Importing the Dependencies

```
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
```

```
# loading the dataset to a Pandas Dataframe
ccdp = pd.read_csv('/content/CreditCardDefault.csv')
```

```
# first 5 rows of the dataset
ccdp.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.09
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.08
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.24
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.37
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.27

5 rows × 31 columns

```
ccdp.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
5843	6339	-2.262193	2.436048	-1.230114	-1.257178	1.999487	3.164989	-0.326910	
5844	6340	-1.134217	0.166310	1.306848	1.667260	-0.570757	0.814229	2.103197	-
5845	6345	-0.865862	0.295617	3.940337	3.606141	-0.672490	1.242731	-0.897963	
5846	6345	1.203971	0.927268	0.041463	1.669881	-0.007861	-1.477041	0.300909	-
5847	6347	1.122792	-0.064138	1.035259	1.775286	-0.649522	0.282818	-0.613878	

5 rows × 31 columns

```
# dataset information
ccdp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5848 entries, 0 to 5847
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Time    5848 non-null   int64
 1   V1      5848 non-null   float64
 2   V2      5848 non-null   float64
 3   V3      5848 non-null   float64
 4   V4      5848 non-null   float64
 5   V5      5848 non-null   float64
 6   V6      5848 non-null   float64
 7   V7      5848 non-null   float64
 8   V8      5848 non-null   float64
 9   V9      5848 non-null   float64
 10  V10     5848 non-null   float64
 11  V11     5848 non-null   float64
 12  V12     5848 non-null   float64
 13  V13     5848 non-null   float64
 14  V14     5848 non-null   float64
 15  V15     5848 non-null   float64
 16  V16     5848 non-null   float64
 17  V17     5848 non-null   float64
 18  V18     5848 non-null   float64
 19  V19     5848 non-null   float64
 20  V20     5848 non-null   float64
 21  V21     5848 non-null   float64
 22  V22     5848 non-null   float64
 23  V23     5848 non-null   float64
 24  V24     5848 non-null   float64
 25  V25     5848 non-null   float64
 26  V26     5847 non-null   float64
 27  V27     5847 non-null   float64
 28  V28     5847 non-null   float64
 29  Amount  5847 non-null   float64
 30  Class   5847 non-null   float64
dtypes: float64(30), int64(1)
memory usage: 1.4 MB
```

```
# checking the number of missing value in each column
ccdp.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
```

```
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       1
V27       1
V28       1
Amount    1
Class     1
dtype: int64
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
# anlalyse the distribution of data in the column from V21 to V28
fig, ax = plt.subplots(figsize=(8,8))
sns.distplot(ccdp.V20)
```

```
ccdp['V20'].fillna(ccdp['V20'].mean(),inplace=True)
```

```
ccdp.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
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       1
V27       1
V28       1
Amount    1
Class     1
dtype: int64
```

```
sns.distplot(ccdp.V21)
ccdp['V21'].fillna(ccdp['V21'].mean(),inplace=True)
ccdp.isnull().sum()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
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       1
V27       1
V28       1
Amount    1
Class     1
dtype: int64
```


```
sns.distplot(ccdp.V22)
ccdp['V22'].fillna(ccdp['V22'].mean(),inplace=True)
ccdp.isnull().sum()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
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       1
V27       1
V28       1
Amount    1
```

```
sns.distplot(ccdp.V23)
ccdp['V23'].fillna(ccdp['V23'].mean(),inplace=True)
ccdp.isnull().sum()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
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
```

```
sns.distplot(ccdp.V24)
ccdp['V24'].fillna(ccdp['V24'].mode(),inplace=True)
ccdp.isnull().sum()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
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
```

```
sns.distplot(ccdp.V25)
ccdp['V25'].fillna(ccdp['V25'].mean(),inplace=True)
ccdp.isnull().sum()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
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
```

```
sns.distplot(ccdp.V26)
ccdp['V26'].fillna(ccdp['V26'].mode(),inplace=True)
ccdp.isnull().sum()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
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
```

```
sns.distplot(ccdp.V27)
ccdp['V27'].fillna(ccdp['V27'].mean(),inplace=True)
ccdp.isnull().sum()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
```

```
sns.distplot(ccdp.V28)
ccdp['V28'].fillna(ccdp['V28'].mean(),inplace=True)
ccdp.isnull().sum()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
```

```
sns.distplot(ccdp.Amount)
ccdp['Amount'].fillna(ccdp['Amount'].mean(),inplace=True)
ccdp.isnull().sum()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
Time      0
V1        0
V2        0
```

```
sns.distplot(ccdp.Class)
ccdp['Class'].fillna(ccdp['Class'].mean(),inplace=True)
ccdp.isnull().sum()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
```

```
# distribution of legit transactions and fraudulent transactions
ccdp['Class'].value_counts()
```

```
0.000000    5844
1.000000       3
0.000513       1
Name: Class, dtype: int64
V7          0
```

This dataset is highly unbalanced.

0 -> normal transactions

1 -> fraudulent transactions

```
V14         0
```

```
# separating the data for analysis
legit = ccdp[ccdp.Class == 0]
fraud = ccdp[ccdp.Class == 1]
```

```
V19         0
```

```
print(legit.shape)
print(fraud.shape)
```

```
(5844, 31)
(3, 31)
V20         1
```

```
# statistical messages of the data
legit.Amount.describe()
```

```
count    5844.000000
mean       65.002214
std       193.404607
min         0.000000
25%         4.397500
50%        15.655000
75%        56.570000
max      7712.430000
Name: Amount, dtype: float64
```

```
fraud.Amount.describe()
```

```
count      3.000000
mean     256.310000
std      264.880121
min        0.000000
25%      119.965000
50%      239.930000
75%      384.465000
max      529.000000
Name: Amount, dtype: float64
```

```
# compare the values for both transactions
ccdp.groupby('Class').mean()
```

Class	Time	V1	V2	V3	V4	V5	V6	
0.000000	2595.144422	-0.261024	0.280833	0.843675	0.088231	-0.003062	0.188308	0
0.000513	6347.000000	1.122792	-0.064138	1.035259	1.775286	-0.649522	0.282818	-0
1.000000	1780.000000	-2.553039	0.184644	-0.293711	2.872264	0.005330	-0.855718	-0

3 rows × 30 columns

Under-Sampling

Build a sample dataset containing similar distribution of normal taransactions and fraudulent transactions

Number of fraudulent transactions -> 81

```
legit_sample = legit.sample(n=81)
```

Concatenating two Dataframes

```
new_dataset = pd.concat([legit_sample, fraud], axis=0)
```

```
new_dataset.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
5755	6117	-0.812460	0.280784	1.274797	1.310739	-0.580612	0.038560	1.597415	-(
5630	5829	-0.688969	-0.162450	1.389613	-3.740448	0.306288	-0.554524	0.495774	-(
764	574	-1.062129	-0.618574	0.615388	-3.335834	0.746649	-0.540531	0.705932	(
4491	3789	1.125017	0.155286	1.467918	2.004638	-0.832711	0.052231	-0.723182	(
3658	3126	1.017152	-0.572347	1.058118	0.318101	-1.169185	-0.315466	-0.420007	-(

5 rows × 31 columns

```
new_dataset.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7
5594	5756	-1.069526	-0.149317	1.679008	-2.419008	0.789008	-0.217896	1.113164
3859	3410	1.501353	-1.108203	0.814055	-1.309409	-1.894297	-0.826509	-1.291526
541	406	-2.312227	1.951992	-1.609851	3.997906	-0.522188	-1.426545	-2.537387
623	472	-3.043541	-3.157307	1.088463	2.288644	1.359805	-1.064823	0.325574
4920	4462	-2.303350	1.759247	-0.359745	2.330243	-0.821628	-0.075788	0.562320

5 rows × 31 columns

```
new_dataset['Class'].value_counts()
```

```
0.0    81
1.0     3
Name: Class, dtype: int64
```

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

Class	Time	V1	V2	V3	V4	V5	V6	
0.0	2539.54321	-0.288968	0.356879	1.028599	0.256676	0.006768	0.285301	0.0377
1.0	1780.00000	-2.553039	0.184644	-0.293711	2.872264	0.005330	-0.855718	-0.5498

2 rows × 30 columns

Splitting the data into Features and Target

```
x = new_dataset.drop(columns='Class',axis=1)
y = new_dataset['Class']
```

```
print(x)
```

```
      Time        V1        V2        V3        V4        V5        V6  \
5755  6117 -0.812460  0.280784  1.274797  1.310739 -0.580612  0.038560
5630  5829 -0.688969 -0.162450  1.389613 -3.740448  0.306288 -0.554524
764    574 -1.062129 -0.618574  0.615388 -3.335834  0.746649 -0.540531
4491  3789  1.125017  0.155286  1.467918  2.004638 -0.832711  0.052231
3658  3126  1.017152 -0.572347  1.058118  0.318101 -1.169185 -0.315466
...    ...       ...       ...       ...       ...       ...       ...
5594  5756 -1.069526 -0.149317  1.679008 -2.419008  0.789008 -0.217896
3859  3410  1.501353 -1.108203  0.814055 -1.309409 -1.894297 -0.826509
541    406 -2.312227  1.951992 -1.609851  3.997906 -0.522188 -1.426545
623    472 -3.043541 -3.157307  1.088463  2.288644  1.359805 -1.064823
```

```
4920  4462 -2.303350  1.759247 -0.359745  2.330243 -0.821628 -0.075788

            V7        V8        V9  ...       V20       V21       V22  \
5755  1.597415 -0.195905  0.496023  ...  0.585061  0.035528 -0.138674
5630  0.495774 -0.290648  1.250978  ... -0.301001 -0.708265 -0.926938
764   0.705932  0.032525  1.334181  ...  0.311059 -0.114269 -0.661122
4491 -0.723182  0.119859  1.088655  ... -0.152402  0.013169  0.347485
3658 -0.420007 -0.054179  0.974584  ...  0.220932 -0.155981 -0.347512
...        ...       ...       ...  ...       ...       ...       ...
5594  1.113164 -0.610826  3.102446  ... -0.348998 -0.379733 -0.289215
3859 -1.291526 -0.111048 -1.512937  ... -0.358382 -0.067718  0.198207
541  -2.537387  1.391657 -2.770089  ...  0.126911  0.517232 -0.035049
623   0.325574 -0.067794 -0.270953  ...  2.102339  0.661696  0.435477
4920  0.562320 -0.399147 -0.238253  ... -0.430022 -0.294166 -0.932391

           V23       V24       V25       V26       V27       V28  Amount
5755  0.769209  0.286840 -0.484240 -0.597981  0.064021  0.186632  303.90
5630 -0.256803 -0.547499  0.323130 -0.425269  0.125349 -0.080506   18.79
764   0.136250 -1.377245  0.263820 -1.115516  0.079213  0.116638  131.37
4491 -0.023424  0.539720  0.198475  1.025332 -0.072794  0.001137    5.03
3658 -0.025914  0.517160  0.148134  0.926493 -0.043377  0.035135  109.63
...        ...       ...       ...       ...       ...       ...     ...
5594 -0.161039 -0.416205 -0.136385 -1.291803 -0.818651 -0.380878   45.01
3859  0.007487  0.362487  0.322289 -0.101170  0.048733  0.025776   10.50
541  -0.465211  0.320198  0.044519  0.177840  0.261145 -0.143276    0.00
623   1.375966 -0.293803  0.279798 -0.145362 -0.252773  0.035764  529.00
4920  0.172726 -0.087330 -0.156114 -0.542628  0.039566 -0.153029  239.93

[84 rows x 30 columns]
```

```
print(y)
```

```
5755    0.0
5630    0.0
764     0.0
4491    0.0
3658    0.0
       ...
5594    0.0
3859    0.0
541     1.0
623     1.0
4920    1.0
Name: Class, Length: 84, dtype: float64
```

Split the data into Training data & Testing data

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y, rando
```

```
print(x.shape, x_train.shape, x_test.shape)
```

```
(84, 30) (67, 30) (17, 30)
```

Model Training

Logistic Regression

```
model = LogisticRegression()
```

```
# training the logistic regression model with training data
model.fit(x_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
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
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression()
```

Model Evaluation

Accuracy Score

```
# accuracy on training data
x_train_prediction·=·model.predict(x_train)
training_data_accuracy·=·accuracy_score(x_train_prediction,·y_train)
```

```
print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  1.0
```

```
# accuracy on test data
x_test_prediction = model.predict(x_test)
test_data_accuracy·= accuracy_score(x_test_prediction, y_test)
```

```
print('Accuracy score on Test data : ', test_data_accuracy)
```

```
Accuracy score on Test data :  0.9411764705882353
```

0s completed at 12:59 PM