```
#Task-5 Decision trees and Random forests
#1.Train a Decision Tree Classifier and visualize the tree.

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/heart.csv')
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

1025 rows × 14 columns

Next steps: Generate code with df View recommended plots New interactive sheet

```
x = df.drop("target", axis=1)
y = df["target"]
x
y
```

	target
0	0
1	0
2	0
3	0
4	0
...	...
1020	1
1021	0
1022	0
1023	1
1024	0

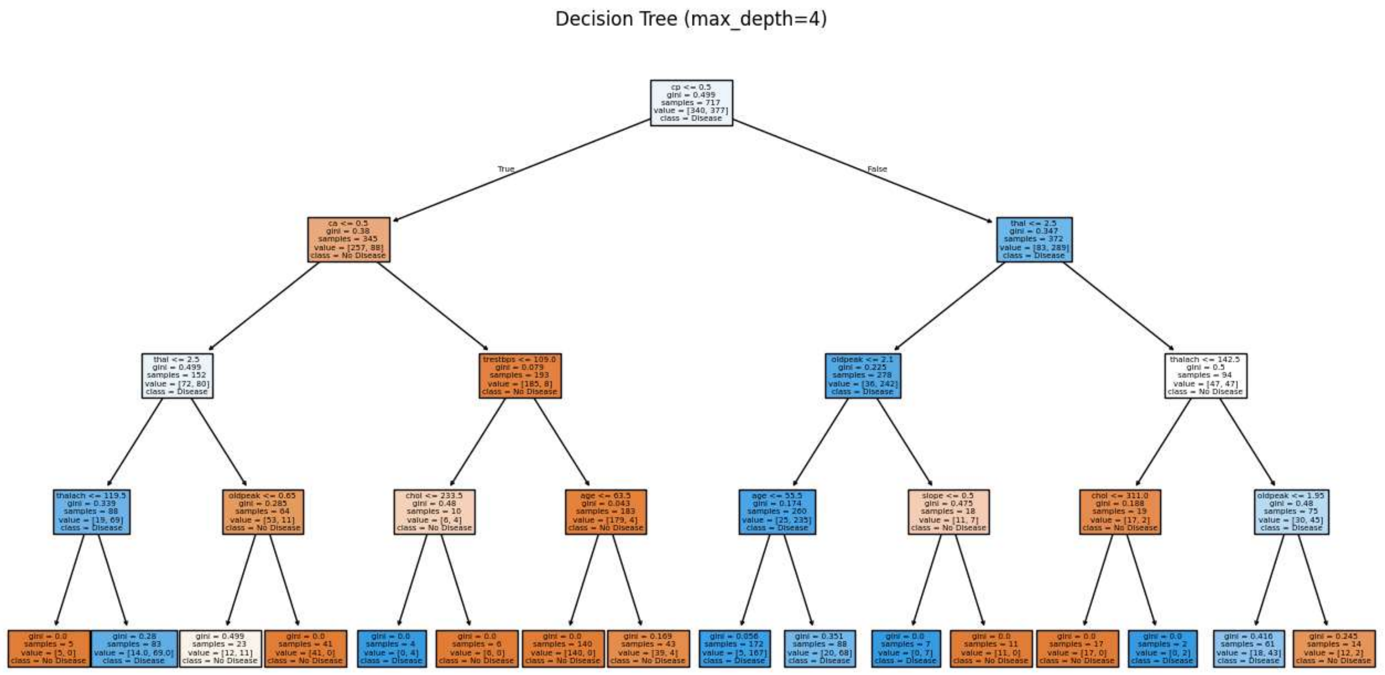
1025 rows × 1 columns

dtype: int64

```
#2.Analyze overfitting and control tree depth.
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 accuracy_score
# 2. Split into Train/Test
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
dt_clf = DecisionTreeClassifier(max_depth=4, random_state=42)
dt_clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=4, random_state=42)
```

```
#3.Train a Random Forest and compare accuracy
plt.figure(figsize=(16, 8))
plot_tree(dt_clf, feature_names=x.columns, class_names=["No Disease", "Disease"], filled=True)
plt.title("Decision Tree (max_depth=4)")
plt.show()
```


Decision Tree (max_depth=4)
cp <= 0.5 gini = 0.499 samples = 717 value = [340, 377] class = Disease
True
False
ca <= 0.5 gini = 0.38 samples = 345 value = [257, 88] class = No Disease
thal <= 2.5 gini = 0.347 samples = 372 value = [83, 289] class = Disease
thal <= 2.5 gini = 0.499 samples = 152 value = [72, 80] class = Disease
trestbps <= 109.0 gini = 0.079 samples = 193 value = [185, 8] class = No Disease
oldpeak <= 2.1 gini = 0.225 samples = 278 value = [36, 242] class = Disease
thalach <= 142.5 gini = 0.5 samples = 94 value = [47, 47] class = No Disease
thalach <= 119.5 gini = 0.339 samples = 88 value = [19, 69] class = Disease
oldpeak <= 0.65 gini = 0.285 samples = 64 value = [53, 11] class = No Disease
chol <= 233.5 gini = 0.48 samples = 10 value = [6, 4] class = No Disease
age <= 63.5 gini = 0.043 samples = 183 value = [179, 4] class = No Disease
age <= 55.5 gini = 0.174 samples = 260 value = [25, 235] class = Disease
slope <= 0.5 gini = 0.475 samples = 18 value = [11, 7] class = No Disease
chol <= 311.0 gini = 0.188 samples = 19 value = [17, 2] class = No Disease
oldpeak <= 1.95 gini = 0.48 samples = 75 value = [30, 45] class = Disease
gini = 0.0 samples = 5 value = [5, 0] class = No Disease
gini = 0.28 samples = 83 value = [14.0, 69.0] class = Disease
gini = 0.499 samples = 23 value = [12, 11] class = No Disease
gini = 0.0 samples = 41 value = [41, 0] class = No Disease
gini = 0.0 samples = 4 value = [0, 4] class = Disease
gini = 0.0 samples = 6 value = [6, 0] class = No Disease
gini = 0.0 samples = 140 value = [140, 0] class = No Disease
gini = 0.169 samples = 43 value = [39, 4] class = No Disease
gini = 0.056 samples = 172 value = [5, 167] class = Disease
gini = 0.351 samples = 88 value = [20, 68] class = Disease
gini = 0.0 samples = 7 value = [0, 7] class = Disease
gini = 0.0 samples = 11 value = [11, 0] class = No Disease
gini = 0.0 samples = 17 value = [17, 0] class = No Disease
gini = 0.0 samples = 2 value = [0, 2] class = Disease
gini = 0.416 samples = 61 value = [18, 43] class = Disease
gini = 0.245 samples = 14 value = [12, 2] class = No Disease

```
# 4. Overfitting Analysis
deep = DecisionTreeClassifier(random_state=42)
deep.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

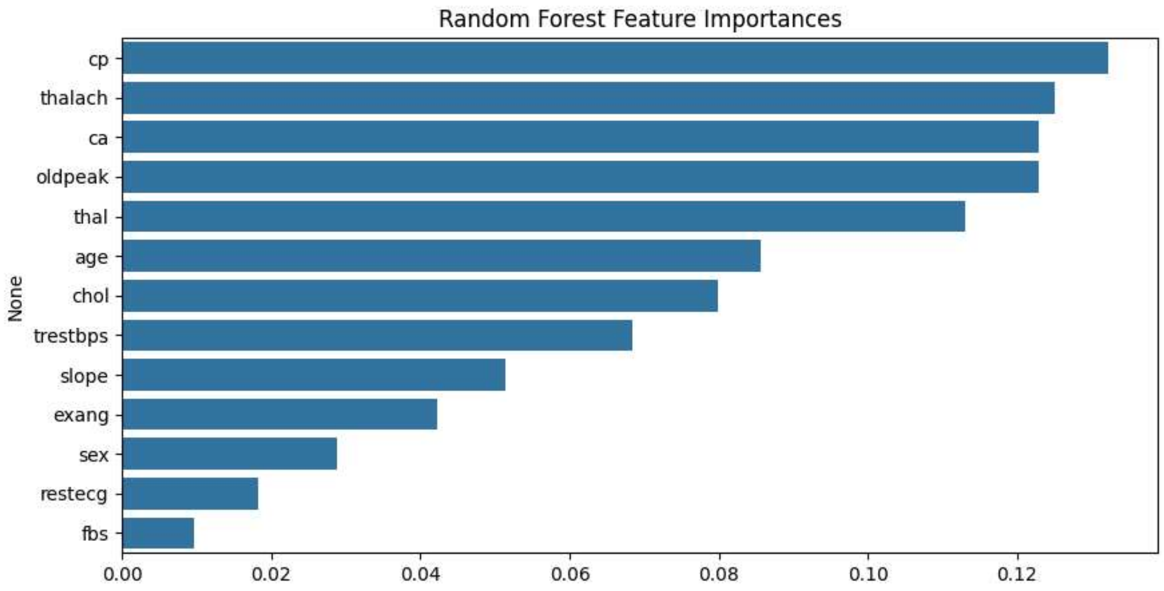
```
print(f"Shallow Tree Accuracy: {accuracy_score(y_test, dt_clf.predict(X_test)):.2f}")
print(f"Deep Tree Accuracy: {accuracy_score(y_test, deep.predict(X_test)):.2f}")
```

```
Shallow Tree Accuracy: 0.83
Deep Tree Accuracy: 0.97
```

```
# 5. Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_clf.fit(X_train, y_train)
rf_acc = accuracy_score(y_test, rf_clf.predict(X_test))
print(f"Random Forest Accuracy: {rf_acc:.2f}")
```

```
Random Forest Accuracy: 0.98
```

```
 #6. Feature Importance
importances = pd.Series(rf_clf.feature_importances_, index=x.columns).sort_values(ascending=False)
plt.figure(figsize=(10, 5))
sns.barplot(x=importances.values, y=importances.index)
plt.title("Random Forest Feature Importances")
plt.show()
```



```
# 7. Cross-Validation Scores
cv_dt = cross_val_score(dt_clf, x, y, cv=5)
cv_rf = cross_val_score(rf_clf, x, y, cv=5)

print(f"Decision Tree CV Accuracy: {cv_dt.mean():.2f}")
print(f"Random Forest CV Accuracy: {cv_rf.mean():.2f}")
```

```
Decision Tree CV Accuracy: 0.83
Random Forest CV Accuracy: 1.00
```