40100433 amir mahdi eslmai

Question 3)

(a) Overall Gini Index:

$$Gini = 1 - (0.5^2 + 0.5^2) = 0.5$$

B)

The Customer ID attribute is unique for each example, meaning each value splits the data into one example (itself) and the rest. For each Customer ID:

- If the example is in C0, the "rest" will have 9 C0 and 10 C1 examples.
- If the example is in C1, the "rest" will have 10 C0 and 9 C1 examples.

The Gini index for a split is the weighted average of the Gini indices of the resulting subsets. For Customer ID, each subset has only one example, so:

- For the subset with the example: Gini = 1−(1^2+0^2)=0 (pure).
- For the "rest" subset: Gini depends on the class distribution.

However, since Customer ID is a unique identifier, it perfectly splits the data into pure subsets (one example per subset), leading to a Gini index of 0 for each split. The weighted average Gini index for the attribute is:

$$\text{gini} = \sum_{i=1}^{20} \frac{1}{20} * 0 = 0$$

c)

The gini for Male is $1 - (0.4)^2 - (0.6)^2 = 0.48$.
The gini for Female is $1 - (0.4)^2 - (0.6)^2 = 0.48$.
Therefore, the overall gini for Gender is $0.5 \times 0.48 + 0.5 \times 0.48 = 0.48$

d)

car Type has three values: Family, Sports, and Luxury. We calculate the Gini index for each subset and then take the weighted average.

Car Type = Family:

- Total Family = 4 (Customer IDs: 1, 11, 12, 13).
  - C0: 1 → 1 example.
  - C1: 11,12,13 → 3 examples.

  Gini for family : $1 - \left(\left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) = 0.375$

Car Type = Sports:

- Total Sports = 8 (Customer IDs: 2,3,4,5,6,7,8,9).
  - C0: 2,3,4,5,6,7,8,9 → 8 examples.
  - C1: None.

  Gini for sports : $1 - (1^2 + 0) = 0$

Car Type = Luxury:

- Total Luxury = 8 (Customer IDs: 10,14,15,16,17,18,19,20).
  - C0*C*0: 10 → 1 example.
  - C1*C*1: 14,15,16,17,18,19,20 → 7 examples.

  Gini for luxury : $1 - \left(\left(\frac{1}{8}\right)^2 + \left(\frac{7}{8}\right)^2\right) = 0.2188$

Weighted gini :

Gini car type : $\left(\frac{4}{20}\right) * 0.375 + \left(\frac{8}{20}\right) * 0 + \left(\frac{8}{20}\right) * 0.2188 = 0.1625$

e)

shirt size = small :

$$1 - (\left(\frac{3}{5}\right)^2 + \left(\frac{2}{5}\right)^2) = 0.48$$

shirt size = medium :

$$1 - \left(\left(\frac{3}{7}\right)^2 + \left(\frac{4}{7}\right)^2\right) = 0.4898$$

shirt size = larg :

$$1 - \left(\left(\frac{2}{4}\right)^2 + \left(\frac{2}{4}\right)^2\right) = 0.5$$

shirt size = extra larg :

$$1 - \left(\left(\frac{2}{4}\right)^2 + \left(\frac{2}{4}\right)^2\right) = 0.5$$

weighted :

$$\left(\frac{5}{20}\right) * 48 + \left(\frac{7}{20}\right) * 0.4898 + \left(\frac{4}{20}\right) * 0.5 + \left(\frac{4}{20}\right) * 0.5 = 0.4914$$

f)

Car Type because it has the lowest gini among the three attributes.

g)

The Customer ID attribute has a Gini index of 0 because it uniquely identifies each example, perfectly splitting the data into pure subsets (each containing one example). However, this is a case of overfitting:

- Customer ID does not generalize to unseen data. It memorizes the training examples rather than learning a meaningful pattern.
- In practice, such a split would result in a useless model for prediction, as new customers will have different IDs not seen during training.

Customer ID should not be used because it is a unique identifier that overfits the training data and provides no predictive power for new examples.

Question 5)

Entropy measures the uncertainty or impurity in a dataset. A node has high entropy when the data inside it is mixed (e.g., equal number of different classes), and low entropy when it's more pure (mostly one class).

When we split a node based on an attribute (e.g., color, age, etc.), we divide the data into smaller subsets (child nodes), each of which contains data that shares a specific value of the attribute.

Each child node may have less uncertainty (i.e., lower entropy), especially if the split groups similar class labels together.

$$E(Y) = -\sum_{j=1}^{c} P(y_j) \log_2 P(y_j) = \sum_{j=1}^{c} \sum_{i=1}^{k} P(x_i, y_j) \log_2 P(y_j),$$

$$E(Y|x_i) = -\sum_{j=1}^{c} P(y_j|x_i) \log_2 P(y_j|x_i)$$

Even though some child nodes might still be impure, we take a weighted average of the entropies of all child nodes (based on how much data falls into each one). This is called the expected entropy after splitting.

This average is always less than or equal to the original entropy of the parent node.

Splitting lets us gain information about the class by grouping similar data together. In information theory, gaining information always reduces uncertainty, so entropy either:

- Decreases (we learn something useful), or
- Stays the same (the split doesn’t help, i.e., random labels remain random after the split)

But it never increases, because that would mean we're somehow losing information by looking at more data — which doesn't make sense.

$$
\begin{aligned}
E(Y|X) &= \sum_{i=1}^{\tilde{}} P(x_i) E(Y|x_i) \\
&= -\sum_{i=1}^{k} \sum_{j=1}^{c} P(x_i) P(y_j|x_i) \log_2 P(y_j|x_i) \\
&= -\sum_{i=1}^{k} \sum_{j=1}^{c} P(x_i, y_j) \log_2 P(y_j|x_i),
\end{aligned}
$$

$$
\begin{aligned}
E(Y|X) - E(Y) &= -\sum_{i=1}^{k} \sum_{j=1}^{c} P(x_i, y_j) \log_2 P(y_j|x_i) + \sum_{i=1}^{k} \sum_{j=1}^{c} P(x_i, y_j) \log_2 P(y_j) \\
&= \sum_{i=1}^{k} \sum_{j=1}^{c} P(x_i, y_j) \log_2 \frac{P(y_j)}{P(y_j|x_i)} \\
&= \sum_{i=1}^{k} \sum_{j=1}^{c} P(x_i, y_j) \log_2 \frac{P(x_i)P(y_j)}{P(x_i, y_j)}.
\end{aligned}
$$

Using Jensen's inequality (a mathematical tool for averages and convex functions), we can prove that the average entropy after the split is never greater than the original entropy. This is a formal way of expressing the idea that averaging over smaller, more specific groups can't increase the overall disorder.

Splitting a node in a decision tree never increases entropy. It either reduces it (which helps us make better decisions) or leaves it the same (which tells us the split wasn't useful).

$$
\sum_{k=1}^{d} a_k \log(z_k) \leq \log \left(\sum_{k=1}^{d} a_k z_k \right),
$$

$$E(Y|X) - E(Y) \leq \log_2 \left[\sum_{i=1}^{k} \sum_{j=1}^{c} P(x_i, y_j) \cdot \frac{P(x_i)P(y_j)}{P(x_i, y_j)} \right]$$

$$= \log_2 \left[\sum_{i=1}^{k} P(x_i) \sum_{j=1}^{c} P(y_j) \right] = \log_2(1) = 0$$

$$E(Y|X) - E(Y) \leq 0$$

Question 8)

Step 1: Calculate Classification Error Rate for Root Node

- Total examples = (5+40) + (0+15) + (10+5) + (45+0) + (10+5) + (25+0) + (5+20) + (0+15) = 200
- Majority class = C2 (40+15+5+0+5+0+20+15 = 100) vs C1 (5+0+10+45+10+25+5+0 = 100)
- Root node error rate = min(100/200, 100/200) = 0.5

Step 2: Evaluate All Possible First Splits

1. **Split on X:**
    - X=0: C1=60, C2=60 → Error=0.5
    - X=1: C1=40, C2=40 → Error=0.5
    - Weighted error = (120/200)*0.5 + (80/200)*0.5 = 0.5
2. **Split on Y:**
    - Y=0: C1=40, C2=60 → Error=40/100=0.4
    - Y=1: C1=60, C2=40 → Error=40/100=0.4
    - Weighted error = 0.4
3. **Split on Z:**
    - Z=0: C1=30, C2=70 → Error=30/100=0.3
    - Z=1: C1=70, C2=30 → Error=30/100=0.3
    - Weighted error = 0.3

Best First Split: Z (lowest error 0.3)

Step 2: Second Level Splits

For Z=0 branch (30 C1, 70 C2):

- Both X and Y splits yield error rate = (15 + 15)/100 = 0.3 (30%)
- Predictions:
    - For X=0 or Y=0: Predict C2 (Error = 15/60)
    - For X=1 or Y=1: Predict C2 (Error = 15/40)

For Z=1 branch (70 C1, 30 C2):

- Both X and Y splits yield error rate = (15 + 15)/100 = 0.3 (30%)
- Predictions:
    - For X=0 or Y=1: Predict C1 (Error = 15/60)
    - For X=1 or Y=0: Predict C1 (Error = 15/40)

Final Tree Structure:

1. First split on Z
    - If Z=0:
        - Split on X (or Y)
            - X=0: Predict C2
            - X=1: Predict C2
    - If Z=1:
        - Split on X (or Y)
            - X=0: Predict C1
            - X=1: Predict C1

Overall Error Rate: 30% (60 errors out of 200 examples)

b)

Here, x is fixed as the first splitting attribute, and the best remaining attribute (y or z) is chosen for the second level.

1. **Level 1 Splitting Attribute:** x.
    - X=0 C1 = 60, C2 = 60 → Error = min(60, 60) = 60
    - X=1: C1 = 40, C2 = 40 → Error = min(40, 40) = 40
    - Total error = 60+40=100→ Error rate = 100/200=0.5.
2. **Level 2 Splitting Attribute for X=0:**
    - **Attribute Y:**
        - Y=0: C1 = 5, C2 = 55 → Error = min(5, 55) = 5
        - Y=1: C1 = 55, C2 = 5 → Error = min(55, 5) = 5
        - Total error = 5+5=10 → Error rate = 10/120≈0.083.
    - **Attribute Z:**
        - Z=0 C1 = 15, C2 = 45 → Error = min(15, 45) = 15
        - Z=1: C1 = 45, C2 = 15 → Error = min(45, 15) = 15
        - Total error = 15+15=30 → Error rate = 30/120=0.25
          Conclusion: Choose y for X=0 (lower error rate).
3. **Level 2 Splitting Attribute for X=1:**
    - **Attribute Y:**
        - Y=0: C1 = 35, C2 = 5 → Error = min(35, 5) = 5
        - Y=1: C1 = 5, C2 = 35 → Error = min(5, 35) = 5
        - Total error = 5+5=10 → Error rate = 10/80=0.125
    - **Attribute Z:**
        - Z=0: C1 = 15, C2 = 25 → Error = min(15, 25) = 15
        - Z=1: C1 = 25, C2 = 15 → Error = min(25, 15) = 15
        - Total error = 15+15=30 → Error rate = 30/80=0.375
          Conclusion: Choose y for X=1 (lower error rate).
4. **Induced Tree Structure:**
    - Root: x
        - X=0 Split on y

        - Y=0Y=0: Predict c2 (error = 5)
        - Y=1Y=1: Predict c1 (error = 5)
    - X=1: Split on y
        - Y=0*:* Predict c1 (error = 5)
        - Y=1: Predict c2 (error = 5)

Overall Error Rate:

- Total errors = 5+5+5+5=20 → Error rate = 20/200=0.1.

c)

Answer:

- Part (a): Error rate = 0.3 (using greedy selection of z first).
- Part (b): Error rate = 0.1 (using x first, then y).

Comparison and Suitability of Greedy Heuristic:

1. The greedy heuristic in part (a) selects z as the root due to its immediate lowest error rate (0.3), but this leads to a suboptimal tree with higher error compared to part (b).
2. Part (b) shows that forcing x as the root and then selecting y for splits yields a significantly better tree (error rate = 0.1).
3. Limitation of Greedy Heuristic: The greedy approach myopically optimizes for the immediate split without considering future splits, which can result in globally suboptimal trees.
4. Suitability: While the greedy heuristic is computationally efficient and works well in many cases, this example demonstrates that it may not always find the optimal solution. Alternatives like looking ahead or using global optimization criteria (e.g., information gain) could mitigate this issue.

Conclusion: The greedy heuristic is practical but not foolproof; its performance depends on the data distribution and the chosen splitting criterion.

Question 9)

a)

The classification error rate for a node is calculated as:

$$\text{Error}=1-\max(p,1-p)$$

where p is the proportion of the majority class in the node.

Total instances:100

Base error rate (before split):
Majority class is "-" with 50 instances, "+" with 50 instances.

$$\text{Base Error}=1-\max(0.5,0.5)=0.5$$

For each attribute, calculate the weighted error rate after splitting:

1. **Attribute A:**
    - **A = T:** Instances: (5, 0, 20, 0, 0, 0, 0, 0) → "+" = 25, "-" = 0 → Error = 1 - 25/25 = 0
    - **A = F:** Instances: (0, 20, 0, 5, 25, 0, 0, 25) → "+" = 25, "-" = 50 → Error = 1 - 50/75 = 0.333
    - **Weighted Error for A:** $(25/100)\times0+(75/100)\times0.333=0.25$
    - **Gain in Error Rate:** $0.5-0.25=0.250.5-0.25=0.25$
2. **Attribute B:**
    - **B = T:** Instances: (5, 0, 0, 20, 0, 25, 0, 0) → "+" = 30, "-" = 20 → Error = 1 - 30/50 = 0.4
    - **B = F:** Instances: (20, 0, 0, 5, 0, 0, 0, 25) → "+" = 20, "-" = 30 → Error = 1 - 30/50 = 0.4
    - **Weighted Error for B:** $(50/100)\times0.4+(50/100)\times0.4=0.4$
    - **Gain in Error Rate:** $0.5-0.4=0.1$
3. **Attribute C:**

- **C = T:** Instances: (5, 0, 20, 0, 0, 20, 0, 5) → "+" = 25, "-" = 25 → Error = 1 - 25/50 = 0.5
- **C = F:** Instances: (0, 0, 0, 0, 25, 0, 0, 25) → "+" = 25, "-" = 25 → Error = 1 - 25/50 = 0.5
- **Weighted Error for C:** $(50/100)\times0.5+(50/100)\times0.5=0.5$
- **Gain in Error Rate:** $0.5-0.5=0$

Conclusion for (a): Attribute A has the highest gain in error rate (0.25), so it is chosen as the first splitting attribute.

(b) Splitting the two children of the root node (A)

1. **A = T:**
   - Instances: (T,T,T)=5+, (T,F,T)=20+, (T,T,F)=0, (T,F,F)=0 → All "+" (25 instances).
   - Pure node (all "+"), no further splitting needed.
2. **A = F:**
   - Instances: (F,T,T)=20-, (F,F,T)=5-, (F,T,F)=25+, (F,F,F)=25- → "+"=25, "-"=50.
   - Calculate error rate for splitting on B or C:
     - **Base Error for A=F:** $1-50/75=0.333$
     - **Split on B:**
       - B=T: (F,T,T)=20-, (F,T,F)=25+ → "+"=25, "-"=20 → Error = 1 - 25/45 ≈ 0.444.
       - B=F: (F,F,T)=5-, (F,F,F)=25- → "+"=0, "-"=30 → Error = 1 - 30/30 = 0.
       - Weighted Error: $(45/75)\times0.444+(30/75)\times0=0.266$
       - Gain: $0.333-0.266=0.067$
     - **Split on C:**
       - C=T: (F,T,T)=20-, (F,F,T)=5- → "+"=0, "-"=25 → Error = 0.
       - C=F: (F,T,F)=25+, (F,F,F)=25- → "+"=25, "-"=25 → Error = 0.5.
       - Weighted Error: $(25/75)\times0+(50/75)\times0.5\approx0.333$

        - Gain: 0.333−0.333=0
    - **Conclusion:** Split on B (higher gain).

(c) Misclassified instances in the resulting tree

- **Root (A):**
    - A=T: All 25 instances correctly classified as "+".
    - A=F: Split on B:
        - B=T: Predict "+" (majority), but 20 "-" are misclassified.
        - B=F: Predict "-", all 30 correctly classified.
- **Total Misclassified:** 20 (from B=T under A=F) and total rate is 20/100

(d)

Step 1: First Split on C

The data is split into two child nodes based on C*C*:

1. **C=T:**
    - Instances:
      (A=T,B=T,C=T):5+
      (A=F,B=T,C=T):20−
      (A=T,B=F,C=T):20+
      (A=F,B=F,C=T):5−
    - Total: 25+, 25-
    - **Initial Error Rate:**
      Eorig=1−max(0.5,0.5)=0.5
2. **C=F:**
    - Instances:
      (A=F,B=T,C=F):25+
      (A=F,B=F,C=F):25−
    - Total: 25+, 25−.

- **Initial Error Rate:**
  Eorig=1−max(0.5,0.5)=0.5

Step 2: Splitting the C=T Node

For the C=T node, we evaluate splits on A and B:

1. **Split on *A*:**
   - **A=T:**
     Instances: (T,T,T)=5+ (T,F,T)=20+
     Total: 25+,0−
     Error: 0
   - **A=F:**
     Instances: (F,T,T)=20−, (F,F,T)=5−
     Total: 0+, 25-
     Error: 0
   - **Weighted Error:**
     0
   - **Gain in Error Rate:**
     0.5
2. **Split on B:**
   - **B=T:**
     Instances: (T,T,T)=5+, (F,T,T)=20−(-
     Total: 5+, 20−
     Error: 0.2
   - **B=F:**
     Instances: (T,F,T)=20+, (F,F,T)=5−
     Total: 20+, 5−
     Error:0.2
   - **Weighted Error:**
     0.2
   - **Gain in Error Rate:**
     0.3

Conclusion for C=T:
Split on A (higher gain of 0.50 vs. 0.30 for B).

Step 3: Splitting the C=F Node

For the C=F*C=F* node, we evaluate splits on A*A* and B*B*:

1. **Split on A:**
    - **A=T:**
      No instances (all A=Ffor *C=F*).
      Error: Undefined (no split possible).
    - **A=F:**
      All instances: (F,T,F)=25+, (F,F,F)=25−
      Error: 0.5
    - **Gain in Error Rate:**
      No improvement (cannot split on A).
2. **Split on B:**
    - **B=T:**
      Instances: (F,T,F)=25+
      Total: 25+,0-
      Error: EB=T=0.
    - **B=F:**
      Instances: (F,F,F)=25−
      Total: 0+, 25−
      Error: EB=F=0
    - **Weighted Error:**
      0
    - **Gain in Error Rate:**
      0.5

Conclusion for C=F:
Split on B (gain of 0.50).

Step 4: Resulting Decision Tree

1. **Root:** Split on C.
    - **C=T:** Split on A.
        - A=T: Predict +.
        - A=F: Predict −.
    - **C=F:** Split on B.
        - B=T: Predict +.
        - B=F: Predict −.

Step 5: Misclassified Instances

- **C=T, A=T:**
  25+, 0−→ All correct.
- **C=T, A=F:**
  0+, 25− → All correct.
- **C=F, B=T:**
  25+, 0− → All correct.
- **C=F, *B=F*:**
  0+, 25− → All correct.

Total Misclassified: 0.

(e) Conclusion about the greedy nature of the algorithm

The greedy algorithm does not always find the globally optimal tree (e.g., a tree splitting on B first might be better, but the greedy choice is A