✅ Task 5: Decision Trees and Random Forests

🔷 Objective:

Learn and work with **tree-based models** for **classification** and **regression**.

🔷 Tools:

Scikit-learn – for building models

Graphviz / plot_tree – for visualizing trees

📘 Detailed Explanation / Guide

1 Train a Decision Tree Classifier and Visualize the Tree

✔ What is a Decision Tree?

A decision tree splits the data into smaller groups by asking **yes/no questions** at each node.
Example: *Is Age > 30?* → Yes → No → ... and so on.

✔ Steps to train a Decision Tree

1. Load your dataset

2. Split into training and testing
3. Fit the model
4. Visualize the tree

✔ Code Example

```
from sklearn.datasets import
load_iris

from sklearn.model_selection
import train_test_split
from sklearn.tree import
DecisionTreeClassifier,
plot_tree

import matplotlib.pyplot as plt

# Load data
data = load_iris()
X = data.data
y = data.target

# Train-test split

X_train, X_test, y_train,
y_test = train_test_split(X,
y, test_size=0.2,
random_state=42)

# Train model
dt =
DecisionTreeClassifier(max_dept
h=3)
dt.fit(X_train, y_train)

# Visualize tree
plt.figure(figsize=(12, 8))

plot_tree(dt, filled=True,
feature_names=data.feature_name
s,
class_names=data.target_names)
plt.show()
```

2 Analyze Overfitting and Control Tree Depth

✔ What is Overfitting?

Overfitting happens when the tree becomes **too deep** and learns unnecessary details from training data.

✔ Symptoms of Overfitting:

Very high training accuracy

Low test accuracy

✔ How to control overfitting?

Use **hyperparameters**:

max_depth

→ limit levels of tree

min_samples_split

→ minimum samples needed to split

min_samples_leaf

→ minimum samples in leaf

✔ Example

```
dt =
DecisionTreeClassifier(max_dept
h=4, min_samples_leaf=3)
dt.fit(X_train, y_train)
```

$\boxed{3}$ Train a Random Forest and Compare Accuracy

✔ What is a Random Forest?

A Random Forest builds **multiple decision trees** and averages their predictions.
This reduces overfitting and improves accuracy.

✔ Why Random Forest is better?

More stable than a single tree

Less overfitting

More accurate in most cases

✔ Code Example

```
from sklearn.ensemble import
RandomForestClassifier
from sklearn.metrics import
accuracy_score
```

```
rf =
RandomForestClassifier(n_estima
tors=100)
rf.fit(X_train, y_train)


y_pred_dt = dt.predict(X_test)

y_pred_rf = rf.predict(X_test)


print("Decision Tree
Accuracy:",
accuracy_score(y_test,
y_pred_dt))
print("Random Forest
Accuracy:",
accuracy_score(y_test,
y_pred_rf))
```

4 Interpret Feature Importances

✔ What is Feature Importance?

It tells which features contribute most to predictions.

✔ Code Example

```
import pandas as pd
import numpy as np


importance =
rf.feature_importances_
for i, score in
enumerate(importance):


print(f"{data.feature_names[i]}
 : {score:.4f}")
```

✔ Output Example (Iris dataset):

petal length → highest importance
petal width → also important
This helps understand which inputs influence the model most.

5 Evaluate using Cross-Validation

✔ What is Cross-Validation?

Instead of using a single train-test split, you split your data into **k folds**.
Model trains multiple times and average accuracy is taken.

✔ Code Example

```
from sklearn.model_selection
import cross_val_score


scores = cross_val_score(rf,
X, y, cv=5)
print("Cross-validation
accuracy:", scores.mean())
```

☆ Final Summary

Step	Task	What You Learn
1	Train Decision Tree	How a tree works & how to visualize it
2	Analyze Overfitting	Control tree complexity
3	Random Forest	Improve accuracy & stability
4	Feature Importance	Understand which features matter
5	Cross-Validation	Evaluate model properly