**PREDICTING PRODUCT DEMAND USING BATCH-PROCERSSED   
DATA IN AZURE DATABRICKS ML**  ***A Major Project Report submitted to***

***Jawaharlal Nehru Technological University***

***in partial fulfillment of the requirements for the award of Degree of***

### BACHELOR OF TECHNOLOGY

#### in

**COMPUTER SCIENCE AND ENGINEERING**

**(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)**

**Submitted by**

**AARTHI VELPULA 21831A6661**

**E.N SNEHA 21831A6622**

**ARUN KUMAR MADAGONI 21831A6644**

***Under the Guidance of***

#### Mr.N.RAJESH

**Assistant Professor**

**(CSE-AIML)**

****

**Department of COMPUTER SCIENCE AND ENGINEERING**

***(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)***

**Guru Nanak Institute of Technology**

**Ibrahimpatnam, Hyderabad, R.R. District – 501506**

# MAY, 2025

**** **GURU NANAK INSTITUTE OF TECHNOLOGY**

**(Autonomous)**

###### Ibrahimpatnam, Ranga Reddy District-501506

**CERTIFICATE**

This is to certify that the project entitled “ Predicting Product Demand Using Batch-Procerssed

Data in Azure Databricks ML” is being submitted by Ms.AARTHI VELPULA, bearing Roll No.

21831A6661,Ms.E.NSNEHA, bearing Roll No. 21831A6622,Mr. ARUM KUMAR MADAGONI bearing Roll No. 21831A6644, in partial fulfillment for the award of the Degree of Bachelor of the Technology in Computer Science and Engineering (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING) to the Jawaharlal Nehru Technological University is a record of Bonafide work carried out by them under my guidance and supervision

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

**Mr.N.RAJESH Ms. UZMA YASEEN BHAT**

##### INTERNAL GUIDE PROJECT COORDINATOR

**Dr. MAHESH LOKHANDE**

**HEAD OF DEPARTMENT EXTERNAL EXAMINER**

 **GURU NANAK INSTITUTE OF TECHNOLOGY**

**(Autonomous)**

###### Ibrahimpatnam, Ranga Reddy District-501506

###### Declaration of Student

We, BODANAPALLI HARSHITHA (21831A6614) , KATAKAM MOHAN (21831A6635) , M.SRIKAR (20831A6635) , hereby declare that the major project titled “A Proxy-Assisted Secure Deduplication Framework For Encrypted Cloud Storage” has been carried out by us as part of the requirements for the award of the Degree of Bachelor of Technology in the Department of Computer Science and Engineering (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING) at Guru

Nanak Institute of Technology, Hyderabad

We confirm the following:

1. The project was undertaken by us under the supervision of our guide, Mr.N.RAJESH, from the selection of the topic to the completion of the final report.
2. We have ensured that the results presented in the report are accurate and based on our original work.
3. To the best of our knowledge, the content of this report is free from plagiarism and adheres to ethical standards.
4. Each member of the team has contributed significantly and appropriately to the project work.
5. The project report has been prepared with diligence, ensuring clarity, accuracy, and adherence to academic standards.

We further declare that this report has not been submitted, in part or full, to any other institution or university for the award of any degree or diploma.

AARTHI VELPULA

21831A6661 Signature Student 1

E.N SNEHA

21831A6622 Signature Student 2

ARUN KUMAR MADAGONI

21831A6644 Signature Student 3

Date:

Place:Ibrahimpatnam, Hyderabad.

# GURU NANAK INSTITUTE OF TECHNOLOGY

**(Autonomous)**

**Ibrahimpatnam, Ranga Reddy District-501506**

## Declaration of Guide

I, Mr.N.RAJESH, hereby declare that I have guided the major project titled “Predicting Product Demand Using Batch Processed Data in Azure Databricks” undertaken by AARTHI VELPULA (21831A6661),E.N.SNEHA (21831A6622) ,ARUN KUMAR MADAGONI (21831A6644). This project was carried out towards the fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING) at Guru Nanak Institute of Technology, Hyderabad.

As the guide, I confirm the following:

1. I have overseen the entire project process, from the selection of the project title to the submission of the final report.
2. I have reviewed and certified the accuracy and relevance of the results presented in the report.
3. The work carried out is original, free from plagiarism, and adheres to ethical guidelines.
4. The contributions of each student have been appropriately recognized and assessed.
5. The project report has been prepared under my supervision, ensuring adherence to high standards of quality, clarity, and structure.

I further certify that this project report has not been previously submitted in part or full for the award of any degree or diploma by any institution or university.

Name of Guide:Mr.N.RAJESH

Date: Signature of the Guide Date:

Place:Ibrahimpatnam, Hyderabad.

Name of HOD: Dr. MAHESH LOKHANDE Signature of the HOD Department:CSE (AI&ML)

Date:

Place:Ibrahimpatnam, Hyderabad.

# GURU NANAK INSTITUTE OF TECHNOLOGY

**(Autonomous)**

**Ibrahimpatnam, Ranga Reddy District-501506**

**ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to our internal guide, **Mr. N. RAJESH** Assistant Professor, Computer Science and Engineering (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING), for his valuable guidance, encouragement, and continuous support throughout the duration of this project.

We are also thankful to **Dr. MAHESH LOKHANDE**, Head of the Department, Computer Science and Engineering (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING), for his expert supervision and helpful suggestions, which contributed significantly to the successful completion of this project.

We would also like to thank the faculty members of the **Computer Science and Engineering (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)** and

the Lab Technicians for their assistance and cooperation during the practical work of our project.

We are grateful to our friends and well-wishers for their encouragement, collaboration, and useful feedback throughout the project journey.

Lastly, we sincerely thank our parents for their constant support, patience, and motivation,

which helped us complete this project successfully.

AARTHI VELPULA 21831A6661

E.N. SNEHA 21831A6622

ARUN KUMAR MADAGONI 2183IA6644

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **CERTIFICATE**  **DECLARATION OF STUDENT**  **DECLARATION OF GUIDE**  **ACKNOWLEDGEMENT**  **ABSTRACT**  **LIST OF FIGURES**  **LIST OF NOTATIONS**  **ABBREVATIONS** | **i**  **ii**  **iv**  **vi**  **viii**  **ix**  **x**  **xiii** |
| 1. | **CHAPTER 1: INTRODUCTION**   * 1. GENERAL   2. EXISTING SYSTEM   1.2.1 EXISTING SYSTEM DISADVANTAGES | 1 - 3  1  3  3 |
| 2. | **CHAPTER 2: LITERATURE SURVEY**  2.1 LITERATURE SURVEY’S | 4 - 9  4 |
| 3. | **CHAPTER 3: DESIGN AND DEVELOPEMENT**  3.1 PROBLEM STATEMENT & NEED ANALYSIS  3.1.1 PROBLEM STATEMENT  3.1.2 OBJECTIVE  3.1.3 SCOPE OF THE PROJECT  3.1.4 HARDWARE REQUIREMENTS  3.1.5 SOFTWARE REQUIREMENTS  3.1.6 FUNCTIONAL REQUIREMENTS  3.1.7 NON-FUNCTIONAL REQUIREMENTS  3.2 SYSTEM ARCHITECTURE  3.2.1 SCALABILITY  3.3 DESIGN METHODOLOGY  3.3.1 MODULES NAMES  3.4 DESIGN PROCESS & UML DIAGRAMS  3.4.1 USE CASE DIAGRAM  3.4.2 CLASS DIAGRAM  3.4.3 OBJECT DIAGRAM  3.4.4COMPONENT DIAGRAM  3.4.5 DEPLOYEMENT DIAGRAM  3.4.6 SEQUENCE DIAGRAM  3.4.7 COLLABORATION DIAGRAM  3.4.8 STATE DIAGRAM  3.4.9 ACTIVITY DIAGRAM  3.4.10 DATA FLOW DIAGRAM  3.4.11 E.R DIAGRAM  3.5 TOOLS AND TECHNOLOGIES USED  3.5.1 FEATURES OF JAVA  3.5.2OBJECTIVES OF JAVA  3.5.3 JAVA SWING OVERVIEW  3.5.4 COLLECTION FRAMEWORK  3.5.4.1 THREE TYPES OF COMPONENTS  3.6 CODE STRUCTURE  3.7 TESTING AND VALIDATION  3.7.1 DEVELOPING METHODOLOGIES  3.7.2 TYPES OF TESTING  3.7.2.1 UNIT TESTING  3.7.2.2 FUNCTINOAL TESTING  3.7.2.3 SYSTEM TEST  3.7.2.4 PERFORMANCE TEST  3.7.2.5 INTEGRATION TESTING  3.7.2.6 ACCEPTANCE TESTING  3.2.7.7 BUILD THE TEST PLAN  3.8 SNAPSHOTS | 10 - 49  10 - 14  10  11  11  12  12  13  14  15 - 16  16  17  17  22 - 34  22  23  24  25  26  27  28  29  30  31  34  36 - 39  36  37  38  39  39  42  43 - 46  43  44  45  45  45  46  46  46  46  47 - 49 |
| 4. | **CHAPTER 4: RESULTS AND DISCUSSIONS**  4.1 FUTURE ENHANCEMENTS | 50  50 |
| 5. | **CHAPTER 5: CONCLUSION**  5.1 CONCLUSION | 51  51 |
|  | **REFERENCES** | 52 - 54 |

# ABSTRACT

Accurate demand forecasting is critical for inventory optimization, cost reduction, and customer

satisfaction in data-driven enterprises. Traditional statistical approaches often fall short in managing

high-volume data, handling complex seasonality, or incorporating external factors. This project

addresses these limitations by building a scalable and automated demand forecasting system using

Azure Databricks. Leveraging its Apache Spark-powered batch processing engine, Delta Lake for

reliable data storage, and machine learning frameworks like MLflow, the system enables efficient

data preprocessing, model training, and deployment. By analyzing historical sales data, the model

predicts future product demand with improved accuracy and reduced manual effort, making it suitable

for modern enterprise applications in retail and supply chain management.

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO** |
| 4.2 | SYSTEM ARCHITECTURE | 12 |
| 4.3.1 | USE CASE DIAGRAM | 13 |
| 4.3.2 | CLASS DIAGRAM | 14 |
| 4.3.3 | OBJECT DIAGRAM | 15 |
| 4.3.4 | COMPONENT DIAGRAM | 16 |
| 4.3.5 | DEPLOYMENT DIAGRAM | 17 |
| 4.3.6 | SEQUENCE DIAGRAM | 18 |
| 4.3.7 | COLLABORATION DIAGRAM | 19 |
| 4.3.8 | STATE DIAGRAM | 20 |
| 4.3.9 | ACTIVITY DIAGRAM | 21 |
| 4.4 | DATA FLOW DIAGRAM | 22 |
| 4.5 | E-R DIAGRAM | 23 |
| 5.1.1 | NUMPY, PANDAS, MATPLOTLIB, SCIKIT-LEARN | 25 |
| 7.1.1 | CALCULATE TOTAL SALES FOR EACH PRODUCT | 31 |
| 7.1.2 | COUNT THE NUMBER OF ORDERS PER CUSTOMER | 32 |
| 7.1.3 | IDENTIFY THE MOST POPULAR PRODUCTS BY QUANTITY SOLD | 33 |

## LIST OF SYMBOLS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **NOTATION NAME** | **NOTATION** | | | **DESCRIPTION** |
| 1. | Class |  |  | *Class Name* | Represents a collection of similar |
|  |  | *+ public* | *-attribute* | entities grouped together. |
|  |  | *-private* | *-attribute* |  |
|  |  | *# protected* |  |  |
|  |  |  | *+operation* |  |
|  |  |  | *+operation* |  |
|  |  |  | *+operation* |  |
| 2. | Association | Class A  Class A | NAME | Class B  Class B | Associations represent static relationships between classes. Roles represent the way the two classes see each other. |
| 3. | Actor |  | | | Interaction between the system and external environment |
| 4. | Aggregation | Class A  Class B |  | Class A  Class B | It aggregates several classes into a single class. |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation (uses) | uses | Used for additional process communication. |
| 6. | Relation (extends) | extends | Extends relationship is used when one the use case is similar to another use  case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processes. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use case | Uses case | Interaction between the system and external environment. |
| 14. | Component |  | Represents physical modules which are a collection of components. |

|  |  |  |  |
| --- | --- | --- | --- |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard, sensors, etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

# CHAPTER 1

# INTRODUCTION

## 

## GENERAL

his project delves into the strategic use of batch-processed data within **Azure Databricks** to

develop accurate **product demand forecasting models** for large-scale datasets. In today's data-driven world, where e-commerce platforms and retail giants handle **enormous volumes of transactional data**,there is an urgent need to transform raw information into actionable insights that can support timely business decisions.

This project addresses this need by harnessing **Azure Databricks’ powerful integration of Apache Spark** with Microsoft’s cloud ecosystem to create an end-to-end demand prediction pipeline.The core of this project revolves around the analysis of **historical sales data**, encompassing various attributes such as order frequency, product categories, customer purchase behavior, and seasonal trends.. With **in-memory computation, parallel processing, and autoscaling clusters**, Azure Databricks proves to be an ideal platform for building and executing machine learning models on large-scale datasets without performance bottlenecks.

By implementing **predictive algorithms** such as Random Forest, Decision Trees, and XGBoost, the project forecasts future product demand with high accuracy. These models learn from historical sales patterns to predict the quantity of each product likely to be demanded in upcoming cycles. Such insights are essential for **stock optimization, reducing overstock or understock risks, minimizing holding costs**, and improving overall customer satisfaction through better availability.

To enhance interpretability, the project incorporates **data visualization** tools and dashboards using platforms like **Power BI or Databricks notebooks**, enabling stakeholders to easily view demand forecasts, seasonal patterns, and category-level trends.

Overall, this project showcases a **robust, scalable, and cloud-native approach** to predictive analytics using **Azure Databricks ML**. It not only emphasizes the benefits of batch-processing architecture in big data environments but also presents a practical solution for **forecasting product demand**, leading to **smarter inventory decisions, optimized operations, and enhanced profitability** for modern businesses

## OBJECTIVE

## To implement a scalable demand forecasting solution using Azure Databricks that processes large-scale historical sales data through batch processing.

## To leverage Apache Spark for distributed computing to efficiently transform, clean, and prepare data for machine learning.

## To utilize Delta Lake for reliable and consistent data storage that supports ACID transactions and time-travel capabilities.

## To apply advanced machine learning algorithms including ARIMA, Facebook Prophet, Random Forest, and XGBoost for modeling product demand and improving forecasting accuracy.

## To integrate MLflow for automating the machine learning pipeline including experiment tracking, hyperparameter tuning, model versioning, and evaluation.

## To compare and analyze the performance of the proposed system with traditional forecasting methods in terms of accuracy, speed, and scalability.

## To provide actionable insights from the forecast results that can help organizations make informed inventory and supply chain decisions.

# CHAPTER 2 LITERATURE SURVEY

## REVIEW OF RELATED RESEARCH PAPERS

**Title:** Designing Scalable Data Engineering Pipelines Using Azure andDatabricks

**Author:** Santosh Kumar Singu

**Year:** 2021

**Description:**

This paper explores the design and implementation of scalable data engineering pipelines using the robust capabilities of Azure and Apache Databricks. The author presents a systematic approach to tackling the challenges that arise while handling large-scale data workloads, particularly in industries like retail and e-commerce, where massive volumes of structured and semi-structured data need to be processed daily.The paper emphasizes the architecture of data pipelines that incorporate Azure Data Lake for cost-effective storage, Azure Data Factory for orchestration, and Databricks for processing and transformation. Spark jobs are optimized through parallel processing, schema evolution handling, and adaptive execution planning, which is especially useful for iterative machine learning workflows and batch data processing.An end-to-end pipeline is demonstrated that includes data ingestion, cleaning, transformation, and finally, data modeling for predictive analytics. This modular approach is particularly suitable for demand forecasting systems where periodic ingestion of sales logs and customer activity logs is critical. The author showcases how Azure's autoscaling and Databricks' cluster configuration features can significantly reduce infrastructure overhead while maintaining high performance and fault tolerance.This paper contributes a valuable perspective on using cloud-native tools in modern data engineering workflows, particularly relevant to businesses aiming to predict product demand using batch-processed pipelines in Azure Databricks

**Title:** Predicting the Ratings of Amazon Products Using Big Data

**Author:** Jongwook Woo, Monika Mishra

**Year:** 2020

**Description:**

This research explores how predictive modeling techniques can be employed on large-scale datasets to forecast product ratings on e-commerce platforms like Amazon. The authors use a real-world dataset of user reviews, product metadata, and star ratings, which are often used as indicators of product demand and consumer sentiment. The dataset is massive, consisting of millions of records, which necessitates the use of big data tools to handle storage, preprocessing, and model training.The study leverages Oracle Big Data and Azure Cloud services to preprocess and transform the data. It then implements machine learning models, including Random Forest, Gradient Boosting, and Decision Trees, to predict product ratings. The models are trained on various features like product category, user behavior, and review length, making them suitable for capturing nonlinear relationships between different factors affecting customer satisfaction and product demand.Azure's scalable infrastructure allows parallel training and testing of these models, reducing processing time and improving prediction accuracy. While the study primarily focuses on rating prediction, it has strong implications for product demand forecasting, as high ratings often correlate with increased demand.This paper is instrumental in showing how big data and machine learning, when paired with Azure's ecosystem, can be used to forecast product-related metrics, offering strong relevance to any demand prediction model designed using batch-processed data

**Title:** Demand Forecasting with Machine Learning

**Author:** K Chang

**Year:** 2024

**Description**:

In this study, the focus is placed on implementing machine learning for demand forecasting in supply chain and inventory management. The research outlines a step-by-step methodology beginning with defining the business problem, understanding the characteristics of available data, and applying traditional forecasting techniques like ARIMA and exponential smoothing as a baseline.Once the baselines are established, the study transitions into machine learning territory, employing models such as XGBoost, LSTM (Long Short-Term Memory networks), and Random Forest to improve forecasting precision. The models are evaluated on their ability to handle temporal data, detect seasonality, and adapt to changing trends, which are essential components in accurately predicting product demand.A key contribution of this work is its emphasis on model evaluation and the importance of feature engineering. The study highlights how temporal features, promotional events, and even weather data can significantly enhance the forecasting capability. The implementation is performed in a cloud-based environment using services compatible with Azure Databricks, emphasizing scalability, automation, and integration with batch-processing pipelines.

This paper serves as a contemporary reference for using intelligent systems to forecast demand, aligning beautifully with the goals of projects focusing on predictive analytics in Azure Databricks ML. It emphasizes not just prediction accuracy but also the importance of efficient pipeline design and processing architecture

# CHAPTER 3

# DESIGN AND DEVELOPMENT

## GENERAL

## Design and development cover a wide spectrum of activities, beginning with the initial concept and planning, followed by prototyping, testing, and final product delivery. This process includes problem identification, exploring potential solutions, and continuously refining through iterative design and development cycles to ensure the end product aligns with user requirements and business objectives

**3.2 EXISTING SYSTEM**

In the existing system, Azure Databricks is utilized for handling large workloads efficiently using Apache Spark. It supports batch and real-time data processing, provides shared workspaces for team collaboration, and integrates seamlessly with Azure services. Azure Databricks also supports popular programming languages such as Python and SQL for fast and efficient data processing.

The traditional approach to processing large-scale e-commerce data involves collecting data using ETL frameworks or custom scripts and storing it in RDBMS systems like MySQL or distributed storage like HDFS. Data cleaning and transformation are done using SQL or standalone tools, often requiring complex queries. Batch processing with tools like Apache Hive or Hadoop MapReduce handles analysis, but the reliance on disk operations makes it slow. Insights are derived from static datasets using BI tools, limiting real-time analytics. Machine learning is typically applied in isolated environments on static data, restricting scalability and integration.

This approach faces challenges such as poor scalability, high latency, resource-intensive maintenance, and a lack of flexibility for real-time analytics. These limitations highlight the need for modern solutions like PySpark for efficient, scalable, and real-time data processing.

## 3.2.1 EXISTING SYSTEM DISADVANTAGES

* + - * Performance
      * Scalability
      * Latency
      * High Maintenance

## PROPOSED SYSTEM

The proposed system leverages PySpark, the Python API for Apache Spark, to optimize data processing pipelines for large-scale e-commerce datasets. Unlike traditional approaches, this system harnesses the power of in-memory distributed computing to process massive datasets with low latency and high efficiency. By implementing tailored algorithms, it ensures effective data transformation and preparation specific to e-commerce analytics. The system supports real-time data processing using Spark Streaming, enabling immediate insights into customer behavior, sales trends, and inventory needs.

PySpark enables the pipeline to handle vast datasets seamlessly by distributing the workload across multiple nodes, making it scalable and well-suited for growing e-commerce data demands. The system is designed to streamline data transformation and cleaning processes, using advanced algorithms that are optimized for e-commerce-specific needs, such as revenue analysis, customer behavior tracking, and product popularity assessments.

This open-source solution also reduces reliance on proprietary platforms, making it cost-effective and adaptable across different environments. Overall, the proposed system offers enhanced performance, scalability, and flexibility, addressing the shortcomings of traditional methods while enabling businesses to make data-driven decisions in real time.

## PROPOSED SYSTEM ADVANTAGES:

## Better Performance

* Enhanced Scalability
* Improved Insights
* Flexibility

## REQUIREMENTS

#### HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should be what the system does and not how it should be implemented.

PROCESSOR : DUAL CORE 2 DUOS. RAM : 2-4GB DDR RAM HARD DISK : 250 GB

#### SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

OPERATING SYSTEM : WINDOWS 7/8/10

PLATFORM : AZURE DATABRICKS PROGRAMMING LANGUAGE : PYTHON, SQL FRAMEWORK : PYSPARK,MATPLOTLIB

## FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Valid Azure Databricks (ADB)

Account with required Permissions and Actions. Uninterrupted Internet Connection with Good Bandwidth.

Ensure the pipeline is scalable to handle increasing data volumes using PySpark's distributed computing capabilities.Develop efficient data transformation and cleansing processes to prepare raw e-commerce data for analysis and downstream systems.

## NON-FUNCTIONAL REQUIREMENTS

The major non-functional Requirements of the system are as follows

###### Usability

The system is designed to have an automated data processing workflow using PySpark on Azure Databricks. With minimal user intervention, the process ensures efficiency and ease of use, enabling seamless log analysis and visualization.

###### Reliability

The system leverages Python and PySpark, ensuring reliability through robust code and the fault-tolerant nature of Spark’s distributed processing framework. Azure Databricks, being a managed cloud service, adds additional reliability with its highly available infrastructure.

###### Performance

The use of PySpark ensures high performance, enabling efficient processing of large volumes of web server log data. Combined with Azure Databricks, the system delivers near real-time results with minimal latency, providing fast responses to user queries and insights.

###### Supportability

The system is designed to be cross-platform and highly supportable. As PySpark and Azure Databricks work seamlessly across different cloud environments and support various data formats, the system can run on any hardware or software environment that supports Python and Spark.

###### Implementation

The system is implemented in a cloud environment using Azure Databricks for distributed data processing and analysis. Python is used as the primary programming language, and PySpark handles the data transformation tasks. Azure Blob Storage and Databricks clusters are utilized for data storage and processing, ensuring a scalable and secure implementation.

###### Interface

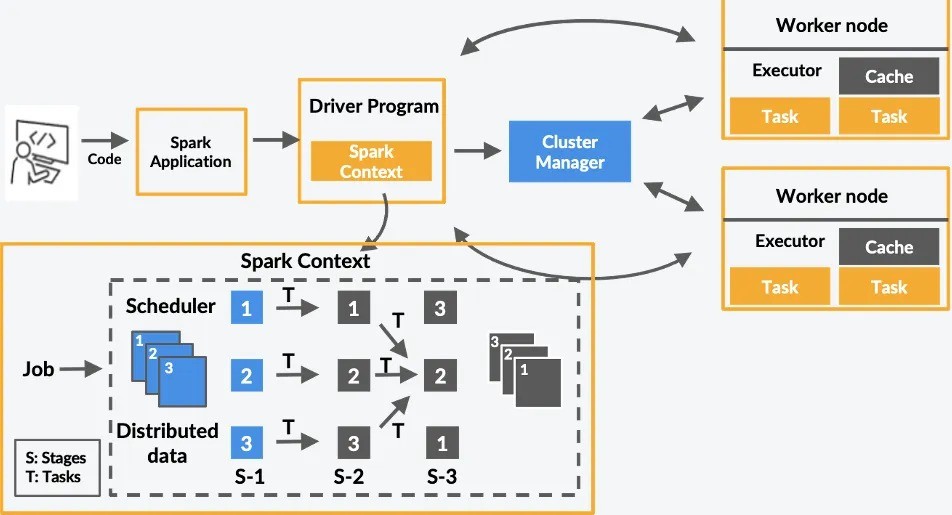
The user interface is integrated with Databricks notebooks, which provide a collaborative, interactive, and user-friendly platform for writing Python scripts, running PySpark jobs, and visualizing results. The interface supports rich visualizations for analyzing trends and patterns in log data, ensuring better usability for end users.

# CHAPTER 4 DESIGN ENGINEERING

## GENERAL

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished products.

## SYSTEM ARCHITECTURE

****

###### Figure - 4.2: SYSTEM ARCHITECTURE

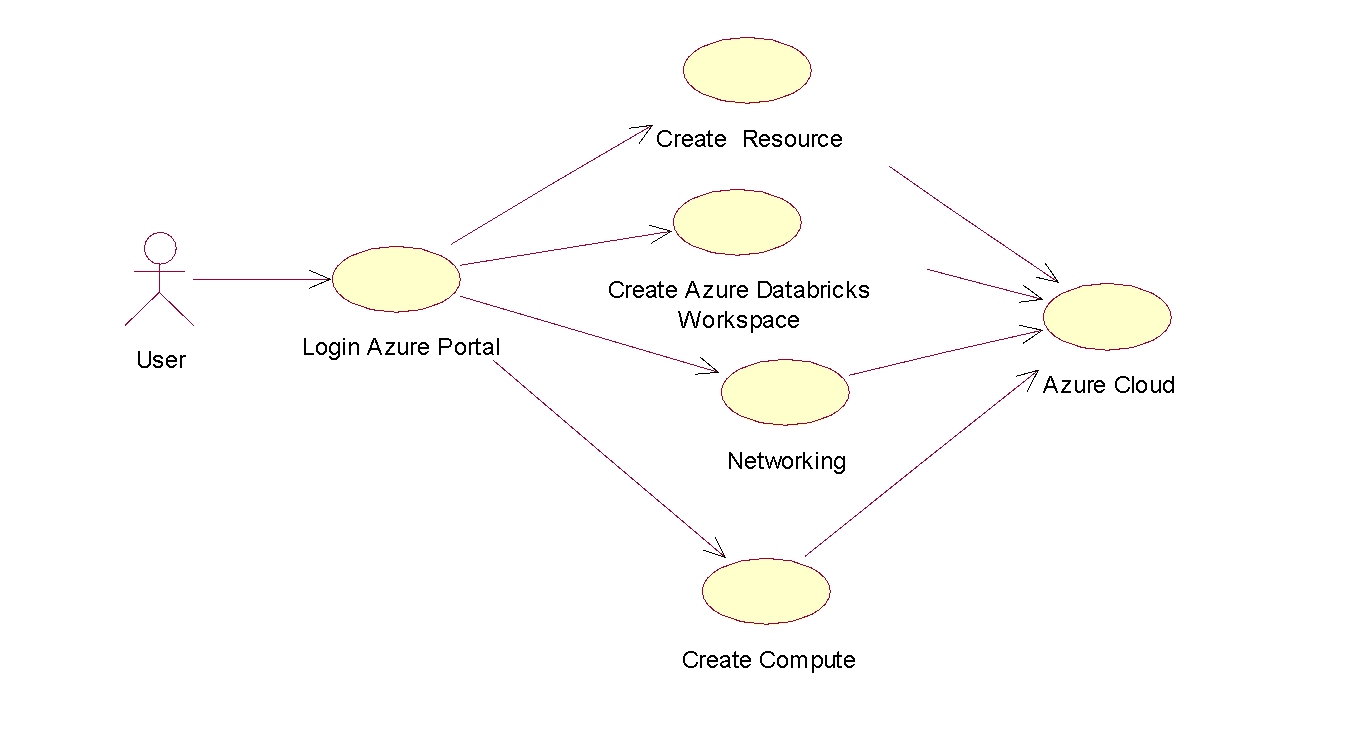
Apache Spark is a distributed computing framework that processes large datasets efficiently across a cluster of machines. At the core of its architecture is the Driver Program, which acts as the brain of the Spark application. The driver breaks the big job (like analyzing data) into smaller tasks, manages them, and communicates with the Cluster Manager to request resources like CPU and memory.

The Cluster Manager coordinates with multiple machines, called Worker Nodes, where the actual work happens. Each worker node runs Executors, which execute tasks and temporarily store data. Spark divides the job into Stages, and these stages are further split into Tasks, which operate on smaller chunks of data called partitions. This division allows Spark to process data in parallel, making it highly scalable and fast.

Data and tasks flow between the Driver Program, Cluster Manager, and Worker Nodes, enabling efficient execution. To optimize performance, Spark caches intermediate data in memory, avoiding repetitive computations. This combination of parallel processing and caching makes Spark powerful for large-scale data processing, such as filtering, aggregating, and analyzing datasets.

## UML DIAGRAMS

* + 1. **USE CASE DIAGRAM**

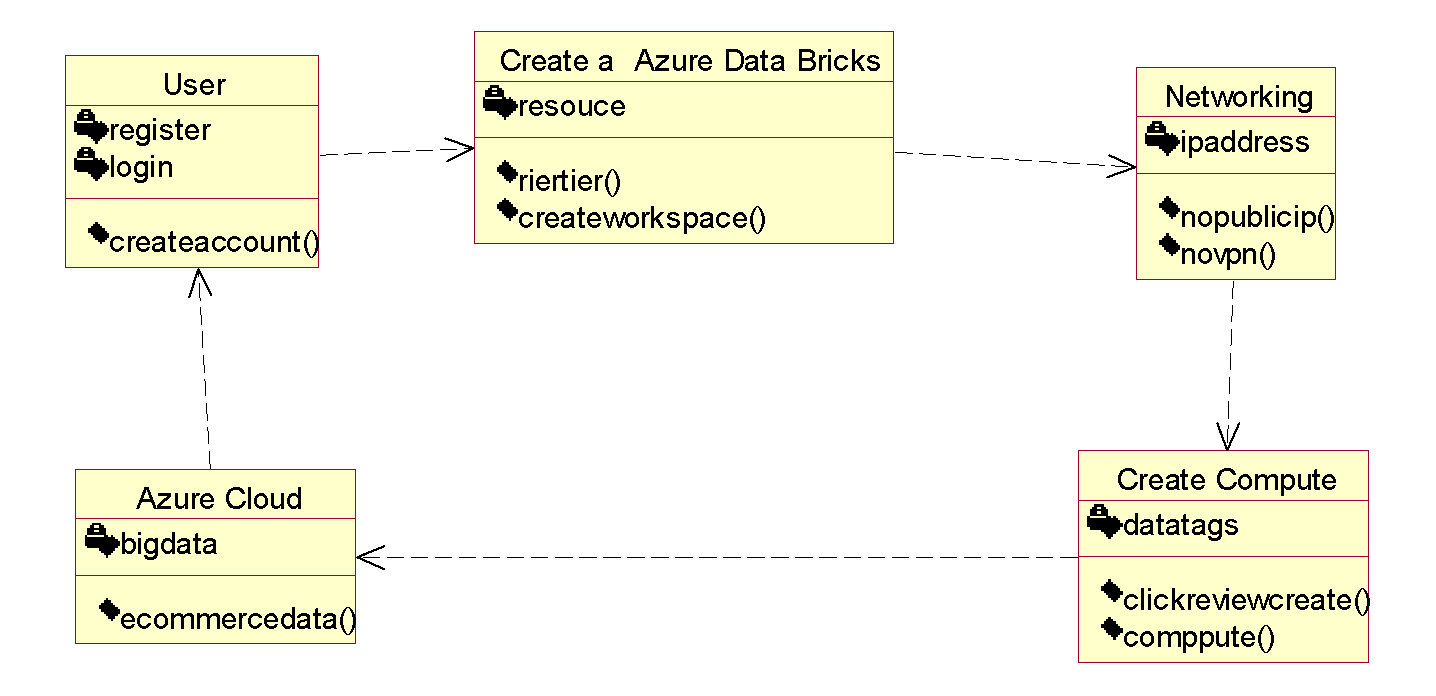
****

**Figure - 4.3.1: USE CASE DIAGRAM**

## EXPLANATION:

A use case diagram in the Unified Modeling Language (UML) is a model that shows how a user interacts with a system. It's a dynamic diagram that depicts the behavior of a system and how it affects interactions and changes in a process.

## CLASS DIAGRAM

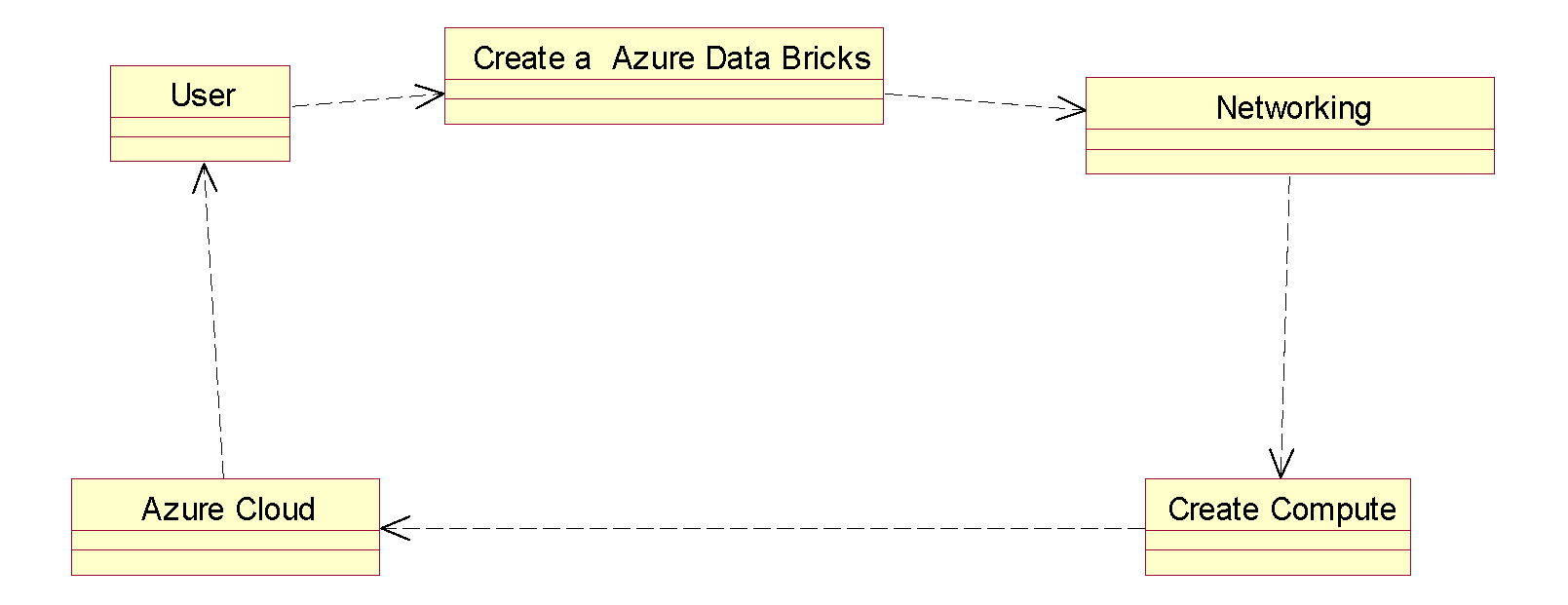
****

**Figure - 4.3.2: CLASS DIAGRAM**

## EXPLANATION:

A class diagram in Unified Modeling Language (UML) is a graphical representation of the structure of a system, including classes, interfaces, and objects. Class diagrams are used to model the static structure of a system, and are a fundamental part of the object modeling process.

## OBJECT DIAGRAM

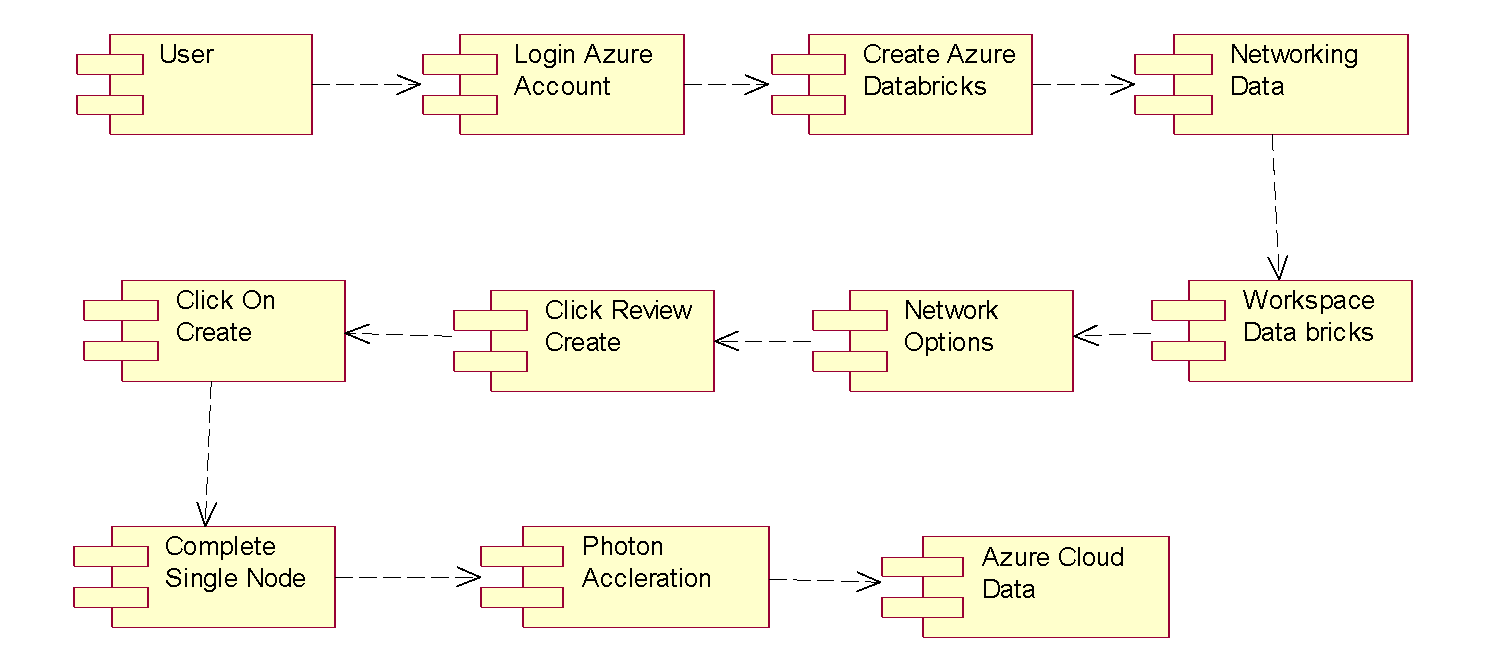
****

**Figure - 4.3.3: OBJECT DIAGRAM**

## EXPLANATION:

An object diagram in the Unified Modeling Language (UML) is a visual representation of the instances of a system's classes and interfaces at a specific point in time. Object diagrams are similar to class diagrams, but they focus on specific instances of classes and the links between the data.

## COMPONENT DIAGRAM

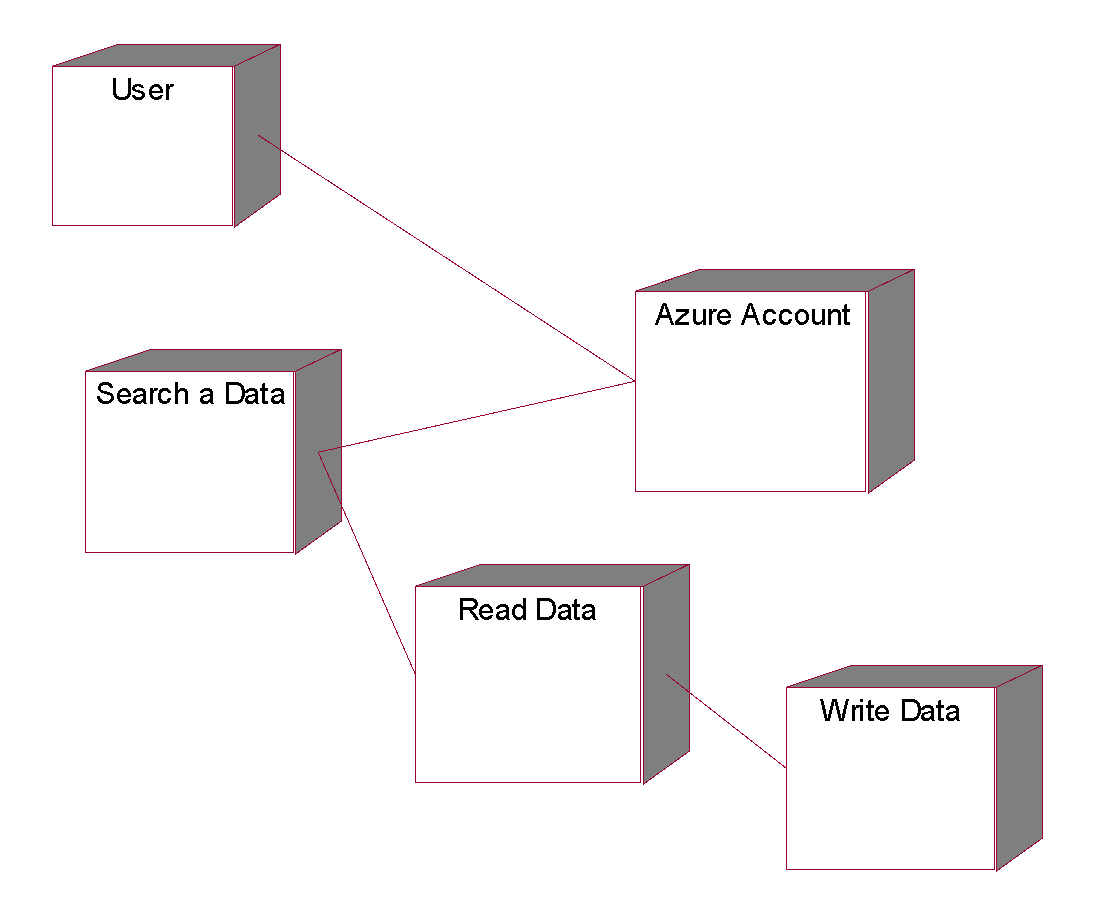
****

**Figure - 4.3.4: COMPONENT DIAGRAM**

## EXPLANATION:

Component diagrams can be used to model software systems at a high level or at a lower package level. They can help define the reusable and executable aspects of a software system, and can reveal configuration issues.

## DEPLOYMENT DIAGRAM

****

**Figure - 4.3.5: DEPLOYMENT DIAGRAM**

## EXPLANATION:

Deployment diagrams are typically created during the implementation phase of development. They are used by software and system engineers to understand and detail the complex network setups and infrastructure where the system will be deployed.

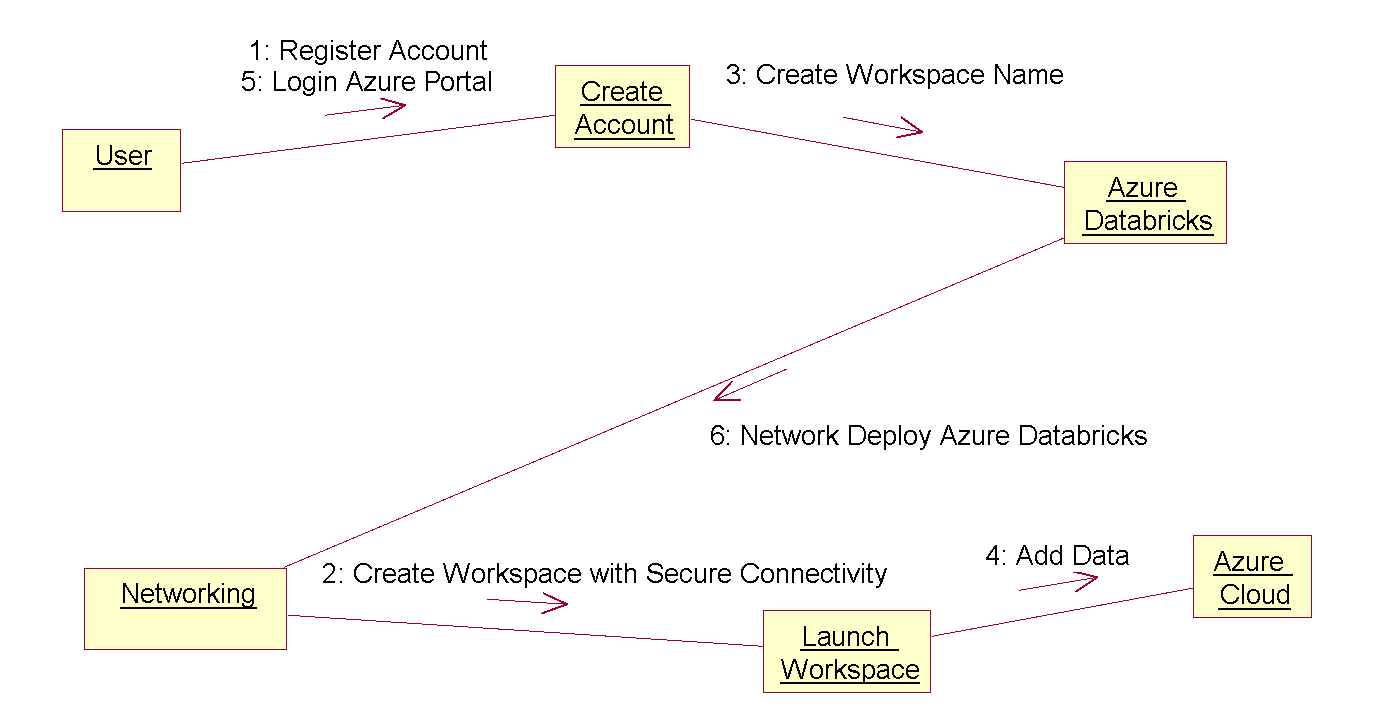
## SEQUENCE DIAGRAM

**Figure - 4.3.6: SEQUENCE DIAGRAM**

## EXPLANATION:

Sequence diagrams can help ensure that all stakeholders have a clear understanding of the system's behaviour, which can aid in development, troubleshooting, and modification. You can build a sequence diagram using a UML sequence template or starter diagram.

## COLLABORATION DIAGRAM

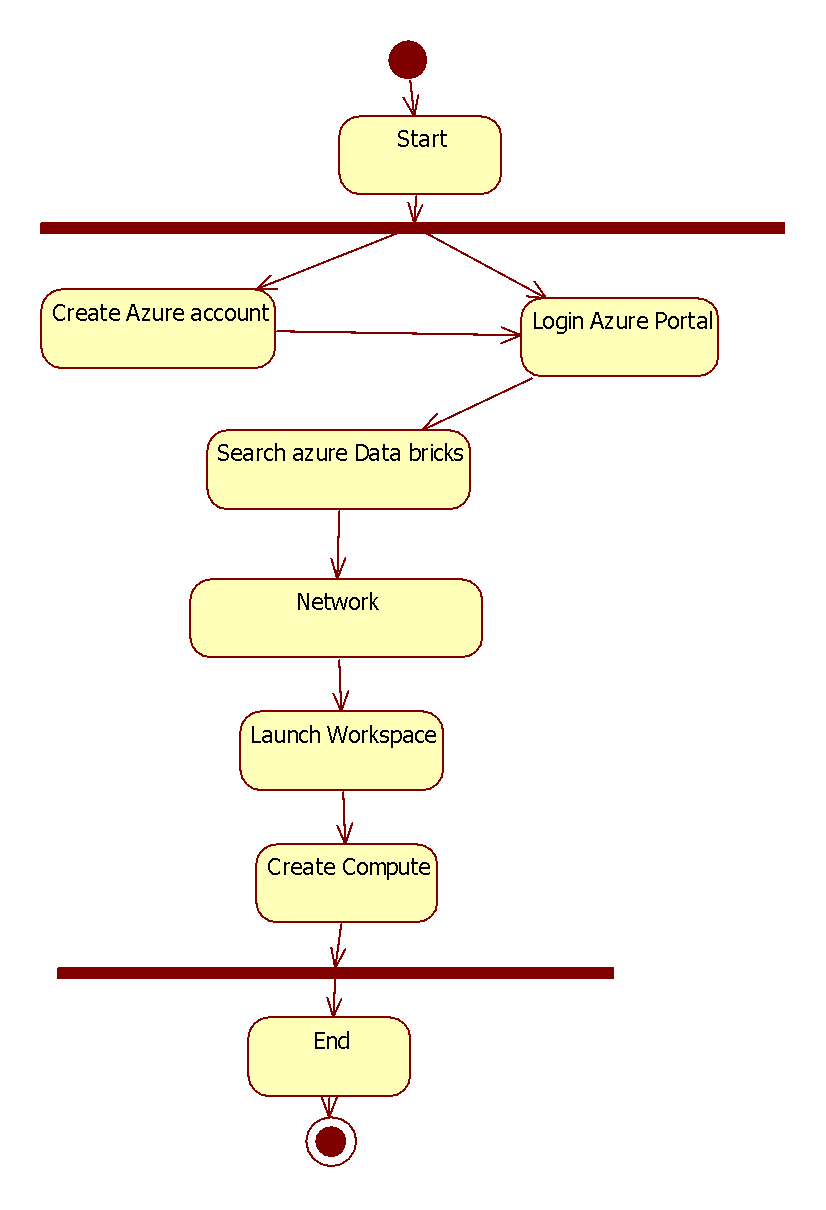
****

**Figure - 4.3.7: COLLABORATION DIAGRAM**

## EXPLANATION:

Collaboration diagrams are a key part of UML, along with sequence diagrams, which are used to show the order in which messages are sent between objects. Both are known as interaction diagrams.

## STATE DIAGRAM

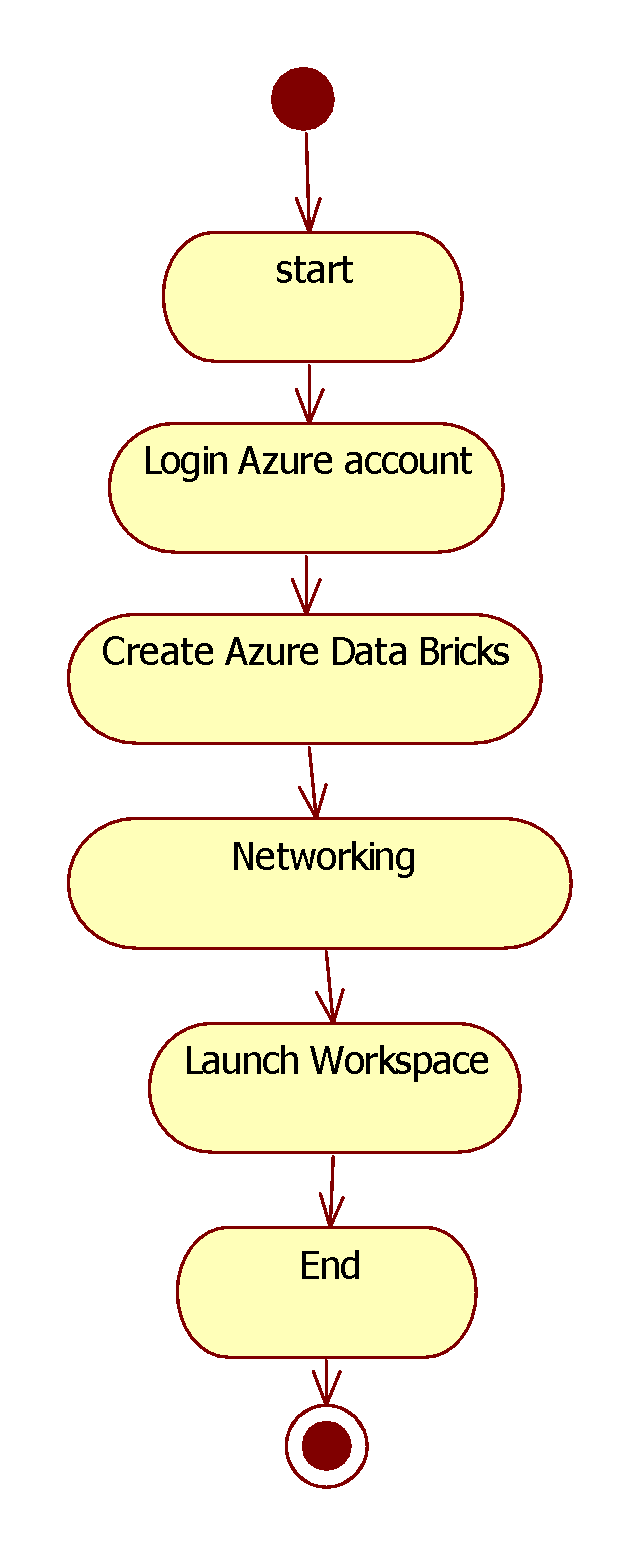
****

**Figure - 4.3.8: STATE DIAGRAM**

## EXPLANATION:

State diagrams typically start with a dark circle for the initial state and end with a bordered circle for the final state. They include states, transitions, events, and activities.

## ACTIVITY DIAGRAM

****

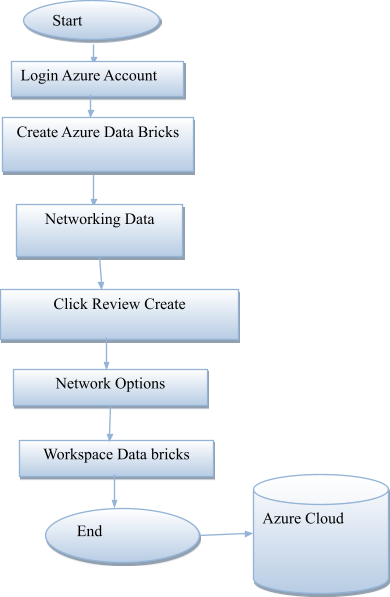
**Figure - 4.3.9: ACTIVITY DIAGRAM**

## EXPLANATION:

Activity diagrams can be used to model computational and organizational processes, and are often used in business modelling. They can be created at different stages of a project, such as before starting, during requirements, and during analysis and design.

## DATA FLOW DIAGRAM

**Level 1**

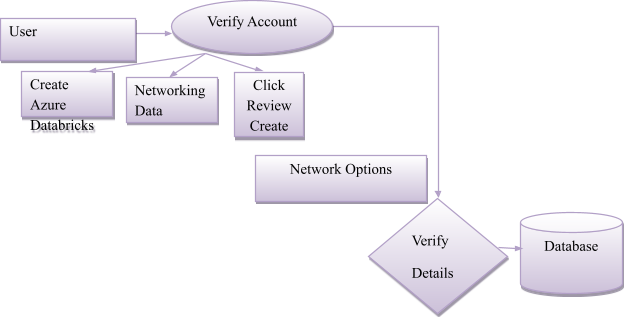
****

**Figure - 4.4: DATA FLOW DIAGRAM**

## EXPLANATION:

The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes. The data-flow diagram is a tool that is part of structured analysis and data modeling. When using UML, the activity diagram typically takes over the role of the data-flow diagram.

## E-R DIAGRAM

****

**Figure - 4.5: E-R DIAGRAM**

## EXPLANATION:

UML is used for planning software development, and is used in many different diagrams for various purposes. ER diagrams do not focus on the software, but rather the modelling of databases, which are usually part of a software system. UML diagrams are broader, and have many uses, and ERDs only have one purpose.

# CHAPTER 5 IMPLEMENTATION

## TECHNOLOGY USED:

###### Python:

Python is a high-level, interpreted, interactive, and object-oriented programming language, known for its simplicity and readability. It uses English keywords instead of punctuation, making it easier to understand compared to other programming languages. Python was developed by Guido van Rossum in the late 1980s and early 1990s at the National Research Institute for Mathematics and Computer Science in the Netherlands. It draws inspiration from several languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell scripting. The language is open-source and distributed under the GNU General Public License (GPL), with maintenance handled by a core development team, while van Rossum continues to play a key role in its progress.

###### Importance of Python

Python is an interpreted language, meaning programs are executed directly by the interpreter without prior compilation, similar to PERL and PHP. Its interactive nature allows users to write and test code in real time via a Python prompt. Additionally, Python supports object-oriented programming, encapsulating code within objects to promote modularity and reuse. Its simplicity and ease of use make it an ideal choice for beginner programmers, with applications ranging from text processing and web browsers to games.

###### Features of Python

Python is designed to be easy to learn, with a simple syntax and few keywords, making it accessible for beginners. Its code is both easy to read and maintain. The language comes with a broad standard library that is portable and compatible across platforms like UNIX, Windows, and macOS. Python supports an interactive mode for real-time testing and debugging, enhancing the development experience.

It is highly portable, running seamlessly on various hardware platforms, and extendable, allowing developers to add custom modules for improved efficiency. Python also supports GUI programming and provides robust tools for building scalable applications. Additionally, Python is versatile, supporting functional, structured, and object-oriented programming methods, and integrates easily with languages like C, C++, and Java.

###### Libraries Used in Python

Python has an extensive ecosystem of libraries that simplify complex tasks. Popular libraries include **NumPy,** which is used for handling N-dimensional arrays, and **Pandas,** a data analysis library featuring structures like DataFrames. **Matplotlib** is widely used for creating high-quality 2D plots, while **Seaborn** simplifies statistical data visualization with aesthetically pleasing charts. For machine learning tasks, **Scikit-learn** provides efficient algorithms for data analysis and mining.



###### Figure - 5.1.1: NumPy, Pandas, Matplotlib, Scikit-learn

**What is PySpark?**

PySpark is the Python API for Apache Spark, a distributed computing framework designed for big data processing. It leverages Spark's scalability and performance while providing the simplicity of Python for ease of use. PySpark enables developers to perform advanced data analysis and machine learning on massive datasets using Spark's distributed computing capabilities.

###### Uses of PySpark in Big Data Analysis

PySpark excels at processing massive datasets through distributed computing, breaking large tasks into smaller ones handled across a cluster of machines. It supports both batch and real-time data processing, making it highly versatile for big data workflows.

The DataFrame API in PySpark allows for efficient data manipulation and query-like operations, similar to SQL. It integrates with MLlib, Spark’s machine learning library, enabling distributed machine learning operations. PySpark’s scalability makes it ideal for handling terabytes or even petabytes of data, ensuring fault tolerance and minimizing the risk of data loss during hardware failures.

###### How PySpark Helps in Big Data Analysis

PySpark allows businesses to process and analyze vast datasets faster than traditional tools. It facilitates advanced analytics and machine learning, providing insights into structured, semi-structured, and unstructured data. PySpark integrates seamlessly with tools like Hadoop Distributed File System (HDFS) and cloud platforms like Azure Databricks, making it a preferred choice for organizations working with large-scale data. It supports live data stream analysis, enabling real-time decision-making and actionable insights, making it an essential tool in the modern era of big data.

# CHAPTER 6

**CODE IMPLEMENTATION**

**6.1 Input CSV file**

dbutils.fs.put("/FileStore/ecommerce\_transactions.csv", """

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| product\_id | store\_id | price | promotion | day\_of\_week | past\_sales | demand | date |
| P2 | S2 | 38.08 | 1 | 6 | 149 | 259 | ######## |
| P1 | S5 | 40.06 | 1 | 0 | 86 | 230 | ######## |
| P1 | S3 | 23.81 | 0 | 1 | 145 | 269 | ######## |
| P3 | S4 | 10.32 | 1 | 2 | 72 | 240 | ######## |
| P2 | S1 | 12.84 | 0 | 3 | 35 | 224 | ######## |
| P8 | S2 | 27.84 | 0 | 4 | 129 | 243 | ######## |
| P3 | S4 | 46.16 | 0 | 5 | 106 | 205 | ######## |
| P4 | S4 | 12.2 | 1 | 6 | 113 | 257 | ######## |
| P10 | S4 | 25.78 | 1 | 0 | 35 | 204 | ######## |
| P8 | S3 | 15.57 | 0 | 1 | 122 | 251 | ######## |
| P9 | S1 | 41.17 | 1 | 2 | 41 | 199 | ######## |
| P5 | S1 | 7.11 | 1 | 3 | 68 | 247 | ######## |
| P10 | S4 | 14.47 | 0 | 4 | 178 | 278 | ######## |
| P4 | S1 | 19.67 | 1 | 5 | 133 | 287 | ######## |
| P9 | S1 | 19.15 | 0 | 6 | 32 | 214 | ######## |
| P4 | S4 | 15.45 | 1 | 0 | 102 | 255 | ######## |
| P8 | S3 | 7.76 | 0 | 1 | 196 | 278 | ######## |
| P10 | S4 | 47.16 | 1 | 2 | 77 | 220 | ######## |
| P2 | S2 | 18.64 | 0 | 3 | 190 | 284 | ######## |
| P7 | S5 | 9.83 | 0 | 4 | 82 | 224 | ######## |
| P2 | S2 | 41.16 | 1 | 5 | 197 | 269 | ######## |
| P6 | S1 | 6.39 | 0 | 6 | 112 | 253 | ######## |
| P2 | S2 | 8.77 | 1 | 0 | 170 | 300 | ######## |
| P10 | S5 | 18.92 | 1 | 1 | 159 | 284 | ######## |
| P3 | S1 | 20.98 | 1 | 2 | 85 | 223 | ######## |
| P5 | S4 | 35.91 | 1 | 3 | 84 | 216 | ######## |
| P9 | S2 | 15 | 0 | 4 | 25 | 208 | ######## |
| P9 | S4 | 22.79 | 1 | 5 | 80 | 231 | ######## |
| P4 | S4 | 6.09 | 0 | 6 | 33 | 216 | ######## |
| P7 | S1 | 32.56 | 0 | 0 | 107 | 223 | ######## |
| P8 | S1 | 26.73 | 1 | 1 | 53 | 221 | ######## |
| P8 | S1 | 28.86 | 0 | 2 | 41 | 194 | ######## |
| P2 | S2 | 7.65 | 0 | 3 | 85 | 230 | ######## |
| P6 | S2 | 49.86 | 1 | 4 | 34 | 190 | ######## |
| P7 | S3 | 26.44 | 1 | 5 | 96 | 239 | ######## |
| P6 | S1 | 42.4 | 1 | 6 | 94 | 241 | ######## |
| P8 | S3 | 36.7 | 1 | 0 | 62 | 219 | ######## |
| P10 | S1 | 33.33 | 1 | 1 | 30 | 199 | ######## |
| P7 | S5 | 48.68 | 0 | 2 | 134 | 193 | ######## |
| P9 | S4 | 28.3 | 0 | 3 | 55 | 209 | ######## |
| P4 | S5 | 26.51 | 0 | 4 | 149 | 243 | ######## |
| P5 | S5 | 38.12 | 0 | 5 | 97 | 178 | ######## |
| P9 | S3 | 15.47 | 0 | 6 | 81 | 237 | ######## |
| P1 | S4 | 5.95 | 1 | 0 | 105 | 262 | ######## |
| P3 | S2 | 43.31 | 1 | 1 | 38 | 188 | ######## |
| P5 | S1 | 29.25 | 1 | 2 | 111 | 235 | ######## |
| P8 | S4 | 37.34 | 1 | 3 | 28 | 190 | ######## |
| P9 | S5 | 15.64 | 0 | 4 | 33 | 216 | ######## |
| P8 | S5 | 29.57 | 0 | 5 | 153 | 225 | ######## |
| P4 | S5 | 36.91 | 1 | 6 | 50 | 208 | ######## |
| P5 | S4 | 37.79 | 0 | 0 | 42 | 172 | ######## |
| P10 | S2 | 8.51 | 1 | 1 | 91 | 239 | ######## |
| P10 | S2 | 38.39 | 0 | 2 | 25 | 156 | ######## |
| P9 | S1 | 45.81 | 1 | 3 | 66 | 216 | ######## |
| P9 | S3 | 17.1 | 0 | 4 | 123 | 262 | ######## |
| P7 | S1 | 42.9 | 0 | 5 | 39 | 182 | ######## |
| P6 | S3 | 28.92 | 0 | 6 | 189 | 256 | ######## |
| P2 | S2 | 42.4 | 1 | 0 | 197 | 278 | ######## |
| P3 | S4 | 28.03 | 1 | 1 | 97 | 248 | ######## |
| P4 | S4 | 49.91 | 0 | 2 | 30 | 155 | ######## |
| P4 | S3 | 8.16 | 1 | 3 | 76 | 237 | ######## |
| P4 | S3 | 26.5 | 1 | 4 | 193 | 293 | ######## |
| P2 | S5 | 47.79 | 0 | 5 | 121 | 226 | ######## |
| P1 | S2 | 5.16 | 0 | 6 | 156 | 285 | ######## |
| P3 | S3 | 21.32 | 0 | 0 | 84 | 216 | ######## |
| P7 | S2 | 40.28 | 1 | 1 | 169 | 261 | ######## |
| P2 | S4 | 27.9 | 0 | 2 | 134 | 228 | ######## |
| P7 | S2 | 47.14 | 1 | 3 | 197 | 272 | ######## |
| P10 | S5 | 12.73 | 0 | 4 | 135 | 258 | ######## |
| P9 | S5 | 8.9 | 0 | 5 | 183 | 285 | ######## |
| P4 | S2 | 27.42 | 0 | 6 | 150 | 243 | ######## |
| P3 | S5 | 37.33 | 1 | 0 | 186 | 282 | ######## |
| P8 | S2 | 34.18 | 1 | 1 | 33 | 192 | ######## |
| P4 | S2 | 23.09 | 0 | 2 | 66 | 212 | ######## |
| P8 | S3 | 35.97 | 1 | 3 | 22 | 187 | ######## |
| P3 | S3 | 5.95 | 0 | 4 | 97 | 250 | ######## |
| P6 | S4 | 35.27 | 0 | 5 | 133 | 227 | ######## |
| P10 | S5 | 6.89 | 1 | 6 | 95 | 257 | ######## |
| P2 | S4 | 45.86 | 0 | 0 | 111 | 199 | ######## |
| P5 | S2 | 18.91 | 0 | 1 | 22 | 181 | ######## |
| P2 | S3 | 21.11 | 1 | 2 | 64 | 215 | ######## |
| P2 | S1 | 33.29 | 1 | 3 | 158 | 270 | ######## |
| P1 | S5 | 40.05 | 0 | 4 | 39 | 179 | ######## |
| P10 | S2 | 8.19 | 1 | 5 | 22 | 218 | ######## |
| P4 | S4 | 21.74 | 0 | 6 | 181 | 275 | ######## |
| P4 | S5 | 16.47 | 0 | 0 | 166 | 267 | ######## |
| P1 | S3 | 28.04 | 0 | 1 | 59 | 199 | ######## |
| P5 | S5 | 36.83 | 0 | 2 | 86 | 209 | ######## |
| P8 | S2 | 33.41 | 1 | 3 | 164 | 274 | ######## |
| P3 | S2 | 17.26 | 0 | 4 | 198 | 292 | ######## |
| P3 | S4 | 26.53 | 1 | 5 | 114 | 250 | ######## |
| P1 | S4 | 7.4 | 0 | 6 | 68 | 228 | ######## |
| P8 | S2 | 8.59 | 1 | 0 | 180 | 313 | ######## |
| P8 | S1 | 46.93 | 1 | 1 | 69 | 204 | ######## |
| P9 | S4 | 49.09 | 0 | 2 | 90 | 204 | ######## |
| P8 | S3 | 24.06 | 1 | 3 | 59 | 236 | ######## |
| P8 | S4 | 42.57 | 1 | 4 | 99 | 217 | ######## |
| P3 | S3 | 44.72 | 0 | 5 | 73 | 206 | ######## |
| P5 | S2 | 10.07 | 1 | 6 | 66 | 253 | ######## |
| P5 | S2 | 47.29 | 1 | 0 | 136 | 232 | ######## |

""", overwrite=True)

**6.2 CODE:**

from pyspark.sql import SparkSession

from pyspark.sql.types import \*

from pyspark.sql.functions import to\_date

# Create Spark session

spark = SparkSession.builder.appName("DemandPredictionRealTime").getOrCreate()

# Path to your real-time sales data (CSV format with string-type date)

file\_path = "dbfs:/user/hive/warehouse/real\_sales\_data\_matched\_columns"

 # Update if your path is different

# Define schema assuming 'sales\_date' is string in CSV

schema = StructType([

StructField("product\_id", StringType(), True),

StructField("store\_id", StringType(), True),

StructField("date", StringType(), True),  # We'll cast this below

StructField("price", DoubleType(), True),

StructField("promotion", IntegerType(), True),

StructField("day\_of\_week", IntegerType(), True),

StructField("past\_sales", IntegerType(), True),

StructField("demand", IntegerType(), True),])

# Read CSV data

real\_time\_df = spark.read.format('delta').load(file\_path)

# Convert sales\_date string to DateType

real\_time\_df = real\_time\_df.withColumn("date", to\_date("date", "yyyy-MM-dd"))

# Show the data

n = int(input("Enter the number of rows to display: "))

real\_time\_df.show(n)

from pyspark.ml.regression import RandomForestRegressor

from pyspark.ml import Pipeline

from pyspark.ml.feature import StringIndexer, VectorAssembler

# Define indexers for categorical columns

product\_indexer = StringIndexer(inputCol="product\_id", outputCol="product\_index", handleInvalid="keep")

store\_indexer   = StringIndexer(inputCol="store\_id",   outputCol="store\_index",   handleInvalid="keep")

# Assemble feature columns into one feature vector

assembler = VectorAssembler(

inputCols=["product\_index", "store\_index", "price", "promotion", "day\_of\_week", "past\_sales"],

outputCol="features")

Random Forest Regression model

rf = RandomForestRegressor(featuresCol="features", labelCol="demand")

# Pipeline for training

pipeline = Pipeline(stages=[product\_indexer, store\_indexer, assembler, rf])

print("\n##### STEP 2 OUTPUT: Pipeline Constructed #####")

print(pipeline)

# If n isn't defined above, define a default value

try:

n

except NameError:

n = 10  # default number of rows to show

# Build a mini-pipeline to show how features are constructed from the real data

from pyspark.ml import Pipeline as \_MiniPipeline

\_MiniPipeline(stages=[product\_indexer, store\_indexer, assembler]) \

.fit(real\_time\_df) \

.transform(real\_time\_df) \

.select("features", "demand") \

.show(n, truncate=False)

from pyspark.ml.evaluation import RegressionEvaluator

from pyspark.ml.tuning import ParamGridBuilder, CrossValidator

# --- Split the real-time DataFrame into train & test ---

train\_data, test\_data = real\_time\_df.randomSplit([0.8, 0.2], seed=42)

# --- Define the parameter grid for hyperparameter tuning ---

paramGrid = ParamGridBuilder() \

.addGrid(rf.numTrees, [50, 100]) \

.addGrid(rf.maxDepth, [5, 10]) \

.build()

# --- Set up CrossValidator with the full pipeline ---

cv = CrossValidator(

estimator=pipeline,

estimatorParamMaps=paramGrid,

evaluator=RegressionEvaluator(labelCol="demand", predictionCol="prediction", metricName="rmse"),

numFolds=3

)

# --- Train the model with hyperparameter tuning ---

cv\_model = cv.fit(train\_data)

# --- Get the best model from CrossValidator ---

best\_model = cv\_model.bestModel

# --- Save the trained model to DBFS or local path ---

from pyspark.ml.evaluation import RegressionEvaluator

# — Make predictions on the test data

predictions = cv\_model.transform(test\_data)

# — Show a few prediction results

print("\n##### STEP 4 OUTPUT: Predictions #####")

predictions.select("product\_id", "store\_id", "price", "promotion", "day\_of\_week",

"past\_sales", "demand", "prediction").show(10, truncate=False)

# — Evaluate model with RMSE and R2

evaluator\_rmse = RegressionEvaluator(labelCol="demand", predictionCol="prediction", metricName="rmse")

evaluator\_r2   = RegressionEvaluator(labelCol="demand", predictionCol="prediction", metricName="r2")

rmse = evaluator\_rmse.evaluate(predictions)

r2   = evaluator\_r2.evaluate(predictions)

print(f"\nModel Evaluation Metrics:")

print(f"✔️ Root Mean Squared Error (RMSE): {rmse}")

print(f"✔️ R-Squared (R²): {r2}")

# Extract the best model from the cross-validator pipeline

best\_pipeline\_model = cv\_model.bestModel

# Get the RandomForest model stage (assumes it's the last stage in the pipeline)

rf\_model = best\_pipeline\_model.stages[-1]

Get feature importances

importances = rf\_model.featureImportances.toArray()

# List the feature names used in the VectorAssembler

feature\_names = ["product\_index", "store\_index", "price", "promotion", "day\_of\_week", "past\_sales"]

# Display feature importances

print("\n--- Feature Importances ---")

for name, importance in zip(feature\_names, importances):

print(f"{name:<15}: {importance:.4f}")

import matplotlib.pyplot as plt

# Plotting feature importances

plt.figure(figsize=(8, 5))

plt.barh(feature\_names, importances, color="cornflowerblue")

plt.title("Feature Importances - Random Forest")

plt.xlabel("Importance Score")

plt.grid(True)

plt.gca().invert\_yaxis()

plt.tight\_layout()

plt.show()

# Save selected columns from predictions to CSV

predictions.select(

"product\_id", "store\_id", "price", "promotion", "day\_of\_week",

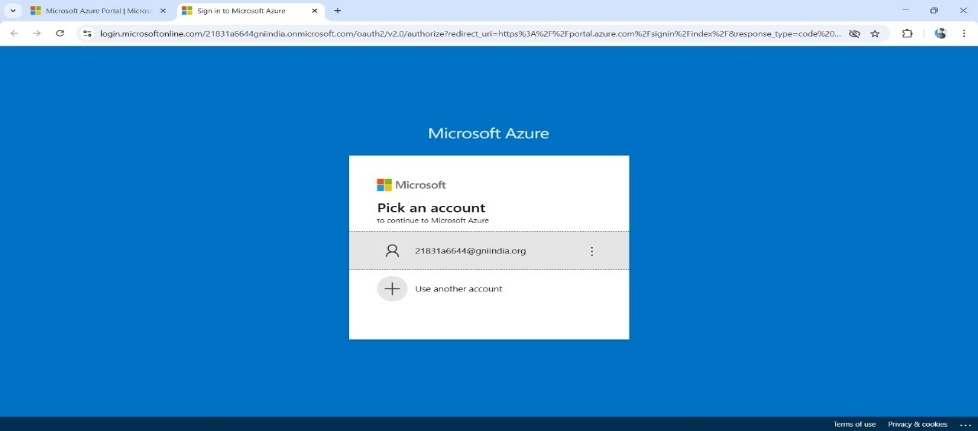
"past\_sales", "demand", "prediction", "date"

).coalesce(1).write.mode("overwrite").option("header", True).csv("/dbfs/FileStore/tables/predicted\_demand\_output")

print("\n Predictions successfully saved to: /dbfs/FileStore/tables/predicted\_demand\_output"

**7.1 SNAPSHOTS:**

# CHAPTER 7 SNAPSHOTS



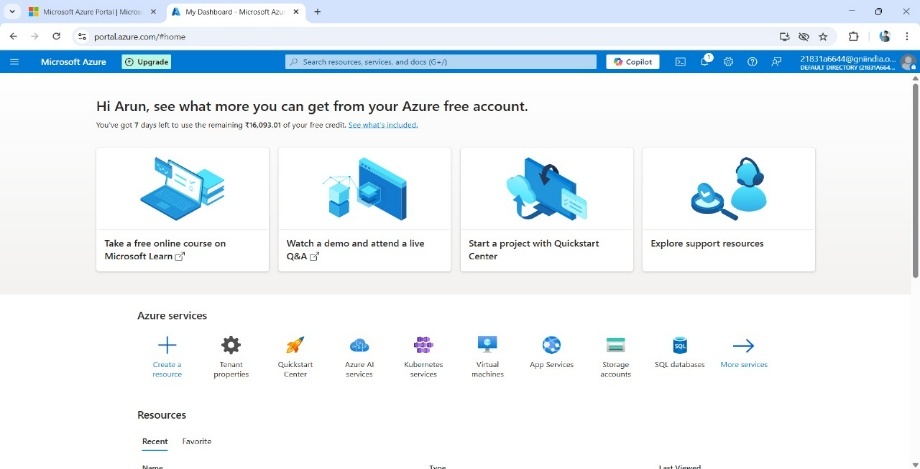
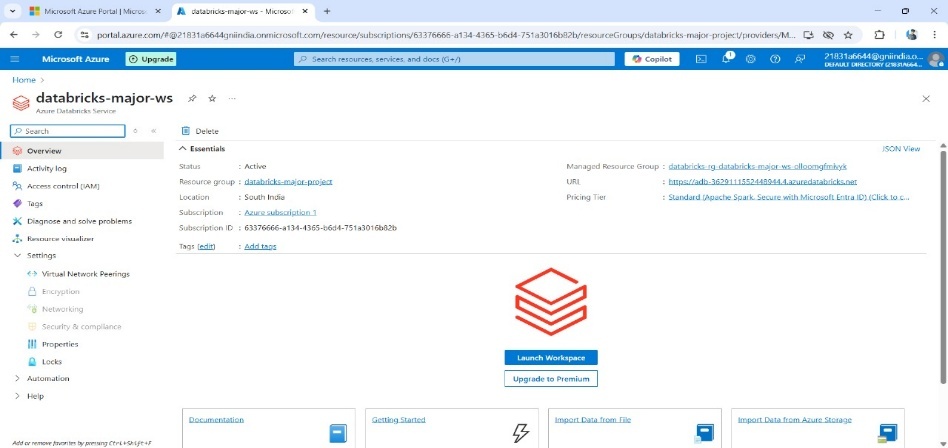


Fig 7.1.1 **:**Login to Azure and access Databricks service

##### EXPLANATION:

In this first step, we log in to the Azure portal and navigate to the Databricks service. Authentication is required to access cloud resources, so users either sign in with organizational credentials or personal Microsoft accounts. This step ensures a secure environment to manage and run big data workloads. Logging in provides access to the workspace, clusters, notebooks, and other Databricks features. It’s the starting point for any development work on AzureDatabricks.



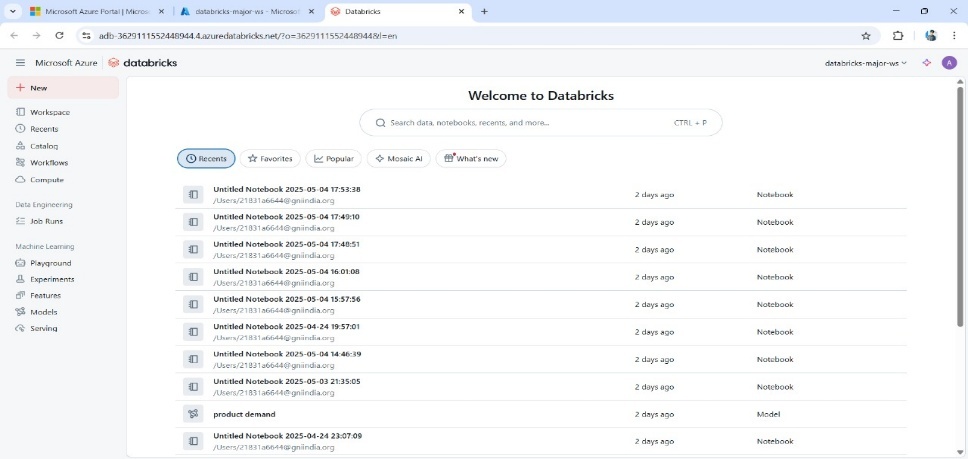
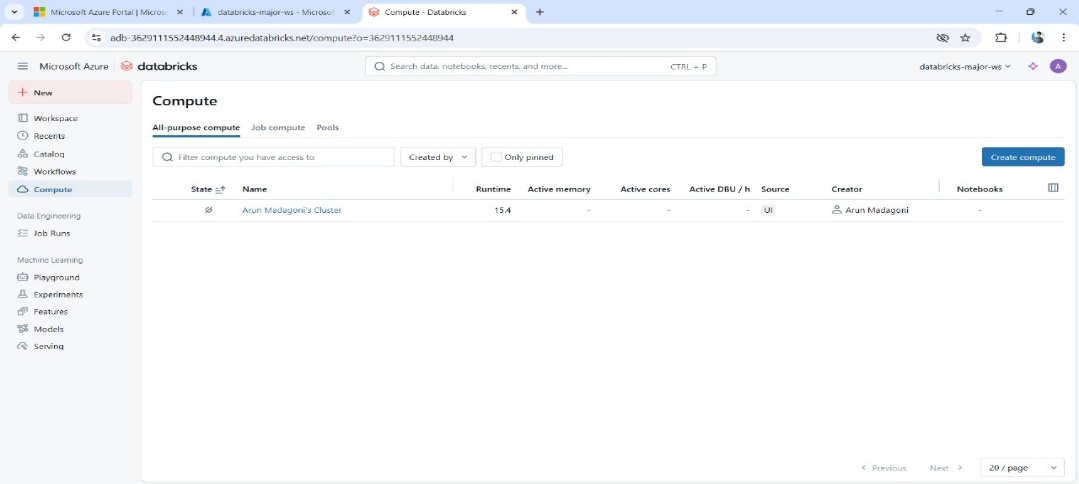


Fig 7.1.2: Launching the Databricks workspace after authentication

##### EXPLANATION:

After logging in, we access the **Databricks workspace**, which is a unified environment for running big data workloads. It includes tabs for clusters, notebooks, jobs, data, and models. The workspace serves as the developer’s playground to build, train, and test ML workflows. Here, users can organize files, create folders, and collaborate in real-time. It provides a centralized interface to interact with Spark and ML features seamlessly



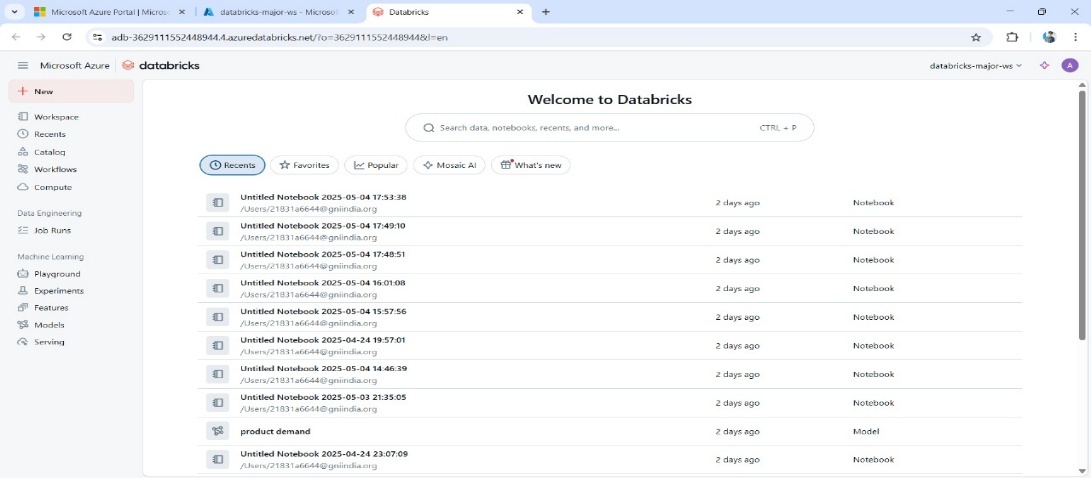
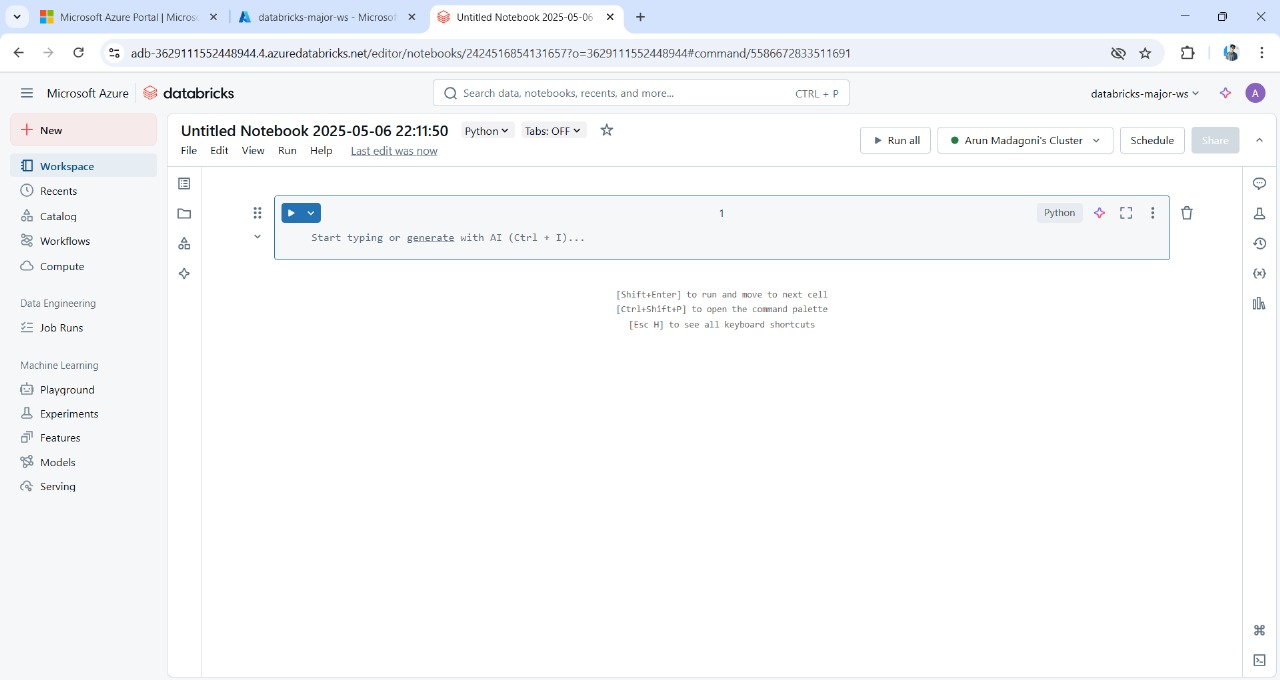


Fig 7.1.3: Creating and Running a Databricks Cluster

##### EXPLANATION:

A cluster is created by selecting the Databricks runtime version, instance types, and configuration settings. Once launched, the cluster provides the compute power for executing code in notebooks. It uses Apache Spark to process big data in parallel, ensuring fast computation. Running a cluster is crucial before running any ML model or transformation, as all operations depend on it. Clusters can also be autoscaled based on workload requirements.



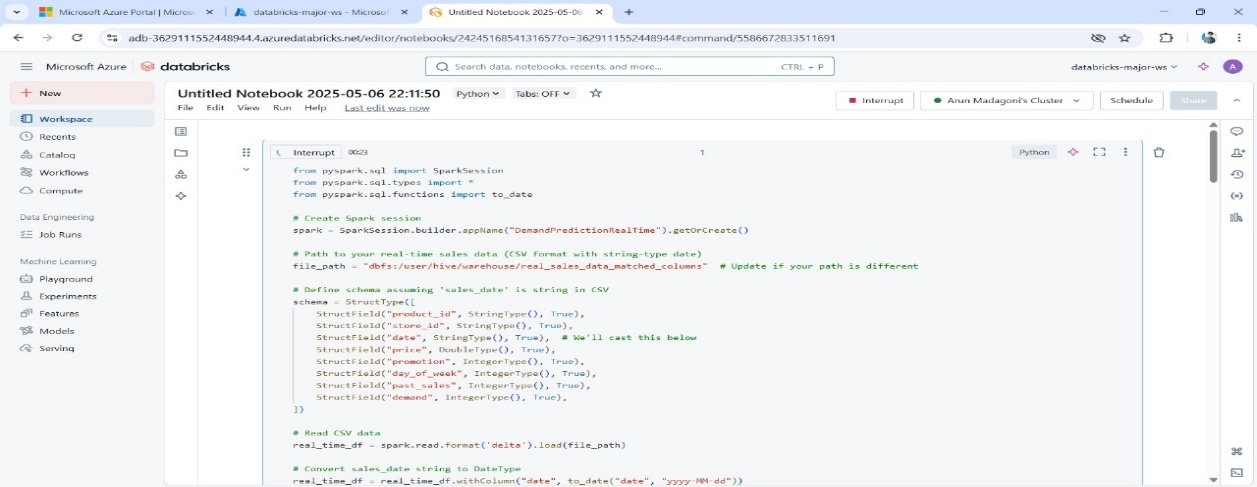
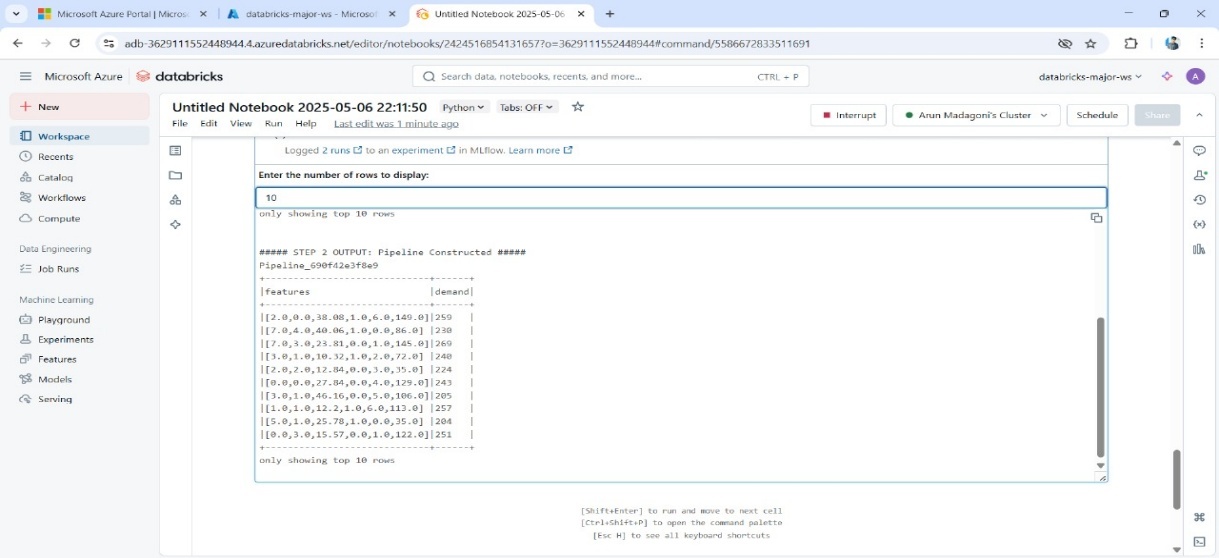


Fig 7.1.4 : Creating and opening a new notebook inside the workspace

**EXPLANTAION:**

Inside the workspace, we create or open a **notebook**, which acts like your personal scratchpad for writing code. It supports multiple languages like Python, SQL, and Scala. The notebook is where you type, document, and visualize your entire data journey. It can be shared with teammates or exported for reports. This step sets the foundation for interacting with data, running transformations, and training machine learning models.



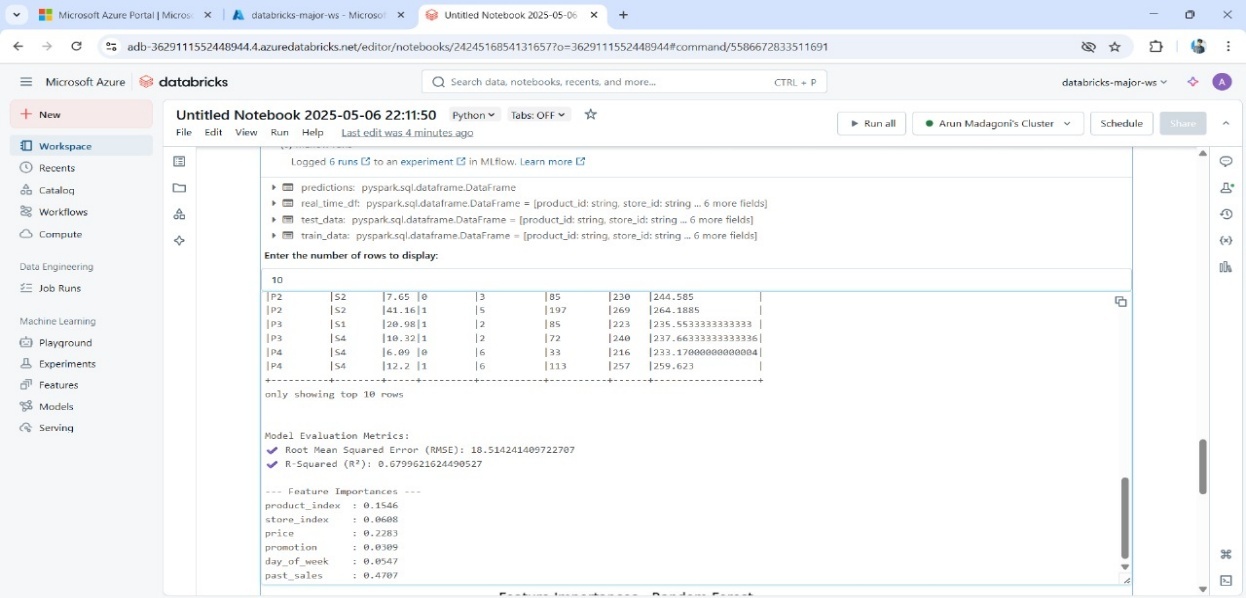


Fig 7.1.5 : Executing Code Cell-by-Cell

**EXPLANATION:**

Code inside the notebook is executed **cell by cell** using a connected cluster. This lets you run and debug code in smaller chunks, making it easier to monitor and tweak. Each cell can display visual outputs, error messages, or status info. Running code incrementally gives better control over logic and ensures accuracy in every step. You can also re-run individual cells without affecting the rest of the notebook.

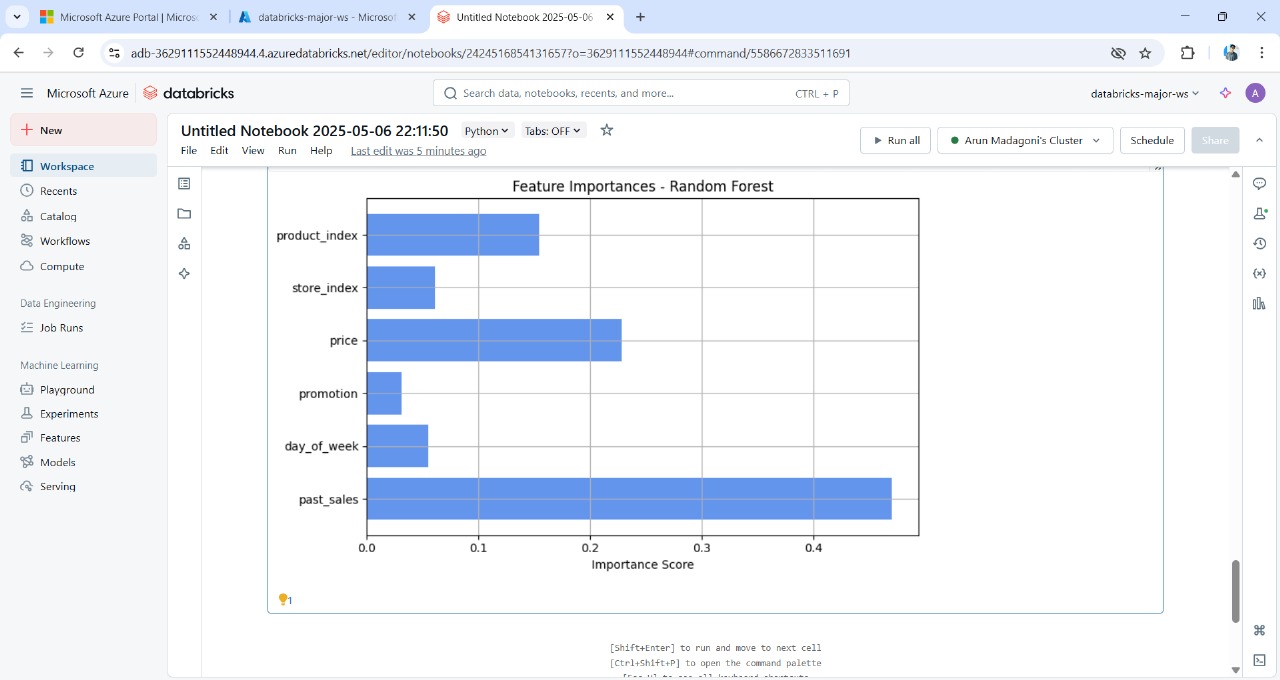


Fig 7.1.6 : Final visualization showcasing product popularity trends

**EXPLANATION:**

Finally, the results of the executed code are visible directly below the respective cell. Outputs could include tables, charts, logs, or model predictions depending on what was run. These outputs give real-time feedback and guide next steps — whether it’s tweaking parameters, visualizing results, or exporting insights

# 

# CHAPTER 8 SOFTWARE TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

## DEFINITION

Software testing is the process of verifying a system with the purpose of identifying any errors, gaps or missing requirements versus the actual requirement.

## SOFTWARE TESTING:

It is the process of testing the functionality and correctness of software by running it. Process of executing a program with the intent of finding an error.

Software Testing is usually performed for one of two reasons:

* + - Defect detection
    - Reliability estimation

## BLACK BOX TESTING

Applies to software systems or modules, tests functionality in terms of inputs and outputs at interfaces. Test reveals if the software function is fully operational with reference to requirements specification.

## WHITE BOX TESTING

Knowing the internal workings i.e., to test if all internal operations are performed according to program structures and data structures. To test if all internal components have been adequately exercised.

## SOFTWARE TESTING STRATEGIES

A strategy for software testing will begin in the following order:

1. Unit testing
2. Integration testing
3. System testing
4. Validation testing

## UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

## INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications,

e.g. components in a software system or – one step up – software applications at the company level – interact without error.

## SYSTEM TESTING:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

## TEST CASES

**Test Case 1:** Valid Input

**Input**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Product\_id | store\_id | price | promotion | day\_of\_week | past\_sales | demand | date |

1,101,201,2,15.99,2023-09-01 10:00:01

2,102,202,1,25.50,2023-09-01 10:05:03

3,101,203,4,5.00,2023-09-01 10:10:15

4,103,204,1,40.00,2023-09-01 10:15:22

5,102,201,3,15.99,2023-09-01 10:20:45

6,104,202,2,25.50,2023-09-01 10:25:30

7,101,203,5,5.00,2023-09-01 10:30:00

8,105,204,1,40.00,2023-09-01 10:35:15

### Expected Output:

"Data loaded successfully."

**Test Case 2:** Non-CSV File

**Input**: "/FileStore/ecommerce\_transactions.xls", """

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| product\_id | store\_id | price | promotion | day\_of\_week | past\_sales | demand | date |

1,101,201,2,15.99,2023-09-01 10:00:01

2,102,202,1,25.50,2023-09-01 10:05:03

3,101,203,4,5.00,2023-09-01 10:10:15

4,103,204,1,40.00,2023-09-01 10:15:22

5,102,201,3,15.99,2023-09-01 10:20:45

6,104,202,2,25.50,2023-09-01 10:25:30

7,101,203,5,5.00,2023-09-01 10:30:00

8,105,204,1,40.00,2023-09-01 10:35:15

### Expected Output:

"Error: Only comma-separated files (.csv) are supported."

**Test Case 3:** Duplicate order\_id

**Input**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| product\_id | store\_id | price | promotion | day\_of\_week | past\_sales | demand | date |

1,101,201,2,15.99,2023-09-01 10:00:01

1,102,202,1,25.50,2023-09-01 10:05:03

3,101,203,4,5.00,2023-09-01 10:10:15

4,103,204,1,40.00,2023-09-01 10:15:22

5,102,201,3,15.99,2023-09-01 10:20:45

6,104,202,2,25.50,2023-09-01 10:25:30

7,101,203,5,5.00,2023-09-01 10:30:00

8,105,204,1,40.00,2023-09-01 10:35:15

### Expected Output:

"Error: order\_id must be unique."

**Test Case 4:** Incorrect Data Types

**Input**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| product\_id | store\_id | price | promotion | day\_of\_week | past\_sales | demand | date |

1,101,201,2,15.99,2023-09-01 10:00:01

2,102,202,one,25.50,2023-09-01 10:05:03

3,101,203,4,5.00,2023-09-01 10:10:15

4,103,204,1,40.00,2023-09-01 10:15:22

5,102,201,3,15.99,2023-09-01 10:20:45

6,104,202,2,25.50,2023-09-01 10:25:30

7,101,203,5,5.00,2023-09-01 10:30:00

8,105,204,1,40.00,2023-09-01 10:35:15

### Expected Output:

"Error: Column quantity must be of type int."

# CHAPTER 9 CONCLUSION

# 9.1 CONCLUSION:

The project successfully demonstrates how Azure Databricks, combined with modern machine learning techniques and batch-processing architectures, can significantly enhance the accuracy and scalability of product demand forecasting. By integrating tools like Apache Spark, Delta Lake, and MLflow, the system achieves efficient data handling, reliable model tracking, and reproducible results across massive datasets. Unlike traditional forecasting models that often fail under complexity or scale, this modern approach proves robust against data volume, missing values, and dynamic demand patterns. The predictive insights generated can empower businesses to optimize inventory, reduce waste, and meet customer expectations more effectively. This project paves the way for scalable and intelligent demand forecasting in data-driven industries such as retail, manufacturing, and logistics.

**Key Takeaways:**

1. Scalable Demand Forecasting Achieved:

Azure Databricks enabled the processing of large-scale sales data efficiently using distributed

computing with Apache Spark.

2. Enhanced Accuracy with Machine Learning Models:

Advanced ML algorithms like XGBoost and Random Forest outperformed traditional methods

in handling complex, non-linear demand patterns.

3. Seamless Data Pipeline Integration:

Batch processing pipelines were successfully integrated with Delta Lake for reliable, version-controlled data management.

4. Improved Experiment Tracking and Reproducibility:

MLflow provided a robust framework to manage experiments, track metrics, and deploy models consistently.

5. Business Value Through Predictive Insights

The forecasting solution has practical applications in inventory optimization, reduced stockouts, and better supply chain planning.

# CHAPTER 10

# FUTURE ENHANCEMENT

## FUTURE ENHANCEMENT

Looking ahead, this project holds immense potential for further innovation and scalability. A key enhancement would be the integration of **real-time data streams** using platforms like Apache Kafka or Azure Event Hubs, enabling instantaneous demand forecasting and faster market response. Incorporating **deep learning models** such as LSTMs or Transformers could further refine prediction accuracy by capturing intricate temporal patterns. To empower decision-makers, a **dynamic visualization dashboard** using tools like Power BI or Tableau can be added for real-time insights and reporting. Automating the model optimization process through **AutoML** or advanced hyperparameter tuning frameworks like Optuna would significantly reduce manual intervention while improving performance. The system can also be expanded to support **regional and multi-product forecasting**, offering more granular and targeted insights. Implementing **MLOps pipelines** for continuous integration, deployment, and monitoring would ensure the scalability and reliability of the solution in production environments. Additionally, enhancing **data security and compliance** with role-based access control and GDPR alignment is essential for enterprise adoption. Enriching the dataset with **external variables** such as weather, holidays, or market trends can boost forecasting accuracy. Finally, introducing **intelligent alert systems** and a structured **A/B testing framework** would help monitor anomalies, evaluate multiple models in parallel, and ensure continuous improvement in business outcomes

# REFERENCES

* + 1. Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., & Shahabi, C., Big data and its technical challenges. Communications of the ACM (2014).
    2. Dean, J., & Ghemawat, S. , MapReduce: Simplified data processing on large clusters. Communications of the ACM, (2008).
    3. Breiman, L. ,Random forests. Machine Learning, (2001).
    4. Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. Supervised machine learning: A review of classification techniques. Emerging Artificial Intelligence Applications in Computer Engineering (2007).
    5. Han, J., Kamber, M., & Pei, J. , Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann (2011).
    6. Provost, F., & Fawcett, T., Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking(2013).
    7. Min, H. , Artificial intelligence in supply chain management: Theory and applications. International Journal of Logistics: Research and Applications (2010).
    8. Manyika, J., Chui, M., Brown, B., et al. ,Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute (2011).
    9. Russom, P, Big Data Analytics. TDWI Best Practices Report. The Data Warehousing Institute (2011).
    10. Chen, M., Mao, S., & Liu, Y. , Big data: A survey. Mobile Networks and Applications (2014).
    11. Box, G.E.P., Jenkins, G.M., Reinsel, G.C., & Ljung, G.M. (2015). Time Series Analysis: Forecasting and Control, Wiley.
    12. Taylor, S.J., & Letham, B. (2018). Forecasting at Scale, The American Statistician.
    13. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., & Stoica, I. (2016). Apache Spark: A Unified Engine for Big Data Processing, Communications of the ACM.
    14. Armbrust, M., Das, T., Zhu, S., et al. (2020). Delta Lake: High-Performance ACID Table Storage Over Cloud Object Stores, VLDB Endowment.
    15. Karau, H., & Warren, R. (2017). High Performance Spark: Best Practices for Scaling and Optimizing Apache Spark.
    16. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system, Proceedings of the 22nd ACM SIGKDD Conference.
    17. Guller, M. (2015). Big Data Analytics with Spark: A Practitioner's Guide to Using Spark for Large-Scale Data Analysis.
    18. Hyndman, R.J., & Athanasopoulos, G. (2021). Forecasting: Principles and Practice.
    19. Zhang, G., Eddy Patuwo, B., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting.
    20. Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. National Institute of Standards and Technology, U.S. Department of Commerce.
    21. Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical Machine Learning Tools and Techniques.
    22. Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. International Journal of Information Management.
    23. Delen, D., & Demirkan, H. (2013). Data, information and analytics as services. Decision Support Systems.