Rohini Vishwanathan

Knowledge Discovery and Data Mining

IST 340

IST340 Computer Exercise (CE2)

Decision Trees Part 2

Q1.

1. Evaluation Approach

Performance Measures Table

Measure	Description	Definition of Value Function	Weight	Threshold
Loss	Measures misclassification cost	(FP+2 x FN)/ (TP + TN +FP +FN)	0.60	Lower is better
Simplicity	Prefers simpler tress (lower depth)	1/ Tree Depth	0.30	Higher is better
Stability	Measures consistency across train and test	(1− (Train Accuracy−Test Accuracy)	0.10	Higher is better
Combination Function	Overall Score = $w_{Accuracy}$*$Score_{Accuracy}$ + $w_{Simplicity}$*$Score_{Simplicity}$… 0.60 x (1 - Loss) + 0.30 x Simplicity + 0.10 x Stability			

2. Summary of Results

Results Table (Using Loss Matrix)

DT Label/ Description	Performance Measures						Overall Score
	Loss (or Profit)		Simplicity		Stability		
	Value	Score	No of Leaves	Score	Score	Justification	
Entropy (3%)	0.2623	0.7377	20	0.1429	0.9312	High stability as the model generalizes well across train and test data	0.5786
Entropy (6%)	0.4426	0.5574	11	0.2000	0.9237	Slightly lower stability but still acceptable, less complex model	0.4868

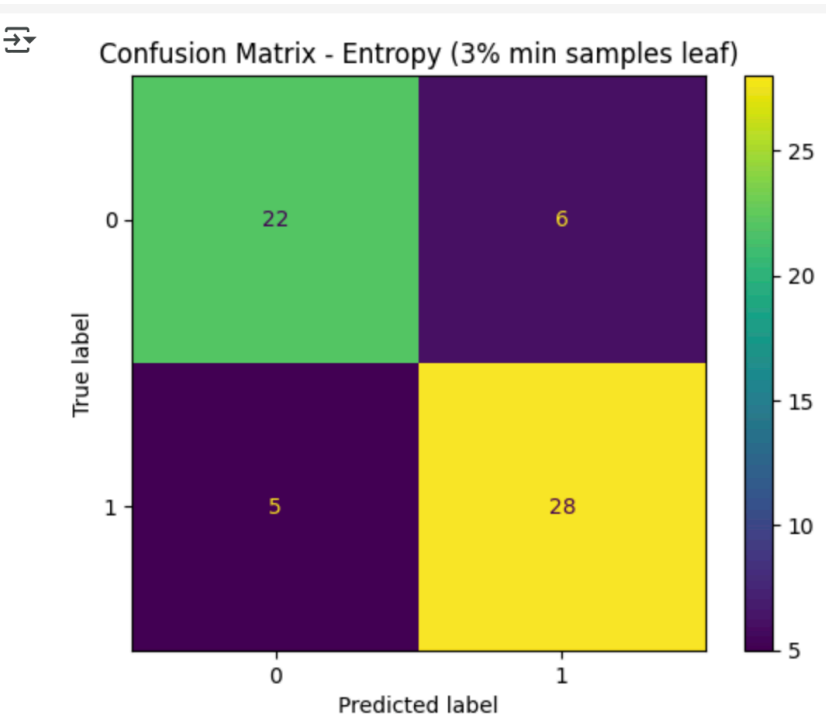
3. Description of the best decision tree

- The best Decision Tree was trained using Entropy with min_samples_leaf=3%.
- It had the lowest Loss (0.2623), meaning fewer costly misclassifications.
- While slightly deeper, it remained highly stable (0.9312).
- Its final performance score was 0.5786, making it the best model for Q1.

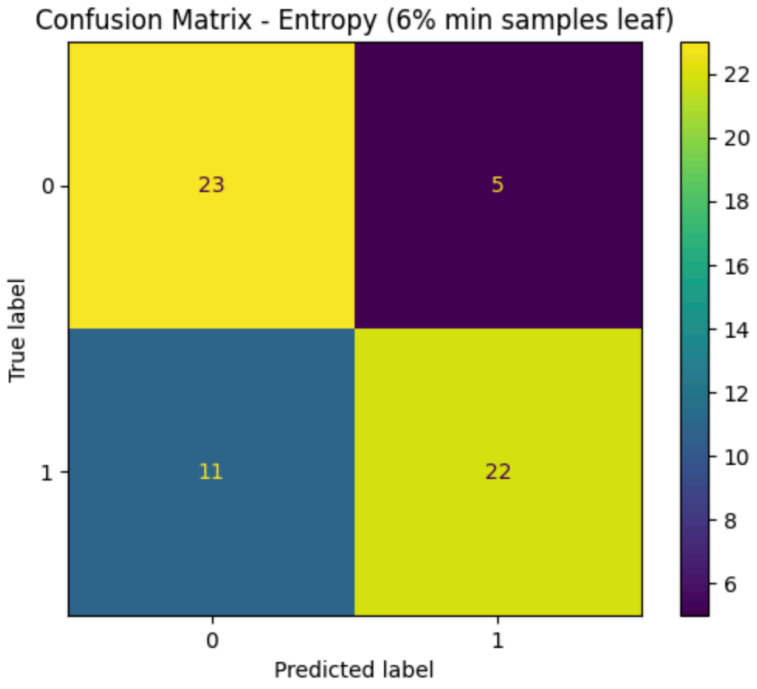
4. Evidence of Experimentation

- Confusion Matrices for both models (demonstrating classification performance).

Entropy 3%


Confusion Matrix - Entropy (3% min samples leaf)
22
6
5
28
0
1
True label
0
1
Predicted label
25
20
15
10
5

Entropy 6%


Confusion Matrix - Entropy (6% min samples leaf)
23
5
11
22
0
1
True label
0
1
Predicted label
22
20
18
16
14
12
10
8
6

- **Tree depths evaluation (to assess simplicity).**

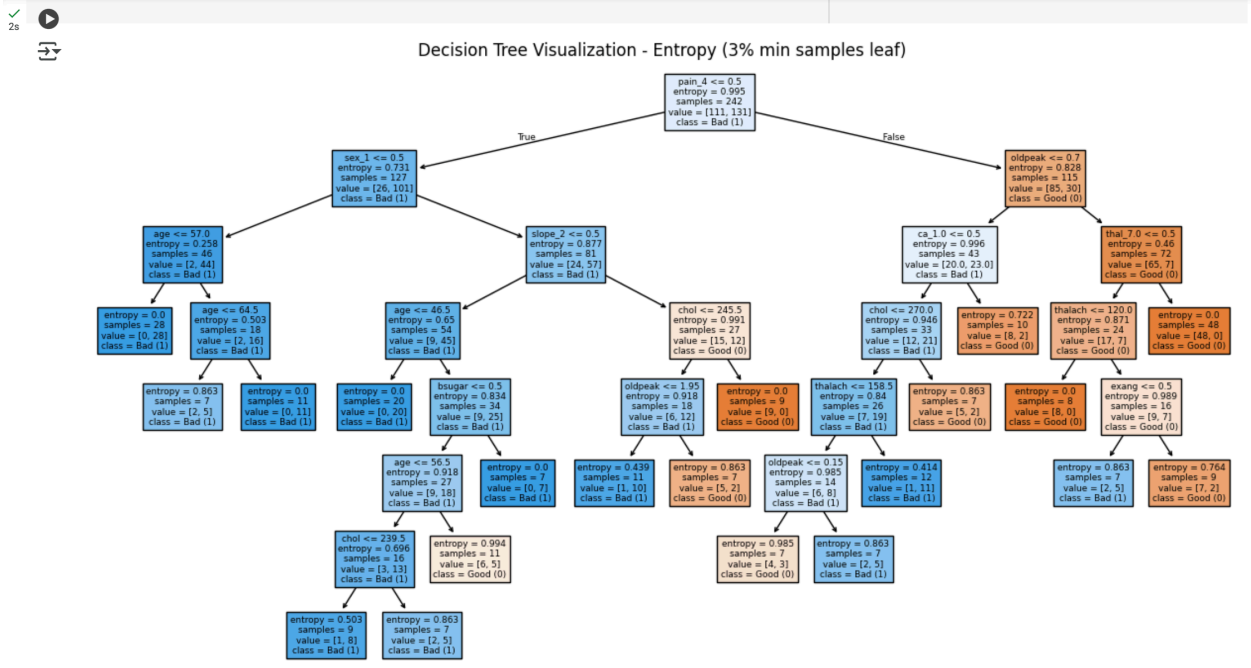
```
# Simplicity = Inverse of tree depth (lower depth is better)
simplicity_entropy_3 = 1 / dt_entropy.get_depth()
simplicity_entropy_6 = 1 / dt_entropy_6.get_depth()

print(f"Simplicity (Entropy, 3% min samples leaf): {simplicity_entropy_3:.4f}")
print(f"Simplicity (Entropy, 6% min samples leaf): {simplicity_entropy_6:.4f}")
```

```
Simplicity (Entropy, 3% min samples leaf): 0.1429
Simplicity (Entropy, 6% min samples leaf): 0.2000
```

- **Decision tree visualization- (best model)**

Entropy 3%



- **Final weighted scores calculation (to validate performance scoring).**

```
[14] # Compute final weighted performance score for both models
     final_score_entropy_3 = (0.60 * (1 - loss_entropy_3)) + (0.30 * simplicity_entropy_3) + (0.10 * stability_entropy_3)
     final_score_entropy_6 = (0.60 * (1 - loss_entropy_6)) + (0.30 * simplicity_entropy_6) + (0.10 * stability_entropy_6)

     print(f"Final Score (Entropy, 3% min samples leaf): {final_score_entropy_3:.4f}")
     print(f"Final Score (Entropy, 6% min samples leaf): {final_score_entropy_6:.4f}")
```

```
Final Score (Entropy, 3% min samples leaf): 0.5786
Final Score (Entropy, 6% min samples leaf): 0.4868
```

- **Train-Test Split verification screenshot (showing dataset partitioning)**

```
[9]  from sklearn.model_selection import train_test_split

     # Define Features (X) and Target (y)
     X = df.drop("target", axis=1)  # All features except target
     y = df["target"]  # Target variable (BAD = 1, GOOD = 0)

     # Perform an 80%-20% train-test split with stratification
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

     # Verify dataset sizes
     print("Training Set Size:", X_train.shape)
     print("Testing Set Size:", X_test.shape)
```

```
Training Set Size: (242, 20)
Testing Set Size: (61, 20)
```

Q2

1. Evaluation Approach

Performance Measures Table

Measure	Description	Definition of Value Function	Weight	Threshold
Loss	Measures prediction error	RMSE (Root Mean Squared Error) / MAE (Mean Absolute Error)	0.55	Lower is better
Simplicity	Prefers simpler trees (lower depth)	1/ tree depth	0.25	Higher is better
Stability	Measures consistency across train and test	1- (Train accuracy - Test Accuracy)	0.20	Higher the better
Combination Function	Overall Score = $w_{Accuracy}$*$Score_{Accuracy}$ + $w_{Simplicity}$*$Score_{Simplicity}$… 0.55 x (1- Loss) + 0.25 x Simplicity + 0.20 x Stability			

2. Summary of Results

Results Table

DT Label/ Description	Performance Measures						Overall Score
	Loss		Simplicity		Stability		
	Value	Score	No of Leaves	Score	Score	Justification	
Squared Error (2% min samples leaf)	0.1011	0.8989	25	0.1429	0.3652	Moderate stability; more leaves (25) make the tree slightly sensitive to variations.	0.1643

Squared Error, 5% min samples leaf	0.2367	0.7633	12	0.2500	0.6940	Highest stability; fewer leaves (12) ensure better generalization	0.3315
Absolute Error, 2% min samples leaf)	-0.1553	1.1553	47	0.0769	0.2774	Lower stability; more prone to overfitting due to high sensitivity.	-0.0107
Absolute Error, 5% min samples leaf	-0.2543	1.2543	16	0.1429	0.4613	Improved stability but still sensitive compared to squared error models.	-0.0119

3. Description of the best decision tree

- Best Model Selected: Squared Error, 5% min samples leaf
  - Chosen due to highest final score (0.3315), better stability (0.6940), and balanced accuracy and simplicity.

Key Decision Rules from the Best Model

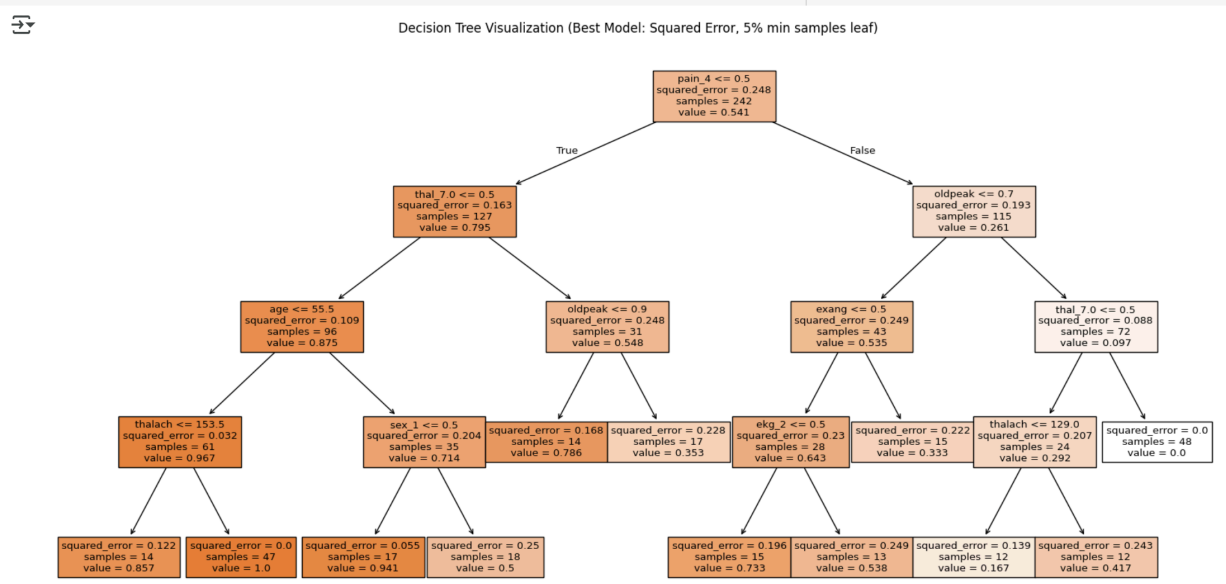
- If pain_4 <= 0.5 and thal_7.0 <= 0.5 and age <= 55.5, then thalach <= 153.50 — High probability of positive outcome.
- If pain_4 > 0.5 and oldpeak > 0.7 and thalach <= 129.0, then lower probability of positive outcome.
- Splits involving thalach, oldpeak, age, and pain_4 are strong predictors of the outcome.

Justification for Selection

- Good Rules-
  - pain_4, oldpeak, thalach, and age are strong medical predictors of heart disease risk.
  - Splits at thalach ≤ 153.50 and oldpeak ≤ 0.7 effectively separate patient risk levels.

4. Evidence of Experimentation

- **Decision Tree Visualizations of the best model**



- **Extracted Decision Rules (Best Model)**

```
Decision Tree Rules for the Best Model:

|--- pain_4 <= 0.50
|   |--- thal_7.0 <= 0.50
|   |   |--- age <= 55.50
|   |   |   |--- thalach <= 153.50
|   |   |   |   |--- value: [0.86]
|   |   |   |--- thalach >  153.50
|   |   |   |   |--- value: [1.00]
|   |   |--- age >  55.50
|   |   |   |--- sex_1 <= 0.50
|   |   |   |   |--- value: [0.94]
|   |   |   |--- sex_1 >  0.50
|   |   |   |   |--- value: [0.50]
|   |--- thal_7.0 >  0.50
|   |   |--- oldpeak <= 0.90
|   |   |   |--- value: [0.79]
|   |   |--- oldpeak >  0.90
|   |   |   |--- value: [0.35]
|--- pain_4 >  0.50
|   |--- oldpeak <= 0.70
|   |   |--- exang <= 0.50
|   |   |   |--- ekg_2 <= 0.50
|   |   |   |   |--- value: [0.73]
|   |   |   |--- ekg_2 >  0.50
|   |   |   |   |--- value: [0.54]
|   |   |--- exang >  0.50
|   |   |   |--- value: [0.33]
|   |--- oldpeak >  0.70
|   |   |--- thal_7.0 <= 0.50
|   |   |   |--- thalach <= 129.00
|   |   |   |   |--- value: [0.17]
|   |   |   |--- thalach >  129.00
|   |   |   |   |--- value: [0.42]
|   |   |--- thal_7.0 >  0.50
|   |   |   |--- value: [0.00]
```

- **Final Weighted Scores Computation**

```
Accuracy (Squared Error, 2% min samples leaf): 0.1011
Simplicity (Squared Error, 2% min samples leaf): 0.1429
Stability (Squared Error, 2% min samples leaf): 0.3652
Final Score (Squared Error, 2% min samples leaf): 0.1643

Accuracy (Absolute Error, 2% min samples leaf): -0.1553
Simplicity (Absolute Error, 2% min samples leaf): 0.0769
Stability (Absolute Error, 2% min samples leaf): 0.2774
Final Score (Absolute Error, 2% min samples leaf): -0.0107

Accuracy (Squared Error, 5% min samples leaf): 0.2367
Simplicity (Squared Error, 5% min samples leaf): 0.2500
Stability (Squared Error, 5% min samples leaf): 0.6940
Final Score (Squared Error, 5% min samples leaf): 0.3315

Accuracy (Absolute Error, 5% min samples leaf): -0.2543
Simplicity (Absolute Error, 5% min samples leaf): 0.1429
Stability (Absolute Error, 5% min samples leaf): 0.4613
Final Score (Absolute Error, 5% min samples leaf): -0.0119
```

- **Train-Test Split Verification**

```
# Import necessary libraries
from sklearn.tree import DecisionTreeRegressor
import numpy as np

# Define min_samples_leaf as 2% and 5% of training data
min_samples_2 = int(0.02 * len(X_train))
min_samples_5 = int(0.05 * len(X_train))

# Train Decision Tree using Squared Error (2% min samples leaf)
dt_squared_error_2 = DecisionTreeRegressor(criterion="squared_error", min_samples_leaf=min_samples_2, random_state=42)
dt_squared_error_2.fit(X_train, y_train)

# Train Decision Tree using Absolute Error (2% min samples leaf)
dt_absolute_error_2 = DecisionTreeRegressor(criterion="absolute_error", min_samples_leaf=min_samples_2, random_state=42)
dt_absolute_error_2.fit(X_train, y_train)

# Train Decision Tree using Squared Error (5% min samples leaf)
dt_squared_error_5 = DecisionTreeRegressor(criterion="squared_error", min_samples_leaf=min_samples_5, random_state=42)
dt_squared_error_5.fit(X_train, y_train)

# Train Decision Tree using Absolute Error (5% min samples leaf)
dt_absolute_error_5 = DecisionTreeRegressor(criterion="absolute_error", min_samples_leaf=min_samples_5, random_state=42)
dt_absolute_error_5.fit(X_train, y_train)

# Display trained models
dt_squared_error_2, dt_absolute_error_2, dt_squared_error_5, dt_absolute_error_5
```

```
(DecisionTreeRegressor(min_samples_leaf=4, random_state=42),
 DecisionTreeRegressor(criterion='absolute_error', min_samples_leaf=4,
                       random_state=42),
 DecisionTreeRegressor(min_samples_leaf=12, random_state=42),
 DecisionTreeRegressor(criterion='absolute_error', min_samples_leaf=12,
                       random_state=42))
```

Google Colab Link-

https://colab.research.google.com/drive/1-5HXe0g_OnIcHkbrRR5w95X6cv-WrnKt?usp=sharing