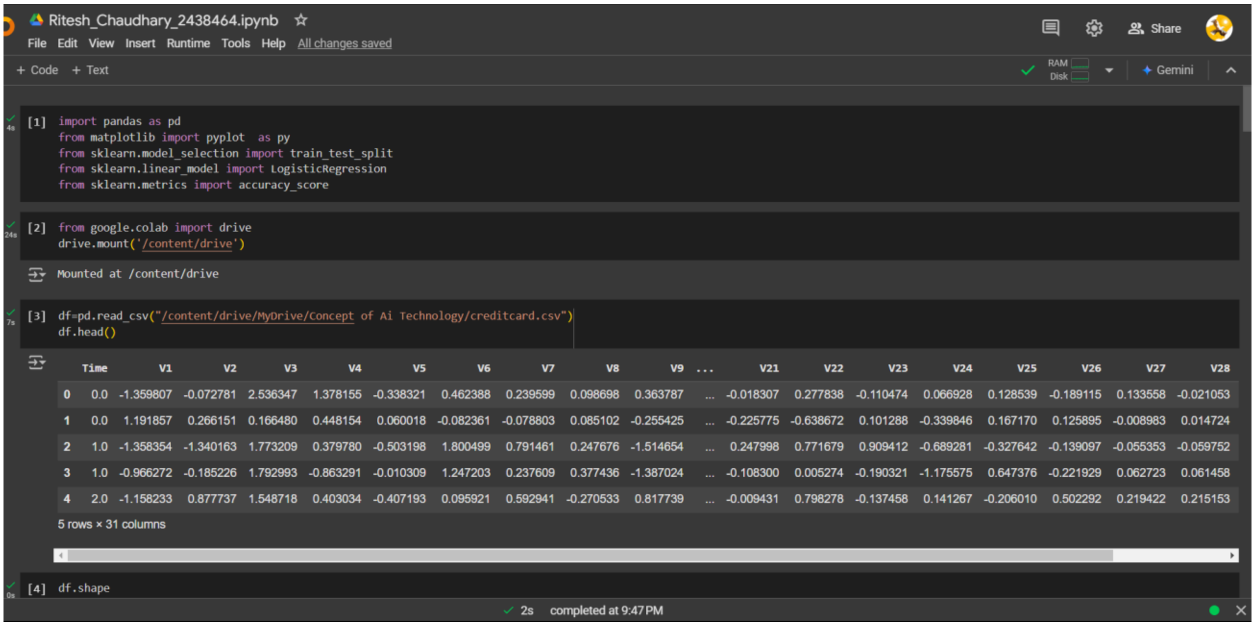
**Name: Ritesh Chaudhary**

**Id: 2438464**

# Workshop week-7



Ritesh_Chaudhary_2438464.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
RAM
Disk
Gemini

```
[1] import pandas as pd
    from matplotlib import pyplot  as py
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import accuracy_score
```

```
[2] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

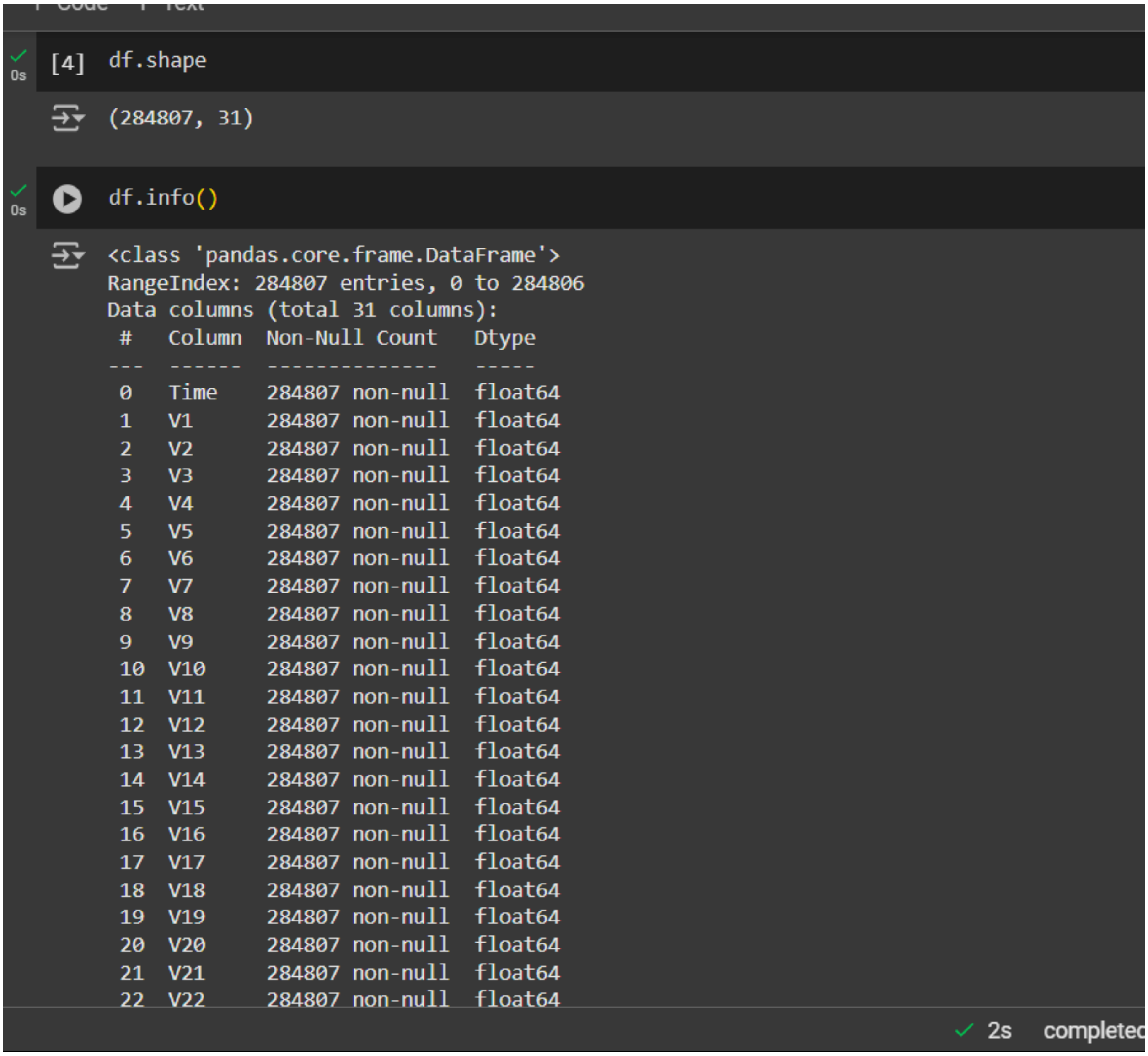
```
[3] df=pd.read_csv("/content/drive/MyDrive/Concept of Ai Technology/creditcard.csv")
    df.head()
```

|  | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | ... | V21 | V22 | V23 | V24 | V25 | V26 | V27 | V28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0.363787 | ... | -0.018307 | 0.277838 | -0.110474 | 0.066928 | 0.128539 | -0.189115 | 0.133558 | -0.021053 |
| 1 | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.255425 | ... | -0.225775 | -0.638672 | 0.101288 | -0.339846 | 0.167170 | 0.125895 | -0.008983 | 0.014724 |
| 2 | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.514654 | ... | 0.247998 | 0.771679 | 0.909412 | -0.689281 | -0.327642 | -0.139097 | -0.055353 | -0.059752 |
| 3 | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.387024 | ... | -0.108300 | 0.005274 | -0.190321 | -1.175575 | 0.647376 | -0.221929 | 0.062723 | 0.061458 |
| 4 | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.817739 | ... | -0.009431 | 0.798278 | -0.137458 | 0.141267 | -0.206010 | 0.502292 | 0.219422 | 0.215153 |

5 rows × 31 columns

```
[4] df.shape
```

✓ 2s completed at 9:47 PM

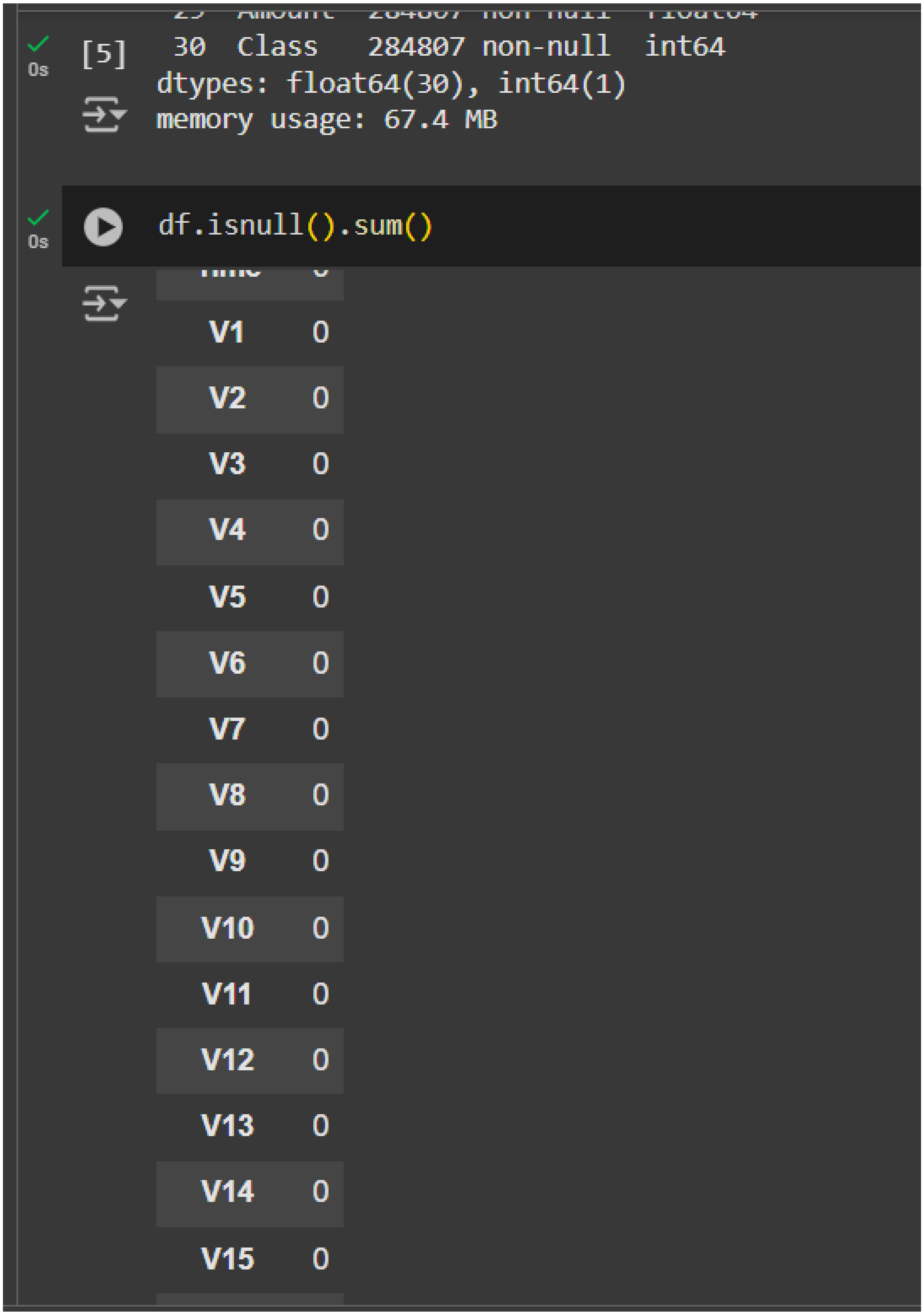


```
[4] df.shape
```

```
(284807, 31)
```

```
df.info()
```

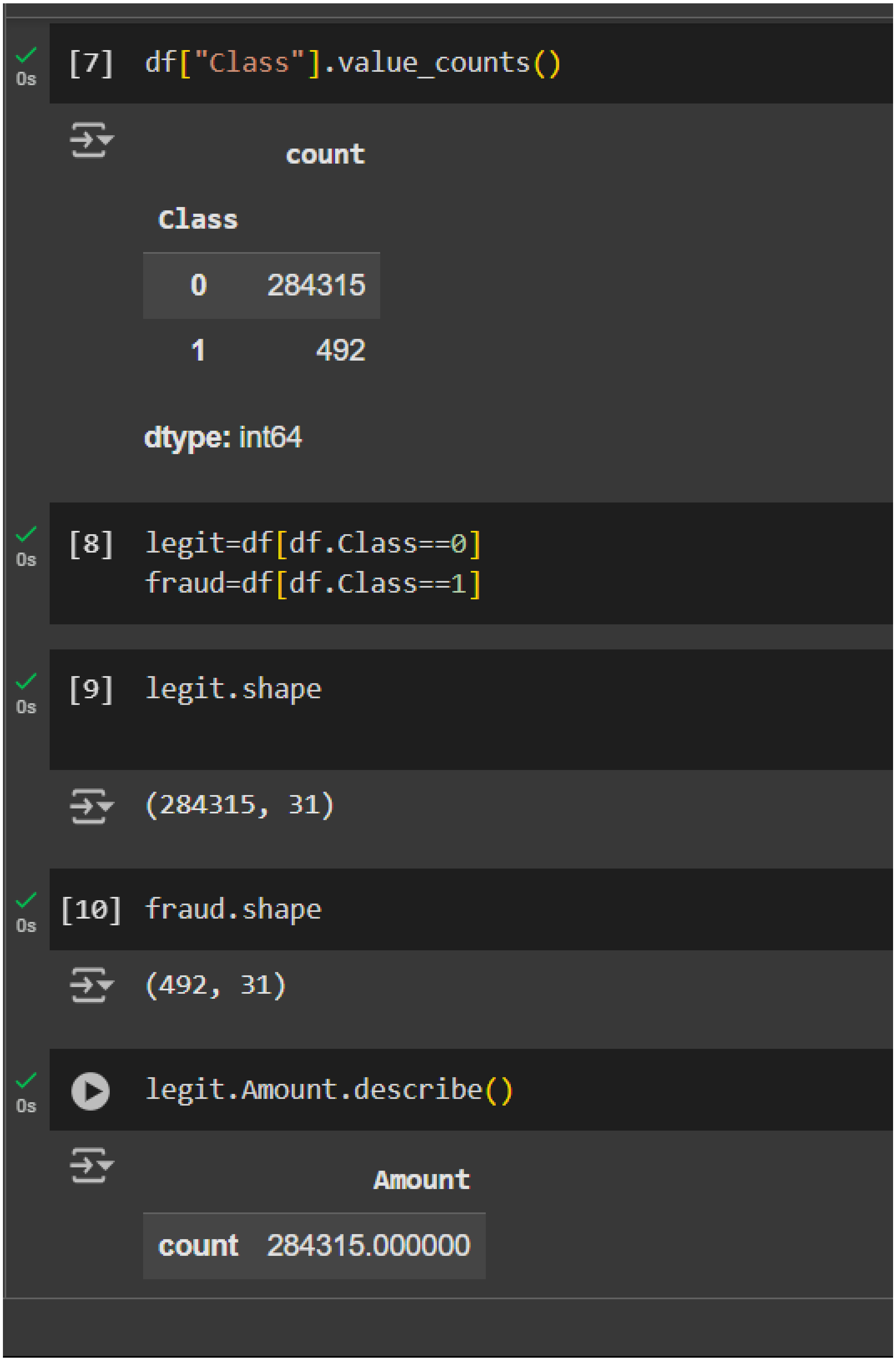
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
```



```
 29  Amount   284807 non-null  float64
 30  Class    284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

```
df.isnull().sum()
```

```
Time   0
  V1   0
  V2   0
  V3   0
  V4   0
  V5   0
  V6   0
  V7   0
  V8   0
  V9   0
 V10   0
 V11   0
 V12   0
 V13   0
 V14   0
 V15   0
```



```
[7] df["Class"].value_counts()
```

```
          count
   Class
       0  284315
       1     492

   dtype: int64
```

```
[8] legit=df[df.Class==0]
    fraud=df[df.Class==1]
```

```
[9] legit.shape
```

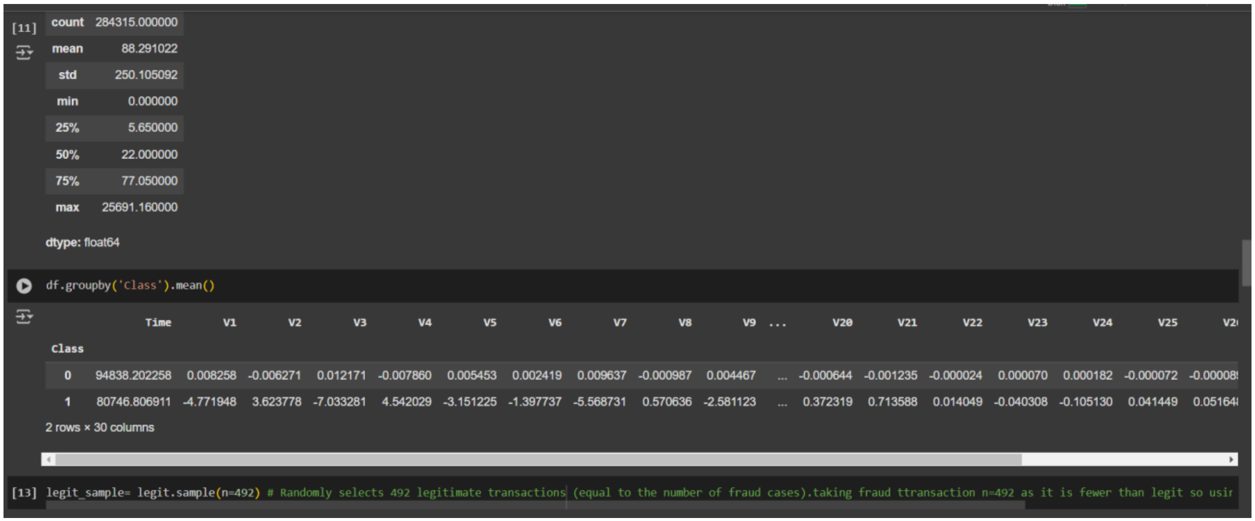
```
(284315, 31)
```

```
[10] fraud.shape
```

```
(492, 31)
```

```
legit.Amount.describe()
```

```
                   Amount
   count  284315.000000
```



[11]

| | |
|---|---|
| **count** | 284315.000000 |
| **mean** | 88.291022 |
| **std** | 250.105092 |
| **min** | 0.000000 |
| **25%** | 5.650000 |
| **50%** | 22.000000 |
| **75%** | 77.050000 |
| **max** | 25691.160000 |

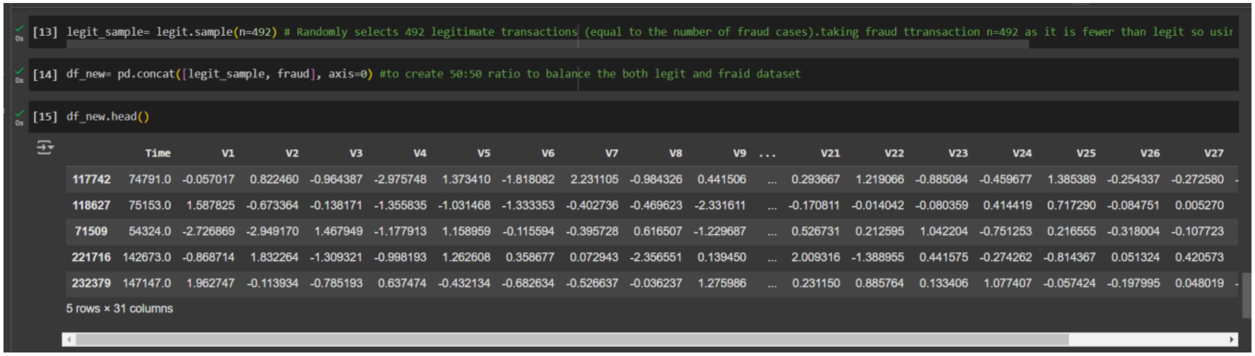
**dtype:** float64

```
df.groupby('Class').mean()
```

| Class | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | ... | V20 | V21 | V22 | V23 | V24 | V25 | V2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 94838.202258 | 0.008258 | -0.006271 | 0.012171 | -0.007860 | 0.005453 | 0.002419 | 0.009637 | -0.000987 | 0.004467 | ... | -0.000644 | -0.001235 | -0.000024 | 0.000070 | 0.000182 | -0.000072 | -0.00008 |
| **1** | 80746.806911 | -4.771948 | 3.623778 | -7.033281 | 4.542029 | -3.151225 | -1.397737 | -5.568731 | 0.570636 | -2.581123 | ... | 0.372319 | 0.713588 | 0.014049 | -0.040308 | -0.105130 | 0.041449 | 0.05164 |

2 rows × 30 columns

```
[13] legit_sample= legit.sample(n=492) # Randomly selects 492 legitimate transactions (equal to the number of fraud cases).taking fraud ttransaction n=492 as it is fewer than legit so usir
```



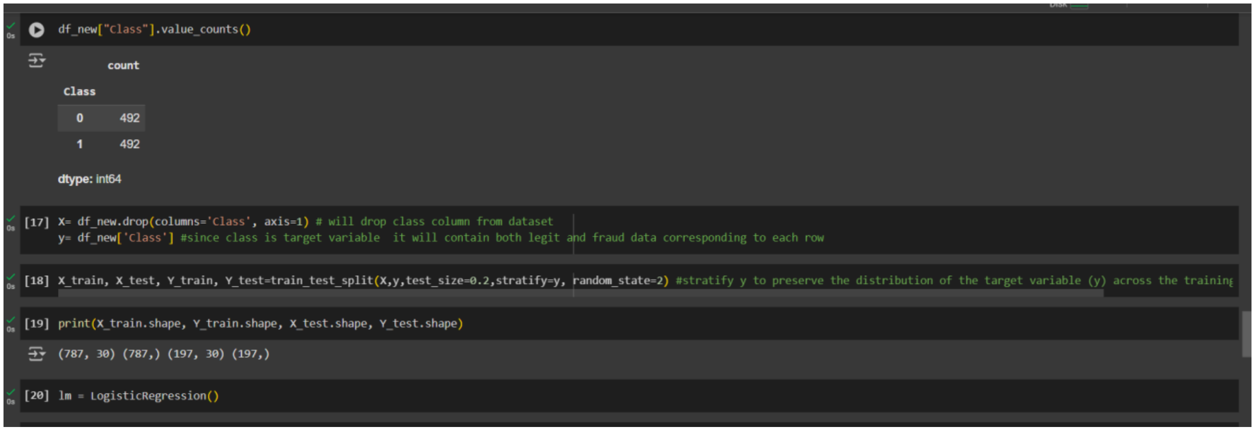
```
[13] legit_sample= legit.sample(n=492) # Randomly selects 492 legitimate transactions (equal to the number of fraud cases).taking fraud ttransaction n=492 as it is fewer than legit so usir
```

```
[14] df_new= pd.concat([legit_sample, fraud], axis=0) #to create 50:50 ratio to balance the both legit and fraid dataset
```

```
[15] df_new.head()
```

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | ... | V21 | V22 | V23 | V24 | V25 | V26 | V27 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **117742** | 74791.0 | -0.057017 | 0.822460 | -0.964387 | -2.975748 | 1.373410 | -1.818082 | 2.231105 | -0.984326 | 0.441506 | ... | 0.293667 | 1.219066 | -0.885084 | -0.459677 | 1.385389 | -0.254337 | -0.272580 |
| **118627** | 75153.0 | 1.587825 | -0.673364 | -0.138171 | -1.355835 | -1.031468 | -1.333353 | -0.402736 | -0.469623 | -2.331611 | ... | -0.170811 | -0.014042 | -0.080359 | 0.414419 | 0.717290 | -0.084751 | 0.005270 |
| **71509** | 54324.0 | -2.726869 | -2.949170 | 1.467949 | -1.177913 | 1.158959 | -0.115594 | -0.395728 | 0.616507 | -1.229687 | ... | 0.526731 | 0.212595 | 1.042204 | -0.751253 | 0.216555 | -0.318004 | -0.107723 |
| **221716** | 142673.0 | -0.868714 | 1.832264 | -1.309321 | -0.998193 | 1.262608 | 0.358677 | 0.072943 | -2.356551 | 0.139450 | ... | 2.009316 | -1.388955 | 0.441575 | -0.274262 | -0.814367 | 0.051324 | 0.420573 |
| **232379** | 147147.0 | 1.962747 | -0.113934 | -0.785193 | 0.637474 | -0.432134 | -0.682634 | -0.526637 | -0.036237 | 1.275986 | ... | 0.231150 | 0.885764 | 0.133406 | 1.077407 | -0.057424 | -0.197995 | 0.048019 |

5 rows × 31 columns



```
df_new["Class"].value_counts()
```

| Class | count |
|---|---|
| **0** | 492 |
| **1** | 492 |

**dtype:** int64

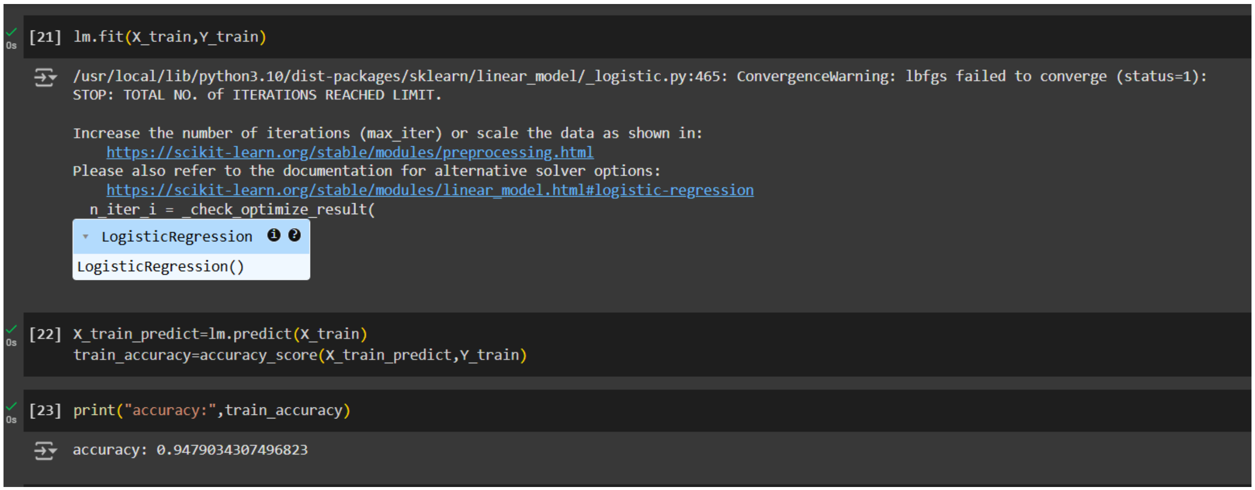
```
[17] X= df_new.drop(columns='Class', axis=1) # will drop class column from dataset
     y= df_new['Class'] #since class is target variable  it will contain both legit and fraud data corresponding to each row
```

```
[18] X_train, X_test, Y_train, Y_test=train_test_split(X,y,test_size=0.2,stratify=y, random_state=2) #stratify y to preserve the distribution of the target variable (y) across the training
```

```
[19] print(X_train.shape, Y_train.shape, X_test.shape, Y_test.shape)
```

```
(787, 30) (787,) (197, 30) (197,)
```

```
[20] lm = LogisticRegression()
```



```
lm.fit(X_train,Y_train)
```

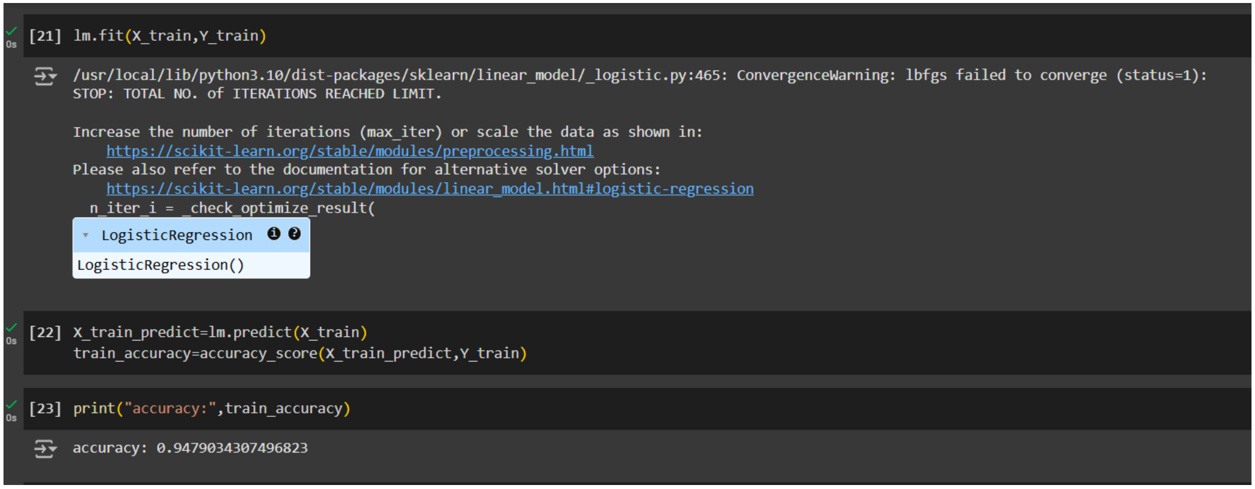
```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
X_train_predict=lm.predict(X_train)
train_accuracy=accuracy_score(X_train_predict,Y_train)
```

```
print("accuracy:",train_accuracy)
```

```
accuracy: 0.9479034307496823
```



```
lm.fit(X_train,Y_train)
```

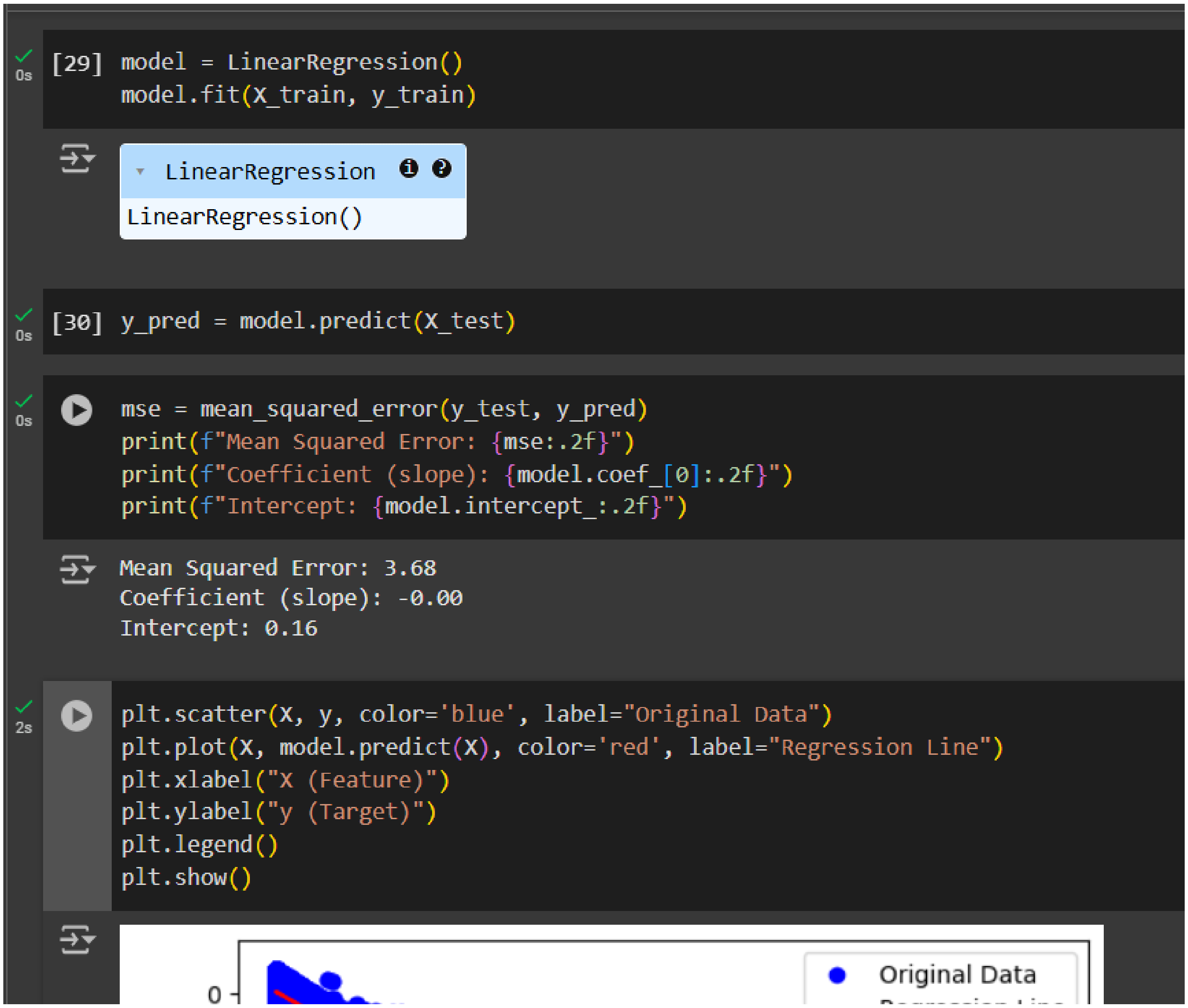
```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
X_train_predict=lm.predict(X_train)
train_accuracy=accuracy_score(X_train_predict,Y_train)
```

```
print("accuracy:",train_accuracy)
```

```
accuracy: 0.9479034307496823
```



```
[29] model = LinearRegression()
     model.fit(X_train, y_train)
```

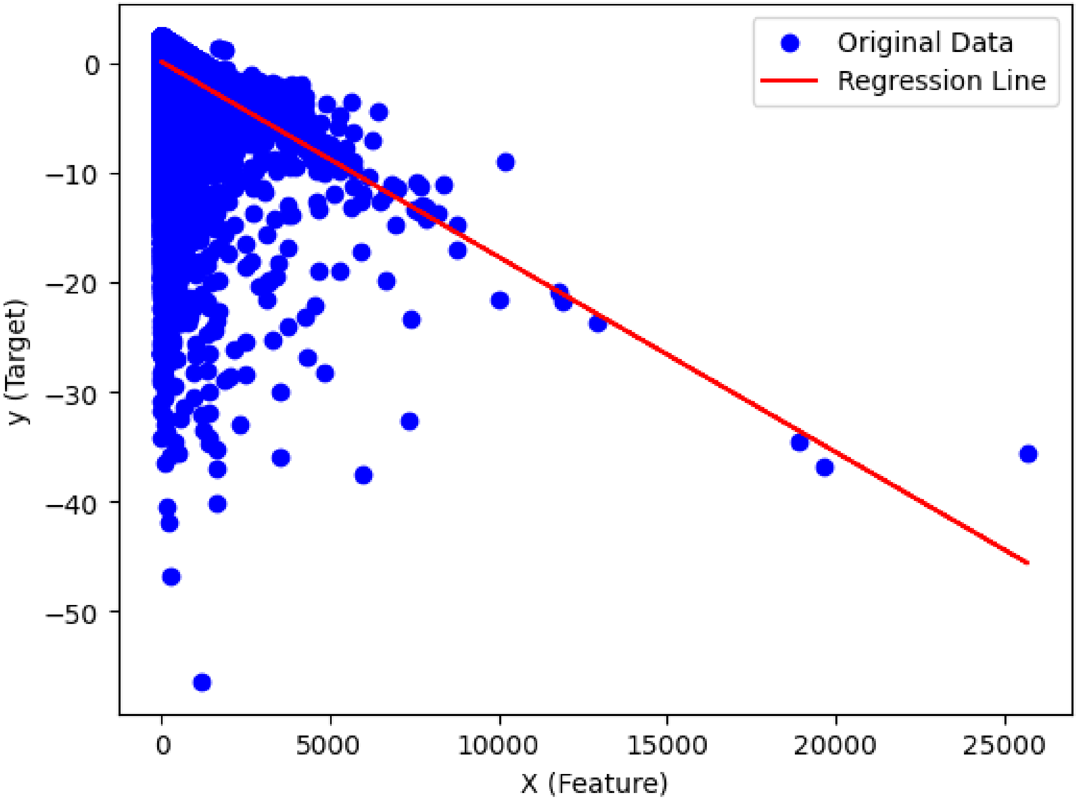
```
▾ LinearRegression
LinearRegression()
```

```
[30] y_pred = model.predict(X_test)
```

```
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse:.2f}")
print(f"Coefficient (slope): {model.coef_[0]:.2f}")
print(f"Intercept: {model.intercept_:.2f}")
```

```
Mean Squared Error: 3.68
Coefficient (slope): -0.00
Intercept: 0.16
```

```
plt.scatter(X, y, color='blue', label="Original Data")
plt.plot(X, model.predict(X), color='red', label="Regression Line")
plt.xlabel("X (Feature)")
plt.ylabel("y (Target)")
plt.legend()
plt.show()
```


Original Data
Regression Line
y (Target)
X (Feature)
0
−10
−20
−30
−40
−50
0
5000
10000
15000
20000
25000