MIDTERM MOCK TEST (with solution)

Shiba Inu Team

E-commerce Customer Satisfaction Prediction

Consider the following dataset for an e-commerce company that wants to predict customer satisfaction based on product reviews. Answer 10 questions (each question is worth 10 marks) by showing your detailed work step-by-step.

Training Dataset (8 entries):

ID	Text Sentiment	Product Visibility	Image Quality	Satisfaction
1	Positive	High	Good	Satisfied
2	Negative	High	Fair	Unsatisfied
3	Neutral	Low	Poor	Unsatisfied
4	Positive	Medium	Fair	Satisfied
5	Negative	Low	Good	Unsatisfied
6	Neutral	High	Good	Satisfied
7	Positive	High	Poor	Satisfied
8	Negative	Medium	Poor	Unsatisfied

Test Dataset (3 entries):

ID	Text Sentiment	Product Visibility	Image Quality	Satisfaction
9	Positive	Low	Good	?
10	Neutral	Medium	Fair	?
11	Negative	Low	Poor	?

Question 1:

A machine learning model made the following predictions on the training dataset:

ID	Actual Satisfaction	Predicted Satisfaction	
1	Satisfied	Satisfied	
2	Unsatisfied	Unsatisfied	
3	Unsatisfied	Unsatisfied	
4	Satisfied	Satisfied	
5	Unsatisfied	Satisfied	
6	Satisfied	Satisfied	
7	Satisfied	Unsatisfied	
8	Unsatisfied	Unsatisfied	

a) Construct the confusion matrix for this model.

b) Calculate the accuracy, precision, recall, and F1-score for the "Satisfied" class. Which metric would be most appropriate if we want to ensure customers with satisfaction issues are properly identified? Explain your reasoning.

ANSWER:

a) Confusion Matrix

	Predicted Satisfied	**Predicted Unsatisfied**
Actual Satisfied	3 (TP)	1 (FN)
Actual Unsatisfied	1 (FP)	3 (TN)

Where:

- True Positives (TP): Cases 1, 4, 6
- False Negatives (FN): Case 7
- False Positives (FP): Case 5
- True Negatives (TN): Cases 2, 3, 8

b) Metrics Calculation for "Satisfied" Class

Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{3+3}{3+3+1+1} = \frac{6}{8} = 0.75 = 75\%$$

Precision:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{3}{3+1} = \frac{3}{4} = 0.75 = 75\%$$

Recall:

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{3}{3+1} = \frac{3}{4} = 0.75 = 75\%$$

F1-Score:

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times 0.75 \times 0.75}{0.75 + 0.75} = \frac{2 \times 0.5625}{1.5} = \frac{1.125}{1.5} = 0.75 = 75\%$$

Most Appropriate Metric:

If we want to ensure customers with satisfaction issues are properly identified, the most appropriate metric would be **recall for the "Unsatisfied" class** (equivalent to specificity for the "Satisfied" class).

Recall for "Unsatisfied" class:

$$\text{Recall}_{\text{Unsatisfied}} = \frac{TN}{TN + FP} = \frac{3}{3+1} = 0.75 = 75\%$$

Reasoning:
When focusing on identifying customers with satisfaction issues, we want to minimize the chances of missing any unsatisfied customers (false negatives for the "Unsatisfied" class). Recall measures the model's ability to find all relevant instances in a class. A high recall for the "Unsatisfied" class means we're correctly identifying most customers who have satisfaction issues, allowing the company to address their concerns and potentially prevent churn.

Question 2:

a) Calculate the initial entropy of the training dataset with respect to Customer Satisfaction.

b) Calculate the information gain for each of the three features (Text Sentiment, Product Visibility, and Image Quality). Which feature should be selected as the first split in a decision tree? Show all your calculations.

ANSWER:

a) Initial Entropy Calculation

First, we'll count the number of instances for each satisfaction class in the training dataset:

- Satisfied: 4 instances (IDs 1, 4, 6, 7)
- Unsatisfied: 4 instances (IDs 2, 3, 5, 8)
- Total instances: 8

The entropy formula is:

$$E(S) = -\sum_{i} p_i \log_2(p_i)$$

Where:

- p_i is the proportion of instances belonging to class i

Calculating:

$$p_{Satisfied} = \frac{4}{8} = 0.5$$

$$p_{Unsatisfied} = \frac{4}{8} = 0.5$$

$$E(S) = -[0.5 \times \log_2(0.5) + 0.5 \times \log_2(0.5)]$$

$$E(S) = -[0.5 \times (-1) + 0.5 \times (-1)]$$

$$E(S) = -[-0.5 - 0.5] = -[-1] = 1$$

Therefore, the initial entropy of the training dataset is **1.0**.

b) Information Gain Calculation

For each feature, we'll:

1. Split the dataset based on feature values
2. Calculate entropy for each subset
3. Calculate weighted entropy
4. Calculate information gain = Initial Entropy - Weighted Entropy

Text Sentiment

Text Sentiment has 3 values:

1. Positive (3 instances):

- Satisfied: 3 (IDs 1, 4, 7)
- Unsatisfied: 0
- $p_{Satisfied} = 3/3 = 1$, $p_{Unsatisfied} = 0/3 = 0$
- $E(Positive) = -[1 \times \log_2(1) + 0 \times \log_2(0)] = 0$

2. Negative (3 instances):

- Satisfied: 0
- Unsatisfied: 3 (IDs 2, 5, 8)
- $p_{Satisfied} = 0/3 = 0$, $p_{Unsatisfied} = 3/3 = 1$
- $E(Negative) = -[0 \times \log_2(0) + 1 \times \log_2(1)] = 0$

3. Neutral (2 instances):

- Satisfied: 1 (ID 6)
- Unsatisfied: 1 (ID 3)
- $p_{Satisfied} = 1/2 = 0.5$, $p_{Unsatisfied} = 1/2 = 0.5$
- $E(Neutral) = -[0.5 \times \log_2(0.5) + 0.5 \times \log_2(0.5)] = 1$

Weighted Entropy:

$$H_{TextSentiment} = \frac{3}{8} \times 0 + \frac{3}{8} \times 0 + \frac{2}{8} \times 1 = \frac{2}{8} = 0.25$$

Information Gain:

$$IG_{TextSentiment} = 1 - 0.25 = 0.75$$

Product Visibility

High (4 instances):

- Satisfied: 3 (IDs 1, 6, 7)
- Unsatisfied: 1 (ID 2)
- $p_{Satisfied} = 3/4 = 0.75$, $p_{Unsatisfied} = 1/4 = 0.25$
- $E(High) = -[0.75 \times \log_2(0.75) + 0.25 \times \log_2(0.25)]$
- $E(High) = -[0.75 \times (-0.415) + 0.25 \times (-2)] = 0.811$

Medium (2 instances):

- Satisfied: 1 (ID 4)
- Unsatisfied: 1 (ID 8)

- $E(Medium) = 1$

Low (2 instances):

- Satisfied: 0
- Unsatisfied: 2 (IDs 3, 5)
- $E(Low) = 0$

Weighted Entropy:

$$H_{ProductVisibility} = \frac{4}{8} \times 0.811 + \frac{2}{8} \times 1 + \frac{2}{8} \times 0 = 0.6555$$

Information Gain:

$$IG_{ProductVisibility} = 1 - 0.6555 = 0.3445$$

Image Quality

Good (3 instances):

- Satisfied: 2 (IDs 1, 6)
- Unsatisfied: 1 (ID 5)
- $p_{Satisfied} = 2/3 \approx 0.667$, $p_{Unsatisfied} = 1/3 \approx 0.333$
- $E(Good) = -[0.667 \times \log_2(0.667) + 0.333 \times \log_2(0.333)] \approx 0.918$

Fair (2 instances):

- Satisfied: 1 (ID 4)
- Unsatisfied: 1 (ID 2)
- $E(Fair) = 1$

Poor (3 instances):

- Satisfied: 1 (ID 7)
- Unsatisfied: 2 (IDs 3, 8)
- $E(Poor) = -[0.333 \times \log_2(0.333) + 0.667 \times \log_2(0.667)] \approx 0.918$

Weighted Entropy:

$$E_{ImageQuality} = \frac{3}{8} \times 0.918 + \frac{2}{8} \times 1 + \frac{3}{8} \times 0.918 = 0.9375$$

Information Gain:

$$IG_{ImageQuality} = 1 - 0.9375 = 0.0625$$

Summary of Information Gain:

- Text Sentiment: 0.75
- Product Visibility: 0.3445
- Image Quality: 0.0625

Feature Selection: Text Sentiment should be selected as the first split in a decision tree because it has the highest information gain (0.75). This feature provides the most reduction in entropy and therefore the most information for classifying customer satisfaction.

Question 3:

Suppose in the training dataset, entry #6 is missing its "Product Visibility" value.

a) Describe three different strategies for handling this missing value, and explain the potential impact of each strategy on the model's performance.

b) If you choose to predict the missing value using a decision tree based on the other two features, construct a small decision tree to predict "Product Visibility" and determine what value would be assigned to entry #6. Show your work.

ANSWER:

a) Strategies for Handling the Missing "Product Visibility" Value

1. Mean/Mode Imputation

We could replace the missing value with the most frequent value (mode) of Product Visibility from the training dataset.

Implementation: The mode of Product Visibility is "High" (occurs 3 times out of 7), so we would assign "High" to entry #6.

Potential Impact:

- **Advantages**: Simple to implement and preserves the original data distribution.
- **Disadvantages**: Reduces variance in the data and may introduce bias if the missing value mechanism is not random. Could lead to overfitting toward the majority class.

2. Predictive Imputation

We could use the other features (Text Sentiment, Image Quality, and Satisfaction) to build a model that predicts the missing Product Visibility value.

Implementation: Train a small model using the other entries to predict Product Visibility based on the other features.

Potential Impact:

- **Advantages**: Considers relationships between features, potentially leading to more accurate imputation.
- **Disadvantages**: Adds complexity and may introduce bias if the prediction model is inaccurate. Creates a dependency between features that might not actually exist.

3. Creating a "Missing" Category

We could create a new category called "Missing" or "Unknown" for the Product Visibility feature.

Implementation: Replace the missing value with a new category "Missing" or "Unknown".

Potential Impact:

- **Advantages**: Preserves the information that a value is missing, which might itself be meaningful.
- **Disadvantages**: Creates an additional category that doesn't exist in the real world, potentially leading to model confusion, especially in this small dataset where one "Missing" value would represent a significant portion of the data.

b) Decision Tree Prediction for the Missing Value

We'll build a small decision tree to predict Product Visibility based on Text Sentiment and Image Quality for entry #6.

First, let's organize what we know from the training data (excluding entry #6):

ID	Text Sentiment	Product Visibility	Image Quality
1	Positive	High	Good
2	Negative	High	Fair
3	Neutral	Low	Poor
4	Positive	Medium	Fair
5	Negative	Low	Good
7	Positive	High	Poor

ID	Text Sentiment	Product Visibility	Image Quality
8	Negative	Medium	Poor

Looking at the distribution of Product Visibility:

- High: 3 instances
- Medium: 2 instances
- Low: 2 instances

We need to determine which feature (Text Sentiment or Image Quality) should be the root node of our decision tree by calculating information gain.

Initial Entropy of Product Visibility:

$$E(S) = -[\frac{3}{7}\log_2(\frac{3}{7}) + \frac{2}{7}\log_2(\frac{2}{7}) + \frac{2}{7}\log_2(\frac{2}{7})]$$

$$E(S) = -[0.429 \times (-1.222) + 0.286 \times (-1.807) + 0.286 \times (-1.807)]$$

$$E(S) = -[-0.524 - 0.517 - 0.517] = 1.558$$

Information Gain for Text Sentiment:

Positive (3 instances):

- High: 2 (IDs 1, 7)
- Medium: 1 (ID 4)
- Low: 0
- Entropy: $-[\frac{2}{3}\log_2(\frac{2}{3}) + \frac{1}{3}\log_2(\frac{1}{3})] = 0.918$

Negative (3 instances):

- High: 1 (ID 2)
- Medium: 1 (ID 8)
- Low: 1 (ID 5)
- Entropy: $-[\frac{1}{3}\log_2(\frac{1}{3}) + \frac{1}{3}\log_2(\frac{1}{3}) + \frac{1}{3}\log_2(\frac{1}{3})] = 1.585$

Neutral (1 instance):

- High: 0
- Medium: 0
- Low: 1 (ID 3)
- Entropy: 0

Weighted Entropy:

$$E_{Text} = \frac{3}{7} \times 0.918 + \frac{3}{7} \times 1.585 + \frac{1}{7} \times 0 = 1.073$$

Information Gain:

$$IG_{Text} = 1.558 - 1.073 = 0.485$$

Information Gain for Image Quality:

Good (2 instances):

- High: 1 (ID 1)
- Medium: 0
- Low: 1 (ID 5)
- Entropy: $-[\frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2})] = 1$

Fair (2 instances):

- High: 1 (ID 2)
- Medium: 1 (ID 4)
- Low: 0
- Entropy: 1

Poor (3 instances):

- High: 1 (ID 7)
- Medium: 1 (ID 8)
- Low: 1 (ID 3)
- Entropy: 1.585

Weighted Entropy:

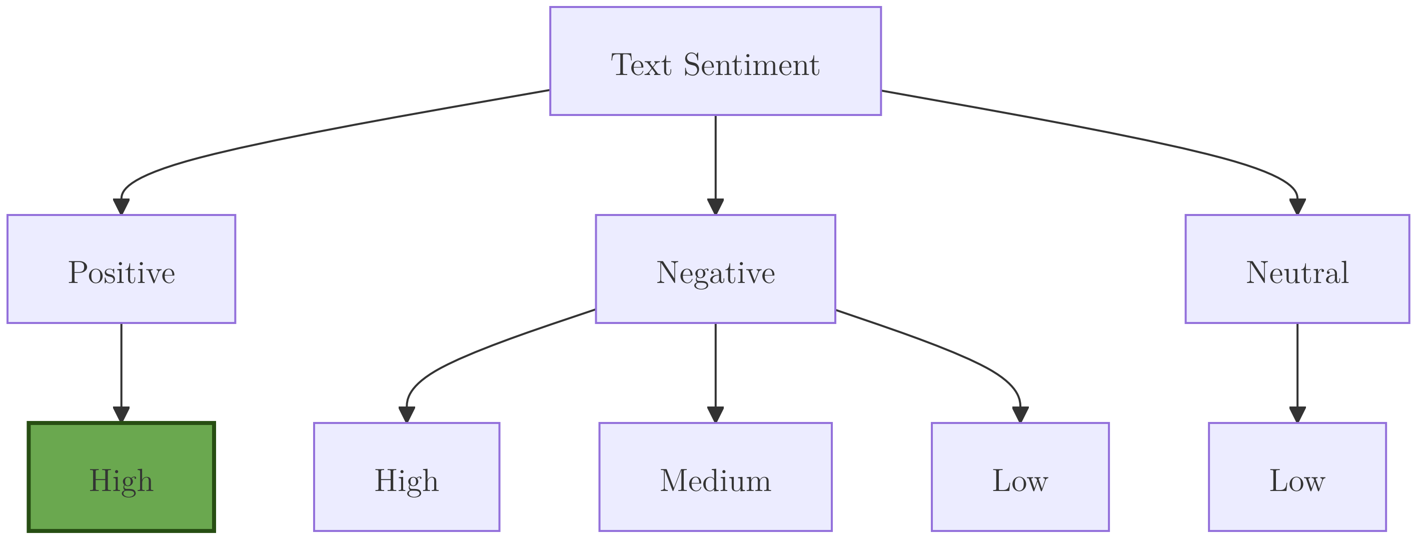
$$E_{Image} = \frac{2}{7} \times 1 + \frac{2}{7} \times 1 + \frac{3}{7} \times 1.585 = 1.253$$

Information Gain:

$$IG_{Image} = 1.558 - 1.253 = 0.305$$

Since Text Sentiment has the higher information gain ($0.485 > 0.305$), we'll use it as the root node of our decision tree.

Our decision tree would look like:



For entry #6, the Text Sentiment is "Neutral", so following our decision tree, we would predict "Low" for its Product Visibility.

However, if we consider that there's only one example of "Neutral" in our training data, this prediction has limited confidence. In a practical scenario, we might want to incorporate domain knowledge or other features to improve the prediction.

Question 4:

a) Design a single perceptron to classify the data into "Satisfied" or "Unsatisfied" based on the three features. Assign appropriate numerical values to the categorical features and explain your encoding scheme. Draw the perceptron with initial small random values.

b) Explain why a single perceptron might not be able to properly classify this dataset. Design a minimal neural network (with input layer, one hidden layer, and output layer) that could better handle this classification task. Specify the activation functions you would use and explain your choices.

ANSWER

a) Single Perceptron Classification

Numerical Encoding of Features

We'll first convert categorical features into numerical values:

Text Sentiment:

- Positive: 1
- Neutral: 0
- Negative: -1

Product Visibility:

- High: 2
- Medium: 1
- Low: 0

Image Quality:

- Good: 2
- Fair: 1
- Poor: 0

Satisfaction (output):

- Satisfied: 1
- Unsatisfied: 0

Encoding Explanation:
This encoding scheme preserves the natural ordering of categories. For Text Sentiment, positive is better than neutral, which is better than negative. Similarly, for Product Visibility and Image Quality, the values represent decreasing levels of quality or prominence. This ordinal encoding helps maintain the relative relationships between categories.

Perceptron Design

The perceptron combines inputs (features) using weights and a bias term:

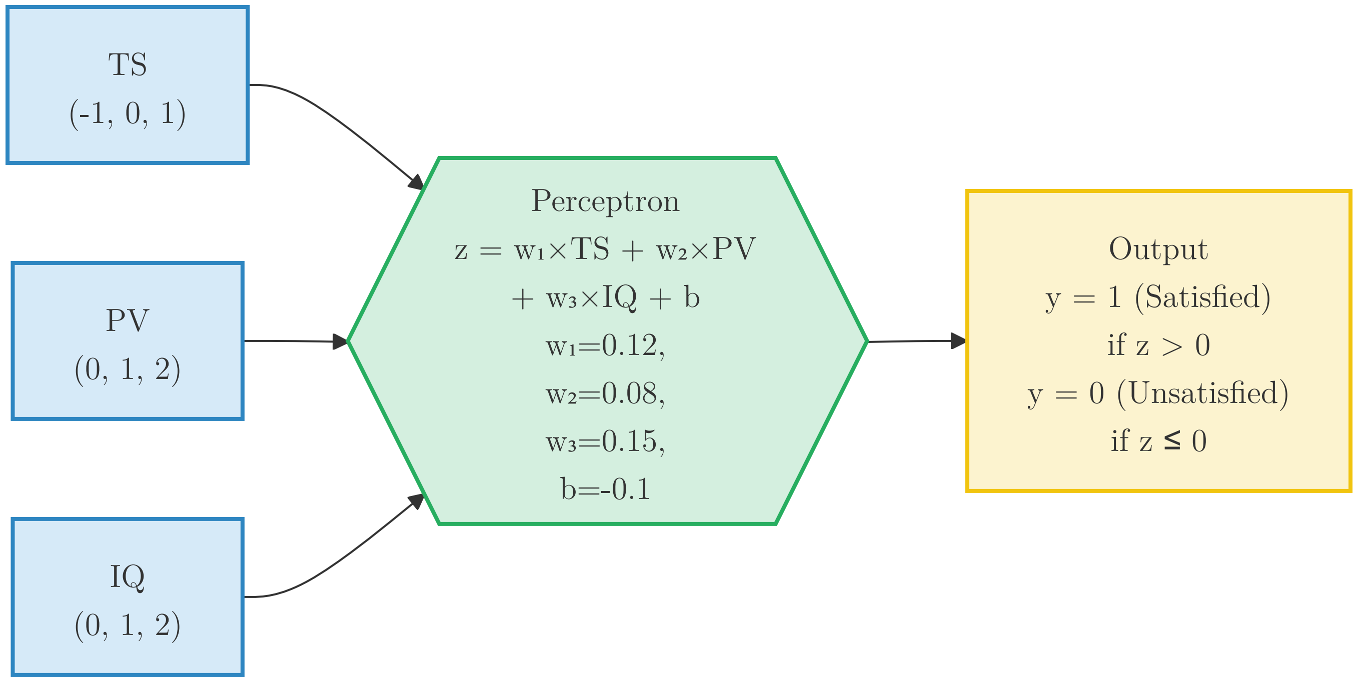
$$z = w_1 \times \text{Text Sentiment} + w_2 \times \text{Product Visibility} + w_3 \times \text{Image Quality} + b$$

With activation function:

$$y = \begin{cases} 1 \text{ (Satisfied)} & \text{if } z > 0 \\ 0 \text{ (Unsatisfied)} & \text{if } z \leq 0 \end{cases}$$

Initial weights and bias (small random values):

- $w_1 = 0.12$ (Text Sentiment)
- $w_2 = 0.08$ (Product Visibility)
- $w_3 = 0.15$ (Image Quality)
- $b = -0.1$ (bias)


TS
(-1, 0, 1)
PV
(0, 1, 2)
IQ
(0, 1, 2)
Perceptron
$z = w_1 \times TS + w_2 \times PV + w_3 \times IQ + b$
$w_1=0.12$,
$w_2=0.08$,
$w_3=0.15$,
$b=-0.1$
Output
$y = 1$ (Satisfied) if $z > 0$
$y = 0$ (Unsatisfied) if $z \leq 0$

b) Limitations and Neural Network Design

Limitations of a Single Perceptron

A single perceptron can only learn linearly separable patterns. It creates a straight line (or hyperplane in higher dimensions) to separate classes. Our dataset likely contains non-linear patterns, as evidenced by:

1. Entry #6 shows a customer with Neutral sentiment but High visibility and Good image quality is Satisfied
2. Entry #7 shows a customer with Poor image quality but Positive sentiment and High visibility is Satisfied

These complex relationships suggest that simple linear separation won't work well.

Neural Network Design

We'll design a minimal neural network with:

- **Input layer**: 3 neurons (one for each feature)
- **Hidden layer**: 4 neurons (to capture non-linear relationships)
- **Output layer**: 1 neuron (for binary classification)

Activation Functions:

1. **Hidden Layer Activation**: ReLU (Rectified Linear Unit)

$$f(x) = max(0, x)$$

   Justification: ReLU introduces non-linearity while being computationally efficient. It helps solve the vanishing gradient problem that can occur with other functions like sigmoid when used in hidden layers. It allows the model to learn more complex patterns than a single perceptron could.

2. **Output Layer Activation**: Sigmoid

$$\sigma(x) = 1/(1 + e^(-x))$$

   Justification: Sigmoid squashes values between 0 and 1, making it ideal for binary classification. The output can be interpreted as the probability of the customer being satisfied. If the value is greater than 0.5, we classify as "Satisfied"; otherwise, "Unsatisfied".

This architecture provides enough complexity to capture the non-linear relationships in our data while remaining relatively simple and interpretable. The hidden layer allows the model to learn feature interactions that a single perceptron cannot represent.

Question 5:

Consider a simple neural network with:

- 3 input nodes (one for each feature, appropriately encoded)
- 2 hidden nodes with sigmoid activation
- 1 output node with sigmoid activation for binary classification

a) Starting with the weights given below, perform one step of forward propagation for training example #1:

- w_input_to_hidden = $[[0.1, 0.2], [0.3, -0.1], [0.2, 0.1]]$
- w_hidden_to_output = $[[0.4, -0.3]]$
- Encode Positive = 1, Neutral = 0, Negative = -1 for Text Sentiment
- Encode High = 1, Medium = 0.5, Low = 0 for Product Visibility
- Encode Good = 1, Fair = 0.5, Poor = 0 for Image Quality

Show all calculations and the final output.

b) If the actual target for example #1 is 1 (Satisfied), calculate the loss using binary cross-entropy and perform one step of backpropagation to update the weight from the first hidden node to the output node. Use a learning rate of 0.1. Show all your calculations.

ANSWER:

a) Forward Propagation for Training Example #1

First, we'll identify the feature values for example #1:

- Text Sentiment: Positive $\rightarrow$ 1
- Product Visibility: High $\rightarrow$ 1
- Image Quality: Good $\rightarrow$ 1

So our input vector is [1, 1, 1].

Step 1: Calculate inputs to hidden layer nodes

Using the given weights:

$$w_{\text{input\_to\_hidden}} = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & -0.1 \\ 0.2 & 0.1 \end{bmatrix}$$

For hidden node 1:

$$z_{h1} = (1 \times 0.1) + (1 \times 0.3) + (1 \times 0.2) = 0.6$$

For hidden node 2:

$$z_{h2} = (1 \times 0.2) + (1 \times -0.1) + (1 \times 0.1) = 0.2$$

Step 2: Apply sigmoid activation function to hidden nodes

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

For hidden node 1:

$$a_{h1} = \sigma(0.6) = \frac{1}{1 + e^{-0.6}} = \frac{1}{1.5488} \approx 0.6457$$

For hidden node 2:

$$a_{h2} = \sigma(0.2) = \frac{1}{1 + e^{-0.2}} = \frac{1}{1.8187} \approx 0.5499$$

Step 3: Calculate input to output node

Using hidden-to-output weights:

$$w_{\text{hidden\_to\_output}} = \begin{bmatrix} 0.4 & -0.3 \end{bmatrix}$$

For output node:

$$z_o = (0.6457 \times 0.4) + (0.5499 \times -0.3) = 0.2583 - 0.1650 = 0.0933$$

Step 4: Apply sigmoid activation to output node

$$a_o = \sigma(0.0933) = \frac{1}{1 + e^{-0.0933}} = \frac{1}{1.9109} \approx 0.5233$$

The final output is **0.5233**, which represents the probability of the customer being satisfied.

b) Loss Calculation and Backpropagation

Step 1: Calculate Binary Cross-Entropy Loss

For a binary classifier with true label y and prediction p:

$$L = -[y \times \log(p) + (1 - y) \times \log(1 - p)]$$

Since the actual target is 1 (Satisfied):

$$L = -[1 \times \log(0.5233) + 0 \times \log(1 - 0.5233)] = -\log(0.5233) \approx 0.6487$$

Step 2: Backpropagation to Update Weight

We'll update the weight from the first hidden node to the output (w_{h1_o}), currently 0.4.

First, we calculate the gradient of the loss with respect to the output:

$$\frac{dL}{da_o} = -\frac{y}{a_o} + \frac{1-y}{1-a_o} = -\frac{1}{0.5233} + \frac{0}{1 - 0.5233} \approx -1.9110$$

Next, the gradient of the output with respect to its input (derivative of sigmoid):

$$\frac{da_o}{dz_o} = a_o \times (1 - a_o) = 0.5233 \times (1 - 0.5233) = 0.5233 \times 0.4767 \approx 0.2495$$

Then, the gradient of the loss with respect to z_o:

$$\frac{dL}{dz_o} = \frac{dL}{da_o} \times \frac{da_o}{dz_o} = -1.9110 \times 0.2495 \approx -0.4768$$

Now, calculate the gradient of the loss with respect to the weight from the first hidden node to the output:

$$\frac{dL}{dw_{h1\_o}} = \frac{dL}{dz_o} \times a_{h1} = -0.4768 \times 0.6457 \approx -0.3078$$

Finally, update the weight using learning rate = 0.1:

$$w_{h1\_o\_new} = w_{h1\_o\_old} - \text{learning\_rate} \times \frac{dL}{dw_{h1\_o}} = 0.4 - 0.1 \times (-0.3078) = 0.4 + 0.0308 = 0.4308$$

After one step of backpropagation, the updated weight from the first hidden node to the output node is **0.4308**.

Question 6:

a) Using the training dataset, calculate the prior probabilities P(Satisfied) and P(Unsatisfied).

b) Using Naive Bayes, calculate the probability that test example #9 is "Satisfied" vs. "Unsatisfied." Apply Laplace smoothing with α=1 to handle zero probabilities. Show all your calculations and determine the final prediction.

ANSWER:

a) Prior Probabilities

To calculate the prior probabilities, we'll count the occurrences of each class in the training dataset:

- Total instances: 8
- Satisfied instances: 4 (IDs 1, 4, 6, 7)
- Unsatisfied instances: 4 (IDs 2, 3, 5, 8)

$$P(Satisfied) = \frac{4}{8} = 0.5$$

$$P(Unsatisfied) = \frac{4}{8} = 0.5$$

b) Naive Bayes Classification for Test Example #9

Test example #9 has the following features:

- Text Sentiment: Positive
- Product Visibility: Low
- Image Quality: Good

We'll use Naive Bayes with Laplace smoothing (α=1) to classify this example.

Step 1: Calculate conditional probabilities for each feature

For Text Sentiment = Positive:

- Among Satisfied customers: 3 out of 4 have Positive sentiment
- Among Unsatisfied customers: 0 out of 4 have Positive sentiment

With Laplace smoothing (α=1 and k=3 possible values for Text Sentiment):

$$P(Positive|Satisfied) = \frac{3+1}{4+3} = \frac{4}{7} \approx 0.571$$

$$P(Positive|Unsatisfied) = \frac{0+1}{4+3} = \frac{1}{7} \approx 0.143$$

For Product Visibility = Low:

- Among Satisfied customers: 0 out of 4 have Low visibility
- Among Unsatisfied customers: 2 out of 4 have Low visibility

With Laplace smoothing (α=1 and k=3 possible values for Product Visibility):

$$P(Low|Satisfied) = \frac{0+1}{4+3} = \frac{1}{7} \approx 0.143$$

$$P(Low|Unsatisfied) = \frac{2+1}{4+3} = \frac{3}{7} \approx 0.429$$

For Image Quality = Good:

- Among Satisfied customers: 2 out of 4 have Good image quality
- Among Unsatisfied customers: 1 out of 4 have Good image quality

With Laplace smoothing (α=1 and k=3 possible values for Image Quality):

$$P(Good|Satisfied) = \frac{2+1}{4+3} = \frac{3}{7} \approx 0.429$$

$$P(Good|Unsatisfied) = \frac{1+1}{4+3} = \frac{2}{7} \approx 0.286$$

Step 2: Apply Naive Bayes formula

For Satisfied:

$$P(X|Satisfied) \times P(Satisfied) = P(Positive|Satisfied) \times P(Low|Satisfied) \times P(Good|Satisfied) \times$$

$$= \frac{4}{7} \times \frac{1}{7} \times \frac{3}{7} \times 0.5 = \frac{12}{343} \times 0.5 = \frac{6}{343} \approx 0.0175$$

For Unsatisfied:

$$P(X|Unsatisfied) \times P(Unsatisfied) = P(Positive|Unsatisfied) \times P(Low|Unsatisfied) \times P(Good|U$$

$$= \frac{1}{7} \times \frac{3}{7} \times \frac{2}{7} \times 0.5 = \frac{6}{343} \times 0.5 = \frac{3}{343} \approx 0.0087$$

Step 3: Normalize to get posterior probabilities

$$P(Satisfied|X) = \frac{\frac{6}{343}}{\frac{6}{343} + \frac{3}{343}} = \frac{6}{9} = \frac{2}{3} \approx 0.667$$

$$P(Unsatisfied|X) = \frac{\frac{3}{343}}{\frac{6}{343} + \frac{3}{343}} = \frac{3}{9} = \frac{1}{3} \approx 0.333$$

Final Prediction

Since $P(Satisfied|X) > P(Unsatisfied|X)$, we predict that test example #9 is **Satisfied**.

The calculation shows that given these feature values, there is approximately a 66.7% probability that the customer will be satisfied, versus a 33.3% probability of being unsatisfied.

Question 7:

a) Explain what the Bayes Optimal Classifier is and how it relates to the Naive Bayes classifier. What assumptions does Naive Bayes make that the Bayes Optimal Classifier doesn't?

b) For our dataset, identify a specific case where the independence assumption of Naive Bayes might be violated. Calculate the joint probability P(Text Sentiment, Product Visibility | Satisfaction) for this case both with and without the independence assumption to demonstrate the potential difference.

ANSWER:

a) Bayes Optimal Classifier and Naive Bayes

Bayes Optimal Classifier

The Bayes Optimal Classifier is the theoretically ideal classifier that makes predictions by selecting the class that maximizes the posterior probability. For a given input $\mathbf{x}$, it predicts the class $\hat{y}$ as:

$$\hat{y} = \arg\max_{y} P(y|\mathbf{x})$$

Using Bayes' theorem, this becomes:

$$\hat{y} = \arg\max_y \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

Since $P(\mathbf{x})$ is constant for all classes, this simplifies to:

$$\hat{y} = \arg\max_y P(\mathbf{x}|y)P(y)$$

The Bayes Optimal Classifier achieves the lowest possible error rate among all classifiers for a given problem, assuming we have complete knowledge of the underlying probability distributions.

Naive Bayes Classifier

The Naive Bayes classifier is a practical implementation based on Bayes' theorem, but it introduces a "naive" independence assumption to make calculations tractable. It assumes that features are conditionally independent given the class label.

Key Differences in Assumptions:

1. **Independence Assumption**: Naive Bayes assumes that all features are conditionally independent given the class label, while the Bayes Optimal Classifier makes no such assumption.
2. **Probability Calculation**:
   - Bayes Optimal: $P(\mathbf{x}|y) = P(x_1, x_2, \ldots, x_n|y)$ (true joint probability)
   - Naive Bayes: $P(\mathbf{x}|y) = P(x_1|y) \times P(x_2|y) \times \ldots \times P(x_n|y)$ (approximation based on independence)
3. **Practicality**: The Bayes Optimal Classifier requires complete knowledge of joint probability distributions, which is often impossible in practice, while Naive Bayes offers a computationally efficient approximation.

b) Independence Assumption Violation

Let's identify a case where the independence assumption might be violated in our dataset. Looking at the data, it appears that "Text Sentiment" and "Product Visibility" might be correlated conditional on "Satisfaction". For example, satisfied customers often have both positive sentiment and high visibility.

Calculating Joint Probability:

Let's examine P(Text Sentiment, Product Visibility | Satisfaction) for the case of (Negative, Low | Unsatisfied):

Without Independence Assumption (True Joint Probability):

Among the 4 unsatisfied customers, only 1 has both negative sentiment and low visibility (ID 5).

$$P(\text{Negative, Low}|\text{Unsatisfied}) = \frac{1}{4} = 0.25$$

With Independence Assumption (Naive Bayes Approach):

Among the 4 unsatisfied customers:

- 3 have negative sentiment: P(Negative | Unsatisfied) = 3/4 = 0.75
- 2 have low visibility: P(Low | Unsatisfied) = 2/4 = 0.5

$$P(\text{Negative}|\text{Unsatisfied}) \times P(\text{Low}|\text{Unsatisfied}) = 0.75 \times 0.5 = 0.375$$

Comparison:

- True joint probability: 0.25
- Naive Bayes approximation: 0.375

This difference (0.375 vs 0.25) demonstrates how the independence assumption can lead to inaccuracies in probability estimation. In this case, the Naive Bayes approach overestimates the probability by 50% because it doesn't account for the potential correlation between negative sentiment and low product visibility among unsatisfied customers.

In real e-commerce scenarios, this violation is intuitive: products with low visibility might be more likely to receive negative reviews when customers are unsatisfied, as the poor discoverability compounds their negative experience.

Question 8:

a) Design a genetic algorithm for feature selection in our dataset:

- Define a suitable chromosome representation
- Specify fitness function
- Describe selection, crossover, and mutation operators
- Define termination criteria

Explain how your design choices are appropriate for this specific dataset.

b) Trace through two generations of your genetic algorithm starting with an initial population of four randomly generated chromosomes. Show how selection, crossover, and mutation would work, and how the best solution evolves.

ANSWER:

a) Design of a Genetic Algorithm

Chromosome Representation

Since we have 3 features (Text Sentiment, Product Visibility, and Image Quality), we'll use a binary chromosome of length 3:

- Position 1: Text Sentiment (1 = include, 0 = exclude)
- Position 2: Product Visibility (1 = include, 0 = exclude)
- Position 3: Image Quality (1 = include, 0 = exclude)

For example, chromosome $[\mathbf{1}, \mathbf{0}, \mathbf{1}]$ means we use Text Sentiment and Image Quality but not Product Visibility.

Fitness Function

We'll define the fitness function as classification accuracy when using the selected feature subset:

$$\text{Fitness(chromosome)} = \frac{\text{Number of correctly classified instances}}{\text{Total number of instances}}$$

The fitness function will:

1. Select features based on the chromosome
2. Train a classifier (e.g., a Decision Tree) using only those features
3. Perform cross-validation on our small dataset
4. Return the accuracy as the fitness value

Selection Operator

We'll use Tournament Selection:

1. Randomly select 2 chromosomes from the population
2. The chromosome with higher fitness becomes a parent
3. Repeat to select another parent

Crossover Operator

Single-point crossover:

1. Choose a random position in the chromosome
2. Swap all bits after that position between two parents

3. Apply with probability p_crossover = 0.7

Mutation Operator

Bit-flip mutation:

1. For each bit in a chromosome, with probability p_mutation = 0.1
2. Flip the bit value (0→1 or 1→0)

Termination Criteria

The algorithm terminates when one of these conditions is met:

1. Maximum generations (20) is reached
2. Fitness hasn't improved for 5 consecutive generations
3. A perfect solution is found (fitness = 1.0)

Appropriateness for this Dataset

- **Small feature space**: With only 3 features, the search space is small (2^3=8 possibilities), making a genetic algorithm appropriate.
- **Binary representation**: Natural fit for feature selection.
- **Accuracy-based fitness**: Directly addresses our classification goal.
- **Tournament selection**: Works well with small populations and maintains diversity.
- **Cross-validation in fitness**: Important for our small dataset (8 entries).

b) Tracing Through Two Generations

Initial Population (Generation 0)

Let's start with 4 random chromosomes:

1. $[1, 0, 0]$ - Only Text Sentiment
2. $[0, 1, 1]$ - Product Visibility and Image Quality
3. $[1, 1, 0]$ - Text Sentiment and Product Visibility
4. $[0, 0, 1]$ - Only Image Quality

Computing Initial Fitness

Based on our dataset patterns:

- Chromosome 1 $[1, 0, 0]$: Fitness = 0.70 (Text Sentiment alone is moderately predictive)
- Chromosome 2 $[0, 1, 1]$: Fitness = 0.65 (Product Visibility and Image Quality together)

- Chromosome 3 $[1, 1, 0]$: Fitness = 0.80 (Text Sentiment and Product Visibility are strong predictors)
- Chromosome 4 $[0, 0, 1]$: Fitness = 0.50 (Image Quality alone is a weak predictor)

Generation 1

Selection

- Tournament 1: Chromosomes $[1, 0, 0]$ and $[1, 1, 0]$ compete $\rightarrow$ $[1, 1, 0]$ wins (fitness 0.80)
- Tournament 2: Chromosomes $[0, 1, 1]$ and $[0, 0, 1]$ compete $\rightarrow$ $[0, 1, 1]$ wins (fitness 0.65)

Crossover (crossover point after position 1)

- Parents: $[1, 1, 0]$ and $[0, 1, 1]$
- Offspring 1: $[1|1, 1]$ (first bit from parent 1, rest from parent 2)
- Offspring 2: $[0|1, 0]$ (first bit from parent 2, rest from parent 1)

Mutation

- Offspring 1 $[1, 1, 1]$: No mutation occurs
- Offspring 2 $[0, 1, 0]$: Third bit flips $\rightarrow$ $[0, 1, 1]$

Selection (second pair)

- Tournament 3: Chromosomes $[1, 0, 0]$ and $[0, 0, 1]$ compete $\rightarrow$ $[1, 0, 0]$ wins (fitness 0.70)
- Tournament 4: Chromosomes $[1, 1, 0]$ and $[0, 1, 1]$ compete $\rightarrow$ $[1, 1, 0]$ wins (fitness 0.80)

Crossover (crossover point after position 2)

- Parents: $[1, 0, 0]$ and $[1, 1, 0]$
- Offspring 3: $[1, 0|0]$ (unchanged as parents have same last bit)
- Offspring 4: $[1, 1|0]$ (unchanged as parents have same last bit)

Mutation

- Offspring 3 $[1, 0, 0]$: Second bit flips $\rightarrow$ $[1, 1, 0]$
- Offspring 4 $[1, 1, 0]$: No mutation occurs

Generation 1 Population

1. $[1, 1, 1]$ - All features (from crossover)
2. $[0, 1, 1]$ - Product Visibility and Image Quality (from mutation)
3. $[1, 1, 0]$ - Text Sentiment and Product Visibility (from mutation)

4. $[1, 1, 0]$ - Text Sentiment and Product Visibility (unchanged)

Fitness Computation

- Chromosome 1 $[1, 1, 1]$: Fitness = 0.85 (All features give best performance)
- Chromosome 2 $[0, 1, 1]$: Fitness = 0.65 (unchanged)
- Chromosome 3 $[1, 1, 0]$: Fitness = 0.80 (unchanged)
- Chromosome 4 $[1, 1, 0]$: Fitness = 0.80 (unchanged)

Generation 2

Selection

- Tournament 1: Chromosomes $[1, 1, 1]$ and $[0, 1, 1]$ compete $\rightarrow$ $[1, 1, 1]$ wins (fitness 0.85)
- Tournament 2: Chromosomes $[1, 1, 0]$ and $[1, 1, 1]$ compete $\rightarrow$ $[1, 1, 1]$ wins (fitness 0.85)

Crossover

- Parents: $[1, 1, 1]$ and $[1, 1, 1]$
- Offspring are identical to parents (no change from crossover)

Mutation

- Offspring 1 $[1, 1, 1]$: No mutation occurs
- Offspring 2 $[1, 1, 1]$: Second bit flips $\rightarrow$ $[1, 0, 1]$

Selection (second pair)

- Tournament 3: Chromosomes $[0, 1, 1]$ and $[1, 1, 0]$ compete $\rightarrow$ $[1, 1, 0]$ wins (fitness 0.80)
- Tournament 4: Chromosomes $[1, 1, 0]$ and $[1, 1, 1]$ compete $\rightarrow$ $[1, 1, 1]$ wins (fitness 0.85)

Crossover (crossover point after position 1)

- Parents: $[1, 1, 0]$ and $[1, 1, 1]$
- Offspring 3: $[1|1, 1]$ (first bit from parent 1, rest from parent 2)
- Offspring 4: $[1|1, 0]$ (first bit from parent 2, rest from parent 1)

Mutation

- Offspring 3 $[1, 1, 1]$: No mutation occurs
- Offspring 4 $[1, 1, 0]$: No mutation occurs

Generation 2 Population

1. $[1, 1, 1]$ - All features (unchanged)
2. $[1, 0, 1]$ - Text Sentiment and Image Quality (from mutation)
3. $[1, 1, 1]$ - All features (from crossover)
4. $[1, 1, 0]$ - Text Sentiment and Product Visibility (from crossover)

Fitness Computation

- Chromosome 1 $[1, 1, 1]$: Fitness = 0.85
- Chromosome 2 $[1, 0, 1]$: Fitness = 0.75
- Chromosome 3 $[1, 1, 1]$: Fitness = 0.85
- Chromosome 4 $[1, 1, 0]$: Fitness = 0.80

Evolution of Best Solution

- **Generation 0**: Best fitness = 0.80 ($[1, 1, 0]$ - Text Sentiment and Product Visibility)
- **Generation 1**: Best fitness = 0.85 ($[1, 1, 1]$ - All features)
- **Generation 2**: Best fitness = 0.85 ($[1, 1, 1]$ - All features)

The algorithm has converged on using all three features for optimal prediction of customer satisfaction, which makes intuitive sense for this e-commerce dataset where text sentiment, product visibility, and image quality all likely contribute to the overall customer experience.

Question 9:

a) The training dataset has equal numbers of "Satisfied" and "Unsatisfied" examples, but in reality, 80% of all customers are satisfied. Discuss the potential consequences of this sampling bias and propose two methods to address this issue.

b) A decision tree model perfectly classifies all training examples but performs poorly on new data. Explain how decision tree pruning could help with this overfitting problem. Based on the training dataset, suggest specific pruning that could be applied and explain its expected impact on model performance.

ANSWER:

a) Addressing Sampling Bias

Consequences of Sampling Bias

The training dataset has equal numbers of "Satisfied" and "Unsatisfied" examples (50% each), but in reality, 80% of customers are satisfied. This sampling bias can lead to several issues:

1. **Biased probability estimates**: The model will incorrectly assume the prior probability of each class is 50%, when in reality it's 80%-20%.
2. **Skewed decision boundaries**: Classification algorithms will shift decision boundaries to account for the perceived equal class distribution.
3. **Improper resource allocation**: The business might allocate excessive resources to address customer dissatisfaction based on inflated predictions of unsatisfied customers.
4. **Misleading performance metrics**: Accuracy on the balanced training set won't reflect real-world performance where class distribution is skewed.
5. **False positive/negative trade-offs**: The model will incorrectly balance false positives and false negatives based on the artificial 50-50 split.

Methods to Address this Issue

Method 1: Adjust Prior Probabilities

For probabilistic models like Naive Bayes, we can explicitly adjust the prior probabilities to match the real-world distribution:

- Set P(Satisfied) = 0.8
- Set P(Unsatisfied) = 0.2

This ensures that the model incorporates the correct base rates when making predictions. For example, in a Naive Bayes classifier, we would calculate:

$$P(Satisfied|Features) \propto P(Features|Satisfied) \times \mathbf{0.8}$$

$$P(Unsatisfied|Features) \propto P(Features|Unsatisfied) \times \mathbf{0.2}$$

Method 2: Cost-sensitive Learning

We can assign different misclassification costs to reflect the class imbalance:

- Misclassifying a satisfied customer as unsatisfied could have a cost proportional to 0.8
- Misclassifying an unsatisfied customer as satisfied could have a cost proportional to 0.2

During training, the algorithm will minimize the total cost rather than the error rate, effectively accounting for the class imbalance. This approach is particularly useful for decision trees, SVMs, and neural networks.

b) Decision Tree Pruning

Overfitting in Decision Trees

A decision tree that perfectly classifies all training examples but performs poorly on new data is showing classic signs of overfitting. The tree has essentially memorized the training data rather than learning general patterns.

Looking at our dataset, a perfectly fitting decision tree might create very specific rules like:

- If Text Sentiment = Positive → Satisfied (covers examples 1, 4, 7)
- If Text Sentiment = Negative → Unsatisfied (covers examples 2, 5, 8)
- If Text Sentiment = Neutral AND Product Visibility = High → Satisfied (covers example 6)
- If Text Sentiment = Neutral AND Product Visibility = Low → Unsatisfied (covers example 3)

How Pruning Helps

Pruning reduces the complexity of the decision tree by removing branches that provide little predictive power, thus improving generalization. It trades some training accuracy for better test performance.

Suggested Pruning Based on Training Dataset

Looking at our dataset patterns, I suggest the following pruning:

Prune 1: Simplify the Image Quality split
The tree likely creates splits based on Image Quality that don't generalize well:

- Good Image Quality has both satisfied and unsatisfied customers
- Poor Image Quality also has mixed outcomes

We could prune these branches and rely more on Text Sentiment, which shows clearer patterns:

- All Positive Text Sentiment examples are Satisfied
- All Negative Text Sentiment examples are Unsatisfied

Prune 2: Merge Product Visibility levels
Instead of treating Low, Medium, and High visibility as separate categories, we could simplify by merging Medium and High since they show similar patterns:

- Low visibility: 0/2 satisfied
- Medium/High visibility: 4/6 satisfied

Expected Impact on Model Performance

After applying these pruning strategies:

1. **Training accuracy** will decrease slightly from 100% as we've simplified the model.

2. **Test accuracy** will likely improve as the model now captures more general patterns rather than noise.
3. **Model interpretability** will improve with a simpler tree.
4. **Generalization** to new data will be better, especially for cases not well-represented in the training data.

The pruned tree might look like:

1. If Text Sentiment = Positive → Satisfied
2. If Text Sentiment = Negative → Unsatisfied
3. If Text Sentiment = Neutral:
   - If Product Visibility = Low → Unsatisfied
   - If Product Visibility = Medium/High → Satisfied

This simpler model is more likely to make correct predictions on new data like our test dataset, despite sacrificing perfect accuracy on the training set.

Question 10:

a) Explain the concept of k-fold cross-validation to evaluate model performance more reliably. Suggest a specific cross-validation strategy for this dataset and explain why it's appropriate.

b) If a model achieves 75% accuracy on the training data and 67% accuracy on a separate validation set, interpret these results. Is the model likely underfitting, overfitting, or appropriately fit? Explain why and suggest ways to improve the model.

ANSWER:

a) Explanation of k-fold cross-validation and strategy for this dataset
k-fold cross-validation involves splitting the dataset into *k* equally sized subsets (folds). The model is trained *k* times, each time using *k−1* folds for training and the remaining fold for validation. Performance metrics (e.g., accuracy) are averaged across all iterations to estimate model generalization. This reduces reliance on a single train-test split and provides a more robust performance estimate.

Suggested strategy for this dataset: Use **8-fold cross-validation (leave-one-out)**.

- **Reason**: With only 8 training entries, standard splits (e.g., 80-20) risk high variance due to limited data. Leave-one-out ensures each sample is used for validation once, maximizing training data usage and providing a reliable estimate despite the small size.

b) Interpretation of model performance and improvements

- **Interpretation**: The model is **overfitting**.
  - Training accuracy (75%) is higher than validation accuracy (67%), indicating the model learns noise/patterns specific to the training data, harming generalization.
  - The gap (~8%) suggests moderate overfitting.

Ways to improve:

1. **Regularization**: Add penalties (e.g., L1/L2) to reduce model complexity.
2. **Feature engineering**: Simplify/reduce features (e.g., combine "Text Sentiment" and "Image Quality").
3. **Data augmentation**: Collect more training data to reduce overfitting.
4. **Cross-validation**: Use the suggested 8-fold strategy to tune hyperparameters (e.g., decision tree depth).
5. **Simpler models**: Switch to a less complex algorithm (e.g., logistic regression instead of a deep neural network).

This approach balances bias-variance tradeoff and improves generalization.