A6:Implement a random forest classifier using sci-kit-learn to classify a dataset and understand ensemble learning concepts.

Understanding Ensemble Learning

Ensemble learning combines multiple models (called base learners) to produce better results than any single model alone. Random Forest is an ensemble of Decision Trees.

Key Concepts:

Bagging (Bootstrap Aggregation): Each decision tree is trained on a random subset of data (with replacement). This reduces overfitting.

Random Feature Selection: At each split, only a random subset of features is considered. This introduces diversity among trees.

Voting: For classification, each tree votes for a class, and the majority vote becomes the final prediction.

In [12]:
```
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

In [13]:
```
# Step 1: Load a sample dataset (Iris dataset)
iris = load_iris()
X = iris.data        # features
y = iris.target      # target labels
```

In [14]:
```
X
```

```
Out[14]:  array([[5.1, 3.5, 1.4, 0.2],
                 [4.9, 3. , 1.4, 0.2],
                 [4.7, 3.2, 1.3, 0.2],
                 [4.6, 3.1, 1.5, 0.2],
                 [5. , 3.6, 1.4, 0.2],
                 [5.4, 3.9, 1.7, 0.4],
                 [4.6, 3.4, 1.4, 0.3],
                 [5. , 3.4, 1.5, 0.2],
                 [4.4, 2.9, 1.4, 0.2],
                 [4.9, 3.1, 1.5, 0.1],
                 [5.4, 3.7, 1.5, 0.2],
                 [4.8, 3.4, 1.6, 0.2],
                 [4.8, 3. , 1.4, 0.1],
                 [4.3, 3. , 1.1, 0.1],
                 [5.8, 4. , 1.2, 0.2],
                 [5.7, 4.4, 1.5, 0.4],
                 [5.4, 3.9, 1.3, 0.4],
                 [5.1, 3.5, 1.4, 0.3],
                 [5.7, 3.8, 1.7, 0.3],
                 [5.1, 3.8, 1.5, 0.3],
                 [5.4, 3.4, 1.7, 0.2],
                 [5.1, 3.7, 1.5, 0.4],
                 [4.6, 3.6, 1. , 0.2],
                 [5.1, 3.3, 1.7, 0.5],
                 [4.8, 3.4, 1.9, 0.2],
                 [5. , 3. , 1.6, 0.2],
                 [5. , 3.4, 1.6, 0.4],
                 [5.2, 3.5, 1.5, 0.2],
                 [5.2, 3.4, 1.4, 0.2],
                 [4.7, 3.2, 1.6, 0.2],
                 [4.8, 3.1, 1.6, 0.2],
                 [5.4, 3.4, 1.5, 0.4],
                 [5.2, 4.1, 1.5, 0.1],
                 [5.5, 4.2, 1.4, 0.2],
                 [4.9, 3.1, 1.5, 0.2],
                 [5. , 3.2, 1.2, 0.2],
                 [5.5, 3.5, 1.3, 0.2],
                 [4.9, 3.6, 1.4, 0.1],
                 [4.4, 3. , 1.3, 0.2],
                 [5.1, 3.4, 1.5, 0.2],
                 [5. , 3.5, 1.3, 0.3],
                 [4.5, 2.3, 1.3, 0.3],
                 [4.4, 3.2, 1.3, 0.2],
                 [5. , 3.5, 1.6, 0.6],
                 [5.1, 3.8, 1.9, 0.4],
                 [4.8, 3. , 1.4, 0.3],
                 [5.1, 3.8, 1.6, 0.2],
                 [4.6, 3.2, 1.4, 0.2],
                 [5.3, 3.7, 1.5, 0.2],
                 [5. , 3.3, 1.4, 0.2],
                 [7. , 3.2, 4.7, 1.4],
                 [6.4, 3.2, 4.5, 1.5],
                 [6.9, 3.1, 4.9, 1.5],
                 [5.5, 2.3, 4. , 1.3],
                 [6.5, 2.8, 4.6, 1.5],
                 [5.7, 2.8, 4.5, 1.3],
                 [6.3, 3.3, 4.7, 1.6],
                 [4.9, 2.4, 3.3, 1. ],
                 [6.6, 2.9, 4.6, 1.3],
                 [5.2, 2.7, 3.9, 1.4],
                 [5. , 2. , 3.5, 1. ],
                 [5.9, 3. , 4.2, 1.5],
                 [6. , 2.2, 4. , 1. ],
                 [6.1, 2.9, 4.7, 1.4],
                 [5.6, 2.9, 3.6, 1.3],
                 [6.7, 3.1, 4.4, 1.4],
                 [5.6, 3. , 4.5, 1.5],
                 [5.8, 2.7, 4.1, 1. ],
                 [6.2, 2.2, 4.5, 1.5],
                 [5.6, 2.5, 3.9, 1.1],
                 [5.9, 3.2, 4.8, 1.8],
                 [6.1, 2.8, 4. , 1.3],
                 [6.3, 2.5, 4.9, 1.5],
                 [6.1, 2.8, 4.7, 1.2],
                 [6.4, 2.9, 4.3, 1.3],
                 [6.6, 3. , 4.4, 1.4],
                 [6.8, 2.8, 4.8, 1.4],
                 [6.7, 3. , 5. , 1.7],
                 [6. , 2.9, 4.5, 1.5],
                 [5.7, 2.6, 3.5, 1. ],
                 [5.5, 2.4, 3.8, 1.1],
                 [5.5, 2.4, 3.7, 1. ],
                 [5.8, 2.7, 3.9, 1.2],
                 [6. , 2.7, 5.1, 1.6],
                 [5.4, 3. , 4.5, 1.5],
                 [6. , 3.4, 4.5, 1.6],
                 [6.7, 3.1, 4.7, 1.5],
                 [6.3, 2.3, 4.4, 1.3],
                 [5.6, 3. , 4.1, 1.3],
                 [5.5, 2.5, 4. , 1.3],
                 [5.5, 2.6, 4.4, 1.2],
                 [6.1, 3. , 4.6, 1.4],
                 [5.8, 2.6, 4. , 1.2],
                 [5. , 2.3, 3.3, 1. ],
                 [5.6, 2.7, 4.2, 1.3],
                 [5.7, 3. , 4.2, 1.2],
                 [5.7, 2.9, 4.2, 1.3],
                 [6.2, 2.9, 4.3, 1.3],
                 [5.1, 2.5, 3. , 1.1],
                 [5.7, 2.8, 4.1, 1.3],
                 [6.3, 3.3, 6. , 2.5],
                 [5.8, 2.7, 5.1, 1.9],
                 [7.1, 3. , 5.9, 2.1],
                 [6.3, 2.9, 5.6, 1.8],
                 [6.5, 3. , 5.8, 2.2],
                 [7.6, 3. , 6.6, 2.1],
                 [4.9, 2.5, 4.5, 1.7],
                 [7.3, 2.9, 6.3, 1.8],
                 [6.7, 2.5, 5.8, 1.8],
                 [7.2, 3.6, 6.1, 2.5],
                 [6.5, 3.2, 5.1, 2. ],
                 [6.4, 2.7, 5.3, 1.9],
                 [6.8, 3. , 5.5, 2.1],
                 [5.7, 2.5, 5. , 2. ],
                 [5.8, 2.8, 5.1, 2.4],
                 [6.4, 3.2, 5.3, 2.3],
                 [6.5, 3. , 5.5, 1.8],
                 [7.7, 3.8, 6.7, 2.2],
                 [7.7, 2.6, 6.9, 2.3],
                 [6. , 2.2, 5. , 1.5],
                 [6.9, 3.2, 5.7, 2.3],
                 [5.6, 2.8, 4.9, 2. ],
                 [7.7, 2.8, 6.7, 2. ],
                 [6.3, 2.7, 4.9, 1.8],
                 [6.7, 3.3, 5.7, 2.1],
                 [7.2, 3.2, 6. , 1.8],
                 [6.2, 2.8, 4.8, 1.8],
                 [6.1, 3. , 4.9, 1.8],
                 [6.4, 2.8, 5.6, 2.1],
                 [7.2, 3. , 5.8, 1.6],
                 [7.4, 2.8, 6.1, 1.9],
                 [7.9, 3.8, 6.4, 2. ],
                 [6.4, 2.8, 5.6, 2.2],
                 [6.3, 2.8, 5.1, 1.5],
                 [6.1, 2.6, 5.6, 1.4],
                 [7.7, 3. , 6.1, 2.3],
                 [6.3, 3.4, 5.6, 2.4],
                 [6.4, 3.1, 5.5, 1.8],
                 [6. , 3. , 4.8, 1.8],
                 [6.9, 3.1, 5.4, 2.1],
                 [6.7, 3.1, 5.6, 2.4],
                 [6.9, 3.1, 5.1, 2.3],
                 [5.8, 2.7, 5.1, 1.9],
                 [6.8, 3.2, 5.9, 2.3],
                 [6.7, 3.3, 5.7, 2.5],
                 [6.7, 3. , 5.2, 2.3],
                 [6.3, 2.5, 5. , 1.9],
                 [6.5, 3. , 5.2, 2. ],
                 [6.2, 3.4, 5.4, 2.3],
                 [5.9, 3. , 5.1, 1.8]])
```

In [15]:
```
y
```

Out[15]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [16]:
```
# Step 2: Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [17]:
```
# Step 3: Create a Random Forest Classifier
rf_model = RandomForestClassifier(
    n_estimators=100,       # number of decision trees
    criterion='gini',       # metric to measure quality of split
    max_depth=None,         # expand trees fully unless restricted
    random_state=42
)
```

In [18]:
```
# Step 4: Train the model
rf_model.fit(X_train, y_train)
```

Out[18]:

```
▾ RandomForestClassifier                 i ?

▸ Parameters
```

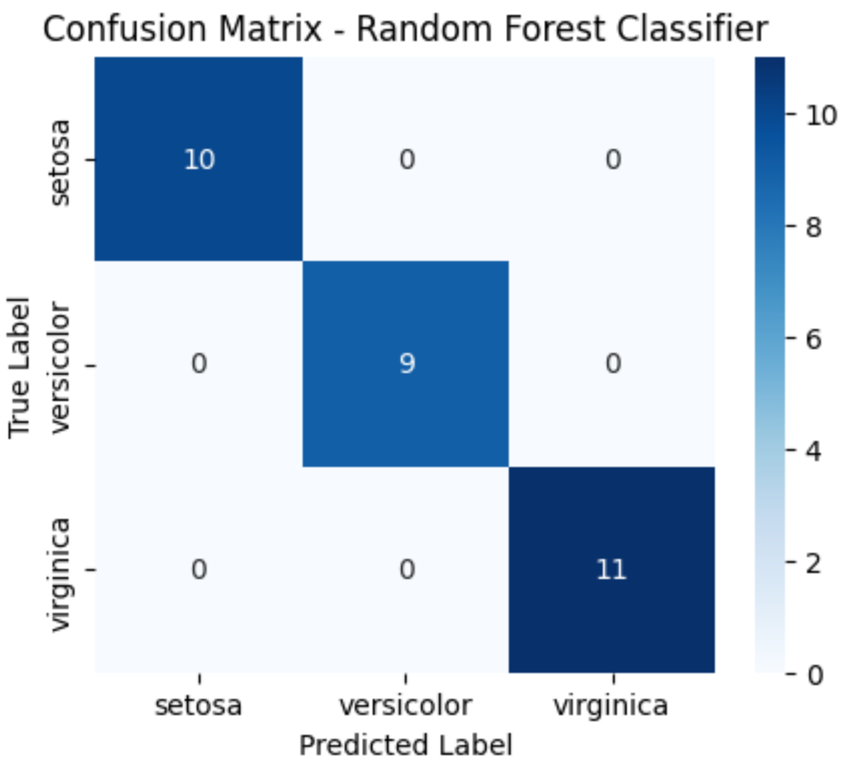
	n_estimators	100
	criterion	'gini'
	max_depth	None
	min_samples_split	2
	min_samples_leaf	1
	min_weight_fraction_leaf	0.0
	max_features	'sqrt'
	max_leaf_nodes	None
	min_impurity_decrease	0.0
	bootstrap	True
	oob_score	False
	n_jobs	None
	random_state	42
	verbose	0
	warm_start	False
	class_weight	None
	ccp_alpha	0.0
	max_samples	None
	monotonic_cst	None

In [19]:
```
# Step 5: Make predictions
y_pred = rf_model.predict(X_test)
```

In [20]:
```
# Step 6: Evaluate the model
print("✅ Accuracy:", accuracy_score(y_test, y_pred))
print("\n📊 Classification Report:\n", classification_report(y_test, y_pred))
print("\n🔷 Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

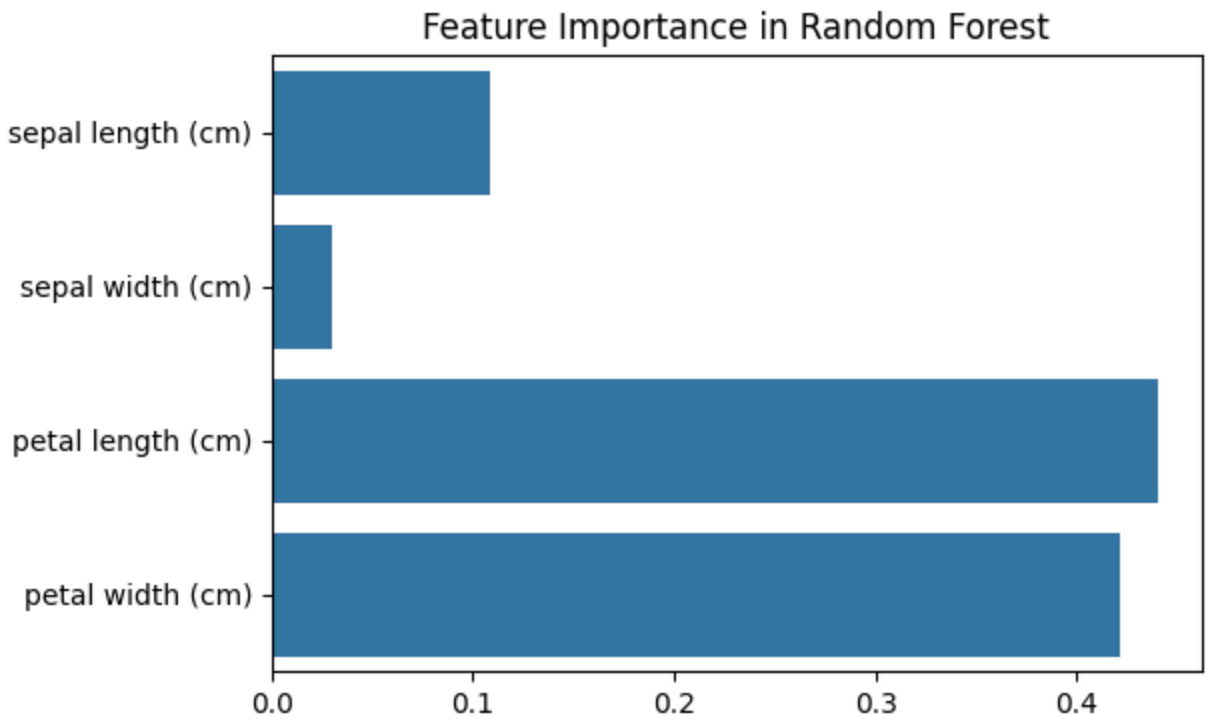
```
✅ Accuracy: 1.0

📊 Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


🔷 Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [21]:
```
#Step 7: Visualize confusion matrix
plt.figure(figsize=(5,4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blues', fmt='d',
            xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix - Random Forest Classifier")
plt.show()
```



In [22]:

```
# Step 8: Feature importance visualization
importances = rf_model.feature_importances_
plt.figure(figsize=(6,4))
sns.barplot(x=importances, y=iris.feature_names)
plt.title("Feature Importance in Random Forest")
plt.show()
```


Feature Importance in Random Forest
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
0.0
0.1
0.2
0.3
0.4