Experiment-10

10.Consider Fish dataset build,test,evaluate, SVM for Multi Class Classification

In [1]:

```
import pandas as pd
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn import metrics
fish = pd.read_csv('C:/Users/Guru Kiran/All CSV files/dataset_Fish.csv')
fish.head()
```

Out[1]:

	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 [2]:

```
fish['Species'].unique()
```

Out[2]:

```
array(['Bream', 'Roach', 'Whitefish', 'Parkki', 'Perch', 'Pike', 'Smelt'],
      dtype=object)
```

In [3]:

```
fish.isnull().sum()
```

Out[3]:

```
Species    0
Weight     0
Length1    0
Length2    0
Length3    0
Height     0
Width      0
dtype: int64
```

In [4]:

```
X = fish.drop('Species',axis=1)
y = fish['Species']
```

In [5]:

```
X.head()
```

Out[5]:

	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 [6]:

```
## Scaling the input features using MinMaxScaler
```

In [7]:

```
scaler = MinMaxScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
```

In [8]:

```
###Splitting into train and test datasets using train_test_split
```

In [9]:

```
X_train, X_test, y_train, y_test= train_test_split(X_scaled, y, test_size=0.2, r
```

In [10]:

```
# -----------------------------
# 4. SVM Model
# -----------------------------
svm = SVC()
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
```

In [11]:

```
# -----------------------------
# 5. Predictions & Evaluation
# -----------------------------
y_pred = svm.predict(X_test)
```

In [12]:

```
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: 0.875

Confusion Matrix:
 [[10  0  0  0  0  0  0]
 [ 0  1  0  0  0  0  0]
 [ 0  0  9  0  0  0  0]
 [ 0  0  0  3  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       1.00      1.00      1.00         1
       Perch       0.69      1.00      0.82         9
        Pike       1.00      1.00      1.00         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                           0.88        32
   macro avg       0.67      0.71      0.69        32
weighted avg       0.79      0.88      0.82        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])
```