Title of the Project: Hill and Valley Prediction using Logistic Regression method

Objective: TO develop an Machine Learning Model to predict Hill and Valley using Logistic Regression method that can accurately classify geographical locations as either hills or valleys based on a set of input features. The model will be trained on a labeled dataset of geographical features and their corresponding classifications, and then evaluated on a seperate test dataset to measure its performance. The ultimate goal of this project is to provde a useful tool for identifying hills and valleys in various geographic locations, which can have important applications in fields such as geology, agriculture, and urban planning.

Data source: YBI foundation github

Import Library

```
import pandas as pd
```

```
import numpy as np
```

Import Library

```
df = pd.read_csv('https://github.com/YBIFoundation/Dataset/raw/main/Hill%20Valley%20Dataset.csv')
```

```
df.head()
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V92	V93	
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	
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	
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	7
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	4
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 rows × 101 columns

```
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
```

Describe Data

```
df.describe()
```

	V1	V2	V3	V4	V5	V6	V7	V8
count	1212.000000	1212.000000	1212.000000	1212.000000	1212.000000	1212.000000	1212.000000	1212.000000

```
df.columns
```

```
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)
```

50%	301.425000	295.205000	297.260000	299.720000	295.115000	294.380000	295.935000	290.850000

```
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
```

```
df.shape
```

```
(1212, 101)
```

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

```
0    606
1    606
Name: Class, dtype: int64
```

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

Class	V1	V2	V3	V4	V5	V6	V7	V8	V9	
0	7913.333251	7825.339967	7902.497294	7857.032079	7775.610198	7875.436337	7804.166584	7722.324802	7793.328416	768
1	8424.850512	8463.272558	8482.810182	8496.705396	8480.984224	8470.623680	8572.998911	8644.958284	8516.011716	855

2 rows × 100 columns

Define Target Variable(y) and Feature Variable(x)

```
y=df['Class']
```

```
y.shape
```

```
(1212,)
```

```
y
```

```
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
```

```
x=df[['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21
```

```
x.shape
```

```
(1212, 100)
```

```
x
```

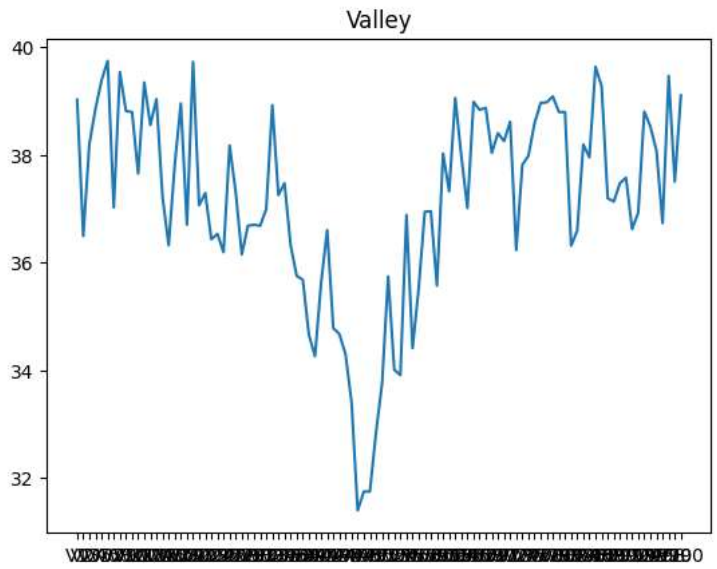
	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V91	V92
0	39.02	36.49	38.20	38.85	39.38	39.74	37.02	39.53	38.81	38.79	...	37.57	36.62
1	1.83	1.71	1.77	1.77	1.68	1.78	1.80	1.70	1.75	1.78	...	1.71	1.80
2	68177.69	66138.42	72981.88	74304.33	67549.66	69367.34	69169.41	73268.61	74465.84	72503.37	...	69384.71	73438.88
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
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
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
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
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
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
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

1212 rows × 100 columns

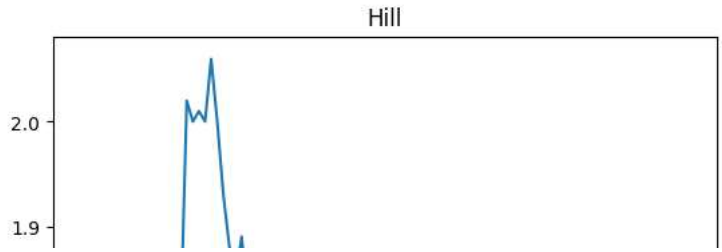
Data Visualisation

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

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




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



Data Preprocessing

```
from sklearn.preprocessing import StandardScaler
```

```
ss = StandardScaler()
```

```
x=ss.fit_transform(x)
```

```
x
```

```
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]])
```

```
x.shape
```

```
(1212, 100)
```

Train Test Spilt

```
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, random_state=2529)
```

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((848, 100), (364, 100), (848,), (364,))
```

Modeling

```
from sklearn.linear_model import LogisticRegression
```

```
lr = LogisticRegression()
```

Model Evaluation

```
lr.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

Prediction

```
y_pred = lr.predict(x_test)
```

```
y_pred.shape
```

```
(364,)
```

```
y_pred
```

```
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])
```

```
lr.predict_proba(x_test)
```

```
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]
```

```
from sklearn.metrics import confusion_matrix, classification_report
```

```
print(confusion_matrix(y_test,y_pred))
```

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

```
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
```

Future Prediction

```
x_new = df.sample()
```

```
x_new
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V92	V93	V94	V95	V96	V97
612	409.53	393.75	388.72	388.45	413.1	406.14	415.73	413.29	411.99	411.11	...	411.53	391.8	400.67	409.16	393.57	405.5

1 rows × 101 columns

```
x_new.shape
```

```
(1, 101)
```

```
x_new = x_new.drop('Class', axis = 1)
```

```
x_new
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V91	V92	V93	V94	V95	V9
612	409.53	393.75	388.72	388.45	413.1	406.14	415.73	413.29	411.99	411.11	...	388.55	411.53	391.8	400.67	409.16	393.5

1 rows × 100 columns

```
x_new.shape
```

```
(1, 100)
```

```
x_new = ss.fit_transform(x_new)
```

```
y_pred_new = lr.predict(x_new)
```

```
y_pred_new
```

```
array([1])
```

```
lr.predict_proba(x_new)
```

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

Explanation: This project is based on the Machine Learning using python programming language. The main purpose of creating this project is to develop a predictive model using logistic regression that can accurately classify geographical locations as either hills or valleys based on a set of input features. The model will be trained on a labeled dataset of geographical features and their corresponding classifications, and then evaluated on a seperate test data set to measure its performance. The ultimate goal of this poject is to provide a useful tool for identifying hills and valleys in various geographical locations, which can have important applications in fields such as geology, agriculture, and urban planning.