Dr. D. Y. Patil Pratishthan's
D. Y. PATIL COLLEGE OF ENGINEERING, AKURDI, PUNE-44.

DEPARTMENT OF
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

LAB MANUAL
Computer Laboratory-IV
(B. E.)

Academic Year 2024-25
SEM II

Lab Incharge
Dr. Bhagyashree A. Tingare

Computer Laboratory - IV

Course Code	Course Name	Teaching Scheme(Hrs./ Week)	Credits
417531(B)	Computer Laboratory-IV : Big Data Analytics	2	2
417531(D)	Computer Laboratory-IV : Deep Learning	2	2
417532(B)	Computer Laboratory-IV : Business Intelligence	2	2

Course Objectives:

- To understand Big Data Analytics Concepts.
- To understand fundamental concepts of Deep Learning.
- To introduce the concepts and components of Business Intelligence (BI).

Course Outcomes:

After completion of the course, learners should be able to

- CO1: Apply basic principles of elective subjects to problem solving and modeling
- CO3: Design and develop applications on subjects of their choice.

D Y Patil College of Engineering, Akurdi, Pune

Department of Artificial Intelligence & Data Science

Vision of Institute

"Empowerment through knowledge"

Mission of Institute

1. To educate the students to transform them as professionally competent and quality conscious engineers.
2. To Provide Conducive Environment for Teaching Learning and overall personality development.
3. To culminate the Institute into an International seat of excellence

VISION of Department

Developing highly skilled and competent IT professional for sustainable growth in the field of Artificial Intelligence and Data science.

MISSION of Department

1. To empower students for developing intelligent systems and innovative products for societal problems.
2. To build strong foundation in Data computation, Intelligent Systems that enables self-development entrepreneurship and Intellectual property.
3. To develop competent and skilled IT professional by imparting global skills and technologies for serving society.

D Y Patil College of Engineering, Akurdi, Pune
Department of Artificial Intelligence & Data Science

Program Outcomes (PO's)

PO1	Engineering knowledge	Apply the knowledge of mathematics, science, Engineering fundamentals, and an Engineering specialization to the solution of complex Engineering problems.
PO2	Problem analysis	Identify, formulate, review research literature and analyze complex Engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and Engineering sciences.
PO3	Design / Development of Solutions	Design solutions for complex Engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and Environmental considerations.
PO4	Conduct Investigations of Complex Problems	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	Modern Tool Usage	Create, select, and apply appropriate techniques, resources, and modern Engineering and IT tools including prediction and modeling to complex Engineering activities with an understanding of the limitations.
PO6	The Engineer and Society	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practices.
PO7	Environment and Sustainability	Understand the impact of the professional Engineering solutions in societal and Environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	Ethics	Apply ethical principles and commit to professional ethics and responsibilities and norms of Engineering practice.
PO9	Individual and Team Work	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	Communication Skills	Communicate effectively on complex Engineering activities with the Engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	Project Management and Finance	Demonstrate knowledge and understanding of Engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary Environments.
PO12	Life-long Learning	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

D Y Patil College of Engineering, Akurdi, Pune
Department of Artificial Intelligence & Data Science

Program Specific Outcomes (PSO's)

PSO1	Professional Skills	The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, networking, artificial intelligence and data science for efficient design of computer-based systems of varying complexities.
PSO2	Problem-Solving Skills	The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.
PSO3	Successful Career and Entrepreneurship	The ability to employ modern computer languages, environments and platforms in creating innovative career paths to be an entrepreneur and to have a zest for higher studies.

Program Educational Objective (PEO's)

PEO1	To prepare globally competent graduates having strong fundamentals and domain knowledge to provide effective solutions for engineering problems.
PEO2	To prepare the graduates to work as a committed professional with strong professional ethics and values, sense of responsibilities, understanding of legal, safety, health, societal, cultural and environmental issues.
PEO3	To prepare committed and motivated graduates with research attitude, lifelong learning, investigative approach, and multidisciplinary thinking.
PEO4	To prepare the graduates with strong managerial and communication skills to work effectively as individual as well as in teams.

D.Y. Patil College of Engineering, Akurdi, Pune 44.
Department of AI & DS Engineering

SUBJECT : 417531(B) - Big Data Analytics

INDEX

Sr. No.	Date of Performance	Title of Assignment	Page No.	Sign/ Remarks
1.		Set up and Configuration Hadoop Using CloudEra/ Google Cloud BigQuery. Databricks Lakehouse Platform. Snowflake. Amazon Redshift.		
2.		Develop a MapReduce program to calculate the frequency of a given word in a given file.		
3.		Implement Matrix Multiplication using Map-Reduce		
4.		Develop a MapReduce program to find the grades of students.		
5.		Mini Project		

CERTIFICATE

Certified that Mr./Miss._____ Roll No._____ of _____ class has carried out above practical/term work within the four wall of the college. His/Her performance is satisfactory & attendance is_____%.

DATE Staff Member in-charge Head of the Department Principal

D.Y. Patil College of Engineering, Akurdi, Pune 44.

Department of AI & DS Engineering

SUBJECT : 417531(D) - Deep Learning

INDEX

CERTIFICATE

Certified that Mr./Miss._____ Roll No._____ of _____ class has carried out above practical/term work within the four wall of the college. His/Her performance is satisfactory & attendance is_____%.

DATE Staff Member in-charge Head of the Department Principal

D.Y. Patil College of Engineering, Akurdi, Pune 44.

Department of AI & DS Engineering

SUBJECT : 417532(B) - Business Intelligence

INDEX

CERTIFICATE

Certified that Mr./Miss._____ Roll No._____ of _____ class has carried out above practical/term work within the four wall of the college. His/Her performance is satisfactory & attendance is_____%.

DATE Staff Member in-charge Head of the Department Principal

Lab Assignment No.	1
Title	Set up and Configuration of Hadoop Using CloudEra, Google Cloud BigQuery, Databricks Lakehouse Platform, Snowflake, and Amazon Redshift
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

Lab – 1:- Set up and Configuration Hadoop Using CloudEra/ Google Cloud BigQuery. Databricks Lakehouse Platform. Snowflake. Amazon Redshift.

1.1 Title:
Set up and Configuration of Hadoop Using CloudEra, Google Cloud BigQuery, Databricks Lakehouse Platform, Snowflake, and Amazon Redshift

1.2 Problem Definition:
In the era of big data, organizations face challenges in managing, processing, and analyzing massive datasets. Hadoop and modern data platforms like CloudEra, Google Cloud BigQuery, Databricks Lakehouse, Snowflake, and Amazon Redshift provide scalable solutions for these tasks. However, the effective setup, configuration, and integration of these tools remain complex. This project addresses the challenges of configuring these platforms to create a unified ecosystem for big data processing and analytics.

1.3 Requirements:
1. **Software Requirements:**
   - Hadoop framework (HDFS, MapReduce, YARN)
   - CloudEra Manager
   - Google Cloud BigQuery account
   - Databricks Lakehouse Platform access
   - Snowflake trial/paid account
   - Amazon Redshift cluster
2. **Hardware Requirements:**
   - Minimum 8-core CPU, 16 GB RAM for Hadoop setup (on-premises or virtualized)
   - Stable internet connection for cloud platform access
3. **Technical Skills:**
   - Understanding of Hadoop ecosystem
   - Familiarity with cloud platforms (Google Cloud, AWS, Databricks)
   - SQL and Python for querying and scripting

1.4 Learning Objectives:
- Understand the installation and configuration of Hadoop clusters using CloudEra.
- Explore Google Cloud BigQuery for large-scale data analytics.
- Learn to set up and utilize the Databricks Lakehouse Platform for unified data analytics.
- Configure and query Snowflake for efficient data warehousing.
- Deploy and manage Amazon Redshift clusters for big data processing.
- Compare and integrate multiple big data platforms to achieve an optimized data architecture.

1.5 Outcomes:
- Successfully set up Hadoop using CloudEra and explore its core components (HDFS, MapReduce, YARN).
- Perform advanced analytics on structured and semi-structured data using Google Cloud BigQuery.
- Leverage Databricks Lakehouse for a unified approach to data engineering, machine learning, and BI.

- Implement scalable data warehousing solutions on Snowflake and Amazon Redshift.
- Gain hands-on experience in configuring cloud-based data platforms for business use cases.

1.6 Test Case Table:

Test Case ID	Scenario	Expected Outcome	Result
TC01	Set up Hadoop cluster using CloudEra	Hadoop cluster successfully configured and operational	Pass/Fail
TC02	Execute SQL queries on Google Cloud BigQuery	Queries return accurate results within the expected time	Pass/Fail
TC03	Configure Databricks Lakehouse for analytics	Platform configured, and a basic ML pipeline runs	Pass/Fail
TC04	Set up Snowflake and perform data queries	Snowflake configured, and queries return expected outputs	Pass/Fail
TC05	Create Amazon Redshift cluster and query data	Cluster operational, and queries execute successfully	Pass/Fail
TC06	Integrate platforms for data flow	Data seamlessly flows between platforms without issues	Pass/Fail

1.7 Theory Concepts:

Hadoop Ecosystem:

The Hadoop ecosystem is a framework of tools and technologies that allows the storage, processing, and analysis of large datasets in a distributed manner. It comprises the following core components:

1. HDFS (Hadoop Distributed File System):
   - A scalable, fault-tolerant storage system that splits large files into blocks and distributes them across multiple nodes.
   - Ensures redundancy by replicating data blocks across nodes.
2. MapReduce:
   - A programming model for processing large datasets in parallel across a Hadoop cluster.
   - Divides tasks into smaller sub-tasks (Map phase) and combines the results (Reduce phase).
3. YARN (Yet Another Resource Negotiator):
   - Manages cluster resources and schedules tasks.
   - Allows multiple data processing engines (MapReduce, Spark) to run simultaneously.

Hadoop is often used for batch processing of unstructured or semi-structured data.

CloudEra Manager:

CloudEra Manager simplifies the management of Hadoop clusters through an intuitive interface. It offers:

- Cluster Deployment: Automates the installation and configuration of Hadoop and its ecosystem components.
- Monitoring: Provides real-time dashboards to monitor cluster health, resource utilization, and job performance.

- Security: Integrates with Kerberos for authentication and supports data encryption.
- Data Governance: Offers tools like Apache Atlas for metadata management and lineage tracking.

  CloudEra Manager enhances the operational efficiency of Hadoop clusters, making them enterprise-ready.

Google Cloud BigQuery:

BigQuery is a fully managed, serverless data warehouse designed for large-scale data analytics. Key features include:

- Scalability: Handles petabytes of data with ease.
- Speed: Executes queries in seconds using distributed query execution.
- SQL-Based Interface: Supports standard SQL, making it accessible to data analysts.
- Integration: Seamlessly integrates with Google Cloud services, including Dataflow, Cloud Storage, and AI tools.
- Pay-as-You-Go Model: Charges based on data storage and query execution, optimizing costs.

  BigQuery is ideal for real-time analytics, data lake integration, and handling structured datasets.

Databricks Lakehouse Platform:

The Databricks Lakehouse Platform unifies data engineering, machine learning, and analytics into a single environment. It merges the best features of data lakes and data warehouses:

- Data Lake: Stores raw, unstructured, and semi-structured data at scale.
- Data Warehouse: Provides structured data models for analytics and business intelligence.
- Unified Workspace: Allows collaboration among data engineers, data scientists, and analysts.
- Delta Lake: An optimized storage layer that ensures ACID transactions and reliable data pipelines.
- Machine Learning Integration: Built-in ML libraries and support for frameworks like TensorFlow and PyTorch.

  The Lakehouse approach simplifies data workflows, reducing duplication and costs.

Snowflake:

Snowflake is a cloud-native data warehouse designed to handle diverse workloads. Key features include:

- Separation of Compute and Storage: Allows independent scaling of storage and compute resources.
- Multi-Cloud Compatibility: Supports AWS, Azure, and Google Cloud, enabling cross-cloud operations.
- Data Sharing: Provides secure and seamless sharing of data across organizations.
- SQL-Based: Easy to use for developers and analysts familiar with SQL.
- Data Integration: Connects with ETL tools, BI platforms, and streaming services.

  Snowflake's elasticity and performance make it ideal for ad hoc analysis and large-scale reporting.

Amazon Redshift:

Amazon Redshift is a fully managed data warehouse designed for high-performance analytics.

Key features include:

- Massively Parallel Processing (MPP): Executes queries across multiple nodes in parallel for faster results.
- Columnar Storage: Optimized for analytical queries by storing data in columns instead of rows.
- Scalability: Easily scales up or down by adding/removing nodes.
- Integration with AWS Ecosystem: Works seamlessly with services like S3, Glue, and Lambda.
- Cost-Effectiveness: Offers reserved instances and on-demand pricing to optimize costs.
  Redshift is suitable for data lake integration, complex queries, and business intelligence reporting.

Data Integration:

Modern big data architectures often require seamless data flow between multiple platforms. Integration strategies include:

- ETL (Extract, Transform, Load): Moving data from source systems to target platforms while transforming it to meet requirements.
- API-Based Integration: Leveraging platform APIs for data exchange in real time.
- Data Federation: Querying data across multiple systems without physically moving it.
- Hybrid Architectures: Combining on-premises and cloud platforms to meet specific needs.
  Effective integration ensures data consistency, reduces latency, and optimizes analytics workflows.

1.8 Conclusion:

This project demonstrates the setup, configuration, and effective utilization of modern big data platforms. It provides insights into the strengths and limitations of each tool, offering a practical foundation for businesses to design scalable and efficient data ecosystems. Mastery of these platforms equips individuals with the ability to address real-world challenges in big data analytics, engineering, and storage.

Lab Assignment No.	2
Title	Develop a MapReduce Program to Calculate the Frequency of a Given Word in a Given File
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

Lab – 2: Develop a MapReduce program to calculate the frequency of a given word in a given file.

1.1 Title:

Develop a MapReduce Program to Calculate the Frequency of a Given Word in a Given File

1.2 Problem Definition:

Large-scale text data processing often involves counting the occurrence of specific words to derive insights. This project aims to develop a MapReduce program that calculates the frequency of a specified word in a given file. The program will split the file into smaller chunks for parallel processing, count the occurrences of the target word in each chunk, and then aggregate the results to determine the total frequency efficiently.

1.3 Requirements:

1. **Software Requirements:**
   - Hadoop framework (installed and configured)
   - Java Development Kit (JDK) or Python for MapReduce code
   - Text file for input
2. **Hardware Requirements:**
   - Hadoop-compatible cluster with sufficient nodes and resources
   - Minimum 8-core CPU, 16 GB RAM (for single-node setups)
3. **Technical Skills:**
   - Understanding of Hadoop and its MapReduce programming model
   - Proficiency in Java or Python
   - Basic text file manipulation

1.4 Learning Objectives:

- Understand the MapReduce programming paradigm for distributed data processing.
- Learn how to write a MapReduce job in Java or Python.
- Explore techniques for efficiently processing large text files in Hadoop.
- Gain experience with the Hadoop ecosystem, including setting up jobs and retrieving output.
- Analyze word frequency data to solve real-world text processing problems.

1.5 Outcomes:

- **D**evelop a working MapReduce program that calculates the frequency of a given word in a large text file.
- Successfully execute the program on a Hadoop cluster, processing input in parallel.
- Gain insights into the performance and scalability of MapReduce for text processing tasks.
- Understand how to deploy and troubleshoot MapReduce jobs in a distributed environment.

1.6 Test Case Table:

Test Case ID	Scenario	Expected Outcome	Result
TC01	Input file contains the target word multiple times	Total frequency matches the count in the file	Pass/Fail
TC02	Input file is empty	Program outputs a frequency of zero	Pass/Fail
TC03	Target word is not present in the input file	Program outputs a frequency of zero	Pass/Fail
TC04	File contains the target word in mixed cases	Frequency count is case-insensitive if specified	Pass/Fail
TC05	Very large input file	Program completes execution without performance issues	Pass/Fail

1.7 Theory Concepts:

MapReduce Programming Model:

MapReduce is a programming paradigm for processing large datasets in a distributed environment. It involves two main phases:

1. **Map Phase:**
   - The input data is split into chunks, and each chunk is processed independently by a mapper.
   - The mapper processes key-value pairs and produces intermediate key-value pairs.
   - In this program, the mapper reads lines of text, splits them into words, and emits a key-value pair for each occurrence of the target word (e.g., ("word", 1)).
2. **Reduce Phase:**
   - Reducers aggregate the intermediate results to produce the final output.
   - For this program, the reducer sums the counts of the target word across all mappers to calculate the total frequency.

Hadoop Ecosystem:

- Hadoop Distributed File System (HDFS) stores the input file and intermediate results across multiple nodes.
- YARN (Yet Another Resource Negotiator) manages the resources required for executing the MapReduce job.

Word Frequency Calculation:

- The program focuses on identifying occurrences of a specific word in the input text.
- **Steps involved:**
  1. Read the file line by line.
  2. Tokenize each line into words.

3. Emit (word, 1) for the target word found in each line.
4. Aggregate all 1s for the word in the Reduce phase.

Case Sensitivity:

- By default, word matching can be case-sensitive.
- To make the program case-insensitive, convert all words to lowercase during the Map phase.

Handling Large Files:

- Hadoop automatically splits large input files into smaller blocks for processing.
- Each block is processed by a separate mapper, enabling parallel computation.

1.8 Conclusion:

The MapReduce program successfully demonstrates the distributed processing capabilities of Hadoop. By calculating the frequency of a target word, the program showcases the scalability and efficiency of the MapReduce paradigm for text processing tasks. This solution can be extended to handle more complex data analytics workflows, making it valuable for big data applications.

Lab Assignment No.	3
Title	Implement Matrix Multiplication Using MapReduce
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

Lab 3 - Implement Matrix Multiplication using Map-Reduce

1.1 Title:

Implement Matrix Multiplication Using MapReduce

1.2 Problem Definition:

Matrix multiplication is a fundamental operation in various domains such as scientific computing, machine learning, and data analysis. For large matrices, the operation becomes computationally expensive. This project aims to implement matrix multiplication using the MapReduce programming model to distribute the computation across multiple nodes in a Hadoop cluster. The program will efficiently calculate the product of two matrices AAA and BBB, leveraging the parallel processing power of Hadoop.

1.3 Requirements:

1. **Software Requirements:**
   - **Hadoop framework (installed and configured)**
   - **Java Development Kit (JDK) or Python for MapReduce programming**
   - **Input matrices in a text file format**
2. **Hardware Requirements:**
   - Hadoop-compatible cluster with multiple nodes
   - Minimum 8-core CPU, 16 GB RAM (for single-node setups)
3. **Technical Skills:**
   - Knowledge of matrix multiplication algorithms
   - Familiarity with MapReduce programming
   - Proficiency in Java or Python

1.4 Learning Objectives:

- Understand the MapReduce framework and its suitability for distributed computation.
- Learn to express matrix multiplication as a distributed computation using MapReduce.
- Explore the use of key-value pairs for efficient data partitioning and communication between mappers and reducers.
- Gain hands-on experience with Hadoop cluster setup and execution of MapReduce jobs.

1.5 Outcomes:

- Implement a MapReduce program for matrix multiplication that handles large matrices efficiently.
- Successfully execute the program on a Hadoop cluster, demonstrating scalability and parallelism.
- Understand the trade-offs and challenges of distributed computation in MapReduce.
- Develop a reusable solution for large-scale matrix operations in distributed environments.

1.6 Test Case Table:

Test Case ID	Scenario	Expected Outcome	Result
TC01	Multiplication of two square matrices	Program outputs the correct product matrix	Pass/Fail
TC02	Multiplication of two non-square matrices	Program outputs the correct product matrix	Pass/Fail
TC03	Input matrices with zeros	Output matrix contains zero values where applicable	Pass/Fail
TC04	Very large input matrices	Program completes execution without memory issues	Pass/Fail
TC05	Mismatched dimensions	Program throws an error or handles gracefully	Pass/Fail

1.7 Theory Concepts:

Matrix Multiplication Basics:

Matrix multiplication involves computing the dot product of rows from matrix AAA with columns from matrix BBB. Given matrices AAA (dimensions m×nm \times nm×n) and BBB (dimensions n×pn \times pn×p), the resulting matrix CCC will have dimensions m×pm \times pm×p. Each element C[i][j]C[i][j]C[i][j] is computed as:

$C[i][j]=\Sigma k=1nA[i][k]\cdot B[k][j]C[i][j] = \sum_{k=1}^n A[i][k] \cdot B[k][j]C[i][j]=k=1\Sigma n\ A[i][k]\cdot$B[k][j]

MapReduce Programming for Matrix Multiplication:

The MapReduce framework splits the computation into two phases:

1. **Map Phase:**
   - The mapper reads the input matrices AAA and BBB.
   - For matrix AAA, the mapper emits key-value pairs of the form (i,k),(A,j,A[i][j])(i, k), (A, j, A[i][j])(i,k),(A,j,A[i][j]).
   - For matrix BBB, the mapper emits key-value pairs of the form (k,j),(B,i,B[k][j])(k, j), (B, i, B[k][j])(k,j),(B,i,B[k][j]).
2. **Reduce Phase:**
   - The reducer receives all key-value pairs with the same key (e.g., (i,j)(i, j)(i,j)).
   - It computes the product of matching elements and sums the results to produce each element of the output matrix.

Hadoop Ecosystem for Matrix Multiplication:

- HDFS: Distributes the input data (matrices) across the cluster nodes for parallel processing.
- YARN: Manages the computation resources and schedules the MapReduce job.

Key-Value Pair Design:

- Input matrices are represented in a text file format where each line corresponds to an element:
  For matrix AAA: i,j,A[i][j]i, j, A[i][j]i,j,A[i][j]
  For matrix BBB: j,k,B[j][k]j, k, B[j][k]j,k,B[j][k]
- Output key-value pairs are:
  Key: (i,j)(i, j)(i,j)
  Value: The computed value of C[i][j]C[i][j]C[i][j].

Handling Large Matrices:

- Partitioning: Hadoop automatically splits the input matrices into smaller chunks for parallel processing.
- Scalability: Adding more nodes to the cluster increases the processing capacity for larger matrices.

Error Handling and Validation:

- The program checks the compatibility of input matrix dimensions.
- Invalid or incomplete input data is logged and ignored to ensure smooth execution.

1.8 Conclusion:

The MapReduce implementation of matrix multiplication showcases the power of distributed computing for large-scale mathematical operations. By leveraging Hadoop's scalability and fault tolerance, the program efficiently handles matrices too large for single-node computation. This approach provides a foundation for implementing other complex numerical algorithms in a distributed environment.

Lab Assignment No.	4
Title	Develop a MapReduce Program to Find the Grades of Students
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

Lab – 4:- Develop a MapReduce program to find the grades of students

1.1 Title:

Develop a MapReduce Program to Find the Grades of Students

1.2 Problem Definition:

Student grading is a common task in educational institutions, where marks are aggregated across subjects to determine overall grades. For large datasets of student scores, manual processing is inefficient. This project aims to create a MapReduce program that calculates students' total or average marks and assigns grades based on a predefined grading scale. The program processes the dataset in a distributed manner, ensuring scalability and efficiency.

1.3 Requirements:

1. **Software Requirements:**
   - Hadoop framework (installed and configured)
   - Java Development Kit (JDK) or Python for MapReduce programming
   - Input file containing student scores
2. **Hardware Requirements:**
   - Hadoop-compatible cluster with sufficient nodes
   - Minimum 8-core CPU, 16 GB RAM (for single-node setups)
3. **Technical Skills:**
   - Knowledge of MapReduce programming
   - Understanding of grading scales and data aggregation
   - Proficiency in Java or Python

1.4 Learning Objectives:

- Understand how MapReduce processes data in a distributed environment.
- Learn to design a MapReduce workflow to aggregate scores and determine grades.
- Explore methods for efficient key-value pair manipulation and sorting in MapReduce.
- Gain experience in handling large datasets in Hadoop.

1.5 Outcomes:

- Implement a functional MapReduce program that processes student data to determine grades.
- Successfully execute the program on a Hadoop cluster, demonstrating scalability and parallelism.
- Understand the relationship between data distribution and computation efficiency in MapReduce.
- Develop a reusable framework for analyzing and processing large-scale academic datasets.

1.6 Test Case Table:

Test Case ID	Scenario	Expected Outcome	Result
TC01	Input file contains valid student data	Program outputs correct grades for all students	Pass/Fail
TC02	Input file is empty	Program outputs an empty result	Pass/Fail
TC03	Input contains invalid or incomplete records	Program skips invalid data and processes valid records	Pass/Fail
TC04	Dataset is very large	Program executes without performance issues	Pass/Fail
TC05	Grading scale is modified	Program adjusts grades according to the new scale	Pass/Fail

---

1.7 Theory Concepts:

MapReduce Programming Model for Grading:

The program processes student scores using the following steps:

1. **Map Phase:**
   - The input file contains records in the format: StudentID, Subject, Marks.
   - The mapper processes each record and emits key-value pairs:
     Key: StudentID
     Value: Marks
2. **Shuffle and Sort:**
   - Hadoop groups all marks by StudentID, ensuring all marks for a student are processed together in the Reduce phase.
3. **Reduce Phase:**
   - The reducer aggregates the marks for each student (e.g., total marks or average).
   - Based on the aggregated marks, the program assigns a grade according to the predefined grading scale (e.g., A, B, C).

Grading Scale:

A typical grading scale may look like this:

- 90–100: A
- 80–89: B
- 70–79: C
- 60–69: D
- Below 60: F

Hadoop Ecosystem:

- HDFS: Distributes the input file containing student data across nodes.
- YARN: Manages cluster resources for efficient MapReduce job execution.

Data Representation:

- Input Format: A text file where each line contains:
  StudentID, Subject, Marks
  Example:

101, Math, 95
101, English, 88
102, Math, 72

- Intermediate Key-Value Pairs:
  Key: StudentID
  Value: Marks
- Output Format: A text file where each line contains:
  StudentID, Total/Average Marks, Grade
  Example:

101, 91.5, A
102, 72, C

Error Handling:

- Skips records with invalid formats or missing fields.
- Logs invalid records for review.

Scalability:

- The program can process large datasets efficiently by distributing computation across Hadoop cluster nodes.
- Adding nodes to the cluster enhances processing capacity for larger datasets.

1.8 Conclusion:

The MapReduce program for grading students demonstrates the efficiency of distributed computing in handling large academic datasets. By leveraging Hadoop's fault-tolerance and scalability, the program ensures accurate and timely grade computation, making it a valuable tool for educational institutions and large-scale assessments.

Lab Assignment No.	Mini Project
Title	
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Big Data Analytics
Assessment Marks	
Assessor's Sign	

Mini Project

Mini Project Instructions

Project Title Options:

1. Facebook Sentiment Analysis System – Using PySpark for real-time sentiment analysis to assist in crisis management, service adjustments, and target marketing.
2. Medical Insurance Fraud Detection, Traffic Control Using Big Data, or Data Warehouse Design for an E-Commerce Site – Utilizing MapReduce techniques to process large-scale datasets.
3. Disease Prediction Based on Symptoms / Recommendation System / Smart Cities Using Big Data – Leveraging big data analytics to enhance decision-making and services.

Problem Statement:

Develop an application that adheres to the following requirements:

1. Select a problem statement from the above options.
2. Follow the Software Development Life Cycle (SDLC) and apply Software Engineering principles throughout the implementation.
3. Develop the application using:
    - Front End: Python (Flask/Django)/Java/PHP/.NET or any suitable language.
    - Backend: MongoDB/MySQL/Oracle/Any SQL or NoSQL database.
4. Implement real-time data processing where applicable, using Big Data tools (PySpark, Hadoop, MapReduce).
5. Conduct testing and validation using manual/automated testing approaches.

.

Project Report Requirements:

1. Title of the Project
2. Abstract
3. Introduction
4. Scope
5. Software/Hardware Requirements Specification
6. Source Code
7. Data Reports
8. Testing Document
9. Future Enhancements
10. Conclusion
11. References/Bibliography

Expected Outcome:

- Develop a functional Big Data-driven application using real-time data processing techniques.
- Implement Data Analytics and Machine Learning models where applicable.
- Ensure adherence to Software Development Life Cycle (SDLC).

Lab Assignment No.	1
Title	House Price Prediction using Linear Regression
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Deep Learning
Assessment Marks	
Assessor's Sign	

- Successfully validate, test, and document the project.

ASSIGNMENT No: 01
Title: House Price Prediction using Linear Regression
Problem Statement:
Predict house prices in the USA using a dataset and implement a Linear Regression model to estimate the selling price of a house.
Prerequisite:
Basics of Python and Machine Learning
Software Requirements:
Jupyter Notebook, Python Libraries (pandas, numpy, scikit-learn, matplotlib)
Hardware Requirements:
PIV, 2GB RAM, 500 GB HDD
Learning Objectives:

- Learn to preprocess and analyze housing data.
- Understand the working of a Linear Regression model.
- Build a predictive model for house price estimation.

Outcomes:
After completing this assignment, students will be able to:

- Build and evaluate a Linear Regression model for predicting house prices.
- Understand how to interpret model coefficients and performance metrics.

Theory:

Linear Regression for House Price Prediction
1. Introduction
Linear Regression is a fundamental **supervised machine learning algorithm** used for predictive modeling. It establishes a relationship between **a dependent variable** (house price) and **one or more independent variables** (house features) by fitting a linear equation to the data.
This model is widely used in real estate to predict house prices based on factors like **size, location, number of bedrooms, amenities, etc.**

---

2. Understanding the Linear Regression Model
Mathematical Equation:
The general equation of a linear regression model is:
$y=\beta0+\beta1x1+\beta2x2+...+\betanxn+\varepsilon$ y = \beta_0 + \beta_1x_1 + \beta_2x_2 + ... + \beta_nx_n + \varepsilon
Where:

- yy = **Dependent variable** (House price)
- x1,x2,...,xnx_1, x_2, ..., x_n = **Independent variables** (House features like size, location, number of rooms)
- β0\beta_0 = **Intercept** (Value of yy when all independent variables are zero)
- β1,β2,...,βn\beta_1, \beta_2, ..., \beta_n = **Coefficients** (Weights assigned to each feature)
- ε\varepsilon = **Error term** (Difference between actual and predicted values)

Types of Linear Regression:

1. **Simple Linear Regression:** One independent variable (e.g., predicting house price based only on square footage). $y=\beta0+\beta1x+\varepsilon$ y = \beta_0 + \beta_1x + \varepsilon
2. **Multiple Linear Regression:** More than one independent variable (e.g., predicting house price using size, location, and number of rooms).

y=β0+β1x1+β2x2+β3x3+...+βnxn+εy = \beta_0 + \beta_1x_1 + \beta_2x_2 + \beta_3x_3 + ... + \beta_nx_n + \varepsilon

---

3. Applications of Linear Regression in Real Estate

Linear Regression is widely used in real estate for various purposes:

✅ **Predicting house prices** based on size, number of bedrooms, location, and amenities.

✅ **Understanding the impact of different factors** on house pricing.

✅ **Identifying overpriced or underpriced properties** by comparing predicted vs. actual prices.

✅ **Forecasting future trends** in the housing market.

---

4. Evaluation Metrics

To measure the performance of a linear regression model, we use the following metrics:

Mean Squared Error (MSE):

Measures the average squared difference between actual and predicted house prices. Lower MSE indicates a better model.

MSE=1nΣ(yi−y^i)2MSE = \frac{1}{n} \sum (y_i - \hat{y}_i)^2

Root Mean Squared Error (RMSE):

The square root of MSE, making it easier to interpret.

RMSE=MSERMSE = \sqrt{MSE}

Mean Absolute Error (MAE):

Measures the average absolute difference between actual and predicted values.

MAE=1nΣ | yi−y^i | MAE = \frac{1}{n} \sum |y_i - \hat{y}_i|

R-squared (R^2):

Represents the proportion of variance in the dependent variable that is explained by the independent variables.

R2=1−SSresidualSStotalR^2 = 1 - \frac{SS_{residual}}{SS_{total}}

- R2R^2 close to 1 indicates a good fit.
- R2R^2 close to 0 means the model does not explain the data well.

---

5. Steps to Build a Linear Regression Model for House Price Prediction

Step 1: Load and Preprocess the Dataset

- Handle missing values
- Encode categorical variables
- Normalize numerical features

Step 2: Split the Dataset

- Divide data into **training** and **testing** sets (e.g., 80% train, 20% test).

Step 3: Train the Linear Regression Model

- Fit the model using **training data** to learn the relationship between features and house price.

Step 4: Evaluate the Model

- Use metrics like **MSE, RMSE, and R^2** to assess model performance.

Step 5: Predict House Prices for New Data

Lab Assignment No.	2
Title	Multiclass Classification using Convolutional Neural Networks (CNN)
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Deep Learning
Assessment Marks	
Assessor's Sign	

- Use the trained model to predict prices of new houses based on their features.

Conclusion:
I have understood how to build a Linear Regression model to predict house prices and evaluate its performance using appropriate metrics.

ASSIGNMENT No: 02
Title: Multiclass Classification using Convolutional Neural Networks (CNN)
Problem Statement:
Build a Multiclass Classifier using a CNN model. Use the MNIST dataset or any other suitable dataset to classify images into multiple classes.
Prerequisite:
Basics of Python, Deep Learning, and CNNs
Software Requirements:

- Jupyter Notebook or Google Colab
- Python Libraries: TensorFlow/Keras, pandas, numpy, matplotlib, scikit-learn

Hardware Requirements:

- GPU-enabled system (optional for faster training)
- 4 GB RAM (minimum)
- 500 GB HDD

Learning Objectives:

1. Learn to preprocess data for multiclass classification tasks.
2. Understand how to define and train a CNN model.
3. Evaluate multiclass classification performance using metrics like the confusion matrix.

Outcomes:

1. Build a CNN for multiclass classification tasks.
2. Preprocess datasets effectively for image-based models.
3. Evaluate classification results using appropriate metrics.

Theory:

Theory: Multiclass Classification with CNN

Convolutional Neural Networks (CNNs) are powerful deep learning models widely used in image classification tasks due to their ability to automatically learn spatial hierarchies of features. In a **multiclass classification** setting, the model assigns each input image to one of multiple possible categories. Unlike binary classification, which distinguishes between two classes, multiclass classification deals with three or more distinct categories.

Key Features of CNN for Multiclass Classification

1. **Convolutional Layers:**
    - These layers extract key features from images by applying filters (kernels) that detect patterns such as edges, textures, and shapes.
    - Multiple convolutional layers stacked together help capture more complex patterns.

2. **Pooling Layers:**
    - Pooling layers (such as Max Pooling) reduce the spatial dimensions of feature maps, making computations more efficient and reducing the risk of overfitting.
    - They help retain essential features while discarding less relevant details.
3. **Fully Connected Layers:**
    - After convolutional and pooling layers, fully connected layers (dense layers) help map extracted features to the final output classes.
    - These layers allow the network to learn complex relationships between high-level features.
4. **Softmax Activation in the Output Layer:**
    - The final output layer uses a **Softmax** activation function, which converts raw logits into a probability distribution across all possible classes.
    - The class with the highest probability is assigned as the model's prediction.

Applications of Multiclass Classification Using CNN

CNNs are widely used in various real-world applications requiring classification across multiple categories:

- **Digit recognition** using the MNIST dataset (0-9 handwritten digits).
- **Handwritten text classification** to recognize different characters or words.
- **Object classification** in real-world images (e.g., classifying animals, vehicles, or furniture).
- **Medical imaging** to classify diseases from X-rays, MRI scans, or CT scans.
- **Fashion classification** using datasets like Fashion-MNIST to categorize clothing items.

Evaluation Metrics for Multiclass Classification

Evaluating the performance of a multiclass classification model requires multiple metrics:

1. **Accuracy:**
    - Measures the overall percentage of correctly classified images.
    - Formula: Accuracy=Correct PredictionsTotal PredictionsAccuracy = \frac{Correct\ Predictions}{Total\ Predictions}Accuracy=Total PredictionsCorrect Predictions
2. **Confusion Matrix:**
    - A table that provides a detailed breakdown of how each class is classified.
    - Helps identify misclassification patterns by displaying True Positives, False Positives, False Negatives, and True Negatives.
3. **Precision, Recall, and F1-Score:**
    - **Precision**: The proportion of correctly predicted instances for a given class out of all instances predicted as that class.

- **Recall**: The proportion of correctly predicted instances for a given class out of all actual instances of that class.
- **F1-Score**: The harmonic mean of Precision and Recall, providing a balanced measure for imbalanced datasets.

Steps to Implement Multiclass Classification Using CNN

1. Select a Dataset

Choose a dataset suitable for multiclass classification, such as:

- **MNIST** (handwritten digits from 0-9).
- **Fashion-MNIST** (10 classes of clothing items).
- **CIFAR-10 or CIFAR-100** (objects like airplanes, cats, dogs, cars, etc.).
- **Custom datasets** depending on the application.

2. Perform Data Preprocessing

- *Normalize pixel values to a range of [0,1][0,1][0,1] or [-1,1][-1,1][-1,1] for better training stability.*
- Resize images if needed to ensure uniform input dimensions.
- Split the dataset into **training**, **validation**, and **testing** sets.
- Apply **data augmentation** (rotation, flipping, zooming) to improve generalization.

3. Define the CNN Model Architecture

A standard CNN architecture for multiclass classification typically includes:

- **Convolutional layers** with different kernel sizes to extract features.
- **Batch normalization** to stabilize activations and speed up training.
- **Pooling layers** (e.g., Max Pooling) to reduce dimensionality.
- **Dropout layers** to prevent overfitting.
- **Fully connected layers** to make final predictions.
- **Softmax activation** in the output layer for class probabilities.

4. Train the Model

- Use an appropriate **loss function**, such as **Categorical Cross-Entropy**, which is ideal for multiclass classification.
- Select an **optimizer** like **Adam, RMSprop, or SGD** to minimize the loss function.
- Train the model for multiple epochs and monitor the loss and accuracy.
- Use **early stopping** or **learning rate scheduling** to prevent overfitting.

5. Evaluate the Model

- Test the trained model on a separate test dataset.
- Compute **accuracy**, **confusion matrix**, and **precision-recall-F1 score** to assess performance.
- Visualize the confusion matrix using **matplotlib** or **seaborn** to analyze classification mistakes.

6. Make Predictions on New Data

- Input unseen images to the trained CNN and obtain class predictions.
- Convert predicted probabilities into class labels using `argmax()`.
- Analyze real-world performance and optimize the model if necessary.

Conclusion:
I have successfully built a CNN model for multiclass classification using a suitable dataset. I also evaluated its performance with metrics like accuracy and a confusion matrix, demonstrating the model's effectiveness in classifying images into multiple categories.

Lab Assignment No.	3
Title	Image Classification using Convolutional Neural Networks (CNN)
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Deep Learning
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 03
Title: Image Classification using Convolutional Neural Networks (CNN)
Problem Statement:
Design and implement a CNN for Image Classification a) Select a suitable image classification dataset (medical imaging, agricultural, etc.). b) Optimized with different hyper-parameters including learning rate, filter size, no. of layers, optimizers, dropouts, etc.
Prerequisite:
Basic understanding of Deep Learning, Neural Networks, and Python programming.
Software Requirements:

- Jupyter Notebook or Google Colab
- Python Libraries: TensorFlow/Keras, pandas, numpy, matplotlib, scikit-learn

Hardware Requirements:

- GPU-enabled system (optional for faster training)
- 4 GB RAM (minimum)
- 500 GB HDD

Learning Objectives:

1. Understand the architecture and working of CNNs for image classification.
2. Learn to preprocess image datasets for model training.
3. Optimize CNN performance using hyperparameter tuning.

Outcomes:

1. Design and train a CNN for image classification tasks.
2. Understand and apply techniques for improving model accuracy and efficiency.
3. Evaluate CNN performance using appropriate metrics.

Theory: Convolutional Neural Networks (CNNs)

Introduction to CNNs

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing **image data**. Unlike traditional fully connected neural networks, CNNs leverage spatial hierarchies in images, allowing them to efficiently learn and extract meaningful patterns such as edges, textures, and object structures.

CNNs have revolutionized computer vision by achieving state-of-the-art performance in various applications, including **image classification, object detection, and medical imaging**. Their architecture is inspired by the **visual cortex of the human brain**, which processes visual information in layers.

Key Features of CNNs

CNNs consist of multiple specialized layers that process images through a series of transformations to extract relevant features and perform classification. The core layers include:

1. Convolutional Layer

- The convolutional layer is the **foundation** of CNNs.
- It applies **kernels/filters** (small matrices) that slide over the image to detect patterns such as edges, curves, and textures.
- Each filter learns to detect different features at varying levels of abstraction.

Example:

- Early layers may detect simple features like edges.
- Deeper layers detect complex features like faces, objects, or text.

2. Pooling Layer

- The pooling layer is responsible for **reducing the spatial dimensions** of feature maps while retaining the most important information.
- This helps in reducing computational complexity and prevents overfitting.
- The most common type is **Max Pooling**, which selects the highest value in a region, ensuring that key features are preserved.

3. Fully Connected Layer

- Once feature extraction is complete, the CNN flattens the extracted features and passes them to a fully connected (dense) layer.
- These layers map the extracted features to output predictions by learning complex relationships.
- The final layer often uses **Softmax activation** for classification tasks.

4. Activation Functions

- Activation functions introduce **non-linearity**, enabling CNNs to learn complex patterns.
- **ReLU (Rectified Linear Unit)** is the most widely used activation function in CNNs because it helps mitigate the vanishing gradient problem and speeds up training.
- The final output layer typically uses **Softmax** for multi-class classification or **Sigmoid** for binary classification.

Applications of CNNs in Image Classification

CNNs have a wide range of applications across various fields:

1. **Medical Imaging:**
   - Tumor detection in MRI or CT scans.
   - Identifying pneumonia in X-ray images.
   - Retinal disease classification using fundus images.
2. **Agriculture and Remote Sensing:**
   - Crop disease detection from leaf images.
   - Soil quality analysis using satellite images.
   - Identifying plant species and growth stages.
3. **Object Recognition:**
   - Recognizing handwritten digits (e.g., MNIST dataset).
   - Identifying faces in security systems.
   - Classifying animals, vehicles, and other objects in real-world images (e.g., CIFAR-10).
4. **Autonomous Vehicles:**
   - Traffic sign recognition.
   - Pedestrian and obstacle detection.
5. **Retail and E-commerce:**
   - Automated product classification in online stores.
   - Visual search engines that recommend similar products based on images.

Evaluation Metrics for CNN Models

Evaluating a CNN's performance involves various metrics to ensure robust classification:

1. **Accuracy:**
   - Measures the percentage of correctly classified images.
   - Formula: Accuracy=Correct PredictionsTotal PredictionsAccuracy = \frac{Correct\ Predictions}{Total\ Predictions}Accuracy=Total PredictionsCorrect Predictions
2. **Precision, Recall, and F1-Score:**
   - **Precision**: Measures the proportion of correctly predicted instances of a specific class among all predicted instances of that class.
   - **Recall (Sensitivity)**: Measures the proportion of actual instances correctly identified by the model.
   - **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure for imbalanced datasets.
3. **Confusion Matrix:**
   - A **visual representation** of model performance, showing how many predictions were correct or misclassified.
   - Useful for identifying specific classes where the model struggles.

Steps to Implement Image Classification Using CNN

1. Select a Suitable Dataset

- Choose an appropriate dataset depending on the classification problem. Examples include:
  - **MNIST** (handwritten digit classification, 10 classes).
  - **CIFAR-10/CIFAR-100** (real-world object classification, 10/100 classes).
  - **Fashion-MNIST** (classification of clothing items, 10 classes).
  - **Medical imaging datasets** (e.g., Chest X-ray, Retinal OCT).
  - **Custom datasets** (collected from various sources).

2. Preprocess the Dataset

- Resize images to a consistent shape (e.g., 28×28 for MNIST, 32×32 for CIFAR-10).
- Normalize pixel values to a range of [0,1] or [-1,1] to enhance training stability.
- Split data into **training, validation, and testing** sets.
- Apply **data augmentation** (e.g., rotation, flipping, zooming, shifting) to improve generalization and reduce overfitting.

3. Design the CNN Architecture

A typical CNN architecture consists of:

- **Convolutional Layers** (to extract features).
- **Pooling Layers** (to reduce dimensions).
- **Fully Connected Layers** (to classify images).
- **Dropout Layers** (to prevent overfitting).
- **Softmax Activation** (to generate class probabilities).

4. Optimize the CNN

- Experiment with **hyperparameters** such as learning rates, filter sizes, the number of layers, and optimizers (Adam, RMSprop, SGD).
- Use **dropout layers** to reduce overfitting.
- Adjust **batch sizes** and **epochs** for better training stability.

5. Train the Model

- Use **categorical cross-entropy** as the loss function for multi-class classification.
- Select an **optimizer** like Adam or RMSprop.

- Train for multiple epochs while monitoring performance through **loss and accuracy metrics**.
- Implement **early stopping** or **learning rate scheduling** to optimize training.

6. Evaluate the Model

- Test the trained CNN on the **test dataset**.
- Compute **accuracy, precision, recall, F1-score**, and generate a **confusion matrix**.
- Identify misclassified images and analyze areas for improvement.

7. Make Predictions on New Data

- Input unseen images into the trained model.
- Convert predicted probabilities into class labels.
- Use visualization tools like **Grad-CAM** to understand how the model makes decisions.

Conclusion:
I have successfully designed and implemented a CNN for image classification. I also optimized its performance using hyperparameter tuning and evaluated it using metrics like accuracy and F1-score.

Lab Assignment No.	4
Title	Sentiment Analysis using Recurrent Neural Networks (RNN)
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Deep Learning
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 04
Title: Sentiment Analysis using Recurrent Neural Networks (RNN)
Problem Statement:
Design and implement a Recurrent Neural Network (RNN) to perform sentiment analysis on text data. Use a suitable dataset and analyze the sentiment (positive, negative, neutral) expressed in the text.
Prerequisite:
Basic understanding of Natural Language Processing (NLP), Deep Learning, RNNs, and Python programming.
Software Requirements:

- Jupyter Notebook or Google Colab
- Python Libraries: TensorFlow/Keras, pandas, numpy, matplotlib, scikit-learn, nltk/spacy

Hardware Requirements:

- GPU-enabled system (optional for faster training)
- 4 GB RAM (minimum)
- 500 GB HDD

Learning Objectives:

1. Understand the architecture and working of RNNs for sequential data analysis.
2. Learn to preprocess text data for sentiment analysis.
3. Optimize the performance of RNN models through hyperparameter tuning.

Outcomes:

1. Build and train an RNN for sentiment analysis tasks.
2. Preprocess and vectorize text data for use in RNN models.
3. Evaluate model performance using appropriate metrics.

Theory: Recurrent Neural Networks (RNNs) for Sentiment Analysis

Introduction to RNNs

Recurrent Neural Networks (RNNs) are a type of deep learning model designed for **sequential data** processing. Unlike traditional neural networks, which treat each input independently, RNNs maintain a **hidden state** that allows them to capture **temporal dependencies**. This makes them particularly useful for tasks where order and context matter, such as **natural language processing (NLP), time-series forecasting, and speech recognition**.

Sentiment analysis is a key NLP application where RNNs are widely used. It involves classifying text data (e.g., reviews, tweets, or feedback) into different sentiment categories, such as **positive, negative, or neutral**.

Key Features of RNNs for Sentiment Analysis

1. Hidden Layers for Sequential Data Processing

- RNNs process data **one step at a time** while maintaining information about previous steps through **hidden states**.
- This allows the model to understand the **context** of words within a sentence.

2. Activation Functions

- RNNs use non-linear activation functions like **Tanh** or **ReLU** to introduce non-linearity and capture complex relationships in the data.
- Tanh is commonly used in standard RNNs, while ReLU can help in reducing the **vanishing gradient problem**.

3. Variants: LSTMs and GRUs

- Standard RNNs struggle with **long-term dependencies** due to the **vanishing gradient problem**.
- Two advanced RNN architectures, **Long Short-Term Memory (LSTM)** and **Gated Recurrent Units (GRU)**, help overcome this issue:
    - **LSTM**: Uses memory cells, input gates, output gates, and forget gates to selectively retain or discard information.
    - **GRU**: A simplified version of LSTM that combines input and forget gates into a **single update gate**, making it computationally efficient.

4. Embedding Layer for Word Representation

- Words in text data need to be converted into numerical representations before being processed by an RNN.
- **Word embeddings** (such as **Word2Vec, GloVe, or Keras Embedding Layer**) transform words into dense vector representations that capture their semantic meaning.

Applications of RNNs in Sentiment Analysis

1. Customer Feedback Analysis

- Companies analyze customer reviews to understand product performance, customer satisfaction, and areas for improvement.

2. Product Review Classification

- RNNs help classify product reviews into categories such as **positive, negative, or neutral**, aiding businesses in sentiment-driven decision-making.

3. Social Media Sentiment Analysis

- Businesses, politicians, and organizations use sentiment analysis on platforms like **Twitter, Facebook, and Reddit** to track public opinion and trends.

4. Financial Market Predictions

- Sentiment analysis on financial news and social media posts can help predict stock market trends.

Evaluation Metrics for Sentiment Analysis

1. Accuracy

- Measures the percentage of correctly classified sentiments.

2. Precision, Recall, and F1-Score

- **Precision**: Measures how many predicted positive instances are actually positive.
- **Recall (Sensitivity)**: Measures how well the model captures all positive instances.
- **F1-Score**: A balanced metric combining **precision and recall**, useful for imbalanced datasets.

3. Confusion Matrix

- A visual representation of model performance that shows how many instances were correctly classified and misclassified.

Steps to Implement Sentiment Analysis Using RNN

1. Select a Suitable Dataset

Choose a dataset that contains text samples labeled with sentiments. Common datasets include:

- **IMDB Movie Reviews** (positive/negative sentiment classification).

- **Twitter Sentiment Analysis Dataset** (tweets labeled as positive, neutral, or negative).
- **Amazon/Yelp Product Reviews** (analyzing customer sentiment).
- **Custom datasets** collected from different sources.

---

2. Preprocess the Text Data

Since raw text data cannot be directly fed into an RNN, preprocessing is necessary:

- **Text Cleaning**:
  - Convert text to **lowercase**.
  - Remove **punctuation, special characters, and stop words**.
- **Tokenization**:
  - Split sentences into individual words or tokens.
- **Convert Words to Sequences**:
  - Use **word embeddings** like **Word2Vec, GloVe, or Keras Embedding Layer** to convert words into numerical vectors.
- **Padding Sequences**:
  - Since RNNs process sequences of fixed lengths, shorter sentences are **padded** with zeros.

---

3. Design the RNN Model Architecture

A typical RNN model for sentiment analysis consists of:

- **Embedding Layer**: Converts words into dense vectors.
- **RNN (or LSTM/GRU) Layer**: Processes sequential text data while retaining context.
- **Dense Layer**: Maps the output to a sentiment classification.
- **Softmax or Sigmoid Activation**:
  - **Softmax** is used for **multi-class classification** (positive, neutral, negative).
  - **Sigmoid** is used for **binary classification** (positive vs. negative).

---

4. Optimize the RNN Model

- **Experiment with Hyperparameters**:
  - Number of **layers, units per layer, and dropout rates**.
  - Learning rate adjustments using optimizers like **Adam or RMSprop**.
  - Batch size and number of epochs for training stability.
- **Use Dropout Regularization**:

- Dropout layers help prevent **overfitting** by randomly deactivating some neurons during training.

---

5. Train the Model

- Train the model on the **training dataset**.
- Use **loss functions** like **binary cross-entropy** for binary sentiment analysis or **categorical cross-entropy** for multi-class classification.
- Implement **early stopping** to halt training when validation accuracy stops improving.

---

6. Evaluate the Model

- Test the model on the **test dataset**.
- Compute evaluation metrics like **accuracy, precision, recall, F1-score**, and generate a **confusion matrix**.

---

7. Use the Model for Sentiment Prediction

- Input **new text samples** into the trained model.
- Convert text to sequences and pass them through the RNN to predict sentiment.

1.

Output:

Conclusion:

I have successfully designed and implemented an RNN for sentiment analysis. The model was optimized using hyperparameter tuning and evaluated using metrics such as accuracy and F1-score. It demonstrated an ability to classify text sentiment effectively, showcasing the power of RNNs for NLP tasks.

Lab Assignment No.	
Title	
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Deep Learning
Assessment Marks	
Assessor's Sign	

Mini Project

Mini Project Instructions

Project Title Options:

1. Gender Recognition Using Voice – Implement a machine learning model to classify gender based on voice features.
2. Crop Disease Detection – Develop an AI-based system to identify plant diseases using image processing techniques.
3. Music Genre Classification System – Build a system that categorizes music into different genres based on audio features.

Problem Statement:

Develop an application that adheres to the following requirements:

1. Select a problem statement from the above options.
2. Follow the Software Development Life Cycle (SDLC) and apply Software Engineering principles throughout the implementation.
3. Develop the application using:
    - Front End: Python (Flask/Django)/Java/PHP/.NET or any suitable language.
    - Backend: MongoDB/MySQL/Oracle/Any SQL or NoSQL database.
4. Implement machine learning/deep learning models for data classification and prediction.
5. Conduct testing and validation using manual/automated testing approaches.

Project Report Requirements:

1. Title of the Project
2. Abstract
3. Introduction
4. Scope
5. Software/Hardware Requirements Specification
6. Source Code
7. Data Reports
8. Testing Document
9. Future Enhancements
10. Conclusion
11. References/Bibliography

Lab Assignment No.	1
Title	Data Visualization from Extraction, Transformation, and Loading (ETL) Process
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Business Intelligence
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 01

Title: Data Visualization from Extraction, Transformation, and Loading (ETL) Process

Problem Statement:

Visualize the results of the Extraction, Transformation, and Loading (ETL) process to provide insights into the data after it has been extracted, cleaned, transformed, and loaded into a targeted system.

Prerequisite:

- Basic understanding of Python programming.
- Familiarity with ETL processes and data visualization concepts.
- Knowledge of visualization libraries in Python (e.g., Matplotlib, Seaborn).

Software Requirements:

- **Jupyter Notebook** for implementation.
- Python libraries: pandas, matplotlib, seaborn, plotly, sqlalchemy, pyodbc.

Hardware Requirements:

- Processor: PIV or above.
- RAM: At least 2 GB.
- Storage: Minimum 500 GB HDD.

Learning Objectives:

1. Understand the ETL process and its stages—Extraction, Transformation, and Loading.
2. Learn how to visualize data at different stages of the ETL process.
3. Use Python visualization libraries to create meaningful charts and graphs to represent data insights.

Theory:

ETL Process and Data Visualization: A Comprehensive Guide

The Extract, Transform, Load (ETL) process is a key component of data engineering, ensuring that raw data is properly collected, cleaned, and stored for analysis. Once the data is processed, data visualization helps in extracting meaningful insights by identifying trends, patterns, and anomalies. This guide expands on the ETL process, visualization techniques, and evaluation metrics.

ETL Process: A Step-by-Step Guide

1. Extraction

Data extraction involves gathering raw data from various sources, such as:

- Databases: Structured data stored in SQL or NoSQL databases.
- APIs: Web services that provide structured or semi-structured data.
- Flat Files: Data stored in CSV, JSON, XML, or Excel files.

- Web Scraping: Collecting publicly available data from websites.

Successful data extraction ensures the completeness and consistency of raw data before processing.

2. Transformation

Once the data is extracted, it undergoes processing to improve quality and usability. Transformation includes:

- Data Cleaning: Handling missing values, duplicates, and inconsistencies.
- Data Standardization: Converting different formats into a uniform structure.
- Data Aggregation: Summarizing data to make it more useful for analysis.
- Feature Engineering: Creating new meaningful variables from existing data.

Effective transformation ensures that the dataset is accurate, structured, and ready for meaningful insights.

3. Loading

The transformed data is stored in a target system for future analysis. Common storage destinations include:

- Databases: Relational databases (e.g., MySQL, PostgreSQL) or NoSQL systems (e.g., MongoDB).
- Data Warehouses: Scalable storage solutions like Amazon Redshift or Google BigQuery.
- Cloud Storage: Platforms like AWS S3 or Google Cloud Storage for large datasets.
- Flat Files: CSV, JSON, or Parquet files for lightweight storage.

Proper data loading ensures that the processed data is easily retrievable and efficiently stored for visualization.

Data Visualization: Techniques and Tools

Data visualization makes it easier to interpret processed data, helping businesses and analysts make informed decisions.

1. Types of Visualizations

Visualization Type	Purpose
Bar Charts	Compare categorical data (e.g., sales by region).
Line Charts	Display trends over time (e.g., stock price movements).
Scatter Plots	Show relationships between two numerical variables.
Heatmaps	Visualize data density or correlations.
Pie Charts	Represent proportions of a whole (use sparingly).
Box Plots	Identify outliers and data distribution.

Each visualization type serves a different analytical purpose, making it essential to choose the right chart based on the dataset.

2. Popular Libraries for Visualization

- Matplotlib: A fundamental library for creating simple yet powerful static visualizations.
- Seaborn: Built on Matplotlib, it provides aesthetically appealing statistical graphs.
- Plotly: Enables interactive and web-based visualizations for deeper insights.

Selecting the appropriate visualization tool helps in making data more engaging and interpretable.

Evaluation Metrics for Success

1. Data Integration

- Ensuring extracted data is complete, consistent, and relevant.
- Properly handling missing, duplicate, or inconsistent records.

2. Effective Visualizations

- Clear and accurate representation of key data insights.
- Choosing suitable visualization types based on the dataset.

3. Clarity & Readability

- Well-labeled charts with appropriate titles, legends, and annotations.
- Easy-to-interpret graphs that facilitate decision-making.

4. Performance Optimization

- Efficient processing and storage of large datasets.
- Using optimized storage formats to enhance retrieval speed.

Implementation Workflow

1. Set Up the Environment
    - Install necessary tools and libraries.
    - Prepare the dataset for analysis after the ETL process.
2. Extract Data from Source
    - Gather data from databases, APIs, or files.
    - Validate extracted data for completeness and consistency.
3. Transform Data for Analysis
    - Cleanse and preprocess data to ensure accuracy.
    - Convert and standardize formats as needed.
4. Load Data into Target System
    - Store transformed data in a database, warehouse, or file.
    - Ensure data is accessible for visualization and reporting.
5. Create Visualizations for Insights
    - Generate charts and graphs to identify trends and patterns.
    - Use interactive visualizations for a more detailed analysis.
6. Validate and Present Results
    - Ensure clarity and correctness of visual representations.
    - Use dashboards or reports to communicate insights effectively.

Output:

Conclusion:

This assignment offers hands-on experience in both the ETL process and the creation of data visualizations. By integrating the ETL process with effective visualizations, you will be able to generate actionable insights that can drive decision-making. Understanding how to visualize data after extraction, transformation, and loading is crucial in uncovering valuable

patterns and trends. This process is essential in real-world data science and analytics workflows, where visual storytelling of data plays a vital role in communicating findings to stakeholders.

Lab Assignment No.	2
Title	Data Analysis and Visualization using Advanced Excel
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Business Intelligence
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 02

Title: Data Analysis and Visualization using Advanced Excel

Problem Statement:

Perform data analysis and create visualizations using advanced Excel features to extract insights and communicate findings effectively.

Prerequisite:

- Basic knowledge of Excel.
- Familiarity with data analysis techniques.
- Understanding of advanced Excel features like pivot tables, advanced charts, and functions.

Software Requirements:

- Microsoft Excel (2016 or newer).

Hardware Requirements:

- Processor: PIV or above.
- RAM: At least 2 GB.
- Storage: Minimum 500 GB HDD.

Learning Objectives:

1. Learn to perform data analysis using advanced Excel features.
2. Understand how to create meaningful visualizations in Excel.
3. Gain the ability to generate insights and communicate them effectively using Excel.

Theory:

Comprehensive Guide to Data Analysis in Excel

Microsoft Excel is one of the most widely used tools for **data analysis**, offering an extensive range of features to process, analyze, and visualize data efficiently. From simple spreadsheet functions to **advanced data modeling and visualization**, Excel enables users to **extract insights, identify trends, and make informed decisions**. This guide expands on **Excel's data analysis techniques, visualization methods, and advanced tools**, along with evaluation metrics and a step-by-step approach.

---

1. Data Analysis in Excel

Key Features for Data Analysis

Excel provides several built-in tools to help users analyze data quickly and effectively:

1. Pivot Tables

Pivot Tables are powerful tools that allow users to **summarize large datasets** by grouping, filtering, and aggregating information. They help in:

- **Summarizing numerical data** (e.g., sales figures by region).
- **Calculating totals, averages, counts, and other aggregate functions.**
- **Creating interactive reports** by easily reorganizing data.

2. Advanced Functions

Excel provides a range of functions to **manipulate data, perform lookups, and execute complex calculations**:

- **VLOOKUP & HLOOKUP:** Search for values in rows and columns.
- **INDEX-MATCH:** A more flexible alternative to VLOOKUP for retrieving values.
- **IF Statements:** Apply conditional logic to datasets.
- **Array Functions:** Perform calculations on multiple values simultaneously.

3. Data Validation

This feature **ensures data integrity** by setting constraints for data entry:

- Creating **drop-down lists** to standardize input.
- Enforcing **date range restrictions** to prevent incorrect entries.
- Setting numerical constraints to **avoid outliers** in data.

4. Conditional Formatting

Conditional formatting allows users to highlight data based on specific criteria, such as:

- **Color coding values** (e.g., marking high and low sales figures).
- **Using icon sets** to indicate trends or performance.
- **Highlighting duplicates** to detect errors in data entry.

---

2. Data Visualization in Excel

Effective visualization techniques help translate raw data into meaningful insights. Excel offers a variety of **charts and visual aids** to improve data interpretation.

1. Charts for Data Representation

Excel provides multiple chart types for **visualizing trends and patterns**:

Chart Type	Purpose
Bar Charts	Compare categorical data (e.g., sales by region).
Line Graphs	Display trends over time (e.g., stock price movements).
Scatter Plots	Show relationships between two numerical variables.
PieCharts	Represent proportions within a dataset (use sparingly).
Histograms	Show frequency distributions in numerical data.

Choosing the right chart type ensures clarity and accuracy in **data interpretation**.

2. Conditional Formatting with Data Bars and Color Scales

- **Data Bars:** Represent values using horizontal bars within a cell.
- **Color Scales:** Apply color gradients to show variations in numerical values.
- **Icon Sets:** Use symbols (e.g., arrows, checkmarks) to indicate changes.

These techniques **enhance readability** and help users **spot trends instantly**.

3. Sparklines

Sparklines are **miniature charts** embedded within a single cell to show trends in small datasets. They are ideal for:

- **Tracking performance over time** (e.g., monthly sales).
- **Comparing trends** across multiple data points.
- **Identifying seasonal variations** in business data.

4. Dashboards

Dashboards combine multiple charts, tables, and metrics into a **single interactive view**, allowing users to:

- **Summarize key performance indicators (KPIs).**
- **Use slicers and filters** to explore data interactively.
- **Monitor trends across different categories** in a unified display.

---

3. Advanced Excel Tools for Data Analysis

Excel includes **add-ins and specialized tools** that extend its analytical capabilities, making it suitable for **large-scale data processing**.

1. Power Query (ETL in Excel)

Power Query is an **extract, transform, and load (ETL) tool** within Excel that allows users to:

- **Import data from multiple sources** (databases, web, APIs, files).
- **Filter, clean, and reshape data** before analysis.
- **Automate repetitive data transformation** tasks.

2. Power Pivot (Advanced Data Modeling)

Power Pivot enhances Excel's ability to **analyze large datasets** and create **multi-table relationships**. It enables:

- **Creation of data models** with complex relationships.
- **Use of DAX (Data Analysis Expressions) functions** for advanced calculations.
- **Integration of data from multiple tables** without traditional VLOOKUPs.

---

4. Evaluation Metrics for Success

1. Data Processing Efficiency

- Successful **removal of duplicates, missing values, and inconsistencies**.
- Proper **application of data validation techniques** to prevent errors.

2. Advanced Function Utilization

- Effective **use of lookup functions** to connect multiple datasets.
- Application of **conditional logic and calculations** for deeper insights.

3. Visualization Effectiveness

- Clear and accurate representation of **trends and relationships**.
- Selection of appropriate **chart types** based on dataset characteristics.

4. Dashboard & Reporting Clarity

- **Interactive dashboards** that allow for flexible exploration.
- Well-structured layouts with **easy-to-read summaries and insights**.

---

5. Step-by-Step Implementation Workflow

Step 1: Import the Dataset

- Open Excel and load data via **file import, database connection, or web query**.
- Ensure **data structure is well-defined** for analysis.

Step 2: Data Cleaning & Preprocessing

- **Remove duplicates and correct inconsistencies.**
- **Handle missing values** using imputation or filtering.
- **Convert data types** (e.g., text to numbers, dates).

Step 3: Apply Advanced Functions

- Use **VLOOKUP/INDEX-MATCH** for data retrieval.
- Apply **IF statements and logical functions** to classify data.
- Leverage **array formulas** for large-scale calculations.

Step 4: Create Pivot Tables

- Summarize **key metrics** such as total sales, average revenue, etc.
- Group data by **categories, time periods, or locations**.
- Apply **filters and slicers** for interactive exploration.

Step 5: Design Charts & Visualizations

- Choose **bar charts, line graphs, or scatter plots** based on data patterns.
- Apply **conditional formatting** to highlight outliers and trends.
- Use **sparklines** for quick insights into data changes.

Step 6: Build an Interactive Dashboard

- Combine **multiple charts, tables, and KPIs** into a structured dashboard.
- Add **slicers and filters** to allow user interaction.
- Ensure **clarity, readability, and accuracy** of the final report

1.

Output:

Conclusion:

This assignment provides hands-on experience in using advanced Excel features to perform data analysis and create compelling visualizations. By mastering pivot tables, advanced functions, and visualization techniques, you can derive valuable insights from complex datasets. Excel is a versatile tool for data analysis, and using these advanced features enhances its capabilities for processing, analyzing, and presenting data in an intuitive manner. This skill is crucial for making data-driven decisions and effectively communicating findings in any business or analytical context.

Lab Assignment No.	3
Computer Laboratory-IV	*B.E.(Sem-II)* **[2024-25]**
Title	Data Classification Using a Classification Algorithm
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Business Intelligence
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 03

Title: Data Classification Using a Classification Algorithm

Problem Statement:

Implement and evaluate a classification algorithm to classify data into predefined categories or classes. The focus will be on building, training, and testing a classification model, then visualizing the results to gain insights into the model's performance.

Prerequisite:

- Basic understanding of Python programming.
- Familiarity with machine learning concepts, especially classification.
- Knowledge of Python libraries for data processing and machine learning (e.g., pandas, scikit-learn, matplotlib, seaborn).

Software Requirements:

- Jupyter Notebook for implementation.
- Python libraries: pandas, scikit-learn, matplotlib, seaborn, numpy, plotly.

Hardware Requirements:

- Processor: PIV or above.
- RAM: At least 2 GB.
- Storage: Minimum 500 GB HDD**.**

Learning Objectives:

1. Understand the concept of classification in machine learning.
2. Learn how to preprocess data for a classification task.
3. Implement a classification algorithm using scikit-learn.
4. Visualize and interpret the results of a classification model.

Theory:

Classification in Machine Learning

1. Introduction to Classification

Classification is a supervised machine learning technique used to categorize data into predefined classes or groups. A classification model is trained on labeled data, meaning the input data has known outcomes or categories. Once trained, the model can classify new, unseen data into the appropriate category.

1.1 Importance of Classification

Classification is widely used in various real-world applications, including:

- Spam Detection: Identifying whether an email is spam or not.
- Sentiment Analysis: Determining if a customer review is positive, neutral, or negative.
- Fraud Detection: Detecting fraudulent transactions in banking and finance.
- Medical Diagnosis: Classifying diseases based on patient data.

- Image Recognition: Identifying objects in images (e.g., facial recognition, handwriting recognition).

---

2. Steps in a Classification Task

2.1 Data Collection

The first step in building a classification model is collecting and preparing the dataset.

- Structured Data: Includes tabular data with numerical and categorical variables (e.g., customer transactions, medical records).
- Unstructured Data: Includes images, text, and audio that require preprocessing before classification (e.g., sentiment analysis, object recognition).
- Sources: Data can be gathered from databases, CSV files, APIs, web scraping, or existing datasets from sources like Kaggle and UCI Machine Learning Repository.

2.2 Data Preprocessing

Raw data often contains inconsistencies, missing values, and irrelevant features. Data preprocessing ensures the data is cleaned and structured for optimal model performance. Key preprocessing steps include:

Handling Missing Values

- Remove missing values if they are minimal and do not affect data distribution.
- Impute missing values using mean, median, mode, or predictive methods.

Encoding Categorical Variables

- One-Hot Encoding: Converts categorical values into binary columns (suitable for nominal data).
- Label Encoding: Assigns numerical labels to categorical variables (used for ordinal data).

Feature Scaling & Normalization

- Standardization (Z-score normalization): Centers data around zero using the formula: $X' = \frac{X - \mu}{\sigma}$
- Min-Max Scaling: Rescales values between 0 and 1.

Feature Engineering & Selection

- Remove irrelevant or highly correlated features.

- Create new meaningful features from existing ones.

2.3 Model Selection & Training

After preprocessing the dataset, the next step is selecting a suitable classification algorithm and training the model.

Common Classification Algorithms

Algorithm	Description	Use Case
Logistic Regression	A simple statistical model for binary classification.	Spam detection, loan approval.
Decision Tree	A tree-based model that splits data into branches based on feature values.	Credit scoring, customer segmentation.
Random Forest	An ensemble of multiple decision trees to improve accuracy and reduce overfitting.	Fraud detection, medical diagnosis.
Support Vector Machine (SVM)	Finds the optimal hyperplane to separate different classes.	Image classification, text categorization.
K-Nearest Neighbors (KNN)	Classifies data based on the majority class of its nearest neighbors.	Pattern recognition, recommendation systems.
Naïve Bayes	Uses probability distributions to classify data.	Sentiment analysis, spam filtering.

The choice of algorithm depends on the size of the dataset, feature complexity, and interpretability.

2.4 Model Evaluation

Once the model is trained, it must be evaluated using performance metrics to ensure it generalizes well to unseen data.

Key Evaluation Metrics

1. Accuracy
    - Measures the percentage of correctly classified instances:
2. Accuracy=TP+TNTP+TN+FP+FNAccuracy = \frac{TP + TN}{TP + TN + FP + FN}Accuracy=TP+TN+FP+FNTP+TN

    - Best for balanced datasets, but not ideal for imbalanced classes.
3. Precision, Recall, and F1-Score
    - Precision: Measures how many predicted positive cases are actually positive. Precision=TPTP+FPPrecision = \frac{TP}{TP + FP}Precision=TP+FPTP
    - Recall (Sensitivity): Measures how well the model identifies actual positive cases. Recall=TPTP+FNRecall = \frac{TP}{TP + FN}Recall=TP+FNTP
    - F1-Score: Harmonic mean of precision and recall, balancing both metrics. F1=2×Precision×RecallPrecision+RecallF1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}F1=2×Precision+RecallPrecision×Recall
4. Confusion Matrix
    - A table representation of classification results, showing true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).
    - Helps in diagnosing model errors and misclassifications.
5. ROC Curve & AUC Score
    - The Receiver Operating Characteristic (ROC) curve plots the true positive rate (TPR) vs. the false positive rate (FPR).
    - The Area Under the Curve (AUC) measures the ability to distinguish between classes (higher is better).
6. Feature Importance Analysis
    - Helps interpret which variables influence the model's decision-making.
    - Techniques include SHAP values, permutation importance, and tree-based feature importance.

---

3. Visualizing Model Performance

3.1 Confusion Matrix

- A heatmap representation to understand false positives, false negatives, and overall model accuracy.
- Helps in identifying biases or class imbalances in predictions.

3.2 ROC Curve

- Used to analyze the trade-off between true positive and false positive rates.
- A higher AUC (Area Under Curve) score indicates a better model.

3.3 Feature Importance Plots

- Identifies the most influential variables in decision-making.
- Used to eliminate less relevant features, improving model interpretability.

---

4. Model Optimization & Hyperparameter Tuning

Once the model is evaluated, further improvements can be made through optimization.

4.1 Hyperparameter Tuning Techniques

1. Grid Search:
    - Tests all possible hyperparameter combinations.
2. Random Search:
    - Selects a subset of hyperparameter combinations for faster tuning.
3. Bayesian Optimization:
    - Uses probability distributions to find optimal hyperparameters efficiently.

4.2 Handling Imbalanced Datasets

- Oversampling (SMOTE): Generates synthetic samples for the minority class.
- Undersampling: Reduces the number of majority class samples.
- Class Weight Adjustment: Assigns higher weight to the minority class.

---

5. Presenting Results & Insights

After training and optimizing the model, the results should be presented in a structured and insightful way:

5.1 Summarizing Key Findings

- Highlight performance metrics (accuracy, precision, recall, F1-score, AUC-ROC).
- Show model limitations (e.g., high false negatives in fraud detection).
- Interpret feature importance to explain key influencing factors.

5.2 Creating Visual Reports

- Use charts, heatmaps, and plots to communicate findings effectively.
- Provide actionable insights for decision-making or business applications.

Output:

Conclusion:

This assignment provides practical experience with implementing a classification algorithm, from preprocessing data to evaluating and visualizing model performance. By applying classification techniques, you will gain valuable insights into how machine learning models can classify data accurately and effectively. The skills learned in this assignment are essential

for solving real-world problems where classification plays a critical role in predictive analytics.

Lab Assignment No.	4
Title	Data Clustering Using a Clustering Algorithm
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Business Intelligence
Assessment Marks	
Assessor's Sign	

ASSIGNMENT No: 04

Title: Data Clustering Using a Clustering Algorithm

Problem Statement:

Implement and evaluate a clustering algorithm to group similar data points into clusters based on their features. The goal is to identify inherent patterns or groupings within the data without relying on labeled outputs**.**

Prerequisite:

- Basic understanding of Python programming.
- Familiarity with machine learning concepts, especially unsupervised learning.
- Knowledge of Python libraries for data processing and machine learning (e.g., pandas, scikit-learn, matplotlib, seaborn).

Software Requirements:

- Jupyter Notebook for implementation.
- Python libraries: pandas, scikit-learn, matplotlib, seaborn, numpy, plotly.

Hardware Requirements:

- Processor: PIV or above.
- RAM: At least 2 GB.
- Storage: Minimum 500 GB HDD.

Learning Objectives:

1. Understand the concept of clustering in unsupervised learning.
2. Learn how to preprocess data for a clustering task.
3. Implement a clustering algorithm using Python libraries.
4. Visualize and interpret the results of clustering models.

Theory:

1. Introduction to Clustering

Clustering is an unsupervised learning technique used to group data points into distinct clusters based on their similarities. Unlike classification, where data points are assigned to predefined categories, clustering works with unlabeled data and identifies natural groupings within the dataset.

1.1 Importance of Clustering

Clustering is widely used in various real-world applications, including:

- Customer Segmentation: Grouping customers based on purchasing behavior, demographics, or preferences.
- Anomaly Detection: Identifying unusual patterns in network security, fraud detection, or medical diagnosis.
- Market Research: Understanding market trends by grouping similar products or consumers.
- Image Segmentation: Dividing an image into different regions for object recognition.

- Genomic Data Analysis: Clustering genetic sequences to identify species similarities or mutations.

---

2. Steps in a Clustering Task

2.1 Data Collection

The first step in clustering is collecting and preparing the dataset. Data can be obtained from various sources, including:

- Structured Data: Customer databases, transaction records, survey results.
- Unstructured Data: Images, social media posts, sensor readings.
- Data Repositories: Public datasets from Kaggle, UCI Machine Learning Repository, or government databases.

2.2 Data Preprocessing

Before applying clustering algorithms, data must be cleaned and transformed.

Handling Missing Values

- Imputation: Fill missing values using mean, median, mode, or predictive models.
- Removal: Drop rows or columns with excessive missing values if necessary.

Feature Scaling & Normalization

- Standardization (Z-score normalization): $X'=X-\mu\sigma X' = \frac{X - \mu}{\sigma}X'=\sigma X-\mu$
- Min-Max Scaling: Rescales values between 0 and 1 to ensure features contribute equally to clustering.

Outlier Detection & Removal

- Z-score method: Removes values beyond a certain threshold (e.g., 3 standard deviations from the mean).
- Interquartile Range (IQR) method: Removes extreme values using quartile-based filtering.

Dimensionality Reduction (if necessary)

- Principal Component Analysis (PCA): Reduces high-dimensional data while retaining important information.

- t-SNE (t-distributed Stochastic Neighbor Embedding): Useful for visualizing high-dimensional data in 2D/3D space.

2.3 Model Selection & Clustering Algorithms

Several clustering algorithms exist, each with its own strengths and weaknesses.

Common Clustering Algorithms

Algorithm	Description	Use Case
K-Means	Partitions data into K clusters, assigning points to the nearest centroid.	Customer segmentation, document clustering.
DBSCAN (Density-Based Spatial Clustering of Applications with Noise)	Groups data based on density and identifies anomalies as noise.	Anomaly detection, geospatial clustering.
Agglomerative Hierarchical Clustering	Builds a hierarchy of clusters by iteratively merging or splitting them.	Taxonomy classification, social network analysis.
Mean-Shift Clustering	Identifies high-density regions without requiring a predefined number of clusters.	Image segmentation, object tracking.
Gaussian Mixture Models (GMM)	Uses probability distributions to model complex clusters with soft assignments.	Anomaly detection, financial risk analysis.

The choice of algorithm depends on factors like dataset size, shape of clusters, and computational efficiency.

2.4 Model Evaluation

Clustering algorithms must be evaluated to determine how well they separate data points.

Key Evaluation Metrics

1. Silhouette Score
    - Measures how similar a point is to its own cluster compared to other clusters.
    - Ranges from -1 (poor clustering) to 1 (well-separated clusters).

2. $S(i)=b(i)-a(i)max⁡\{a(i),b(i)\}S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}S(i)=max(a(i),b(i))b(i)−a(i)$
    - Where a(i) is the mean intra-cluster distance, and b(i) is the mean nearest-cluster distance.
3. Elbow Method
    - Used to determine the optimal number of clusters K in K-Means.
    - Plots the within-cluster sum of squares (WCSS) against the number of clusters.
    - The "elbow" point in the graph indicates the best K value.
4. Davies-Bouldin Index
    - Measures the ratio of intra-cluster distances to inter-cluster distances.
    - Lower values indicate better clustering.
5. Visual Inspection
    - Scatter Plots: Helps visualize clusters in two-dimensional space.
    - Pair Plots: Shows relationships between multiple features across clusters.
    - Heatmaps: Displays correlations between clusters and features.

---

3. Visualizing Clustering Results

3.1 Scatter Plots

- Used for 2D or 3D data to plot clusters with different colors.
- Helps in identifying cluster overlap or separation.

3.2 Pair Plots

- Plots feature distributions across clusters for better understanding.
- Useful for datasets with multiple numerical features.

3.3 Heatmaps

- Shows feature correlations within clusters.
- Helps identify which features contribute the most to each cluster.

3.4 Dendrograms (Hierarchical Clustering Visualization)

- Used in Agglomerative Hierarchical Clustering to display the merging process of clusters.
- The cut-off point in the tree determines the optimal number of clusters.

---

4. Model Optimization & Hyperparameter Tuning

To improve clustering performance, hyperparameter tuning and optimizations are necessary.

4.1 Choosing the Right Number of Clusters

- Use the Elbow Method for K-Means to determine optimal K.
- Silhouette Analysis to assess cohesion and separation.
- Gap Statistic Method to compare WCSS with a random clustering baseline.

4.2 Handling Overlapping Clusters

- Use Gaussian Mixture Models (GMM) for probabilistic cluster assignments.
- Try t-SNE or UMAP (Uniform Manifold Approximation and Projection) for better visualization.

4.3 Dealing with Noise & Outliers

- Use DBSCAN to separate noise from real clusters.
- Apply feature engineering to remove unnecessary attributes.

---

5. Presenting Clustering Insights

Once clustering is complete, insights should be communicated effectively.

5.1 Key Findings

- Describe distinct clusters and their characteristics.
- Highlight patterns or trends found in the clusters.

5.2 Business Insights & Recommendations

- Customer Segmentation: Tailor marketing strategies for different customer groups.
- Fraud Detection: Flag anomalies that require further investigation.
- Product Recommendations: Identify products frequently purchased together.

5.3 Visual Reports

- Use dashboards and graphs to present findings clearly.
- Provide actionable insights for decision-makers.

Output:

Conclusion:

This assignment offers practical experience with unsupervised learning techniques by implementing a clustering algorithm. By identifying and visualizing inherent patterns within the data, you will gain insights into data segmentation and distribution. Clustering is a powerful tool for exploring unlabeled datasets and uncovering meaningful groupings that support decision-making and further analysis.

Lab Assignment No.	Mini Project
Title	
Roll No.	
Class	BE
Date of Completion	
Subject	Computer Laboratory-IV : Business Intelligence
Assessment Marks	
Assessor's Sign	

Mini Project

Detailed Case Study on a Business Intelligence (BI) Tool

Objective:

Conduct a detailed case study on any one Business Intelligence (BI) tool (such as Pentaho, Power BI, Tableau, or Apache Superset) and prepare a BI Report outlining the following steps.

Problem Statement:

Develop a comprehensive BI report by following these requirements:

1. Select an open-source or proprietary BI tool and explore its functionalities.
2. Define a problem statement and determine which data mining task is required (e.g., classification, clustering, regression, association rule mining).
3. Identify and use a standard data mining dataset suitable for the chosen problem.
4. Use the selected BI tool to perform data extraction, transformation, visualization, and analysis.
5. Evaluate insights generated from the BI tool and document findings in the final report.
6. Submit a detailed case study report covering all phases of BI tool implementation.

BI Report Requirements:

1. Title of the Case Study
2. Abstract
3. Introduction
4. Problem Definition and Required Data Mining Task
5. Selected Data Mining Dataset
6. BI Tool Overview and Justification
7. Data Processing and Analysis Steps
8. Visualization and Insights Generated
9. Interpretation of Results
10. Future Enhancements and Use Cases
11. Conclusion
12. References/Bibliography