Logistic Regression

February 24, 2025

[1]:
```
import pandas as pd
```

[3]:
```
df = pd.read_csv("Social_Network_Ads.csv")
```

[4]:
```
df
```

[4]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
..	…	…	…	…	…
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

[400 rows x 5 columns]

[5]:
```
df = df.drop(columns = 'User ID')   # used to delete a column
```

[7]:
```
df # Print without 'User ID' column
```

[7]:

	Gender	Age	EstimatedSalary	Purchased
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0
..	…	…	…	…
395	Female	46	41000	1
396	Male	51	23000	1
397	Female	50	20000	1
398	Male	36	33000	0
399	Female	49	36000	1

```
[400 rows x 4 columns]
```

[8]:
```
df.isnull().sum()   # For check how many null value available. If want to drop␣
↪null value command -> df.dropna()
```

[8]:
```
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

[9]:
```
# Now we should change all String to number as category ways
```

[10]:
```
df['Gender'] = df['Gender'].astype('category')
df['Gender'] = df['Gender'].cat.codes
```

[11]:
```
df
```

[11]:

	Gender	Age	EstimatedSalary	Purchased
0	1	19	19000	0
1	1	35	20000	0
2	0	26	43000	0
3	0	27	57000	0
4	1	19	76000	0
..	…	…	…	…
395	0	46	41000	1
396	1	51	23000	1
397	0	50	20000	1
398	1	36	33000	0
399	0	49	36000	1

```
[400 rows x 4 columns]
```

[12]:
```
X = df.drop(columns = 'Purchased')  # Pick the independent variables
```

[13]:
```
X
```

[13]:

	Gender	Age	EstimatedSalary
0	1	19	19000
1	1	35	20000
2	0	26	43000
3	0	27	57000
4	1	19	76000
..	…	…	…
395	0	46	41000
396	1	51	23000
397	0	50	20000
398	1	36	33000

```
399       0   49            36000

[400 rows x 3 columns]
```

[14]:
```
Y = df['Purchased']  # Pick the Predicted Variable
```

[15]:
```
Y
```

[15]:
```
0      0
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Purchased, Length: 400, dtype: int64
```

[16]:
```
from sklearn.model_selection import train_test_split
```

[17]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.3,␣
↪random_state = 21) # test_size = 0.3 means 30% will be used for test and 70%␣
↪for training. random_state can be any integer, by using random_state we can␣
↪find same train-test_split
```

[18]:
```
X_train
```

[18]:
```
     Gender  Age  EstimatedSalary
113       1   37            55000
26        1   49            28000
178       1   24            23000
95        0   35            44000
29        1   31            18000
..      …  …              …
368       1   38            71000
48        1   30           135000
260       0   35            77000
312       0   38            50000
207       0   52           114000

[280 rows x 3 columns]
```

[19]:
```
X_test
```

[19]:
```
     Gender  Age  EstimatedSalary
106       0   26            35000
9         0   35            65000
61        1   25            87000
224       0   35            60000
37        1   30            49000
..      …  …              …
23        0   45            22000
157       1   29            75000
349       1   38            61000
255       0   52            90000
180       1   26            16000

[120 rows x 3 columns]
```

[20]:
```
Y_train
```

[20]:
```
113    0
26     1
178    0
95     0
29     0
      ..
368    0
48     1
260    0
312    0
207    0
Name: Purchased, Length: 280, dtype: int64
```

[21]:
```
Y_test
```

[21]:
```
106    0
9      0
61     0
224    0
37     0
      ..
23     1
157    0
349    0
255    1
180    0
Name: Purchased, Length: 120, dtype: int64
```

[22]:
```
# For all numbers into an fixed range we can do some Scalling for pre-processing
```

[23]:
```
from sklearn.preprocessing import StandardScaler
```

[24]:
```
scaler = StandardScaler()
```

[28]:
```
X_train_scaled = scaler.fit_transform(X_train) # Use fit_transform() to know␣
↪the train sets 'standard deviation' and 'mean'
```

[27]:
```
X_test_scaled = scaler.transform(X_test) # Use only transform() so that model␣
↪should not know anything about 'test set'
```

[29]:
```
X_train_scaled
```

[29]:
```
array([[ 0.97182532, -0.11728762, -0.48154649],
       [ 0.97182532,  1.06615502, -1.26799962],
       [ 0.97182532, -1.39935048, -1.41363908],
       [-1.02899151, -0.31452806, -0.80195332],
       [ 0.97182532, -0.70900894, -1.55927855],
       [ 0.97182532, -1.0048696 ,  0.21752295],
       [-1.02899151,  0.4744337 ,  1.79042919],
       [ 0.97182532, -0.21590784, -0.33590703],
       [-1.02899151, -1.0048696 ,  0.36316241],
       [-1.02899151, -0.31452806, -1.41363908],
       [-1.02899151, -0.61038872,  0.4214182 ],
       [ 0.97182532, -1.9910718 , -0.56893017],
       [ 0.97182532, -0.31452806, -1.44276698],
       [ 0.97182532, -1.0048696 ,  1.49915025],
       [ 0.97182532, -1.39935048, -0.48154649],
       [ 0.97182532, -1.30073026,  0.21752295],
       [ 0.97182532, -1.20211004,  0.24665084],
       [-1.02899151,  0.86891458,  1.03310396],
       [ 0.97182532, -0.70900894,  0.07188348],
       [ 0.97182532,  2.15097745,  0.88746449],
       [-1.02899151, -0.61038872,  1.32438289],
       [ 0.97182532, -0.0186674 , -0.0155002 ],
       [ 0.97182532,  1.75649657, -0.33590703],
       [ 0.97182532,  0.17857304,  1.03310396],
       [ 0.97182532, -1.10348982, -1.50102276],
       [-1.02899151,  1.95373701,  0.68356924],
       [ 0.97182532,  0.37581348, -0.21939545],
       [-1.02899151,  0.27719326, -0.59805807],
       [-1.02899151,  1.85511679, -1.12236015],
       [-1.02899151,  0.77029436,  0.07188348],
       [-1.02899151, -0.21590784, -0.51067439],
       [-1.02899151,  2.05235723, -0.86020911],
       [ 0.97182532, -0.31452806, -0.97672068],
       [ 0.97182532,  2.05235723,  2.08170812],
       [-1.02899151, -0.61038872,  1.84868498],
       [-1.02899151, -0.31452806, -0.42329071],
       [-1.02899151, -0.0186674 ,  1.17874343],
```

```
[ 0.97182532, -1.39935048,  0.36316241],
[-1.02899151,  2.15097745,  1.06223185],
[ 0.97182532,  1.06615502,  0.07188348],
[-1.02899151,  0.37581348,  1.06223185],
[-1.02899151,  0.17857304,  2.05258023],
[ 0.97182532, -1.20211004,  0.27577873],
[-1.02899151, -1.20211004,  0.36316241],
[ 0.97182532,  0.07995282, -0.86020911],
[ 0.97182532, -0.31452806,  0.21752295],
[-1.02899151, -1.4979707 , -0.68544175],
[-1.02899151,  1.16477524, -1.03497647],
[ 0.97182532,  0.86891458, -1.50102276],
[-1.02899151, -1.4979707 , -0.16113967],
[ 0.97182532,  1.85511679,  0.07188348],
[ 0.97182532,  0.9675348 , -1.12236015],
[ 0.97182532,  1.95373701,  2.11083602],
[-1.02899151, -0.5117685 ,  1.20787132],
[-1.02899151, -1.10348982, -1.18061594],
[ 0.97182532, -0.21590784, -0.56893017],
[ 0.97182532, -0.90624938,  0.24665084],
[ 0.97182532,  0.86891458, -0.83108121],
[ 0.97182532, -1.69521114,  0.01362769],
[ 0.97182532, -0.61038872, -1.55927855],
[-1.02899151,  0.86891458, -0.65631385],
[ 0.97182532, -0.11728762, -0.53980228],
[ 0.97182532,  0.9675348 ,  0.53792977],
[ 0.97182532,  0.67167414, -1.3262554 ],
[-1.02899151, -1.20211004, -1.64666223],
[ 0.97182532, -0.31452806,  0.10101137],
[ 0.97182532,  0.86891458, -1.41363908],
[-1.02899151,  0.9675348 ,  1.38263868],
[-1.02899151, -1.0048696 , -0.36503492],
[-1.02899151, -0.41314828, -0.83108121],
[-1.02899151, -0.11728762, -0.0155002 ],
[-1.02899151, -0.61038872,  2.28560338],
[-1.02899151, -1.0048696 ,  0.45054609],
[ 0.97182532, -0.31452806, -0.53980228],
[ 0.97182532,  1.06615502,  0.47967399],
[ 0.97182532,  0.07995282,  1.81955708],
[-1.02899151, -1.0048696 ,  0.39229031],
[-1.02899151, -0.0186674 ,  0.24665084],
[ 0.97182532, -0.31452806,  0.47967399],
[ 0.97182532, -1.30073026, -1.44276698],
[-1.02899151,  0.27719326,  0.01362769],
[ 0.97182532, -0.41314828,  1.17874343],
[-1.02899151,  1.55925613, -1.3262554 ],
[-1.02899151, -1.20211004, -0.83108121],
```

```
[-1.02899151,  0.9675348 ,  1.81955708],
[ 0.97182532, -0.31452806,  1.06223185],
[-1.02899151, -1.20211004, -1.58840644],
[ 0.97182532, -0.61038872, -1.55927855],
[-1.02899151, -0.70900894,  1.35351079],
[-1.02899151, -0.31452806,  0.01362769],
[ 0.97182532, -0.11728762, -0.56893017],
[ 0.97182532, -0.11728762,  2.11083602],
[-1.02899151, -1.79383136, -1.03497647],
[ 0.97182532,  1.65787635, -0.94759279],
[ 0.97182532,  0.86891458,  0.97484817],
[-1.02899151,  1.16477524, -1.50102276],
[-1.02899151,  0.67167414, -0.77282543],
[-1.02899151,  0.27719326, -1.20974383],
[ 0.97182532, -1.89245158, -1.3553833 ],
[-1.02899151, -0.11728762, -0.42329071],
[ 0.97182532,  0.9675348 , -0.889337  ],
[-1.02899151,  1.36201569, -1.47189487],
[ 0.97182532, -1.10348982,  0.47967399],
[ 0.97182532,  0.67167414, -1.15148804],
[ 0.97182532,  0.9675348 , -1.12236015],
[-1.02899151, -1.10348982, -0.39416281],
[-1.02899151, -0.70900894, -0.10288388],
[ 0.97182532,  0.67167414,  0.21752295],
[ 0.97182532, -1.79383136,  0.4214182 ],
[-1.02899151,  1.46063591, -1.09323225],
[-1.02899151, -1.20211004,  1.35351079],
[ 0.97182532,  0.17857304, -0.71456964],
[ 0.97182532, -0.5117685 , -1.18061594],
[-1.02899151,  1.95373701, -1.41363908],
[ 0.97182532,  0.37581348,  0.94572028],
[ 0.97182532, -1.9910718 ,  0.30490663],
[ 0.97182532,  0.37581348,  0.24665084],
[-1.02899151, -1.10348982,  0.71269713],
[ 0.97182532,  0.17857304,  0.18839505],
[-1.02899151,  0.4744337 ,  1.17874343],
[-1.02899151,  1.95373701, -0.71456964],
[-1.02899151, -1.10348982,  1.90694076],
[ 0.97182532, -1.59659092,  0.27577873],
[-1.02899151,  0.9675348 ,  0.71269713],
[-1.02899151, -0.21590784,  1.58653393],
[ 0.97182532, -0.31452806, -1.29712751],
[ 0.97182532,  0.77029436,  0.47967399],
[ 0.97182532, -1.89245158,  0.39229031],
[ 0.97182532, -1.0048696 , -0.36503492],
[-1.02899151,  0.27719326,  0.01362769],
[-1.02899151, -0.31452806, -0.0155002 ],
```

```
[ 0.97182532, -0.80762916, -1.64666223],
[ 0.97182532,  0.27719326, -0.77282543],
[ 0.97182532,  0.9675348 ,  0.07188348],
[ 0.97182532,  0.07995282,  1.00397607],
[ 0.97182532, -1.20211004,  0.4214182 ],
[ 0.97182532, -0.11728762, -0.04462809],
[-1.02899151,  0.67167414,  1.7321734 ],
[-1.02899151, -0.0186674 , -0.19026756],
[ 0.97182532,  0.17857304, -0.0155002 ],
[-1.02899151,  0.27719326, -0.24852335],
[-1.02899151, -0.0186674 ,  1.20787132],
[-1.02899151, -0.5117685 , -1.26799962],
[ 0.97182532, -0.0186674 , -0.59805807],
[ 0.97182532,  0.77029436,  1.32438289],
[ 0.97182532, -1.20211004, -1.20974383],
[ 0.97182532,  2.05235723,  1.70304551],
[ 0.97182532, -1.89245158, -1.53015066],
[-1.02899151, -0.5117685 , -0.59805807],
[ 0.97182532, -1.10348982,  0.53792977],
[ 0.97182532, -0.21590784, -1.12236015],
[ 0.97182532,  2.15097745, -1.09323225],
[-1.02899151, -1.59659092, -0.24852335],
[-1.02899151, -1.20211004,  0.24665084],
[-1.02899151,  0.07995282, -0.36503492],
[-1.02899151, -0.11728762,  0.24665084],
[ 0.97182532, -1.0048696 , -1.15148804],
[-1.02899151,  0.37581348,  0.21752295],
[-1.02899151,  0.9675348 ,  1.7321734 ],
[ 0.97182532,  0.17857304, -0.42329071],
[ 0.97182532, -1.89245158, -0.04462809],
[-1.02899151, -1.20211004,  0.01362769],
[-1.02899151, -0.90624938,  0.33403452],
[-1.02899151, -1.10348982, -0.42329071],
[-1.02899151,  0.37581348,  0.10101137],
[ 0.97182532,  0.07995282, -0.86020911],
[-1.02899151,  0.07995282, -0.30677913],
[-1.02899151,  2.15097745, -0.74369753],
[ 0.97182532, -0.5117685 , -0.83108121],
[ 0.97182532,  0.07995282, -0.0155002 ],
[ 0.97182532,  0.37581348, -0.51067439],
[-1.02899151,  0.86891458, -0.71456964],
[ 0.97182532, -0.11728762,  0.15926716],
[ 0.97182532, -0.11728762,  0.07188348],
[ 0.97182532,  1.85511679, -0.33590703],
[-1.02899151, -0.70900894,  0.50880188],
[-1.02899151,  1.26339546,  1.81955708],
[ 0.97182532,  0.07995282,  1.47002236],
```

```
[ 0.97182532, -1.39935048, -0.39416281],
[ 0.97182532, -0.21590784,  1.55740604],
[-1.02899151, -1.4979707 ,  0.30490663],
[ 0.97182532, -0.31452806,  0.56705767],
[ 0.97182532, -1.0048696 ,  0.50880188],
[-1.02899151,  0.9675348 ,  1.93606866],
[-1.02899151, -1.69521114, -1.61753434],
[ 0.97182532,  1.06615502, -1.26799962],
[ 0.97182532, -1.30073026,  0.53792977],
[-1.02899151,  0.86891458, -1.20974383],
[-1.02899151,  0.37581348, -0.04462809],
[-1.02899151,  0.86891458,  1.20787132],
[ 0.97182532, -0.31452806, -1.50102276],
[ 0.97182532,  1.75649657,  1.79042919],
[-1.02899151, -0.41314828, -1.3553833 ],
[-1.02899151,  0.77029436, -0.889337  ],
[-1.02899151, -0.11728762, -1.12236015],
[ 0.97182532, -0.80762916,  0.45054609],
[-1.02899151,  1.16477524, -0.80195332],
[-1.02899151,  1.26339546,  2.1690918 ],
[-1.02899151, -0.21590784, -0.24852335],
[ 0.97182532,  0.77029436, -1.26799962],
[ 0.97182532, -0.41314828,  1.26612711],
[ 0.97182532, -1.10348982, -0.39416281],
[ 0.97182532, -0.11728762,  0.21752295],
[-1.02899151,  1.65787635,  1.70304551],
[-1.02899151,  0.27719326,  0.24665084],
[-1.02899151, -0.21590784, -0.62718596],
[-1.02899151,  2.05235723,  0.33403452],
[ 0.97182532, -1.10348982,  0.50880188],
[ 0.97182532,  0.37581348, -0.19026756],
[ 0.97182532, -0.31452806, -0.30677913],
[-1.02899151,  2.05235723,  0.47967399],
[-1.02899151, -0.5117685 ,  2.25647548],
[ 0.97182532, -1.59659092, -1.55927855],
[ 0.97182532, -1.30073026, -1.12236015],
[-1.02899151, -1.0048696 , -0.80195332],
[-1.02899151, -1.39935048, -1.29712751],
[ 0.97182532, -0.11728762,  0.01362769],
[ 0.97182532,  0.27719326, -0.56893017],
[-1.02899151, -0.70900894, -0.0155002 ],
[-1.02899151,  1.06615502, -1.03497647],
[ 0.97182532, -0.61038872,  1.41176657],
[ 0.97182532, -0.0186674 , -0.30677913],
[ 0.97182532, -0.61038872,  0.82920871],
[-1.02899151,  0.57305392,  1.96519655],
[-1.02899151,  0.77029436,  0.71269713],
```

```
[-1.02899151,  0.57305392, -0.94759279],
[-1.02899151, -0.31452806,  2.1982197 ],
[ 0.97182532,  0.07995282,  0.15926716],
[-1.02899151, -0.11728762, -0.27765124],
[ 0.97182532,  0.37581348, -0.19026756],
[ 0.97182532,  0.17857304, -0.36503492],
[ 0.97182532,  0.77029436,  0.21752295],
[-1.02899151,  0.27719326, -0.33590703],
[ 0.97182532,  2.15097745,  0.33403452],
[ 0.97182532, -0.80762916,  1.03310396],
[ 0.97182532, -0.90624938, -0.83108121],
[ 0.97182532,  1.46063591,  0.01362769],
[-1.02899151,  2.05235723,  0.13013927],
[-1.02899151, -0.0186674 , -0.48154649],
[ 0.97182532,  0.37581348,  2.25647548],
[-1.02899151,  0.86891458,  2.11083602],
[ 0.97182532,  0.27719326, -0.56893017],
[ 0.97182532,  1.06615502,  0.50880188],
[ 0.97182532, -0.80762916,  0.24665084],
[ 0.97182532, -0.21590784,  2.11083602],
[-1.02899151, -0.0186674 , -0.62718596],
[ 0.97182532,  1.26339546, -1.41363908],
[ 0.97182532,  0.17857304, -0.42329071],
[-1.02899151, -1.39935048, -1.15148804],
[-1.02899151,  1.85511679, -1.3262554 ],
[-1.02899151,  0.37581348,  0.24665084],
[-1.02899151, -1.69521114, -0.10288388],
[-1.02899151, -0.90624938, -0.71456964],
[-1.02899151, -0.5117685 , -0.33590703],
[-1.02899151,  1.95373701,  0.8583366 ],
[-1.02899151,  0.77029436, -1.15148804],
[ 0.97182532, -0.90624938, -0.30677913],
[-1.02899151,  0.07995282,  0.10101137],
[-1.02899151, -0.80762916, -0.27765124],
[ 0.97182532,  1.55925613, -0.04462809],
[-1.02899151,  0.77029436,  0.30490663],
[-1.02899151,  0.17857304, -0.33590703],
[-1.02899151,  0.17857304,  0.10101137],
[-1.02899151, -0.31452806, -0.62718596],
[-1.02899151,  1.06615502, -0.94759279],
[ 0.97182532,  1.06615502,  0.4214182 ],
[-1.02899151, -0.11728762,  2.1690918 ],
[ 0.97182532, -0.31452806, -0.48154649],
[-1.02899151,  0.07995282, -0.0155002 ],
[ 0.97182532, -0.21590784,  1.35351079],
[-1.02899151,  0.07995282,  1.81955708],
[ 0.97182532, -0.0186674 , -0.36503492],
```

```
       [ 0.97182532, -0.31452806,  0.04275559],
       [ 0.97182532, -0.31452806,  0.01362769],
       [-1.02899151, -0.21590784,  0.10101137],
       [ 0.97182532, -0.0186674 , -0.0155002 ],
       [ 0.97182532, -0.80762916,  1.84868498],
       [-1.02899151, -0.31452806,  0.15926716],
       [-1.02899151, -0.0186674 , -0.62718596],
       [-1.02899151,  1.36201569,  1.23699921]])
```

[30]:
```
X_test_scaled
```

[30]:
```
array([[-1.02899151, -1.20211004, -1.06410436],
       [-1.02899151, -0.31452806, -0.19026756],
       [ 0.97182532, -1.30073026,  0.45054609],
       [-1.02899151, -0.31452806, -0.33590703],
       [ 0.97182532, -0.80762916, -0.65631385],
       [-1.02899151,  1.16477524,  0.47967399],
       [-1.02899151,  1.06615502,  2.02345234],
       [-1.02899151, -0.90624938,  0.33403452],
       [-1.02899151,  1.55925613,  1.06223185],
       [-1.02899151,  1.46063591,  2.08170812],
       [-1.02899151,  0.07995282,  0.21752295],
       [-1.02899151, -1.0048696 , -1.00584857],
       [-1.02899151,  0.77029436, -1.44276698],
       [ 0.97182532,  0.07995282,  0.71269713],
       [ 0.97182532,  0.27719326,  0.01362769],
       [-1.02899151, -1.79383136,  0.30490663],
       [-1.02899151,  0.07995282,  0.04275559],
       [ 0.97182532,  0.86891458, -1.3553833 ],
       [ 0.97182532,  0.17857304,  0.10101137],
       [-1.02899151, -0.5117685 , -0.07375599],
       [-1.02899151, -1.4979707 , -1.26799962],
       [ 0.97182532, -1.79383136,  0.07188348],
       [ 0.97182532,  0.27719326, -0.36503492],
       [-1.02899151, -1.9910718 , -0.80195332],
       [ 0.97182532, -0.31452806,  0.10101137],
       [ 0.97182532, -0.31452806, -0.62718596],
       [ 0.97182532,  1.36201569,  2.28560338],
       [ 0.97182532,  2.15097745, -0.86020911],
       [-1.02899151, -1.10348982, -0.51067439],
       [-1.02899151, -0.41314828,  0.01362769],
       [-1.02899151, -1.9910718 , -0.10288388],
       [-1.02899151, -1.10348982, -1.58840644],
       [ 0.97182532, -1.79383136, -0.65631385],
       [ 0.97182532,  0.67167414, -1.44276698],
       [ 0.97182532, -1.4979707 , -0.24852335],
       [-1.02899151, -0.5117685 , -0.889337  ],
```

```
[ 0.97182532,  0.17857304, -0.42329071],
[-1.02899151, -0.31452806, -1.3553833 ],
[-1.02899151, -0.11728762,  0.24665084],
[ 0.97182532,  0.77029436, -0.36503492],
[-1.02899151,  1.75649657,  0.94572028],
[-1.02899151, -1.20211004, -0.56893017],
[ 0.97182532, -1.30073026,  0.24665084],
[-1.02899151, -1.39935048, -0.48154649],
[ 0.97182532, -0.90624938, -0.83108121],
[-1.02899151, -1.79383136, -1.41363908],
[ 0.97182532, -1.39935048, -1.53015066],
[-1.02899151,  1.65787635,  1.55740604],
[ 0.97182532,  0.17857304, -0.19026756],
[ 0.97182532,  0.27719326,  0.45054609],
[ 0.97182532,  0.37581348,  0.04275559],
[ 0.97182532, -0.31452806, -0.39416281],
[ 0.97182532, -0.70900894, -0.39416281],
[-1.02899151, -0.70900894, -1.09323225],
[-1.02899151,  0.86891458, -0.59805807],
[ 0.97182532,  0.86891458, -1.09323225],
[-1.02899151,  0.9675348 , -1.20974383],
[ 0.97182532, -0.21590784,  0.80008081],
[-1.02899151, -1.0048696 , -0.48154649],
[ 0.97182532, -0.90624938,  2.22734759],
[ 0.97182532, -0.31452806, -0.94759279],
[ 0.97182532,  0.4744337 ,  1.67391761],
[-1.02899151,  0.9675348 , -1.06410436],
[-1.02899151, -0.61038872,  1.32438289],
[ 0.97182532,  0.27719326,  0.21752295],
[-1.02899151,  1.85511679,  1.47002236],
[-1.02899151, -0.31452806,  0.74182503],
[ 0.97182532, -1.69521114,  0.47967399],
[ 0.97182532, -0.70900894, -0.16113967],
[-1.02899151,  1.95373701, -0.97672068],
[ 0.97182532, -0.31452806, -0.36503492],
[ 0.97182532, -1.20211004, -1.64666223],
[-1.02899151,  0.17857304,  0.01362769],
[ 0.97182532, -1.20211004, -1.15148804],
[ 0.97182532,  0.9675348 ,  2.02345234],
[ 0.97182532, -1.4979707 , -1.50102276],
[ 0.97182532,  0.37581348, -0.51067439],
[ 0.97182532, -0.11728762,  0.10101137],
[-1.02899151, -1.79383136,  0.30490663],
[-1.02899151,  0.86891458, -0.62718596],
[-1.02899151, -1.89245158, -1.3262554 ],
[-1.02899151,  1.36201569, -0.97672068],
[ 0.97182532,  2.15097745, -0.86020911],
```

```
       [ 0.97182532, -0.80762916, -1.58840644],
       [-1.02899151, -0.11728762,  0.62531345],
       [-1.02899151, -1.89245158, -1.47189487],
       [ 0.97182532, -0.41314828, -0.83108121],
       [-1.02899151, -1.10348982,  0.36316241],
       [-1.02899151,  0.27719326, -0.0155002 ],
       [-1.02899151, -1.59659092, -1.29712751],
       [-1.02899151, -1.59659092, -0.48154649],
       [-1.02899151,  1.46063591,  0.94572028],
       [-1.02899151,  0.37581348,  0.53792977],
       [ 0.97182532,  0.77029436, -1.41363908],
       [-1.02899151,  1.55925613,  0.94572028],
       [ 0.97182532, -1.89245158,  0.13013927],
       [-1.02899151, -1.9910718 ,  0.4214182 ],
       [-1.02899151, -0.90624938, -1.26799962],
       [-1.02899151, -0.11728762,  0.18839505],
       [-1.02899151,  2.05235723, -1.23887172],
       [-1.02899151, -1.39935048,  0.50880188],
       [ 0.97182532,  1.06615502, -0.19026756],
       [-1.02899151,  0.37581348, -0.53980228],
       [-1.02899151, -0.31452806, -0.71456964],
       [-1.02899151, -0.80762916,  1.295255  ],
       [-1.02899151,  1.36201569,  1.93606866],
       [-1.02899151, -0.70900894, -1.64666223],
       [-1.02899151, -0.11728762,  1.90694076],
       [-1.02899151,  1.46063591,  0.30490663],
       [-1.02899151, -0.80762916,  0.21752295],
       [-1.02899151,  0.27719326,  0.01362769],
       [ 0.97182532, -0.80762916,  0.50880188],
       [-1.02899151,  0.9675348 , -1.23887172],
       [ 0.97182532,  0.17857304, -0.30677913],
       [ 0.97182532, -0.70900894,  0.13013927],
       [-1.02899151,  0.67167414, -1.44276698],
       [ 0.97182532, -0.90624938,  0.10101137],
       [ 0.97182532, -0.0186674 , -0.30677913],
       [-1.02899151,  1.36201569,  0.53792977],
       [ 0.97182532, -1.20211004, -1.61753434]])
```

[31]:
```
from sklearn.linear_model import LogisticRegression
```

[36]:
```
logistic_reg = LogisticRegression(random_state = 0).fit(X_train_scaled,Y_train)
# Training Complete
```

[34]:
```
logistic_reg.predict(X_train_scaled)
```

[34]:
```
array([0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1], dtype=int64)
```

[35]:
```
logistic_reg.predict(X_test_scaled)
```

[35]:
```
array([0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

[37]:
```
logistic_reg.score(X_train_scaled,Y_train)  # Calculating Accuracy
```

[37]:
```
0.8214285714285714
```

[38]:
```
# 0.8214285714285714 means 82.1428..% accurate
```

[39]:
```
logistic_reg.score(X_test_scaled,Y_test)  # Calculating Accuracy
```

[39]:
```
0.85
```

[40]:
```
# 0.85 = 85% => The model is said to be best the more close accuracy match with␣
↪'train' and 'test'
```

[41]:
```
# We can add some more features for increasing accuracy =>
```

[61]:
```
logistic_reg1 = LogisticRegression(random_state = 0, C = 1, fit_intercept =␣
↪True).fit(X_train_scaled,Y_train) # Model Trained
```

[45]:
```
# C = regularization parameter, Regularization is a technique used in machine␣
↪learning to prevent overfitting. Overfitting happens when a model learns the␣
↪training data too well, including the noise and outliers, which causes it to␣
↪perform poorly on new data. In simple terms, regularization adds a penalty␣
↪to the model for being too complex, encouraging it to stay simpler and more␣
↪general. This way, it's less likely to make extreme predictions based on the␣
↪noise in the data and '1' its default value.
# fit_intercept = is the intersaction (y = mx + c). True means there should be␣
↪intersaction.
```

[62]:
```
logistic_reg1.score(X_train_scaled,Y_train)  # Calculating Accuracy
```

[62]:
```
0.8214285714285714
```

[63]:
```
logistic_reg1.score(X_test_scaled,Y_test)  # Calculating Accuracy
```

[63]:
```
0.85
```

[67]:
```
logistic_reg2 = LogisticRegression(random_state = 0, C = 0.01, fit_intercept =␣
↪True).fit(X_train_scaled,Y_train) # Model Trained
```

[68]:
```
logistic_reg2.score(X_train_scaled,Y_train)  # Calculating Accuracy
```

[68]:
```
0.7607142857142857
```

[69]:
```
logistic_reg2.score(X_test_scaled,Y_test)  # Calculating Accuracy
```

[69]:
```
0.8166666666666667
```

[73]:
```
logistic_reg3 = LogisticRegression(random_state = 0, C = 50, fit_intercept =␣
↪True).fit(X_train_scaled,Y_train) # Model Trained
```

[74]:
```
logistic_reg3.score(X_train_scaled,Y_train)  # Calculating Accuracy
```

[74]:
```
0.8321428571428572
```

[75]:
```
logistic_reg3.score(X_test_scaled,Y_test)  # Calculating Accuracy
```

[75]:
```
0.8416666666666667
```

[76]:
```
# logistic_reg3 is more accurate then all
```