Notebook

February 17, 2025

[1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

1 Data Loading And Preprocessing

[3]:
```
sonar=pd.read_csv(r"C:\Users\arjun\Downloads\Copy of sonar data.
↪csv",header=None)
sonar.head()
```

[3]:
```
       0       1       2       3       4       5       6       7       8   \
0  0.0200  0.0371  0.0428  0.0207  0.0954  0.0986  0.1539  0.1601  0.3109
1  0.0453  0.0523  0.0843  0.0689  0.1183  0.2583  0.2156  0.3481  0.3337
2  0.0262  0.0582  0.1099  0.1083  0.0974  0.2280  0.2431  0.3771  0.5598
3  0.0100  0.0171  0.0623  0.0205  0.0205  0.0368  0.1098  0.1276  0.0598
4  0.0762  0.0666  0.0481  0.0394  0.0590  0.0649  0.1209  0.2467  0.3564

       9   …      51      52      53      54      55      56      57  \
0  0.2111  …  0.0027  0.0065  0.0159  0.0072  0.0167  0.0180  0.0084
1  0.2872  …  0.0084  0.0089  0.0048  0.0094  0.0191  0.0140  0.0049
2  0.6194  …  0.0232  0.0166  0.0095  0.0180  0.0244  0.0316  0.0164
3  0.1264  …  0.0121  0.0036  0.0150  0.0085  0.0073  0.0050  0.0044
4  0.4459  …  0.0031  0.0054  0.0105  0.0110  0.0015  0.0072  0.0048

       58      59 60
0  0.0090  0.0032  R
1  0.0052  0.0044  R
2  0.0095  0.0078  R
3  0.0040  0.0117  R
4  0.0107  0.0094  R

[5 rows x 61 columns]
```

[5]:
```
sonar[60].value_counts()
```

[5]:
```
60
M    111
R     97
Name: count, dtype: int64
```

[7]:
```
sonar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 208 entries, 0 to 207
Data columns (total 61 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       208 non-null    float64
 1   1       208 non-null    float64
 2   2       208 non-null    float64
 3   3       208 non-null    float64
 4   4       208 non-null    float64
 5   5       208 non-null    float64
 6   6       208 non-null    float64
 7   7       208 non-null    float64
 8   8       208 non-null    float64
 9   9       208 non-null    float64
 10  10      208 non-null    float64
 11  11      208 non-null    float64
 12  12      208 non-null    float64
 13  13      208 non-null    float64
 14  14      208 non-null    float64
 15  15      208 non-null    float64
 16  16      208 non-null    float64
 17  17      208 non-null    float64
 18  18      208 non-null    float64
 19  19      208 non-null    float64
 20  20      208 non-null    float64
 21  21      208 non-null    float64
 22  22      208 non-null    float64
 23  23      208 non-null    float64
 24  24      208 non-null    float64
 25  25      208 non-null    float64
 26  26      208 non-null    float64
 27  27      208 non-null    float64
 28  28      208 non-null    float64
 29  29      208 non-null    float64
 30  30      208 non-null    float64
 31  31      208 non-null    float64
 32  32      208 non-null    float64
 33  33      208 non-null    float64
 34  34      208 non-null    float64
 35  35      208 non-null    float64
 36  36      208 non-null   float64
 37  37      208 non-null   float64
 38  38      208 non-null   float64
 39  39      208 non-null   float64
 40  40      208 non-null   float64
 41  41      208 non-null   float64
 42  42      208 non-null   float64
 43  43      208 non-null   float64
 44  44      208 non-null   float64
 45  45      208 non-null   float64
 46  46      208 non-null   float64
 47  47      208 non-null   float64
 48  48      208 non-null   float64
 49  49      208 non-null   float64
 50  50      208 non-null   float64
 51  51      208 non-null   float64
 52  52      208 non-null   float64
 53  53      208 non-null   float64
 54  54      208 non-null   float64
 55  55      208 non-null   float64
 56  56      208 non-null   float64
 57  57      208 non-null   float64
 58  58      208 non-null   float64
 59  59      208 non-null   float64
 60  60      208 non-null   object
dtypes: float64(60), object(1)
memory usage: 99.2+ KB
```

[11]:
```
x=sonar.drop(columns=60,axis=1)
y=sonar[60]
```

2 Training And Testing

[12]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪2,random_state=1,stratify=y)
```

3 Model Training

[16]:
```
model=LogisticRegression()
model.fit(x_train,y_train)
```

[16]:
```
LogisticRegression()
```

4 Model Evaluation

[22]:
```
y_pred=model.predict(x_train)
y_pred
```

[22]:
```
array(['M', 'R', 'M', 'M', 'R', 'R', 'R', 'M', 'R', 'M', 'R', 'R', 'M',
       'M', 'M', 'M', 'R', 'M', 'R', 'R', 'M', 'R', 'R', 'R', 'R', 'R',
       'M', 'M', 'R', 'R', 'M', 'M', 'R', 'R', 'R', 'M', 'M', 'R', 'R',
       'R', 'M', 'M', 'M', 'M', 'M', 'M', 'R', 'M', 'R', 'M', 'R', 'M',
       'M', 'M', 'M', 'R', 'M', 'M', 'M', 'M', 'R', 'R', 'R', 'M', 'R',
       'R', 'M', 'M', 'R', 'R', 'M', 'R', 'M', 'R', 'M', 'M', 'M', 'M',
       'M', 'R', 'M', 'R', 'M', 'R', 'R', 'M', 'M', 'M', 'R', 'R', 'M',
       'M', 'R', 'M', 'R', 'R', 'M', 'M', 'M', 'M', 'M', 'M', 'R', 'R',
       'M', 'R', 'R', 'R', 'M', 'R', 'M', 'R', 'M', 'M', 'M', 'M', 'R',
       'M', 'M', 'M', 'R', 'R', 'M', 'R', 'R', 'R', 'R', 'M', 'R', 'M',
       'R', 'M', 'R', 'R', 'M', 'M', 'M', 'R', 'M', 'R', 'M', 'R', 'M',
       'R', 'R', 'R', 'R', 'M', 'M', 'M', 'R', 'M', 'M', 'R', 'M', 'R',
       'R', 'M', 'M', 'M', 'M', 'R', 'M', 'M', 'M', 'M'], dtype=object)
```

[23]:
```
accuracy=accuracy_score(y_pred,y_train)
accuracy
```

[23]: 0.8433734939759037

[24]:
```
input_data = (0.0307,0.0523,0.0653,0.0521,0.0611,0.0577,0.0665,0.0664,0.1460,0.
 ↪2792,0.3877,0.4992,0.4981,0.4972,0.5607,0.7339,0.8230,0.9173,0.9975,0.9911,0.
 ↪8240,0.6498,0.5980,0.4862,0.3150,0.1543,0.0989,0.0284,0.1008,0.2636,0.2694,0.
 ↪2930,0.2925,0.3998,0.3660,0.3172,0.4609,0.4374,0.1820,0.3376,0.6202,0.4448,0.
 ↪1863,0.1420,0.0589,0.0576,0.0672,0.0269,0.0245,0.0190,0.0063,0.0321,0.0189,0.
 ↪0137,0.0277,0.0152,0.0052,0.0121,0.0124,0.0055)

# changing the input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the np array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]=='R'):
  print('The object is a Rock')
else:
  print('The object is a mine')
```

```
['M']
The object is a mine
```

This notebook was converted with convert.ploomber.io