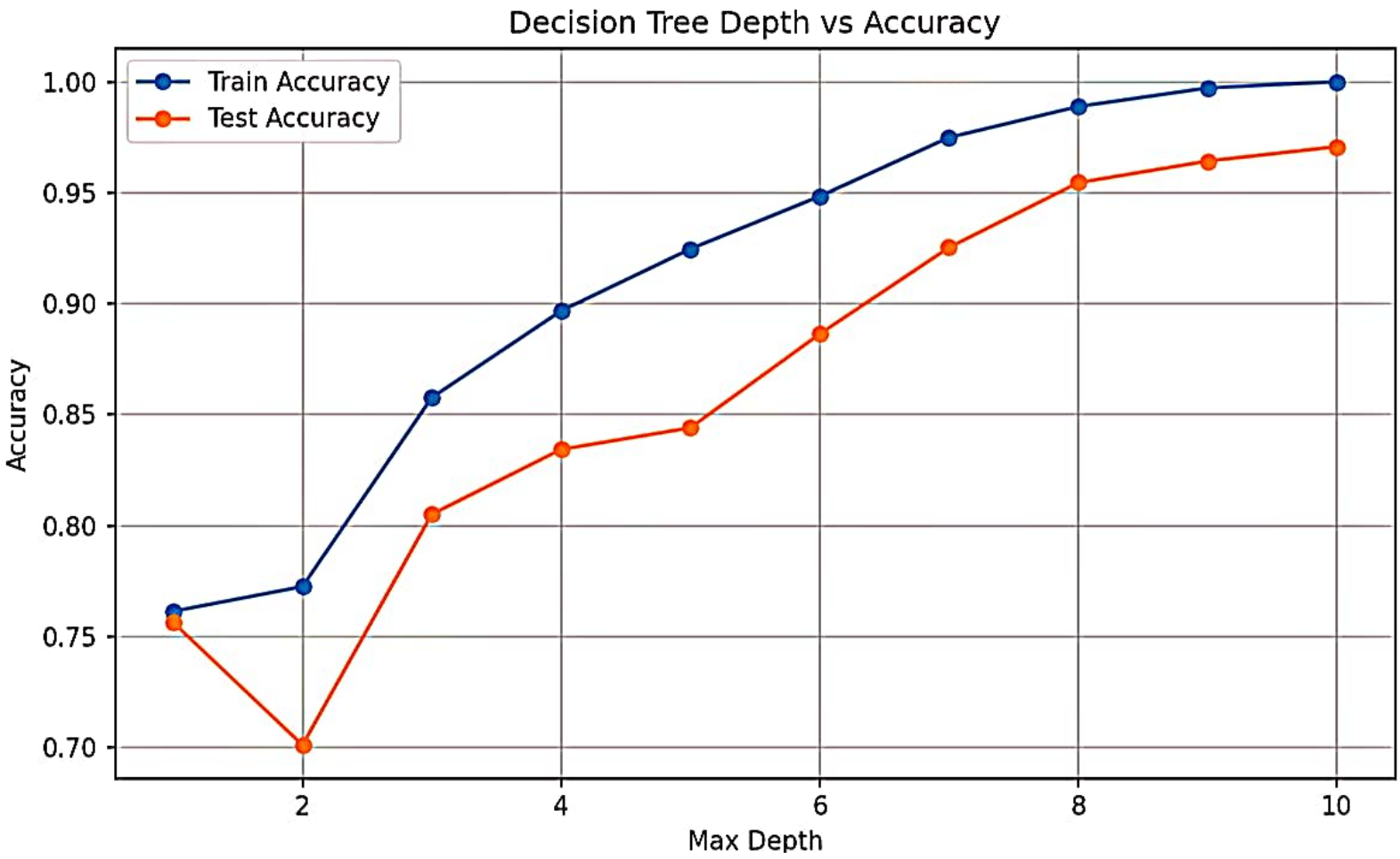
decisontee_randomforets.py > ...

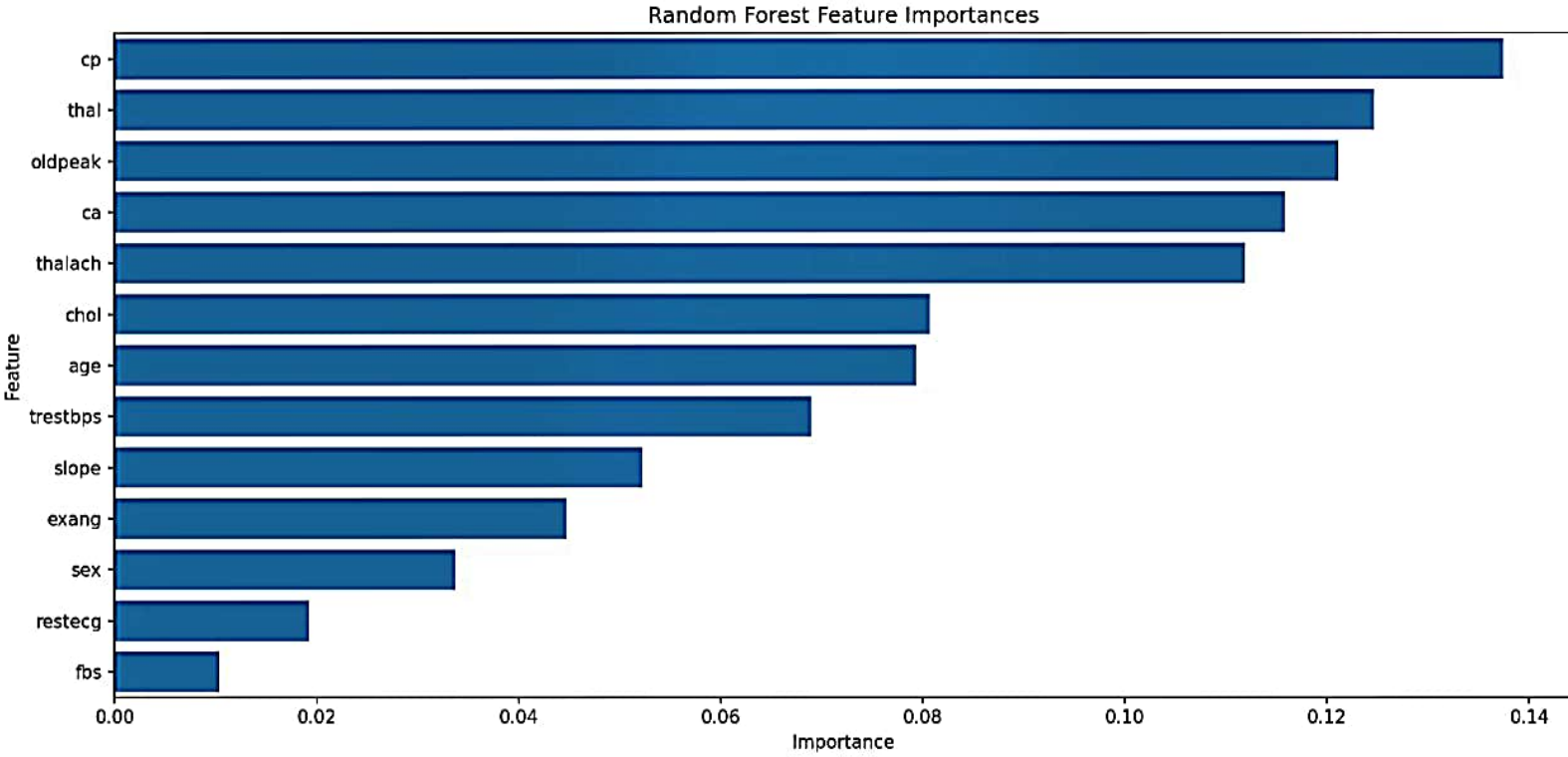
```python
# Task 5: Decision Trees and Random Forests using heart.csv

# Import required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import graphviz

# Step 1: Load the dataset
df = pd.read_csv("heart.csv")

# Step 2: Prepare features and labels
X = df.drop('target', axis=1)
y = df['target']

# Step 3: Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Step 4: Train Decision Tree Classifier
dt_model = DecisionTreeClassifier(random_state=0)
dt_model.fit(X_train, y_train)

# Step 5: Evaluate Decision Tree
dt_pred = dt_model.predict(X_test)
dt_accuracy = accuracy_score(y_test, dt_pred)
print(f"Decision Tree Accuracy: {dt_accuracy:.4f}")

# Step 6: Visualize the Decision Tree
dot_data = export_graphviz(dt_model, out_file=None,
                           feature_names=X.columns,
                           class_names=["No Disease", "Disease"],
                           filled=True, rounded=True,
```

```
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("decision_tree_heart", format="png", cleanup=True)
print("Decision tree visual saved as 'decision_tree_heart.png'")

# Step 7: Analyze Overfitting by controlling max_depth
depths = range(1, 11)
train_scores = []
test_scores = []

for depth in depths:
    model = DecisionTreeClassifier(max_depth=depth, random_state=0)
    model.fit(X_train, y_train)
    train_scores.append(model.score(X_train, y_train))
    test_scores.append(model.score(X_test, y_test))

# Plot accuracy vs max depth
plt.figure(figsize=(8, 5))
plt.plot(depths, train_scores, marker='o', label='Train Accuracy')
plt.plot(depths, test_scores, marker='o', label='Test Accuracy')
plt.xlabel("Max Depth")
plt.ylabel("Accuracy")
plt.title("Decision Tree Depth vs Accuracy")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig("tree_depth_vs_accuracy.png")
plt.show()

# Step 8: Train Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=0)
rf_model.fit(X_train, y_train)

# Step 9: Evaluate Random Forest
rf_pred = rf_model.predict(X_test)
```

```
rf_accuracy = accuracy_score(y_test, rf_pred)
print(f"Random Forest Accuracy: {rf_accuracy:.4f}")

# Step 10: Feature Importance from Random Forest
importances = rf_model.feature_importances_
feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

# Plot feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title("Random Forest Feature Importances")
plt.tight_layout()
plt.savefig("feature_importances.png")
plt.show()

# Step 11: Cross-validation scores
dt_cv_scores = cross_val_score(dt_model, X, y, cv=5)
rf_cv_scores = cross_val_score(rf_model, X, y, cv=5)

print(f"Decision Tree CV Accuracy: {dt_cv_scores.mean():.4f} ± {dt_cv_scores.std():.4f}")
print(f"Random Forest CV Accuracy: {rf_cv_scores.mean():.4f} ± {rf_cv_scores.std():.4f}")
```

```
[Running] python -u "d:\titanic-preprossesing\decisontee_randomforets.py"
Decision Tree Accuracy: 0.9708
Decision tree visual saved as 'decision_tree_heart.png'
Random Forest Accuracy: 0.9805
Decision Tree CV Accuracy: 0.9971 ◆ 0.0059
Random Forest CV Accuracy: 0.9941 ◆ 0.0072

[Done] exited with code=0 in 119.544 seconds
```


Decision Tree Depth vs Accuracy
Train Accuracy
Test Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
Accuracy
2
4
6
8
10
Max Depth


Random Forest Feature Importances
cp
thal
oldpeak
ca
thalach
chol
age
trestbps
slope
exang
sex
restecg
fbs
Feature
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Importance