Day 8 – Decision Tree & Random Forest Classification

📓 Today's Highlights

In today's session, we explored **tree-based classification models** — specifically:

- 🌳 **Decision Tree Classifier**
- 🌲 **Random Forest Classifier**

We also performed **data analysis and cleaning**, and visualized the Iris dataset using **Seaborn** before applying the models.

🌳 What is Decision Tree?

A **Decision Tree** is a supervised machine learning algorithm used for both **classification** and **regression** tasks.

- It works like a flowchart where:
  - **Nodes** represent features
  - **Branches** represent decision rules
  - **Leaves** represent outcomes (class labels)
- The tree splits data based on the feature that gives the **maximum information gain** or **Gini index**.
- Easy to interpret but can overfit on noisy datasets.

🌲 What is Random Forest?

Random Forest is an **ensemble** learning method built using multiple decision trees.

- It combines predictions from multiple trees to improve accuracy and control overfitting.
- Each tree is trained on a **random subset** of the data (bagging).
- It is more accurate and robust than a single Decision Tree.

□ Step-by-Step Implementation: Data Preprocessing

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv('Iris.csv')
```

```
# Initial exploration
df.head()
df.info()
df.describe().T
df.isna().sum()
df.duplicated().sum()

# Drop duplicates
df.drop_duplicates(inplace=True)

# Visualize class distribution
sns.countplot(x='species', data=df, palette='Set2')
plt.title("Species Count Distribution")
plt.show()
```

🔍 Feature Analysis & Visualization

```
# Compare sepal length by species
sns.barplot(x="species", y="sepal_length", data=df, palette=["blue",
"green", "salmon"])
plt.title("Mean Sepal Length by Species")
plt.show()

# Box plot
sns.boxplot(data=df, x="species", y="sepal_length", palette=["blue",
"green", "salmon"])
plt.title("Sepal Length Distribution by Species")
plt.show()
```

🌳 Decision Tree Classifier

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder

# Prepare data
X = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y = df['species']

le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Split data
x_train, x_test, y_train, y_test = train_test_split(X, y_encoded,
test_size=0.2, random_state=42)

# Train Decision Tree
dt = DecisionTreeClassifier(random_state=42)
dt.fit(x_train, y_train)

# Evaluate
y_pred = dt.predict(x_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred))
```

📊 Decision Tree Confusion Matrix

```
conf_mat = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_mat, annot=True, fmt=".0f", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Decision Tree Confusion Matrix")
plt.show()
```

🌲 Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

# Train Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(x_train, y_train)

# Evaluate
y_pred_rf = rf.predict(x_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

📊 Random Forest Confusion Matrix

```
conf_mat_rf = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(conf_mat_rf, annot=True, fmt=".0f", cmap="Greens")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Random Forest Confusion Matrix")
plt.show()
```

✅ Output & Observations

```
Decision Tree Accuracy: 1.00
Random Forest Accuracy: 1.00
```

Both models gave **100% accuracy** on the Iris dataset (due to its clean and balanced nature).

📝 Conclusion

- Learned the working and difference between **Decision Tree** and **Random Forest** classifiers.
- Applied them successfully on the **Iris dataset**.
- Visualized results using **confusion matrices**.
- Observed that **Random Forest reduces overfitting** and improves performance by aggregating decisions from multiple trees.