As per the New Choice Based Credit System (CBCS) Syllabus of Mumbai University w.e.f. academic year 2023 - 2024

Data Science

(USCS601) **(Core Subject)**

T.Y.B.Sc. (Computer Science) Semester VI

Dr. Shraddha Sable **Vaishali C. Shelar**

TechKnowledge™
Publications

Data Science

(USCS601) (Core Subject)

Semester VI – T. Y. B.Sc. (Computer Science) (Mumbai University)

Choice Based Credit System with effect from Academic Year 2023-2024

Dr. Shraddha Sable

Ph. D. (Computer Science)

Assistant Professor

S.K. College of Science and Commerce,

Nerul, Navi Mumbai, Maharashtra, India

Vaishali Chandrakant Shelar

Trainer & Mentor of change in

Atal Innovation Mission (AIM), Mumbai.

TechKnowledge™ Publications

MCE37A Price ₹ 225/-

SYLLABUS

Course Code	Course Title	Credits	Lectures/Week
USCS601	Data Science	2	3

Unit	Topics	No. of Lectures
I	**Introduction to Data Science and Data Preprocessing** **What is Data Science?:** Definition and scope of Data Science, Applications and domains of Data Science, Comparison with other fields like Business Intelligence (BI), Artificial Intelligence (AI), Machine Learning (ML), and Data Warehousing/Data Mining (DW-DM) **Data Types and Sources :** Different types of data: structured, unstructured, semi-structured, Data sources: databases, files, APIs, web scraping, sensors, social media **Data Preprocessing :** Data cleaning: handling missing values, outliers, duplicates, Data transformation: scaling, normalization, encoding categorical variables, Feature selection: selecting relevant features/columns, Data merging: combining multiple datasets **Data Wrangling and Feature Engineering :** Data wrangling techniques: reshaping, pivoting, aggregating, Feature engineering: creating new features, handling time-series data Dummification: converting categorical variables into binary indicators, Feature scaling: standardization, normalization **Tools and Libraries :** Introduction to popular libraries and technologies used in Data Science like Pandas, NumPy, Sci-kit Learn, etc.	15
II	**Data Analysis and Machine Learning** **Exploratory Data Analysis (EDA) :** Data visualization techniques: histograms, scatter plots, box plots, etc., Descriptive statistics: mean, median, mode, standard deviation, etc., Hypothesis testing: t-tests, chi-square tests, ANOVA, etc. **Introduction to Machine Learning :** Supervised learning: classification and regression, Unsupervised learning: clustering and dimensionality reduction, Bias-variance tradeoff, underfitting, and overfitting **Regression Analysis :** Simple linear regression, Multiple linear regression, Stepwise regression, Logistic regression for classification **Model Evaluation and Selection :** Techniques for evaluating model performance: accuracy, precision, recall, F1-score, Confusion matrix and ROC curve analysis, Cross-validation: k-fold cross-validation, stratified cross-validation, Hyperparameter tuning and model selection **Machine Learning Algorithms :** Decision Trees and Random Forests, Support Vector Machines (SVM), Artificial Neural Networks (ANN), Ensemble Learning: Boosting and Bagging, K-Nearest Neighbors (K-NN), Gradient Descent for optimization	15

Unit	Topics	No. of Lectures
III	**Model Evaluation, Data Visualization, and Management** **Model Evaluation Metrics :** Accuracy, precision, recall, F1-score, Area Under the Curve (AUC), Evaluating models for imbalanced datasets **Data Visualization and Communication :** Principles of effective data visualization, Types of visualizations: bar charts, line charts, scatter plots, etc. Visualization tools: matplotlib, seaborn, Tableau, etc. Data storytelling: communicating insights through visualizations **Data Management :** Introduction to data management activities, Data pipelines: data extraction, transformation, and loading (ETL), Data governance and data quality assurance, Data privacy and security considerations.	15

❑❑❑

Lab Syllabus

Course Code	Course Title	Credits	Lectures/Week
USCSP601	Data Science - Practical	1	3

1	Introduction to Excel • Perform conditional formatting on a dataset using various criteria. • Create a pivot table to analyze and summarize data. • Use VLOOKUP function to retrieve information from a different worksheet or table. • Perform what-if analysis using Goal Seek to determine input values for desired output.
2	Data Frames and Basic Data Pre-processing • Read data from CSV and JSON files into a data frame. • Perform basic data pre-processing tasks such as handling missing values and outliers. • Manipulate and transform data using functions like filtering, sorting, and grouping..
3	Feature Scaling and Dummification • Apply feature-scaling techniques like standardization and normalization to numerical features. • Perform feature dummification to convert categorical variables into numerical representations.

4	Hypothesis Testing • Formulate null and alternative hypotheses for a given problem. • Conduct a hypothesis test using appropriate statistical tests (e.g., t-test, chi-square test). • Interpret the results and draw conclusions based on the test outcomes.
5	ANOVA (Analysis of Variance) • Perform one-way ANOVA to compare means across multiple groups. • Conduct post-hoc tests to identify significant differences between group means.
6	Regression and Its Types • Implement simple linear regression using a dataset. • Explore and interpret the regression model coefficients and goodness-of-fit measures. • Extend the analysis to multiple linear regression and assess the impact of additional predictors.
7	Logistic Regression and Decision Tree • Build a logistic regression model to predict a binary outcome. • Evaluate the model's performance using classification metrics (e.g., accuracy, precision, recall). • Construct a decision tree model and interpret the decision rules for classification.
8	K-Means Clustering • Apply the K-Means algorithm to group similar data points into clusters. • Determine the optimal number of clusters using elbow method or silhouette analysis. • Visualize the clustering results and analyze the cluster characteristics.
9	Principal Component Analysis (PCA) • Perform PCA on a dataset to reduce dimensionality. • Evaluate the explained variance and select the appropriate number of principal components. • Visualize the data in the reduced-dimensional space.
10	Data Visualization and Storytelling • Create meaningful visualizations using data visualization tools • Combine multiple visualizations to tell a compelling data story. • Present the findings and insights in a clear and concise manner.

❑❑❑

Unit - I

Chapter 1 : Introduction to Data Science and Data Preprocessing 1-1 to 1-36

What is Data Science? - Definition and scope of Data Science, Applications and domains of Data Science, Comparison with other fields like Business Intelligence (BI), Artificial Intelligence (AI), Machine Learning (ML), and Data Warehousing/Data Mining (DW-DM).

Data Types and Sources : Different types of data : structured, unstructured, semi-structured, Data sources : databases, files, APIs, web scraping, sensors, social media

Data Preprocessing : Data cleaning : handling missing values, outliers, duplicates, Data transformation : scaling, normalization, encoding categorical variables, Feature selection : selecting relevant features/columns, Data merging : combining multiple datasets

Data Wrangling and Feature Engineering : Data wrangling techniques : reshaping, pivoting, aggregating, Feature engineering : creating new features, handling time-series data Dummification : converting categorical variables into binary indicators, Feature scaling : standardization, normalization

Tools and Libraries : Introduction to popular libraries and technologies used in Data Science like Pandas, NumPy, Sci-kit Learn, etc.

Unit - II

Exploratory Data Analysis (EDA) : Data visualization techniques: histograms, scatter plots, box plots, etc., Descriptive statistics: mean, median, mode, standard deviation, etc., Hypothesis testing: t-tests, chi-square tests, ANOVA, etc.

Introduction to Machine Learning : Supervised learning: classification and regression, Unsupervised learning: clustering and dimensionality reduction, Bias-variance tradeoff, underfitting, and overfitting

Regression Analysis : Simple linear regression, Multiple linear regression, Stepwise regression, Logistic regression for classification

Model Evaluation and Selection : Techniques for evaluating model performance: accuracy, precision, recall, F1-score, Confusion matrix and ROC curve analysis, Cross-validation: k-fold cross-validation, stratified cross-validation, Hyperparameter tuning and model selection

Machine Learning Algorithms : Decision Trees and Random Forests, Support Vector Machines (SVM), Artificial Neural Networks (ANN), Ensemble Learning: Boosting and Bagging, K-Nearest Neighbors (K-NN), Gradient Descent for optimization

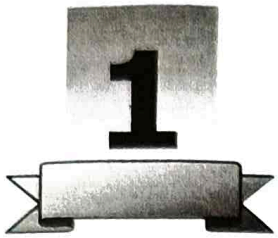
Unit - III

Chapter 3 : Model Evaluation, Data Visualization and Management 3-1 to 3-15

Model Evaluation Metrics : Accuracy, precision, recall, F1-score, Area Under the Curve (AUC), Evaluating models for imbalanced datasets

Data Visualization and Communication : Principles of effective data visualization, Types of visualizations : bar charts, line charts, scatter plots, etc. Visualization tools : matplotlib, seaborn, Tableau, etc. Data storytelling : communicating insights through visualizations

Data Management : Introduction to data management activities, Data pipelines : data extraction, transformation, and loading (ETL), Data governance and data quality assurance, Data privacy and security considerations



Introduction to Data Science and Data Preprocessing

Syllabus

What is Data Science? - Definition and scope of Data Science, Applications and domains of Data Science, Comparison with other fields like Business Intelligence (BI), Artificial Intelligence (AI), Machine Learning (ML), and Data Warehousing/Data Mining (DW-DM).

Data Types and Sources : Different types of data : structured, unstructured, semi-structured, Data sources : databases, files, APIs, web scraping, sensors, social media

Data Preprocessing : Data cleaning : handling missing values, outliers, duplicates, Data transformation : scaling, normalization, encoding categorical variables, Feature selection : selecting relevant features/columns, Data merging : combining multiple datasets

Data Wrangling and Feature Engineering : Data wrangling techniques : reshaping, pivoting, aggregating, Feature engineering : creating new features, handling time-series data Dummification : converting categorical variables into binary indicators, Feature scaling : standardization, normalization

Tools and Libraries : Introduction to popular libraries and technologies used in Data Science like Pandas, NumPy, Sci-kit Learn, etc.

1.1. What is Data Science?

1.1.1 Definition and Scope of Data Science

Data Science is a multidisciplinary field that involves using various techniques, algorithms, processes, and systems to extract valuable insights and knowledge from structured and unstructured data. It combines elements from statistics, computer science, mathematics, domain expertise, and data engineering to analyze and interpret large volumes of data and make data-driven decisions.

Let's break down the concept of Data Science further :

1. **Data Collection :** The first step in data science is gathering relevant data. This data can be sourced from a wide range of places, including databases, sensors, websites, social media, and more.
2. **Data Cleaning :** Raw data is often messy, with missing values, inconsistencies, and errors. Data scientists clean and pre-process the data to make it suitable for analysis.

3. **Data Exploration :** After cleaning, data scientists explore the data to understand its characteristics. They use statistical methods, visualization tools, and domain knowledge to gain insights into the data's patterns and trends.
4. **Feature Engineering :** This involves selecting, transforming, or creating new features (variables) from the data that are relevant for the analysis. Feature engineering is crucial for building accurate predictive models.
5. **Model Building :** Data scientists use various machine learning and statistical modeling techniques to build predictive models or uncover patterns in the data. Common algorithms include regression, decision trees, neural networks, and clustering.
6. **Model Evaluation :** Once models are built, they need to be evaluated for their accuracy and performance. This involves using metrics such as accuracy, precision, recall, and F1-score to assess how well the model is doing.
7. **Model Deployment :** Successful models are deployed into production systems where they can make real-time predictions or provide insights for decision-making.
8. **Continuous Monitoring and Improvement :** Data scientists monitor the performance of deployed models and continuously refine them as new data becomes available or as the business environment changes.

Examples of Data Science in Action :

1. **Recommendation Systems :** Netflix and Amazon use data science to recommend movies or products based on your previous choices and user behaviour.
2. **Predictive Analytics :** Financial institutions use data science to predict credit risk and detect fraudulent transactions.
3. **Healthcare :** Data scientists analyze patient data to predict disease outcomes and optimize treatment plans.
4. **Social Media :** Social media platforms use data science to personalize content feeds, target ads, and analyze user engagement.

In summary, data science is a powerful field that leverages data to gain insights, make predictions, and drive decision-making across various industries. It combines skills in data manipulation, statistics, machine learning, and domain expertise to extract valuable information from data and create actionable insights.

1.1.2 Applications and Domains of Data Science

Data science has a wide range of applications across various domains, revolutionizing industries, and decision-making processes. In this chapter, we will explore some of the most prominent applications of data science and the domains where data science plays a crucial role.

Understanding Data Science Applications

Data science is all about extracting valuable insights and knowledge from data. Its applications are diverse and versatile, making it a transformative field across numerous sectors.

1. **Data Science in Healthcare**

   I. **Predictive Analytics for Disease Diagnosis**

   Data science leverages machine learning algorithms to analyse patient data, including medical history, symptoms, and test results. By identifying patterns and correlations, it assists in early disease detection and diagnosis.

   Example :

   Predicting the likelihood of diabetes in a patient based on their genetic markers and lifestyle factors.

   II. **Drug Discovery and Development**

   Data science accelerates drug discovery by analyzing molecular data, genetic information, and clinical trial results. These speeds up the identification of potential drug candidates.

   Example :

   Analyzing the genomic data of patients to discover biomarkers for targeted cancer therapies.

   III. **Healthcare Fraud Detection**

   Data science helps healthcare providers and insurers identify fraudulent claims through anomaly detection and pattern recognition.

   Example :

   Detecting abnormal billing patterns that suggest fraudulent insurance claims.

2. **Data Science in Finance**

   I. **Algorithmic Trading**

   Data science is used to build predictive models for stock price movements, enabling automated trading strategies.

   Example :

   Developing trading algorithms that analyze market data and execute buy/sell orders based on predefined rules.

   II. **Credit Risk Assessment**

   Data science assesses the credit worthiness of individuals and businesses by analyzing credit history and other relevant data.

   Example :

   Using machine learning to predict the likelihood of a borrower defaulting on a loan based on their financial history.

   III. **Fraud Detection**

   Data science helps financial institutions detect fraudulent transactions by identifying unusual spending patterns and anomalies.

   Example :

   Monitoring credit card transactions in real-time to flag suspicious activity.

3. Data Science in Marketing

I. Customer Segmentation

Data science divides a customer base into segments based on behaviour, demographics, or preferences. This allows for more targeted marketing campaigns.

Example :

Segmenting e-commerce customers into "frequent shoppers," "one-time buyers," and "window shoppers."

II. Personalized Recommendations

Data science powers recommendation systems that suggest products, services, or content based on user behaviour and preferences.

Example :

Netflix using machine learning to recommend movies and TV shows to its subscribers based on viewing history.

III. A/B Testing

Data science helps marketers conduct experiments to compare two or more variations of a webpage, email, or advertisement to determine which performs better.

Example :

Testing two different email subject lines to see which one results in higher open rates.

4. Data Science in Education

I. Adaptive Learning

Data science personalizes education by analyzing student data and adapting the curriculum to individual learning needs.

Example :

An online learning platform using data analytics to adjust the difficulty of math problems based on a student's progress.

II. Predictive Student Success

Data science predicts student success and identifies at-risk students by analyzing academic performance and engagement metrics.

Example :

A university using data to identify students who may need additional support to improve their chances of graduation.

5. Data Science in Manufacturing

I. Predictive Maintenance

Data science predicts when machines and equipment are likely to fail, reducing downtime and maintenance costs.

Example :

Using sensor data and machine learning to predict when an industrial machine needs maintenance.

II. Quality Control

Data science monitors product quality and identifies defects in real-time by analyzing data from sensors and cameras.

Example :

A manufacturing plant using computer vision to inspect product quality on the assembly line.

6. Data Science in Social Media

I. Sentiment Analysis

Data science analyses social media content to understand public sentiment and opinions.

Example :

Tracking social media mentions of a brand to gauge public perception and sentiment.

II. Content Recommendation

Data science powers content recommendation algorithms, such as those used by YouTube and Facebook, to suggest posts, videos, or articles to users.

Example :

Facebook recommending friends to connect with based on mutual interests and connections.

7. Data Science in Government

I. Crime Prediction

Data science models use historical crime data to predict areas with a high likelihood of criminal activity, assisting law enforcement agencies.

Example :

Predicting hotspots for criminal activity to allocate police resources effectively.

8. Healthcare Management

Data science is employed in healthcare administration for resource allocation, patient management, and disease tracking.

Example :

Analyzing healthcare data to identify areas at risk of disease outbreaks and directing vaccination campaigns.

Data science's applications are widespread and continually expanding into new areas. It empowers decision-making, drives efficiency, and unlocks insights that were previously hidden within vast datasets. As the field of data science continues to evolve, its impact on various domains will only become more significant, transforming the way industries operate and make informed choices. Understanding the potential applications of data science is key to harnessing its power for the benefit of society and business alike.

1.1.3 A Comparison with Other Fields like BI, AI, ML, DW-DM

Sr. No.	Aspect	Business Intelligence (BI)	Artificial Intelligence (AI)	Machine Learning (ML)	Data Warehousing/ Data Mining (DW-DM)	Data Science
1.	Definition	Analyzes historical data and provides insights into past performance.	Simulates human intelligence and performs tasks that typically require human intelligence.	Focuses on developing algorithms that can learn from and make predictions on data.	Involves collecting, storing, and managing data, and extracting valuable information through querying and analysis.	Multidisciplinary field that combines various techniques to extract insights from data.
2.	Purpose	Supports decision-making by providing insights and reports.	Aims to replicate human cognitive functions, such as reasoning, problem-solving, and perception.	Focuses on creating predictive models and making data-driven predictions.	Primarily deals with data storage, retrieval, and mining.	Covers the entire data lifecycle, from data collection to analysis to decision-making.
3.	Interactivity	Typically provides static reports and dashboards.	Can interact with users through natural language processing (NLP) and conversational interfaces.	Can adapt and learn from new data.	Mostly query-based with limited interactivity.	Interactive data exploration and visualization.
4.	Automation	Limited automation, primarily for reporting.	High degree of automation for complex tasks.	Automation of model building and predictions.	Automation of data extraction and pattern discovery.	Combines automation with human expertise.
5.	Data Type	Relies on structured data sources.	Can work with structured, semi-structured, and unstructured data.	Can handle various data types, including structured and unstructured.	Focuses on structured data for analysis.	Works with structured and unstructured data.

Sr. No.	Aspect	Business Intelligence (BI)	Artificial Intelligence (AI)	Machine Learning (ML)	Data Warehousing/ Data Mining (DW-DM)	Data Science
6.	Problem Solving	Primarily descriptive analytics.	Solves complex problems through reasoning and learning.	Focuses on predictive and prescriptive analytics.	Focuses on pattern recognition and knowledge discovery.	Addresses a wide range of problems, from descriptive to predictive to prescriptive.
7.	Learning	Typically does not learn or adapt over time.	Learns from data and experience, improving performance.	Learns patterns and makes predictions based on data.	Does not learn but extracts patterns from data.	Combines learning with traditional statistical methods.
8.	Complexity	Generally lower complexity tasks, such as reporting and dashboard creation.	Handles complex tasks like natural language understanding and image recognition.	Complex algorithms for predictive modeling and pattern recognition.	Complex data storage and retrieval systems.	Integrates complex processes from data collection to analysis.
9.	Use of Algorithms	Limited use of advanced algorithms.	Relies on a wide range of algorithms, including neural networks and deep learning.	Utilizes machine learning algorithms for prediction and classification.	Utilizes algorithms for data transformation and pattern discovery.	Leverages various algorithms, including ML, for analysis.
10.	Decision Support	Supports historical and descriptive decision-making.	Supports predictive and prescriptive decision-making.	Supports predictive and prescriptive decision-making.	Facilitates data-driven decision-making.	Supports data-driven decision-making across domains.
11.	Data Integration	Integrates data from various sources for reporting.	Integrates data from multiple sources for AI applications.	Requires data integration for model training.	Integrates and stores data for mining and analysis.	Integrates data for analysis and modeling.

Sr. No.	Aspect	Business Intelligence (BI)	Artificial Intelligence (AI)	Machine Learning (ML)	Data Warehousing/ Data Mining (DW-DM)	Data Science
12.	Time Horizon	Focuses on historical and current data.	Can handle both historical and real-time data.	Focuses on future predictions based on past data.	Primarily focuses on historical data for mining.	Addresses historical and future data needs.
13.	Domain Expertise	Less emphasis on domain expertise.	Requires domain expertise for problem framing and data labeling.	Benefits from domain knowledge for feature engineering and model evaluation.	Requires domain knowledge for effective data mining.	Collaborates closely with domain experts.
14.	Scalability	Scales well for reporting and dash boarding.	Scales for various applications depending on computational resources.	Scalable for large-scale machine learning tasks.	Scalable for large data volumes but may require significant hardware resources.	Scalable for diverse data science projects.
15.	Governance and Compliance	Supports data governance and compliance through reporting.	May require ethical considerations for AI decision-making.	Compliance is essential in handling sensitive data.	Compliance is crucial for data privacy and security.	Adheres to data governance and compliance standards.
16.	Hardware Requirements	Typically runs on standard servers.	May require specialized hardware for deep learning tasks.	Benefits from GPU and cloud computing for model training.	Requires storage infrastructure for data warehousing.	May utilize cloud computing resources for scalability.
17.	Industry Applications	Commonly used in business, finance, and marketing.	Applied across various industries, including healthcare, finance, and automotive.	Widely used in healthcare, finance, and tech industries.	Applied in finance, retail, and customer relationship management.	Applicable in virtually every industry with data-driven needs.

1.2 What is Data?

- Data is defined as raw facts and figures collected and stored in database. Data can be in structured, semi structured, and unstructured format. Data are records which are collected by various ways, large number of resources generates data, and this data is in different formats.
- For example, if number of males and females are counted in specific location then the values represented as no of males and females is data as it is a fact.
- Data can be processed to get information. Processing can include combining data from various sources, collecting data, converting, or transforming data into a specific format, summarizing, modeling etc.
- Data can be measured, collected, presented, and analysed by using various tools. Once the data is collected from various sources it is in the raw format and hence it is also known as raw data.
- Raw data then must undergo through the important phase of cleaning. Data cleaning method is used to remove the garbage and unwanted values.
- Raw data collected from various source will not always provide the proper values and hence data cleaning and then data evaluating phase after collection provides assurance about the genuineness of the data. Data is usually evaluated by comparing it with some standard values or by validating it through the experts.
- Fig. 1.2.1 shows the knowledge data and information pyramid.

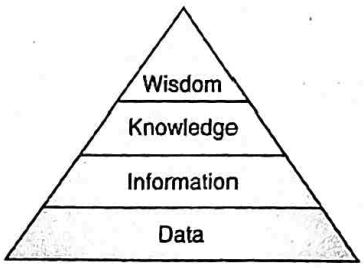


Fig. 1.2.1 : Data, information, knowledge triangle

1. **Data**

   Raw Facts and Figures.

2. **Information**

   Processed Data provides Information.

3. **Knowledge**

   Mastering the use of information in particular fashion provides knowledge.

4. **Wisdom**

   Application of the knowledge is known as wisdom.

1.2.1 Different Kinds of Data

Data at its highest classification levels is of two types :

1. Quantitative data
2. Qualitative data

1. Quantitative Data

- It is represented using numbers or anything through which someone can measure various dimensions such as height, weight, width, length, temperature, humidity, cost etc.
- Quantitative data are further classified as discrete data and continuous data.
  (a) Discrete data
  (b) Continuous data

(a) Discrete Data : Data which can be counted completely is discrete data it is mostly the integer values for example number of children in family, no of players in cricket team etc, are the discrete values.

(b) Continuous Data : Continuous data is divided into the finer levels, and they are usually floating-point values. Example of continuous data can be height, weight, length etc.

2. Qualitative Data

- It provides the characteristics and descriptors which cannot be easily measured. Qualitative data can be observed subjectively. For example, smell, taste, textures, color etc.
- Qualitative data are further classified as binomial, nominal, and ordinal data.
  (a) Binomial data
  (b) Nominal data
  (c) Ordinal data

(a) Binomial Data : Binomial means two values which are like binary data, two values can be true or false, yes or no, except or reject, right or wrong etc.

(b) Nominal Data : It is also known as unordered data here every individual element will not have a kind of ranking, but it will have some categories. For example, let us say there are 10 items which are having different colours so they can be categories according to colour if the next value comes then it is easily categories.

(c) Ordinal Data : Ordinal data is also known as ordered data here every element have some kind of order for example short, medium, tall can be three categories for height and now if look at their names they follow some order.

Data can be also further classified into three main types which are :

i) Structured Data

ii) Unstructured Data

iii) Semi-Structured Data

i) Structured Data : Structured data is highly organized and follows a specific schema or data model. It is typically found in relational databases and spreadsheets. Key characteristics include :

- **Tabular Structure :** Data is organized into rows and columns.
- **Fixed Schema :** Data adheres to a predefined structure with well-defined data types for each column.
- **Ease of Querying :** Structured data is easy to query using SQL or similar languages.

Examples of Structured Data

- **Relational Database :** A customer database in an e-commerce system. It includes structured data such as customer IDs, names, addresses, purchase history, and order details organized in tables.
- **Excel Spreadsheet :** A financial statement containing structured data, including date, income, expenses, and profit columns in a tabular format.

ii) **Unstructured Data :** Unstructured data lacks a predefined structure or format. It is typically in the form of text, images, audio, or video and does not fit neatly into rows and columns. Key characteristics include :

- **Lack of Structure :** Data doesn't follow a specific format or schema.
- **High Complexity :** Unstructured data can be complex and require advanced techniques for analysis.
- **Natural Language :** Text data is a common form of unstructured data, including documents, social media posts, and emails.

Examples of Unstructured Data

- **Text Documents :** A collection of customer reviews for a product, where each review is in free-text form, making it unstructured.
- **Image Data :** A repository of medical images, such as X-rays, MRIs, or histopathology slides. These images contain unstructured visual information.

iii) **Semi-Structured Data :** Semi-structured data lies between structured and unstructured data. It has some level of structure but is more flexible than structured data. Key characteristics include :

- **Partially Defined Schema :** Semi-structured data may have some structure, but it is not as rigid as structured data.
- **Flexibility :** Data elements can vary in structure and attributes.
- **Use of Tags or Markers :** XML, JSON, and YAML are common formats for representing semi-structured data.

Examples of Semi-Structured Data

- **JSON Data :** An API response containing product information, where each product has a name, price, and description, but additional attributes can vary.

```
{
  "product1": {
  "name": "Smartphone",
  "price": 499.99,
  "description": "High-end mobile phone with advanced features."
},
"product2": {
  "name": "Laptop",
  "price": 999.99,
  "description": "Powerful laptop for work and gaming.",
```

```
    "color": "Silver"
  }
}
```

- **XML Document :** An RSS feed that contains news articles, where each article has a title, publication date, and content, but can include optional elements like author and tags.

```
<rss>
  <channel>
   <item>
    <title>Breaking News</title>
   <pubDate>2023-10-15</pubDate>
   <description>Important news update.</description>
   <author>John Doe</author>
   <tags>
   <tag>Politics</tag>
  <tag>Economy</tag>
 </tags>
 </item>
</channel>
</rss>
```

Understanding the types of data - structured, unstructured, and semi-structured - is crucial in data science. Each type presents unique challenges and opportunities. Structured data offers simplicity and ease of analysis, unstructured data holds a wealth of information, and semi-structured data strikes a balance between structure and flexibility. Effective data handling and analysis often involve dealing with all three data types, depending on the specific use case and the insights you seek to extract from your data.

1.2.2 Data Sources

Data can be gathered from a wide range of sources, and understanding these sources is essential in data science. Data sources are the resources through which the data is made available for analysis or any other use. Data available from various sources can be in structured, semi structured or unstructured format. There are various data sources which are available which includes,

- **File Systems :** File systems are for example CSV, excel files, etc.
- **Relational Systems :** They include various databases such as Oracle, Sql Server, DB2, etc.
- **Cloud Systems :** In this data source data is store on the cloud various system include here are Windows Azure, Google BigQuery, etc.
- **Websites :** Various websites provides data in various formats.
- **Live Streaming Data :** These are the data available from various electronic gadget available in the digital format.
- **Electronic Sensors :** Various electronic devices such as television, mobile phones generate various signals and these signals are used as the source for the data.

1. **Databases**

Databases are structured repositories for storing and managing data. They can be relational databases (SQL) or NoSQL databases, and they have several key characteristics :

- **Structured Data :** Databases store structured data in tables with predefined schemas.
- **ACID Transactions :** They support ACID (Atomicity, Consistency, Isolation, Durability) transactions for data integrity.
- **Querying Language :** SQL (Structured Query Language) is commonly used to interact with relational databases.

Examples of Databases

- **MySQL :** A company's customer database that contains tables for customer information, orders, and payment details.
- **MongoDB :** A NoSQL database used for storing unstructured or semi-structured data, such as user profiles in a social media application.

2. **Files**

Files are used to store data in various formats, including text, CSV, Excel, and more. Key characteristics include :

- **Flexibility :** Files can store structured, semi-structured, or unstructured data.
- **Portability :** Files can be easily shared and transferred between systems.
- **Variety of Formats :** Files can be in formats like .txt, .csv, .xlsx, .json, .xml, and many more.

Examples of Files

- **CSV File :** A CSV file containing sales data with columns for date, product, quantity, and price.
- **JSON File :** A JSON file storing configuration settings for a web application, with nested structures for different modules.

3. **APIs (Application Programming Interfaces)**

APIs allow for structured data retrieval and interaction with remote services. Key characteristics include :

- **Structured Data Access :** APIs provide a structured way to request and receive data.
- **Authentication and Authorization :** Many APIs require authentication to access data securely.
- **HTTP Requests :** RESTful APIs often use HTTP methods (GET, POST, PUT, DELETE) for data exchange.

Examples of APIs

- **Twitter API :** Accessing real-time Twitter data through the Twitter API to gather tweets, user profiles, or trending topics.
- **OpenWeatherMap API :** Retrieving weather data for a specific location using the OpenWeatherMap API, which provides current conditions, forecasts, and historical weather data.

4. **Web Scraping**

Web scraping involves extracting data from websites. Key characteristics include :

- **Unstructured Data :** Websites often present data in an unstructured or semi-structured format.
- **HTML Parsing :** Web scraping typically involves parsing HTML or other markup languages.

- **Ethical Considerations :** Web scraping should be performed ethically and with respect to website terms of use.

Examples of Web Scraping

- **News Headlines :** Scraping news headlines from a news website to analyse trends or create a news aggregator.
- **Real Estate Listings :** Extracting property information from real estate websites to analyse housing market trends.

5. **Sensors**

Sensors collect data from the physical world. Key characteristics include :

- **Real-time Data :** Sensors often provide real-time data from the environment.
- **Various Types :** Sensors can measure variables like temperature, pressure, humidity, or GPS location.
- **IoT (Internet of Things) :** Many IoT devices incorporate sensors to collect and transmit data.

Examples of Sensors

- **Temperature Sensor :** A temperature sensor in a smart home system measures room temperature and sends data to a central controller for climate control.
- **GPS Sensor :** A GPS sensor in a smartphone collects location data, enabling mapping and location-based services.

6. **Social Media**

Social media platforms generate vast amounts of data. Key characteristics include :

- **User-Generated Content :** Data is created and shared by users, including text, images, videos, and interactions.
- **Temporal Data :** Social media data often has a timestamp, allowing analysis of trends over time.
- **API Access :** Social media platforms offer APIs for data access, though with limitations.

Examples of Social Media Data

- **Twitter Posts :** Collecting tweets using the Twitter API to analyse sentiment or track public opinions on various topics.
- **Instagram Images :** Gathering images and captions from Instagram posts to understand visual trends or user preferences.

Data science relies on a wide array of data sources, each with its own characteristics and methods for data collection. Understanding these sources and their unique properties is crucial in data science projects. Effective data scientists know how to access, manipulate, and analyse data from various sources to derive valuable insights and drive informed decision-making.

1.3 Data Preprocessing

- Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task. Real-world datasets are generally messy, raw, incomplete, inconsistent, and unusable.

- It can contain manual entry errors, missing values, inconsistent schema, etc. Data Preprocessing is the process of converting raw data into a format that is understandable and usable. It is a crucial step in any Data Science project to carry out an efficient and accurate analysis. It ensures that data quality is consistent before applying any **Machine Learning** or **Data Mining techniques**.

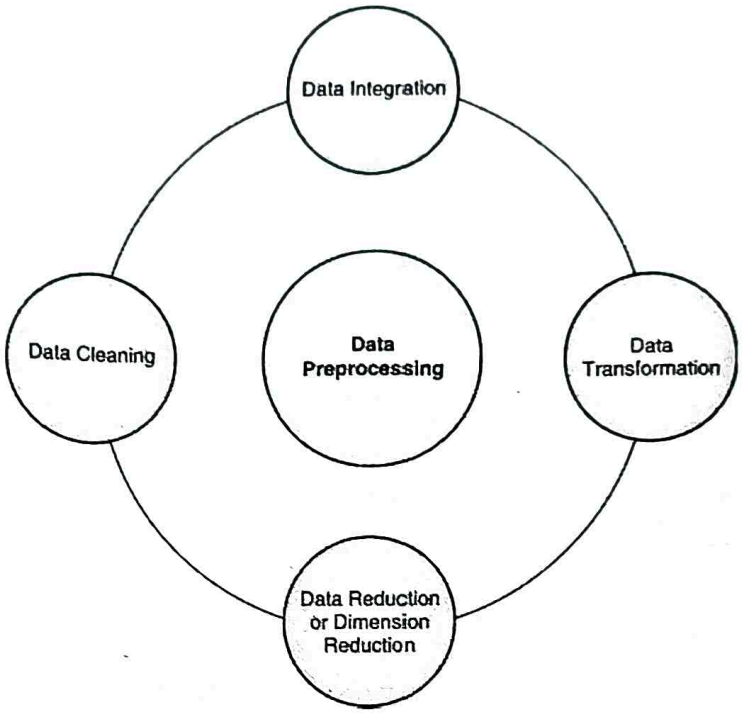


Fig. 1.3.1

Data Preprocessing Techniques :

- Data cleaning
  - Handling missing values
  - Outliers
  - Duplicates
- Data transformation
  - Scaling
  - Normalization
  - Encoding categorical variables
- Feature selection
  - Selecting relevant features/columns
- Data merging
  - Combining multiple datasets

1.3.1 Data Cleaning

This involves identifying and correcting errors or inconsistencies in the data, such as missing values, outliers, and duplicates. Various techniques can be used for data cleaning, such as imputation, removal, and transformation. Data Cleaning uses methods to handle incorrect, incomplete, inconsistent, or missing values.

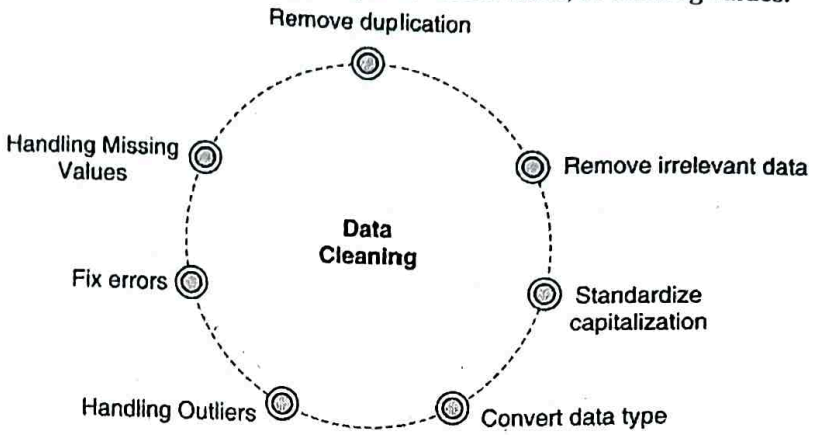


Fig. 1.3.2

1. Handling Missing Values

Input data can contain missing or **NULL** values, which must be handled before applying any Machine Learning or Data Mining techniques. Missing values can be handled by many techniques, such as removing rows/columns containing NULL values and imputing NULL values using mean, mode, regression, etc.

2. Handling Outliers

Outliers are data points that stand out from the rest. They are unusual values that don't follow the overall pattern of your data. Identifying outliers in Data science is important because they can skew results and mislead analyses. Once found, following options can handle outliers :

- **Transform the Data :** Apply log, square root, or other transformations to compress the range of values and reduce outlier impact.
- **Use Robust Statistics :** Choose statistical methods less influenced by outliers like median, Mode, and interquartile range instead of mean and standard deviation.
- **Impute Missing Values :** For outliers caused by missing or erroneous values, you can estimate replacements using the mean, median, or most frequent values.

3. Handling Duplicates

- When you are working with large datasets, working across multiple data sources, or have not implemented any quality checks before adding an entry, your data will likely show duplicated values.
- These duplicated values add redundancy to your data and can make your calculations go wrong. Duplicate serial numbers of products in a dataset will give you a higher count of products than the actual numbers.
- Duplicate email IDs or mobile numbers might cause your communication to look more like spam. We take care of these duplicate records by keeping just one occurrence of any unique observation in our data.

1.3.2 Data Transformation

This involves converting the data into a suitable format for analysis. Common techniques used in data transformation include normalization, standardization, and discretization. Normalization is used to scale the data to a common range, while standardization is used to transform the data to have zero mean and unit variance. Discretization is used to convert continuous data into discrete categories.

1. **Scaling**

   Scaling is useful when you want to compare two different variables on equal grounds. This is especially useful with variables which use distance measures. For example, models that use Euclidean Distance are sensitive to the magnitude of distance, so scaling helps even the weight of all the features. This is important because if one variable is more heavily weighted than the other, it introduces bias into our analysis.

2. **Normalization**

   Normalization is used to scale the data to a common range, while standardization is used to transform the data to have zero mean and unit variance. This involves scaling the data to a common range, such as between 0 and 1 or -1 and 1. Normalization is often used to handle data with different units and scales. Common normalization techniques include min-max normalization, z-score normalization, and decimal scaling.

3. **Encoding Categorical Variables**

   The process of encoding categorical data into numerical data is called "categorical encoding." It involves transforming categorical variables into a numerical format suitable for machine learning models. Encoding categorical data is a process of converting categorical data into integer format so that the data with converted categorical values can be provided to the different models.

1.3.3 Feature Selection

This involves selecting a subset of relevant features from the dataset. Feature selection is often performed to remove irrelevant or redundant features from the dataset. It can be done using various techniques such as correlation analysis, mutual information, and principal component analysis (PCA).Feature selection is a crucial step in the data preprocessing phase of a data science project. It involves selecting the most relevant features or columns from your dataset to build a predictive model or perform analysis.

There are mainly two types of Feature Selection techniques, which are :

- **Supervised Feature Selection Technique :** Supervised Feature selection techniques consider the target variable and can be used for the labelled dataset.
- **Unsupervised Feature Selection Technique :** Unsupervised Feature selection techniques ignore the target variable and can be used for the unlabelled dataset.

1.3.4 Selecting Relevant Features/Columns, Data

Common techniques for selecting relevant features :

- **Univariate Selection :** This method selects features based on univariate statistical tests like chi-square test, ANOVA, or correlation coefficients. You typically set a threshold for p-values or correlation coefficients to select the most significant features.

- **Feature Importance :** Techniques like decision trees, random forests, or gradient boosting machines can provide feature importance scores. You can select the top features based on these scores.
- **Recursive Feature Elimination (RFE) :** RFE recursively removes features and builds a model on the remaining features until the desired number of features is reached. It uses model accuracy as a metric to rank features.
- **Principal Component Analysis (PCA) :** PCA is a dimensionality reduction technique that can also be used for feature selection. It transforms the original features into a lower-dimensional space while retaining most of the variance. You can select the principal components that explain the majority of the variance in the data.
- **SelectKBest :** This is a scikit-learn function that selects the top k features based on statistical tests. It can be used with various scoring functions like chi-square, ANOVA F-value, or mutual information.
- **Forward Selection/Backward Elimination :** These are sequential feature selection techniques where you start with an empty set of features and add/remove features one by one based on their individual performance or contribution to the model.
- **Domain Knowledge :** Sometimes, domain knowledge can guide feature selection. Subject matter experts may identify features that are most relevant to the problem at hand.

1.3.5 Merging : Combining Multiple Datasets

Merging, also known as joining, is a fundamental operation in data science where you combine data from multiple datasets based on a common attribute or key.

There are several types of merges commonly used :

- **Inner Join :** This type of merge returns only the rows where the key exists in both datasets.

  merged_data = pd.merge(left_dataframe, right_dataframe, on='common_key', how='inner')
- **Outer Join :** An outer join returns all rows from both datasets, merging records when the key matches and filling in missing values with NaNs where there is no match.

  merged_data = pd.merge(left_dataframe, right_dataframe, on='common_key', how='outer')
- **Left Join :** A left join returns all rows from the left dataset and matching rows from the right dataset.

  merged_data = pd.merge(left_dataframe, right_dataframe, on='common_key', how='left')
- **Right Join :** A right join is the opposite of a left join; it returns all rows from the right dataset and matching rows from the left dataset.

  merged_data = pd.merge(left_dataframe, right_dataframe, on='common_key', how='right')
- **Concatenation :** Concatenation is used to combine datasets along a particular axis (usually rows). It does not merge based on any key; rather, it just stacks the datasets together.

  concatenated_data = pd.concat([dataframe1, dataframe2], axis=0)

Merging is essential when dealing with relational datasets or when integrating data from multiple sources. It allows you to create a unified dataset for analysis or modeling purposes. However, it is crucial to ensure that the keys used for merging are consistent and that you handle missing values appropriately.

1.4 Data Wrangling and Feature Engineering

- Data Wrangling is referred to as data munging. It is the process of transforming and mapping data from one "raw" data form into another format to make it more appropriate and valuable for various downstream purposes such as analytics. The goal of data wrangling is to assure quality and useful data.
- The process of data wrangling may include further munging, data visualization, data aggregation, training a statistical model, and many other potential uses. Data wrangling typically follows a set of general steps, which begin with extracting the raw data from the data source, "munging" the raw data (e.g., sorting) or parsing the data into predefined data structures, and finally depositing the resulting content into a data sink for storage and future use.

1.4.1 Data Wrangling Techniques

1. **Reshaping :** Data Reshaping is about changing the way data is organized into rows and columns. It is easy to extract data from the rows and columns of a data frame but there are situations when we need the data frame in a format that is different from format in which we received it. There are many functions to split, merge and change the rows to columns and vice-versa in a data frame.
2. **Pivoting :** Pivoting can be used to restructure a DataFrame, such that the rows can be converted into additional column headings where a chosen column is displayed in these new column headings. Pivoting aids data understanding and presentation. Pivoting is where you take a long data file (lots of rows, few columns) and make it wider. Or where you take a wide data file (lots of columns, few rows) and make it longer.
3. **Aggregating :** Data aggregation is the process of collecting data to present it in summary form. This information is then used to conduct statistical analysis and can also help company executives make more informed decisions about marketing strategies, price settings, and structuring operations, among other things.

   Aggregating data is a useful tool for data exploration. Aggregation is sometimes done to allow for analysis to be completed at a higher level of the data. For example, if an analysis of the size of school districts in a region is to be done, the number of students from the schools with in the district is summed (aggregated.)

1.4.2 Feature Engineering

Feature Engineering is the process of creating new features or transforming existing features to improve the performance of a machine-learning model. It involves selecting relevant information from raw data and transforming it into a format that can be easily understood by a model. The goal is to improve model accuracy by providing more meaningful and relevant information.

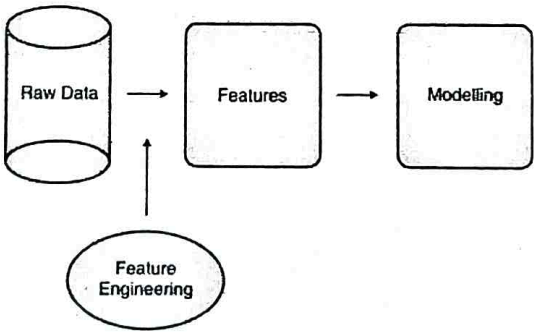


Fig. 1.4.1

1. **Creating New Features**

Feature Creation is the process of generating new features based on domain knowledge or by observing patterns in the data. It is a form of feature engineering that can significantly improve the performance of a machine-learning model.

Types of Feature Creation

- **Domain-Specific :** Creating new features based on domain knowledge, such as creating features based on business rules or industry standards.
- **Data-Driven :** Creating new features by observing patterns in the data, such as calculating aggregations or creating interaction features.
- **Synthetic :** Generating new features by combining existing features or synthesizing new data points.

2. **Handling Time-Series Data**

Handling time-series data in feature engineering involves creating additional features that capture relevant information from the temporal nature of the data.

Following are several techniques used in feature engineering for time-series data :

- **Lag Features :** Lag features involve incorporating past values of the target variable or other relevant variables into the dataset. Lag features allow the model to capture temporal dependencies and patterns in the data.
- **Rolling Window Statistics :** Rolling window statistics involve calculating summary statistics (e.g.,mean, median, standard deviation) over a rolling window of past observations.
- **Exponential Moving Average (EMA) :** EMA is a weighted moving average that gives more weight to recent observations and less weight to older observations.EMA features capture recent trends and smooth out noise in the time series.
- **Seasonal Features :** Seasonal features capture periodic patterns and seasonality in the time series data. Seasonal features help the model capture recurring patterns and improve forecast accuracy.
- **Time-based Features :** Time-based features capture temporal characteristics of the data, such as the time of day, day of the week, month, or year.
- **Window Statistics :** Window statistics involve calculating summary statistics over fixed windows of time.

1.4.3 Dummification

In data science, "dummification" typically refers to the process of converting categorical variables into a set of binary variables, also known as dummy variables. This process is essential because many machine learning algorithms cannot directly handle categorical data; they require numerical input.

How dummification works?

- **Identify Categorical Variables :** These are variables that represent categories or labels rather than numerical values. For example, "gender" with categories "male" and "female", or "color" with categories "red", "green", and "blue".

- **Encode Categorical Variables :** Each categorical variable is encoded into a set of binary variables, with one variable for each category. For example, for the "gender" variable, we might create two binary variables: "is_male" and "is_female". If a data point belongs to a particular category, the corresponding binary variable is set to 1; otherwise, it is set to 0.
- **Drop One Category :** To avoid multicollinearity (a situation where one predictor variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy), one category is typically dropped. This is because the information from the dropped category is redundant once we know the values of all the other binary variables.
- **Use Dummy Variables in Analysis :** The resulting dummy variables are used as input features in machine learning algorithms or statistical analysis.

1.4.4 Feature Scaling

- Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units.
- If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values. Feature scaling is a preprocessing step in machine learning that involves transforming the features of a dataset to have similar scales or ranges.
- This step is essential for algorithms that are sensitive to the scale of input features, such as gradient descent-based optimization algorithms and distance-based algorithms like k-nearest neighbours

1. Standardization

- This method of scaling is basically based on the central tendencies and variance of the data. First, we should calculate the mean and standard deviation of the data we would like to normalize. Then we are supposed to subtract the mean value from each entry and then divide the result by the standard deviation.
- This helps us achieve a normal distribution (if it is already normal but skewed) of the data with a mean equal to zero and a standard deviation equal to 1.

$$X_{scaled} = \frac{X_i - X_{mean}}{0}$$

2. Normalization

This method is more or less the same as the previous method but here instead of the minimum value, we subtract each entry by the mean value of the whole data and then divide the results by the difference between the minimum and the maximum value. This method scales the features to a fixed range, usually between 0 and 1.

$$X_{scaled} = \frac{X_i - X_{mean}}{X_{max} - X_{min}}$$

1.5 Introduction to Tools and Libraries

- In the dynamic field of Data Science, practitioners utilize a powerful array of tools and libraries to extract meaningful insights from data. Offering an overview of some widely used libraries and technologies in Data Science, the chapter delves into key tools like Pandas, NumPy, and Sci-kit Learn, exploring their functionalities and their contribution to the robust toolkit driving the data-driven revolution.
- Empowered by a comprehensive toolbox, the field relies on various tools and libraries. Serving as a gateway to essential technologies, it guides you through the fundamentals crucial for data-driven exploration, analysis, and modelling. Whether handling datasets or implementing machine learning algorithms, understanding these tools is pivotal for any aspiring data scientist.

1.5.1 NumPy

- NumPy, short for Numerical Python, serves as Python's essential package for numerical computation. At its core, it boasts a potent N-dimensional array object. GitHub hosts approximately 18,000 comments, reflecting a vibrant community featuring 700 active contributors.
- This versatile package caters to a broad array of tasks in array processing, supplying high-performance multidimensional objects labeled as arrays, along with accompanying tools for seamless interaction. To counter slowness, NumPy introduces multidimensional arrays and enhances operational efficiency through functions and operators tailored for these arrays.

Features

- Rapid, precompiled functions for numerical operations.
- Embraces array-oriented computing, enhancing efficiency.
- Adopts an object-oriented approach.
- Executes compact and swift computations through vectorization.

Pros

- NumPy is highly optimized, making it an excellent tool for data scientists to perform scientific computations with numeric arrays.
- It is efficient for use in popular packages like sci-kit-learn and TensorFlow, where NumPy arrays serve as input.
- The ndarray object in NumPy provides numerous supporting functions, such as elementwise addition and multiplication, as well as the computation of the Kronecker product, which are not available with Python lists.

Cons

- NumPy supports the use of NaN (Not a Number), which, while supported by NumPy, is not as widely supported by other packages, making it challenging to interpret and work with for users.
- Continuous allocation of memory is required, and when memory is allocated contiguously, the insertion and deletion of memory become costly due to the need for shifting.

Applications

- Widely utilized in data analysis.
- Establishes robust N-dimensional arrays.
- Serves as the foundational framework for other libraries like SciPy and scikit-learn.
- Proves to be a viable alternative to MATLAB, especially when integrated with SciPy and matplotlib.

For example, consider the following code snippet demonstrating NumPy's functionality :

```
import numpy as np
# Create a 2D array
array_2d = np.array([[1, 2, 3], [4, 5, 6]])

# Perform element-wise multiplication
result_array = array_2d * 2

print("Original Array:")
print(array_2d)

print("\nResult after multiplication:")
print(result_array)
```

Additionally, NumPy's extensive documentation provides comprehensive insights into its usage, further aiding developers in harnessing its capabilities effectively.

1.5.2 Pandas

Pandas, a crucial component in the data science life cycle, stands as the most widely embraced Python library for data science. Alongside NumPy and Matplotlib, it constitutes a foundational toolset. With approximately 17,00 comments on GitHub and an engaged community featuring 1,200 contributors, Pandas plays a pivotal role in data analysis and cleaning.

Features

- **Eloquent Syntax and Rich Functionalities :** Pandas offers an eloquent syntax and diverse functionalities, providing the flexibility to handle missing data seamlessly.
- **Custom Function Creation :** Users can devise and execute their own functions across a series of data.
- **High-Level Abstraction :** It offers high-level abstraction, making it conducive for efficient data manipulation.
- **Data Structures and Manipulation Tools :** Pandas comprises advanced data structures and tools for data manipulation.

Pros

- Python offers a straightforward representation of data by condensing various data types into a simple data frame.

- This simplification enhances our ability to visualize and comprehend data more efficiently.
- Additionally, the Pandas library provides powerful features, encompassing all commands necessary for data manipulation, including filtering, grouping, and segmenting.
- Furthermore, Pandas excels in efficiently handling large datasets, a core purpose for its development.

Cons

- While Pandas boasts powerful capabilities, it does come with a steep learning curve.
- Users new to Pandas may require some time to familiarize themselves with the library's operational intricacies.
- Another drawback lies in the imperfect documentation of Pandas, which may be attributed to the extensive capabilities of the library.
- However, users familiar with specific applications can find multiple use cases to reference.
- Pandas faces a significant limitation in its compatibility with 3D matrices, making it less suitable for applications requiring the processing of multi-dimensional arrays.
- In such cases, alternative packages like NumPy are preferred for their better handling of 3D matrices.

Applications

- **General Data Wrangling and Cleaning :** Pandas is extensively used for general data wrangling and cleaning tasks.
- **ETL Jobs (Extract, Transform, Load) :** Its robust support for loading CSV files into data frames makes it ideal for ETL jobs involving data transformation and storage.
- **Wide Industry Adoption :** Pandas finds application in diverse academic and commercial domains, including statistics, finance, and neuroscience.
- **Time-Series Functionality :** It specializes in time-series operations, offering features like date range generation, moving window analysis, linear regression, and date shifting.

Examples

Code Snippet for Handling Missing Data :

```
import pandas as pd
# Creating a DataFrame with missing values
df = pd.DataFrame({'A': [1, 2, None], 'B': [4, None, 6]})

# Dropping rows with any missing values
df_cleaned = df.dropna()
```

CSV File Loading :

```
import pandas as pd
# Loading a CSV file into a Pandas DataFrame
df_csv = pd.read_csv('your_data.csv')
```

Time-Series Operation :

```
import pandas as pd
# Creating a time series with date range
date_range = pd.date_range('2023-01-01', '2023-01-10')
```

By integrating Pandas into the data science workflow, users gain a powerful tool for handling data effectively, from basic cleaning to advanced analytical tasks across various domains.

1.5.3 Matplotlib

Matplotlib, a Python plotting library, boasts powerful and aesthetically pleasing visualizations. Widely embraced for data visualization, it has a robust community of approximately 700 contributors and garners around 26,000 comments on GitHub.

Features

- Serves as a cost-free, open-source alternative to MATLAB.
- Offers extensive support with numerous backends and output types, ensuring compatibility across operating systems and output formats.
- Facilitates the integration of plots into applications through its object-oriented API.
- Pandas can act as a wrapper around the MATLAB API, enhancing code clarity.
- Demonstrates low memory consumption and superior runtime behaviour.

Pros

- Matplotlib, built on NumPy, provides a relatively straightforward entry point for beginners.
- It offers an intuitive experience for individuals familiar with graph plotting tools like MATLAB.
- Users can achieve a high level of customization through code.

Cons

- Matplotlib does not support interactivities for all visualizations.
- Achieving visually appealing results can be challenging due to its low-level interface.
- Plotting non-basic graphs in Matplotlib may become complex, leading to code-heavy implementations.

Applications

- Conduct correlation analysis of variables.
- Visualize 95 percent confidence intervals of models.
- Detect outliers through scatter plots.
- Gain immediate insights into data distribution.

```
import matplotlib.pyplot as plt
import numpy as np
# Generate sample data
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Create a simple plot
```

```
plt.plot(x, y)
plt.title('Sine Wave')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```

1.5.4 SciPy

SciPy, an open-source Python library for data science, is widely utilized for advanced computations. With approximately 19,000 comments on GitHub and an active community boasting around 600 contributors, it extends NumPy, offering user-friendly and efficient routines for scientific calculations.

Features

- Utilizes NumPy extension for Python, providing a collection of algorithms and functions.
- Offers high-level commands for data manipulation and visualization.
- Employs the SciPy ndimage submodule for multidimensional image processing.
- Includes built-in functions for solving differential equations.

Pros

- SciPy offers classes designed for efficient visualization and data manipulation.
- It demonstrates enhanced cross-functionality with other Python libraries.
- The inclusion of parallel programming options for specific database and web routines.
- SciPy is known for its quick and straightforward learning curve.

Cons

- The use of NaN (Not a Number) can pose challenges.
- While supported by SciPy and NumPy, it may not be as widely supported by other packages, making it difficult to interpret and work with for some users.
- SciPy can be complex for individuals without a background in mathematics.
- While it is a powerful tool for scientific and mathematical exploration, users without fundamental knowledge in these areas may find it challenging to leverage effectively.

Applications

- Multidimensional image operations.
- Solving differential equations and Fourier transforms.
- Optimization algorithms.
- Linear algebra.
- SciPy's significance in data science is evident through its vast GitHub comments and the active involvement of a diverse contributor community. It excels in various applications, ranging from multidimensional image operations to complex mathematical problem-solving. Below are a few examples to illustrate its usage :

Utilizing SciPy for Multidimensional Image Processing

```
import numpy as np
from scipy import ndimage

# Create a sample 3D array
image_array = np.random.rand(3, 3, 3)

# Apply a Gaussian filter for smoothing
smoothed_image = ndimage.gaussian_filter(image_array, sigma=1)

# Display the original and smoothed images
print("Original Image:")
print(image_array)
print("\nSmoothed Image:")
print(smoothed_image)
```

Solving a Differential Equation with SciPy

```
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# Define a simple differential equation
def simple_differential_equation(t, y):
    return -2 * y + np.sin(t)

# Set initial conditions
initial_conditions = [0]

# Define the time span
time_span = (0, 10)

# Solve the differential equation using SciPy
solution = solve_ivp(simple_differential_equation, time_span, initial_conditions, t_eval=np.linspace(0, 10, 100))

# Plot the solution
plt.plot(solution.t, solution.y[0])
plt.xlabel('Time')
plt.ylabel('Solution')
plt.title('Solving a Differential Equation with SciPy')
plt.show()
```

1.5.5 Scikit-Learn

Scikit-learn, an open-source machine learning library, is extensively used for predictive data analysis. Built upon tools like NumPy, SciPy, and Matplotlib, it offers a versatile set of features and applications.

Features

- Supporting various predictive data analytics applications, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.
- Scikit-learn provides a comprehensive suite of algorithms.
- These encompass logistic regression, decision trees, bagging, boosting, random forest, XGBoost, and Support Vector Machine (SVM), complemented by a diverse array of classification metrics.

Pros

- With a licensing structure that allows free usage and minimal legal restrictions.
- Scikit-learn stands out as a widely adopted package in the realm of machine learning.
- It serves as an excellent toolkit for modeling.

Cons

- While Scikit-learn serves as a robust foundational package
- it may not be the preferred library for in-depth machine learning tasks.
- It encounters challenges in scaling to large datasets and can present confusion for data scientists working with NumPy and Pandas data frames, with less optimal compatibility with the latter.

Applications

- Scikit-learn finds application in diverse fields, such as spam detection, image recognition, drug response analysis, stock price prediction, customer segmentation, and experimental grouping.

Examples

Logistic Regression in Scikit-learn :

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

Random Forest Classifier :

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
```

1.5.6 Seaborn

Seaborn, a high-level interface built on top of matplotlib, facilitates the creation of informative statistical graphs and visually appealing data visualizations.

Features

- Seaborn includes various plots like relational, categorical, distribution, regression, matrix, and multi-plot grids.
- Provides themes for styling matplotlib visualizations.

- Capable of plotting linear regression models and statistical time series, working seamlessly with NumPy and Pandas data structures.
- Efficient at visualizing both univariate and bivariate data.

Pros

- Significantly faster as a visualization tool; processes entire datasets with minimal effort.
- Offers interactive and informative representations for quick data visualization.

Cons

- Lack of true interactivity in visualizations.
- Limited customization options restricted to seaborn's predefined styles.

Applications

- Used to aesthetically visualize data.
- Employed in various Integrated Development Environments (IDEs).

Examples

Coding Example for Plotting a Relational Plot :

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Create a DataFrame (assuming df is your dataset)
sns.relplot(x="x_column", y="y_column", data=df, kind="scatter")

# Show the plot
plt.show()
```

Example of Applying a Seaborn Theme to a Matplotlib Visualization

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a basic matplotlib plot
plt.plot([1, 2, 3, 4], [10, 15, 25, 30])

# Apply seaborn theme
sns.set_theme()
# Show the updated plot
plt.show()
```

1.5.7 TensorFlow

TensorFlow stands as a preeminent open-source framework, commanding a pivotal role in the realms of machine learning and data science. Developed by the Google Brain team, it is revered for its flexibility and efficiency in crafting, training, and deploying machine learning models.

Features

- **Versatile Functionality :** TensorFlow seamlessly handles diverse tasks, encompassing data preparation, model construction, deployment, and the intricate field of MLOps.
- **User-Friendly Design :** Boasting an intuitive high-level Keras API, TensorFlow simplifies the model-building process, rendering it accessible even to those new to the world of machine learning.
- **Deployment Flexibility :** Its deployment capabilities span the web, mobile devices, edge computing, and traditional servers, providing a wide-ranging solution for various application scenarios.

Pros

- **Efficient Model Building :** TensorFlow's high-level Keras API ensures that constructing models is an intuitive and streamlined process, enhancing productivity in machine learning development.
- **Robust Production :** With a focus on robust machine learning production, TensorFlow assures reliability in deploying models at scale.
- **Scalability and Debugging :** TensorFlow facilitates easy debugging, offers extensive scalable architectural support, and excels in library management, contributing to a smooth development process.

Cons

- **Windows Limitations :** TensorFlow encounters limitations when operating on Windows, a factor to consider when choosing a development environment.
- **Performance Comparison :** In comparison to alternative frameworks, TensorFlow may exhibit relatively slower performance and occasional inconsistencies.
- **Architectural Constraints :** TensorFlow's architectural design restricts its functionality to model execution, omitting the ability to directly train models within the framework.

Applications

- **Airbnb :** TensorFlow is harnessed by Airbnb for tasks such as image classification and object detection, showcasing its applicability in diverse domains.
- **Airbus :** Airbus leverages TensorFlow to extract valuable insights from satellite images, providing clients with informed and data-driven perspectives.
- **GE Healthcare :** GE employs TensorFlow in the medical field, specifically for identifying and analysing brain anatomy in MRI scans, demonstrating its importance in healthcare applications.

```
Building a neural network using TensorFlow
import tensorflow as tf

model = tf.keras.Sequential([
  tf.keras.layers.Dense(128, activation='relu', input_shape=(784,)),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

# Example 2: Loading a pre-trained model
pretrained_model = tf.keras.applications.MobileNetV2(weights='imagenet')

# Example 3: Deploying a model for image classification
image = load_and_preprocess_image('path/to/image.jpg')
predictions = pretrained_model.predict(image)
```

1.5.8 Keras

Keras serves as a user-friendly interface for Artificial Neural Networks (ANNs) and functions as a bridge to the powerful TensorFlow library. Optimized to minimize cognitive load, Keras facilitates seamless implementation of Deep Learning with minimal user intervention.

Features

- **Simplicity and Flexibility :** Keras is designed to be simple, flexible, and powerful, allowing users to conduct experiments swiftly and efficiently in Deep Learning.
- **Integration with TensorFlow 2 :** Built on top of TensorFlow 2, Keras seamlessly integrates with the library and can seamlessly scale to meet the demands of large-scale production settings.
- **Versatile Deployment :** Keras offers the flexibility of deployment across various platforms, including websites, Android and iOS phones, embedded devices, and as a web API.

Pros

- **End-to-End Solutions :** Tightly integrated with TensorFlow 2, Keras provides comprehensive coverage for end-to-end machine learning solutions, simplifying the development process.
- **Ease of Use :**
- Known for its user-friendly design, Keras is recognized as one of the most accessible entry points into the field of deep learning.
- **Pre-trained Models and Hardware Support :** Keras comes with pre-trained models and robust support for multiple GPUs and TPUs, enhancing its capability for various machine learning tasks.

Cons

- **Low-Level API Challenges :** Users may encounter low-level backend issues while working with Keras, particularly when attempting operations beyond its original design scope.

- **Areas for Improvement** : Keras could enhance certain features like data pre-processing, basic machine learning algorithms, dynamic chart creation, etc., to further augment its utility.
- **Trade-off Between Speed and User-Friendliness** : The emphasis on user-friendliness in Keras may, in some cases, result in a trade-off with speed, where certain applications sacrifice speed for enhanced user accessibility.

Applications

- **Image Recognition** : Keras finds significant utility in image recognition applications, leveraging pre-trained models such as Xception, ResNet, MobileNet, and ImageNet to enhance accuracy and efficiency. These attributes collectively position Keras as a powerful yet user-friendly tool for delving into the complexities of Artificial Neural Networks, making it particularly valuable in diverse applications, from image recognition to large-scale production environments.

Simple coding example using Keras to create a basic neural network for image classification

```
# Import necessary libraries
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Load and preprocess your dataset (replace this with your own dataset loading and preprocessing code)
# X_train, y_train = load_and_preprocess_dataset('path/to/dataset')

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
# Build a simple neural network using Keras
model = models.Sequential()
model.add(layers.Flatten(input_shape=(your_image_width, your_image_height, number_of_channels)))  # Adjust input shape accordingly
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(number_of_classes, activation='softmax'))

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',  # Use 'categorical_crossentropy' if labels are one-hot encoded
              metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))

# Evaluate the model on the test set
```

```
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {test_acc}')

# Plot training history
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

1.5.9 Pytorch

PyTorch, a machine learning framework, is based on Meta's torch library, streamlining the transition from prototyping and research to production and deployment.

Features

- **Production Readiness :** PyTorch is meticulously designed for production use, offering convenient tools for deploying models in a cloud-agnostic manner.
- **Comprehensive Support :** It provides support for essential features like metrics, logging, multi-model serving, and the creation of RESTful endpoints.
- **Distributed Training :** Training models in a distributed fashion is a strength of PyTorch, and it boasts a robust ecosystem to support such scenarios.

Pros

- **Versatile Frontend :** Despite having a C++ frontend, PyTorch also features a Pythonic frontend, allowing users to extend functionalities as needed.
- **User-Friendly and Supportive Community :** PyTorch is renowned for its ease of learning, supported by a vibrant and strong community. Debugging is simplified in PyTorch workflows.
- **GPU and CPU Scalability :** It provides support for both CPU and GPU, demonstrating excellent scalability in diverse computing environments.

Cons

- **Relatively New Framework :** PyTorch, introduced in 2016, is relatively new compared to other frameworks, resulting in lower awareness within the developer community.
- **Monitoring and Dashboard Limitations :** Unlike TensorFlow, PyTorch lacks a built-in monitoring and dashboard tool similar to TensorBoard, which can be a drawback for some users.
- **Smaller Developer Community :** The developer community for PyTorch is comparatively smaller when compared to other widely adopted machine learning frameworks.

Applications

- **Computer Vision** : PyTorch finds extensive applications in computer vision tasks, leveraging its capabilities in image recognition and analysis.
- **Natural Language Processing** : In the realm of natural language processing, PyTorch proves to be a valuable tool for tasks such as text analysis and language modeling.
- **Reinforcement Learning** : PyTorch is utilized in reinforcement learning applications, showcasing its adaptability in training agents for decision-making scenarios.

These attributes collectively position PyTorch as a powerful framework with a focus on production readiness, flexibility, and ease of use, making it particularly valuable across various domains, from computer vision to natural language processing and reinforcement learning.

Simple example of building a neural network using PyTorch for a basic classification task

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torch.utils.data import DataLoader
from torchvision import datasets, transforms

# Define a simple neural network
class SimpleNet(nn.Module):
  def __init__(self):
    super(SimpleNet, self).__init__()
    self.fc1 = nn.Linear(28 * 28, 128)
    self.fc2 = nn.Linear(128, 10)

  def forward(self, x):
    x = x.view(-1, 28 * 28)
    x = F.relu(self.fc1(x))
    x = self.fc2(x)
    return F.log_softmax(x, dim=1)

# Set device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load and preprocess the MNIST dataset
transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,), (0.5,))])
train_dataset = datasets.MNIST('./data', train=True, download=True, transform=transform)
test_dataset = datasets.MNIST('./data', train=False, download=True, transform=transform)
```

```
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)

# Instantiate the model, loss function, and optimizer
model = SimpleNet().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Training loop
epochs = 5
for epoch in range(epochs):
    running_loss = 0.0
    for data in train_loader:
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)

        optimizer.zero_grad()

        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    print(f'Epoch {epoch+1}/{epochs}, Loss: {running_loss / len(train_loader)}')
# Testing loop
correct = 0
total = 0
with torch.no_grad():
    for data in test_loader:
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = model(inputs)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
print(f'Accuracy on the test set: {100 * correct / total}%')
```

Review Questions

Q. 1 What is Data Science?

Q. 2 What is the scope of Data Science?

Q. 3 Give some common applications of Data Science.

Q. 4 Explain various domains of Data Science.

Q. 5 How does Data Science differ from Business Intelligence (BI)?

Q. 6 What is the difference between Data Science and Artificial Intelligence (AI)?

Q. 7 How is Data Science related to Machine Learning (ML)?

Q. 8 How does Data Science relate to Data Warehousing and Data Mining (DW-DM)?

Q. 9 What is the fundamental definition of data, and how is it categorized based on its format?

Q. 10 Explain the concept of the data, information, knowledge pyramid. How does data transition into knowledge and wisdom?

Q. 11 Differentiate between quantitative and qualitative data.How are they utilized in data analysis?

Q. 12 How are formats like JSON and XML used to represent semi-structured data?

Q. 13 What are the key characteristics of databases, and how do they differ from other data sources?

Q. 14 Provide examples of data stored in different file formats and their use cases.

Q. 15 Give examples of APIs and their applications in accessing real-time data.

Q. 16 List and explain any 2 Libraries commonly used in Data Science.

Q. 17 Explain NumPy with help of an example.

Q. 18 Explain Pandas with help of an example.

Q. 19 Explain SciPy with help of an example.

Q. 20 Explain TensorFlow with help of an example.

Q. 21 Explain Scikit-Learn with help of an example.

Q. 22 Explain Matplotlib with help of an example.

Q. 23 Explain Seaborn with help of an example.

Q. 24 Explain Keras with help of an example.

Q. 25 Explain Pytorch with help of an example.

❑❑❑

Data Analysis and Machine Learning

Syllabus

Exploratory Data Analysis (EDA) : Data visualization techniques: histograms, scatter plots, box plots, etc., Descriptive statistics: mean, median, mode, standard deviation, etc., Hypothesis testing: t-tests, chi-square tests, ANOVA, etc.

Introduction to Machine Learning : Supervised learning: classification and regression, Unsupervised learning: clustering and dimensionality reduction, Bias-variance tradeoff, underfitting, and overfitting

Regression Analysis : Simple linear regression, Multiple linear regression, Stepwise regression, Logistic regression for classification

Model Evaluation and Selection : Techniques for evaluating model performance: accuracy, precision, recall, F1-score, Confusion matrix and ROC curve analysis, Cross-validation: k-fold cross-validation, stratified cross-validation, Hyperparameter tuning and model selection

Machine Learning Algorithms : Decision Trees and Random Forests, Support Vector Machines (SVM), Artificial Neural Networks (ANN), Ensemble Learning: Boosting and Bagging, K-Nearest Neighbors (K-NN), Gradient Descent for optimization

2.1 Exploratory Data Analysis (EDA)

Definition :

- The goal of Exploratory Data Analysis (EDA), which typically makes use of statistical and visual methodologies, is to analyze datasets in a way that highlights their salient aspects. Before advancing to more complex statistical analyses or modelling, the main goal of EDA is to gain a deeper understanding, identify patterns, find anomalies, and appreciate the inherent structure of the data.
- EDA includes the process of carefully analyzing data to describe its properties, distributions, correlations between variables, and possible problems, such as missing values or outliers. This methodical investigation supports the generation of hypotheses, well-informed decision-making, and directs further analysis to glean insightful information from the dataset.

2.1.1 Types of EDA

Depending on the number of columns we are analyzing we can divide EDA into two types. EDA, or Exploratory Data Analysis, refers back to the method of analyzing and analyzing information units to uncover styles, pick out relationships, and gain insights. There are various sorts of EDA strategies that can be hired relying on the nature of the records and the desires of the evaluation. Here are some not unusual kinds of EDA :

1. **Statistical Analysis :** Exploring data characteristics numerically to understand distributions, central tendencies, and variability.
2. **Data Visualization Techniques :** Visualizing data to glean insights, discover patterns, and identify trends through plots, graphs, and charts.
3. **Pattern Recognition :** Unveiling hidden structures and trends within the data, often utilizing techniques like clustering, dimensionality reduction, and anomaly detection.
4. **Statistical Testing :** Using various statistical tests to validate hypotheses, compare groups, and make inferences about the data.
5. **Correlation Analysis :** Assessing relationships between variables to understand how they change together or influence each other.
6. **Data Cleaning and Preprocessing :** Preparing data by handling missing values, outliers, and transforming variables to make it suitable for analysis.

2.1.2 Data Visualization Techniques

Data Visualization involves analyzing data through the creation of graphs or maps, facilitating a more accessible comprehension of trends or patterns within the data. The techniques are as follow us :

1. Histograms
2. Scatter plots
3. Box plots
4. Handling Outliers
5. Removing Outliers

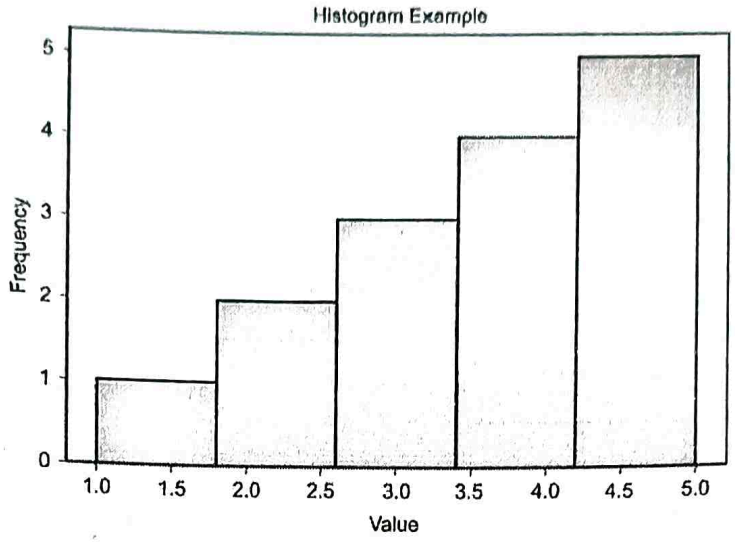
Let's see some commonly used graphs –

1. Histogram

A histogram is applicable for both univariate and bivariate analyses. The **hist** function in Matplotlib is used to create histograms. Here's a basic example along with some explanation :

```
import matplotlib.pyplot as plt

# Sample data
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

# Plotting histogram
plt.hist(data, bins=5, color='blue', edgecolor='black')

# Adding labels and title
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram Example')

# Display the plot
plt.show()
```

Output :



Explanation :

- **plt.hist(data, bins=5, color='blue', edgecolor='black') :** This line creates the histogram. data is your dataset, and bins is the number of bins or classes in the histogram. You can adjust the color and edge color as per your preference.
- **plt.xlabel('Value') and plt.ylabel('Frequency') :** These lines add labels to the x-axis and y-axis, respectively.
- **plt.title('Histogram Example'):** This line adds a title to the plot.
- **plt.show():** This function displays the plot.

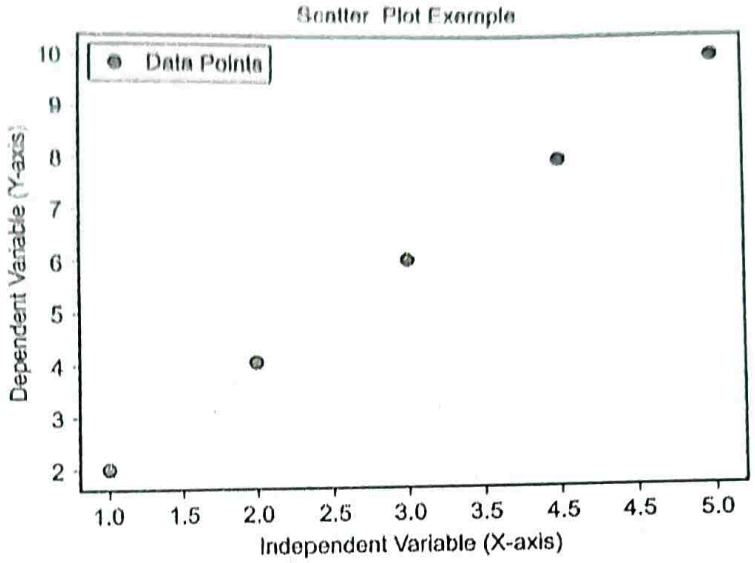
2. Scatter Plots

Scatter plots depict the correlation between two variables within a dataset by plotting data points on a two-dimensional plane or Cartesian system. The X-axis represents the independent variable or attribute, while the Y-axis represents the dependent variable.

Example :

```
: import matplotlib.pyplot as plt

# Sample data
x_values = [1, 2, 3, 4, 5]
y_values = [2, 4, 6, 8, 10]

# Plotting scatter plot
plt.scatter(x_values, y_values, color='blue', marker='o', label='Data Points')

# Adding labels and title
plt.xlabel('Independent Variable (X-axis)')
plt.ylabel('Dependent Variable (Y-axis)')
plt.title('Scatter Plot Example')

# Adding a legend (optional)
plt.legend()

# Display the plot
plt.show()
```

Output :



Explanation :

- **plt.scatter(x_values, y_values, color='blue', marker='o', label='Data Points') :** This line creates a scatter plot. **x_values** and **y_values** are your datasets for the X and Y axes. You can customize the color, marker style, and label as needed.
- **plt.xlabel('Independent Variable (X-axis)') and plt.ylabel('Dependent Variable (Y-axis)') :** These lines add labels to the x-axis and y-axis, respectively.
- **plt.title('Scatter Plot Example') :** This line adds a title to the plot.
- **plt.legend() :** This line adds a legend if you have labeled your data points.
- **plt.show() :** This function displays the plot.

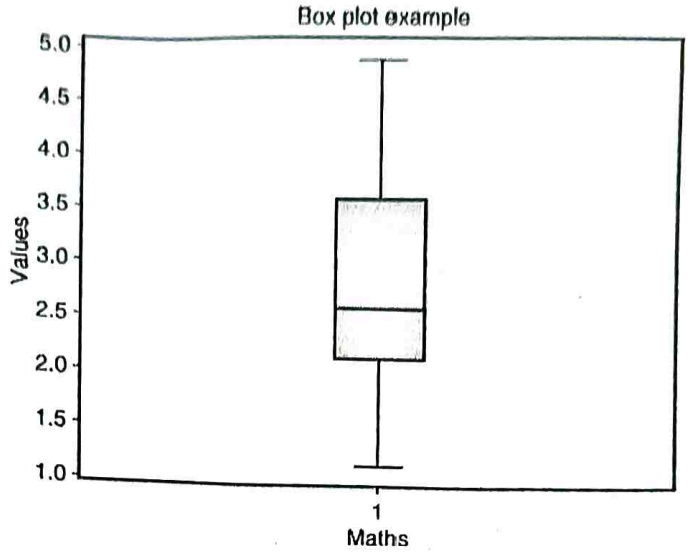
3. Box Plot

It is versatile and applicable for both univariate and bivariate analyses.

i) Matplotlib Example :

```
import matplotlib.pyplot as plt

# Sample data
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

# Creating a box plot
plt.boxplot(data)

# Adding labels and title
plt.xlabel('Variable')
plt.ylabel('Values')
plt.title('Box Plot Example')

# Display the plot
plt.show()
```

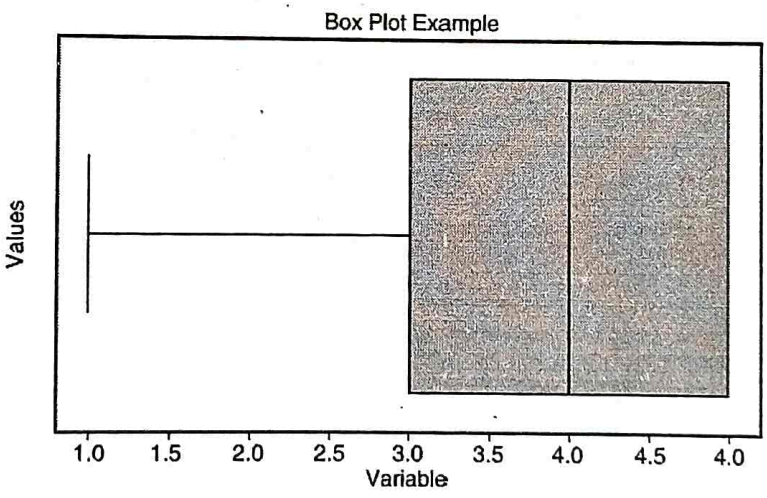
OUTPUT :



ii) Seaborn Example :

```
: import seaborn as sns
import matplotlib.pyplot as plt

# Sample data
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

# Creating a box plot using Seaborn
sns.boxplot(x=data)

# Adding labels and title
plt.xlabel('Variable')
plt.ylabel('Values')
plt.title('Box Plot Example')

# Display the plot
plt.show()
```

OUTPUT



Explanation :

- **plt.boxplot(data) :** This line creates a box plot using Matplotlib. Replace 'data' with your actual dataset.
- **sns.boxplot(x=data) :** This line creates a box plot using Seaborn. Again, replace 'data' with your dataset.
- **plt.xlabel('Variable') and plt.ylabel('Values') :** These lines add labels to the x-axis and y-axis, respectively.
- **plt.title('Box Plot Example') :** This line adds a title to the plot.
- **plt.show() :** This function displays the plot.

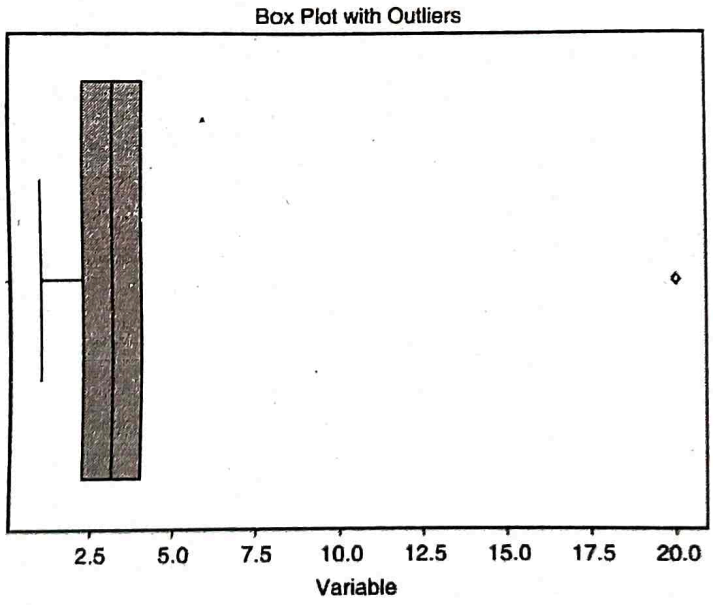
4. Handling Outliers

Handling outliers involves managing data points that significantly deviate from the majority of the dataset, commonly known as "normal" objects. Outliers may arise due to measurement or execution errors, and the process of identifying them is referred to as outlier mining. Various methods exist for outlier detection, and the process of removing these outliers from a pandas DataFrame is akin to eliminating a data item from the dataset.

Identify and Remove Outliers Example :

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Sample DataFrame with outliers
data = {'Values': [1, 2, 2, 3, 3, 3, 4, 4, 4, 20]}
df = pd.DataFrame(data)

# Create a box plot
plt.figure(figsize=(8, 6))
sns.boxplot(x=df['Values'])

# Add labels and title
plt.xlabel('Variable')
plt.title('Box Plot with Outliers')

# Display the plot
plt.show()
```

OUTPUT



Explanation :

- In this example, the box plot allows you to visualize the distribution of the data, including the presence of outliers. Outliers are typically shown as individual points beyond the "whiskers" of the box plot.
- If you want to handle outliers by removing them, you can use the approach mentioned in the previous response. After removing outliers, you can create another visualization to observe the updated distribution.

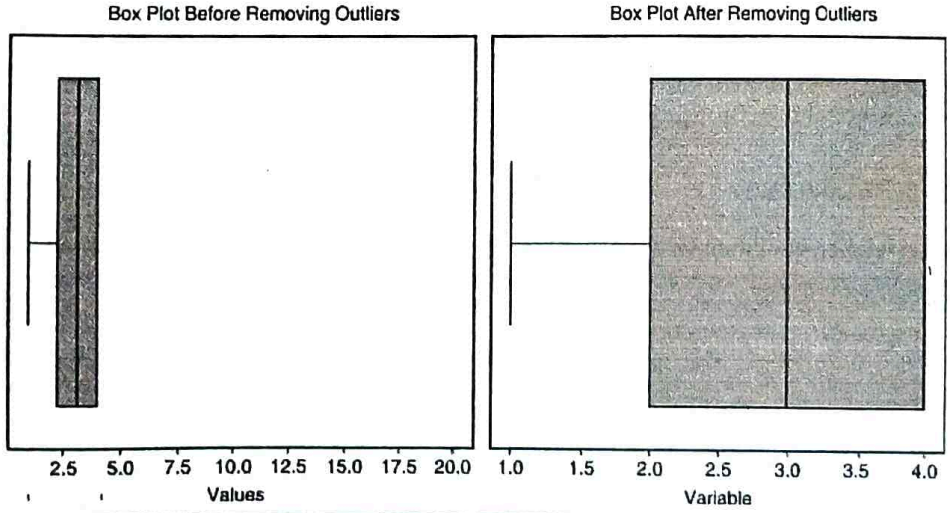
5. Removing Outliers

To eliminate outliers, the process involves removing a specific entry from the dataset based on its exact position, as the outlier detection methods yield a list of data items that meet the outlier criteria according to the applied method.

Example :

```
: import seaborn as sns
  import matplotlib.pyplot as plt
  import pandas as pd

  # Sample DataFrame with outliers
  data = {'Values': [1, 2, 2, 3, 3, 3, 4, 4, 4, 20]}
  df = pd.DataFrame(data)

  # Create a box plot before removing outliers
  plt.figure(figsize=(12, 6))
  plt.subplot(1, 2, 1)
  sns.boxplot(x=df['Values'])
  plt.title('Box Plot Before Removing Outliers')

  # Identify outliers using the interquartile range (IQR)
  Q1 = df['Values'].quantile(0.25)
  Q3 = df['Values'].quantile(0.75)
  IQR = Q3 - Q1

  # Define a threshold to identify outliers
  threshold = 1.5

  # Filter out outliers
  filtered_data = df[(df['Values'] >= Q1 - threshold * IQR) & (df['Values'] <= Q3 + threshold * IQR)]

  # Create a box plot after removing outliers
  plt.subplot(1, 2, 2)
  sns.boxplot(x=filtered_data['Values'])
  plt.title('Box Plot After Removing Outliers')

  # Display the plots
  plt.tight_layout()
  plt.show()
```

OUTPUT



Explanation

In this example :

- The first box plot represents the data with outliers.
- The second box plot is created after removing outliers using the IQR method.

You can observe how the distribution changes after the removal of outliers. This allows you to visually assess the impact of outlier removal on your dataset.

2.1.3 Descriptive Statistics

- Descriptive analysis involves employing numerical techniques to derive insights from data. This method entails summarizing the values of numerical variables, particularly in the context of sales data for a vehicle company.
- In the realm of descriptive analytics literature, key questions revolve around determining statistics such as the mean, mode, and median of selling prices for specific car types or calculating the revenue generated from the sale of a particular automobile model.
- Through descriptive analysis, one can ascertain the central tendency and dispersion of numerical variables within the dataset. This approach proves valuable in gaining a comprehensive understanding of the data, offering practical insights for various data science.

Types of Descriptive Statistics

1. Measures of Central Tendency
2. Measure of Variability
3. Measures of Frequency Distribution

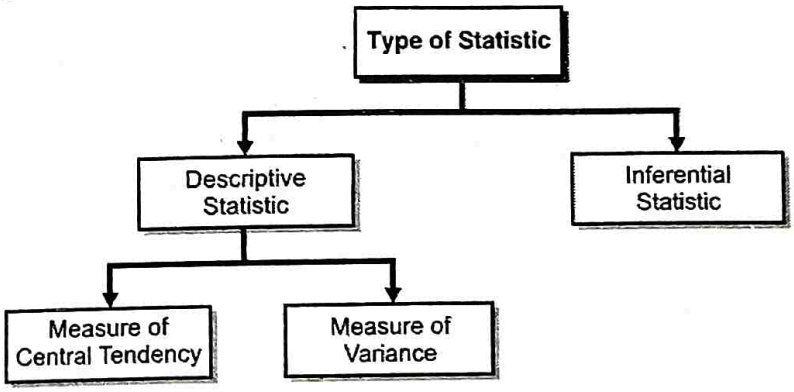


Fig. 2.1.1 : Types of Descriptive Statistics

1. Measures of Central Tendency

It represents the whole set of data by a single value. It gives us the location of the central points. There are three main measures of central tendency :

i) Mean

ii) Mode

iii) Median

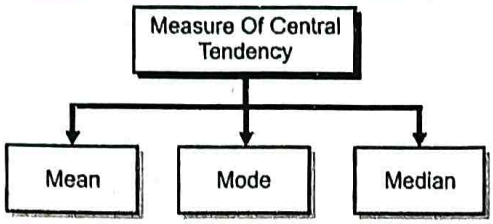


Fig. 2.1.2 : Measure of central tendency

i) Mean

It is the sum of observations divided by the total number of observations. It is also defined as average which is the sum divided by count.

$$\bar{x} = \frac{\sum x}{n}$$

where,

x = Observations

n = number of terms

Python Code Example :

```
import numpy as np

# Sample Data
arr = [5, 6, 11]

# Mean
mean = np.mean(arr)

print("Mean = ", mean)
```

```
Mean =  7.333333333333333
```

ii) Mode

It is the value that has the highest frequency in the given data set. The data set may have no mode if the frequency of all data points is the same. Also, we can have more than one mode if we encounter two or more data points having the same frequency.

Python Code Example

```
from scipy import stats

# sample Data
arr = [1, 2, 2, 3]

# Mode
mode = stats.mode(arr)
print("Mode = ", mode)
```

```
Mode =  ModeResult(mode=array([2]), count=array([2]))
```

iii) Median

It is the middle value of the data set. It splits the data into two halves. If the number of elements in the data set is odd then the center element is the median and if it is even then the median would be the average of two central elements.

Python code example

```
import numpy as np

# sample Data
arr = [1, 2, 3, 4]

# Median
median = np.median(arr)

print("Median = ", median)
```

```
Median =  2.5
```

2. Standard Deviation

It is defined as the square root of the variance. It is calculated by finding the Mean, then subtracting each number from the Mean which is also known as the average, and squaring the result. Adding all the values and then dividing by the no of terms followed by the square root

$$\sigma = \sqrt{\frac{\sum(x-\mu)^2}{N}}$$

where, x = Observation under consideration

N = number of terms

mu = Mean

Python code Example :

```
import statistics

# sample data
arr = [1, 2, 3, 4, 5]

# Standard Deviation
print("Std = ", (statistics.stdev(arr)))
```

```
Std =  1.5811388300841898
```

2.1.4 Hypothesis Testing

- Hypothesis testing serves as a statistical tool for drawing inferences about population data, offering an analytical approach to assess assumptions and ascertain the likelihood of a given outcome with a specified level of accuracy.
- This method enables the validation of experimental results, ensuring their reliability and significance.

- Hypothesis testing uses sample data from the population to draw useful conclusions regarding the population probability distribution.
- It tests an assumption made about the data using different types of hypothesis testing methodologies.
- The hypothesis testing results in either rejecting or not rejecting the null hypothesis.
- An example of hypothesis testing is setting up a test to check if a new medicine works on a disease in a more efficient manner.

i) Null Hypothesis

- The null hypothesis is a concise mathematical statement that is used to indicate that there is no difference between two possibilities.
- In other words, there is no difference between certain characteristics of data. This hypothesis assumes that the outcomes of an experiment are based on chance alone. It is denoted as H0.
- Hypothesis testing is used to conclude if the null hypothesis can be rejected or not.
- Suppose an experiment is conducted to check if girls are shorter than boys at the age of 5. The null hypothesis will say that they are the same height.

ii) Alternative Hypothesis

- The alternative hypothesis is an alternative to the null hypothesis.
- It is used to show that the observations of an experiment are due to some real effect.
- It indicates that there is a statistical significance between two possible outcomes and can be denoted as H1 or Ha.

For the above-mentioned example, the alternative hypothesis would be that girls are shorter than boys at the age of 5.

Hypothesis Testing P Value

- In hypothesis testing, the p value is used to indicate whether the results obtained after conducting a test are statistically significant or not.
- It also indicates the probability of making an error in rejecting or not rejecting the null hypothesis. This value is always a number between 0 and 1.
- The p value is compared to an alpha level, α or significance level.
- The alpha level can be defined as the acceptable risk of incorrectly rejecting the null hypothesis. The alpha level is usually chosen between 1% to 5%.

Hypothesis Testing Critical region

All sets of values that lead to rejecting the null hypothesis lie in the critical region. Furthermore, the value that separates the critical region from the non-critical region is known as the critical value.

2.1.5 Hypothesis Testing Formula

Depending upon the type of data available and the size, different types of hypothesis testing are used to determine whether the null hypothesis can be rejected or not. The hypothesis testing formula for some important test statistics are given below:

- $z = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}}$ · $\bar{X}$ is the sample mean, μ is the population mean, σ is the population standard deviation and n is the size of the sample.
- $t = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{n}}}$ · S is the sample standard deviation.
- $X^2 = \sum \frac{(O_i - E_i)^2}{E_i}$ · O_i is the observed value and E_i is the expected value.

Types of Hypothesis Testing

Selecting the correct test for performing hypothesis testing can be confusing. These tests are used to determine a test statistic on the basis of which the null hypothesis can either be rejected or not rejected. Some of the important tests used for hypothesis testing are given below :

i) Hypothesis Testing Z Test

A z test is a way of hypothesis testing that is used for a large sample size ($n \geq 30$). It is used to determine whether there is a difference between the population mean and the sample mean when the population standard deviation is known. It can also be used to compare the mean of two samples. It is used to compute the z test statistic.

The formulas are given as follows :

One sample :

$$z = \frac{(\bar{x_1} - \bar{x_2}) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

ii) Hypothesis Testing t Test

The t test is another method of hypothesis testing that is used for a small sample size ($n < 30$). It is also used to compare the sample mean and population mean. However, the population standard deviation is not known. Instead, the sample standard deviation is known. The mean of two samples can also be compared using the t test.

One sample :

$$t = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{n}}}$$

Two samples :

$$t = \frac{(\bar{x_1} - \bar{x_2}) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

iii) Hypothesis Testing Chi Square

The Chi square test is a hypothesis testing method that is used to check whether the variables in a population are independent or not. It is used when the test statistic is chi-squared distributed.

iv) One Tailed Hypothesis Testing

One tailed hypothesis testing is done when the rejection region is only in one direction. It can also be known as directional hypothesis testing because the effects can be tested in one direction only. This type of testing is further classified into the right tailed test and left tailed test.

v) Right Tailed Hypothesis Testing

The right tail test is also known as the upper tail test. This test is used to check whether the population parameter is greater than some value. The null and alternative hypotheses for this test are given as follows :

H_0 : The population parameter is ≤ some value

H_1 : The population parameter is > some value.

If the test statistic has a greater value than the critical value then the null hypothesis is rejected

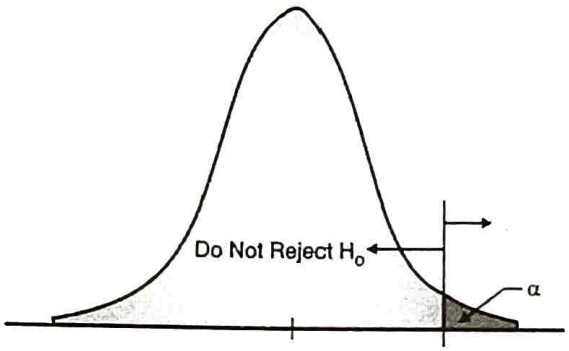


Fig. 2.1.3 : Right tail hypothesis testing

vi) Left Tailed Hypothesis Testing

The left tail test is also known as the lower tail test. It is used to check whether the population parameter is less than some value. The hypotheses for this hypothesis testing can be written as follows :

H_0 : The population parameter is ≥ some value

H_1 : The population parameter is < some value.

The null hypothesis is rejected if the test statistic has a value lesser than the critical value.

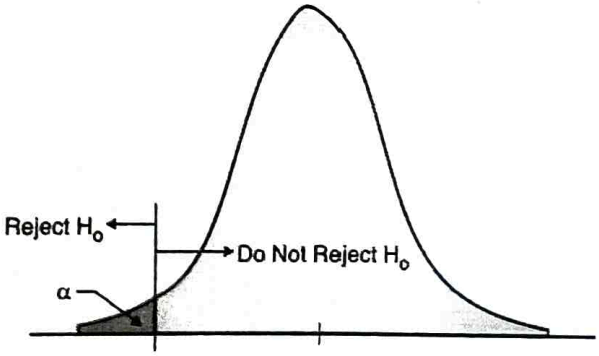


Fig. 2.1.4 : Left tail hypothesis testing

vii) Two Tailed Hypothesis Testing

In this hypothesis testing method, the critical region lies on both sides of the sampling distribution. It is also known as a non - directional hypothesis testing method. The two-tailed test is used when it needs to be determined if the population parameter is assumed to be different than some value. The hypotheses can be set up as follows:

H_0 : the population parameter = some value

H_1 : the population parameter ≠ some value

The null hypothesis is rejected if the test statistic has a value that is not equal to the critical value.

2.1.6 Hypothesis Testing Steps

Hypothesis testing can be easily performed in five simple steps. The most important step is to correctly set up the hypotheses and identify the right method for hypothesis testing. The basic steps to perform hypothesis testing are as follows :

Step 1 : Set up the null hypothesis by correctly identifying whether it is the left-tailed, right-tailed, or two-tailed hypothesis testing.

Step 2 : Set up the alternative hypothesis.

Step 3 : Choose the correct significance level, α, and find the critical value.

Step 4 : Calculate the correct test statistic (z, t or χ) and p-value.

Step 5 : Compare the test statistic with the critical value or compare the p-value with α to arrive at a conclusion. In other words, decide if the null hypothesis is to be rejected or not.

Hypothesis Testing Example

Ex.2.1.1 : The best way to solve a problem on hypothesis testing is by applying the 5 steps mentioned in the previous section. Suppose a researcher claims that the mean average weight of men is greater than 100kgs with a standard deviation of 15 kgs. 30 men are chosen with an average weight of 112.5 Kgs. Using hypothesis testing, check if there is enough evidence to support the researcher's claim. The confidence interval is given as 95%.

Soln. :

Step 1 : This is an example of a right-tailed test. Set up the null hypothesis as H0 : $\mu = 100$.

Step 2 : The alternative hypothesis is given by H1: $\mu > 100$.

Step 3 : As this is a one-tailed test, $\alpha = 100\% - 95\% = 5\%$. This can be used to determine the critical value.

$1 - \alpha = 1 - 0.05 = 0.95$

0.95 gives the required area under the curve. Now using a normal distribution table, the area 0.95 is at z = 1.645. A similar process can be followed for a t-test. The only additional requirement is to calculate the degrees of freedom given by n - 1.

Step 4 : Calculate the z test statistic. This is because the sample size is 30. Furthermore, the sample and population means are known along with the standard deviation.

$$z = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

$$\mu = 100,\ \bar{X} = 112.5,\ n = 30,\ \sigma = 15$$

$$z = \frac{112.5 - 100}{\frac{15}{\sqrt{30}}} = 4.56$$

Step 5 : Conclusion. As $4.56 > 1.645$ thus, the null hypothesis can be rejected.

Ex. 2.1.2 : A survey on cars had conducted in 2011 and determined that 60% of car owners have only one car, 28% have two cars, and 12% have three or more. Supposing that you have decided to conduct your own survey and have collected the data below, determine whether your data supports the results of the study.

Use a significance level of 0.05. Also, given that, out of 129 car owners, 73 had one car and 38 had two cars.

Soln. :

Let us state the null and alternative hypotheses.

H_0 : The proportion of car owners with one, two or three cars is 0.60, 0.28 and 0.12 respectively.

H_1 : The proportion of car owners with one, two or three cars does not match the proposed model.

A Chi-Square goodness of fit test is appropriate because we are examining the distribution of a single categorical variable.

Let's tabulate the given information and calculate the required values.

	Observed (O_i)	**Expected (E_i)**	$O_i - E_i$	$(O_i - E_i)^2$	$(O_i - E_i)^2/E_i$
One car	73	0.60 × 129 = 77.4	-4.4	19.36	0.2501
Two cars	38	0.28 × 129 = 36.1	1.9	3.61	0.1
Three or more cars	18	0.12 × 129 = 15.5	2.5	6.25	0.4032
Total	129				0.7533

Therefore, $x^2 = \frac{\sum(O_i - E_i)^2}{E_i} = 0.7533$

Let's compare it to the chi-square value for the significance level 0.05.

The degrees for freedom = 3 – 1 = 2

Using the table, the critical value for a 0.05 significance level with df = 2 is 5.99.

That means that 95 times out of 100, a survey that agrees with a sample will have a χ^2 value of 5.99 or less.

The Chi-square statistic is only 0.7533, so we will accept the null hypothesis.

2.1.7 Hypothesis Testing - Analysis of Variance (ANOVA) Top of Form

- ANOVA (Analysis of Variance) provides a statistical test of whether two or more population means are equalThe hypothesis is based on available information and the investigator's belief about the population parameters. The specific test considered here is called analysis of variance (ANOVA) and is a test of hypothesis that is appropriate to compare means of a continuous variable in two or more independent comparison groups.

- For example, in some clinical trials there are more than two comparison groups. In a clinical trial to evaluate a new medication for asthma, investigators might compare an experimental medication to a placebo and to a standard treatment (i.e., a medication currently being used). In an observational study such as the Framingham Heart Study, it might be of interest to compare mean blood pressure or mean cholesterol levels in persons who are underweight, normal weight, overweight and obese.
- The technique to test for a difference in more than two independent means is an extension of the two independent samples procedure discussed previously which applies when there are exactly two independent comparison groups.
- The ANOVA technique applies when there are two or more than two independent groups.
- The ANOVA procedure is used to compare the means of the comparison groups and is conducted using the same five step approach used in the scenarios discussed in previous sections. Because there are more than two groups, however, the computation of the test statistic is more involved.
- The test statistic must take into account the sample sizes, sample means and sample standard deviations in each of the comparison groups.
- If one is examining the means observed among, say three groups, it might be tempting to perform three separate group to group comparisons, but this approach is incorrect because each of these comparisons fails to take into account the total data, and it increases the likelihood of incorrectly concluding that there are statistically significate differences, since each comparison adds to the probability of a type I error.
- Analysis of variance avoids these problems by asking a more global question, i.e., whether there are significant differences among the groups, without addressing differences between any two groups in particular (although there are additional tests that can do this if the analysis of variance indicates that there are differences among the groups).
- Consider an example with four independent groups and a continuous outcome measure. The independent groups might be defined by a particular characteristic of the participants such as BMI (e.g., underweight, normal weight, overweight, obese) or by the investigator (e.g., randomizing participants to one of four competing treatments, call them A, B, C and D). Suppose that the outcome is systolic blood pressure, and we wish to test whether there is a statistically significant difference in mean systolic blood pressures among the four groups. The sample data are organized as follows:

	Group 1	**Group 2**	**Group 3**	**Group 4**
Sample Size	n_1	n_2	n_3	n_4
Sample Mean	$\bar{X}_1$	$\bar{X}_2$	$\bar{X}_3$	$\bar{X}_4$
Sample Standard Deviation	s_1	s_2	s_3	s_4

- The hypotheses of interest in an ANOVA are as follows :
  - $H_0 : \mu_1 = \mu_2 = \mu_3 \ldots = \mu_k$
  - H_1 : Means are not all equal.

where k = the number of independent comparison groups.

- In this example, the hypotheses are :
  - $H_0 : \mu_1 = \mu_2 = \mu_3 = \mu_4$
  - H_1 : The means are not all equal.
- The null hypothesis in ANOVA is always that there is no difference in means. The research or alternative hypothesis is always that the means are not all equal and is usually written in words rather than in mathematical symbols. The research hypothesis captures any difference in means and includes, for example, the situation where all four means are unequal, where one is different from the other three, where two are different, and so on. The alternative hypothesis, as shown above, capture all possible situations other than equality of all means specified in the null hypothesis.

2.1.8 Test Statistic for ANOVA

The test statistic for testing $H_0 : \mu_1 = \mu_2 = ... = \mu_k$ is :

$$F = \frac{\sum n_j (\bar{X}_j - \bar{X})^2/(k-1)}{\sum\sum (X - \bar{X}_j)^2/(N-k)}$$

and the critical value is found in a table of probability values for the F distribution with (degrees of freedom) $df_1 = k-1$, $df_2 = N-k$. The table can be found in "Other Resources" on the left side of the pages.

In the test statistic, n_j = the sample size in the j^{th} group (e.g., j = 1, 2, 3, and 4 when there are 4 comparison groups), $\bar{X}_j$ is the sample mean in the j^{th} group, and $\bar{X}$ is the overall mean. k represents the number of independent groups (in this example, k = 4), and N represents the total number of observations in the analysis. Note that N does not refer to a population size, but instead to the total sample size in the analysis (the sum of the sample sizes in the comparison groups, e.g., $N = n_1 + n_2 + n_3 + n_4$). The test statistic is complicated because it incorporates all of the sample data. While it is not easy to see the extension, the F statistic shown above is a generalization of the test statistic used for testing the equality of exactly two means.

The ANOVA Procedure

We will next illustrate the ANOVA procedure using the five step approach. Because the computation of the test statistic is involved, the computations are often organized in an ANOVA table. The ANOVA table breaks down the components of variation in the data into variation between treatments and error or residual variation. Statistical computing packages also produce ANOVA tables as part of their standard output for ANOVA, and the ANOVA table is set up as follows :

Table 2.1.1

Source of Variation	Sums of Squares (SS)	Degrees of Freedom (df)	Mean Squares (MS)	F
Between Treatments	$SSB = \sum n_j (\bar{X}_j - \bar{X})^2$	k-1	$MSB = \frac{SSB}{k-1}$	$F = \frac{MSB}{MSE}$
Error (or Residual)	$SSE = \sum\sum (x - \bar{X}_j)^2$	N-k	$MSE = \frac{MSE}{N-k}$	
Total	$SST = \sum\sum (x - \bar{X})^2$	N-1		

where

$\bar{x}$ = individual observation,

$\bar{x}_j$ = sample mean of the j^{th} treatment (or group),

$\bar{x}$ = overall sample mean,

k = the number of treatments or independent comparison groups, and

N = total number of observations or total sample size.

The ANOVA table above is organized as follows :

- The first column is entitled "Source of Variation" and delineates the between treatment and error or residual variation. The total variation is the sum of the between treatment and error variation.
- The second column is entitled "Sums of Squares (SS)". The between treatment sums of squares is

$$SSB = \sum n_j (\bar{X}_j - \bar{X})^2$$

and is computed by summing the squared differences between each treatment (or group) mean and the overall mean. The squared differences are weighted by the sample sizes per group (n_j). The error sums of squares is :

$$SSE = \sum\sum (X - \bar{X}_j)^2$$

and is computed by summing the squared differences between each observation and its group mean (i.e., the squared differences between each observation in group 1 and the group 1 mean, the squared differences between each observation in group 2 and the group 2 mean, and so on).

The double summation (*SS*) indicates summation of the squared differences within each treatment and then summation of these totals across treatments to produce a single value. (This will be illustrated in the following examples).

The total sums of squares is :

$$SST = \sum\sum (X - \bar{X})^2$$

and is computed by summing the squared differences between each observation and the overall sample mean. In an ANOVA, data are organized by comparison or treatment groups. If all of the data were pooled into a single sample, SST would reflect the numerator of the sample variance computed on the pooled or total sample. SST does not figure into the F statistic directly. However, SST = SSB + SSE, thus if two sums of squares are known, the third can be computed from the other two.

- The third column contains degrees of freedom. The between treatment degrees of freedom is df_1 = k-1. The error degrees of freedom is df_2 = N - k. The total degrees of freedom is N-1 (and it is also true that (k-1) + (N-k) = N-1).
- The fourth column contains "Mean Squares (MS)" which are computed by dividing sums of squares (SS) by degrees of freedom (df), row by row. Specifically, MSB = $\frac{SSB}{(k-1)}$ and MSE = $\frac{SSE}{(N-k)}$. Dividing $\frac{SST}{(N-1)}$ produces the variance of the total sample. The F statistic is in the rightmost column of the ANOVA table and is computed by taking the ratio of MSB/MSE.

Examples

Ex. 2.1.3 : A clinical trial is run to compare weight loss programs and participants are randomly assigned to one of the comparison programs and are counselled on the details of the assigned program. Participants follow the assigned program for 8 weeks. The outcome of interest is weight loss, defined as the difference in weight measured at the start of the study (baseline) and weight measured at the end of the study (8 weeks), measured in pounds. Three popular weight loss programs are considered. The first is a low-calorie diet. The second is a low-fat diet and the third is a low carbohydrate diet. For comparison purposes, a fourth group is considered as a control group. Participants in the fourth group are told that they are participating in a study of healthy behaviours with weight loss only one component of interest. The control group is included here to assess the placebo effect (i.e., weight loss due to simply participating in the study). A total of twenty patients agree to participate in the study and are randomly assigned to one of the four diet groups. Weights are measured at baseline and patients are counselled on the proper implementation of the assigned diet (with the exception of the control group). After 8 weeks, each patient's weight is again measured and the difference in weights is computed by subtracting the 8-week weight from the baseline weight. Positive differences indicate weight losses and negative differences indicate weight gains. For interpretation purposes, we refer to the differences in weights as weight losses and the observed weight losses are shown below.

Low Calorie	Low Fat	Low Carbohydrate	Control
8	2	3	2
9	4	5	2
6	3	4	-1
7	5	2	0
3	1	3	3

Is there a statistically significant difference in the mean weight loss among the four diets? We will run the ANOVA using the five-step approach.

Soln. :

Step 1 : Set up hypotheses and determine level of significance

$H_0 : \mu_1 = \mu_2 = \mu_3 = \mu_4$ H_1: Means are not all equal $\alpha = 0.05$

Step 2 : Select the appropriate test statistic.

The test statistic is the F statistic for ANOVA, $F = \frac{MSB}{MSE}$.

Step 3 : Set up decision rule.

The appropriate critical value can be found in a table of probabilities for the F distribution (see "Other Resources"). In order to determine the critical value of F we need degrees of freedom, $df_1 = k - 1$ and $df_2 = N - k$. In this example, $df_1 = k - 1 = 4 - 1 = 3$ and $df_2 = N-k = 20 - 4 = 16$. The critical value is 3.24 and the decision rule is as follows : Reject H_0 if $F \geq 3.24$.

Step 4 : Compute the test statistic.

To organize our computations we complete the ANOVA table. In order to compute the sums of squares we must first compute the sample means for each group and the overall mean based on the total sample.

	Low Calorie	Low Fat	Low Carbohydrate	Control
n	5	5	5	5
Group mean	6.6	3.0	3.4	1.2

If we pool all N = 20 observations, the overall mean is $\bar{X} = 3.6$.

We can now compute

$$SSB = \sum n_j (\bar{X}_j - \bar{X})^2$$

So, in this case :

$$SSB = 5(6.6 - 3.6)^2 + 5(3.0 - 3.6)^2 + 5(3.4 - 3.6)^2 + 5(1.2 - 3.6)^2$$

$$SSB = 45.0 + 1.8 + 0.2 + 28.8 = 75.8$$

Next we compute,

$$SSE = \sum\sum (X - \bar{X}_j)^2$$

SSE requires computing the squared differences between each observation and its group mean. We will compute SSE in parts. For the participants in the low-calorie diet :

Low Calorie	(X - 6.6)	$(X - 6.6)^2$
8	1.4	2.0
9	2.4	5.8
6	-0.6	0.4
7	0.4	0.2
3	-3.6	13.0
Totals	0	21.4

Thus, $\sum(X - \bar{X}_1)^2 = 21.4$

For the participants in the low fat diet :

Low Fat	(X - 3.0)	$(X - 3.0)^2$
2	-1.0	1.0
4	1.0	1.0
3	0.0	0.0
5	2.0	4.0
1	-2.0	4.0
Totals	0	10.0

Thus, $\sum(X - \bar{X}_2)^2 = 10.0$

For the participants in the low carbohydrate diet :

Low Carbohydrate	(X - 3.4)	$(X - 3.4)^2$
3	-0.4	0.2
5	1.6	2.6
4	0.6	0.4
2	-1.4	2.0
3	-0.4	0.2
Totals	0	5.4

Thus, $\sum(X - \bar{X}_3)^2 = 5.4$

For the participants in the control group :

Control	(X - 1.2)	$(X - 1.2)^2$
2	0.8	0.6
2	0.8	0.6
-1	-2.2	4.8
0	-1.2	1.4
3	1.8	3.2
Totals	0	10.6

Thus, $\sum(X - \bar{X}_4)^2 = 10.6$

Therefore, $SSE = \sum\sum(X - \bar{X}_j)^2 = 21.4 + 10.0 + 5.4 + 10.6 = 47.4$

We can now construct the **ANOVA table.**

Source of Variation	Sums of Squares (SS)	Degrees of Freedom (df)	Means Squares (MS)	F
Between Treatment	75.8	4 – 1 = 3	75.8/3 = 25.3	25.3/3.0 = 8.43
Error (or Residual)	47.4	20 – 4 = 16	47.4/16 = 3.0	
Total	123.2	20 – 1 = 19		

Step 5 : Conclusion.

We reject H_0 because $8.43 \geq 3.24$. We have statistically significant evidence at $\alpha = 0.05$ to show that there is a difference in mean weight loss among the four diets.

ANOVA is a test that provides a global assessment of a statistical difference in more than two independent means. In this example, we find that there is a statistically significant difference in mean weight loss among the four diets considered. In addition to reporting the results of the statistical test of hypothesis (i.e., that there is a statistically significant difference in mean weight losses at $\alpha = 0.05$), investigators should also report the observed sample means to facilitate interpretation of the results.

Ex.2.1.4 : Calcium is an essential mineral that regulates the heart, is important for blood clotting and for building healthy bones. The National Osteoporosis Foundation recommends a daily calcium intake of 1000-1200 mg/day for adult men and women. While calcium is contained in some foods, most adults do not get enough calcium in their diets and take supplements. Unfortunately some of the supplements have side effects such as gastric distress, making them difficult for some patients to take on a regular basis.

A study is designed to test whether there is a difference in mean daily calcium intake in adults with normal bone density, adults with osteopenia (a low bone density which may lead to osteoporosis) and adults with osteoporosis. Adults 60 years of age with normal bone density, osteopenia and osteoporosis are selected at random from hospital records and invited to participate in the study. Each participant's daily calcium intake is measured based on reported food intake and supplements. The data are shown below.

Normal Bone Density	Osteopenia	Osteoporosis
1200	1000	890
1000	1100	650
980	700	1100
900	800	900
750	500	400
800	700	350

Is there a statistically significant difference in mean calcium intake in patients with normal bone density as compared to patients with osteopenia and osteoporosis? We will run the ANOVA using the five-step approach.

Soln. :

Step 1 : Set up hypotheses and determine level of significance

$H_0 : \mu_1 = \mu_2 = \mu_3$ H_1: Means are not all equal $\alpha = 0.05$

Step 2 : Select the appropriate test statistic.

The test statistic is the F statistic for ANOVA, F=MSB/MSE.

Step 3 : Set up decision rule.

In order to determine the critical value of F we need degrees of freedom, $df_1 = k - 1$ and $df_2 = N - k$. In this example, $df_1 = k - 1 = 3 - 1 = 2$ and $df_2 = N - k = 18 - 3 = 15$. The critical value is 3.68 and the decision rule is as follows : Reject H_0 if $F \geq 3.68$.

Step 4 : Compute the test statistic.

To organize our computations we will complete the ANOVA table. In order to compute the sums of squares we must first compute the sample means for each group and the overall mean.

Normal Bone Density	Osteopenia	Osteoporosis
$n_1 = 6$	$n_2 = 6$	$n_3 = 6$
$\bar{X}_1 = 938.3$	$\bar{X}_2 = 800.0$	$\bar{X}_3 = 715.0$

If we pool all N = 18 observations, the overall mean is 817.8.

We can now compute :

$$SSB = \sum n_j(\bar{X}_j - \bar{X})^2$$

Substituting :

$$SSB = 6(938.3333 - 817.7778)^2 + 6(800.0 - 817.7778)^2 + 6(715.0 - 817.7778)^2$$

Finally,

$$SSB = 87201.77 + 63379.66 + 1896.301 = 152477.7$$

Next,

$$SSE = \sum\sum(X - \bar{X}_j)^2$$

SSE requires computing the squared differences between each observation and its group mean. We will compute SSE in parts. For the participants with normal bone density:

Normal Bone Density	(X - 938.3)	$(X - 938.3333)^2$
1200	261.6667	68,486.9
1000	61.6667	3,806.9
980	41.6667	1,738.9
900	-38.3333	1,466.9
750	-188.333	35,456.9
800	-138.333	19,126.9
Total	0	130,083.3

Thus, $\sum(X - \bar{X}_1)^2 = 130083.3$

For participants with osteopenia:

Osteopenia	(X - 800.0)	$(X - 800.0)^2$
1000	200	40,000
1100	300	90,000
700	-100	10,000
800	0	0
500	-300	90,000
700	-100	10,000
Total	0	240,000

Thus, $\sum(X - \bar{X}_2)^2 = 240000.0$

For participants with osteoporosis:

Osteoporosis	(X - 715.0)	$(X - 715.0)^2$
890	175	30,625
650	-65	4,225
1100	385	148,225
900	185	34,225
400	-315	99,225
350	-365	133,225
Total	0	449,750

Thus, $\sum(X - \bar{X}_3)^2 = 449750.0$

$$SSE = \sum\sum(X - \bar{X}_j)^2 = 130083.3 + 240000.0 + 449750.0 = 819833.3$$

We can now construct the ANOVA table.

Source of Variation	Sums of Squares (SS)	Degrees of freedom (df)	Mean Squares (MS)	F
Between Treatments	152,477.7	2	76,238.6	1.395
Error or Residual	819,833.3	15	54,655.5	
Total	972,311.0	17		

Step 5 : Conclusion.

We do not reject H_0 because $1.395 < 3.68$.

1) One Way ANOVA

A one way ANOVA is used to compare two means from two independent (unrelated) groups using the F-distribution. The null hypothesis for the test is that the two means are equal. Therefore, a significant result means that the two means are unequal.

Examples of when to use a one way ANOVA

- **Situation 1 :** You have a group of individuals randomly split into smaller groups and completing different tasks. For example, you might be studying the effects of tea on weight loss and form three groups: green tea, black tea, and no tea.
- **Situation 2 :** Similar to situation 1, but in this case the individuals are split into groups based on an attribute they possess. For example, you might be studying leg strength of people according to weight. You could split participants into weight categories (obese, overweight and normal) and measure their leg strength on a weight machine.

2) Two Way ANOVA

- A two-way ANOVA tests the effect of two independent variables on a dependent variable. A two-way ANOVA test analyses the effect of the independent variables on the expected outcome along with their relationship to the outcome itself. Random factors would be considered to have no statistical influence on a data set, while systematic factors would be considered to have statistical significance. By using ANOVA, a researcher is able to determine whether the variability of the outcomes is due to chance or to the factors in the analysis. ANOVA has many applications in finance, economics, science, medicine, and social science.
- For example, suppose a botanist wants to explore how sunlight exposure and watering frequency affect plant growth. She plants 40 seeds and lets them grow for two months under different conditions for sunlight exposure and watering frequency. After two months, she records the height of each plant.
- In this case, we have the following variables :
  - **Response variable :** plant growth
  - **Factors :** sunlight exposure, watering frequency
- And we would like to answer the following questions:
  - Does sunlight exposure affect plant growth?
  - Does watering frequency affect plant growth?
  - Is there an interaction effect between sunlight exposure and watering frequency? (e.g. the effect that sunlight exposure has on the plants is dependent on watering frequency)
- We would use a two-way ANOVA for this analysis because we have **two** factors. If instead we wanted to know how only watering frequency affected plant growth, we would use a one-way ANOVA since we would only be working with one factor.

2.2 Introduction to Machine Learning

- Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience.
- The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data, or both.
- Arthur Samuel, an early American leader in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM.
- He defined machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed." However, there is no universally accepted definition for machine learning. Different authors define the term differently.

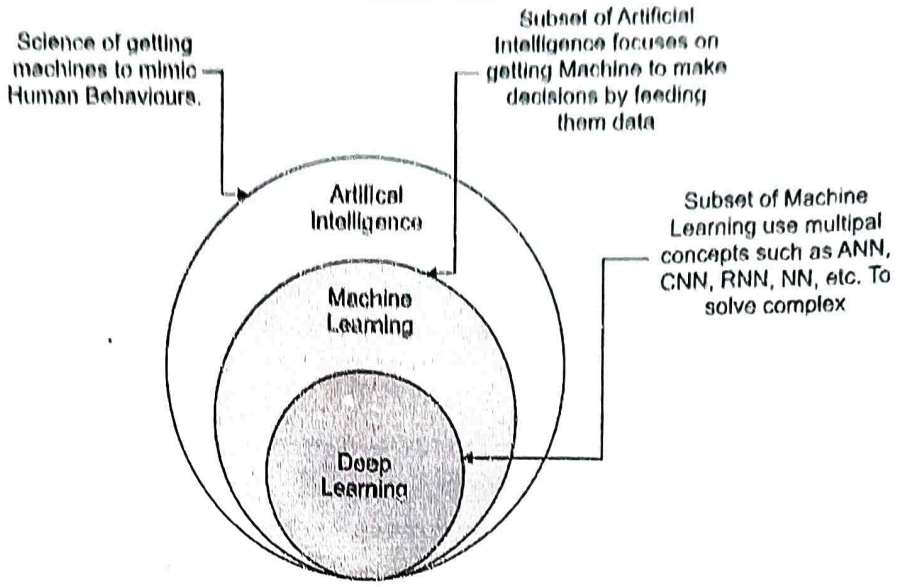


Fig. 2.2.1 : Introduction Of Machine learning

Definition of Learning

Definition : *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E. Examples*

i) Handwriting recognition learning problem

- Task T : Recognising and classifying handwritten words within images
- Performance P : Percent of words correctly classified
- Training experience E : A dataset of handwritten words with given classifications

ii) A robot driving learning problem

- Task T : Driving on highways using vision sensors
- Performance measure P : Average distance travelled before an error
- Training experience : A sequence of images and steering commands recorded while observing a human driver

iii) A chess learning problem

- Task T : Playing chess
- Performance measure P : Percent of games won against opponents
- Training experience E : Playing practice games against itself

Definition : *A computer program which learns from experience is called a machine learning program or simply a learning program. Such a program is sometimes also referred to as a learner.*

2.2.1 Types of Machine Learning

Basically, Machine Learning is divided into three parts.

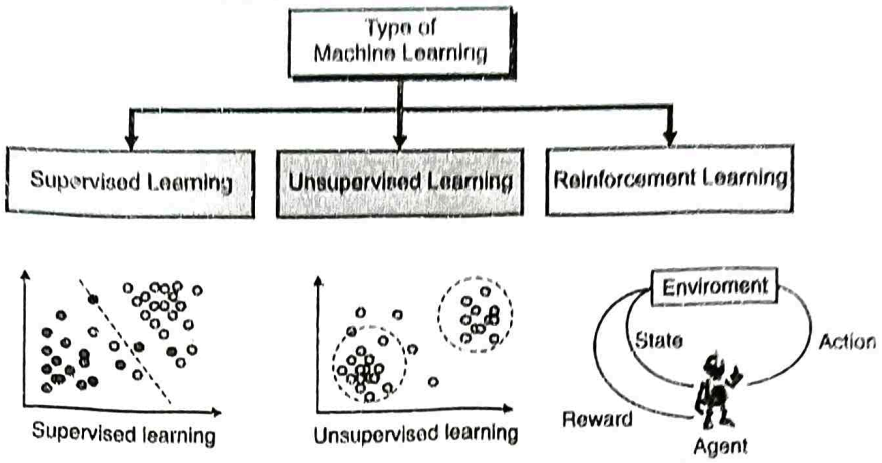


Fig. 2.2.2 : Types of Machine Learning

2.2.1(A) Supervised Learning

- Supervised Learning model has sets of input variables **(x)** and an output variable **(y)**. An algorithm identifies the mapping function between **(x)** and **(y)** variables. The relationship is **y = f(x)**.
- Supervised Learning algorithms are trained using labeled examples, such as an input where the desired output is known. That means within our dataset we are going to have some historical features with historical labels.
- Let's say for example we are creating an E-mail spam detection system. Now we will provide information such as a segment of text which has a category label. So we take a bunch of previous E-mails and someone has already gone by those E-mails and classified them using the correct labels as spam or ham *(Non-Spam)*. So there were some E-mails which are were classified as the spam or ham *(Non-Spam)* so the idea would be for the future text information such as future E-mails using the historical labeled data and Machine Learning algorithm which will learn using historical data in order to predict the new data whether it is a spam or ham.
- Supervised Learning is commonly used in applications where historical data predicts likely future events.

Supervised Learning Process in Machine Learning

- So let's go in and go through this step by step
  - Get data and it actually depends on what domain you are working in from where this data actually comes from. This data can come from your customer's feedback, research or it would have been collected from some online database, etc. So at some point the data has to be acquired.
  - Once we actually acquire the data then we need to clean and format the data and often we do this using **Pandas**.

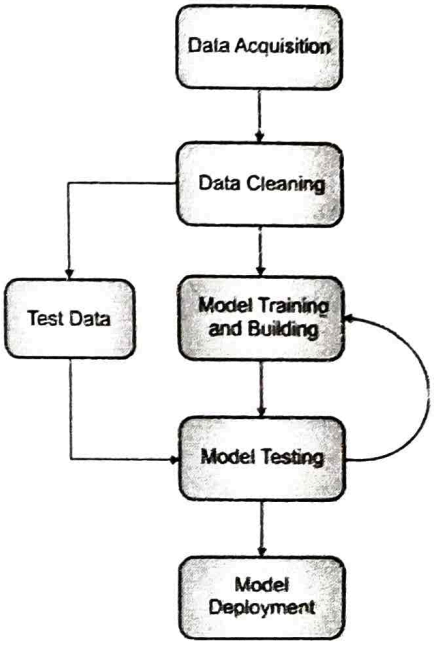


Fig. 2.2.3 : Supervised Learning Process in Machine Learning

- Then we split the data into Training data and Testing data. What we do here is, we take some portion of our data like 30% to make it our test data, and then we will take the remaining data which is 70% as our training data. Now what we are going to do here is we are going to use that specific training set on our machine learning model in order to fit a model to that training data.

  Then we want to know how our model performs. So then we run that test data through the model and compare the model predictions to the actual correct label that the test data have because remember we actually know the correct label for the test data. So we can run that test data features through the model, get our models predictions and compare it to the correct answer and then we can evaluate the model and then maybe you want to go back based off that performance and adjust the model parameters such as decreasing the amount of test size to 20% or 25%.

- And once we are satisfied by this we can then deploy the model to the real world.

Supervised Learning and it's types

As we know from earlier Supervised Learning algorithms are trained using labeled examples, such as an input where the desired output is known.

Types of Supervised Learning :

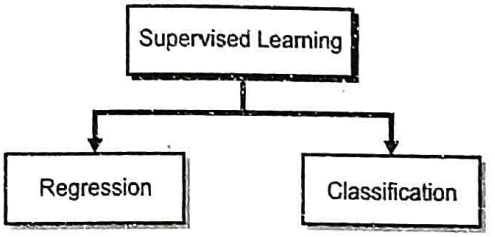


Fig. 2.2.4 : Types of Supervised Learning

Supervised Learning is divided into 2 types.

(i) Classification :

Classification is a process of categorizing a given set of data into classes, It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label, or categories. The classification predictive modelling is a task of approximating the mapping function from input variables to discrete output variables and the main goal is to identify which class or category the new data will fall into.

To simplify this, Let's look at some examples: Is that email spam or not, Does a patient is suffering from heart diseases or not, etc.

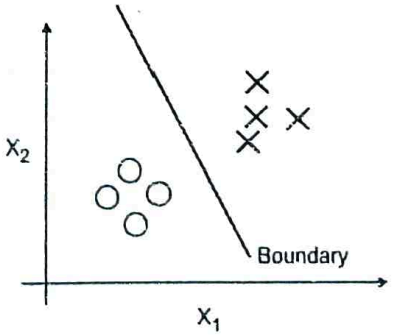


Fig. 2.2.5

Classification problems can be solved with a numerous amount of algorithms. Any algorithm you choose it actually depends on the situation. Here are some of the popular classification algorithms.

- Support Vector Machine
- Decision Tree
- K-Nearest Neighbour
- Random Forest

1. **Support Vector Machine (SVM) :** This algorithm finds a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.
2. **Decision Tree :** It's a flowchart-like tree structure where an internal node represents a feature(or attribute), a branch represents a decision rule, and each leaf node represents the outcome.
3. **K-Nearest Neighbor (KNN) :** It classifies data points based on how their neighbors are classified. It's a simple algorithm that stores all available cases and classifies new cases based on a similarity measure.
4. **Random Forest :** This is an ensemble learning method. It constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

(ii) Regression :

Regression Analysis is a **predictive modeling** technique. It estimates the relationship between a dependent variable we can also call it as **target** & an independent variable which is also known as a **predictor.**

Regression algorithm goal is to predict the continuous number such as sales, price, scores, etc. The equation for basic linear regression can be also written as :

$$y = m * x + c$$

Where;

y : Dependent variable

x : Independent variable

c : y-Intercept

m : Co-efficient

For simple regression problems such as this, the model predictions are represented by the line of best fit. For models using two independent variables, the plane will be used. Finally, for a model using more than two independent variables, a hyperplane will be used.

To simplify this, Let's look at some examples: Predicting house prices. predicting student scores, etc.

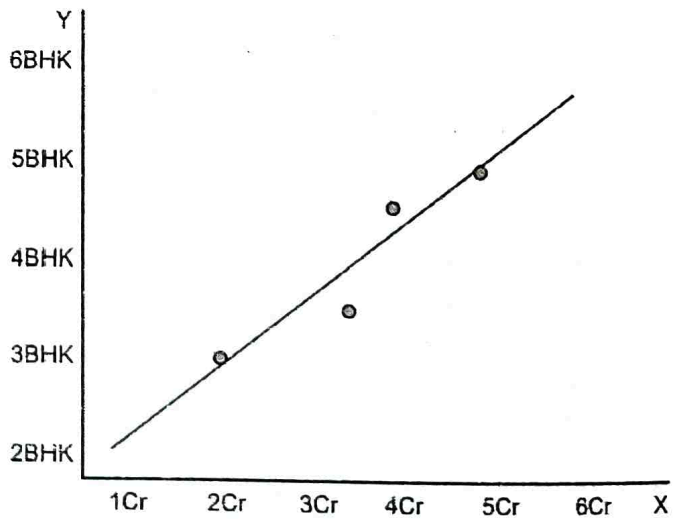


Fig. 2.2.6

There are many different types of Regression Algorithms. The three most common are listed down below :

a) Linear Regression

- Linear regression is a standard method that depicts the linear relationship between the continuous output variable and the input factors.
- Simple linear regression only requires one input variable, whereas multiple linear regression requires several. The model computes the coefficients using the least squares method to minimize the sum of squared residuals.
- The interpretation of coefficients enables the understanding of how input elements influence output. Polynomial regression extends linear regression to consider non-linear connections by employing polynomial terms. Regularization methods like ridge regression and lasso regression avoid overfitting by introducing penalty terms.

b) Decision Trees for Regression

- Regression tasks using continuous rather than categorical output variables can also be performed using decision trees.
- Regression trees divide the data depending on features to predict constant values. Minimizing mean squared error or mean absolute error is a common splitting criterion for regression trees.
- Pruning techniques, such as cost-complexity pruning, help prevent overfitting and simplify the tree structure.
- Multiple regression trees are used in ensemble approaches, such as random forests, to increase prediction accuracy.

c) Support Vector Regression (SVR)

- Support Vector regression applies SVM principles to regression problems. SVR looks for a function that approximately approximates the relationship between the continuous output variable and the input variables at a given level of error tolerance.
- Linear SVR looks for the best linear approximation, whereas non-linear SVR maps the data into higher-dimensional feature spaces using kernel functions. Tuning hyperparameters like the penalty parameter (C) and epsilon value is crucial for achieving the best SVR performance. SVR has uses in many industries, such as anomaly identification, energy load forecasting, and stock market forecasting.

d) Neural Networks for Regression

- Neural networks are strong models that can capture complex relationships between inputs and outputs. In regression problems, neural networks learn to approximate the underlying function that converts discrete input to continuous output values. Feed-forward neural networks, which have interconnected layers of artificial neurons, have several applications.
- Activation functions like sigmoid, ReLU, or tanh incorporate non-linearities to capture complex patterns.
- Depending on the difference between the expected and actual output values, the network weights are modified using the learning technique known as backpropagation. Overfitting can be prevented using regularization strategies like dropout and L1/L2 regularization.

e) Gradient Boosting Regression

- By integrating several weak regression models, frequently decision trees, the ensemble learning technique known as gradient boosting regression produces a superior prediction model.
- The method learns by repeatedly fitting the base models to the residuals of the prior models. Gradient boosting variables like the learning rate and tree count must be appropriately tuned for the best results.
- Gradient-increasing techniques like XGBoost and LightGBM further improve regression accuracy and computing efficiency.

Regression vs Classification

Let's see the Difference between Regression and Classification.

Table 2.2.1

Regression	Classification
Regression is the task of predicting a continuous quantity.	Classification is the task of predicting a discrete class label.
Regression means to predict the output value using training data.	Classification means to group the output into a clsss.
A regression problem requires the prediction of a quantity.	In a classification problem data is labelled into one of two or more class.
If it is real number or continuous then it is regression problem.	If it is discrete or categorical varibale, then it is classification problem.
A regression problem with multiple input variables is called a multivariable regression problem.	A classification problem with two classes is called binary, more than 2 classes is called as multi-classification problem.
Example : Predict the house prices.	**Example :** Is than E-mail is spam or not a spam.

2.2.1(B) Unsupervised Learning

Unsupervised Learning is a type of Machine Learning that looks for previously undetected patterns in a dataset with no pre-existing labels and minimum of human supervision. The Unsupervised Learning algorithms are not provided any "Answers" to learn from; it must make sense of the data just by observations.

1. Clustering

Clustering is the assignment of objects to homogeneous groups (called clusters) while making sure that objects in different groups is not similar. Clustering is considered an unsupervised task as it aims to describe the hidden structure of the objects. Each object is described by a set of characters called features. The first step of dividing objects into clusters is to define the distance between the different objects. Defining an adequate distance measure is crucial for the success of the clustering process.

i) k-means

- There are many clustering algorithms, each has its advantages and disadvantages. A popular algorithm for clustering is k-means, which aims to identify the best k cluster centers in an iterative manner. Cluster centers are served as "representative" of the objects associated with the cluster. k-means' key features are also its drawbacks:
  - The number of clusters (k) must be given explicitly. In some cases, the number of different groups is unknown.
  - k-means iterative nature might lead to an incorrect result due to convergence to a local minimum.
  - The clusters are assumed to be spherical.
- Despite these drawbacks, k-means remains the right and popular choice in many cases. An example of clustering using k-means on spherical data can be seen in Fig. 2.2.7.

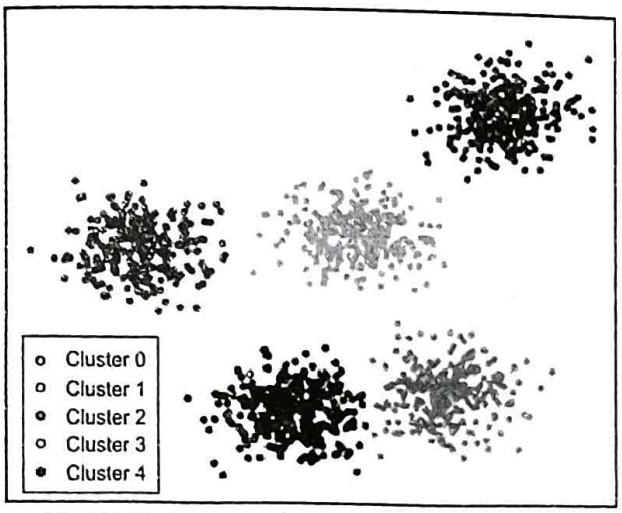


Fig. 2.2.7 : k-means clustering on spherical data

ii) OPTICS

- A different clustering algorithm is OPTICS, which is a density-based clustering algorithm. Density-based clustering, unlike centroid-based clustering, works by identifying "dense" clusters of points, allowing it to learn clusters of arbitrary shape and densities. OPTICS can also identify outliers (noise) in the data by identifying scattered objects.

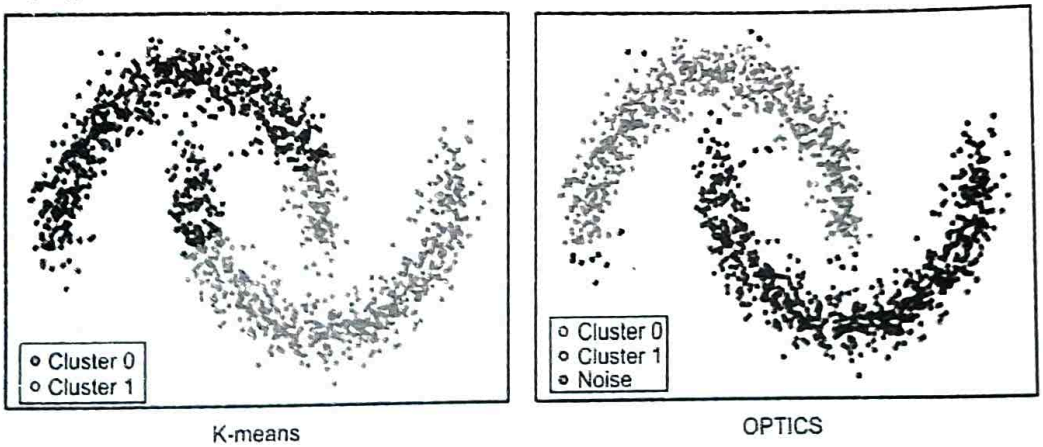


Fig. 2.2.8 : k-means versus OPTICS on moon-like data

- The OPTICS approach yields a very different grouping of data points than k-means; it classifies outliers and more accurately represents clusters that are by nature not spherical. An example of running k-means versus OPTICS on moon-like data is presented in Fig. 2.2.8 :

2. Dimensionality Reduction

- In the field of machine learning, it is useful to apply a process called dimensionality reduction to highly dimensional data. The purpose of this process is to reduce the number of features under consideration, where each feature is a dimension that partly represents the objects. Why is dimensionality reduction important? As more features are added, the data becomes very sparse and analysis suffers from the curse of dimensionality. Additionally, it is easier to process smaller data sets.
- Dimensionality reduction can be executed using two different methods :
  - Selecting from the existing features (feature selection)
  - Extracting new features by combining the existing features (feature extraction)
- The main technique for feature extraction is the Principle Component Analysis (PCA). PCA guarantees finding the best linear transformation that reduces the number of dimensions with a minimum loss of information. Sometimes the information that was lost is regarded as noise – information that does not represent the phenomena we are trying to model, but is rather a side effect of some usually unknown processes. PCA process can be visualized as follows (Fig. 2.2.9):

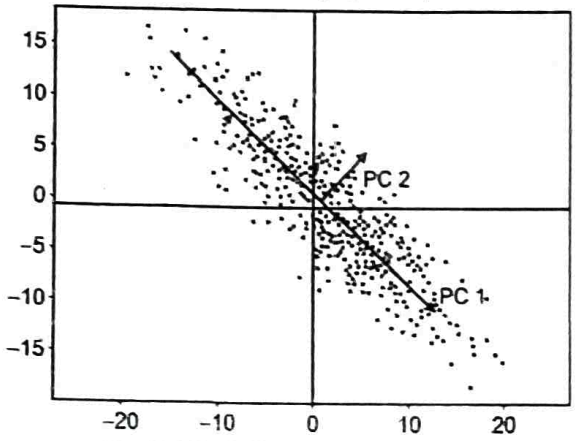


Fig. 2.2.9 : PCA process visualized

- Following the process in the example, we might be content with just PC1 – one feature instead of originally two.
- There is a great choice of dimensionality reduction techniques: some are linear like PCA, some are nonlinear and lately methods using deep learning are gaining popularity

2.2.2 Bias-variance Trade-off

The bias is known as the difference between the prediction of the values by the Machine Learning model and the correct value. Being high in biasing gives a large error in training as well as testing data. It recommended that an algorithm should always be low-biased to avoid the problem of underfitting. By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as the Underfitting of Data. This happens when the hypothesis is too simple or linear in nature. Refer to the graph given below for an example of such a situation.

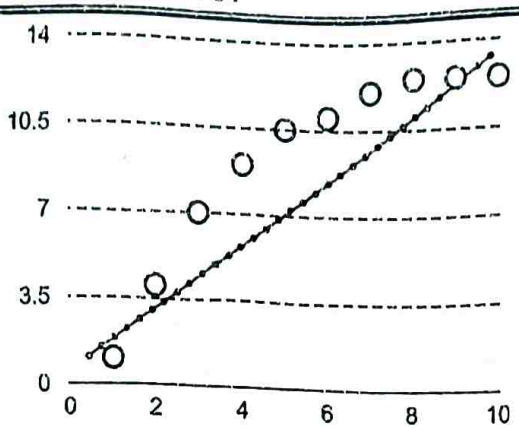


Fig. 2.2.10 : High Bias in the Model

In such a problem, a hypothesis looks like follows

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Variance

The variability of model prediction for a given data point which tells us the spread of our data is called the variance of the model. The model with high variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn't seen before. As a result, such models perform very well on training data but have high error rates on test data. When a model is high on variance, it is then said to as Overfitting of Data. Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high. While training a data model variance should be kept low. The high variance data looks as follows.

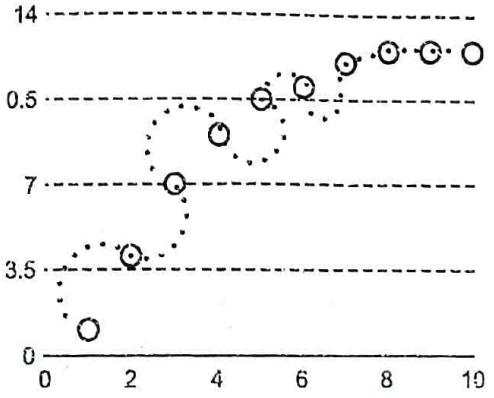


Fig. 2.2.11 : High Variance in the Model

In such a problem, a hypothesis looks like follow

Bias Variance Tradeoff

- If the algorithm is too simple (hypothesis with linear equation) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree equation) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well.
- Well, there is something between both of these conditions, known as a Trade-off or Bias Variance Trade-off. This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time. For the graph, the perfect tradeoff will be like this.

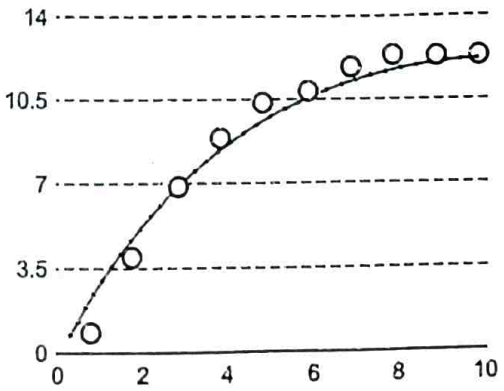
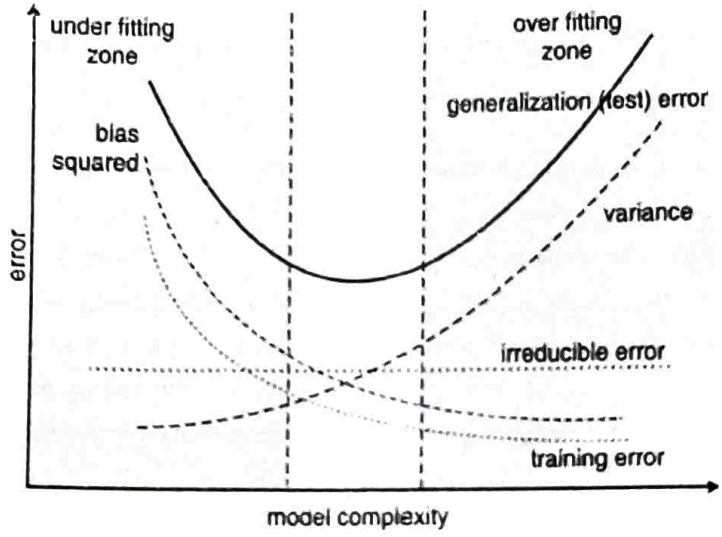


Fig. 2.2.12

- We try to optimize the value of the total error for the model by using the Bias-Variance Tradeoff.
- The best fit will be given by the hypothesis on the tradeoff point. The error to complexity graph to show trade-off is given as –



Region for the least value of total error

Fig. 2.2.13 : Region for the Least Value of Total Error

- This is referred to as the best point chosen for the training of the algorithm which gives low error in training as well as testing data.

2.2.3 Underfitting and Overfitting

A statistical model or a machine learning algorithm is said to have underfitting when a model is too simple to capture data complexities. It represents the inability of the model to learn the training data effectively result in poor performance both on the training and testing data. In simple terms, an underfit model's are inaccurate, especially when applied to new, unseen examples. It mainly happens when we uses very simple model with overly simplified assumptions. To address underfitting problem of the model, we need to use more complex models, with enhanced feature representation, and less regularization.

Note : The underfitting model has High bias and low variance.

Reasons for Underfitting

1. The model is too simple, So it may be not capable to represent the complexities in the data.
2. The input features which is used to train the model is not the adequate representations of underlying factors influencing the target variable.
3. The size of the training dataset used is not enough.
4. Excessive regularization are used to prevent the overfitting, which constraint the model to capture the data well.
5. Features are not scaled.

Techniques to Reduce Underfitting

1. Increase model complexity.
2. Increase the number of features, performing feature engineering.
3. Remove noise from the data.
4. Increase the number of epochs or increase the duration of training to get better results.

Overfitting in Machine Learning

- A statistical model is said to be overfitted when the model does not make accurate predictions on testing data. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. And when testing with test data results in High variance. Then the model does not categorize the data correctly, because of too many details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models. A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.
- In a nutshell, Overfitting is a problem where the evaluation of machine learning algorithms on training data is different from unseen data.

Reasons for Overfitting :

1. High variance and low bias.
2. The model is too complex.
3. The size of the training data.

Techniques to Reduce Overfitting

1. Increase training data.
2. Reduce model complexity.
3. Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
4. Ridge Regularization and Lasso Regularization.
5. Use dropout for neural networks to tackle overfitting.

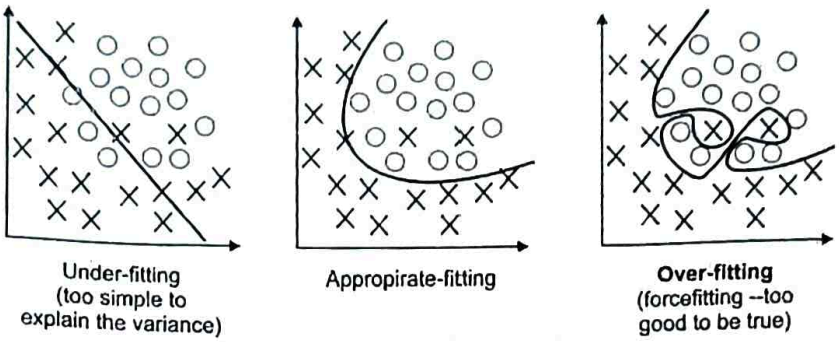
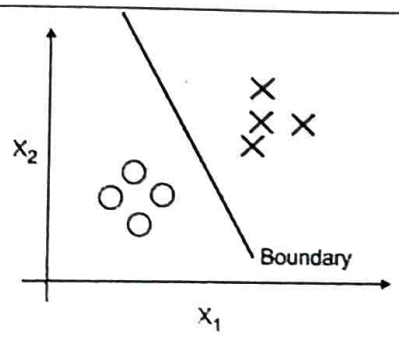
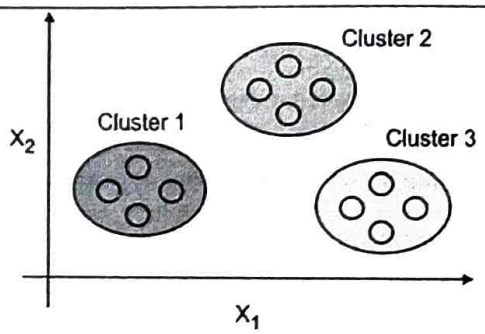


Fig 2.2.14 : Underfitting and Overfitting

2.2.4 Supervised Learning vs Unsupervised Learning

Now Let's see the difference between Supervised Learning and Unsupervised Learning.

Supervised Learning	Unsupervised Learning
Input data is labelled	Input data is unlabelled
Uses training Dataset	Uses just input dataset
Data is classified based on training dataset	Uses properties of given data to classify it
Used for prediction	Used for analysis
Divided into 2 types regression and classification	Divided into 2 types clustering and association
Known number of classes	Unknown number of classes
Use off line analysis of data	Use real time analysis of data
 Fig. 2.2.15	 Fig. 2.2.16

Supervised Learning Examples in Real Life

So as we saw that there are 2 types of supervised learning, let's see their implementation in the real world.

(a) **Real-world implementation of Supervised Learning with respect to Classification :**

- Rating of a product or a movie.
- Fraud detection.
- Spam or not spam.
- Risk analysis.

(b) Real-world implementation of Supervised Learning with respect to Regression

- House price prediction.
- The amount of Fraud.
- Probability of Risk.
- Detecting the increase or decrease of Crime Rates.

Advantages of Supervised Learning

- You can get very **specific** about the definition of the classes, which means that you can train the classifier in a way that has a **perfect decision boundary** to distinguish different classes accurately.
- You can specifically determine **how many classes** you want to have.
- After training, you don't necessarily need to keep the training examples in a memory. You can keep the **decision boundary as a mathematical formula** and that would be enough for classifying future inputs.

Disadvantages of Supervised Learning

- Your decision boundary **might be overtrained.** This means that if your training set is not including some examples that you want to have in a class when you use those examples after training, you might not get the correct class label.
- When this an input which is not from any of the classes in reality, then **it might get a wrong class label** after classification.
- You have to select a lot of good examples from each class while you are training the classifier. If you consider the classification of **big data** that can be a real challenge.
- Training needs a lot of **computation time**, so do the classification.
- You might need to use a cloud and leave the training algorithm work overnight or nights before obtaining a good decision boundary model.

2.3 Regression Analysis

- Relationship between a dependent (target) and independent variable (s) (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables. For example, relationship between rash driving and number of road accidents by a driver is best studied through regression.

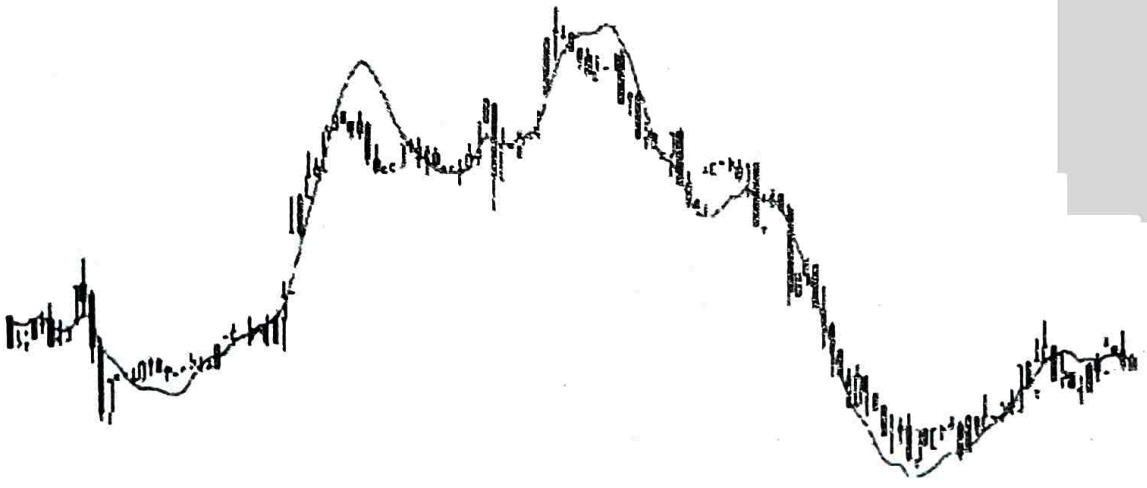


Fig. 2.3.1

- Regression analysis is an important tool for modelling and analyzing data. Here, we fit a curve / line to the data points, in such a manner that the differences between the distances of data points from the curve or line is minimized.
- As mentioned above, regression analysis estimates the relationship between two or more variables. Let's understand this with an easy example:
- Let's say, you want to estimate growth in sales of a company based on current economic conditions. You have the recent company data which indicates that the growth in sales is around two and a half times the growth in the economy. Using this insight, we can predict future sales of the company based on current & past information.
- There are multiple benefits of using regression analysis. They are as follows :
  1. It indicates the **significant relationships** between dependent variable and independent variable.
  2. It indicates the **strength of impact** of multiple independent variables on a dependent variable.
- Regression analysis also allows us to compare the effects of variables measured on different scales, such as the effect of price changes and the number of promotional activities. These benefits help market researchers / data analysts / data scientists to eliminate and evaluate the best set of variables to be used for building predictive models.

2.3.1 Types of Regression Techniques

- There are various kinds of regression techniques available to make predictions. These techniques are mostly driven by three metrics (number of independent variables, type of dependent variables and shape of regression line). We'll discuss them in detail in the following sections.

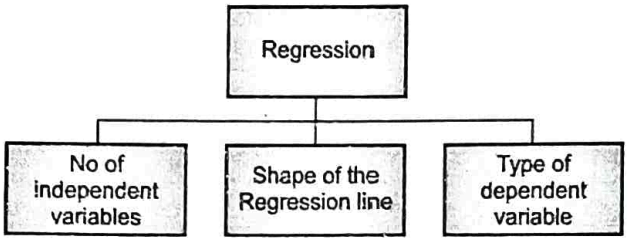


Fig. 2.3.2

- For the creative ones, you can even cook up new regressions, if you feel the need to use a combination of the parameters above, which people haven't used before. But before you start that, let us understand the most commonly used regressions.

1. Linear Regression

- It is one of the most widely known modeling technique. Linear regression is usually among the first few topics which people pick while learning predictive modeling. In this technique, the dependent variable is continuous, independent variable(s) can be continuous or discrete, and nature of regression line is linear.
- Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as regression line).
- It is represented by an equation Y = a + b * X + e, where a is intercept, b is slope of the line and e is error term. This equation can be used to predict the value of target variable based on given predictor variable(s).

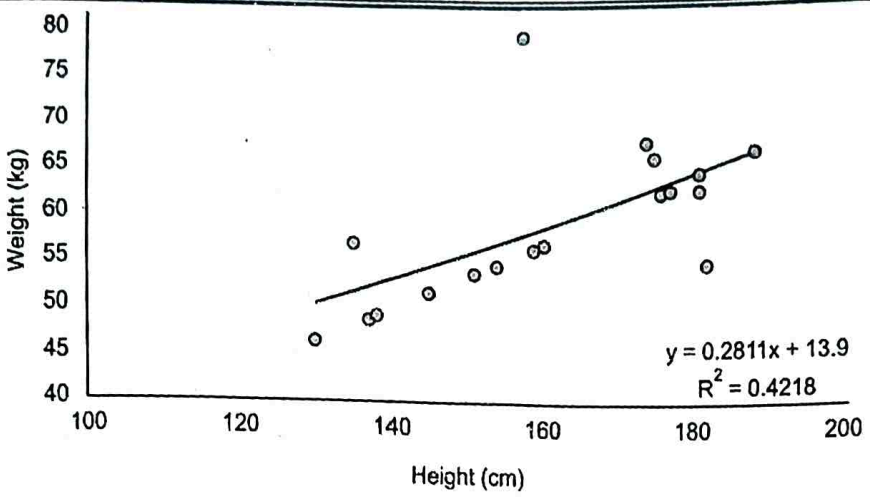


Fig. 2.3.3 : Relation B/w weight and height

- The difference between simple linear regression and multiple linear regression is that, multiple linear regression has (>1) independent variables, whereas simple linear regression has only 1 independent variable. Now, the question is "How do we obtain best fit line?".
- How to obtain best fit line (Value of a and b)?
- This task can be easily accomplished by Least Square Method. It is the most common method used for fitting a regression line. It calculates the best-fit line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line. Because the deviations are first squared, when added, there is no cancelling out between positive and negative values.

$$\min_{w} || Xw - y ||_2^2$$

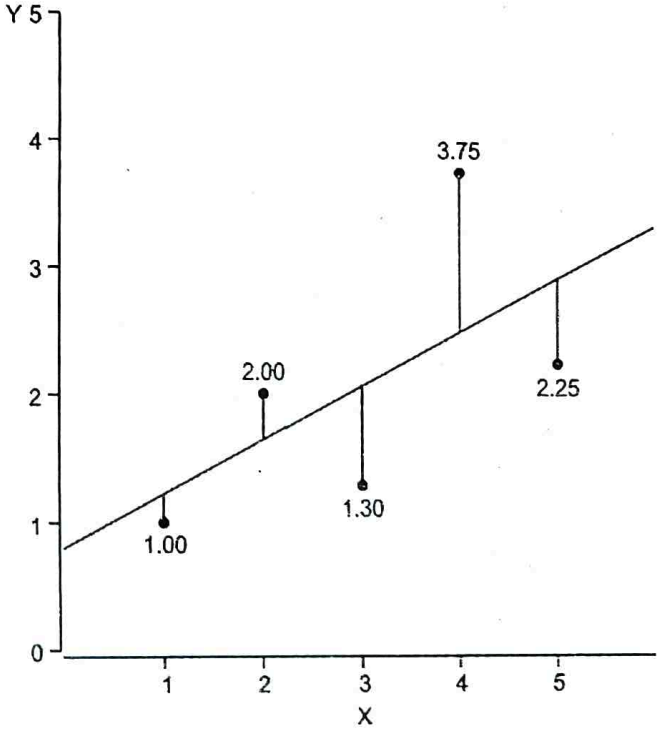


Fig. 2.3.4

- We can evaluate the model performance using the metric R-square.

Important Points :

- There must be linear relationship between independent and dependent variables
- Multiple regression suffers from **multicollinearity, autocorrelation, heteroskedasticity.**
- Linear Regression is very sensitive to Outliers. It can terribly affect the regression line and eventually the forecasted values.
- Multicollinearity can increase the variance of the coefficient estimates and make the estimates very sensitive to minor changes in the model. The result is that the coefficient estimates are unstable
- In case of multiple independent variables, we can go with **forward selection, backward elimination** and **step wise approach** for selection of most significant independent variables.

2. Logistic Regression

- Logistic regression is used to find the probability of event=Success and event=Failure. We should use logistic regression when the dependent variable is binary (0/ 1, True/ False, Yes/ No) in nature. Here the value of Y ranges from 0 to 1 and it can represented by following equation.

$$\text{odds} = p/(1-p) = \text{probability of event occurrence / probability of not event occurrence}$$

$$\ln(\text{odds}) = \ln(p/(1-p))$$

$$\text{logit}(p) = \ln(p/(1-p)) = b0 + b1X1 + b2X2 + b3X3.... + bkXk$$

- Above, p is the probability of presence of the characteristic of interest. A question that you should ask here is "why have we used log in the equation?".
- Since we are working here with a binomial distribution (dependent variable), we need to choose a link function which is best suited for this distribution. And, it is **logit** function. In the equation above, the parameters are chosen to maximize the likelihood of observing the sample values rather than minimizing the sum of squared errors (like in ordinary regression).

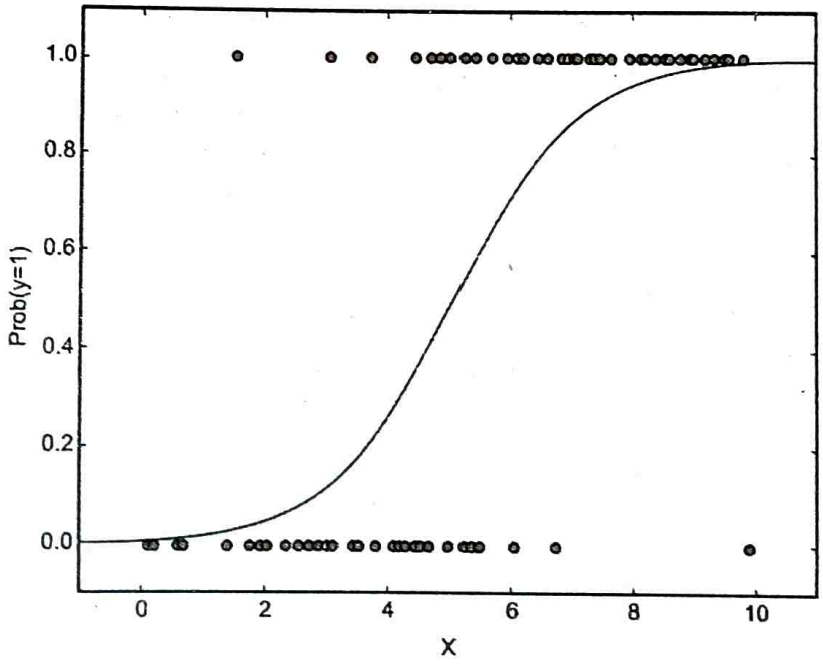


Fig. 2.3.5

Important Points :

- Logistic regression is widely used for **classification problems**
- Logistic regression doesn't require linear relationship between dependent and independent variables. It can handle various types of relationships because it applies a non-linear log transformation to the predicted odds ratio
- To avoid over fitting and under fitting, we should include all significant variables. A good approach to ensure this practice is to use a step wise method to estimate the logistic regression
- It requires **large sample sizes** because maximum likelihood estimates are less powerful at low sample sizes than ordinary least square
- The independent variables should not be correlated with each other i.e. **no multi collinearity.** However, we have the options to include interaction effects of categorical variables in the analysis and in the model.
- If the values of dependent variable is ordinal, then it is called as **Ordinal logistic regression**
- If dependent variable is multi class then it is known as **Multinomial Logistic regression.**

3. Polynomial Regression

- A regression equation is a polynomial regression equation if the power of independent variable is more than 1. The equation below represents a polynomial equation :

  $$y = a + b * x^2$$

- In this regression technique, the best fit line is not a straight line. It is rather a curve that fits into the data points.

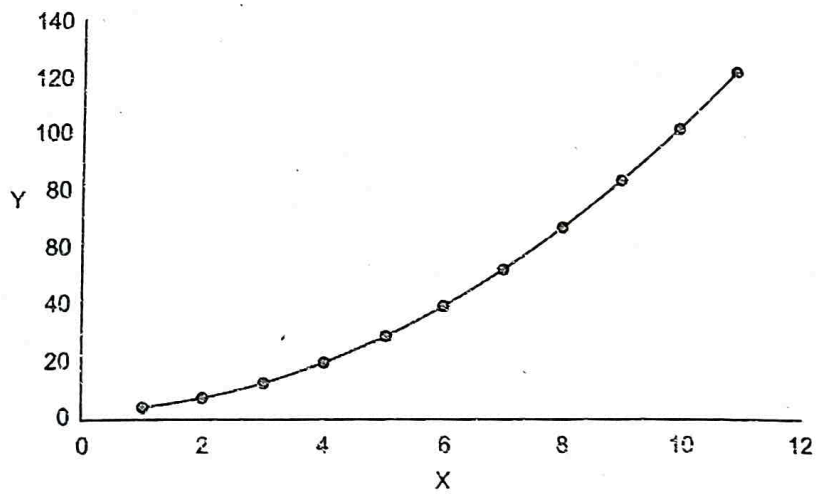


Fig. 2.3.6

Important Points :

- While there might be a temptation to fit a higher degree polynomial to get lower error, this can result in over-fitting. Always plot the relationships to see the fit and focus on making sure that the curve fits the nature of the problem. Here is an example of how plotting can help :

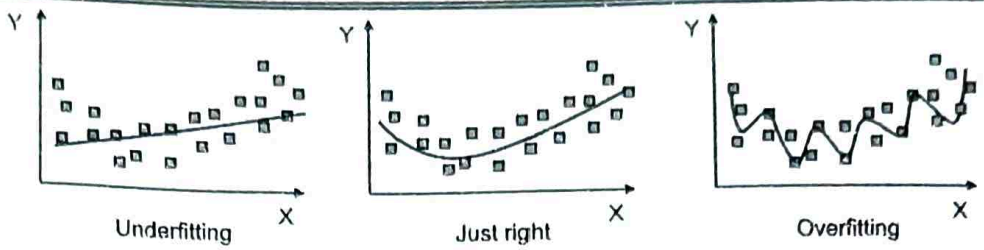


Fig. 2.3.7

- Especially look out for curve towards the ends and see whether those shapes and trends make sense. Higher polynomials can end up producing wierd results on extrapolation.

4. Stepwise Regression

- This form of regression is used when we deal with multiple independent variables. In this technique, the selection of independent variables is done with the help of an automatic process, which involves *no* human intervention.
- This feat is achieved by observing statistical values like R-square, t-stats and AIC metric to discern significant variables. Stepwise regression basically fits the regression model by adding/dropping co-variates one at a time based on a specified criterion. Some of the most commonly used Stepwise regression methods are listed below :
  - Standard stepwise regression does two things. It adds and removes predictors as needed for each step.
  - Forward selection starts with most significant predictor in the model and adds variable for each step.
  - Backward elimination starts with all predictors in the model and removes the least significant variable for each step.
- The aim of this modeling technique is to maximize the prediction power with minimum number of predictor variables. It is one of the method to handle higher dimensionality of data set.

2.4 Model Evaluation and Selection

- Model Selection and Evaluation is a hugely important procedure in the machine learning workflow. This is the section of our workflow in which we will analyse our model.
- We look at more insightful statistics of its performance and decide what actions to take in order to improve this model.
- Model selection in machine learning is the process of choosing the most suitable model or algorithm from a set of candidates to address a specific problem.
- It involves evaluating and comparing different models based on their performance and selecting the one that achieves the highest accuracy or predictive power.
- The choice of model is critical as different models have varying levels of complexity, underlying assumptions, and capabilities.
- The aim is to find a model that fits the training data well and generalizes effectively to new data. A model that is too simple might underfit the data and perform poorly in prediction, while a model that is too complex may overfit the data and fail to generalize.

- The goal is to select a model that not only performs well on the training data but also generalizes effectively to new, unseen data. Here's a comprehensive overview :

2.4.1 Model Evaluation

- **Performance Metrics** : Depending on the nature of your problem (classification, regression, etc.), different metrics are used:
- **Classification** : Accuracy, Precision, Recall, F1-Score, ROC-AUC, etc.
- **Regression** : Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), R^2 (Coefficient of Determination), etc.
- **Confusion Matrix** : Particularly for classification problems, it helps in understanding the type of errors made by the model (False Positives and False Negatives).
- **ROC Curve and AUC** : Useful for evaluating the performance of a classification model at various thresholds settings.
- **Error Analysis** : Involves looking at the specific instances where the model performed poorly to understand the underlying reasons.
- **Cross-Validation** : Techniques like k-fold or stratified k-fold cross-validation are used to ensure that the model's performance is consistent across different subsets of the dataset.

Model Selection

1. **Experiment with Multiple Models** : Start with a variety of models to understand which ones perform best for your specific dataset. This can include simple linear models to more complex ones like ensemble methods and neural networks.
2. **Feature Importance and Selection** : Analyze which features are contributing most to the model's predictions. Removing irrelevant or less important features can improve model performance.
3. **Hyperparameter Tuning** : Use techniques like Grid Search, Random Search, or Bayesian Optimization to find the optimal set of hyperparameters for your models.
4. **Model Validation** : Validate the model on a separate validation set that wasn't used during training. This step is crucial to check for overfitting.
5. **Learning Curves** : Analyzing learning curves can help in understanding if the model is benefiting from more data (underfitting) or suffering due to increased complexity (overfitting).
6. **Cost-Benefit Analysis** : Sometimes, the best model might not be the one with the highest accuracy, but the one that offers the best trade-off between performance and cost (like computational resources, time, etc.).

Considerations in Model Selection

1. **Bias-Variance Tradeoff** : Balancing the trade-off between underfitting (high bias) and overfitting (high variance).
2. **Interpretability vs Performance** : Sometimes, more complex models are less interpretable. Depending on the application, you might prefer a slightly less accurate model if it's more interpretable.
3. **Computational Efficiency** : For real-time applications, the computational complexity of the model is a crucial factor.

4. **Robustness and Generalizability** : The model should perform consistently across various sets of data and under different conditions.

Tools and Frameworks

1. **Scikit-learn** : Widely used in Python for model evaluation and selection, offering a variety of tools and metrics.
2. **TensorFlow and PyTorch** : For more complex models, especially deep learning.
3. **Automated Machine Learning (AutoML)** : Tools like AutoML offer automated solutions for model selection and hyperparameter tuning.
4. Model evaluation and selection is an iterative and sometimes subjective process, depending on the specific requirements and constraints of your project or application.

2.4.2 Model Performance Metrics

a) **Accuracy**

- **Definition** : The ratio of correctly predicted observations to the total observations.
- **Formula** : Accuracy = (True Positives + True Negatives) / Total Observations
- **Use Case** : Good for balanced datasets.

b) **Precision**

- **Definition** : The ratio of correctly predicted positive observations to the total predicted positives.
- **Formula** : Precision = True Positives / (True Positives + False Positives)
- **Use Case** : Important when the cost of False Positives is high (e.g., spam detection).

c) **Recall (Sensitivity)**

- **Definition** : The ratio of correctly predicted positive observations to all observations in the actual class.
- **Formula** : Recall = True Positives / (True Positives + False Negatives)
- **Use Case** : Critical when the cost of False Negatives is high (e.g., disease diagnosis).

d) **F1-Score**

- **Definition** : The weighted average of Precision and Recall.
- **Formula** : F1 Score = 2 * (Precision * Recall) / (Precision + Recall)
- **Use Case** : Useful when seeking a balance between Precision and Recall, and there's an uneven class distribution.

e) **Confusion Matrix**

- **Description** : A table layout that visualizes the performance of an algorithm.
- **True Positives (TP)** : Correctly predicted positive observations.
- **True Negatives (TN)** : Correctly predicted negative observations.
- **False Positives (FP)** : Incorrectly predicted positive observations.
- **False Negatives (FN)** : Incorrectly predicted negative observations.

f) ROC Curve and AUC

- **ROC Curve** : Plots the True Positive Rate (Recall) against the False Positive Rate.
- **AUC (Area Under Curve)** : Represents degree or measure of separability by the model.
- **Steps :**
  - Calculate TPR and FPR for various threshold values.
  - Plot TPR vs. FPR.

2.4.3 Hyperparameter Tuning and Model Selection

- Hyperparameter Tuning
- **Description** : Process of selecting the set of optimal hyperparameters for a learning algorithm.

Methods :

- **Grid Search** : Exhaustive search over a specified parameter grid.
- **Random Search** : Random exploration of parameters.
- **Bayesian Optimization** : Uses probability to find the best parameters.

2.4.4 Confusion Matrix

A confusion matrix is an N X N matrix, where N is the number of predicted classes. For the problem in hand, we have N = 2, and hence we get a 2 X 2 matrix. It is a performance measurement for machine learning classification problems where the output can be two or more classes. Confusion matrix is a table with 4 different combinations of predicted and actual values. It is extremely useful for measuring precision-recall, Specificity, Accuracy, and most importantly, AUC-ROC curves.

Here are a few definitions you need to remember for a confusion matrix :

- **True Positive :** You predicted positive, and it's true.
- **True Negative :** You predicted negative, and it's true.
- **False Positive : (Type 1 Error) :** You predicted positive, and it's false.
- **False Negative : (Type 2 Error) :** You predicted negative, and it's false.
- **Accuracy :** The proportion of the total number of correct predictions that were correct.
- **Positive Predictive Value or Precision :** The proportion of positive cases that were correctly i
- **Negative Predictive Value :** The proportion of negative cases that were correctly identified.
- **Sensitivity or Recall :** The proportion of actual positive cases which are correctly identified.
- **Specificity :** The proportion of actual negative cases which are correctly identified.
- **Rate :** It is a measuring factor in a confusion matrix. It has also 4 types TPR, FPR, TNR, and FNR.

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	Positive predictive value	a/(a + b)
	Negative	c	d	Negative predictive value	d/(c + d)
		Sensitivity	Specificity	Accuracy = (a + d)/(a + b + c + d)	
		a/(a + c)	d/(b + d)		

Count of ID	Target		
Model	1	0	Grand Total	
1	3,834	639	4,473	85.7%
0	16	951	967	1.7%
Grand Total	3,850	1,590	5,440	
	99.6%	40.19%		88.0%

- The accuracy for the problem in hand comes out to be 88%. As you can see from the above two tables, the Positive Predictive Value is high, but the negative predictive value is quite low. The same holds for Sensitivity and Specificity. This is primarily driven by the threshold value we have chosen. If we decrease our threshold value, the two pairs of starkly different numbers will come closer.
- In general, we are concerned with one of the above-defined metrics. For instance, in a pharmaceutical company, they will be more concerned with a minimal wrong positive diagnosis. Hence, they will be more concerned about high Specificity. On the other hand, an attrition model will be more concerned with Sensitivity. Confusion matrices are generally used only with class output models.

2.4.5 F1 Score

- In the last section, we discussed precision and recall for classification problems and also highlighted the importance of choosing a precision/recall basis for our use case. What if, for a use case, we are trying to get the best precision and recall at the same time? F1-Score is the harmonic mean of precision and recall values for a classification problem. The formula for F1-Score is as follows :

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2}\right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- Now, an obvious question that comes to mind is why you are taking a harmonic mean and not an arithmetic mean. This is because HM punishes extreme values more. Let us understand this with an example. We have a binary classification model with the following results :

  Precision: 0, Recall: 1

- Here, if we take the arithmetic mean, we get 0.5. It is clear that the above result comes from a dumb classifier that ignores the input and predicts one of the classes as output. Now, if we were to take HM, we would get 0 which is accurate as this model is useless for all purposes.

- This seems simple. There are situations, however, for which a data scientist would like to give a percentage more importance/weight to either precision or recall. Altering the above expression a bit such that we can include an adjustable parameter beta for this purpose, we get :

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

- Fbeta measures the effectiveness of a model with respect to a user who attaches β times as much importance to recall as precision.

2.4.6 Cross Validation

- Let's first understand the importance of cross-validation. Due to my busy schedule these days, I don't get much time to participate in data science competitions. A long time back, I participated in TFI Competition on Kaggle. Without delving into my competition performance, I would like to show you the dissimilarity between my public and private leaderboard scores.

Here is an Example of Scoring on Kaggle!

- For the TFI competition, the following were three of my solution and scores (the lesser, the better) :

Submission	Files	Public Score	Public Score	Selected?
Mon, 04 May 2015 12:59:31 Edit description	submissio n_all_wit h_sai3.csv	1649776.86428	1809956.02878	☑
Mon, 04 May 2015 11:48:54 Edit description	submissio n_all_wit	1649776.86428	1809956.02878	☑
Mon, 13 Apr 2015 13:28:08 Edit description	submissio n_all.csv	1677138.71291	1795007.23155	☑

- You will notice that the third entry which has the worst Public score turned out to be the best model on Private ranking. There were more than 20 models above the "submission_all.csv", but I still chose "submission_all.csv" as my final entry (which really worked out well). What caused this phenomenon? The dissimilarity in my public and private leaderboards is caused by over-fitting.
- Over-fitting is nothing, but when your model becomes highly complex that it starts capturing noise, also. This 'noise' adds no value to the model but only inaccuracy.
- In the following section, I will discuss how you can know if a solution is an over-fit or not before we actually know the test set results.

The Concept of Cross-Validation

- Cross Validation is one of the most important concepts in any type of data modeling. It simply says, try to leave a sample on which you do not train the model and test the model on this sample before finalizing the model.

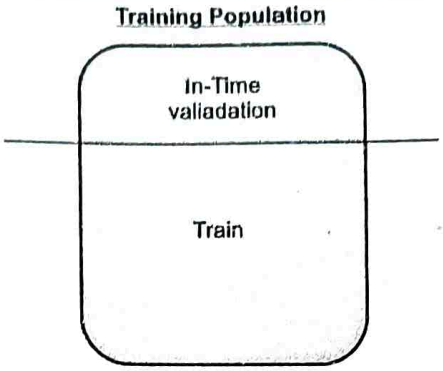


Fig. 2.4.1

- The above diagram shows how to validate the model with the in-time sample. We simply divide the population into 2 samples and build a model on one sample. The rest of the population is used for in-time validation.

(i) K-Fold Cross-Validation

- Let's extrapolate the last example to k-fold from 2-fold cross-validation. Now, we will try to visualize how a k-fold validation work.

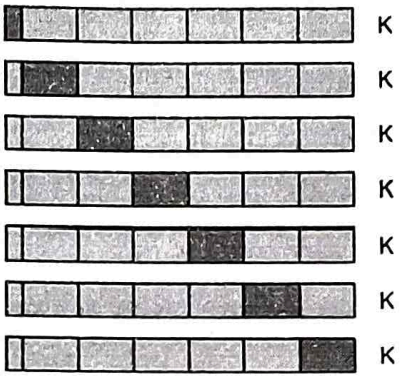


Fig. 2.4.2

- This is a 7-fold cross-validation.
- Here's what goes on behind the scene: we divide the entire population into 7 equal samples. Now we train models on 6 samples (Green boxes) and validate on 1 sample (grey box). Then, at the second iteration, we train the model with a different sample held as validation. In 7 iterations, we have basically built a model on each sample and held each of them as validation. This is a way to reduce the selection bias and reduce the variance in prediction power. Once we have all 7 models, we take an average of the error terms to find which of the models is best.

ii) Stratified k-fold cross-validation

- This technique is similar to k-fold cross-validation with some little changes. This approach works on stratification concept, it is a process of rearranging the data to ensure that each fold or group is a good representative of the complete dataset. To deal with the bias and variance, it is one of the best approaches.
- It can be understood with an example of housing prices, such that the price of some houses can be much high than other houses. To tackle such situations, a stratified k-fold cross-validation technique is useful.

iii) Holdout Method

- This method is the simplest cross-validation technique among all. In this method, we need to remove a subset of the training data and use it to get prediction results by training it on the rest part of the dataset.
- The error that occurs in this process tells how well our model will perform with the unknown dataset. Although this approach is simple to perform, it still faces the issue of high variance, and it also produces misleading results sometimes.

iv) Hyper parameter Tuning

- Hyper parameter tuning is the process of tuning the parameters present as the tuples while we build machine learning models. These parameters are defined by us which can be manipulated according to programmer wish. Machine learning algorithms never learn these parameters. These are tuned so that we could get good performance by the model.
- A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.
- However, there is another kind of parameters, known as ***Hyper parameters***, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.
- Some examples of model hyper parameters include :
  - The penalty in Logistic Regression Classifier i.e. L1 or L2 regularization
  - The learning rate for training a neural network.
  - The C and sigma hyper parameters for support vector machines.
  - The k in k-nearest neighbors.
  - The aim of this article is to explore various strategies to tune hyper parameter for Machine learning model.
  - Models can have many hyper parameters and finding the best combination of parameters can be treated as a search problem. Two best strategies for Hyper parameter tuning are :

a) Grid Search CV

- In Grid Search CV approach, machine learning model is evaluated for a range of hyper parameter values. This approach is called Grid Search CV, because it searches for best set of hyper parameters from a grid of hyper parameters values.
- For example, if we want to set two hyper parameters C and Alpha of Logistic Regression Classifier model, with different set of values. The grid search technique will construct many versions of the model with all possible combinations of hyper parameters, and will return the best one.
- As in the image, for C = [0.1, 0.2, 0.3, 0.4, 0.5] and Alpha = [0.1, 0.2, 0.3, 0.4]. For a combination ***C = 0.3 and Alpha = 0.2***, performance score comes out to be 0.726(Highest), therefore it is selected.

C \ Alpha	0.1	0.2	0.3	0.4
0.5	0.701	0.703	0.697	0.696
0.4	0.699	0.702	0.698	0.702
0.3	0.721	0.726	0.713	0.703
0.2	0.706	0.705	0.704	0.701
0.1	0.698	0.692	0.688	0.675

Fig. 2.4.3

- ***Drawback*** : Grid Search CV will go through all the intermediate combinations of hyper parameters which makes grid search computationally very expensive.

b) Randomized Search CV

Randomized Search CV solves the drawbacks of Grid Search CV, as it goes through only a fixed number of hyper parameter settings. It moves within the grid in random fashion to find the best set hyper parameters. This approach reduces unnecessary computation.

2.5 Machine Learning Algorithms

Machine learning algorithms are sophisticated computational models enabling computers to recognize patterns and make predictions or decisions based on data, eliminating the need for explicit programming. They constitute the core of contemporary artificial intelligence and have a broad spectrum of applications. These include image and speech recognition, natural language processing, recommendation systems, fraud detection, and autonomous vehicles, among others. These algorithms empower machines to improve their performance and decision-making capabilities through experience, mirroring human learning to a certain extent. Their versatility and adaptability make them integral to the advancement of various technology-driven fields.

2.5.1 Decision Trees

- Decision Tree is a supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

Note : A decision tree can contain categorical data (YES/NO) as well as numeric data.

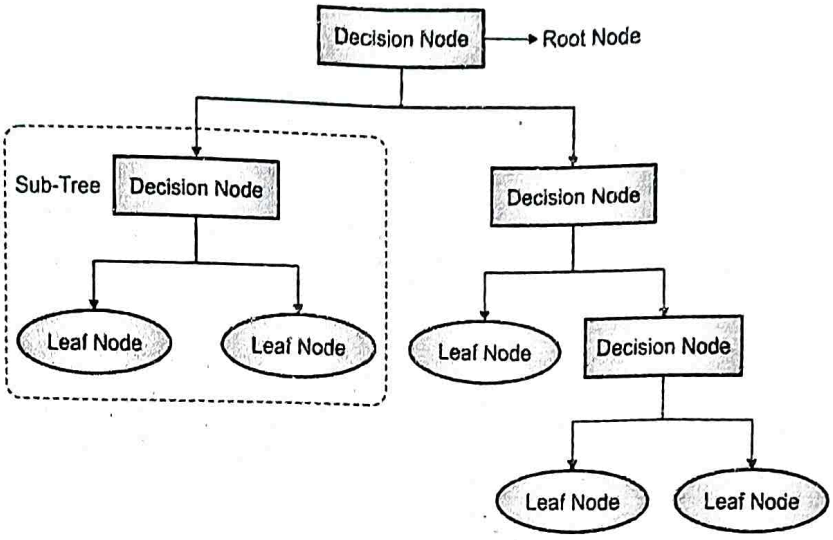


Fig. 2.5.1

Why use Decision Trees?

- There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree :
  - Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
  - The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

- **Root Node :** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node :** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting :** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree :** A tree formed by splitting the tree.
- **Pruning :** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node :** The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work?

- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

- For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm :
  - **Step 1 :** Begin the tree with the root node, says S, which contains the complete dataset.
  - **Step 2 :** Find the best attribute in the dataset using Attribute Selection Measure (ASM).
  - **Step 3 :** Divide the S into subsets that contains possible values for the best attributes.
  - **Step 4 :** Generate the decision tree node, which contains the best attribute.
  - **Step 5 :** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.
- **Example :** Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram :

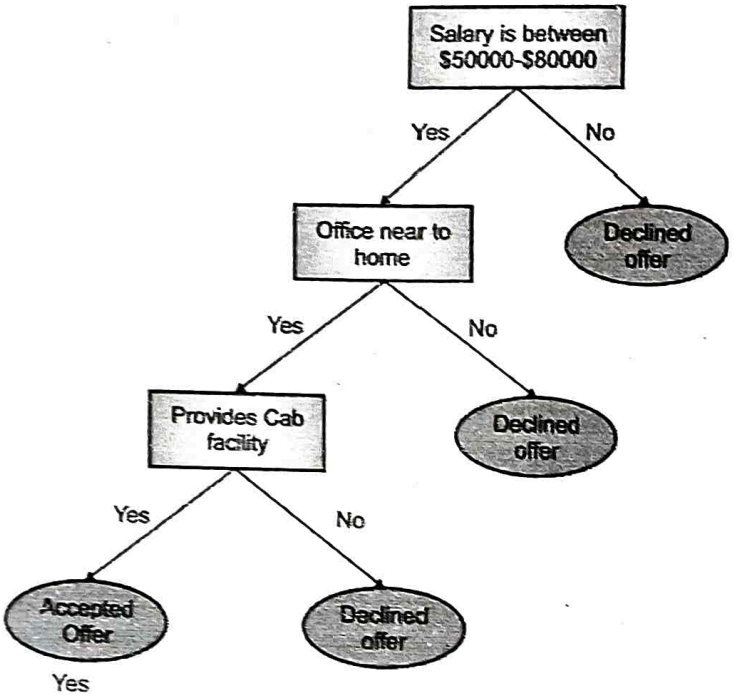


Fig. 2.5.2

Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are :

1. Information Gain
2. Gini Index

1. Information Gain :

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

$$\text{Information Gain} = \text{Entropy(S)} - [(\text{Weighted Avg}) * \text{Entropy(each feature)}]$$

- **Entropy :** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as :

$$\text{Entropy(s)} = -P(\text{yes})\log_2 P(\text{yes}) - P(\text{no}) \log_2 P(\text{no})$$

Where,

S = Total number of samples

P(yes) = probability of yes

P(no) = probability of no

2. Gini Index :

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

- **Pruning :** Getting an Optimal Decision tree

  Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

- A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree pruning technology used :

  i) Cost Complexity Pruning

  ii) Reduced Error Pruning.

- **Advantages of the Decision Tree**
  - It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
  - It can be very useful for solving decision-related problems.
  - It helps to think about all the possible outcomes for a problem.
  - There is less requirement of data cleaning compared to other algorithms.

- **Disadvantages of the Decision Tree**
  - The decision tree contains lots of layers, which makes it complex.
  - It may have an overfitting issue, which can be resolved using the Random Forest algorithm.
  - For more class labels, the computational complexity of the decision tree may increase.

2.5.2 Random Forest

Random Forest is a powerful machine learning algorithm that falls under the category of supervised learning. It can be effectively utilized for both classification and regression tasks. The algorithm is based on the concept of ensemble learning, a technique where multiple classifiers (in this case, decision trees) are combined to tackle complex problems and enhance the model's performance.

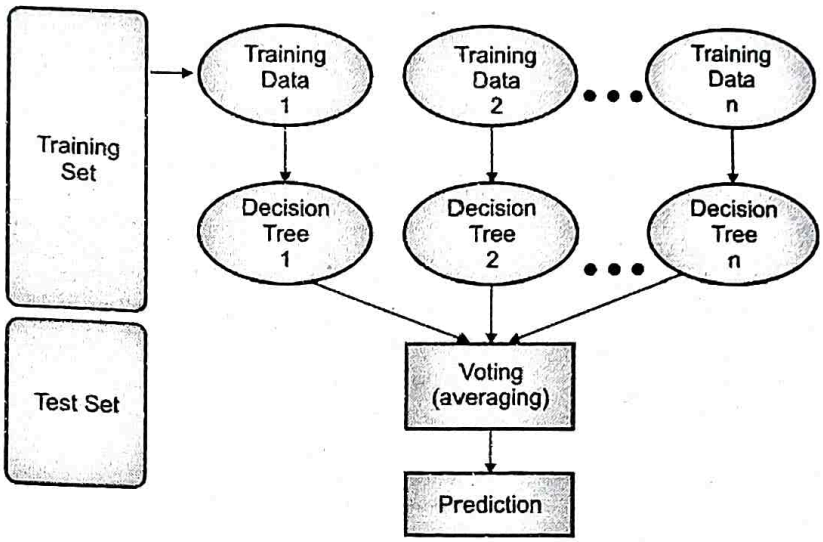


Fig. 2.5.3

Here's a more refined explanation of the Random Forest algorithm :

Concept :

- **Random Forest** operates by constructing numerous decision trees on different subsets of the given dataset and averages the results to boost the dataset's predictive accuracy.
- Unlike relying on a single decision tree, Random Forest makes predictions based on the majority vote from all trees. This approach improves accuracy and helps prevent overfitting.

Key Features :

- **Higher Accuracy** : The more trees in the forest, generally, the higher the accuracy and robustness of the model.
- **Overfitting Prevention** : The ensemble approach reduces the risk of overfitting compared to using a single decision tree.

Working of Random Forest :

- **Random Selection** : Randomly select 'K' data points from the training set.
- **Build Decision Trees** : Construct decision trees associated with these data points (subsets).

- **Choose Number of Trees** : Decide on 'N', the number of decision trees to build.
- **Repeat Steps** : Execute steps 1 and 2 multiple times.
- **Majority Voting for Prediction** : For a new data point, each tree gives a prediction. The final output is determined by the majority vote among all trees.

Example Use-Case :

Consider a dataset containing various fruit images. The Random Forest classifier splits this dataset into subsets, each fed into a different decision tree. During training, each tree predicts an outcome. For a new fruit image, the classifier takes the majority decision from all trees to make the final prediction.

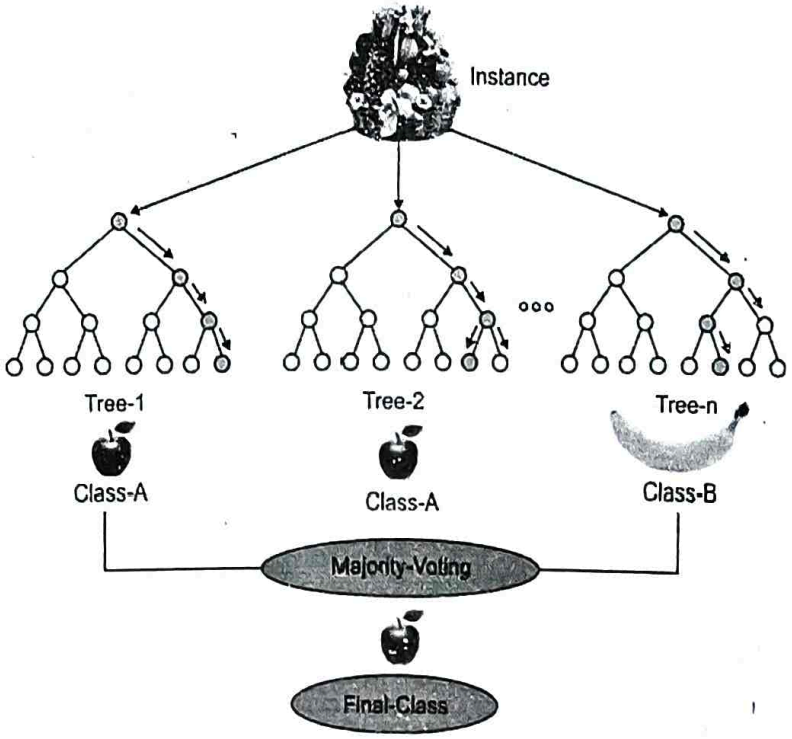


Fig. 2.5.4

Applications of Random Forest

- **Banking** : Identifying loan risks.
- **Medicine** : Detecting disease trends and assessing risks.
- **Land Use** : Identifying areas of similar land use.
- **Marketing** : Determining market trends and consumer preferences.

Advantages

- Capable of both classification and regression.
- Handles large datasets and high dimensionality efficiently.
- Improves accuracy and mitigates overfitting issues.

Disadvantages

While Random Forest is versatile for classification and regression, it may be less optimal for regression tasks due to its nature of averaging results, which might not capture the nuances in continuous variable predictions.

2.5.3 Support Vector Machine

Support Vector Machine (SVM) is a versatile supervised learning algorithm predominantly used for classification problems, but it can also handle regression tasks. The essence of SVM is to determine the best decision boundary (hyperplane) that can segregate classes in an n-dimensional space, enhancing the model's ability to correctly classify new data points.

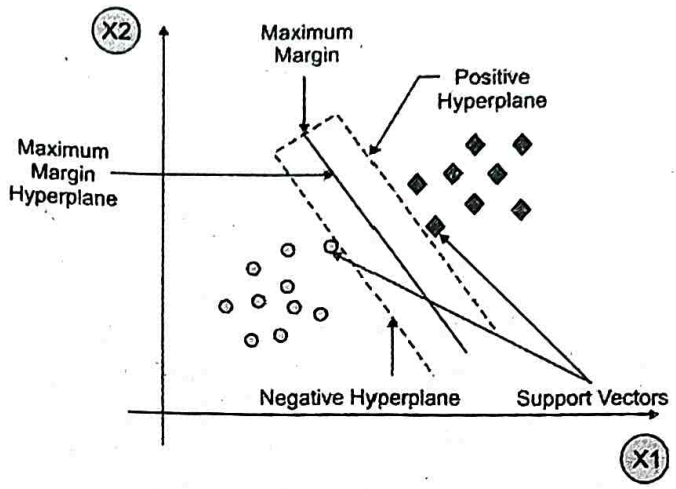


Fig. 2.5.5

Concept of SVM :

- **Hyperplane** : In SVM, the decision boundary is known as a hyperplane. This hyperplane is an n-dimensional line (n is the number of features) that separates different classes in the feature space. The goal is to find a hyperplane that best divides the classes with the maximum margin.
- **Support Vectors** : These are the data points nearest to the hyperplane, which influence the position and orientation of the hyperplane. These points essentially "support" the creation of the hyperplane, hence the name.
- **Margin** : It's the gap between the two lines on the closest class points. This is an area where no data points are present and is maximized in SVM.

Working of SVM :

- **Linear SVM :**
  - Linear SVM is used when the data is linearly separable, which means it can be separated using a straight line in 2D (or a hyperplane in higher dimensions).
  - The SVM algorithm finds this line by first computing the hyperplane that maximizes the margin between the classes. The data points closest to the hyperplane (the support vectors) guide its creation.
- **Non-Linear SVM :**
  - Non-linear SVM is applied when data is not linearly separable.
  - It uses a technique called the kernel trick to transform the data into a higher dimension where a hyperplane can be used to separate the data.
  - For example, it might add a new dimension (z) calculated from the existing features (e.g., $z = x^2 + y^2$ for two-dimensional data), which allows a non-linear separation in the original feature space.

Example :

Imagine a set of images of cats and dogs. An SVM model is trained on these images, learning to distinguish the features that separate cats from dogs. When presented with a new, unseen image (say, a cat with features similar to a dog), the SVM uses the hyperplane and support vectors it has established to classify the image either as a cat or a dog.

Applications :

- **Face Detection** : Identifying faces within images by classifying each image region as a face or non-face.
- **Image Classification** : Categorizing images into different groups.
- **Text Categorization** : Classifying text documents, such as emails or news articles, into different categories.

Types of SVM :

i) **Linear SVM** : Used when data is linearly separable.

ii) **Non-linear SVM** : Used when data cannot be separated by a straight line. This type often involves using kernel functions like polynomial, radial basis function (RBF), or sigmoid.

Advantages :

- Effective in high-dimensional spaces.
- Works well with clear margin of separation and is effective in cases where the number of dimensions is g than the number of samples.

Disadvantages :

- Not suitable for large datasets because of its high training time.
- Less effective on noisier datasets with overlapping classes.

SVM's ability to find complex relationships and its flexibility in handling both linear and non-linear data make it a powerful tool in the machine learning toolkit. The choice of kernel and regularization parameters can significantly affect the performance, requiring careful tuning for optimal results.

2.5.4 Artificial Neural Networks (ANN)

An Artificial Neural Network (ANN) is a computational model inspired by the way biological neural networks in the human brain process information. It is a key technology in the field of machine learning and artificial intelligence, capable of performing complex tasks such as pattern recognition, language processing, and image analysis.

An ANN is composed of a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection between artificial neurons can transmit a signal from one to another. The receiving neuron processes the signal and signals downstream neurons connected to it. Neurons are organized in layers – input, hidden, and output.

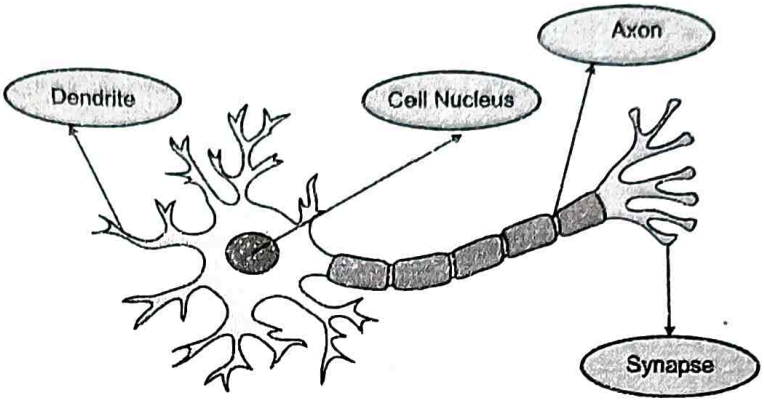


Fig. 2.5.6

a) Components of an ANN :

- **Input Layer** : Receives the input signals and passes them on to the next layer.
- **Hidden Layer(s)** : Intermediate layer(s) that perform computations and transfer information from the input nodes to the output nodes.
- **Output Layer** : Delivers the final output of the neural network.

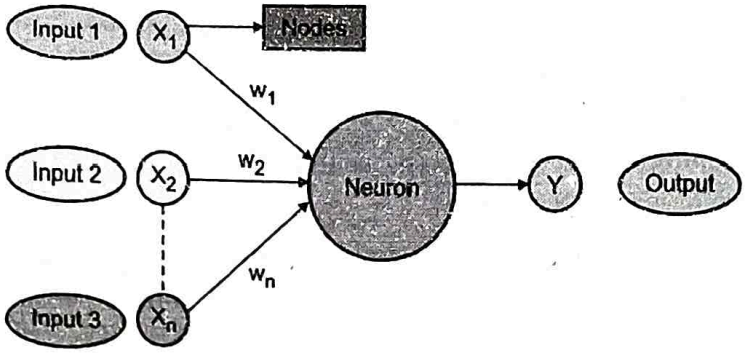


Fig. 2.5.7

b) Working of an ANN :

1. **Input Processing** : Inputs are received by the input layer, each input associated with a weight signifying its importance.
2. **Weighted Sum** : These inputs are multiplied by their respective weights and then summed.
3. **Adding Bias** : A bias (akin to an intercept in linear models) is usually added to the weighted sum to help the model in a way that it can better fit the data.
4. **Activation Function** : The result is passed through an activation function, which determines the neuron's output. Common activation functions include sigmoid, hyperbolic tangent (tanh), and Rectified Linear Unit (ReLU).
5. **Output Generation** : The process continues through the network until the output layer produces the final output.

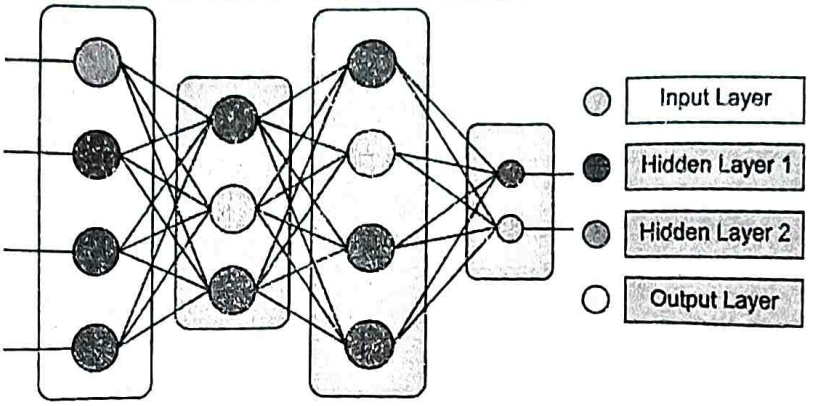


Fig. 2.5.8

c) **Types of ANN :**

i) **Feed forward Neural Networks** : The simplest type of ANN, where the data moves in one direction from input to output nodes.

ii) **Recurrent Neural Networks (RNNs)** : Designed for processing sequential data, the outputs from neurons can loop back into the network, creating a 'memory' of previous inputs.

iii) **Convolutional Neural Networks (CNNs)** : Primarily used in image recognition and processing, they ar structured to pick up on spatial hierarchies in data.

Applications of ANN :

1. **Image and Voice Recognition** : Used extensively in computer vision for tasks like image classification and face recognition, and in voice recognition systems.
2. **Natural Language Processing** : For translating languages, sentiment analysis, and text generation.
3. **Predictive Analytics** : In finance for predicting stock prices, in healthcare for predicting disease outbreaks or patient diagnosis, etc.

Advantages of ANN :

1. **Ability to Learn and Model Non-linear Relationships** : Crucial for complex problems where the relationship between inputs and outputs is not linear.
2. **Ability to Generalize** : After training, ANNs can make predictions on new, unseen data.
3. **Parallel Processing** : ANNs can perform more than one task simultaneously, making them efficient for complex problem solving.

Disadvantages of ANN :

1. **Opaque Nature (Black Box)** : ANNs often do not reveal how they reached a particular decision, which can be problematic in certain applications.
2. **Hardware Dependencies** : They require processors with parallel processing power, in line with their structure.
3. **Data and Computation Intensive** : ANNs require large amounts of data and computational power for training.

5.5 Ensemble Learning

1. Bagging (Bootstrap Aggregating)

Bagging is an ensemble technique primarily used to reduce the variance of our predictions by combining the result of multiple classifiers modelled on different sub-samples of the same data set. The key idea is to train the same algorithm multiple times using different subsets of the training data.

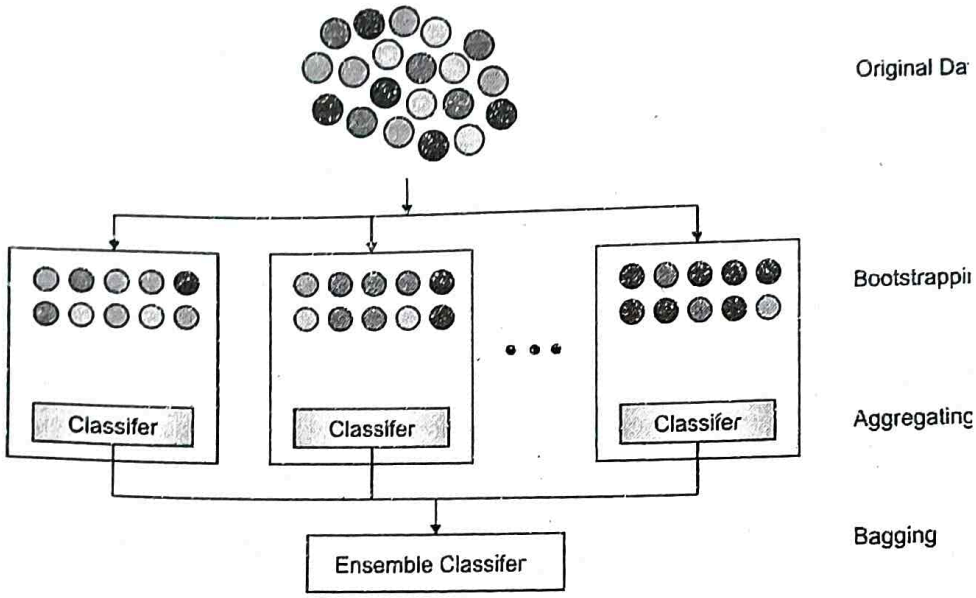


Fig. 2.5.9 : An illustration for the concept of bootstrap aggregating (Bagging)

How it Works?

- **Data Sampling** : Multiple subsets of the original dataset are created using bootstrap sampling (i.e., sampling with replacement).
- **Model Training** : Each subset is used to train a separate model. Importantly, all these models are of the same type.
- **Parallel Learning** : The models are learned independently and in parallel.
- **Aggregation** : The final output prediction is averaged (regression) or voted (classification) across all models.

Example : Random Forest

Random Forest is a classic example of Bagging, where numerous decision trees (high variance models) are combined. It introduces randomness by not only sampling the rows but also the features for splitting nodes.

2. Boosting

- Boosting is another ensemble technique designed to improve the accuracy of machine learning algorithms. It converts weak learners into strong learners.
- Unlike Bagging, Boosting focuses on reducing bias and builds models sequentially, where each model attempts to correct the errors of the previous one.

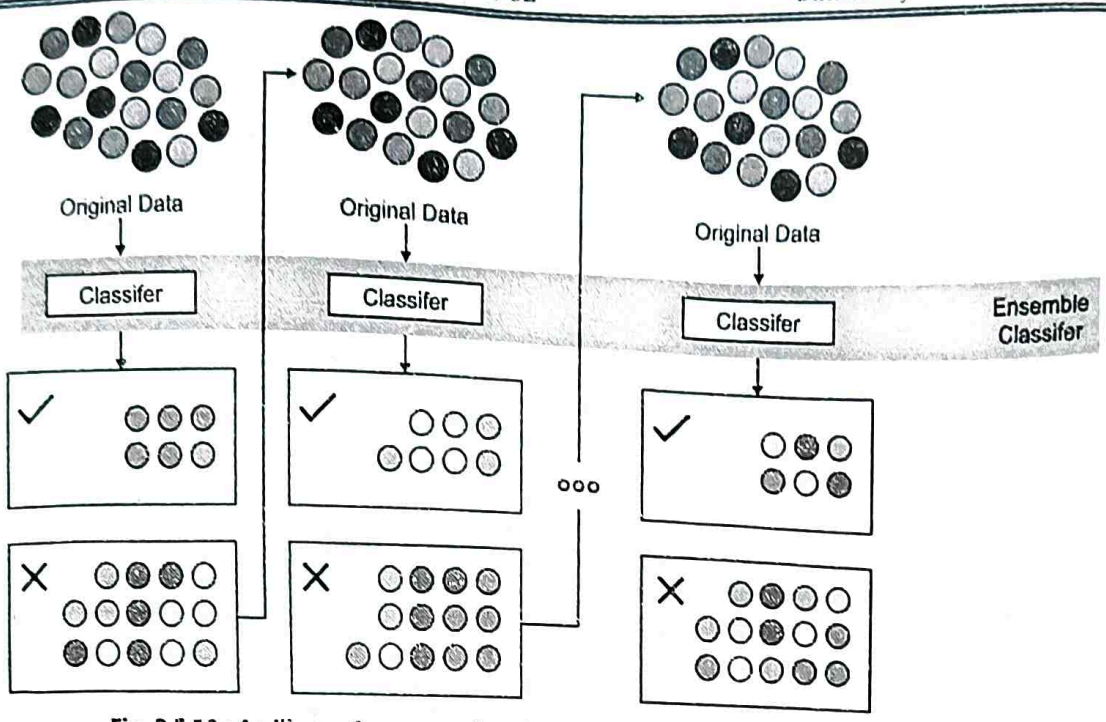


Fig. 2.5.10 : An illustration presenting the intuition behind the boosting algorithm, consisting of the parallel learners and weighted dataset.

How it Works?

- **Initial Model :** The first model is trained on the entire dataset.
- **Sequential Learning** : Each subsequent model is trained with a focus on accurately predicting the instances where the previous model performed poorly.
- **Weight Adjustment** : The weights of instances are adjusted according to the errors; the weight increases for misclassified instances and decreases for correctly classified instances.
- **Aggregation** : The final model is a weighted sum of all the sequential models.

Example: AdaBoost

AdaBoost (Adaptive Boosting) is a popular boosting variant. It focuses on classification problems and aims to convert a set of weak classifiers into a strong one.

Similarities between Bagging and Boosting :

- **Ensemble Nature** : Both methods use a collection of models to make final predictions.
- **Sampling Technique** : They employ a technique of sampling data sets and running algorithms on them.
- **Final Decision Making :** Decisions are made by averaging or voting among the individual models.
- **Stability and Accuracy** : Both methods aim to improve prediction stability and reduce variance, thus improving the accuracy over individual models.

Differences

- **Learning Technique** : Bagging models are independent of each other and learned in parallel, while Boosting models are learned sequentially.
- **Focus** : Bagging aims to reduce variance and is less sensitive to outliers, whereas Boosting focuses on reducing bias and is sensitive to outliers.

Model Weighting : In Boosting, models are weighted based on their performance, whereas in Bagging, each model has equal weight.

Model Type : Bagging often uses complex models (like fully grown decision trees), while Boosting uses simple models (weak learners).

Differences between Bagging and Boosting

Table 2.5.1

Sr. No.	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.
5.	Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.
6.	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
9	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

2.5.6 K-Nearest Neighbors Algorithm

The K-Nearest Neighbors (KNN) algorithm is indeed a fundamental yet powerful tool in machine learning for both classification and regression tasks. Let's break down your comprehensive overview to clarify and expand on some points.

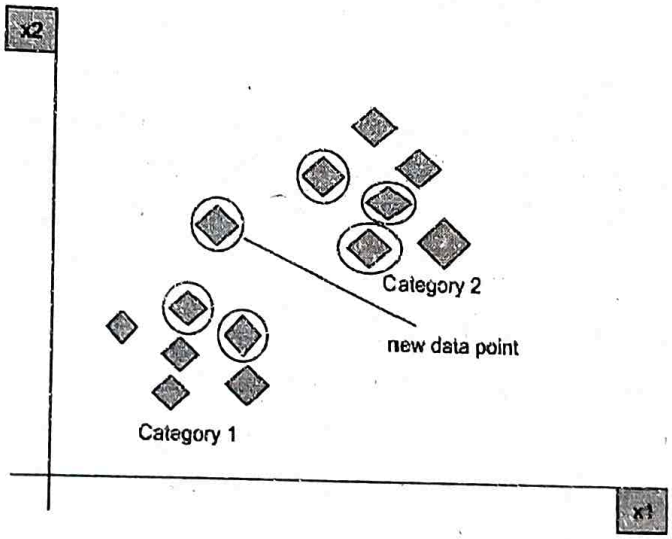


Fig. 2.5.11 : KNN algorithm working visualization

Intuition Behind KNN Algorithm

KNN operates on a simple principle : it classifies a data point based on how its neighbors are classified. This method assumes that similar data points can be found near each other. Hence, it looks at the K closest data points (neighbors) to determine the classification of a new point.

A) Distance Metrics Used in KNN

i) Euclidean Distance

This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{ij})^2}$$

ii) Manhattan Distance

Manhattan Distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

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

iii) Minkowski Distance

- We can say that the Euclidean, as well as the Manhattan distance, are special cases of the Minkowski distance.

$$d(x, y) = \left(\sum_{i=1}^{n} (x_i - y_i)^p\right)^{\frac{1}{p}}$$

- From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

  The above-discussed metrics are most common while dealing with a Machine Learning problem but there are other distance metrics as well like Hamming Distance which come in handy while dealing with problems that require overlapping comparisons between two vectors whose contents can be Boolean as well as string values.

Choosing the Value of K in KNN

Selecting the right K value is crucial. A small value of K makes the model sensitive to noise, while a large K makes it computationally expensive and might include points from other classes. Cross-validation is a common method to find an optimal K value. It's advisable to choose an odd number for K in binary classification to avoid ties.

B) Working of the KNN Algorithm

1. **Select K** : Choose the number of neighbors (K).
2. **Calculate Distance** : Compute the distance between the query instance and all the training samples.
3. **Find Nearest Neighbors** : Identify the K nearest neighbors from the calculated distances.
4. **Vote or Average** : For classification, take the majority vote of these K neighbors. For regression, compute the average of these K nearest neighbors.

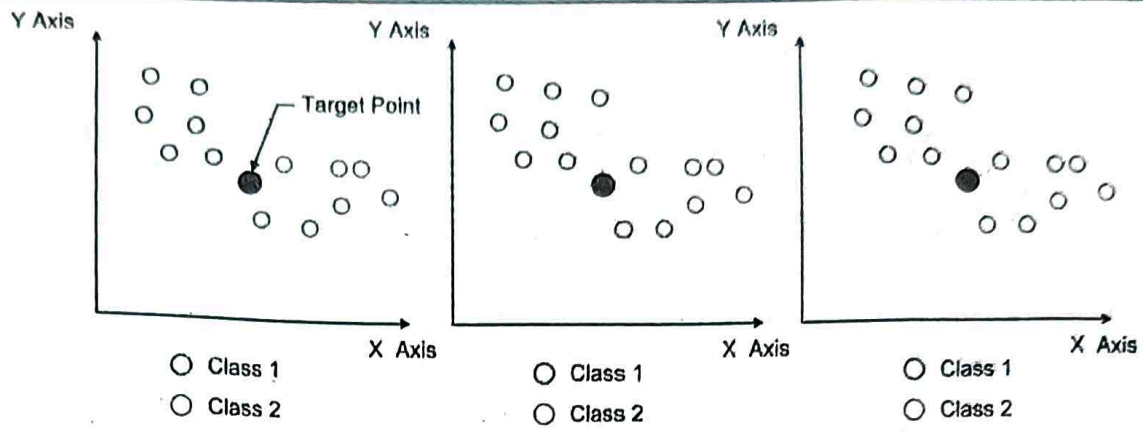


Fig. 2.5.12

Advantages of KNN

1. **Simplicity** : KNN is straightforward to understand and implement.
2. **Versatility** : It works well with both numerical and categorical data.
3. **Non-parametric** : KNN makes no assumptions about the underlying data distribution, which is useful in real-world scenarios.

Disadvantages of KNN

1. **Scalability** : KNN can be computationally expensive, especially with large datasets, as it stores all training data.
2. **Curse of Dimensionality** : Its performance degrades with the increase in the number of features due to the increased distance between data points.
3. **Sensitive to Imbalanced Data** : KNN can be biased towards the majority class in cases of imbalanced datasets.
4. **Need for Feature Scaling** : KNN is affected by the scale of the features, so proper feature scaling is essential.

2.5.7 Gradient Descent for Optimization

Gradient Descent is a widely used optimization algorithm for machine learning models. However, there are several optimization techniques that can be used to improve the performance of Gradient Descent.

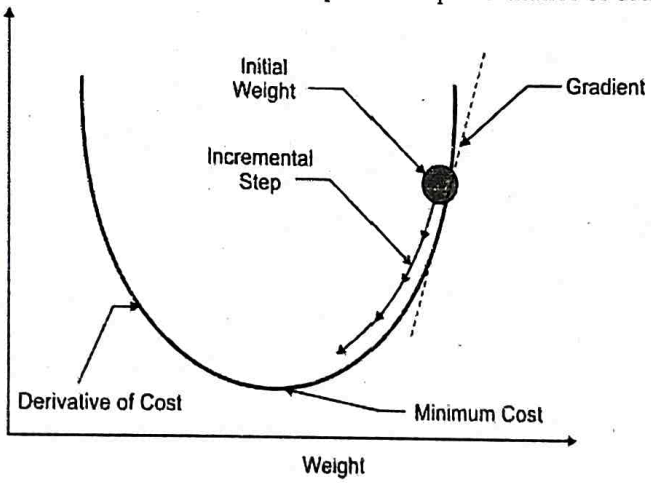


Fig. 2.5.13

1. **Gradient Descent - A Deeper Dive :**
   - **Historical Context :** Augustin-Louis Cauchy's introduction of this algorithm was a pivotal moment in optimization techniques. Recognizing its historical context can help in understanding its evolution and application.
   - **Intuition Behind the Algorithm :** Imagine walking blindfolded on a hill; your goal is to find the lowest point. You feel the slope under your feet and take steps accordingly. Gradient Descent works similarly by iteratively moving towards the lowest point of the function (cost function in ML).
2. **Cost Function - The Core of Gradient Descent :**
   - **Real-World Analogies :** Think of the cost function like a GPS system in a car, guiding you towards your destination (optimal model performance) while avoiding traffic (errors).
   - **Detailed Differences from Loss Function :** Expand on how the cost function's role in a model's training phase differs from the loss function's role in evaluating individual training examples.
3. **Mechanics of Gradient Descent:**
   - **Starting Point :** Emphasize the importance of the initial starting point and how different starting points can lead to different solutions.
   - **Role of Derivatives :** Discuss how derivatives (gradients) provide directional information and how th guides the optimization process.

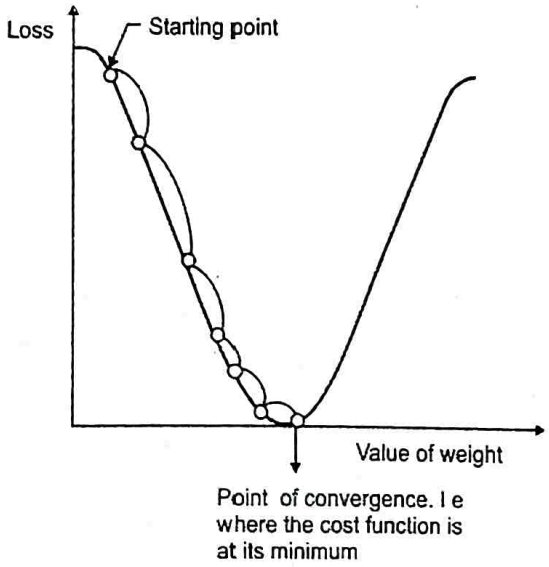


Fig. 2.5.14

4. **Learning Rate - Balancing Act :**
   - **Analogy :** Compare the learning rate to the accelerator pedal in a car. Press too hard (high learning rate), and you might overshoot your destination; press too gently (low learning rate), and you might arrive too slowly.
   - **Adaptive Learning Rates :** Introduce concepts like learning rate decay or advanced optimizers like Adam, which adjust the learning rate as the training progresses.

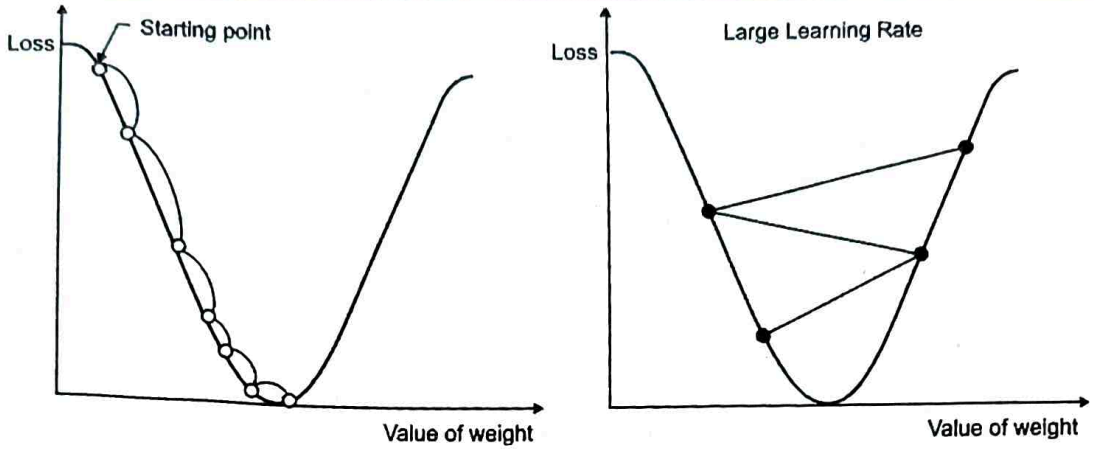


Fig. 2.5.15

5. **Variants of Gradient Descent :**
   - **Visual Examples :** Provide visual illustrations of how each variant (Batch, Stochastic, MiniBatch) processes data differently.
   - **Practical Applications :** Discuss when each type is preferred based on the size and nature of the dataset.
6. **Challenges and Solutions :**
   - **Local Minima and Saddle Points :** Offer strategies for overcoming these challenges, such as using momentum or advanced optimizers like RMSprop.
   - **Vanishing/Exploding Gradients :** Discuss normalization techniques or the use of specific activation functions like ReLU to mitigate these issues.
7. **Advanced Topics :**
   - **Gradient Descent in Deep Learning :** Delve into how gradient descent is adapted for complex models like deep neural networks.
   - **Beyond Gradient Descent :** Introduce alternative optimization methods like Quasi-Newton methods for certain types of problems.

By delving into these aspects, the tutorial becomes more comprehensive, providing learners with a deeper understanding of Gradient Descent and its pivotal role in Machine Learning.

Review Questions

Q. 1 Explain the concept and importance of histograms in EADA. Provide an example scenario where a histogram is crucial for data analysis.

Q. 2 Discuss the role of scatter plots, box plots, and descriptive statistics (like mean, median, mode, and standard deviation) in EDA. Illustrate with an example how these tools can be used together to gain insights into a complex dataset.

Q. 3 Define supervised learning and give one example each of classification and regression.

Q. 4 Elaborate on unsupervised learning, discussing clustering and dimensionality reduction. Include a case study to demonstrate their application.

Q. 5 Briefly describe simple linear regression with an example of its application in predictive analysis.

Q. 6 Compare and contrast multiple linear regression, stepwise regression, and logistic regression. Provide examples where each method would be most appropriate.

Q. 7 Define and explain the importance of model evaluation metrics such as accuracy and precision.

Q. 8 Discuss in detail the concepts of the confusion matrix, ROC curve analysis, and k-fold cross-validation. Provide a case study or example to illustrate these concepts in practice.

Q. 9 Describe the basic concept of Decision Trees in machine learning.

Q. 10 Provide an in-depth analysis of Ensemble Learning techniques, particularly focusing on Boosting and Bagging. Include examples to highlight their applications and differences.

❑❑❑

Model Evaluation, Data Visualization and Management

Syllabus

Model Evaluation Metrics : Accuracy, precision, recall, F1-score, Area Under the Curve (AUC), Evaluating models for imbalanced datasets

Data Visualization and Communication : Principles of effective data visualization, Types of visualizations : bar charts, line charts, scatter plots, etc. Visualization tools : matplotlib, seaborn, Tableau, etc. Data storytelling : communicating insights through visualizations

Data Management : Introduction to data management activities, Data pipelines : data extraction, transformation, and loading (ETL), Data governance and data quality assurance, Data privacy and security considerations

3.1 Model Evaluation Metrics

Model evaluation metrics are essential in assessing the performance of machine learning models. The choice of metrics depends on the specific problem and the goals of the model. Here are some commonly used evaluation metrics in machine learning :

1. **Classification Metrics**
   - i) Accuracy
   - ii) Precision
   - iii) Recall (Sensitivity or True Positive Rate)
   - iv) F1 Score
   - v) Area Under the Receiver Operating Characteristic (ROC-AUC)
2. **Regression Metrics**
   - i) Mean Absolute Error (MAE)
   - ii) Mean Squared Error (MSE)
   - iii) Root Mean Squared Error (RMSE)
   - iv) R-squared (R2)
3. **Clustering Metrics**
   - i) Silhouette Score
   - ii) Davies-Bouldin Index

4. **Multi-class Classification Metrics**
   i) Macro/Micro/Average Precision, Recall, F1 Score
   ii) Cohen's Kappa
5. **Anomaly Detection Metrics**
   i) Area under the Precision-Recall Curve (AUC-PR)
6. **Ranking Metrics**
   i) Mean Reciprocal Rank (MRR)

Classification Metrics

In a classification task, our main task is to predict the target variable which is in the form of discrete values. A very common example of binary classification is spam detection, where the input data could include the email text and metadata (sender, sending time), and the output label is either "spam" or "not spam." Sometimes, people use some other names also for the two classes: "positive" and "negative," or "class 1" and "class 0."

- To evaluate the performance of such a model there are metrics as mentioned below :
  - Classification Accuracy
  - Logarithmic loss
  - Area under Curve
  - F1 score
    - Precision
    - Recall
  - Confusion Matrix

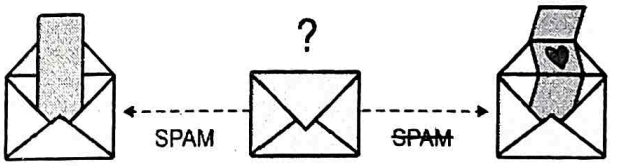


Fig. 3.1.1

3.1.1 Accuracy

- Accuracy simply measures how often the classifier correctly predicts. Accuracy is the ratio of the number of correct predictions and the total number of predictions. It is suitable for balanced datasets but may be misleading for imbalanced datasets.
- When any model gives an accuracy rate of 99%, you might think that model is performing very good but this is not always true and can be misleading in some situations.

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

- Consider a binary classification problem, where a model can achieve only two results, either model gives a correct or incorrect prediction. Now imagine we have a classification task to predict if an image is a dog or cat.

- In a supervised learning algorithm, we first **fit/train** a model on training data, then **test** the model on **testing data**. Once we have the model's predictions from the **X_test** data, we compare them to the **true y_values** (the correct labels).

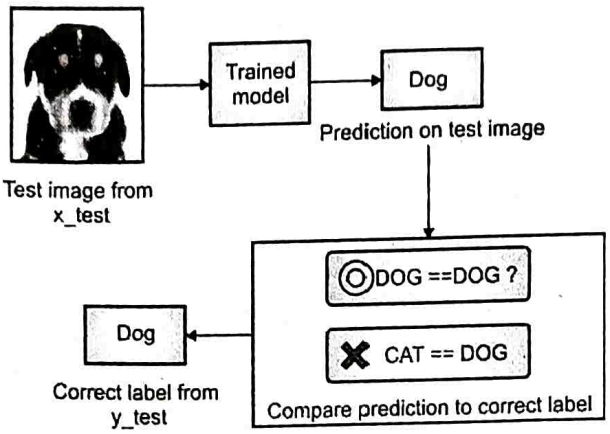


Fig. 3.1.2

- We feed the image of the dog into the training model. Suppose the model predicts that this is a dog, and then we compare the prediction to the correct label. If the model predicts that this image is a cat and then we again compare it to the correct label and it would be incorrect.
- We repeat this process for all images in X_test data. Eventually, we'll have a count of correct and incorrect matches. Accuracy is useful when the target class is well balanced but is not a good choice for the unbalanced classes.
- Imagine the scenario where we had 99 images of the dog and only 1 image of a cat present in our training data. Then our model would always predict the dog, and therefore we got 99% accuracy. In reality, data is always imbalanced for example spam email, credit card fraud, and medical diagnosis.

3.1.2 Precision

- Precision is a measure of a model's performance that tells you how many of the positive predictions made by the model are actually correct. It is calculated as the number of true positive predictions divided by the number of true positive and false positive predictions.
- The importance of Precision is in music or video recommendation systems, e-commerce websites, etc. where wrong results could lead to customer churn and this could be harmful to the business.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

3.1.3 Recall

- It explains how many of the actual positive cases we were able to predict correctly with our model. It is important in medical cases where it doesn't matter whether we raise a false alarm but the actual positive cases should not go undetected.
- Recall for a label is defined as the number of true positives divided by the total number of actual positives.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

3.1.4 F1-score

It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall. F1 Score is the harmonic mean of precision and recall. The F1 score punishes extreme values more.

- F1 Score could be an effective evaluation metric in the following cases :
  - When FP and FN are equally costly.
  - Adding more data doesn't effectively change the outcome
  - True Negative is high

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

3.1.5 Area under the Curve (AUC)

- It is one of the widely used metrics and basically used for binary classification. The AUC of a classifier is defined as the probability of a classifier will rank a randomly chosen positive example higher than a negative example.
- **True Positive Rate :** Also called or termed sensitivity. True Positive Rate is considered as a portion of positive data points that are correctly considered as positive, with respect to all data points that are positive.

$$TPR = \frac{TP}{TP + FN}$$

- **True Negative Rate :** Also called or termed specificity. False Negative Rate is considered as a portion of negative data points that are correctly considered as negative, with respect to all data points that are negatives.

$$TNR = \frac{TN}{TN + FP}$$

- **False-Positive Rate :** False Negative Rate is considered as a portion of negative data points that are mistakenly considered as negative, with respect to all data points that are negative.

$$FPR = \frac{FP}{FP + TN}$$

- False Positive Rate and True Positive Rate both have values in the range [0, 1]. A U C is a curve plotted between False Positive Rate Vs True Positive Rate at all different data points with a range of [0, 1]. Greater the value of AUC better the performance of the model.

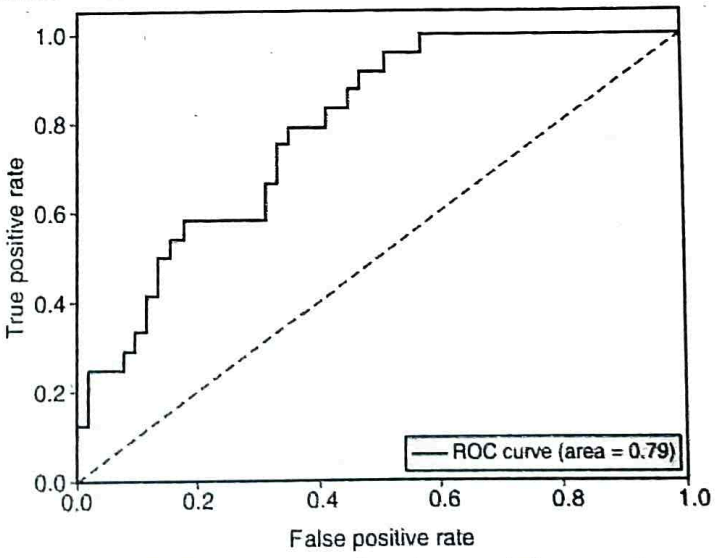


Fig. 3.1.3 : Receiver operating characteristic example

3.1.6 Evaluating Models for Imbalanced Datasets

Evaluating machine learning models on imbalanced datasets requires careful consideration to ensure that the model's performance is adequately assessed, especially regarding the minority class.

Here are some key evaluation strategies and metrics for imbalanced datasets :

1. Confusion Matrix
   - True Positives (TP)
   - False Positives (FP)
   - True Negatives (TN)
   - False Negatives (FN)

   Metrics for Imbalanced Datasets
   - Precision
   - Recall (Sensitivity or True Positive Rate)
   - F1 Score
3. Area Under the Precision-Recall Curve (AUC-PR)
4. Receiver Operating Characteristic (ROC) Curve
5. Precision-Recall Curve
6. Stratified Sampling and Cross-Validation
7. Cost-sensitive Learning
8. Ensemble Methods
9. Resampling Techniques
10. Use Specific Evaluation Metrics for Imbalanced Data

3.2 Data Visualization and Communication

Data visualization and communication play crucial roles in the field of machine learning. Effective visualization not only helps in understanding and exploring the data but also aids in communicating the results and insights derived from machine learning models. Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics, and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.

3.2.1 Principles of Effective Data Visualization

1. **Understand the Problem Domain :** Gain a deep understanding of the problem domain and the goals of your machine learning project to create visualizations that are relevant and impactful.
2. **Visualize Model Performance :** Use visualizations to communicate the performance of machine learning models. ROC curves, precision-recall curves, and confusion matrices are particularly useful for classification models.

3. **Feature Importance :** Visualize feature importance to communicate the contribution of each feature to the model's predictions. This can help stakeholders understand the factors driving the model's decisions.
4. **Model Interpretability :** Employ visualizations that enhance the interpretability of complex models. Techniques like partial dependence plots, SHAP (SHapley Additive exPlanations), and LIME (Local Interpretable Model-agnostic Explanations) can be effective.
5. **Evaluate Bias and Fairness :** Use visualizations to assess and communicate potential biases in the model predictions. Visualize performance metrics across different demographic groups to identify and address fairness concerns.
6. **Dynamic Model Exploration :** Create interactive visualizations that allow users to explore model predictions, understand decision boundaries, and investigate how changes in input features impact the output.
7. **Visualizing Time Series and Temporal Patterns :** If working with time-series data, use visualizations such as line charts, stacked area charts, or heat maps to highlight temporal patterns and trends.
8. **Ensemble Model Visualization :** When using ensemble models, visualize the combined output of multiple models. Understanding the decision-making process of an ensemble can be valuable.
9. **Uncertainty Visualization :** If your model provides uncertainty estimates (e.g., Bayesian models), visualize the uncertainty to convey the level of confidence in predictions.
10. **Data Quality and Preprocessing :** Use visualizations to explore the distribution of your input features, identify outliers, and assess the effectiveness of preprocessing steps.
11. **Interactive Reporting for Stakeholders :** Create interactive reports and dashboards to present machine learning results to stakeholders. Allow them to interact with the data and model outputs for deeper insights.
12. **Avoid Overfitting :** Visualize learning curves to check for overfitting or underfitting. Understanding the model's behavior during training can inform adjustments to hyper parameters.
13. **Communication with Domain Experts :** Collaborate with domain experts to create visualizations that are meaningful and align with the domain-specific understanding of the problem.
14. **Version Control for Visualizations :** If your machine learning project involves multiple iterations or models, consider version control for visualizations to track changes and improvements over time.
15. **Ethical Considerations and Transparency :** Clearly communicate the limitations, assumptions, and potential biases of the model through visualizations. Address ethical considerations transparently.

By following these principles, you can create visualizations that not only aid in understanding machine learning models but also facilitate effective communication with various stakeholders involved in the project.

3.2.2 Types of Visualizations

Following are few types of visualizations :

- Drawing Plot
- Bar Chart
- Histogram
- Distribution or Density plot

- Box Plot
- Scatter Plot
- Pair Plot
- Correlation and Heat Map
- Line Charts

1. Line Charts

In a line chart, each data point is represented by a point on the graph, and these points are connected by a line. We may find patterns and trends in the data across time by using line charts. Time-series data is frequently displayed using line charts.

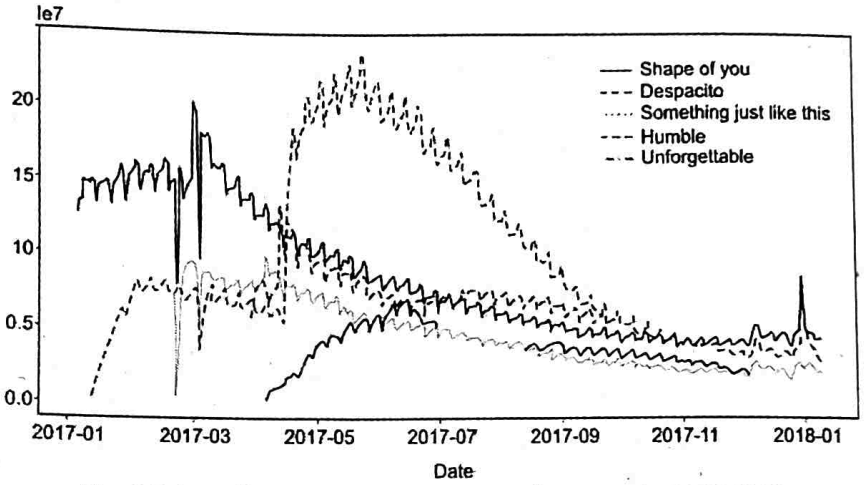


Fig. 3.2.1 : Daily Global Streams of popular songs in 2017-2018

2. Scatter Plots

A quick and efficient method of displaying the relationship between two variables is to use scatter plots. With one variable plotted on the x-axis and the other variable drawn on the y-axis, each data point in a scatter plot is represented by a point on the graph. We may use scatter plots to visualize data to find patterns, clusters, and outliers.

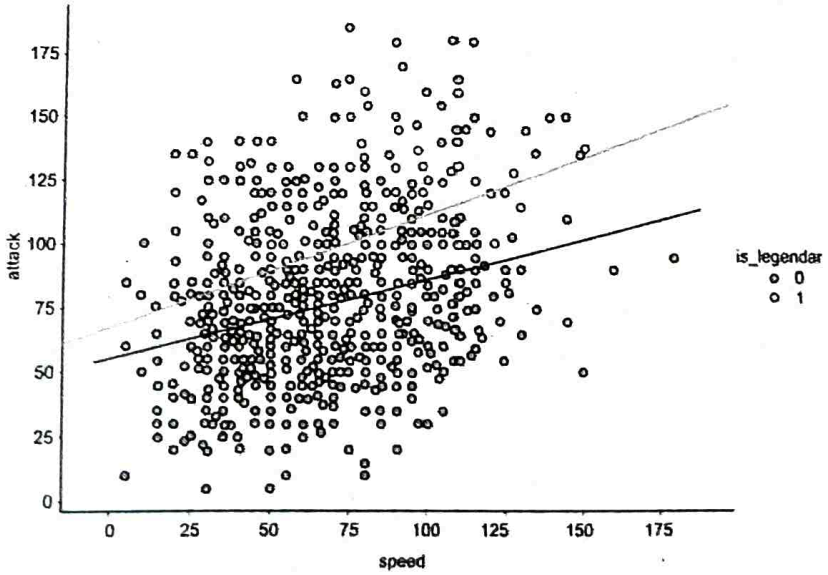


Fig. 3.2.2

3. Bar Charts

Bar charts are a common way of displaying categorical data. In a bar chart, each category is represented by a bar, with the height of the bar indicating the frequency or proportion of that category in the data. Bar graphs are useful for comparing several categories and seeing patterns over time.

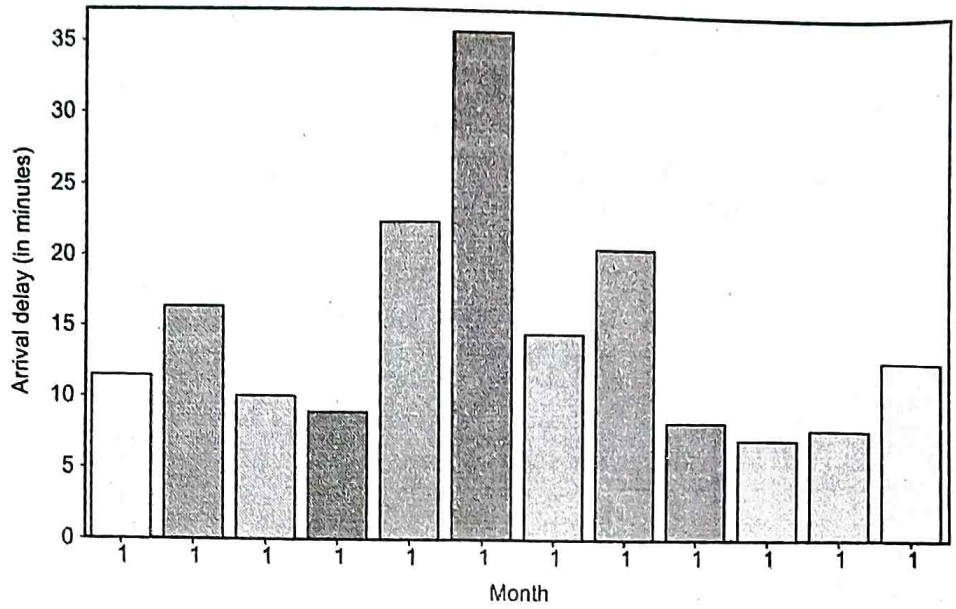


Fig. 3.2.3 : Average Arrival Delay for Spirit Airlines Flights, by Month

4. Box Plots

Box plots are a graphical representation of the distribution of a set of data. In a box plot, the median is shown by a line inside the box, while the center box depicts the range of the data. The whiskers extend from the box to the highest and lowest values in the data, excluding outliers. Box plots can help us to identify the spread and skewness of the data.

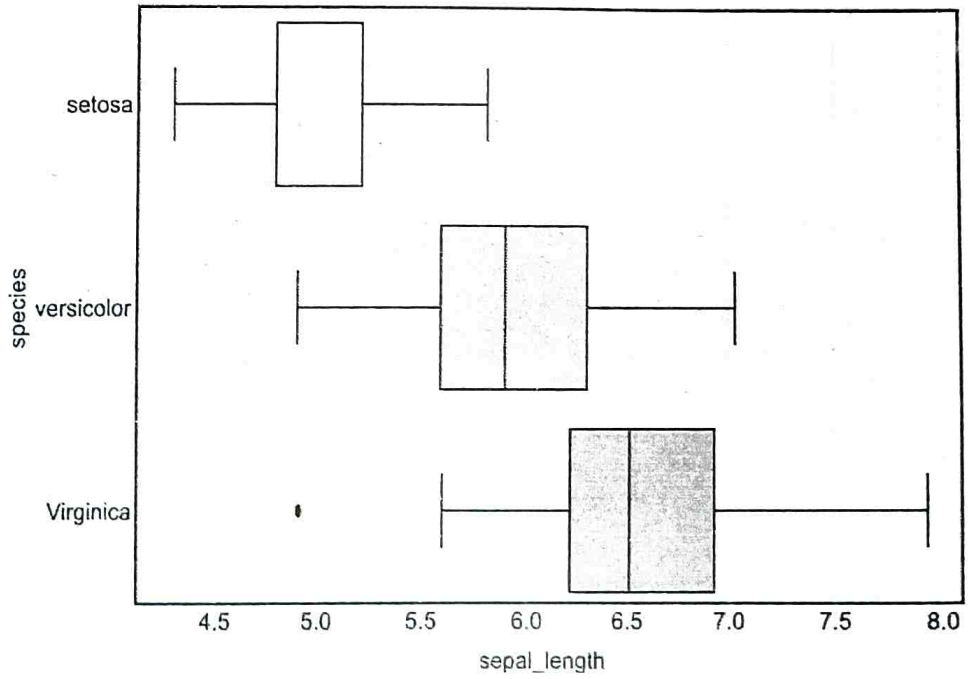


Fig. 3.2.4

5. Histogram

A histogram is a plot that shows the frequency distribution of a set of continuous variables. The histogram gives an insight into the underlying distribution of the variable, outliers, skewness, etc.

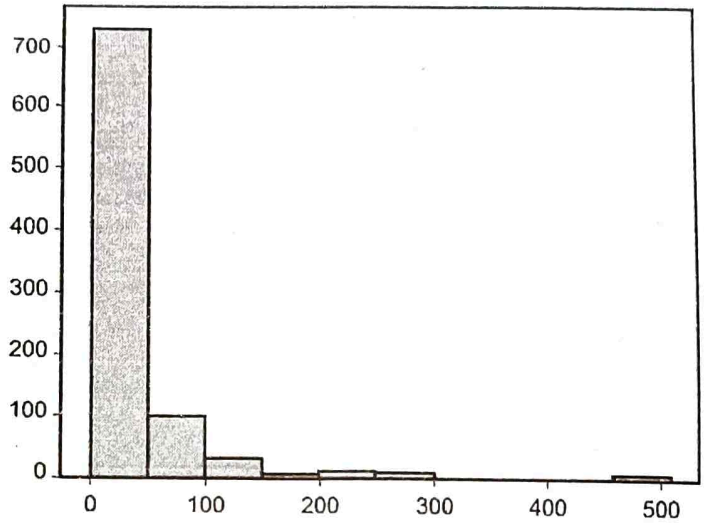


Fig. 3.2.5

6. Heat Maps

A heatmap is a common and beautiful matrix plot that can be used to graphically summarize the relationship between two variables. The degree of correlation between two variables is represented by a color code. For example, this heat extracted from our analyzes the occupation of the guests of the Daily Show during the 1999-2012 period. As expected, guests from the acting and media industries are the most frequent attendants.

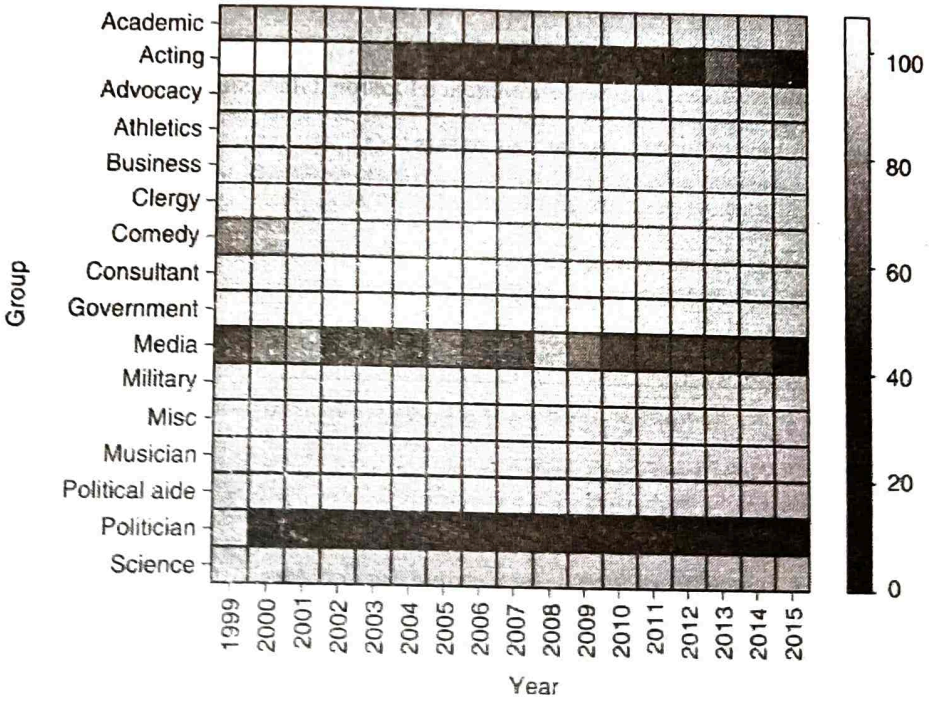


Fig. 3.2.6

7. Tree Maps

Tree maps are suitable to show part-to-whole relationships in data. They display hierarchical data as a set of rectangles. Each rectangle is a category within a given variable, whereas the area of the rectangle is proportional to the size of that category. Compared to similar visualizations, like pie charts, tree maps are considered more intuitive and preferable.

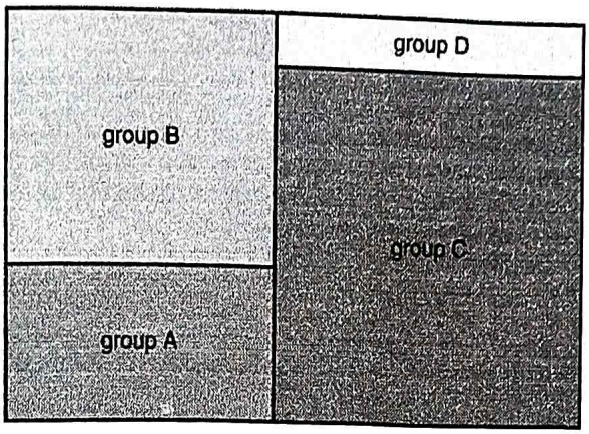


Fig. 3.2.7

3.2.3 Visualization Tools

1. Tableau

It is an interactive data visualization software. This tool is used for effective data analysis and data visualization in the industry. It has a drag and drop interface and this feature helps it to perform tasks easily and very fast. The software doesn't force its users to write codes. The software is compatible with a lot of data sources. The tool is a bit expensive but it is the most preferred choice of a top company like Amazon.

Key features of Tableau

- Tableau is known as the simplest business intelligence tool for data visualization
- Data scientists do not need to write custom code in this tool
- The tool is also a real-time collaboration along with data mixing

2. Microsoft Power BI

It is a set of business analytics tools that can simplify data, prepare and analyze instantly. It is the most preferred tool as it can easily integrate with Microsoft tools and is absolutely free to use and download. The tool is available for both mobile and desktop versions. So if a business uses Microsoft tools it can be a big benefit for them.

Key Features of Microsoft Power BI

- Generate interactive data visualizations across multiple data centers
- It offers enterprise data analytics as well as self-service on a single platform
- Even non-data scientists can easily create machine learning models

3. **Matplotlib**

- Matplotlib library is used to create static 2D plots, although it does have some support for 3D visualizations. It makes producing both simple and advanced plots straightforward and intuitive. It can be used in Python scripts, Jupyter notebook, and web application servers.
- Matplotlib is a low-level library of Python which is used for data visualization. It is easy to use and emulates MATLAB like graphs and visualization. This library is built on the top of NumPy arrays and consist of several plots like line chart, bar chart, histogram, etc. It provides a lot of flexibility but at the cost of writing more code.

Seaborn

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It is built on the top of matplotlib library and also closely integrated into the data structures from pandas. Seaborn helps to visualize the statistical relationships, To understand how variables in a dataset are related to one another and how that relationship is dependent on other variables, we perform statistical analysis. This Statistical analysis helps to visualize the trends and identify various patterns in the dataset.

3.2.4 Data Storytelling and Communicating Insights through Visualizations

Data storytelling is a method of communicating insights and information derived from data through the use of compelling narratives, visuals, and data-driven evidence. It involves presenting data in a way that makes it easier for people to understand, engage with, and draw meaningful conclusions from the information presented. By weaving data into a cohesive and persuasive story, data storytelling enables organizations and individuals to make informed decisions, influence stakeholders, and create impactful presentations.

Data Storytelling Benefits

- Data storytelling makes complex data more accessible and understandable, allowing audiences to grasp insights easily.
- Engaging narratives and compelling visuals keep audiences hooked, increasing retention and attention.
- Storytelling with data empowers better decision-making by presenting evidence-based insights.
- Stories are memorable; dat5a combined with storytelling creates a lasting impact on the audience.
- Convincing narratives backed by data build trust and influence stakeholders effectively.
- Data storytelling adds a human touch, making data relatable and emotionally compelling.
- Data-driven stories create empathy, helping organizations connect with their customers' needs and experiences.
- Storytelling reveals trends and patterns hidden within data, leading to valuable discoveries.
- Data stories help identify and address challenges, promoting innovative problem-solving.
- Storytelling unifies teams around data, enabling better collaboration and shared vision.
- Data storytelling is a potent tool for advocating social causes and driving positive change.
- Organizations that excel in data storytelling gain a competitive edge in their industries.
- Data storytelling eliminates jargon and simplifies complexities, promoting clear communication.

- Understand customer preferences and behavior better through data-driven stories.
- Engaging data stories form the basis of powerful marketing campaigns.

3.2.5 Storytelling Visualization Examples

Table 3.2.1

Sr. No.	Visualization Type	Description	Example Scenario
1.	Timeline Storytelling	Represents the progression of events or trends over time.	Tracking the growth of a company's revenue over the years.
2.	Geospatial Storytelling	Utilizes maps to convey information related to geographic locations.	Displaying the spread of a disease or regional sales performance.
3.	Interactive Dashboards	Provides users with interactive tools to explore data on their own.	Business performance dashboard allowing users to interact with metrics.
4.	Flowcharts and Decision Trees	Visualizes decision processes or workflows.	Illustrating the decision-making process in a marketing strategy.
5.	Comparative Visualizations	Highlights contrasts and trends through visual comparisons.	Comparing sales figures between different product categories.
6.	Word Clouds	Emphasizes the frequency or importance of words in a dataset.	Analyzing customer reviews to identify common feedback themes.
7.	Sankey Diagrams	Illustrates the flow of resources or processes using interconnected lines.	Displaying the distribution of budget allocations in a project.
8.	Heatmaps	Displays patterns and variations in data using color gradients.	Representing website user activity with a heatmap.
9.	Storyboarding	Creates a sequence of visualizations that unfold sequentially to tell a story.	Illustrating the stages of a product lifecycle or project development.
10.	Infographics	Combines text and visuals in a concise format to convey information effectively.	Summarizing key insights from a research study in an infographic.
11.	Network Graphs	Visualizes relationships and connections between entities.	Mapping social network connections or dependencies in a supply chain.
12.	Pictorial Charts	Enhances visual appeal by using images and icons in charts.	Representing demographic data using icons of people for engagement.
13.	Animated Visualizations	Incorporates animation to dynamically show changes in data over time.	Animated chart illustrating the growth of a social media platform's user base.
14.	Circular Visualizations	Represents data in a circular format for a unique perspective.	Displaying market share distribution among competitors in a circular chart.

3.3 Data Management

Data management refers to the processes and activities involved in acquiring, organizing, storing, securing, and analyzing data throughout its lifecycle. Proper data management is crucial for ensuring the reliability, accuracy, and accessibility of information within an organization.

3.3.1 Introduction to Data Management Activities

Data management activities encompass a wide range of practices related to the acquisition, storage, organization, processing, and analysis of data. These activities are crucial for ensuring that data is accurate, accessible, and secure. Here are some key data management activities :

Data Collection

- Identify and define the data needed for your organization or project.
- Establish methods for collecting data, whether it's through surveys, sensors, logs, user interactions, or other sources.

2. **Data Storage**

- Choose appropriate data storage solutions (databases, data warehouses, file systems) based on the type and volume of data.
- Implement secure and scalable storage infrastructure.

3. **Data Quality Assurance**

- Ensure the accuracy, consistency, and completeness of data through validation and verification processes.
- Implement data cleaning and deduplication techniques to maintain high data quality.

4. **Data Security**

- Establish access controls and encryption mechanisms to protect sensitive data.
- Regularly audit and monitor access to prevent unauthorized use.
- Develop and implement a robust data backup and recovery plan.

5. **Data Organization and Cataloging**

- Create a data catalog that documents metadata, including data definitions, formats, and relationships.
- Use consistent naming conventions and taxonomies for easy data discovery.

6. **Data Integration**

- Integrate data from various sources to create a unified and comprehensive view.
- Implement Extract, Transform, Load (ETL) processes for data integration.

7. **Data Governance**

- Develop and enforce policies and procedures for data management.
- Assign responsibilities for data stewardship and establish a data governance framework.

8. **Data Privacy and Compliance**
   - Ensure compliance with data protection laws and regulations (e.g., GDPR, HIPAA).
   - Implement privacy controls to protect personally identifiable information (PII).
9. **Data Retrieval and Analysis**
   - Develop tools and systems for querying and retrieving data efficiently.
   - Perform data analysis and reporting to derive insights and support decision-making.
10. **Data Archiving and Purging**
   - Establish a data retention policy to determine how long data should be stored.
   - Archive historical data and purge obsolete data to optimize storage resources.
11. **Data Documentation**
   - Document data sources, definitions, and transformations for future reference.
   - Maintain documentation to facilitate collaboration and knowledge transfer.
12. **Data Auditing and Monitoring**
   - Regularly audit data to ensure compliance with quality standards and policies.
   - Implement monitoring systems to detect anomalies or unauthorized activities.

3.3.2 Data Pipelines

A data pipeline is a set of processes and tools for ingesting, processing, transforming, and moving data from one or multiple sources to a destination, typically a storage or analytics platform. Data pipelines are essential components in modern data architecture, enabling organizations to handle and analyze large volumes of data efficiently.

1. **Data Source**
   - **Definition :** The origin of the data.
   - **Examples :** Databases, log files, external APIs, streaming platforms.
2. **Data Ingestion**
   - **Definition :** The process of bringing data into the pipeline from various sources.
   - **Examples :** Batch ingestion (e.g., scheduled ETL jobs) or real-time ingestion (e.g., streaming).
3. **Data Processing**
   - **Definition :** The manipulation and transformation of raw data to prepare it for analysis.
   - **Examples :** Cleaning, filtering, aggregating, and enriching data.
4. **ETL (Extract, Transform, Load)**
   - **Definition :** A common data processing paradigm that involves extracting data from source systems, transforming it, and loading it into a destination, such as a data warehouse.
   - **Examples :** Using tools like Apache Spark, Apache Flink, or commercial ETL tools.

5. **Data Transformation**
   - **Definition :** The process of converting raw data into a format suitable for analysis.
   - **Examples :** Changing data types, handling missing values, creating derived features.
6. **Data Storage**
   - **Definition :** Where processed data is stored for future retrieval and analysis.
   - **Examples :** Data warehouses (e.g., Amazon Redshift, Google BigQuery), data lakes.
7. **Data Movement**
   - **Definition :** Transferring data from one location to another within the pipeline.
   - **Examples :** Data replication, file transfers.
8. **Data Security**
   - **Definition :** Measures to protect data confidentiality, integrity, and availability within the pipeline.
   - **Examples :** Encryption, access controls, secure connections.
9. **Data Governance**
   - **Definition :** Policies and processes for ensuring data compliance, privacy, and ethical use.
   - **Examples :** Metadata management, data lineage tracking.

Data pipelines are critical for maintaining a smooth and reliable flow of data within an organization, supporting analytics, reporting, and decision-making processes. They play a key role in handling the complexity of managing and processing large volumes of data efficiently.

Review Questions

Q. 1 Explain various model evaluation metrics.

Q. 2 Define the term accuracy.

Q. 3 Explain the terms :- precision, recall, F1-score, AUC.

Q. 4 What are the principles of effective data visualization?

Q. 5 Explain the various types of data visualizations.

Q. 6 Explain the various tools used for data visualizations.

Q. 7 Write down importance of data storytelling and its benefits.

Q. 8 What is the need of data management and how it can be achieved?

Q. 9 Explain the concept of data pipelines.

❑❑❑

Lab Manual

LIST OF PRACTICALS

Practical No. 1

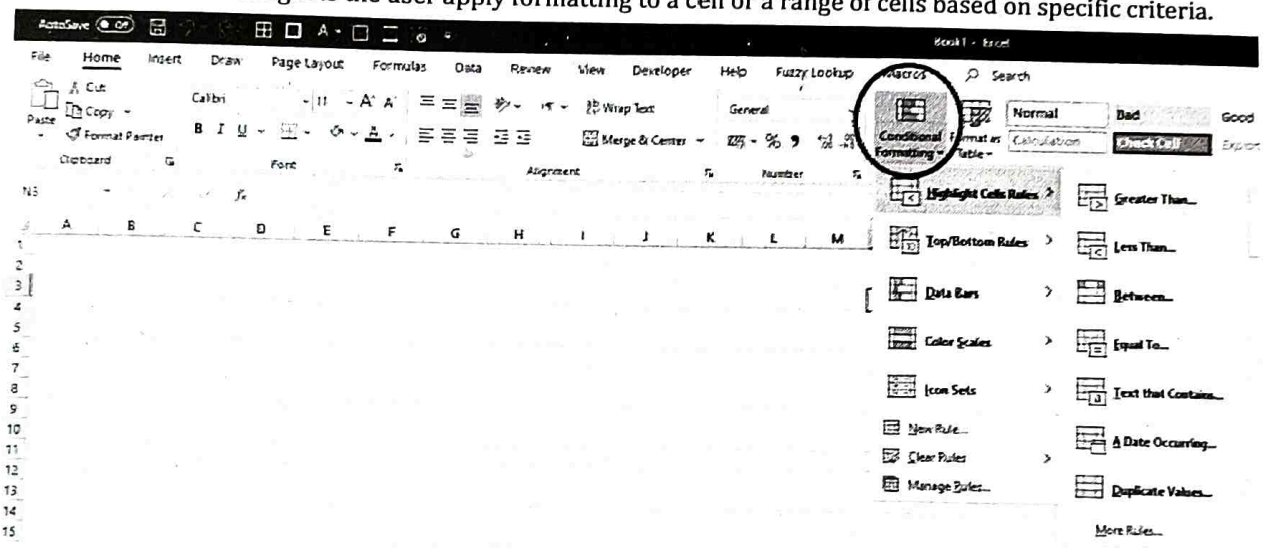
1 Introduction to Excel

- Perform conditional formatting on a dataset using various criteria.
- Create a pivot table to analyze and summarize data.
- Use VLOOKUP function to retrieve information from a different worksheet or table.
- Perform what-if analysis using Goal Seek to determine input values for desired output.

Solution :

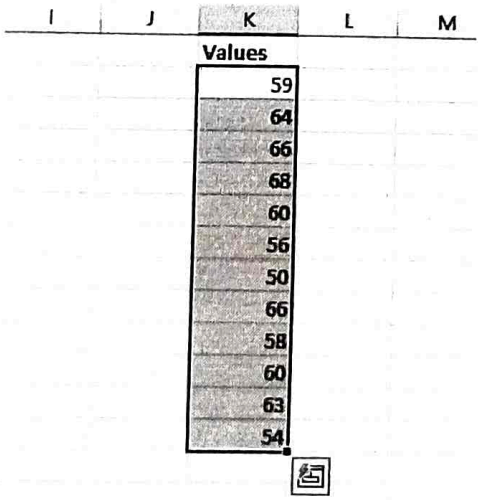
- **Perform conditional formatting on a dataset using various criteria.**

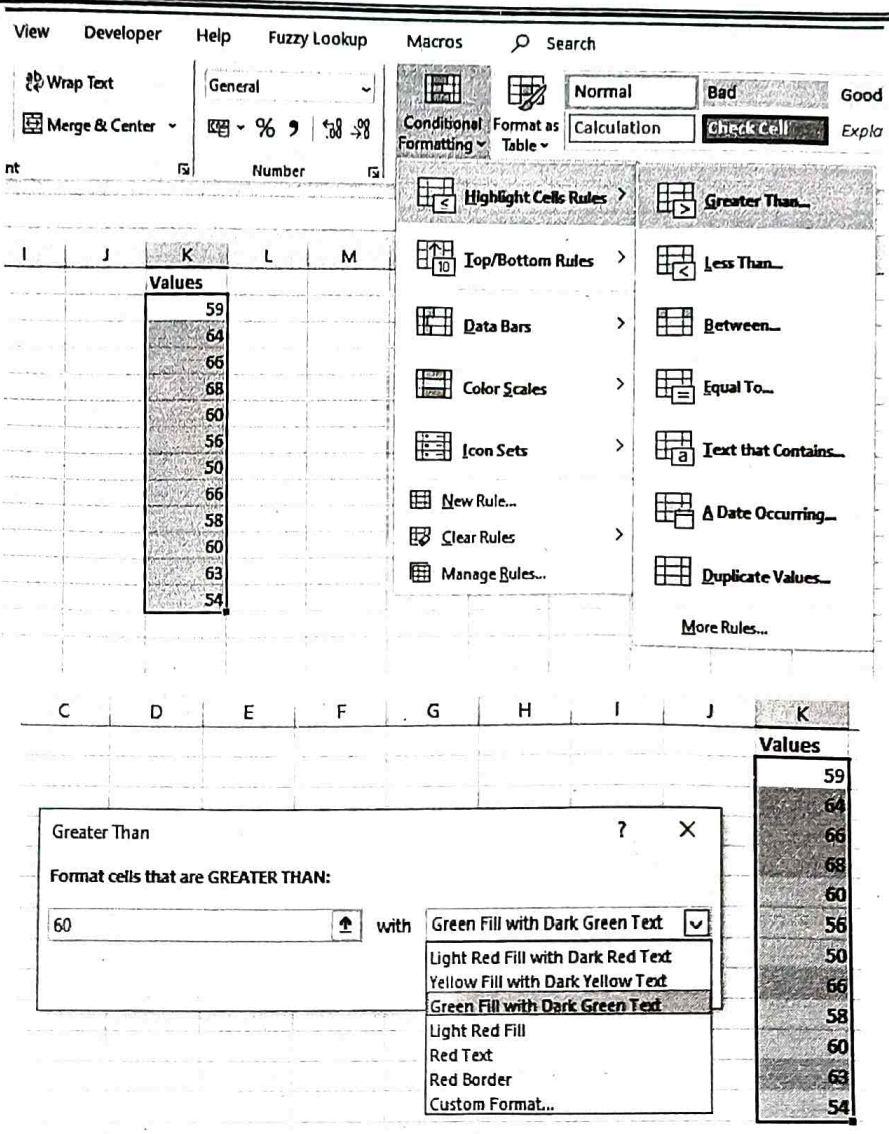
Conditional Formatting lets the user apply formatting to a cell or a range of cells based on specific criteria.



Formatting cells based on their values

Here, we can change the background color of all values greater than 60 to green:



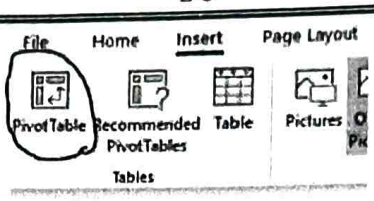


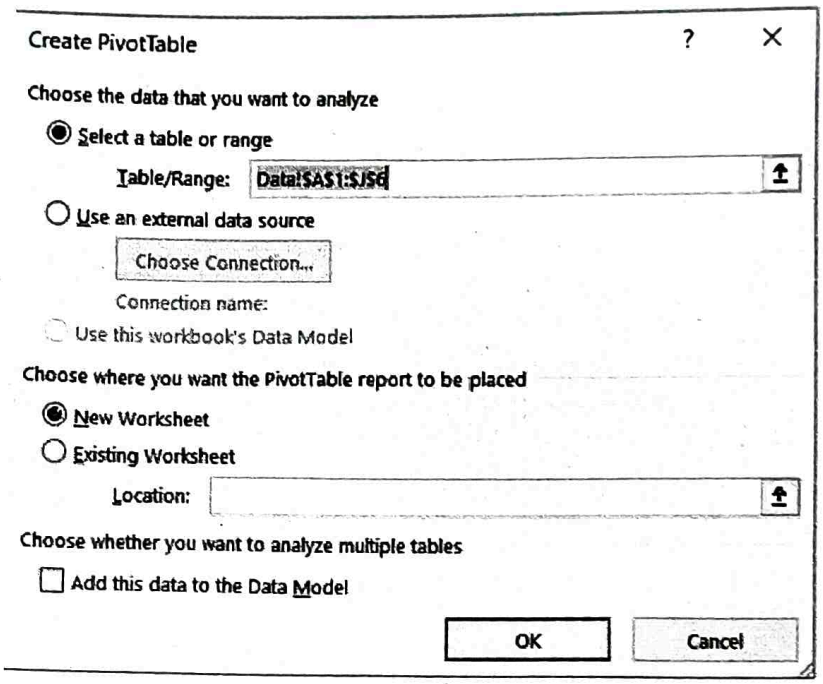
- **Create a pivot table to analyze and summarize data.**

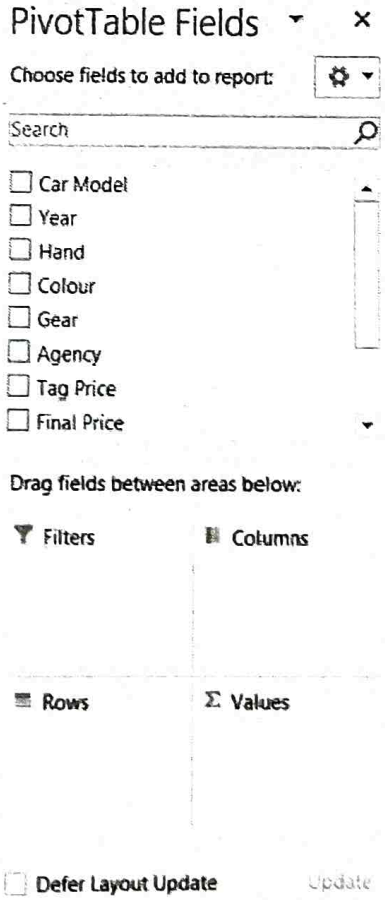
  Pivot Table is a great tool for summarizing and analyzing data in Excel.

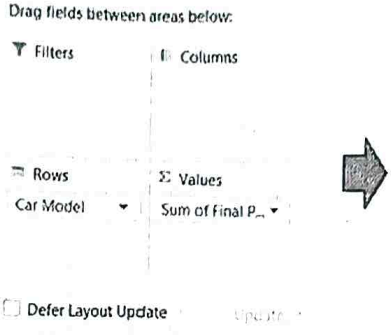
  We can use a Pivot Table to perform calculations on our data based on certain criteria.

	A	B	C	D	E	F	G	H	I	J
1	Car Model	Year	Hand	Colour	Gear	Agency	Tag Price	Final Price	Arrival Date	Sale Date
2	Toyota Yaris	2015	3	Yellow	Automatic	London	9500	9400	25/04/2017	04/05/2017
3	Mini Cooper	2017	2	Blue	Manual	Birmingham	21000	19500	25/10/2017	26/10/2018
4	Hyundai Sonata	2018	4	Black	Manual	Birmingham	9700	9600	26/02/2018	18/11/2018
5	Ford Edge	2018	2	Blue	Manual	Manchester	22000	18600	13/12/2018	02/02/2019
6	Hyundai i25	2017	1	Red	Manual	Liverpool	14500	12100	04/05/2017	30/01/2018
7	Hyundai Sonata	2015	1	Yellow	Automatic	Manchester	9700	9200	20/02/2018	12/03/2018
8	Mini Cooper	2017	2	Yellow	Manual	Birmingham	21000	20300	25/09/2017	12/10/2017
9	Ford Edge	2018	2	Blue	Manual	Manchester	22000	21100	03/12/2018	03/03/2019
10	Hyundai i25	2018	2	Blue	Manual	London	14500	12000	02/05/2017	12/06/2017
11	Hyundai Sonata	2017	3	Black	Manual	London	9700	8600	24/01/2018	16/02/2018
12	Mini Cooper	2015	4	Black	Manual	Manchester	21000	18700	06/10/2017	06/11/2017
13	Hyundai i25	2017	1	Red	Manual	Liverpool	14500	12400	08/05/2017	06/08/2017
14	Hyundai Sonata	2018	1	Red	Automatic	London	9700	9500	02/02/2018	13/05/2018
15	Hyundai i25	2018	2	Red	Manual	London	14500	14400	20/04/2017	09/06/2017
16										

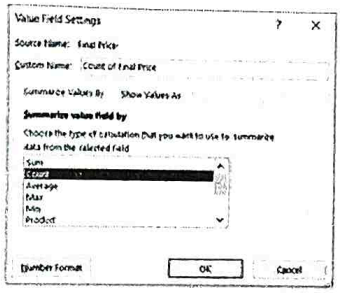

File
Home
Insert
Page Layout
PivotTable
Recommended PivotTables
Table
Pictures
Tables


Create PivotTable
Choose the data that you want to analyze
Select a table or range
Table/Range: Data!$A$1:$J$6
Use an external data source
Choose Connection...
Connection name:
Use this workbook's Data Model
Choose where you want the PivotTable report to be placed
New Worksheet
Existing Worksheet
Location:
Choose whether you want to analyze multiple tables
Add this data to the Data Model
OK
Cancel

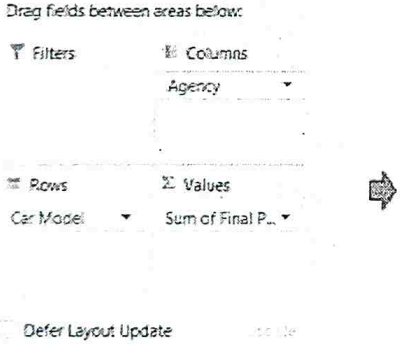

PivotTable Fields
Choose fields to add to report:
Search
Car Model
Year
Hand
Colour
Gear
Agency
Tag Price
Final Price
Drag fields between areas below:
Filters
Columns
Rows
Values
Defer Layout Update
Update



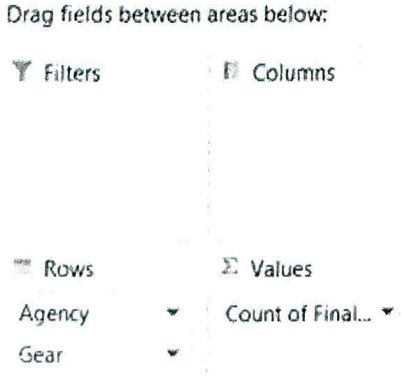
Row Labels	Sum of Final Price
Ford Edge	39700
Hyundai i25	50900
Hyundai Sonata	36900
Mini Cooper	58500
Toyota Yaris	9400
Grand Total	**195400**



Row Labels	Count of Final Price
Ford Edge	2
Hyundai i25	4
Hyundai Sonata	4
Mini Cooper	3
Toyota Yaris	1
Grand Total	**14**



Sum of Final Price	Column Labels				
Row Labels	Birmingham	Liverpool	London	Manchester	Grand Total
Ford Edge				39700	39700
Hyundai i25		24500	26400		50900
Hyundai Sonata	9600		18100	9200	36900
Mini Cooper	39800			18700	58500
Toyota Yaris			9400		9400
Grand Total	**49400**	**24500**	**53900**	**67600**	**195400**



Row Labels	Count of Final Price
⊟ **Birmingham**	**3**
Manual	3
⊟ **Liverpool**	**2**
Manual	2
⊟ **London**	**5**
Automatic	2
Manual	3
⊟ **Manchester**	**4**
Automatic	1
Manual	3
Grand Total	**14**

- **Use VLOOKUP function to retrieve information from a different worksheet or table.**

VLOOKUP helps us lookup a value in table, and return a corresponding value.

Syntax

=VLOOKUP(lookup_value,table_array,col_index_num,[range_lookup]

37	Name	Age	Company
38	Sean	35	Amazon
39	Sarah	25	Google
40	Yoav	48	Microsoft
41	Joe	37	Apple
42	Dani	52	Netflix
43	Tina	32	Disney
44	Sharon	33	Tesla
45	Gil	45	Facebook

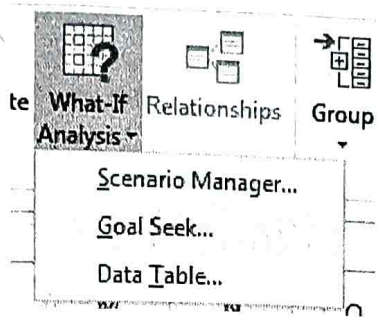
Name	Workplace	
Dani		=VLOOKUP(A32,$A$37:$C$45,3,0)
Tina		=VLOOKUP(A33,$A$37:$C$45,3,0)

	A	B	C
1	Name	ID	Age
2	Jean Claude	253	58
3	Arnold	646	72
4	Steven	235	67
5	Sylvester	356	73
6			
7	What is the Age of ID 646?		
8			
9	=VLOOKUP(646,B1:C5,2,0)		
10			

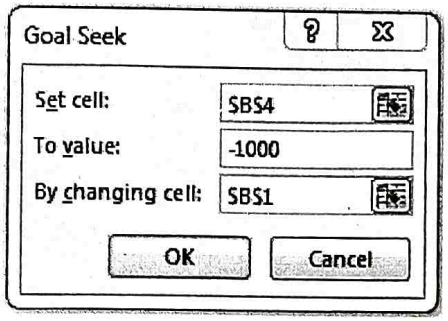
- **Perform what-if analysis using Goal Seek to determine input values for desired output.**

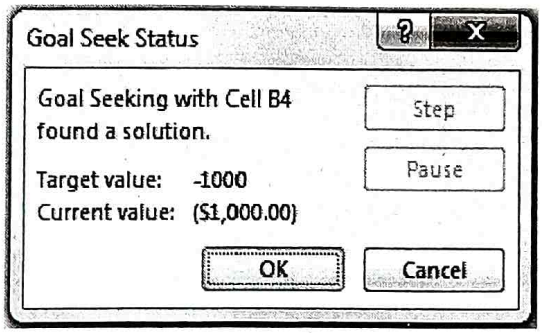
Goal Seek in Excel, as the name suggests, helps you in finding a target value by editing a dependent value.

Go to Data –> Data Tools –> What-If Analysis –> Goal Seek



- **Set Cell:** B4 (this is the cell with the goal/target).
- **To Value:** -1,000 (this is the goal value. Its negative here as it is an outflow).
- **By Changing Cell:** B1 (this is the loan amount that we want to change to achieve the goal value).



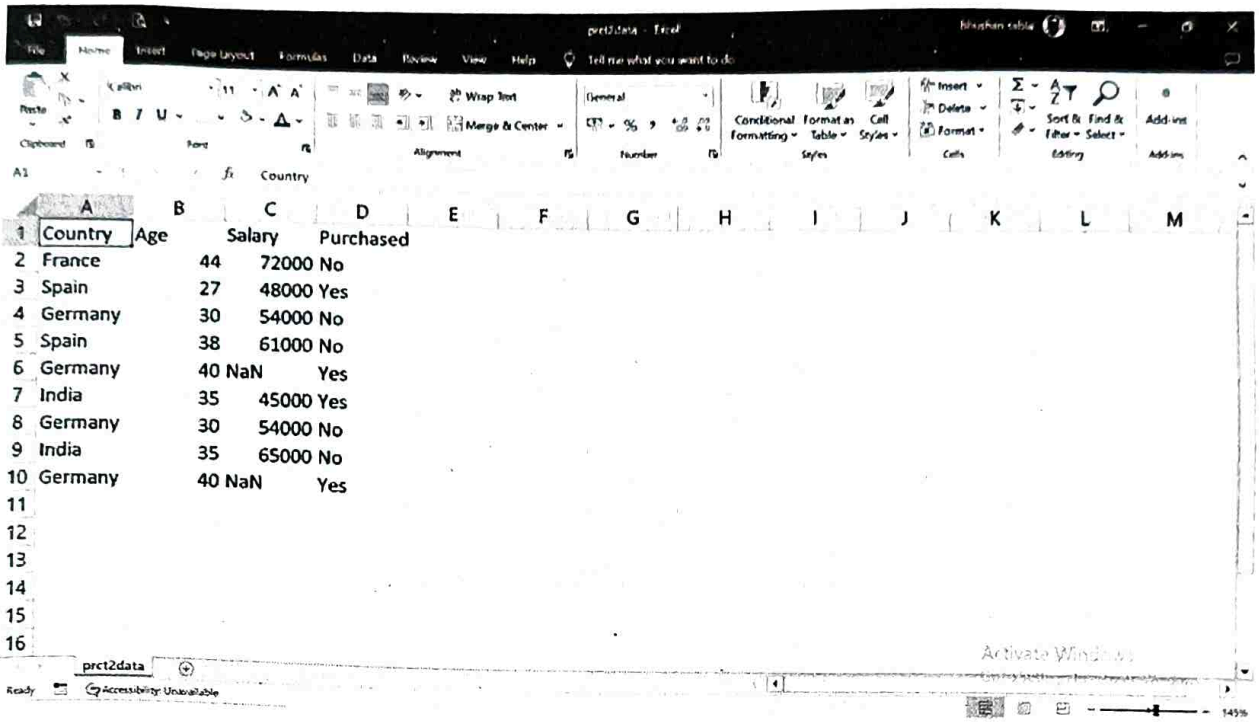


	A	B
1	Loan Amount	$51,726
2	Interest Rate	6%
3	Number of Monthly Payments	60
4	Monthly Payment	($1,000.00)

Practical No. 2

2. Data Frames and Basic Data Pre-processing
   - Read data from CSV and JSON files into a data frame.
   - Perform basic data pre-processing tasks such as handling missing values and outliers.
   - Manipulate and transform data using functions like filtering, sorting, and grouping.

Solution :



	A	B	C	D
1	Country	Age	Salary	Purchased
2	France	44	72000	No
3	Spain	27	48000	Yes
4	Germany	30	54000	No
5	Spain	38	61000	No
6	Germany	40	NaN	Yes
7	India	35	45000	Yes
8	Germany	30	54000	No
9	India	35	65000	No
10	Germany	40	NaN	Yes

```
//Read data from CSV and JSON files into a data frame.
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

ds = pd.read_csv('prct2data.csv')

print("Data Head: \n",ds.head())

print("\n Data Describe:",ds.describe())

X = ds.iloc[:, :-1].values
Y = ds.iloc[:, 3].values
print('\nInput : ',X)
print('\nOutput : ', Y)

//handling missing values
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing_values = np.nan, strategy = 'mean')

imputer = imputer.fit(X[:, 1:3])
```

```
X[:,1:3] = imputer.fit_transform(X[:, 1:3])
print("\n New Input with Mean value for NaN: ",X)

//outliers
import sklearn
from sklearn.datasets import load_diabetes
import pandas as pd
import matplotlib.pyplot as plt

db = load_diabetes()

column_name = db.feature_names
df_db = pd.DataFrame(db.data)
df_db.columns = column_name
df_db.head()

import seaborn as sns

sns.boxplot(df_db['bmi'])

import numpy as np

print(np.where(df_db['bmi']>0.12))

//sorting
display(df_db)

sorted = df_db.sort_values(by=['age'])
display(sorted)

//filterting rows
a = df_db.query('age > 0')
display(a)
//filtering columns
b = df_db.filter(['age','bp'])
display(b)
//grouping data
g = df_db.groupby('age')
g.first()
```

Practical No. 3

3. Feature Scaling and Dummification

- Apply feature-scaling techniques like standardization and normalization to numerical features.
- Perform feature dummification to convert categorical variables into numerical representations.

Solution :

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
data = pd.DataFrame(data=np.c_[iris['data'], iris['target']], columns=iris['feature_names'] + ['target'])

# Extract numerical features
numerical_features = iris['feature_names']

# Separate features and target variable
X = data[numerical_features]
y = data['target']

# Standardization
scaler_standard = StandardScaler()
X_standardized = scaler_standard.fit_transform(X)

# Normalization (Min-Max scaling)
scaler_minmax = MinMaxScaler()
X_normalized = scaler_minmax.fit_transform(X)

# Visualize the original, standardized, and normalized features
plt.figure(figsize=(12, 4))

plt.subplot(131)
plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=y, cmap='viridis')
plt.title('Original Features')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
```

```
plt.subplot(132)
plt.scatter(X_standardized[:, 0], X_standardized[:, 1], c=y, cmap='viridis')
plt.title('Standardized Features')
plt.xlabel('Feature 1 (Standardized)')
plt.ylabel('Feature 2 (Standardized)')

plt.subplot(133)
plt.scatter(X_normalized[:, 0], X_normalized[:, 1], c=y, cmap='viridis')
plt.title('Normalized Features')
plt.xlabel('Feature 1 (Normalized)')
plt.ylabel('Feature 2 (Normalized)')

plt.tight_layout()
plt.show()

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Sample dataset with categorical variables
data = {
  'Color': ['Red', 'Blue', 'Green', 'Red', 'Blue'],
  'Size': ['Small', 'Large', 'Medium', 'Medium', 'Small'],
  'Label': [1, 0, 1, 0, 1]
}

df = pd.DataFrame(data)

# Perform one-hot encoding (feature dummification)
df_encoded = pd.get_dummies(df, columns=['Color', 'Size'], drop_first=True)

# Display the original and encoded dataframes
print("Original DataFrame:")
print(df)
print("\nDataFrame after Feature Dummification:")
print(df_encoded)

# Split the dataset into features (X) and target variable (y)
```

```
X = df_encoded.drop('Label', axis=1)
y = df_encoded['Label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a simple model for demonstration (e.g., RandomForestClassifier)
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("\nModel Accuracy:", accuracy)
```

Practical No. 4

4. Hypothesis Testing

- Formulate null and alternative hypotheses for a given problem.
- Conduct a hypothesis test using appropriate statistical tests (e.g., t-test, chisquare test).
- Interpret the results and draw conclusions based on the test outcomes.

Solution :

- **Hypothesis Testing :**Hypothesis testing is a statistical method used to make inferences about population parameters based on sample data. It involves the formulation of a null hypothesis (H0) and an alternative hypothesis (H1), and the collection of sample data to assess the evidence against the null hypothesis. The goal is to determine whether there is enough evidence to reject the null hypothesis in favor of the alternative hypothesis.

1. **Formulate Hypotheses:**
   - **Null Hypothesis (H0):** The average caffeine content per serving is 80 mg ($\mu=80$).
   - **Alternative Hypothesis (H1):** The average caffeine content per serving is different from 80 mg ($\mu\neq80$).
2. **Statistical Test:**
   - A t-test is appropriate since you are comparing a sample mean to a known population mean, and the sample size is small.
3. **Data Collection:**
   - Randomly select 30 cans of the energy drink and measure the caffeine content in each.

4. **Conducting the Hypothesis Test:**
   a. **Collect Data:**Calculate the sample mean ($\bar{x}$) and standard deviation (s) from the 30 samples.
   b. **Set Significance Level (α):**Choose a significance level (α=0.05,0.01,0.10).
   c. **Calculate the Test Statistic (t-value):**Use the formula $t=s/n\bar{x}-\mu$.
   d. **Determine Degrees of Freedom:**For a one-sample t-test, degrees of freedom (df) isn−1.
   e. **Find Critical Values or P-value:**Use a t-table or statistical software to find the critical t-values for a two-tailed test at the chosen significance level.
   f. **Make a Decision:**If the t-value falls outside the critical region, reject the null hypothesis. If it falls inside, fail to reject.
   g. **Interpretation:**If you reject the null hypothesis, there is enough evidence to suggest that the average caffeine content per serving is different from 80 mg. If you fail to reject the null hypothesis, there is not enough evidence to suggest a difference in the average caffeine content.
5. **Conclusion:**

   Draw conclusions about the energy drink's caffeine content, considering both statistical and practical significance. Consider decisions relevant to the context of the problem.

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
# Generate two samples for demonstration purposes
np.random.seed(42)
sample1 = np.random.normal(loc=10, scale=2, size=30)
sample2 = np.random.normal(loc=12, scale=2, size=30)
# Perform a two-sample t-test
t_statistic, p_value = stats.ttest_ind(sample1, sample2)
# Set the significance level
alpha = 0.05
print("Results of Two-Sample t-test:")
print(f"t-statistic: {t_statistic}")
print(f"p-value: {p_value}")
print(f"Degrees of Freedom: {len(sample1) + len(sample2) - 2}")
# Plot the distributions
plt.figure(figsize=(10, 6))
plt.hist(sample1, alpha=0.5, label='Sample 1', color='blue')
plt.hist(sample2, alpha=0.5, label='Sample 2', color='orange')
plt.axvline(np.mean(sample1), color='blue', linestyle='dashed', linewidth=2)
plt.axvline(np.mean(sample2), color='orange', linestyle='dashed', linewidth=2)
plt.title('Distributions of Sample 1 and Sample 2')
plt.xlabel('Values')
plt.ylabel('Frequency')
```

```
plt.legend()
# Highlight the critical region if null hypothesis is rejected
if p_value< alpha:
critical_region = np.linspace(min(sample1.min(), sample2.min()), max(sample1.max(), sample2.max()), 1000)
plt.fill_between(critical_region, 0, 5, color='red', alpha=0.3, label='Critical Region')
# Show the observed t-statistic
plt.text(11, 5, f'T-statistic: {t_statistic:.2f}', ha='center', va='center', color='black', backgroundcolor='white')
# Show the plot
plt.show()
# Draw Conclusions
# Drawing Conclusions
if p_value< alpha:
  if np.mean(sample1) >np.mean(sample2):
print("Conclusion: There is significant evidence to reject the null hypothesis.")
print("Interpretation: The mean caffeine content of Sample 1 is significantly higher than that of Sample 2.")
    # Additional context and practical implications can be added here.
  else:
print("Conclusion: There is significant evidence to reject the null hypothesis.")
print("Interpretation: The mean caffeine content of Sample 2 is significantly higher than that of Sample 1.")
    # Additional context and practical implications can be added here.
else:
print("Conclusion: Fail to reject the null hypothesis.")
print("Interpretation: There is not enough evidence to claim a significant difference between the means.")
```

Practical No. 5

5. ANOVA (Analysis of Variance)

- Perform one-way ANOVA to compare means across multiple groups.
- Conduct post-hoc tests to identify significant differences between group means.

Solution :

```
import scipy.stats as stats
from statsmodels.stats.multicomp import pairwise_tukeyhsd

# Sample data (replace this with your actual data)
group1 = [23, 25, 29, 34, 30]
group2 = [19, 20, 22, 25, 24]
group3 = [15, 18, 20, 21, 17]
group4 = [28, 24, 26, 30, 29]
```

```
# Combine all data into a single array
all_data = group1 + group2 + group3 + group4

# Create corresponding group labels
group_labels = ['Group1'] * len(group1) + ['Group2'] * len(group2) + ['Group3'] * len(group3) + ['Group4'] * len(group4)

# Perform one-way ANOVA
f_statistic, p_value = stats.f_oneway(group1, group2, group3, group4)

# Print ANOVA results
print("One-way ANOVA:")
print("F-statistic:", f_statistic)
print("P-value:", p_value)

# Perform Tukey-Kramer post-hoc test
tukey_results = pairwise_tukeyhsd(all_data, group_labels)

# Print Tukey-Kramer results
print("\nTukey-Kramer post-hoc test:")
print(tukey_results)
```

Practical No. 6

6. Regression and Its Types

- Implement simple linear regression using a dataset
- Explore and interpret the regression model coefficients and goodness-of-fit measures.
- Extend the analysis to multiple linear regression and assess the impact of additional predictors.

Solution :

Simple Linear Regression

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the California Housing dataset
```

```
housing = fetch_california_housing()
housing_df = pd.DataFrame(housing.data, columns=housing.feature_names)
print(housing_df)
housing_df['PRICE'] = housing.target
# Selecting the feature and target variable
X = housing_df[['AveRooms']]  # Average number of rooms
y = housing_df['PRICE'] # Target variable - Price

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Creating a linear regression model
model = LinearRegression()

# Fitting the model
model.fit(X_train, y_train)

# Predicting on the test set
y_pred = model.predict(X_test)

# Model evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
print("Intercept:", model.intercept_)
print("Coefficient:", model.coef_)
```

Multiple Linear Regression

```
# Selecting multiple features
X = housing_df.drop('PRICE', axis=1)  # Using all features except the target variable
y = housing_df['PRICE']  # Target variable - Price

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Creating a linear regression model
model = LinearRegression()
# Fitting the model
```

```
model.fit(X_train, y_train)
# Predicting on the test set
y_pred = model.predict(X_test)
# Model evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)
```

Practical No. 7

7. Logistic Regression and Decision Tree

- Build a logistic regression model to predict a binary outcome.
- Evaluate the model's performance using classification metrics (e.g.. accuracy, precision, recall).
- Construct a decision tree model and interpret the decision rules for classification.

Solution :

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, classification_report

# Load the Iris dataset
iris = load_iris()
iris_df = pd.DataFrame(data= np.c_[iris['data'], iris['target']], columns= iris['feature_names'] + ['target'])

# For binary classification, let's consider only two classes (0 and 1)
binary_df = iris_df[iris_df['target'] != 2]

# Selecting features and target variable
X = binary_df.drop('target', axis=1)
y = binary_df['target']
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Logistic Regression model
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)

# Predictions
y_pred_logistic = logistic_model.predict(X_test)

# Evaluate logistic regression model
print("Logistic Regression Metrics:")
print("Accuracy:", accuracy_score(y_test, y_pred_logistic))
print("Precision:", precision_score(y_test, y_pred_logistic))
print("Recall:", recall_score(y_test, y_pred_logistic))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_logistic))

# Decision Tree model
decision_tree_model = DecisionTreeClassifier()
decision_tree_model.fit(X_train, y_train)

# Predictions
y_pred_tree = decision_tree_model.predict(X_test)

# Evaluate decision tree model
print("\nDecision Tree Metrics:")
print("Accuracy:", accuracy_score(y_test, y_pred_tree))
print("Precision:", precision_score(y_test, y_pred_tree))
print("Recall:", recall_score(y_test, y_pred_tree))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_tree))
```

Practical No. 8

8. K-Means Clustering

- Apply the K-Means algorithm to group similar data points into clusters.
- Determine the optimal number of clusters using elbow method or silhouette analysis.
- Visualize the clustering results and analyze the cluster characteristics.

Solution :

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load the Iris dataset
iris = load_iris()
iris_df = pd.DataFrame(data= np.c_[iris['data'], iris['target']], columns= iris['feature_names'] + ['target'])

# Selecting features
X = iris_df.drop('target', axis=1)

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Determine the optimal number of clusters using the elbow method
wcss = []
for i in range(1, 11):
kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
kmeans.fit(X_scaled)
wcss.append(kmeans.inertia_)

# Plot the elbow method
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')  # Within cluster sum of squares
plt.show()

# Based on the elbow method, let's choose the optimal number of clusters as 3

# Apply K-Means clustering with the optimal number of clusters
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
kmeans.fit(X_scaled)

# Visualize the clustering results
```

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
centroids = kmeans.cluster_centers_
labels = kmeans.labels_
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='viridis', s=50, alpha=0.5)
plt.scatter(centroids[:, 0], centroids[:, 1], marker='o', c='red', s=200, edgecolor='k')
plt.title('K-Means Clustering')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar()
plt.show()

# Analyze cluster characteristics
cluster_df = iris_df.copy()
cluster_df['cluster'] = labels

for cluster in sorted(cluster_df['cluster'].unique()):
print(f"\nCluster {cluster}:")
  print(cluster_df[cluster_df['cluster'] == cluster].describe())
```

Practical No. 9

9. Principal Component Analysis (PCA)

- Perform PCA on a dataset to reduce dimensionality.
- Evaluate the explained variance and select the appropriate number of principal components.
- Visualize the data in the reduced-dimensional space.

Solution :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load the Iris dataset
iris = load_iris()
iris_df = pd.DataFrame(data= np.c_[iris['data'], iris['target']], columns= iris['feature_names'] + ['target'])
# Selecting features and target variable
X = iris_df.drop('target', axis=1)
```

```
y = iris_df['target']
# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Perform PCA
pca = PCA()
X_pca = pca.fit_transform(X_scaled)

# Explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_

# Plot explained variance ratio
plt.figure(figsize=(8, 6))
plt.plot(np.cumsum(explained_variance_ratio), marker='o', linestyle='--')
plt.title('Explained Variance Ratio')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.grid(True)
plt.show()

# Select the appropriate number of principal components
cumulative_variance_ratio = np.cumsum(explained_variance_ratio)
n_components = np.argmax(cumulative_variance_ratio>= 0.95) + 1

print(f"Number of principal components to explain 95% variance: {n_components}")

# Reduce dimensionality using the selected number of principal components
pca = PCA(n_components=n_components)
X_reduced = pca.fit_transform(X_scaled)

# Visualize data in the reduced-dimensional space
plt.figure(figsize=(8, 6))
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y, cmap='viridis', s=50, alpha=0.5)
plt.title('Data in Reduced-dimensional Space')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(label='Target')
plt.show()
```

Practical No. 10

10. Data Visualization and Storytelling

- Create meaningful visualizations using data visualization tools
- Combine multiple visualizations to tell a compelling data story.
- Present the findings and insights in a clear and concise manner.

Solution :

Consider an example scenario where we analyze customer churn data for a telecommunications company.

Step 1: Data Understanding and Preparation

Assuming we have a dataset containing information about customer demographics, services subscribed, and whether they churned or not.

Step 2 : Exploratory Data Analysis (EDA)

Perform exploratory data analysis to understand the characteristics of the dataset and identify patterns, trends, and correlations.

Step 3 : Creating Meaningful Visualizations

- **Churn Rate by Gender:** A bar chart showing the churn rate by gender.
- **Churn Rate by Age Group:** A histogram or box plot showing the distribution of age among churned and retained customers.
- **Churn Rate by Service Type:** A pie chart or bar chart showing the percentage of churned customers based on the type of services subscribed.
- **Correlation Matrix:** A heatmap showing the correlation between different variables, highlighting factors that are strongly correlated with churn.
- **Customer Tenure vs. Churn:** A scatter plot showing the relationship between customer tenure and churn status.
- **Customer Segmentation:** Use clustering algorithms like K-Means to segment customers based on their behavior and visualize the clusters using a scatter plot with different colors for each cluster.

Step 4 : Combining Visualizations

Combine the above visualizations into a dashboard or a series of slides to tell a compelling data story. For example:

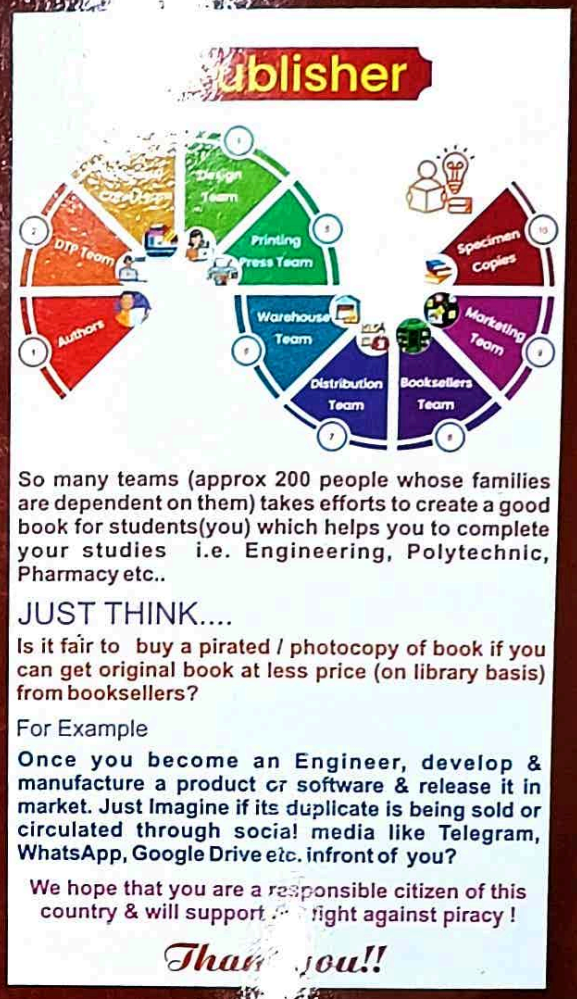
- Start with an overview of the churn rate and its distribution across different demographics.
- Dive deeper into specific factors contributing to churn, such as age, services subscribed, and tenure.
- Highlight any correlations or patterns observed in the data.
- Conclude with actionable insights and recommendations for reducing churn.

Step 5 : Presenting Findings

Present the findings and insights in a clear and concise manner, using storytelling techniques to engage the audience and emphasize key points. Use titles, labels, and annotations to guide the audience through the visualizations and explain the significance of each insight.

```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Churn rate by gender (using seaborn)
sns.countplot(data=df, x='gender', hue='churn')
plt.title('Churn Rate by Gender')

# Churn rate by age group (using plotly)
fig = px.histogram(df, x='age', color='churn', nbins=20, histnorm='percent')
fig.update_layout(title='Churn Rate by Age Group', xaxis_title='Age', yaxis_title='% of Customers')

# Churn rate by service type (using matplotlib)
service_churn = df.groupby('service_type')['churn'].mean()
plt.pie(service_churn, labels=service_churn.index, autopct='%1.1f%%')
plt.title('Churn Rate by Service Type')

# Correlation matrix (using seaborn)
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')

# Customer tenure vs. churn (using plotly)
fig = px.scatter(df, x='tenure', y='churn', color='churn')
fig.update_layout(title='Customer Tenure vs. Churn', xaxis_title='Tenure', yaxis_title='Churn (1=Yes, 0=No)')

# Customer segmentation (using plotly)
fig = px.scatter(df, x='feature1', y='feature2', color='cluster')
fig.update_layout(title='Customer Segmentation')
```

❑❑❑



Head Office - TECHKNOWLEDGE PUBLICATIONS :
B/5, Maniratna Complex, Taware Colony,
Aranyeshwar Corner, Pune - 411009. Maharashtra, India.
Tel. : 91-20-24221234, 91-20-24225678. M: 93703 14831.

PRINTED ON RECYCLED PAPER

ISBN : 978-93-5563-910-3

Price ₹ 225/-

MCE37A

Distributors

Vidyarthi Sales Agencies : T. : (022) 23867279, 98197 76110

Ved Book Distributors : M.: 80975 71421 / 92208 77214 / 80973 75002 (For Library Orders Also)

Our Branches : Pune | Mumbai | Kolhapur | Nagpur | Solapur | Nashik

Email : info@techknowledgebooks.com
Website : www.techknowledgebooks.com

Books are also available at : amazon

Like us at: TechknowledgePublications