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\HP\OneDrive\Documents\Ionospear.csv")
df
```

Out[2]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.0376	...
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...
...	...	...	...	...	...	...	...	...	...	...	...
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...

350 rows × 35 columns

In [3]:

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

In [4]:

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

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

In [5]:

```
df.head()
```

Out[5]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.0376	0.85243
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.5087
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.7308
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.0000
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.5279
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.0378

In [6]:

```
feature_matrix=df.iloc[:,0:34]
target_vector=df.iloc[:,-1]
```

In [7]:

```
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).res
```

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

In [8]:

```
feature_matrix_standardized = StandardScaler().fit_transform(feature_matrix)
```

In [9]:

```
algorithm = LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_intercept
```

In [10]:

```
Logistic_Regression_Model = algorithm.fit(feature_matrix_standardized,target_vector
```

In [11]:

```
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,
               -0.38542,0.58219,-0.32192,0.56971,-0.29674,0.36946,-0.47357,
              0.56811,-0.51171,0.41078000000000003,-0.46168000000000003,0.21266,
               -0.3409,0.42267,-0.54487,0.18641,-0.453]]
```

In [12]:

```
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 [13]:

```
print('Algorithm was Trained To predict one of the two classes : %s'%(algorithm.cla
```

```
Algorithm was Trained To predict one of the two classes : ['b' 'g']
```

In [14]:

```
print("""The Model ssays the probability of the observation we passed Belonging To (
print()
print("""The Model ssays the probability of the observation we passed Belonging To (
```

```
The Model ssays the probability of the observation we passed Belonging To
class['b'] Is 0.008044378633976335

The Model ssays the probability of the observation we passed Belonging To
class['g'] Is 0.9919556213660237
```

In [19]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
feature=df.columns[0:3]
target=df.columns[-1]
x=df[feature].values
y=df[target].values
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
lr = LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x-test,y_test))
print(lr.score(x_train,y_train))
```

```
---------------------------------------------------------------------
--
ValueError                                Traceback (most recent call las
t)
Cell In[19], line 9
      7 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
25)
      8 lr = LinearRegression()
----> 9 lr.fit(x_train,y_train)
     10 print(lr.score(x-test,y_test))
     11 print(lr.score(x_train,y_train))

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\linear_model\_base.py:648, in LinearRegression.fit(self, X, y, sample_we
ight)
    644 n_jobs_ = self.n_jobs
    646 accept_sparse = False if self.positive else ["csr", "csc", "coo"]
--> 648 X, y = self._validate_data(
    649     X, y, accept_sparse=accept_sparse, y_numeric=True, multi_outp
ut=True
    650 )
    652 sample_weight = _check_sample_weight(
    653     sample_weight, X, dtype=X.dtype, only_non_negative=True
    654 )
    656 X, y, X_offset, y_offset, X_scale = _preprocess_data(
    657     X,
    658     y,
   (...)
    661     sample_weight=sample_weight,
    662 )

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\base.py:584, in BaseEstimator._validate_data(self, X, y, reset, validate
_separately, **check_params)
    582         y = check_array(y, input_name="y", **check_y_params)
    583     else:
--> 584         X, y = check_X_y(X, y, **check_params)
    585     out = X, y
    587 if not no_val_X and check_params.get("ensure_2d", True):

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\utils\validation.py:1122, in check_X_y(X, y, accept_sparse, accept_large
_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi
_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
   1102     raise ValueError(
   1103         f"{estimator_name} requires y to be passed, but the targe
t y is None"
   1104     )
   1106 X = check_array(
   1107     X,
   1108     accept_sparse=accept_sparse,
   (...)
   1119     input_name="X",
   1120 )
-> 1122 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric, e
stimator=estimator)
   1124 check_consistent_length(X, y)
   1126 return X, y

File ~\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\utils\validation.py:1147, in _check_y(y, multi_output, y_numeric, estima
```

```
tor)
   1145     _ensure_no_complex_data(y)
   1146 if y_numeric and y.dtype.kind == "O":
-> 1147     y = y.astype(np.float64)
   1149 return y

ValueError: could not convert string to float: 'g'
```

In [ ]:

```
1
```