# Logistic Regression

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [2]:

```
df=pd.read_csv(r"C:\Users\91949\Downloads\archive.zip")
df
```

Out[2]:

| | 1 | 0 | 0.99539 | -0.05889 | 0.85243 | 0.02306 | 0.83398 | -0.37708 | 1.1 | 0.03760 | ... | -0.51171 | 0.41078 | -0.46 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 1.00000 | -0.18829 | 0.93035 | -0.36156 | -0.10868 | -0.93597 | 1.00000 | -0.04549 | ... | -0.26569 | -0.20468 | -0.18 |
| **1** | 1 | 0 | 1.00000 | -0.03365 | 1.00000 | 0.00485 | 1.00000 | -0.12062 | 0.88965 | 0.01198 | ... | -0.40220 | 0.58984 | -0.22 |
| **2** | 1 | 0 | 1.00000 | -0.45161 | 1.00000 | 1.00000 | 0.71216 | -1.00000 | 0.00000 | 0.00000 | ... | 0.90695 | 0.51613 | 1.00 |
| **3** | 1 | 0 | 1.00000 | -0.02401 | 0.94140 | 0.06531 | 0.92106 | -0.23255 | 0.77152 | -0.16399 | ... | -0.65158 | 0.13290 | -0.53 |
| **4** | 1 | 0 | 0.02337 | -0.00592 | -0.09924 | -0.11949 | -0.00763 | -0.11824 | 0.14706 | 0.06637 | ... | -0.01535 | -0.03240 | 0.09 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **345** | 1 | 0 | 0.83508 | 0.08298 | 0.73739 | -0.14706 | 0.84349 | -0.05567 | 0.90441 | -0.04622 | ... | -0.04202 | 0.83479 | 0.00 |
| **346** | 1 | 0 | 0.95113 | 0.00419 | 0.95183 | -0.02723 | 0.93438 | -0.01920 | 0.94590 | 0.01606 | ... | 0.01361 | 0.93522 | 0.04 |
| **347** | 1 | 0 | 0.94701 | -0.00034 | 0.93207 | -0.03227 | 0.95177 | -0.03431 | 0.95584 | 0.02446 | ... | 0.03193 | 0.92489 | 0.02 |
| **348** | 1 | 0 | 0.90608 | -0.01657 | 0.98122 | -0.01989 | 0.95691 | -0.03646 | 0.85746 | 0.00110 | ... | -0.02099 | 0.89147 | -0.07 |
| **349** | 1 | 0 | 0.84710 | 0.13533 | 0.73638 | -0.06151 | 0.87873 | 0.08260 | 0.88928 | -0.09139 | ... | -0.15114 | 0.81147 | -0.04 |

350 rows × 35 columns

In [3]:

```
df.head()
```

Out[3]:

| | 1 | 0 | 0.99539 | -0.05889 | 0.85243 | 0.02306 | 0.83398 | -0.37708 | 1.1 | 0.03760 | ... | -0.51171 | 0.41078 | -0.46168 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 1.00000 | -0.18829 | 0.93035 | -0.36156 | -0.10868 | -0.93597 | 1.00000 | -0.04549 | ... | -0.26569 | -0.20468 | -0.18401 |
| **1** | 1 | 0 | 1.00000 | -0.03365 | 1.00000 | 0.00485 | 1.00000 | -0.12062 | 0.88965 | 0.01198 | ... | -0.40220 | 0.58984 | -0.22145 |
| **2** | 1 | 0 | 1.00000 | -0.45161 | 1.00000 | 1.00000 | 0.71216 | -1.00000 | 0.00000 | 0.00000 | ... | 0.90695 | 0.51613 | 1.00000 |
| **3** | 1 | 0 | 1.00000 | -0.02401 | 0.94140 | 0.06531 | 0.92106 | -0.23255 | 0.77152 | -0.16399 | ... | -0.65158 | 0.13290 | -0.53206 |
| **4** | 1 | 0 | 0.02337 | -0.00592 | -0.09924 | -0.11949 | -0.00763 | -0.11824 | 0.14706 | 0.06637 | ... | -0.01535 | -0.03240 | 0.09223 |

5 rows × 35 columns

In [4]:

```
df.tail()
```

Out[4]:

| | 1 | 0 | 0.99539 | -0.05889 | 0.85243 | 0.02306 | 0.83398 | -0.37708 | 1.1 | 0.03760 | ... | -0.51171 | 0.41078 | -0.46168 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **345** | 1 | 0 | 0.83508 | 0.08298 | 0.73739 | -0.14706 | 0.84349 | -0.05567 | 0.90441 | -0.04622 | ... | -0.04202 | 0.83479 | 0.00123 |
| **346** | 1 | 0 | 0.95113 | 0.00419 | 0.95183 | -0.02723 | 0.93438 | -0.01920 | 0.94590 | 0.01606 | ... | 0.01361 | 0.93522 | 0.04925 |
| **347** | 1 | 0 | 0.94701 | -0.00034 | 0.93207 | -0.03227 | 0.95177 | -0.03431 | 0.95584 | 0.02446 | ... | 0.03193 | 0.92489 | 0.02542 |
| **348** | 1 | 0 | 0.90608 | -0.01657 | 0.98122 | -0.01989 | 0.95691 | -0.03646 | 0.85746 | 0.00110 | ... | -0.02099 | 0.89147 | -0.07760 |
| **349** | 1 | 0 | 0.84710 | 0.13533 | 0.73638 | -0.06151 | 0.87873 | 0.08260 | 0.88928 | -0.09139 | ... | -0.15114 | 0.81147 | -0.04822 |

5 rows × 35 columns

In [5]:

```
df.describe()
```

Out[5]:

| | 1 | 0 | 0.99539 | -0.05889 | 0.85243 | 0.02306 | 0.83398 | -0.37708 | 1.1 | ( |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 350.000000 | 350.0 | 350.000000 | 350.000000 | 350.000000 | 350.000000 | 350.000000 | 350.000000 | 350.000000 | 350. |
| **mean** | 0.891429 | 0.0 | 0.640330 | 0.044667 | 0.600350 | 0.116154 | 0.549284 | 0.120779 | 0.510453 | 0. |
| **std** | 0.311546 | 0.0 | 0.498059 | 0.442032 | 0.520431 | 0.461443 | 0.493124 | 0.520816 | 0.507117 | 0. |
| **min** | 0.000000 | 0.0 | -1.000000 | -1.000000 | -1.000000 | -1.000000 | -1.000000 | -1.000000 | -1.000000 | -1. |
| **25%** | 1.000000 | 0.0 | 0.471517 | -0.065388 | 0.412555 | -0.024868 | 0.209105 | -0.053483 | 0.086785 | -0. |
| **50%** | 1.000000 | 0.0 | 0.870795 | 0.016700 | 0.808620 | 0.021170 | 0.728000 | 0.015085 | 0.682430 | 0. |
| **75%** | 1.000000 | 0.0 | 1.000000 | 0.194727 | 1.000000 | 0.335317 | 0.970445 | 0.451572 | 0.950555 | 0. |
| **max** | 1.000000 | 0.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1. |

8 rows × 34 columns

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350 entries, 0 to 349
Data columns (total 35 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   1          350 non-null    int64  
 1   0          350 non-null    int64  
 2   0.99539    350 non-null    float64
 3   -0.05889   350 non-null    float64
 4   0.85243    350 non-null    float64
 5   0.02306    350 non-null    float64
 6   0.83398    350 non-null    float64
 7   -0.37708   350 non-null    float64
 8   1.1        350 non-null    float64
 9   0.03760    350 non-null    float64
 10  0.85243.1  350 non-null    float64
 11  -0.17755   350 non-null    float64
 12  0.59755    350 non-null    float64
 13  -0.44945   350 non-null    float64
 14  0.60536    350 non-null    float64
 15  -0.38223   350 non-null    float64
 16  0.84356    350 non-null    float64
 17  -0.38542   350 non-null    float64
 18  0.58212    350 non-null    float64
 19  -0.32192   350 non-null    float64
 20  0.56971    350 non-null    float64
 21  -0.29674   350 non-null    float64
 22  0.36946    350 non-null    float64
 23  -0.47357   350 non-null    float64
 24  0.56811    350 non-null    float64
 25  -0.51171   350 non-null    float64
 26  0.41078    350 non-null    float64
 27  -0.46168   350 non-null    float64
 28  0.21266    350 non-null    float64
 29  -0.34090   350 non-null    float64
 30  0.42267    350 non-null    float64
 31  -0.54487   350 non-null    float64
 32  0.18641    350 non-null    float64
 33  -0.45300   350 non-null    float64
 34  g          350 non-null    object 
dtypes: float64(32), int64(2), object(1)
memory usage: 95.8+ KB
```

In [7]:

```
df.isna().any()
```

Out[7]:

```
1             False
0             False
0.99539       False
-0.05889      False
0.85243       False
0.02306       False
0.83398       False
-0.37708      False
1.1           False
0.03760       False
0.85243.1     False
-0.17755      False
0.59755       False
-0.44945      False
0.60536       False
-0.38223      False
0.84356       False
-0.38542      False
0.58212       False
-0.32192      False
0.56971       False
-0.29674      False
0.36946       False
-0.47357      False
0.56811       False
-0.51171      False
0.41078       False
-0.46168      False
0.21266       False
-0.34090      False
0.42267       False
-0.54487      False
0.18641       False
-0.45300      False
g             False
dtype: bool
```

In [8]:

```
x=df.iloc[:,[2,3]].values
y=df.iloc[:,4].values
```

In [9]:

```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [10]:

```
print('The DataFrame has %d Rows and %d columns'%(df.shape))
```

```
The DataFrame has 350 Rows and 35 columns
```

In [15]:

```
print('The Features matrix has %d Rows and %d column(s)'%(feature_matrix.shape))
print('The target matrix has %d Rows and %d column(s)'%(np.array(target_vector).reshape(-1,1).shape))
```

```
The Features matrix has 350 Rows and 34 column(s)
The target matrix has 350 Rows and 1 column(s)
```

In [17]:

```
features_matrix = df.iloc[:,0:34]
```

In [18]:

```
target_vector = df.iloc[:,-1]
```

In [19]:

```
print('The Features Matrix Has %d Rows And %d columns(s)'%(features_matrix.shape))
```

```
The Features Matrix Has 350 Rows And 34 columns(s)
```

In [20]:

```
print('The Target Matrix Has %d Rows And %d Columns(s)'%(np.array(target_vector).reshape(-1, 1).shape)
```

```
The Target Matrix Has 350 Rows And 1 Columns(s)
```

In [21]:

```
features_matrix_standardized = StandardScaler().fit_transform(features_matrix)
```

In [22]:

```
algorithm = LogisticRegression(penalty=None,dual=False, tol=1e-4,C=1.0, fit_intercept=True,intercept_s
class_weight=None,random_state=None,solver='lbfgs',max_iter=10000,
multi_class='auto',verbose=0, warm_start=False, n_jobs=None,l1_ratio=None)
```

In [23]:

```
Logistic_Regression_Model = algorithm.fit(features_matrix_standardized,target_vector)
```

In [24]:

```
observation = [[1, 0, 0.99539, -0.05889, 0.8524299999999999, 0.02306, 0.8339799999999999, -0.37708, 1.
0.8524299999999999, -0.17755, 0.59755, -0.44945, 0.60536, -0.38223, 0.8435600000000001, -0.38542,
0.58212, -0.32192, 0.56971, -0.29674, 0.36946, -0.47357, 0.56811, -0.51171, 0.4107800000000003,
-0.4616800000000003, 0.21266, -0.3409,0.112267,-0.54487,0.18641,-0.453]]
```

In [25]:

```
predictions = Logistic_Regression_Model.predict(observation)
print('The Model predicted The observation To Belong To Class %s'%(predictions))
```

```
The Model predicted The observation To Belong To Class ['g']
```

In [26]:

```
print('The Algorithm Was Trained To predict The One Of The Classes: %s'%(algorithm.classes_))
```

```
The Algorithm Was Trained To predict The One Of The Classes: ['b' 'g']
```

In [27]:

```
print("""The Model Says The Probability Of The observation We Passed belonging To The Class ['b'] is %
%(algorithm.predict_proba(observation)[0][0]))
print()
```

```
The Model Says The Probability Of The observation We Passed belonging To The Class ['b'] is
2.657963595609214e-05
```

In [28]:

```
print("""The Model Says The Probability Of The observation We Passed belonging To The Class ['g'] is %
%(algorithm.predict_proba(observation)[0][1]))
```

```
The Model Says The Probability Of The observation We Passed belonging To The Class ['g'] is
0.9999734203640439
```