Inosphere

In [1]:
```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [2]:
```python
df=pd.read_csv(r"C:\Users\anu\Downloads\archive (1).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	-0.46168	0.21266	-0.34090	0.42267	-0.54487
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	-0.19040	-0.11593	-0.16626	-0.06288
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	0.43100	-0.17365	0.60436	-0.24180
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	1.00000	-0.20099	0.25682	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	0.02431	-0.62197	-0.05707	-0.59573
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	-0.07859	0.00732	0.00000	0.00000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	1.00000	0.12815	0.86660	-0.10714
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	0.93159	0.08168	0.94066	-0.00035
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	0.92120	0.02242	0.92459	0.00442
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	0.82983	-0.17238	0.96022	-0.03757
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	0.78207	-0.00703	0.75747	-0.06678

350 rows × 35 columns

In [3]:
```python
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [4]:
```python
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

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

In [5]:
```python
df.head()
```

Out[5]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243.1	-0.17755	0.59755	-0.44945	0.60536	-0.38223	0.84356	-0.385
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.67743	0.34432	-0.69707	-0.51685	-0.97515	0.05499	-0.622
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.05346	0.85443	0.00827	0.54591	0.00299	0.83775	-0.136
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	-1.00000	0.14516	0.54094	-0.393
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.20275	0.56409	-0.00712	0.34395	-0.27457	0.52940	-0.217
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	-0.06302	0.00000	0.00000	-0.04572	-0.15540	-0.00343	-0.101

In [6]:
```python
features_matrix = df.iloc[:,0:34]
```

In [7]:
```python
target_vector = df.iloc[:,-1]
```

In [8]:
```python
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 [9]:
```python
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 [10]:
```python
features_matrix_standardized = StandardScaler().fit_transform(features_matrix)
```

In [11]:
```python
algorithm = LogisticRegression(penalty=None,dual=False, tol=1e-4,C=1.0, fit_intercept=True,intercept_scaling=1,
                               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 [12]:
```python
Logistic_Regression_Model = algorithm.fit(features_matrix_standardized,target_vector)
```

```
In [14]: observation = [[1, 0, 0.99539, -0.05889, 0.8524299999999999, 0.02306, 0.8339799999999999, -0.37708, 1.0, 0.0376,
                         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 [15]: 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 [16]: 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 [17]: print("""The Model Says The Probability Of The observation We Passed belonging To The Class ['b'] is %s"""
               %(algorithm.predict_proba(observation)[0][0]))
         print()
```

```
The Model Says The Probability Of The observation We Passed belonging To The Class ['b'] is 2.8223920200298735e-05
```

```
In [18]: print("""The Model Says The Probability Of The observation We Passed belonging To The Class ['g'] is %s"""
               %(algorithm.predict_proba(observation)[0][1]))
```

```
The Model Says The Probability Of The observation We Passed belonging To The Class ['g'] is 0.9999717760797997
```

```
In [ ]:
```