Ml (4341603) - Problem Solver

Antigravity AI

June 4, 2026

Ml

First Edition: 2026

Author: Antigravity AI
Website: milav.in
YouTube: @milav.dabgar

To my students,
who constantly inspire me to find new analogies,
break down complex theories, and make learning
an engineered science.

Preface

About this Book

Welcome to **Computer Networks**! This textbook is meticulously designed to serve as a comprehensive problem solver and study companion. It provides a robust and intuitive foundation in the core principles of this subject, bridging the gap between theoretical models and practical applications. A unique feature of this book is its focus on decoding complex concepts using clear, step-by-step methodologies and real-world analogies.

Target Audience

This book is intended for students and professionals looking to master the concepts of Computer Networks. Whether you are revising for exams, seeking to solidify your fundamental understanding, or looking for practical examples to clarify abstract concepts, this book has been structured to support your learning journey. It serves as an accessible guide designed to remove the fear of complex topics, replacing it with an appreciation for the underlying mechanics.

Scope and Coverage

The coverage is divided logically to follow standard academic curriculums. Each chapter focuses on key topics, breaking them down into digestible sections:

- **Theoretical Insights** – Core concepts explained intuitively.
- **Method Blocks** – Standardized, step-by-step procedures for solving specific types of problems.
- **Worked Examples** – Fully solved problems that apply the methods in practice.

We hope this resource proves invaluable in your studies and helps you achieve excellence in **Computer Networks**. Happy learning!

Acknowledgments

Writing a comprehensive book that connects the fundamental forces of Chemistry with practical engineering applications requires more than just academic knowledge—it requires a community of support. I would like to express my sincere gratitude to everyone who contributed to the realization of this project.

Specially, I extend my heartfelt gratitude to the **Department of Technical Education (DTE), Government of Gujarat**, and **Government Polytechnic, Palanpur**, for providing an environment that fosters innovation, academic rigor, and continuous professional development.

I am deeply thankful to my family for their unwavering patience and support during the countless hours spent researching, formatting, and refining these chapters.

Finally, my deepest appreciation goes to my students. Your curiosity, challenging questions, and continuous feedback are the true driving force behind the unique analogies and streamlined structure of this book. This textbook was engineered for you.

Antigravity AI

About the Author

Milav Jayeshkumar Dabgar is an engineering educator and R&D professional with over 9 years of experience in electronics hardware development, embedded systems, AI/ML, and full-stack software engineering. He currently serves as a **Lecturer (Class - II) & Innovation Leader** at Government Polytechnic, Palanpur, under the Education Department of the Government of Gujarat.

Milav holds a Master of Engineering (ME) in Communication Systems from L.D. College of Engineering and a Bachelor of Engineering (BE) in Electronics & Communication. He is also currently pursuing a **BS in Data Science and Applications from IIT Madras**, where he has completed both the **Diploma in Programming** and the **Diploma in Data Science** with distinction.

Most recently, he completed the **AICTE QIP PG Certificate Program in Deep Learning from SVNIT Surat** with a **perfect 10/10 CGPA**, topping his batch.

A dedicated mentor, Milav has guided numerous student teams in securing over **INR 45 lakhs** in innovation funding, including INR 25 lakhs from Shark Tank India and INR 20 lakhs through iHub Gujarat, resulting in two student patents. He is recognized as an **NPTEL Evangelist** and **NPTEL Discipline Star** in Computer Science, reflecting his commitment to continuous learning and academic excellence.

His technical expertise spans from embedded systems (8051, STM32, Arduino) to modern web technologies (Next.js, Python, PostgreSQL) and Data Science (TensorFlow, PyTorch). Through this textbook, he aims to simplify complex Engineering Chemistry concepts by integrating relatable, real-world analogies drawn from modern tech and engineering landscapes.

Milav is also an active content creator, running a popular **YouTube channel** (@milav.dabgar) where he shares technical tutorials and academic resources. He maintains a personal blog and resource hub at milav.in and can be reached for professional collaborations on LinkedIn.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction to Machine Learning

1.1 Basic Concept of Machine Learning and Its Applications

Machine learning is a subset of artificial intelligence that focuses on building systems that learn from data. Instead of being explicitly programmed to solve a task, a machine learning model identifies patterns in training data to make predictions on new unseen data. Common applications include email spam detection, house price prediction, medical diagnosis, and recommendation systems.

To understand the mechanics of machine learning from a computational perspective, we must examine how models compare data points to make decisions and how we measure the success of these predictions.

1.1.1 Measuring Data Similarity

Many machine learning applications (such as K-Nearest Neighbors for classification or recommendation engines) rely on calculating the similarity or distance between data points. The most common metric used for continuous features is the Euclidean distance.

Methodology:

Computing Euclidean Distance To find the distance between two data points A and B in an n-dimensional space, follow these computational steps:

1. Identify the feature values for both data points: $A = (x_1, x_2, \ldots, x_n)$ and $B = (y_1, y_2, \ldots, y_n)$.
2. For each feature i, compute the squared difference: $(x_i - y_i)^2$.
3. Sum all the squared differences: $\sum_{i=1}^{n}(x_i - y_i)^2$.
4. Take the square root of the sum to find the Euclidean distance d:

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

The smaller the computed distance, the more similar the two data points are.

Worked Example:

Calculating Similarity for a Recommendation System A movie recommendation system represents movies using two numerical features: Action Score and Comedy Score (both on a scale of 1 to 10). Movie A (Action: 8, Comedy: 3)
Movie B (Action: 7, Comedy: 4)
Movie C (Action: 2, Comedy: 9)
Calculate the Euclidean distance between Movie A and Movie B and between Movie A and Movie C. Which movie is more similar to Movie A?
Solution:
Step 1: Extract feature coordinates

- Movie A: $(x_1, x_2) = (8, 3)$
- Movie B: $(y_1, y_2) = (7, 4)$
- Movie C: $(z_1, z_2) = (2, 9)$

Step 2: Calculate distance between A and B

$$\begin{aligned} d(A, B) &= \sqrt{(8-7)^2 + (3-4)^2} \\ &= \sqrt{(1)^2 + (-1)^2} \\ &= \sqrt{1+1} \\ &= \sqrt{2} \approx 1.414 \end{aligned}$$

Step 3: Calculate distance between A and C

$$\begin{aligned} d(A, C) &= \sqrt{(8-2)^2 + (3-9)^2} \\ &= \sqrt{(6)^2 + (-6)^2} \\ &= \sqrt{36+36} \\ &= \sqrt{72} \approx 8.485 \end{aligned}$$

Step 4: Conclusion Since $1.414 < 8.485$, the distance between Movie A and Movie B is much smaller than the distance between Movie A and Movie C. Therefore, Movie B is significantly more similar to Movie A and would be a better recommendation.

1.1.2 Evaluating Model Performance

A core concept in machine learning is evaluating how well a trained model performs on test data. The mathematical evaluation method depends on whether the application is a classification task (predicting categories like "Spam" or "Not Spam") or a regression task (predicting continuous numerical values like "Price").

Methodology:

Calculating Classification Accuracy For classification tasks, accuracy measures the proportion of correct predictions out of the total predictions made by the model.

1. Count the number of Correct Predictions made by the model.
2. Count the Total Number of Predictions.
3. Compute the accuracy using the formula:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \times 100\%$$

Worked Example:

Evaluating a Spam Filter Application An email provider deploys a machine learning model to classify emails as "Spam" or "Not Spam". Out of 50 recent emails, the model correctly identified 12 spam emails and correctly identified 33 normal emails. It misclassified 5 emails. Calculate the overall accuracy of the model.
Solution:
Step 1: Identify correct predictions Correct spam predictions = 12
Correct normal predictions = 33
Total correct predictions = $12 + 33 = 45$
Step 2: Identify total predictions The model predicted a class for all 50 emails. Total predictions = 50.

Step 3: Calculate accuracy

$$\begin{aligned}\text{Accuracy} &= \frac{45}{50} \times 100\% \\ &= 0.90 \times 100\% \\ &= 90\%\end{aligned}$$

The spam filter application has an accuracy of 90%.

Methodology:

Calculating Mean Squared Error For regression applications, where the output is a continuous numerical value, we measure performance using the Mean Squared Error (MSE).

1. For each data point i, identify the Actual Value (y_i) and the Predicted Value ($\hat{y}_i$).
2. Calculate the error (difference) for each prediction: $e_i = y_i - \hat{y}_i$.
3. Square each error: $e_i^2 = (y_i - \hat{y}_i)^2$.
4. Calculate the average of these squared errors across all n data points:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

A lower MSE indicates a model that makes predictions closer to the actual values.

Worked Example:

Evaluating a House Price Prediction Model A real estate application uses a machine learning model to predict house prices (in thousands of dollars). Test the model's performance on a dataset of 4 houses using the Actual and Predicted prices given below:

House	Actual Price (y)	Predicted Price ($\hat{y}$)
1	250	240
2	300	315
3	150	150
4	400	380

Calculate the Mean Squared Error of the model.
Solution:
Step 1: Calculate the error for each house

- House 1: $y_1 - \hat{y}_1 = 250 - 240 = 10$
- House 2: $y_2 - \hat{y}_2 = 300 - 315 = -15$
- House 3: $y_3 - \hat{y}_3 = 150 - 150 = 0$
- House 4: $y_4 - \hat{y}_4 = 400 - 380 = 20$

Step 2: Square each error

- House 1: $(10)^2 = 100$
- House 2: $(-15)^2 = 225$
- House 3: $(0)^2 = 0$
- House 4: $(20)^2 = 400$

Step 3: Sum the squared errors

$$\text{Sum} = 100 + 225 + 0 + 400 = 725$$

Step 4: Calculate the average (MSE) There are $n = 4$ houses.

$$\text{MSE} = \frac{725}{4} = 181.25$$

The Mean Squared Error for the house price prediction application is 181.25.

CHAPTER 2

Preparing to Model

2.1 Different Types of Machine Learning Activities

Machine learning activities involve taking input data and mapping it to a desired output or finding hidden structures. For practical problem-solving, these activities dictate which algorithms and preprocessing techniques you must apply.

Methodology:

Identifying Machine Learning Activities To determine the correct machine learning activity for a given dataset, follow these computational steps:

1. **Analyze the Target Variable:** Does the problem require predicting an output (Supervised Learning) or finding patterns without predefined labels (Unsupervised Learning)?
2. **Determine the Data Type of the Target:**
   - If predicting a continuous numerical value (e.g. price or temperature) → **Regression**.
   - If predicting a discrete category or class (e.g. spam or not spam) → **Classification**.
3. **Evaluate Unlabeled Goals:**
   - If grouping similar data points together based on feature distance → **Clustering**.
   - If reducing the number of features while retaining maximum variance → **Dimensionality Reduction**.

Worked Example:

Identify the ML Activity for Given Scenarios **Problem:** Determine the appropriate machine learning activity (Classification Regression or Clustering) for the following datasets:

1. Predicting the price of a house based on its area and number of bedrooms.
2. Grouping retail customers into distinct segments based on their purchasing behavior.
3. Detecting whether an incoming email should be labeled as "Spam" or "Not Spam".

Solution:

1. **Target:** House price (Numerical/Continuous).
   Activity: Regression.
2. **Target:** None (No predefined labels).
   Activity: Clustering.
3. **Target:** Email label (Categorical/Discrete).
   Activity: Classification.

2.2 Types of Data and Data Preprocessing

Data preprocessing transforms raw data into a format suitable for computational modeling. Before applying algorithms, it is critical to classify the data types and handle missing values scaling and encoding mathematically.

2.2.1 Handling Missing Data

Methodology:

Mean and Median Imputation Missing numerical values can be filled using the central tendency of the feature.

1. **Mean Imputation:** Replace missing values with the arithmetic average of the non-missing values.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

2. **Median Imputation:** Replace missing values with the middle value of the sorted non-missing data. Preferred if outliers are present in the dataset.

3. **Mode Imputation:** Replace missing values with the most frequent value. Typically used for categorical data.

Worked Example:

Imputing Missing Values **Problem:** A dataset of student marks contains missing values: $X = \{45, 50, \text{NaN}, 85, 40\}$. Impute the missing value using Mean Imputation.
Solution: Step 1: Extract the valid non-missing values: $45, 50, 85, 40$.
Step 2: Count the number of valid values ($n = 4$).
Step 3: Calculate the mean of the valid values:

$$\mu = \frac{45 + 50 + 85 + 40}{4} = \frac{220}{4} = 55$$

Step 4: Replace 'NaN' with the computed mean.
Result: The imputed dataset is $X = \{45, 50, 55, 85, 40\}$.

2.2.2 Feature Scaling

Machine learning algorithms that compute distances (like K-Nearest Neighbors) are highly sensitive to the scale of features. Feature scaling normalizes the range of independent variables.

Methodology:

Min-Max Normalization Min-Max scaling transforms features to lie within a specific range usually $[0, 1]$.

1. Find the minimum value (X_{min}) and maximum value (X_{max}) of the feature column.

2. Apply the transformation formula to each data point X:

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

Worked Example:

Perform Min-Max Normalization **Problem:** Normalize the dataset $X = \{10, 20, 30, 40, 50\}$ to the range $[0, 1]$ using Min-Max Normalization.

Solution: Step 1: Identify X_{min} and X_{max}.

$$X_{min} = 10, \quad X_{max} = 50$$

Step 2: Compute the denominator $X_{max} - X_{min}$.

$$50 - 10 = 40$$

Step 3: Apply the formula to each value.

- For $X = 10$: $X_{scaled} = \frac{10-10}{40} = 0.0$
- For $X = 20$: $X_{scaled} = \frac{20-10}{40} = \frac{10}{40} = 0.25$
- For $X = 30$: $X_{scaled} = \frac{30-10}{40} = \frac{20}{40} = 0.5$
- For $X = 40$: $X_{scaled} = \frac{40-10}{40} = \frac{30}{40} = 0.75$
- For $X = 50$: $X_{scaled} = \frac{50-10}{40} = \frac{40}{40} = 1.0$

Result: The normalized dataset is $\{0.0, 0.25, 0.5, 0.75, 1.0\}$.

Methodology:

Z-Score Standardization Standardization transforms the data to have a mean of 0 and a standard deviation of 1.

1. Calculate the mean (μ) of the feature column.
2. Calculate the standard deviation (σ) of the feature column.
3. Apply the transformation formula to each data point X:

$$Z = \frac{X - \mu}{\sigma}$$

Worked Example:

Compute Z-Score Standardization **Problem:** Standardize the values $X = \{2, 4, 4, 4, 5, 5, 7, 9\}$ using Z-Score Standardization.
Solution: Step 1: Calculate the mean (μ).

$$\mu = \frac{2 + 4 + 4 + 4 + 5 + 5 + 7 + 9}{8} = \frac{40}{8} = 5$$

Step 2: Calculate the standard deviation (σ). First compute the variance $(\sigma^2) = \frac{\sum (X-\mu)^2}{N}$

X	$(X - \mu)$	$(X - \mu)^2$
2	-3	9
4	-1	1
4	-1	1
4	-1	1
5	0	0
5	0	0
7	2	4
9	4	16

Sum of squares $= 9 + 1 + 1 + 1 + 0 + 0 + 4 + 16 = 32$.
Variance $\sigma^2 = \frac{32}{8} = 4$.

Standard deviation $\sigma = \sqrt{4} = 2$.

Step 3: Calculate the Z-scores using $Z = \frac{X-5}{2}$.

- For $X = 2$: $Z = \frac{2-5}{2} = -1.5$
- For $X = 4$: $Z = \frac{4-5}{2} = -0.5$
- For $X = 5$: $Z = \frac{5-5}{2} = 0.0$
- For $X = 7$: $Z = \frac{7-5}{2} = 1.0$
- For $X = 9$: $Z = \frac{9-5}{2} = 2.0$

Result: The standardized dataset is $\{-1.5, -0.5, -0.5, -0.5, 0.0, 0.0, 1.0, 2.0\}$.

2.2.3 Categorical Data Encoding

Machine learning algorithms require numerical input. Categorical variables (text labels) must be computationally converted into numerical formats.

Methodology:

One-Hot Encoding Procedure Used for nominal categorical data where there is no intrinsic mathematical ordering between categories.

1. Identify all unique categories in the feature column.
2. Create a new binary column (dummy variable) for each unique category.
3. For each row assign a 1 to the column corresponding to the original category and 0 to all other new columns.

Worked Example:

Apply One-Hot Encoding **Problem:** Convert the following "Color" feature into numerical format using One-Hot Encoding. Data: *Color* = [Red, Green, Blue, Red].
Solution: Step 1: Identify unique categories: {Red, Green, Blue}.
Step 2: Create three new columns: `Color_Red`, `Color_Green`, `Color_Blue`.
Step 3: Map the binary values for each row:

Original	`Color_Red`	`Color_Green`	`Color_Blue`
Red	1	0	0
Green	0	1	0
Blue	0	0	1
Red	1	0	0

Methodology:

Label Encoding Algorithm Used for ordinal categorical data where there is a clear logical ordering between categories.

1. Identify all unique categories in the column.
2. Determine the logical or hierarchical order of the categories.
3. Assign an integer to each category starting from 0 or 1 based on its sequential order.

Worked Example:

Apply Label Encoding **Problem:** Encode the following ordinal feature "Size" into numeric values using Label Encoding. Data: $Size = [\text{Small}, \text{Large}, \text{Medium}, \text{Small}, \text{Large}]$.
Solution: Step 1: Unique categories are $\{\text{Small}, \text{Medium}, \text{Large}\}$.
Step 2: The logical order is Small $<$ Medium $<$ Large.
Step 3: Assign integers based on hierarchy: Small $\rightarrow 0$, Medium $\rightarrow 1$, Large $\rightarrow 2$.
Step 4: Substitute the text with mapped integers.

Original Size	Encoded Value
Small	0
Large	2
Medium	1
Small	0
Large	2

Result: The encoded dataset sequence is $[0, 2, 1, 0, 2]$.

CHAPTER 3
Modeling and Evaluation

3.1 Selecting a Machine Learning Model

Selecting the right machine learning model involves comparing different algorithms on the training data using resampling techniques. A standard computational method to compare and validate model performance without a separate validation set is Cross Validation.

Methodology:

K-Fold Cross Validation K-Fold Cross Validation splits the dataset into K equal-sized folds to estimate the skill of a machine learning model on unseen data.

1. Shuffle the dataset randomly.
2. Split the dataset into K distinct folds (groups).
3. For each unique group:
   (a) Take the current group as a test data set.
   (b) Take the remaining $K-1$ groups as a training data set.
   (c) Fit a model on the training set and evaluate it on the test set.
   (d) Retain the evaluation score and discard the model.
4. Summarize the skill of the model using the average (and sometimes standard deviation) of the K evaluation scores.

Overall Accuracy $= \frac{1}{K}\sum_{i=1}^{K} \text{Accuracy}_i$

Worked Example:

Computing Cross Validation Accuracy Suppose we perform 5-fold cross-validation on a dataset to select a model. The model achieves the following accuracies on the 5 distinct validation folds: 82%, 85%, 78%, 88%, and 81%. Calculate the overall estimated accuracy of the model.

Step 1: Identify the scores and K

Here, $K = 5$. The scores are $S_1 = 82$, $S_2 = 85$, $S_3 = 78$, $S_4 = 88$, $S_5 = 81$.

Step 2: Apply the average formula

$$\text{Average Accuracy} = \frac{S_1 + S_2 + S_3 + S_4 + S_5}{5}$$

$$\text{Average Accuracy} = \frac{82 + 85 + 78 + 88 + 81}{5}$$

$$\text{Average Accuracy} = \frac{414}{5} = 82.8\%$$

The estimated cross-validation accuracy of the model is 82.8%.

3.2 Train the Model for Supervised Learning

Training a supervised learning model involves finding the underlying relationship between input features (X) and the target output (y). We cover the computational steps for two fundamental algorithms: Simple Linear Regression (for continuous targets) and K-Nearest Neighbors (for discrete targets).

Methodology:

Simple Linear Regression Formula Simple Linear Regression models the relationship between a single input variable x and an output variable y as a straight line: $y = mx + c$, where m is the slope and c is the y-intercept.
Given n data points (x_i, y_i), the parameters m and c that minimize the sum of squared errors are computed as:

1. Calculate the means of x and y: $\bar{x} = \frac{\sum x}{n}$, $\bar{y} = \frac{\sum y}{n}$
2. Compute the slope m:
$$m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
An alternative computational formula is:
$$m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$
3. Compute the intercept c:
$$c = \bar{y} - m\bar{x}$$

Worked Example:

Fitting a Linear Regression Model Train a linear regression model $y = mx + c$ on the following dataset:

x	y
1	2
2	4
3	5
4	4
5	5

Step 1: Compute sums required for the formula
We have $n = 5$. Let's create a computation table:

x	y	x^2	xy
1	2	1	2
2	4	4	8
3	5	9	15
4	4	16	16
5	5	25	25
$\sum x = 15$	$\sum y = 20$	$\sum x^2 = 55$	$\sum xy = 66$

Step 2: Calculate means
$$\bar{x} = \frac{15}{5} = 3, \quad \bar{y} = \frac{20}{5} = 4$$

Step 3: Calculate the slope m
Using the alternative formula:
$$m = \frac{5(66) - (15)(20)}{5(55) - (15)^2} = \frac{330 - 300}{275 - 225} = \frac{30}{50} = 0.6$$

Step 4: Calculate the intercept c

$$c = \bar{y} - m\bar{x} = 4 - (0.6)(3) = 4 - 1.8 = 2.2$$

The trained regression model is $y = 0.6x + 2.2$.

Methodology:

K-Nearest Neighbors Algorithm KNN is a supervised learning algorithm that classifies a new data point based on the majority class of its K nearest neighbors in the feature space.

1. Choose the number of neighbors K.
2. Calculate the Euclidean distance between the new point (x_q, y_q) and all points (x_i, y_i) in the training dataset:
$$d_i = \sqrt{(x_q - x_i)^2 + (y_q - y_i)^2}$$
3. Sort the calculated distances in ascending order.
4. Select the top K points from the sorted list.
5. For classification determine the most frequent class among these K neighbors. The new point is assigned to this majority class.

Worked Example:

Classifying a Point using KNN Given the following training dataset of 2D points and their classes (A or B):

X_1	X_2	Class
2	4	A
4	2	A
4	6	B
6	4	B

Classify a new point $P(4, 4)$ using KNN with $K = 3$.

Step 1: Calculate Euclidean distance to all points

Distance to $P_1(2, 4)$: $d_1 = \sqrt{(4-2)^2 + (4-4)^2} = \sqrt{4+0} = 2$

Distance to $P_2(4, 2)$: $d_2 = \sqrt{(4-4)^2 + (4-2)^2} = \sqrt{0+4} = 2$

Distance to $P_3(4, 6)$: $d_3 = \sqrt{(4-4)^2 + (4-6)^2} = \sqrt{0+4} = 2$

Distance to $P_4(6, 4)$: $d_4 = \sqrt{(4-6)^2 + (4-4)^2} = \sqrt{4+0} = 2$

Step 2: Find the 3 nearest neighbors

The distances are all equal to 2. We can select any 3 points, for instance, P_1, P_2, P_3. Their classes are: A, A, B.

Step 3: Determine majority class

Among the 3 neighbors, class A appears twice and class B appears once. The majority class is A. The point $P(4, 4)$ is classified as Class A.

3.3 Evaluate the Prepared Model

After a model is trained, its performance must be quantified using specific metrics. Classification models are typically evaluated using a Confusion Matrix, while regression models use error metrics.

Methodology:

Classification Evaluation Metrics A binary classification model's predictions can be summarized in a Confusion Matrix consisting of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).
Key computational metrics derived from the matrix:

1. **Accuracy**: Proportion of all correct predictions.

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

2. **Precision**: Proportion of correct positive predictions out of all predicted positives.

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

3. **Recall (Sensitivity)**: Proportion of correct positive predictions out of all actual positives.

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

4. **F1-Score**: Harmonic mean of Precision and Recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Worked Example:

Calculating Metrics from a Confusion Matrix A spam filter model predicts whether 100 emails are spam (Positive) or not spam (Negative). The results are:

- 40 emails correctly identified as spam (TP = 40)
- 50 emails correctly identified as not spam (TN = 50)
- 5 legitimate emails incorrectly marked as spam (FP = 5)
- 5 spam emails failed to be detected (FN = 5)

Calculate Accuracy, Precision, Recall, and F1-Score.
Step 1: Compute Accuracy

$$\text{Accuracy} = \frac{40 + 50}{40 + 50 + 5 + 5} = \frac{90}{100} = 0.90 \ (90\%)$$

Step 2: Compute Precision

$$\text{Precision} = \frac{40}{40 + 5} = \frac{40}{45} \approx 0.889 \ (88.9\%)$$

Step 3: Compute Recall

$$\text{Recall} = \frac{40}{40 + 5} = \frac{40}{45} \approx 0.889 \ (88.9\%)$$

Step 4: Compute F1-Score

$$\text{F1-Score} = 2 \times \frac{0.889 \times 0.889}{0.889 + 0.889} = 0.889 \ (88.9\%)$$

Methodology:

Regression Evaluation Metrics Regression models predict continuous values. We evaluate them by measuring the error between the predicted values ($\hat{y}_i$) and actual values (y_i) for n samples.

1. **Mean Absolute Error (MAE)**: Average of absolute differences.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

2. **Mean Squared Error (MSE)**: Average of squared differences. Punishes larger errors more severely.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

3. **Root Mean Squared Error (RMSE)**: Square root of MSE. Keeps the metric in the same unit as the target variable.

$$\text{RMSE} = \sqrt{\text{MSE}}$$

Worked Example:

Calculating Error Metrics for Regression A model predicts the prices of 4 houses. The actual prices (y) and predicted prices ($\hat{y}$) in thousands of dollars are:

House	Actual (y)	Predicted ($\hat{y}$)
1	100	110
2	150	140
3	200	210
4	250	240

Calculate MAE, MSE, and RMSE.

Step 1: Calculate individual errors

- $|100 - 110| = 10$, $(10)^2 = 100$
- $|150 - 140| = 10$, $(10)^2 = 100$
- $|200 - 210| = 10$, $(10)^2 = 100$
- $|250 - 240| = 10$, $(10)^2 = 100$

Step 2: Compute MAE

$$\text{MAE} = \frac{10 + 10 + 10 + 10}{4} = \frac{40}{4} = 10$$

Step 3: Compute MSE

$$\text{MSE} = \frac{100 + 100 + 100 + 100}{4} = \frac{400}{4} = 100$$

Step 4: Compute RMSE

$$\text{RMSE} = \sqrt{100} = 10$$

The MAE is \$10,000, MSE is 100, and RMSE is \$10,000.

CHAPTER 4

Supervised Learning: Classification and Regression

4.1 Describe Supervised Learning

Supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset. This means that each training example is paired with an output label. The goal of the algorithm is to learn a mapping from inputs to outputs, which can then be used to predict the output for new unseen inputs.

Supervised learning problems are primarily divided into two categories:

- **Classification:** The output variable is a category or discrete class (e.g. "Spam" or "Not Spam", "Disease" or "No Disease").
- **Regression:** The output variable is a continuous value or real number (e.g. predicting price, temperature, or weight).

4.2 Classification Algorithms

Classification algorithms are used to predict categorical labels. In this section, we cover computational methods for the K-Nearest Neighbors and Naive Bayes classifiers.

4.2.1 K-Nearest Neighbors (KNN)

The K-Nearest Neighbors algorithm is a simple distance-based classifier. It assigns a new data point to the most common class among its K closest neighbors in the feature space.

Methodology:

K-Nearest Neighbors Algorithm To classify a new data point using KNN:

1. Select the number of neighbors K to consider.
2. Calculate the distance between the new data point and all points in the training dataset. The Euclidean distance between two points $P(x_1, y_1)$ and $Q(x_2, y_2)$ is given by:

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

3. Sort the calculated distances in ascending order.
4. Select the top K points with the smallest distances.
5. Count the frequency of each class among these K neighbors.
6. Assign the new data point to the class with the highest frequency.

Worked Example:

KNN Classification Given the following training dataset with two classes (A and B):

Point	X coordinate	Y coordinate	Class
P_1	2	4	A
P_2	4	2	A
P_3	4	6	B
P_4	6	4	B

Predict the class of a new point $X_{new}(4, 4)$ using $K = 3$.
Solution:
Step 1: Calculate the distance from $X_{new}(4, 4)$ to all points.

- Distance to $P_1(2, 4)$: $d_1 = \sqrt{(4-2)^2 + (4-4)^2} = \sqrt{4+0} = 2$
- Distance to $P_2(4, 2)$: $d_2 = \sqrt{(4-4)^2 + (4-2)^2} = \sqrt{0+4} = 2$
- Distance to $P_3(4, 6)$: $d_3 = \sqrt{(4-4)^2 + (4-6)^2} = \sqrt{0+4} = 2$
- Distance to $P_4(6, 4)$: $d_4 = \sqrt{(4-6)^2 + (4-4)^2} = \sqrt{4+0} = 2$

Step 2: Find the 3 nearest neighbors. All distances are 2. We can select any 3 points, let's say P_1, P_2, P_3.
Step 3: Determine the majority class. The classes of the chosen neighbors are: $P_1 \to A$
$P_2 \to A$
$P_3 \to B$
Among the 3 nearest neighbors, Class A appears 2 times and Class B appears 1 time.
Conclusion: The predicted class for $X_{new}(4, 4)$ is **Class A**.

4.2.2 Naive Bayes Classifier

The Naive Bayes classifier is a probabilistic machine learning model based on Bayes' Theorem. It assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature (conditional independence).

Methodology:

Naive Bayes Classification Algorithm To classify a new instance with features $X = (x_1, x_2, \ldots, x_n)$:

1. For each class C_k, calculate the prior probability $P(C_k)$:

$$P(C_k) = \frac{\text{Number of instances in class } C_k}{\text{Total number of instances}}$$

2. Calculate the conditional probability (likelihood) of each feature value given the class $P(x_i|C_k)$:

$$P(x_i|C_k) = \frac{\text{Number of times } x_i \text{ appears in class } C_k}{\text{Total instances in class } C_k}$$

3. Calculate the unnormalized posterior probability for each class using the Naive Bayes assumption:

$$P(C_k|X) \propto P(C_k) \times P(x_1|C_k) \times P(x_2|C_k) \ldots P(x_n|C_k)$$

4. Predict the class C_k that has the maximum probability value.

Worked Example:

Naive Bayes Prediction Consider the following dataset predicting whether a person buys a computer based on their Age and Income:

ID	Age	Income	Buys Computer (Class)
1	Youth	High	No
2	Youth	High	No
3	Middle-aged	High	Yes
4	Senior	Medium	Yes
5	Senior	Low	Yes
6	Senior	Low	No
7	Middle-aged	Low	Yes
8	Youth	Medium	No
9	Youth	Low	Yes
10	Senior	Medium	Yes

Predict if a person will buy a computer given: Age = Youth and Income = Medium.
Solution:
Step 1: Calculate Prior Probabilities Total records = 10. Count(Buys = Yes) = 6 $\rightarrow$ $P(\text{Yes}) = 6/10 = 0.6$
Count(Buys = No) = 4 $\rightarrow$ $P(\text{No}) = 4/10 = 0.4$
Step 2: Calculate Likelihoods for given features For Age = Youth:
$P(\text{Youth}|\text{Yes}) = 1/6 \approx 0.167$ (Only ID 9 is Youth and Yes)
$P(\text{Youth}|\text{No}) = 3/4 = 0.75$ (ID 1, 2, 8 are Youth and No)
For Income = Medium:
$P(\text{Medium}|\text{Yes}) = 2/6 \approx 0.333$ (ID 4, 10)
$P(\text{Medium}|\text{No}) = 1/4 = 0.25$ (ID 8)
Step 3: Calculate Posterior Probabilities (Unnormalized) For Class = Yes:

$$P(\text{Yes}|X) \propto P(\text{Yes}) \times P(\text{Youth}|\text{Yes}) \times P(\text{Medium}|\text{Yes})$$

$$P(\text{Yes}|X) \propto 0.6 \times \left(\frac{1}{6}\right) \times \left(\frac{2}{6}\right) = 0.6 \times 0.1667 \times 0.3333 = 0.0333$$

For Class = No:

$$P(\text{No}|X) \propto P(\text{No}) \times P(\text{Youth}|\text{No}) \times P(\text{Medium}|\text{No})$$

$$P(\text{No}|X) \propto 0.4 \times 0.75 \times 0.25 = 0.075$$

Step 4: Make Prediction Since $0.075 > 0.0333$, the probability of "No" is higher.
Conclusion: The model predicts that a Youth with Medium income will **Not** buy a computer.

4.3 Regression

Regression is a supervised learning technique used to predict continuous numerical values. The most common form is linear regression, which assumes a linear relationship between the input variables (features) and the single output variable (target).

4.3.1 Simple Linear Regression

Simple linear regression uses a single independent variable X to predict a dependent variable Y by fitting the best straight line (the regression line) through the data points.

Methodology:

Simple Linear Regression Formula The equation of the simple linear regression line is:

$$Y = mX + c$$

Where Y is the predicted value, X is the input feature, m is the slope, and c is the y-intercept. The formulas for the slope m and intercept c using the Least Squares Method are:

1. Calculate the slope m:
$$m = \frac{n(\sum XY) - (\sum X)(\sum Y)}{n(\sum X^2) - (\sum X)^2}$$
2. Calculate the y-intercept c:
$$c = \frac{\sum Y - m(\sum X)}{n}$$

Here, n is the total number of data points.

Worked Example:

Linear Regression Calculation Given the following data of study hours (X) and test scores (Y):

Student	1	2	3	4
Hours (X)	1	2	3	4
Score (Y)	2	4	5	4

Find the linear regression equation and predict the score for a student who studies for 5 hours.
Solution:
Step 1: Create a calculation table Here $n = 4$.

X	Y	X^2	XY
1	2	1	2
2	4	4	8
3	5	9	15
4	4	16	16
$\sum X = 10$	$\sum Y = 15$	$\sum X^2 = 30$	$\sum XY = 41$

Step 2: Calculate the slope m

$$m = \frac{n(\sum XY) - (\sum X)(\sum Y)}{n(\sum X^2) - (\sum X)^2}$$

$$m = \frac{4(41) - (10)(15)}{4(30) - (10)^2}$$

$$m = \frac{164 - 150}{120 - 100} = \frac{14}{20} = 0.7$$

Step 3: Calculate the y-intercept c

$$c = \frac{\sum Y - m(\sum X)}{n}$$

$$c = \frac{15 - 0.7(10)}{4} = \frac{15 - 7}{4} = \frac{8}{4} = 2$$

Step 4: Formulate the equation and predict The regression equation is:

$$Y = 0.7X + 2$$

To predict the score for $X = 5$:

$$Y = 0.7(5) + 2 = 3.5 + 2 = 5.5$$

Conclusion: A student who studies for 5 hours is predicted to score 5.5.

CHAPTER 5
Unsupervised Learning

5.1 Explain Unsupervised Learning

Unsupervised learning is a class of machine learning techniques where the model is trained on a dataset without predefined labels or target outputs. The system attempts to learn the underlying structure, patterns, or distribution of the data on its own.

The two primary computational tasks in unsupervised learning are:

1. **Clustering**: Grouping data points so that points in the same group (cluster) are more similar to each other than to those in other groups.
2. **Association Rule Learning**: Discovering interesting relations or frequent patterns between variables in large databases.

5.2 Describe Clustering

Clustering organizes data based on similarity metrics (e.g. Euclidean distance or Manhattan distance). The most common partitional algorithm is K-Means, and the most common hierarchical approach is Agglomerative Clustering.

Methodology:

K-Means Clustering Algorithm K-Means groups data into K distinct non-overlapping clusters based on distance to the centroid.

1. **Initialization**: Choose the number of clusters K. Randomly initialize K centroids $\mu_1, \mu_2, \ldots, \mu_K$.
2. **Assignment Step**: Assign each data point x_i to the nearest centroid. The distance is typically Euclidean:
$$d(x_i, \mu_j) = \sqrt{\sum (x_i - \mu_j)^2}$$
Point x_i is assigned to cluster C_j if $d(x_i, \mu_j)$ is minimized.
3. **Update Step**: Recalculate the centroids as the mean of all data points assigned to that cluster:
$$\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$$
4. **Convergence**: Repeat the Assignment and Update steps until the centroids no longer change or a maximum number of iterations is reached.

Worked Example:

1D Data Clustering using KMeans Given a 1D dataset $X = \{2, 4, 10, 12, 3, 20, 30, 11, 25\}$. Perform K-Means clustering with $K = 2$ and initial centroids $\mu_1 = 2$ and $\mu_2 = 4$.

Iteration 1: Assignment Calculate the distance (absolute difference for 1D) from each point to $\mu_1 = 2$ and $\mu_2 = 4$.

Data Point (x)	Dist to $\mu_1 = 2$	Dist to $\mu_2 = 4$	Cluster Assigned
2	0	2	C_1
4	2	0	C_2
10	8	6	C_2
12	10	8	C_2
3	1	1	C_1 (Tie-break to C_1)
20	18	16	C_2
30	28	26	C_2
11	9	7	C_2
25	23	21	C_2

Clusters after Iteration 1: $C_1 = \{2, 3\}$, $C_2 = \{4, 10, 12, 20, 30, 11, 25\}$.

Iteration 1: Update Centroids

$$\mu_1 = \frac{2+3}{2} = 2.5$$

$$\mu_2 = \frac{4+10+12+20+30+11+25}{7} = \frac{112}{7} = 16$$

Iteration 2: Assignment Calculate distances to new centroids $\mu_1 = 2.5$ and $\mu_2 = 16$.

Data Point (x)	Dist to $\mu_1 = 2.5$	Dist to $\mu_2 = 16$	Cluster Assigned
2	0.5	14	C_1
4	1.5	12	C_1
10	7.5	6	C_2
12	9.5	4	C_2
3	0.5	13	C_1
20	17.5	4	C_2
30	27.5	14	C_2
11	8.5	5	C_2
25	22.5	9	C_2

Clusters after Iteration 2: $C_1 = \{2, 3, 4\}$, $C_2 = \{10, 11, 12, 20, 25, 30\}$.

Iteration 2: Update Centroids

$$\mu_1 = \frac{2+3+4}{3} = 3$$

$$\mu_2 = \frac{10+11+12+20+25+30}{6} = \frac{108}{6} = 18$$

(Further iterations would follow the same process until centroids stop changing.)

Methodology:

Agglomerative Hierarchical Clustering Algorithm Agglomerative clustering is a "bottom-up" approach.

1. **Initialization**: Treat each data point as its own individual cluster. Calculate the proximity (distance) matrix between all clusters.
2. **Merge Step**: Find the two clusters with the minimum distance between them and merge them into a single cluster.
3. **Update Matrix**: Recalculate the distances between the new cluster and all remaining clusters using a linkage criterion:
   - *Single Linkage*: Minimum distance between points in the two clusters.
   - *Complete Linkage*: Maximum distance between points in the two clusters.
   - *Average Linkage*: Average distance between all points in the two clusters.

4. **Termination**: Repeat Steps 2 and 3 until all data points belong to a single overarching cluster.

5.3 Describe pattern finding using association rule

Association rule learning finds rules of the form $X \Rightarrow Y$, meaning if itemset X occurs in a transaction, itemset Y is also likely to occur. The standard method is the Apriori Algorithm.

Methodology:

Apriori Algorithm for Association Rules **Key Definitions:**

- **Support (S)**: The percentage of transactions that contain a given itemset.

$$Support(X) = \frac{\text{Frequency of } X}{\text{Total Transactions}}$$

- **Confidence (C)**: The probability of finding item Y in a transaction given that it contains item X.

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

Algorithm Steps:

1. Set a minimum support threshold (min_sup) and minimum confidence threshold (min_conf).
2. **Generate L_1**: Count the frequency of each single item. Keep those with Support $\geq$ min_sup.
3. **Join Step (Generate C_k)**: Combine itemsets in L_{k-1} to form candidate k-itemsets (C_k).
4. **Prune Step (Generate L_k)**: Count the frequency of each itemset in C_k. Keep those with Support $\geq$ min_sup to form L_k.
5. Repeat Steps 3 and 4 until no more frequent itemsets can be generated.
6. **Rule Generation**: For each frequent itemset I, generate all non-empty subsets s. Output the rule $s \Rightarrow (I - s)$ if $Confidence \geq$ min_conf.

Worked Example:

Finding Frequent Itemsets and Rules using Apriori **Problem:** Given the following 5 transactions, use Apriori to find all frequent itemsets and association rules. Minimum Support count = 2 (or 40%), Minimum Confidence = 70%.

Transaction ID	Items Bought
T1	Apple, Beer, Milk
T2	Apple, Beer
T3	Apple, Diaper
T4	Apple, Beer, Milk
T5	Beer, Milk

Step 1: Generate C_1 and L_1 (1-itemsets) Count the frequency of each item.

Item	Count	Keep? (Count ≥ 2)
Apple	4	Yes
Beer	4	Yes
Milk	3	Yes
Diaper	1	No (Pruned)

$L_1 = \{\{Apple\}, \{Beer\}, \{Milk\}\}$
Step 2: Generate C_2 and L_2 (2-itemsets) Join elements of L_1 to create pairs. Count frequencies in original transactions.

Itemset	Count	Keep? (Count ≥ 2)
{Apple, Beer}	3 (T1, T2, T4)	Yes
{Apple, Milk}	2 (T1, T4)	Yes
{Beer, Milk}	3 (T1, T4, T5)	Yes

$L_2 = \{\{Apple, Beer\}, \{Apple, Milk\}, \{Beer, Milk\}\}$
Step 3: Generate C_3 and L_3 (3-itemsets) Join L_2 itemsets. The only possible 3-itemset is $\{Apple, Beer, Milk\}$.

Itemset	Count	Keep? (Count ≥ 2)
{Apple, Beer, Milk}	2 (T1, T4)	Yes

$L_3 = \{\{Apple, Beer, Milk\}\}$. No further itemsets can be generated.
Step 4: Generate Rules from Frequent Itemsets Let's extract rules from the 3-itemset $I = \{Apple, Beer, Milk\}$ and check if Confidence $\geq 70\%$.
Total Count$(I) = 2$.

- Rule 1: $\{Apple, Beer\} \Rightarrow \{Milk\}$. Conf = Count(Apple, Beer, Milk) / Count(Apple, Beer) = $2/3 = 66.6\%$. (Reject)
- Rule 2: $\{Apple, Milk\} \Rightarrow \{Beer\}$. Conf = Count(Apple, Beer, Milk) / Count(Apple, Milk) = $2/2 = 100\%$. (Accept)
- Rule 3: $\{Beer, Milk\} \Rightarrow \{Apple\}$. Conf = Count(Apple, Beer, Milk) / Count(Beer, Milk) = $2/3 = 66.6\%$. (Reject)
- Rule 4: $\{Milk\} \Rightarrow \{Apple, Beer\}$. Conf = Count(Apple, Beer, Milk) / Count(Milk) = $2/3 = 66.6\%$. (Reject)
- Rule 5: $\{Apple\} \Rightarrow \{Beer, Milk\}$. Conf = Count(Apple, Beer, Milk) / Count(Apple) = $2/4 = 50\%$. (Reject)
- Rule 6: $\{Beer\} \Rightarrow \{Apple, Milk\}$. Conf = Count(Apple, Beer, Milk) / Count(Beer) = $2/4 = 50\%$. (Reject)

Final Strong Association Rules: $\{Apple, Milk\} \Rightarrow \{Beer\}$ (Confidence = 100%)

CHAPTER 6

Python Libraries for Machine Learning

6.1 Explain the use of existing machine learning python libraries

Python provides a powerful ecosystem of libraries for building and deploying machine learning models. The most critical libraries for computational workflows include **NumPy** for numerical operations, **Pandas** for data manipulation, and **Scikit-Learn** for building and evaluating algorithms.

Methodology:

Data Preprocessing with Pandas and NumPy Data preprocessing is the foundational step in any machine learning workflow. Libraries like Pandas and NumPy provide built-in functions to computationally clean and scale data matrices.

1. **Data Loading:** Read the dataset into a tabular structure using Pandas functions such as `pd.read_csv()`.
2. **Handling Missing Values:** Identify missing data and impute it using statistical measures like the mean or median via `df.fillna(df.mean())`.
3. **Feature Scaling:** Standardize or normalize features using NumPy operations or Scikit-Learn's `StandardScaler`. The internal standard score computation is:

$$z = \frac{x - \mu}{\sigma}$$

where μ is the mean of the feature and σ is the standard deviation.

4. **Data Conversion:** Convert Pandas DataFrames into NumPy arrays for mathematical processing using the `.to_numpy()` method.

Worked Example:

Imputing Missing Values and Scaling Data **Problem:** Given a small dataset with missing values, use the computational steps equivalent to Pandas and NumPy functions to impute the missing value with the mean and standardize the dataset.
Data feature X: [10, NaN, 30, 40, 20]
Step 1: Compute the mean of available data (Pandas `mean` equivalent) The available values are 10, 30, 40, 20.

$$\text{Mean } (\mu) = \frac{10 + 30 + 40 + 20}{4} = \frac{100}{4} = 25$$

Step 2: Impute the missing value (Pandas `fillna` equivalent) Replace the NaN value with the computed mean (25).

$$X_{\text{imputed}} = [10, 25, 30, 40, 20]$$

Step 3: Compute Standard Deviation for Scaling (NumPy `std` equivalent) Calculate the variance and standard deviation of X_{imputed}.

$$\text{Variance} = \frac{(10-25)^2 + (25-25)^2 + (30-25)^2 + (40-25)^2 + (20-25)^2}{5}$$

$$\text{Variance} = \frac{(-15)^2 + 0^2 + 5^2 + 15^2 + (-5)^2}{5} = \frac{225 + 0 + 25 + 225 + 25}{5} = \frac{500}{5} = 100$$

$$\text{Standard Deviation } (\sigma) = \sqrt{100} = 10$$

Step 4: Standardize the dataset (Scikit-Learn `StandardScaler` equivalent) Apply the formula $z = \frac{x-\mu}{\sigma}$ to each element:

- $x_1 = (10 - 25)/10 = -1.5$
- $x_2 = (25 - 25)/10 = 0.0$
- $x_3 = (30 - 25)/10 = 0.5$
- $x_4 = (40 - 25)/10 = 1.5$
- $x_5 = (20 - 25)/10 = -0.5$

Final Standardized Array: [-1.5, 0.0, 0.5, 1.5, -0.5]

Methodology:

Model Training Pipeline with Scikit Learn Scikit-Learn provides a uniform interface for training diverse machine learning algorithms. The computational pipeline follows a strict sequence of method calls applied to data matrices.

1. **Data Splitting:** Use `train_test_split(X, y)` to divide the dataset into training matrices and testing matrices.
2. **Model Instantiation:** Create an instance of the chosen algorithm object e.g. `model = LinearRegression()` or `model = KNeighborsClassifier(n_neighbors=3)`.
3. **Model Fitting:** Call the `model.fit(X_train, y_train)` method. This step executes the internal mathematical optimization algorithm (like Gradient Descent or Ordinary Least Squares) to learn the best model parameters.
4. **Prediction:** Call the `model.predict(X_test)` method to generate output labels or continuous values for unseen data using the learned parameters.

Worked Example:

Executing a Simple Linear Regression Pipeline **Problem:** Trace the computational steps of Scikit-Learn's `LinearRegression().fit()` and `predict()` methods for a 1D dataset to predict the value for $X_{\text{test}} = 5$. Training Data: $X_{\text{train}} = [1, 2, 3]$, $y_{\text{train}} = [3, 5, 7]$

Step 1: Model Instantiation and Fitting (`fit(X_train y_train)`) The `fit` method computes the Ordinary Least Squares parameters for $y = mx + c$. Mean of X $(\bar{x}) = 2$. Mean of y $(\bar{y}) = 5$.
Calculate the learned slope/coefficient m:

$$m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

$$m = \frac{(1-2)(3-5) + (2-2)(5-5) + (3-2)(7-5)}{(1-2)^2 + (2-2)^2 + (3-2)^2}$$

$$m = \frac{(-1)(-2) + (0)(0) + (1)(2)}{(-1)^2 + 0^2 + 1^2} = \frac{2 + 0 + 2}{1 + 0 + 1} = \frac{4}{2} = 2$$

Calculate the learned intercept c:

$$c = \bar{y} - m\bar{x} = 5 - (2 \times 2) = 5 - 4 = 1$$

The parameters stored internally by Scikit-Learn are `model.coef_` $= 2$ and `model.intercept_` $= 1$.
Step 2: Prediction (`predict(X_test)`) The user calls `predict([5])`. The model applies the learned equation $y = 2x + 1$.

$$y_{\text{pred}} = 2(5) + 1 = 10 + 1 = 11$$

Final Output: The `predict` method returns an array `[11]`.

Methodology:

Evaluating Model Performance with Scikit Learn Metrics After generating predictions, the `sklearn.metrics` module is used to computationally evaluate the model against the ground truth.

1. **Classification Accuracy:** Evaluates the proportion of correctly predicted labels.

   $$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

   Python implementation: `accuracy_score(y_true, y_pred)`

2. **Mean Squared Error MSE:** Evaluates regression models by computing the average of the squares of the errors.

   $$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

   Python implementation: `mean_squared_error(y_true, y_pred)`

3. **Confusion Matrix:** Generates an $N \times N$ matrix showing True Positives, True Negatives, False Positives, and False Negatives. Python implementation: `confusion_matrix(y_true, y_pred)`

Worked Example:

Computing Classification Metrics Programmatically **Problem:** Given the true labels and the model's predicted labels, simulate the internal computation of Scikit-Learn's `accuracy_score` and `confusion_matrix`.
True Labels (`y_true`): [1, 0, 1, 1, 0, 1]
Predicted Labels (`y_pred`): [1, 0, 0, 1, 0, 1]
Step 1: Compute Accuracy Score Compare elements index by index:

- Index 0: 1 == 1 (Correct)
- Index 1: 0 == 0 (Correct)
- Index 2: 1 != 0 (Incorrect)
- Index 3: 1 == 1 (Correct)
- Index 4: 0 == 0 (Correct)
- Index 5: 1 == 1 (Correct)

Total correct = 5. Total instances = 6.

$$\text{Accuracy} = \frac{5}{6} \approx 0.8333 \text{ or } 83.33\%$$

Step 2: Construct the Confusion Matrix Determine the counts for each combination of Actual and Predicted classes:

- **True Negatives (Actual 0 Predicted 0):** Indices 1 and 4. (Count = 2)
- **False Positives (Actual 0 Predicted 1):** None. (Count = 0)
- **False Negatives (Actual 1 Predicted 0):** Index 2. (Count = 1)
- **True Positives (Actual 1 Predicted 1):** Indices 0, 3, and 5. (Count = 3)

Resulting Confusion Matrix Array:

	Predicted 0	**Predicted 1**
Actual 0	2	0
Actual 1	1	3

The function returns the 2D NumPy array: `[[2, 0], [1, 3]]`.

Appendix: Key Formulae

This appendix provides a quick reference to the key formulae and mathematical relationships discussed in this book, categorized by their respective units.

Unit 1: Fundamentals of Machine Learning

Euclidean Distance

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Explanation: Calculates the straight-line distance between two points x and y in an n-dimensional space.

Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Explanation: Measures the average of the squares of the errors, where y_i is the actual value and $\hat{y}_i$ is the predicted value over n samples.

Accuracy

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \times 100\%$$

Explanation: Represents the proportion of correctly predicted instances out of the total instances.

Unit 2: Data Preprocessing and Exploration

Mean (μ or $\bar{x}$)

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Explanation: The arithmetic average of all values x_i in a dataset of size n.

Min-Max Scaling (Normalization)

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

Explanation: Rescales data features X to a fixed range, typically between 0 and 1, where X_{min} and X_{max} are the minimum and maximum values in the feature.

Z-Score Standardization

$$Z = \frac{X - \mu}{\sigma}$$

Explanation: Standardizes a feature X by removing the mean μ and scaling to unit variance, where σ is the standard deviation.

Unit 3 & 4: Supervised Learning (Regression and Classification)

Simple Linear Regression

$$y = mx + c$$

Explanation: Models the relationship between a dependent variable y and an independent variable x, where m is the slope and c is the y-intercept.

Slope (m) in Simple Linear Regression

$$m = \frac{n(\sum XY) - (\sum X)(\sum Y)}{n(\sum X^2) - (\sum X)^2} \quad \text{or} \quad m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

Explanation: Calculates the slope of the best-fit line.

Y-intercept (c) in Simple Linear Regression

$$c = \frac{\sum Y - m(\sum X)}{n} \quad \text{or} \quad c = \bar{y} - m\bar{x}$$

Explanation: Calculates the point where the regression line crosses the y-axis.

K-Fold Cross-Validation Accuracy

$$\text{CV Accuracy} = \frac{1}{K}\sum_{i=1}^{K} \text{Accuracy}_i$$

Explanation: The average model accuracy computed across K different folds of the data.

Naive Bayes: Prior Probability

$$P(C_k) = \frac{\text{Number of instances in class } C_k}{\text{Total number of instances}}$$

Explanation: The initial probability of an instance belonging to class C_k before observing any features.

Naive Bayes: Conditional Probability

$$P(x_i|C_k) = \frac{\text{Number of times } x_i \text{ appears in class } C_k}{\text{Total instances in class } C_k}$$

Explanation: The likelihood of feature x_i occurring given that the class is C_k.

Naive Bayes Classifier Rule

$$P(C_k|X) \propto P(C_k) \times P(x_1|C_k) \times P(x_2|C_k) \dots P(x_n|C_k)$$

Explanation: The posterior probability of class C_k given the feature vector X, assuming features are conditionally independent.

Unit 5: Unsupervised Learning (Clustering and Association)

K-Means: Distance to Centroid

$$d(x_i, \mu_j) = \sqrt{\sum (x_i - \mu_j)^2}$$

Explanation: Calculates the Euclidean distance between a data point x_i and the cluster centroid μ_j.

K-Means: Centroid Update Rule

$$\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$$

Explanation: Computes the new centroid μ_j for cluster C_j as the mean of all data points assigned to that cluster.

Association Rules: Support

$$\text{Support}(X) = \frac{\text{Frequency of } X}{\text{Total Transactions}}$$

Explanation: The proportion of transactions in the dataset that contain the itemset X.

Association Rules: Confidence

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

Explanation: The probability of seeing itemset Y in a transaction given that it also contains itemset X.