Task 5: Decision Trees and Random Forests

Objective

Learn tree-based models for classification & regression using **Decision Trees** and **Random Forests**.

Steps & Implementation

1. Dataset

- Dataset Used: **Heart Disease Dataset**
- Target: Presence (1) or Absence (0) of Heart Disease

2. Decision Tree Classifier

- Trained using `DecisionTreeClassifier` (Scikit-learn)
- Controlled `max_depth` to reduce overfitting
- Visualized using **Graphviz** and `plot_tree`

3. Random Forest Classifier

- Trained using `RandomForestClassifier` (100 trees)
- Compared accuracy with single Decision Tree
- Extracted **feature importances**

4. Evaluation

- Used **train-test split** and **cross-validation**
- Metrics: Accuracy, Confusion Matrix, Precision, Recall, F1-score
- Example results:
- Decision Tree Accuracy: 78%
- Random Forest Accuracy: 85%

5. Feature Importance

- Top features contributing to prediction: **Age, Cholesterol, Max Heart Rate, Blood Pressure**

Interview Questions & Answers

1. **How does a decision tree work?**

2. Splits data using features based on information gain or Gini index until stopping criteria are met.

3. **What is entropy and information gain?**

4. Entropy measures impurity. Information gain = reduction in entropy after a split.

5. **How is random forest better than a single tree?**

6. Uses bagging and multiple trees → reduces variance, improves generalization.

7. **What is overfitting and how do you prevent it?**

8. Overfitting: Model learns noise, not patterns. Prevent via `max_depth`, pruning, or ensembles.

9. **What is bagging?**

10. Bootstrap Aggregation: train multiple models on random subsets of data, aggregate results.

11. **How do you visualize a decision tree?**

12. Using `plot_tree` in sklearn or Graphviz.

13. **How do you interpret feature importance?**

14. Each feature's contribution to prediction is calculated from impurity reduction across splits.

15. **What are the pros/cons of random forests?**

16. ✅Pros: High accuracy, robust to noise, less overfitting
17. ❌Cons: Slower, less interpretable

Code (Colab / Jupyter Notebook)

```
# Decision Trees and Random Forest - Heart Disease Dataset
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Load dataset
data = pd.read_csv('heart.csv')  # or your dataset path
X = data.drop('target', axis=1)
y = data['target']

# 2. 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)

# 3. Decision Tree
```

```python
clf_tree = DecisionTreeClassifier(max_depth=4, random_state=42)
clf_tree.fit(X_train, y_train)

# 4. Random Forest
clf_rf = RandomForestClassifier(n_estimators=100, random_state=42)
clf_rf.fit(X_train, y_train)

# 5. Predictions
y_pred_tree = clf_tree.predict(X_test)
y_pred_rf = clf_rf.predict(X_test)

# 6. Evaluation
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_tree))
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))

print("\nDecision Tree Classification Report:\n",
classification_report(y_test, y_pred_tree))
print("\nRandom Forest Classification Report:\n",
classification_report(y_test, y_pred_rf))

# 7. Confusion Matrix for Random Forest
cm = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Random Forest Confusion Matrix')
plt.show()

# 8. Visualize Decision Tree
plt.figure(figsize=(12,8))
plot_tree(clf_tree, feature_names=X.columns, class_names=['No
Disease','Disease'], filled=True)
plt.show()

# 9. Feature Importance
importances = clf_rf.feature_importances_
feat_imp = pd.Series(importances,
index=X.columns).sort_values(ascending=False)
print("Feature Importances:\n", feat_imp)

# Plot feature importance
feat_imp.plot(kind='barh')
plt.title('Random Forest Feature Importance')
plt.show()

# 10. Cross-validation
cv_scores = cross_val_score(clf_rf, X, y, cv=5)
print("Random Forest CV Accuracy:", np.mean(cv_scores))
```

Repository Structure

```
☣ DecisionTree-RandomForest-Task
├ 営 decision_tree_random_forest.ipynb   # Code
├ 営 README.md                           # Explanation
├ 🈂 outputs.png                         # Plots (tree, feature importance)
└ ☣ data                                # Dataset
```

Key Learning

- Decision trees are simple and interpretable but prone to overfitting.
- Random forests improve accuracy and robustness using bagging.
- Feature importance helps understand model decisions.