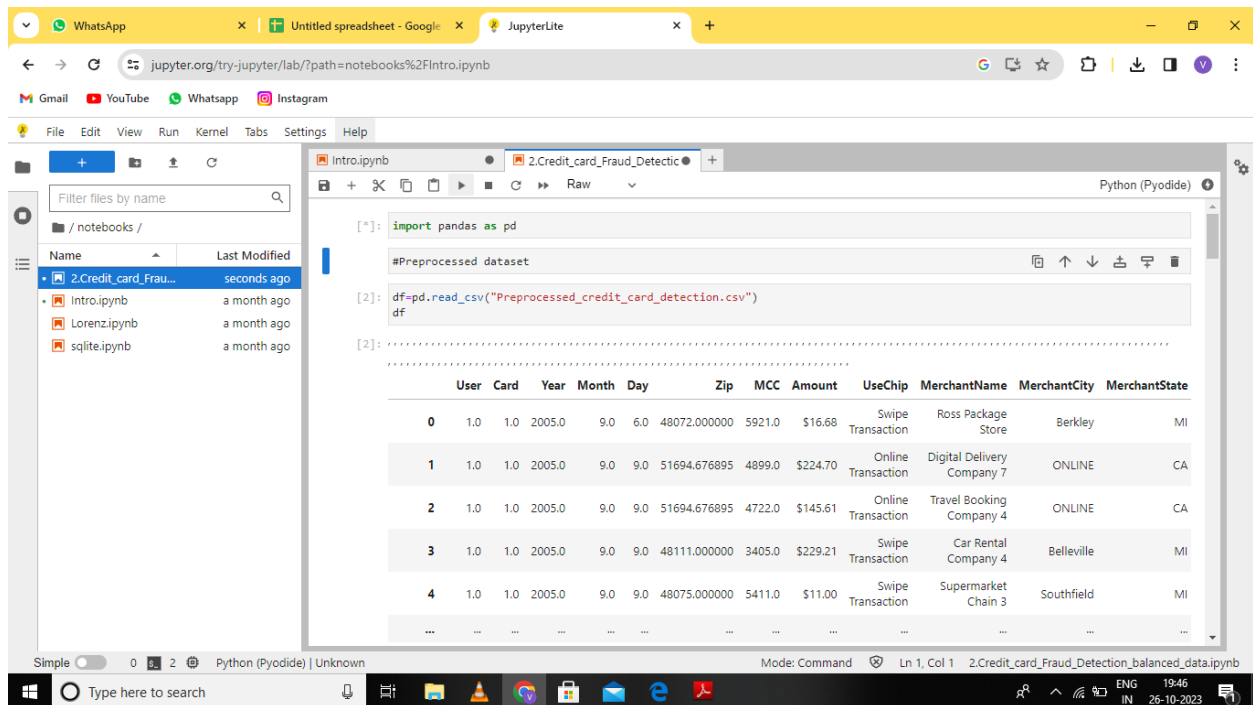


# CREDIT CARD FRAUD DETECTION

Credit card fraud detection is a crucial process in the financial industry designed to identify and prevent unauthorized or fraudulent transactions using credit cards. It involves the use of advanced algorithms and machine learning techniques to analyze transaction data, detect suspicious patterns, and flag potentially fraudulent activities, ultimately safeguarding both cardholders and financial institutions from financial losses and security breaches.

Starting with our coding, we have already collected our dataset and have preprocessed the data. With the preprocessed dataset we have done our coding.

Let's start with the coding explanation and the output that we have got from Jupyter notebook. First of all we have imported the pandas to gain access to many functions for performing data analysis in Python.



```
[*]: import pandas as pd

#Preprocessed dataset

[2]: df=pd.read_csv("Preprocessed_credit_card_detection.csv")
df
```

	User	Card	Year	Month	Day	Zip	MCC	Amount	UseChip	MerchantName	MerchantCity	MerchantState
0	1.0	1.0	2005.0	9.0	6.0	48072.000000	5921.0	\$16.68	Swipe Transaction	Ross Package Store	Berkley	MI
1	1.0	1.0	2005.0	9.0	9.0	51694.676895	4899.0	\$224.70	Online Transaction	Digital Delivery Company 7	ONLINE	CA
2	1.0	1.0	2005.0	9.0	9.0	51694.676895	4722.0	\$145.61	Online Transaction	Travel Booking Company 4	ONLINE	CA
3	1.0	1.0	2005.0	9.0	9.0	48111.000000	3405.0	\$229.21	Swipe Transaction	Car Rental Company 4	Belleville	MI
4	1.0	1.0	2005.0	9.0	9.0	48075.000000	5411.0	\$11.00	Swipe Transaction	Supermarket Chain 3	Southfield	MI
...	...	...	...	...	...	...	...	...	...	...	...	...

After importing pandas, we have coded to read the CSV file to have the code user efficient. So for reading the data set is coded. The output diagram is shown below.

The screenshot shows a JupyterLab interface with a file browser on the left and a code editor on the right. The file browser lists several notebooks: '2.Credit\_card\_Frau...', 'Intro.ipynb', 'Lorenz.ipynb', and 'sqlite.ipynb'. The code editor displays the following Python code:

```
[*]: import pandas as pd

#Preprocessed dataset

[*]: df=pd.read_csv("Preprocessed_credit_card_detection.csv")
df

[3]: df.columns

[3]: Index(['User', 'Card', 'Year', 'Month', 'Day', 'Zip', 'MCC', 'Amount',
         'UseChip', 'MerchantName', 'MerchantCity', 'MerchantState',
         'IssFraud?'],
         dtype='object')

[4]: df["IssFraud?"].value_counts()

[4]: IssFraud?
     No    691048
     Yes      872
     Name: count, dtype: int64

[5]: df.isnull().sum()

[5]: User      0
     Card      0
     Year      0
     Month     0
```

The status bar at the bottom indicates the mode is 'Command' and the current file is '2.Credit\_card\_Fraud\_Detection\_balanced\_data.ipynb'.

Next we are looking over to the columns. We have coded for the columns to read in the dataset for the user to have more efficient ways to look into the dataset.

The screenshot shows a JupyterLab interface with a file browser on the left and a code editor on the right. The file browser lists several notebooks: '2.Credit\_card\_Frau...', 'Intro.ipynb', 'Lorenz.ipynb', and 'sqlite.ipynb'. The code editor displays the following Python code:

```
[*]: import pandas as pd

#Preprocessed dataset

[*]: df=pd.read_csv("Preprocessed_credit_card_detection.csv")
df

[*]: df.columns

[4]: df["IssFraud?"].value_counts()

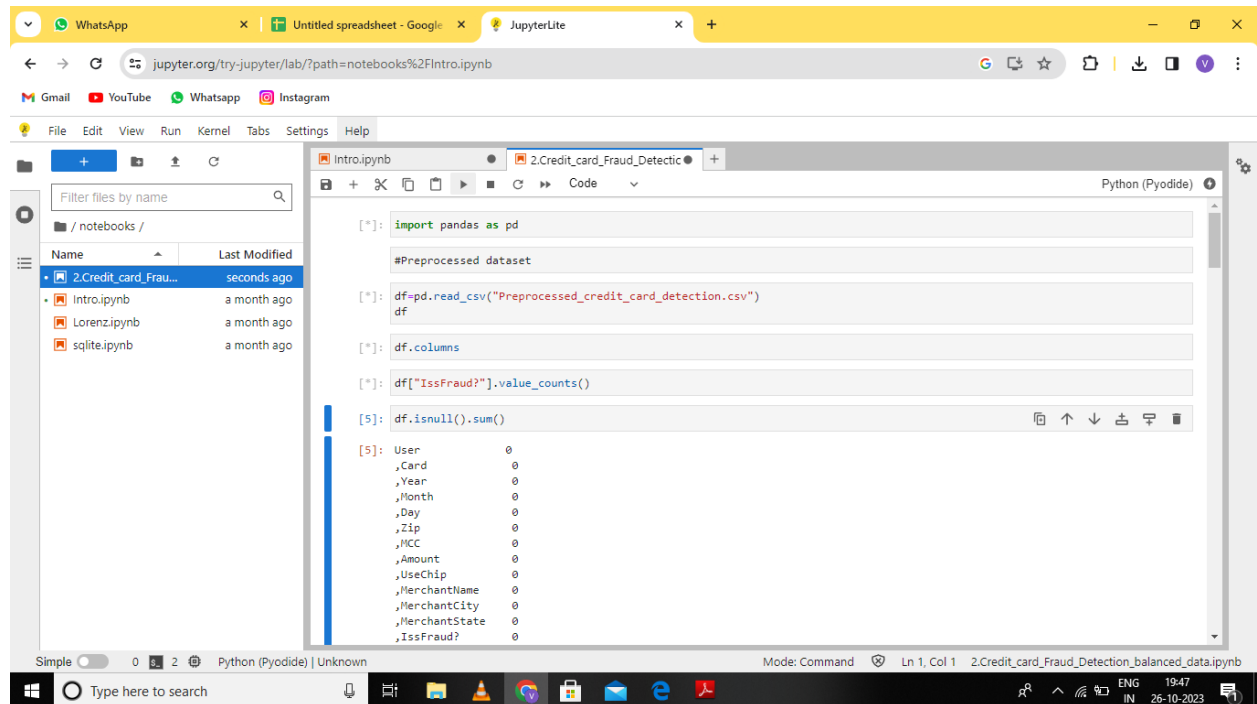
[4]: IssFraud?
     No    691048
     Yes      872
     Name: count, dtype: int64

[5]: df.isnull().sum()

[5]: User      0
     Card      0
     Year      0
     Month     0
     Day       0
     Zip       0
     MCC       0
     Amount    0
```

The status bar at the bottom indicates the mode is 'Command' and the current file is '2.Credit\_card\_Fraud\_Detection\_balanced\_data.ipynb'.

Next, we are heading to the number of fraudness that we have in our collection of our data. So we have to count the number of fraudness for the code to be satisfied with the output. We have coded for the count of the fraudness.



The screenshot shows a JupyterLab interface with a notebook titled '2.Credit\_card\_Fraud\_Detectic'. The code in the notebook is as follows:

```
[*]: import pandas as pd

#Preprocessed dataset

[*]: df=pd.read_csv("Preprocessed_credit_card_detection.csv")
df

[*]: df.columns

[*]: df[["IssFraud?"]].value_counts()

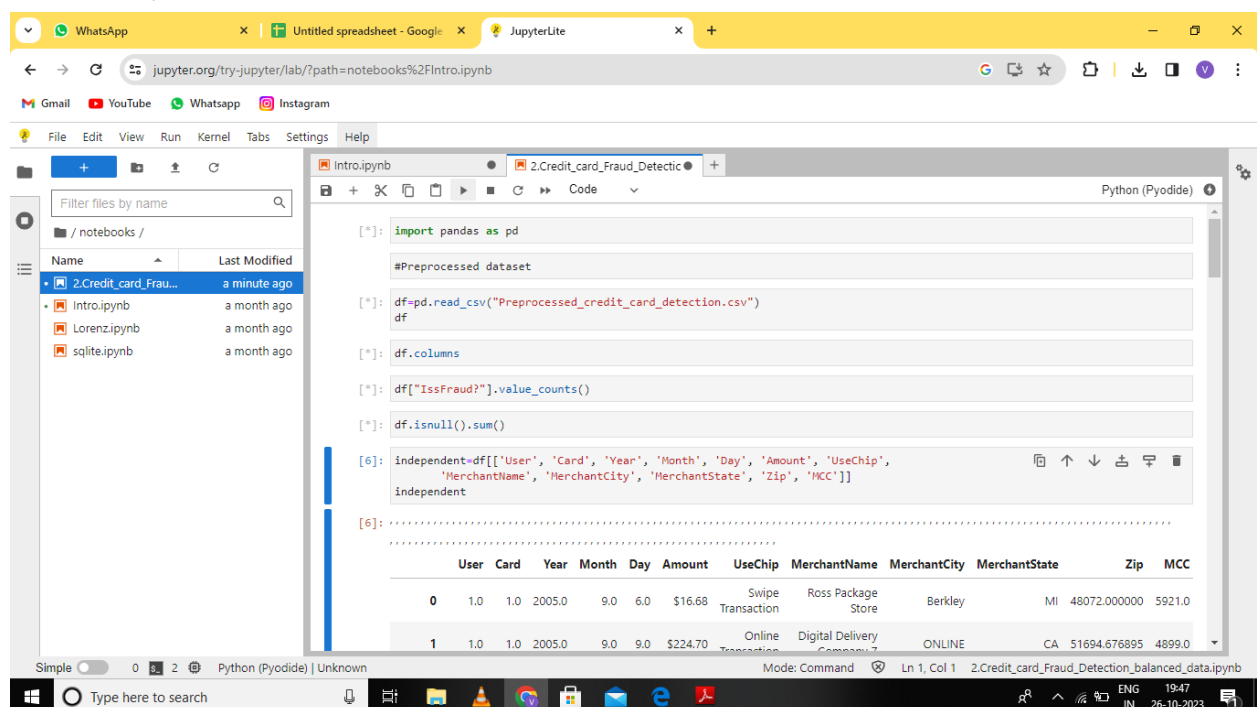
[5]: df.isnull().sum()

[5]: User          0
     ,Card         0
     ,Year         0
     ,Month        0
     ,Day          0
     ,Zip          0
     ,MCC          0
     ,Amount       0
     ,UseChip      0
     ,MerchantName  0
     ,MerchantCity  0
     ,MerchantState 0
     ,IssFraud?    0
```

The left sidebar shows a file explorer with the following files:

Name	Last Modified
2.Credit_card_Frau...	seconds ago
Intro.ipynb	a month ago
Lorenz.ipynb	a month ago
sqlite.ipynb	a month ago

Now, to start the count we have to start with zero. So coding was continued with null values i.e., zero and then to the count.



The screenshot shows a JupyterLab interface with a notebook titled '2.Credit\_card\_Fraud\_Detectic'. The code in the notebook is as follows:

```
[*]: import pandas as pd

#Preprocessed dataset

[*]: df=pd.read_csv("Preprocessed_credit_card_detection.csv")
df

[*]: df.columns

[*]: df[["IssFraud?"]].value_counts()

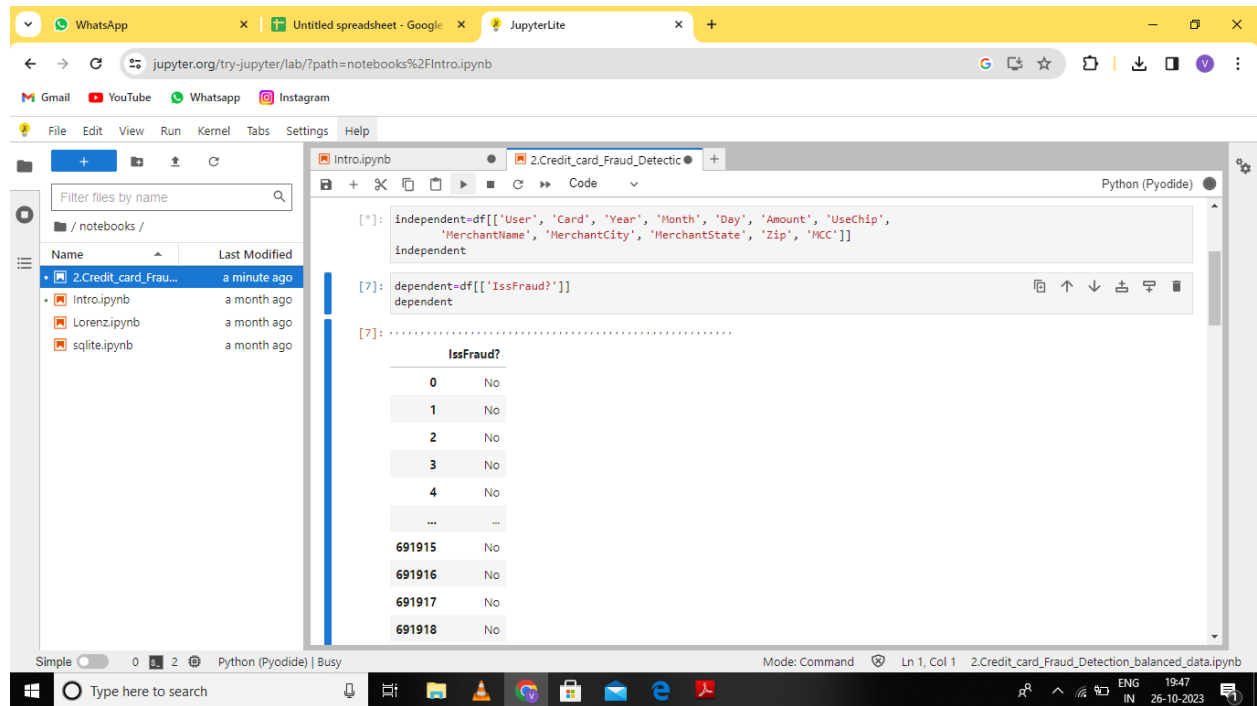
[*]: df.isnull().sum()

[6]: independent=df[["User", "Card", "Year", "Month", "Day", "Amount", "UseChip",
    'MerchantName', 'MerchantCity', 'MerchantState', 'Zip', 'MCC']]
independent

[6]:
```

	User	Card	Year	Month	Day	Amount	UseChip	MerchantName	MerchantCity	MerchantState	Zip	MCC
0	1.0	1.0	2005.0	9.0	6.0	\$16.68	Swipe Transaction	Ross Package Store	Berkley	MI	48072.000000	5921.0
1	1.0	1.0	2005.0	9.0	9.0	\$224.70	Online Transaction	Digital Delivery Company	ONLINE	CA	51694.676895	4899.0

Next forwarding we are heading over to the headings or the title of each column and printing those columns as the next analysis.



The screenshot displays a JupyterLab environment. On the left, a file explorer shows a directory named 'notebooks' containing several files: '2.Credit\_card\_Fraud\_Detect...', 'Intro.ipynb', 'Lorenz.ipynb', and 'sqlite.ipynb'. The main area is a code editor with two cells. The first cell contains the following code:

```
[*]: independent=df[['User', 'Card', 'Year', 'Month', 'Day', 'Amount', 'UseChip',  
                    'MerchantName', 'MerchantCity', 'MerchantState', 'Zip', 'MCC']]  
independent
```

The second cell contains the following code:

```
[7]: dependent=df[['IsFraud?']]  
dependent
```

The console output on the right shows the result of the second cell, displaying a DataFrame with the column 'IsFraud?'. The output is as follows:

	IsFraud?
0	No
1	No
2	No
3	No
4	No
...	...
691915	No
691916	No
691917	No
691918	No

We are now considering the title of columns in the name of the variable independent and the number of counts or the value where fraudness happens corresponding to our dataset in the name of the variable dependent.

The screenshot shows a JupyterLab interface with a notebook titled '2.Credit\_card\_Fraud\_Detectio'. The code in the notebook is as follows:

```
[*]: independent=df[['User', 'Card', 'Year', 'Month', 'Day', 'Amount', 'UseChip',
'MerchantName', 'MerchantCity', 'MerchantState', 'Zip', 'MCC']]
independent

[*]: dependent=df[['IssFraud?']]
dependent

#Using undersampling balance the data

[8]: from imblearn.under_sampling import RandomUnderSampler
ros=RandomUnderSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(independent,dependent)

[9]: x_ros.shape

[9]: (1744, 12)

[10]: y_ros.value_counts()

[10]: IssFraud?
,No      872
,Yes     872
,Name: count, dtype: int64

[11]: x_ros
```

The output of the code shows the shape of the independent variable as (1744, 12) and the value counts for the dependent variable 'IssFraud?' as No: 872 and Yes: 872.

Next we are creating `x_ros` and `y_ros` variables to denote the variables independent and dependent which we have explained above with the picture of the output for the given data set.

The screenshot shows a JupyterLab interface with a notebook titled '2.Credit\_card\_Fraud\_Detectio'. The code in the notebook is as follows:

```
#Using undersampling balance the data

[8]: from imblearn.under_sampling import RandomUnderSampler
ros=RandomUnderSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(independent,dependent)

[9]: x_ros.shape

[9]: (1744, 12)

[10]: y_ros.value_counts()

[10]: IssFraud?
,No      872
,Yes     872
,Name: count, dtype: int64

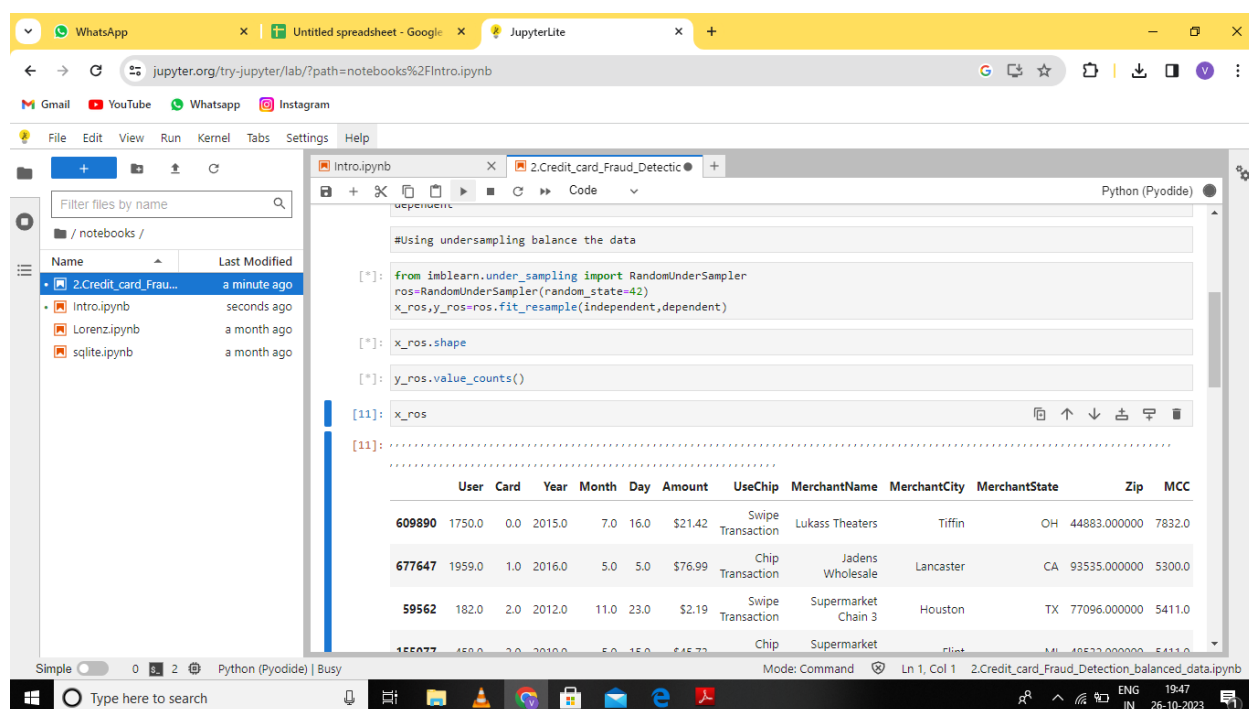
[11]: x_ros

[11]: .....
```

The output of the code shows the shape of the independent variable as (1744, 12) and the value counts for the dependent variable 'IssFraud?' as No: 872 and Yes: 872. A sample data row is also shown at the bottom:

	User	Card	Year	Month	Day	Amount	UseChip	MerchantName	MerchantCity	MerchantState	Zip	MCC
609890	1750.0	0.0	2015.0	7.0	16.0	\$21.42	Swipe Transaction	Lukass Theaters	Tiffin	OH	44883.000000	7832.0

After denoting the variable x\_ros and y\_ros, we are coding for the shape of x\_ros and then we are evaluating the number of counts for y\_ros.



The screenshot shows a JupyterLab environment with a Python notebook. The code in the notebook is as follows:

```
#Using undersampling balance the data

[*]: from imblearn.under_sampling import RandomUnderSampler
ros=RandomUnderSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(independent,dependent)

[*]: x_ros.shape

[*]: y_ros.value_counts()

[11]: x_ros

[11]:
```

The output of the last cell shows a preview of the dataset:

	User	Card	Year	Month	Day	Amount	UseChip	MerchantName	MerchantCity	MerchantState	Zip	MCC
609890	1750.0	0.0	2015.0	7.0	16.0	\$21.42	Swipe Transaction	Lukass Theaters	Tiffin	OH	44883.000000	7832.0
677647	1959.0	1.0	2016.0	5.0	5.0	\$76.99	Chip Transaction	Jadens Wholesale	Lancaster	CA	93535.000000	5300.0
59562	182.0	2.0	2012.0	11.0	23.0	\$2.19	Swipe Transaction	Supermarket Chain 3	Houston	TX	77096.000000	5411.0
155077	150.0	2.0	2010.0	5.0	15.0	\$15.73	Chip	Supermarket	Flint	MI	48532.000000	5411.0

Before finishing the last topic of our code with the corresponding given dataset let us look at a short note on Feature engineering.

Feature engineering is a fundamental step in machine learning and data analysis. It involves selecting, transforming, and creating relevant features from raw data to improve the performance of predictive models. Effective feature engineering can enhance the model's ability to extract patterns and make accurate predictions by presenting the data in a more meaningful and informative way, leading to better outcomes in various applications such as classification, regression, and clustering.

Feature engineering is the process of creating new, informative features from existing data to improve the performance of machine learning models. It involves selecting, transforming, or generating relevant features that capture important information from the data. Effective feature engineering can lead to better model accuracy and generalization. It often requires domain knowledge and creativity to identify the most meaningful features for a given problem. Common techniques include one-hot encoding, scaling, and creating interaction terms.

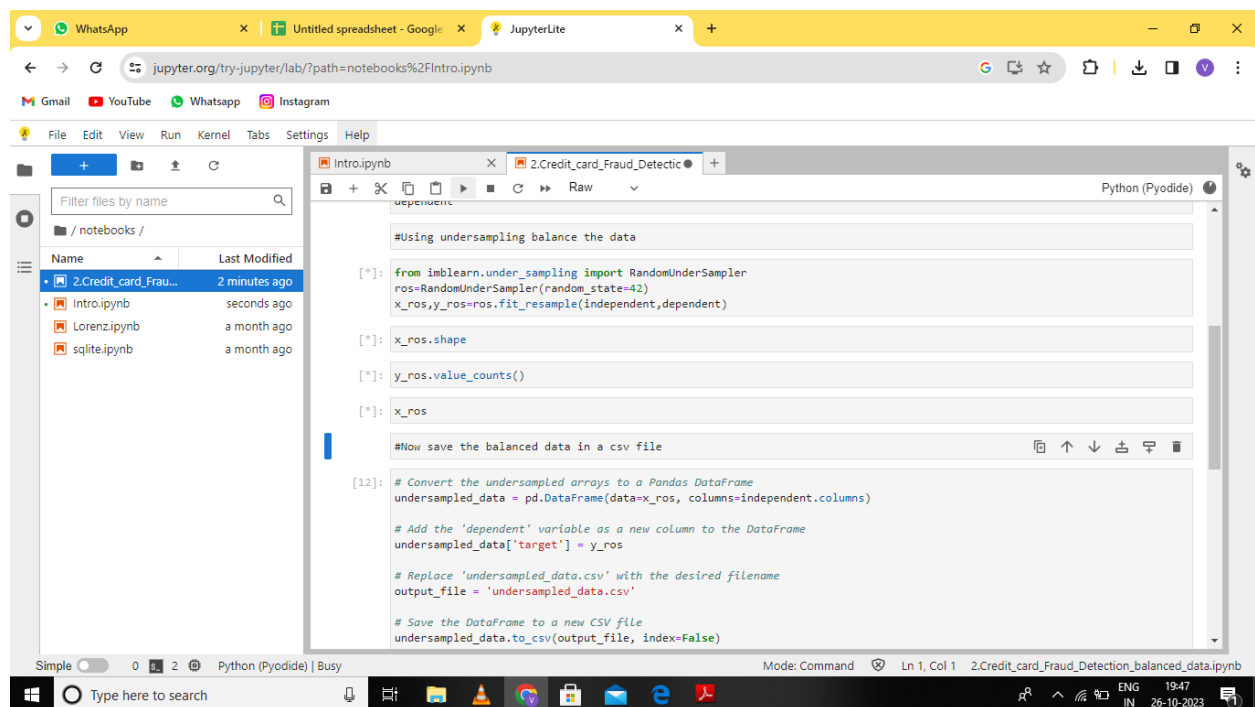
Feature engineering is a crucial step in the machine learning pipeline, as the quality of features can significantly impact the performance of a model. Here are some key aspects to consider:

1. **\*Feature Selection:\*** Sometimes, it's important to choose the most relevant features and discard irrelevant ones to reduce dimensionality and computational complexity. Techniques like correlation analysis, mutual information, and feature importance from tree-based models can help in selecting the most important features.
2. **\*Feature Transformation:\*** This involves converting or scaling features to make them suitable for modeling. Common transformations include normalization, standardization, and logarithmic scaling.
3. **\*Feature Generation:\*** In some cases, you can create new features from existing ones. For instance, in a time series dataset, you can generate features like moving averages, time lags, or statistical moments (mean, variance) to capture temporal patterns.
4. **\*One-Hot Encoding:\*** Categorical variables need to be encoded numerically for machine learning models. One-hot encoding is a common technique where each category is transformed into a binary vector, with each category represented as a binary variable.
5. **\*Handling Missing Data:\*** Dealing with missing values is crucial. You can choose to impute missing data with mean, median, mode, or use more advanced methods like regression imputation or techniques like K-nearest neighbors.
6. **\*Engineering Interaction Features:\*** Creating interaction features can help capture relationships between different variables. For example, if you have variables for "distance" and "time," you can create a new feature for "speed."
7. **\*Domain Knowledge:\*** Often, domain-specific knowledge is invaluable for feature engineering. Understanding the problem domain can help you identify which features are likely to be important and how they should be manipulated.
8. **\*Regularization:\*** When using linear models, L1 or L2 regularization can automatically perform feature selection by penalizing or shrinking less important features.
9. **\*Cross-Validation:\*** Feature engineering should be done within cross-validation loops to ensure that the performance improvements generalize to unseen data.

10. \*Automated Feature Engineering:\* There are tools and libraries that can automatically generate and select features. AutoML platforms often incorporate these capabilities.

The goal of feature engineering is to provide the model with the most relevant and informative data, which can lead to improved model accuracy, interpretability, and generalization to new, unseen data. It's often an iterative process that involves experimenting with different feature combinations and transformations to find the best representation of the data for the task at hand.

So the last code line that we have done to implement credit card fraud detection is to print the independent and dependent as we have given them those two variables as `x_ros` and `y_ros`. So we have shown the output of those two variables.



The screenshot shows a JupyterLab interface with a browser window at the top displaying the URL `jupyter.org/try-jupyter/lab/?path=notebooks%2FIntro.ipynb`. Below the browser, there's a sidebar with a file explorer showing a list of notebooks: `2.Credit_card_Frau...` (2 minutes ago), `Intro.ipynb` (seconds ago), `Lorenz.ipynb` (a month ago), and `sqlite.ipynb` (a month ago). The main area displays a notebook with the following code:

```
#Using undersampling balance the data

[*]: from imblearn.under_sampling import RandomUnderSampler
ros=RandomUnderSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(independent,dependent)

[*]: x_ros.shape

[*]: y_ros.value_counts()

[*]: x_ros

#Now save the balanced data in a csv file

[12]: # Convert the undersampled arrays to a Pandas DataFrame
undersampled_data = pd.DataFrame(data=x_ros, columns=independent.columns)

# Add the 'dependent' variable as a new column to the DataFrame
undersampled_data['target'] = y_ros

# Replace 'undersampled_data.csv' with the desired filename
output_file = 'undersampled_data.csv'

# Save the DataFrame to a new CSV file
undersampled_data.to_csv(output_file, index=False)
```

So, we have much more code to implement this project i.e., Credit card fraud detection. But to show the implementation of our project with the given dataset we have to print the number of fraudness counts that start with null value and detect whether any fraudness has happened in credit cards. Credit card fraud detection is crucial for two main reasons:

Financial Protection: It safeguards against unauthorized transactions, preventing financial losses for both cardholders and financial institutions.



Regulatory Compliance: Implementing fraud detection helps institutions comply with legal requirements and industry standards, reducing the risk of penalties and maintaining customer trust.

To finish off this we are implementing this project.