31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

**Credit Card Fraud Detection**

In [1]:

*# Importing the Libraries :*

**import** numpy **as** np

**import** pandas **as** pd

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.metrics **import** accuracy\_score

# numpy = use to make arrays

# pandas = use to make data frames or structures for data analysis # The next codes are for :

      splitting the data set into train set and test set ;       I'm gonna use Logistics Regression this analysis for accuracy ;       accuracy score will help us to check the performance of our model

In [3]:

*#Import the Data Set :*

df**=**pd.read\_csv("/Users/ranitaghosh/Desktop/my folders/Project/creditcard.csv") print("The data is read successfully")

The data is read successfully

In [12]:

*# to see the number of rows & columns :*

df.shape

Out[12]:

(284807, 31)

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 1/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

In [4]:

*# to see the first 5 rows of the data set :*

df.head()

Out[4]:

**Time V1 V2 V3 V4 V5 V6 V7 V8 0** 0.0 -1.359807 -0.072781 2.536347 1.378155 -0.338321 0.462388 0.239599 0.098698 **1** 0.0 1.191857 0.266151 0.166480 0.448154 0.060018 -0.082361 -0.078803 0.085102 - **2** 1.0 -1.358354 -1.340163 1.773209 0.379780 -0.503198 1.800499 0.791461 0.247676 - **3** 1.0 -0.966272 -0.185226 1.792993 -0.863291 -0.010309 1.247203 0.237609 0.377436 - **4** 2.0 -1.158233 0.877737 1.548718 0.403034 -0.407193 0.095921 0.592941 -0.270533

5 rows × 31 columns

In [5]:

*# to see the last 5 rows of the data set :*

df.tail()

Out[5]:

**Time V1 V2 V3 V4 V5 V6 V7 284802** 172786.0 -11.881118 10.071785 -9.834783 -2.066656 -5.364473 -2.606837 -4.918215 **284803** 172787.0 -0.732789 -0.055080 2.035030 -0.738589 0.868229 1.058415 0.024330 **284804** 172788.0 1.919565 -0.301254 -3.249640 -0.557828 2.630515 3.031260 -0.296827 **284805** 172788.0 -0.240440 0.530483 0.702510 0.689799 -0.377961 0.623708 -0.686180 **284806** 172792.0 -0.533413 -0.189733 0.703337 -0.506271 -0.012546 -0.649617 1.577006 -

5 rows × 31 columns

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 2/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

In [6]:

*# to see some information of the data set :* 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 23 V23 284807 non-null float64 24 V24 284807 non-null float64 25 V25 284807 non-null float64 26 V26 284807 non-null float64 27 V27 284807 non-null float64 28 V28 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

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 3/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

In [7]:

*# checking the number of missing values in each column :*

df.isnull().sum()

Out[7]:

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

V16 0

V17 0

V18 0

V19 0

V20 0

V21 0

V22 0

V23 0

V24 0

V25 0

V26 0

V27 0

V28 0

Amount 0

Class 0

dtype: int64

In [13]:

*# to see the distribution of legit transaction and fraudulent transaction :*

df["Class"].value\_counts()

Out[13]:

0 284315

1 492

Name: Class, dtype: int64

# Here, Label = 0 represents legit transaction ; Label = 1 represents fraudulent transaction.

# There are two target variables.

# But we can see that, over 90% of the data points is in legit transaction or in one particular target variable.

# So, we can't fit the data set into Machine Learing model.

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 4/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

# If I train my ML model with this data, it won't recognise the fraud transaction cause it has very less data points.

# So whenever I put a new data, it will predict legit transaction which is not the case.

**Therefore, this data set is highly unbalanced.**

In [16]:

*# Separate all the variables (ie. legit & fraud transaction) from this data set :*

legit**=**df[df.Class **==** 0] *# storing all the legit transcation in this particular varia* fraud**=**df[df.Class **==** 1] *# storing all the fraudulent transcation in this particular*

# Data Frame name = df;

In the data frame df , we have Class column;

So, in the class column of value 0, all the values in that paticular row will be stored in legit;

Same goes for fraud.

In [17]:

*# rows & columns of legit & fraud :*

print(legit.shape)

print(fraud.shape)

(284315, 31)

(492, 31)

**Statistical Measurements of the Data :**

In [19]:

*# for legit transaction :*

legit.Amount.describe()

Out[19]:

count 284315.000000

mean 88.291022

std 250.105092

min 0.000000

25% 5.650000

50% 22.000000

75% 77.050000

max 25691.160000

Name: Amount, dtype: float64

# found the measurements of Amount column in which the money has been transacted in that particular transaction.

# there it shows mean , standard deviation , max value , min value , the number of data points we have which is count.

# 25 percentile doesn't mean 25 percentage ;

      # 25% = 25 percentage of transaction amount is less than 5.650000       # same for 50% and 75%

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 5/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

In [20]:

*# for fraudulent transaction :*

fraud.Amount.describe()

Out[20]:

count 492.000000

mean 122.211321

std 256.683288

min 0.000000

25% 1.000000

50% 9.250000

75% 105.890000

max 2125.870000

Name: Amount, dtype: float64

In [21]:

*# compare the measurements for both transactions :*

df.groupby("Class").mean()

Out[21]:

**Time V1 V2 V3 V4 V5 V6 V7**

**Class**

**0** 94838.202258 0.008258 -0.006271 0.012171 -0.007860 0.005453 0.002419 0.009637 **1** 80746.806911 -4.771948 3.623778 -7.033281 4.542029 -3.151225 -1.397737 -5.568731

2 rows × 30 columns

We can compare all the rest measurements. But here, I compare only mean values because there is a high difference

between legit mean and fraud mean. And this difference is very important and this is how ML algorithm can predict whether a transaction is either legit or fraudulent transaction.

**Dealing with this unbalanced Data : Under-Sampling**

In [23]:

*# Build a sample data set containing similar distribution of legit & fraud transacti # We'll take 492 transactions in legit variable ; which will be same as fraud.*

legit\_sample **=** legit.sample(n**=**492)

In [27]:

*# Now concatenating legit & fraud data frames :*

df\_sample **=** pd.concat([legit\_sample,fraud],axis**=**0)

        #axis = 0 means data frames will be added below "legit\_sample" one by one row wise (axis = 1 means column wise).

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 6/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

In [35]:

df\_sample.head()

Out[35]:

**Time V1 V2 V3 V4 V5 V6 V7**

**151388** 95480.0 -3.657405 0.587003 -0.077778 -0.108495 0.616579 -0.088736 0.104411 0.41 **71742** 54426.0 -1.271430 0.458375 2.167371 -1.477542 -1.750212 0.046554 0.096646 -0.13 **71662** 54389.0 -0.740383 1.259907 2.101816 0.880800 0.129886 -0.933158 1.114343 -0.52

**112815** 72817.0 -1.146653 0.408907 1.582569 0.200762 0.991270 -0.392608 0.410024 0.15 **50249** 44409.0 1.135426 0.608113 0.205817 2.598191 -0.041364 -1.034539 0.586538 -0.28

5 rows × 31 columns

In [37]:

df\_sample.tail()

Out[37]:

**Time V1 V2 V3 V4 V5 V6 V7**

**279863** 169142.0 -1.927883 1.125653 -4.518331 1.749293 -1.566487 -2.010494 -0.882850 0.6 **280143** 169347.0 1.378559 1.289381 -5.004247 1.411850 0.442581 -1.326536 -1.413170 0.2 **280149** 169351.0 -0.676143 1.126366 -2.213700 0.468308 -1.120541 -0.003346 -2.234739 1.2 **281144** 169966.0 -3.113832 0.585864 -5.399730 1.817092 -0.840618 -2.943548 -2.208002 1.0 **281674** 170348.0 1.991976 0.158476 -2.583441 0.408670 1.151147 -0.096695 0.223050 -0.0

5 rows × 31 columns

So, all the samples are randomly picked ; we can say this by seeing the serial number.

In [31]:

*# checking if the distribution of this sample data set :*

df\_sample["Class"].value\_counts()

Out[31]:

1 492

0 492

Name: Class, dtype: int64

                # So now we have uniformly distrubuted data with 492 legit and 492 fraudulent transaction.

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 7/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

In [38]:

*# again compare mean values on new data set :*

df\_sample.groupby("Class").mean()

Out[38]:

**Time V1 V2 V3 V4 V5 V6 V7**

**Class**

**0** 95372.867886 -0.049723 0.131796 -0.068673 0.019697 0.025820 0.071185 -0.031394 - **1** 80746.806911 -4.771948 3.623778 -7.033281 4.542029 -3.151225 -1.397737 -5.568731

2 rows × 30 columns

  # when compare the population mean it had high difference in mean values between legit & fraud transactions ;

  # and now that we compare sample means ; it also shows high difference in mean values.

  # So, the nature of the data set is not changed; the difference is there   # It helps our ML model to detect correctly whether a transaction is either legit or fraud;

  # So it is a good sample; otherwise it would be bad and we have to pick another random sample

In [40]:

*# Splitting the data set in Features(X) and target(Y) :*

X **=** df\_sample.drop(columns**=**"Class",axis **=** 1) *# for dropping the colum "Class" becau* Y **=** df\_sample["Class"]

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 8/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

In [41]:

print(X)

Time V1 V2 V3 V4 V5 V6 \

151388 95480.0 -3.657405 0.587003 -0.077778 -0.108495 0.616579 -0. 088736

71742 54426.0 -1.271430 0.458375 2.167371 -1.477542 -1.750212 0. 046554

71662 54389.0 -0.740383 1.259907 2.101816 0.880800 0.129886 -0. 933158

112815 72817.0 -1.146653 0.408907 1.582569 0.200762 0.991270 -0. 392608

50249 44409.0 1.135426 0.608113 0.205817 2.598191 -0.041364 -1. 034539

... ... ... ... ... ... ... ...

279863 169142.0 -1.927883 1.125653 -4.518331 1.749293 -1.566487 -2. 010494

280143 169347.0 1.378559 1.289381 -5.004247 1.411850 0.442581 -1. 326536

280149 169351.0 -0.676143 1.126366 -2.213700 0.468308 -1.120541 -0. 003346

281144 169966.0 -3.113832 0.585864 -5.399730 1.817092 -0.840618 -2. 943548

281674 170348.0 1.991976 0.158476 -2.583441 0.408670 1.151147 -0. 096695

V7 V8 V9 ... V20 V21 V2 2 \

151388 0.104411 0.413151 2.657495 ... -0.728626 -0.943409 -0.64438 4

71742 0.096646 -0.139487 -0.977292 ... -0.088383 0.533227 1.27294 4

71662 1.114343 -0.524002 -0.758370 ... 0.116045 0.109028 0.47786 9

112815 0.410024 0.150814 -0.358890 ... -0.344274 0.068177 0.32884 6

50249 0.586538 -0.284289 -1.001115 ... -0.136601 0.022452 -0.06482 4

... ... ... ... ... ... ... ...

279863 -0.882850 0.697211 -2.064945 ... 1.252967 0.778584 -0.31918 9

280143 -1.413170 0.248525 -1.127396 ... 0.226138 0.370612 0.02823 4

280149 -2.234739 1.210158 -0.652250 ... 0.247968 0.751826 0.83410 8

281144 -2.208002 1.058733 -1.632333 ... 0.306271 0.583276 -0.26920 9

281674 0.223050 -0.068384 0.577829 ... -0.017652 -0.164350 -0.29513 5

V23 V24 V25 V26 V27 V28 Am ount

151388 0.952668 0.657374 0.449978 0.095636 0.667097 0.520954 0.01

71742 -0.360121 0.572534 0.201054 -0.115470 -0.655651 0.036225 11 9.96

71662 -0.310764 0.964034 0.440585 -0.283772 -0.352498 -0.137751

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 9/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

3.50

112815 -0.254316 0.237920 -0.062435 -0.603473 0.071375 0.108723 1.00

50249 -0.090029 0.697745 0.654846 0.046429 -0.049603 0.018543 3 9.48

... ... ... ... ... ... ... ...

279863 0.639419 -0.294885 0.537503 0.788395 0.292680 0.147968 39 0.00

280143 -0.145640 -0.081049 0.521875 0.739467 0.389152 0.186637 0.76

280149 0.190944 0.032070 -0.739695 0.471111 0.385107 0.194361 7 7.89

281144 -0.456108 -0.183659 -0.328168 0.606116 0.884876 -0.253700 24 5.00

281674 -0.072173 -0.450261 0.313267 -0.289617 0.002988 -0.015309 4 2.53

[984 rows x 30 columns]

In [42]:

print(Y)

151388 0

71742 0

71662 0

112815 0

50249 0

..

279863 1

280143 1

280149 1

281144 1

281674 1

Name: Class, Length: 984, dtype: int64

In [47]:

*# now splitting the data into training data and test data :*

X\_train,X\_test,Y\_train,Y\_test**=** train\_test\_split(X,Y, test\_size**=**0.2,stratify**=**Y,random

#the splitting will be random

#test\_size = amount of testing data we want;

#test\_size = 0.2 means 20 % of the data go to test data and 80% of the data go to training data.

80% of the data will be stored in X\_train and the correspoding labels(0 & 1) will be stored in Y\_train ;

20% of the data will be stored in Y\_train and the correspoding labels will be stored in Y\_train

stratify = Y ; I want to statisfy this data based on Y ; If I dont mention Y the distribution of 0 & 1 can be very

  different in the training data and test data.

  when I say stratify = Y ; it means there will be a evenly distributed into two classiers in both X\_train & X\_test

# random\_state = 2 ; split in one manner

random\_state = 3 ; split in another manner ; it is used for reproduce our code as it is

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 10/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

In [45]:

print(X.shape , X\_train.shape , X\_test.shape)

(984, 30) (787, 30) (197, 30)

In [46]:

print(Y.shape , Y\_train.shape , Y\_test.shape)

(984,) (787,) (197,)

In [48]:

*# Model Training : Logistics Regression (genarally we use logistic regression for bi* model **=** LogisticRegression() *# we are one instance of this logistic regression model*

In [50]:

*# training the logistic regression model with training data*

model.fit(X\_train,Y\_train)

Out[50]:

LogisticRegression()

In [ ]:

*# Model Evaluation based on Accuracy score*

In [54]:

*# Accuracy on training data :*

X\_train\_prediction **=** model.predict(X\_train)

training\_data\_accuracy **=** accuracy\_score(X\_train\_prediction,Y\_train) print("The Accuracy on Training Data :",training\_data\_accuracy)

The Accuracy on Training Data : 0.9237611181702668

Over 75% or 80% accuracy is good accuracy

Accuracy means out of 100 predictions , our model can predit correctly for 92.3 predictions

In [55]:

*# Accuracy on test data :*

X\_test\_prediction **=** model.predict(X\_test)

test\_data\_accuracy **=** accuracy\_score(X\_test\_prediction,Y\_test)

print("The Accuracy on Test Data :",test\_data\_accuracy)

The Accuracy on Test Data : 0.9137055837563451

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 11/12

31/08/2021 Project\_Credit Card Fraud Detection - Jupyter Notebook

In [ ]:

*# We can see the accuracy of training data and test data is very similar ; # If we see that training accuracy is highly larger than test accuracy then ou #If we see that training accuracy is highly smaller than test accuracy then ou*

In [ ]:

Example : A student **is** practicing all the maths problems **in** a book **=** X\_train ; the teacher put another equation to solve **in** exam which **is not in** that boo **is** based on how he answers **in** exam; So it **is =** X\_test

In this case , evaluate the model **with** the data which it hasn't seen befor

Over**-**fitted example : A student **is** practising only previous year question paper , s this year's examination if he's been asked questions **from** pr different questions he can't perform well.

localhost:8888/notebooks/Desktop/Project\_Credit Card Fraud Detection.ipynb 12/12