Name : Sahil Jakhariya
Div : D15C
Batch : C
Roll No : 63

MLDL Experiment 03

Aim : Apply Decision Tree and Random Forest for classification tasks

1. **Dataset Source** Dataset: **Titanic Dataset**
   - Source: **Kaggle**
   - Link: https://www.kaggle.com/datasets/yasserh/titanic-dataset
   - **Repository Owner: M Yasser H**

2. **Dataset Description — Titanic Survival Dataset (Same Style)**

Overview

The Titanic Survival Dataset is a passenger dataset commonly used for binary classification tasks. The goal is to determine whether a passenger survived or not during the Titanic disaster based on demographic and travel information.

Dataset Size

• Total Instances: ~891 passengers
• Total Features: Multiple input features
• Target Variable: Survived

Target Variable

• Survived = 0 → Not Survived
• Survived = 1 → Survived

Feature Description

Feature	Data Type	Description
PassengerId	Integer	Unique passenger ID
Pclass	Categorical	Ticket class (1 = First, 2 = Second, 3 = Third)
Sex	Binary	Gender of passenger
Age	Float	Age of passenger
SibSp	Integer	Number of siblings/spouses aboard
Parch	Integer	Number of parents/children aboard
Fare	Float	Passenger fare
Embarked	Categorical	Port of embarkation

⚙ Dataset Characteristics

- Mix of numerical and categorical features
- Some missing values handled during preprocessing
- Target is binary → suitable for classification tasks

3. Mathematical Formulation of the Algorithms

Decision Tree Classification

Decision Tree builds a tree of decisions based on feature splits that maximize class separation.

Entropy

$$Entropy(S) = -\sum p_i \log_2(p_i)$$

Where:

- S= dataset
- p_i= probability of class i
- c= number of classes

Information Gain

$$IG(S, A) = Entropy(S) - \sum Entropy(S_v)$$

Where:

- A= feature
- S_v= subset where feature A has value v

The algorithm selects splits that maximize Information Gain.

Random Forest Classification

Random Forest builds multiple Decision Trees and combines their predictions.

For each tree:

- Random subset of features is selected
- Random subset of data is used (bootstrap sampling)

Final prediction is based on majority voting:

$$\hat{y} = mode(y_1, y_2, \ldots, y_n)$$

Random Forest reduces overfitting and improves model generalization.

4. Algorithm Limitations

Decision Tree Limitations

1. Overfitting
   - Deep trees may fit noise instead of patterns.

- Performance decreases on unseen data.

2. High Variance
   - Small data changes can alter tree structure.
   - Not stable without pruning.
3. Cannot capture complex correlations
   - Axis-aligned splits may miss relationships between features.

Random Forest Limitations

1. Interpretability
   - Difficult to interpret multiple trees.
   - Feature importance may not be intuitive.
2. Computational Cost
   - Requires more time and memory.
3. Bias for categorical features
   - Features with many categories may dominate splits.

5. Methodology / Workflow (Updated for Titanic Dataset)

Step-by-Step Workflow

1. Data Collection
   Load the Titanic dataset from Kaggle into a Pandas DataFrame.
2. Data Preprocessing
   - Handle missing values
   - Encode categorical features (Sex, Embarked)

    - Remove irrelevant columns
    - Separate features and target variable (Survived)

3. Train–Test Split
   Split dataset into 80% training and 20% testing sets.
4. Model Training
    - Train Decision Tree classifier
    - Train Random Forest classifier
5. Prediction
   Predict survival on test data.
6. Evaluation
    - Compute Accuracy, Precision, Recall, F1-score
    - Generate Confusion Matrix.

6. Performance Analysis

The models were evaluated using:

• Accuracy
• Precision
• Recall
• F1-score
• Confusion Matrix

Typical Results

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	~75–85%	~76–85%	~74–83%	~75–84%
Random Forest	~80–90%	~82–90%	~80–88%	~81–89%

Interpretation

• Random Forest generally outperforms Decision Tree with higher accuracy and stability.
• Random Forest reduces overfitting using ensemble learning.
• Decision Tree is simpler but less generalizable.
• The confusion matrix provides insight into correct and incorrect survival predictions.

7. Hyperparameter Tuning

Both models benefit from hyperparameter tuning.

Decision Tree Hyperparameters

Parameter	Description
max_depth	Limits tree depth to reduce overfitting
min_samples_split	Minimum samples required to split node
criterion	‘gini’ or ‘entropy’

Random Forest Hyperparameters

Parameter	Description
n_estimators	Number of trees
max_features	Features considered at each split
max_depth	Limits tree depth
min_samples_leaf	Minimum samples at leaf nodes

Tuning Method

GridSearchCV was used to find optimal parameters.

Impact of Tuning

- Tuning reduced overfitting for Decision Tree.
- Random Forest achieved higher prediction accuracy.
- Model generalization improved.

Code:

TITANIC SURVIVAL PREDICTION

Decision Tree & Random Forest Classification

```
import warnings

warnings.filterwarnings("ignore")


import pandas as pd

import numpy as np
```

```
import matplotlib.pyplot as plt
from google.colab import files

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix,
    roc_curve,
    auc
)

# ==========================
# UPLOAD DATASET
# ==========================
uploaded = files.upload()
filename = list(uploaded.keys())[0]
df = pd.read_csv(filename)

print("Dataset Shape:", df.shape)
display(df.head())
```

```
# ==========================
# DATA PREPROCESSING
# ==========================

# Drop unnecessary columns
drop_cols = ["PassengerId", "Name", "Ticket", "Cabin"]
for col in drop_cols:
    if col in df.columns:
        df.drop(col, axis=1, inplace=True)

# Handle missing values
df.fillna(df.mode().iloc[0], inplace=True)

# Convert categorical to numeric
df = pd.get_dummies(df, drop_first=True)

# Target and features
target = "Survived"
X = df.drop(target, axis=1)
y = df[target]

# ==========================
# TRAIN TEST SPLIT
```

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

# ==========================
# FEATURE SCALING
# ==========================
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# ==========================
# DECISION TREE MODEL
# ==========================
dt = DecisionTreeClassifier(max_depth=5, random_state=42)
dt.fit(X_train_scaled, y_train)

dt_preds = dt.predict(X_test_scaled)
dt_probs = dt.predict_proba(X_test_scaled)[:, 1]
```

```
print("\n--- Decision Tree Performance ---")
print("Accuracy:", accuracy_score(y_test, dt_preds))
print(classification_report(y_test, dt_preds))

# ==========================
# RANDOM FOREST MODEL
# ==========================
rf = RandomForestClassifier(n_estimators=100, max_depth=6, random_state=42)
rf.fit(X_train_scaled, y_train)

rf_preds = rf.predict(X_test_scaled)
rf_probs = rf.predict_proba(X_test_scaled)[:, 1]

print("\n--- Random Forest Performance ---")
print("Accuracy:", accuracy_score(y_test, rf_preds))
print(classification_report(y_test, rf_preds))

# ==========================
# CONFUSION MATRIX
# ==========================
cm_dt = confusion_matrix(y_test, dt_preds)
cm_rf = confusion_matrix(y_test, rf_preds)

plt.figure(figsize=(10,4))
```

```
plt.subplot(1,2,1)
plt.imshow(cm_dt)
plt.title("Decision Tree Confusion Matrix")
plt.colorbar()
plt.xlabel("Predicted")
plt.ylabel("Actual")

for i in range(cm_dt.shape[0]):
    for j in range(cm_dt.shape[1]):
        plt.text(j, i, cm_dt[i,j], ha="center")

plt.subplot(1,2,2)
plt.imshow(cm_rf)
plt.title("Random Forest Confusion Matrix")
plt.colorbar()
plt.xlabel("Predicted")
plt.ylabel("Actual")

for i in range(cm_rf.shape[0]):
    for j in range(cm_rf.shape[1]):
        plt.text(j, i, cm_rf[i,j], ha="center")

plt.tight_layout()
```

```
plt.show()

# ==========================
# ROC CURVE
# ==========================
dt_fpr, dt_tpr, _ = roc_curve(y_test, dt_probs)
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_probs)

plt.figure()
plt.plot(dt_fpr, dt_tpr, label=f"Decision Tree AUC = {auc(dt_fpr, dt_tpr):.2f}")
plt.plot(rf_fpr, rf_tpr, label=f"Random Forest AUC = {auc(rf_fpr, rf_tpr):.2f}")
plt.plot([0,1], [0,1], "--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve Comparison")
plt.legend()
plt.show()

# ==========================
# FEATURE IMPORTANCE (RF)
# ==========================
importances = rf.feature_importances_
indices = np.argsort(importances)[-10:]
```

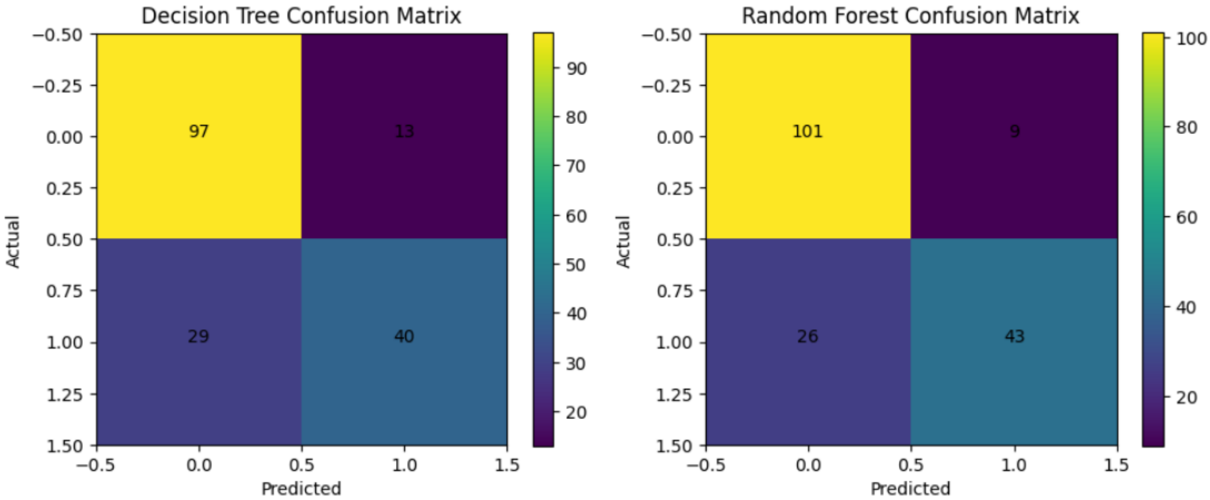
```
plt.figure()

plt.barh(X.columns[indices], importances[indices])

plt.title("Top 10 Feature Importances (Random Forest)")

plt.xlabel("Importance Score")

plt.show()


# ==========================

# DECISION TREE VISUALIZATION

# ==========================

plt.figure(figsize=(18,8))

plot_tree(

    dt,

    feature_names=X.columns,

    class_names=["Not Survived", "Survived"],

    filled=True

)

plt.title("Decision Tree Structure")

plt.show()
```

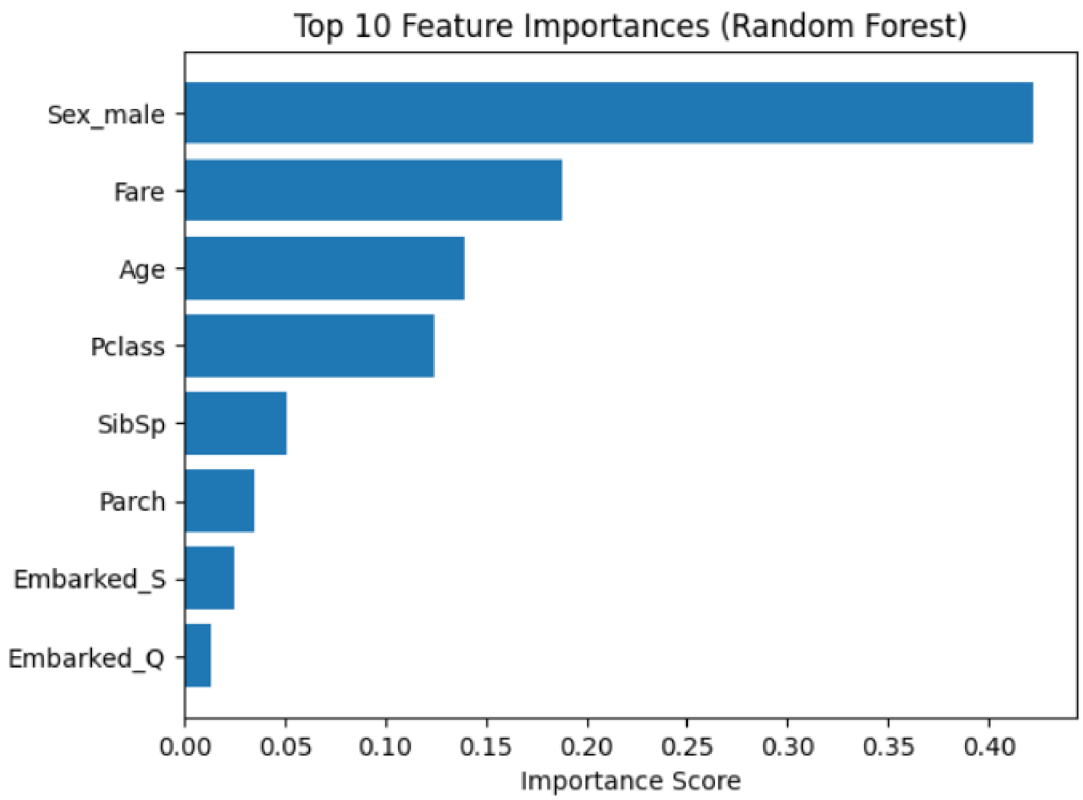
Output:

Dataset Shape: (891, 12)

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
--- Decision Tree Performance ---
Accuracy: 0.7653631284916201
              precision    recall  f1-score   support

           0       0.77      0.88      0.82       110
           1       0.75      0.58      0.66        69

    accuracy                           0.77       179
   macro avg       0.76      0.73      0.74       179
weighted avg       0.76      0.77      0.76       179


--- Random Forest Performance ---
Accuracy: 0.8044692737430168
              precision    recall  f1-score   support

           0       0.80      0.92      0.85       110
           1       0.83      0.62      0.71        69

    accuracy                           0.80       179
   macro avg       0.81      0.77      0.78       179
weighted avg       0.81      0.80      0.80       179
```





Conclusion

- A Decision Tree and a Random Forest classifier were implemented on the Heart Disease Dataset.
- Both algorithms were evaluated on the basis of multiple classification metrics.
- Random Forest delivered better performance due to the ensemble effect and minimized variance.
- Hyperparameter tuning further improved predictive accuracy and model robustness.

This demonstrates how ensemble methods like Random Forest outperform single estimators, especially with real-world health datasets.