Experiment-9

9a) Consider play tennis dataset build,test,evaluate the logestic regression for binary classification

b)Consider Fish dataset build,test,evaluate the logestic regression for multi-class classification

logistic Regression for binary classification

In [1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

In [2]:

```
df = pd.read_csv("C:/Users/Guru Kiran/All CSV files/Play Tennis.csv")
df.head()
```

Out[2]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes

In [3]:

```
# 2. Preprocessing
x = df.drop(columns=['Day','Play_Tennis'])   # features
y = df['Play_Tennis']                        # target
```

In [4]:

```
# One-hot encode categorical features
x = pd.get_dummies(x, drop_first=True)
x
```

Out[4]:

	Outlook_Rain	Outlook_Sunny	Temprature_Hot	Temprature_Mild	Humidity_Normal
0	False	True	True	False	False
1	False	True	True	False	False
2	False	False	True	False	False
3	True	False	False	True	False
4	True	False	False	False	True
5	True	False	False	False	True
6	False	False	False	False	True
7	False	True	False	True	False
8	False	True	False	False	True
9	True	False	False	True	True
10	False	True	False	True	True
11	False	False	False	True	False
12	False	False	True	False	True
13	True	False	False	True	False

In [5]:

```
# Split into Train/Test
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,random_s
```

In [6]:

```
# ----------------------------
# 3. Logistic Regression Model
# ----------------------------
log_reg = LogisticRegression()
log_reg.fit(x_train, y_train)

# Predictions
y_pred = log_reg.predict(x_test)
# y_prob = log_reg.predict_proba(x_test)
```

In [7]:

```
# 4. Evaluation
# ----------------------------
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", metrics.confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.6

Confusion Matrix:
 [[0 2]
 [0 3]]
```

In [8]:

```
print("\nClassification Report:\n",
      metrics.classification_report(y_test, y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

          No       0.00      0.00      0.00         2
         Yes       0.60      1.00      0.75         3

    accuracy                           0.60         5
   macro avg       0.30      0.50      0.38         5
weighted avg       0.36      0.60      0.45         5
```

```
C:\Users\Guru Kiran\AppData\Local\Programs\Python\Python313\Lib\site-packages\skl
earn\metrics\_classification.py:1706: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
C:\Users\Guru Kiran\AppData\Local\Programs\Python\Python313\Lib\site-packages\skl
earn\metrics\_classification.py:1706: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
C:\Users\Guru Kiran\AppData\Local\Programs\Python\Python313\Lib\site-packages\skl
earn\metrics\_classification.py:1706: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
```

b) a) Consider Fish dataset build,test,evaluate the logestic regression for multi-class classification

Logistic Regression for multi class classification

In [9]:

```python
import pandas as pd
df1 = pd.read_csv('C:/Users/Guru Kiran/All CSV files/dataset_Fish.csv')
df1.head()
```

Out[9]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

In [10]:

```python
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159 entries, 0 to 158
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Species  159 non-null    object 
 1   Weight   159 non-null    float64
 2   Length1  159 non-null    float64
 3   Length2  159 non-null    float64
 4   Length3  159 non-null    float64
 5   Height   159 non-null    float64
 6   Width    159 non-null    float64
dtypes: float64(6), object(1)
memory usage: 8.8+ KB
```

In [11]:
```
x = df1.drop('Species',axis=1)
y = df1['Species']
```

In [12]:
```
x.head()
```

Out[12]:

	Weight	Length1	Length2	Length3	Height	Width
0	242.0	23.2	25.4	30.0	11.5200	4.0200
1	290.0	24.0	26.3	31.2	12.4800	4.3056
2	340.0	23.9	26.5	31.1	12.3778	4.6961
3	363.0	26.3	29.0	33.5	12.7300	4.4555
4	430.0	26.5	29.0	34.0	12.4440	5.1340

In [13]:
```
### Scaling the input features using MinMaxScaler
```

In [14]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(x)
x_scaled = scaler.transform(x)
```

In [15]:
```
x_scaled[0:5]
```

Out[15]:
```
array([[0.14666667, 0.30485437, 0.30909091, 0.35810811, 0.56833405,
        0.41897835],
       [0.17575758, 0.32038835, 0.32545455, 0.37837838, 0.62405535,
        0.45923545],
       [0.20606061, 0.3184466 , 0.32909091, 0.37668919, 0.61812335,
        0.51427887],
       [0.22      , 0.36504854, 0.37454545, 0.41722973, 0.63856611,
        0.48036479],
       [0.26060606, 0.36893204, 0.37454545, 0.42567568, 0.6219658 ,
        0.57600361]])
```

In [16]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x_scaled, y, test_size=0.2, r
```

In [17]:
```
from sklearn.linear_model import LogisticRegression
log_Reg = LogisticRegression()
# training the model
log_Reg.fit(x_train, y_train)
```

```
y_pred = log_Reg.predict(x_test)
```

In [18]:
```
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", metrics.confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", metrics.classification_report(y_test, y_pred
```

```
Accuracy: 0.8125

Confusion Matrix:
 [[10  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0]
 [ 0  0  9  0  0  0  0]
 [ 0  0  1  2  0  0  0]
 [ 0  0  1  0  0  0  0]
 [ 0  0  0  0  0  5  0]
 [ 0  0  3  0  0  0  0]]

Classification Report:
               precision    recall  f1-score   support

       Bream       1.00      1.00      1.00        10
      Parkki       0.00      0.00      0.00         1
       Perch       0.60      1.00      0.75         9
        Pike       1.00      0.67      0.80         3
       Roach       0.00      0.00      0.00         1
       Smelt       1.00      1.00      1.00         5
   Whitefish       0.00      0.00      0.00         3

    accuracy                           0.81        32
   macro avg       0.51      0.52      0.51        32
weighted avg       0.73      0.81      0.75        32
```

```
C:\Users\Guru Kiran\AppData\Local\Programs\Python\Python313\Lib\site-packages\skl
earn\metrics\_classification.py:1706: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
C:\Users\Guru Kiran\AppData\Local\Programs\Python\Python313\Lib\site-packages\skl
earn\metrics\_classification.py:1706: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
C:\Users\Guru Kiran\AppData\Local\Programs\Python\Python313\Lib\site-packages\skl
earn\metrics\_classification.py:1706: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
```