In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:

```
d=pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Hill%20Valley%2
```

In [80]:

```
d.head()
```

Out[80]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	
0	39.02	36.49	38.20	38.85	39.38	39.74	37.02	39.53	38.81	3
1	1.83	1.71	1.77	1.77	1.68	1.78	1.80	1.70	1.75	
2	68177.69	66138.42	72981.88	74304.33	67549.66	69367.34	69169.41	73268.61	74465.84	7250
3	44889.06	39191.86	40728.46	38576.36	45876.06	47034.00	46611.43	37668.32	40980.89	3846
4	5.70	5.40	5.28	5.38	5.27	5.61	6.00	5.38	5.34	

5 rows × 101 columns

In [81]:

```
d.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 [82]:

```
d.describe()
```

Out[82]:

	V1	V2	V3	V4	V5	V6
count	1212.000000	1212.000000	1212.000000	1212.000000	1212.000000	1212.000000
mean	8169.091881	8144.306262	8192.653738	8176.868738	8128.297211	8173.030008
std	17974.950461	17881.049734	18087.938901	17991.903982	17846.757963	17927.114105
min	0.920000	0.900000	0.850000	0.890000	0.880000	0.860000
25%	19.602500	19.595000	18.925000	19.277500	19.210000	19.582500
50%	301.425000	295.205000	297.260000	299.720000	295.115000	294.380000
75%	5358.795000	5417.847500	5393.367500	5388.482500	5321.987500	5328.040000
max	117807.870000	108896.480000	119031.350000	110212.590000	113000.470000	116848.390000

8 rows × 101 columns

In [83]:
```
d.columns
```

Out[83]:
```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       ...
       'V92', 'V93', 'V94', 'V95', 'V96', 'V97', 'V98', 'V99', 'V100',
       'Class'],
      dtype='object', length=101)
```

In [84]:
```
print(d.columns.tolist())
```

```
['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11', 'V12', 'V1
3', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21', 'V22', 'V23', 'V2
4', 'V25', 'V26', 'V27', 'V28', 'V29', 'V30', 'V31', 'V32', 'V33', 'V34', 'V3
5', 'V36', 'V37', 'V38', 'V39', 'V40', 'V41', 'V42', 'V43', 'V44', 'V45', 'V4
6', 'V47', 'V48', 'V49', 'V50', 'V51', 'V52', 'V53', 'V54', 'V55', 'V56', 'V5
7', 'V58', 'V59', 'V60', 'V61', 'V62', 'V63', 'V64', 'V65', 'V66', 'V67', 'V6
8', 'V69', 'V70', 'V71', 'V72', 'V73', 'V74', 'V75', 'V76', 'V77', 'V78', 'V7
9', 'V80', 'V81', 'V82', 'V83', 'V84', 'V85', 'V86', 'V87', 'V88', 'V89', 'V9
0', 'V91', 'V92', 'V93', 'V94', 'V95', 'V96', 'V97', 'V98', 'V99', 'V100', 'Cla
ss']
```

In [85]:
```
d.shape
```

Out[85]:
```
(1212, 101)
```

Get Unique Values in y Variable

In [86]:
```
d['Class'].value_counts()
```

Out[86]:
```
0    606
1    606
Name: Class, dtype: int64
```

In [87]:
```
d.groupby('Class').mean()
```

Out[87]:

	V1	V2	V3	V4	V5	V6	V7
Class							
0	7913.333251	7825.339967	7902.497294	7857.032079	7775.610198	7875.436337	7804.166584
1	8424.850512	8463.272558	8482.810182	8496.705396	8480.984224	8470.623680	8572.998911

2 rows × 100 columns

Define y and x

In [88]:
```
y=d['Class']
```

In [89]:
```
y.shape
```

Out[89]:
```
(1212,)
```

In [90]:
```
y
```

Out[90]:
```
0       0
1       1
2       1
3       0
4       0
       ..
1207    1
1208    0
1209    1
1210    1
1211    0
Name: Class, Length: 1212, dtype: int64
```

In [91]:
```
x=d.drop(['Class'],axis=1)
```

In [92]:
```
x.shape
```

Out[92]:
```
(1212, 100)
```

In [93]:
```
x
```

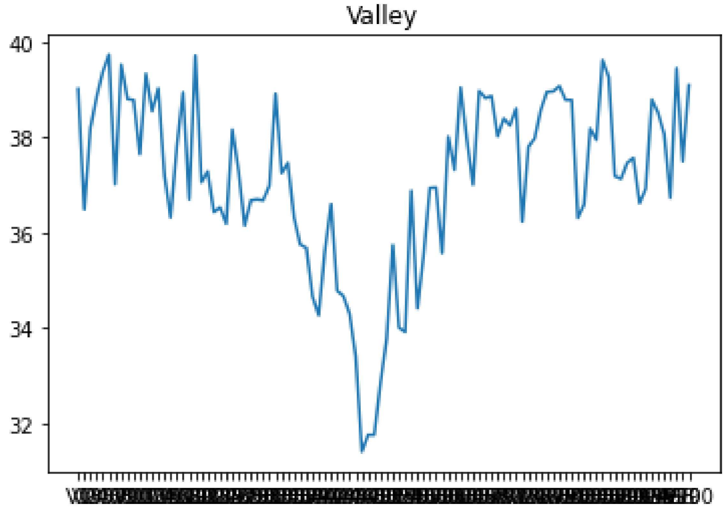
Out[93]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	
0	39.02	36.49	38.20	38.85	39.38	39.74	37.02	39.53	38.81	
1	1.83	1.71	1.77	1.77	1.68	1.78	1.80	1.70	1.75	
2	68177.69	66138.42	72981.88	74304.33	67549.66	69367.34	69169.41	73268.61	74465.84	7
3	44889.06	39191.86	40728.46	38576.36	45876.06	47034.00	46611.43	37668.32	40980.89	3
4	5.70	5.40	5.28	5.38	5.27	5.61	6.00	5.38	5.34	
...	...	...	...	...	...	...	...	...	...	
1207	13.00	12.87	13.27	13.04	13.19	12.53	14.31	13.33	13.63	
1208	48.66	50.11	48.55	50.43	50.09	49.67	48.95	48.65	48.63	
1209	10160.65	9048.63	8994.94	9514.39	9814.74	10195.24	10031.47	10202.28	9152.99	
1210	34.81	35.07	34.98	32.37	34.16	34.03	33.31	32.48	35.63	
1211	8489.43	7672.98	9132.14	7985.73	8226.85	8554.28	8838.87	8967.24	8635.14	

1212 rows × 100 columns

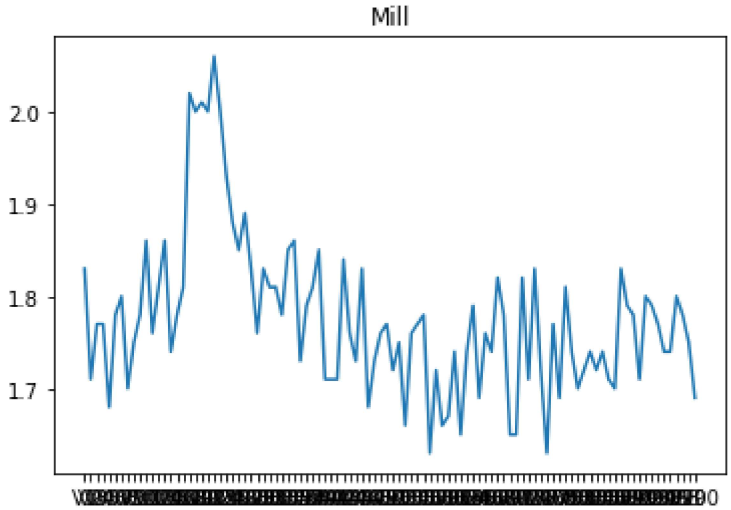
In [94]:
```
plt.plot(x.iloc[0,:])
plt.title('Valley')
```

Out[94]:
```
Text(0.5, 1.0, 'Valley')
```



In [95]:
```
plt.plot(x.iloc[1,:])
plt.title('Mill')
```

Out[95]:
```
Text(0.5, 1.0, 'Mill')
```



Get x Variables standardised

In [96]:
```
from sklearn.preprocessing import StandardScaler
```

In [97]:
```
l=StandardScaler()
```

In [98]:
```
x=l.fit_transform(x)
```

In [99]:
```
x
```

Out[99]:
```
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 [100]:
```
x.shape
```

Out[100]:
```
(1212, 100)
```

Train Test Sample

In [101]:
```
from sklearn.model_selection import  train_test_split
```

In [102]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=252
```

In [103]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[103]:
```
((848, 100), (364, 100), (848,), (364,))
```

Model Train

In [104]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
```

In [105]:
```
lr.fit(x_train,y_train)
```

Out[105]:
```
▾ LogisticRegression
LogisticRegression()
```

Model Prediction

In [106]:
```
y_pred=lr.predict(x_test)
```

In [107]:
```
y_pred.shape
```

Out[107]:
```
(364,)
```

In [108]:
```
y_pred
```

Out[108]:
```
array([1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

Probability Of Each Predicted Class

In [110]:
```
lr.predict_proba(x_test)
```

```
       [0.40808422, 0.59191578],
       [0.50493538, 0.49506462],
       [0.50418094, 0.49581906],
       [0.50456299, 0.49543701],
       [0.50454562, 0.49545438],
       [0.99663629, 0.00336371],
       [0.50457384, 0.49542616],
       [0.0062339 , 0.9937661 ],
       [0.5045384 , 0.4954616 ],
       [0.52126881, 0.47873119],
       [0.30440945, 0.69559055],
       [0.31548267, 0.68451733],
       [0.8842735 , 0.1157265 ],
       [0.50460229, 0.49539771],
       [0.5045896 , 0.4954104 ],
       [0.50352471, 0.49647529],
       [0.5054642 , 0.4945358 ],
       [0.56872655, 0.43127345],
       [0.49685317, 0.50314683],
       [0.50579842, 0.49420158]])
```

Model Evaluation

In [111]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

In [112]:
```
print(confusion_matrix(y_test,y_pred))
```

```
[[176   4]
 [ 92  92]]
```

In [113]:
```
print(classification_report(y_test,y_pred))
```

```
              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
```

Future Predictions

In [114]:
```
x_new=d.sample(1)
```

In [115]:
```
x_new
```

Out[115]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V92	V
492	433.71	392.07	401.92	445.66	419.42	409.89	422.95	404.18	393.58	395.29	...	438.21	424

1 rows × 101 columns

In [116]:
```
x_new.shape
```

Out[116]:
```
(1, 101)
```

In [117]:
```
x_new=x_new.drop('Class',axis=1)
```

In [118]:
```
x_new
```

Out[118]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V91	V
492	433.71	392.07	401.92	445.66	419.42	409.89	422.95	404.18	393.58	395.29	...	431.15	438

1 rows × 100 columns

```
In [119]: x_new.shape

Out[119]: (1, 100)

In [120]: x_new=l.fit_transform(x_new)

In [121]: y_pred_n=lr.predict(x_new)

In [123]: y_pred_n

Out[123]: array([1], dtype=int64)

In [124]: lr.predict_proba(x_new)

Out[124]: array([[0.49604115, 0.50395885]])

  In [ ]:

  In [ ]:
```