In [1]:
```python
import pandas as pd
import numpy as np                          # For mathematical calculations
import seaborn as sns                       # For data visualization
import matplotlib.pyplot as plt
import seaborn as sn                        # For plotting graphs
import io
%matplotlib inline
import warnings                             # To ignore any warnings
warnings.filterwarnings("ignore")
```

In [2]:
```python
filepath2 = r"C:\Users\91623\OneDrive\Desktop\projects\hill and valley\hill.csv"
df= pd.read_csv(filepath2)
print(df)
```

```
            V1        V2        V3        V4        V5        V6        V7  \
0        39.02     36.49     38.20     38.85     39.38     39.74     37.02
1         1.83      1.71      1.77      1.77      1.68      1.78      1.80
2     68177.69  66138.42  72981.88  74304.33  67549.66  69367.34  69169.41
3     44889.06  39191.86  40728.46  38576.36  45876.06  47034.00  46611.43
4         5.70      5.40      5.28      5.38      5.27      5.61      6.00
...        ...       ...       ...       ...       ...       ...       ...
1207     13.00     12.87     13.27     13.04     13.19     12.53     14.31
1208     48.66     50.11     48.55     50.43     50.09     49.67     48.95
1209  10160.65   9048.63   8994.94   9514.39   9814.74  10195.24  10031.47
1210     34.81     35.07     34.98     32.37     34.16     34.03     33.31
1211   8489.43   7672.98   9132.14   7985.73   8226.85   8554.28   8838.87

            V8        V9       V10  ...       V92       V93       V94  \
0        39.53     38.81     38.79  ...     36.62     36.92     38.80
1         1.70      1.75      1.78  ...      1.80      1.79      1.77
2     73268.61  74465.84  72503.37  ...  73438.88  71053.35  71112.62
3     37668.32  40980.89  38466.15  ...  42625.67  40684.20  46960.73
4         5.38      5.34      5.87  ...      5.17      5.67      5.60
...        ...       ...       ...  ...       ...       ...       ...
1207     13.33     13.63     14.55  ...     12.48     12.15     13.15
1208     48.65     48.63     48.61  ...     46.93     49.61     47.16
1209  10202.28   9152.99   9591.75  ...   9068.11   9191.80   9275.04
1210     32.48     35.63     32.48  ...     32.76     35.03     32.89
1211   8967.24   8635.14   8544.37  ...   8609.73   9209.48   8496.33

           V95       V96       V97       V98       V99      V100  Class
0        38.52     38.07     36.73     39.46     37.50     39.10      0
1         1.74      1.74      1.80      1.78      1.75      1.69      1
2     74916.48  72571.58  66348.97  71063.72  67404.27  74920.24      1
3     44546.80  45410.53  47139.44  43095.68  40888.34  39615.19      0
4         5.94      5.73      5.22      5.30      5.73      5.91      0
...        ...       ...       ...       ...       ...       ...    ...
1207     12.35     13.58     13.86     12.88     13.87     13.51      1
1208     48.17     47.94     49.81     49.89     47.43     47.77      0
1209   9848.18   9074.17   9601.74  10366.24   8997.60   9305.77      1
1210     31.91     33.85     35.28     32.49     32.83     34.82      1
1211   8724.01   8219.99   8550.86   8679.43   8389.31   8712.80      0

[1212 rows x 101 columns]
```

In [3]:
```python
df.head()
```

Out[3]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V92	V93	V94	V95
0	39.02	36.49	38.20	38.85	39.38	39.74	37.02	39.53	38.81	38.79	...	36.62	36.92	38.80	38.52
1	1.83	1.71	1.77	1.77	1.68	1.78	1.80	1.70	1.75	1.78	...	1.80	1.79	1.77	1.74
2	68177.69	66138.42	72981.88	74304.33	67549.66	69367.34	69169.41	73268.61	74465.84	72503.37	...	73438.88	71053.35	71112.62	74916.48
3	44889.06	39191.86	40728.46	38576.36	45876.06	47034.00	46611.43	37668.32	40980.89	38466.15	...	42625.67	40684.20	46960.73	44546.80
4	5.70	5.40	5.28	5.38	5.27	5.61	6.00	5.38	5.34	5.87	...	5.17	5.67	5.60	5.94

5 rows × 101 columns

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1212 entries, 0 to 1211
Columns: 101 entries, V1 to Class
dtypes: float64(100), int64(1)
memory usage: 956.5 KB
```

In [5]:
```python
df.dropna()
```

Out[5]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V92	V93	V94	
0	39.02	36.49	38.20	38.85	39.38	39.74	37.02	39.53	38.81	38.79	...	36.62	36.92	38.80	3
1	1.83	1.71	1.77	1.77	1.68	1.78	1.80	1.70	1.75	1.78	...	1.80	1.79	1.77	
2	68177.69	66138.42	72981.88	74304.33	67549.66	69367.34	69169.41	73268.61	74465.84	72503.37	...	73438.88	71053.35	71112.62	7491
3	44889.06	39191.86	40728.46	38576.36	45876.06	47034.00	46611.43	37668.32	40980.89	38466.15	...	42625.67	40684.20	46960.73	4454
4	5.70	5.40	5.28	5.38	5.27	5.61	6.00	5.38	5.34	5.87	...	5.17	5.67	5.60	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1207	13.00	12.87	13.27	13.04	13.19	12.53	14.31	13.33	13.63	14.55	...	12.48	12.15	13.15	1
1208	48.66	50.11	48.55	50.43	50.09	49.67	48.95	48.65	48.63	48.61	...	46.93	49.61	47.16	4
1209	10160.65	9048.63	8994.94	9514.39	9814.74	10195.24	10031.47	10202.28	9152.99	9591.75	...	9068.11	9191.80	9275.04	984
1210	34.81	35.07	34.98	32.37	34.16	34.03	33.31	32.48	35.63	32.48	...	32.76	35.03	32.89	3
1211	8489.43	7672.98	9132.14	7985.73	8226.85	8554.28	8838.87	8967.24	8635.14	8544.37	...	8609.73	9209.48	8496.33	872

1212 rows × 101 columns

In [8]:
```python
y=df['Class']
```

In [9]:
```python
x = df.drop(['Class'], axis = 1)
```

In [10]:
```python
x
```

Out[10]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V91	V92	V93	
0	39.02	36.49	38.20	38.85	39.38	39.74	37.02	39.53	38.81	38.79	...	37.57	36.62	36.92	3
1	1.83	1.71	1.77	1.77	1.68	1.78	1.80	1.70	1.75	1.78	...	1.71	1.80	1.79	
2	68177.69	66138.42	72981.88	74304.33	67549.66	69367.34	69169.41	73268.61	74465.84	72503.37	...	69384.71	73438.88	71053.35	7111
3	44889.06	39191.86	40728.46	38576.36	45876.06	47034.00	46611.43	37668.32	40980.89	38466.15	...	47653.60	42625.67	40684.20	4696
4	5.70	5.40	5.28	5.38	5.27	5.61	6.00	5.38	5.34	5.87	...	5.52	5.17	5.67	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1207	13.00	12.87	13.27	13.04	13.19	12.53	14.31	13.33	13.63	14.55	...	12.89	12.48	12.15	1
1208	48.66	50.11	48.55	50.43	50.09	49.67	48.95	48.65	48.63	48.61	...	47.45	46.93	49.61	4
1209	10160.65	9048.63	8994.94	9514.39	9814.74	10195.24	10031.47	10202.28	9152.99	9591.75	...	10413.41	9068.11	9191.80	927
1210	34.81	35.07	34.98	32.37	34.16	34.03	33.31	32.48	35.63	32.48	...	33.18	32.76	35.03	3
1211	8489.43	7672.98	9132.14	7985.73	8226.85	8554.28	8838.87	8967.24	8635.14	8544.37	...	7747.70	8609.73	9209.48	849

1212 rows × 100 columns

In [11]:
```python
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

In [15]:
```python
x=StandardScaler().fit(x).transform(x)
x
```

Out[15]:
```
array([[-0.45248681, -0.45361784, -0.45100881, ..., -0.45609618,
        -0.45164274, -0.45545496],
       [-0.45455665, -0.45556372, -0.45302369, ..., -0.45821768,
        -0.45362255, -0.45755405],
       [ 3.33983504,  3.24466709,  3.58338069, ...,  3.5427869 ,
         3.27907378,  3.74616847],
       ...,
       [ 0.11084204,  0.0505953 ,  0.04437307, ...,  0.12533312,
         0.04456025,  0.06450317],
       [-0.45272112, -0.45369729, -0.45118691, ..., -0.45648861,
        -0.45190136, -0.45569511],
       [ 0.01782872, -0.02636986,  0.05196137, ...,  0.03036056,
         0.01087365,  0.03123129]])
```

In [16]:
```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =train_test_split(x,y, test_size=0.3,random_state=2529)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[16]:
```
((848, 100), (364, 100), (848,), (364,))
```

In [17]:
```python
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
y_pred = lr.predict(x_test)
from sklearn.metrics import confusion_matrix ,classification_report
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[176   4]
 [ 92  92]]
              precision    recall  f1-score   support

           0       0.66      0.98      0.79       180
           1       0.96      0.50      0.66       184

    accuracy                           0.74       364
   macro avg       0.81      0.74      0.72       364
weighted avg       0.81      0.74      0.72       364
```

In [18]:
```python
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
rfc_pred=rfc.predict(x_test)
print(confusion_matrix(y_test,rfc_pred))
print(classification_report(y_test,rfc_pred))
```

```
[[105  75]
 [ 71 113]]
              precision    recall  f1-score   support

           0       0.60      0.58      0.59       180
           1       0.60      0.61      0.61       184

    accuracy                           0.60       364
   macro avg       0.60      0.60      0.60       364
weighted avg       0.60      0.60      0.60       364
```

In [19]:
```python
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf.fit(x_train,y_train)
clf_pred=clf.predict(x_test)
print(confusion_matrix(y_test,clf_pred))
print(classification_report(y_test,clf_pred))
```

```
[[100  80]
 [ 92  92]]
              precision    recall  f1-score   support

           0       0.52      0.56      0.54       180
           1       0.53      0.50      0.52       184

    accuracy                           0.53       364
   macro avg       0.53      0.53      0.53       364
weighted avg       0.53      0.53      0.53       364
```

In [20]:
```python
from sklearn.svm import SVC
svC=SVC()
svC.fit(x_train,y_train)
svC_pred=svC.predict(x_test)
from sklearn.metrics import confusion_matrix ,classification_report
print(confusion_matrix(y_test,svC_pred))
print(classification_report(y_test,svC_pred))
```

```
[[162  18]
 [159  25]]
              precision    recall  f1-score   support

           0       0.50      0.90      0.65       180
           1       0.58      0.14      0.22       184

    accuracy                           0.51       364
   macro avg       0.54      0.52      0.43       364
weighted avg       0.54      0.51      0.43       364
```

In [21]:
```python
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train,y_train)
gnb_pred=gnb.predict(x_test)
print(confusion_matrix(y_test,gnb_pred))
print(classification_report(y_test,gnb_pred))
```

```
[[121  59]
 [102  82]]
              precision    recall  f1-score   support

           0       0.54      0.67      0.60       180
           1       0.58      0.45      0.50       184

    accuracy                           0.56       364
   macro avg       0.56      0.56      0.55       364
weighted avg       0.56      0.56      0.55       364
```

In [ ]: