Session 9: Decision Trees and Random Forests

1. Introduction

In this session, we will explore **Decision Trees** and **Random Forests**, two powerful algorithms in machine learning.

- **Decision Trees** help make predictions by learning simple decision rules from data.
- **Random Forests** improve the performance of Decision Trees by combining multiple trees to reduce overfitting and increase accuracy.

By the end of this lesson, you will:
✅ Understand how Decision Trees work and how they split data.
✅ Learn how Random Forests improve predictions.
✅ Train both models using Python.
✅ Interpret results and understand feature importance.

---

2. Decision Trees

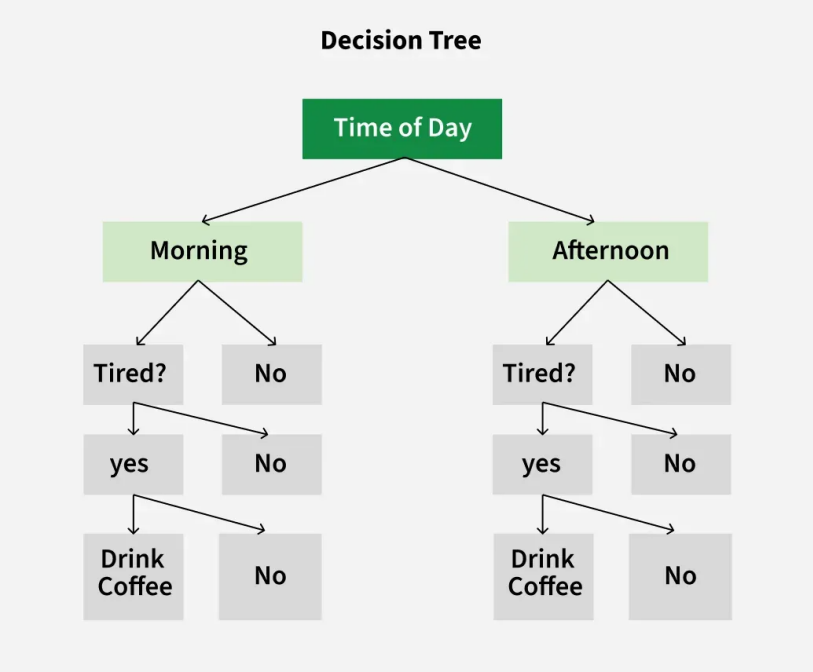
What is a Decision Tree?

A **Decision Tree** is a model that makes decisions based on a series of questions. It works like a flowchart where each step leads to a new decision.

For example, imagine you are deciding whether to go outside:

1. **Is it raining?**
    - Yes → Stay inside.
    - No → Go to the next question.
2. **Is it too hot?**
    - Yes → Stay inside.
    - No → Go outside.

The same logic applies to machine learning models. A Decision Tree asks a series of questions about the data and learns patterns to make predictions.



How Does a Decision Tree Work?

1. **Splitting the Data**
   - The model starts by looking at all available features (characteristics) in the dataset.
   - It chooses the feature that best separates the data into different groups.
   - The dataset is split into two or more parts based on this feature.
2. **Repeating the Process**
   - Each new group (node) is analyzed again, and another split is made.
   - This process continues until the tree reaches a stopping condition.
3. **Making Predictions**
   - Once the tree is fully grown, it assigns each new data point to one of the final groups (leaf nodes).
   - Each leaf node represents a decision or a predicted class.

Example of a Decision Tree

Let's consider the **Iris dataset**, which contains different types of flowers classified based on petal length, petal width, and other characteristics.

- A **Decision Tree** might first ask, **"Is petal length greater than 2 cm?"**
- If **yes**, it might then ask, **"Is petal width greater than 1 cm?"**
- Based on these conditions, it continues dividing the dataset until it reaches a conclusion.

This structured decision-making helps classify new flower samples based on learned rules.

---

3. Training a Decision Tree (Hands-on Example)

We will now train a **Decision Tree** to classify flowers in the **Iris dataset**.

Step 1: Import Libraries

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Step 2: Load the Dataset

```
# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Step 3: Train a Decision Tree Model

```
# Train a Decision Tree classifier
dt_model = DecisionTreeClassifier(max_depth=3, random_state=42)
dt_model.fit(X_train, y_train)

# Make predictions
y_pred = dt_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Decision Tree Accuracy: {accuracy:.2f}")
```

Step 4: Visualize the Decision Tree

```
plt.figure(figsize=(12,8))
plot_tree(dt_model, feature_names=iris.feature_names,
class_names=iris.target_names, filled=True)
plt.show()
```

---

4. Problems with Decision Trees

🔻 **Overfitting** – If the tree is too deep, it memorizes the training data instead of learning general patterns.

🔻 **Underfitting** – If the tree is too shallow, it misses important patterns.
🔻 **Sensitive to Noise** – Small changes in data can lead to very different trees.

To solve these problems, we use **Random Forests**.

---

5. Random Forests

What is a Random Forest?

A **Random Forest** is a collection of multiple **Decision Trees** that work together to improve predictions.

Instead of relying on one tree, Random Forest:
✅ **Trains multiple Decision Trees on random parts of the dataset.**
✅ **Combines their predictions to make a final decision.**
✅ **Reduces overfitting and increases accuracy.**

How Does Random Forest Work?

1. **Creates multiple Decision Trees using different parts of the data.**
2. **Uses different features for each tree to ensure diversity.**
3. **Combines the predictions of all trees.**
    - For classification: Takes the most common prediction (majority vote).
    - For regression: Takes the average prediction.

---

6. Training a Random Forest Model

Step 1: Import the Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

# Train a Random Forest model with 100 trees
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions
y_pred_rf = rf_model.predict(X_test)

# Calculate accuracy
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f"Random Forest Accuracy: {accuracy_rf:.2f}")
```

Step 2: Compare Decision Tree and Random Forest

```
print(f"Decision Tree Accuracy: {accuracy:.2f}")
```

```
print(f"Random Forest Accuracy: {accuracy_rf:.2f}")
```

🔺 **Random Forest usually performs better than a single Decision Tree because it averages out mistakes and reduces overfitting.**

---

7. Feature Importance in Random Forest

One advantage of Random Forest is that it shows which features are most important in making decisions.

Step 1: Get Feature Importance Scores

```
# Get feature importance
importances = rf_model.feature_importances_

# Print importance of each feature
for feature, importance in zip(iris.feature_names, importances):
    print(f"{feature}: {importance:.3f}")
```

Step 2: Visualize Feature Importance

```
plt.figure(figsize=(8,6))
plt.barh(iris.feature_names, importances, color='skyblue')
plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("Feature Importance in Random Forest")
plt.show()
```

🔺 **More important features have a higher impact on predictions.**

---

8. Summary

Model	Strengths	Weaknesses
Decision Tree	Easy to understand, fast	Overfits easily
Random Forest	Reduces overfitting, improves accuracy	Slower than a single tree

Key Takeaways

✅ **Decision Trees** are simple but prone to **overfitting**.
✅ **Random Forests** improve predictions by averaging multiple trees.
✅ **Feature importance** helps understand which variables are important.

---

9. Activities

1. **Modify the Decision Tree Model**
    - Change `max_depth` and observe the impact on accuracy.
    - Try `criterion="entropy"` instead of `gini` and compare the results.
2. **Experiment with Random Forest**
    - Train with `n_estimators=50` and `n_estimators=200` and compare accuracy.
3. **Overfitting and Underfitting**
    - Compare training accuracy vs test accuracy for both models.

---

10. Discussion Questions

1. Why does Random Forest perform better than a single Decision Tree?
2. How does feature importance help in model interpretation?
3. What happens if we use too many trees in a Random Forest?