HILL AND VALLEY PREDICTION

Objective: To analyze the hills and valleys

Data Source: YBI Foundation through Github

Import Library

Import Data

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv(r'https://raw.githubusercontent.com/YBIFoundation/Dataset/main/Hil
df.head()
```

Out[ ]:

	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
3	44889.06	39191.86	40728.46	38576.36	45876.06	47034.00	46611.43	37668.32	40980.89
4	5.70	5.40	5.28	5.38	5.27	5.61	6.00	5.38	5.34

5 rows × 101 columns

Describe Data

In [ ]:
```
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 [ ]:
```
df.describe()
```

Out[ ]:

	V1	V2	V3	V4	V5	
count	1212.000000	1212.000000	1212.000000	1212.000000	1212.000000	1212.
mean	8169.091881	8144.306262	8192.653738	8176.868738	8128.297211	8173.
std	17974.950461	17881.049734	18087.938901	17991.903982	17846.757963	17927.
min	0.920000	0.900000	0.850000	0.890000	0.880000	0.
25%	19.602500	19.595000	18.925000	19.277500	19.210000	19.
50%	301.425000	295.205000	297.260000	299.720000	295.115000	294.
75%	5358.795000	5417.847500	5393.367500	5388.482500	5321.987500	5328.
max	117807.870000	108896.480000	119031.350000	110212.590000	113000.470000	116848.

8 rows × 101 columns

In [ ]:
```
df.columns
```

Out[ ]:
```
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 [ ]:
```
print(df.columns.tolist())
```

```
['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', 'V29', 'V30', 'V31', 'V32', 'V33', 'V34', 'V35', 'V36', 'V37',
'V38', 'V39', 'V40', 'V41', 'V42', 'V43', 'V44', 'V45', 'V46', 'V47', 'V48', 'V49',
'V50', 'V51', 'V52', 'V53', 'V54', 'V55', 'V56', 'V57', 'V58', 'V59', 'V60', 'V61',
'V62', 'V63', 'V64', 'V65', 'V66', 'V67', 'V68', 'V69', 'V70', 'V71', 'V72', 'V73',
'V74', 'V75', 'V76', 'V77', 'V78', 'V79', 'V80', 'V81', 'V82', 'V83', 'V84', 'V85',
'V86', 'V87', 'V88', 'V89', 'V90', 'V91', 'V92', 'V93', 'V94', 'V95', 'V96', 'V97',
'V98', 'V99', 'V100', 'Class']
```

In [ ]:
```
df.shape
```

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

In [ ]:
```
df['Class'].value_counts()
```

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

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

Out[ ]:

Class	V1	V2	V3	V4	V5	V6	
0	7913.333251	7825.339967	7902.497294	7857.032079	7775.610198	7875.436337	7804.
1	8424.850512	8463.272558	8482.810182	8496.705396	8480.984224	8470.623680	8572.

2 rows × 100 columns

Define Target Variable (y) and Feature Variables (X)

In [ ]:
```
y = df['Class']
y.shape
```

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

In [ ]:
```
y
```

Out[ ]:
```
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 [ ]:
```
X = df.drop('Class',axis=1)
X.shape
```

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

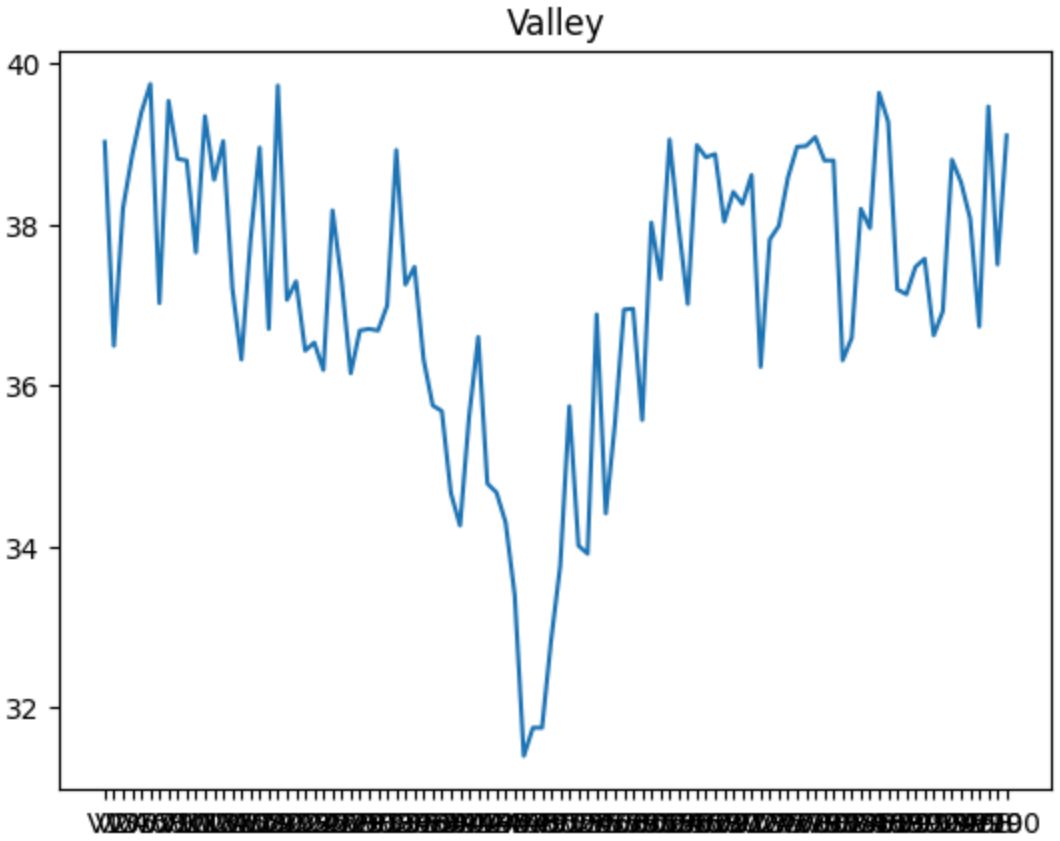
In [ ]:
```
X
```

Out[ ]:

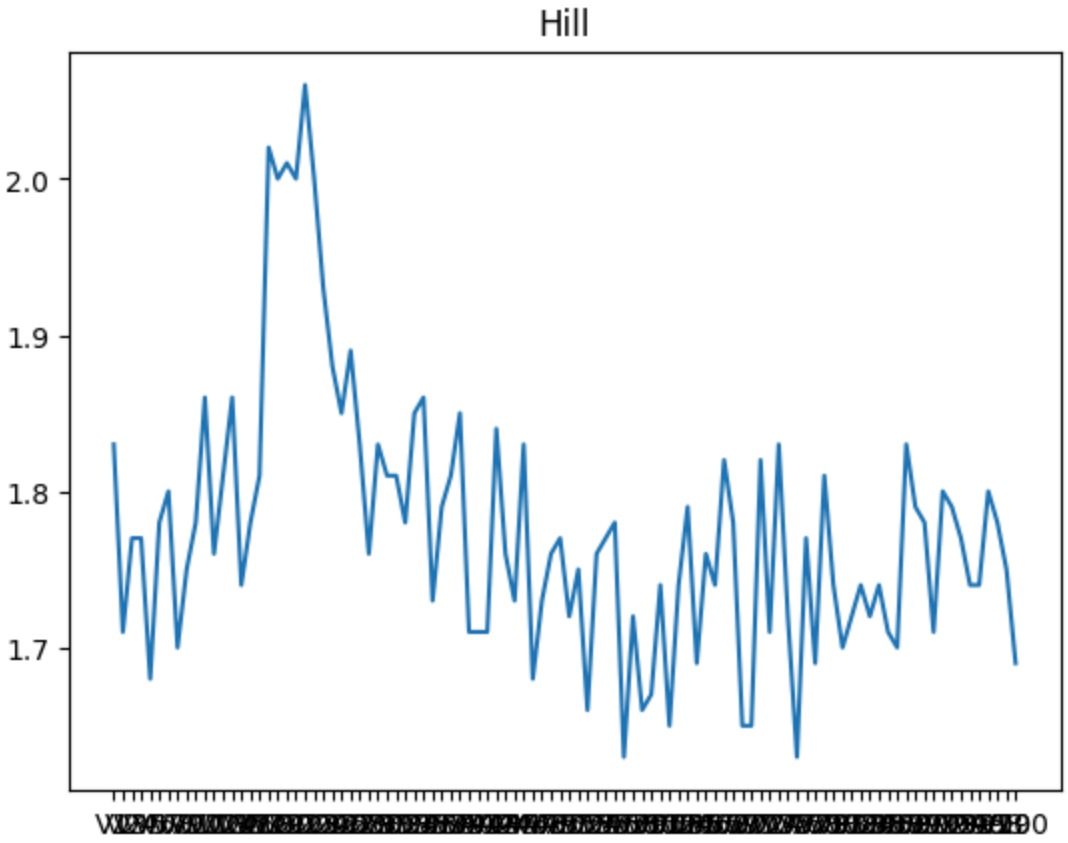
	V1	V2	V3	V4	V5	V6	V7	V8	
0	39.02	36.49	38.20	38.85	39.38	39.74	37.02	39.53	3
1	1.83	1.71	1.77	1.77	1.68	1.78	1.80	1.70	
2	68177.69	66138.42	72981.88	74304.33	67549.66	69367.34	69169.41	73268.61	7446
3	44889.06	39191.86	40728.46	38576.36	45876.06	47034.00	46611.43	37668.32	4098
4	5.70	5.40	5.28	5.38	5.27	5.61	6.00	5.38	
...	...	...	...	...	...	...	...	...	
1207	13.00	12.87	13.27	13.04	13.19	12.53	14.31	13.33	1
1208	48.66	50.11	48.55	50.43	50.09	49.67	48.95	48.65	4
1209	10160.65	9048.63	8994.94	9514.39	9814.74	10195.24	10031.47	10202.28	915
1210	34.81	35.07	34.98	32.37	34.16	34.03	33.31	32.48	3
1211	8489.43	7672.98	9132.14	7985.73	8226.85	8554.28	8838.87	8967.24	863

1212 rows × 100 columns

In [ ]:
```
import matplotlib.pyplot as plt
plt.plot(X.iloc[0,:])
plt.title('Valley');
```



In [ ]:
```
plt.plot(X.iloc[1,:])
plt.title('Hill');
```



Train Test Split

In [ ]:
```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X = ss.fit_transform(X)
X
```

Out[ ]:
```
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 [ ]:
```
X.shape
```

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

In [ ]:
```
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, stratify= y,
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

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

In [ ]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

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

Prediction

In [ ]:
```
y_pred = lr.predict(X_test)
y_pred.shape
```

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

In [ ]:
```
y_pred
```

Out[ ]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 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, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1])
```

In [ ]:
```
lr.predict_proba(X_test)
```

Out[ ]:
```
array([[0.56336744, 0.43663256],
       [0.50327039, 0.49672961],
       [0.57446514, 0.42553486],
       [0.50737525, 0.49262475],
       [0.50767478, 0.49232522],
       [0.5087066 , 0.4912934 ],
       [0.50793217, 0.49206783],
       [0.60357917, 0.39642083],
       [0.51009655, 0.48990345],
       [0.50964836, 0.49035164],
       [0.50721213, 0.49278787],
       [0.51503419, 0.48496581],
       [0.93595857, 0.06404143],
       [0.50968822, 0.49031178],
       [0.52004959, 0.47995041],
       [0.73731198, 0.26268802],
       [0.47389171, 0.52610829],
       [0.50781847, 0.49218153],
       [0.50862145, 0.49137855],
       [0.5086342 , 0.4913658 ],
       [0.29771935, 0.70228065],
       [0.38273299, 0.61726701],
       [0.50865396, 0.49134604],
       [0.28367974, 0.71632026],
       [0.50873182, 0.49126818],
       [0.50707761, 0.49292239],
       [0.50896136, 0.49103864],
       [0.50811697, 0.49188303],
       [0.50861558, 0.49138442],
       [0.5074842 , 0.4925158 ],
       [0.41565133, 0.58434867],
       [0.51322175, 0.48677825],
       [0.19965039, 0.80034961],
       [0.74863308, 0.25136692],
       [0.50865392, 0.49134608],
       [0.50862564, 0.49137436],
       [0.50868082, 0.49131918],
       [0.50853411, 0.49146589],
       [0.51269831, 0.48730169],
       [0.51582682, 0.48417318],
       [0.50858125, 0.49141875],
       [0.52031811, 0.47968189],
       [0.28012043, 0.71987957],
       [0.51125979, 0.48874021],
       [0.54087677, 0.45912323],
       [0.46730929, 0.53269071],
       [0.50822765, 0.49177235],
       [0.5238823 , 0.4761177 ],
       [0.50104301, 0.49895699],
       [0.50875872, 0.49124128],
       [0.50864302, 0.49135698],
       [0.54043012, 0.45956988],
       [0.50846686, 0.49153314],
       [0.50733903, 0.49266097],
       [0.51454789, 0.48545211],
       [0.50856525, 0.49143475],
```

```
[0.50860437, 0.49139563],
[0.50893174, 0.49106826],
[0.5586017 , 0.4413983 ],
[0.35542757, 0.64457243],
[0.50448798, 0.49551202],
[0.50859471, 0.49140529],
[0.50862332, 0.49137668],
[0.33508012, 0.66491988],
[0.51012545, 0.48987455],
[0.50853881, 0.49146119],
[0.56765074, 0.43234926],
[0.47833062, 0.52166938],
[0.50867152, 0.49132848],
[0.50644915, 0.49355085],
[0.509056  , 0.490944  ],
[0.50862813, 0.49137187],
[0.51023788, 0.48976212],
[0.5082914 , 0.4917086 ],
[0.50959783, 0.49040217],
[0.59324195, 0.40675805],
[0.40833642, 0.59166358],
[0.36255743, 0.63744257],
[0.77441815, 0.22558185],
[0.54037267, 0.45962733],
[0.5085901 , 0.4914099 ],
[0.50863832, 0.49136168],
[0.50820561, 0.49179439],
[0.4964397 , 0.5035603 ],
[0.5085909 , 0.4914091 ],
[0.50399121, 0.49600879],
[0.50865059, 0.49134941],
[0.50908459, 0.49091541],
[0.50847808, 0.49152192],
[0.50056184, 0.49943816],
[0.64764177, 0.35235823],
[0.50865275, 0.49134725],
[0.50962532, 0.49037468],
[0.5217569 , 0.4782431 ],
[0.50861733, 0.49138267],
[0.00188839, 0.99811161],
[0.45882122, 0.54117878],
[0.60418366, 0.39581634],
[0.50866312, 0.49133688],
[0.53064522, 0.46935478],
[0.51532306, 0.48467694],
[0.49165318, 0.50834682],
[0.54605773, 0.45394227],
[0.50882289, 0.49117711],
[0.50537031, 0.49462969],
[0.55736619, 0.44263381],
[0.5083356 , 0.4916644 ],
[0.50016455, 0.49983545],
[0.50861668, 0.49138332],
[0.50792357, 0.49207643],
[0.27493137, 0.72506863],
[0.50848806, 0.49151194],
```

```
[0.51027351, 0.48972649],
[0.50866435, 0.49133565],
[0.55368443, 0.44631557],
[0.14525857, 0.85474143],
[0.50812566, 0.49187434],
[0.50425641, 0.49574359],
[0.83669283, 0.16330717],
[0.51807391, 0.48192609],
[0.50934292, 0.49065708],
[0.38916501, 0.61083499],
[0.50855269, 0.49144731],
[0.48813824, 0.51186176],
[0.50865032, 0.49134968],
[0.50869167, 0.49130833],
[0.5074832 , 0.4925168 ],
[0.51478613, 0.48521387],
[0.50957087, 0.49042913],
[0.56654947, 0.43345053],
[0.50531205, 0.49468795],
[0.52184111, 0.47815889],
[0.50901839, 0.49098161],
[0.47509921, 0.52490079],
[0.54368227, 0.45631773],
[0.50843518, 0.49156482],
[0.51352798, 0.48647202],
[0.50857309, 0.49142691],
[0.48070577, 0.51929423],
[0.50821319, 0.49178681],
[0.50859266, 0.49140734],
[0.61181147, 0.38818853],
[0.50871788, 0.49128212],
[0.50965742, 0.49034258],
[0.50876595, 0.49123405],
[0.26235726, 0.73764274],
[0.508621  , 0.491379  ],
[0.71522876, 0.28477124],
[0.5088026 , 0.4911974 ],
[0.50865615, 0.49134385],
[0.50858526, 0.49141474],
[0.50864688, 0.49135312],
[0.88654953, 0.11345047],
[0.50865943, 0.49134057],
[0.50865124, 0.49134876],
[0.10671285, 0.89328715],
[0.50865425, 0.49134575],
[0.41188852, 0.58811148],
[0.55572713, 0.44427287],
[0.51395264, 0.48604736],
[0.50861696, 0.49138304],
[0.89581583, 0.10418417],
[0.50894481, 0.49105519],
[0.5096374 , 0.4903626 ],
[0.50845822, 0.49154178],
[0.62869575, 0.37130425],
[0.56013681, 0.43986319],
[0.41050707, 0.58949293],
```

```
[0.51594206, 0.48405794],
[0.51034503, 0.48965497],
[0.51006378, 0.48993622],
[0.50868313, 0.49131687],
[0.53572343, 0.46427657],
[0.5088925 , 0.4911075 ],
[0.57188349, 0.42811651],
[0.49876233, 0.50123767],
[0.80087408, 0.19912592],
[0.50865013, 0.49134987],
[0.733861  , 0.266139  ],
[0.58443597, 0.41556403],
[0.50888943, 0.49111057],
[0.50875194, 0.49124806],
[0.50856455, 0.49143545],
[0.99289372, 0.00710628],
[0.50862518, 0.49137482],
[0.50902015, 0.49097985],
[0.42178743, 0.57821257],
[0.51195925, 0.48804075],
[0.50693383, 0.49306617],
[0.49439744, 0.50560256],
[0.50864169, 0.49135831],
[0.34064698, 0.65935302],
[0.50050909, 0.49949091],
[0.47805584, 0.52194416],
[0.50862875, 0.49137125],
[0.50882094, 0.49117906],
[0.51986751, 0.48013249],
[0.5087651 , 0.4912349 ],
[0.52968383, 0.47031617],
[0.50624454, 0.49375546],
[0.50866737, 0.49133263],
[0.52908124, 0.47091876],
[0.51997503, 0.48002497],
[0.29951874, 0.70048126],
[0.49517002, 0.50482998],
[0.50677621, 0.49322379],
[0.50861995, 0.49138005],
[0.50847963, 0.49152037],
[0.50858711, 0.49141289],
[0.57059462, 0.42940538],
[0.60694762, 0.39305238],
[0.50848908, 0.49151092],
[0.45270204, 0.54729796],
[0.5982575 , 0.4017425 ],
[0.16132575, 0.83867425],
[0.50863962, 0.49136038],
[0.50972408, 0.49027592],
[0.50865437, 0.49134563],
[0.44337178, 0.55662822],
[0.46571331, 0.53428669],
[0.50860372, 0.49139628],
[0.50273773, 0.49726227],
[0.65421583, 0.34578417],
[0.4883913 , 0.5116087 ],
```

```
[0.50863483, 0.49136517],
[0.0332069 , 0.9667931 ],
[0.50858267, 0.49141733],
[0.50815533, 0.49184467],
[0.55923855, 0.44076145],
[0.50893217, 0.49106783],
[0.07230376, 0.92769624],
[0.50858321, 0.49141679],
[0.4224533 , 0.5775467 ],
[0.42780692, 0.57219308],
[0.50889484, 0.49110516],
[0.52570584, 0.47429416],
[0.49822124, 0.50177876],
[0.30161901, 0.69838099],
[0.50821596, 0.49178404],
[0.50888748, 0.49111252],
[0.1753728 , 0.8246272 ],
[0.50710602, 0.49289398],
[0.78121492, 0.21878508],
[0.48397767, 0.51602233],
[0.50854707, 0.49145293],
[0.51044181, 0.48955819],
[0.80138593, 0.19861407],
[0.50656725, 0.49343275],
[0.50568628, 0.49431372],
[0.50864418, 0.49135582],
[0.50865874, 0.49134126],
[0.30934702, 0.69065298],
[0.50848121, 0.49151879],
[0.05533464, 0.94466536],
[0.50864408, 0.49135592],
[0.45699689, 0.54300311],
[0.53046093, 0.46953907],
[0.50863729, 0.49136271],
[0.50861171, 0.49138829],
[0.50890203, 0.49109797],
[0.49228277, 0.50771723],
[0.55054228, 0.44945772],
[0.49563697, 0.50436303],
[0.50864186, 0.49135814],
[0.50547373, 0.49452627],
[0.50705965, 0.49294035],
[0.50850825, 0.49149175],
[0.50863514, 0.49136486],
[0.6070427 , 0.3929573 ],
[0.50861908, 0.49138092],
[0.50836716, 0.49163284],
[0.50845779, 0.49154221],
[0.51338135, 0.48661865],
[0.50859941, 0.49140059],
[0.50550932, 0.49449068],
[0.37826994, 0.62173006],
[0.50859178, 0.49140822],
[0.91470492, 0.08529508],
[0.50860396, 0.49139604],
[0.50846494, 0.49153506],
```

```
[0.50858272, 0.49141728],
[0.50860138, 0.49139862],
[0.51559009, 0.48440991],
[0.50927127, 0.49072873],
[0.5093133 , 0.4906867 ],
[0.51131576, 0.48868424],
[0.50854686, 0.49145314],
[0.50861361, 0.49138639],
[0.43520623, 0.56479377],
[0.50865075, 0.49134925],
[0.51146009, 0.48853991],
[0.5081865 , 0.4918135 ],
[0.46802482, 0.53197518],
[0.50863676, 0.49136324],
[0.40298303, 0.59701697],
[0.52052627, 0.47947373],
[0.44184753, 0.55815247],
[0.45694071, 0.54305929],
[0.34477066, 0.65522934],
[0.50856495, 0.49143505],
[0.32961822, 0.67038178],
[0.50842322, 0.49157678],
[0.50862839, 0.49137161],
[0.50861587, 0.49138413],
[0.49982741, 0.50017259],
[0.50640416, 0.49359584],
[0.00939576, 0.99060424],
[0.50865029, 0.49134971],
[0.40727561, 0.59272439],
[0.50783678, 0.49216322],
[0.50863236, 0.49136764],
[0.50881992, 0.49118008],
[0.5062655 , 0.4937345 ],
[0.48327858, 0.51672142],
[0.49892383, 0.50107617],
[0.50862417, 0.49137583],
[0.51207822, 0.48792178],
[0.87769406, 0.12230594],
[0.50885621, 0.49114379],
[0.50775443, 0.49224557],
[0.50885088, 0.49114912],
[0.45953645, 0.54046355],
[0.50860029, 0.49139971],
[0.49737448, 0.50262552],
[0.50902558, 0.49097442],
[0.03768716, 0.96231284],
[0.46144808, 0.53855192],
[0.49909602, 0.50090398],
[0.71375106, 0.28624894],
[0.50867123, 0.49132877],
[0.63597662, 0.36402338],
[0.50894067, 0.49105933],
[0.53221095, 0.46778905],
[0.4735918 , 0.5264082 ],
[0.51131842, 0.48868158],
[0.47711039, 0.52288961],
```

```
       [0.50836868, 0.49163132],
       [0.51791139, 0.48208861],
       [0.50929629, 0.49070371],
       [0.77824423, 0.22175577],
       [0.50878251, 0.49121749],
       [0.53444807, 0.46555193],
       [0.68401993, 0.31598007],
       [0.50837605, 0.49162395],
       [0.49955891, 0.50044109],
       [0.66733466, 0.33266534],
       [0.51287013, 0.48712987],
       [0.5100684 , 0.4899316 ],
       [0.49493863, 0.50506137],
       [0.50415413, 0.49584587],
       [0.50875424, 0.49124576],
       [0.50859968, 0.49140032],
       [0.51714632, 0.48285368],
       [0.49828539, 0.50171461],
       [0.50868242, 0.49131758],
       [0.5887849 , 0.4112151 ],
       [0.50578145, 0.49421855],
       [0.50864404, 0.49135596],
       [0.78691964, 0.21308036],
       [0.50865314, 0.49134686],
       [0.5085737 , 0.4914263 ],
       [0.48168718, 0.51831282],
       [0.50838591, 0.49161409],
       [0.05333433, 0.94666567]])
```

Explanation

In [ ]:
```
from sklearn.metrics import confusion_matrix, classification_report
print(confusion_matrix(y_test, y_pred))
```

```
[[181   1]
 [106  76]]
```

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

```
              precision    recall  f1-score   support

           0       0.63      0.99      0.77       182
           1       0.99      0.42      0.59       182

    accuracy                           0.71       364
   macro avg       0.81      0.71      0.68       364
weighted avg       0.81      0.71      0.68       364
```

In [ ]:
```
X_new = df.sample(1)
X_new
```

Out[ ]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V92	V93	V94	V95	V
409	9.63	8.97	8.48	8.5	9.79	10.15	8.74	9.52	10.18	8.7	...	8.43	8.48	8.81	9.26	8.

1 rows × 101 columns

In [ ]:
```
X_new.shape
```

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

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

Out[ ]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V91	V92	V93	V94	V
409	9.63	8.97	8.48	8.5	9.79	10.15	8.74	9.52	10.18	8.7	...	9.05	8.43	8.48	8.81	9.

1 rows × 100 columns

In [ ]:
```
X_new.shape
```

Out[ ]:
```
(1, 100)
```

In [ ]:
```
X_new = ss.fit_transform(X_new)
y_pard_new = lr.predict(X_new)
y_pard_new
```

Out[ ]:
```
array([1])
```

In [ ]:
```
lr.predict_proba(X_new)
```

Out[ ]:
```
array([[0.49714993, 0.50285007]])
```