import Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

import Linear Regression

In [2]:

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

In [3]:

```
lgr=LogisticRegression()
```

Select Required data from certain columns

In [4]:

```
a=pd.read_csv("iono.csv")
a
```

Out[4]:

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

350 rows × 35 columns

In [5]:

```
fm=a.iloc[:,0:2]
tv=a.iloc[:,-1]
```

Shape

In [6]:

```
fm.shape
```

Out[6]:

```
(350, 2)
```

In [7]:

```
tv.shape
```

Out[7]:

```
(350,)
```

To make the data in order (feature matrix)

from sklearn.preprocessing import StandardScaler

fs=StandardScaler().fit_transform(fm)

Imply Logistic Regression

In [8]:

```
lgr.fit(fm,tv)
```

Out[8]:

```
LogisticRegression()
```

Prediction

In [9]:

```
ab=[[3,90]]
```

In [10]:

```
pre=lgr.predict(ab)
```

In [11]:

```
print(pre)
```

```
['g']
```

To check the output var we have got

In [12]:

```
lgr.classes_
```

Out[12]:

```
array(['b', 'g'], dtype=object)
```

Prediction in Probablity value

In [14]:

```
lgr.predict_proba(ab)[0][1]
```

Out[14]:

```
0.9993816276539712
```

Logistic Regression-2:

In [40]:

```
import re
import numpy as np
import pandas as pd
from sklearn.datasets import load_digits
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

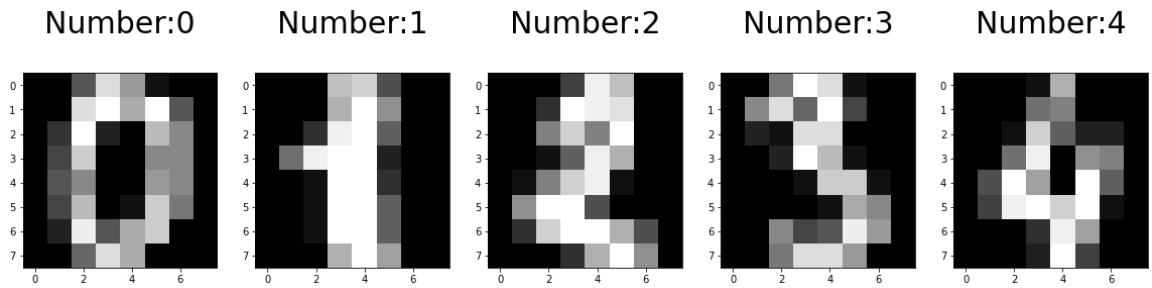
In [30]:

```
dig=load_digits()
dig
```

```
       ...,

       [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ...,  2.,  1.,  0.],
        [ 0.,  0., 16., ..., 16.,  5.,  0.],
        ...,
        [ 0.,  0., 16., ..., 15.,  0.,  0.],
        [ 0.,  0., 15., ..., 16.,  0.,  0.],
        [ 0.,  0.,  2., ...,  6.,  0.,  0.]],

       [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
        [ 0.,  0., 14., ..., 15.,  1.,  0.],
        [ 0.,  4., 16., ..., 16.,  7.,  0.],
        ...,
        [ 0.,  0.,  0., ..., 16.,  2.,  0.],
        [ 0.,  0.,  4., ..., 16.,  2.,  0.],
        [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

       [[ 0.,  0., 10., ...,  1.,  0.,  0.],
        [ 0.,  2., 16., ...,  1.,  0.,  0.],
```

In [33]:

```
plt.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(dig.data[0:5],dig.target[0:5])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title('Number:%i\n'%label,fontsize=30)
```



In [35]:

```
x_train,x_test,y_train,y_test=train_test_split(dig.data,dig.target,test_size=0.30)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(1257, 64)
(1257,)
(540, 64)
(540,)
```

Fit the model

In [43]:

```
lgre=LogisticRegression(max_iter=10000)
lgre.fit(x_train,y_train)
```

Out[43]:

```
LogisticRegression(max_iter=10000)
```

Confusion Matrix

In [44]:

```
print(lgre.predict(x_test))
```

```
[1 9 1 7 2 6 0 9 6 6 5 0 5 2 5 5 9 7 2 6 7 7 5 1 9 4 2 7 9 4 4 7 1 5 5 8 5
 2 9 9 1 5 0 2 4 3 0 5 2 4 3 2 7 0 2 4 3 2 8 8 2 6 2 7 4 5 6 3 0 6 5 8 0 5
 5 2 3 3 8 7 8 8 7 7 2 1 9 6 6 0 5 9 0 0 6 9 1 4 9 3 9 0 0 9 0 5 3 4 2 7 8
 8 3 6 3 5 5 3 0 3 3 5 7 5 9 5 2 4 2 6 6 8 7 7 3 3 1 3 4 9 9 0 0 5 6 8 3 9
 4 2 6 5 9 8 4 3 6 2 2 6 6 9 7 5 4 1 9 8 5 7 4 7 5 0 7 2 5 8 2 4 7 2 4 4 7
 8 2 0 0 1 7 2 6 4 0 7 0 1 5 1 3 4 6 7 9 0 4 8 4 6 0 2 7 5 1 2 1 6 3 1 9 6
 4 9 6 1 9 3 4 4 4 1 2 7 6 2 7 3 4 7 6 2 4 1 0 8 0 5 0 3 1 6 4 0 4 9 1 4 0
 9 3 7 9 0 6 8 1 9 8 3 0 7 0 7 1 6 7 7 1 7 8 5 7 5 5 1 5 0 4 3 4 7 1 1 0 1
 6 9 6 5 8 5 7 3 3 8 4 8 2 3 8 3 6 7 4 5 3 6 5 1 6 3 5 5 0 2 4 4 6 1 6 8 3
 8 7 0 6 6 7 3 2 2 6 9 9 8 0 3 7 3 5 9 1 7 2 5 7 4 5 2 0 7 1 2 6 7 7 2 1 0
 8 6 7 1 5 9 1 5 4 4 7 3 3 2 0 9 3 2 4 3 4 6 8 9 1 5 6 8 7 0 1 6 2 7 4 7 2
 4 1 6 1 8 9 3 9 8 9 1 2 0 9 5 4 4 2 9 1 8 8 1 8 4 9 4 9 4 8 3 2 5 5 4 7 1
 3 2 6 2 5 9 7 9 4 3 2 8 8 6 5 7 1 2 8 0 2 9 2 0 4 3 8 2 7 4 0 0 8 6 6 6 8
 0 2 4 2 8 9 5 6 0 2 9 9 4 2 5 7 7 1 3 1 6 8 4 3 6 9 5 3 8 2 5 1 8 9 8 8 0
 6 5 1 3 2 9 8 0 2 2 5 8 9 8 7 7 9 7 6 1 6 9]
```

Accuracy

In [45]:

```
print(lgre.score(x_test,y_test))
```

```
0.9611111111111111
```