Data Mining and Machine Learning: Exam Preparation

Tauan Torres Mendes

Contents

1 Metric Distance Functions

1.1 Question 1

- Which of the following is **NOT** a property of a *metric* distance function? **BOUNDEDNESS**

1.2 Answer

Boundedness is not a property of a metric distance function. The fundamental properties of a metric distance function are:

1. **Non-negativity:** $d(x, y) \geq 0 \quad \forall x, y$
2. **Identity of indiscernibles:** $d(x, y) = 0 \iff x = y$
3. **Symmetry:** $d(x, y) = d(y, x)$
4. **Triangle inequality:** $d(x, z) \leq d(x, y) + d(y, z)$

Boundedness is not required for a function to be a metric.

1.3 Theory Recap

A metric distance function must satisfy four properties:

- Non-negativity
- Identity of indiscernibles
- Symmetry
- Triangle inequality

Boundedness is not a requirement, as metrics are not constrained to finite values.

2 Similarity Measures

2.1 Question 2

- Given the two binary vectors below, which is their similarity according to the *Simple Matching Coefficient*? **0.5**

	a	b	c	d	e	f	g	h	i	j
Vector 1	1	0	0	0	1	0	1	1	0	1
Vector 2	1	0	1	1	1	0	1	0	1	0

2.2 Answer

The similarity according to the *Simple Matching Coefficient* (SMC) is:

$$\text{SMC} = \frac{\text{Number of matches}}{\text{Total elements}} = \frac{5}{10} = 0.5$$

2.3 Theory Recap

The **Simple Matching Coefficient (SMC)** is a measure used to calculate the similarity between two binary vectors. It considers both matches in 1s and 0s equally important. The formula is:

$$\text{SMC} = \frac{a+d}{a+b+c+d}$$

Where:

- a: Number of positions where both vectors have 1 (1,1 matches).
- d: Number of positions where both vectors have 0 (0,0 matches).
- b: Number of positions where the first vector has 1 and the second has 0 (1,0 mismatches).
- c: Number of positions where the first vector has 0 and the second has 1 (0,1 mismatches).

2.4 Key Concepts

- **Binary Vectors:** Represent objects in binary form (0s and 1s).
- **Matching Measures:** SMC equally considers matches of 1s and 0s, unlike other measures like the Jaccard index, which focuses on 1s.
- **Applications:** Used in comparing binary data sets, such as presence/absence matrices in machine learning.

2.5 Question 3

- Given the two binary vectors below, which is their similarity according to the *Jaccard Coefficient*? **0.375**

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*
Vector 1	1	0	0	0	1	0	1	1	0	1
Vector 2	1	0	1	1	1	0	1	0	1	0

2.6 Answer

The similarity according to the *Jaccard Coefficient* (JC) is:

$$\text{JC} = \frac{\text{Number of matches in 1's}}{\text{Total number of positions with at least one 1}} = \frac{3}{8} = 0.375$$

2.7 Theory Recap

The **Jaccard Coefficient (JC)** measures the similarity between two binary vectors by focusing only on the positions where at least one of the vectors has a 1. It is defined as:

$$\text{JC} = \frac{a}{a+b+c}$$

Where:

- a: Number of positions where both vectors have 1 (1,1 matches).
- b: Number of positions where the first vector has 1 and the second has 0 (1,0 mismatches).
- c: Number of positions where the first vector has 0 and the second has 1 (0,1 mismatches).

2.8 Key Differences

- JC focuses only on 1's and ignores 0,0 matches, making it more suitable for sparse binary data.
- Applications include text mining and clustering.

3 Clustering Methods

3.1 Question 4

- What is the *single linkage*? **A method to compute the distance between two sets of items; it can be used in hierarchical clustering.**

3.2 Answer

Single linkage is a method used to compute the distance between two clusters by taking the smallest distance between any two points in the two clusters. It is commonly used in hierarchical clustering.

3.3 Theory Recap

Single Linkage is a clustering technique used in hierarchical clustering to measure the similarity (or distance) between two clusters. The method considers the closest pair of points between the clusters. Formally:

$$d(A, B) = \min_{a \in A, b \in B} d(a, b)$$

- Applications include exploratory data analysis.

4 Evaluation Metrics

4.1 Question 5

- Given the definitions below:
  - TP = True Positives
  - TN = True Negatives
  - FP = False Positives
  - FN = False Negatives

  Which of the formulas below computes the *recall* (or *hit rate* or *sensitivity*) of a binary classifier? **TP / (TP + FN)**

4.2 Answer

The formula for recall (or sensitivity or hit rate) is:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Where:

- TP: The number of true positives (correctly predicted positive cases).
- FN: The number of false negatives (positive cases that were incorrectly predicted as negative).

4.3 Theory Recap

Recall, also known as sensitivity or hit rate, is a measure of a model's ability to identify all relevant instances in a dataset. It focuses on the proportion of true positive predictions relative to the total actual positives.

The formula is:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

4.3.1 Key Points:

- **High Recall:** Indicates that the model correctly identifies most of the positive cases.
- **Low Recall:** Means that many positive cases are missed (high FN).

4.3.2 Applications of Recall

- Recall is crucial in scenarios where missing positive cases has a high cost, such as:
  - Disease detection.
  - Fraud detection.

4.3.3 Relation to Other Metrics

Recall is often analyzed alongside precision to evaluate a model's overall performance. Together, they form the basis for other metrics such as the F1-Score.

4.4 Question 6

- Given the definitions below:
  - TP = True Positives
  - TN = True Negatives
  - FP = False Positives
  - FN = False Negatives

  Which of the formulas below computes the *accuracy* of a binary classifier? **(TP + TN) / (TP + FP + TN + FN)**

4.5 Answer

The formula for accuracy is:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

Where:

- TP: Number of true positives (correctly predicted positive cases).
- TN: Number of true negatives (correctly predicted negative cases).
- FP: Number of false positives (negative cases incorrectly predicted as positive).
- FN: Number of false negatives (positive cases incorrectly predicted as negative).

4.6 Theory Recap

Accuracy is a fundamental metric used to evaluate the performance of a classification model. It measures the proportion of correctly predicted cases (both positives and negatives) out of the total number of cases.

The formula is:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

4.6.1 Key Points:

- **High Accuracy:** Indicates that the model is correctly classifying most cases.
- **Limitations:** Accuracy can be misleading when dealing with imbalanced datasets, where one class significantly outweighs the other.

4.6.2 Applications of Accuracy

- Accuracy is a general-purpose metric, often used when the dataset is balanced, and all errors have equal importance.
- Less useful in cases like:
  - Fraud detection.
  - Rare disease diagnosis.

4.6.3 Relation to Other Metrics

Accuracy is often analyzed alongside precision, recall, and the F1-score to get a complete understanding of model performance, especially in imbalanced datasets.

4.7 Question 7

- Given the definitions below:
  - TP = True Positives
  - TN = True Negatives
  - FP = False Positives
  - FN = False Negatives

  Which of the formulas below computes the *precision* (or *positive predictive value*) of a binary classifier? **TP / (TP + FP)**

4.8 Answer

The formula for precision is:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Where:

- TP: Number of true positives (correctly predicted positive cases).
- FP: Number of false positives (negative cases incorrectly predicted as positive).

4.9 Theory Recap

Precision, also known as the positive predictive value, is a measure of the accuracy of positive predictions. It evaluates how many of the predicted positive cases are actually positive.

The formula is:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

4.9.1 Key Points:

- **High Precision:** Indicates that most of the predicted positives are correct.
- **Low Precision:** Means that there are many false positives.

4.9.2 Applications of Precision

- Precision is crucial in scenarios where false positives are costly, such as:
  - Email spam filters.
  - Predictive policing.

4.9.3 Relation to Other Metrics

Precision is often analyzed alongside recall. Together, they provide a balance through the F1-score, especially when there is a trade-off between false positives and false negatives.

4.10 Question 8

- Given the definitions below:
  - TP = True Positives
  - TN = True Negatives
  - FP = False Positives
  - FN = False Negatives

  Which of the formulas below computes the *specificity* of a binary classifier? **TN / (TN + FP)**

4.11 Answer

The formula for specificity is:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Where:

- TN: Number of true negatives (correctly predicted negative cases).
- FP: Number of false positives (negative cases incorrectly predicted as positive).

4.12 Theory Recap

Specificity, also known as the true negative rate, measures a model's ability to correctly identify negative cases. It evaluates the proportion of true negative predictions relative to all actual negative cases.

The formula is:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

4.12.1 Key Points:

- **High Specificity:** Indicates that the model correctly identifies most of the negative cases.
- **Low Specificity:** Means that many negative cases are incorrectly classified as positive (high FP).

4.12.2 Applications of Specificity

- Specificity is crucial in scenarios where avoiding false positives is important, such as:
  - Diagnostic tests where a false positive could lead to unnecessary treatment.
  - Fraud detection systems to reduce false alarms.

4.12.3 Relation to Other Metrics

Specificity is often analyzed alongside sensitivity (recall) to assess a model's overall performance, especially in imbalanced datasets. Together, they provide insight into how well the model distinguishes between the positive and negative classes.

5 Extra Theory Recap

5.1 Definitions of TP, FP, FN, TN

Let's consider a medical test for a disease:

- **Positive**: The person has the disease.
- **Negative**: The person does not have the disease.

5.1.1 1. True Positive (TP):

- **Definition**: Cases where the model correctly predicts "Positive."
- **Example**: The test predicts that a person has the disease, and the person actually has the disease.

5.1.2 2. False Positive (FP):

- **Definition**: Cases where the model predicts "Positive," but it is actually "Negative."
- **Example**: The test predicts that a person has the disease, but the person does not have it. This is also called a **Type I Error**.

5.1.3 3. False Negative (FN):

- **Definition**: Cases where the model predicts "Negative," but it is actually "Positive."
- **Example**: The test predicts that a person does not have the disease, but the person actually has it. This is also called a **Type II Error**.

5.1.4 4. True Negative (TN):

- **Definition**: Cases where the model correctly predicts "Negative."
- **Example**: The test predicts that a person does not have the disease, and the person actually does not have it.

5.2 Metrics Explanation

Let's now visualize each metric in a real-life scenario.

5.2.1 Example Data:

We test 10 people with a medical test:

- 5 people have the disease.
- 5 people do not have the disease.
- Model predictions:
  - Correctly predicts 3 people with the disease ($TP = 3$).

  - Predicts the disease for 2 people without the disease ($FP = 2$).
  - Misses 2 people with the disease ($FN = 2$).
  - Correctly predicts 3 people without the disease ($TN = 3$).

5.2.2 1. Accuracy

- **Formula**: $\text{Accuracy} = \frac{\text{TP}+\text{TN}}{N}$
- **What it means**: Fraction of correct predictions (both positive and negative) out of all predictions.
- **In the example**:

$$\text{Accuracy} = \frac{3+3}{10} = 0.6 \ (60\%)$$

- **Interpretation**: The model is correct 60% of the time.

5.2.3 2. Error Rate

- **Formula**: $\text{Error Rate} = 1 - \text{Accuracy}$
- **What it means**: Fraction of incorrect predictions out of all predictions.
- **In the example**:

$$\text{Error Rate} = 1 - 0.6 = 0.4 \ (40\%)$$

- **Interpretation**: The model is incorrect 40% of the time.

5.2.4 3. Precision

- **Formula**: $\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$
- **What it means**: Of all the cases the model predicted as positive, how many were actually positive.
- **In the example**:

$$\text{Precision} = \frac{3}{3+2} = 0.6 \ (60\%)$$

- **Interpretation**: When the model predicts a disease, it is correct 60% of the time.

5.2.5 4. Recall (Sensitivity)

- **Formula**: Recall $= \frac{\text{TP}}{\text{TP+FN}}$
- **What it means**: Of all the actual positive cases, how many the model correctly predicted as positive.
- **In the example**:

$$\text{Recall} = \frac{3}{3+2} = 0.6 \ (60\%)$$

- **Interpretation**: The model identifies 60% of the actual disease cases.

5.2.6 5. Specificity

- **Formula**: Specificity $= \frac{\text{TN}}{\text{TN+FP}}$
- **What it means**: Of all the actual negative cases, how many the model correctly predicted as negative.
- **In the example**:

$$\text{Specificity} = \frac{3}{3+2} = 0.6 \ (60\%)$$

- **Interpretation**: The model identifies 60% of the healthy cases correctly.

5.2.7 6. F1 Score

- **Formula**: $F1 = 2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision+Recall}}$
- **What it means**: Harmonic mean of precision and recall. It balances the trade-off between precision and recall.
- **In the example**:

$$F1 = 2 \cdot \frac{0.6 \cdot 0.6}{0.6 + 0.6} = 0.6 \ (60\%)$$

- **Interpretation**: The model has a balanced performance in predicting positive cases.

5.3 Key Takeaways

- **Precision**: How much you can trust the model when it predicts positive.
- **Recall**: How good the model is at finding all the positives.
- **Specificity**: How good the model is at finding all the negatives.
- **F1 Score**: Overall balance between precision and recall.

6 Clustering

6.1 Question 9

- Why do we *prune* a decision tree? **To eliminate parts of the tree where the decisions could be influenced by random effects.**

6.2 Answer

Pruning is used to remove sections of a decision tree that may be overfitting the training data. It helps generalize the model to perform better on unseen data.

6.3 Theory Recap

Pruning reduces the complexity of a decision tree by:

- Eliminating nodes that provide little predictive power.
- Reducing the risk of overfitting.

Pruning can be done using techniques like cost complexity pruning or setting a minimum node size.

—

6.4 Question 10

- A Decision Tree is... **A tree-structured plan of tests on single attributes to forecast the target.**

6.5 Answer

A decision tree is a predictive modeling technique that uses a tree-like structure to make decisions based on conditions or features of the data.

6.6 Theory Recap

Decision Trees:

- Split data into subsets based on the value of input features.
- Predict outcomes using a series of decision rules.
- Common algorithms: CART, ID3, and C4.5.

—

6.7 Question 11

- When training a neural network, what is the *learning rate*? **A multiplying factor of the correction to be applied to the connection weights.**

6.8 Answer

The learning rate controls how much the model's weights are adjusted during training.

6.9 Theory Recap

Learning Rate:

- Small learning rates ensure convergence but may be slow.
- Large learning rates speed up training but risk overshooting the optimal point.

The learning rate is often tuned using techniques like learning rate scheduling or adaptive optimizers (e.g., Adam, RMSprop).

—

6.10 Question 12

- Which of the following is a strength of the clustering algorithm DBSCAN?
  - **Ability to find clusters with concavities.**
  - **Ability to separate outliers from regular data.**

6.11 Answer

DBSCAN is robust in finding clusters of arbitrary shapes and identifying outliers.

6.12 Theory Recap

DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

- Identifies dense regions as clusters.
- Handles noise and outliers effectively.
- Parameters: ε (neighborhood radius) and *minPts* (minimum points in a cluster).

—

6.13 Question 13

- Which of the following is **not** a strength point of DBSCAN with respect to k-means? **The efficiency even in large datasets.**

6.14 Answer

DBSCAN is less efficient on large datasets because it computes distances between all points, resulting in higher computational cost compared to k-means.

6.15 Theory Recap

DBSCAN vs K-Means:

- **Strengths of DBSCAN**: Handles arbitrary-shaped clusters and outliers.

- **Weakness**: Higher computational complexity compared to k-means.
- **Strengths of K-Means**: Efficient on large datasets and easy to implement.

—

6.16 Question 14

- Which of the following characteristics of data can reduce the effectiveness of K-Means? **Presence of outliers.**

6.17 Answer

K-Means is sensitive to outliers because they can significantly distort the cluster centroids.

6.18 Theory Recap

Challenges of K-Means:

- Outliers can pull centroids away from actual cluster centers.
- Preprocessing techniques like removing outliers or using robust clustering algorithms (e.g., DBSCAN) can mitigate this issue.

—

6.19 Question 15

- After fitting DBSCAN with the default parameter values the result are: 0 clusters, 100% of noise points. Which will be your next trial?
  - **Reduce the minimum number of objects in the neighborhood.**
  - **Increase the radius of the neighborhood.**

6.20 Answer

Adjust the DBSCAN parameters (*minPts* and ε) to identify clusters better.

6.21 Theory Recap

Parameter Tuning in DBSCAN:

- *minPts*: Decrease this value to allow smaller clusters to form.
- ε: Increase this value to expand the neighborhood radius and include more points in clusters.

Parameter tuning is essential for DBSCAN to work effectively on different datasets.

7 Extra Theory Recap: Decision Tree

7.1 What is a Decision Tree?

A **Decision Tree** is a tool used in machine learning to make predictions or decisions based on a series of yes/no questions or splits. It's similar to playing "20 Questions," where each question narrows down the possibilities.

7.1.1 Structure of a Decision Tree

- **Root Node**: The starting point of the tree.
- **Internal Nodes**: Points where the data is split based on a condition.
- **Leaf Nodes**: The endpoints that represent the final decision or prediction.

7.2 Example of a Decision Tree

Imagine you want to decide whether to go outside based on the weather.

- **Root Node**: Is it raining?
  - Yes → Go to the next question.
  - No → Decision: Go outside.
- **Next Question (if raining)**: Do you have an umbrella?
  - Yes → Decision: Go outside.
  - No → Decision: Stay inside.

This simple tree gives us a decision based on two conditions: weather and whether you have an umbrella.

7.3 Building a Decision Tree

A decision tree is built using data. Let's say you have the following data about whether people go outside:

Weather	Umbrella	Went Outside?
Rain	Yes	Yes
Rain	No	No
Sunny	-	Yes
Cloudy	-	Yes

7.3.1 Step 1: Start with the Root Node

What's the best question to ask first? The goal is to ask a question that splits the data into groups where each group has similar answers (e.g., all "Yes" or all "No").

7.3.2 Step 2: Calculate a Splitting Criterion

Common metrics include **Gini Index** or **Entropy (Information Gain)**.

7.4 How Do Splits Work?

7.4.1 Gini Index (A Measure of Impurity):

Gini Index measures how "pure" a group is. A pure group means all data points have the same label.

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} p_i^2$$

Where p_i is the proportion of data points belonging to class i.

7.4.2 Example Calculation

For the dataset:

- **At the Root (Before Splitting):**

$$p_{\text{Yes}} = \frac{3}{4}, \quad p_{\text{No}} = \frac{1}{4}$$

$$\text{Gini Index} = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 1 - 0.5625 - 0.0625 = 0.375$$

- **After Splitting (e.g., by asking ”Is it raining?”)**:
  - Group 1 (Rain): 1 “Yes,” 1 “No”

$$\text{Gini} = 1 - \frac{1}{2}^2 - \frac{1}{2}^2 = 0.5$$

  - Group 2 (Not Rain): 2 “Yes,” 0 “No”

$$\text{Gini} = 1 - 1^2 - 0^2 = 0$$

7.5 What is Pruning?

A decision tree might grow too complex if we keep splitting until every data point fits perfectly. This can lead to **overfitting**, where the model memorizes the training data but performs poorly on new data.

7.5.1 Pruning:

Pruning simplifies the tree by removing splits (branches) that don’t add significant value. Think of pruning like trimming a plant: you cut away the unnecessary parts to make it healthier.

7.6 Types of Pruning

- **Pre-Pruning:** Stop growing the tree early, e.g., limit the maximum depth or minimum number of data points in a leaf.
- **Post-Pruning:** Grow the full tree, then remove branches that contribute little to the prediction.

7.7 Example of Pruning

Let’s revisit the weather example. Imagine you split based on “Is it cloudy?” but the results don’t help improve predictions (e.g., the Gini Index doesn’t improve). In this case, pruning would remove the “Cloudy” split, simplifying the tree.

7.7.1 Before Pruning:

```
Root: Is it raining?
  Yes → Umbrella?
    Yes → Go outside.
    No → Stay inside.
  No → Go outside.
```

7.7.2 After Pruning:

Root: Is it raining?
Yes → Stay inside.
No → Go outside.

7.8 Why is Pruning Important?

- Prevents overfitting.
- Improves interpretability (simpler trees are easier to understand).
- Reduces computational cost during predictions.

7.9 Key Takeaways

- Decision trees split data based on conditions.
- Over-splitting can lead to overfitting.
- Pruning simplifies the tree while retaining its predictive power.

8 Extra Theory Recap: Neural Networks

8.1 What is a Neural Network?

A **Neural Network** is a machine learning model inspired by the structure of the human brain. It is designed to learn patterns from data through layers of interconnected nodes, known as neurons.

8.1.1 Basic Structure of a Neural Network

- **Input Layer**: Receives the input data. Each neuron represents a feature of the data (e.g., age, height, weight).
- **Hidden Layers**: Perform computations on the input data using weights and biases. The network can have one or more hidden layers.
- **Output Layer**: Produces the final output or prediction (e.g., a class label or a numeric value).

8.2 How Neural Networks Work

8.2.1 Forward Propagation

- The input data passes through the network, layer by layer.
- Each neuron computes a weighted sum of its inputs:

  $$z = \sum_{i=1}^{n} w_i x_i + b$$

  Where:

  - w_i: Weight for input x_i.
  - b: Bias term.
- The result, z, is passed through an activation function (e.g., sigmoid, ReLU) to introduce non-linearity.

8.2.2 Activation Functions

Add non-linearity to the model, allowing it to learn complex patterns. Examples:

- Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$
- ReLU: $f(z) = \max(0, z)$

8.2.3 Loss Calculation

The difference between the predicted output and the true output is calculated using a loss function (e.g., Mean Squared Error for regression or Cross-Entropy for classification).

8.2.4 Backward Propagation

The model updates its weights and biases to minimize the loss. This is done using a process called **Gradient Descent**.

—

8.3 Learning Rate

The **Learning Rate** is a critical hyperparameter in neural networks. It controls the step size during the weight update process in gradient descent.

8.3.1 Gradient Descent

Gradient Descent minimizes the loss by updating weights in the opposite direction of the gradient (the slope) of the loss function:

$$w_{\text{new}} = w_{\text{old}} - \eta \frac{\partial L}{\partial w}$$

Where:

- w_{new}: Updated weight.
- w_{old}: Current weight.
- η: Learning rate.
- $\frac{\partial L}{\partial w}$: Gradient of the loss L with respect to the weight w.

—

8.3.2 Importance of the Learning Rate

- **Small Learning Rate**: Ensures convergence but can make training very slow. The model may get stuck in a local minimum.
- **Large Learning Rate**: Speeds up training but risks overshooting the optimal solution, leading to divergence.

8.4 Visualizing the Learning Rate

Imagine you're descending a hill:

- A small learning rate is like taking tiny steps—you're less likely to stumble but might take forever to reach the bottom.
- A large learning rate is like jumping—fast but risky.

—

8.5 Examples

8.5.1 Learning Rate Too High

The model might oscillate around the minimum without converging. Example:

- Initial Loss: 10
- Steps: 10, 15, 8, 12 (oscillating and unstable).

8.5.2 Learning Rate Too Low

The model converges very slowly. Example:

- Initial Loss: 10
- Steps: 10, 9.8, 9.6, 9.4 (progressing slowly).

8.5.3 Optimal Learning Rate

The model converges quickly and smoothly to the minimum. Example:

- Initial Loss: 10
- Steps: 10, 8, 5, 3, 0 (stable and efficient).

—

8.6 How to Choose a Learning Rate

- **Learning Rate Scheduling**: Gradually decrease the learning rate during training (e.g., exponential decay).
- **Adaptive Optimizers**: Use optimizers like Adam or RMSprop, which adjust the learning rate dynamically based on gradients.
- **Grid Search**: Test multiple learning rates to find the best one.

—

8.7 Example with Numbers

Assume a single neuron with:

- Weight $w = 0.5$
- Input $x = 2$
- Target output $y = 1$

8.7.1 Forward Pass

$$z = wx = 0.5 \times 2 = 1$$

Prediction: $\hat{y} = 1$

8.7.2 Loss (Mean Squared Error)

$$L = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(1 - 1)^2 = 0$$

8.7.3 Backpropagation

Gradient:

$$\frac{\partial L}{\partial w} = x(y - \hat{y}) = 2(1 - 1) = 0$$

8.7.4 Weight Update

With $\eta = 0.1$:

$$w_{\text{new}} = w_{\text{old}} - \eta \frac{\partial L}{\partial w} = 0.5 - 0.1 \times 0 = 0.5$$

The weight doesn't change because the prediction was already correct! —

8.8 Key Takeaways

- Neural networks learn by adjusting weights using forward and backward passes.
- The learning rate determines how fast weights are updated.
- An optimal learning rate is crucial for stable and efficient training.

9 Extra Theory Recap: DBSCAN

9.1 What is DBSCAN?

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups data points based on their density in the feature space. Unlike other algorithms (like K-Means), DBSCAN does not assume that clusters have a particular shape and can handle outliers effectively.

9.2 Core Concepts of DBSCAN

DBSCAN identifies clusters based on the density of points. It uses two main parameters:

- ε (**epsilon**): The radius of the neighborhood around a point.
- **minPts**: The minimum number of points required within the ε-neighborhood for a point to be considered a core point.

9.2.1 Types of Points in DBSCAN

- **Core Point**: A point that has at least **minPts** points (including itself) within its ε-neighborhood.
- **Border Point**: A point that does not have enough points within its ε-neighborhood to be a core point but is within the ε-neighborhood of a core point.
- **Noise Point** (Outlier): A point that is neither a core point nor a border point. It is considered an outlier.

9.3 Steps of the DBSCAN Algorithm

1. **Start with an Unvisited Point**:
   - Pick an unvisited point in the dataset.
2. **Check the Neighborhood**:
   - Count the number of points within the ε-neighborhood of the point.
3. **Classify the Point**:
   - If the point has at least **minPts** points in its neighborhood, it becomes a **core point**, and a new cluster is started.
   - If not, it is labeled as a **noise point** (temporarily).
4. **Expand the Cluster**:
   - For a core point, add all points in its ε-neighborhood to the cluster.
   - Recursively repeat this process for all new core points discovered in the neighborhood.

5. **Continue Until All Points Are Visited**:
   - Once all points have been visited, the clusters and noise points are finalized.

9.4 Example of DBSCAN

9.4.1 Dataset

Let's say we have the following points in 2D space:

$$(1, 1), (1, 2), (2, 2), (8, 8), (8, 9), (9, 9), (20, 20)$$

9.4.2 Parameters

- $\varepsilon = 2$ (radius of neighborhood)
- minPts = 3 (minimum points to form a cluster)

9.4.3 Step-by-Step Process

1. **Start with point** $(1, 1)$:
   - Check its ε-neighborhood: $(1, 2), (2, 2)$.
   - Total points in the neighborhood = 3 (including itself).
   - $(1, 1)$ is a core point. Start a new cluster: Cluster 1.
2. **Expand Cluster 1**:
   - Add $(1, 2)$ and $(2, 2)$ to Cluster 1.
   - Check if $(1, 2)$ or $(2, 2)$ are core points:
     - Both points have 3 points in their neighborhoods $\rightarrow$ They are core points.
   - All neighbors of $(1, 2)$ and $(2, 2)$ are already in the cluster.
3. **Move to point** $(8, 8)$:
   - Check its ε-neighborhood: $(8, 9), (9, 9)$.
   - Total points in the neighborhood = 3.
   - $(8, 8)$ is a core point. Start a new cluster: Cluster 2.
4. **Expand Cluster 2**:

- Add $(8,9)$ and $(9,9)$ to Cluster 2.
- Both points are core points since their neighborhoods satisfy the density requirement.

5. **Move to point** $(20,20)$:

   - Check its ε-neighborhood: No points within $\varepsilon = 2$.
   - $(20,20)$ is a noise point (outlier).

9.4.4 Final Result

- **Cluster 1**: $(1,1),(1,2),(2,2)$
- **Cluster 2**: $(8,8),(8,9),(9,9)$
- **Noise**: $(20,20)$

9.5 Advantages of DBSCAN

- **Handles Arbitrary Shapes**: DBSCAN can identify clusters of any shape (e.g., circular, elongated).
- **Identifies Outliers**: Points not belonging to any cluster are labeled as noise.
- **No Need to Specify the Number of Clusters**: Unlike K-Means, DBSCAN doesn't require the user to predefine the number of clusters.

9.6 Limitations of DBSCAN

- **Parameter Sensitivity**: The results depend heavily on the choice of ε and minPts.
- **Scalability**: DBSCAN is computationally expensive for large datasets because it calculates the distance between all points.
- **Varying Densities**: Struggles to detect clusters when densities vary significantly.

9.7 Applications of DBSCAN

- **Geospatial Clustering**: Grouping GPS coordinates into meaningful clusters.
- **Outlier Detection**: Identifying fraud or anomalies in datasets.
- **Image Segmentation**: Clustering pixels in an image based on their density.

10 Extra Theory Recap: K-Means

10.1 What is K-Means?

K-Means is a clustering algorithm that partitions a dataset into k clusters, where each cluster is defined by its centroid (a central point representing the cluster).

10.2 Core Concepts of K-Means

- **Centroid**: The "center" of a cluster, calculated as the mean of all points in the cluster.
- **Cluster**: A group of data points that are closer to one centroid than to others.
- **Objective**: Minimize the within-cluster sum of squares (WCSS), also called inertia:

  $$WCSS = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \mu_k\|^2$$

  Where:

  - K: Number of clusters.
  - C_k: Points in cluster k.
  - μ_k: Centroid of cluster k.

10.3 Steps of the K-Means Algorithm

1. **Initialize Centroids**: Randomly select k points from the dataset as initial centroids.
2. **Assign Points to Clusters**:

   - For each point in the dataset, assign it to the cluster whose centroid is closest (based on distance, typically Euclidean).

3. **Update Centroids**:

   - For each cluster, calculate the mean of all points assigned to it. The new mean becomes the updated centroid.

4. **Repeat**:

   - Repeat steps 2 and 3 until:
     - The centroids no longer change.
     - Or the maximum number of iterations is reached.

10.4 Example of K-Means

10.4.1 Dataset

We have the following 2D points:

$$(1,2),(2,1),(4,5),(6,8),(7,9)$$

10.4.2 Parameters

- $k = 2$ (we want 2 clusters).
- Randomly pick two points as initial centroids:

$$\text{Centroid 1: } (1,2), \quad \text{Centroid 2: } (7,9)$$

10.4.3 Step-by-Step Process

1. **Step 1: Assign Points to Clusters**

   - Calculate the distance from each point to the centroids (using Euclidean distance):

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

   Example for point $(4,5)$:

$$\text{Distance to Centroid 1} = \sqrt{(4-1)^2 + (5-2)^2} = \sqrt{3^2 + 3^2} = \sqrt{18} \approx 4.24$$

$$\text{Distance to Centroid 2} = \sqrt{(4-7)^2 + (5-9)^2} = \sqrt{3^2 + 4^2} = \sqrt{25} = 5$$

Based on distances:

$$\text{Cluster 1} : (1, 2), (2, 1), (4, 5)$$

$$\text{Cluster 2} : (6, 8), (7, 9)$$

2. **Step 2: Update Centroids**

   - Calculate the new centroids by taking the mean of the points in each cluster:

   $$\text{New Centroid 1: } \left(\frac{1+2+4}{3}, \frac{2+1+5}{3}\right) = (2.33, 2.67)$$

   $$\text{New Centroid 2: } \left(\frac{6+7}{2}, \frac{8+9}{2}\right) = (6.5, 8.5)$$

3. **Step 3: Repeat**

   - Reassign points to the updated centroids and update centroids again.
   - Continue until the centroids do not change significantly.

10.4.4 Final Result

- **Cluster 1**: Points closer to $(2.33, 2.67)$.
- **Cluster 2**: Points closer to $(6.5, 8.5)$.

10.5 Advantages of K-Means

- **Simplicity**: Easy to implement and understand.
- **Efficiency**: Works well with large datasets.
- **Scalability**: Can be extended to higher dimensions.

10.6 Limitations of K-Means

- **Number of Clusters (k)**: The user must specify k in advance, which may not always be known.
- **Sensitivity to Initialization**: Poor initialization can lead to suboptimal clustering. This can be mitigated using techniques like K-Means++.

- **Assumes Spherical Clusters**: Works best when clusters are roughly spherical and evenly distributed.
- **Outliers**: Sensitive to outliers, which can distort cluster centroids.

10.7 Applications of K-Means

- **Image Compression**: Cluster pixels into groups to reduce colors in an image.
- **Customer Segmentation**: Group customers based on purchasing behavior.
- **Document Clustering**: Group similar documents based on their content.

10.8 Visualizing K-Means

Imagine the data points are magnets, and the centroids are metal balls. Each metal ball is pulled toward the magnets it attracts (i.e., the data points assigned to it). Over time, the balls (centroids) settle in the center of their respective groups.

11 Association Rules

11.1 Question 16

- Which of the following statements regarding the discovery of association rules is true (one or more)?
  - **The confidence of a rule can be computed starting from the supports of itemsets.**
  - **The support of an itemset is anti-monotonic with respect to the composition of the itemset.**

11.2 Answer

Both statements are correct:

1. The confidence of a rule can be computed from the supports of itemsets:

$$\text{Confidence} = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

2. The support of an itemset is anti-monotonic, meaning:

   If an itemset is frequent, all its subsets must also be frequent.

   Conversely, if an itemset is infrequent, all its supersets are infrequent.

11.3 Theory Recap

Association Rule Mining is a method in data mining to discover relationships between variables in a dataset. It is often used in market basket analysis to find associations between items purchased together.

11.3.1 Key Metrics in Association Rule Mining

- **Support**: The proportion of transactions containing a specific itemset:

$$\text{Support}(X) = \frac{\text{Number of transactions containing } X}{\text{Total number of transactions}}$$

- **Confidence**: The likelihood that item Y is purchased given that item X is purchased:

$$\text{Confidence}(X \to Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

- **Lift**: The strength of an association rule compared to random co-occurrence:

$$\text{Lift}(X \to Y) = \frac{\text{Confidence}(X \to Y)}{\text{Support}(Y)}$$

11.3.2 Anti-Monotonicity of Support

The support of an itemset exhibits the **anti-monotonic property**:

- If an itemset X is infrequent, any superset of X (e.g., $X \cup \{a\}$) will also be infrequent.
- This property is crucial for pruning in algorithms like **Apriori**, which eliminates candidates early based on their subsets.

11.3.3 Example

Consider the following transactions:

Transaction ID	Items
1	$\{A, B, C\}$
2	$\{A, C\}$
3	$\{A, B\}$
4	$\{B, C\}$
5	$\{A, B, C\}$

- **Support Calculation**:
  - Support($\{A\}$) $= \frac{4}{5} = 0.8$
  - Support($\{A, B\}$) $= \frac{3}{5} = 0.6$
- **Confidence Calculation**:
  - Rule: $\{A\} \rightarrow \{B\}$

$$\text{Confidence} = \frac{\text{Support}(\{A, B\})}{\text{Support}(\{A\})} = \frac{0.6}{0.8} = 0.75$$

- **Anti-Monotonicity of Support**:
  - If $\{A, B, C\}$ is infrequent, then any superset (e.g., $\{A, B, C, D\}$) will also be infrequent.

11.4 Key Takeaways

- **Confidence and support** are fundamental metrics in association rule mining.
- **Anti-monotonicity of support** is used to optimize algorithms like Apriori by reducing the search space.
- Association rules are widely applied in domains like market basket analysis, recommendation systems, and customer behavior modeling.

11.5 Question 17

- Consider the transactional dataset below:

ID	Items
1	A, B, C
2	A, B, D
3	B, D, E
4	C, D
5	A, C, D, E

Which is the **confidence** of the rule $A, C \Rightarrow B$? **50%**

11.6 Answer

The **confidence** of a rule is calculated as:

$$\text{Confidence}(X \Rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

For the rule $A, C \Rightarrow B$:

- $X = \{A, C\}$, $Y = \{B\}$.
- $X \cup Y = \{A, C, B\}$.

Step 1: Calculate Support for $\{A, C\}$:

- Transactions containing $\{A, C\}$: 1, 5.
- $\text{Support}(\{A, C\}) = \frac{2}{5} = 0.4$ (40%).

Step 2: Calculate Support for $\{A, C, B\}$:

- Transactions containing $\{A, C, B\}$: 1.
- $\text{Support}(\{A, C, B\}) = \frac{1}{5} = 0.2$ (20%).

Step 3: Calculate Confidence:

$$\text{Confidence}(A, C \Rightarrow B) = \frac{\text{Support}(\{A, C, B\})}{\text{Support}(\{A, C\})} = \frac{0.2}{0.4} = 0.5 \ (50\%)$$

11.7 Theory Recap

- **Confidence** measures the likelihood that Y occurs given that X has occurred.
- It is a key metric in association rule mining to evaluate the strength of a rule.

11.7.1 Steps to Calculate Confidence

1. Identify X (antecedent) and Y (consequent).
2. Compute the support of $X \cup Y$ (both antecedent and consequent together).
3. Compute the support of X (antecedent only).
4. Divide the support of $X \cup Y$ by the support of X:

$$\text{Confidence}(X \Rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

11.8 Question 18

- Consider the transactional dataset below:

ID	Items
1	A, B, C
2	A, B, D
3	B, D, E
4	C, D
5	A, C, D, E

Which is the **support** of the rule $A, C \Rightarrow B$? **20%**

11.9 Answer

The **support** of a rule $X \Rightarrow Y$ is calculated as:

$$\text{Support}(X \Rightarrow Y) = \text{Support}(X \cup Y)$$

For the rule $A, C \Rightarrow B$:

- $X = \{A, C\}$, $Y = \{B\}$.
- $X \cup Y = \{A, C, B\}$.

Step 1: Identify Transactions Containing $\{A, C, B\}$:

- Transactions containing $\{A, C, B\}$: 1.

Step 2: Calculate Support:

$$\text{Support}(\{A, C, B\}) = \frac{\text{Number of transactions containing } \{A, C, B\}}{\text{Total number of transactions}}$$

$$\text{Support}(\{A, C, B\}) = \frac{1}{5} = 0.2 \ (20\%)$$

11.10 Theory Recap

- **Support** measures the proportion of transactions in the dataset that contain a specific itemset.
- It is used to evaluate how frequently an itemset or rule occurs in the dataset.
- For a rule $X \Rightarrow Y$, the support is equivalent to the support of $X \cup Y$, as both antecedent and consequent must occur together.

11.10.1 Steps to Calculate Support

1. Identify the itemset $X \cup Y$ (antecedent and consequent together).
2. Count the number of transactions containing $X \cup Y$.
3. Divide this count by the total number of transactions:

$$\text{Support}(X \cup Y) = \frac{\text{Number of transactions containing } X \cup Y}{\text{Total number of transactions}}$$

11.11 Key Takeaway

The support of a rule provides an absolute measure of how frequently the rule is observed in the dataset, making it a fundamental metric in association rule mining.

11.12 Question 19

- Consider the transactional dataset below:

ID	Items
1	A, B, C
2	A, B, D
3	B, D, E
4	C, D
5	A, C, D, E

Which is the **confidence** of the rule $B \Rightarrow E$? **33%**

11.13 Answer

The **confidence** of a rule is calculated as:

$$\text{Confidence}(X \Rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

For the rule $B \Rightarrow E$:

- $X = \{B\}$, $Y = \{E\}$.
- $X \cup Y = \{B, E\}$.

Step 1: Calculate Support for $\{B\}$:

- Transactions containing $\{B\}$: 1, 2, 3.
- $\text{Support}(\{B\}) = \frac{3}{5} = 0.6$ (60%).

Step 2: Calculate Support for $\{B, E\}$:

- Transactions containing $\{B, E\}$: 3.
- $\text{Support}(\{B, E\}) = \frac{1}{5} = 0.2$ (20%).

Step 3: Calculate Confidence:

$$\text{Confidence}(B \Rightarrow E) = \frac{\text{Support}(\{B, E\})}{\text{Support}(\{B\})} = \frac{0.2}{0.6} = 0.33\ (33\%)$$

11.14 Theory Recap

- **Confidence** measures the likelihood that Y occurs given that X has occurred.
- It is a ratio of the number of transactions containing $X \cup Y$ to the number of transactions containing X.

11.14.1 Steps to Calculate Confidence

1. Identify X (antecedent) and Y (consequent).
2. Compute the support of $X \cup Y$ (both antecedent and consequent together).
3. Compute the support of X (antecedent only).
4. Divide the support of $X \cup Y$ by the support of X:

$$\text{Confidence}(X \Rightarrow Y) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

11.15 Key Takeaway

The confidence of a rule provides a conditional measure of the strength of the rule, indicating how often Y occurs when X is present.

12 Topic: Polynomial Regression

12.1 Question 20

- When is polynomial regression appropriate? **When the relationship between the predicting variable and the target cannot be approximated as linear.**

12.2 Answer

Polynomial regression is appropriate when the data shows a non-linear relationship between the predictor variable(s) and the target variable. A linear model would not capture the patterns or trends in such cases, and a polynomial model is better suited.

—

12.3 Theory Recap

Polynomial Regression is a type of regression that models the relationship between the predictor variable X and the response variable Y as an n-th degree polynomial.

12.3.1 General Form of Polynomial Regression

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_n X^n + \epsilon$$

Where:

- $\beta_0, \beta_1, \ldots, \beta_n$: Coefficients of the polynomial.
- X^n: Predictor variable raised to the n-th power.
- ϵ: Error term.

—

12.4 When to Use Polynomial Regression

- When scatter plots show a clear curved or non-linear trend.
- When a simple linear regression model yields high residuals and low goodness-of-fit (e.g., low R^2).

—

12.5 Example

Let's consider a dataset where:

$$X = [1, 2, 3, 4, 5], \quad Y = [2, 5, 10, 17, 26]$$

12.5.1 Linear Regression

Using a linear model:

$$Y = \beta_0 + \beta_1 X$$

This would result in high residuals because the relationship between X and Y is quadratic, not linear.

12.5.2 Polynomial Regression (Degree 2)

Using a polynomial model:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2$$

This captures the quadratic trend in the data:

$$Y = 1 + 1X + 1X^2$$

$$\text{Predicted Values: } [2, 5, 10, 17, 26]$$

—

12.6 Advantages of Polynomial Regression

- Can model non-linear relationships effectively.
- Still easy to implement using tools like Scikit-learn.

12.7 Disadvantages of Polynomial Regression

- **Overfitting**: High-degree polynomials can fit noise in the data.
- **Extrapolation Issues**: Predictions beyond the range of data can behave erratically.

12.8 Key Takeaways

- Use polynomial regression when data shows a clear non-linear trend.
- Regularize or limit the polynomial degree to avoid overfitting.
- Always visualize the data to determine the best degree for the model.

13 Topic: Data Discretization

13.1 Question 21

- Which is the purpose of discretization/discretisation? **Reduce the number of distinct values in an attribute, in order to put in evidence possible patterns and regularities.**

13.2 Answer

The purpose of discretization is to transform continuous data into discrete bins or intervals. This reduces the number of distinct values in an attribute, highlighting potential patterns and regularities that might not be evident in raw continuous data.

—

13.3 Theory Recap

Discretization is a pre-processing technique in data mining and machine learning. It converts continuous attributes into discrete categories or intervals.

13.3.1 Why Use Discretization?

- **Simplify Data Analysis**: By reducing the number of distinct values, it simplifies patterns and trends in the data.

- **Improve Interpretability**: Discrete categories are often easier to understand and interpret.
- **Enhance Algorithm Compatibility**: Some machine learning algorithms, such as decision trees, work better with discrete features.

—

13.3.2 Methods of Discretization

1. **Equal Width Binning**:
   - Divides the range of values into intervals of equal width.
   - Example: For a range of $[1, 100]$ with 5 bins:

   $$\text{Bins: } [1, 20], [21, 40], [41, 60], [61, 80], [81, 100]$$

2. **Equal Frequency Binning**:
   - Divides the data such that each bin has approximately the same number of data points.
   - Example: For the values $[1, 2, 3, 4, 5, 6, 7, 8, 9]$ with 3 bins:

   $$\text{Bins: } [1, 3], [4, 6], [7, 9]$$

3. **Clustering-Based Discretization**:
   - Uses clustering techniques (e.g., K-Means) to group continuous values into discrete clusters.

4. **Entropy-Based Discretization**:
   - Uses information gain (from decision trees) to determine the optimal splitting points for intervals.

—

13.4 Example

Let's consider the following dataset of continuous ages:

$$\text{Ages: } [12, 15, 20, 25, 30, 35, 40, 50, 60]$$

13.4.1 Equal Width Binning (3 Bins)

$$\text{Bins: } [12, 28], [29, 45], [46, 60]$$

13.4.2 Equal Frequency Binning (3 Bins)

$$\text{Bins: } [12, 20], [25, 35], [40, 60]$$

—

13.5 Advantages of Discretization

- Simplifies complex datasets.
- Reduces computational complexity for some algorithms.
- Helps identify patterns that might be obscured in continuous data.

13.6 Disadvantages of Discretization

- Loss of information due to binning.
- Choice of binning strategy can introduce bias.

13.7 Key Takeaways

- Discretization transforms continuous attributes into discrete ones to simplify analysis.
- Proper selection of binning strategy is crucial to preserve meaningful information in the data.

14 Topic: CRISP-DM Methodology

14.1 Question 22

- In which part of the CRISP-DM methodology do we perform the *test design* activity? **Modelling**

14.2 Answer

The *test design* activity is performed during the **Modelling** phase of the CRISP-DM methodology. This phase focuses on selecting and applying appropriate modeling techniques to the dataset and testing their effectiveness. —

14.3 Theory Recap

CRISP-DM (Cross Industry Standard Process for Data Mining) is a widely used methodology in data mining and machine learning projects. It provides a structured approach divided into six phases.

14.3.1 Phases of CRISP-DM

1. **Business Understanding**:
   - Define the objectives and requirements of the project.
   - Translate business goals into data mining goals.
2. **Data Understanding**:
   - Collect initial data.
   - Explore the data to identify patterns, anomalies, or quality issues.
3. **Data Preparation**:
   - Clean and preprocess the data for analysis.
   - Create the final dataset to be used in the modeling phase.
4. **Modelling**:
   - Select appropriate modeling techniques.
   - Build and test models.
   - Perform **test design** to validate the performance of models.
5. **Evaluation**:
   - Assess the models in the context of the business objectives.
   - Determine whether the models meet the project goals.
6. **Deployment**:

  - Implement the results of the project in a production environment.
  - Deliver the final report or system to stakeholders.

—

14.3.2 Focus on the Modelling Phase

The Modelling phase is critical as it involves:

- Selecting the right algorithms for the task (e.g., regression, clustering, classification).
- Designing tests to validate the model's performance on unseen data.
- Iterating between model building and evaluation to optimize performance.

—

14.4 Example of Test Design

Suppose we are building a classification model to predict whether a customer will churn:

- During test design, we split the dataset into:
  - Training Set (70%): Used to train the model.
  - Test Set (30%): Used to evaluate the model's performance.
- Metrics such as accuracy, precision, recall, and F1-score are calculated on the test set to determine the effectiveness of the model.

—

14.5 Key Takeaways

- The *test design* activity is essential for ensuring the model's validity and reliability.
- It is a part of the **Modelling** phase in the CRISP-DM methodology.
- CRISP-DM provides a structured and iterative approach, ensuring that data mining projects align with business objectives.

15 Topic: Standardization of Attributes

15.1 Question 23

- Which is the main reason for the *standardization* of numeric attributes? **Map all the numeric attributes to a new range such that the mean is zero and the variance is one.**

15.2 Answer

The main reason for standardizing numeric attributes is to transform them into a standardized scale where:

- The **mean** of the attributes becomes 0.
- The **variance** of the attributes becomes 1.

This ensures that all attributes contribute equally to the model, especially when dealing with algorithms sensitive to the scale of features.

—

15.3 Theory Recap

Standardization (also called *Z-score normalization*) is a data preprocessing technique that transforms numeric attributes into a standard scale.

15.3.1 Formula for Standardization

For each value x_i in an attribute:

$$z_i = \frac{x_i - \mu}{\sigma}$$

Where:

- z_i: Standardized value.
- x_i: Original value.
- μ: Mean of the attribute.
- σ: Standard deviation of the attribute.

—

15.4 Why Standardize Data?

- **Equal Contribution to Model**: Standardization ensures that attributes with larger scales do not dominate those with smaller scales.
- **Improves Convergence in Gradient Descent**: Models like logistic regression and neural networks converge faster when data is standardized.
- **Required by Algorithms**: Algorithms like Support Vector Machines (SVMs) and Principal Component Analysis (PCA) require features to be on the same scale for optimal performance.

—

15.5 Example

Original Data:

$$X = [10, 15, 20, 25, 30]$$

Step 1: Calculate Mean and Standard Deviation:

$$\mu = \frac{10 + 15 + 20 + 25 + 30}{5} = 20$$

$$\sigma = \sqrt{\frac{(10-20)^2 + (15-20)^2 + (20-20)^2 + (25-20)^2 + (30-20)^2}{5}} = 7.07$$

Step 2: Apply Standardization Formula:

$$z_1 = \frac{10-20}{7.07} = -1.41, \quad z_2 = \frac{15-20}{7.07} = -0.71, \quad z_3 = \frac{20-20}{7.07} = 0$$

$$z_4 = \frac{25-20}{7.07} = 0.71, \quad z_5 = \frac{30-20}{7.07} = 1.41$$

Standardized Data:

$$Z = [-1.41, -0.71, 0, 0.71, 1.41]$$

—

15.6 Advantages of Standardization

- Improves the performance of distance-based algorithms (e.g., K-Means, KNN).
- Reduces bias in models caused by scale differences.

15.7 Disadvantages of Standardization

- Can distort the meaning of the original data for interpretability.
- Sensitive to outliers, which can significantly affect μ and σ.

—

15.8 Key Takeaways

- Standardization transforms numeric attributes into a scale with a mean of 0 and variance of 1.
- It is crucial for algorithms sensitive to feature scales.
- Always check if standardization is necessary based on the model and dataset characteristics.

16 Topic: Gini Index

16.1 Question 24

- What is *Gini Index*? **An impurity measure of a dataset alternative to the *Information Gain* and to the *Misclassification Index.***

16.2 Answer

The **Gini Index** is a measure of impurity or heterogeneity of a dataset. It is commonly used in decision tree algorithms to evaluate splits at each node. It serves as an alternative to the *Information Gain* and the *Misclassification Index.*

—

16.3 Theory Recap

The Gini Index evaluates how often a randomly chosen element from the dataset would be incorrectly classified if it were labeled based on the distribution of classes in the dataset.

16.3.1 Formula for Gini Index

$$Gini = 1 - \sum_{i=1}^{n} p_i^2$$

Where:

- p_i: Proportion of instances belonging to class i.
- n: Number of unique classes.

16.3.2 Example

Consider a dataset with two classes:

$$\text{Class Distribution: } [60\%, 40\%]$$

$$Gini = 1 - (0.6^2 + 0.4^2) = 1 - (0.36 + 0.16) = 1 - 0.52 = 0.48$$

A lower Gini Index indicates better purity of the node.
—

16.4 Comparison with Information Gain

- **Information Gain**: Based on entropy, it measures the reduction in uncertainty after a split.
- **Gini Index**: Focuses on class distributions and is computationally less intensive.

16.5 Key Takeaways

- The Gini Index is widely used in decision trees like CART (Classification and Regression Trees).
- It is an effective and computationally efficient impurity measure.

16.6 Question 25

- Which of the following measures can be used as an alternative to the *Information Gain*? **Gini Index**

16.7 Answer

The **Gini Index** can be used as an alternative to the *Information Gain*. It is particularly favored in decision tree algorithms like CART due to its computational simplicity and effective impurity measurement.

—

16.8 Theory Recap

- **Information Gain**: Based on entropy, measures the reduction in uncertainty.
- **Gini Index**: Measures impurity using class probabilities and is less computationally expensive.
- Both measures are effective, but the choice often depends on the algorithm or the computational requirements.

—

16.9 Applications of Gini Index

- Used in decision trees for classification tasks.
- Evaluates splits at each node to determine the best split.

17 Topic: Decision Trees

17.1 Question 26

- In a decision tree, the number of objects in a node... **is smaller than the number of objects in its ancestor.**

17.2 Answer

In a decision tree, at each split, the dataset is divided into smaller subsets. As a result:

- The number of objects in a child node is always **smaller than or equal to** the number of objects in its parent (ancestor) node.
- This reduction occurs because some data points are excluded from each branch based on the splitting criterion.

—

17.3 Theory Recap

Decision Trees are hierarchical models used for classification and regression tasks. They split the dataset at each node based on the value of an attribute to create branches, leading to subsets of the data.

17.3.1 How Splitting Works

- At each node, a decision rule divides the data into two or more subsets.
- These subsets are sent to child nodes.
- This process continues until:
  - A stopping criterion is met (e.g., maximum depth, minimum samples per node).
  - All data points in a subset belong to the same class.

17.3.2 Key Point

Each child node contains a subset of the data from its parent node. Therefore:

$$\text{Objects in child node} \leq \text{Objects in parent node}$$

—

17.4 Example

Let's consider a dataset with 100 objects. At the root node:

- A splitting rule (e.g., $X > 5$) divides the data into two subsets:
  - Left child: 60 objects ($X \leq 5$).
  - Right child: 40 objects ($X > 5$).
- The number of objects in each child node is smaller than in the parent node (100 objects).

—

17.5 Key Takeaways

- In a decision tree, the number of objects decreases as you move from parent nodes to child nodes.
- This property ensures that the tree structure effectively narrows down the data to make predictions.

18 Topic: Bayesian Classifier

18.1 Question 27

- Which of the following is a base hypothesis for a Bayesian classifier? **The attributes must be statistically independent inside each class.**

18.2 Answer

The base hypothesis for a Bayesian classifier, particularly the **Naive Bayes Classifier**, is that the attributes (features) are **statistically independent** given the class. This assumption simplifies the computation of probabilities and makes the model efficient.

—

18.3 Theory Recap

Bayesian Classifiers are probabilistic models based on Bayes' theorem. They predict the probability of a data point belonging to a class by using prior probabilities and the likelihood of features given the class.

18.3.1 Bayes' Theorem

The formula for Bayes' theorem is:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Where:

- $P(C|X)$: Posterior probability (probability of class C given the feature vector X).
- $P(X|C)$: Likelihood (probability of the feature vector X given class C).
- $P(C)$: Prior probability (probability of class C).
- $P(X)$: Evidence (probability of the feature vector X).

—

18.3.2 Naive Bayes Assumption

The **Naive Bayes Classifier** assumes that the features $X_1, X_2, \ldots, X_n$ are conditionally independent given the class C:

$$P(X|C) = P(X_1|C) \cdot P(X_2|C) \cdot \cdots \cdot P(X_n|C)$$

This simplifies the calculation of $P(X|C)$ and makes the classifier computationally efficient.

—

18.4 Example

Let's classify whether an email is **Spam** or **Not Spam** based on two features:

- X_1: Contains the word "discount".
- X_2: Contains the word "offer".

Dataset:

Email Class	X_1(discount)	X_2(offer)
Spam	1	1
Spam	1	0
Not Spam	0	1
Not Spam	0	0

Calculations:

- $P(\text{Spam}) = 2/4 = 0.5$
- $P(\text{Not Spam}) = 2/4 = 0.5$
- $P(X_1 = 1|\text{Spam}) = 1$, $P(X_2 = 1|\text{Spam}) = 0.5$
- $P(X_1 = 1|\text{Not Spam}) = 0$, $P(X_2 = 1|\text{Not Spam}) = 0.5$

To classify a new email with $X_1 = 1$ and $X_2 = 1$:

$$P(\text{Spam}|X) \propto P(X_1 = 1|\text{Spam}) \cdot P(X_2 = 1|\text{Spam}) \cdot P(\text{Spam}) = 1 \cdot 0.5 \cdot 0.5 = 0.25$$

$$P(\text{Not Spam}|X) \propto P(X_1 = 1|\text{Not Spam}) \cdot P(X_2 = 1|\text{Not Spam}) \cdot P(\text{Not Spam}) = 0 \cdot 0.5 \cdot 0.5 = 0$$

Prediction: The email is classified as **Spam**.

—

18.5 Key Takeaways

- The independence assumption in Naive Bayes simplifies probability computations.
- Despite its simplicity, Naive Bayes performs well in many scenarios, such as text classification.
- It works best when features are actually independent or when the independence assumption is reasonably accurate.

19 Topic: Clustering Metrics

19.1 Question 28

- With reference to the total *sum of squared errors* and *separation* of a clustering scheme, which of the statements below is true? **They are strictly correlated, if, changing the clustering scheme, one increases, then the other decreases.**

19.2 Answer

The *sum of squared errors* (SSE) and *separation* in clustering are inversely correlated. Specifically:

- If the **SSE increases**, it means the clustering quality decreases, and the **separation decreases**.
- Conversely, if the **separation increases**, the **SSE decreases**, indicating better-defined clusters.

—

19.3 Theory Recap

Sum of Squared Errors (SSE) and **Separation** are metrics used to evaluate the quality of clustering schemes.

19.3.1 Sum of Squared Errors (SSE)

- SSE measures the compactness of clusters by summing the squared distances of each point from its cluster centroid:

$$\text{SSE} = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \mu_k\|^2$$

  Where:

  - K: Number of clusters.
  - C_k: Points in cluster k.
  - μ_k: Centroid of cluster k.

- A lower SSE indicates tighter and more compact clusters.

19.3.2 Separation

- Separation measures how distinct or well-separated clusters are.
- It is typically calculated as the minimum distance between centroids of different clusters.

$$\text{Separation} = \min_{i \neq j} \|\mu_i - \mu_j\|$$

  Where:

  - μ_i, μ_j: Centroids of clusters i and j.

- Higher separation indicates that clusters are well-separated.

—

19.3.3 Relationship Between SSE and Separation

- **Inverse Relationship:**
  - Reducing SSE typically increases separation, as it ensures points are closer to their cluster centroid and further from other clusters.
  - Increasing separation results in tighter clusters and lower SSE.
- This trade-off is crucial in optimizing clustering algorithms.

—

19.4 Example

Consider a dataset with two clusters:

$$\text{Cluster 1: } \{(1,2),(2,3),(3,3)\}, \quad \text{Cluster 2: } \{(7,8),(8,9),(9,8)\}$$

Step 1: Calculate SSE for Each Cluster

- Centroid of Cluster 1: $(2, 2.67)$
- Centroid of Cluster 2: $(8, 8.33)$
- SSE:
$$\text{SSE} = \sum_{x \in C_1} \|x - \mu_1\|^2 + \sum_{x \in C_2} \|x - \mu_2\|^2$$

Step 2: Calculate Separation

- Separation = Distance between centroids of Cluster 1 and Cluster 2:
$$\text{Separation} = \|(2, 2.67) - (8, 8.33)\| \approx 8.6$$

Observation:

- If SSE decreases (tighter clusters), separation increases.
- If SSE increases, separation decreases.

—

19.5 Key Takeaways

- SSE and Separation are critical metrics for clustering evaluation.
- They are inversely correlated, highlighting the balance between compactness and distinctiveness of clusters.
- Optimizing clustering schemes requires managing this trade-off effectively.

20 Topic: K-Means Clustering

20.1 Question 29

- Which of the statements below is true (one or more)?
  - **Sometimes K-Means stops at a configuration which does not give the minimum distortion for the chosen value of the number of clusters.**
  - **K-Means is quite efficient even for large datasets.**
  - **K-Means is very sensitive to the initial assignment of the centers.**

20.2 Answer

All the statements are true:

1. **K-Means does not always find the global minimum distortion:** Due to its reliance on local optimization, K-Means may converge to a local minimum rather than the global minimum, depending on the initial centroids.
2. **K-Means is efficient:** The algorithm is computationally efficient, with a complexity of $O(n \cdot k \cdot t)$, where:
   - n: Number of data points.
   - k: Number of clusters.
   - t: Number of iterations (typically small).
3. **K-Means is sensitive to initial centroid assignment:** The outcome depends heavily on the initial placement of centroids, which can lead to different cluster configurations.

—

20.3 Theory Recap

K-Means Clustering is an unsupervised learning algorithm that partitions data into k clusters by minimizing the within-cluster sum of squared errors (SSE).

20.3.1 Algorithm Steps

1. Initialize k centroids (randomly or using methods like K-Means++ to improve results).
2. Assign each data point to the nearest centroid based on Euclidean distance.
3. Recompute centroids as the mean of all points assigned to the cluster.
4. Repeat the assignment and centroid recomputation steps until convergence.

—

20.3.2 Key Challenges in K-Means

- **Local Minima:** K-Means minimizes distortion iteratively but may converge to a local minimum, missing the optimal solution.
- **Initialization Sensitivity:** Different initializations can lead to different clustering results. Methods like K-Means++ are designed to address this issue by initializing centroids more effectively.
- **Efficiency:** K-Means is computationally efficient and scales well with the size of the dataset, making it suitable for large datasets.

—

20.4 Example

Consider the following 2D data points:

$$(1,1),(1,2),(2,2),(6,6),(6,7),(7,7)$$

Step 1: Initialize Centroids (Random Initialization)

$$\text{Centroid 1: }(1,1),\quad \text{Centroid 2: }(6,6)$$

Step 2: Assign Points to Nearest Centroid

$$\text{Cluster 1: }\{(1,1),(1,2),(2,2)\},\quad \text{Cluster 2: }\{(6,6),(6,7),(7,7)\}$$

Step 3: Recompute Centroids

$$\text{New Centroid 1: }(1.33,1.67),\quad \text{New Centroid 2: }(6.33,6.67)$$

Repeat until centroids stabilize or the change in SSE is below a threshold.
—

20.5 Key Takeaways

- K-Means is efficient but can converge to suboptimal solutions due to its reliance on initialization.
- Using K-Means++ for better initialization and running the algorithm multiple times with different seeds can improve results.
- Despite its limitations, K-Means remains one of the most popular and widely used clustering algorithms.

21 Topic: DBSCAN Clustering

21.1 Question 30

- Which of the statements below is true (one or more)?
  - **Sometimes DBSCAN stops at a configuration which does not include any cluster.**
  - **DBSCAN can give good performance when clusters have concavities.**
  - **Increasing the radius of the neighborhood can decrease the number of noise points.**

21.2 Answer

All the statements are true:

1. **DBSCAN may result in no clusters:** If the parameters ε (radius of the neighborhood) and $MinPts$ (minimum points) are not appropriately tuned, DBSCAN might classify all points as noise.
2. **DBSCAN performs well with concave clusters:** Unlike K-Means, DBSCAN can identify clusters with non-linear shapes, including concavities, due to its density-based approach.
3. **Increasing ε reduces noise:** A larger radius of the neighborhood increases the density reachability of points, reducing the number of points classified as noise.

—

21.3 Theory Recap

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that groups points based on their density connectivity.

21.3.1 Key Parameters

- ε (eps): Defines the radius of the neighborhood.
- $MinPts$: The minimum number of points required to form a dense region (core point).

21.3.2 DBSCAN Algorithm Steps

1. Identify core points: Points with at least $MinPts$ neighbors within a radius of ε.
2. Expand clusters from core points by connecting them to other core points and density-reachable points.
3. Label points that are not density-reachable as noise.

—

21.4 Example

Dataset:

$$\text{Points: } \{(1,1),(2,1),(1,2),(10,10),(11,11),(50,50)\}$$

Parameters:

$$\varepsilon = 2, \quad MinPts = 2$$

Results:

- Cluster 1: $(1,1),(2,1),(1,2)$
- Cluster 2: $(10,10),(11,11)$
- Noise: $(50,50)$

If ε is increased to 5, point $(50,50)$ could become part of a cluster.
—

21.4.1 Strengths of DBSCAN

- Can detect clusters with arbitrary shapes, including concavities.
- Automatically identifies noise points.

21.4.2 Weaknesses of DBSCAN

- Sensitive to parameter tuning (ε and $MinPts$).
- Struggles with clusters of varying densities.

—

21.5 Key Takeaways

- Proper parameter selection is crucial for DBSCAN to identify meaningful clusters.
- DBSCAN's ability to detect noise and handle non-linear clusters makes it suitable for a wide range of applications.
- Increasing ε reduces the number of noise points but might merge distinct clusters.

22 Topic: Association Rules

22.1 Question 31

- What is the meaning of the statement: "*the support is anti-monotone*"? **The support of an itemset never exceeds the support of its subsets.**

22.2 Answer

The statement "*the support is anti-monotone*" means that:

- If an itemset X has a certain support, then any superset of X (e.g., $X \cup \{A\}$) will have equal or lower support.
- This property is crucial in pruning the search space during frequent itemset mining.

—

22.3 Theory Recap

Support measures how frequently an itemset appears in a transactional dataset. The anti-monotonicity of support means that adding more items to an itemset will not increase its frequency (support).

22.3.1 Definition of Support

For an itemset X:

$$\text{Support}(X) = \frac{\text{Number of transactions containing } X}{\text{Total number of transactions}}$$

22.3.2 Anti-Monotonicity of Support

If $X \subseteq Y$, then:

$$\text{Support}(Y) \leq \text{Support}(X)$$

Example: Consider the transactions:

ID	Items
1	A, B, C
2	A, B
3	B, C
4	A, C
5	B, C

Support Calculation:

- Support({B}) $= \frac{4}{5} = 0.8$.
- Support({B, C}) $= \frac{3}{5} = 0.6$.
- Support({A, B, C}) $= \frac{1}{5} = 0.2$.

From the calculations:

$$\text{Support}(\{B, C\}) \leq \text{Support}(\{B\}), \quad \text{Support}(\{A, B, C\}) \leq \text{Support}(\{B, C\})$$

—

22.3.3 Importance in Frequent Itemset Mining

The anti-monotonicity property is used in algorithms like **Apriori**:

- If an itemset X is infrequent, all its supersets Y $(X \subseteq Y)$ are also infrequent.
- This helps reduce the computational complexity by pruning unnecessary candidate itemsets.

—

22.4 Key Takeaways

- The anti-monotonicity of support ensures that adding more items to an itemset cannot increase its frequency.
- This property is fundamental for optimizing frequent itemset mining algorithms like Apriori and FP-Growth.
- Understanding this property helps in efficiently mining association rules from large datasets.

23 Topic: Coefficient of Determination ($\mathbf{R^2}$)

23.1 Question 32

- What is the coefficient of determination R^2? **Provide an index of goodness for a linear regression model.**

23.2 Answer

The coefficient of determination R^2 is a statistical measure that evaluates how well a linear regression model explains the variability of the dependent variable (Y) in relation to the independent variable(s) (X).

—

23.3 Theory Recap

Definition: R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

23.3.1 Formula for R^2

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

Where:

- SS_{res}: Residual sum of squares, calculated as:

$$\text{SS}_{\text{res}} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- SS_{tot}: Total sum of squares, calculated as:

$$\text{SS}_{\text{tot}} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

- $\hat{y}_i$: Predicted value for observation i.
- y_i: Actual value for observation i.
- $\bar{y}$: Mean of the actual values.

—

23.3.2 Range of R^2

- $R^2 \in [0, 1]$:
  - $R^2 = 0$: The model explains none of the variability in Y.
  - $R^2 = 1$: The model perfectly explains the variability in Y.
- Negative R^2: This can occur if the model fits worse than a horizontal line at $\bar{y}$.

—

23.4 Example

Dataset:

$$\text{Observations: } X = [1, 2, 3, 4], \quad Y = [2, 4, 6, 8]$$

Step 1: Fit a Linear Model

$$Y = 2X$$

Step 2: Calculate Predicted Values

$$\hat{Y} = [2, 4, 6, 8]$$

Step 3: Compute $\mathbf{SS_{res}}$ and $\mathbf{SS_{tot}}$

- Residual Sum of Squares ($\mathrm{SS_{res}}$):

$$\mathrm{SS_{res}} = \sum (Y - \hat{Y})^2 = 0$$

- Total Sum of Squares ($\mathrm{SS_{tot}}$):

$$\mathrm{SS_{tot}} = \sum (Y - \bar{Y})^2 = 20$$

Step 4: Compute R^2

$$R^2 = 1 - \frac{\mathrm{SS_{res}}}{\mathrm{SS_{tot}}} = 1 - \frac{0}{20} = 1$$

Interpretation: The model perfectly explains the variability in Y.
—

23.5 Key Takeaways

- R^2 provides a measure of how well the regression model fits the data.
- A higher R^2 value indicates a better fit, but it does not guarantee that the model is appropriate or useful.
- R^2 should be considered alongside other metrics, such as residual plots and p-values, for comprehensive evaluation.

24 Topic: K-Means Optimization

24.1 Question 33

- What does K-means try to minimize? **The distortion, that is the sum of the squared distances of each point with respect to its centroid.**

24.2 Answer

K-means clustering minimizes the **distortion**, also known as the **within-cluster sum of squared errors (SSE)**. This metric measures the compactness of the clusters by calculating the sum of the squared distances between each data point and its assigned cluster centroid.

—

24.3 Theory Recap

Distortion (or SSE) is the objective function minimized by K-means. It is calculated as:

$$J = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \mu_k\|^2$$

Where:

- J: Objective function (distortion).
- K: Number of clusters.
- C_k: Cluster k.
- x: Data point in cluster C_k.
- μ_k: Centroid of cluster C_k.
- $\|x - \mu_k\|^2$: Squared distance between data point x and its cluster centroid.

—

24.4 Example

Dataset:

$$\text{Points: } \{(1,1),(2,2),(3,3),(8,8),(9,9)\}$$

Step 1: Initialize Centroids (Randomly)

$$\text{Centroid 1: } (1,1), \quad \text{Centroid 2: } (8,8)$$

Step 2: Assign Points to Nearest Centroid

$$\text{Cluster 1: } \{(1,1),(2,2),(3,3)\}, \quad \text{Cluster 2: } \{(8,8),(9,9)\}$$

Step 3: Compute Distortion (SSE)

$$J = \sum_{x \in C_1} \|x - \mu_1\|^2 + \sum_{x \in C_2} \|x - \mu_2\|^2$$

For Cluster 1:

$$\mu_1 = \frac{(1,1) + (2,2) + (3,3)}{3} = (2,2)$$

SSE for Cluster 1 $= \|(1,1)-(2,2)\|^2+\|(2,2)-(2,2)\|^2+\|(3,3)-(2,2)\|^2 = 2$

For Cluster 2:

$$\mu_2 = \frac{(8,8) + (9,9)}{2} = (8.5, 8.5)$$

SSE for Cluster 2 $= \|(8,8) - (8.5,8.5)\|^2 + \|(9,9) - (8.5,8.5)\|^2 = 0.5$

Total Distortion:

$$J = 2 + 0.5 = 2.5$$

—

24.5 Key Takeaways

- K-means minimizes distortion (SSE) to ensure compact and well-separated clusters.
- A lower distortion indicates better clustering, but the algorithm might converge to a local minimum.
- Multiple runs with different initializations (e.g., K-Means++) can help find better solutions.

25 Topic: CRISP-DM Methodology

25.1 Question 34

- Which of the activities below is part of *Business Understanding* in the CRISP methodology? **Which are the resources available (manpower, hardware, software, ...).**

25.2 Answer

In the **Business Understanding** phase of the CRISP-DM methodology, evaluating the available resources, such as manpower, hardware, and software, is a critical activity. This ensures that the project goals are feasible within the constraints of the organization's resources.

—

25.3 Theory Recap

CRISP-DM (Cross Industry Standard Process for Data Mining) is a widely adopted methodology for data mining and machine learning projects. The first phase, **Business Understanding**, focuses on defining the objectives and planning the project.

25.3.1 Objectives of Business Understanding

- Define the **business goals** and expected outcomes.
- Translate business goals into **data mining objectives**.
- Assess the **resources available**, including:
  - **Manpower:** Expertise and team size.
  - **Hardware:** Computational resources.
  - **Software:** Tools and platforms for analysis.
  - **Time:** Project timeline and deadlines.
- Identify the success criteria and evaluation metrics.

—

25.3.2 Example

Scenario: A retail company wants to implement a customer segmentation model.

Steps in Business Understanding:

- Define the goal: Segment customers to improve targeted marketing.
- Identify constraints:
  - **Manpower:** Two data scientists and one marketing expert.

- **Hardware:** A server with 32 GB RAM and 1 TB storage.
- **Software:** Python with scikit-learn and Tableau for visualization.

- Translate the goal into a data mining objective:
  - Perform clustering to group customers based on purchase behavior.
- Define success criteria:
  - Model performance: Cohesion and separation of clusters.
  - Business impact: Increased sales from targeted campaigns.

—

25.4 Key Takeaways

- The **Business Understanding** phase lays the foundation for a successful project by aligning technical objectives with business goals.
- Evaluating resources ensures that the project is feasible and achievable within the given constraints.
- Clear definition of goals and success criteria enables better planning and evaluation.

26 Topic: Outliers and Noise

26.1 Question 35

- Which of the following statements is *true* (one or more)?
  - **Outliers can be due to noise.**
  - **The noise can generate outliers.**

26.2 Answer

Both statements are true:

1. **Outliers can be due to noise:** Noise in the data, such as measurement errors or random variations, can create values that significantly deviate from the rest of the data, appearing as outliers.

2. **The noise can generate outliers:** Random or systematic noise in a dataset may produce extreme values that are identified as outliers.

—

26.3 Theory Recap

Outliers are data points that deviate significantly from the majority of the dataset. They can arise due to various reasons, including noise, errors, or rare events.

26.3.1 Types of Outliers

- **Global Outliers:** Points that are significantly different from all other data points.
- **Contextual Outliers:** Points that are unusual within a specific context (e.g., time-series data).
- **Collective Outliers:** A group of points that deviate together from the dataset.

26.3.2 Noise and Its Impact

Noise refers to random or systematic disturbances in data that do not represent the true underlying process. Noise can:

- Lead to incorrect measurements.
- Create artificial outliers by distorting the original data.
- Reduce the overall quality of the dataset.

—

26.4 Example

Scenario: Consider a dataset of student test scores, where the majority of scores range from 50 to 90. However, a single score of 10 is observed.

Analysis:

- If the score of 10 is due to a recording error (noise), it is an outlier caused by noise.

- Alternatively, if the score of 10 reflects an actual rare event (e.g., a student who didn't attempt the test), it is a legitimate outlier not caused by noise.

—

26.5 Key Takeaways

- Noise is a common source of outliers in datasets.
- Identifying and handling noise and outliers is crucial for improving data quality and model performance.
- Methods like statistical tests, visualization (e.g., box plots), and clustering can help identify and address outliers.

27 Topic: Information Gain and Indices

27.1 Question 36

- In which mining activity the *Information Gain* can be useful? **Classification.**

27.2 Answer

Information Gain is primarily used in **classification tasks**, particularly in decision tree algorithms such as **ID3** and **C4.5**. It measures the reduction in entropy after a dataset is split on an attribute, helping to identify the most informative attribute for making decisions at each node of the tree.

—

27.3 Theory Recap

27.3.1 Information Gain

Information Gain (IG) is a criterion for selecting the best attribute to split the data in decision tree construction. It is based on the concept of **Entropy**, which measures the impurity or randomness in a dataset.

Entropy Formula:

$$H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

Where:

- S: Dataset.
- p_i: Proportion of class i in S.
- c: Number of classes.

Information Gain Formula:

$$IG(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

Where:

- $IG(S, A)$: Information gain of attribute A.
- S_v: Subset of S where $A = v$.
- $|S_v|/|S|$: Proportion of instances with value v for attribute A.

—

27.3.2 Gini Index

The **Gini Index** is another impurity measure used in decision tree algorithms such as **CART** (Classification and Regression Trees). It measures the probability of incorrectly classifying a randomly chosen instance if it is labeled randomly according to the class distribution.

Gini Index Formula:

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2$$

Where:

- p_i: Proportion of class i in dataset S.

Key Characteristics:

- Lower values of Gini Index indicate higher purity.
- It is computationally less intensive than Entropy and does not require logarithmic calculations.

—

27.3.3 Differences Between Information Gain and Gini Index

- **Mathematical Basis:**
  - Information Gain uses Entropy to measure impurity, requiring logarithmic computations.
  - Gini Index uses squared probabilities, making it simpler to calculate.
- **Bias:**
  - Information Gain tends to favor attributes with many values.
  - Gini Index is less biased towards attributes with multiple values.
- **Use Case:**
  - Information Gain is commonly used in **ID3** and **C4.5**.
  - Gini Index is used in **CART**.

—

27.4 Example Comparison

Consider a dataset with two classes (A and B) and an attribute X with three values (X1, X2, X3):

X	Class A	Class B
$X1$	4	0
$X2$	0	4
$X3$	1	1

Step 1: Calculate Entropy and Information Gain

$$H(S) = -\left(\frac{5}{10}\log_2\frac{5}{10} + \frac{5}{10}\log_2\frac{5}{10}\right) = 1$$

For X:

$$IG(S, X) = 1 - \left(\frac{4}{10}\cdot 0 + \frac{4}{10}\cdot 0 + \frac{2}{10}\cdot 1\right) = 0.8$$

Step 2: Calculate Gini Index

$$Gini(S) = 1 - \left(\frac{5}{10}^2 + \frac{5}{10}^2\right) = 0.5$$

For X:

$$Gini(S, X) = \frac{4}{10} \cdot 0 + \frac{4}{10} \cdot 0 + \frac{2}{10} \cdot 0.5 = 0.1$$

Observation: Both metrics indicate that X is a good splitting attribute, but their exact values and computational requirements differ.

—

27.5 Key Takeaways

- Both Information Gain and Gini Index are widely used for splitting criteria in decision tree algorithms.
- Information Gain is more informative but computationally intensive, while Gini Index is simpler and faster.
- The choice between them depends on the algorithm and the specific problem being addressed.

28 Topic: Cross Validation

28.1 Question 37

- What is *cross validation*? **A technique to obtain a good estimation of the performance of a classifier when it will be used with data different from the training set.**

28.2 Answer

Cross Validation is a statistical technique used to evaluate the performance of a machine learning model by testing its ability to generalize to unseen data. It helps estimate the model's performance on data different from the training set, thus reducing overfitting.

—

28.3 Theory Recap

Cross Validation is essential for assessing how well a model will perform in practice. It divides the dataset into subsets (folds) to ensure the model is trained and tested on different portions of the data.

28.3.1 Steps in Cross Validation

1. Split the dataset into k equal-sized subsets (folds).
2. For each fold:
    - Use the fold as the test set.
    - Use the remaining $k - 1$ folds as the training set.
3. Train the model on the training set and evaluate it on the test set.
4. Repeat for all k folds and compute the average performance metric.

28.3.2 Common Types of Cross Validation

- **K-Fold Cross Validation:**
    - The dataset is divided into k folds, and the process above is repeated k times.
    - Commonly used with $k = 5$ or $k = 10$.
- **Stratified K-Fold Cross Validation:**
    - Similar to K-Fold, but ensures each fold maintains the class distribution of the dataset.
- **Leave-One-Out Cross Validation (LOOCV):**
    - Each data point is used as a test set once, and the rest serve as the training set.
    - Provides an unbiased estimate but can be computationally expensive.

—

28.4 Example

Consider a dataset with 10 instances and a 5-Fold Cross Validation:

Dataset:

$$\text{Instances: } \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

Step 1: Divide the dataset into 5 folds:

$$\text{Fold 1: } \{1, 2\}, \quad \text{Fold 2: } \{3, 4\}, \quad \text{Fold 3: } \{5, 6\}, \quad \text{Fold 4: } \{7, 8\}, \quad \text{Fold 5: } \{9, 10\}$$

Step 2: Perform Cross Validation:

- Iteration 1: Use Fold 1 as the test set and Folds 2–5 as the training set.
- Iteration 2: Use Fold 2 as the test set and Folds 1, 3–5 as the training set.
- Repeat until all folds are used as the test set.

Step 3: Compute Average Metric: Calculate the performance metric (e.g., accuracy) for each iteration and average the results.

—

28.5 Advantages of Cross Validation

- Provides a more reliable estimate of model performance compared to a single train-test split.
- Reduces the risk of overfitting by testing the model on unseen data in each fold.
- Ensures all data points are used for training and testing.

28.6 Disadvantages of Cross Validation

- Computationally expensive, especially with large datasets or complex models.
- LOOCV can be particularly slow for large datasets.

—

28.7 Key Takeaways

- Cross Validation is a powerful tool for evaluating model performance.
- K-Fold Cross Validation is widely used due to its balance between computational cost and performance estimation.
- Proper cross validation can help detect and reduce overfitting, leading to more robust models.

29 Topic: Naive Bayes Classifier

29.1 Question 38

- Which of the following preprocessing activities is useful to build a Naive Bayes classifier if the independence hypothesis is violated? **Feature selection.**

29.2 Answer

When the independence assumption of the Naive Bayes classifier is violated, **feature selection** can help improve performance by reducing redundant or correlated features. This helps align the data more closely with the classifier's assumptions.

—

29.3 Theory Recap

Naive Bayes Classifier is a probabilistic model based on Bayes' Theorem, assuming that features are conditionally independent given the class label.

29.3.1 Bayes' Theorem

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Where:

- $P(C|X)$: Posterior probability of class C given features X.
- $P(X|C)$: Likelihood of observing X given class C.
- $P(C)$: Prior probability of class C.

- $P(X)$: Evidence (normalizing constant).

—

29.3.2 Independence Assumption

The Naive Bayes classifier assumes that all features are conditionally independent given the class:

$$P(X|C) = \prod_{i=1}^{n} P(X_i|C)$$

Where:

- X_i: Individual feature.
- n: Total number of features.

If the independence assumption is violated (e.g., when features are highly correlated), the classifier's performance may degrade.

—

29.4 Feature Selection to Address Violations

Feature selection is a preprocessing step to identify and retain the most relevant features for classification, reducing noise and redundancy.

29.4.1 Common Feature Selection Methods

- **Mutual Information:** Measures the dependency between a feature and the target variable.
- **Chi-Square Test:** Evaluates the statistical independence between categorical features and the target.
- **Correlation Analysis:** Identifies and removes highly correlated features.
- **Recursive Feature Elimination (RFE):** Iteratively removes the least important features.

—

29.5 Example

Dataset:

$$\text{Features: } \{X_1, X_2, X_3, X_4\}, \quad \text{Target: } Y$$

Scenario:

- X_1 and X_2 are highly correlated.
- X_3 is irrelevant to Y.

Feature Selection Process:

1. Perform correlation analysis and remove one of X_1 or X_2 (e.g., remove X_2).
2. Calculate mutual information between X_3 and Y. If it is low, remove X_3.
3. Retain X_1 and X_4 as the final features.

—

29.6 Key Takeaways

- The independence assumption in Naive Bayes is often violated in real-world datasets.
- Feature selection is an effective technique to reduce redundancy and improve the model's performance.
- Methods like mutual information and correlation analysis are commonly used to select relevant features.

30 Topic: Min-Max Scaling

30.1 Question 39

- Which is the main reason for the *Min-Max scaling* (also known as *rescaling*) of attributes? **Map all the numeric attributes to the same range, in order to prevent the attributes with higher range from having prevalent influence.**

30.2 Answer

Min-Max scaling is a preprocessing technique used to normalize numerical attributes by rescaling their values to a specified range, typically [0, 1]. This ensures that attributes with larger ranges do not dominate those with smaller ranges during model training.

—

30.3 Theory Recap

Min-Max Scaling (Rescaling) transforms numeric data into a fixed range, maintaining the relative relationships between values while reducing the impact of attributes with higher ranges.

30.3.1 Formula for Min-Max Scaling

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

Where:

- X: Original value of the attribute.
- X_{min}: Minimum value of the attribute in the dataset.
- X_{max}: Maximum value of the attribute in the dataset.
- X_{scaled}: Rescaled value in the range [0, 1].

—

30.3.2 Advantages of Min-Max Scaling

- Ensures that all features contribute equally to the model, avoiding dominance by features with larger ranges.
- Maintains the relative distribution of the original data.
- Suitable for algorithms sensitive to the scale of attributes, such as:
  - Gradient Descent-based models (e.g., Logistic Regression, Neural Networks).
  - Distance-based models (e.g., K-Means, K-Nearest Neighbors).

—

30.4 Example

Dataset:

Attribute (X)	Value
Feature 1	20
Feature 2	50
Feature 3	80
Feature 4	100

Step 1: Identify Min and Max Values

$$X_{\text{min}} = 20, \quad X_{\text{max}} = 100$$

Step 2: Apply Min-Max Scaling Formula

$$\text{For Feature 1: } X_{\text{scaled}} = \frac{20 - 20}{100 - 20} = 0$$

$$\text{For Feature 2: } X_{\text{scaled}} = \frac{50 - 20}{100 - 20} = 0.375$$

$$\text{For Feature 3: } X_{\text{scaled}} = \frac{80 - 20}{100 - 20} = 0.75$$

$$\text{For Feature 4: } X_{\text{scaled}} = \frac{100 - 20}{100 - 20} = 1$$

Rescaled Dataset:

Attribute (X)	Scaled Value
Feature 1	0
Feature 2	0.375
Feature 3	0.75
Feature 4	1

—

30.5 Comparison to Other Scaling Techniques

- **Standardization:**
  - Rescales data to have a mean of 0 and a variance of 1.
  - Formula:
    $$X_{\text{standardized}} = \frac{X - \mu}{\sigma}$$
    Where μ is the mean and σ is the standard deviation.

- **Min-Max Scaling:**
  - Rescales data to a fixed range, typically [0, 1].
  - Does not assume a Gaussian distribution.
- **Robust Scaling:**
  - Uses the median and interquartile range to scale data.
  - Less sensitive to outliers.

—

30.6 Key Takeaways

- Min-Max Scaling is critical for models sensitive to attribute magnitudes.
- It is simple to implement and ensures uniform contribution of attributes during model training.
- Consider the range of values and distribution of your dataset when choosing a scaling technique.

31 Topic: Normalization

31.1 Question 40

- Which is the main reason for the *normalization* (also known as *rescaling*) of numeric attributes? **Map all the numeric attributes to the same range, in order to prevent the attributes (without altering the distribution) with higher range from having prevalent influence.**

31.2 Answer

Normalization is a data preprocessing technique used to scale numeric attributes into a specific range, typically [0, 1]. This ensures that features with higher ranges do not dominate the learning process of machine learning models.

—

31.3 Theory Recap

Normalization is a scaling technique that adjusts the range of numerical features without altering their distribution. It is particularly useful for algorithms sensitive to the magnitude of input features.

31.3.1 Formula for Normalization

$$X_{\text{normalized}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

Where:

- X: Original value of the attribute.
- X_{min}: Minimum value of the attribute in the dataset.
- X_{max}: Maximum value of the attribute in the dataset.
- $X_{\text{normalized}}$: Rescaled value in the range [0, 1].

—

31.3.2 Advantages of Normalization

- Prevents attributes with larger ranges from dominating those with smaller ranges.
- Maintains the relative distribution of data points.
- Reduces numerical instability in algorithms that rely on distance measures (e.g., K-Means, K-Nearest Neighbors).

—

31.4 Example

Dataset:

Attribute (X)	Value
Feature 1	10
Feature 2	50
Feature 3	100
Feature 4	150

Step 1: Identify Min and Max Values

$$X_{\text{min}} = 10, \quad X_{\text{max}} = 150$$

Step 2: Apply Normalization Formula

$$\text{For Feature 1: } X_{\text{normalized}} = \frac{10 - 10}{150 - 10} = 0$$

$$\text{For Feature 2: } X_{\text{normalized}} = \frac{50 - 10}{150 - 10} = 0.2857$$

$$\text{For Feature 3: } X_{\text{normalized}} = \frac{100 - 10}{150 - 10} = 0.6429$$

$$\text{For Feature 4: } X_{\text{normalized}} = \frac{150 - 10}{150 - 10} = 1$$

Normalized Dataset:

Attribute (X)	Normalized Value
Feature 1	0
Feature 2	0.2857
Feature 3	0.6429
Feature 4	1

—

31.5 Comparison to Standardization

- **Normalization:**
  - Rescales data to a fixed range, such as [0, 1].
  - Maintains the original distribution of data points.
- **Standardization:**
  - Transforms data to have a mean of 0 and a standard deviation of 1.
  - Useful for algorithms that assume normally distributed data.
  - Formula:

    $$X_{\text{standardized}} = \frac{X - \mu}{\sigma}$$

    Where μ is the mean and σ is the standard deviation.

—

31.6 Key Takeaways

- Normalization ensures all features contribute equally to the model.
- It is particularly useful for distance-based models and gradient descent algorithms.
- Choose the scaling method based on the requirements of your machine learning algorithm and dataset characteristics.

32 Topic: Feature Selection

32.1 Question 41

- Which of the following is *not* an objective of feature selection? **Select the features with higher range, which have more influence on the computations.**

32.2 Answer

The objective of feature selection is to identify the most relevant and informative features for a model, not to select features based on their range or influence on computations. Features with higher ranges might dominate computations but are not necessarily more informative.

—

32.3 Theory Recap

Feature Selection is a preprocessing technique used in machine learning to select a subset of relevant features for building a model. It reduces dimensionality, improves interpretability, and enhances model performance by eliminating irrelevant or redundant features.

—

32.3.1 Objectives of Feature Selection

1. **Improve Model Performance:**
   - Reduce overfitting by removing noisy or irrelevant features.
   - Enhance model accuracy and generalization.
2. **Reduce Computational Complexity:**

- Lower the time and memory requirements for training.

3. **Improve Model Interpretability:**
   - Simplify the model by focusing on the most important features.
4. **Handle High-Dimensional Data:**
   - Mitigate the curse of dimensionality.

—

32.3.2 Methods for Feature Selection

- **Filter Methods:**
  - Evaluate features based on statistical measures.
  - Examples:
    - Mutual Information.
    - Chi-Square Test.
- **Wrapper Methods:**
  - Use a predictive model to evaluate feature subsets.
  - Examples:
    - Recursive Feature Elimination (RFE).
- **Embedded Methods:**
  - Perform feature selection as part of the model training process.
  - Examples:
    - LASSO (L1 regularization).

—

32.3.3 Misconception: Influence of Feature Range

- Selecting features based on their range or scale can introduce bias.
- Features with large ranges may dominate distance-based algorithms (e.g., K-Means) but are not inherently more informative.
- **Normalization or Standardization** can address the issue of differing feature ranges.

—

32.4 Example

Dataset:

Feature	Range	Importance
Feature 1	$[0, 1]$	*High*
Feature 2	$[1000, 10000]$	*Low*

Analysis:

- Feature 2 has a larger range but does not provide useful information for the target variable.
- Feature 1, despite its smaller range, is more relevant and should be retained.

—

32.5 Key Takeaways

- Feature selection aims to retain the most relevant and informative features, not those with higher ranges.
- Proper scaling techniques, such as normalization, ensure fair contribution of features during training.
- Understanding feature importance is critical for effective model building and interpretation.

33 Topic: Distance Functions

33.1 Question 42

- For each type of data, choose the best-suited distance function:
  - Vector space with real values: **Euclidean Distance.**
  - Boolean data: **Jaccard Coefficient.**
  - Vectors of terms representing documents: **Cosine Distance.**
  - High-dimensional spaces: **Manhattan Distance.**

33.2 Answer

Different distance functions are suitable for different types of data and tasks. Below are the appropriate distance functions for each case along with their formulas and usage scenarios.

—

33.3 Theory Recap and Formulas

33.3.1 1. Euclidean Distance

Formula:

$$d_{\text{Euclidean}}(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

Where:

- p and q are points in n-dimensional space.
- p_i and q_i are the coordinates of p and q in dimension i.

When to Use:

- Suitable for continuous real-valued data.
- Ideal for low-dimensional spaces where the geometry of data is important.

Example: **Points:** $p = (2,3), q = (5,7)$

$$d_{\text{Euclidean}}(p,q) = \sqrt{(5-2)^2 + (7-3)^2} = \sqrt{3^2 + 4^2} = \sqrt{9+16} = 5$$

—

33.3.2 2. Jaccard Coefficient

Formula:

$$\text{Jaccard}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Where:

- A and B are sets (e.g., binary vectors for Boolean data).
- $|A \cap B|$: Number of elements common to both sets.
- $|A \cup B|$: Number of elements in either set.

When to Use:

- Suitable for Boolean data (presence/absence or binary values).
- Effective in applications like text similarity and market basket analysis.

Example: **Sets:** $A = \{1, 1, 0, 1\}, B = \{1, 0, 1, 1\}$

$$\text{Jaccard}(A, B) = \frac{|\{1, 1\}|}{|\{1, 1, 0, 1\}|} = \frac{2}{4} = 0.5$$

—

33.3.3 3. Cosine Distance

Formula:

$$\text{Cosine}(A, B) = 1 - \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Where:

- A and B are vectors.
- A_i, B_i: Components of the vectors.

When to Use:

- Suitable for high-dimensional, sparse data like text documents.
- Measures the similarity in orientation (not magnitude) of vectors.

Example: **Vectors:** $A = [1, 0, 1], B = [1, 1, 0]$

$$\text{Cosine}(A, B) = 1 - \frac{(1 \cdot 1) + (0 \cdot 1) + (1 \cdot 0)}{\sqrt{1^2 + 0^2 + 1^2} \cdot \sqrt{1^2 + 1^2 + 0^2}}$$

$$= 1 - \frac{1}{\sqrt{2} \cdot \sqrt{2}} = 1 - \frac{1}{2} = 0.5$$

—

33.3.4 4. Manhattan Distance

Formula:

$$d_{\text{Manhattan}}(p, q) = \sum_{i=1}^{n} |p_i - q_i|$$

Where:

- p and q are points in n-dimensional space.
- p_i and q_i are the coordinates of p and q in dimension i.

When to Use:

- Effective in high-dimensional spaces.
- Suitable when differences in individual dimensions are more important than geometric distance.

Example: **Points:** $p = (2, 3), q = (5, 7)$

$$d_{\text{Manhattan}}(p, q) = |5 - 2| + |7 - 3| = 3 + 4 = 7$$

—

33.4 Key Takeaways

- Different distance functions are suited for different data types and applications.
- **Euclidean Distance:** Continuous, low-dimensional data.
- **Jaccard Coefficient:** Boolean data and set similarity.
- **Cosine Distance:** Textual data and sparse vectors.
- **Manhattan Distance:** High-dimensional spaces and feature importance.

34 Topic: Overfitting in Classifiers

34.1 Question 43

- When developing a classifier, which of the following is a symptom of *overfitting*? **The error rate in the test set is much greater than the error rate in the training set.**

34.2 Answer

Overfitting occurs when a model learns the training data too well, capturing noise and specific patterns that do not generalize to unseen data. This is evidenced by a low error rate in the training set and a significantly higher error rate in the test set.

—

34.3 Theory Recap

Overfitting is a common problem in machine learning, where a model performs exceptionally well on the training data but fails to generalize to new, unseen data. This typically happens when the model is overly complex relative to the amount of training data or the complexity of the underlying patterns.

—

34.3.1 Symptoms of Overfitting

- The error rate (or loss) on the **training set** is very low.
- The error rate (or loss) on the **test set** is significantly higher than on the training set.
- The model shows high variance, meaning it is highly sensitive to small changes in the training data.

—

34.3.2 Causes of Overfitting

1. **Model Complexity:** A model with too many parameters relative to the data can memorize the training data instead of learning general patterns.
2. **Insufficient Training Data:** When the training data is limited, the model may capture noise instead of meaningful trends.
3. **Lack of Regularization:** Without techniques like L1/L2 regularization or dropout, the model may overemphasize certain patterns.

—

34.3.3 Methods to Prevent Overfitting

- **Cross-Validation:** Use techniques like k-fold cross-validation to evaluate model performance on different subsets of data.
- **Regularization:** Add penalties to the loss function for large parameter values.
- **Pruning (for Decision Trees):** Remove branches that add little value to generalization.

- **Data Augmentation:** Increase the diversity of the training data by applying transformations.
- **Simplify the Model:** Reduce the number of parameters or layers in the model.
- **Early Stopping:** Monitor the validation loss during training and stop when it begins to increase.

—

34.4 Example

Scenario: Suppose you train a classifier to predict whether an email is spam using the following datasets:

- Training Set Accuracy: 98%
- Test Set Accuracy: 65%

Observation: - The model performs very well on the training set, but its performance drops significantly on the test set. This discrepancy is a clear symptom of overfitting.

Steps to Address Overfitting: 1. Use regularization (e.g., L2 penalty). 2. Reduce the model complexity (e.g., fewer features or simpler architecture). 3. Gather more training data if possible. 4. Apply k-fold cross-validation to ensure robust evaluation.

—

34.5 Key Takeaways

- Overfitting is indicated by a large gap between training and test performance.
- Prevent overfitting by balancing model complexity and training data size.
- Techniques like regularization, cross-validation, and pruning can help mitigate overfitting.

—

34.5.1 Example: Visualizing Overfitting with Numbers

Dataset: - A dataset consists of 1000 samples, split into: - 800 samples for training. - 200 samples for testing.

Model A (Simple Model): - Training Set Accuracy: 85% - Test Set Accuracy: 83%

Model B (Overfitting Model): - Training Set Accuracy: 99% - Test Set Accuracy: 60%

—

Step-by-Step Explanation:

1. **Model A (Good Generalization):** - This model captures the underlying patterns in the training data without memorizing noise. - The accuracy on the training set (85%) is similar to the accuracy on the test set (83%), indicating that the model generalizes well to unseen data.

2. **Model B (Overfitting):** - This model has learned the training data almost perfectly (99%), including noise and irrelevant details. - However, when applied to the test set, its performance drops drastically (60%). This indicates that the model cannot generalize to new data and has overfitted the training set.

—

34.5.2 Numerical Calculation Example

Training Data:

$$\text{Training Data (Subset)} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{800}, y_{800})\}$$

Let's assume Model B correctly classifies 792 out of 800 samples:

$$\text{Training Accuracy} = \frac{\text{Correct Predictions on Training Set}}{\text{Total Training Samples}} = \frac{792}{800} = 99\%.$$

Test Data:

$$\text{Test Data (Subset)} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{200}, y_{200})\}$$

Let's assume Model B correctly classifies only 120 out of 200 samples:

$$\text{Test Accuracy} = \frac{\text{Correct Predictions on Test Set}}{\text{Total Test Samples}} = \frac{120}{200} = 60\%.$$

Observation: - The model has memorized the training data but performs poorly on new data, a clear indication of overfitting.

—

34.5.3 Key Metrics Comparison

Metric	Model A (Simple Model)	Model B (Overfitting Model)
Training Accuracy	85%	99%
Test Accuracy	83%	60%

—

34.6 How to Address Overfitting

- **Regularization:** Introduce penalties for overly complex models (e.g., L1/L2 regularization).
- **Simplify the Model:** Reduce the number of parameters or features.
- **Cross-Validation:** Use k-fold cross-validation to ensure robust evaluation.
- **Early Stopping:** Stop training when validation accuracy stops improving.
- **Data Augmentation:** Increase the diversity of training data.

—

35 Topic: Decision Trees - Node Attributes

35.1 Question 44

- In a decision tree, an attribute which is used only in nodes near the leaves... **gives little insight with respect to the target.**

35.2 Answer

Attributes that are used only in nodes near the leaves tend to have minimal impact on the overall decision-making process of the tree. These attributes provide limited information about the target variable.

—

35.3 Theory Recap

Decision trees are hierarchical models that use attributes to split data at various levels. Each attribute contributes to reducing uncertainty about the target variable. Attributes that appear near the root of the tree typically provide more significant insights compared to attributes used near the leaves.

—

35.3.1 Key Points:

1. **Attributes near the root:** - These attributes result in large information gain. - They partition the dataset into subsets that significantly reduce entropy or impurity. - Example: In a decision tree for predicting whether a person buys a product, the attribute ”Age” might appear near the root because it significantly influences purchasing behavior.

2. **Attributes near the leaves:** - These attributes deal with smaller subsets of data. - They generally have less influence on the target variable. - Their selection may be due to minor variations or noise in the data.

—

35.3.2 Example: Decision Tree for Weather Prediction

Consider a dataset for predicting whether to play tennis based on weather conditions. The attributes are:

- **Outlook (Sunny, Overcast, Rainy)**
- **Temperature (Hot, Mild, Cool)**
- **Humidity (High, Normal)**
- **Wind (Weak, Strong)**

The constructed decision tree might look like this:

$$\text{Outlook} \rightarrow \begin{cases} \text{Sunny} & \text{Humidity (High/Normal)} \\ \text{Overcast} & \text{Play Tennis} \\ \text{Rainy} & \text{Wind (Weak/Strong)} \end{cases}$$

1. **Root Node:** - Attribute ”Outlook” is at the root because it provides the largest information gain, splitting the dataset into meaningful groups. 2. **Near Leaves:** - Attributes ”Humidity” and ”Wind” appear near the leaves, providing additional but less significant insights.

—

35.3.3 Insight into the Question:

- Attributes near the leaves are often chosen to resolve fine-grained distinctions in smaller subsets of data. - They contribute minimally to the overall predictive power of the model because they deal with fewer instances and smaller information gain.

—

35.3.4 Key Takeaways:

- Attributes near the root have a greater impact on decision-making and generalization.
- Attributes near the leaves often deal with noise or less critical distinctions.
- Pruning techniques can eliminate attributes near the leaves if they add complexity without improving accuracy.

36 Topic: Hierarchical Agglomerative Clustering

36.1 Question 45

- Which of the statements below about *Hierarchical Agglomerative Clustering* is true? **Requires the definition of distance between set of objects.**

36.2 Answer

Hierarchical Agglomerative Clustering (HAC) requires a definition of distance or similarity between sets of objects. This is critical for determining how clusters are merged at each step of the algorithm.

—

36.3 Theory Recap

Hierarchical Agglomerative Clustering (HAC) is a clustering technique that builds a hierarchy of clusters in a bottom-up manner. Initially, each object is treated as a single cluster, and the algorithm iteratively merges the closest clusters until all objects are grouped into a single cluster or a predefined number of clusters is reached.

—

36.3.1 Steps in HAC

1. **Start with individual clusters:** Each data point is treated as its own cluster.
2. **Calculate distances:** Compute the pairwise distances between all clusters using a distance metric (e.g., Euclidean, Manhattan, etc.).
3. **Merge clusters:** Merge the two closest clusters based on a linkage criterion (e.g., single, complete, average linkage).
4. **Repeat:** Recompute distances and merge until a stopping condition is met (e.g., a single cluster remains or a specified number of clusters is achieved).

—

36.3.2 Linkage Criteria

The linkage criterion determines how the distance between clusters is computed:

- **Single Linkage:** Distance between the closest points in two clusters.

$$d_{\text{single}}(C_1, C_2) = \min_{x \in C_1, y \in C_2} d(x, y)$$

- **Complete Linkage:** Distance between the farthest points in two clusters.

$$d_{\text{complete}}(C_1, C_2) = \max_{x \in C_1, y \in C_2} d(x, y)$$

- **Average Linkage:** Average distance between all pairs of points in the two clusters.

$$d_{\text{average}}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{x \in C_1} \sum_{y \in C_2} d(x, y)$$

- **Centroid Linkage:** Distance between the centroids of the two clusters.

$$d_{\text{centroid}}(C_1, C_2) = d(\text{centroid of } C_1, \text{centroid of } C_2)$$

—

36.4 Example: Hierarchical Agglomerative Clustering

Dataset: Assume we have 4 points in a 2D space:

$$P_1 = (1,1), \quad P_2 = (2,2), \quad P_3 = (5,5), \quad P_4 = (8,8)$$

Steps: 1. Treat each point as its own cluster:

$$C_1 = \{P_1\},\ C_2 = \{P_2\},\ C_3 = \{P_3\},\ C_4 = \{P_4\}$$

2. Compute distances between all clusters (using Euclidean distance):

$$d(P_1, P_2) = \sqrt{(2-1)^2 + (2-1)^2} = \sqrt{2} \approx 1.41$$

$$d(P_1, P_3) = \sqrt{(5-1)^2 + (5-1)^2} = \sqrt{32} \approx 5.66$$

Similar calculations yield:

$$d(P_2, P_3) \approx 4.24, \quad d(P_3, P_4) \approx 4.24, \quad d(P_1, P_4) \approx 9.90$$

3. Merge the closest clusters: - C_1 and C_2 are merged because $d(P_1, P_2) = 1.41$.

4. Recompute distances: - Use the selected linkage criterion (e.g., single linkage, complete linkage) to determine distances between the new clusters and the remaining clusters.

5. Repeat until all clusters are merged or the desired number of clusters is reached.

—

36.5 Applications of HAC

- **Gene Expression Data:** Identifying similar gene patterns.
- **Document Clustering:** Grouping similar documents based on textual content.
- **Image Segmentation:** Grouping pixels based on intensity or color similarity.

—

36.6 Key Takeaways

- HAC builds a hierarchy of clusters, requiring a definition of distance between sets of objects.
- Linkage criteria significantly affect the clustering results.
- HAC is computationally expensive for large datasets, as it requires $O(n^2 \log n)$ computations.

37 Topic: Agglomerative and Divisive Clustering

37.1 Theory Recap

Hierarchical Clustering is a method of clustering that builds a hierarchy of clusters. It can be performed in two main ways:

1. **Agglomerative Clustering:** A *bottom-up* approach.
2. **Divisive Clustering:** A *top-down* approach.

—

37.1.1 1. Agglomerative Clustering

Definition: Agglomerative clustering starts with each object as its own cluster. At each step, the closest clusters are merged until a single cluster remains or a stopping condition is met.

Steps:

1. Treat each data point as a single cluster.
2. Calculate the pairwise distance between clusters using a distance metric (e.g., Euclidean, Manhattan).
3. Merge the two closest clusters based on a linkage criterion (e.g., single, complete, average linkage).
4. Repeat until all data points belong to a single cluster or a specified number of clusters is reached.

—

37.1.2 2. Divisive Clustering

Definition: Divisive clustering starts with all objects in a single cluster. At each step, a cluster is split into smaller clusters until each object is in its own cluster or a stopping condition is met.

Steps:

1. Start with all data points in a single cluster.
2. Divide the cluster into two smaller clusters based on a splitting criterion.
3. Repeat the process on each resulting cluster until every object is in its own cluster or the desired number of clusters is achieved.

—

37.2 Key Differences Between Agglomerative and Divisive Clustering

Aspect	**Agglomerative Clustering**	**Divisive Clustering**
Approach	Bottom-up (start with individual points)	Top-down (start with one c
Complexity	Computationally less expensive	Computationally more exp
Granularity	Merges clusters iteratively	Splits clusters iterative
Example Applications	Gene clustering, document grouping	Rarely used due to high

—

37.3 Example: Comparing Agglomerative and Divisive Clustering

Dataset: Assume we have 6 points:

$$P_1 = (1,1),\ P_2 = (2,2),\ P_3 = (5,5),\ P_4 = (6,6),\ P_5 = (8,8),\ P_6 = (10,10)$$

—

37.3.1 Agglomerative Clustering:

1. **Step 1:** Start with each point as its own cluster:

$$C_1 = \{P_1\},\ C_2 = \{P_2\},\ C_3 = \{P_3\}, \dots,\ C_6 = \{P_6\}$$

2. **Step 2:** Calculate distances and merge the closest clusters:

$$d(P_1, P_2) = \sqrt{(2-1)^2 + (2-1)^2} = \sqrt{2}$$

   Merge P_1 and P_2: $C_1 = \{P_1, P_2\}$.

3. **Step 3:** Repeat until one cluster remains:

$$C_1 = \{P_1, P_2\},\ C_2 = \{P_3, P_4\},\ C_3 = \{P_5, P_6\}$$

—

37.3.2 Divisive Clustering:

1. **Step 1:** Start with all points in one cluster:

$$C = \{P_1, P_2, P_3, P_4, P_5, P_6\}$$

2. **Step 2:** Split the cluster into two subclusters based on a splitting criterion (e.g., k-means clustering):

$$C_1 = \{P_1, P_2, P_3\},\ C_2 = \{P_4, P_5, P_6\}$$

3. **Step 3:** Recursively split each cluster until each point is in its own cluster:

$$C_1 = \{P_1, P_2\},\ C_2 = \{P_3\}, \ldots,\ C_6 = \{P_6\}$$

—

37.4 Applications of Divisive Clustering

- **Hierarchical Models:** Situations where a top-down structure is more intuitive (e.g., taxonomy of species).
- **Small Datasets:** Due to its computational cost, divisive clustering is more suited for smaller datasets.

—

37.5 Key Takeaways

- Agglomerative clustering is more commonly used due to its computational efficiency and intuitive bottom-up approach.
- Divisive clustering can be useful in specific scenarios but is computationally expensive and less frequently applied.
- Both methods require a well-defined distance metric and linkage criteria to determine cluster relationships.

38 Topic: Rule Evaluation

38.1 Question 46

- Match the rule evaluation formulas with their names:
  - $\frac{\text{sup}(A \Rightarrow C)}{\text{sup}(A)}$: **Confidence**
  - $\frac{\text{sup}(A \Rightarrow C)}{\text{sup}(C)}$: **Lift**
  - $\text{sup}(A \cup C) - \text{sup}(A) \cdot \text{sup}(C)$: **Leverage**
  - $\frac{1-\text{sup}(C)}{1-\text{sup}(A \Rightarrow C)}$: **Conviction**

38.2 Answer

The correct matching of rule evaluation formulas to their names is:

- $\frac{\text{sup}(A \Rightarrow C)}{\text{sup}(A)}$: **Confidence**
- $\frac{\text{sup}(A \Rightarrow C)}{\text{sup}(C)}$: **Lift**
- $\text{sup}(A \cup C) - \text{sup}(A) \cdot \text{sup}(C)$: **Leverage**
- $\frac{1-\text{sup}(C)}{1-\text{sup}(A \Rightarrow C)}$: **Conviction**

—

38.3 Theory Recap

Rule evaluation metrics are essential in association rule mining to assess the strength, relevance, and interestingness of the discovered rules.

—

38.3.1 1. Confidence

Definition: Measures the reliability of the rule $A \Rightarrow C$. It is the proportion of transactions containing A that also contain C.

$$\text{Confidence}(A \Rightarrow C) = \frac{\text{sup}(A \cup C)}{\text{sup}(A)}$$

Interpretation: A higher confidence indicates a stronger rule.

—

38.3.2 2. Lift

Definition: Measures how much more likely C is to occur with A compared to its independent occurrence.

$$\text{Lift}(A \Rightarrow C) = \frac{\text{Confidence}(A \Rightarrow C)}{\text{sup}(C)}$$

Interpretation:

- Lift > 1: A and C are positively correlated.
- Lift $= 1$: A and C are independent.
- Lift < 1: A and C are negatively correlated.

—

38.3.3 3. Leverage

Definition: Measures the difference between the observed support of $A \cup C$ and the expected support if A and C were independent.

$$\text{Leverage}(A \Rightarrow C) = \text{sup}(A \cup C) - \text{sup}(A) \cdot \text{sup}(C)$$

Interpretation: Positive leverage indicates a stronger association than expected by chance.

—

38.3.4 4. Conviction

Definition: Measures how strongly the presence of A implies the absence of $\neg C$. It compares the probability of A being associated with C against its expected occurrence by chance.

$$\text{Conviction}(A \Rightarrow C) = \frac{1 - \sup(C)}{1 - \text{Confidence}(A \Rightarrow C)}$$

Interpretation: Conviction values closer to infinity indicate stronger rules, while values close to 1 indicate weak rules.

—

38.3.5 Example: Rule Evaluation

Dataset:

Transaction ID	A	C	$A \cup C$
1	1	1	1
2	1	0	0
3	0	1	0
4	1	1	1
5	0	0	0

Calculations:

- $\sup(A) = \frac{3}{5} = 0.6$
- $\sup(C) = \frac{2}{5} = 0.4$
- $\sup(A \cup C) = \frac{2}{5} = 0.4$

Metrics:

- Confidence:
$$\text{Confidence}(A \Rightarrow C) = \frac{\sup(A \cup C)}{\sup(A)} = \frac{0.4}{0.6} = 0.67$$
- Lift:
$$\text{Lift}(A \Rightarrow C) = \frac{\text{Confidence}(A \Rightarrow C)}{\sup(C)} = \frac{0.67}{0.4} = 1.675$$
- Leverage:
$$\text{Leverage}(A \Rightarrow C) = \sup(A \cup C) - \sup(A) \cdot \sup(C) = 0.4 - (0.6 \cdot 0.4) = 0.16$$
- Conviction:
$$\text{Conviction}(A \Rightarrow C) = \frac{1 - \sup(C)}{1 - \text{Confidence}(A \Rightarrow C)} = \frac{1 - 0.4}{1 - 0.67} = \frac{0.6}{0.33} = 1.82$$

—

38.4 Key Takeaways

- **Confidence** indicates reliability but does not account for baseline probabilities.
- **Lift** evaluates how A and C are related beyond chance.
- **Leverage** quantifies the strength of association compared to independence.
- **Conviction** interprets how strongly A predicts C.

39 Topic: Data Preprocessing

39.1 Question 47

- In data preprocessing, which of the following is **not** an objective of the *aggregation* of attributes? **Obtain a more detailed description of data.**

39.2 Answer

The objective of aggregation in data preprocessing is to simplify data, reduce its dimensionality, and improve processing efficiency. It is **not** aimed at obtaining a more detailed description of the data.

—

39.3 Theory Recap

Aggregation in data preprocessing is the process of combining multiple attributes or objects into a single attribute or object. This is typically done to simplify data representation, reduce noise, or improve the efficiency of data analysis.

—

39.3.1 Objectives of Aggregation

- **Dimensionality Reduction:** By combining attributes, the dataset can be simplified, reducing the number of dimensions and improving computational performance.
- **Noise Reduction:** Aggregating data helps to smooth out noise by considering the combined value instead of individual fluctuations.

- **Improved Analysis Efficiency:** Aggregation reduces the complexity of the dataset, making analysis faster and more efficient.
- **Trend Identification:** Aggregation can highlight general trends or patterns that may not be visible in the raw data.

—

39.3.2 Non-Objective:

Obtaining a **more detailed description of data** is not an objective of aggregation. In fact, aggregation typically reduces the granularity of the data, moving from detailed to more generalized representations.

—

39.4 Example: Aggregation in Action

Dataset: Consider daily temperature readings for a city over a week:

$$\text{Day 1: } 28°\text{C}, \quad \text{Day 2: } 30°\text{C}, \quad \text{Day 3: } 31°\text{C}, \ldots, \text{Day 7: } 29°\text{C}.$$

Aggregation: Calculate the weekly average temperature:

$$\text{Average Temperature} = \frac{28 + 30 + 31 + 29 + \ldots}{7} = 29.43°\text{C}.$$

Impact:

- The aggregation simplifies the dataset from daily values to a single weekly average.
- This reduces dimensionality and smooths out daily fluctuations, focusing on the overall trend.
- However, the detailed day-to-day variations are lost in the process.

—

39.5 Key Takeaways

- Aggregation simplifies and reduces the dimensionality of the dataset.
- It is useful for reducing noise and identifying broader trends.
- It does **not** aim to provide a more detailed description of the data but rather a more generalized view.

40 Topic: Data Preprocessing

40.1 Question 48

- In data preprocessing, which of the following **is** an objective of the *aggregation* of attributes?
  - Obtain a less detailed scale.
  - Reduce the variability of data.
  - Reduce the number of attributes or distinct values.

40.2 Answer

The objectives of aggregation in data preprocessing include:

- Obtaining a less detailed scale.
- Reducing the variability of data.
- Reducing the number of attributes or distinct values.

—

40.3 Theory Recap

Aggregation in data preprocessing is used to simplify data, reduce noise, and improve analysis efficiency by combining multiple attributes or values into a single representative value.

—

40.3.1 Objectives of Aggregation

- **Obtain a Less Detailed Scale:** Aggregation reduces the granularity of data, providing a higher-level overview.
- **Reduce Variability:** By aggregating data points, noise and fluctuations are smoothed, making patterns more apparent.
- **Reduce the Number of Attributes or Distinct Values:** Aggregation combines multiple attributes into a single one, reducing dimensionality and simplifying the dataset.

—

40.4 Example: Aggregation Objectives in Action

Dataset: Consider daily sales data for a product over a month:

$$\text{Daily Sales: } \{150, 200, 220, 180, 170, 190, \ldots\}$$

Aggregation:

1. **Obtain a Less Detailed Scale:** Group the sales data into weekly totals:

$$\text{Week 1 Total: } 150 + 200 + 220 + 180 + 170 + 190 = 1110.$$

2. **Reduce Variability:** Use the weekly averages to smooth fluctuations:

$$\text{Week 1 Average: } \frac{1110}{7} = 158.57.$$

3. **Reduce Attributes:** Instead of daily sales data, store only the weekly totals or averages.

—

40.5 Key Takeaways

- Aggregation simplifies data by reducing the level of detail and variability.
- It is a key step in preprocessing to improve computational efficiency and highlight patterns in the data.
- The goals of aggregation align with reducing the complexity of the dataset while retaining meaningful information.

41 Topic: Dimensionality Reduction

41.1 Question 49

- In order to reduce the dimensionality of a dataset, which is the advantage of Multi Dimensional Scaling (MDS), with respect to Principal Component Analysis (PCA)? **MDS can be used also with categorical data, provided that the matrix of the distance is available, while PCA is limited to vector spaces.**

41.2 Answer

MDS is more versatile than **PCA** as it can handle categorical data as long as a distance matrix is available, while **PCA** is restricted to numeric vector spaces.

—

41.3 Theory Recap

Dimensionality reduction techniques simplify high-dimensional data into fewer dimensions, retaining as much variance or information as possible.

—

41.3.1 1. Principal Component Analysis (PCA)

Definition: PCA is a linear dimensionality reduction technique that transforms the dataset into a new coordinate system where the greatest variance lies along the first axis (principal component), the second greatest along the second axis, and so on.

Steps:

1. Standardize the data.
2. Compute the covariance matrix.
3. Calculate eigenvalues and eigenvectors of the covariance matrix.
4. Select the top k eigenvectors (principal components).
5. Transform the data into the new space using these components.

Formula:

$$Z = X \cdot W$$

Where:

- Z: Transformed data.
- X: Original data matrix.
- W: Matrix of eigenvectors.

Applications:

- Dimensionality reduction for numeric data.

- Visualizing high-dimensional data.
- Noise reduction.

Limitations:

- Works only with numeric data.
- Sensitive to outliers.
- Assumes linear relationships.

—

41.3.2 2. Multi Dimensional Scaling (MDS)

Definition: MDS is a technique for reducing dimensions while preserving pairwise distances or dissimilarities as faithfully as possible.

Steps:

1. Compute a dissimilarity matrix D, where $D(i, j)$ represents the distance between data points i and j.
2. Find coordinates in a lower-dimensional space that best preserve these distances.

Optimization Objective:

$$\text{minimize} \sum_{i,j} \left(d_{ij} - \hat{d}_{ij} \right)^2$$

Where:

- d_{ij}: Original distance between points i and j.
- $\hat{d}_{ij}$: Distance between points in the reduced space.

Advantages:

- Handles both numeric and categorical data (if a distance matrix is provided).
- Does not assume linear relationships.

Limitations:

- Computationally expensive for large datasets.
- Requires a precomputed distance matrix.

—

41.4 Example: PCA vs MDS

Dataset: Consider a dataset with 5 points:

$$\text{Points: } (1,2),(2,3),(3,5),(6,8),(8,9)$$

PCA Example:

1. Standardize the data: Center the points by subtracting the mean.
2. Compute the covariance matrix:
$$\text{Cov} = \begin{bmatrix} 6.8 & 7.6 \\ 7.6 & 8.8 \end{bmatrix}$$
3. Calculate eigenvalues and eigenvectors:
$$\lambda_1 = 14,\ \lambda_2 = 1, \quad \text{Eigenvectors: } [0.8, 0.6], [0.6, -0.8]$$
4. Transform data using the first eigenvector.

MDS Example:

1. Compute the pairwise Euclidean distances:
$$D = \begin{bmatrix} 0 & 1.4 & 3.6 & 7.2 & 10 \\ 1.4 & 0 & 2.2 & 5.8 & 8.6 \\ 3.6 & 2.2 & 0 & 3.6 & 6.4 \\ 7.2 & 5.8 & 3.6 & 0 & 2.8 \\ 10 & 8.6 & 6.4 & 2.8 & 0 \end{bmatrix}$$
2. Find lower-dimensional coordinates that preserve these distances.

—

41.5 Key Takeaways

- **PCA:** Works in vector spaces and is ideal for numeric data. It assumes linear relationships and focuses on maximizing variance.
- **MDS:** Works with any type of data if a distance matrix is available. It does not assume linearity and focuses on preserving pairwise distances.
- **When to Use:**
  - Use PCA for numeric datasets with linear relationships.
  - Use MDS for datasets with mixed or categorical data or when distances are meaningful.

42 Topic: Methods

42.1 Question 50

- Which is different from the others?
  - **Decision tree** *(only supervised method - between K-means, Expectation Maximization, Apriori).*
  - **Decision tree** *(not a clustering method - between K-means, Expectation Maximization, DBSCAN).*

42.2 Answer

Decision tree is different from the others because:

- It is a **supervised learning method**, while the other methods are unsupervised.
- It is **not a clustering method**, whereas K-means, Expectation Maximization, and DBSCAN are clustering techniques.

—

42.3 Theory Recap

42.3.1 1. Apriori Algorithm

Definition: Apriori is an algorithm used in association rule mining to find frequent itemsets and derive association rules based on the frequent itemsets.

Steps:

1. Identify the frequent itemsets using a minimum support threshold.
2. Generate association rules from the frequent itemsets that satisfy a minimum confidence threshold.

Key Concepts:

- **Support:** Fraction of transactions that contain an itemset.

$$\text{Support}(A) = \frac{\text{Number of transactions containing } A}{\text{Total number of transactions}}$$

- **Confidence:** Probability that item C is purchased given A.

$$\text{Confidence}(A \to C) = \frac{\text{Support}(A \cup C)}{\text{Support}(A)}$$

- **Lift:** Measures the strength of association between two items.

$$\text{Lift}(A \to C) = \frac{\text{Confidence}(A \to C)}{\text{Support}(C)}$$

Applications:

- Market basket analysis.
- Recommendation systems.

—

42.3.2 2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Definition: DBSCAN is a clustering algorithm that groups points that are closely packed together while marking points in low-density regions as outliers.

Parameters:

- ε: Maximum radius of a neighborhood for a point to be included in a cluster.
- **MinPts:** Minimum number of points required to form a dense region.

Steps:

1. Identify core points with at least **MinPts** neighbors within ε.
2. Form clusters by connecting core points and their neighbors.
3. Label points that are not reachable from any core point as noise.

Advantages:

- Identifies clusters of arbitrary shapes.
- Handles noise effectively.

Limitations:

- Sensitive to the choice of ε and **MinPts**.
- Computationally expensive for large datasets.

—

42.4 Example: Apriori and DBSCAN

Apriori Example:

- **Dataset:**

Transaction ID	Items Purchased
1	A, B, C
2	A, C
3	B, C, D
4	A, B
5	A, B, C, D

- **Frequent Itemsets:**

$$\text{Support}(A, B) = \frac{3}{5} = 0.6, \quad \text{Support}(A, C) = \frac{4}{5} = 0.8$$

- **Rule:** $A \to C$, **Confidence:**

$$\text{Confidence} = \frac{\text{Support}(A, C)}{\text{Support}(A)} = \frac{0.8}{0.8} = 1.0$$

DBSCAN Example:

- **Dataset:** Points in 2D space: $(1, 1), (2, 2), (2, 3), (8, 8), (8, 9), (25, 80)$.
- **Parameters:** $\varepsilon = 2$, **MinPts** $= 2$.
- **Clusters:**
  - Cluster 1: $(1, 1), (2, 2), (2, 3)$.
  - Cluster 2: $(8, 8), (8, 9)$.
  - Noise: $(25, 80)$.

—

42.5 Key Takeaways

- **Apriori:** Best for finding frequent itemsets and association rules in transactional datasets.
- **DBSCAN:** Ideal for clustering spatial or non-linear data with noise.
- Decision trees differ as they are a supervised learning method, not a clustering or association rule method.

43 Topic: Methods

43.1 Question 51

- Which is different from the others?
  - **DBSCAN** *(not a classification method - between SVM, Neural Network, Decision Tree).*
  - **Silhouette Index** *(not an index for the evaluation of purity - between Misclassification Error, Gini Index, Entropy).*

43.2 Answer

DBSCAN is different because it is a clustering method, not a classification method, while SVM, Neural Networks, and Decision Trees are classification methods.

Silhouette Index is different because it evaluates the quality of clustering, not the purity of a classification model. Misclassification Error, Gini Index, and Entropy are used to evaluate classification purity.

—

43.3 Theory Recap

43.3.1 1. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Recap: As previously discussed, DBSCAN is a clustering algorithm that identifies clusters based on density. It is not used for classification tasks.

—

43.3.2 2. Silhouette Index

Definition: The Silhouette Index measures the quality of clustering by evaluating how similar a point is to its cluster compared to other clusters.

Formula:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Where:

- $a(i)$: Average distance of point i to all other points in the same cluster.
- $b(i)$: Average distance of point i to all points in the nearest cluster.

Interpretation:

- $s(i)$ ranges from -1 to 1:
  - $s(i) \approx 1$: Point is well-clustered.
  - $s(i) \approx 0$: Point lies between clusters.
  - $s(i) \approx -1$: Point is misclassified.

Applications:

- Evaluating clustering quality.
- Comparing different clustering algorithms.

—

43.3.3 3. Indices for Classification Purity

Gini Index:

$$\text{Gini}(t) = 1 - \sum_{i=1}^{C} p_i^2$$

Where p_i is the proportion of class i in node t. It measures impurity, with 0 being pure.

Entropy:

$$\text{Entropy}(t) = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

Measures the level of uncertainty or randomness in a node.

Misclassification Error:

$$\text{Error}(t) = 1 - \max(p_i)$$

Measures the fraction of misclassified instances.

Applications:

- Used in Decision Trees to determine splits.
- Helps identify the purity of nodes.

—

43.4 Key Takeaways

- **DBSCAN:** A clustering method, not suitable for classification tasks.
- **Silhouette Index:** Evaluates the quality of clustering, not classification purity.
- **Classification Indices (Gini, Entropy, Misclassification Error):** Evaluate the purity of classification models.

44 Topic: Overview of Methods

44.1 Supervised Methods

Supervised learning methods are used for tasks where the model is trained on labeled data. The goal is to map inputs to outputs based on the provided labels.

- **Decision Tree:** A tree-structured model where nodes represent tests on attributes, branches represent outcomes, and leaves represent predicted classes or values. Suitable for both classification and regression tasks.
- **Support Vector Machines (SVM):** A classifier that separates classes using a hyperplane with the largest margin. Useful for high-dimensional spaces and works well with both linear and non-linear data using kernels.
- **Neural Networks:** Models inspired by the human brain, consisting of layers of interconnected neurons. They are used for complex tasks like image recognition, natural language processing, and more.
- **Linear Regression:** A method to predict a continuous target variable by fitting a linear equation to the data. It assumes a linear relationship between input variables and the target.
- **Naive Bayes:** A probabilistic classifier based on Bayes' theorem, assuming independence among features. Works well for text classification and spam detection.

—

44.2 Clustering Methods

Clustering methods group data into clusters based on similarity, without needing labels.

- **K-Means:** A partitioning method that minimizes the sum of squared distances between points and their cluster centroids. It is sensitive to the initial placement of centroids.
- **DBSCAN:** Density-Based Spatial Clustering of Applications with Noise identifies clusters as regions of high point density. It handles outliers and non-spherical clusters well.
- **Hierarchical Clustering:** Builds a hierarchy of clusters either through agglomerative (bottom-up) or divisive (top-down) approaches. The result is a dendrogram representing cluster relationships.
- **Expectation-Maximization (EM):** An iterative approach that assumes data is generated from a mixture of distributions. It finds clusters by maximizing the likelihood of the data under the mixture model.

—

44.3 Classification Methods

Classification methods are a subset of supervised learning, aiming to categorize data into predefined classes.

- **Decision Tree:** As described earlier, used for classifying data based on a series of attribute tests.
- **Neural Networks:** Handles classification tasks with layers and neurons, especially effective for large and complex datasets.
- **SVM:** As described earlier, effective for binary and multi-class classification problems.
- **K-Nearest Neighbors (KNN):** A non-parametric method that classifies a point based on the majority class among its nearest neighbors.
- **Logistic Regression:** Predicts probabilities of a binary outcome using a logistic function. It works well for linearly separable data.

—

44.4 Evaluation Indices

Evaluation indices measure the performance of clustering or classification models.

- **Silhouette Index:** Measures clustering quality based on cohesion (within clusters) and separation (between clusters). A value close to 1 indicates good clustering.
- **Gini Index:** Measures impurity in a node. Used in decision trees to select splits. Lower values indicate purer nodes.
- **Entropy:** Measures uncertainty or randomness in a node. Like Gini, it is used in decision trees for splitting.
- **Misclassification Error:** Fraction of misclassified instances. Used to evaluate classification performance.
- **Lift:** Used in association rule mining. It evaluates how much more likely the antecedent and consequent occur together than expected by chance.
- **Confidence:** The probability that the consequent occurs given the antecedent in association rules.
- **Conviction:** Measures the strength of an implication in association rules by comparing the probability of the rule's consequent occurring with its absence.

—

44.5 Key Takeaways

- **Supervised Methods:** Use labeled data to predict outcomes. Include Decision Trees, SVMs, Neural Networks, etc.
- **Clustering Methods:** Group data into clusters without labels. Include K-Means, DBSCAN, Hierarchical Clustering, etc.
- **Classification Methods:** A subset of supervised methods for categorizing data into classes.
- **Evaluation Indices:** Metrics like Gini, Silhouette Index, and Confidence evaluate the performance of methods.

45 Topic: Frequent Itemsets

45.1 Question 52

- How does *pruning* work when generating frequent itemsets? **If an itemset is not frequent, then none of its supersets can be frequent, therefore the frequencies of the supersets are not evaluated.**

45.2 Answer

Pruning in the context of frequent itemset generation leverages the anti-monotonic property of support. The property states:

- If an itemset I is not frequent, then any superset of I cannot be frequent.

This allows algorithms like Apriori to avoid evaluating the support of supersets of infrequent itemsets, significantly reducing computational complexity.

45.3 Theory Recap

Pruning in Frequent Itemset Mining:

- Frequent itemset mining is used in association rule mining to find sets of items that frequently appear together in a transactional dataset.
- Pruning is a technique to eliminate candidate itemsets that cannot be frequent, based on the anti-monotonicity property of support.

45.3.1 Example

Consider a transactional database:

Transaction ID	Items
1	A, B, C
2	A, C
3	B, C
4	A, B
5	A, B, C

Step 1: Generate 1-itemsets and their supports.

$$\text{Support}(\{A\}) = 4/5, \quad \text{Support}(\{B\}) = 4/5, \quad \text{Support}(\{C\}) = 4/5$$

Step 2: Generate 2-itemsets and their supports.

$$\text{Support}(\{A, B\}) = 3/5, \quad \text{Support}(\{A, C\}) = 3/5, \quad \text{Support}(\{B, C\}) = 3/5$$

Step 3: Generate 3-itemsets.

- Candidate: $\{A, B, C\}$
- Support: 3/5

Pruning: If $\{A, C\}$ or $\{B, C\}$ is found to be infrequent (below a minimum support threshold), we can skip evaluating $\{A, B, C\}$.

45.4 Key Takeaways

- **Anti-Monotonicity Property:** If an itemset is not frequent, none of its supersets can be frequent.
- **Efficiency:** Pruning reduces the computational burden by eliminating unnecessary calculations.
- **Applications:** Used in algorithms like Apriori and FP-Growth for association rule mining.

46 Topic: Expectation Maximization (EM)

46.1 Question 53

- What measure is maximized by the *Expectation Maximization (EM)* algorithm for clustering? **The likelihood of a class label, given the values of the attributes of the example.**

46.2 Answer

The *Expectation Maximization (EM)* algorithm aims to maximize the likelihood function, which represents the probability of the observed data given the parameters of a probabilistic model.

46.3 Theory Recap

Expectation Maximization (EM) Algorithm: The EM algorithm is an iterative optimization method used to find the maximum likelihood estimates of parameters in probabilistic models, especially when the data is incomplete or has hidden variables.

46.3.1 Steps of the EM Algorithm

1. **Initialization:** - Start with initial estimates for the model parameters (e.g., means and variances for Gaussian clusters).

2. **Expectation Step (E-Step):** - Calculate the expected value of the hidden variables (e.g., cluster assignments) based on the current parameter estimates. - For each data point, compute the probability it belongs to each cluster using:

$$P(Z = k|X = x) = \frac{P(X = x|Z = k)P(Z = k)}{\sum_{j=1}^{K} P(X = x|Z = j)P(Z = j)}$$

3. **Maximization Step (M-Step):** - Update the parameters of the model to maximize the likelihood of the data, based on the probabilities computed in the E-step. - For example, for Gaussian Mixture Models (GMMs), update the mean (μ), variance (σ^2), and mixing coefficient (π) of each cluster:

$$\mu_k = \frac{\sum_{i=1}^{N} P(Z_i = k|X_i)X_i}{\sum_{i=1}^{N} P(Z_i = k|X_i)}$$

$$\sigma_k^2 = \frac{\sum_{i=1}^{N} P(Z_i = k|X_i)(X_i - \mu_k)^2}{\sum_{i=1}^{N} P(Z_i = k|X_i)}$$

$$\pi_k = \frac{\sum_{i=1}^{N} P(Z_i = k|X_i)}{N}$$

4. **Repeat:** - Alternate between the E-step and M-step until convergence.

46.3.2 Key Concepts

- **Likelihood Function:** Represents the probability of the observed data given the parameters of the model. The goal of EM is to find parameters that maximize this function.

$$L(\theta) = P(X|\theta)$$

- **Applications:** - Clustering (e.g., Gaussian Mixture Models). - Estimating hidden variables in incomplete datasets. - Latent variable modeling in natural language processing and computer vision.

46.4 Example

Consider a dataset of 2D points:

$$X = \{(1.0, 1.0), (1.2, 1.1), (5.0, 5.0), (4.9, 5.1)\}$$

We assume there are two Gaussian clusters with initial parameters:

$$\mu_1 = (1.0, 1.0), \quad \mu_2 = (5.0, 5.0)$$

Step 1: E-Step Compute the probability of each point belonging to each cluster using the Gaussian distribution.

Step 2: M-Step Update the means, variances, and mixing coefficients based on the probabilities computed in the E-step.

Result: After several iterations, the EM algorithm will converge to a solution where the likelihood of the observed data is maximized.

46.5 Key Takeaways

- EM is a powerful method for clustering, especially when data is noisy or incomplete.
- The algorithm alternates between estimating hidden variables (E-step) and optimizing model parameters (M-step).
- EM maximizes the likelihood of the data, making it a probabilistic approach to clustering.
- Common applications include Gaussian Mixture Models, Hidden Markov Models, and missing data imputation.

47 Topic: Information Gain

47.1 Question 54

- The *information gain* is used to ... **select the attribute which maximizes, for a given training set, the ability to predict the class value**.

47.2 Answer

Information Gain is a metric used in decision trees to select the attribute that provides the highest reduction in entropy (uncertainty) when splitting the dataset.

47.3 Theory Recap

Information Gain: Information Gain measures the effectiveness of an attribute in classifying the dataset. It is defined as the reduction in entropy after a dataset is split based on an attribute.

47.3.1 Formulas:

1. **Entropy:**

$$H(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

Where:

- p_i: Probability of class i in the dataset S.
- c: Number of classes.

2. **Information Gain:**

$$IG(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

Where:

- S: Original dataset.
- A: Attribute being evaluated.
- S_v: Subset of S where attribute A has value v.

47.3.2 Steps to Calculate Information Gain:

1. Calculate the entropy of the original dataset $H(S)$. 2. For each attribute A:

- Split the dataset S into subsets S_v based on the values of A.
- Compute the weighted average entropy of the subsets.
- Calculate the information gain:

$$IG(S, A) = H(S) - \text{Weighted Average Entropy of Subsets}$$

3. Select the attribute with the highest information gain for splitting.

47.4 Example

Consider the following dataset:

Outlook	Temperature	Play?
Sunny	*Hot*	*No*
Sunny	*Hot*	*No*
Overcast	*Hot*	*Yes*
Rainy	*Mild*	*Yes*
Rainy	*Cool*	*Yes*
Rainy	*Cool*	*No*
Overcast	*Cool*	*Yes*
Sunny	*Mild*	*No*
Sunny	*Cool*	*Yes*
Rainy	*Mild*	*Yes*
Sunny	*Mild*	*Yes*
Overcast	*Mild*	*Yes*
Overcast	*Hot*	*Yes*
Rainy	*Mild*	*No*

1. Calculate $H(S)$ for the target variable ”Play?”:

$$H(S) = -\left(\frac{9}{14}\log_2\frac{9}{14} + \frac{5}{14}\log_2\frac{5}{14}\right) \approx 0.94$$

2. For the attribute ”Outlook,” split the dataset into subsets:

- $H(\text{Sunny}) \approx 0.97$
- $H(\text{Overcast}) = 0$
- $H(\text{Rainy}) \approx 0.97$

Weighted average entropy:

$$\text{Weighted Entropy} = \frac{5}{14}\cdot 0.97 + \frac{4}{14}\cdot 0 + \frac{5}{14}\cdot 0.97 \approx 0.69$$

3. Calculate $IG(S, \text{Outlook})$:

$$IG(S, \text{Outlook}) = 0.94 - 0.69 = 0.25$$

Repeat for other attributes and select the one with the highest IG.

47.5 Key Takeaways

- **Information Gain** helps in selecting the attribute that best splits the data.
- It is commonly used in decision trees like ID3, C4.5, and CART.
- The attribute with the highest IG reduces uncertainty the most.

48 Topic: Normalization

48.1 Question 55

- In *data preparation*, which is the effect of **normalization**? **Map all the numeric attributes to the same range, without altering the distribution, in order to avoid that attributes with large ranges have more influence.**

48.2 Answer

Normalization ensures that all numeric attributes are rescaled to a specific range, typically [0, 1] or [-1, 1], without distorting their distribution. This prevents attributes with larger ranges from dominating computations.

48.3 Theory Recap

Normalization: Normalization is a data preparation technique used to scale numeric data so that it lies within a specific range. This is particularly important for machine learning algorithms sensitive to the scale of input features, such as distance-based models.

48.3.1 Formula for Min-Max Normalization:

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)}$$

Where:

- x: Original value.
- $\min(X)$: Minimum value of the feature.
- $\max(X)$: Maximum value of the feature.
- x': Normalized value.

48.4 Example

Consider a dataset with a feature X containing the following values:

$$X = \{10, 20, 30, 40, 50\}$$

1. Calculate $\min(X) = 10$ and $\max(X) = 50$. 2. Apply the normalization formula to each value:

- $x' = \frac{10-10}{50-10} = 0.0$
- $x' = \frac{20-10}{50-10} = 0.25$
- $x' = \frac{30-10}{50-10} = 0.5$
- $x' = \frac{40-10}{50-10} = 0.75$
- $x' = \frac{50-10}{50-10} = 1.0$

Resulting normalized feature:

$$X' = \{0.0, 0.25, 0.5, 0.75, 1.0\}$$

48.4.1 Advantages of Normalization:

- Prevents attributes with larger scales from dominating computations.
- Speeds up the convergence of gradient-based optimization algorithms.
- Improves the performance of distance-based models like k-Nearest Neighbors and k-Means clustering.

48.5 Key Takeaways

- Normalization scales numeric features to a consistent range, ensuring balanced influence in calculations.
- It is essential for algorithms sensitive to feature magnitude, especially those relying on distance metrics.
- Unlike standardization, normalization maintains the original distribution of the data.

49 Topic: Clustering Methods

49.1 Question 56

- Which of the following clustering methods is **not** based on distances between objects? **Expectation Maximization**

49.2 Answer

Expectation Maximization (EM) is a clustering method that is not based on direct distance calculations but instead utilizes probabilistic models to cluster data. It assigns probabilities to data points belonging to each cluster and maximizes the likelihood of the observed data.

49.3 Theory Recap

Clustering Methods: Clustering is an unsupervised learning technique used to group similar data points into clusters. Most clustering algorithms are distance-based, but some methods like Expectation Maximization use probabilistic approaches.

49.3.1 Distance-Based Methods:

- **K-Means:** Minimizes the within-cluster sum of squared distances.
- **Hierarchical Clustering:** Groups objects based on distance matrices using linkage criteria.
- **DBSCAN:** Groups points based on density, requiring a minimum number of points within a radius.

49.3.2 Expectation Maximization (EM):

- **Overview:** EM is a probabilistic clustering algorithm based on Gaussian Mixture Models (GMM).
- **Process:**
  1. **Expectation Step (E-Step):** Assign probabilities to data points for belonging to each cluster based on current parameters.
  2. **Maximization Step (M-Step):** Update the parameters (e.g., means, variances) to maximize the likelihood of the data given the cluster assignments.

- **Output:** The algorithm outputs soft cluster assignments, meaning each data point has a probability of belonging to multiple clusters.

49.4 Example of EM Algorithm:

Dataset:

$$\text{Points: } \{(1,2),(2,1),(8,9),(9,8)\}$$

Steps:

1. Initialize cluster parameters (e.g., means and variances of two Gaussian distributions).
2. **E-Step:** Calculate the probability of each point belonging to each cluster.
3. **M-Step:** Update the cluster parameters to maximize the likelihood of the data.
4. Repeat E-Step and M-Step until convergence.

49.5 Key Takeaways

- Distance-based clustering methods like K-Means rely on distance metrics such as Euclidean distance.
- Expectation Maximization uses probabilistic models instead of distances to determine clusters.
- EM is suitable for datasets with overlapping clusters or when a probabilistic interpretation of clusters is desired.

50 Topic: Feature Selection

50.1 Question 57

- In a dataset with D attributes, how many subsets of attributes should be considered for feature selection according to an exhaustive search? $O(2^D)$

50.2 Answer

The number of subsets of attributes to consider for an exhaustive search is 2^D, where D is the total number of attributes.

50.3 Theory Recap

Feature Selection: Feature selection is the process of selecting a subset of relevant attributes (features) to improve the performance of a model, reduce complexity, and avoid overfitting.

50.3.1 Exhaustive Search:

- Involves evaluating every possible subset of features.
- If there are D features, the total number of subsets is:

$$2^D = \binom{D}{0} + \binom{D}{1} + \binom{D}{2} + \ldots + \binom{D}{D}$$

- This includes subsets of size 0 (no features) to size D (all features).

50.3.2 Drawbacks of Exhaustive Search:

- Computationally expensive for large D.
- The complexity grows exponentially, making it infeasible for datasets with high dimensionality.

50.4 Example

Dataset:

$$\text{Features: } \{A, B, C\}$$

All Possible Subsets:

$$\{\}, \{A\}, \{B\}, \{C\}, \{A, B\}, \{A, C\}, \{B, C\}, \{A, B, C\}$$

Total Subsets:

$$2^3 = 8$$

50.5 Alternative Methods:

To address the limitations of exhaustive search, alternative methods include:

- **Heuristic Approaches:** Greedy search, forward selection, backward elimination.
- **Randomized Methods:** Genetic algorithms, simulated annealing.
- **Embedded Methods:** Regularization techniques like Lasso, which select features during model training.

50.6 Key Takeaways:

- Exhaustive search guarantees finding the best subset of features but is computationally expensive.
- For high-dimensional datasets, heuristic and embedded methods are preferred to reduce computational overhead.

51 Topic: Curse of Dimensionality

51.1 Question 58

- Which is the effect of the *curse of dimensionality*? **When the number of dimensions increases, the Euclidean distance becomes less effective to discriminate between points in the space.**

51.2 Answer

The **curse of dimensionality** refers to the phenomena where the performance of distance-based algorithms deteriorates as the number of dimensions in the dataset increases.

51.3 Theory Recap

What is the Curse of Dimensionality?

- In high-dimensional spaces, data points tend to become equidistant, making it difficult for algorithms to distinguish between them.
- Distance metrics, such as Euclidean distance, lose their discriminative power as the dimensionality increases.

51.4 Effect on Euclidean Distance

- In a low-dimensional space, the distance between points varies significantly, allowing for effective discrimination.
- As dimensionality increases, the range of distances between points shrinks, making all points appear almost equidistant.

51.5 Example

Dataset: Consider a dataset with $n = 1000$ points uniformly distributed in a hypercube.

Case 1: 2 Dimensions

- Points are spread across the plane.
- The difference between the minimum and maximum distances is large.

Case 2: 100 Dimensions

- Points are spread across a high-dimensional space.
- The difference between the minimum and maximum distances becomes negligible.
- All points appear nearly equidistant.

51.6 Mathematical Insight

- The volume of a hypersphere in d-dimensions is given by:

$$V = \frac{\pi^{d/2}}{\Gamma(d/2+1)} r^d$$

- As d increases, the volume of the hypersphere becomes concentrated near its surface, leading to sparsity in the interior.
- This sparsity reduces the effectiveness of algorithms that rely on density or proximity.

51.7 Key Takeaways

- High-dimensional datasets require techniques like dimensionality reduction (e.g., PCA, t-SNE) to mitigate the curse of dimensionality.
- Algorithms based on distances (e.g., k-NN, clustering) are particularly affected.
- Careful feature selection or engineering can help reduce dimensionality and improve algorithm performance.

52 Topic: Bayesian Classification

52.1 Question 59

- Which is the main purpose of *smoothing* in Bayesian classification? **Classifying an object containing attribute values which are missing from some classes in the training set.**

52.2 Answer

The main purpose of smoothing in Bayesian classification is to handle cases where some attribute values are missing from the training set for certain classes, ensuring the classifier can still make meaningful predictions.

52.3 Theory Recap

What is Smoothing?

- Smoothing is a technique used to adjust probabilities to prevent zero probabilities when certain attribute values are not observed in the training data for specific classes.
- It ensures that every possible attribute value has a small, non-zero probability.

52.4 Why Smoothing is Necessary?

- In Bayesian classification, the probability of a class is calculated as:

$$P(C|A_1, A_2, \ldots, A_n) \propto P(C) \prod_{i=1}^{n} P(A_i|C)$$

- If any $P(A_i|C)$ is zero, the entire product becomes zero, causing the classifier to fail for that instance.
- Smoothing avoids this problem by assigning a small probability to unseen values.

52.5 Types of Smoothing

- **Laplace Smoothing:**
  - Adds a small constant (usually 1) to all counts.

  - Adjusted probability formula:

$$P(A_i|C) = \frac{\text{count}(A_i, C) + 1}{\text{count}(C) + k}$$

    where k is the total number of possible attribute values.

- **Additive Smoothing:**
  - Generalizes Laplace smoothing by adding a constant α instead of 1.
  - Adjusted probability formula:

$$P(A_i|C) = \frac{\text{count}(A_i, C) + \alpha}{\text{count}(C) + \alpha k}$$

52.6 Example

Dataset:

- Class C_1: $\{A_1, A_2\}$
- Class C_2: $\{A_2, A_3\}$

Scenario:

- Attribute A_3 is not observed in class C_1.
- Without smoothing:

$$P(A_3|C_1) = 0 \implies P(C_1|\{A_3\}) = 0$$

- With Laplace smoothing:

$$P(A_3|C_1) = \frac{0 + 1}{\text{count}(C_1) + k} > 0$$

52.7 Key Takeaways

- Smoothing ensures the classifier is robust to missing attribute values in the training data.
- Laplace smoothing is widely used for categorical data.
- It balances the need to handle unseen values while preserving the observed data's distribution.

53 Topic: DBSCAN Clustering

53.1 Question 60

- Which of the following characteristics of data can reduce the effectiveness of DBSCAN? **Presence of clusters with different densities.**

53.2 Answer

The presence of clusters with different densities can reduce the effectiveness of DBSCAN, as the algorithm relies on density parameters (ϵ and MinPts) to identify clusters. If clusters have varying densities, a single set of parameters may fail to correctly identify all clusters.

53.3 Theory Recap

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups points based on density. It is effective in identifying clusters of arbitrary shapes and separating noise from meaningful clusters.

53.3.1 How DBSCAN Works:

- Two main parameters:
  - ϵ: Maximum distance between two points for them to be considered neighbors.
  - MinPts: Minimum number of points required to form a dense region (cluster core).
- Definitions:
  - **Core Point**: A point with at least MinPts neighbors within ϵ.
  - **Border Point**: A point that is not a core point but is within ϵ of a core point.
  - **Noise Point**: A point that is neither a core point nor a border point.
- Process:
  - Start with an arbitrary point.

- If the point is a core point, create a cluster by expanding its neighbors iteratively.
- Label points as noise if they cannot be assigned to any cluster.

53.3.2 Challenges with Clusters of Different Densities:

- DBSCAN uses a single ϵ and MinPts for the entire dataset.
- If clusters have different densities:
  - A low ϵ may fail to merge sparse clusters.
  - A high ϵ may incorrectly merge dense clusters or include noise points.
- Example:
  - Cluster A has 20 points tightly packed, requiring $\epsilon = 0.1$.
  - Cluster B has 50 points spread out, requiring $\epsilon = 0.5$.
  - Using a single ϵ leads to misclassification.

53.4 Key Takeaways

- DBSCAN is highly effective for datasets with well-separated clusters of similar densities.
- When clusters have different densities, consider:
  - Preprocessing the data to normalize densities.
  - Using advanced density-based algorithms like HDBSCAN (Hierarchical DBSCAN) which adapt to varying densities.

54 Topic: Distance Metrics

54.1 Question 61

- Which of the following types of data allows the use of the **Euclidean distance**? **Point in a vector space.**

54.2 Answer

Euclidean distance is most suitable for data represented as points in a vector space, where all attributes are numeric and continuous. It calculates the straight-line distance between two points in this space.

54.3 Theory Recap

Euclidean Distance: Definition and Formula

- Formula for two points $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ in an n-dimensional space:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

- It assumes:
  - The data points are embedded in a vector space.
  - Attributes are on the same scale or standardized.

Example Calculation

- Points: $\mathbf{p} = (2, 3, 5)$, $\mathbf{q} = (6, 1, 4)$
- Distance:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(2-6)^2 + (3-1)^2 + (5-4)^2} = \sqrt{(-4)^2 + 2^2 + 1^2} = \sqrt{16+4+1} = \sqrt{21} \approx 4.5$$

54.4 Applications

- Clustering algorithms like K-Means.
- Nearest-neighbor classification (e.g., K-Nearest Neighbors).
- Geometric computations in multidimensional spaces.

54.5 Key Considerations

- **Scaling**: Euclidean distance is sensitive to differences in scale between attributes. Standardization or normalization is necessary when the attributes have different units or ranges.
- **High Dimensionality**: In high-dimensional spaces, Euclidean distance can lose its discriminative power due to the *curse of dimensionality.*

55 Topic: Curse of Dimensionality

55.1 Question 62

- Which is the effect of the **curse of dimensionality**? **When the number of dimensions increases, the Euclidean distance becomes less effective to discriminate between points in the space.**

55.2 Answer

The **curse of dimensionality** refers to various phenomena that arise when analyzing data in high-dimensional spaces. A key effect is that distance metrics like Euclidean distance lose their discriminative power as the number of dimensions increases.

55.3 Theory Recap

Definition of Curse of Dimensionality

- In high-dimensional spaces:
  - All points tend to become equidistant.
  - The spread of distances diminishes, making it hard to distinguish between points.
- This affects algorithms that rely on distance, such as:
  - K-Means Clustering
  - Nearest-Neighbor Classification
  - PCA-based dimensionality reduction

Mathematical Intuition

- As dimensions increase, the volume of the space increases exponentially, but the data points remain sparse.
- For a dataset of N points in d-dimensions, the ratio of the nearest neighbor distance to the farthest neighbor distance approaches 1 as d increases:

$$\frac{\text{Nearest Distance}}{\text{Farthest Distance}} \to 1, \text{ as } d \to \infty$$

Example

- Consider 100 points uniformly distributed in a unit cube:
  - In 2D, points are spread across an area of 1.
  - In 10D, the same 100 points occupy a 10-dimensional hypercube, leaving most of the space empty.
- Result: The distance between points becomes nearly identical in higher dimensions.

55.4 Key Effects

- **Loss of Discrimination**: Distance metrics like Euclidean distance become ineffective.
- **Sparsity of Data**: Data points occupy a tiny fraction of the feature space.
- **Computational Overhead**: High-dimensional data requires more computational resources.

55.5 Mitigation Strategies

- **Dimensionality Reduction**: Use techniques like PCA or t-SNE to project data into lower dimensions.
- **Feature Selection**: Retain only the most relevant features.
- **Distance Alternatives**: Use Manhattan or cosine similarity, which can be more effective in high dimensions.

56 Topic: Neural Networks

56.1 Question 63

- What are the hyperparameters of a Neural Network (possibly non-exhaustive)? **Hidden layer structure, learning rate, activation function, number of epochs.**

56.2 Answer

The hyperparameters of a neural network are settings that determine the architecture and learning process of the model but are not learned during training. These parameters are specified before the training process begins.

56.3 Theory Recap

Key Hyperparameters of Neural Networks

- **Hidden Layer Structure**
  - Refers to the number of hidden layers and the number of neurons in each layer.
  - Determines the model's capacity to learn complex patterns.
  - Example: A network with 2 hidden layers, each containing 128 neurons.
- **Learning Rate**
  - Controls the size of the steps taken during optimization.
  - A small learning rate ensures convergence but can be slow, while a large learning rate may lead to overshooting the minimum.
  - Example: Typical values range from 10^{-1} to 10^{-6}.
- **Activation Function**
  - Defines the non-linearity applied to neurons.
  - Examples:
    * ReLU ($f(x) = \max(0, x)$): Effective in deep networks.
    * Sigmoid ($f(x) = \frac{1}{1+e^{-x}}$): Used in binary classification.
    * Tanh ($f(x) = \frac{e^x-e^{-x}}{e^x+e^{-x}}$): Centered around zero.
- **Number of Epochs**
  - Refers to the number of times the entire training dataset is passed through the network.
  - Example: Training for 50 epochs means the dataset is used 50 times.

56.4 Examples

- **Scenario 1: Small Network for Simple Patterns**
  - Hidden Layers: 1 layer with 32 neurons.
  - Learning Rate: 0.01.
  - Activation Function: ReLU.

  - Epochs: 10.

- **Scenario 2: Deep Network for Complex Data**
  - Hidden Layers: 3 layers with 128, 64, and 32 neurons respectively.
  - Learning Rate: 0.001.
  - Activation Function: ReLU for hidden layers, Softmax for output.
  - Epochs: 100.

56.5 Key Considerations

- Hyperparameters significantly impact the model's performance and must often be tuned.
- Common tuning methods include grid search, random search, and Bayesian optimization.

57 Topic: Regression Models

57.1 Question 64

- How can we measure the quality of a trained regression model? **With a formula elaborating the difference between the forecast values and the true ones.**

57.2 Answer

The quality of a trained regression model is typically measured using error metrics that quantify the difference between the predicted values ($\hat{y}$) and the actual true values (y).

57.3 Theory Recap

Common Metrics for Regression Quality

- **Mean Absolute Error (MAE)**

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

  - Measures the average magnitude of errors.

- Example: For $y = [3, 5, 7]$ and $\hat{y} = [4, 4, 6]$,

$$\text{MAE} = \frac{|3-4| + |5-4| + |7-6|}{3} = \frac{1+1+1}{3} = 1$$

- **Mean Squared Error (MSE)**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  - Penalizes larger errors more heavily than MAE.
  - Example: For $y = [3, 5, 7]$ and $\hat{y} = [4, 4, 6]$,

$$\text{MSE} = \frac{(3-4)^2 + (5-4)^2 + (7-6)^2}{3} = \frac{1+1+1}{3} = 1$$

- **Root Mean Squared Error (RMSE)**

$$\text{RMSE} = \sqrt{\text{MSE}}$$

  - Provides the error in the same units as the target variable.
  - Example:

$$\text{RMSE} = \sqrt{1} = 1$$

- **R-squared (R^2)**

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

  - Measures the proportion of variance in the dependent variable explained by the model.
  - Example: For a perfect fit, $R^2 = 1$, and for a model predicting the mean, $R^2 = 0$.

57.4 Key Considerations

- Low values of error metrics (e.g., MAE, MSE, RMSE) indicate better model performance.
- High R^2 values indicate that the model explains most of the variability in the data.

58 Topic: Classification vs Regression

58.1 Question 65

- What is the difference between classification and regression? **Classification has a categorical target, while regression has a numeric target.**

58.2 Answer

The fundamental difference between classification and regression lies in the type of target variable being predicted:

- **Classification:** Predicts categorical labels or classes (e.g., spam vs non-spam, disease vs no disease).
- **Regression:** Predicts continuous numerical values (e.g., house prices, stock prices).

58.3 Theory Recap

Classification

- The goal is to assign inputs into predefined categories or labels.
- Examples:
  - Email classification: Spam ($y = 1$) vs Not Spam ($y = 0$).
  - Disease detection: Positive ($y = 1$) vs Negative ($y = 0$).
- Algorithms:
  - Logistic Regression.
  - Support Vector Machines (SVM).
  - Decision Trees and Random Forests.
  - Neural Networks.
- Performance Metrics:
  - Accuracy.
  - Precision, Recall, and F1-Score.
  - ROC-AUC.

Regression

- The goal is to predict a continuous numerical output.
- Examples:
  - Predicting house prices based on features like size and location.
  - Estimating temperature based on historical data.
- Algorithms:
  - Linear Regression.
  - Polynomial Regression.
  - Support Vector Regression (SVR).
  - Neural Networks.
- Performance Metrics:
  - Mean Absolute Error (MAE).
  - Mean Squared Error (MSE).
  - R^2 (Coefficient of Determination).

58.4 Example Comparison

- **Classification:**
  - Task: Predict if a student will pass ($y = 1$) or fail ($y = 0$) based on hours studied.
  - Input: $X = [1, 2, 3, 4, 5]$ (hours studied).
  - Output: $Y = [0, 0, 1, 1, 1]$ (pass/fail labels).
- **Regression:**
  - Task: Predict the score of a student based on hours studied.
  - Input: $X = [1, 2, 3, 4, 5]$ (hours studied).
  - Output: $Y = [20, 40, 60, 80, 100]$ (scores).

58.5 Key Takeaways

- Classification deals with predicting categories, while regression deals with predicting numerical values.
- The choice of algorithm and evaluation metric depends on the nature of the problem and the type of target variable.

59 Topic: Principal Component Analysis (PCA)

59.1 Question 66

- In feature selection, what is the Principal Component Analysis? **A mathematical technique used to transform a set of numeric attributes into a smaller set of numeric attributes which capture most of the variability in data.**

59.2 Answer

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional form, while retaining as much variability as possible. It achieves this by identifying the directions (principal components) along which the variance in the data is maximized.

59.3 Theory Recap

Steps in PCA:

1. **Standardize the data:** Transform the features so that they have zero mean and unit variance.

2. **Compute the covariance matrix:** The covariance matrix represents the relationships between features.

$$\text{Cov}(X) = \frac{1}{n-1} X^T X$$

3. **Find the eigenvalues and eigenvectors:** The eigenvalues represent the amount of variance captured by each principal component, and the eigenvectors represent the directions of these components.

4. **Sort and select components:** Order the eigenvalues in descending order, and select the top k components that capture the desired amount of variance.

5. **Transform the data:** Project the original data onto the selected principal components to obtain the reduced representation.

$$X_{\text{reduced}} = X W_k$$

   where W_k is the matrix of the top k eigenvectors.

59.4 Example

Original Data:

- Attributes: Height (cm), Weight (kg).
- Data:

$$\begin{bmatrix} 170 & 65 \\ 180 & 75 \\ 160 & 55 \\ 150 & 50 \end{bmatrix}$$

Steps:

1. Standardize the data:

$$\begin{bmatrix} 0.5 & 0.4 \\ 0.8 & 0.6 \\ -0.5 & -0.4 \\ -0.8 & -0.6 \end{bmatrix}$$

2. Compute the covariance matrix:

$$\text{Cov}(X) = \begin{bmatrix} 0.75 & 0.6 \\ 0.6 & 0.48 \end{bmatrix}$$

3. Find eigenvalues and eigenvectors:
   - Eigenvalues: $\lambda_1 = 1.2, \lambda_2 = 0.03$.
   - Eigenvectors: $v_1 = [0.8, 0.6], v_2 = [-0.6, 0.8]$.

4. Select top $k = 1$ component and transform the data:

$$X_{\text{reduced}} = X \cdot v_1 = \begin{bmatrix} 1.0 \\ 1.2 \\ -1.0 \\ -1.2 \end{bmatrix}$$

59.5 Key Takeaways

- PCA reduces the dimensionality of data by transforming it into a smaller set of principal components.
- It helps eliminate redundant features, improves computational efficiency, and retains most of the data's variability.
- PCA works best with numerical data and when relationships between variables are linear.

60 Topic: Backpropagation

60.1 Question 67

- In a Neural Network, what is the **backpropagation**? **The technique used to adjust the connection weights according to the difference between the desired output and the output generated by the network.**

60.2 Theory Recap

Backpropagation is a supervised learning algorithm used in training artificial neural networks. It works by propagating the error signal (difference between actual and predicted output) backward through the network, allowing the weights to be updated in a way that minimizes the error.

60.3 Steps of Backpropagation

1. **Forward Pass:**

    - Input data is passed through the network layer by layer to produce an output.
    - Compute the loss L, typically using a loss function such as Mean Squared Error (MSE):

    $$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2. **Backward Pass (Error Computation):**

    - Compute the gradient of the loss function with respect to the network's output.

    $$\frac{\partial L}{\partial \hat{y}_i}$$

    - Using the chain rule of calculus, compute the gradient of the loss with respect to the weights and biases in each layer.

      For a weight w:

    $$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w}$$

3. **Weight Update:**

- Update the weights using the gradient and the learning rate η:

$$w_{\text{new}} = w_{\text{old}} - \eta \cdot \frac{\partial L}{\partial w}$$

60.4 Example

Network Setup:

- Single-layer network with inputs x_1, x_2, weights w_1, w_2, bias b, and output $\hat{y}$.
- Activation function: Sigmoid:

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

- Target: $y = 1$.
- Initial weights: $w_1 = 0.5, w_2 = 0.3$, $b = 0.1$.
- Inputs: $x_1 = 1, x_2 = 0$.

Forward Pass:

- Compute weighted sum:

$$z = w_1x_1 + w_2x_2 + b = (0.5 \cdot 1) + (0.3 \cdot 0) + 0.1 = 0.6$$

- Compute output:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-0.6}} \approx 0.645$$

Loss Computation:

- Loss (MSE):

$$L = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(1 - 0.645)^2 \approx 0.063$$

Backward Pass:

- Compute gradient of loss with respect to $\hat{y}$:

$$\frac{\partial L}{\partial \hat{y}} = -(y - \hat{y}) = -(1 - 0.645) \approx -0.355$$

- Compute gradient with respect to z:

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} = -0.355 \cdot \hat{y}(1-\hat{y}) \approx -0.355 \cdot 0.645 \cdot 0.355 \approx -0.082$$

- Compute gradient with respect to weights:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z} \cdot x_1 \approx -0.082 \cdot 1 = -0.082$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z} \cdot x_2 \approx -0.082 \cdot 0 = 0$$

Weight Update:

- Update weights using a learning rate $\eta = 0.1$:

$$w_1 = w_1 - \eta \cdot \frac{\partial L}{\partial w_1} \approx 0.5 - (0.1 \cdot -0.082) \approx 0.5082$$

$$w_2 = w_2 - \eta \cdot \frac{\partial L}{\partial w_2} \approx 0.3$$

60.5 Key Takeaways

- Backpropagation iteratively adjusts weights to minimize error, improving the network's predictive accuracy.
- It is efficient for training multi-layer networks.
- Backpropagation requires a differentiable activation function.