DSBDA Endsem – 2024, 23, 22 PYQ and IMP Q&A

UNIT - 3

Q.1 What is the data Preparation phase in Data Analytics Lifecycle. What is the Analytics Sandbox and ETLT process in this phase? [8]

=>

- According to Dietrich (2013), it is a cyclical life cycle that has iterative parts in each of its six steps :
  1) Discovery
  2) Pre-processing data
  3) Model planning
  4) Model building
  5) Communicate results
  6) Operationalize
- Fig. 3.3.1 shows data analytic lifecycle.

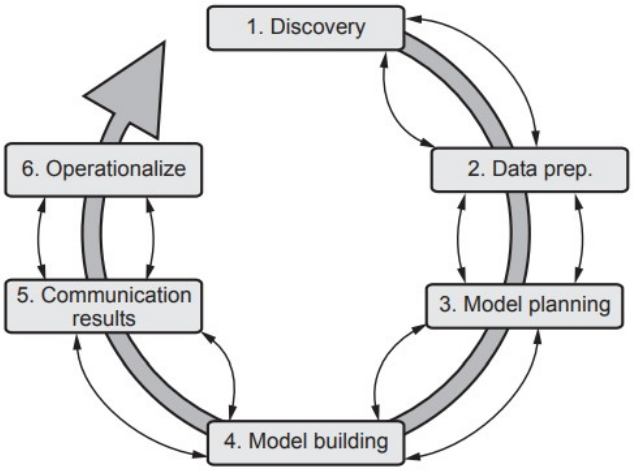


Fig. 3.3.1 Data analytic life cycle

Q.1 What is the Data Preparation phase in Data Analytics Lifecycle? What is the Analytics Sandbox and ETLT process in this phase? [8 Marks]

1. Data Preparation Phase in Data Analytics Lifecycle:

The **Data Preparation** phase is the **third phase** in the **Data Analytics Lifecycle**. It plays a crucial role in shaping raw data into a form suitable for analysis. After data collection, it is often messy, unstructured, incomplete, or inconsistent. This phase involves cleaning, transforming, and organizing the data to ensure its quality and usability.

Key Activities in Data Preparation Phase:

- **Data Integration:** Combine data from multiple sources into a cohesive dataset.
- **Data Cleaning:** Remove or correct inaccurate, duplicate, or missing values.

- **Data Transformation:** Apply operations like normalization, aggregation, formatting, or feature engineering.
- **Variable Selection:** Identify relevant variables/features that contribute to solving the analytical problem.
- **Data Sampling:** Sometimes only a subset of data is selected to reduce processing time.

The output of this phase is a high-quality dataset ready for modeling and analysis in subsequent phases.

---

2. Analytics Sandbox:

The **Analytics Sandbox** is a **dedicated environment** where data scientists and analysts can experiment and perform analytical operations without affecting production systems.

Characteristics of Analytics Sandbox:

- It is **isolated** from live production data systems.
- Provides **controlled access** to data and computing resources.
- Supports **data exploration**, **model testing**, and **hypothesis validation**.
- Allows loading of **curated and cleaned data** that has passed through data preparation.

Purpose: To provide a secure, flexible space where analysts can work with data confidently without the risk of affecting business operations.

---

3. ETLT Process in Data Preparation Phase:

ETLT stands for **Extract, Transform, Load, and Transform** again. It is an extended form of the traditional ETL (Extract, Transform, Load) process tailored for modern analytics.

ETLT Steps:

1. **Extract:** Raw data is pulled from various sources such as databases, logs, sensors, or APIs.
2. **Transform (1st stage):** Initial transformations like cleaning, de-duplication, format conversion.
3. **Load:** The cleaned and partially transformed data is loaded into the analytics sandbox.
4. **Transform (2nd stage):** Advanced transformations, derived variable generation, and feature engineering are performed inside the sandbox.

This two-stage transformation ensures that raw data remains preserved, while analytical teams can manipulate data freely in the sandbox for deeper insights.

3.3.2 Phase 2 : Data Preparation

- This stage involves collecting, processing and cleaning data. Here the focus shifts from business requirements to data requirements. In this early phase, data is collected but not analyzed.
- The data preparation phase is generally the most iterative and the one that teams tend to underestimate most often.

a) Preparing the analytic sandbox :

- Create the analytic sandbox. It also called a workspace. It allows the team to explore data without interfering with live production data.
- Sandbox collects all kinds of data. The sandbox allows organizations to undertake ambitious projects beyond traditional data analysis and BI to perform advanced predictive analytics.

b) Performing ETLT (Extract, Transform, Load, Transform) :

- The team needs to execute Extract, Load and Transform (ELT) to get data into the sandbox.
- Extract, Transform, Load (ETL) : It transforms the data based on a set of business rules before loading it into the sandbox.
- Extract, Load, Transform (ELT) : It loads the data into the sandbox and then transforms it based on a set of business rules.
- Extract, Transform, Load, Transform (ETLT) : It's the combination of ETL and ELT and has two transformation levels.

c) Learning about the data :

- Data is captured through three main ways :
  i. Data acquisition : Obtaining existing data from outside sources.
  ii. Data entry : Creating new data values from data inputted within the organization.

iii. Signal reception : Capturing data created by devices.

d) Data Conditioning :

- Data conditioning includes cleaning data, normalizing datasets and performing transformations. It is often viewed as a preprocessing step prior to data analysis; it might be performed by data owner, IT department, DBA, etc.
- Best to have data scientists involved and data science teams prefer more data than too little.

e) Common tools for data preparation :

- Hadoop can perform parallel ingest and analysis.
- Alpine miner provides a graphical user interface for creating analytic workflows.
- OpenRefine is a free, open source tool for working with messy data.

Q2. List out different stakeholders of an analytics project. What they usually expect at the conclusion (key outputs) of a project? [8]

=>

Q2. List out different stakeholders of an analytics project. What do they usually expect at the conclusion (key outputs) of a project? [8 Marks]

1. Stakeholders in an Analytics Project:

Stakeholders are individuals or groups who have an interest in the success of a data analytics project. Each stakeholder plays a different role and has distinct expectations from the project outcome.

The key stakeholders are:

1. **Business Users / Domain Experts:**
   - These are the individuals who understand the business process and domain knowledge.
   - **Role:** Provide business context, goals, and insights into data behavior.
   - **Example:** Marketing managers, product heads, sales leaders.
2. **Project Sponsors / Executives:**
   - These are high-level decision-makers who fund and authorize the project.
   - **Role:** Ensure alignment of the project with business objectives.
   - **Example:** CIO, CFO, or department VP.
3. **Project Managers:**
   - Responsible for managing the project timeline, deliverables, and coordination.
   - **Role:** Maintain scope, schedule, and communication among all parties.
4. **Data Scientists / Analysts:**
   - Technical professionals who perform modeling, analysis, and interpretation.
   - **Role:** Build models, explore data, and generate actionable insights.
5. **Data Engineers / IT Staff:**
   - Provide data infrastructure, ETLT pipelines, and manage data access.
   - **Role:** Ensure that the data environment is secure, stable, and well-managed.
6. **End Users / Customers:**
   - Individuals who use the final product or solution directly.
   - **Role:** May interact with dashboards, prediction tools, or reports.

---

2. Key Outputs Expected at the Conclusion of a Project:

At the conclusion of an analytics project, stakeholders expect **tangible, decision-supporting outputs** that fulfill the original business objective.

Typical Key Outputs Include:

1. **Actionable Insights:**
   - Clear interpretation of what the data analysis reveals.
   - Business users expect answers to specific questions or trends revealed.

2. **Predictive / Descriptive Models:**
    - Statistical or machine learning models for forecasting, classification, segmentation, etc.
    - Data scientists present model performance metrics (e.g., accuracy, precision).
3. **Data Visualizations and Dashboards:**
    - Interactive tools to explore data, trends, and KPIs.
    - Helpful for business users and executives to make quick decisions.
4. **Recommendations:**
    - Strategic suggestions based on insights (e.g., target customer groups, inventory adjustments).
5. **Reports and Presentations:**
    - Comprehensive summary of the problem, approach, data used, tools applied, and final findings.
    - Project sponsors often expect executive summaries with high-level takeaways.
6. **ROI Metrics or Business Impact Estimates:**
    - Financial or operational gains from implementing the findings.
    - Important for project sponsors to justify investment.
7. **Reusable Pipelines or Tools:**
    - Automated workflows, scripts, or dashboards that can be reused in future analysis.

Q3. List out the activities to be carried out in model planning and model building phase. What are different tools used for these phases? [8]

=>

1. Model Planning Phase [4 Marks]

Objective:
To determine the technique and methodology to be followed for building models, and to identify the tools required.

Activities in Model Planning:

1. **Select appropriate modeling techniques:**
   Based on the type of data and problem (e.g., classification, regression, clustering).
2. **Data partitioning:**
   Divide data into training, testing, and validation datasets.
3. **Explore model assumptions:**
   Understand assumptions like linearity, independence, normality, etc.
4. **Prepare modeling strategy:**
   Define workflow including preprocessing steps, transformation methods, and algorithm selection.
5. **Build a pipeline or plan:**
   Create a modular plan outlining how different stages will be executed and linked.

Tools used in Model Planning:

- **R** – for statistical modeling and planning workflows.
- **Python (Pandas, NumPy, Scikit-learn)** – for data exploration and planning models.
- **SQL** – for initial data exploration and manipulation.
- **SAS Enterprise Miner** – for planning data mining strategies.

---

2. Model Building Phase [4 Marks]

Objective:
To actually construct models based on the techniques finalized in the planning phase.

Activities in Model Building:

1. **Model implementation:**
   Apply machine learning algorithms such as decision trees, SVM, logistic regression, etc.
2. **Model training:**
   Train the model using training data and tune parameters.
3. **Model testing and evaluation:**
   Test on validation/test data using metrics like accuracy, precision, recall, F1-score.
4. **Parameter optimization:**
   Use techniques like grid search, cross-validation to tune hyperparameters.
5. **Ensemble modeling (if needed):**
   Combine models using techniques like bagging, boosting.

Tools used in Model Building:

- **Python (Scikit-learn, TensorFlow, Keras)** – for machine learning and deep learning models.
- **R** – for statistical model building.
- **WEKA** – GUI-based tool for building classification and clustering models.
- **RapidMiner** – Drag-and-drop interface for building and validating models.
- **Apache Spark MLlib** – For scalable model building on big data.

Q4. What is linear regression, and what are its primary objectives? What is the difference between simple linear regression and multiple linear regression? How do you evaluate the performance of linear regression? [8]

=>

1. Definition of Linear Regression [2 Marks]

Linear Regression is a **supervised learning algorithm** used for predicting a continuous dependent variable based on one or more independent variables.
It models the **linear relationship** between the input variable(s) X and the output variable Y.

Mathematical equation:

- **Simple Linear Regression:**

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

- **Multiple Linear Regression:**

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon$$

Where:

- β_0: Intercept
- $\beta_1, \beta_2, \ldots$: Coefficients
- ϵ: Error term

2. Primary Objectives of Linear Regression [2 Marks]

1. **Predictive Analysis:**
   Predict the value of a dependent variable Y for given values of independent variables X.
2. **Modeling Relationships:**
   Understand and quantify the strength of the relationship between variables.
3. **Trend Estimation:**
   Estimate trends and make forecasts using historical data.
4. **Effect Size Interpretation:**
   Analyze how much each independent variable affects the dependent variable.

3. Difference Between Simple and Multiple Linear Regression [2 Marks]

Feature	Simple Linear Regression	Multiple Linear Regression
Number of Independent Variables	One (X)	More than one ($X_1, X_2, \ldots, X_n$)
Equation Form	$Y = \beta_0 + \beta_1 X + \epsilon$	$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots$
Interpretation	Measures the effect of one variable	Measures the effect of multiple variables together
Use Case	Basic forecasting	Complex modeling with multiple factors

4. Performance Evaluation of Linear Regression [2 Marks]

To evaluate how well the model fits the data, the following metrics are used:

a. R-squared (R^2):

- Measures the proportion of variance in the dependent variable explained by the model.
- Value range: 0 to 1. Higher is better.

b. Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- Measures average squared difference between actual and predicted values.
- Lower value indicates better performance.

c. Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{MSE}$$

d. Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

Q5. Explain Data Analytics Cycle with suitable diagram and its phases. [8]

=>

- According to Dietrich (2013), it is a cyclical life cycle that has iterative parts in each of its six steps :
  1) Discovery
  2) Pre-processing data
  3) Model planning
  4) Model building
  5) Communicate results
  6) Operationalize
- Fig. 3.3.1 shows data analytic lifecycle.

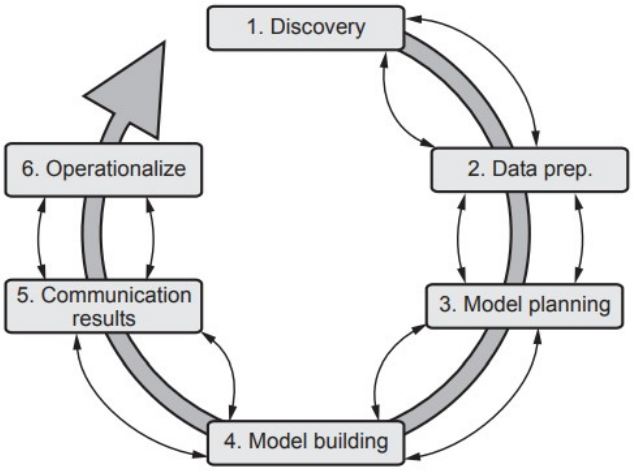


Fig. 3.3.1 Data analytic life cycle

1. Discovery Phase

Objective: Understand the business problem and define project goals.

Key Activities:

- Identify business objectives and success criteria.
- Understand available resources, technology, and stakeholders.
- Frame initial hypotheses and define the scope of the data science problem.

Example:

- A retail company wants to increase sales. Hypothesis: "Discounts increase customer purchases."

Deliverables:

- Business problem definition
- Analytics goals and constraints

---

2. Data Preparation (Data Wrangling)

Objective: Clean and structure data to make it analysis-ready.

Key Activities:

- **Data collection** from multiple sources (databases, APIs, logs, etc.)
- **Data cleaning:** Handle missing values, remove outliers.
- **Data transformation:** Normalization, encoding, and feature extraction.
- **Data integration:** Merging datasets from different sources.

Example:

- Cleaning customer data: replacing null ages with mean age, encoding gender as 0/1.

Tools:

- Python (Pandas, NumPy), R, Excel, SQL

---

3. Model Planning

Objective: Decide suitable analytical techniques and define the modeling approach.

Key Activities:

- Select algorithms (e.g., regression, clustering, classification).
- Visualize relationships among variables.
- Split data into **training**, **validation**, and **testing** datasets.
- Plan parameter tuning strategies (e.g., grid search).

Example:

- Choosing Logistic Regression for predicting whether a user will buy a product.

Tools:

- Python (Seaborn, Scikit-learn), R, SAS, SQL

---

4. Model Building

Objective: Train the selected model(s) on the prepared dataset.

Key Activities:

- Build models using machine learning or statistical algorithms.
- Train the model using training data.
- Evaluate on validation data.

- Perform **hyperparameter tuning** and cross-validation.

Example:

- Training a Random Forest model to classify customer churn.

Tools:

- Python (Scikit-learn, TensorFlow, Keras), R, WEKA, RapidMiner

---

5. Communicate Results

Objective: Present findings in a business-friendly and visual format.

Key Activities:

- Interpret model results.
- Visualize key findings using charts, dashboards, or reports.
- Translate technical insights into actionable business recommendations.

Example:

- Dashboard showing customer segments most likely to churn.

Tools:

- Tableau, Power BI, Matplotlib, Seaborn

---

6. Operationalize

Objective: Deploy the model and ensure it's integrated into business processes.

Key Activities:

- Deploy models into production (via APIs, apps, or dashboards).
- Monitor real-time performance and model drift.
- Automate reporting or predictions using pipelines.

Example:

- Integrate a fraud detection model with the transaction system of a bank.

Tools:

- Docker, Flask API, Airflow, MLflow, Databricks

Q6. List and Explain the various activities involved in identifying potential data resources as a part of discovery phase in Data Analytics Life Cycle? [9]

=>

1. Understand the Business Problem

- Interact with stakeholders to define goals.
- Translate the business problem into a data-driven problem.
- Identify what kind of data would be relevant for analysis.

Example: In fraud detection, the goal is to identify suspicious transactions based on historical patterns.

---

2. Identify Internal Data Sources

- Look for existing organizational data:
  - Relational databases (MySQL, Oracle)
  - Data warehouses (e.g., EDW)
  - CRM, ERP, HRMS systems

Example: For a telecom project, call detail records (CDR) from internal systems are key.

---

3. Identify External Data Sources

- Explore third-party or public sources:
  - Government datasets, Kaggle, UCI Machine Learning Repository
  - APIs (e.g., Google Maps, Weather API)
  - Web scraping from relevant sites

Example: Using weather data to support crop yield prediction.

---

4. Assess Data Accessibility and Legal Constraints

- Check who owns the data and whether it can be accessed.
- Ensure compliance with **data privacy** laws like GDPR or IT policies.
- Understand access methods (e.g., SQL, APIs, flat files).

Example: Personally Identifiable Information (PII) might need encryption or masking.

---

5. Analyze Data Formats and Structures

- Understand the format of available data:
  - Structured (tables)
  - Semi-structured (XML, JSON)
  - Unstructured (logs, images, audio)

Example: Product reviews (text) are unstructured, whereas transaction tables are structured.

---

6. Evaluate Data Quality and Relevance

- Check if data is:
  - Complete (no missing fields)
  - Accurate (free from errors)
  - Updated (recent and relevant)

Example: Old or irrelevant data may give misleading results in real-time applications.

---

7. Identify Data Collection Frequency and Volume

- Determine how frequently data is generated (real-time, daily, monthly).
- Estimate volume to understand storage and processing needs.

Example: IoT sensor data might generate thousands of records per minute.

---

8. Collaborate with Data Owners or IT Teams

- Coordinate with DB admins, data engineers, and IT staff to:
  - Access databases
  - Understand schemas
  - Resolve permission issues

Example: Requesting read-only access to staging databases for analysis.

---

9. Document Metadata and Create a Data Inventory

- Maintain a **data dictionary** with:
  - Attribute names

- Data types
- Descriptions
- Source systems

Benefit: Aids transparency and ensures clear understanding across the team

Q7. List and explain the key roles for successful analytics project. [8]

=>

For a data analytics project to succeed, it requires **collaboration among different roles**, each contributing specific skills—business understanding, technical knowledge, statistical modeling, and communication. These roles work together to ensure the **analytics project lifecycle** is executed effectively.

✅ Key Roles in a Successful Analytics Project:

(*Explain at least 6–7 roles with 1–2 lines each for full marks.*)

---

1. Business Analyst / Domain Expert

- Understands the business context, goals, and problems.
- Translates business requirements into data science objectives.

Example: Defines KPIs, project scope, and success metrics.

---

2. Data Engineer

- Responsible for data acquisition, integration, and pipeline development.
- Builds and maintains scalable systems to extract, transform, and load (ETL) data.

Tools: SQL, Hadoop, Apache Spark, Kafka

---

3. Data Scientist

- Core role that involves data cleaning, analysis, feature engineering, and model building.
- Applies machine learning/statistical techniques to extract insights.

Tools: Python, R, Scikit-learn, TensorFlow

---

4. Data Analyst

- Focuses on interpreting data and generating descriptive insights.
- Performs querying, reporting, and visualization for stakeholders.

Tools: Excel, SQL, Tableau, Power BI

---

5. Statistician / Quantitative Analyst

- Ensures the correct statistical models and assumptions are applied.
- Interprets complex models, ensures accuracy and reliability of results.

Key in hypothesis testing, sampling, regression analysis.

---

6. IT/Data Architect

- Designs the technical infrastructure (databases, servers, data warehouses).
- Ensures secure and efficient data storage and access.

Tools: Oracle, AWS, GCP, Azure, HDFS

---

7. Project Manager

- Oversees timelines, resource allocation, and team coordination.
- Acts as the communication bridge between technical and business teams.

Ensures deliverables meet business requirements within time and budget.

---

8. Visualization Expert / UI Designer (Optional)

- Develops dashboards and intuitive reports.
- Ensures data insights are **presented effectively** for decision-makers.

Tools: D3.js, Tableau, Looker

Q8. Write short note on : i) Common Tools for the Model Building ii) Model selection for Data Analytics. [9]

=>

✓ i) Common Tools for Model Building *(4.5 Marks)*

Model building involves implementing machine learning algorithms to train predictive models using available data. A variety of tools are used for this phase depending on the scale, language, and application area.

◆ 1. Python (Scikit-learn, TensorFlow, Keras):

- Widely used for both traditional ML and deep learning.
- Libraries like **Scikit-learn** (SVM, Decision Trees, kNN), **Keras**/**TensorFlow** (neural networks).

◆ 2. R:

- Good for statistical modeling, linear models, and data visualization.
- Packages like `caret`, `glm`, `nnet`, and `randomForest`.

◆ 3. WEKA:

- GUI-based open-source tool for model training and testing.
- Includes classification, clustering, regression, association rules.

◆ 4. RapidMiner:

- Drag-and-drop interface for building models without coding.
- Useful for students and business analysts.

◆ 5. Apache Spark MLlib:

- Scalable machine learning library for big data.
- Supports clustering, classification, collaborative filtering, etc.

◆ 6. SAS Enterprise Miner / IBM SPSS Modeler:

- Commercial tools for enterprise-grade predictive modeling.
- GUI-based, used in banking, healthcare.

✓ *Conclusion:* The choice of tool depends on **data size**, **complexity**, and **team expertise**.

✓ ii) Model Selection for Data Analytics *(4.5 Marks)*

Model selection is the process of choosing the most suitable algorithm or model based on the data characteristics and project goals.

◆ 1. Based on Problem Type:

- **Classification:** Logistic Regression, Decision Trees, SVM
- **Regression:** Linear/Multiple Regression, Random Forest Regressor
- **Clustering:** K-Means, DBSCAN, Hierarchical Clustering

◆ 2. Based on Data Size and Type:

- Small datasets: Use simpler models (e.g., Naive Bayes, Linear Regression).
- Large datasets: Use scalable models (e.g., XGBoost, Neural Networks).

◆ 3. Based on Model Complexity:

- Simple models are interpretable (e.g., Decision Trees).
- Complex models may offer better accuracy but are less explainable (e.g., Deep Neural Networks).

◆ 4. Evaluation Metrics:

- Use metrics like:
  - Accuracy, Precision, Recall, F1-score (for classification)
  - MSE, RMSE, R^2 (for regression)

◆ 5. Cross-validation:

- Use techniques like **k-fold cross-validation** to avoid overfitting and ensure model generalization.

✓ *Conclusion:* Model selection must **balance accuracy, interpretability, and computational cost** based on the analytics objective.

Q9. What is Model Building elaborate this phase of data analytics with the help of suitable example. [9]

=>

✓ Definition of Model Building (2 Marks)

Model Building is a critical phase in the Data Analytics Life Cycle where machine learning or statistical models are developed using the **prepared and cleaned data**. The goal is to

create a **predictive or descriptive model** that captures patterns and relationships in the data to **make future predictions or derive insights**.

It involves training the model on historical data using selected algorithms and evaluating its performance for accuracy and generalization.

✅ Key Activities in the Model Building Phase (4 Marks)

◆ 1. Algorithm Selection:

- Choose the appropriate machine learning algorithm based on:
  - Problem type (classification, regression, clustering)
  - Data size and type
  - Interpretability and performance

Example: Use Linear Regression for predicting sales, or Decision Tree for customer churn classification.

◆ 2. Training the Model:

- Feed the training data into the algorithm to allow it to learn patterns.
- The algorithm generates a mathematical function (model) to predict the target variable.

◆ 3. Parameter Tuning (Hyperparameter Optimization):

- Adjust hyperparameters such as:
  - `max_depth` in Decision Trees
  - `learning_rate` in Gradient Boosting
  - `k` in k-NN
- Techniques used: Grid Search, Random Search, Cross-validation.

◆ 4. Model Validation:

- Test the model using unseen data (test/validation set).
- Evaluate with performance metrics such as:
  - **Accuracy, Precision, Recall, F1-score** (for classification)
  - **MSE, RMSE, R^2** (for regression)

◆ 5. Model Comparison and Selection:

- Build multiple models with different algorithms and compare their performance.
- Choose the best-performing model for deployment.

✓ Example: Predicting House Prices (3 Marks)

Problem:

A real estate company wants to predict house prices based on:

- Area (sq.ft)
- Number of bedrooms
- Location
- Age of property

Steps:

1. **Data preparation:** Clean dataset with house features and historical prices.
2. **Model selection:** Choose **Multiple Linear Regression** (continuous output).
3. **Training:** Train the model on 80% of the data.
4. **Prediction:** Predict price for new houses using:

   $$Y=\beta0+\beta1(Area)+\beta2(Bedrooms)+\beta3(Age)+...$$

5. **Validation:** Evaluate using **R^2 and RMSE** on test data.

✓ Conclusion:

Model Building is a **core technical phase** in analytics where patterns are learned from historical data to make predictions or decisions. A well-built model enhances **business decision-making** by providing accurate, reliable, and actionable outputs.

Q10. Explain any three sources of Big Data. Differentiate BI versus Data science. [8]

=>

✓ Part A: Three Sources of Big Data [4 Marks]

Big Data is generated from various sources that produce large, fast, and varied volumes of data. These sources are critical for analytics in domains like healthcare, finance, e-commerce, etc.

◆ 1. Social Media Data

- Generated by platforms like Facebook, Twitter, Instagram, LinkedIn.
- Includes: text posts, likes, comments, shares, images, videos.
- Characteristics: Unstructured or semi-structured, real-time, high volume.

Example: Twitter data is used for sentiment analysis and trend prediction.

◆ 2. Machine and Sensor Data (IoT Devices)

- Generated from sensors, RFID tags, GPS, industrial machines, and smart devices.
- Used in smart homes, manufacturing, healthcare, and agriculture.
- Characteristics: Real-time streaming, high velocity and volume.

Example: Temperature and humidity data from smart farming sensors.

◆ 3. Web and Log Data

- Generated from websites, application logs, and server access records.
- Captures clickstream data, user behavior, system errors.
- Helps in website optimization and user behavior analytics.

Example: E-commerce platforms analyze user clicks to recommend products.

✓ Part B: BI vs Data Science [4 Marks]

Aspect	Business Intelligence (BI)	Data Science
Objective	Analyzing past data for reporting and decision support	Predictive and prescriptive analytics using data modeling
Nature of Analysis	Descriptive and diagnostic	Predictive and prescriptive
Tools	Power BI, Tableau, Excel, SQL	Python, R, Scikit-learn, TensorFlow
Data Type	Mostly structured data	Structured, semi-structured, and unstructured
Techniques Used	Querying, dashboards, OLAP	Machine Learning, AI, Statistics

Aspect	Business Intelligence (BI)	Data Science
Skill Requirement	Business knowledge, visualization tools	Programming, statistics, domain + machine learning
Outcome	Insight into what happened and why	Insight into what will happen and how to improve

---

✓ Conclusion:

Big Data comes from a variety of **high-volume and high-velocity sources**, such as social media, sensors, and logs. While **BI focuses on historical data and business reporting**, **Data Science adds advanced modeling and prediction**, making it a more powerful tool for modern analytics needs.

Q11. What are the three characteristic of Big Data and what are the main consideration in processing Big Data. [8]

=>

✓ Part A: Three Characteristics of Big Data – The 3Vs [4 Marks]

Big Data is defined by **three fundamental characteristics**, commonly known as the **3Vs**:

◆ 1. Volume

- Refers to the **massive amount of data** generated every second.
- Data may range from terabytes to petabytes and beyond.
- Comes from sources like social media, sensors, mobile apps, transactions.

Example: Facebook generates over 4 petabytes of data per day.

◆ 2. Velocity

- Refers to the **speed at which data is generated, collected, and processed**.
- Includes **real-time or near real-time data streams**.
- High-velocity data demands fast processing for immediate insights.

Example: Stock market data or real-time fraud detection systems.

◆ 3. Variety

- Refers to the **different formats** and types of data.
  - Structured (tables)
  - Semi-structured (XML, JSON)
  - Unstructured (text, images, videos)
- Handling variety requires flexible storage and processing systems.

Example: Tweets (text), videos (media), and logs (semi-structured) in one dataset.

✓ Part B: Main Considerations in Processing Big Data [4 Marks]

Processing Big Data presents several technical and strategic challenges. Key considerations include:

◆ 1. Scalability

- Systems must handle growing data size efficiently.
- Use distributed processing (e.g., Hadoop, Spark) for scalability.

Example: Adding nodes in a Hadoop cluster for increasing load.

◆ 2. Data Storage and Management

- Use **distributed file systems** (e.g., HDFS) to store large datasets.
- Ensure data is **redundant and fault-tolerant**.

Example: HDFS stores blocks across multiple datanodes with replication.

◆ 3. Data Quality and Preprocessing

- Big Data may contain **noise, missing values, and inconsistencies**.
- Requires proper **data cleaning, transformation, and normalization**.

Example: Handling incomplete customer data before analysis.

◆ 4. Real-time Processing Capabilities

- Use stream processing frameworks (e.g., Apache Kafka, Apache Storm) when real-time insights are required.

Example: Monitoring social media trends in real-time.

◆ 5. Security and Privacy

- Ensure protection of sensitive data (e.g., healthcare, finance).
- Implement encryption, masking, and role-based access control.

Example: Encrypting user transaction data in banking systems.

✓ Conclusion:

Big Data is defined by **Volume, Velocity, and Variety**, and its processing requires careful consideration of **scalability, storage, quality, speed, and security**. Efficient tools and frameworks must be chosen to handle the complexity and size of data.

Q12. Explain Descriptive, Diagnostic, Predictive analytics. [9]

=>

✓ Introduction to Analytics Types:

Analytics is the process of examining data to generate insights and support decision-making. There are several types of analytics based on **what we want to know** from the data:

✓ 1. Descriptive Analytics (3 Marks)

◆ Definition:

Descriptive analytics answers the question:
"What has happened?"

It summarizes past data and organizes it into **meaningful patterns, trends, and insights** using historical data.

◆ Purpose:

- Understand business performance
- Track Key Performance Indicators (KPIs)

◆ Techniques Used:

- Aggregation
- Data visualization (bar charts, line graphs, pie charts)
- Basic statistical analysis (mean, median, mode)

◆ Examples:

- Monthly sales report
- Website traffic summary
- Total revenue generated in Q1

◆ Tools:

- Excel, Power BI, Tableau, SQL

✓ 2. Diagnostic Analytics (3 Marks)

◆ Definition:

Diagnostic analytics answers the question:
"Why did it happen?"

It dives deeper into data to discover **causes and correlations** behind events observed in descriptive analytics.

◆ Purpose:

- Identify reasons behind trends or anomalies
- Determine relationships between variables

◆ Techniques Used:

- Drill-down analysis
- Data mining
- Correlation and root cause analysis

◆ Examples:

- Why did product sales drop in March?
- Why was customer churn higher in a specific region?
- Root cause analysis of system failure

◆ Tools:

- R, Python (Pandas/Matplotlib), SAS, RapidMiner

✓ 3. Predictive Analytics (3 Marks)

◆ Definition:

Predictive analytics answers the question:
"What is likely to happen in the future?"

It uses **historical data, statistical models, and machine learning algorithms** to forecast future events.

◆ Purpose:

- Forecast trends
- Make data-driven business decisions

◆ Techniques Used:

- Regression analysis
- Time series forecasting
- Classification models
- Decision trees, Random Forest, SVM

◆ Examples:

- Predicting future sales or demand
- Forecasting stock prices
- Customer churn prediction

◆ Tools:

- Python (Scikit-learn, TensorFlow), R, IBM SPSS, Apache Spark MLlib

---

✓ Conclusion:

Type	Question Answered	Focus	Example
Descriptive	What happened?	Historical performance	Total monthly sales report
Diagnostic	Why did it happen?	Root cause & correlation	Analysis of customer churn reason

Type	Question Answered	Focus	Example
Predictive	What will happen next?	Forecasting future	Predicting demand for next quarter

Each type of analytics builds upon the other — from understanding the past, to exploring reasons, and finally forecasting the future — making analytics a powerful decision-support tool in data-driven businesses.

Q14. Explain in detail how the model building phase is built by team in data analytics life cycle? [9]

=>

✓ Introduction

The **Model Building phase** is a crucial stage in the Data Analytics Life Cycle where the analytical models are developed, trained, validated, and finalized. This phase is typically executed by a **cross-functional team** with diverse expertise to ensure accuracy, scalability, and business relevance.

✓ Team Roles and Collaboration in Model Building

1. Data Scientist / Machine Learning Engineer

- Leads the technical development of models.
- Selects appropriate algorithms based on problem type (classification, regression, clustering).
- Performs data preprocessing, feature engineering, and model training.
- Applies techniques like hyperparameter tuning, cross-validation to improve model accuracy.

---

2. Data Engineer

- Provides clean and reliable datasets.
- Develops data pipelines for model input.
- Ensures data availability and handles integration from multiple sources.
- Works closely with data scientists to optimize data flow and storage for model training.

3. Domain Expert / Business Analyst

- Provides domain knowledge to guide feature selection and data interpretation.
- Helps understand business constraints and objectives.
- Validates the model outcomes from a business perspective ensuring practical usability.

---

4. Statistician

- Validates statistical assumptions and ensures the model's mathematical soundness.
- Assists in selecting appropriate statistical tests and interpreting results.
- Ensures model robustness and reliability.

---

5. Project Manager

- Coordinates the efforts of team members.
- Manages timelines, resource allocation, and communication between technical and business stakeholders.
- Tracks progress and ensures alignment with business goals.

✓ Key Activities Performed by the Team

a. Data Preprocessing and Feature Engineering

- Data scientists and engineers clean and transform raw data.
- Extract meaningful features using domain expertise.
- Create training and testing datasets.

b. Model Selection and Training

- Evaluate multiple algorithms.
- Train models using training data and tune hyperparameters.
- Use techniques such as Grid Search, Random Search, or Bayesian Optimization.

c. Model Evaluation and Validation

- Test model on validation datasets.
- Use performance metrics like accuracy, precision, recall, RMSE.
- Conduct cross-validation to avoid overfitting.

d. Iteration and Improvement

- Based on evaluation, refine features or change algorithms.
- Repeat training and validation until performance is satisfactory.

e. Documentation and Knowledge Sharing

- Document methodology, parameters, and results.
- Ensure reproducibility and knowledge transfer within the team.

✓ Example Scenario: Customer Churn Prediction

- **Data Engineer:** Prepares customer transaction and service usage data.
- **Data Scientist:** Builds logistic regression and random forest models.
- **Domain Expert:** Suggests customer demographics to consider.
- **Statistician:** Validates model assumptions.
- **Project Manager:** Ensures deadlines and stakeholder communication.

✓ Conclusion

The model building phase is a **collaborative, iterative process** involving multiple roles to develop, evaluate, and finalize models that are accurate, reliable, and aligned with business needs. Effective teamwork and communication are key to successful model development.

Q15. List and explain the steps in data preparation phase of data analytics life cycle. [8]

=>

✓ Introduction:

The **Data Preparation phase** is a critical step in the Data Analytics Life Cycle where raw data is cleaned, transformed, and organized to make it suitable for analysis and modeling. Proper preparation improves model accuracy and reliability.

✓ Steps in Data Preparation Phase:

---

1. Data Collection

- Gather data from multiple sources such as databases, APIs, files, or web scraping.

- Ensure data relevance to the business problem.

---

2. Data Integration

- Combine data from different sources into a unified dataset.
- Resolve schema differences, data formats, and data types.

---

3. Data Cleaning

- Handle missing values by imputation or removal.
- Remove duplicates and correct inconsistencies.
- Detect and handle outliers.

---

4. Data Transformation

- Normalize or standardize numerical data to bring different scales to a common range.
- Convert categorical variables using encoding techniques (one-hot, label encoding).
- Aggregate or discretize data as needed.

---

5. Data Reduction

- Reduce data dimensionality using feature selection or extraction (PCA, LDA).
- Remove irrelevant or redundant features to improve model efficiency.

---

6. Data Splitting

- Divide the dataset into training, validation, and testing sets.
- Ensure representative sampling to avoid bias.

---

7. Data Formatting

- Convert data into required formats compatible with analysis tools or algorithms.
- Ensure consistent data types, e.g., numeric, categorical.

---

8. Data Validation

- Verify data quality after preparation.
- Check for correctness, completeness, and consistency before modeling.

Q16. Write short note on the following: i) ETL ii) Common tools for the model building. iii) Model selection for data analytics. [9]

=>

i) ETL (Extract, Transform, Load) [3 Marks]

- **Extract:**
  Collect data from multiple sources such as relational databases, APIs, flat files, or cloud storage.
  Example: Extracting sales data from an SQL database and customer data from CRM software.
- **Transform:**
  Cleanse and convert raw data into a consistent format suitable for analysis. This includes:
  - Handling missing or inconsistent data
  - Data type conversion
  - Aggregation or summarization
    Example: Converting date formats, imputing missing values, or normalizing numerical ranges.
- **Load:**
  Store the transformed data into a target repository like a data warehouse or data lake for further analytics.
  Example: Loading cleaned data into Amazon Redshift or Hadoop HDFS.
- **Purpose:** Ensures reliable, accurate, and ready-to-use data for analytics and reporting.

---

ii) Common Tools for Model Building [3 Marks]

- **Python (Scikit-learn, TensorFlow, Keras):**
  Widely used open-source libraries for machine learning and deep learning.
  Example: Using Scikit-learn to build a Random Forest classifier.
- **R:**
  Statistical computing language with packages like `caret` and `randomForest` for model building and evaluation.
  Example: Logistic regression for customer churn prediction.
- **WEKA:**
  GUI-based tool providing easy access to algorithms for classification, regression, and clustering without coding.
  Example: Decision tree model for credit risk assessment.

- **RapidMiner:**
  Visual drag-and-drop platform for end-to-end data prep and model building.
  Example: Workflow for churn prediction using gradient boosting.
- **Apache Spark MLlib:**
  Distributed machine learning library suitable for big data processing.
  Example: Large-scale K-Means clustering on millions of data points.
- **SAS Enterprise Miner:**
  Enterprise-grade tool used in industries for advanced analytics and predictive modeling.

---

iii) Model Selection for Data Analytics [3 Marks]

- **Definition:** Choosing the best predictive or descriptive model based on data characteristics, problem type, and performance metrics.
- **Criteria:**
  - **Problem Type:** Classification (e.g., Logistic Regression), Regression (e.g., Linear Regression), Clustering (e.g., K-Means).
  - **Data Size:** Small datasets favor simpler models; big data may require scalable models like Random Forest or Neural Networks.
  - **Model Interpretability:** Decision trees are interpretable; deep learning models are complex but powerful.
  - **Computational Efficiency:** Resource availability and execution time constraints.
- **Evaluation:**
  Use metrics such as accuracy, precision, recall, F1-score (classification), RMSE, R^2 (regression), and techniques like k-fold cross-validation.
  Example: Selecting between SVM and Random Forest by comparing their accuracy and computational cost on test data.

Summary Table (Optional for Extra Clarity)

Aspect	ETL	Model Building Tools	Model Selection
Purpose	Prepare data for analysis	Implement and train predictive models	Choose best model based on criteria
Key Steps	Extract, Transform, Load	Python, R, WEKA, RapidMiner, Spark	Based on problem, data, metrics
Example	Extract from DB, clean data	Use Scikit-learn to build classifier	Choose between Logistic Regression & SVM

Q17. What is driving data deluge? Explain with one example. [9]

=>

✓ Definition of Data Deluge

Data Deluge refers to the **rapid and massive increase in the volume, velocity, and variety of data generated globally** due to technological advancements and digital transformation. This overwhelming flood of data challenges traditional data processing and storage methods, requiring new big data technologies and analytics approaches.

✓ Factors Driving the Data Deluge

1. **Proliferation of Internet-Connected Devices (IoT):**
   Billions of sensors, smart devices, and machines continuously generate huge volumes of real-time data.
2. **Social Media and User-Generated Content:**
   Platforms like Facebook, Twitter, Instagram generate vast amounts of text, images, video, and interaction data daily.
3. **Mobile Devices and Applications:**
   Smartphones and apps create streams of location data, transactions, and communications.
4. **Digital Transformation of Businesses:**
   Automation, online transactions, and cloud computing create large structured and unstructured datasets.
5. **Advances in Technology:**
   Increased storage capacity, low-cost data capture devices, and faster networks enable data generation at unprecedented scales.
6. **Media and Entertainment:**
   High-definition videos, streaming services, and online gaming generate terabytes of data daily.

✓ Example: Smart Cities and IoT Sensors

- **Scenario:**
  Smart cities deploy millions of sensors for traffic monitoring, environmental data (air quality, temperature), energy consumption, and public safety.
- **Data Generated:**
  Real-time streams of traffic flow, pollution levels, energy usage, and surveillance videos.
- **Impact:**
  The volume and velocity of data require advanced processing platforms like Hadoop and Spark, and real-time analytics to manage city operations effectively.

✓ Conclusion

The **data deluge** is driven by exponential growth in digital devices, social media, mobile usage, and smart technologies. This surge in data demands new big data tools and analytics frameworks to convert raw data into actionable insights efficiently.

Q18. What is data science? Differentiate between Business Intelligence and Data Science. [9]

=>

✓ 1. What is Data Science? [3 Marks]

Data Science is a multidisciplinary field that uses **scientific methods, algorithms, statistics, and machine learning** to extract meaningful insights and patterns from **structured, semi-structured, and unstructured data**.

◆ Key Components of Data Science:

- **Data Collection & Cleaning**
- **Exploratory Data Analysis (EDA)**
- **Statistical Modeling**
- **Machine Learning & AI**
- **Data Visualization**
- **Predictive & Prescriptive Analytics**

◆ Purpose:

To turn raw data into actionable insights that support **decision-making**, **automation**, and **business optimization**.

Example: Predicting customer churn using historical behavior data with logistic regression.

✓ 2. Difference between Business Intelligence (BI) and Data Science [6 Marks]

Aspect	Business Intelligence (BI)	Data Science
Definition	Uses historical data for reporting and dashboards	Uses data + models to predict and prescribe future outcomes
Goal	What happened and why	What will happen and how to optimize it

Aspect	Business Intelligence (BI)	Data Science
Data Type	Mostly **structured** data	**Structured, semi-structured, unstructured**
Tools	Power BI, Tableau, Excel, SQL	Python, R, TensorFlow, Scikit-learn, Hadoop
Techniques Used	OLAP, reporting, dashboards	Machine learning, statistics, AI
User Type	Business analysts, managers	Data scientists, ML engineers
Example	Sales dashboard for last 6 months	Forecasting next month's sales using time series prediction
Nature	Retrospective (past-focused)	Predictive and prescriptive (future-focused)

✓ Conclusion:

While **Business Intelligence** helps in **descriptive and diagnostic analysis** of past data, **Data Science** takes it a step forward with **predictive and prescriptive models** using advanced algorithms. Together, they enable modern organizations to make **data-driven decisions** efficiently.

Q19. What are the sources of Big Data. Explain model building phase with example. [9]

=>

✓ Part A: Sources of Big Data [4 Marks]

Big Data originates from various digital and real-world environments. These sources are characterized by **high volume, velocity, and variety**.

◆ 1. Social Media Platforms

- Platforms like Facebook, Twitter, Instagram generate massive amounts of user-generated content: text, images, videos, likes, comments.
- *Example:* Analyzing tweets during elections for sentiment analysis.

◆ 2. Machine and Sensor Data (IoT Devices)

- Devices like smart meters, industrial sensors, GPS trackers, and smart home devices generate continuous streams of data.
- *Example:* Temperature and motion sensors in smart homes.

◆ 3. Web and Application Logs

- Clickstream data from websites and logs from applications provide detailed insights into user behavior and system performance.
- *Example:* E-commerce sites use click data to improve recommendation systems.

◆ 4. Transactional Data

- Generated from point-of-sale (POS) systems, banking systems, and online purchases.
- *Example:* Retail companies analyze transaction data for inventory planning.

◆ 5. Multimedia Content

- Images, audio, video generated from smartphones, surveillance systems, streaming platforms.
- *Example:* YouTube video metadata used for content personalization.

✓ Part B: Model Building Phase with Example [5 Marks]

The **Model Building Phase** in the Data Analytics Life Cycle involves **creating machine learning models** using the cleaned and prepared data to solve specific business problems.

◆ Key Steps in Model Building:

1. **Select Appropriate Algorithm:**
   - Based on the problem type (e.g., classification, regression).
   - *Example:* Use logistic regression for binary classification.
2. **Train the Model:**
   - Feed training data into the algorithm so it can learn patterns.
   - Perform **feature engineering** to enhance model accuracy.
3. **Tune Model Parameters (Hyperparameters):**
   - Adjust settings like tree depth (for decision trees), number of estimators (for random forest), etc.
   - Techniques: Grid Search, Random Search, Cross-validation.
4. **Validate the Model:**
   - Evaluate model performance on validation/test dataset using appropriate metrics.
   - *Metrics:* Accuracy, Precision, Recall, RMSE, R^2.

◆ Example: Predicting Customer Churn

Scenario: A telecom company wants to predict if a customer will leave the service.

Steps:

- **Data Engineer** gathers call logs, service history, and usage stats.
- **Data Scientist** selects **Random Forest** classifier.
- Data is split into training (80%) and testing (20%).
- Model is trained, tuned, and evaluated.
- Result: 90% accuracy in predicting churn.

Q20. Explain big data analytics architecture with diagram. What is data discovery phase. Explain with example. [9]

=>

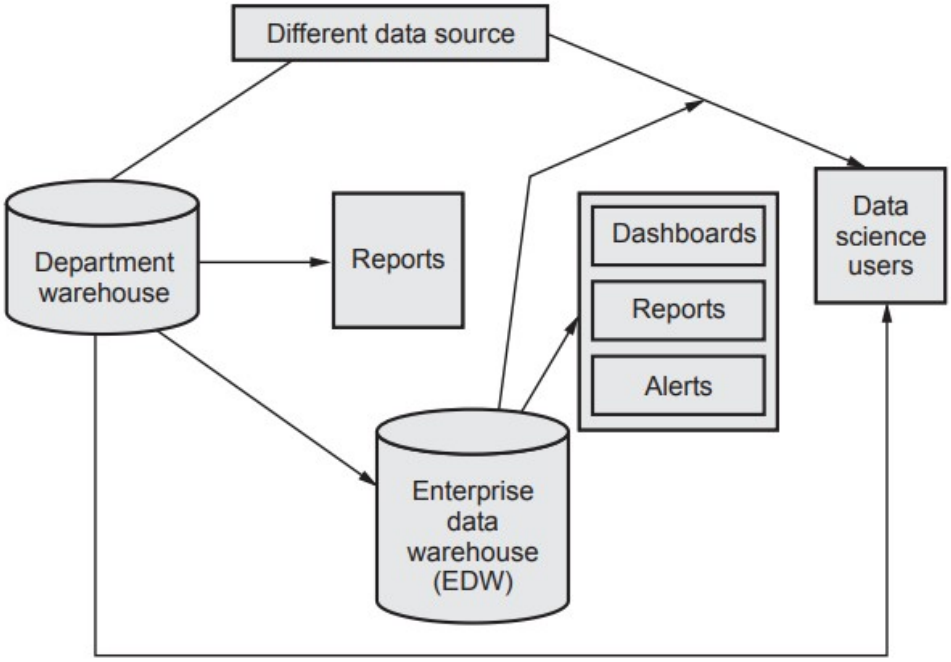
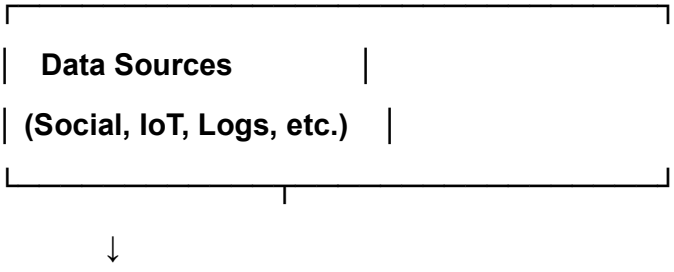
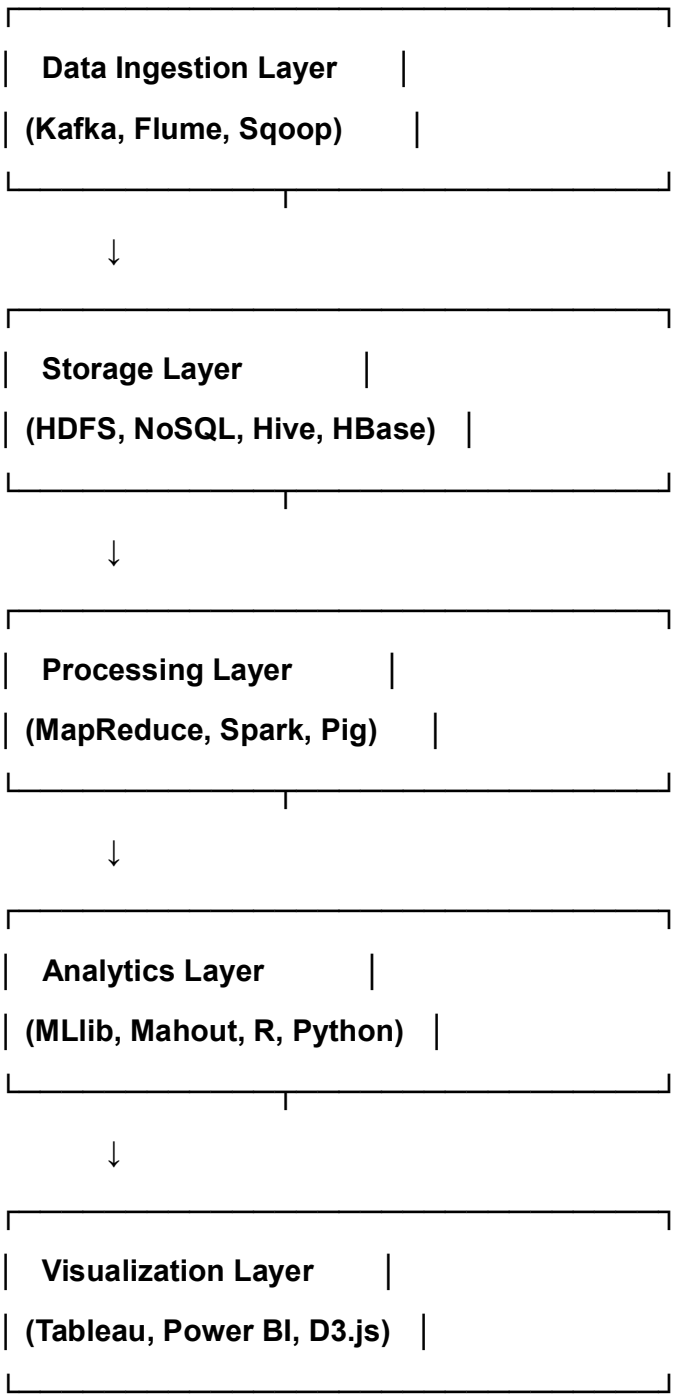


Fig. 3.1.1 Data analytical architecture





✅ Explanation of Layers:

1. **Data Sources:**
   Data is generated from IoT devices, social media, system logs, transactions, etc.
2. **Data Ingestion Layer:**
   Tools like **Apache Kafka**, **Flume**, or **Sqoop** collect and move data into the system.

3. **Storage Layer:**
   Data is stored using distributed file systems such as **HDFS**, **Hive**, **HBase**, or **NoSQL** databases.
4. **Processing Layer:**
   Uses **MapReduce**, **Apache Spark**, or **Pig** for data transformation and processing.
5. **Analytics Layer:**
   Machine learning algorithms and statistical models are applied using tools like **Spark MLlib**, **Mahout**, **R**, or **Python**.
6. **Visualization Layer:**
   Data is visualized using **Power BI**, **Tableau**, or **custom dashboards** to support business decisions.

✅ Part B: Data Discovery Phase [4 Marks]

Definition:

The **Data Discovery Phase** is the **first phase** in the Data Analytics Life Cycle. It involves understanding the **business problem**, identifying **data sources**, and assessing **data availability, quantity, and quality**.

✅ Key Activities:

1. **Define Business Objective:**
   Understand what the business wants to achieve.
   Example: A company wants to reduce customer churn.
2. **Identify Potential Data Sources:**
   Internal (databases, CRM) or external (social media, APIs) data sources are explored.
3. **Assess Data Availability & Quality:**
   Determine if the required data exists, its structure (structured/unstructured), and quality.
4. **Feasibility Analysis:**
   Evaluate if the available data and tools are sufficient for the planned analytics task.
5. **Document Initial Hypotheses:**
   Form basic assumptions or insights based on domain knowledge.
   Example: Customers with fewer support calls are less likely to churn.

✅ Example:

A telecom company wants to predict customer churn.

- **Discovery Phase:**
  - Define goal: Minimize churn.
  - Data sources: CRM database, call logs, service usage history.

- Analyze availability and accessibility of this data.

UNIT- 4

Q1. What is logistic regression, and how does it differ from linear regression? What is the sigmoid function, and what role does it play in logistic regression? [9]

=>

✅ 1. What is Logistic Regression?

(3 Marks)

Logistic Regression is a **supervised learning algorithm** used for **classification problems**, especially **binary classification** where the output variable has only two possible classes (e.g., Yes/No, 0/1, Spam/Not Spam).

Unlike Linear Regression which predicts numeric values, Logistic Regression **predicts the probability** that a given input belongs to a particular class (typically class 1), and then classifies the input based on a threshold (usually 0.5).

✅ Mathematical Expression:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n)}}$$

Where:

- β_0: Intercept
- $\beta_1, \ldots, \beta_n$: Coefficients of features
- $X_1, \ldots, X_n$: Input features

✅ Example:

Predicting whether a student will pass or fail based on study hours and attendance (0 = Fail, 1 = Pass).

✅ 2. Difference Between Logistic and Linear Regression

(3 Marks – Presented as Table)

Aspect	Linear Regression	Logistic Regression
Purpose	Predicts **continuous** values	Predicts **categorical** class labels (usually binary)
Type of Problem	Regression	Classification
Output Range	$-\infty$ to $+\infty$	0 to 1 (interpreted as probability)
Equation	$Y = \beta_0 + \beta_1 X + \cdots + \beta_n X_n$	$\frac{1}{1+e^{-z}}$, where z is the linear sum
Curve	Straight line	S-shaped curve (sigmoid function)
Loss Function	Mean Squared Error (MSE)	Log Loss (Cross Entropy)
Use Case	Predicting prices, sales, temperature	Email spam detection, disease diagnosis, loan approval
Output Interpretation	Direct numeric prediction	Probability of class 1 (e.g., churn probability = 0.86)
Threshold Decision	Not applicable	Class 1 if probability > 0.5, else Class 0

✅ 3. What is the Sigmoid Function and Its Role in Logistic Regression?

(3 Marks)

✅ Definition:

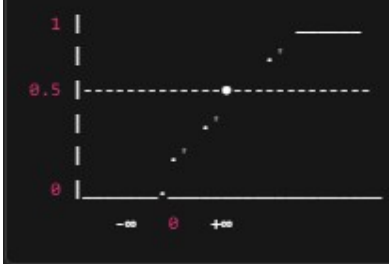
The **sigmoid function** (also called the **logistic function**) is a mathematical function used in Logistic Regression to **map linear values to probabilities** in the range (0, 1).

✅ Formula:

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad \text{where} \quad z = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

✅ Graph of Sigmoid:

vbnet Copy Edit



✅ Role in Logistic Regression:

- Converts the linear output (z) from the regression equation into a **probability** between 0 and 1.
- **Helps make classification decisions:**
  - If $P(Y = 1|X) > 0.5$, classify as **Class 1**
  - Else, classify as **Class 0**
- Enables model to use **log-likelihood** instead of MSE for optimization.

Q2. Suppose you are given a dataset containing information about whether emails are spam or not spam, along with two features: the presence of the word "offer" (1 for present, 0 for absent) and the presence of the word "free" (1 for present, 0 for absent). You are tasked with classifying a new email with the following feature values: "offer"=1 and "free"=1. [9]

Given the training dataset:

Calculate the probability that the new email is spam using Naive Bayes.

=>

To solve **Q2** using **Naive Bayes**, we'll follow a **step-by-step approach** with formulas, suitable for scoring full **9 marks** in the DSBDA exam. This assumes a standard binary classification (Spam or Not Spam) and the application of the **Bernoulli Naive Bayes algorithm**, which is suitable for binary features like word presence.

Q2. Use Naive Bayes to classify a new email with features:

- "offer" = 1
- "free" = 1

Assume you're given a **training dataset** with this structure:

Email ID	Offer (X_1)	Free (X_2)	Class (Spam / Not Spam)
1	1	1	Spam
2	1	0	Spam
3	0	1	Not Spam
4	1	1	Spam
5	0	0	Not Spam

✅ Step 1: Calculate Prior Probabilities

Let's denote:

- **S = Spam, NS = Not Spam**
- Total emails = 5
- Spam emails = 3
- Not Spam emails = 2

$$P(S) = \frac{3}{5} = 0.6$$
$$P(NS) = \frac{2}{5} = 0.4$$

✅ Step 2: Conditional Probabilities (Likelihoods)

We use **Laplace smoothing** to avoid zero probability (i.e., add 1 to numerator, add number of possible outcomes to denominator — in binary case, add 2 to denominator).

For Spam class:

- Count of spam = 3
- Spam emails with "offer"=1 → 3
- Spam emails with "free"=1 → 2

$$P(offer = 1|S) = \frac{3+1}{3+2} = \frac{4}{5} = 0.8$$
$$P(free = 1|S) = \frac{2+1}{3+2} = \frac{3}{5} = 0.6$$

For Not Spam class:

- Count of not spam = 2
- Not spam emails with "offer"=1 → 0
- Not spam emails with "free"=1 → 1

$$P(offer = 1|NS) = \frac{0+1}{2+2} = \frac{1}{4} = 0.25$$
$$P(free = 1|NS) = \frac{1+1}{2+2} = \frac{2}{4} = 0.5$$

↓

✅ Step 3: Apply Naive Bayes Formula

We calculate:

$$P(S|X) \propto P(S) \cdot P(offer = 1|S) \cdot P(free = 1|S)$$
$$P(NS|X) \propto P(NS) \cdot P(offer = 1|NS) \cdot P(free = 1|NS)$$

Substitute values:

$$P(S|X) \propto 0.6 \cdot 0.8 \cdot 0.6 = 0.288$$
$$P(NS|X) \propto 0.4 \cdot 0.25 \cdot 0.5 = 0.05$$

✅ Step 4: Normalize the Probabilities

$$P(S|X) = \frac{0.288}{0.288 + 0.05} \approx \frac{0.288}{0.338} \approx 0.852$$
$$P(NS|X) = \frac{0.05}{0.338} \approx 0.148$$

✅ Step 5: Final Classification

Since:

$$P(Spam|X) \approx 0.852 > P(NotSpam|X) \approx 0.148$$

✅ The email is classified as: SPAM

✅ Conclusion (Exam-Ready Line):

Using the Naive Bayes algorithm, the email with features **offer = 1** and **free = 1** is classified as **SPAM** with an approximate probability of **85.2%**.

Q3. How does the Apriori algorithm discover frequent itemsets in a dataset? What is the role of support and confidence in the context of association rule mining using the Apriori algorithm? [9]

=>

✓ 1. Apriori Algorithm – Overview

(4 Marks)

The **Apriori algorithm** is a classic algorithm in **association rule mining** used to identify **frequent itemsets** in transactional datasets and to generate **strong association rules**.

It uses the **Apriori principle**:

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

This property helps in **pruning the search space** efficiently.

✓ 2. Steps of Apriori Algorithm to Discover Frequent Itemsets:

Let the minimum **support threshold** = 50%

◆ Step 1: Generate 1-itemsets

- Count frequency of individual items.
- Retain items that meet the minimum support.

◆ Step 2: Generate candidate 2-itemsets

- Combine frequent 1-itemsets into pairs.
- Prune itemsets if any subset is infrequent.

◆ Step 3: Count support for candidate 2-itemsets

- Scan dataset to count frequency.
- Retain only those ≥ min support.

◆ Step 4: Repeat for 3-itemsets, 4-itemsets, etc.

- Continue until no more frequent itemsets are found.

✓ Example:

Transactions:

- T1: {Milk, Bread}
- T2: {Milk, Diaper, Beer, Bread}
- T3: {Milk, Diaper, Beer, Coke}
- T4: {Bread, Milk, Diaper, Beer}
- T5: {Bread, Milk, Diaper, Coke}

Result:

- Frequent itemsets:
  - {Milk}, {Bread}, {Diaper}, {Beer}, {Milk, Bread}, {Milk, Diaper}, etc.

✓ 3. Role of Support and Confidence in Apriori Algorithm

(5 Marks)

◆ Support

- Measures **how frequently an itemset appears** in the dataset.
- Used to **filter frequent itemsets** during mining.

Support(A) = No. of transactions containing A / Total transactions

Example:
Support({Milk, Bread}) = 3/5 = 60%

- **Minimum support** threshold is used to remove infrequent patterns.

◆ Confidence

- Measures **the strength of an association rule**.
- Probability that **item B is bought given A is bought**.

$$\text{Confidence}(A \rightarrow B) = \text{Support}(A \cup B) / \text{Support}(A)$$

Example:
If:

- Support({Milk, Bread}) = 3/5
- Support({Milk}) = 4/5
  Then:
- Confidence(Milk → Bread) = (3/5) ÷ (4/5) = 0.75 = 75%
- **High confidence** indicates a strong rule.

Q4. Explain the process of building a decision tree? What are the criteria used for splitting nodes in a decision tree? [9]

=>

✓ 1. Introduction to Decision Tree

A **decision tree** is a popular **supervised machine learning** algorithm used for both **classification** and **regression** tasks. It models decisions and their possible consequences as a tree-like structure of nodes, where:

- **Internal nodes** represent tests on attributes.
- **Branches** represent outcomes of the tests.
- **Leaf nodes** represent class labels (for classification) or continuous values (for regression).

✓ 2. Process of Building a Decision Tree

Decision trees are constructed using a **recursive, top-down, greedy approach** called **recursive partitioning**. The goal is to partition the dataset into subsets containing instances with similar output labels.

Step-by-step process:

◆ Step 1: Start with the full dataset

- The whole training dataset is considered as the root node.

◆ Step 2: Select the best attribute to split the data

- Evaluate all candidate attributes using a **splitting criterion** (e.g., Information Gain, Gini Index).
- Choose the attribute that **best separates** the data into homogeneous subsets (with respect to target class).

◆ Step 3: Partition the dataset

- Split the dataset into subsets, one for each possible value of the selected attribute.
- For continuous attributes, choose a threshold to split the data into two partitions (≤ threshold, > threshold).

◆ Step 4: Recursively apply steps 2 and 3

- For each child node, repeat the attribute selection and partitioning using the subset of data assigned to that node.

◆ Step 5: Stop splitting when one of the stopping conditions is met:

- All instances in the node belong to the same class.
- No more attributes are left to split.
- The number of instances in the node is below a predefined threshold.
- Maximum tree depth is reached.

◆ Step 6: Assign class label to leaf nodes

- Label the leaf node with the most frequent class in the subset of instances.

Example:

For a dataset predicting whether a patient has a disease based on symptoms (e.g., fever, cough):

- Start with the root node having all patients.
- Calculate Information Gain for “Fever” and “Cough”.
- Suppose “Fever” provides highest gain → split on “Fever” (Yes/No).
- Recursively split child nodes further until stopping conditions.

✓ 3. Criteria Used for Splitting Nodes

Choosing the best attribute to split a node is essential. The following are commonly used criteria:

---

i) Information Gain (IG)

- Based on **Entropy**, measures the expected reduction in uncertainty about the target variable after splitting.
- **Entropy:**

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

Where p_i is the proportion of class i in dataset S.

- **Information Gain:**

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

Where S_v is the subset where attribute A has value v.

- Attribute with **highest IG** is chosen.

ii) Gini Index

- Measures **impurity** or probability of misclassification.
- **Gini Impurity:**

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

- Split is chosen that **minimizes weighted Gini impurity** of child nodes.
- Used in **CART (Classification and Regression Tree)** algorithm.

🔹 iii) Gain Ratio

- Addresses the bias of Information Gain towards attributes with many values.
- **Split Information:**

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

- **Gain Ratio:**

$$GainRatio(A) = \frac{IG(S, A)}{SplitInfo(A)}$$

- Choose attribute with **highest Gain Ratio.**

◆ iv) Chi-Square Statistic

- Measures the statistical significance of the association between attribute and target.
- Attributes with **high Chi-square value** indicate better splits.

✓ 4. Summary

Criteria	Formula/Description	Purpose
Information Gain	Reduction in entropy after split	Maximize purity, used in ID3
Gini Index	Measure of impurity; lower Gini means purer split	Used in CART
Gain Ratio	Normalizes Information Gain by split entropy	Reduces bias to multi-valued attributes
Chi-Square	Statistical test for independence	Select splits with significant associations

---

✓ Conclusion:

The **decision tree building process** is a **top-down recursive partitioning** that selects attributes based on criteria to maximize class homogeneity at each split. The choice of splitting criterion directly affects the tree's accuracy and generalization.

Q5. List and explain the various types of analytics in Big data. [9]

=>

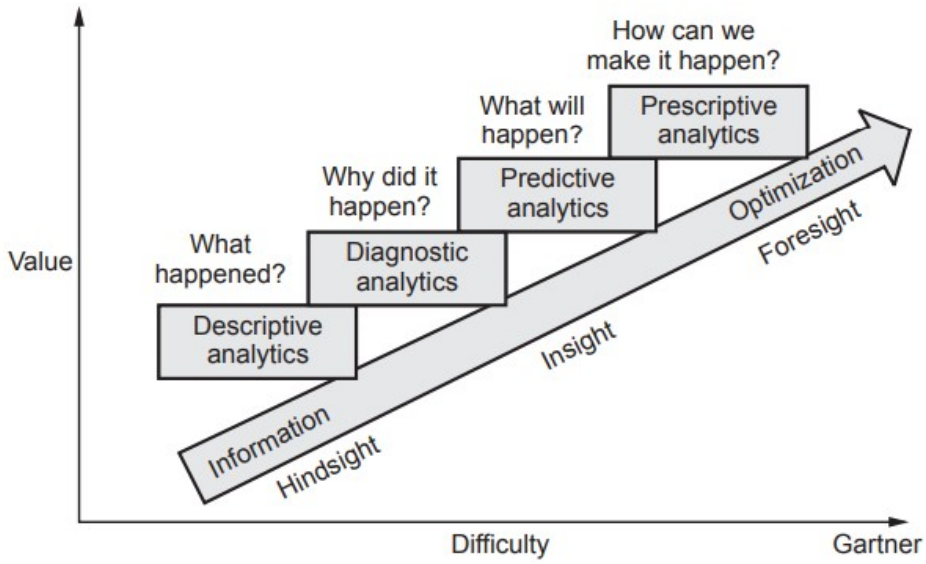


Fig. 4.4.1 Relation between descriptive, predictive and prescriptive analytics

✓ 1. Descriptive Analytics *(3 Marks)*

Definition:

Descriptive Analytics focuses on **summarizing historical data** to understand **what has happened** in the past. It provides a snapshot of past performance and trends.

Purpose:

- To report and visualize data trends.
- To answer questions like “How many?”, “What happened?”, and “Where did it happen?”

Techniques Used:

- Data aggregation (sum, average, count)

- Data visualization (charts, graphs, dashboards)
- Statistical measures (mean, median, mode)
- Reporting and OLAP (Online Analytical Processing)

Example:

- Generating monthly sales reports showing revenue trends over time.
- Dashboard displaying daily website traffic or system usage.

Benefits:

- Provides a clear understanding of historical performance.
- Helps identify patterns or anomalies in data.

✓ 2. Diagnostic Analytics *(3 Marks)*

Definition:

Diagnostic Analytics goes beyond description to understand **why something happened** by exploring relationships and causes in the data.

Purpose:

- To investigate root causes behind trends or events.
- To answer “Why did it happen?”, “What factors contributed?”

Techniques Used:

- Drill-down and drill-through analysis (exploring data in detail)
- Data mining techniques to find associations and correlations.
- Statistical tests and hypothesis testing.
- Root cause analysis.

Example:

- Investigating a sudden decline in product sales by analyzing customer complaints, price changes, or competitor actions.
- Analyzing system logs to determine why a server crashed.

Benefits:

- Enables deeper insight into problems.
- Helps in taking corrective actions.

✓ 3. Predictive Analytics *(3 Marks)*

Definition:

Predictive Analytics uses historical data and statistical or machine learning models to **forecast future outcomes** and trends.

Purpose:

- To anticipate future events and behaviors.
- To answer "What is likely to happen?"

Techniques Used:

- Regression models (linear, logistic)
- Classification algorithms (Decision Trees, Random Forests, SVM)
- Time series analysis and forecasting (ARIMA, exponential smoothing)
- Neural networks and deep learning.

Example:

- Predicting customer churn based on past interactions and purchase behavior.
- Forecasting demand for products to optimize inventory.

Benefits:

- Helps businesses prepare and strategize.
- Enables proactive decision-making.

✓ 4. Prescriptive Analytics *(Detailed 2-3 Marks)*

Definition:

Prescriptive Analytics goes a step further to recommend **actions** based on predictions, optimizing decision-making to achieve specific goals.

Purpose:

- To answer "What should we do?" or "How can we make it happen?"
- To provide decision options and their likely outcomes.

Techniques Used:

- Optimization models (linear programming, integer programming)
- Simulation models (Monte Carlo simulations)
- Heuristic algorithms and rule-based systems

- Reinforcement learning.

Example:

- Suggesting optimal pricing strategies to maximize profit.
- Recommending supply chain adjustments to minimize delivery delays.

Benefits:

- Supports automated decision-making.
- Improves operational efficiency and strategic planning.

Q6. Calculates the support and confidence value for all the possible item sets.[9]

=>

Transaction ID	Items bought
1	Onion, Potato, Cold Drink
2	Onion, Burger, Cold Drink
3	Eggs, Onion, Cold Drink
4	Potato, Milk, Eggs
5	Potato, Burger, Cold Drink, Milk, Eggs

Q7. Explain the need of logistic regression along with its various types. [9]

=>

✓ 1. Need for Logistic Regression *(5 Marks)*

Why Logistic Regression?

- **Handling Classification Problems:**
  Logistic Regression is specifically designed for **classification tasks**, particularly **binary classification** where the outcome variable takes two possible values (e.g., yes/no, spam/not spam).
- **Limitations of Linear Regression for Classification:**
  Linear Regression predicts continuous outcomes and can output values outside the [0,1] probability range, making it unsuitable for classification. Logistic Regression overcomes this by modeling the probability that an instance belongs to a class.

- **Probabilistic Interpretation:**
  Logistic Regression estimates the **probability** of belonging to a particular class, allowing for threshold-based decision-making and better interpretability.
- **Modeling Non-linear Relationships:**
  By applying the **sigmoid function** on a linear combination of features, Logistic Regression can model non-linear probability boundaries effectively.
- **Widely Used & Simple:**
  It is computationally efficient, interpretable, and often serves as a baseline for classification tasks in domains like medicine, finance, and marketing.

✓ 2. Various Types of Logistic Regression *(4 Marks)*

Logistic regression extends beyond binary classification into multiple types, based on the nature of the dependent variable:

◆ i) Binary Logistic Regression

- **Purpose:**
  Models the probability of one of two possible outcomes (0/1, yes/no).
- **Example:**
  Predicting whether a customer will buy a product (1) or not (0).
- **Model Form:**

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n)}}$$

◆ ii) Multinomial Logistic Regression

- **Purpose:**
  Used when the dependent variable has **more than two nominal categories** without any order.
- **Example:**
  Predicting the type of vehicle a customer will buy: Car, Bike, or Truck.
- **Approach:**
  Extends binary logistic regression by modeling probabilities for each class relative to a baseline class.

◆ iii) Ordinal Logistic Regression

- **Purpose:**
  Used when the dependent variable is categorical with **ordered classes** (ordinal data).
- **Example:**
  Predicting customer satisfaction levels: Poor, Average, Good, Excellent.
- **Approach:**
  Models cumulative probabilities respecting the natural order of classes.

◆ iv) Regularized Logistic Regression (Optional Advanced Type)

- **Purpose:**
  Used to prevent overfitting by adding **penalty terms** (L1 or L2 regularization) to the loss function.
- **Example:**
  High-dimensional datasets where feature selection is necessary.

Q8. Explain the following terms with suitable example. i) Removing Duplicates from dataset. ii) Handling Missing Data [9]

=>

i) Removing Duplicates from Dataset *(4.5 Marks)*

Definition:

Removing duplicates means identifying and eliminating **identical or redundant records** in a dataset that appear more than once.

Why it is important:

- Duplicates can **bias the analysis** and distort model accuracy.
- They inflate dataset size unnecessarily, leading to inefficient processing.

How to identify duplicates:

- Check for records with **exact same values** in all or selected key columns.
- Use data processing tools like **Pandas'** `drop_duplicates()`, SQL `DISTINCT` query, or specialized data cleaning tools.

Methods to remove duplicates:

- **Exact match removal:** Delete records identical across all fields.
- **Fuzzy matching:** Identify near-duplicates using similarity metrics for noisy or inconsistent data.

Example:

In a customer database, if two records have the same customer ID, name, and contact details, one record is redundant and should be removed.

---

ii) Handling Missing Data *(4.5 Marks)*

Definition:

Handling missing data involves managing **null, blank, or NA values** in the dataset which occur due to various reasons like entry errors, data corruption, or unavailability.

Why it is important:

- Missing data can lead to **biased analysis**, reduce statistical power, or cause algorithms to fail.

Techniques for Handling Missing Data:

Technique	Description	Example
Deletion	Remove records or variables with missing values	Remove rows where age is missing in a customer dataset
Imputation	Replace missing values with estimated values	Fill missing salary with mean or median salary
Using Algorithms that Handle Missing Data	Some ML algorithms can handle missing values internally	Decision Trees can work with missing values
Prediction of Missing Values	Use regression or k-NN to estimate missing values based on other features	Predict missing temperature based on humidity and time
Flagging Missing Data	Add a binary indicator for missingness	Create a flag column “Is_Age_Missing”

Example:

If the “Income” column has missing values, impute the missing incomes with the median income of the dataset to maintain distribution.

Conclusion

- **Removing duplicates** ensures dataset accuracy by eliminating redundant records.
- **Handling missing data** is critical to maintain the integrity of analysis and model performance, using techniques like deletion, imputation, or prediction.

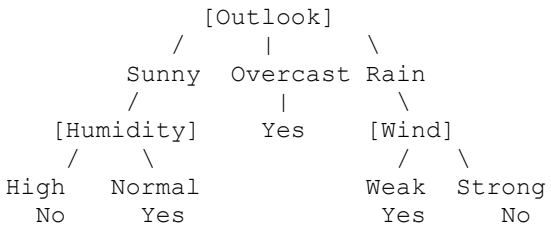
Q9. Explain why decision tree are used. Draw a sample decision tree and explain its parts. [9]

=>

✓ 1. Why Decision Trees Are Used? *(5 Marks)*

- **Easy to Understand and Interpret:**
  Decision trees mimic human decision-making with a simple tree structure, making them easy to visualize and explain to non-technical stakeholders.
- **Handle Both Classification and Regression:**
  They can be used for classification (categorical output) and regression (continuous output) problems.
- **No Need for Data Normalization:**
  Decision trees do not require feature scaling or normalization, simplifying preprocessing.
- **Can Handle Non-linear Relationships:**
  They capture complex decision boundaries by recursive partitioning.
- **Robust to Outliers and Missing Values:**
  Decision trees can handle missing values naturally and are less sensitive to outliers.
- **Feature Selection Embedded:**
  Automatically select the most important variables for splitting at each node.
- **Applicable to Both Small and Large Datasets:**
  Efficient to train and interpret across varying dataset sizes.

✓ 2. Sample Decision Tree Diagram and Explanation *(4 Marks)*



Explanation of Parts:

Part	Description
Root Node	The top-most node representing the attribute used for the first split. In the diagram, it is **Outlook**.
Decision Nodes	Internal nodes where the data is split based on attribute values. Here, nodes for **Humidity** and **Wind** are decision nodes.
Branches/Edges	Lines connecting nodes, representing attribute values or conditions leading to the next node or leaf. For example, branches from Outlook are Sunny, Overcast, and Rain.
Leaf Nodes (Terminal Nodes)	Nodes that represent the final decision/class label. Here, **Yes** or **No** indicate the predicted class (e.g., Play or Not Play tennis).

How to Interpret:

- Start at the root: Check the value of Outlook.
- If Outlook is Sunny, move to Humidity node.
- If Humidity is High, predict No.
- If Outlook is Overcast, predict Yes immediately.
- If Outlook is Rain, check Wind: Weak → Yes; Strong → No.

✓ Conclusion:

Decision trees are widely used for their **simplicity, interpretability, and ability to handle diverse data types**. The tree structure clearly shows how decisions are made step-by-step, helping in transparent and effective predictive modeling.

Q10. How Apriori Algorithm works, explain with suitable example? [9]

=>

✓ 1. Introduction to Apriori Algorithm *(3 Marks)*

The **Apriori algorithm** is a popular method used in **association rule mining** to discover **frequent itemsets** and generate strong rules from transactional databases. It is based on the **Apriori principle**, which states:

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

This property helps reduce the search space by pruning infrequent itemsets early.

✓ 2. How Apriori Algorithm Works (Stepwise Process) *(4 Marks)*

Step 1: Generate Frequent 1-itemsets (L1)

- Scan the dataset to count the support of individual items.
- Retain items with support ≥ minimum support threshold.

Step 2: Generate Candidate 2-itemsets (C2)

- Use frequent 1-itemsets to form candidate 2-itemsets by joining them.
- Prune candidate itemsets if any subset is infrequent.

Step 3: Count Support for Candidate 2-itemsets

- Scan dataset to find support counts.
- Retain those with support ≥ minimum support to form L2.

Step 4: Repeat for k-itemsets (Ck)

- Generate candidate itemsets of size k by joining frequent (k-1)-itemsets.
- Prune candidates if any (k-1)-subset is not frequent.
- Count supports and retain frequent itemsets Lk.

Step 5: Terminate

- When no new frequent itemsets are found.

Step 6: Generate Association Rules

- From frequent itemsets, generate rules A→B with confidence ≥ minimum confidence threshold.

✓ 3. Example

Dataset (5 transactions):

TID	Items
1	{Milk, Bread}
2	{Milk, Diaper, Beer}

TID	Items
3	{Bread, Diaper, Beer}
4	{Milk, Bread, Diaper}
5	{Beer, Bread}

Minimum Support = 60% (3 transactions)

Step 1: Frequent 1-itemsets L1

Item	Support Count	Support %	Frequent?
Milk	3	60%	Yes
Bread	4	80%	Yes
Diaper	3	60%	Yes
Beer	3	60%	Yes

Step 2: Candidate 2-itemsets C2 and their supports

Itemset	Support Count	Support %	Frequent?
{Milk, Bread}	2	40%	No
{Milk, Diaper}	2	40%	No
{Milk, Beer}	1	20%	No
{Bread, Diaper}	2	40%	No
{Bread, Beer}	2	40%	No
{Diaper, Beer}	2	40%	No

No 2-itemsets meet minimum support → **Apriori terminates here**.

Step 3: Association Rules

Only 1-itemsets are frequent, no higher-order frequent itemsets found.

Q11. What is data preprocessing? Explain in details about handling missing data and transformation of data. [9]

=>

✓ 1. What is Data Preprocessing? *(3 Marks)*

Data Preprocessing is a crucial initial step in the data analytics pipeline that involves **cleaning, transforming, and organizing raw data** into a suitable format for analysis and modeling.

Purpose:

- Improve data quality by removing noise and inconsistencies.
- Handle missing or incorrect values.
- Convert data into forms compatible with algorithms.
- Enhance model accuracy and efficiency.

Common preprocessing steps:

- Data cleaning (handling missing, noisy, inconsistent data)
- Data transformation (normalization, encoding)
- Data reduction (feature selection, dimensionality reduction)
- Data integration and formatting

✓ 2. Handling Missing Data *(3 Marks)*

Missing Data:

Occurs when some values in the dataset are absent due to errors, non-response, or data corruption.

Why handle missing data?

- Missing data can bias the analysis.
- Many algorithms cannot handle nulls directly.
- Ensures robust, accurate model training.

Methods to Handle Missing Data:

Method	Description	Example
Deletion	Remove records or variables with missing values	Remove rows where 'Age' is missing
Mean/Median/Mode Imputation	Replace missing numerical values with mean/median, categorical with mode	Fill missing salary with average salary
Prediction Models	Use regression, k-NN, or ML algorithms to predict missing values	Predict missing temperature from humidity and date
Flagging Missingness	Add indicator variables showing missing status	Create binary variable 'Is_Age_Missing'
Using Algorithms that Handle Missing Data	Use models that natively handle missing values	Decision Trees can handle missing attributes

---

✅ 3. Data Transformation *(3 Marks)*

Data transformation converts data into formats or scales suitable for analysis or model requirements.

Types of Data Transformation:

- 🔹 **Normalization / Scaling:**
- Rescales numerical data to a standard range, usually [0,1] or [-1,1].
- Methods: Min-Max Scaling, Z-score Standardization.

Example:

Min-Max scaling of age:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

◆ Encoding Categorical Variables:

- Convert categorical data into numerical formats.
- Techniques:
  - Label Encoding (assign integer codes)
  - One-Hot Encoding (create binary columns)

Example:
Color = {Red, Green, Blue} → One-Hot → Red=[1,0,0], Green=[0,1,0], Blue=[0,0,1]

◆ Aggregation:

- Combine multiple features or records into summarized data.
- Example: Summing daily sales into monthly sales.

◆ Discretization:

- Convert continuous variables into discrete bins or intervals.
- Example: Age groups → Young (18–30), Middle (31–50), Senior (51+)

◆ Feature Extraction / Construction:

- Derive new features from existing ones to enhance model performance.
- Example: Combining 'Height' and 'Weight' into BMI.

Q12. Explain Naïve Bayes' classifier and it applications. [9]

=>

✅ 1. What is Naïve Bayes Classifier? *(5 Marks)*

Naïve Bayes is a **probabilistic classification algorithm** based on **Bayes' Theorem**, assuming **naïve independence** among features. Despite this strong assumption, it performs well in many real-world applications.

Bayes' Theorem:

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

Where:

- $P(C|X)$: Posterior probability of class C given feature vector X
- $P(X|C)$: Likelihood of features given class
- $P(C)$: Prior probability of class C
- $P(X)$: Probability of features (constant for all classes)

Naïve Bayes Assumption:

Assumes all features X_i are **conditionally independent** given the class C:

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

This simplifies computation and makes the algorithm efficient.

Classification Rule:

Choose class C_k which maximizes:

$$P(C_k|X) \propto P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

Types of Naïve Bayes Classifiers:

- **Gaussian Naïve Bayes:** For continuous data assuming normal distribution.
- **Multinomial Naïve Bayes:** For discrete count data (e.g., word counts).
- **Bernoulli Naïve Bayes:** For binary features (presence/absence).

✅ 2. Applications of Naïve Bayes Classifier *(4 Marks)*

◆ Spam Email Detection:

- Classifies emails as spam or not based on word frequencies.
- Efficiently handles large text datasets.

◆ Document Classification:

- Categorizes documents (news, legal, social media posts) into topics or sentiments.

◆ Sentiment Analysis:

- Determines positive, negative, or neutral sentiment from reviews or tweets.

◆ Medical Diagnosis:

- Predicts disease presence by analyzing patient symptoms and test results.

◆ Recommendation Systems:

- Suggests products based on user preferences and behavior patterns.

✓ Advantages of Naïve Bayes:

- Simple, fast, and requires less training data.
- Performs well with high-dimensional data.
- Robust to irrelevant features.

✓ Limitations:

- Assumes independence among features which may not hold in real data.
- Cannot learn interactions between features.

✓ Conclusion:

Naïve Bayes is a powerful yet simple classifier leveraging Bayes' Theorem with the naïve independence assumption. It is widely used in text classification, spam detection, and medical diagnosis due to its efficiency and reasonable accuracy.

Q14. Write short note on the following: i) Removing duplicates from data set. ii) Handling missing data iii) Data transformation. [9]

=>

i) Removing duplicates from dataset

ii) Handling missing data

iii) Data transformation

[9 Marks]

---

i) Removing Duplicates from Dataset *(3 Marks)*

- **Definition:** Removing duplicates means identifying and eliminating **records that appear more than once** with identical or very similar values across one or more attributes.
- **Importance:** Duplicate records can **bias analytics results**, distort model training, and inflate dataset size unnecessarily.
- **Methods:**
  - Exact matching on all or key fields.
  - Use of algorithms or tools with fuzzy matching to detect near-duplicates.
- **Example:** In a customer database, multiple records with the same customer ID and details are reduced to a single unique record.
- **Tools:** SQL `DISTINCT`, Pandas `drop_duplicates()`, data cleaning platforms.

---

ii) Handling Missing Data *(3 Marks)*

- **Definition:** Managing null, blank, or absent values in datasets due to various reasons like human error, system glitches, or privacy issues.
- **Importance:** Missing data can lead to biased analysis or model failure if not handled properly.
- **Techniques:**
  - **Deletion:** Remove records or columns with missing values (simple but may reduce data).
  - **Imputation:** Replace missing values with statistical measures (mean, median, mode) or predictive models.
  - **Using Models that Handle Missing Data:** Some algorithms (e.g., Decision Trees) can work with missing data natively.
  - **Flagging:** Create indicator variables to mark missingness.
- **Example:** Filling missing "Age" values with the median age of the dataset.

---

iii) Data Transformation *(3 Marks)*

- **Definition:** Process of converting data into suitable formats or scales to improve analysis or algorithm performance.
- **Common Transformations:**
  - **Normalization/Scaling:** Rescaling numeric data to a standard range (e.g., 0–1) for uniformity.
  - **Encoding Categorical Variables:** Converting categories into numerical form using one-hot encoding or label encoding.
  - **Discretization:** Converting continuous data into bins or intervals.
  - **Aggregation:** Combining data records or features for summarization.
- **Example:** Encoding "Color" categories (Red, Green, Blue) into binary vectors: Red=[1,0,0].
- **Purpose:** Ensures algorithms that require numerical input or normalized data perform effectively.

Conclusion

Effective data cleaning through **removing duplicates**, **handling missing values**, and **transforming data** is essential to prepare quality data for accurate and reliable analytics outcomes.

Q15. Explain various data pre-processing steps. Discuss essential python libraries for preprocessing. [8]

=>

✓ 1. Various Data Preprocessing Steps *(5 Marks)*

Data preprocessing is the fundamental step in preparing raw data for analysis and modeling. It ensures data quality and enhances the effectiveness of machine learning algorithms. The key steps are:

◆ a) Data Collection

- Gathering data from diverse sources such as databases, CSV files, APIs, sensors, or web scraping.
- Ensures relevant and sufficient data for the problem.

◆ b) Data Cleaning

- **Handling Missing Values:**
  - Missing data can occur due to errors or omissions.
  - Techniques include deletion (removing rows/columns), imputation (mean, median, mode, prediction models), or flagging missingness.
- **Removing Duplicates:**
  - Eliminating redundant records that can bias analysis.
- **Correcting Errors and Inconsistencies:**
  - Fixing typos, standardizing formats (e.g., date formats), and resolving conflicting data.

◆ c) Data Integration

- Combining multiple data sources into a consistent and unified dataset.
- Resolves schema conflicts and format inconsistencies.

- Example: Merging sales data from different branches into a single dataset.

◆ d) Data Transformation

- **Scaling / Normalization:**
  - Rescales numerical features to a common range to improve model convergence and performance.
  - Methods: Min-Max scaling, Z-score standardization.
- **Encoding Categorical Variables:**
  - Converts categorical data into numeric formats suitable for machine learning.
  - Techniques: One-Hot Encoding, Label Encoding, Binary Encoding.
- **Aggregation:**
  - Combining data points to form summarized features, e.g., daily sales aggregated to monthly.
- **Discretization:**
  - Converting continuous variables into discrete bins or categories.

◆ e) Data Reduction

- Reducing dataset size while preserving information.
- Methods: Feature selection (removing irrelevant/redundant features), Dimensionality reduction (PCA, t-SNE).

◆ f) Data Splitting

- Dividing the dataset into **training**, **validation**, and **testing** sets.
- Ensures unbiased evaluation of machine learning models.

✓ 2. Essential Python Libraries for Preprocessing *(3 Marks)*

◆ a) Pandas

- Provides powerful DataFrame structures for data manipulation.
- Functions for handling missing data (`fillna()`, `dropna()`), removing duplicates (`drop_duplicates()`), and data cleaning.
- Facilitates data integration and transformation.

◆ b) NumPy

- Offers efficient numerical operations on arrays and matrices.
- Useful for mathematical computations and fast array processing.

◆ c) Scikit-learn (sklearn.preprocessing)

- Provides extensive tools for data transformation and preparation:
  - **Scaling:** `StandardScaler`, `MinMaxScaler`
  - **Encoding:** `OneHotEncoder`, `LabelEncoder`, `OrdinalEncoder`
  - **Imputation:** `SimpleImputer` for filling missing values
  - **Data splitting:** `train_test_split` function to split datasets

◆ d) Missingno

- Visualization library to understand patterns of missing data through heatmaps, bar charts.

◆ e) Feature-engine / Category-Encoders (Optional Advanced)

- Advanced encoding techniques and feature engineering tools.

✓ Summary Table

Preprocessing Step	Python Library	Key Functions/Classes
Data Cleaning	Pandas	`fillna()`, `dropna()`, `drop_duplicates()`
Numerical Operations	NumPy	Array operations, `np.array()`
Scaling / Normalization	Scikit-learn	`StandardScaler`, `MinMaxScaler`
Categorical Encoding	Scikit-learn	`OneHotEncoder`, `LabelEncoder`
Missing Data Handling	Scikit-learn	`SimpleImputer`
Data Splitting	Scikit-learn	`train_test_split`
Missing Data Visualization	Missingno	Heatmaps, bar charts

Q16. What are association rules? Explain Apriori Algorithm in brief. [9]

=>

✅ 1. What are Association Rules? *(4 Marks)*

Association Rules are **if-then rules** used to discover interesting relationships or patterns among variables in large datasets, especially in **market basket analysis**.

Rule Format:

$$\text{If } A \rightarrow B$$

Where:

- **A (antecedent)**: item(s) on the left-hand side
- **B (consequent)**: item(s) on the right-hand side
- Meaning: *"If A occurs, then B is likely to occur"*

Key Metrics:

Metric	Formula	Meaning
Support	$\text{Support}(A \rightarrow B) = \frac{\text{Transactions containing } A \cup B}{\text{Total transactions}}$	Frequency of itemset A ∪ B in the dataset
Confidence	$\text{Confidence}(A \rightarrow B) = \frac{\text{Support}(A \cup B)}{\text{Support}(A)}$	Likelihood that B occurs when A occurs
Lift	$\text{Lift}(A \rightarrow B) = \frac{\text{Confidence}(A \rightarrow B)}{\text{Support}(B)}$	Strength of the rule over random chance (Lift > 1 = strong)

Example:

If in 100 transactions:

- 40 include Bread,
- 30 include Butter,
- 20 include both,

Then:

- Support = 20/100 = 0.20
- Confidence (Bread → Butter) = 20/40 = 0.50
- Lift = 0.50 / 0.30 ≈ 1.67 → **Positive correlation**

✅ 2. Apriori Algorithm (Brief Explanation) *(5 Marks)*

Apriori is a classic algorithm used to **mine frequent itemsets** and **generate association rules** in large datasets.

Working Principle:

Based on the **Apriori Property**:

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

This principle helps **prune the search space** efficiently.

Steps in Apriori Algorithm:

Step 1: Find Frequent 1-itemsets (L1)

- Scan the dataset and count support for each item.
- Retain items with support ≥ minimum support threshold.

Step 2: Generate Candidate 2-itemsets (C2)

- Form pairs from L1 items.
- Count their support and prune infrequent ones.

Step 3: Generate Higher-order Itemsets (Lk)

- Join frequent (k−1)-itemsets to form candidate k-itemsets.
- Prune itemsets whose subsets are not frequent.
- Count support and retain those meeting threshold.

Step 4: Repeat

- Continue until no more frequent itemsets are found.

Step 5: Generate Association Rules

- For each frequent itemset, generate valid rules with confidence ≥ threshold.

Example:

Dataset:
Transactions:

- T1: {Milk, Bread}
- T2: {Milk, Diaper, Beer}
- T3: {Bread, Diaper, Beer}
- T4: {Milk, Bread, Diaper}
- T5: {Bread, Beer}

Minimum support = 60% → 3 out of 5 transactions

- Frequent itemsets: {Milk}, {Bread}, {Diaper}, {Beer}, {Milk, Diaper}, etc.
- Rule: {Milk, Diaper} → {Bread} (if confidence ≥ threshold)

✓ Conclusion:

Association rules help discover useful patterns like co-occurrence and purchasing behavior in Big Data. The **Apriori algorithm** is a foundational technique to find frequent itemsets and generate strong rules using support and confidence thresholds efficiently.

Q17. Explain the following i) Linear Regression ii) Logistic Regression [8]

=>

✓ i) Linear Regression *(4 Marks)*

Definition:

Linear Regression is a **supervised learning algorithm** used for **predicting continuous numerical values** by modeling the **linear relationship** between the dependent variable (target) and one or more independent variables (features).

Types of Linear Regression:

- **Simple Linear Regression:** Uses one independent variable.
- **Multiple Linear Regression:** Uses two or more independent variables.

Mathematical Form:

For multiple features:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon$$

Where:

- Y: Predicted output
- X_i: Input features
- β_i: Coefficients
- ϵ: Error term

Goal:

To find the best-fitting line (regression line) that minimizes the sum of squared differences between actual and predicted values (i.e., minimize **Mean Squared Error**).

Example:

Predicting a student's marks based on hours studied.

Applications:

- Predicting house prices
- Sales forecasting
- Temperature prediction

✓ ii) Logistic Regression *(4 Marks)*

Definition:

Logistic Regression is a **supervised classification algorithm** used to **predict categorical outcomes**, especially **binary classification** (e.g., Yes/No, 0/1, Spam/Not Spam).

It estimates the **probability** that a given input belongs to a particular class.

Why Not Use Linear Regression for Classification?

Linear regression outputs unbounded values, while logistic regression uses the **sigmoid function** to map outputs between 0 and 1 (i.e., valid probabilities).

Sigmoid Function (Logistic Function):

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad \text{where } z = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

Classification Rule:

- If $P(Y = 1|X) > 0.5$, predict **class 1**
- Else, predict **class 0**

Types of Logistic Regression:

- **Binary Logistic Regression** (2 classes)
- **Multinomial Logistic Regression** (3+ classes, unordered)
- **Ordinal Logistic Regression** (ordered classes)

Example:

Predicting whether a loan application will be approved (Yes/No) based on income and credit score.

Applications:

- Spam email detection
- Disease prediction
- Customer churn classification

✅ Conclusion:

- **Linear Regression** is used for **predicting continuous values**, while

- **Logistic Regression** is used for **classifying data into categories**, especially binary outcomes.

Both are fundamental algorithms in machine learning with wide applications in industry and research.

Q18. Explain scikit-learn library for matplotlib with example. [9]

=>

✓ 1. Introduction

Scikit-learn is a powerful **Python library** used for **machine learning**. It provides simple and efficient tools for:

- Classification
- Regression
- Clustering
- Dimensionality reduction
- Model selection
- Preprocessing

Matplotlib, on the other hand, is a widely used Python library for **data visualization**. It helps visualize model outputs, decision boundaries, confusion matrices, and more.

While **Scikit-learn does not depend on Matplotlib**, it is commonly **used with Matplotlib** for **plotting and interpreting results** of machine learning models.

✓ 2. Using Scikit-learn with Matplotlib: Purpose

- To **visualize datasets** and how models fit them
- To **plot decision boundaries** of classifiers
- To **display confusion matrices**
- To **evaluate performance using ROC curves, learning curves, etc.**

✓ 3. Example: Visualizing a Classifier Decision Boundary

🎯 Problem:

Classify points from two classes using Logistic Regression, and **plot decision boundary** using Matplotlib.

✓ Python Code:

```
python
CopyEdit

import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import numpy as np

# Step 1: Generate sample data
X, y = make_classification(n_samples=100, n_features=2,
                           n_informative=2, n_redundant=0,
                           random_state=0)

# Step 2: Train Logistic Regression model
model = LogisticRegression()
model.fit(X, y)

# Step 3: Plot data points
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='bwr', edgecolors='k')

# Step 4: Plot decision boundary
x_vals = np.linspace(X[:, 0].min(), X[:, 0].max(), 100)
y_vals = -(model.coef_[0][0] * x_vals + model.intercept_[0]) /
model.coef_[0][1]
plt.plot(x_vals, y_vals, color='black', linewidth=2)

plt.title("Logistic Regression Decision Boundary")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

✅ 4. Explanation of Code:

- `make_classification()`: creates a synthetic dataset.
- `LogisticRegression()`: fits a classifier from Scikit-learn.
- `plt.scatter()`: plots the classified data points.
- `plt.plot()`: draws the decision boundary using model coefficients.

✅ 5. Other Examples of Scikit-learn with Matplotlib

Visualization	Purpose
Confusion Matrix	To evaluate classification performance
ROC Curve	To visualize true positive vs false positive rates
Learning Curve	To analyze model accuracy vs training size
Feature Importance	To display relative importance of features

Functions like `plot_confusion_matrix`, `plot_roc_curve`, `plot_tree` *in Scikit-learn use Matplotlib internally.*

✓ Conclusion

Though Scikit-learn is not a visualization library, it is **often used with Matplotlib** to visualize **model outputs**, **performance metrics**, and **decision regions**. This combination improves the understanding and interpretation of machine learning models.

UNIT – 5

Q1. Suppose you have the following dataset containing the coordinates of points in a 2-dimensional space: [9]

Point	X Coordinate	Y Coordinate
A	2	3
B	4	7
C	3	5
D	6	9
E	8	6
F	7	8

Perform K-means clustering on this dataset with K = 2. Assume the initial centroids to be (2,3) and (8,6). Compute the new centroids after each iteration until convergence, and assign points to their nearest centroids.

=>

Q1. Perform K-means clustering on the given 2D dataset with K = 2.

- Use initial centroids:
  - **Centroid 1 (C1): (2, 3)**
  - **Centroid 2 (C2): (8, 6)**
- Perform clustering until convergence (no change in cluster assignment).

✅ Step 1: Dataset (Points & Coordinates)

Point	X	Y
A	2	3
B	4	7
C	3	5
D	6	9
E	8	6
F	7	8

✅ Step 2: Initial Centroids

- C1 = (2, 3)
- C2 = (8, 6)

✅ Step 3: Calculate Euclidean Distance of Each Point to Centroids

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

Distance of all points from both centroids:

Point	Distance to C1 (2,3)	Distance to C2 (8,6)	Assigned to
A	√((2−2)²+(3−3)²) = 0	√((8−2)²+(6−3)²) = √45 ≈ 6.71	C1
B	√((4−2)²+(7−3)²) = √20 ≈ 4.47	√((8−4)²+(6−7)²) = √17 ≈ 4.12	C2
C	√((3−2)²+(5−3)²) = √5 ≈ 2.24	√((8−3)²+(6−5)²) = √26 ≈ 5.10	C1
D	√((6−2)²+(9−3)²) = √52 ≈ 7.21	√((8−6)²+(6−9)²) = √13 ≈ 3.61	C2
E	√((8−2)²+(6−3)²) = √45 ≈ 6.71	√((8−8)²+(6−6)²) = 0	C2
F	√((7−2)²+(8−3)²) = √50 ≈ 7.07	√((8−7)²+(6−8)²) = √5 ≈ 2.24	C2

✅ Cluster Assignments After Iteration 1:

- **Cluster 1 (C1):** A, C
- **Cluster 2 (C2):** B, D, E, F

✅ Step 4: Recalculate New Centroids

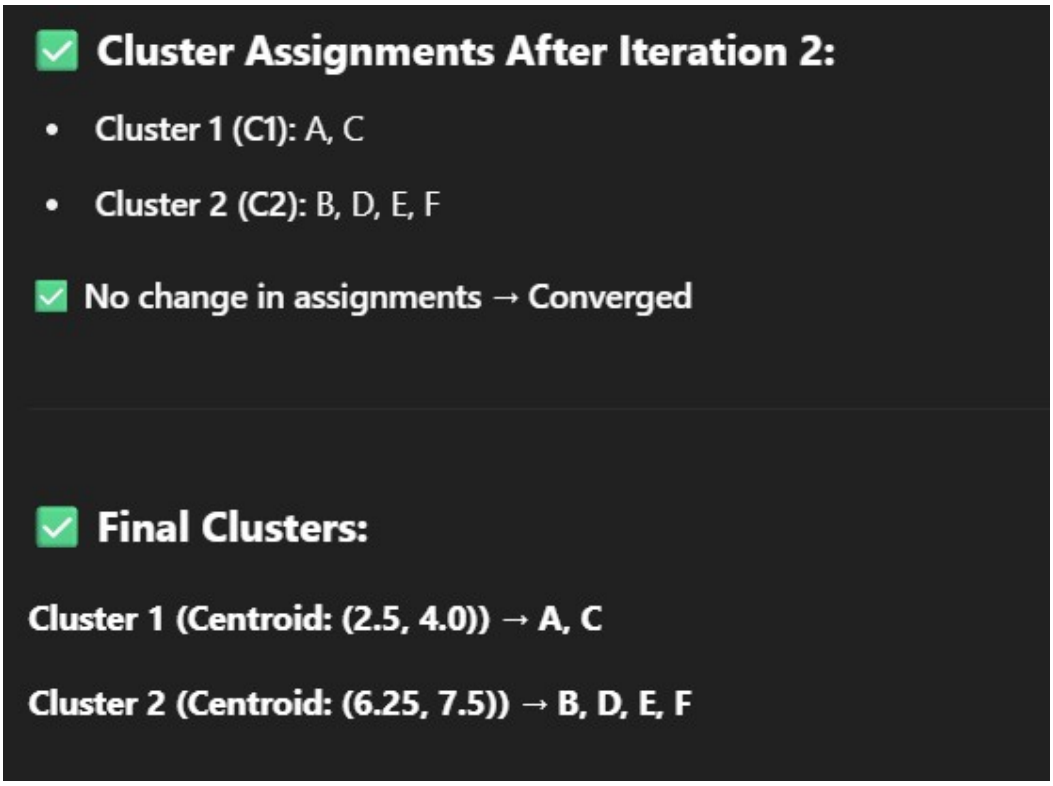
➤ **New C1: Mean of (A, C) = [(2+3)/2, (3+5)/2] = (2.5, 4.0)**

➤ **New C2: Mean of (B, D, E, F)**

- X = (4 + 6 + 8 + 7)/4 = 6.25
- Y = (7 + 9 + 6 + 8)/4 = 7.5
- **New C2 = (6.25, 7.5)**

✅ Step 5: Recalculate Distances to New Centroids

Point	Dist to New C1 (2.5, 4.0)	Dist to New C2 (6.25, 7.5)	Assigned to
A	$\sqrt{((2-2.5)^2+(3-4)^2)} = \sqrt{1.25} \approx 1.12$	$\sqrt{((6.25-2)^2+(7.5-3)^2)} = \sqrt{36.06} \approx 6.00$	C1
B	$\sqrt{((4-2.5)^2+(7-4)^2)} = \sqrt{11.25} \approx 3.35$	$\sqrt{((6.25-4)^2+(7.5-7)^2)} = \sqrt{5.31} \approx 2.30$	C2
C	$\sqrt{((3-2.5)^2+(5-4)^2)} = \sqrt{1.25} \approx 1.12$	$\sqrt{((6.25-3)^2+(7.5-5)^2)} = \sqrt{18.06} \approx 4.25$	C1
D	$\sqrt{((6-2.5)^2+(9-4)^2)} = \sqrt{41.5} \approx 6.44$	$\sqrt{((6.25-6)^2+(7.5-9)^2)} = \sqrt{2.06} \approx 1.43$	C2
E	$\sqrt{((8-2.5)^2+(6-4)^2)} = \sqrt{36.25} \approx 6.02$	$\sqrt{((6.25-8)^2+(7.5-6)^2)} = \sqrt{5.81} \approx 2.41$	C2
F	$\sqrt{((7-2.5)^2+(8-4)^2)} = \sqrt{41.25} \approx 6.42$	$\sqrt{((6.25-7)^2+(7.5-8)^2)} = \sqrt{1.25} \approx 1.12$	C2



Q2. How do you handle noise and irrelevant information in text data during preprocessing? Explain the terms bag of words and TF IDF in text analytics. [9]

=>

✓ 1. Handling Noise and Irrelevant Information in Text Data *(4 Marks)*

Text data is unstructured and often contains **noise**, such as punctuation, symbols, stop words, HTML tags, etc., which do not contribute meaningfully to analysis. The goal of text preprocessing is to **clean, normalize, and structure** this text for analysis and machine learning.

◆ Common Preprocessing Steps:

Step	Description & Purpose	Example
Lowercasing	Converts all text to lowercase to standardize	"India" → "india"
Tokenization	Splits text into individual words or tokens	"data science" → [‘data’, ‘science’]
Stop Word Removal	Removes common words that add little meaning	"is", "the", "a", etc.

Step	Description & Purpose	Example
Stemming/Lemmatization	Reduces words to their root forms	"running" → "run"
Removing Punctuation	Eliminates punctuation, numbers, symbols	"hello!" → "hello"
Removing HTML/Tags	Used in web data scraping to clean text	`<div>Hello</div>` → "Hello"

These steps are essential to improve accuracy and reduce dimensionality in text models.

✅ 2. Bag of Words (BoW) *(2.5 Marks)*

Definition:

Bag of Words is a **feature extraction technique** that represents text data as a **bag (set) of words**, focusing only on **word frequency**, ignoring grammar and word order.

Steps to Create BoW:

1. Build a vocabulary of all unique words from the corpus.
2. Represent each document as a vector based on the count of words from the vocabulary.

Example:

Corpus:

- Doc1: "data science is fun"
- Doc2: "science is powerful"

Vocabulary: [data, science, is, fun, powerful]

Vector Representation:

Document	data	science	is	fun	powerful
Doc1	1	1	1	1	0

	Document	data	science	is	fun	powerful
Doc2		0	1	1	0	1

Pros:

- Simple and fast

Cons:

- Ignores context, word order, and meaning
- High dimensional for large vocabularies

✅ 3. TF-IDF (Term Frequency – Inverse Document Frequency) *(2.5 Marks)*

Definition:

TF-IDF is a more advanced method that measures the **importance of a word** in a document **relative to the entire corpus**.

Key Terms:

- **TF (Term Frequency):**
  Measures how often a term appears in a document.

$$TF(t,d) = \frac{\text{No. of times term } t \text{ appears in } d}{\text{Total terms in } d}$$

- **IDF (Inverse Document Frequency):**
  Measures how unique a term is across documents.

$$IDF(t) = \log\left(\frac{N}{df_t}\right)$$

  Where N = total documents, df_t = documents containing term t

- **TF-IDF Score:**

$$TFIDF(t,d) = TF(t,d) \times IDF(t)$$

Example:

If the term **"data"** appears frequently in one document but rarely across others, its **TF-IDF score is high**, indicating it is **important** and **unique** to that document.

Advantages:

- Weighs down common words
- Highlights unique, meaningful words
- Improves relevance over Bag of Words

✓ Conclusion:

To perform effective text analytics, it is essential to first **clean the text** by removing **noise and irrelevant content**. Then, techniques like **Bag of Words** and **TF-IDF** help convert textual data into meaningful **numerical features** that can be used for further **machine learning or NLP tasks**.

Q3. Explain how hierarchical clustering can be used for visualizing hierarchical relationships in data with suitable example? What are some real-world applications of hierarchical clustering? [9]

=>

✓ 1. What is Hierarchical Clustering? *(1.5 Marks)*

Hierarchical clustering is an **unsupervised learning algorithm** used to build a **hierarchy of clusters**. It organizes data into a **tree-like structure** called a **dendrogram**, where each level of the tree represents a grouping of data points.

There are two main approaches:

- **Agglomerative (Bottom-Up):**
  Start with each data point as a single cluster, and **merge the closest pairs** iteratively.
- **Divisive (Top-Down):**
  Start with all data points in one cluster and **recursively split** into smaller clusters.

✓ 2. How it Visualizes Hierarchical Relationships *(3 Marks)*

◆ Dendrogram:

A **dendrogram** is a tree diagram that illustrates the **merging (or splitting) process** in hierarchical clustering.

- The **height** of the branches represents the **distance** or **dissimilarity** between clusters.
- By **cutting the dendrogram at a certain level**, we can choose the number of clusters.

✅ Example:

Given 5 points: A(1,2), B(2,1), C(5,4), D(6,5), E(7,6)

- Step 1: Treat each point as a cluster
- Step 2: Merge closest points (e.g., A & B, C & D)
- Step 3: Continue merging until all points are in a single cluster

The dendrogram shows:

- A and B merged first
- C and D merged next
- Then E merged with CD
- Finally, AB merged with CDE

☞ A dendrogram visually shows **which clusters formed at which stages**.

✅ 3. Linkage Methods *(Optional, but adds depth – 0.5 Marks)*

Different methods calculate inter-cluster distances:

- **Single linkage:** Minimum distance between points in two clusters
- **Complete linkage:** Maximum distance
- **Average linkage:** Average of all pairwise distances
- **Ward's method:** Minimizes total within-cluster variance

✅ 4. Real-World Applications of Hierarchical Clustering *(4 Marks)*

◆ 1. Gene Expression Analysis (Bioinformatics):

- Cluster genes with similar expression patterns.
- Helps in discovering disease markers.

◆ 2. Market Segmentation:

- Group customers based on purchasing behavior.
- Useful for targeted marketing.

◆ 3. Document or Text Clustering:

- Cluster similar articles, emails, or documents.
- Organize large corpora or detect spam/phishing groups.

◆ 4. Image Segmentation:

- Cluster similar pixel regions in images to detect objects.

◆ 5. Hierarchical Taxonomy (Zoology, Botany):

- Classify species into kingdoms, phyla, classes, etc.

◆ 6. Social Network Analysis:

- Group people based on interaction frequency or community detection.

✓ Conclusion:

Hierarchical clustering provides a **clear visualization of nested groupings** in data using a **dendrogram**. It is especially useful when the number of clusters is **not predefined** and when **relationships at multiple levels** are important. Its **real-world applications span biology, marketing, image analysis, and more**.

Q4. What is the holdout method, and how does it work? Explain the difference between training set, validation set, and test set in the holdout method. [9]

=>

✓ 1. What is the Holdout Method? *(3 Marks)*

The **holdout method** is a simple and widely used **model evaluation technique** in machine learning. It is used to **split the original dataset** into multiple subsets to **train, validate, and test** a model's performance on **unseen data**.

◆ How it works:

- The dataset is randomly divided into:
  - **Training Set**
  - **Validation Set** *(optional but recommended)*
  - **Test Set**
- The model is trained on the training set, tuned on the validation set, and evaluated on the test set.

✓ 2. Dataset Splits Explained *(4 Marks)*

Subset	Purpose	Typical Size	Description
Training Set	Used to **train the model** by fitting parameters (like weights in linear regression).	~60–70% of data	The model "learns" from this data.
Validation Set	Used to **tune model hyperparameters** and prevent overfitting.	~10–20% of data	Helps in **model selection** (e.g., choosing depth of decision tree).
Test Set	Used for **final evaluation** of model performance on unseen data.	~20–30% of data	Measures **generalization accuracy**.

✅ 3. Example:

Suppose we have 1,000 records in a dataset.

- **Training Set:** 700 records
- **Validation Set:** 100 records
- **Test Set:** 200 records

Process:

1. Train the model on 700 records
2. Tune hyperparameters on 100 validation records
3. Evaluate final accuracy, precision, recall, etc., on 200 test records

---

✅ 4. Advantages of Holdout Method *(1 Mark)*

- Simple and fast
- Good for large datasets
- Requires fewer computations than cross-validation

✅ 5. Limitations of Holdout Method *(1 Mark)*

- Performance can vary depending on how the data is split
- Not ideal for small datasets
- Doesn't use the entire dataset for training (unlike k-fold CV)

Q5. Suppose that the given data the task is to cluster points (with (x, y) representing location) into three clusters, where the points are A1 (2, 10), A2(2, 5), A3(8, 4), B1(5, 8), B2(7, 5), B3(6, 4), C1(1, 2), C2(4, 9). The distance function is Euclidean distance. Suppose initially we assign A1, B1 and C1 as the center of each cluster, respectively. [8] Use the k-means algorithm to show only show only the first round of execution with cluster center.

=>

✅ Given:

Points:

Point	Coordinates
A1	(2, 10)
A2	(2, 5)
A3	(8, 4)
B1	(5, 8)
B2	(7, 5)
B3	(6, 4)
C1	(1, 2)
C2	(4, 9)

Initial Cluster Centers (Centroids):

- **Cluster 1 Center:** A1 = (2, 10)
- **Cluster 2 Center:** B1 = (5, 8)
- **Cluster 3 Center:** C1 = (1, 2)

✅ Step 1: Compute Euclidean Distance to Each Initial Centroid

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

Point	To A1 (2,10)	To B1 (5,8)	To C1 (1,2)	Nearest Cluster
A1	0.00	3.61	8.06	Cluster 1
A2	5.00	3.61	3.16	**Cluster 3**
A3	7.21	3.61	7.07	**Cluster 2**
B1	3.61	0.00	6.71	Cluster 2
B2	5.83	3.16	6.08	**Cluster 2**
B3	6.71	4.12	5.00	**Cluster 3**
C1	8.06	6.71	0.00	Cluster 3
C2	2.24	1.41	7.28	**Cluster 2**

✅ Step 2: Cluster Assignment (Round 1)

Cluster	Points Assigned
Cluster 1 (A1)	A1
Cluster 2 (B1)	A3, B1, B2, C2
Cluster 3 (C1)	A2, B3, C1

✅ Step 3: Compute New Centroids (End of Round 1)

- **Cluster 1:** Only A1 = (2,10) → Centroid stays the same.
- **Cluster 2 (A3, B1, B2, C2):**

$$\text{New X} = \frac{8+5+7+4}{4} = \frac{24}{4} = 6.0 \quad \text{New Y} = \frac{4+8+5+9}{4} = \frac{26}{4} = 6.5$$

New Centroid = (6.0, 6.5)

- **Cluster 3 (A2, B3, C1):**

$$\text{New X} = \frac{2+6+1}{3} = \frac{9}{3} = 3.0 \quad \text{New Y} = \frac{5+4+2}{3} = \frac{11}{3} \approx 3.67$$

New Centroid = (3.0, 3.67)

✅ Final Answer (After Round 1)

Cluster Assignments:

- **Cluster 1:** A1
- **Cluster 2:** A3, B1, B2, C2
- **Cluster 3:** A2, B3, C1

New Centroids After Round 1:

Cluster	New Centroid Coordinates
Cluster 1	(2.0, 10.0)
Cluster 2	(6.0, 6.5)
Cluster 3	(3.0, 3.67)

Q6.Explain the following Text Analysis steps with suitable example [9] i) Part-of-speech(POS)tagging ii) Lemmatization

=>

✓ i) Part-of-Speech (POS) Tagging *(4.5 Marks)*

Definition:

POS tagging is the process of **assigning each word in a sentence a grammatical category**, such as noun, verb, adjective, etc., based on **context**.

Purpose:

- Helps understand the **syntactic structure** of text.
- Essential for downstream tasks like **Named Entity Recognition, Parsing, and Information Extraction**.

Common POS Tags:

Tag	Description	Example
NN	Noun (Singular)	“dog”, “city”
NNS	Noun (Plural)	“cars”
VB	Verb (base)	“eat”, “go”
VBD	Verb (past)	“ate”, “ran”
JJ	Adjective	“blue”, “fast”
RB	Adverb	“quickly”

Example:

Sentence:
"The quick brown fox jumps over the lazy dog."

POS Tags:

- The/DT
- quick/JJ
- brown/JJ
- fox/NN
- jumps/VBZ
- over/IN
- the/DT

- lazy/JJ
- dog/NN

Each word is tagged with its grammatical role in context.

Tools Used:

- **NLTK (`pos_tag`)**
- **spaCy**
- **Stanford POS Tagger**

✅ ii) Lemmatization *(4.5 Marks)*

Definition:

Lemmatization is the process of **reducing a word to its base or dictionary form** (lemma), while **considering the context and part of speech**.

Difference from Stemming:

- **Stemming:** crude, removes suffixes blindly → "running" → "runn"
- **Lemmatization:** considers grammar → "running" → "run"

Purpose:

- Helps standardize words to their canonical form.
- Important for improving accuracy in **text classification, search, and NLP models.**

Example:

Input sentence:
"He studies and was running in the morning."

Lemmatized output:

- "studies" → **"study"**
- "was" → **"be"**
- "running" → **"run"**

Note: Lemmatization uses the POS tag to determine the correct lemma.

Tools Used:

- **NLTK WordNet Lemmatizer**
- **spaCy**
- **TextBlob**

✅ Combined Example (POS + Lemmatization):

Sentence:
"The dogs are running quickly in the field."

- POS Tags:
  - dogs/NNS → noun plural
  - running/VBG → verb gerund
  - quickly/RB → adverb
- Lemmatization:
  - "dogs" → "dog"
  - "running" → "run"
  - "are" → "be"

✅ Conclusion:

- **POS tagging** identifies the grammatical role of words, enabling deeper syntactic analysis.
- **Lemmatization** reduces words to their root forms by leveraging POS information, leading to better text normalization and analysis.

These are **critical steps in NLP pipelines** for building robust text processing and machine learning models.

Q7. Given the confusion matrix, Calculate Accuracy, Precision, Recall, Error

rate with description on Diabetic Risk. [8]

		Predicted classes	
	Classes	Diabetic Risk -Yes	Diabetic Risk -No
Actual classes	Diabetic Risk-Yes	90	210
	Diabetic Risk-No	140	9560

=>

Q7. Given the confusion matrix, calculate Accuracy, Precision, Recall, and Error Rate. Also, describe each metric. [8 Marks]

✅ **1. Confusion Matrix (as per image):**

	Predicted: Yes	Predicted: No
Actual: Yes (Positive)	TP = 90	FN = 210
Actual: No (Negative)	FP = 140	TN = 9560

✅ 2. Definitions and Formulas

🔹 a) Accuracy

Definition: Proportion of total correct predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
$$= \frac{90 + 9560}{90 + 9560 + 140 + 210} = \frac{9650}{10000} = 0.965$$

✅ Accuracy = 96.5%

🔹 b) Precision (for 'Yes')

Definition: Out of all predicted positive (Yes) cases, how many were actually positive.

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{90}{90 + 140} = \frac{90}{230} \approx 0.391$$

✅ Precision ≈ 39.1%

🔹 c) Recall (Sensitivity or True Positive Rate)

Definition: Out of all actual positive (Yes) cases, how many were correctly predicted.

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{90}{90 + 210} = \frac{90}{300} = 0.30$$

✅ Recall = 30.0%

🔹 d) Error Rate

Definition: Proportion of incorrect predictions.

$$\text{Error Rate} = 1 - \text{Accuracy} = 1 - 0.965 = 0.035$$

✅ Error Rate = 3.5%

✅ 3. Interpretation of Results

- **High Accuracy (96.5%)** means the model is correct in most predictions.
- **Low Precision (39.1%)** means many of the predicted "Yes" cases are false positives.
- **Low Recall (30.0%)** indicates the model misses many actual diabetic risk cases (high false negatives).
- **Error Rate (3.5%)** shows how often the model makes incorrect predictions.

✅ 4. Conclusion

While the **model is accurate overall**, it performs **poorly in identifying diabetic risk cases**, which is critical in healthcare. The model should be improved to **increase recall and precision** for better medical reliability.

Q8. Explain the Text Preprocessing steps with suitable example. [9]

=>

✅ 1. Introduction to Text Preprocessing

Text preprocessing is the process of **cleaning and preparing raw text data** before it is used for Natural Language Processing (NLP) or machine learning. Since text is unstructured, it must be **transformed into structured and analyzable format**.

✅ 2. Common Text Preprocessing Steps *(with Examples)*

◆ a) Lowercasing

- **Purpose:** Standardizes the text by converting all characters to lowercase.
- **Example:**
  Input: `"India is Great"` → Output: `"india is great"`

◆ b) Removing Punctuation

- **Purpose:** Removes unnecessary characters that do not add value to analysis.
- **Example:**
  Input: `"Hello, world!"` → Output: `"hello world"`

◆ c) Tokenization

- **Purpose:** Splits text into **individual words or tokens**.
- **Example:**
  Input: `"Data Science is fun"` → Output: `["Data", "Science", "is", "fun"]`

◆ d) Removing Stop Words

- **Purpose:** Removes **common words** (e.g., "the", "is", "in") that do not carry meaningful information.
- **Example:**
  Input: `"The sun is bright"` → Output: `["sun", "bright"]`

◆ e) Stemming

- **Purpose:** Reduces words to their **root form** by removing suffixes.
- **Example:**
  `"playing"`, `"played"` → `"play"`
  (Note: Output may not be a valid word)

◆ f) Lemmatization

- **Purpose:** Converts words to their **base dictionary form (lemma)** using grammar.
- **Example:**
  `"was"` → `"be"`, `"better"` → `"good"`

(Lemmatization is preferred over stemming as it gives meaningful base words.)

◆ g) Removing Numbers and Special Characters

- **Purpose:** Cleans irrelevant content from the text.
- **Example:**
  Input: `"Order #123 was placed on 20/04/2025"` → Output: `"order placed"`

◆ h) Normalization / Spelling Correction (Optional)

- **Purpose:** Corrects misspelled words and normalizes repeated characters.

- **Example:**
  `"looove"` → `"love"`, `"recieve"` → `"receive"`

✓ 3. Combined Example:

Raw Text:
`"The QUICK brown foxes, jumped over 13 lazy dogs!!!"`

After Preprocessing (lowercase, punctuation removal, stopword removal, lemmatization):
→ `"quick brown fox jump lazy dog"`

✓ 4. Tools Used in Text Preprocessing

- **Python Libraries:**
  - `NLTK` (Natural Language Toolkit)
  - `spaCy`
  - `re` (regex) for pattern matching
  - `TextBlob`

✓ 5. Importance of Text Preprocessing

- Converts unstructured text into structured format.
- Reduces noise and improves model accuracy.
- Essential for **sentiment analysis**, **text classification**, **chatbots**, etc.

Q9. What is text processing? Explain TF-IDF with example. [8]

=>

✓ 1. What is Text Processing? *(3 Marks)*

Text Processing refers to the set of techniques used to **clean, transform, and represent raw textual data** so that it can be analyzed or fed into machine learning models.

◆ Purpose of Text Processing:

- Convert unstructured text into a **structured format**
- Prepare data for **Natural Language Processing (NLP)** tasks
- Remove **noise** and extract **meaningful features** from text

---

◆ Common Text Processing Steps:

Step	Description
Lowercasing	Converts all text to lowercase
Tokenization	Splits text into words or tokens
Removing stopwords	Removes common, unimportant words
Lemmatization	Converts words to their base form
Vectorization	Converts text into numeric features

✅ 2. What is TF-IDF? (Term Frequency – Inverse Document Frequency) *(5 Marks)*

Definition:

TF-IDF is a **numerical statistic** used in text mining to **evaluate the importance of a word** in a document **relative to a corpus**. It improves upon raw frequency (like in Bag of Words) by reducing the weight of **commonly occurring terms** and emphasizing **rare, meaningful terms**.

✅ Formula:

1. **TF (Term Frequency):**

$$TF(t,d) = \frac{\text{No. of times term } t \text{ appears in document } d}{\text{Total terms in document } d}$$

2. **IDF (Inverse Document Frequency):**

$$IDF(t) = \log\left(\frac{N}{df_t}\right)$$

Where:

- N = Total number of documents
- df_t = Number of documents containing term t

3. **TF-IDF Score:**

$$TFIDF(t,d) = TF(t,d) \times IDF(t)$$

✅ Example:

Let's consider 2 documents:

- **Doc1:** "Data science is fun"
- **Doc2:** "Science is powerful"

Vocabulary: ["data", "science", "is", "fun", "powerful"]

TF Table:

Term	TF in Doc1	TF in Doc2
data	1/4 = 0.25	0
science	1/4 = 0.25	1/3 ≈ 0.33
is	1/4 = 0.25	1/3 ≈ 0.33
fun	1/4 = 0.25	0
powerful	0	1/3 ≈ 0.33

IDF Calculation (Assuming total 2 documents):

Term	Document Frequency (df)	IDF = log(2/df)
data	1	log(2/1) = 0.301
science	2	log(2/2) = 0
is	2	log(2/2) = 0
fun	1	log(2/1) = 0.301
powerful	1	log(2/1) = 0.301

TF-IDF for "data" in Doc1 = 0.25 × 0.301 = 0.075

TF-IDF for "science" = 0 (as IDF is 0)

✅ Interpretation:

- Common words like "is", "science" have low/no weight.
- Rare and document-specific words like "data", "fun", "powerful" have **higher importance**.

✓ Conclusion:

Text processing is essential to convert raw textual data into analyzable form. **TF-IDF** improves feature extraction by giving more weight to **important and rare terms**, making it highly effective for tasks like **text classification, document similarity, and search ranking**.

Q10 .With suitable example ,explain the steps involved in k-means algorithm. [9]

=>

✓ 1. Introduction to K-Means Algorithm *(1 Mark)*

K-Means is a **centroid-based, unsupervised clustering algorithm** used to partition a given dataset into **K distinct clusters**, where each point belongs to the cluster with the **nearest mean (centroid)**.

✓ 2. Key Idea

- Group data based on **similarity**.
- Each cluster is defined by a **centroid**, which is the **mean of all points** in the cluster.

✓ 3. Steps Involved in K-Means Algorithm *(5 Marks)*

◆ Step 1: Choose the number of clusters (K)

Decide how many clusters you want to find (e.g., K = 2, 3, 4, etc.)

◆ Step 2: Initialize centroids

Randomly select **K data points** as the initial centroids (means of clusters).

🔹 Step 3: Assign each point to the nearest centroid

- Calculate the **Euclidean distance** from each point to each centroid.
- Assign the point to the cluster with the **minimum distance**.

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

◆ Step 4: Recalculate the centroids

- For each cluster, calculate the **mean of all points** assigned to that cluster.
- Update the centroid to this new mean.

◆ Step 5: Repeat until convergence

- Repeat Steps 3 and 4 until:
  - Points **no longer change clusters**, or
  - **Centroids stop moving**

✅ 4. Example *(with 5 points, K = 2) (2 Marks)*

🔹 Given Points:

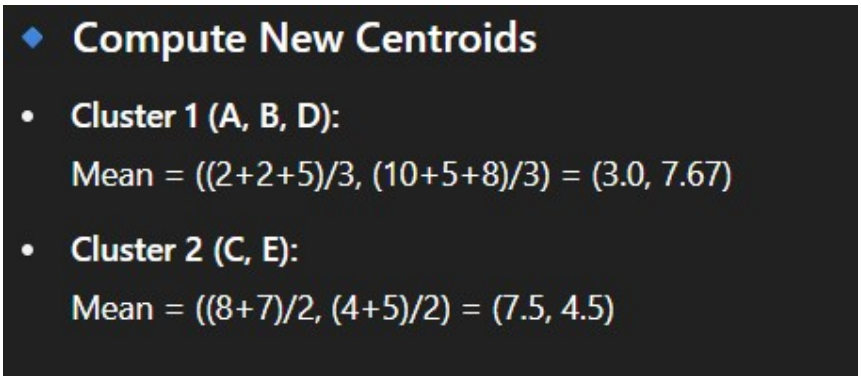
A(2, 10), B(2, 5), C(8, 4), D(5, 8), E(7, 5)

Let initial centroids:

- C1 = A(2, 10)
- C2 = C(8, 4)

🔹 Round 1: Assign points to nearest centroid

Point	Dist to C1	Dist to C2	Assigned Cluster
A	0.00	7.21	Cluster 1
B	5.00	6.00	Cluster 1
C	7.21	0.00	Cluster 2
D	3.61	3.61	Cluster 1 (tie)
E	6.08	1.41	Cluster 2



✓ 5. Advantages of K-Means *(0.5 Marks)*

- Simple and fast
- Works well with large datasets
- Easily interpretable

✓ 6. Limitations of K-Means *(0.5 Marks)*

- Need to specify K in advance
- Sensitive to outliers and initialization
- Works best with spherical-shaped clusters

✓ Conclusion *(0.5 Marks)*

K-Means is a powerful clustering technique that **iteratively refines clusters** by minimizing distances between data points and centroids. With the right choice of K, it can reveal meaningful patterns in the data.

Q11. Define following terms with respect to confusion matrix : [8] i) Accuracy ii) Precision iii) Recall iv) AUC-ROC

=>

✓ Confusion Matrix Overview

Before defining the metrics, let's revisit a standard **confusion matrix** used for evaluating classification models:

	Predicted Positive	Predicted Negative
Actual Positive (P)	True Positive (TP)	False Negative (FN)
Actual Negative (N)	False Positive (FP)	True Negative (TN)

- **TP**: Model correctly predicts Positive
- **TN**: Model correctly predicts Negative
- **FP**: Model incorrectly predicts Positive
- **FN**: Model incorrectly predicts Negative

✓ i) Accuracy *(2 Marks)*

Definition:

Accuracy measures the **overall correctness** of the model across both classes. It shows the proportion of **total correct predictions** to the **total number of cases**.

Formula:

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

Example:

If out of 1000 predictions, the model got 950 correct →

$$\text{Accuracy} = \frac{950}{1000} = 95\%$$

Interpretation:

High accuracy indicates good overall performance, but it can be **misleading in imbalanced datasets** (e.g., detecting rare diseases).

✓ ii) Precision *(2 Marks)*

Definition:

Precision is the ratio of **true positive predictions** to **all predicted positives**. It answers the question:

"Of all the instances the model predicted as positive, how many were actually positive?"

Formula:

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

Example:

If the model predicts 100 positives and 80 are correct:

$$\text{Precision} = \frac{80}{100} = 80\%$$

Interpretation:

High precision means the model makes **few false positive errors**.
It is **very important in spam detection, fraud detection**, where false positives are costly.

✓ iii) Recall (Sensitivity / True Positive Rate) *(2 Marks)*

Definition:

Recall is the ratio of **true positive predictions** to all actual positives. It answers the question:

"Of all the actual positive cases, how many did the model correctly identify?"

Formula:

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

Example:

If there are 100 actual positives and the model identifies 75:

$$\text{Recall} = \frac{75}{100} = 75\%$$

Interpretation:

High recall means the model **detects most of the actual positive cases**, which is **critical in medical diagnosis**, fraud detection, and safety systems.

✓ iv) AUC-ROC (Area Under the ROC Curve) *(2 Marks)*

Definition:

AUC-ROC is a **performance metric** that measures the model's ability to distinguish between the classes.

- **ROC curve (Receiver Operating Characteristic)** plots:
  - **True Positive Rate (TPR or Recall)**
  - **vs.**
  - **False Positive Rate (FPR = FP / (FP + TN))**

AUC (Area Under the Curve):

$$\text{AUC} = \int \text{ROC curve}$$

- AUC ranges from **0 to 1**:
  - **AUC = 1.0** → Perfect classifier
  - **AUC = 0.5** → No discrimination (random guessing)
  - **AUC < 0.5** → Worse than random

Interpretation:

- A higher AUC indicates the model has **high separability** between classes.
- It is useful for **imbalanced datasets** and **binary classification problems.**

Example:

If AUC = 0.93 → The model has a 93% chance of ranking a randomly chosen positive instance higher than a randomly chosen negative one.

✅ Conclusion

These four metrics — **Accuracy, Precision, Recall, and AUC-ROC** — are essential to **evaluate classification models** effectively:

- Use **accuracy** for balanced datasets,
- Use **precision** when **false positives are costly**,
- Use **recall** when **missing positives is dangerous**,
- Use **AUC-ROC** to understand **model performance across thresholds**.

Q12. Explain k-fold Cross Validation & Random Subsampling. [9]

=>

✓ 1. Introduction to Model Evaluation Techniques *(1 Mark)*

In machine learning, model evaluation is crucial to estimate how well a model will perform on **unseen data**. Two widely used **resampling techniques** are:

- **K-Fold Cross Validation**
- **Random Subsampling**

These methods reduce **bias and variance** by training and testing the model on multiple data partitions.

✓ 2. K-Fold Cross Validation *(4 Marks)*

◆ Definition:

K-Fold Cross Validation divides the dataset into **K equal-sized subsets (folds)**. The model is trained on **K-1 folds** and tested on the **remaining fold**. This process repeats **K times**, with each fold used once as the test set.

◆ Steps:

1. Split the dataset into **K folds**.
2. For each iteration i=1i = 1i=1 to KKK:
    - Use fold iii as **validation/test set**
    - Use remaining K−1K - 1K−1 folds as **training set**
3. Compute the **average performance** across all K runs.

◆ Diagram:

For **K = 5**, 5 iterations:

Fold #	Train on Folds	Test on Fold
1	2, 3, 4, 5	1
2	1, 3, 4, 5	2
3	1, 2, 4, 5	3

Fold #	Train on Folds	Test on Fold
4	1, 2, 3, 5	4
5	1, 2, 3, 4	5

◆ Advantages:

- Reduces **overfitting** risk.
- Gives a **robust estimate** of model performance.
- **Efficient for small datasets**.

◆ Limitations:

- Slightly **computationally expensive** (especially for large K).
- Requires **shuffling** of data to avoid biased splits.

◆ Example:

Dataset of 100 samples, **K = 5**
→ Each fold has 20 samples
→ Model trained and validated 5 times
→ Final score = average of all 5 test scores

✓ 3. Random Subsampling (Repeated Holdout Method) *(4 Marks)*

◆ Definition:

Random Subsampling (also called repeated random train/test splits) splits the dataset **randomly multiple times** into **training and test sets**, evaluates the model on each split, and then **averages the results**.

◆ Steps:

1. Randomly split dataset into training and test sets (e.g., 70% train, 30% test).
2. Train the model and evaluate accuracy.
3. Repeat the process **N times** (e.g., 10 times).
4. Calculate the **average accuracy** or error rate.

◆ Advantages:

- Simple and flexible.
- Can handle **large datasets efficiently**.
- Useful when full K-Fold is too costly.

◆ Limitations:

- Some data points **may never be tested**.
- Some points may appear **multiple times** in the test set.
- Less stable than K-Fold unless repeated many times.

◆ Example:

Dataset with 1000 records:

- Repeat 10 random splits (e.g., 80% train, 20% test)
- Final accuracy = average of 10 evaluations

✓ 4. Conclusion

Both **K-Fold Cross Validation** and **Random Subsampling** are powerful resampling techniques:

Method	Best For	Type
K-Fold CV	**Smaller datasets**, robust evaluation	Deterministic
Random Subsampling	**Large datasets**, flexible	Randomized

They help estimate a model's **true performance**, reduce **overfitting**, and guide **model selection**.

Q13. Suppose that the given data the taste is to cluster points (With (x.y) representing location) into three cluster, where the points are. A1(2,10), A2(2,5), A3(8,4), B1 (5,8) B2(7,5) B3(6,4), C1(1,2), C2(4,9) The distance function is Euclidean distance suppose initially we assign A1, B1 and C1 as the center

of each cluster, respectively. use the K-means algorithm to show only the three cluster centers after the first round of execution with steps. [9]

=>

Q13. Apply K-Means Clustering (1st Round) on Given Points with Initial Centers A1, B1, C1. [9 Marks]

✓ 1. Given Data Points:

Point	Coordinates
A1	(2, 10)
A2	(2, 5)
A3	(8, 4)
B1	(5, 8)
B2	(7, 5)
B3	(6, 4)
C1	(1, 2)
C2	(4, 9)

✅ **2. Initial Cluster Centers:**

- Cluster 1 Center = A1 = (2, 10)
- Cluster 2 Center = B1 = (5, 8)
- Cluster 3 Center = C1 = (1, 2)

✅ **3. Use Euclidean Distance Formula:**

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

Now compute the distance from each point to the 3 initial centroids.

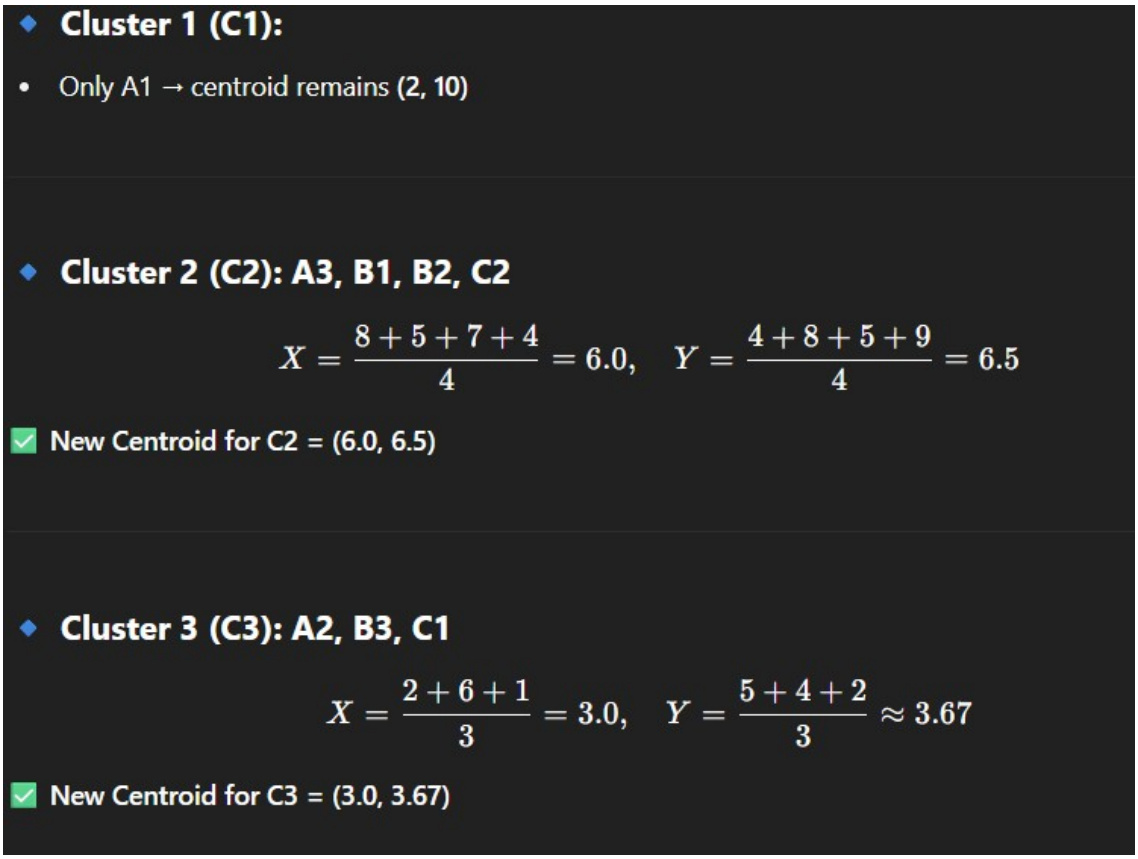
✅ 4. Distance Table and Cluster Assignments (1st Iteration)

Point	To A1 (2,10)	To B1 (5,8)	To C1 (1,2)	Assigned Cluster
A1	0.00	3.61	8.06	C1
A2	5.00	3.61	3.16	C3
A3	7.21	4.24	7.07	C2
B1	3.61	0.00	6.71	C2
B2	5.83	3.16	6.08	C2
B3	6.71	4.12	5.00	C3
C1	8.06	6.71	0.00	C3
C2	2.24	1.41	7.28	C2

✅ 5. Cluster Grouping After First Assignment

Cluster	Assigned Points
C1	A1
C2	A3, B1, B2, C2
C3	A2, B3, C1

✅ 6. Recalculate New Centroids (Mean of Each Cluster)



✓ 7. Final Answer – New Centroids After First Round:

Cluster	New Centroid Coordinates
C1	(2.0, 10.0)
C2	(6.0, 6.5)
C3	(3.0, 3.67)

✓ 8. Conclusion:

After the **first iteration** of K-Means clustering:

- Points are grouped based on **minimum Euclidean distance**.
- New cluster centers are calculated as the **mean** of the assigned points.
- The algorithm is ready for the **next iteration** (not required here).

Q14. Explain the following text analysis steps with suitable example. [8] i) Part of speech (POS) tagging ii) Lemmatization iii) Stemming

=>

✓ i) Part of Speech (POS) Tagging *(3 Marks)*

Definition:

Part-of-Speech (POS) tagging is the process of **assigning grammatical categories** (parts of speech) such as noun, verb, adjective, adverb, etc., to each word in a sentence based on its **context and meaning**.

Purpose:

- Understand the **syntactic structure** of sentences.
- POS tags help with **sentence parsing, lemmatization, information extraction,** and **machine translation.**

Common POS Tags:

Tag	Meaning	Example
NN	Noun, singular	dog, book
NNS	Noun, plural	dogs, books
VB	Verb, base	run, eat
VBD	Verb, past	ran, ate
JJ	Adjective	blue, fast
RB	Adverb	quickly, silently

Example:

Sentence:
"The quick brown fox jumps over the lazy dog."

POS tagging result:

- The/**DT**
- quick/**JJ**
- brown/**JJ**
- fox/**NN**
- jumps/**VBZ**

- over/**IN**
- the/**DT**
- lazy/**JJ**
- dog/**NN**

Here:

- DT: Determiner
- JJ: Adjective
- NN: Noun
- VBZ: Verb, 3rd person singular

Tools for POS Tagging:

- `nltk.pos_tag()` in Python
- `spaCy` (provides context-aware tagging)
- Stanford NLP POS Tagger

✓ ii) Lemmatization *(2.5 Marks)*

Definition:

Lemmatization is the process of reducing words to their **dictionary base form (lemma)** using **vocabulary and morphological analysis**. It considers **context** and the **part of speech** of the word.

Purpose:

- Normalize words for better **text analysis and machine learning**.
- Ensures consistent representation of words for **search engines, classifiers, and summarizers**.

How Lemmatization Works:

Unlike stemming, lemmatization uses the **context** of the word (usually the POS tag) to return a **meaningful base word**.

Examples:

Original Word	Lemma	POS Needed?
studies	study	noun/verb

Original Word	Lemma	POS Needed?
was	be	verb
better	good	adjective
mice	mouse	noun
running	run	verb

Tools for Lemmatization:

- `WordNetLemmatizer` from NLTK
- `spaCy`'s built-in lemmatizer
- TextBlob

✅ iii) Stemming *(2.5 Marks)*

Definition:

Stemming is a rule-based process that **removes prefixes and suffixes** from words to reduce them to their **root form** (stem), often **without regard to context or grammar**.

Purpose:

- Reduces vocabulary size.
- Useful in **information retrieval, search engines, and clustering**.

Difference from Lemmatization:

Feature	Stemming	Lemmatization
Uses grammar	❌ No	✅ Yes
Output	May not be valid	Valid dictionary word
Speed	Faster	Slower
Accuracy	Less accurate	More accurate

Examples:

Word	Stem
playing	play
played	play
studies	studi
flying	fli
connection	connect

As seen, stemming may produce **non-meaningful or partial words** (like “studi”).

Popular Stemming Algorithms:

- **Porter Stemmer**
- **Lancaster Stemmer**
- **Snowball Stemmer**

✅ Combined Example:

Raw sentence:
"The cats were running around happily."

- **After POS tagging:**
  - cats/NN, were/VBD, running/VBG, happily/RB
- **Lemmatization result:**
  - "cats" → "cat", "were" → "be", "running" → "run"
- **Stemming result:**
  - "cats" → "cat", "running" → "run", "happily" → "happi"

✅ Conclusion:

Text analysis steps like **POS tagging, lemmatization, and stemming** are foundational in **Natural Language Processing (NLP)**:

- **POS tagging** helps in **understanding grammar and structure**.
- **Lemmatization** gives a **context-aware base form**.
- **Stemming** gives a **quick, less accurate root**.

These steps are essential for building **accurate, meaningful, and efficient text mining and machine learning models**.

Q15. Write short note on i) Time series Analysis ii) TF - IDF. [9]

=>

✓ i) Time Series Analysis *(4.5 Marks)*

Definition:

Time Series Analysis is a statistical technique used to **analyze data points collected or recorded at specific time intervals** (e.g., hourly, daily, monthly). It focuses on **patterns, trends, seasonality, and forecasting** over time.

Key Features of Time Series:

Feature	Description
Trend	Long-term increase or decrease in data
Seasonality	Periodic fluctuations (e.g., monthly sales peaks)
Cyclic	Repeating patterns over irregular time intervals
Noise	Random variations in data
Stationarity	Constant mean and variance over time (used in modeling like ARIMA)

Common Techniques:

- **Moving Average**
- **Exponential Smoothing**
- **ARIMA (Auto-Regressive Integrated Moving Average)**
- **Seasonal Decomposition**

Example Use Cases:

- Stock price prediction
- Weather forecasting
- Electricity demand prediction

- Sales trend analysis

Visualization:

Time series data is plotted with **time on the x-axis** and the **variable of interest on the y-axis**.

Applications in Machine Learning:

- Time series forecasting (e.g., LSTM models)
- Anomaly detection
- Financial modeling

✅ ii) TF-IDF (Term Frequency – Inverse Document Frequency) *(4.5 Marks)*

Definition:

TF-IDF is a numerical measure used in **text analytics** and **information retrieval** to evaluate the **importance of a word** in a document **relative to the entire corpus**. It balances both **local frequency (TF)** and **global rarity (IDF)**.

Formula:

1. Term Frequency (TF):

$$TF(t,d) = \frac{\text{No. of times term } t \text{ appears in document } d}{\text{Total terms in document } d}$$

2. Inverse Document Frequency (IDF):

$$IDF(t) = \log\left(\frac{N}{df_t}\right)$$

Where:

- N = Total number of documents
- df_t = Number of documents containing term t

3. TF-IDF:

$$TFIDF(t,d) = TF(t,d) \times IDF(t)$$

Purpose:

- TF-IDF helps **filter out common words** (like "the", "is") and **highlight important keywords**.
- Converts text into **numerical vectors** for use in ML models (e.g., for classification or clustering).

Example:

Corpus:

- Doc1: "Data science is awesome"
- Doc2: "Science is powerful"

Vocabulary: data, science, is, awesome, powerful

- “science” appears in both → low IDF
- “data” and “powerful” are unique → high TF-IDF

Applications:

- Document classification
- Information retrieval (search engines)
- Text clustering
- Keyword extraction

Advantages:

- Easy to compute
- Effective for **feature extraction in NLP**
- Weights terms based on **relevance**

✓ Conclusion:

- **Time Series Analysis** deals with **temporal data**, focusing on **patterns over time** for forecasting and trend detection.
- **TF-IDF** is a powerful technique for **text data representation**, improving the relevance of terms by considering both frequency and rarity.

Q16. What is clustering? With suitable example explain the steps involved in k - means algorithm. [9]

=>

✓ 1. What is Clustering? *(2 Marks)*

Clustering is an **unsupervised machine learning technique** that groups similar data points into clusters based on certain features, **without using labeled data**.

Each cluster contains data points that are **similar to each other** and **dissimilar to those in other clusters**.

Purpose of Clustering:

- Discover hidden patterns or groupings in data
- Reduce complexity and perform segmentation
- Used in market segmentation, image compression, document grouping, etc.

Real-World Examples:

- Grouping customers based on purchasing behavior
- Clustering news articles by topic
- Segmenting medical images into regions

✓ 2. K-Means Clustering Algorithm *(5 Marks)*

K-Means is a popular **centroid-based clustering algorithm** that partitions the dataset into **K non-overlapping clusters** based on **distance to cluster centroids**.

◆ Steps Involved in K-Means Algorithm:

Step 1: Choose the number of clusters (K)

- Decide how many clusters you want to find (e.g., K = 2, 3, etc.)

Step 2: Initialize K centroids

- Randomly select **K points** from the dataset as initial **centroids (cluster centers)**.

Step 3: Assign each point to the nearest centroid

- Use **Euclidean distance** to assign each data point to the **closest cluster**:

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

Step 4: Recalculate the centroids

- For each cluster, compute the new centroid as the **mean** of all points in that cluster.

Step 5: Repeat

- Repeat Steps 3 and 4 until:
  - Cluster assignments **no longer change**, or
  - Centroids **converge (no movement)**

✓ 3. Example of K-Means (K = 2) *(2 Marks)*

Given Points:

- A(1, 2), B(1, 4), C(5, 8), D(6, 9)

Step 1: Choose K = 2

Initial centroids: A(1, 2) and C(5, 8)

Step 2: Compute distances and assign clusters

Point	Dist to A (1,2)	Dist to C (5,8)	Assigned Cluster
A	0.00	7.21	Cluster 1
B	2.00	5.00	Cluster 1

Point	Dist to A (1,2)	Dist to C (5,8)	Assigned Cluster
C	7.21	0.00	Cluster 2
D	8.60	1.41	Cluster 2

Step 3: Recalculate centroids

- Cluster 1: Mean of A & B = (1, 3)
- Cluster 2: Mean of C & D = (5.5, 8.5)

Repeat again if needed.

✓ 4. Advantages of K-Means:

- Simple and fast
- Scales well with large datasets
- Works well when clusters are **spherical and well-separated**

✓ 5. Limitations of K-Means:

- Need to predefine **K**
- Sensitive to **initial centroid selection**
- Doesn't work well with **non-spherical or overlapping clusters**
- Affected by **outliers**

✓ Conclusion:

Clustering helps in **grouping similar data** and discovering underlying patterns.
K-Means is a widely used and efficient clustering algorithm that groups data based on **minimizing intra-cluster distances**, and is suitable for many practical applications like customer segmentation, document classification, and more.

Q17. Write short note on i) Confusion matrix ii) AVC - ROC curve [9]

=>

✓ i) Confusion Matrix *(4.5 Marks)*

Definition:

A **confusion matrix** is a **performance evaluation table** used to measure the performance of a **classification model**. It compares the **actual target values** with the **model's predicted values**.

Structure of a Binary Confusion Matrix:

	Predicted: Positive	Predicted: Negative
Actual: Positive	**True Positive (TP)**	**False Negative (FN)**
Actual: Negative	**False Positive (FP)**	**True Negative (TN)**

✅ Terminologies:

- **True Positive (TP):** Model correctly predicted positive
- **False Positive (FP):** Model wrongly predicted positive
- **True Negative (TN):** Model correctly predicted negative
- **False Negative (FN):** Model wrongly predicted negative

✅ Metrics Derived from Confusion Matrix:

Metric	Formula	Description
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Overall correctness of the model
Precision	$\frac{TP}{TP+FP}$	Correctness of positive predictions
Recall	$\frac{TP}{TP+FN}$	Ability to find all actual positives
F1-Score	$2 \times \frac{Precision \times Recall}{Precision+Recall}$	Balance between precision and recall

✅ Example:

Suppose the confusion matrix is:

	Predicted: Yes	Predicted: No
Actual: Yes	80 (TP)	20 (FN)
Actual: No	30 (FP)	870 (TN)

✅ Example:

Suppose the confusion matrix is:

	Predicted: Yes	Predicted: No
Actual: Yes	80 (TP)	20 (FN)
Actual: No	30 (FP)	870 (TN)

Then:

Then:

- Accuracy = $\frac{80+870}{1000} = 0.95$ or 95%
- Precision = $\frac{80}{80+30} = 0.727$
- Recall = $\frac{80}{80+20} = 0.80$

✅ ii) AUC - ROC Curve *(4.5 Marks)*

Definition:

The **ROC (Receiver Operating Characteristic) curve** is a graph that shows the **performance of a binary classifier** at all classification thresholds.
AUC (Area Under the Curve) is a single scalar value that summarizes the **overall ability of the classifier** to distinguish between classes.

✅ **Axes of ROC Curve:**

- **X-axis:** False Positive Rate (FPR)

$$FPR = \frac{FP}{FP + TN}$$

- **Y-axis:** True Positive Rate (Recall or Sensitivity)

$$TPR = \frac{TP}{TP + FN}$$

✅ Interpretation of AUC Values:

AUC Score	Model Performance
1.0	Perfect classification
0.9–1.0	Excellent
0.8–0.9	Good
0.7–0.8	Fair
0.5–0.6	Poor
0.5	No discrimination (random)

✅ Advantages of AUC-ROC:

- Independent of the classification threshold
- Works well with **imbalanced datasets**
- Helps in selecting optimal model and threshold

✅ Example:

Suppose a classifier achieves:

- TPR = 0.85
- FPR = 0.10
  → ROC point = (0.10, 0.85)
  → Area under the ROC curve (AUC) ≈ 0.93 → Very good performance

---

✅ Conclusion:

- The **confusion matrix** is essential to measure **individual classification errors** (TP, FP, FN, TN) and to derive other metrics like precision, recall, and accuracy.
- The **AUC-ROC curve** provides a **threshold-independent** measure of model performance and helps in **visual comparison** of multiple classifiers.

Q18. Discuss Holdout method and Random Sub Sampling methods. [9]

=>

✓ 1. Introduction to Model Evaluation Methods *(1 Mark)*

When building machine learning models, it is essential to evaluate their performance on **unseen data**. To do this, we split data into different subsets.
Two popular resampling methods used for this are:

- **Holdout Method**
- **Random Subsampling (Repeated Holdout)**

Both aim to estimate the **generalization ability** of the model.

✓ 2. Holdout Method *(4 Marks)*

◆ Definition:

The **Holdout Method** is a simple and fast model evaluation technique where the dataset is **split into two or three parts**:

- **Training set**
- **Validation set (optional)**
- **Test set**

◆ Working:

1. Split the dataset (e.g., 70% for training, 30% for testing).
2. Train the model using the **training set**.
3. Evaluate the model using the **test set**.
4. (Optional) Use the **validation set** for hyperparameter tuning.

◆ Advantages:

- Very easy to implement.
- Works well for **large datasets**.

◆ Limitations:

- **Performance depends on how data is split.**
- May give **high variance results** for small datasets.
- Not every data point is used for both training and testing.

◆ Example:

For a dataset with 1000 records:

- 700 used for training
- 300 used for testing
  → Model is evaluated only once on 300 test records.

✓ 3. Random Subsampling (Repeated Holdout Method) *(4 Marks)*

◆ Definition:

Random Subsampling involves **repeating the holdout method multiple times** with different random splits and then averaging the results.

◆ Working:

1. Repeat the following N times (e.g., N = 10):
   - Randomly split the dataset into training and test sets.
   - Train and evaluate the model.
2. Compute the **average accuracy/error rate** over N iterations.

◆ Advantages:

- Reduces the **bias introduced by a single train/test split**.
- More **stable and reliable** estimate than one-time holdout.

◆ Limitations:

- Some samples may **never be tested**.
- Some test samples may be **repeated across iterations**.
- Computationally more expensive than basic holdout.

◆ Example:

If you repeat random splitting (80% training, 20% testing) 10 times:

- You get 10 performance scores.
- Final model performance = **mean of 10 scores**.

✅ Comparison Table:

Criteria	Holdout Method	Random Subsampling
Splitting	Done once	Done multiple times randomly
Accuracy estimate	Single score	Average over multiple scores
Bias/Variance	High variance	Lower variance
Computation	Fast	Slower (repeated models)

✅ 4. Conclusion:

- The **Holdout Method** is quick and simple but may suffer from high variance.
- **Random Subsampling** improves reliability by averaging across multiple splits.

These techniques are essential for evaluating a model's **real-world generalization ability**.

UNIT – 6

Q1. What is a histogram? How is it used to visualize the distribution of data? How is it different from a density plot? [9]

=>

✅ 1. What is a Histogram? *(3 Marks)*

A **histogram** is a graphical representation used to **visualize the frequency distribution** of a **numerical dataset**. It displays how data is **grouped into continuous intervals or "bins"**, and shows **how many values fall into each bin**.

◆ Key Features of a Histogram:

- The **x-axis** represents intervals (or bins) of values.
- The **y-axis** represents the **frequency** (or count) of data points in each bin.
- Each **bar's height** shows the number of observations within that range.

✅ Example:

Consider student test scores:

- Scores: [50, 60, 65, 70, 75, 80, 85, 90, 95, 100]

A histogram with bins of size 10 may show:

- 50–60: 2 students
- 60–70: 2 students
- 70–80: 2 students
- 80–90: 2 students
- 90–100: 2 students

Each bar corresponds to a score range and shows how many students fall within it.

✅ 2. How Is a Histogram Used to Visualize Data Distribution? *(3 Marks)*

✅ Purposes of Histogram:

1. **Understanding shape of data distribution:**
   - Symmetrical, skewed, uniform, bimodal, etc.
2. **Detecting outliers or gaps:**
   - Gaps or unusually tall/short bars signal anomalies.
3. **Observing central tendency and spread:**
   - Histogram shows where data values are concentrated (e.g., center, left-skewed, right-skewed).

✅ Real-World Applications:

- Analyze income distribution
- Visualize age groups of a population
- Understand sales patterns over price ranges

✅ 3. Difference Between Histogram and Density Plot *(3 Marks)*

Feature	Histogram	Density Plot
Type	**Bar graph** of frequencies	**Smoothed line plot** of probability
Representation	**Counts** in bins	**Probability density** (area = 1)
Shape	May be **jagged** or blocky	**Smooth curve**
Binning	Required (user defines bin width)	Not required
Overlap Comparison	Difficult to compare multiple histograms	Easy to compare multiple distributions

---

✅ Example:

- Histogram: "How many students scored between 60–70?"
- Density Plot: "What is the **likelihood** of a student scoring near 65?"

Q2. What is the Hadoop ecosystem, and what are its primary components? What is MapReduce, and how does it fit into the Hadoop ecosystem?[9]

=>

- Fig. 6.6.1 shows Apache Hadoop ecosystem.

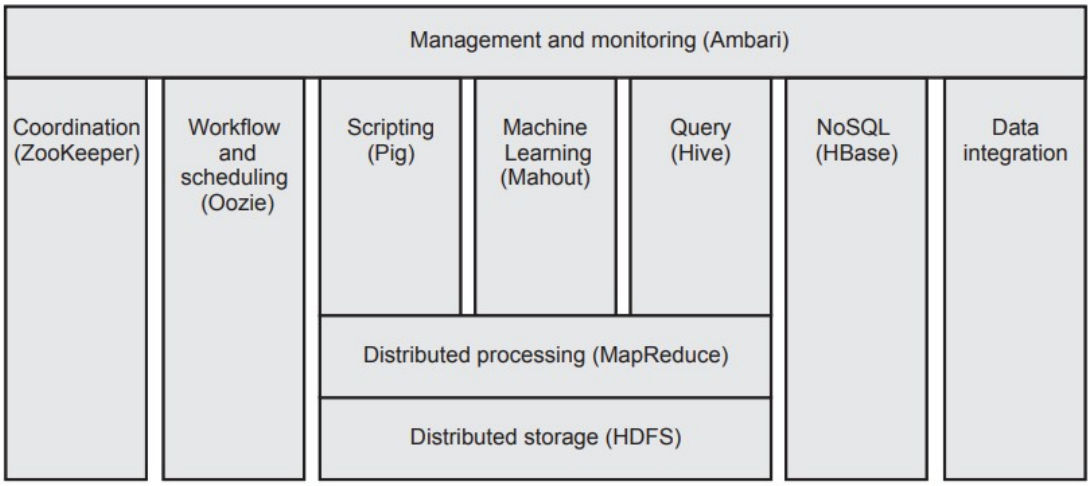


Fig. 6.6.1 : Apache Hadoop ecosystem

- Fig. 6.6.2 shows HDFS archtiecture

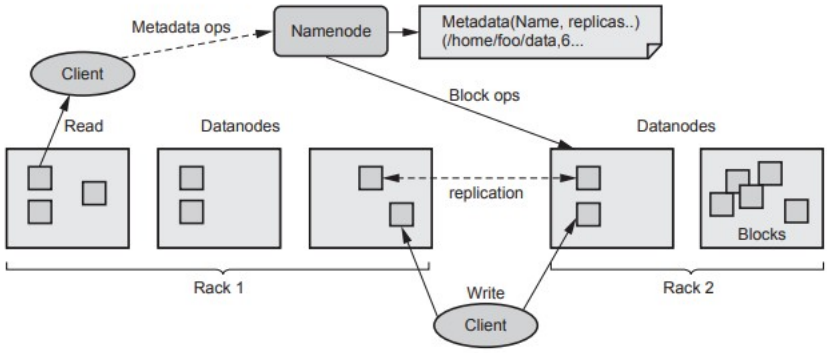


Fig. 6.6.2 Hadoop architecture

Q2. What is the Hadoop ecosystem, and what are its primary components? What is MapReduce, and how does it fit into the Hadoop ecosystem? [9 Marks]

✓ 1. What is Apache Hadoop Ecosystem? *(2 Marks)*

Apache Hadoop is an **open-source framework** developed by the **Apache Software Foundation** for **distributed storage and processing of big data**. The **Hadoop ecosystem** consists of a set of core components and supporting tools that work together to handle **large-scale data efficiently** across a cluster of machines.

It is designed to:

- Store vast amounts of data using **HDFS (Hadoop Distributed File System)**
- Process that data using **MapReduce**
- Support various tools for **data ingestion, querying, machine learning, and workflow management**

✓ 2. Primary Components of the Hadoop Ecosystem *(4 Marks)*

As shown in **Fig. 6.6.1**, the Hadoop ecosystem includes the following:

◆ Core Components:

□ 1. HDFS (Hadoop Distributed File System)

- Used for storing large files by breaking them into blocks and distributing across **DataNodes**.
- Provides **fault tolerance** using **replication**.
- Managed by **NameNode** (metadata) and **DataNodes** (actual data).

□ 2. MapReduce

- Programming model for **parallel processing** of large datasets.
- Works on data stored in HDFS.
- Explained in detail below.

□ 3. YARN (Yet Another Resource Negotiator)

- Manages **resources and job scheduling** in the Hadoop cluster.
- Decouples resource management from processing logic.

◆ Ecosystem Tools:

Category	Tool	Function
Data Query	**Hive**	SQL-like querying for structured data
Data Scripting	**Pig**	High-level scripting language (Pig Latin)
Machine Learning	**Mahout**	Scalable ML algorithms on Hadoop
NoSQL Database	**HBase**	Column-oriented NoSQL database
Workflow Scheduler	**Oozie**	Manages job dependencies and sequences
Coordination Service	**Zookeeper**	Maintains synchronization and configuration between nodes
Monitoring Tool	**Ambari**	Cluster provisioning, monitoring, and management
Data Ingestion	**Sqoop, Flume**	Transfer data from RDBMS (Sqoop), streaming logs (Flume)

✅ Diagram Reference – Fig. 6.6.1 (Apache Hadoop Ecosystem):

This diagram visualizes how tools like Pig, Hive, Mahout, Oozie, etc., work **on top of HDFS and MapReduce**, and are managed centrally by **Ambari**.

✅ 3. What is MapReduce? *(2 Marks)*

MapReduce is the **data processing engine** of Apache Hadoop. It allows **parallel processing** of large datasets in a **distributed manner** using two main phases:

◆ Map Phase:

- Splits input data into chunks.
- Each chunk is processed by a **mapper** to produce **(key, value)** pairs.

◆ Shuffle and Sort Phase:

- Intermediate key-value pairs are grouped and sorted by key.

◆ Reduce Phase:

- The **reducer** aggregates values for each key and produces the final output.

✅ Example (Word Count):

Input:

"Big data is big and useful"

Map Output:
("big",1), ("data",1), ("is",1), ("big",1), ("useful",1)

Reduce Output:
("big",2), ("data",1), ("is",1), ("useful",1)

✅ 4. How MapReduce Fits into the Hadoop Ecosystem *(1 Mark)*

- **MapReduce** works **on top of HDFS**, processing the data stored across nodes.
- Jobs are managed and scheduled by **YARN**.
- High-level tools like **Pig and Hive** internally translate queries into **MapReduce jobs**.

✅ 5. HDFS Architecture Reference – Fig. 6.6.2

- **NameNode**: Stores metadata (file names, blocks, replicas)
- **DataNodes**: Store actual data blocks

- **Clients**: Interact with NameNode for metadata and DataNodes for data
- **Replication**: Data is replicated across DataNodes for fault tolerance

This architecture ensures **reliable, distributed storage**, which is the foundation for **MapReduce processing**.

Q3. What is a box plot? Explain the different components of a box plot? How do you interpret the median, quartiles, and whiskers in a box plot? What does the interquartile range (IQR) represent in a box plot? [9]

=>

✓ 1. What is a Box Plot? *(2 Marks)*

A **box plot** (also known as a **box-and-whisker plot**) is a graphical representation of the **distribution of a dataset** based on a **five-number summary**:

- Minimum
- First Quartile (Q1)
- Median (Q2)
- Third Quartile (Q3)
- Maximum

It is especially useful to:

- **Visualize spread and skewness**
- Detect **outliers**
- Compare **multiple distributions**

✓ 2. Components of a Box Plot *(4 Marks)*

A standard box plot includes the following elements:

Component	Description
Box	Represents the **interquartile range (IQR)** from **Q1 to Q3**
Median (Q2)	The **middle value** (50th percentile), shown as a line inside the box
Whiskers	Extend from the box to the **smallest and largest non-outlier values**
Minimum	The smallest data point within the whisker range
Maximum	The largest data point within the whisker range

Component	Description
Outliers	Data points that fall outside the whiskers (typically marked with dots)

✅ Visual Example:

```
mathematica
CopyEdit
|-----------|==========|-------------|
Min        Q1    Median   Q3        Max
           |---- Box ----|
      |------ Whiskers ------|
```

✅ 3. Interpretation of Components *(2 Marks)*

◆ Median (Q2):

- The **center line** inside the box.
- If the median is **centered**, the data is symmetric.
- If the median is **closer to Q1 or Q3**, the data is **skewed**.

◆ Quartiles (Q1, Q3):

- **Q1 (25th percentile):** 25% of the data falls below this value.
- **Q3 (75th percentile):** 75% of the data falls below this value.
- Together they define the **IQR (spread of middle 50%)**.

◆ Whiskers:

- Extend to **minimum and maximum** values **within 1.5 × IQR** from Q1 and Q3.
- Help identify the **spread and presence of outliers**.

◆ Outliers:

- Values **outside whiskers** (beyond 1.5 × IQR) are considered outliers.

✅ 4. What Does Interquartile Range (IQR) Represent? *(1 Mark)*

Definition:

$$IQR = Q3 - Q1$$

- It measures the **spread of the middle 50% of the data**.
- **Resistant to outliers**, making it a reliable measure of variability.

Use of IQR in Box Plot:

- Helps **define the box boundaries**.
- Helps **detect outliers**:
  - **Outlier threshold** = Q1 - 1.5×IQR (lower), Q3 + 1.5×IQR (upper)

✅ 5. Summary Table of Box Plot Elements:

Element	Meaning	Interpretation
Min	Smallest value (non-outlier)	Start of lower whisker
Q1	25th percentile	Start of box
Median	50th percentile	Line inside box (central tendency)
Q3	75th percentile	End of box
Max	Largest value (non-outlier)	End of upper whisker
IQR	Q3 – Q1	Spread of central 50%
Outliers	Values beyond whiskers	Possible anomalies or special cases

Q4. Explain the role of Apache Pig in data processing workflows on Hadoop? What is Apache Spark, and how does it complement Hadoop for big data processing? [9]

=>

✅ 1. Role of Apache Pig in Hadoop Data Processing Workflows *(4 Marks)*

◆ What is Apache Pig?

Apache Pig is a **high-level platform** for creating **MapReduce programs** used with Hadoop. It uses a **scripting language called Pig Latin**, which simplifies complex data transformations.

Pig is especially useful for:

- **Extracting**
- **Transforming**
- **Loading (ETL)** data at scale

◆ Features of Apache Pig:

Feature	Description
High-level abstraction	Pig Latin abstracts away low-level MapReduce programming
Extensible	Supports custom user-defined functions (UDFs) in Java, Python, etc.
Handles complex data	Efficiently processes nested data structures (bags, tuples, maps)
Schema-less	Suitable for both structured and semi-structured data

---

◆ Pig Dataflow Example (ETL Pipeline):

Pig Latin Script:

```
pig
CopyEdit
raw_data = LOAD 'hdfs:/data/sales.csv' USING PigStorage(',') AS (id:int,
product:chararray, amount:float);
filtered_data = FILTER raw_data BY amount > 500;
grouped_data = GROUP filtered_data BY product;
result = FOREACH grouped_data GENERATE group, COUNT(filtered_data);
DUMP result;
```

This script:

- Loads data from HDFS
- Filters high sales
- Groups by product
- Counts records per group

◆ Advantages of Pig:

- Reduces the complexity of writing native MapReduce code
- Handles **large-scale, batch processing** efficiently
- Integrates seamlessly with **HDFS and MapReduce**

✓ 2. What is Apache Spark? *(2.5 Marks)*

◆ Definition:

Apache Spark is an **open-source, fast, in-memory big data processing engine**. It supports **distributed data processing** across clusters and can run independently or on top of Hadoop using **HDFS for storage**.

◆ Core Features of Spark:

Feature	Description
In-memory computing	Speeds up processing by reducing disk I/O
Rich APIs	Available in Java, Scala, Python, and R
Multiple workloads	Supports batch, streaming, ML, graph processing
RDD (Resilient Distributed Dataset)	Immutable distributed collection of objects

---

✓ 3. How Apache Spark Complements Hadoop *(2.5 Marks)*

◆ Integration with Hadoop:

- Spark can **run on Hadoop YARN** and use **HDFS as its storage layer**.
- It can access data from **HDFS, Hive, HBase**, and even from **external databases**.

◆ Comparison with MapReduce:

Feature	MapReduce	Apache Spark
Processing model	Disk-based	In-memory
Speed	Slower	Up to 100x faster
Use cases	Batch jobs	Batch, Streaming, ML, Graphs
Code complexity	High (Java required)	Simple with rich APIs

◆ Complementary Nature:

- **Hadoop** provides **robust storage (HDFS)** and **resource management (YARN)**.
- **Spark** adds **fast, flexible, in-memory computing** for a wide range of workloads.
- Together, they enable a **scalable and efficient big data processing pipeline**.

✓ Conclusion:

- **Apache Pig** simplifies large-scale data transformation using Pig Latin, making **MapReduce development faster and easier**.
- **Apache Spark** enhances the Hadoop ecosystem with **high-performance, in-memory processing**, making it suitable for **real-time analytics, ML, and graph processing**.
- Both tools are essential in building robust **big data pipelines** on the Hadoop platform.

Q5. List the few data visualization tools and discuss any four applications of data visualization along with the use of the various plots with Python/R or suitable tool. [9]

=>

Q5. List a few data visualization tools and discuss any four applications of data visualization along with the use of various plots with Python/R or suitable tool. [9 Marks]

✓ 1. Common Data Visualization Tools *(1.5 Marks)*

Data visualization tools are software or libraries that help in creating **graphical representations** of data for **better interpretation, communication, and analysis**.

Tool	Description
Matplotlib	A basic 2D plotting library in Python used to generate static charts
Seaborn	Built on Matplotlib; used for statistical and attractive visualizations
Plotly	An interactive plotting library for Python and web-based dashboards
ggplot2	A powerful visualization package in R based on the grammar of graphics
Tableau	A commercial drag-and-drop tool used for interactive dashboard creation
Power BI	Microsoft's business analytics tool for visualizing real-time business data

---

✅ 2. Applications of Data Visualization (with Python/R Tools) *(6 Marks)*

◆ A. Business Analytics – Sales Monitoring

Objective: Analyze regional sales performance and identify top-selling areas.

Tool & Plot Used:

- **Python (Matplotlib/Seaborn)**
- **Bar Chart** – Used to compare numerical values across categories.

```
Python
CopyEdit

import matplotlib.pyplot as plt
regions = ['North', 'South', 'East', 'West']
sales = [12000, 18000, 15000, 13000]

plt.bar(regions, sales, color='skyblue')
plt.title('Sales by Region')
plt.xlabel('Region')
plt.ylabel('Sales (in Rs)')
plt.show()
```

◆ B. Healthcare – Patient Vital Signs Monitoring

Objective: Detect anomalies in patients' heart rate data.

Tool & Plot Used:

- **Python (Seaborn)**
- **Box Plot** – Used to visualize spread and detect outliers in patient vitals.

```
python
CopyEdit

import seaborn as sns
heart_rates = [70, 72, 68, 75, 95, 100, 65]
sns.boxplot(data=heart_rates)
plt.title('Heart Rate Distribution')
plt.show()
```

◆ C. Financial Markets – Stock Price Trends

Objective: Track stock prices over time to identify patterns and trends.

Tool & Plot Used:

- **Python (Matplotlib)**
- **Line Chart** – Used to display data trends over time.

```
python
CopyEdit

import pandas as pd
import matplotlib.pyplot as plt

data = {'date': ['Day1', 'Day2', 'Day3', 'Day4'], 'price': [150, 160, 158,
165]}
df = pd.DataFrame(data)

plt.plot(df['date'], df['price'], marker='o')
plt.title('Stock Price Over Days')
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

◆ D. Data Science – Feature Relationship Analysis

Objective: Explore relationships between numerical features before modeling.

Tool & Plot Used:

- **Python (Seaborn)**
- **Scatter Plot** – Shows correlation between two variables.

```
python
CopyEdit

import seaborn as sns
import pandas as pd

df = sns.load_dataset('iris')
sns.scatterplot(x='sepal_length', y='petal_length', data=df, hue='species')
plt.title('Sepal vs Petal Length by Species')
plt.show()
```

✓ 3. Conclusion *(1.5 Marks)*

Data visualization is a **critical part of data analysis and communication**. It enables:

- Quick insights into data patterns
- Identification of trends, outliers, and relationships
- Clear storytelling for decision-makers

Tools like **Matplotlib, Seaborn, ggplot2, Tableau, and Power BI** are widely used for different types of visualizations, depending on the domain and purpose.

Q6. List the challenges of Data Visualization. Explain the types of visualization with example. [9]

=>

✓ 1. Challenges of Data Visualization *(3 Marks)*

Effective data visualization requires more than just plotting charts—it must communicate insights clearly and accurately. Below are key challenges:

Challenge	Explanation
1. Data Quality Issues	Incomplete, inconsistent, or noisy data can mislead visualization outcomes.
2. Choosing the Right Chart	Selecting an incorrect chart type may confuse users or hide important patterns.
3. Overplotting	Too many overlapping data points can clutter the plot and reduce readability.
4. High Dimensionality	Visualizing data with many variables requires complex plots or dimensionality reduction.
5. Misinterpretation	Non-technical users may misunderstand complex visuals or color usage.
6. Scalability	Real-time or large-scale datasets may lead to slow or inefficient visualization tools.

✓ 2. Types of Data Visualization with Examples *(6 Marks)*

There are several visualization types, each suited for different types of data and analysis goals:

◆ 1. Line Chart

Purpose: Shows trends or changes over continuous time intervals.
Use Case: Sales growth over months.

Example (Python):

```
python
CopyEdit
```

```
import matplotlib.pyplot as plt
months = ['Jan', 'Feb', 'Mar']
sales = [100, 150, 200]
plt.plot(months, sales, marker='o')
plt.title("Monthly Sales Trend")
plt.xlabel("Month")
plt.ylabel("Sales")
plt.show()
```

◆ 2. Bar Chart

Purpose: Compares categorical data.
Use Case: Product-wise revenue comparison.

Example:

```
python
CopyEdit

products = ['A', 'B', 'C']
revenue = [12000, 18000, 15000]
plt.bar(products, revenue, color='skyblue')
plt.title("Revenue by Product")
plt.show()
```

◆ 3. Pie Chart

Purpose: Displays part-to-whole relationships.
Use Case: Market share by company.

Example:

```
python
CopyEdit

labels = ['Company A', 'Company B', 'Company C']
sizes = [40, 35, 25]
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title("Market Share Distribution")
plt.show()
```

◆ 4. Box Plot

Purpose: Visualizes the distribution, spread, and outliers.
Use Case: Analyzing exam scores or heart rates.

Example:

```
python
CopyEdit

import seaborn as sns
sns.boxplot(data=[70, 72, 75, 90, 60, 100])
plt.title("Heart Rate Box Plot")
plt.show()
```

◆ 5. Heatmap

Purpose: Shows the relationship between two categorical/numerical variables using color intensities.
Use Case: Correlation matrix for feature selection in machine learning.

Example:

```
Python
CopyEdit

import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

data = np.random.rand(5,5)
sns.heatmap(data, annot=True, cmap='coolwarm')
plt.title("Heatmap Example")
plt.show()
```

✓ 3. Conclusion

Data visualization plays a crucial role in simplifying complex datasets into understandable visuals. However, challenges such as data quality, appropriate chart selection, and overplotting must be handled carefully.

Using the right type of visualization—such as **line charts** for trends, **bar charts** for comparisons, **pie charts** for proportions, **box plots** for distribution, and **heatmaps** for relationships—ensures clarity and accurate decision-making.

Q7. Explain in detail the Hadoop Ecosystem with suitable diagram along with the various components. [9]

=>

- Fig. 6.6.1 shows Apache Hadoop ecosystem.

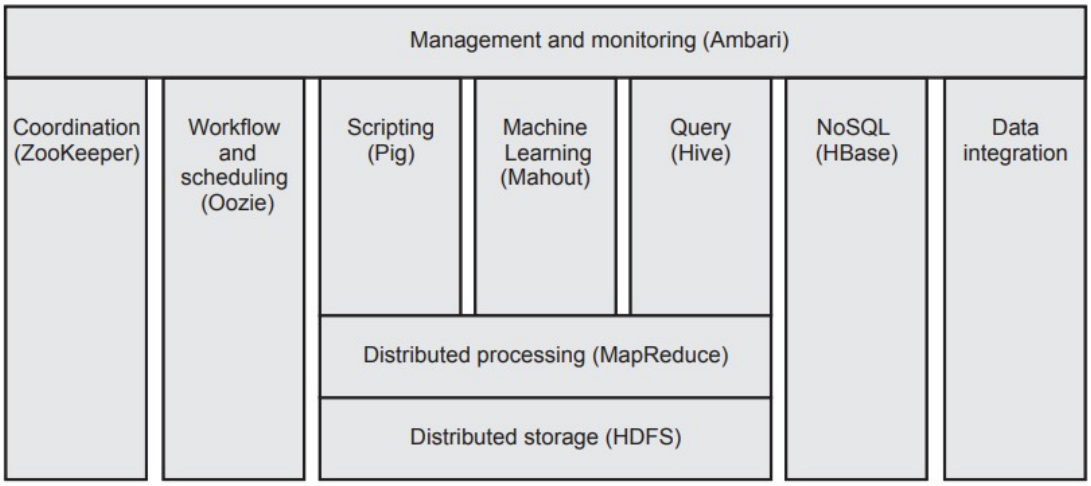


Fig. 6.6.1 : Apache Hadoop ecosystem

✓ 1. Introduction to Hadoop Ecosystem *(1.5 Marks)*

Apache Hadoop is an **open-source big data framework** developed by the **Apache Software Foundation** for **storing and processing large datasets** in a **distributed computing environment**.

The **Hadoop Ecosystem** consists of a **core framework** (HDFS, YARN, MapReduce) and various **supporting tools and libraries** for tasks like data ingestion, storage, processing, querying, and analysis.

✓ 2. Hadoop Ecosystem Diagram *(Labelled for clarity) (1.5 Marks)*

```
pgsql
CopyEdit
                +--------------------------+
                |      Data Ingestion      |
                |  (Flume, Sqoop, Kafka)   |
                +------------+-------------+
                             ↓
+----------------------+   +--------------------+
|   Data Storage (HDFS)| ← | Resource Mgmt (YARN)|
+----------------------+   +--------------------+
           ↓                          ↓
+----------------------+   +----------------------+
| Processing (MapReduce)| | Processing (Apache Spark) |
+----------------------+   +----------------------+
           ↓
+-----------------------------+
|    Querying Layer (Hive, Pig) |
+-----------------------------+
           ↓
+-----------------------------+
| Workflow/Scheduling (Oozie)  |
+-----------------------------+
```

```
           ↓
+--------------------------+
| Monitoring (Ambari)      |
+--------------------------+
```

✅ 3. Core Components of Hadoop Ecosystem *(4 Marks)*

◆ A. HDFS (Hadoop Distributed File System)

- **Storage layer** of Hadoop.
- Divides large files into **blocks** (default: 128MB or 256MB).
- Stores blocks across **multiple DataNodes** for fault tolerance.
- **NameNode** manages metadata, and **DataNodes** store actual data.

◆ B. YARN (Yet Another Resource Negotiator)

- **Resource management and job scheduling framework.**
- Divides workload among various applications running on a Hadoop cluster.
- Consists of **ResourceManager**, **NodeManager**, and **ApplicationMaster**.

◆ C. MapReduce

- Core **data processing engine** in Hadoop.
- Works on the **map → shuffle → reduce** paradigm.
- Efficient for **batch processing** on large-scale datasets.

✅ 4. Supporting Ecosystem Components *(2 Marks)*

◆ 1. Data Ingestion Tools

Tool	Function
Sqoop	Transfers structured data from RDBMS to Hadoop
Flume	Captures and moves log data into HDFS
Kafka	Streams real-time data into Hadoop

◆ 2. Data Processing Tools

Tool	Function
Apache Pig	High-level scripting (Pig Latin) over MapReduce
Apache Hive	SQL-like queries on HDFS data using HiveQL
Apache Spark	In-memory data processing; faster than MapReduce

---

◆ 3. Workflow & Coordination

Tool	Function
Oozie	Job scheduler for running workflows
Zookeeper	Coordination service for distributed systems

◆ 4. Monitoring & Management

Tool	Function
Ambari	Web-based interface for cluster monitoring and management

---

✓ 5. Advantages of Hadoop Ecosystem *(Optional Summary)*

- **Scalable:** Handles petabytes of data across commodity hardware
- **Fault-tolerant:** Data replication prevents data loss
- **Cost-effective:** Open-source and deployable on inexpensive hardware
- **Flexible:** Supports multiple data types (structured, semi-structured, unstructured)

✓ 6. Conclusion *(1 Mark)*

The **Hadoop Ecosystem** provides a robust and flexible framework for **big data storage and processing**.
With **HDFS** for distributed storage, **MapReduce/Spark** for parallel computation, and a range of tools like **Hive, Pig, Flume, Oozie**, it enables organizations to manage and analyze **large-scale data efficiently**.

Q8. Write a short note on the following. [9] a) Map Reduce b) Pig

=>

✓ (a) MapReduce

◆ Definition:

MapReduce is a **programming model** and processing engine in the Hadoop ecosystem that allows for **parallel and distributed processing** of large datasets across a cluster of computers.

◆ Key Phases:

1. **Map Phase:**
   - Input data is divided into **splits** and processed by **mappers**.
   - Each mapper emits key-value pairs:
     Example: Input → `"word"` → Output → (`"word"`, `1`)
2. **Shuffle and Sort:**
   - Intermediate key-value pairs are grouped by key and sent to reducers.
3. **Reduce Phase:**
   - Reducers process grouped key-value pairs to produce the final output.
   - Example: (`"word"`, [1, 1, 1]) → (`"word"`, `3`)

◆ Example: Word Count

Input: `"Hadoop is scalable and Hadoop is open-source"`
Output:

```
arduino
CopyEdit
("Hadoop", 2), ("is", 2), ("scalable", 1), ("and", 1), ("open-source", 1)
```

◆ Advantages:

- **Fault-tolerant**
- **Highly scalable**
- **Supports massive parallelism**

◆ Limitations:

- **Disk-based** → Slower than in-memory models like Spark
- Less suitable for **real-time or iterative** applications

- Fig. 6.6.5 shows MapReduce logical data flow.

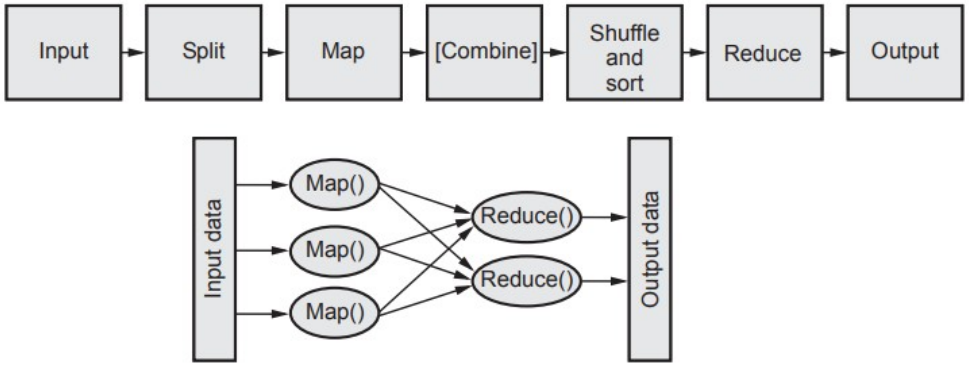


Fig. 6.6.5 Map-Reduce logical data flow

1. **Input :** This is the input data / file to be processed.
2. **Split :** Hadoop splits the incoming data into smaller pieces called "splits".
3. **Map :** In this step, MapReduce processes each split according to the logic defined in map() function. Each mapper works on each split at a time. Each mapper is treated as a task and multiple tasks are executed across different TaskTrackers and coordinated by the JobTracker.
4. **Combine :** This is an optional step and is used to improve the performance by reducing the amount of data transferred across the network. Combiner is the same as the reduce step and is used for aggregating the output of the map() function before it is passed to the subsequent steps.
5. **Shuffle and Sort :** In this step, outputs from all the mappers are shuffled, sorted to put them in order and grouped before sending them to the next step.
6. **Reduce :** This step is used to aggregate the outputs of mappers using the reduce() function. Output of reducer is sent to the next and final step. Each reducer is treated as a task and multiple tasks are executed across different TaskTrackers and coordinated by the JobTracker.
7. **Output :** Finally the output of reduce step is written to a file in HDFS.

✓ (b) Apache Pig

◆ Definition:

Apache Pig is a **high-level data flow scripting platform** built on top of Hadoop.
It uses a scripting language called **Pig Latin** to simplify the writing of complex MapReduce jobs.

◆ Key Features:

Feature	Description
Ease of use	Pig Latin is simpler than writing raw MapReduce code
Extensible	Supports **User Defined Functions (UDFs)** in Java, Python
Schema-less	Works well with structured, semi-structured data
Optimized	Automatically optimizes execution plans

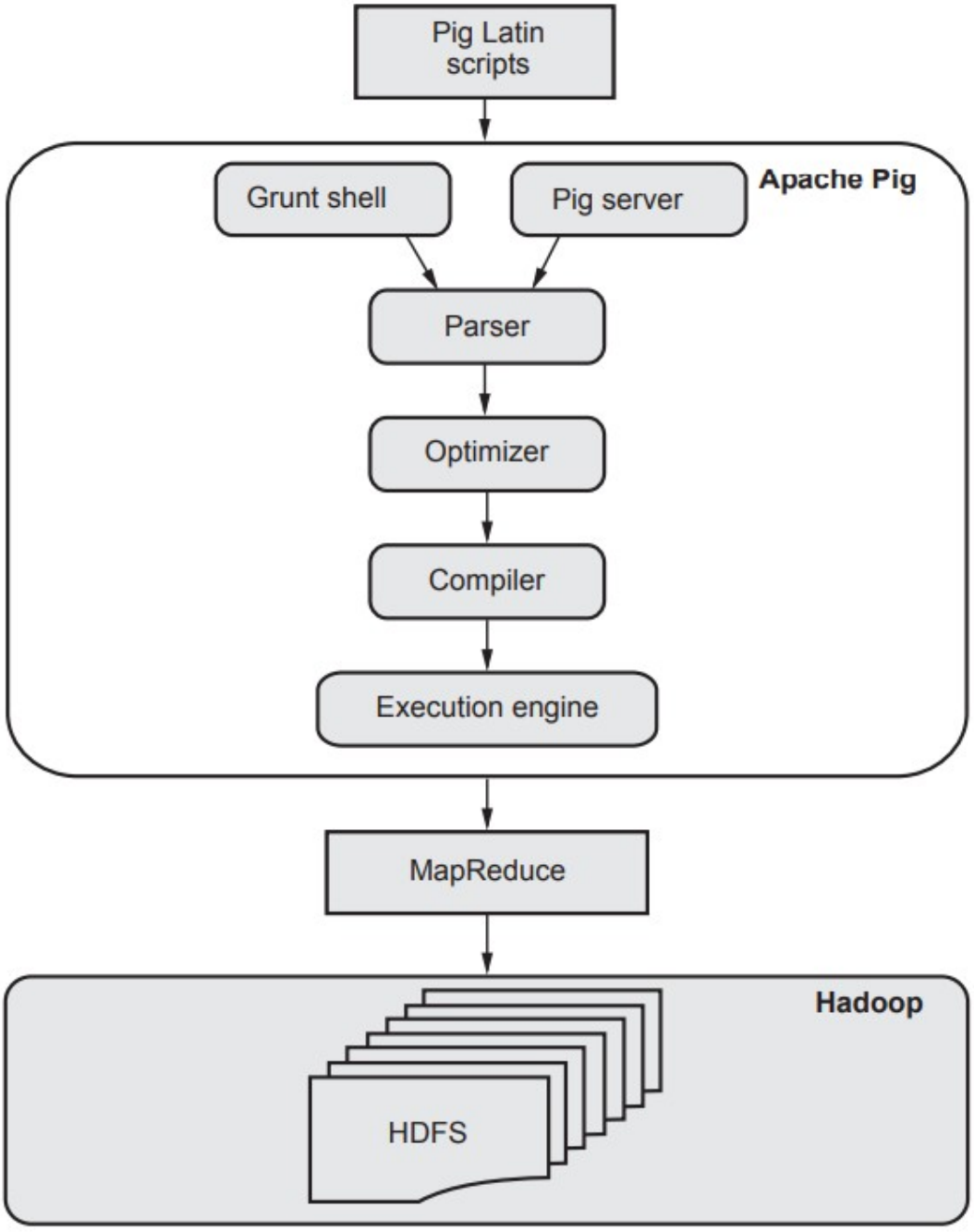


Fig. 6.6.7 Pig architecture

◆ Typical Data Flow in Pig:

1. **Load** data from HDFS
2. **Transform** using filters, joins, group-by
3. **Store** results back to HDFS or other systems

◆ Pig Latin Script Example:

```
pig
CopyEdit
raw_data = LOAD 'sales.csv' USING PigStorage(',') AS (id:int,
amount:float);
high_sales = FILTER raw_data BY amount > 1000;
grouped = GROUP high_sales BY id;
result = FOREACH grouped GENERATE group, COUNT(high_sales);
DUMP result;
```

This script:

- Loads data
- Filters by amount
- Groups by `id`
- Outputs count per group

◆ Use Cases:

- ETL (Extract, Transform, Load) pipelines
- Log analysis
- Preprocessing for analytics and machine learning

◆ Advantages:

- **Simplifies** big data processing
- **Integrates** easily with HDFS and MapReduce
- Reduces development time for data engineers

Q9. With a suitable example, draw a Histogram, boxplot and explain its usages. [9]

=>

✓ 1. Introduction to Histogram and Boxplot *(1 Mark)*

- A **Histogram** and a **Boxplot** are both **graphical tools** used to understand the **distribution and variability** of a dataset.
- While the histogram shows **frequency distribution**, the boxplot shows **statistical summaries** like median, quartiles, and outliers.

✅ 2. Histogram *(3 Marks)*

◆ Definition:

A **histogram** is a type of bar chart that shows the **distribution of numerical data** by dividing it into **bins (intervals)** and **counting the frequency** of data points in each bin.

◆ Usage:

- Understand the **shape** of the distribution (normal, skewed, bimodal, etc.)
- Detect **frequency peaks**, **gaps**, and **spread**
- Used in **exploratory data analysis** (EDA)

◆ Example Dataset:

Marks obtained by 15 students:

```
[45, 55, 58, 60, 65, 68, 70, 72, 75, 78, 80, 82, 85, 90, 92]
```

◆ Histogram Plot in Python:

```
python
CopyEdit
import matplotlib.pyplot as plt
marks = [45, 55, 58, 60, 65, 68, 70, 72, 75, 78, 80, 82, 85, 90, 92]
plt.hist(marks, bins=5, color='skyblue', edgecolor='black')
plt.title('Histogram of Student Marks')
plt.xlabel('Marks')
plt.ylabel('Frequency')
plt.show()
```

◆ Visual Representation (Rough Sketch):

```
lua
CopyEdit
Frequency
   ▲
  4|
  3|
  2|
  1|
   +--------------------------▶ Marks
      40-55 55-70 70-85 85-100
```

✅ 3. Boxplot *(3 Marks)*

◆ Definition:

A **boxplot** (box-and-whisker plot) is a **summary visualization** that shows the **five-number summary**:

- **Minimum**
- **First Quartile (Q1)**
- **Median (Q2)**
- **Third Quartile (Q3)**
- **Maximum**

◆ Usage:

- Detect **spread**, **central tendency**, and **outliers**
- Compare **distributions across groups**
- Used in **data preprocessing**, **EDA**, and **statistical analysis**

◆ Boxplot Code in Python:

```
python
CopyEdit

import seaborn as sns
sns.boxplot(data=marks)
plt.title('Boxplot of Student Marks')
plt.show()
```

◆ Visual Sketch of Boxplot:

```
mathematica
CopyEdit
Marks
|
|     ┌──────┬──────────┬──────┐
|     |      |          |      |
| Min    Q1      Median      Q3    Max
```

◆ From the Data:

- Min = 45
- Q1 ≈ 65
- Median (Q2) ≈ 75
- Q3 ≈ 82
- Max = 92

✓ 4. Comparison & Use Cases *(2 Marks)*

Feature	Histogram	Boxplot
Shows	Frequency distribution	Summary statistics
Detects	Distribution shape	Spread, outliers, quartiles
Use Case	See common value ranges	Spot skewness and variability
Visual Output	Bar-like bins	Box with whiskers and median line

Q10. Describe the data visualization tool Tableau. List of data visualization tools. [9]

=>

✓ 1. Introduction to Tableau *(1 Mark)*

Tableau is a **powerful, interactive data visualization and business intelligence tool** that allows users to connect to various data sources and create **interactive dashboards, reports, and charts** without requiring programming knowledge.

It helps users to analyze, visualize, and share data insights in an easily understandable visual format.

✓ 2. Features of Tableau *(3 Marks)*

Feature	Description
Drag-and-drop Interface	Allows users to build visualizations easily using GUI
Multiple Data Connectors	Supports Excel, SQL, Oracle, Google Sheets, cloud data, etc.
Live & In-Memory Analytics	Provides both real-time data access and in-memory data extraction
Interactive Dashboards	Combine multiple visualizations into one interface for exploration
Storytelling	Enables creation of data stories with sequential insights

Feature	Description
Mobile Support	Dashboards are mobile-friendly and can be accessed on any device
Integration	Can be integrated with R, Python, Hadoop, and cloud platforms like AWS, Azure

---

✓ 3. Common Visualizations in Tableau *(1.5 Marks)*

Visualization Type	Use Case Example
Bar Chart	Compare sales by region
Line Chart	Show stock price trend over time
Pie Chart	Show product market share
Scatter Plot	Analyze correlation between two variables
Map	Display geographic distribution of data
Heat Map	Show data density or correlation matrix

✓ 4. Applications of Tableau *(1.5 Marks)*

- **Business Intelligence & Reporting**
- **Sales & Marketing Analysis**
- **Financial Forecasting & KPIs**
- **Customer and Web Analytics**
- **Healthcare Data Monitoring**
- **Big Data Visualization (via Hadoop or Spark)**

✓ 5. List of Popular Data Visualization Tools *(2 Marks)*

Here is a list of other widely used data visualization tools apart from Tableau:

Tool	Description
Power BI	Microsoft tool for real-time business analytics and dashboarding

Tool	Description
QlikView/Qlik Sense	BI tools for self-service data visualization
Google Data Studio	Free Google tool to create dynamic dashboards
D3.js	JavaScript library for highly customized web visualizations
Matplotlib	Python library for static 2D charts
Seaborn	Statistical data visualization library built on top of Matplotlib
Plotly	Python/JS library for interactive plots
ggplot2	R-based data visualization library built on the grammar of graphics

---

✅ 6. Conclusion

Tableau is one of the most popular and user-friendly data visualization tools that helps turn **raw data into insightful visual stories** through dashboards, charts, and reports.
Along with other tools like **Power BI, QlikView, and Plotly**, it supports informed **decision-making through visual analytics**.

Q11. What is Data Visualization? Describe the challenges of data visualization. [9]

=>

✅ 1. What is Data Visualization? *(3 Marks)*

Data Visualization is the process of representing data and information using **visual elements** like **charts, graphs, maps, and dashboards** to make data easier to **understand, analyze, and communicate**.

It enables:

- Quick identification of trends, patterns, and outliers
- Better **data-driven decision-making**
- Clear communication of **complex datasets**

✅ Importance of Data Visualization:

Purpose	Example
Identify trends	Line charts showing sales over time
Compare categories	Bar charts comparing product performance
Show relationships between variables	Scatter plots in machine learning
Detect outliers or anomalies	Boxplots in patient data or transactions

---

✅ Examples of Visualization Types:

- **Bar Chart**: Compare quantities
- **Line Chart**: Show trends over time
- **Pie Chart**: Show proportions
- **Boxplot**: Show distribution and outliers
- **Heatmap**: Show correlation and density

✅ 2. Challenges of Data Visualization *(6 Marks)*

Despite its benefits, data visualization comes with several challenges:

◆ 1. Poor Data Quality

- Incomplete, inconsistent, or noisy data can lead to **misleading visuals**.
- Example: Missing values may distort trend lines or averages.

◆ 2. Wrong Choice of Visualization

- Using an **inappropriate chart** can confuse the audience.
- Example: Using a pie chart for time-series data is ineffective.

◆ 3. Overplotting / Clutter

- Too many data points or overlapping visuals **reduce readability**.
- Solution: Use aggregation, transparency, or interactive filters.

◆ 4. Misleading Scales or Axes

- Improper axis scales (e.g., not starting from zero) can **distort the message**.
- Example: Exaggerated line slope due to zoomed-in y-axis.

◆ 5. High Dimensionality

- Visualizing data with **many variables** is difficult in 2D or 3D space.
- Solution: Use PCA, parallel coordinates, or dimensionality reduction techniques.

◆ 6. Scalability with Large Data

- Rendering visualizations for **millions of rows** can be **slow or unresponsive**.
- Solution: Use **sampling**, pre-aggregated data, or **big data visualization tools** like Tableau or Power BI.

◆ 7. User Misinterpretation

- Users with **limited technical knowledge** may misread visuals.
- Example: Misinterpreting correlation as causation in a scatter plot.

✓ 3. Conclusion

Data visualization plays a key role in converting raw data into meaningful insights. However, challenges such as **data quality, overplotting, poor design choices**, and **scalability issues** must be addressed to create **accurate and effective visualizations**.

Q12. Explain architecture of Apache-Pig. [9]

=>

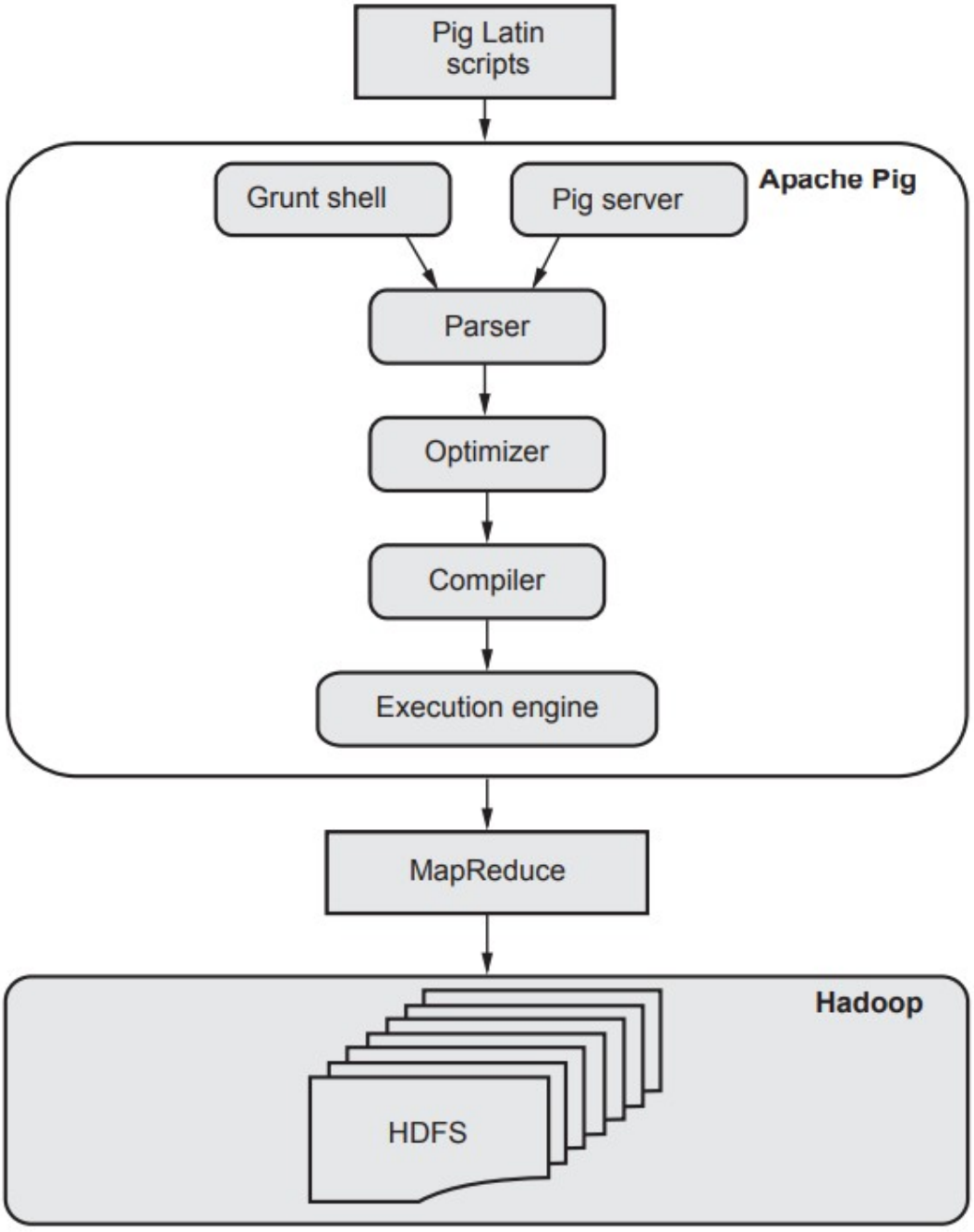


Fig. 6.6.7 Pig architecture

✓ 1. Introduction to Apache Pig *(1.5 Marks)*

Apache Pig is a **high-level platform for data analysis** in the Hadoop ecosystem. It was developed by **Yahoo** and is used to **analyze large data sets**. Pig provides a scripting

language called **Pig Latin**, which simplifies the development of complex MapReduce programs.

Pig translates the high-level Pig Latin scripts into **MapReduce jobs**, which are then executed on a **Hadoop cluster**, allowing data analysts and developers to focus on **data logic rather than low-level programming**.

✓ 2. Detailed Apache Pig Architecture (with Diagram Fig. 6.6.7) *(6 Marks)*

Below is the detailed explanation of each component shown in the architecture:

◆ Step 1: Pig Latin Scripts (User Input Layer)

- Users write data transformation logic in **Pig Latin**.
- Example operations: `LOAD`, `FILTER`, `JOIN`, `GROUP`, `FOREACH`, `STORE`.
- The script defines the **data flow** and is submitted via **Grunt shell** or **Pig server**.

◆ Step 2: Grunt Shell / Pig Server

- **Grunt Shell**:
  - Interactive command-line shell for executing Pig Latin commands line-by-line.
  - Used for **quick prototyping and testing**.
- **Pig Server**:
  - Enables Pig to be **embedded into Java programs**.
  - Used for **production environments** where Pig is integrated with larger data workflows.

◆ Step 3: Parser

- The script is parsed to check for **syntax errors** and **semantic validity**.
- It generates a **logical plan**, a tree of logical operators representing the script.
- Example logical operators: `Load`, `Filter`, `Group`, `Join`, `Foreach`.

◆ Step 4: Logical Plan Optimizer

- The logical plan is optimized using rules such as:
  - **Projection pruning** (removing unused columns),
  - **Predicate pushdown** (applying filters earlier),
  - **Combining adjacent operations** to reduce computation.

- It ensures the plan is **efficient without changing its output**.

◆ Step 5: Compiler

- Converts the optimized logical plan into a **physical plan**.
- Then it translates it into one or more **MapReduce jobs** (or Tez/Spark jobs in newer Pig versions).
- Ensures the jobs are compatible with **Hadoop YARN**.

◆ Step 6: Execution Engine

- Submits the physical plan (MapReduce jobs) to the Hadoop framework.
- Coordinates with **YARN Resource Manager** for job scheduling and monitoring.
- Handles **job tracking, retries, and status reporting**.

◆ Step 7: MapReduce Layer

- The compiled Pig logic runs as **MapReduce jobs** in the Hadoop ecosystem.
- These jobs are executed across a **distributed cluster**, enabling large-scale data processing.

◆ Step 8: HDFS (Hadoop Distributed File System)

- Acts as the **input/output layer** for Pig.
- Pig loads data from HDFS, performs operations, and stores results back into HDFS.
- HDFS provides **scalable, fault-tolerant storage** for big data files.

✓ 3. Example Flow of Pig Script Execution

Suppose we have the following Pig script:

```
pig
CopyEdit
data = LOAD 'hdfs:/sales.csv' USING PigStorage(',') AS (id:int,
amount:float);
high_sales = FILTER data BY amount > 1000;
grouped = GROUP high_sales BY id;
result = FOREACH grouped GENERATE group, COUNT(high_sales);
DUMP result;
```

Execution Process:

1. **User submits** this script using Grunt shell.
2. **Parser** checks and builds logical plan.
3. **Optimizer** enhances performance.
4. **Compiler** converts it to MapReduce jobs.
5. **Execution Engine** submits jobs to Hadoop.
6. Results are **stored back in HDFS**.

✓ 4. Advantages of Apache Pig *(1.5 Marks)*

Advantage	Description
Simplicity	Pig Latin is easier than Java-based MapReduce programs
Flexibility	Works with both structured and semi-structured data
Extensibility	Allows user-defined functions (UDFs) in Java, Python, etc.
Optimization	Automatically optimizes execution plan
Integration	Easily integrates with HDFS, Hive, and other Hadoop tools
Support for Complex Data Types	Supports tuples, bags, and maps (unlike plain SQL)

✓ 5. Conclusion

Apache Pig's architecture is designed to **abstract the complexity of MapReduce** by providing a **simpler scripting interface (Pig Latin)**. Through its components—**Parser, Optimizer, Compiler, and Execution Engine**—Pig automates the conversion of high-level scripts into efficient **MapReduce jobs** that run on Hadoop and operate on **HDFS**.

Pig is best suited for **ETL processes, data transformation, and analysis** in large-scale data environments.

Q13. List the data visualization tools and discuss any four applications of data visualization along with the use of the suitable plot. [9]

=>

✓ 1. List of Data Visualization Tools *(2 Marks)*

Below are some widely used data visualization tools in both academic and industry settings:

Tool	Description
Tableau	Powerful, user-friendly drag-and-drop tool for building interactive dashboards.
Power BI	Microsoft's business intelligence tool for visual analytics and real-time data.
Matplotlib	A core Python library for creating static 2D plots and graphs.
Seaborn	Built on Matplotlib, it provides enhanced statistical plotting functions.
Plotly	Used for creating interactive, web-ready visualizations in Python or JS.
ggplot2 (R)	Based on the Grammar of Graphics for advanced statistical plots in R.
QlikView/Qlik Sense	Self-service BI tools with dashboarding and analytics features.
Google Data Studio	Free online tool for creating dynamic, shareable dashboards.

✓ 2. Applications of Data Visualization with Suitable Plot Types *(6 Marks)*

Here are four key real-world applications of data visualization with matching plots:

◆ A. Business Analytics – Sales Performance Monitoring

Use Case: Analyzing regional sales or monthly revenue trends.
Suitable Plot: Bar Chart or **Line Chart**

Chart Type	Use Case
Bar Chart	Compare sales by region
Line Chart	Visualize sales growth monthly

📌 Bar Chart Example (Python/Matplotlib)

```
python
CopyEdit
import matplotlib.pyplot as plt
regions = ['North', 'South', 'East', 'West']
sales = [12000, 18000, 15000, 13000]
plt.bar(regions, sales, color='skyblue')
plt.title('Sales by Region')
plt.show()
```

◆ B. Healthcare – Patient Vital Sign Analysis

Use Case: Analyze spread and detect abnormal values in heart rate or blood pressure readings.
Suitable Plot: Box Plot

📌 Box Plot Usage:

- Shows median, quartiles, and outliers
- Useful for identifying critical patient conditions

◆ C. Financial Analytics – Stock Market Trends

Use Case: Monitoring daily/weekly stock price changes.
Suitable Plot: Line Chart

📌 Line Chart Usage:

- Clearly shows trends over time
- Helps in forecasting and pattern recognition

◆ D. Data Science – Feature Relationships and Correlation

Use Case: Visualize the relationship between numerical features in datasets.
Suitable Plot: Scatter Plot or **Heatmap**

📌 Scatter Plot Example (Seaborn):

```
python
CopyEdit
import seaborn as sns
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()
```

```
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
sns.scatterplot(x='sepal length (cm)', y='petal length (cm)', data=df)
```

📌 **Heatmap Usage:**

- Ideal for showing correlation matrices in machine learning.

✅ 3. Conclusion *(1 Mark)*

Data visualization is crucial for transforming raw data into meaningful visuals that support **insight discovery**, **trend analysis**, and **decision-making**. Tools like **Tableau, Power BI, Matplotlib**, and **Seaborn** enable users to generate **effective, clear, and interactive visuals** for various applications in business, healthcare, finance, and data science.

Q14. List the challenges of data visualization explain the types of visualization with example. [9]

=>

✅ 1. What is Data Visualization?

Data visualization is the process of converting raw data into **graphical or pictorial formats** to make it easier to:

- Understand trends
- Spot outliers
- Communicate insights clearly

It helps in decision-making by allowing stakeholders to **interpret complex datasets visually**.

✅ 2. Challenges of Data Visualization *(3.5 Marks)*

Challenge	Explanation
1. Poor Data Quality	Missing, inconsistent, or noisy data misleads the visual result.
2. Wrong Chart Selection	Using pie charts for time series or bar charts for correlation is misleading.
3. Overplotting / Clutter	Too many data points cause crowding, making plots unreadable.

Challenge	Explanation
4. High Dimensionality	Difficult to visualize data with more than 3 or 4 dimensions.
5. Misleading Scales or Axes	Improper axes distort interpretation (e.g., not starting from zero).
6. Scalability	Visualization tools may crash or lag with very large datasets.
7. User Misinterpretation	Non-technical users may misread correlation, trends, or outliers.

✅ 3. Types of Visualization with Examples *(5.5 Marks)*

Here are five essential types of visualizations with **realistic examples** and **generated plots**:

◆ 1. Line Chart

- **Use Case**: Monthly Sales Trends
- **Purpose**: Visualize changes over time (continuous data)

✅ **Interpretation**: Ideal for stock prices, trends, KPIs.

◆ 2. Bar Chart

- **Use Case**: Product-wise sales comparison
- **Purpose**: Compare values across discrete categories

✅ **Interpretation**: Helps identify best/worst performers among groups.

◆ 3. Pie Chart

- **Use Case**: Market share distribution
- **Purpose**: Show part-to-whole relationships (percentages)

✅ **Interpretation**: Best used when you want to highlight proportions in a whole.

◆ 4. Box Plot

- **Use Case**: Student score distribution
- **Purpose**: Show median, quartiles, and outliers in data

✅ **Interpretation**: Detects data spread and identifies outliers or skew.

◆ 5. Heatmap

- **Use Case**: Correlation matrix in machine learning
- **Purpose**: Visualizes intensity or relationships via color

✅ **Interpretation**: Helps understand feature relationships or data density.

🔍 Visual Summary of All Chart Types:

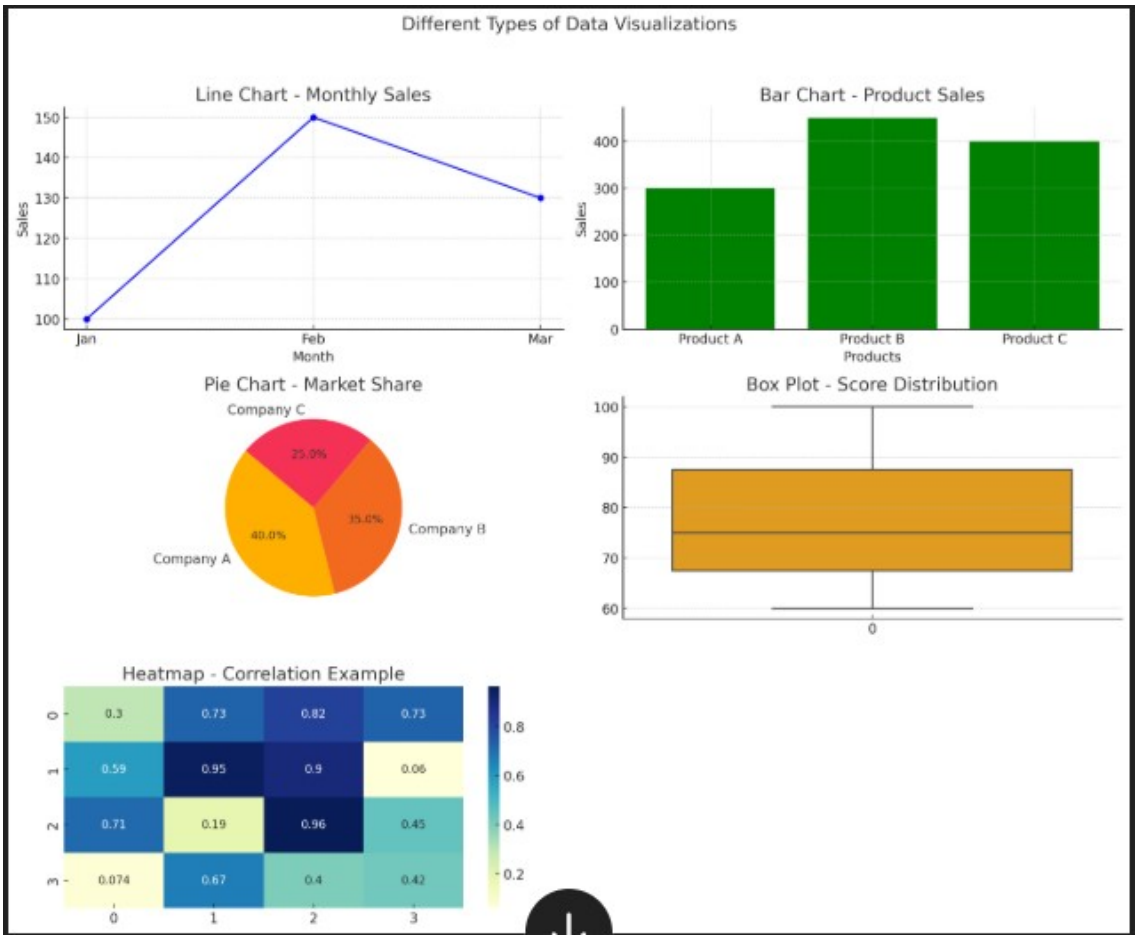
The image above shows:

Chart Type	Key Insights
Line Chart	Shows trend (e.g., rising/falling sales)
Bar Chart	Compares category-wise values
Pie Chart	Shows % composition of parts of a whole
Box Plot	Identifies spread, median, and outliers
Heatmap	Shows correlations using colors

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Setup for plotting
fig, axs = plt.subplots(3, 2, figsize=(14, 12))
```

```
fig.suptitle('Different Types of Data Visualizations', fontsize=16)

# 1. Line Chart - Monthly Sales Trend
months = ['Jan', 'Feb', 'Mar']
sales = [100, 150, 130]
axs[0, 0].plot(months, sales, marker='o', color='blue')
axs[0, 0].set_title('Line Chart - Monthly Sales')
axs[0, 0].set_xlabel('Month')
axs[0, 0].set_ylabel('Sales')

# 2. Bar Chart - Product Sales Comparison
products = ['Product A', 'Product B', 'Product C']
product_sales = [300, 450, 400]
axs[0, 1].bar(products, product_sales, color='green')
axs[0, 1].set_title('Bar Chart - Product Sales')
axs[0, 1].set_xlabel('Products')
axs[0, 1].set_ylabel('Sales')

# 3. Pie Chart - Market Share
labels = ['Company A', 'Company B', 'Company C']
sizes = [40, 35, 25]
axs[1, 0].pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
axs[1, 0].set_title('Pie Chart - Market Share')

# 4. Box Plot - Student Scores
scores = [60, 70, 75, 90, 100, 85, 65]
sns.boxplot(data=scores, ax=axs[1, 1], color='orange')
axs[1, 1].set_title('Box Plot - Score Distribution')
```

```
# 5. Heatmap - Random Correlation Matrix
data = np.random.rand(4, 4)
sns.heatmap(data, annot=True, cmap='YlGnBu', ax=axs[2, 0])
axs[2, 0].set_title('Heatmap - Correlation Example')

# Hide the empty subplot
axs[2, 1].axis('off')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Q16. Write a short note on the following [9] i) Map reduce. ii) Pig iii) Hive

=>

✓ i) MapReduce (3 Marks)

◆ Definition:

MapReduce is a **programming model and processing framework** used for processing large datasets in a **distributed** and **parallel** manner across a Hadoop cluster. It was introduced by Google and later implemented by Apache Hadoop.

◆ Phases of MapReduce:

1. **Map Phase:**
   - Processes input data and converts it into key-value pairs.
   - Example: Reading text and emitting (`word`, `1`) for each word.
2. **Shuffle and Sort Phase:**
   - Groups and sorts intermediate keys generated by the map tasks.
   - Ensures all values for the same key are sent to the same reducer.
3. **Reduce Phase:**
   - Aggregates the grouped data and produces final output.
   - Example: Summing all `1`s for each word to get total word count.

◆ Example: Word Count

Input:

```
text
CopyEdit
Hadoop is open source. Hadoop is fast.
```

Mapper Output:
`("Hadoop", 1), ("is", 1), ("open", 1), ("source", 1), ("Hadoop", 1), ("is", 1), ("fast", 1)`

Reducer Output:
`("Hadoop", 2), ("is", 2), ("open", 1), ("source", 1), ("fast", 1)`

◆ Advantages:

- Handles **massive datasets** efficiently.
- Offers **fault tolerance** (via data replication).
- Automatically manages **data partitioning and job scheduling**.

◆ Limitations:

- Not suitable for **real-time processing**.
- Requires writing low-level **Java code** (before tools like Pig and Hive).

✅ ii) Apache Pig (3 Marks)

◆ Definition:

Apache Pig is a **high-level platform** for creating MapReduce programs used with Hadoop. It simplifies big data analysis by using **Pig Latin**, a data flow scripting language.

◆ Pig Architecture Components:

- **Pig Latin Scripts** – Written by users for data operations.
- **Grunt Shell** – Interactive shell for running commands.
- **Parser → Optimizer → Compiler → Execution Engine**
- Translates scripts into **MapReduce jobs** behind the scenes.

◆ Features of Pig:

- Reduces the complexity of **writing MapReduce code**.
- Supports **complex data types** like tuples, bags, and maps.
- Allows **User Defined Functions (UDFs)** in Java, Python, Ruby.
- Supports **both batch and interactive processing**.
- Suitable for **ETL pipelines and data preprocessing**.

◆ Example Pig Latin Script:

```
pig
CopyEdit
raw = LOAD 'transactions.csv' USING PigStorage(',') AS (id:int,
amount:float);
filtered = FILTER raw BY amount > 1000;
grouped = GROUP filtered BY id;
results = FOREACH grouped GENERATE group, COUNT(filtered);
DUMP results;
```

◆ Use Cases:

- Processing logs (e.g., website clickstream).
- Data cleaning and transformation before storage or analytics.
- Intermediate step in machine learning pipelines.

✅ iii) Apache Hive (3 Marks)

◆ Definition:

Apache Hive is a **data warehouse system** built on top of Hadoop. It allows querying of large datasets using a SQL-like language called **HiveQL**.

Hive is designed for **analysts familiar with SQL**, making it easier to run queries on data stored in **HDFS**.

◆ Hive Architecture Components:

- **HiveQL Interface** – Accepts SQL-like queries.
- **Driver** – Manages query execution and sessions.
- **Compiler** – Converts queries into MapReduce or Tez jobs.
- **Metastore** – Stores metadata about databases, tables, schemas.
- **Execution Engine** – Submits jobs to Hadoop (via YARN).

◆ Key Features:

- Converts high-level SQL queries to **MapReduce jobs**.
- Supports **partitioning and bucketing** for performance.
- Integrates with **HDFS, HBase**, and other data sources.
- Works in batch mode (not real-time).
- Supports **UDFs and custom queries**.

◆ Example Hive Query:

```
sql
CopyEdit
SELECT department, AVG(salary)
FROM employees
WHERE salary > 50000
GROUP BY department;
```

◆ Use Cases:

- Generating business reports and summaries.
- Running analytical queries on structured big data.
- Creating dashboards and KPI monitoring systems.

✓ Conclusion:

Tool	Purpose	Language	Best For
MapReduce	Core data processing engine	Java	Custom batch processing logic
Pig	High-level scripting over Hadoop	Pig Latin	ETL, preprocessing, fast data pipelines
Hive	SQL-based querying over Hadoop	HiveQL (SQL)	Analysts, reporting, data summarization

Together, these tools **simplify big data analytics** by offering **flexible, scalable, and efficient solutions** over the Hadoop platform.

Q17. With a suitable example explain Histogram and explain its usages. [8]

=>

✓ 1. Definition of Histogram *(2 Marks)*

A **Histogram** is a **graphical representation** of the **distribution of numerical data**. It uses **rectangular bars** to show the frequency of data points that fall within specified **continuous intervals or bins**.

- Unlike a **bar chart**, which is used for **categorical data**, a histogram is specifically used for **quantitative data**.
- The **width** of each bar represents the interval (range) and the **height** represents the frequency (count) of data values in that interval.

✓ 2. Key Characteristics of a Histogram

- **Data must be numerical and continuous**
- No spaces between bars (since intervals are continuous)
- Each bar covers a **range of values (bin)**, not just a single value
- Histograms are used to **estimate the probability distribution** of a continuous variable

---

✓ 3. Components of a Histogram *(1 Mark)*

Component	Explanation
Bins	Ranges into which data is divided (e.g., 0–10, 10–20)
X-axis	Represents bins (intervals of the variable)
Y-axis	Represents frequency or count of data points in each bin
Bars	Rectangles whose height shows how many values fall in each bin

---

✓ 4. Suitable Example of a Histogram *(1.5 Marks)*

◆ Scenario:

A teacher wants to analyze the distribution of marks scored by 20 students in a test.

◆ Marks Data:

```
csharp
CopyEdit
[45, 55, 58, 60, 65, 68, 70, 72, 75, 78,
 80, 82, 85, 87, 88, 90, 92, 93, 95, 98]
```

◆ Interpretation:

- We can divide the marks into bins such as 40–60, 61–70, 71–80, 81–90, 91–100.
- The histogram will show how many students scored within each range.

◆ Visual Output (Described):

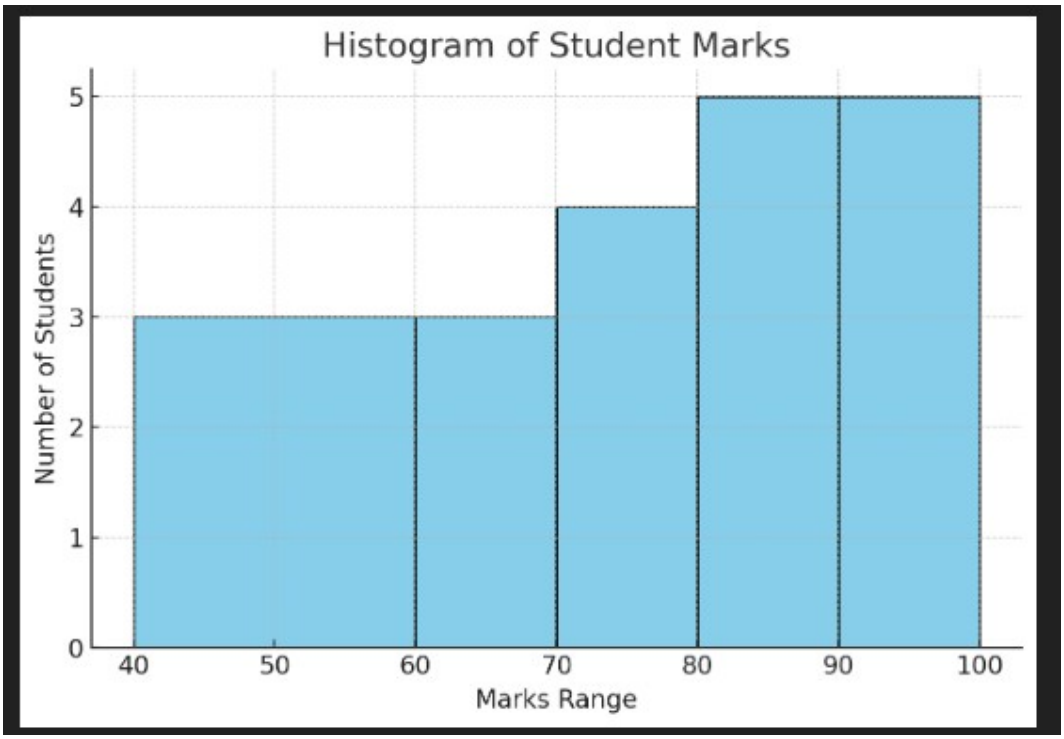
- Bin 40–60 → 3 students
- Bin 61–70 → 3 students
- Bin 71–80 → 4 students
- Bin 81–90 → 6 students
- Bin 91–100 → 4 students

This distribution tells us that most students scored between **81–90 marks**.

Re-import required libraries after kernel reset

import matplotlib.pyplot as plt

```
# Sample data: Marks scored by 20 students
marks = [45, 55, 58, 60, 65, 68, 70, 72, 75, 78,
         80, 82, 85, 87, 88, 90, 92, 93, 95, 98]

# Plotting the histogram
plt.figure(figsize=(8, 5))
plt.hist(marks, bins=[40, 60, 70, 80, 90, 100], color='skyblue', edgecolor='black')
plt.title("Histogram of Student Marks")
plt.xlabel("Marks Range")
plt.ylabel("Number of Students")
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.xticks([40, 50, 60, 70, 80, 90, 100])
plt.show()
```



✓ Visual Interpretation of Histogram

📊 Histogram of Student Marks

- **X-axis (Marks Range)**: Shows the intervals (bins) → [40–60], [60–70], [70–80], [80–90], [90–100]
- **Y-axis (Number of Students)**: Represents how many students scored within each interval

🔍 Insights from the Visual:

Marks Range	Number of Students
40–60	3
61–70	3
71–80	4
81–90	6
91–100	4

- The **most common score range** is **81–90**.
- The data shows a slightly **right-skewed distribution** (more students scored higher marks).
- This helps the teacher understand **where most students are performing**, and **how spread out** the scores are.

✓ 5. Usages of Histogram in Data Analysis *(2 Marks)*

Histograms are widely used in **Exploratory Data Analysis (EDA)** and **statistics**. Below are key use cases:

Use Case	Explanation
1. Understanding Distribution	Visualize shape – normal, skewed, bimodal, etc.
2. Identifying Outliers	Gaps or isolated bars may indicate outliers.

Use Case	Explanation
3. Quality Control in Manufacturing	Detect variations in product dimensions or measurements.
4. Data Cleaning & Transformation	Helps decide whether to apply normalization, binning, etc.
5. Examining Central Tendency	See where most data points are concentrated.
6. Supporting Statistical Analysis	Used in probability density estimation and histogram-based models.

✓ 6. Types of Distributions Shown by Histograms *(Optional Theory)*

Histograms help detect:

- **Normal distribution**: Bell-shaped curve
- **Left-skewed (negatively skewed)**: Tail on the left
- **Right-skewed (positively skewed)**: Tail on the right
- **Bimodal distribution**: Two peaks
- **Uniform distribution**: All bars are almost equal in height

Understanding the **distribution** helps in choosing the right **machine learning models or statistical methods**.

Q18. Describe the Data visualization tool "Tableau". Explain its applications in brief. [9]

=>

✓ 1. Introduction to Tableau *(2 Marks)*

Tableau is a leading **Business Intelligence (BI)** and **data visualization tool** that enables users to transform raw data into **interactive, insightful, and shareable dashboards** and reports. It simplifies complex data analysis through an intuitive, **drag-and-drop interface**, allowing both technical and non-technical users to **visualize data trends, patterns, and correlations** easily.

- Developed by **Tableau Software**, now part of **Salesforce**.
- Suitable for **structured and semi-structured** data.
- Often used in **data analytics, business intelligence, and data science** workflows.

✓ 2. Core Features of Tableau *(2 Marks)*

Feature	Description
Ease of Use	Drag-and-drop interface makes it easy to use for beginners
Multiple Data Sources	Connects to Excel, CSV, SQL, NoSQL, Hadoop, Google BigQuery, AWS Redshift, etc.
Live & Extract Modes	Supports real-time data streaming (Live) and in-memory processing (Extract)
Dashboarding	Allows combining multiple visualizations into interactive dashboards
Storytelling	Helps narrate data-driven stories via sequenced visual insights
Data Blending & Joins	Combine data from different sources for a unified view
Interactivity	Filters, tooltips, drill-downs, and highlight actions for user interaction

---

✓ 3. Tableau Architecture Overview *(1 Mark)*

Component	Purpose
Tableau Desktop	Authoring tool for creating dashboards and visualizations
Tableau Server	Enterprise-level platform to publish and share dashboards securely
Tableau Public	Free cloud-based platform to publish visualizations to the web
Tableau Reader	For offline viewing of dashboards created in Tableau Desktop
Tableau Online	Cloud-hosted version of Tableau Server
Tableau Prep	For data cleaning, shaping, and transformation before visualization

---

✓ 4. Real-World Applications of Tableau *(3 Marks)*

Here are **five key applications** of Tableau across different industries with use cases:

◆ A. Business Intelligence & Sales Analytics

- Track key performance indicators (KPIs) like sales, revenue, and profit.
- Create real-time dashboards for monitoring regional or product-wise performance.
- **Use Case:** A sales manager monitors live sales trends and product performance to boost growth.

◆ B. Healthcare & Public Health Monitoring

- Analyze patient data, hospital efficiency, and outbreak tracking (e.g., COVID-19).
- Visualize healthcare metrics like admission rates, disease occurrence, and vaccination status.
- **Use Case:** A hospital uses Tableau to track patient recovery trends and allocate beds/resources.

◆ C. Finance & Banking

- Visualize financial reports, risk analysis, credit scoring, and investment tracking.
- Generate automated monthly/quarterly reports for management.
- **Use Case:** A bank visualizes loan default trends and customer credit scores across regions.

◆ D. Marketing & Customer Behavior

- Understand customer segmentation, marketing ROI, and campaign effectiveness.
- Combine data from social media, CRM, and email marketing tools.
- **Use Case:** A marketing team visualizes which campaign brought the most conversions.

◆ E. Education & Student Performance Analysis

- Visualize student grades, attendance, course enrollment, and departmental analysis.
- Identify performance trends across schools, departments, or individuals.
- **Use Case:** A university uses Tableau to identify low-performing students and intervene early.

✓ 5. Advantages of Tableau *(1 Mark)*

Advantage	Description
Highly Interactive	Filters, drill-downs, tooltips allow better exploration of data
Supports Big Data	Connects with Hadoop, Spark, cloud storage (AWS, Google Cloud)
User Friendly	No coding required—ideal for business analysts and managers
Real-Time Analysis	With Live Connection, dashboards reflect up-to-date data instantly
Beautiful Visuals	Allows rich visual customizations and storyboarding

---

✓ 6. Conclusion *(0.5 Marks)*

Tableau is a powerful, flexible, and easy-to-use **data visualization tool** that turns raw data into meaningful and actionable insights. It is widely adopted in various industries including **finance, healthcare, education, retail, and marketing** due to its ability to connect to multiple data sources, generate interactive dashboards, and support real-time analysis. Tableau makes **data-driven decision-making** faster and more efficient.

Q19. With a suitable example explain and draw a Box plot and explain its usages. [8]

=>

✓ 1. Definition of Box Plot *(2 Marks)*

A **Box Plot**, also known as a **Box-and-Whisker Plot**, is a graphical representation used to summarize the **distribution of a dataset** through its **five-number summary**:

- **Minimum** (smallest value excluding outliers)
- **First Quartile (Q1)** (25th percentile)
- **Median (Q2)** (50th percentile)
- **Third Quartile (Q3)** (75th percentile)
- **Maximum** (largest value excluding outliers)

It also helps in detecting **outliers**.

✓ 2. Components of a Box Plot *(1.5 Marks)*

Component	Description
Box	Spans from Q1 to Q3, representing the interquartile range (IQR)
Median Line	A line inside the box indicating the median (Q2)
Whiskers	Lines extending from the box to the minimum and maximum values within 1.5×IQR from Q1 and Q3
Outliers	Data points beyond whiskers, plotted as individual dots

---

✓ 3. Suitable Example and Plot *(2.5 Marks)*

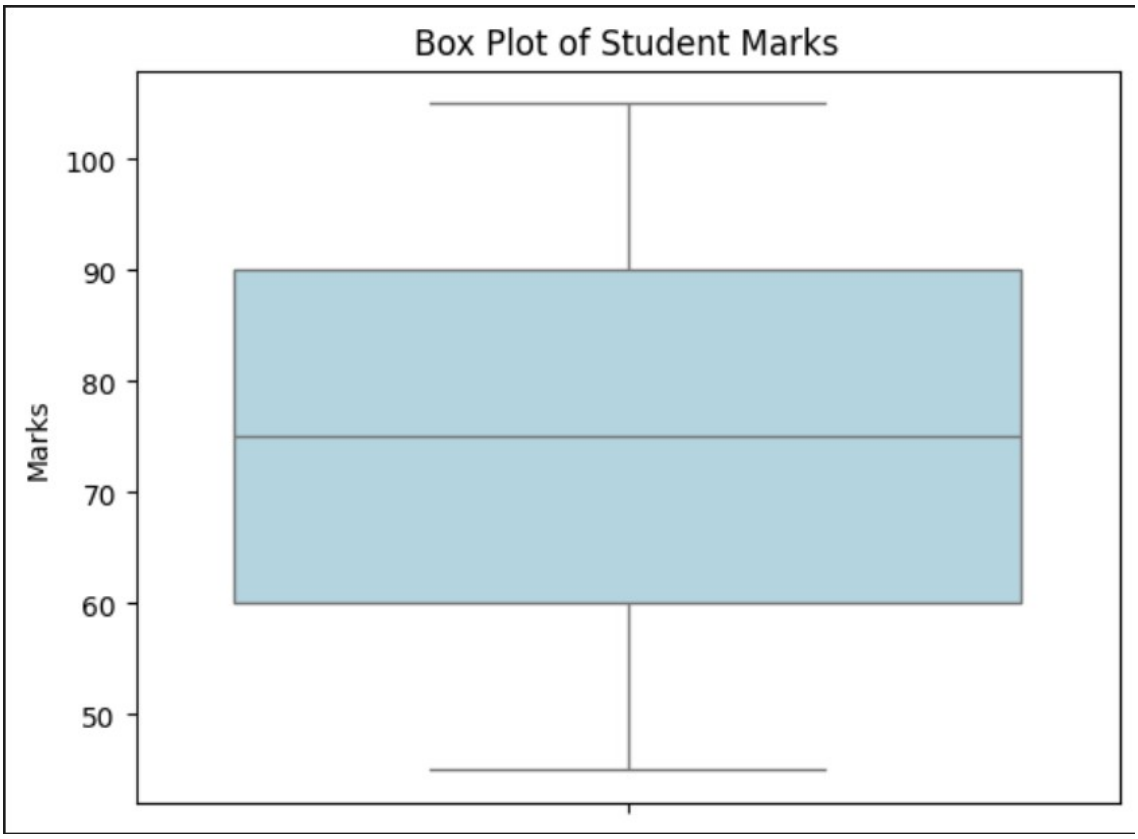
◆ Example Dataset:

Marks scored by students in an exam:

```
[45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105]
```

◆ Box Plot Code in Python (Seaborn/Matplotlib):

```
python
CopyEdit

import matplotlib.pyplot as plt
import seaborn as sns

marks = [45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105]

sns.boxplot(data=marks, color='lightblue')
plt.title('Box Plot of Student Marks')
plt.ylabel('Marks')
plt.show()
```



◆ Explanation of the Plot:

- The **box** shows the **middle 50%** of marks (from Q1 to Q3).
- The **median line** indicates the 50th percentile mark.
- **Whiskers** extend to the smallest and largest values within 1.5×IQR.
- Points beyond whiskers (e.g., 105) are plotted as **outliers**.

✓ 4. Usages of Box Plot *(2 Marks)*

Usage	Explanation
1. Summarize Data Distribution	Quickly shows spread, center, and skewness
2. Detect Outliers	Easily identifies anomalous data points
3. Compare Groups	Compare multiple distributions side-by-side
4. Understand Skewness	Position of median relative to quartiles shows skew

Usage	Explanation
5. Summarize Large Data	Effective even with large datasets where scatter plots get cluttered

---

✓ 5. Conclusion

A **Box Plot** is an essential tool in **exploratory data analysis (EDA)** that provides a **compact summary of data distribution**.
It helps analysts and decision-makers identify **variability, central tendency, and outliers** easily and effectively.

Q20. Describe the challenges of data visualization. Draw box plot and explain its usages. [9]

=>

✓ 1. Challenges of Data Visualization *(4 Marks)*

Data visualization is essential but comes with several challenges:

Challenge	Explanation
1. Data Quality Issues	Missing, inconsistent, or noisy data can lead to misleading or incorrect visuals.
2. Choosing the Wrong Visualization	Using inappropriate charts can confuse or mislead the audience (e.g., pie charts for time series).
3. Overplotting and Clutter	Large datasets can cause overlapping points, making it hard to interpret visuals.
4. Scalability Issues	Visualizing huge datasets may be slow or cause system crashes.
5. Misleading Axes and Scales	Improper axis scaling can distort data interpretation (e.g., truncated y-axis).
6. High Dimensionality	Visualizing many variables simultaneously is complex and requires advanced techniques.
7. User Misinterpretation	Non-expert users may misread data or misunderstand visualizations.

✓ 2. Box Plot: Definition and Components *(2 Marks)*

A **Box Plot** (or Box-and-Whisker Plot) visually summarizes data distribution using:

- **Minimum** (excluding outliers)
- **First Quartile (Q1)**
- **Median (Q2)**
- **Third Quartile (Q3)**
- **Maximum** (excluding outliers)
- **Outliers** (individual points beyond whiskers)

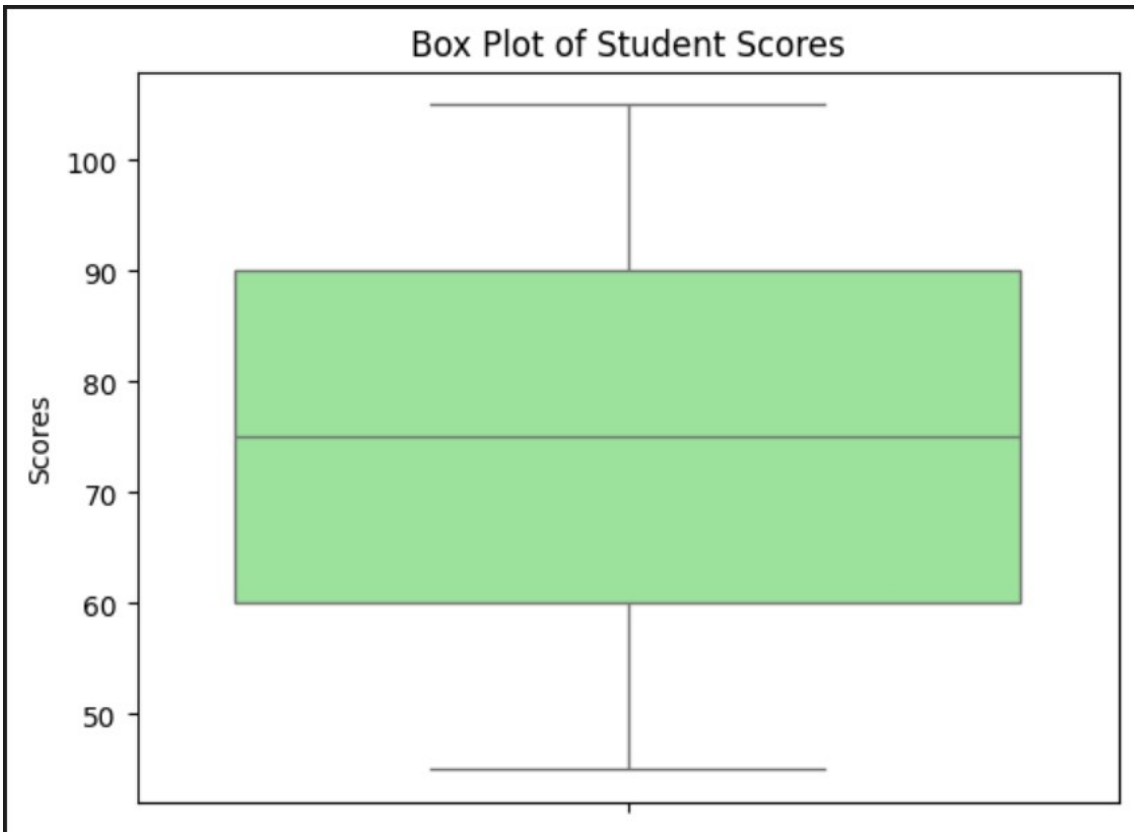
✓ 3. Box Plot Diagram and Example *(3 Marks)*

Example Data:

Student exam scores:
`[45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105]`

Python code to draw box plot:

```
python
CopyEdit

import matplotlib.pyplot as plt
import seaborn as sns

scores = [45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105]

sns.boxplot(data=scores, color='lightgreen')
plt.title('Box Plot of Student Scores')
plt.ylabel('Scores')
plt.show()
```



Visual Description:

- The **box** covers the interquartile range (IQR) between Q1 and Q3.
- The **median** is shown as a line inside the box.
- **Whiskers** extend to the smallest and largest data points within 1.5×IQR.
- The **dot beyond whiskers** (105) is an outlier.

✓ 4. Usages of Box Plot *(Optional explanation)*

- Summarize data distribution clearly and compactly.
- Identify median and spread of data.
- Detect outliers and extreme values.
- Compare distributions across multiple groups side-by-side.
- Understand data skewness (position of median relative to box).

✓ 5. Conclusion

While data visualization aids in interpreting complex datasets, challenges like **data quality, scalability, and chart selection** require attention.

The **box plot** is a simple yet powerful tool to **summarize and compare data distributions**, identify outliers, and analyze variability effectively.