In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data = pd.read_csv("C:\\Users\\amang\\Downloads\\archive (1).zip")
data.head()
```

Out[2]:

	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	0.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.

5 rows × 31 columns

In [3]:
```
data.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]:
```
data["Class"].unique()
```

Out[4]:
```
array([0, 1], dtype=int64)
```

In [6]:
```
data.describe()
```

Out[6]:

	Time	V1	V2	V3	V4	V5
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01

8 rows × 31 columns

In [7]:
```
data.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 [8]:
```
data["Class"].value_counts()
```

Out[8]:
```
0    284315
1       492
Name: Class, dtype: int64
```

In [9]:
```
data.groupby("Class").mean()
```

Out[9]:

Class	Time	V1	V2	V3	V4	V5	V6	V7
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

In [10]:
```
fraud = data[data.Class ==1]
valid = data[data.Class ==0]
```

In [12]:
```
print(fraud.shape)
print(valid.shape)
```

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

Creating a sample for the valid as it is the large dataset

In [13]:
```
valid_new = valid.sample(n=500)
```

In [14]:
```
valid_new.head()
```

Out[14]:

	Time	V1	V2	V3	V4	V5	V6	V7	
41453	40685.0	1.344888	-0.746532	0.001362	-2.673117	-1.172976	-1.572194	-0.182726	-0.3
153189	98140.0	-2.678774	1.746471	-0.556548	-0.569623	-1.417239	-0.300425	-1.210458	1.7
154834	103222.0	2.181700	0.265640	-2.218658	0.139964	1.133575	-0.808198	0.627185	-0.5
83996	60120.0	-3.679631	-0.450747	-1.458111	-3.979915	1.870932	2.778886	-0.217890	0.1
186969	127330.0	2.319370	-0.590469	-1.836586	-1.142508	0.004764	-0.979170	-0.157148	-0.3

5 rows × 31 columns

In [16]:
```
new_data = pd.concat([fraud,valid_new])
```

In [17]:
```
new_data.head()
```

Out[17]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
541	406.0	-2.312227	1.951992	-1.609851	3.997906	-0.522188	-1.426545	-2.537387	1.391657
623	472.0	-3.043541	-3.157307	1.088463	2.288644	1.359805	-1.064823	0.325574	-0.067794
4920	4462.0	-2.303350	1.759247	-0.359745	2.330243	-0.821628	-0.075788	0.562320	-0.399147
6108	6986.0	-4.397974	1.358367	-2.592844	2.679787	-1.128131	-1.706536	-3.496197	-0.248778
6329	7519.0	1.234235	3.019740	-4.304597	4.732795	3.624201	-1.357746	1.713445	-0.496358

5 rows × 31 columns

In [21]:
```
new_data.describe()
```

Out[21]:

	Time	V1	V2	V3	V4	V5	V6	
count	992.000000	992.000000	992.000000	992.000000	992.000000	992.000000	992.000000	9
mean	87721.143145	-2.319080	1.753482	-3.483725	2.248219	-1.530712	-0.723636	
std	48226.092709	5.613462	3.884146	6.197478	3.218437	4.223925	1.727042	
min	406.000000	-36.510583	-40.938048	-31.103685	-4.125352	-22.105532	-6.406267	-
25%	45408.250000	-2.762972	-0.235112	-5.074851	-0.207048	-1.758911	-1.576229	
50%	81368.500000	-0.733893	0.962931	-1.343495	1.240099	-0.447008	-0.641159	
75%	133787.750000	1.089259	2.735735	0.259408	4.229802	0.480477	0.029203	
max	171926.000000	2.368845	22.057729	3.165402	12.114672	11.095089	6.474115	

8 rows × 31 columns

In [22]:
```
new_data.groupby("Class").mean()
```

Out[22]:

	Time	V1	V2	V3	V4	V5	V6	V7	
Class									
0	94583.890000	0.094543	-0.086889	0.009038	-0.008890	0.063873	-0.060322	0.017825	-
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	

2 rows × 30 columns

In [19]:
```
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score, mean_absolute_error
```

In [20]:
```
X = new_data.drop(["Class"],axis=1)
y = new_data["Class"]
```

In [23]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_stat
```

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

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
814: 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 (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

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

In [26]:
```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)
print(mse)
print(rmse)
```

```
0.07537688442211055
0.27454851014367304
```

In [28]:
```
y_prediction = model.predict(X_test)
Accuracy = accuracy_score(y_test,y_prediction)
print("Accuracy:",Accuracy)
```

```
Accuracy: 0.9246231155778895
```