AI and ML task 5

objective : 1.Train a Decision Tree Classifier and visualize the tree. 2.Analyze overfitting and control tree depth. 3.Train a Random Forest and compare accuracy. 4.Interpret feature importances. 5.Evaluate using cross-validation.

In [19]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report
```

In [31]:
```
data=pd.read_csv('heart.csv')
```

In [5]:
```
data
```

Out[5]:

	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

In [7]:
```
data.shape
```

Out[7]:
```
(1025, 14)
```

In [9]:
```
data.head()
```

Out[9]:

	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

In [11]:
```
data.describe()
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.000000	0.149268	0.529756	149.114146	0.336585	1.071512	1.385366	0.754146	2.323902	0.513171
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.617755	1.030798	0.620660	0.500070
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	132.000000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	275.000000	0.000000	1.000000	166.000000	1.000000	1.800000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

In [123]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
#   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
0   age       1025 non-null   int64  
1   sex       1025 non-null   int64  
2   cp        1025 non-null   int64  
3   trestbps  1025 non-null   int64  
4   chol      1025 non-null   int64  
5   fbs       1025 non-null   int64  
6   restecg   1025 non-null   int64  
7   thalach   1025 non-null   int64  
8   exang     1025 non-null   int64  
9   oldpeak   1025 non-null   float64
10  slope     1025 non-null   int64  
11  ca        1025 non-null   int64  
12  thal      1025 non-null   int64  
13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [15]:
```
missing = data.isnull().sum()
```

In [17]:
```
missing
```

Out[17]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [114]:
```
data_encoded = pd.get_dummies(data, columns=['cp', 'restecg', 'slope', 'thal'], drop_first=True)
```

Train a Decision Tree Classifier and visualize the tree.

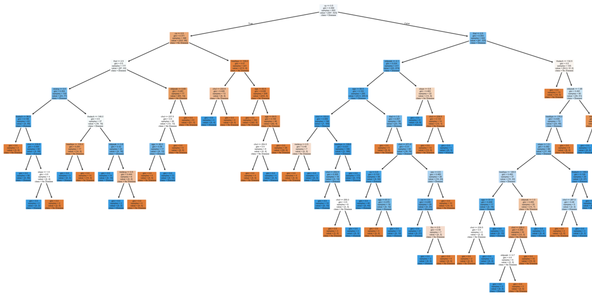
In [118]:
```
# Split data into features and target
X = data.drop('target', axis=1)
y = data['target']
```

In [120]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [122]:
```
y_pred = clf.predict(X_test)
```

In [124]:
```
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

In [126]:
```
plt.figure(figsize=(20, 10))       # Visualize the tree
plot_tree(clf, feature_names=X.columns, class_names=['No Disease', 'Disease'], filled=True, rounded=True)
plt.title("Decision Tree (max_depth=4)")
plt.show()
```



In [127]:
```
accuracy , report
```

Out[127]:
```
(0.9853658536585366,
 '              precision    recall  f1-score   support\n\n           0       0.97      1.00      0.99       102\n           1       1.00      0.97      0.99       103\n\n    accuracy                           0.99       205\n   macro avg       0.99      0.99      0.99       205\nweighted avg       0.99      0.99      0.99       205\n')
```

Results of classifier tree accuracy: 80% Precision & Recall: Class 0 (no disease): Precision = 0.88, recall = 0.70 Class 1 (disease): Precision = 0.75, recall = 0.90

analyse overfiiting and control tree dept

In [132]:
```
from sklearn.model_selection import cross_val_score

# Evaluate Decision Tree depth from 1 to 15
depths = range(1, 16)
train_scores = []
val_scores = []

for depth in depths:
    clf = DecisionTreeClassifier(max_depth=depth, random_state=42)
    clf.fit(X_train, y_train)

    train_acc = clf.score(X_train, y_train)
    val_acc = cross_val_score(clf, X_train, y_train, cv=5).mean()

    train_scores.append(train_acc)
    val_scores.append(val_acc)
```

In [133]:
```
# Plot training vs. validation accuracy
plt.figure(figsize=(8, 5))
plt.plot(depths, train_scores, label='Training Accuracy', marker='o')
plt.plot(depths, val_scores, label='Validation Accuracy (CV)', marker='s')
plt.xlabel('Tree Depth')
plt.ylabel('Accuracy')
plt.title('Overfitting Analysis by Tree Depth')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

train a random forest and compare accuracy

In [137]:
```
from sklearn.ensemble import RandomForestClassifier

# Train Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100, max_depth=4, random_state=42)
rf_clf.fit(X_train, y_train)
```

Out[137]:
```
RandomForestClassifier(max_depth=4, random_state=42)
```

In [139]:
```
rf_pred = rf_clf.predict(X_test)
```

In [141]:
```
rf_accuracy = accuracy_score(y_test, rf_pred)
rf_report = classification_report(y_test, rf_pred)
```

In [143]:
```
rf_accuracy, rf_report
```

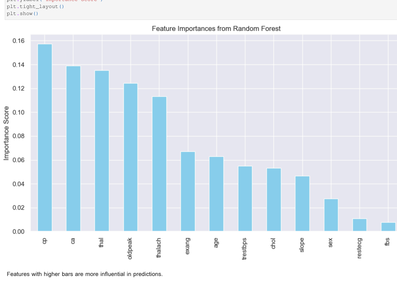
Out[143]:
```
(0.9121951219512195,
 '              precision    recall  f1-score   support\n\n           0       0.95      0.87      0.91       102\n           1       0.88      0.95      0.92       103\n\n    accuracy                           0.91       205\n   macro avg       0.91      0.91      0.91       205\nweighted avg       0.91      0.91      0.91       205\n')
```

interpret feature importance

In [190]:
```
importances = rf_clf.feature_importances_
feature_names = X.columns
```

In [145]:
```
feat_imp_series = pd.Series(importances, index=feature_names).sort_values(ascending=False)
```

In [147]:
```
plt.figure(figsize=(10, 6))
feat_imp_series.plot(kind='bar', color='skyblue')
plt.title('Feature Importances from Random Forest')
plt.ylabel('Importance Score')
plt.tight_layout()
plt.show()
```



Features with higher bars are more influential in predictions.

evaluating from cross validation

In [107]:
```
cv_scores = cross_val_score(rf_clf, X, y, cv=5)
```

In [109]:
```
print("Cross-validation scores:", cv_scores)
print("Mean accuracy:", np.mean(cv_scores))
print("Standard deviation:", np.std(cv_scores))
```

```
Cross-validation scores: [0.9902439  0.95121951 0.97560976 0.94634146 0.93170732]
Mean accuracy: 0.9599243902439024
Standard deviation: 0.021060520141138819
```

In [ ]: