UNIT-1: ANALYTICS AND TYPES OF ANALYTICS

Analytics:

Analytics is the process of converting raw data into meaningful insights to support effective decision-making. It involves examining data systematically to identify patterns, trends, and relationships. Organizations use analytics to make data-driven decisions instead of relying on assumptions or guesswork. By analyzing data, businesses can improve performance, reduce risks, increase profits, and enhance customer satisfaction.

- Helps in understanding past performance
- Identifies patterns and trends
- Supports better decision-making

TYPES OF ANALYTICS

1.Descriptive Analytics

Descriptive Analytics answers the question: **"What happened?"**
It focuses on summarizing historical data to understand past performance.

- Describes past events
- Uses historical data
- Provides reports and dashboards
- Uses basic statistical measures

Example

- Monthly sales reports
- Average marks in the last semester
- Website traffic summary

2. Diagnostic Analytics

Diagnostic Analytics answers the question: **"Why did it happen?"**
It identifies the root causes behind past events.

- Describes past events
- Uses historical data
- Provides reports and dashboards

- Uses basic statistical measures

Example

- Sales dropped due to increased prices
- Students scored low marks because the exam was difficult
- Customer churn increased due to poor service

3.Predictive Analytics

Predictive Analytics answers the question: **"What will happen in the future?"** It uses historical data and statistical models to forecast future outcomes.

- Uses past data to predict future trends
- Applies statistical models and machine learning
- Estimates probabilities of future events

Example

- Predicting next month's sales
- Forecasting customer churn
- Recommending products in online shopping
- Predicting stock market trends

4. Prescriptive Analytics

Prescriptive Analytics answers the question: **"What should we do?"** It suggests actions to achieve desired outcomes.

- Recommends decision options
- Suggests best course of action
- Uses optimization and simulation

Example

- Google Maps suggesting the fastest route
- Businesses recommending discount strategies
- Hospitals optimizing patient scheduling

Difference Between Business Intelligence (BI) and Predictive Analytics

S.No	Business Intelligence (BI)	Predictive Analytics (PA)
1	Focuses on understanding what happened using historical data.	Focuses on what is likely to happen in the future using past data and models.
2	Provides dashboards, reports, and summaries of past performance.	Provides future forecasts, predictions, and scores.
3	Uses descriptive and diagnostic analytics techniques.	Uses advanced statistical modeling and machine learning techniques.
4	Mainly rule-based and visual-based.	Mainly model-based.
5	Helps managers monitor business performance.	Helps managers make future decisions.
6	Answers the question: "What happened?"	Answers the question: "What will happen next?"
7	Example: Shows last month sales.	Example: Predicts next month sales.

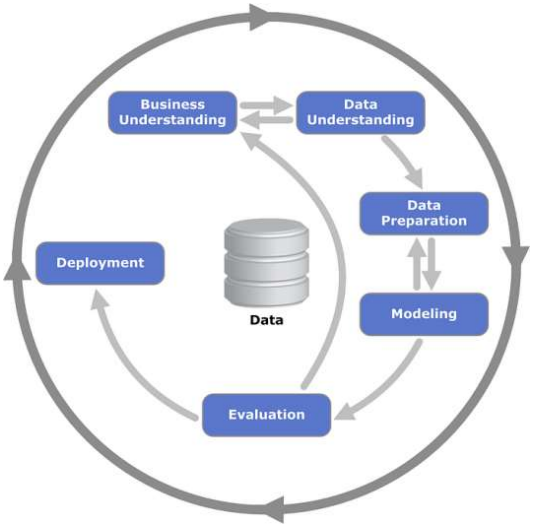
Difference Between Statistics and Predictive Analytics

S.No	Statistics	Predictive Analytics
1	Statistics is the study of collecting, organizing, analyzing, and interpreting data.	Predictive Analytics is the process of using historical data to predict future outcomes.
2	Focuses mainly on describing and analyzing past or existing data.	Focuses mainly on forecasting future events and trends.
3	Uses methods such as mean, median, standard deviation, correlation, and hypothesis testing.	Uses techniques such as regression, classification, time series analysis, and machine learning algorithms.
4	Helps in understanding patterns and relationships in data.	Helps in estimating probabilities and future risks.
5	Primarily used for data analysis and research.	Primarily used for decision-making and future planning.
6	Answers questions like: "What is happening?" or "Why did it happen?"	Answers the question: "What is likely to happen next?"
7	Example: Calculating average marks of students.	Example: Predicting students' future performance.

Difference Between Data Mining and Predictive Analytics

S.No	Data Mining	Predictive Analytics
1	Data Mining is the process of discovering patterns, relationships, and useful information from large datasets.	Predictive Analytics is the process of using historical data to predict future outcomes.
2	Focuses on exploring and extracting hidden patterns from data.	Focuses on forecasting future events using models.
3	Mainly deals with pattern discovery and knowledge extraction.	Mainly deals with prediction and probability estimation.
4	Uses techniques such as clustering, association rules, classification, and anomaly detection.	Uses techniques such as regression, classification models, time series analysis, and machine learning algorithms.
5	Answers questions like: "What patterns exist in the data?"	Answers questions like: "What is likely to happen next?"
6	It is an exploratory process.	It is a future-oriented decision-making process.
7	Example: Finding frequently purchased product combinations in a supermarket.	Example: Predicting which customers are likely to buy a product next month.

CRISP–DM (Cross Industry Standard Process for Data Mining)



CRISP–DM stands for **Cross Industry Standard Process for Data Mining**. It is a structured and widely used methodology for planning and executing data mining and predictive analytics projects. It provides a systematic approach that guides analysts from understanding the business problem to deploying the final solution.

CRISP–DM is industry-independent and can be applied across various domains such as healthcare, finance, marketing, education, and manufacturing.

Phases of CRISP–DM

CRISP–DM consists of **six major phases**:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

Each phase is interconnected and iterative in nature.

1.Business Understanding

This is the first phase of CRISP–DM. It focuses on understanding the business objectives and defining the problem clearly.

- Identify business goals
- Define project objectives
- Convert business problem into data mining problem
- Prepare project plan

Example

If a company wants to reduce customer churn, the business understanding phase defines the objective as predicting which customers are likely to leave.

2. Data Understanding

This phase involves collecting and exploring the data to understand its structure and quality.

- Collect initial data
- Describe data characteristics
- Explore data using statistics and visualization

- Identify data quality issues

Example

Analyzing customer data to check missing values, incorrect entries, or unusual patterns.

3.Data Preparation

Data preparation involves cleaning and transforming raw data into a suitable format for modeling.

- Data cleaning
- Handling missing values
- Removing duplicates
- Data transformation
- Feature selection

4.Modeling

In this phase, various modeling techniques are applied to the prepared data to build predictive models.

- Select appropriate algorithms
- Train models
- Tune parameters
- Compare model performance

5.Evaluation

Evaluation ensures that the model meets business objectives and performs effectively.

- Validate model accuracy
- Compare results with business goals
- Check for errors or bias
- Decide whether model is ready for deployment

6.Deployment

Deployment is the final phase where the model is implemented in a real-world environment.

- Deploy model into system
- Generate reports

- Monitor performance
- Maintain and update model

Example:

Implementing a customer churn prediction model into a company's CRM system.

Defining Data for Predictive Modeling

Defining data is a critical step in the Business Understanding stage of predictive modeling. Before building any model, the data must be structured properly so that predictive algorithms can process it effectively.

Predictive modeling requires data to be organized in a two-dimensional rectangular format, consisting of rows and columns.

1. Structure of Data: Two-Dimensional Format

Predictive modeling data must be:

- Rectangular
- Two-dimensional
- Organized into rows and columns

Rows

Each row represents a Unit of Analysis.

Examples:

- Customer (Customer Analytics)
- Transaction (Fraud Detection)
- Call (Call Center Analytics)
- Survey Response (Survey Analysis)

The unit of analysis is problem-specific and must be clearly defined during the Business Understanding stage.

Columns

Columns are also called:

- Attributes
- Variables
- Fields

- Features
- Descriptors

Each column represents a measure describing the unit of analysis

- All records must have the same number of columns
- Columns must appear in the same order
- Column meanings must be consistent across all records

Data can be loaded in multiple formats:

- Delimited flat files (.csv, tab-delimited)
- Fixed-width flat files
- Binary files (SPSS .sav, SAS .sas7bdat, Matlab .mat)
- Database tables and views (via ODBC/native drivers)

Advantages of database connectivity:

- No need to export large files
- Better version control
- More flexibility in data extraction

Data Leakage :

Data leakage occurs when:

- Future information is used in building the model
- Information not available at prediction time is included

Example:
If phone numbers are collected only after a customer converts, and phone number is used to predict conversion, this leaks future information.

Data leakage leads to:

- Artificially high accuracy
- Model failure in real deployment

Defining Target Variable

Defining the target variable is a crucial step in predictive modeling. The target variable, also known as the dependent variable or response variable, is the outcome that the predictive model aims to predict. Proper definition of the target variable ensures that the model aligns with the business objective and produces meaningful results.

The target variable represents the final output of the predictive model. It is the variable whose value is unknown in the future and needs to be predicted using historical data.

In predictive analytics, the target variable must be:

- Clearly defined
- Measurable
- Relevant to the business objective
- Properly structured

Importance of Defining Target Variable

Defining the target variable correctly is important because:

- It determines the type of predictive model to be used.
- It guides the selection of predictor variables.
- It influences data preparation and modeling techniques.
- Incorrect target definition leads to inaccurate predictions.

A well-defined target variable improves model accuracy and reliability.

Types of Target Variables

1.Binary target variable

A binary target variable has only **two possible outcomes**. It is the most common type used in predictive analytics.

It is usually represented as:

- 1 and 0
- Yes and No
- True and False

Examples:

- Customer churn (Yes / No)
- Fraud detection (Fraud / Not Fraud)
- Loan approval (Approved / Rejected)
- Student result (Pass / Fail)

2. Categorical Target Variable

A categorical target variable has distinct classes or categories.

Examples:

- Customer churn (Yes/No)
- Fraud detection (Fraud/Not Fraud)
- Student result (Pass/Fail)

For such targets, classification models are used.

3. Continuous Target Variable

A continuous target variable has numerical values.

Examples:

- Sales amount
- House price
- Customer lifetime value

For such targets, regression models are used.

Steps in Defining Target Variable

1. Clearly understand the business problem.
2. Identify what exactly needs to be predicted.
3. Ensure the target variable is measurable using available data.
4. Check data availability and quality for the target.
5. Confirm that the target reflects the real-world objective.

Example:

- If a company wants to reduce customer churn, the target variable would be:

  Churn Status (1 = Yes, 0 = No)
- If a business wants to predict next month's revenue, the target variable would be:

  Total Sales Amount (numerical value)

Lapsed Donor in Predictive Analytics-case study

Introduction

A lapsed donor refers to a person who has previously donated to an organization but has stopped donating for a certain period of time. In predictive analytics, identifying lapsed donors is an important application, especially in non-profit and fundraising organizations. Predictive models help organizations analyze donor behavior and take actions to re-engage inactive donors.

Lapsed Donor:

A lapsed donor is defined as: A donor who has not made a donation within a specified time frame, even though they donated in the past.

The time frame may vary depending on the organization. For example:

- No donation in the last 6 months
- No donation in the last 1 year
- No donation within a defined campaign cycle

The definition must be clearly established before building a predictive model.

Importance of Identifying Lapsed Donors

Identifying lapsed donors is important because:

- Retaining existing donors is cheaper than acquiring new donors.
- Organizations can target them with personalized campaigns.
- It improves fundraising efficiency.
- It increases overall donation revenue.

Predictive analytics helps in understanding which donors are likely to lapse and which lapsed donors can be reactivated.

Target Variable in Lapsed Donor Model

In predictive modeling, lapsed donor is usually treated as a **binary target variable**:

- 1 = Lapsed
- 0 = Active

This means the model predicts whether a donor is likely to lapse or remain active.

Predictor Variables Used

To predict lapsed donors, various predictor variables may be used, such as:

- Donation frequency
- Recency of last donation
- Total donation amount
- Demographic details
- Campaign response history

These factors help in understanding donor behavior.

Predictive Modeling for Lapsed Donors

The predictive modeling process typically includes:

1. Defining the target variable (Lapsed / Not Lapsed)
2. Collecting historical donor data
3. Preparing and cleaning data
4. Applying classification models
5. Evaluating model accuracy

The model assigns a probability score indicating the likelihood of a donor becoming lapsed.

Example:

Suppose a donor has not donated in the last 12 months. Based on historical data, the predictive model estimates:

- 75% probability of being a lapsed donor

The organization can then send targeted emails or promotional offers to re-engage the donor.

FRAUD DETECTION – SHORT CASE STUDY

Fraud detection is a major application of predictive analytics used by banks, insurance companies, and e-commerce platforms to identify fraudulent transactions. Since only a small percentage of transactions are fraudulent, manually checking every transaction is costly and inefficient. Therefore, predictive models are developed to estimate the probability of fraud.

In this case study, the unit of analysis is a single transaction. Each record in the dataset represents one transaction with attributes such as transaction amount, time, location, payment method, account age, number of previous transactions, and previous fraud history. The target variable indicates whether the transaction is fraudulent or not.

Business Problem

A financial institution processes thousands of transactions daily. A small percentage of these transactions are fraudulent. Manual investigation of every transaction is impossible due to:

- Large transaction volume
- Limited investigation staff
- High operational cost

The organization needs a predictive model that can:

- Identify suspicious transactions
- Estimate the probability of fraud
- Prioritize high-risk transactions for investigation

The main objective is to reduce fraud losses while minimizing false alarms.

Data Collection

The dataset contains historical transaction data with the following attributes:

- Transaction ID
- Customer ID
- Transaction amount
- Transaction date and time
- Merchant category
- Location of transaction
- Payment method
- Account age
- Number of past transactions
- Previous fraud history
- Target variable (Fraud: Yes/No)

Data Preparation

Data preparation includes the following steps:

1. Handling missing values
2. Removing duplicate records
3. Converting categorical variables into numerical format
4. Creating derived features such as:
    - Number of transactions in last 24 hours
    - Average transaction amount
    - Time difference between transactions
5. Detecting and removing outliers if necessary

Care is taken to avoid data leakage. Only information available at the time of the transaction is used for modeling.

Modeling Approach

Since the target variable is binary (Fraud or Not Fraud), classification algorithms are used. Common algorithms include:

- Logistic Regression
- Decision Trees
- Random Forest
- Gradient Boosting
- Support Vector Machines

The dataset is divided into:

- Training data (to build the model)
- Testing data (to evaluate performance)

Model Evaluation Metrics

Fraud detection is an imbalanced classification problem because fraudulent transactions are rare. Therefore, accuracy alone is not sufficient. Important evaluation metrics include:

- Precision
- Recall (Sensitivity)
- F1-Score
- ROC Curve
- Area Under the Curve (AUC)

Model Deployment

After validation, the model is deployed in the live transaction system.
For each new transaction:

1. The system calculates a fraud score (probability of fraud).
2. If the score exceeds a predefined threshold:

- The transaction is flagged.
- It is sent for manual review.
- Or it may be temporarily blocked.

Challenges in Fraud Detection

- Class imbalance (few fraud cases)
- Changing fraud patterns over time
- Data quality issues
- Risk of false positives affecting genuine customers
- Requirement for real-time prediction

Fraud detection using predictive analytics enables organizations to proactively prevent financial losses. By defining the correct unit of analysis, preparing high-quality data, selecting appropriate algorithms, and carefully evaluating model performance, businesses can build effective fraud detection systems. Successful deployment depends not only on model accuracy but also on operational integration and management support.

REFERENCES:

TEXTBOOK:

1. Dean Abbott, Applied Predictive Analytics, Published by Jhon Wiley &Sons, Inc,2014.

ONLINE REFERENCES:

https://www.datascience-pm.com/crisp-dm-2/

https://www.geeksforgeeks.org/computer-networks/difference-between-business-intelligence-and-predictive-analytics/

UNIT-II

Data Understanding: What the Data Looks Like, Single Variable Summaries, Data Visualization in One Dimension, Histograms, Multiple Variable Summaries, Data Visualization, Two or Higher Dimensions, The Value of Statistical Significance, Pulling It All Together into a Data Audit.

Data Preparation: Variable Cleaning, Feature Creation.

Data Understanding is the second phase of CRISP-DM.

Its purpose is to:

- Examine the raw data
- Understand its structure
- Identify data quality issues
- Discover initial patterns
- Detect problems before modeling

SINGLE VARIABLE SUMMARIES:

Single Variable Summaries refer to analyzing one variable at a time to understand its central tendency, spread, distribution shape, and overall behavior before building predictive models.

It helps us detect:

- Data errors
- Skewness
- Outliers
- Distribution type
- Unusual patterns

When performing Single Variable Summaries, we analyze:

1) Mean
2) Standard Deviation
3) Distribution Shape (Normal / Uniform)
4) Skewness
5) Kurtosis
6) Rank-Ordered Statistics (Median, Percentiles)
7) Categorical Variable Frequencies
8) Histogram (Visualization)

Mean:

Mean is the average value of a numeric variable.

$$Mean = \frac{\sum X}{N}$$

Example:

Marks of 5 students:
60, 70, 80, 90, 100

$$Mean = \frac{60 + 70 + 80 + 90 + 100}{5}$$

$$Mean = \frac{400}{5} = 80$$

So, average marks = 80.

Standard Deviation:

Standard deviation measures how much values deviate from the mean.

$$SD = \sqrt{\frac{\sum (X - \bar{X})^2}{N}}$$

Example:
Data: 2, 4, 6
Step 1: Mean

$$Mean = \frac{2+4+6}{3} = 4$$

Step 2: Deviations

$(2-4)^2 = 4$
$(4-4)^2 = 0$
$(6-4)^2 = 4$

Sum = 8

$$Variance = \frac{8}{3} = 2.67$$

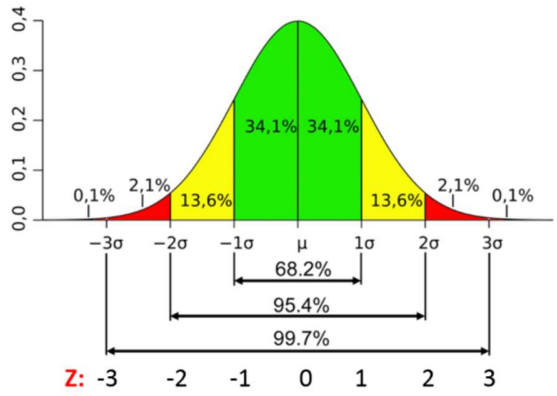
$$SD = \sqrt{2.67} = 1.63$$

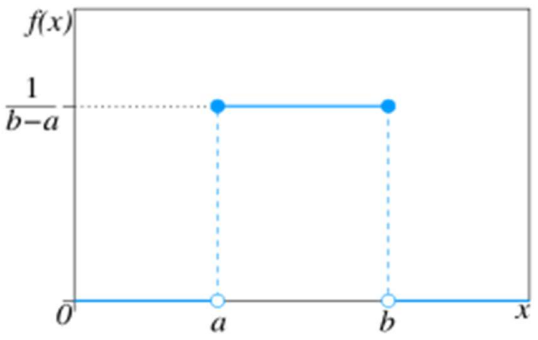
Standard deviation = 1.63

3. Distribution Shape:

A **distribution** describes how the values of a variable are spread across different ranges and how frequently each value occurs.

Normal Distribution	Uniform Distribution
• Bell-shaped curve • Perfectly symmetric • Highest point at the center **Properties** 1. Mean = Median = Mode 2. Symmetric around center 3. Total area under curve = 1 4. 68% data within ±1 SD 5. 95% data within ±2 SD 6. 99.7% data within ±3 SD **Example** Heights of students in a class often follow normal distribution. Most students have average height. Very short and very tall students are fewer.	• Flat shape • All values have equal probability **Properties** 1. No central peak 2. Mean = midpoint 3. Equal frequency across range **Example** Rolling a fair dice: Each number (1–6) has equal probability.





Skewness:

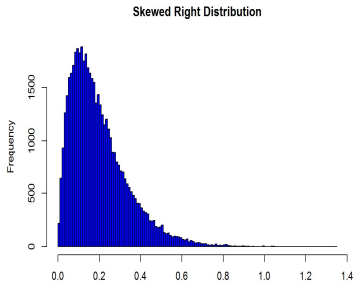
Skewness measures asymmetry of distribution.

$$Skewness = \frac{Mean - Mode}{Standard\ Deviation} \qquad Skewness = \frac{3(Mean - Median)}{Standard\ Deviation}$$

Positive Skew

Tail on right side

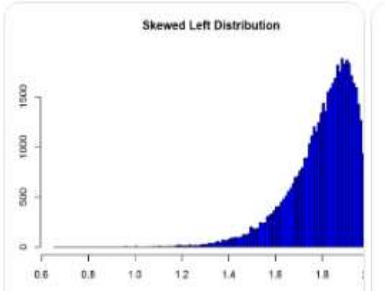
Mean > Median



Negative Skew

Tail on left side

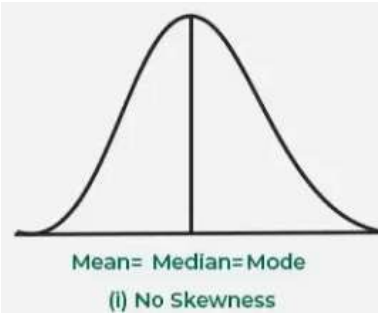
Mean < Median



Zero Skew

Perfectly symmetric

Mean=median=mode



Kurtosis:

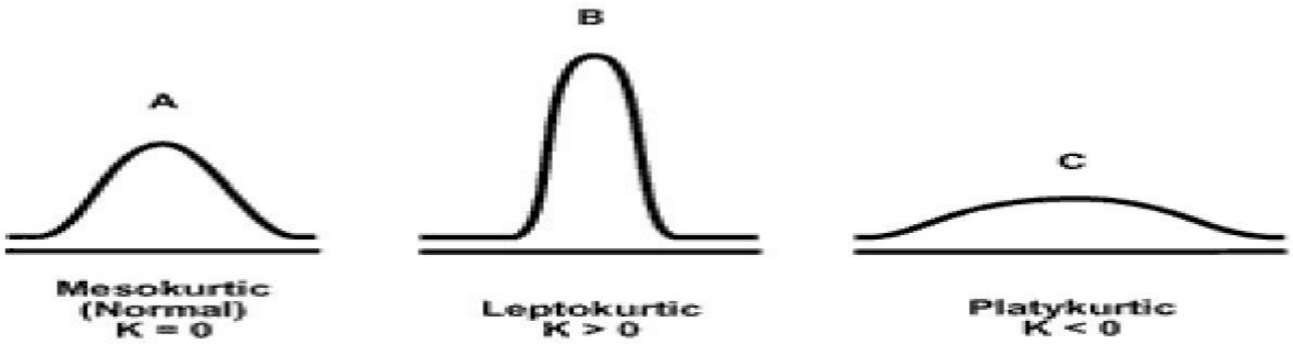
Kurtosis measures the **peakedness and tail heaviness** of a distribution.

It tells us:

- How sharp the peak is
- How heavy the tails are
- Whether extreme values (outliers) are present

$$Kurtosis = \frac{\sum(X_i - \mu)^4}{n\sigma^4} \qquad Excess\ Kurtosis = Kurtosis - 3$$

Mesokurtic Distribution:	**Leptokurtic Distribution**	**Platykurtic Distribution**
Normal distribution shape Moderate peak Moderate tails	Very sharp peak Heavy tails More outliers	Flat peak Thin tails Fewer extreme values
Kurtosis = 3	Kurtosis > 3	Kurtosis < 3
Symmetrical Medium tail thickness Balanced distribution	High central peak Extreme values present High risk data (like stock returns)	Wide distribution Less outliers Low extreme behavior
Example Dataset: Heights of students	**Example Dataset:** 10, 10, 10, 10, 50	**Example Dataset:** 10, 20, 30, 40, 50



Rank Ordered Statistics

Rank Ordered Statistics **are statistics calculated after arranging the data in** ascending or descending order.

- First we sort the data. Then we find position-based measures.

Why Rank ordered statistics?

- Not affected much by extreme values
- Easy to understand
- Useful in skewed data
- Important in predictive analytics

Types of Rank Ordered Statistics

1. Minimum
2. Maximum
3. Median
4. Quartiles
5. Percentiles
6. Interquartile Range (IQR)

Quartiles	**Interquartile Range (IQR)**	**Percentiles**	**Median**
Quartiles divide data into 4 equal parts. $Q_k = \frac{k(n+1)}{4}$ Q1 → 25% data below Q2 → Median Q3 → 75% data below	Measures spread of middle 50% data. $IQR = Q3 - Q1$	Percentile divides data into 100 equal parts. $P_k = \frac{k(n+1)}{100}$	Middle value in the sorted data. M=n+1/2

Data Visualization in One Dimension (1D)

Data Visualization means representing data in a graphical form.So that we can easily understand patterns, trends, and distribution.

Data Visualization in One Dimension (1D) is the graphical representation of a **single variable** to understand its distribution, center, spread, and shape.

In 1D visualization:

- Only **one variable** is analyzed.
- No comparison between two variables.
- The focus is on understanding the **characteristics of that single dataset**.

Common 1D Visualization Methods

- Histogram
- Boxplot
- Dot plot
- Stem-and-leaf plot

Boxplot

A **Boxplot** is a graphical representation that shows the **summary of data using five important values**:

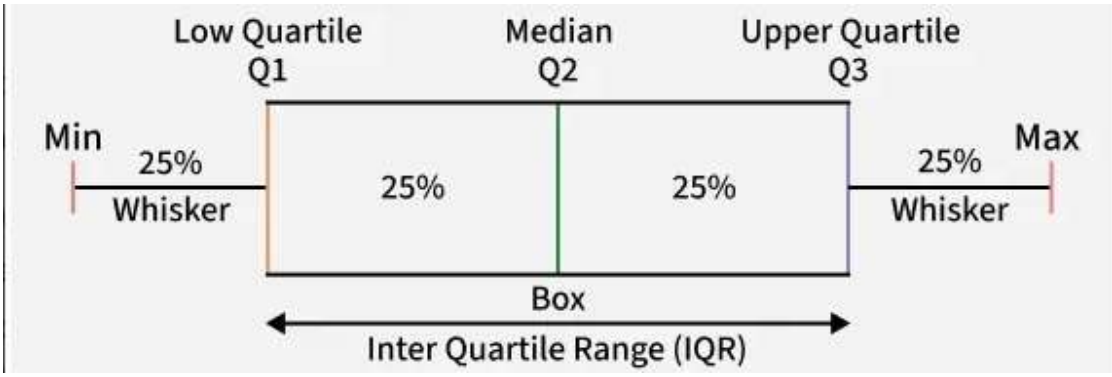
1. Minimum
2. First Quartile (Q1)
3. Median (Q2)
4. Third Quartile (Q3)
5. Maximum

It is also called a **Box and Whisker Plot**.

The box in a box plot represents the Interquartile Range (IQR).

IQR= Q3-Q1

It covers the middle 50% of the data, making it a strong measure of spread that is resistant to extreme values.



Detecting Outliers

Outlier Formula:

Lower Limit = Q1 − 1.5(IQR)
Upper Limit = Q3 + 1.5(IQR)

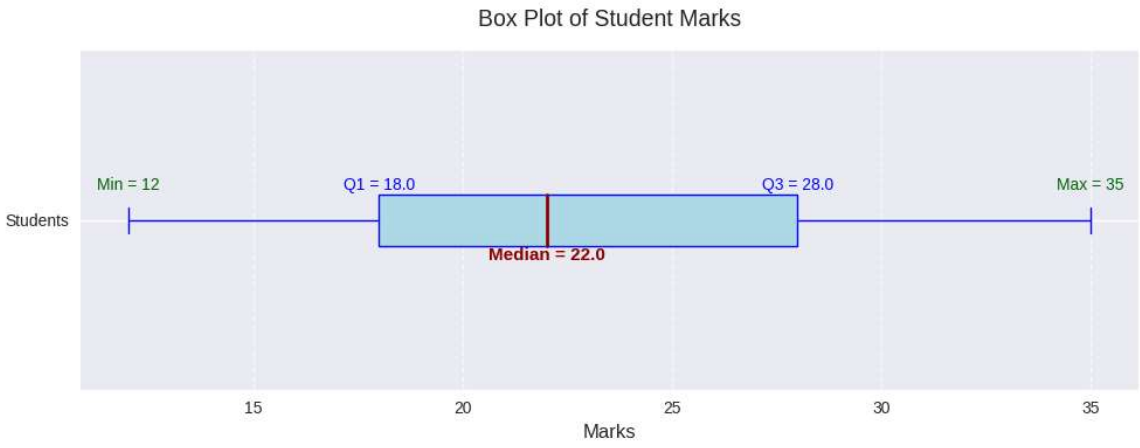
Values outside this range are outliers.

Problem1:

Given Data (Marks of 9 Students)

12, 15, 18, 20, 22, 25, 28, 30, 35

Step 1: Arrange Data The data is already arranged in ascending order:	**Step 2: Find Median (Q2)** Median position = (n + 1) / 2 = (9 + 1) / 2 = 5th value Median (Q2) = 22	**Step 3: Find First Quartile (Q1)** Lower half (excluding median): 12, 15, 18, 20 Q1 = Median of lower half = (15 + 18) / 2 = 16.5
Step 4: Find Third Quartile (Q3) Upper half (excluding median): 25, 28, 30, 35 Q3 = (28 + 30) / 2 = 29	Minimum = 12 Maximum = 35	IQR = Q3 − Q1 = 29 − 16.5 = 12.5



HISTOGRAM:

A **Histogram** is a graphical representation of the distribution of a numeric variable.

It displays how data values are grouped into intervals (called bins) and shows the frequency of observations in each interval.

It is one of the most important tools for understanding **what the data looks like**.

histograms help analyse:

- Understand distribution shape
- Detect skewness
- Identify outliers
- Detect gaps in data
- Observe multimodal patterns
- Detect possible data errors

Before building any predictive model, examining histograms is essential.

A histogram consists of:

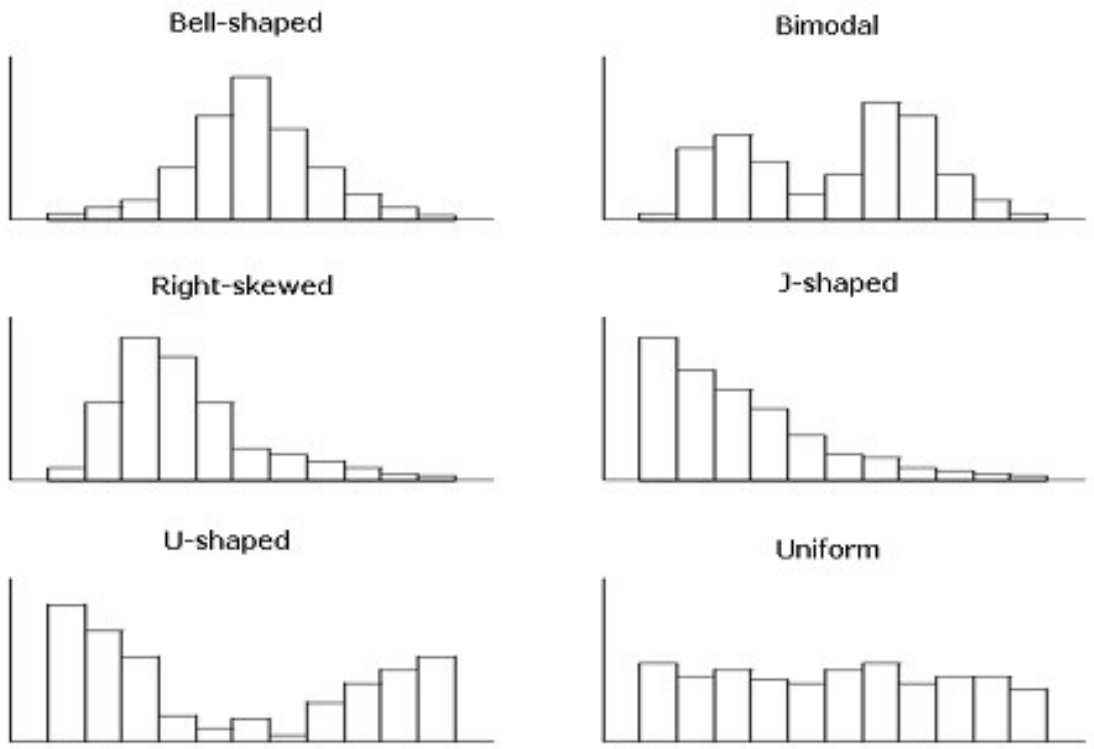
• X-axis → Numeric variable divided into intervals (bins)

• Y-axis → Frequency (count of observations)

• Bars → Represent number of values in each bin

What We Analyze in a Histogram?

1. Shape of Distribution	From histogram shape, we identify: • Normal distribution (bell-shaped) • Right skewed • Left skewed • Uniform • Bimodal
2. Spread of Data	We check: • Are values tightly packed? • Are they widely spread? • Is variance high or low?
3. Skewness	• Long right tail → Positive skew • Long left tail → Negative skew Skewed data may require transformation before modeling.
4. Outliers	Extreme values appear far away from main cluster. Outliers can: • Distort mean • Affect regression models • Indicate data errors
5. Multimodality	If histogram has two peaks: • Data may represent multiple populations • Suggests need for segmentation Example: Combining male and female heights.

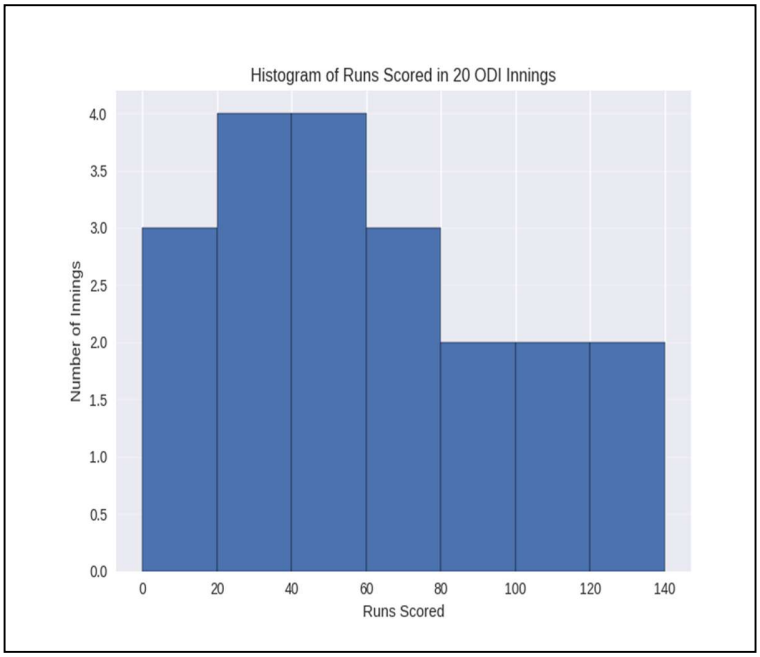


Problem: Cricket Runs Scored

Data: Runs scored in 20 ODI innings: 18, 45, 62, 7, 112, 34, 89, 56, 3, 121, 78, 41, 95, 22, 67, 104, 50, 31, 136, 28.

Group the data into class intervals (bin width = 20): 0–19, 20–39, 40–59, 60–79, 80–99, 100–119, 120–139.

Class Interval (Runs)	Frequency	Innings Examples
0 – 19	3	18, 7, 3
20 – 39	4	34, 22, 31, 28
40 – 59	4	45, 56, 41, 50
60 – 79	3	62, 78, 67
80 – 99	2	89, 95
100 – 119	2	112, 104
120 – 139	2	121, 136
Total	**20**	



Simpson's Paradox

Definition: A statistical phenomenon where trends in subgroups reverse or disappear when data is aggregated (combined). It highlights the importance of controlling for confounding variables.

Key Idea: Correlation in overall data may mislead if subgroups are ignored—often due to lurking variables (e.g., group size differences).

Example:

- **Scenario:** University admissions. Overall, men have higher acceptance rate (45%) than women (35%).
- **Subgroups (by department):**
    - Easy Dept: Women apply more (90% acceptance for both, but more women).
    - Hard Dept: Men apply more (30% acceptance for both).
- **Paradox:** Women actually have equal or higher rates per dept, but overall lower due to applying to easier depts. Aggregating ignores dept difficulty.

Always disaggregate data by potential confounders. Use stratified analysis or regression to control variables.

Multiple Variable Summaries:

Multiple variable summaries (also called multivariate summaries) are ways to describe and show patterns, relationships, or differences when you look at more than one variable at the same time.

In very simple words: It is a summary that answers questions like:

- How do two or more things change together?
- Do different groups behave differently on several measures?
- What is the joint pattern when we consider several variables together?

CORRELATION:

Correlation measures the strength and direction of the linear relationship between two numeric variables. It tells us

- Whether variables move together
- How strongly they are related
- Whether the relationship is positive or negative

r Value	Meaning
+1	Perfect positive correlation
0	No linear correlation
−1	Perfect negative correlation

Formula (Pearson Correlation)

$$r = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{\sqrt{\sum(X - \bar{X})^2 \sum(Y - \bar{Y})^2}}$$

Where:

X = First variable

Y = Second variable

$\bar{X}$ = Mean of X

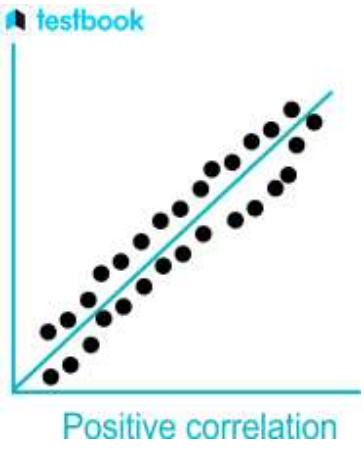
$\bar{Y}$ = Mean of Y

Value of r lies between −1 and +1.

1. Positive Correlation

When one variable increases, the other also increases. Vice versa

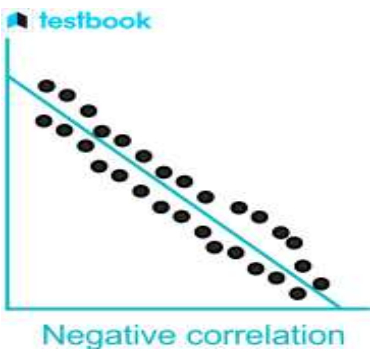
Example: Study hours and marks.



2. Negative Correlation

Negative correlation is a relationship between two variables in which one variable increases as the other decreases, and vice versa.

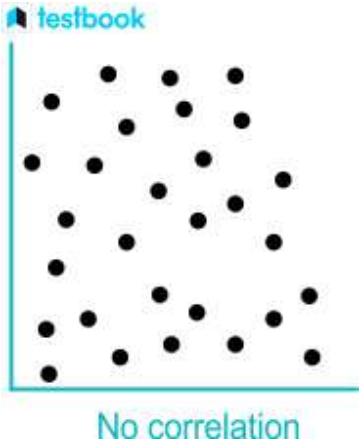
Example: Speed and travel time.



3.No Correlation

There also exists a condition known as no Correlation, where there is no relation or dependence between two variables.

Example: Shoe size and IQ.



PROBLEM: What kind of relationship exist between the following data?

X (Study Hours)	Y (Marks)
2	50
4	60
6	65
8	80
10	90

Step 1: Find Means

Mean of X:

$$\bar{X} = \frac{2+4+6+8+10}{5} = \frac{30}{5} = 6$$

Mean of Y:

$$\bar{Y} = \frac{50+60+65+80+90}{5} = \frac{345}{5} = 69$$

X	Y	X−6	Y−69	(X−6)(Y−69)	(X−6)²	(Y−69)²
2	50	-4	-19	76	16	361
4	60	-2	-9	18	4	81
6	65	0	-4	0	0	16
8	80	2	11	22	4	121
10	90	4	21	84	16	441

Now calculate sums:

$\Sigma(X-6)(Y-69) = 200$

$\Sigma(X-6)^2 = 40$

$\Sigma(Y-69)^2 = 1020$

$$r = \frac{200}{\sqrt{40 \times 1020}}$$

$$r = \frac{200}{\sqrt{40800}}$$

$$r = \frac{200}{202}$$

$$r \approx 0.99$$

Correlation coefficient $r \approx 0.99$

This indicates **very strong positive correlation** between study hours and marks.

CROSS TABS (Cross-Tabulation/Contingency Tables):

A **Crosstab** (Cross-Tabulation) is a table used to examine the relationship between two categorical variables by showing the frequency (count) of observations for each combination of categories.
It helps identify dependency patterns and compare proportions across groups. In predictive analytics, crosstabs are useful in understanding relationships between predictor variables and a categorical target variable.

It helps analyze how one category is distributed across another category.
Crosstabs are used to:

- Identify relationships between categorical variables
- Compare proportions across groups
- Detect dependency between variables
- Support classification problems
- Prepare for statistical tests like Chi-Square

Example :

Survey question: "Do you prefer tea or coffee?" by Gender (n=200 people)

Gender / Preference	Tea	Coffee	Total
Male	30	70	100
Female	60	40	100
Total	90	110	200

Interpretation:

- 30 males prefer tea (30% of males).
- 60 females prefer tea (60% of females).
- Females prefer tea more than males → possible association between gender and preference.

PROBLEM:

A company wants to analyze whether **Gender** is related to **Product Purchase (Yes/No)**.Data collected from 100 customers.

Gender Purchased

Male Yes

Female No

Male Yes

Female Yes

Male No

... ...

SOL:

Step 1: Construct Crosstab

Gender	Purchased Yes	Purchased No	Total
Male	30	20	50
Female	10	40	50
Total	40	60	100

Step 3: Convert to Row Percentages

Row percentages help compare proportions.

For Males:

Purchased Yes = 30 / 50 × 100 = 60%

Purchased No = 40%

For Females:

Purchased Yes = 10 / 50 × 100 = 20%

Purchased No = 80%

DATA VISUALIZATION IN TWO DIMENSIONS:

Data Visualization in Two Dimensions refers to graphical techniques used to examine the relationship between **two numeric variables simultaneously**.

Instead of studying one variable at a time (histogram), we study how two variables interact.

It helps detect:

- Linear relationships
- Nonlinear patterns
- Clusters
- Outliers
- Interactions
- Separation of target classes

Anscombe's Quartet :

Four different datasets can have:

• Same mean
• Same variance
• Same correlation

But completely different scatterplots.

Lesson:
Never rely only on statistics. Always visualize.

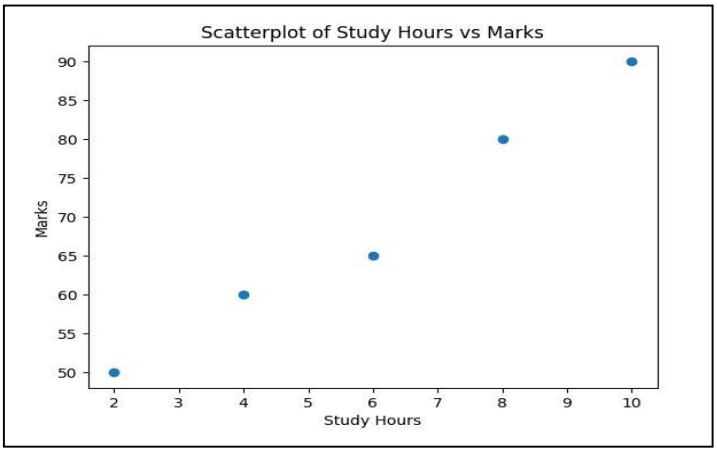
Scatterplot:

A **scatterplot** displays each observation as a point on a graph.

• X-axis → Variable 1
• Y-axis → Variable 2
• Each point → One observation

Problem:

Study Hours	Marks
2	50
4	60
6	65
8	80
10	90



THE VALUE OF STATISTICAL SIGNIFICANCE:

Statistical significance refers to determining whether an observed relationship or pattern in data is likely to be real or occurred by random chance.

It helps analysts decide whether a variable meaningfully contributes to explaining or predicting a target variable.

Statistical significance is used to:

- Test whether relationships between variables are real
- Support decisions about variable selection
- Identify meaningful predictors
- Reduce noise in modeling
- Improve reliability of conclusions

Concept of Hypothesis Testing

Statistical significance is evaluated using hypothesis testing.

Null Hypothesis (H_0): No relationship exists between variables.
Alternative Hypothesis (H_1): A relationship exists.

If the probability (p-value) is small (commonly less than 0.05), we reject the null hypothesis.

P-Value

The p-value measures the probability that the observed result occurred by chance.

- $p < 0.05 \rightarrow$ Statistically significant
- $p > 0.05 \rightarrow$ Not statistically significant

Statistical significance does not always mean practical importance.

In large datasets:

- Very small relationships may appear statistically significant.
- But they may not be practically useful for predictive modeling.

Therefore, analysts must consider:

- Effect size
- Business relevance
- Predictive power

PULLING IT ALL TOGETHER INTO A DATA AUDIT:

A Data Audit is a systematic summary of the dataset after completing data understanding.

It combines:

- Statistical summaries
- Visual analysis
- Data quality checks
- Relationship analysis

The goal is to fully understand strengths and weaknesses of the dataset before modeling.

Purpose of Data Audit

- Identify data quality issues
- Detect bias or imbalance
- Check suitability for modeling
- Identify transformation needs
- Determine variable usefulness

Why Data Audit is Important

- Prevents modeling errors
- Saves time in later stages
- Improves model accuracy
- Ensures consistency in data

A well-executed data audit is essential before moving to Data Preparation

DATA PREPARATION:

Data Preparation is the process of transforming raw data into a clean and structured format suitable for predictive modeling.

It includes:

- Variable Cleaning
- Feature Creation
- Transformation
- Sampling
- Variable Selection

VARIABLE CLEANING:

Variable cleaning refers to correcting errors, inconsistencies, and issues in individual variables to improve data quality.

Steps in Variable Cleaning

1. Handling Incorrect Values

Examples:

- Age = 250
- Negative income
- Invalid category names

These must be corrected or removed.

2. Ensuring Consistency

Examples:

- Male, male, M → Standardize to one format
- Date formats must be consistent

3. Handling Outliers

Outliers are extreme values far from other observations.

They may:

- Be true rare cases
- Be data entry errors

Methods:

- Remove
- Cap
- Transform

4. Handling Missing Values

Missing data can occur due to:

- System errors
- Non-response
- Data entry issues

Methods:

• Delete records
• Replace with mean/median
• Use predictive imputation

Importance of Variable Cleaning

• Improves model performance
• Prevents bias
• Reduces noise
• Ensures reliability

FEATURE CREATION:

Feature Creation (Feature Engineering) is the process of generating new variables from existing data to improve predictive accuracy, model performance, and interpretability.

It transforms raw data into meaningful inputs suitable for modeling.

Feature creation is one of the most powerful steps in predictive analytics because better features often improve model performance more than changing algorithms.

Types of Feature Creation.

- Variable Transformation
- Binning Continuous Variables
- Scaling Numeric Variables
- Creating Interaction Variables
- Date and Time Features

1. Variable Transformation:

Variable Transformation involves applying a mathematical function to an existing numeric variable to change its scale or distribution.

Common Types of Transformations

(a) Log Transformation New Variable = log(X) Used when data is positively skewed. Example: Income values: 10,000; 20,000; 50,000; 2,00,000 The log transformation compresses large values and reduces skew.	**(b) Square Root Transformation** New Variable = √X Used when skewness is moderate.
(c) Square Transformation New Variable = X^2 Used when larger values need more emphasis.	**(d) Reciprocal Transformation** New Variable = 1 / X Used when inverse relationships exist.

2. Binning Continuous Variables

Binning is the process of converting continuous numeric variables into categorical groups (intervals).

Example:
Age →
0–18 = Young
19–40 = Adult
41–60 = Middle-aged
60+ = Senior

Types of Binning:

(a) Equal Width Binning Divide range into equal intervals. Example: Income 0–100,000 divided into 5 equal ranges.	**(b) Equal Frequency Binning** Each bin contains approximately the same number of observations.

3. Scaling Numeric Variables

Scaling adjusts the range of numeric variables so that they are on similar scales.

Without scaling, variables with larger values dominate distance-based algorithms.

(a) Standardization (Z-score Scaling)	(b) Min-Max Normalization
$Z = (X - \text{Mean}) / \text{Standard Deviation}$ After scaling: • Mean = 0 • Standard deviation = 1	$X\_new = (X - \text{Min}) / (\text{Max} - \text{Min})$ Rescales values between 0 and 1.

4. Creating Interaction Variables

Interaction variables are created by combining two or more variables to capture their joint effect.

Example:

Income × Age
Study Hours × Attendance

Mathematical Form

New Variable = $X_1 \times X_2$

Used commonly in regression models.

5. Date and Time Features

Date and Time features are derived from timestamp variables to capture temporal patterns.

Common Extracted Features

From Date:

• Year
• Month
• Day
• Day of Week

• Quarter
• Weekend indicator

From Time:

• Hour
• Time of day (Morning, Afternoon, Night)

Feature creation improves model quality by:

- Reducing skewness
- Capturing nonlinear patterns
- Standardizing variable scales
- Representing interaction effects
- Extracting temporal behavior

In predictive analytics, carefully engineered features often determine model success.

REFERENCES:

TEXTBOOK:

1. Dean Abbott, Applied Predictive Analytics, Published by Jhon Wiley &Sons, Inc,2014.

UNIT-3

Item sets and Association Rules: Terminology, Parameter Settings, How the Data Is Organized, Measures of Interesting Rules, Deploying Association Rules, Problems with Association Rules, Building Classification Rules from Association Rules. Descriptive Modelling: Data Preparation Issues with Descriptive Modelling, Principal Component Analysis, Clustering Algorithms. Interpreting Descriptive Models: Standard Cluster Model Interpretation.

Item sets and Association Rules: Terminology:

Association Rules are a data mining technique used to discover interesting relationships or patterns between variables in large datasets, especially transactional data.

They are commonly used in market basket analysis, where the goal is to identify products that are frequently purchased together.

BASIC TERMINOLOGY IN ASSOCIATION RULES:

1. Item:

An **item** is a single element or product in a dataset.

Example:
Milk, Bread, Butter, Eggs

Each individual product is called an item.

2. Itemset

An **Itemset** is a collection of one or more items.

Example:

- {Milk} → 1-itemset
- {Milk, Bread} → 2-itemset
- {Milk, Bread, Butter} → 3-itemset

Itemsets represent combinations of items that appear together in transactions.

3. Transaction

A **transaction** is a single record containing a set of items purchased together.

Example:

Transaction 1: {Milk, Bread, Butter}

Transaction 2: {Milk, Eggs}

Each row in transactional data is a transaction.

4. Support

Support measures how frequently an itemset appears in the dataset.

Formula:

Support(Itemset): =

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

Example

Suppose there are 5 transactions:

T1: {Milk, Bread}

T2: {Milk, Butter}

T3: {Milk, Bread}

T4: {Bread, Butter}

T5: {Milk, Bread}

Support of {Milk, Bread} = 3 / 5 = 0.60 (60%)

5. Association Rule

An **Association Rule** is an implication of the form:

$$X \rightarrow Y$$

Where:

X = Antecedent (Itemset)

Y = Consequent (Itemset)

It means:

If X occurs, Y is likely to occur.

6. Antecedent (Left-Hand Side – LHS)

The **Antecedent** is the "if" part of an association rule.

Example:

If Milk → then Bread

Milk is the antecedent.

7. Consequent (Right-Hand Side – RHS)

The **Consequent** is the "then" part of an association rule.

Example:

If Milk → then Bread

Bread is the consequent.

8. Confidence(accuracy):

Confidence is a measure used in association rule mining to evaluate the reliability or strength of an association rule.

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

It represents the probability that the consequent (right-hand side) occurs given that the antecedent (left-hand side) has occurred.

In simple terms, confidence tells us:

"How often does Y occur when X has occurred?"

Example:

Suppose:

Out of 100 transactions,

40 customers bought Milk.

30 customers bought both Milk and Bread.

Confidence (Milk → Bread) =30 / 40 = 0.75

This means 75% of customers who bought Milk also bought Bread.

PARAMETER SETTINGS:

Parameter Settings refer to the user-defined thresholds and control values that guide the generation of association rules in itemset mining.

In association rule algorithms (such as Apriori), parameters determine:

- Which itemsets are considered important
- Which rules are generated
- How many rules are produced
- The strength and usefulness of rules

Proper parameter settings are critical because poor settings can result in:

- Too many meaningless rules
- Too few useful rules
- Missed business insights

Main Parameters in Association Rules:

1. Minimum Support:

Minimum Support is the minimum frequency required for an itemset to be considered "frequent."

Only itemsets whose support exceeds this threshold are used to generate rules.

Purpose

- Eliminates rare item combinations
- Reduces computational complexity
- Focuses on meaningful patterns

Effect of Setting Support

If Minimum Support is too high:

- Very few itemsets qualify
- Important but less frequent patterns may be missed

If Minimum Support is too low:

- Too many itemsets are generated
- Computational cost increases

2. Minimum Confidence

Minimum Confidence is the minimum reliability required for a rule to be accepted.

Only rules with confidence greater than the threshold are retained.

Purpose

- Filters weak rules
- Ensures rule reliability
- Improves rule quality

Effect of Setting Confidence

If too high:

- Few rules generated
- May miss useful associations

If too low:

- Weak and unreliable rules included

3. Maximum Length of Itemset

Controls the maximum number of items allowed in an itemset.

Example:

If maximum length = 3

Then itemsets with more than 3 items are not generated.

Purpose

- Prevents overly complex rules
- Improves interpretability
- Reduces computational complexity

HOW THE DATA IS ORGANIZED:

In association rule mining, data must be organized in a **transactional format**, not in the traditional flat-file format used for predictive modeling.

Two Types of Data Organization

1. Standard Predictive Modeling Format

- Rows represent observations
- Columns represent variables
- Used in regression, classification, clustering

- Rows = Transactions
- Columns = Items
- Values = 1 (Purchased) or 0 (Not Purchased)

This is also called **binary format**.

Transaction	Milk	Bread	Butter	Eggs
T1	1	1	0	0
T2	1	0	1	0
T3	0	1	1	1
T4	1	1	1	0
T5	0	0	1	1

The standard 0–1 (flat file) format is not ideal for association rule mining because it stores both the presence and absence of items, resulting in many unnecessary zero values.

In large retail datasets with thousands of products, this creates sparse and high-dimensional data, increasing storage and computational complexity.

Association rule algorithms such as Apriori focus only on items that co-occur in transactions and do not require information about items that were not purchased.

2. Transactional Format

In Transactional Format:

• Each row represents one transaction.
• Each row contains only the list of items purchased.
• There are no separate columns for each item.

This format is required for association rule mining algorithms such as Apriori.

Transaction	**Item Name**
T1	Milk
T1	Bread
T2	Milk
T2	Butter
T3	Bread
T3	Butter
T3	Eggs
T4	Milk
T4	Bread
T4	Butter

Transaction	Item Name
T5	Butter
T5	Eggs

MEASURES OF INTERESTING RULES

Not all generated rules are useful. Measures of interestingness evaluate rule strength and usefulness.

1. Support

Support measures how frequently an itemset appears in the dataset.

Support(X → Y) =Number of transactions containing X and Y / Total transactions

Higher support → Rule is common in dataset.

2. Confidence

Confidence measures rule reliability.

Confidence(X → Y) =Support(X ∪ Y) / Support(X)

It shows how often Y occurs when X occurs.

3. Lift

Lift measures the strength of association compared to random chance.

Lift(X → Y) =Confidence(X → Y) / Support(Y)

Interpretation:

- Lift = 1 → No association
- Lift > 1 → Positive association
- Lift < 1 → Negative association

Lift helps detect meaningful rules beyond simple frequency.

Importance of Interestingness Measures

- Reduce trivial rules
- Identify actionable patterns

- Improve business usefulness
- Prevent misleading conclusions

DEPLOYING ASSOCIATION RULES:

Deploying association rules refers to applying the discovered itemset patterns in real business environments to support decision-making, recommendations, and operational strategies.

Association rules are not built just for analysis; they are built to influence actions.

Purpose of Deployment

The main objective of deploying association rules is to:

• Improve cross-selling and up-selling
• Increase revenue
• Improve customer experience
• Optimize product placement
• Support recommendation systems

Deployment converts discovered patterns into practical business strategies.

Common Applications

1. Cross-Selling

Cross-selling involves recommending related or complementary products based on items already purchased.

Example:

If a customer buys a Laptop → suggest a Mouse or Laptop Bag.

Association rules identify items that are frequently purchased together. Businesses use these rules to increase the total value of each transaction.

Purpose:

Increase average basket size and overall sales.

2. Up-Selling

Up-selling encourages customers to purchase a higher-value or premium version of a product.

Example:

If a customer buys a basic smartphone → suggest a premium model with better features.

Association rules can reveal purchasing patterns that help identify customers likely to upgrade.

Purpose:

Increase profit per customer.

4. Fraud Detection

Association rules can identify unusual combinations of transactions that deviate from normal patterns.

Example:

If a credit card transaction includes an unusual set of purchases that rarely occur together, it may indicate fraud.

By comparing current transactions with normal association patterns, suspicious activity can be flagged.

Purpose:

Reduce financial loss and detect abnormal behavior.

5. Market Basket Analysis

Retailers use association rules to identify products frequently purchased together.

Example:

If customers buy Bread and Butter → they are likely to buy Jam.

Business action:

Place products near each other or create combo offers.

PROBLEMS WITH ASSOCIATION RULES:

1. Too Many Rules

Association rule algorithms can generate a very large number of rules, especially when minimum support and confidence thresholds are low.

- Large datasets produce thousands of rules.
- Many rules may not be meaningful.
- It becomes difficult to interpret and select useful rules.

This makes rule filtering and proper parameter tuning essential.

2. Too Few Rules

If minimum support or confidence thresholds are set too high:

- Very few rules may be generated.
- Important but less frequent patterns may be missed.
- Valuable business insights can be lost.

Thus, overly strict parameter settings may eliminate useful associations.

3. Redundant Rules

Many generated rules may convey essentially the same information.

Example:
Milk → Bread
Milk, Butter → Bread

The second rule may not provide much new insight beyond the first.

Redundant rules:

- Increase complexity
- Reduce clarity
- Make interpretation difficult

Therefore, redundant rules must be filtered before deployment.

BUILDING CLASSIFICATION RULES FROM ASSOCIATION RULES:

Association rules are normally descriptive; they discover relationships among items in transactional data. However, they can be adapted to build classification rules by restricting the rule structure so that the consequent (right-hand side) contains only the target class variable.

Basic Idea:

In traditional association rules:

$X \rightarrow Y$

Both X and Y are itemsets.

In classification rule mining:

$X \rightarrow Class$

Where:

- X = Set of predictor variables
- Class = Target variable

Thus, the rule predicts membership in a specific class.

How It Works

1. Generate frequent itemsets from the dataset.
2. Restrict the consequent to be the class variable only.
3. Calculate support and confidence.
4. Select rules with high confidence and strong support.
5. Use these rules to classify new observations.

Example

Suppose the target variable is Purchase = Yes or No.

A classification rule may look like:

$\{Age < 30, Income = High\} \rightarrow Purchase = Yes$

This means customers with these characteristics are likely to make a purchase.

Here:

- Left side = Predictor variables
- Right side = Class label

Descriptive Modelling:

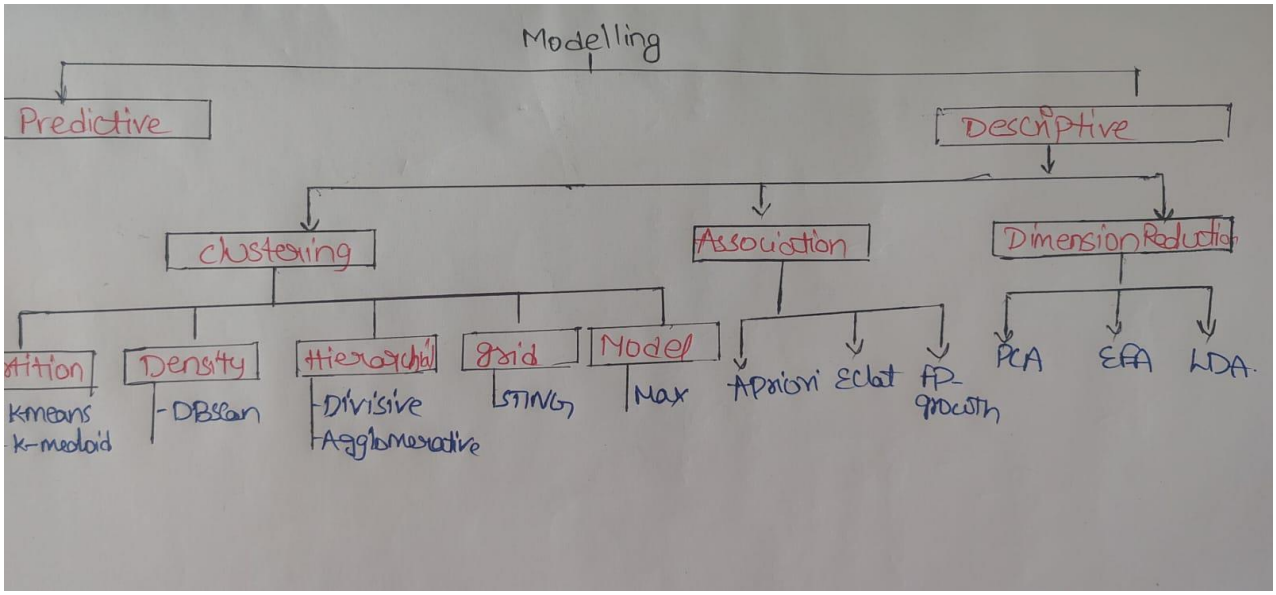
Descriptive modelling algorithms are also called unsupervised learning methods. There is no target feature or label in unsupervised learning. These methods are used to solve descriptive problems. Unsupervised learning is used to:

- describe a dataset
- gain insights from a dataset

Based on all values of all features, interesting patterns and insights are derived about the dataset. Descriptive models group similar data instances together, i.e., data instances having similar values of different features are grouped together.

Examples of descriptive models:

1. K-Means
2. Grouping / clustering music (e.g., music recommendation systems)
3. Market Basket Analysis (Association Rule Learning – often grouped under descriptive/unsupervised techniques)
4. PCA (Principal Component Analysis – dimensionality reduction, often considered a descriptive technique)
5. Kohonen Networks / Self-Organizing Maps (SOM)



Data Preparation Issues in Descriptive Modelling:

Descriptive modelling methods such as clustering, self-organizing maps, and principal component analysis are used to discover patterns and structures in data. Because these techniques rely heavily on similarity and distance calculations, proper data preparation is essential. Poorly prepared data can produce misleading groupings and incorrect interpretations.

1. Inputs Must Be Numeric

Descriptive modelling algorithms require numeric input. However, real-world datasets often contain categorical attributes such as gender, location, or product category.

If categorical data is used directly, the model cannot compute similarity between records.

Solution: Convert categorical variables into numeric form using dummy variables (one-hot encoding).

Example:

Gender → Male = 1, Female = 0

City → Hyderabad, Chennai, Delhi → three binary columns

2. No Missing Values Allowed

Algorithms like clustering and PCA cannot process missing values.

Problem:

- Missing entries distort similarity measurements.

Solutions:

Impute missing values using:

- Mean / median
- Regression estimation
- Domain-based replacement

Example:

If income is missing for some customers, replace with the median income of similar customers.

3. Dealing with Outliers

Outliers are extreme values that differ significantly from other observations. Distance-based methods treat outliers as separate clusters, which can distort results.

Solutions:

- Remove extreme values.
- Cap values at acceptable limits.
- Transform skewed data.

Example:
If most customer spending ranges from ₹500 to ₹5,000 but one record shows ₹2,00,000, it may distort clustering

4. Data Should Be Properly Distributed

Practitioners often transform data so variables are closer to a **normal distribution**.

- **Problem:**
- Skewed data can dominate distance calculations.
- Clusters become biased.
- **Solutions:**
    - Remove extreme outliers
    - Reduce skewness (log or square root transformation)
    - Normalize distributions

5. Correlated and Redundant Variables

Highly correlated variables contain similar information. Including them gives extra weight to certain patterns and biases clustering.

Solutions:

- Remove redundant variables.
- Use Principal Component Analysis (PCA) to reduce dimensionality.

Example:
Total purchase amount and number of items purchased may be highly correlated. Keeping both may overemphasize purchasing behaviour.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical technique used to reduce the number of variables in a dataset while preserving as much information (variance) as possible. It transforms the original correlated variables into a new set of uncorrelated variables called principal components.

What is a Principal Component?

A principal component is a new variable formed as a weighted combination of original variables.

PC1 -captures the maximum possible variance.

PC2 -captures the next highest variance and is perpendicular to PC1.

PC3 -captures the next highest variance and is perpendicular to both PC1 and PC2.

This continues until all variance is explained.

Purpose of PCA:

- Reduce dimensionality of data
- Remove redundancy caused by correlated variables
- Simplify complex datasets
- Improve visualization of high-dimensional data
- Enhance efficiency of further analysis

How Principal Component Analysis Works

PCA uses linear algebra to transform data into new features called principal components. It finds these by calculating eigenvectors (directions) and eigenvalues (importance) from the covariance matrix.

PCA selects the top components with the highest eigenvalues and projects the data onto them simplify the dataset.

Imagine you're looking at a messy cloud of data points like stars in the sky and want to simplify it. PCA helps you find the "most important angles" to view this cloud so you don't miss the big patterns.

Step 1: Standardize the Data

Different features may have different units and scales like salary vs. age. To compare them fairly PCA first standardizes the data by making each feature have:

- A mean of 0
- A standard deviation of 1

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

where:

- μ is the mean of independent features $\mu = \{\mu_1, \mu_2, \cdots, \mu_m\}$

σ is the standard deviation of independent features $\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_m\}$

Step 2: Calculate Covariance Matrix

Next PCA calculates the covariance matrix to see how features relate to each other whether they increase or decrease together. The covariance between two features x_1 and x_2 is:

Where:

- $\bar{x}_1 and \bar{x}_2$ are the mean values of features $x_1 and x_2$
- n is the number of data points

The value of covariance can be positive, negative or zeros.

Step 3: Find the Principal Components

PCA identifies new axes where the data spreads out the most:

- **1st Principal Component (PC1):** The direction of maximum variance (most spread).
- **2nd Principal Component (PC2):** The next best direction, *perpendicular to PC1* and so on.

These directions come from the eigenvectors of the covariance matrix and their importance is measured by eigenvalues. For a square matrix A an eigenvector X (a non-zero vector) and its corresponding eigenvalue λ satisfy:

$$AX = \lambda X$$

This means:

- When *A* acts on X it only stretches or shrinks X by the scalar λ.

- The direction of X remains unchanged hence eigenvectors define "stable directions" of A.

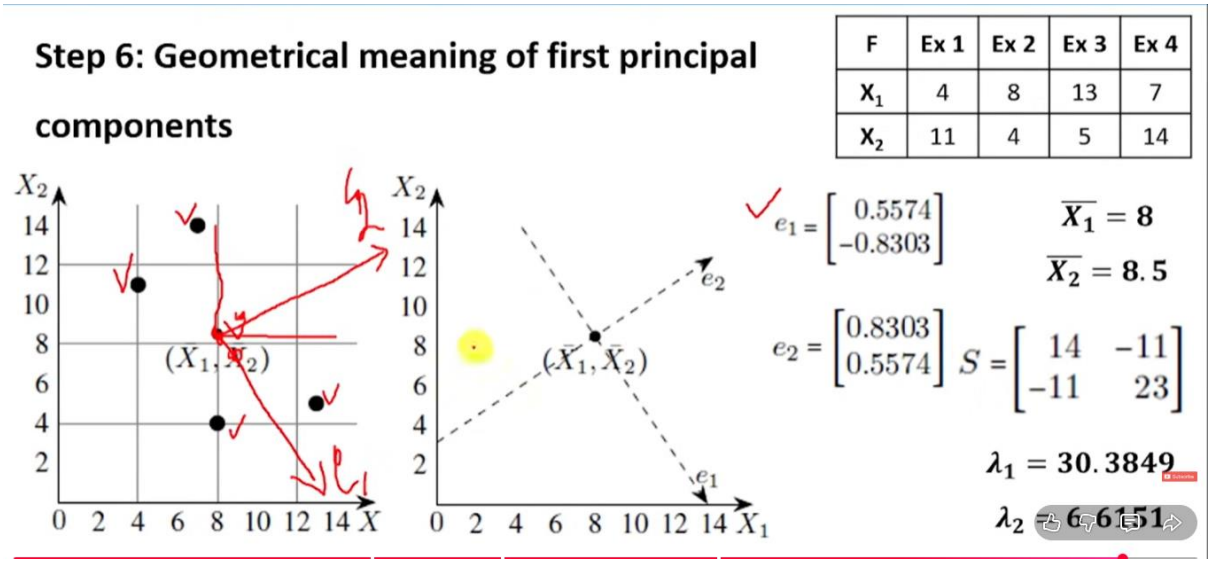
Eigenvalues help rank these directions by importance.

Step 4: Pick the Top Directions & Transform Data

After calculating the eigenvalues and eigenvectors PCA ranks them by the amount of information they capture. We then:

1. Select the top k components that capture most of the variance like 95%.
2. Transform the original dataset by projecting it onto these top components.

This means we reduce the number of features (dimensions) while keeping the important patterns in the data.



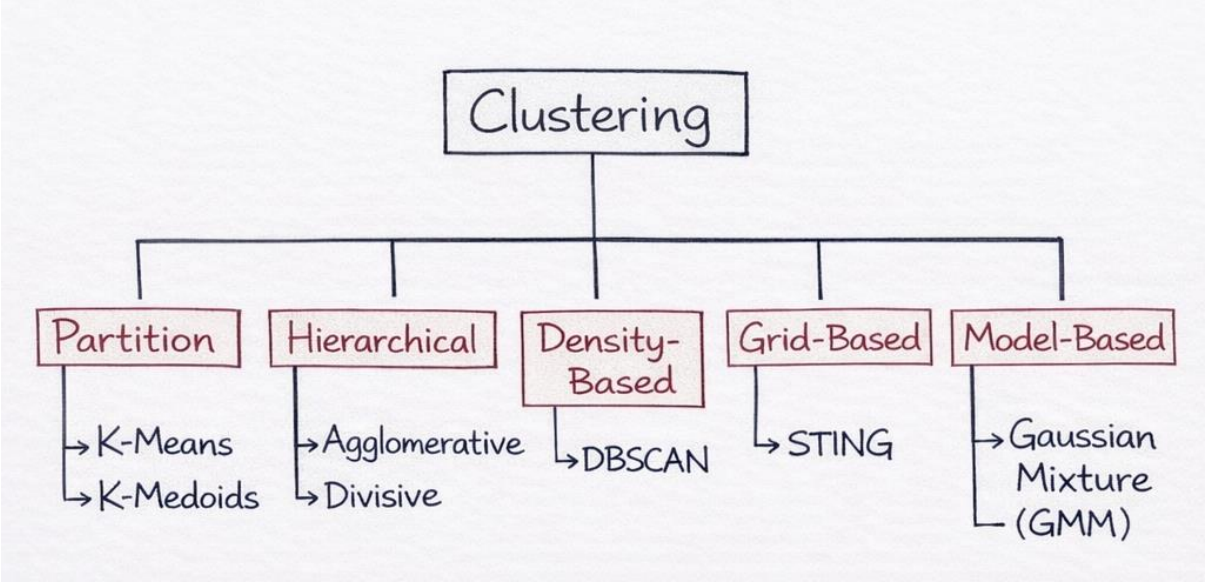
CLUSTERING ALGORITHMS:

Clustering is a descriptive modelling technique used in predictive analytics to discover natural groupings within data when no target variable is available.

It is an **unsupervised learning method**, meaning that there are no predefined classes or labels.

The goal is to group records into similar segments.

It is mainly used for segmentation and pattern discovery.



Choosing Number of Clusters	**Evaluation of Clustering**	**Strengths of Clustering**
□ Elbow Method □ Silhouette Score □ Business Interpretability	Since clustering has no target variable: Evaluation methods include: • Within-cluster variance • Between-cluster separation • Stability of clusters • Business usefulness	✔ Useful for exploration ✔ Helps feature engineering ✔ Simplifies segmentation ✔ Supports predictive modelling

Limitations Of Clustering:

Sensitive To:

- Outliers
- Initial Centroid (K-Means)
- Scaling

ELBOW METHOD:

The **Elbow Method** is a technique to determine the optimal number of clusters (K) in K-Means.

It works by:

1. Running K-Means for different values of K (K = 1, 2, 3, 4, 5 …)
2. For each K, compute:

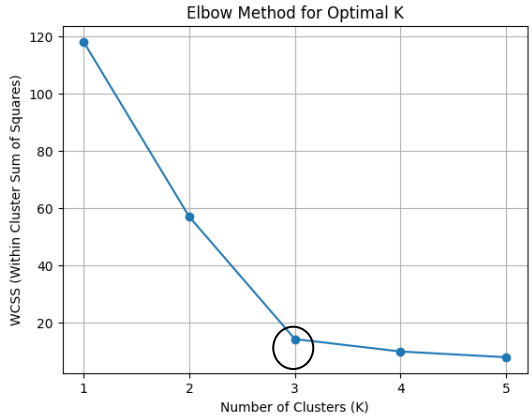
   WCSS (Within-Cluster Sum of Squares)
   Plot:
   - X-axis → Number of Clusters (K)
   - Y-axis → WCSS

When K increase WCSS decreases..

At some point:

- The decrease becomes slow
- The graph forms a bend (like an elbow)

👉 That bend point = Optimal K



K-MEANS CLUSTERING

K-Means clustering is an unsupervised machine learning algorithm used to group similar data points into K clusters based on distance.

It partitions the dataset into K non-overlapping clusters such that each data point belongs to the cluster with the nearest centroid (mean).

It is a centroid-based, partitioning clustering method.

Objective of k means Clustering

- **Grouping similar data points:** K-Means groups data points that are similar to each other to find hidden patterns in the data.
- **Minimizing within-cluster distance:** It tries to keep points inside a cluster as close as possible (usually using Euclidean distance).
- **Maximizing between-cluster distance:** It also tries to keep different clusters far apart so that each cluster is clearly distinct.

K-MEANS ALGORITHM STEPS

Step 1: Select number of clusters K.

Step 2: Randomly initialize K centroids.

Step 3: Compute Euclidean distance of each data point to each centroid.

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

Step 4: Assign each data point to the nearest centroid.

Step 5: Recalculate centroids using the mean formula:

Example:

If a cluster has two 2D points $(1, 2)$ and $(3, 4)$, the new centroid is $(\frac{1+3}{2}, \frac{2+4}{2}) = (2, 3)$.

$$\mu_x = \frac{\sum x_i}{n}$$

$$\mu_y = \frac{\sum y_i}{n}$$

Step 6: Repeat Steps 3 to 5 until cluster assignments no longer change or centroids stabilize.

Distance Metrics Used in K-Means (2D Formulas Only)

Distance Metric	Formula (2D)	When to Use
Euclidean Distance (L2)	$d = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$	Default distance in K-Means. Used for numeric, scaled data and spherical clusters.
Squared Euclidean Distance	$d = (x_2-x_1)^2+(y_2-y_1)^2$	This is the distance actually minimized in K-Means (WCSS).
Manhattan Distance (L1)	$d = \lvert x_2-x_1\rvert + \lvert y_2-y_1\rvert$	Used when data contains outliers. Produces diamond-shaped clusters. Leads to K-Medians.

Problem:1 Apply K–Means Clustering to the following data points:

A1(2,10), A2(2,5), A3(8,4),

B1(5,8), B2(7,5), B3(6,4),

C1(1,2), C2(4,9)

Take **K = 3?**

Sol: Initial centroids are:

C_1 = A1(2,10)

C_2 = B1(5,8)

C_3 = C1(1,2)

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

Distance Table – Iteration:

Point	d to C_1(2,10)	d to C_2(5,8)	d to C_3(1,2)	Cluster
A1(2,10)	0.00	3.61	8.06	C1
A2(2,5)	5.00	4.24	3.16	C3
A3(8,4)	8.49	5.00	7.28	C2
B1(5,8)	3.61	0.00	7.21	C2
B2(7,5)	7.07	3.61	6.71	C2
B3(6,4)	7.21	4.12	5.39	C2
C1(1,2)	8.06	7.21	0.00	C3
C2(4,9)	2.24	1.41	7.62	C2

New C_1: Only A1

C1=(2,10)

New C2:

Points: (8,4), (5,8), (7,5), (6,4), (4,9)

x = (8 + 5 + 7 + 6 + 4)/5 = 30/5 = 6

y = (4 + 8 + 5 + 4 + 9)/5 = 30/5 = 6

C2 = (6, 6)

New C3:

Points: (2,5), (1,2)

x = (2 + 1)/2 = 1.5

y = (5 + 2)/2 = 3.5

C3 = (1.5, 3.5)

Distance Table – Iteration 2:

Point	D To C1(2,10)	D To C_2(6,6)	D To C_3(1.5,3.5)	Old cluster	New Cluster
A1(2,10)	0.00	5.66	6.52	C1	C1
A2(2,5)	5.00	4.12	1.58	C3	C3
A3(8,4)	8.49	2.83	6.52	C2	C2
B1(5,8)	3.61	2.24	5.70	C2	C2
B2(7,5)	7.07	1.41	5.70	C2	C2
B3(6,4)	7.21	2.00	4.53	C2	C2
C1(1,2)	8.06	6.40	1.58	C3	C3
C2(4,9)	2.24	3.61	6.04	C2 →	C1

New C_1:

(2,10), (4,9)

x = (2 + 4)/2 = 3

y = (10 + 9)/2 = 9.5

C1 = (3, 9.5)

New C_2:

(8,4), (5,8), (7,5), (6,4)

x = (8 + 5 + 7 + 6)/4 = 6.5

y = (4 + 8 + 5 + 4)/4 = 5.25

C2 = (6.5, 5.25)

New C3:

C3= (1.5, 3.5)

Distance Table – Iteration 3:

Point	D To C_1 (3,9.5)	D To C_2 (6.5,5.25)	D To C_3 (1.5,3.5)	Old Cluster	New Cluster
A1 (2,10)	1.12	6.54	6.52	C1	C1
A2 (2,5)	4.61	4.51	1.58	C3	C3
A3 (8,4)	7.43	1.95	6.52	C2	C2
B1 (5,8)	2.50	3.13	5.70	C2 →	C1
B2 (7,5)	6.02	0.56	5.70	C2	C2
B3 (6,4)	6.26	1.35	4.53	C2	C2
C1 (1,2)	7.76	6.39	1.58	C3	C3
C2 (4,9)	1.12	4.51	6.04	C1	C1

New C_1:

(2,10), (5,8), (4,9)

x = (2 + 5 + 4) / 3 = 11 / 3 = 3.67

y = (10 + 8 + 9) / 3 = 27 / 3 = 9

C1 = (3.67, 9)

New C_2:

(8,4), (7,5), (6,4)

x = (8 + 7 + 6) / 3 = 21 / 3 = 7

y = (4 + 5 + 4) / 3 = 13 / 3 = 4.33

C2 = (7, 4.33)

New C_3:

(2,5), (1,2)

x = (2 + 1) / 2 = 3 / 2 = 1.5

y = (5 + 2) / 2 = 7 / 2 = 3.5

C3 = (1.5, 3.5)

Distance Table – Iteration 4:

Point	D To C_1 (3.67,9)	D To C_2 (7,4.33)	D To C_3 (1.5,3.5)	Old Cluster	New Cluster
A1 (2,10)	1.94	7.62	6.67	C1	C1
A2 (2,5)	4.18	5.04	1.58	C3	C3
A3 (8,4)	6.61	1.05	6.52	C2	C2
B1 (5,8)	1.67	4.12	5.70	C1	C1
B2 (7,5)	4.96	0.67	5.52	C2	C2
B3 (6,4)	5.49	1.05	4.74	C2	C2
C1 (1,2)	7.34	6.02	1.58	C3	C3
C2 (4,9)	0.33	5.84	6.04	C1	C1

Same

Now stop the iteration.

Final Centroids:

C_1 = (3.67, 9)
C_2 = (7, 4.33)
C_3 = (1.5, 3.5)

Cluster1: A1(2,10),B1(5,8),C2(4,9)

Cluster2:A3(8,4),B2(7,5),B3(7,5)

Cluster 3: A2(8,4),C1(1,2)

INTERPRETING DESCRIPTIVE MODELS

Standard Cluster Model Interpretation:

Descriptive models are used to discover patterns or structure in data without a predefined target variable. One common descriptive technique is **clustering**, which groups similar observations together.

Standard cluster model interpretation refers to the process of understanding, labeling, and explaining the characteristics of each cluster after a clustering algorithm has been applied.

Purpose of Cluster Interpretation

After clustering is completed, the algorithm only provides:

- Cluster numbers (Cluster 1, Cluster 2, etc.)
- Membership of observations

However, the real value comes from interpreting:

- What each cluster represents
- How clusters differ from each other
- What business meaning can be assigned

Interpretation transforms mathematical groups into actionable insights.

Steps in Standard Cluster Model Interpretation

1. **Examine Cluster Profiles**
   Calculate the mean (for numeric variables) and distribution (for categorical variables) within each cluster to understand its characteristics.
2. **Compare Clusters**
   Identify how clusters differ from one another by analyzing differences in key variables.
3. **Identify Key Differentiating Variables**
   Focus on variables that show significant variation across clusters, as they help explain cluster separation.
4. **Label the Clusters**
   Assign meaningful and descriptive names to clusters based on their dominant characteristics.

5. **Check Cluster Size**

   Evaluate the number of observations in each cluster to understand their relative importance.

6. **Validate Business Relevance**

   Ensure clusters are logical, interpretable, and useful for decision-making.

REFERENCES:

TEXTBOOK:

1. Dean Abbott, Applied Predictive Analytics, Published by Jhon Wiley &Sons, Inc,2014.

ONLINE REFERENCES:

https://www.educative.io/answers/what-is-association-rule-mining

https://www.geeksforgeeks.org/machine-learning/apriori-algorithm/

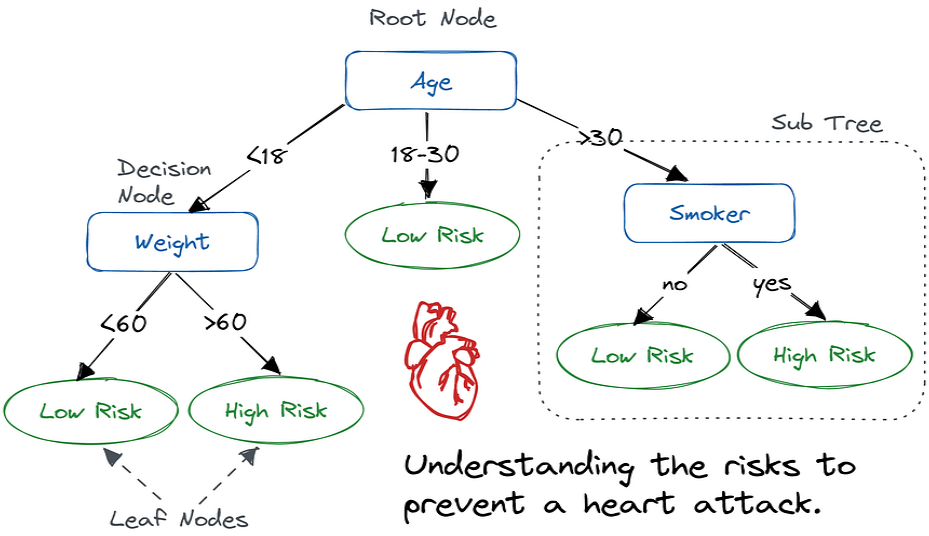
https://www.studocu.com/in/document/savitribai-phule-pune-university/machine-learning/kmean-assignment-we-have-given-a-collection-of-8-points-p10106-p2015071-p300809/26213029

Decision Tree

A **Decision Tree** is a supervised machine learning technique used for:

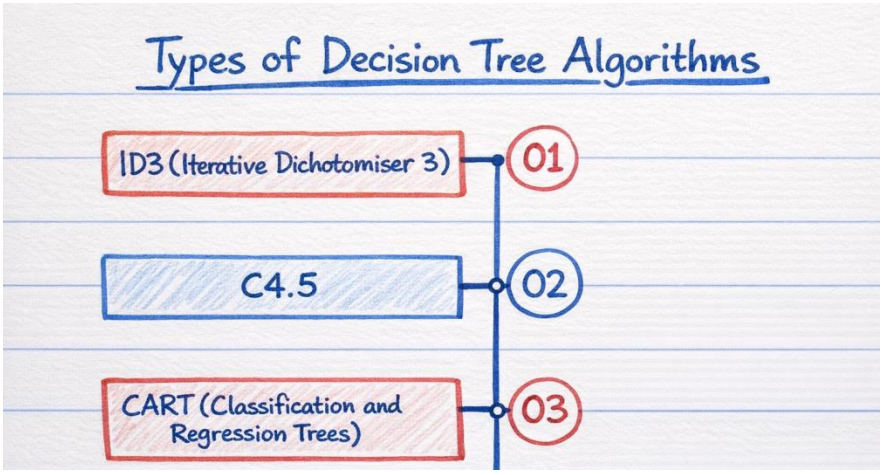
- Classification (Yes/No, Pass/Fail)
- Regression (Predict numerical value)

It represents decisions in a **tree-like structure**.



Parts of Decision Tree:

1. **Root Node**
   - Topmost node
   - Represents best attribute
2. **Internal Node**
   - Decision based on condition
3. **Branches**
   - Outcome of a decision
4. **Leaf Node (Terminal Node)**
   - Final prediction (Yes/No)



1) ID3 Algorithm	2) C4.5 Algorithm	3) CART Algorithm
• Developed by Ross Quinlan • Uses Entropy and Information Gain • Works only with categorical data • Produces multi-branch trees	• Extension of ID3 • Handles continuous & categorical data • Uses Gain Ratio • Supports pruning (reduces overfitting)	• Produces binary trees • Uses Gini Index • Supports classification & regression • Used in Scikit-learn

Entropy: Entropy (S) is a measure of the uncertainty or randomness in a set of data.

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

Where:

S = Dataset

c = Number of classes

p_i =Proportion of class i

For binary classification:

$$Entropy(S) = -p_1 \log_2(p_1) - p_2 \log_2(p_2)$$

Information Gain

Information Gain is a measure used in machine learning, particularly in the construction of decision trees, to determine the effectiveness of an attribute in classifying data.

$$Information\ Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Where:

- S = Parent dataset
- A = Attribute
- S_v = Subset after split on value
- $|S_v|/|S|$ = Weight of subset

Gain Ratio

The **gain ratio** is an attribute selection measure used in the C4.5 decision tree algorithm to address the bias of Information Gain.

$$Gain\ Ratio(S, A) = \frac{Information\ Gain(S, A)}{SplitInfo(S, A)}$$

$$SplitInfo(S, A) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \left(\frac{|S_v|}{|S|} \right)$$

Gini Index (Gini Impurity) :

Gini Impurity is a metric used in decision tree algorithms to measure the frequency at which a randomly chosen element from a set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset.

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

For binary classification:

$$Gini(S) = 1 - (p_1^2 + p_2^2) \qquad \text{(or)} \qquad Gini(S) = 2p_1p_2$$

Problem:

Using the following dataset, construct a Decision Tree using the ID3 algorithm to predict whether a person will Play Tennis or not.

Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Total = 14 records
Yes = 9
No = 5

Entropy of Entire Dataset:
S = − p(Yes) log2 p(Yes) − p(No) log2 p(No)

p(Yes) = 9/14
p(No) = 5/14

Entropy(S)

= − (9/14) log2(9/14) − (5/14) log2(5/14)

= − (0.643 × −0.64) − (0.357 × −1.48)

= 0.940

Entropy(S) = 0.940

Step 2 – Information Gain of Each Attribute:

A) Information Gain (Outlook):

Outlook values:

Sunny (5 records → 2 Yes, 3 No)
Overcast (4 records → 4 Yes, 0 No)
Rain (5 records → 3 Yes, 2 No)

Entropy(Sunny)	Entropy(Overcast)	Entropy(Rain)
= −(2/5) log2(2/5) − (3/5) log2(3/5) = 0.971	= 0 (pure node)	= −(3/5) log2(3/5) − (2/5) log2(2/5) = 0.971

Weighted Entropy: outlook

$$Gain(S, A) = Entropy(S) - \sum [\, p(S|A) \,.\, Entropy(S|A) \,]$$

= (5/14 × 0.971) + (4/14 × 0) + (5/14 × 0.971)

= 0.693

Information Gain: entropy(S)-entropy(outlook)

IG(Outlook) = 0.940 − 0.693
= 0.247

B)Information Gain (Temp):

Outlook values:

Hot → 4 records (2 Yes, 2 No)
Mild → 6 records (4 Yes, 2 No)
Cool → 4 records (3 Yes, 1 No)

Entropy(Hot)	Entropy(Mild)	Entropy(Cool)
p(Yes) = 2/4 p(No) = 2/4 = − (2/4 log2 2/4) − (2/4log2 2/4) = − (0.5 × −1) − (0.5 × −1) = 1.0	p(Yes) = 4/6 p(No) = 2/6 = − (4/6 log2 4/6) − (2/6 log2 2/6) = 0.918	p(Yes) = 3/4 p(No) = 1/4 = − (3/4 log2 3/4) − (1/4 log2 1/4) = 0.811

Weighted Entropy: Temp

$$Gain(S, A) = Entropy(S) - \sum [\, p(S|A) \,.\, Entropy(S|A) \,]$$

= (4/14 × 1.0) + (6/14 × 0.918) + (4/14 × 0.811)

= 0.693

Information Gain: entropy(S)-entropy(outlook)

IG(Temp) = 0.940 − 0.911
= 0.0289

c)Information Gain (Humidity):

Humidity values:
High → 7 records (3 Yes, 4 No)
Normal → 7 records (6 Yes, 1 No)

Entropy(High)	**Entropy(Normal)**
p(Yes) = 3/7 p(No) = 4/7 $= -(3/7 \log_2 3/7) - (4/7 \log_2 4/7)$ $= -(0.429 \times -1.222) - (0.571 \times -0.807)$ $= 0.985$	p(Yes) = 6/7 p(No) = 1/7 $= -(6/7 \log_2 6/7) - (1/7 \log_2 1/7)$ $= -(0.857 \times -0.222) - (0.143 \times -2.807)$ $= 0.591$

Weighted Entropy: Humidity

$= (7/14 \times 0.985) + (7/14 \times 0.591)$

$= (0.5 \times 0.985) + (0.5 \times 0.591)$

$= 0.492 + 0.296$

$= 0.788$

Information Gain: entropy(S) – entropy(Humidity)

IG(Humidity) = 0.940 − 0.788

= 0.1516

c)Information Gain (Wind):

Wind values:
Weak → 8 records (6 Yes, 2 No)
Strong → 6 records (3 Yes, 3 No)

Entropy(Weak)	**Entropy(Strong)**
p(Yes) = 6/8 p(No) = 2/8	p(Yes) = 3/6 p(No) = 3/6

$= -(6/8 \log_2 6/8) - (2/8 \log_2 2/8)$ $= -(0.75 \times -0.415) - (0.25 \times -2)$ $= 0.811$	$= -(3/6 \log_2 3/6) - (3/6 \log_2 3/6)$ $= -(0.5 \times -1) - (0.5 \times -1)$ $= 1.0$

Weighted Entropy: Wind

$= (8/14 \times 0.811) + (6/14 \times 1.0)$

$= (0.571 \times 0.811) + (0.429 \times 1.0)$

$= 0.463 + 0.429$

$= 0.892$

Information Gain: entropy(S) − entropy(Wind)

$IG(Wind) = 0.940 - 0.892$

$= 0.0478$

PA – UNIT V COMPLETE PDF

1. MODEL ENSEMBLES

1.1 Motivation for Ensembles

Machine learning models are often imperfect. A single model can suffer from high bias (underfitting) or high variance (overfitting), leading to poor generalization on unseen data. Ensemble methods address this limitation by combining multiple models — often called 'weak learners' or 'base learners' — into a single, more powerful 'strong learner'.

Why Use Ensembles?

- Reduced Variance: Multiple models trained on different subsets of data or with different parameters tend to make different errors. Averaging their predictions reduces the overall variance.
- Reduced Bias: By combining many slightly biased models, the overall ensemble can achieve a lower bias than any single model.
- Improved Accuracy: Consistently, ensemble methods outperform individual models on benchmark datasets and competitions (e.g., Kaggle).
- Greater Robustness: Ensembles are less sensitive to noise and outliers because the combined output dilutes the impact of any single noisy model.

The Wisdom of the Crowd

The core concept behind ensembles is rooted in the 'wisdom of the crowd' principle. Just as a group of diverse, independent people can collectively make better decisions than a single expert, a diverse collection of models can collectively make better predictions than any single model. Diversity among base learners is the most critical requirement — models that make similar errors provide little benefit when combined.

Conditions for an Effective Ensemble

- Accuracy: Each base learner should perform better than random guessing (accuracy > 50% for binary classification).
- Diversity: The base learners should make different errors on different inputs. If all models make the same mistakes, the ensemble cannot improve upon them.
- Independence: Ideally, the models should be trained somewhat independently so their errors are uncorrelated.

Types of Ensemble Methods

- Bagging (Bootstrap Aggregating): Trains multiple models in parallel on different random subsets of training data.

- Boosting: Trains models sequentially, with each new model focusing on the errors made by previous models.
- Stacking: Trains a meta-model that learns to combine the outputs of multiple base models.
- Voting/Averaging: Combines predictions using majority voting (classification) or averaging (regression).

KEY TAKEAWAYS

- Ensemble methods combine multiple models to produce better predictions than any single model.
- The key is diversity — base learners must make different types of errors.
- Ensembles reduce variance, bias, or both, depending on the method used.
- The three main ensemble strategies are Bagging, Boosting, and Stacking.

1.2 Bagging (Bootstrap Aggregating)

Bagging, proposed by Leo Breiman in 1994, is one of the earliest and most widely used ensemble techniques. The term stands for Bootstrap Aggregating and involves creating multiple versions of a predictor by training on bootstrap samples of the training data, then combining their outputs.

The Bootstrap Sampling Process

A bootstrap sample is created by randomly sampling the original training dataset with replacement. This means:

- Each bootstrap sample has the same size as the original dataset.
- Some training instances may appear multiple times in a bootstrap sample.
- Some training instances may not appear at all (these are called 'out-of-bag' samples, approximately 37% of the original data).

Out-of-bag (OOB) samples serve as a built-in validation set, allowing performance estimation without a separate validation split.

How Bagging Works Step by Step

1. Given training dataset D with N examples and a learning algorithm L.
2. Generate B bootstrap samples: D_1, D_2, ..., D_n by sampling N instances with replacement from D.
3. Train a separate base model M_i on each bootstrap sample D_i.
4. For a new input x, obtain prediction from each model: $M_1(x)$, $M_2(x)$, ..., $M_n(x)$.
5. Combine predictions: use majority vote for classification, or average for regression.

Aggregation Methods

- Majority Voting (Classification): The class predicted by the most models is chosen as the final prediction. Example: if 7 out of 10 models predict 'spam', the final prediction is 'spam'.
- Weighted Voting: Models with higher accuracy on OOB samples may be given more weight.
- Averaging (Regression): The final prediction is the arithmetic mean of all model outputs. This reduces variance significantly.
- Median: Sometimes used instead of mean to reduce the influence of extreme predictions.

Random Forest The Most Famous Bagging Variant

Random Forest extends bagging for decision trees by introducing an additional source of randomness: at each node split, only a random subset of features (typically $\sqrt{m}$ for classification, m/3 for regression, where m is total features) is considered. This creates more diverse trees and further reduces correlation between models.

- Advantages of Random Forest: Handles high-dimensional data, robust to overfitting, handles missing values, provides feature importance scores.
- Disadvantage: Harder to interpret than a single decision tree.

Why Bagging Works Mathematical Intuition

If B independent models each have variance σ^2, their average has variance σ^2/B. While bootstrap samples are not fully independent (they share ~63% of the original data), bagging still significantly reduces variance. Bagging is most effective when base learners have high variance and low bias (e.g., deep, unpruned decision trees).

KEY TAKEAWAYS
• Bagging creates B bootstrap samples and trains a model on each, then aggregates predictions. • Approximately 37% of the original data is excluded from each bootstrap sample (OOB data), used for validation.
• Bagging reduces variance — most useful for high-variance, low-bias models like deep decision trees.
• Random Forest extends bagging with random feature selection at each node for greater diversity.

1.3 Boosting

Boosting is an ensemble technique where models are trained sequentially, with each model focusing on the mistakes made by the previous models. Unlike bagging (which trains models in parallel), boosting builds an additive model iteratively.

Core Idea of Boosting

Initially, all training examples are given equal weight. After each model is trained, misclassified examples are given higher weight so that the next model pays more attention to difficult cases. The final prediction combines all models, typically using a weighted vote or sum.

AdaBoost (Adaptive Boosting)

AdaBoost, introduced by Freund and Schapire (1997), is the classic boosting algorithm. It works as follows:

6. Initialize weights: $w_i = 1/N$ for all N training examples.
7. For t = 1, 2, ..., T iterations: Train a weak classifier h_t on weighted data.
8. Compute weighted error: $\varepsilon_t = \Sigma\, w_i \cdot I(y_i \neq h_t(x_i))$ — sum of weights of misclassified examples.
9. Compute model weight: $\alpha_t = 0.5 \times \ln((1 - \varepsilon_t)/\varepsilon_t)$ — models with lower error get higher weight.
10. Update example weights: increase weights for misclassified examples, decrease for correct ones.
11. Normalize weights so they sum to 1.
12. Final prediction: $H(x) = \text{sign}(\Sigma\, \alpha_t \cdot h_t(x))$.

Gradient Boosting

Gradient Boosting generalizes the boosting concept by framing it as gradient descent in function space. Each new model is trained to predict the residual errors (pseudo-residuals) of the previous ensemble.

- Loss Function: Can be customized — mean squared error for regression, log-loss for classification.
- Learning Rate (η): Shrinks the contribution of each new tree, allowing more trees to be added without overfitting.
- Key Hyperparameters: Number of trees (n_estimators), tree depth (max_depth), learning rate (η), subsampling fraction.

XGBoost, LightGBM, and CatBoost

These are highly optimized implementations of gradient boosting that dominate modern machine learning competitions:

- XGBoost (eXtreme Gradient Boosting): Introduced regularization (L1 and L2) to the loss function to prevent overfitting. Supports parallel processing by computing gain for splits simultaneously. Handles missing values internally.
- LightGBM: Uses 'leaf-wise' tree growth instead of 'level-wise', growing the leaf with the highest gain. Much faster than XGBoost for large datasets. Uses histogram-based algorithms for efficient split finding.
- CatBoost: Designed specifically to handle categorical features without manual encoding. Uses ordered boosting to reduce prediction shift bias.

Differences Between Bagging and Boosting

BAGGING	BOOSTING
Models trained in parallel	Models trained sequentially
Reduces variance	Reduces bias and variance
Each model trained independently	Each model depends on previous
Less prone to overfitting	Can overfit if too many iterations
Example: Random Forest	Example: AdaBoost, XGBoost

KEY TAKEAWAYS

- Boosting trains models sequentially, each focusing on errors of the previous model.
- AdaBoost adjusts sample weights — misclassified examples get higher weight.
- Gradient Boosting fits new models to the residual errors using gradient descent.
- XGBoost, LightGBM, CatBoost are modern implementations that dominate real-world applications.

1.4 Improvements to Bagging and Boosting

Researchers have proposed numerous improvements and extensions to the basic bagging and boosting algorithms to address their shortcomings and adapt them to specific use cases.

Improvements to Bagging

- Feature Randomization (Random Subspaces / Random Forest): Rather than using all features, each base model trains on a random subset of features. This increases diversity and reduces correlation between trees.

- Pasting: Similar to bagging but without replacement in sampling. Can work well when data is abundant and diversity can be achieved without repetition.
- Random Patches: Combines random sampling of both training instances and features, providing maximum diversity.
- Extra-Trees (Extremely Randomized Trees): Further randomizes by selecting both the feature and the split threshold randomly, making trees even more diverse and faster to train.
- Out-of-Bag Evaluation: Using OOB samples for performance estimation eliminates the need for a separate validation set, making more efficient use of training data.

Improvements to Boosting

- Regularization in Gradient Boosting: Adding L1 (Lasso) and L2 (Ridge) penalties to the objective function (as in XGBoost) controls model complexity and reduces overfitting.
- Shrinkage / Learning Rate: Multiplying each tree's contribution by a small learning rate (e.g., 0.01–0.1) and compensating with more trees. This 'shrinkage' technique is highly effective and considered one of the most important improvements.
- Subsampling (Stochastic Gradient Boosting): Introduced by Friedman, training each tree on a random subsample of the training data (similar to bagging within boosting). This introduces variance which helps avoid overfitting and also speeds up training.
- Column (Feature) Subsampling: Only using a random subset of features for each tree, as done in XGBoost's colsample_bytree parameter.
- Tree Pruning and Minimum Gain: Setting a minimum gain threshold for splits prevents unnecessary splits that add complexity without improving accuracy.
- Early Stopping: Monitoring performance on a validation set and stopping when performance stops improving, preventing overfitting.
- Leaf-Wise Growth (LightGBM): Rather than growing all leaves of a tree level by level, LightGBM grows only the leaf with the highest gain, creating asymmetric trees that are more accurate with fewer trees.

Handling Class Imbalance in Ensembles

- Scale Position Weight (XGBoost): Weights the positive class more heavily during gradient computation.
- SMOTEBoost: Combines synthetic minority oversampling (SMOTE) with boosting to handle severely imbalanced datasets.
- EasyEnsemble and BalanceCascade: Specific ensemble approaches designed for imbalanced learning that undersample the majority class strategically.

KEY TAKEAWAYS

- Extra-Trees and Random Patches extend bagging by randomizing features and thresholds.
- Shrinkage (learning rate) and subsampling are the most impactful improvements to boosting.
- Early stopping prevents overfitting in boosting by monitoring validation performance.
- Leaf-wise tree growth in LightGBM makes it significantly faster than level-wise approaches.

1.5 Model Ensembles and Occam's Razor

Occam's Razor is a philosophical principle attributed to William of Ockham (14th century) that states: 'Among competing hypotheses, the one with the fewest assumptions should be selected.' In machine learning, this principle is often interpreted as: prefer simpler models over complex ones when they perform equally well.

The Tension Between Ensembles and Occam's Razor

On the surface, ensemble methods appear to violate Occam's Razor. A Random Forest with 500 trees or an XGBoost model with 1000 boosting rounds is far more complex than a single decision tree. This raises a fundamental question: are ensembles 'better' models, or do they just win by being more complex?

Arguments Against Ensembles from Occam's Razor

- Interpretability: A single decision tree can be visualized and its reasoning explained. A 500-tree Random Forest cannot be easily interpreted — you cannot trace the path from input to output.
- Computational Cost: Ensembles require significantly more memory and computation for both training and inference.
- Maintenance: In production systems, simpler models are easier to update, debug, and maintain.

Counterarguments: When Complexity is Justified

- Predictive Performance: If the goal is maximum predictive accuracy (e.g., in a medical diagnosis system), a complex ensemble that outperforms a simple model is justified.
- Structural Complexity vs. Description Length: Occam's Razor as used in MDL (Minimum Description Length) theory favors models that compress the data best and ensembles sometimes achieve a better compression of the underlying data distribution than simple models.

- Generalization vs. Memorization: Ensembles often generalize better than single complex models, suggesting their complexity captures genuine structure rather than noise.
- Implicit Regularization in Bagging: Combining many high-variance models through bagging often produces a smoother, simpler decision boundary than the individual trees — the ensemble is effectively simpler in terms of functional form even if it consists of many components.

Practical Guidance

In practice, the choice between a simple interpretable model and a complex ensemble depends on the specific application context:

- If interpretability is crucial (e.g., loan approvals, medical decisions, legal contexts): prefer simpler, explainable models.
- If accuracy is the primary goal and interpretability is secondary: ensembles are generally superior.
- SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) can partially address the interpretability gap in ensembles.

KEY TAKEAWAYS

• Occam's Razor favors simpler models, but ensembles often justify their complexity through superior performance.

• Bagging can produce simpler effective decision boundaries despite using many models.

• The choice between simplicity and complexity depends on whether the primary goal is accuracy or interpretability.

• Tools like SHAP and LIME help make ensemble predictions more interpretable.

1.6 Interpreting Model Ensembles

Interpretability is one of the most significant challenges with ensemble methods. While they often achieve superior predictive performance, understanding why they make specific predictions is difficult. Several techniques have been developed to address this limitation.

Feature Importance

One of the most accessible interpretability tools for ensembles is feature importance, which quantifies how much each input feature contributes to predictions.

- Gini Importance (Mean Decrease in Impurity): For tree-based ensembles, sums the impurity reduction achieved by each feature across all trees and all splits. Fast to compute but biased toward high-cardinality features.

- Permutation Importance: Randomly shuffles one feature's values and measures the decrease in model performance. Features that are important will cause a large performance drop when shuffled. Less biased than Gini importance but computationally expensive.
- Drop-Column Importance: Trains the model with each feature removed and measures performance change. Most accurate but extremely slow.

Partial Dependence Plots (PDPs)

PDPs show the marginal effect of one or two features on the predicted outcome, averaging over all other features. They help answer questions like 'How does increasing age affect the predicted probability of loan default?'

- 1D PDP: Shows the relationship between one feature and the predicted output.
- 2D PDP: Shows the interaction effect between two features. Becomes difficult to visualize with more features.
- Limitation: PDPs assume features are independent, which may not hold in practice. Accumulated Local Effects (ALE) plots address this limitation.

SHAP (SHapley Additive exPlanations)

SHAP is currently the most principled framework for explaining individual predictions from ensemble models. Based on game theory's Shapley values, SHAP attributes the prediction of a model to each feature.

- SHAP values tell you: For this specific prediction, how much did each feature push the prediction higher or lower than the average prediction?
- TreeSHAP: An efficient algorithm for computing SHAP values for tree-based models in polynomial time.
- Global SHAP Summary Plots: Show the distribution of SHAP values across the dataset, combining feature importance and effect direction.
- SHAP Dependence Plots: Show how SHAP values for a feature vary with feature values, potentially revealing nonlinear effects.

LIME (Local Interpretable Model-agnostic Explanations)

LIME explains individual predictions by training a simple, interpretable model (like a linear model) locally around the specific prediction point. Steps:

13. Select the prediction to explain.
14. Generate synthetic data points by perturbing the input.
15. Get predictions from the complex ensemble for these synthetic points.
16. Fit a simple linear model to these synthetic points, weighted by their proximity to the original input.
17. Use the linear model's coefficients as the explanation.

Decision Rules and Surrogate Models

- Global Surrogate Models: Train a simple, interpretable model (e.g., decision tree) to mimic the predictions of the ensemble. The surrogate model can then be analyzed to understand the ensemble's behavior.
- Decision Rules Extraction: Extract the most important decision rules from the ensemble for human review.

KEY TAKEAWAYS
• Feature importance (Gini, permutation) identifies which features most influence ensemble predictions.
• Partial Dependence Plots (PDPs) show the marginal effect of individual features on predictions.
• SHAP provides theoretically sound, consistent feature attributions for individual predictions.
• LIME approximates complex ensemble predictions locally with simple interpretable models.

2. TEXT MINING

2.1 Motivation for Text Mining

Text mining (also called text analytics or knowledge discovery from text) refers to the process of deriving high-quality information and patterns from large collections of unstructured text data. It applies machine learning, statistical analysis, and natural language processing (NLP) to extract meaningful insights from text.

The Data Explosion

We live in an era of unprecedented data generation. The vast majority of data generated today — estimated at 80–90% of all data — is unstructured, with text being the most common form:

- Social media posts (Twitter, Facebook, LinkedIn)
- Customer reviews and feedback
- News articles and web content
- Scientific publications and research papers
- Legal documents and contracts
- Email and internal communications
- Medical records and clinical notes

Business Value of Text Mining

- Sentiment Analysis: Understanding customer opinions about products, services, or brands from reviews and social media.
- Topic Modeling: Automatically discovering themes in large document collections (e.g., trending topics in news).
- Information Extraction: Automatically extracting structured information (names, dates, amounts) from unstructured documents.
- Document Classification: Automatically categorizing documents (spam detection, news categorization, support ticket routing).
- Recommendation Systems: Using text from product descriptions and reviews to improve recommendations.
- Risk Management: Mining compliance documents, legal filings, and news for risk signals.

Applications Across Industries

- Healthcare: Mining clinical notes for disease patterns, adverse drug reactions, and treatment outcomes.
- Finance: Analyzing earnings call transcripts, news, and social media for investment signals.
- E-commerce: Analyzing product reviews to improve product quality and customer experience.
- Customer Service: Automatically categorizing and routing support tickets, identifying common customer issues.
- Law: Contract analysis, legal discovery, and compliance monitoring.

KEY TAKEAWAYS
• 80-90% of all business data is unstructured text, making text mining a critical business capability.
• Text mining combines NLP, machine learning, and statistics to extract insights from text.
• Key applications include sentiment analysis, topic modeling, classification, and information extraction.
• Text mining enables automated processing of volumes of text that would be impossible to analyze manually.

2.2 A Predictive Modelling Approach to Text Mining

Predictive modelling in text mining applies supervised machine learning to predict a target variable (label) from text features. This transforms the text mining problem into a standard classification or regression problem.

The General Framework

18. Define the prediction task (e.g., predict whether a review is positive or negative).
19. Collect and label a training dataset of documents.
20. Preprocess and clean the text data.
21. Extract features from the text (convert text to numerical representation).
22. Train a machine learning model on the extracted features.
23. Evaluate the model on held-out test data.
24. Deploy the model to make predictions on new text.

Target Variable Types in Text Mining

- Binary Classification: Spam/not-spam, positive/negative sentiment, relevant/irrelevant document.
- Multi-class Classification: News category (sports, politics, business), product category, customer intent.
- Regression: Predicting a rating score (1–5 stars), predicting document quality score.
- Multi-label Classification: Assigning multiple tags to a document (a blog post can be both 'technology' and 'business').

The Bag-of-Words Model

The most common approach to representing text for predictive modelling is the Bag-of-Words (BoW) model. In BoW, a document is represented as an unordered collection of words — their frequencies are recorded, but their positions and grammar are ignored.

- Vocabulary: The set of all unique words (terms) in the training corpus.
- Document-Term Matrix: A matrix where rows are documents, columns are terms, and each cell contains the count or frequency of that term in that document.
- Limitation: BoW loses word order and context (e.g., 'not good' and 'good not' are treated identically).

Beyond Bag-of-Words: Advanced Representations

- N-grams: Sequences of N consecutive words (bigrams: 'not good', trigrams: 'I am happy'). Partially preserves word order.
- TF-IDF: Term Frequency-Inverse Document Frequency — weights words higher if they are frequent in a document but rare across all documents.
- Word Embeddings (Word2Vec, GloVe, FastText): Dense vector representations where semantically similar words have similar vectors. Captures semantic relationships.
- Transformer-based Models (BERT, GPT): State-of-the-art contextual representations where the same word can have different representations depending on context.

KEY TAKEAWAYS

- Predictive text mining follows the standard ML pipeline: collect → preprocess → featurize → train → evaluate → deploy.
- The Bag-of-Words model is the simplest and most widely used text representation.
- TF-IDF improves BoW by accounting for term rarity across the corpus.
- Modern transformer-based models (BERT, GPT) provide contextual representations and achieve state-of-the-art results.

2.3 Structured vs. Unstructured Data

Understanding the distinction between structured and unstructured data is fundamental to text mining. Each type has different characteristics, storage requirements, and analytical techniques.

Structured Data

Structured data is organized in a predefined format, typically stored in relational databases or spreadsheets. Each data point fits into a defined field with a specific data type.

- Characteristics: Predefined schema, easily searchable with SQL, each record has same fields, data types are consistent (integer, float, date, boolean, string with max length).
- Examples: Customer databases (name, age, income, purchase history), financial records, sensor readings, transaction logs.
- Analysis: Standard machine learning algorithms apply directly after basic preprocessing. Statistical methods and SQL queries work efficiently.
- Tools: Relational databases (MySQL, PostgreSQL), Excel, Pandas, SQL.

Unstructured Data

Unstructured data has no predefined format or organization. It is the most abundant form of data and cannot be easily stored in traditional databases.

- Characteristics: No fixed schema, highly variable format and length, requires specialized processing before analysis.
- Examples: Email messages, social media posts, word processing documents, PDF files, audio recordings, images, video, web pages.
- Analysis: Requires NLP for text, computer vision for images, speech recognition for audio — specialized techniques before ML can be applied.
- Tools: NLTK, spaCy, Hugging Face Transformers, OpenCV, TensorFlow/PyTorch.

Semi-Structured Data

A middle ground exists in the form of semi-structured data, which has some organizational properties but does not conform to a strict relational schema:

- Examples: JSON files, XML documents, HTML web pages, CSV files with variable-length fields, email headers (structured) + body (unstructured).
- Characteristics: Self-describing structure (tags, keys), more flexible than structured but more organized than pure unstructured.

Why the Distinction Matters for Text Mining

Dimension	Structured	Unstructured (Text)
Format	Fixed schema	Variable, free-form
Storage	Relational DB, table	Files, NoSQL, document stores
Query Method	SQL, exact queries	Full-text search, NLP
Preprocessing	Minimal (normalize, impute)	Extensive (tokenize, stem, vectorize)

KEY TAKEAWAYS
• Structured data has a fixed schema and is stored in relational databases; directly usable by ML algorithms.
• Unstructured data (text, images, audio) has no fixed format and requires specialized preprocessing.
• Semi-structured data (JSON, XML) is a middle ground with some organizational properties.
• Text mining bridges unstructured text and structured ML inputs through feature engineering.

2.4 Why Text Mining Is Hard

Natural language is the primary medium of human communication, and it is extraordinarily complex. Text mining faces unique challenges that do not arise (or are less severe) with structured numerical data.

Linguistic Challenges

- Ambiguity: Language is inherently ambiguous at multiple levels. 'Bank' can mean a financial institution or a river bank. 'I saw the man with the telescope' — who has the telescope? Words, phrases, and sentences can have multiple valid interpretations depending on context.

- Polysemy: A single word has multiple related meanings ('crane': bird, construction equipment, to stretch one's neck).
- Synonymy: Multiple words have the same or similar meanings ('car', 'automobile', 'vehicle', 'wheels'). A keyword search for 'car' will miss documents using 'automobile'.
- Anaphora and Coreference: Pronouns and references that point to previously mentioned entities ('John went to the store. He bought milk.' — resolving 'He' = 'John' is complex).
- Idioms and Figurative Language: 'It's raining cats and dogs' means heavy rain, not a literal event. Sarcasm and irony mean the opposite of what is literally said.
- Negation: 'The movie was not bad at all' is positive, but a simple keyword analysis might classify it as negative due to 'bad'.

Preprocessing Challenges

- Spelling Errors and Non-standard Language: Social media text is rife with abbreviations ('lol', 'brb'), misspellings, and slang that standard models struggle with.
- Multilingual Text: Documents may contain multiple languages, or use code-switching (mixing languages mid-sentence).
- Evolving Language: New words, slang, and meanings emerge constantly. Models trained on old data may not recognize new terms.
- Domain-Specific Vocabulary: Medical, legal, and technical texts use specialized terminology not common in everyday language.

High-Dimensionality Challenge

Even a medium-sized text corpus can have a vocabulary of hundreds of thousands of unique words. This creates a very high-dimensional feature space where:

- The document-term matrix is extremely sparse (most documents use only a tiny fraction of all vocabulary words).
- The curse of dimensionality applies — algorithms may need exponentially more data as dimensionality increases.
- Dimensionality reduction (PCA, LSA, word embeddings) is necessary but can lose interpretability.

Semantic and Contextual Challenges

- Context Dependence: The meaning of 'apple' in a sentence about computers is different from one about food. Word meaning is context-dependent.
- Discourse Structure: Understanding documents requires understanding how sentences and paragraphs relate to each other, not just individual sentences in isolation.
- World Knowledge: Many text interpretation tasks require background knowledge that is not present in the text itself (common sense reasoning).

KEY TAKEAWAYS

• Language ambiguity (polysemy, synonymy, negation, idioms) makes text understanding fundamentally difficult.

• High dimensionality from large vocabularies creates sparse, unwieldy feature matrices.

• Non-standard language (social media, domain-specific text) requires specialized preprocessing.

• Contextual meaning requires models that look beyond individual words to their surrounding context.

2.5 Data Preparation Steps

Data preparation (preprocessing) is perhaps the most time-consuming and impactful step in text mining. The quality of the preprocessing pipeline largely determines the quality of the final model. The typical text preprocessing pipeline includes the following steps:

Step 1: Data Collection and Acquisition

- Web Scraping: Automated extraction of text from websites using tools like BeautifulSoup, Scrapy, or Selenium.
- APIs: Accessing data from social media platforms (Twitter API, Reddit API), news sources, or databases.
- Manual Collection: Surveys, transcripts, proprietary documents.
- Public Datasets: Many text mining problems can leverage public datasets (IMDb reviews, Reuters news corpus, Wikipedia, Common Crawl).

Step 2: Text Extraction and Cleaning

- Format Conversion: Extracting plain text from PDFs, Word documents, HTML, or XML. Tools: pdfminer, python-docx, BeautifulSoup.
- Noise Removal: Removing HTML tags, special characters, URLs, email addresses, phone numbers, and other non-informative elements.
- Encoding Normalization: Converting all text to a consistent encoding (UTF-8) and handling special characters (accents, Unicode symbols).
- Language Detection: Identifying and filtering documents by language if working with multilingual corpora.

Step 3: Tokenization

Tokenization is the process of splitting text into individual tokens (words or sub-words). This seems simple but has many edge cases:

- Word Tokenization: Splitting text by whitespace and punctuation. Edge cases: 'New York', 'U.S.A.', 'don't' (should these be split?).
- Sentence Tokenization: Splitting text into sentences. Edge cases: abbreviations ('Dr. Smith went to Washington.').
- Sub-word Tokenization (BPE, WordPiece): Used by modern transformer models, splits rare words into frequent sub-word units. Handles out-of-vocabulary words gracefully.

Step 4: Lowercasing

Converting all text to lowercase so that 'Apple', 'apple', and 'APPLE' are treated as the same word. This reduces vocabulary size and improves term matching. However, case can be meaningful in some contexts (e.g., distinguishing the company 'Apple' from the fruit 'apple' or the acronym 'US' from the word 'us').

Step 5: Stop Word Removal

Stop words are common words that carry little semantic meaning and occur in almost every document (e.g., 'the', 'a', 'is', 'in', 'and', 'of'). Removing them reduces feature space dimensionality without significantly losing meaning.

- Pre-built stop word lists are available in NLTK, spaCy, and other libraries.
- Domain-specific stop words may need to be added (e.g., in legal text, 'whereas', 'hereby', 'therein' are very common).
- Caution: In some tasks (e.g., authorship attribution), stop word patterns are highly informative and should not be removed.

Step 6: Stemming and Lemmatization

These techniques normalize words to their base form so that variations of the same word are treated as one:

- Stemming: Crudely removes word endings using heuristic rules. 'running', 'runs', 'runner' → 'run'. Fast but may produce non-words ('studies' → 'studi'). Popular algorithms: Porter Stemmer, Snowball Stemmer.
- Lemmatization: Reduces words to their dictionary base form using morphological analysis and a vocabulary. 'running' → 'run', 'better' → 'good', 'children' → 'child'. More accurate than stemming but slower and requires a lexicon.

Step 7: Handling Rare and Frequent Terms

- Very Rare Terms: Words appearing in only 1–2 documents are statistically unreliable and often just noise. Removing terms with document frequency below a threshold (e.g., min_df=5) reduces vocabulary size.
- Very Frequent Terms: Words appearing in almost every document (beyond stop words) provide little discriminating power. Setting a max_df threshold (e.g., max_df=0.95) removes these.

Step 8: Feature Extraction — Converting Text to Numbers

- Count Vectorization (Bag-of-Words): Each document is represented as a vector of word counts.
- TF-IDF Vectorization: Word counts are weighted by their inverse document frequency. TF(t,d) = count of term t in document d / total terms in d. IDF(t) = log(N / df(t)), where N is total documents and df(t) is the number containing term t.
- Word Embeddings: Pre-trained or task-trained dense vector representations (Word2Vec, GloVe, FastText). A document embedding can be the average of its word embeddings.
- Transformer Embeddings: Contextual embeddings from BERT, RoBERTa, or GPT models. Represents each word based on its full context.

KEY TAKEAWAYS
• The preprocessing pipeline typically includes: extract → clean → tokenize → lowercase → remove stop words → stem/lemmatize → vectorize.
• Lemmatization is more accurate than stemming but slower; choice depends on application requirements.
• TF-IDF is the standard baseline vectorization method; embeddings are needed for semantic tasks.
• Stop word removal and handling rare/frequent terms significantly reduce feature dimensionality.

2.6 Text Mining Features

Feature engineering is the process of extracting informative numerical features from raw text. The quality and selection of features directly determines the performance of text mining models.

Lexical Features

Lexical features are based on the words and their frequency statistics:

- Term Frequency (TF): Raw count of how often a term appears in a document. Simple but doesn't account for document length.
- Normalized TF: Term count divided by total terms in the document, accounting for document length.
- TF-IDF Score: Balances term frequency with term rarity across the corpus. High TF-IDF means the term is frequent in this document but rare overall — likely an important distinguishing word.
- N-gram Features: Sequences of N consecutive words or characters. Bigrams ('machine learning', 'neural network') and trigrams capture some local context. Character n-grams are useful for handling misspellings and morphological variations.

Syntactic Features

Syntactic features capture grammatical structure:

- Part-of-Speech (POS) Tags: Each word is tagged with its grammatical role (noun, verb, adjective, etc.). The ratio of adjectives to total words can be a proxy for sentiment expressiveness.
- Parse Tree Features: Features derived from the syntactic parse tree (dependency relations between words).
- Named Entity Recognition (NER): Identifying and categorizing named entities — people (PERSON), organizations (ORG), locations (GPE), dates, monetary values. The presence and frequency of specific entity types can be highly predictive.

Semantic Features

- Sentiment Lexicon Scores: Using pre-built lexicons (VADER, AFINN, SentiWordNet) to score text as positive, negative, or neutral. Each word has a pre-assigned sentiment score.
- Word Embedding Features: Dense vector representations (Word2Vec, GloVe) that capture semantic similarity. Documents can be represented as weighted averages of their word embeddings.
- Topic Model Features (LDA): Latent Dirichlet Allocation discovers K latent topics in a corpus. Each document is represented as a distribution over topics — a rich, compact representation.
- Semantic Similarity: Using cosine similarity between document embeddings to capture semantic relatedness beyond keyword overlap.

Structural and Statistical Features

- Document Length: Number of words, sentences, or paragraphs. Longer documents may be more informative or belong to different categories.
- Average Word Length: Can indicate text complexity and domain (scientific texts use longer words).
- Punctuation Features: Frequency of exclamation marks, question marks, or ellipses — relevant for sentiment or style analysis.
- Readability Scores: Flesch-Kincaid Grade Level, Gunning Fog Index — measure text complexity.
- Lexical Diversity (Type-Token Ratio): Ratio of unique words to total words — high diversity suggests more sophisticated vocabulary use.

Feature Selection Methods

With potentially hundreds of thousands of features, selecting the most informative ones improves model performance and reduces overfitting:

- Chi-Squared Test: Tests the independence between each feature and the target variable. Features with high chi-squared values are most dependent on the target.

- Mutual Information: Measures the amount of information shared between a feature and the target.
- Document Frequency Thresholding: Removing very rare and very common terms (as discussed in preprocessing).
- Dimensionality Reduction (LSA/SVD): Latent Semantic Analysis uses Singular Value Decomposition to project the high-dimensional term matrix into a lower-dimensional semantic space.

KEY TAKEAWAYS
• TF-IDF is the most important and widely used lexical feature in text mining.
• N-gram features partially capture word order context beyond simple BoW.
• Semantic features (embeddings, LDA topics) capture meaning beyond surface word patterns.
• Feature selection (chi-squared, mutual information) helps manage high-dimensional text feature spaces.

2.7 Modelling with Text Mining Features

Once text has been converted to numerical features, a wide range of machine learning algorithms can be applied. Different algorithms have different strengths and are suited to different types of text mining tasks.

Naive Bayes

Naive Bayes is one of the oldest and most effective classifiers for text classification. It applies Bayes' theorem with the 'naive' assumption that all features (words) are conditionally independent given the class label.

- Multinomial Naive Bayes: Suited for count data (word frequencies). Common for document classification.
- Bernoulli Naive Bayes: Binary features (word present/absent). Suitable for short texts.
- Advantages: Very fast training and prediction, works well with limited data, strong baseline for text classification.
- Disadvantage: The independence assumption is almost always violated in text (words are correlated with each other).

Logistic Regression

Despite its name, logistic regression is a classification algorithm. It models the log-odds of class membership as a linear combination of features. With L1 regularization (Lasso), it performs automatic feature selection, setting many feature weights to zero — useful when vocabulary is large.

- Advantage: Interpretable weights, strong regularization options, fast training with sparse data.
- Limitation: Linear decision boundary — may miss complex nonlinear patterns.

Support Vector Machines (SVM)

SVMs find the maximum-margin hyperplane that separates classes. With the linear kernel, SVMs are highly effective for high-dimensional sparse text data (like TF-IDF matrices). Linear SVMs are often the best non-neural baseline for text classification.

- Kernel Trick: Non-linear kernels (RBF) can capture nonlinear patterns but are much slower for high-dimensional data.
- Advantage: Excellent performance on high-dimensional sparse data, strong theoretical guarantees.
- Disadvantage: Slow training on large datasets, less intuitive than decision trees.

Decision Trees and Ensemble Methods

- Decision Trees: Can model nonlinear decision boundaries but tend to overfit on text data due to high dimensionality.
- Random Forest: Reduces overfitting through bagging and feature randomization. Useful for text when combined with dense features (embeddings).
- Gradient Boosting (XGBoost, LightGBM): Very effective with dense embedding features; less commonly used with raw TF-IDF due to the high dimensionality.

Deep Learning for Text

- Recurrent Neural Networks (RNNs, LSTMs, GRUs): Process text sequentially, maintaining a hidden state that captures context. LSTMs handle long-range dependencies through gating mechanisms.
- Convolutional Neural Networks (CNNs) for Text: Apply 1D convolutions over word sequences to extract local features (n-gram-like patterns). Fast and effective for classification.
- Transformers (BERT, GPT, RoBERTa, T5): Self-attention mechanisms allow every word to attend to every other word in the sequence. Pre-trained on massive corpora, fine-tuned on specific tasks. Currently achieve state-of-the-art on virtually all NLP benchmarks.

Model Evaluation for Text Classification

- Accuracy: Proportion of correctly classified documents. Misleading with imbalanced classes.
- Precision: Of all documents predicted as positive, what fraction are actually positive? Precision = TP/(TP+FP).
- Recall (Sensitivity): Of all actual positive documents, what fraction did the model correctly identify? Recall = TP/(TP+FN).
- F1-Score: Harmonic mean of precision and recall. F1 = 2 × (Precision × Recall)/(Precision + Recall). Balances both metrics.
- AUC-ROC: Area Under the Receiver Operating Characteristic Curve. Measures the model's ability to discriminate between classes across all decision thresholds.

- Confusion Matrix: A table showing true positives, false positives, true negatives, and false negatives — provides a complete picture of classifier performance.

KEY TAKEAWAYS

- Naive Bayes and Logistic Regression are the strongest traditional baselines for text classification.
- Linear SVMs often outperform other algorithms on high-dimensional TF-IDF features.
- Deep learning (BERT, GPT) achieves state-of-the-art results but requires more data and computation.
- F1-Score is the preferred metric for imbalanced text classification problems.

2.8 Regular Expressions

Regular expressions (regex or regexp) are patterns that describe sets of strings. They are an essential tool for text preprocessing, feature extraction, and information extraction. Regular expressions can search, match, and manipulate text with great precision and efficiency.

Basic Regular Expression Syntax

Pattern	Meaning	Example
.	Any character except newline	c.t matches 'cat', 'cut', 'cot'
^	Start of string / line	^Hello matches 'Hello world'
$	End of string / line	world$ matches 'Hello world'
*	0 or more of previous	ab* matches 'a', 'ab', 'abb'
+	1 or more of previous	ab+ matches 'ab', 'abb' not 'a'
?	0 or 1 of previous (optional)	colou?r matches 'color' or 'colour'
{n,m}	Between n and m repetitions	a{2,4} matches 'aa', 'aaa', 'aaaa'
[abc]	Character class (a, b, or c)	[aeiou] matches any vowel
[^abc]	Negated character class	[^0-9] matches any non-digit
\d	Any digit [0-9]	\d+ matches '123' in 'abc123'
\w	Word character [a-zA-Z0-9_]	\w+ matches words
\s	Whitespace (space, tab, newline)	\s+ matches whitespace
|	Alternation (OR)	cat|dog matches 'cat' or 'dog'
()	Capture group	(\d+) captures a number
(?:)	Non-capturing group	(?:abc)+ matches 'abcabc'

Common Text Mining Applications of Regular Expressions

- Email Extraction: [a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,} — extracts email addresses from text.
- URL Extraction: https?://[^\s]+ — extracts URLs beginning with http or https.
- Phone Number Extraction: (\+?\d{1,3}[\s-]?)?(\(?\d{3}\)?[\s.-]?)\d{3}[\s.-]?\d{4} — matches common phone formats.
- Date Extraction: \d{1,2}[/-]\d{1,2}[/-]\d{2,4} — matches dates like 12/31/2023 or 1-1-24.
- Number Extraction: -?\d+(\.\d+)? — matches integers and decimals, including negative.
- HTML Tag Removal: <[^>]+> — matches HTML tags for removal from web-scraped text.
- Whitespace Normalization: \s+ → ' ' — replaces multiple whitespace characters with a single space.

Regular Expressions in Python

Python's re module provides comprehensive regex support. Key functions include:

- re.search(pattern, string): Searches anywhere in the string, returns Match object or None.
- re.match(pattern, string): Matches only at the beginning of the string.
- re.findall(pattern, string): Returns all non-overlapping matches as a list.
- re.sub(pattern, replacement, string): Replaces all matches with replacement string.
- re.compile(pattern): Compiles regex for reuse — more efficient when applying same pattern many times.

Example — extracting all email addresses: import re; emails = re.findall(r'[\w.-]+@[\w.-]+\.\w+', text)

KEY TAKEAWAYS

• Regular expressions provide powerful pattern matching for text preprocessing and information extraction.

• Core metacharacters: . * + ? ^ $ [] {} | () — each with specific matching behaviors.

• \d (digits), \w (word chars), \s (whitespace) are the most commonly used shorthand character classes.

• Python's re module provides search, match, findall, and sub functions for regex operations.

3. MODEL DEPLOYMENT

3.1 General Deployment Considerations

Model deployment is the process of making a trained machine learning or text mining model available for use in a production environment to serve real-world predictions. It is often called MLOps (Machine Learning Operations) when referring to the full lifecycle of managing deployed models.

The Deployment Gap

A widely recognized problem in industry is the 'deployment gap' — a large fraction (estimated 85–90%) of machine learning models built in organizations never reach production. The reasons include technical complexity, organizational barriers, and failure to account for production requirements during development.

Infrastructure Considerations

- On-Premises vs. Cloud Deployment: On-premises deployment gives maximum control and security but requires significant infrastructure investment. Cloud platforms (AWS SageMaker, Google AI Platform, Azure ML) provide scalable, managed infrastructure with pay-as-you-go pricing.
- Containerization (Docker, Kubernetes): Containers package the model, its dependencies, and runtime environment together, ensuring consistent behavior across development, testing, and production environments. Docker creates containers; Kubernetes orchestrates them at scale.
- Batch vs. Real-Time Serving: Batch inference processes large volumes of data at scheduled intervals (e.g., nightly scoring of all customers). Real-time serving responds to individual requests with low latency (e.g., live spam detection). The choice depends on latency requirements and throughput needs.
- Model Serving Frameworks: MLflow, BentoML, TensorFlow Serving, Seldon Core, ONNX Runtime — tools that simplify model serving, version management, and A/B testing.

Scalability and Performance

- Horizontal Scaling: Adding more server instances to handle increased request volume. Stateless models (those that do not maintain session state) are easy to scale horizontally.
- Vertical Scaling: Upgrading server hardware (more CPU, RAM, GPU) to handle larger individual requests.
- Caching: Storing prediction results for common inputs to avoid recomputation. Especially effective for text queries where the same document may be processed repeatedly.
- Model Compression: Techniques like quantization (reducing numerical precision), pruning (removing small weights), and knowledge distillation (training a smaller model to mimic a larger one) reduce model size and inference time.

- GPU Acceleration: For deep learning models, GPU inference is significantly faster than CPU. Cloud providers offer GPU instances for high-throughput inference.

Data Pipeline Integration

- Feature Consistency: The features computed at training time must be computed identically at inference time. Any discrepancy (different stop word list, different tokenization) will degrade performance. This is called 'training-serving skew'.
- Data Ingestion Pipeline: Mechanisms to ingest new data, preprocess it (using the same preprocessing steps as training), and feed it to the model. Apache Kafka for streaming, Apache Spark for batch processing.
- Feature Store: A centralized repository for storing and serving features, ensuring consistency between training and serving. Tools: Feast, Tecton, Hopsworks.

Model Versioning and Management

- Version Control for Models: Tracking which model version is deployed, what training data was used, what hyperparameters were set. Tools: MLflow, DVC, Weights & Biases.
- A/B Testing: Deploying multiple model versions simultaneously and routing a fraction of traffic to each to compare performance in production. Essential for validating improvements before full rollout.
- Canary Deployment: Gradually increasing traffic to a new model version (e.g., 5% → 20% → 50% → 100%), monitoring for issues at each stage.
- Rollback Capability: The ability to quickly revert to a previous model version if the new version underperforms or causes errors.

Model Monitoring

Once deployed, models must be continuously monitored because real-world data distributions change over time (concept drift or data drift), causing model performance to degrade.

- Data Drift Monitoring: Detecting when the statistical properties of input data change significantly from the training data distribution. Tools: Evidently AI, WhyLabs, Fiddler.
- Concept Drift: The relationship between features and the target variable changes over time (e.g., a fraud detection model may become less effective as fraudsters adapt their behavior).
- Performance Monitoring: Tracking prediction quality metrics (accuracy, F1, AUC) over time using ground truth labels when available.
- Operational Metrics: Monitoring latency, throughput, error rates, and resource utilization to ensure the serving infrastructure is healthy.
- Alerting: Automated alerts when performance metrics fall below thresholds, triggering model retraining or investigation.

Ethical and Regulatory Considerations

- Fairness and Bias: Monitoring whether the model treats different demographic groups (gender, race, age) equitably. Disparate impact analysis identifies when a model disproportionately affects protected groups.
- Transparency and Explainability: Many regulations (EU GDPR, US FCRA) require that automated decisions be explainable to affected individuals. Models making decisions about loans, employment, or healthcare must be able to provide reasons.
- Privacy: Ensuring that personally identifiable information (PII) is not exposed through model outputs. Differential privacy techniques can add mathematical guarantees.
- Regulatory Compliance: Financial models must comply with model risk management regulations (SR 11-7 in the US). Medical models must comply with FDA regulations. Compliance requires extensive documentation and validation.

Model Retraining Strategy

- Scheduled Retraining: Retraining the model on a fixed schedule (daily, weekly, monthly) with the latest available data.
- Performance-Triggered Retraining: Retraining when monitored performance metrics drop below a threshold.
- Online/Incremental Learning: Some models can be updated continuously with new data without full retraining. This is more complex but allows the model to adapt in real time.
- Data Versioning: Tracking what data was used for each model version, enabling reproducibility and auditing.

KEY TAKEAWAYS
• Model deployment bridges the gap between trained models and real-world business applications.
• Training-serving skew (inconsistent feature computation) is the most common deployment failure mode.
• Containerization (Docker/Kubernetes) ensures consistent model behavior across environments.
• Continuous monitoring for data drift and concept drift is essential for maintaining model performance over time.
• Regulatory requirements (fairness, explainability, privacy) must be built into deployment from the start.

4. CASE STUDIES

4.1 Survey Analysis Case Study

Survey analysis is one of the most common and powerful applications of text mining in business settings. Organizations regularly collect free-text survey responses from customers, employees, and stakeholders. While structured survey questions (ratings, multiple choice) are easy to analyze, the richest insights often come from open-ended text responses that traditional analytics tools cannot handle efficiently.

Problem Description

Consider a large organization that conducts an annual employee engagement survey with 10,000+ employees. The survey includes standard Likert-scale questions (rated 1–5) and several open-ended questions such as:

- 'What do you enjoy most about working here?'
- 'What is the biggest challenge you face in your role?'
- 'What would you change to make this a better place to work?'

Manually reading 10,000+ free-text responses is impractical. Text mining automates this analysis.

Data Collection and Preprocessing

- Data Format: Responses collected via online survey platform (SurveyMonkey, Qualtrics), exported as CSV/Excel with columns for employee ID, department, role level, and text responses.
- Anonymization: PII (names, locations, specific identifiers) must be removed before analysis to protect respondent privacy.
- Text Cleaning: Remove leading/trailing whitespace, normalize punctuation, handle abbreviations, correct obvious typos.
- Segmentation: Split multi-sentence responses into individual sentences for finer-grained analysis when needed.

Analytical Techniques Applied

- Sentiment Analysis: Classify each response as positive, neutral, or negative. Using a domain-adapted lexicon or a fine-tuned BERT model, sentiment scores are computed for each response. Aggregating by department reveals which teams have the highest satisfaction vs. dissatisfaction.
- Topic Modeling (LDA): Automatically discover the main themes discussed across all responses. Typical topics in employee surveys might include: 'compensation and benefits', 'work-life balance', 'management and leadership', 'growth opportunities', 'team collaboration'. LDA assigns each response a probability distribution over topics.

- Keyword and Phrase Frequency Analysis: Identify the most frequently mentioned terms and phrases. Word clouds provide a visual representation of term frequency. Bigram/trigram analysis reveals compound concepts ('work from home', 'career development', 'team communication').
- Aspect-Based Sentiment Analysis (ABSA): More granular than document-level sentiment — identifies sentiment toward specific aspects. A response may be positive about 'colleagues' but negative about 'management', and ABSA captures both sentiments separately.
- Correlation with Structured Survey Data: Linking text-derived sentiment scores and topics with Likert-scale ratings to validate the text analysis and identify the topics most strongly correlated with overall satisfaction scores.

Results and Insights

A typical survey analysis might reveal:

- Topic Distribution: 35% of responses mention work-life balance, 28% mention management quality, 22% mention compensation, 15% mention career growth.
- Sentiment by Department: Engineering team responses are 72% positive, whereas the Sales team responses are only 45% positive — suggesting targeted interventions for Sales.
- Emerging Concerns: An unusual spike in mentions of 'remote work', 'flexibility', and 'commute' — actionable intelligence for HR policy updates.
- Correlation Insight: Responses with negative sentiment about 'management' correlate strongly (r=0.68) with low overall satisfaction scores, confirming management quality as the primary driver of dissatisfaction.

Deployment and Reporting

- Dashboard: Results are presented through an interactive dashboard (Tableau, Power BI, or custom web application) that allows HR leaders to filter by department, role level, and sentiment.
- Automated Reports: Periodic (quarterly) automated analysis reports compare current survey results to previous periods, highlighting improvements or deteriorations.
- Action Planning Integration: Identified topics are linked to specific HR action items. For example, negative topics around 'career growth' trigger the HR team to review and update career development programs.

KEY TAKEAWAYS
• Survey text mining automates analysis of thousands of free-text responses that would be impractical to read manually.
• LDA topic modeling discovers the main themes discussed without requiring pre-defined categories.
• Aspect-based sentiment analysis reveals sentiment toward specific topics (management, compensation, etc.) separately. • Linking text-derived insights with structured survey scores validates the analysis and identifies key drivers.

4.2 Help Desk Case Study

Help desk and customer support operations generate massive volumes of text data: tickets, emails, chat logs, call transcripts, and resolution notes. Text mining of this data can dramatically improve support efficiency, reduce resolution time, and improve customer satisfaction.

Problem Description

A mid-sized technology company receives approximately 5,000 support tickets per day across multiple channels (email, web portal, phone). Currently, tickets are manually triaged by a team of agents who read each ticket, classify it by category, assign it to the appropriate team, and prioritize it by urgency. This process takes an average of 15 minutes per ticket and introduces significant inconsistency (different agents classify similar tickets differently).

The goal is to build a text mining system that automatically:

1. Classifies tickets by category (Hardware, Software, Network, Account, Billing, etc.)
2. Assigns urgency level (Critical, High, Medium, Low)
3. Routes to the appropriate support team automatically
4. Suggests relevant knowledge base articles to both agents and customers
5. Identifies emerging issues before they become widespread problems

Data Preparation

- Historical Data: 2 years of historical tickets (approximately 3.6 million tickets) with manually assigned categories, priorities, and resolution times serve as the labeled training dataset.
- Text Fields: The primary text fields are the ticket subject line, the full ticket body, and any customer comments. Agent notes and resolution comments may also be included.
- Preprocessing: Standard pipeline — tokenization, lowercasing, stop word removal (with domain-specific stop words added: 'please', 'thank you', 'dear', 'regards', 'sincerely'), stemming or lemmatization.
- Feature Engineering: TF-IDF features from the ticket body, character n-grams for robustness to typos, metadata features (ticket submission time, customer tier, product type, historical ticket count for this customer).
- Class Imbalance: Categories like 'Critical Hardware Failure' are rare compared to 'Password Reset'. SMOTE oversampling and class-weighted loss functions address this imbalance.

Model Building

- Baseline Model: Logistic Regression with TF-IDF features. Achieves 82% accuracy on category classification. Fast, interpretable, easy to deploy.
- Improved Model: Linear SVM with TF-IDF features. Achieves 87% accuracy. Better at handling the high-dimensional sparse feature space.

- Advanced Model: Fine-tuned DistilBERT (a lighter version of BERT). Achieves 93% accuracy. Captures contextual meaning (e.g., the same word 'crash' means different things for 'system crash' vs. 'price crash'). However, 10x slower than SVM for inference.
- Ensemble Approach: Combining SVM predictions with DistilBERT predictions using soft voting. Achieves 94% accuracy with better calibration than either model alone.

Urgency Detection

Urgency classification is a secondary model trained on the same ticket data. Key features that predict urgency:

- Keyword Detection: Words like 'urgent', 'emergency', 'down', 'outage', 'critical', 'cannot work' are strong signals for high urgency.
- Escalation History: Tickets from customers with a history of escalations are initially treated with higher priority.
- Time Sensitivity Phrases: Phrases like 'deadline today', 'meeting in an hour', 'client presentation' signal urgency.
- Repetition: Customer sending multiple tickets about the same issue signals increasing urgency.

Knowledge Base Recommendation System

A semantic search system recommends relevant knowledge base articles to agents viewing a ticket. This is built using:

- Article Embedding: Each knowledge base article is encoded using a sentence transformer model into a dense vector.
- Query Embedding: Incoming tickets are also encoded using the same model.
- Similarity Search: Cosine similarity between ticket embedding and all article embeddings identifies the most relevant articles. Approximate Nearest Neighbor (ANN) algorithms (FAISS, Annoy) enable fast search over millions of articles.

Emerging Issue Detection

To detect emerging problems before they become widespread:

- Sliding Window Analysis: Monitor the frequency of specific topics and keywords over rolling time windows (e.g., 1-hour and 24-hour windows).
- Anomaly Detection: Statistical anomaly detection (z-score, DBSCAN clustering) identifies sudden spikes in specific error messages or product-related terms.
- Alert System: Automated alerts to technical teams when anomaly detection triggers, potentially identifying a widespread outage before hundreds of tickets arrive.

Business Impact and Results

- Automated Routing Accuracy: 91% of tickets automatically routed to the correct team without human intervention, reducing triage time from 15 minutes to under 1 minute.
- First Contact Resolution Rate: Improved by 18% because agents are immediately presented with the most relevant knowledge base articles.
- Average Handling Time: Reduced by 23% due to automated categorization and knowledge base suggestions.
- Customer Satisfaction (CSAT): Improved by 12 percentage points due to faster resolution and more accurate routing.
- Cost Savings: Reduction in triage team size from 15 agents to 5 agents (the remaining 10 were redeployed to handle complex cases).
- Early Warning: Emerging issue detection identified 3 widespread software bugs within 30 minutes of first user reports, preventing escalation.

Deployment Architecture

- API Endpoint: The classification model is deployed as a REST API endpoint that receives ticket text and returns predicted category, priority, and confidence scores.
- Integration: The API is integrated with the ticketing system (Zendesk, ServiceNow) through webhooks — new tickets automatically trigger a classification request.
- Human-in-the-Loop: For tickets where the model confidence is below 80%, the prediction is flagged for human review before routing. This maintains accuracy while automating high-confidence cases.
- Continuous Learning: Agent corrections to automated classifications are logged and used for periodic model retraining (monthly retraining cycle).

KEY TAKEAWAYS
• Automatic ticket classification and routing can reduce triage time from 15 minutes to under 1 minute.
• Combining traditional ML (SVM) with deep learning (BERT) in an ensemble can achieve near-human accuracy.
• Knowledge base recommendation systems improve agent efficiency and first contact resolution rates.
• Emerging issue detection through anomaly detection on ticket streams can identify widespread problems early.
• Human-in-the-loop design (flagging low-confidence predictions) ensures quality while maximizing automation.

5. GLOSSARY OF KEY TERMS

TERM	DEFINITION
AdaBoost	Adaptive Boosting — a boosting algorithm that sequentially trains weak classifiers, giving higher weight to misclassified examples in each iteration.
Ambiguity	The property of text where a word, phrase, or sentence has multiple valid interpretations depending on context.
Aspect-Based Sentiment Analysis (ABSA)	Identifying sentiment toward specific aspects or entities mentioned in text, rather than overall document sentiment.
Bagging (Bootstrap Aggregating)	An ensemble method that trains multiple models on random bootstrap samples of the training data, then aggregates their predictions.
Bag-of-Words (BoW)	A text representation that records word frequencies while ignoring word order and grammar.
BERT	Bidirectional Encoder Representations from Transformers — a pre-trained deep learning model for NLP that represents words based on their full context.
Bootstrap Sample	A random sample drawn with replacement from the original dataset, the same size as the original.
Boosting	An ensemble method that trains models sequentially, with each model focused on correcting the errors of the previous models.
Concept Drift	A change in the statistical relationship between input features and the target variable in production data over time.
Containerization	Packaging software and its dependencies together (e.g., Docker) to ensure consistent behavior across environments.
Data Drift	A change in the statistical distribution of input features in production compared to the training data distribution.
Document-Term Matrix	A matrix where rows represent documents, columns represent terms (words), and cells contain term frequencies.
Ensemble Method	A machine learning approach that combines multiple models to produce better predictions than any individual model.
Extra-Trees	Extremely Randomized Trees — a bagging variant that randomizes both feature selection and split thresholds.
Feature Importance	A measure of how much each feature contributes to a model's predictions.
Gradient Boosting	A boosting algorithm that trains each new model to predict the residual errors (pseudo-residuals) of the current ensemble.

LDA (Latent Dirichlet Allocation)	A probabilistic topic modeling algorithm that discovers latent topics in a document corpus.
Lemmatization	Reducing words to their dictionary base form using morphological analysis (e.g., 'running' → 'run', 'better' → 'good').
LightGBM	Light Gradient Boosting Machine — a gradient boosting framework using leaf-wise tree growth for faster, more accurate models.
LIME	Local Interpretable Model-agnostic Explanations — a technique for explaining individual predictions of complex models using locally fitted simple models.
Model Deployment	The process of making a trained machine learning model available in a production environment to serve predictions.
N-gram	A contiguous sequence of N items (words or characters) from text (e.g., bigram = 2 words, trigram = 3 words).
Named Entity Recognition (NER)	Identifying and categorizing named entities in text (persons, organizations, locations, dates, etc.).
Occam's Razor	The principle that among competing hypotheses, the one with fewer assumptions should be preferred; in ML: prefer simpler models.
Out-of-Bag (OOB)	Training examples not included in a particular bootstrap sample (approximately 37%), used for model validation.
Partial Dependence Plot (PDP)	A visualization showing the marginal effect of one or two features on the predicted outcome, averaging over all other features.
Polysemy	A property of words where a single word has multiple related meanings (e.g., 'bank' as financial institution or river bank).
Random Forest	A bagging ensemble of decision trees with random feature selection at each node split for greater diversity.
Regular Expression	A pattern that describes a set of strings, used for text searching, matching, and manipulation.
SHAP	SHapley Additive exPlanations — a game-theory-based framework for explaining individual predictions by attributing each feature's contribution.
Stemming	Reducing words to their stem by removing suffixes using heuristic rules (e.g., 'running' → 'run', 'studies' → 'studi').
Stop Words	Common words with little semantic meaning (e.g., 'the', 'a', 'is') that are often removed during text preprocessing.
Synonymy	Multiple different words having the same or similar meaning (e.g., 'car', 'automobile', 'vehicle').
Text Mining	The process of extracting high-quality, structured information and patterns from unstructured text data.

TF-IDF	Term Frequency-Inverse Document Frequency — a feature weight that measures how important a term is to a document relative to a corpus.
Tokenization	The process of splitting text into individual tokens (words, sub-words, or characters).
Topic Modeling	Unsupervised discovery of abstract topics that occur across a collection of documents.
Training-Serving Skew	Discrepancy between how features are computed during training vs. production inference, causing performance degradation.
Unstructured Data	Data without a predefined format or organization (e.g., text, images, audio), requiring specialized processing for analysis.
XGBoost	eXtreme Gradient Boosting — a highly optimized gradient boosting implementation with regularization and parallel processing.

References

Source: WordPress.com https://share.google/vF5DLcnTfKw0349RS

********** every end has new beginning **********