https://www.kaggle.com/mlg-ulb/creditcardfraud?select=creditcard.csv

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
#import missingno
```

In [2]:
```
dt = pd.read_csv('Credit_Card_Data.csv')
dt
```

Out[2]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V2
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.01830
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.22577
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.24799
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.10830
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.00943
...	...	...	...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.21345
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.21420
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.23204
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.26524
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.26105

284807 rows × 31 columns

In [3]:
```
dt.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
```

In [4]:
```
dt.columns
```

Out[4]:
```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

हम float data को integer data में convert करेंगे |

In [5]:
```
columns = ['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class']

for x in columns:
    dt[x] = dt[x].astype('int')


dt.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  int32
 1   V1      284807 non-null  int32
 2   V2      284807 non-null  int32
 3   V3      284807 non-null  int32
 4   V4      284807 non-null  int32
 5   V5      284807 non-null  int32
 6   V6      284807 non-null  int32
 7   V7      284807 non-null  int32
 8   V8      284807 non-null  int32
 9   V9      284807 non-null  int32
 10  V10     284807 non-null  int32
 11  V11     284807 non-null  int32
 12  V12     284807 non-null  int32
 13  V13     284807 non-null  int32
 14  V14     284807 non-null  int32
 15  V15     284807 non-null  int32
 16  V16     284807 non-null  int32
 17  V17     284807 non-null  int32
 18  V18     284807 non-null  int32
 19  V19     284807 non-null  int32
 20  V20     284807 non-null  int32
 21  V21     284807 non-null  int32
 22  V22     284807 non-null  int32
 23  V23     284807 non-null  int32
 24  V24     284807 non-null  int32
 25  V25     284807 non-null  int32
 26  V26     284807 non-null  int32
 27  V27     284807 non-null  int32
 28  V28     284807 non-null  int32
 29  Amount  284807 non-null  int32
 30  Class   284807 non-null  int32
dtypes: int32(31)
memory usage: 33.7 MB
```

In [6]:
```
dt
```

Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0	-1	0	2	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	149	0
1	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	2	0
2	1	-1	-1	1	0	0	1	0	0	-1	...	0	0	0	0	0	0	0	0	378	0
3	1	0	0	1	0	0	1	0	0	-1	...	0	0	0	-1	0	0	0	0	123	0
4	2	-1	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	69	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
284802	172786	-11	10	-9	-2	-5	-2	-4	7	1	...	0	0	1	0	1	0	0	0	0	0
284803	172787	0	0	2	0	0	1	0	0	0	...	0	0	0	-1	0	0	0	0	24	0
284804	172788	1	0	-3	0	2	3	0	0	0	...	0	0	0	0	0	0	0	0	67	0
284805	172788	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	10	0
284806	172792	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	217	0

284807 rows × 31 columns

dt में जो class है उसमें 1 means fraud & 0 means normal

In [7]:
```
dt['Class'].value_counts()
```

Out[7]:
```
0    284315
1       492
Name: Class, dtype: int64
```

In [8]:
```
fraud = dt[dt.Class == 1]
legit = dt[dt.Class == 0]
```

In [9]:
```
print(fraud.shape)
print(legit.shape)
```

```
(492, 31)
(284315, 31)
```

In [10]:
```
legit.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 284315 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284315 non-null  int32
 1   V1      284315 non-null  int32
 2   V2      284315 non-null  int32
 3   V3      284315 non-null  int32
 4   V4      284315 non-null  int32
 5   V5      284315 non-null  int32
 6   V6      284315 non-null  int32
 7   V7      284315 non-null  int32
 8   V8      284315 non-null  int32
 9   V9      284315 non-null  int32
 10  V10     284315 non-null  int32
 11  V11     284315 non-null  int32
 12  V12     284315 non-null  int32
 13  V13     284315 non-null  int32
 14  V14     284315 non-null  int32
 15  V15     284315 non-null  int32
 16  V16     284315 non-null  int32
 17  V17     284315 non-null  int32
 18  V18     284315 non-null  int32
 19  V19     284315 non-null  int32
 20  V20     284315 non-null  int32
 21  V21     284315 non-null  int32
 22  V22     284315 non-null  int32
 23  V23     284315 non-null  int32
 24  V24     284315 non-null  int32
 25  V25     284315 non-null  int32
 26  V26     284315 non-null  int32
 27  V27     284315 non-null  int32
 28  V28     284315 non-null  int32
 29  Amount  284315 non-null  int32
 30  Class   284315 non-null  int32
dtypes: int32(31)
memory usage: 35.8 MB
```

In [11]:
```
fraud.Amount.describe()
```

Out[11]:
```
count     492.000000
mean      121.835366
std       256.622833
min         0.000000
25%         1.000000
50%         9.000000
75%       105.000000
max      2125.000000
Name: Amount, dtype: float64
```

In [12]:
```
legit.Amount.describe()
```

Out[12]:
```
count    284315.000000
mean         87.810875
std         250.133800
min           0.000000
25%           5.000000
50%          22.000000
75%          77.000000
max       25691.000000
```

```
max       25691.160000
Name: Amount, dtype: float64
```

Fraud और legit transaction को Compair करेंगे ।

In [13]:
```
dt.groupby('Class').mean()
```

Out[13]:

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V
0	94838.202258	0.060560	-0.033878	-0.035802	0.020692	0.027353	0.123715	-0.007390	-0.049322	0.024733	...	0.0287
1	80746.806911	-4.479675	3.247967	-6.593496	4.099593	-2.943089	-1.073171	-5.219512	0.378049	-2.186992	...	0.2012

2 rows × 30 columns

In [14]:
```
legit.Amount.count()
```

Out[14]:
```
284315
```

अगर हम किसी Model को बहुत सारे data से train करेंगे तो वो Overfitting हो जाए गा ।

इसलिए हम legit में से 492 sample लेंगे ।
because fraud transaction bhi 492 hai

In [15]:
```
#sample data

legit_sample = legit.sample(n=492)
```

In [16]:
```
n_data = pd.concat([legit_sample, fraud], axis = 0)
```

In [17]:
```
legit_sample.shape
```

Out[17]:
```
(492, 31)
```

In [18]:
```
fraud.shape
```

Out[18]:
```
(492, 31)
```

In [19]:
```
n_data
```

Out[19]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
44440	41944	1	0	0	0	-1	0	0	0	0	...	0	-1	0	0	0	0	0	0	31	0
36271	38468	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	9	0
235896	148582	0	0	0	0	0	0	0	0	0	...	0	0	0	-1	0	0	0	0	9	0
258096	158489	-3	2	1	0	0	0	-1	-4	1	...	4	-2	0	0	0	0	0	0	5	0
129226	79001	1	0	0	0	0	-1	0	0	0	...	0	-1	0	0	0	0	0	0	59	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
279863	169142	-1	1	-4	1	-1	-2	0	0	-2	...	0	0	0	0	0	0	0	0	390	1
280143	169347	1	1	-5	1	0	-1	-1	0	-1	...	0	0	0	0	0	0	0	0	0	1
280149	169351	0	1	-2	0	-1	0	-2	1	0	...	0	0	0	0	0	0	0	0	77	1
281144	169966	-3	0	-5	1	0	-2	-2	1	-1	...	0	0	0	0	0	0	0	0	245	1
281674	170348	1	0	-2	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	42	1

984 rows × 31 columns

Now we are going to split data into Training and Testing

In [20]:
```
x = n_data.iloc[: , :30]
y = n_data['Class']
y
```

Out[20]:
```
44440     0
36271     0
235896    0
258096    0
129226    0
         ..
279863    1
280143    1
```

```
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int32
```

In [21]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify = y, random_state = 2
```

In [22]:
```
print(x.shape, x_train.shape, x_test.shape)
```

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

using Logistic Regression

In [23]:
```
from sklearn.linear_model import LogisticRegression
```

In [24]:
```
model = LogisticRegression()
```

In [25]:
```
model.fit(x_train, y_train)
```

Out[25]:
```
LogisticRegression()
```

Acuracy on Test Data

In [26]:
```
model.score(x_test, y_test)
```

Out[26]:
```
0.9238578680203046
```

In [27]:
```
model.score(x_train, y_train)
```

Out[27]:
```
0.9351969504447268
```

हमारा model overfitted नहीं है ।

In [ ]:

---------- जब test data का score Train Data के score se ज्यादा आए तो वो Model Underfitting होगा ।

In [ ]:

------- जब Train data का score Test Data के score se ज्यादा आए तो वो Model Overfitting होगा ।

In [ ]:

In [ ]:

In [ ]:

----------Thank u So much ----------

7 Nov 2021

Urkarsh Tripathi

In [ ]: