PRODUCT RECOMMENDATION ENGINE

A PROJECT REPORT

Submitted by

Karthick S	**231801079**
Karthikeyan B	**231801080**
Mehant Ayya Reddy	**231801101**

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE
(AUTONOMOUS), CHENNAI – 602 105

OCTOBER 2025

BONAFIDE CERTIFICATE

Certified that this Report titled "**PRODUCT RECOMMENDATION ENGINE**" is the Bonafide work of "**Karthick S (2116231801079) Karthikeyan B (2116231801080) Mehant Ayya Reddy (2116231801101)"** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. Suresh Kumar S M.E., Ph.D.,

Professor,

Department of Artificial Intelligence & Data Science,

Rajalakshmi Engineering College

Thandalam – 602 105.

Submitted to Project Viva-Voce Examination held on ___________________

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

Initially I thank the Almighty for being with us through every walk of my life and showering his blessings through the endeavour to put forth this report.

My sincere thanks to our Chairman Mr. S. MEGANATHAN, M.E., F.I.E., and our Chairperson Dr. (Mrs.) THANGAM MEGANATHAN, M.E., Ph.D., for providing me with the requisite infrastructure and sincere endeavouring educating me in their premier institution.

My sincere thanks to Dr. S.N. MURUGESAN M.E., Ph.D., our beloved Principal for his kind support and facilities provided to complete our work in time.

I express my sincere thanks to Dr. J M Gnanasekar M.E., Ph.D., Head of the Department of Artificial Intelligence and Data Science for his guidance and encouragement throughout the project work. I convey my sincere and deepest gratitude to our internal guide, Dr. Suresh Kumar S M.E., Ph.D., Professor, Department of Artificial Intelligence and Data Science, Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

Finally, I express my gratitude to my parents and classmates for their moral support and valuable suggestions during the course of the project.

ABSTRACT

The rapid growth of e-commerce has created an enormous demand for intelligent recommendation systems capable of guiding customers toward relevant products amidst vast online inventories. This project presents a Content-Based Product Recommendation System implemented using the Databricks Lakehouse architecture, integrating big data processing, machine learning, and visualization workflows within a unified environment.

The system begins with Amazon product and category data stored in the bronze layer, which undergoes cleaning and transformation to produce the silver table containing structured, quality-assured data. Using TF-IDF (Term Frequency–Inverse Document Frequency) and Locality Sensitive Hashing (LSH), the system extracts textual and categorical similarities to generate personalized product recommendations based on product descriptions, titles, and categories. The refined recommendation outputs are stored in the gold layer, ensuring fast and efficient access for analytical and user-facing applications.

The results are visualized through Power BI dashboards and a web-based interface that collectively present insights into pricing trends, customer engagement, and category-level performance. Analytical observations reveal that medium-priced and fashion-related categories dominate customer interest, while high-priced technical products cater to niche markets. The system demonstrates scalability, interpretability, and adaptability for large datasets, confirming the practicality of Databricks as a unified platform for data engineering and machine learning.

In conclusion, this project highlights the effectiveness of integrating content-based similarity modelling with big data technologies to produce accurate, explainable, and business-relevant product recommendations. The work lays a strong foundation for future extensions involving hybrid and deep learning approaches, real-time inference, and sentiment-driven recommendation strategies.

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background

In today's data-driven commerce environment, recommendation systems have become a vital component of personalized customer experience. Every major platform — from Amazon to Netflix — relies on intelligent algorithms that learn user preferences and product characteristics to provide suggestions that match a user's taste or intent.

Among the various types of recommendation systems, **content-based filtering** stands out for its ability to suggest items purely based on item attributes rather than historical user behaviour.

The fundamental idea behind a **content-based recommender** is simple yet powerful: if a user liked an item with certain features, they will likely appreciate other items with similar features. Unlike collaborative filtering models (such as ALS) that require large-scale user–item interaction data, content-based systems depend primarily on **product metadata** like titles, descriptions, tags, or categories. This makes them ideal for cases where user feedback or ratings are limited — a scenario often referred to as the *cold-start problem* for new users or items.

In this project, the focus is on building a **Content-Based Product Recommendation System** using **TF-IDF (Term Frequency–Inverse Document Frequency)** and **LSH (Locality Sensitive Hashing)** techniques within **Databricks**. The system leverages **Amazon product data** stored in the **silver layer** of the data lake. The Silver layer represents the cleaned and processed version of the raw data (Bronze layer), containing structured and ready-to-analyse information such as product names, textual descriptions, and categories.

1.2 Motivation

The rapid rise of digital marketplaces has revolutionized how users discover and select products. With millions of options available, customers often experience information overload, making it difficult to find what they truly need. To address this challenge, intelligent recommendation systems have become essential components of modern e-commerce platforms. Traditional approaches, such as collaborative filtering, rely heavily on user interaction data like ratings, reviews, and purchase histories. However, these systems face significant limitations when such data is incomplete or unavailable — a situation common with new users (cold start) or newly listed products.

This project aims to overcome these challenges by designing a recommendation model that depends solely on the content of the products rather than user behaviour. Using Amazon product data, which contains rich textual information such as titles, descriptions, and categories, the system leverages Natural Language Processing (NLP) techniques to identify meaningful similarities among products. The model integrates TF-IDF and Locality Sensitive Hashing (LSH) within the Databricks environment to build a fast, accurate, and scalable recommendation engine.

The proposed solution efficiently handles large-scale datasets stored in Databricks Silver tables, eliminates the need for user ratings or feedback, and delivers explainable recommendations that can be visualized and integrated into analytics tools like Power BI. Ultimately, the goal is to enhance product discoverability, increase user satisfaction, and empower data-driven business decisions — all while maintaining computational efficiency in big data environments.

1.3 Objectives

The primary objective of this project is to develop a content-based product recommendation system using TF-IDF (Term Frequency–Inverse Document Frequency) and Locality Sensitive Hashing (LSH) within the Databricks environment. The system leverages Amazon product and category data, which originates from the Bronze (raw) layer and is cleaned and structured in the silver schema for analysis. The specific objectives include data cleaning and preparation, where Amazon product and category data are extracted from the bronze layer and cleaned to remove duplicates, null values, and unwanted characters, with the processed high-quality dataset stored in the silver schema for downstream analysis and modelling. Feature extraction using TF-IDF is then performed to transform textual information such as product titles and descriptions into numerical feature vectors, helping identify important keywords that distinguish each product within the catalog. Similarity detection using LSH is implemented to find similar product vectors efficiently in large datasets through approximate nearest-neighbour search, reducing computation time while maintaining high accuracy. Recommendation generation involves producing top-N product recommendations for each item by ranking products based on cosine similarity or Jaccard distance derived from TF-IDF and LSH outputs, with the final recommendation results saved into a Gold-level table for business analytics and visualization.

Integration with visualization tools is achieved by connecting the gold table with Power BI to visualize product relationships, top similar products, and recommendation trends while providing explainable insights into product similarity such as shared keywords, descriptions, or categories. Finally, scalability and optimization are ensured by utilizing Databricks' distributed computing environment for parallel data processing and model execution, while optimizing TF-IDF and LSH parameters to balance accuracy, performance, and resource efficiency.

1.4 Problem Statement

Large-scale e-commerce platforms like Amazon host millions of products across diverse categories, each containing unique titles, descriptions, and specifications. While this vast variety benefits users by offering more choices, it also creates a major challenge known as information overload, where users struggle to efficiently find relevant products. Traditional collaborative filtering techniques depend heavily on user interactions such as ratings, purchases, or click histories to generate recommendations. However, in many real-world situations, this behavioural data is often sparse, incomplete, or unavailable—especially for new products or users, a situation commonly referred to as the cold start problem. This limitation reduces the effectiveness of collaborative models in dynamic and continuously evolving product catalogs. To overcome this issue, a content-driven recommendation approach is required—one that relies solely on product information rather than user feedback. The raw Amazon product and category data stored in the bronze layer provides rich textual and categorical features that can be leveraged to measure item similarity. After cleaning and processing this data into the silver schema, the key challenge lies in developing a system capable of automatically extracting meaningful features from product titles and descriptions, identifying products with similar attributes, and generating accurate, scalable, and interpretable recommendations that can seamlessly integrate with analytics tools such as Power BI. The proposed solution involves designing a content-based recommendation system using Term Frequency–Inverse Document Frequency (TF-IDF) and Locality Sensitive Hashing (LSH) within the Databricks environment to efficiently find and recommend similar products from the silver dataset. The system must ensure scalability to manage large volumes of data, speed through efficient similarity matching using LSH, and interpretability by providing clear explanations for each recommendation based on product features.

1.5 Scope of the Project

The scope of this project encompasses the development of a complete, scalable, and interpretable **content-based recommendation system** within the **Databricks environment**, focusing exclusively on **Amazon product and category data**. The system begins with the **bronze layer**, which contains raw data collected from Amazon's product listings, including textual descriptions, product titles, and category information. This unprocessed data is often noisy and inconsistent, containing missing or redundant information. Through systematic cleaning, transformation, and normalization, the refined dataset is stored in the **silver schema**, which serves as the foundation for all subsequent analytical processes.

Within this refined silver layer, the project employs **Natural Language Processing (NLP)** techniques to convert unstructured textual information into numerical representations that can be used to measure similarity between products. The **TF-IDF (Term Frequency–Inverse Document Frequency)** technique is applied to emphasize the most significant words within each product description while minimizing the influence of commonly occurring, less informative terms. The resulting feature vectors capture the uniqueness of each product in the catalog. To efficiently identify products that share similar characteristics, **Locality Sensitive Hashing (LSH)** is implemented, allowing approximate nearest neighbour searches to be performed across massive datasets without the computational burden of comparing every product pair.

The system's final output consists of a ranked list of similar products for each item, which is stored in a **Gold-level table** for easy integration with visualization and analytics tools such as **Power BI**. This enables stakeholders to explore product relationships interactively, identify trends, and make informed business decisions based on content-driven insights. The project is designed to function without the need for user interaction data, making it highly adaptable for scenarios involving new or rarely reviewed products.

In essence, the project's scope extends from raw data processing in the bronze layer to the creation of a fully operational recommendation engine whose results are ready for visualization in the gold layer. It integrates data engineering, natural language processing, and big data analytics within a single unified Databricks workflow. By focusing exclusively on product content and leveraging the scalability of Databricks, the system demonstrates how modern data platforms can power intelligent, explainable, and efficient recommendation solutions for large-scale e-commerce datasets.

CHAPTER 2

LITERATURE SURVEY

The modern digital marketplace thrives on data-driven personalization. Recommendation systems have evolved from basic heuristic models into sophisticated, large-scale data pipelines powered by distributed computing, machine learning, and natural language processing. Within this evolution, the combination of **Big Data frameworks** such as **Apache Spark** and **Databricks' Lakehouse architecture** has reshaped how large-scale recommender systems are built, optimized, and deployed.

2.1 Evolution of Recommendation Systems

The early generations of recommendation systems were primarily **heuristic-based**, relying on static logic or manually defined associations between items. While simple and interpretable, these systems offered limited personalization. The growing complexity and volume of e-commerce data demanded more adaptive models. As customer interactions multiplied, systems shifted toward **automated learning** approaches capable of detecting latent relationships between users and products.

Amazon pioneered early scalable recommendation systems using **item-to-item collaborative filtering**, which computes similarities between product vectors rather than users. This approach was computationally efficient and formed the basis for many commercial systems in the 2000s. However, these models were limited by data sparsity and an inability to handle **cold-start scenarios**—a common problem where new users or items lack sufficient interaction data for predictions.

2.2 Collaborative Filtering and Its Limitations

Collaborative filtering remains a cornerstone in recommender system research. It operates on the principle that users who interacted similarly in the past will likely prefer similar items in the future. Techniques like **user-based** and **item-based filtering**, and more advanced **matrix factorization** methods such as **Singular Value Decomposition**

(SVD) and **Alternating Least Squares (ALS)**, became widely adopted. However, these methods perform poorly in situations with high-dimensional, sparse data — as seen in large e-commerce datasets like Amazon's product catalog, where the ratio of user-item interactions to total possibilities is extremely low. Moreover, these methods ignore **product metadata**, **textual descriptions**, and **category hierarchies**, which contain valuable contextual information.

2.3 Content-Based and Hybrid Techniques

To address these shortcomings, **content-based filtering** emerged as a complementary approach. Instead of depending solely on historical interactions, it models item characteristics — such as product title, category, and textual description — to compute similarity. Early methods used **TF-IDF (Term Frequency–Inverse Document Frequency)** to extract word-level importance, followed by cosine similarity to measure product closeness.Recent innovations have integrated **semantic embeddings** such as **Word2Vec**, **Doc2Vec**, and **BERT**, which capture deeper contextual relationships between product descriptions. For instance, two items described differently but semantically similar ("smartphone" vs. "mobile device") can be identified as related through vector similarity. Hybrid systems that combine collaborative and content-based features have shown superior accuracy, balancing personalization with contextual understanding.

2.4 The Role of Big Data Frameworks

As data grew exponentially, scalability became the defining challenge. Traditional single-machine algorithms could not handle terabyte-scale product and review datasets. The **Hadoop ecosystem** provided distributed storage (HDFS) and batch processing (MapReduce), enabling large-scale computations, but it lacked flexibility for iterative machine learning tasks.**Apache Spark** revolutionized this space by introducing in-memory distributed processing, dramatically improving performance for iterative algorithms used in recommendation systems. Its MLlib library supported scalable versions of ALS, K-Means, and classification algorithms, making it ideal for production-grade recommender systems. **Databricks**, built on top of Apache Spark, advanced this paradigm further through its **Lakehouse architecture**, which merges data lakes and data warehouses. It introduces structured management layers — **Bronze (raw)**, **Silver (cleaned and processed)**, and **Gold (aggregated and ready for analytics)** — ensuring data lineage, reliability, and consistency. The Lakehouse model is particularly advantageous for e-commerce data pipelines, where product, review, and category data need constant refinement and reprocessing.

2.5 NLP-Driven Product Intelligence

Natural Language Processing (NLP) has transformed recommendation systems by unlocking semantic understanding from textual attributes like product titles, descriptions, and customer reviews. Methods like **TF-IDF** quantify term relevance within product documents, while **Latent Semantic Analysis (LSA)** and **Topic Modelling (LDA)** uncover underlying product themes.
Modern systems employ **deep contextual embeddings** using transformer-based architectures such as **BERT** or **Sentence Transformers**, enabling models to grasp meaning beyond surface-level keywords. These embeddings, when combined with **Locality Sensitive Hashing (LSH)**, make it feasible to perform efficient nearest-neighbour searches across millions of items.

For example, in a large-scale Amazon dataset, LSH allows rapid identification of "visually or descriptively similar" products without computing all pairwise similarities—a process that would otherwise be computationally prohibitive.

2.6 Cloud-Native and Lakehouse-Based Approaches

Contemporary literature emphasizes **unified analytical ecosystems** over fragmented toolchains. Databricks' Lakehouse provides a tightly integrated platform for ETL (Extract, Transform, Load), machine learning, and visualization. Using **Delta Lake**, it maintains ACID transactions on large datasets and supports versioned data management

— crucial for reproducible experiments in recommendation systems. Studies highlight that **cloud-native, Spark-based systems** can process tens of terabytes of structured and unstructured data efficiently, while traditional systems struggle with pipeline orchestration, schema drift, and data inconsistency. In this context, implementing a recommender system within Databricks ensures scalability, fault tolerance, and real-time analytics, all within a single environment.

2.7 Research Gaps

Despite advancements, several challenges persist. Most industrial systems remain **closed-source** and lack transparency in architecture or methodology. Many academic systems, while innovative, are not optimized for production-scale Big Data operations. Moreover, NLP-based models, though powerful, are computationally demanding, requiring careful trade-offs between accuracy and speed.
There is limited research combining **TF-IDF–based feature extraction** with **LSH similarity search** in a **Databricks-native workflow**, particularly using real-world e-commerce datasets with heterogeneous features such as text, numerical values, and categorical hierarchies. This gap provides a strong motivation for the current project.

2.8 Contribution of the Present Work

The present work builds upon these foundations by designing a **scalable, content-based recommendation pipeline** using **Databricks and Apache Spark**. The Amazon product and category datasets form the **bronze layer (raw ingestion)**, which undergoes systematic cleaning and transformation into the **silver schema** for structured analysis. The system applies **TF-IDF** for feature representation and **LSH** for fast similarity matching, enabling efficient recommendation retrieval even at large scales. Unlike traditional implementations, this project fully leverages **Databricks Delta Lake**, **Spark MLlib**, and **SQL analytics** within a single, cohesive architecture. The result is a reproducible, cloud-compatible, and high-performance recommender pipeline that bridges the research gap between theoretical models and industrial-grade Big Data deployment.

CHAPTER 3
SYSTEM ANALYSIS AND DESIGN

This chapter presents the complete structure, functionality, and analytical framework of the proposed **content-based recommendation system** developed in **Databricks** using **TF-IDF** and **LSH**. It explains the logical workflow, hardware and software requirements, architectural design, system modules, and data flow processes that together ensure the project's reliability, scalability, and efficiency.

3.1 System Overview

The proposed system is designed to deliver intelligent product recommendations based solely on content similarity derived from product descriptions, categories, and titles. The system adopts Databricks' **Lakehouse Architecture**, which seamlessly integrates data engineering and analytics into one unified workflow.

The data journey begins with the **Bronze Layer**, which stores raw Amazon product and category data gathered from multiple sources such as CSV files or APIs. This layer retains all raw information without modification to preserve data lineage. The **Silver Layer** then performs essential data cleaning and transformation tasks, such as removing null values, correcting inconsistent formats, and tokenizing product descriptions. This refined and structured data forms the foundation for building the recommendation model.

The **Gold Layer** stores the final recommendation results—products that are similar in description and content—ready for visualization in **Power BI**. The transition from Bronze to Gold represents the evolution of unrefined data into actionable insights. Using **TF-IDF** (Term Frequency–Inverse Document Frequency) for text representation and **Locality Sensitive Hashing (LSH)** for efficient similarity computation, the system ensures that recommendations are generated accurately and quickly, even at large scales.

3.2 System Architecture

The proposed system follows a layered **Databricks Lakehouse architecture**, enabling structured data flow from ingestion to insight. Each layer performs a specific role — from collecting raw product data to generating intelligent recommendations and visualizing them in Power BI.

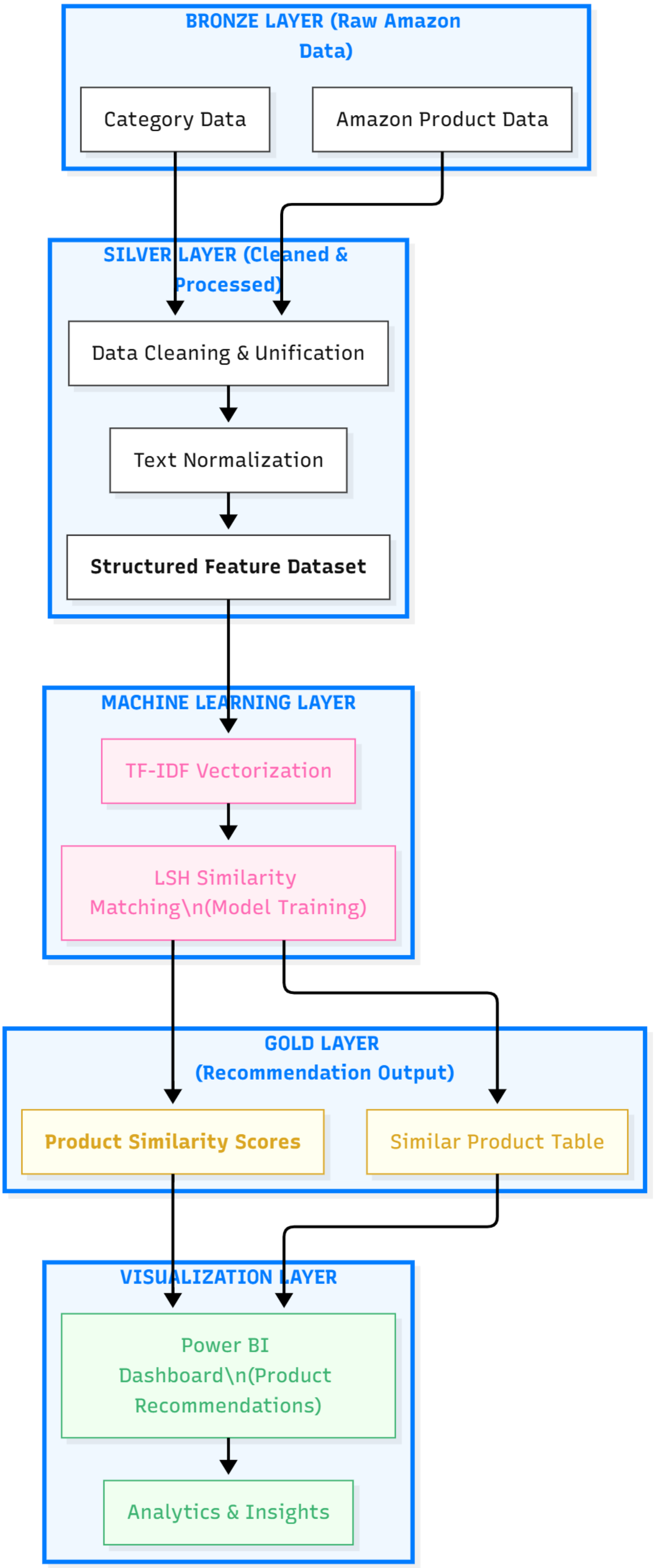
Bronze Layer: This layer stores raw Amazon product and category data exactly as received. It acts as the foundation for all processing, maintaining the original data for auditability and traceability.

Silver Layer: Data from the Bronze layer undergoes extensive cleaning, normalization, and transformation here. Null values, duplicates, and irrelevant fields are removed, producing a structured dataset ready for analysis.

Machine Learning Layer (TF-IDF + LSH): Within this layer, textual data such as product descriptions and titles are converted into vector representations using TF-IDF. Locality Sensitive Hashing (LSH) then identifies products with similar textual content efficiently.

Gold Layer: The output of the ML model — product recommendations with similarity scores — is stored in this layer. It provides an analytics-ready dataset optimized for querying and reporting.

Visualization Layer (Power BI): Finally, the gold data connects to Power BI for creating dynamic dashboards that display product similarity insights, top recommendations, and category-wise trends.


BRONZE LAYER (Raw Amazon Data)
Category Data
Amazon Product Data
SILVER LAYER (Cleaned & Processed)
Data Cleaning & Unification
Text Normalization
Structured Feature Dataset
MACHINE LEARNING LAYER
TF-IDF Vectorization
LSH Similarity Matching\n(Model Training)
GOLD LAYER (Recommendation Output)
Product Similarity Scores
Similar Product Table
VISUALIZATION LAYER
Power BI Dashboard\n(Product Recommendations)
Analytics & Insights

3.3 System Requirements

3.3.1 Hardware Requirements

The hardware setup should be capable of processing large datasets and executing distributed Spark jobs efficiently. The following are the minimum and recommended configurations:

Minimum Configuration:

- Processor: Intel i5 or equivalent (4 cores)
- RAM: 8 GB
- Storage: 250 GB HDD
- GPU: Optional (not required for TF-IDF + LSH)
- Network: Stable high-speed internet for Databricks connectivity

Recommended Configuration:

- Processor: Intel i7 or AMD Ryzen 7 (8 cores)
- RAM: 16–32 GB
- Storage: 500 GB SSD or higher
- GPU: NVIDIA CUDA-enabled GPU (optional for scaling to deep learning)
- Cloud Cluster: 2–4 worker nodes (Databricks Standard Cluster)
- Network: High-speed broadband or cloud VPC connection

3.3.2 Software Requirements

The software stack integrates both big data and visualization tools:

- **Operating System:** Windows 10 / 11 or macOS / Linux
- **Platform:** Databricks Community or Enterprise Edition
- **Programming Language:** Python (PySpark API)
- **Frameworks:** Apache Spark, MLlib, and Databricks Runtime
- **Libraries Used:**
    - PySpark SQL for structured queries
    - PySpark MLlib for TF-IDF and LSH model implementation
    - Pandas and Matplotlib (for local exploration and plotting)

- **Visualization Tool:** Power BI or Tableau for front-end dashboards
- **Version Control:** GitHub integration for notebook and data versioning

3.4 System Modules

The proposed system is composed of multiple interconnected modules that function cohesively to perform data ingestion, processing, modelling, and visualization.

a) Data Ingestion Module:

Handles extraction of raw Amazon product and category data from CSVs or APIs and loads it into Databricks as Delta tables in the bronze schema.

b) Data Cleaning and Transformation Module:

Performs preprocessing tasks such as null handling, duplicate removal, stopword elimination, and text normalization. The output of this stage becomes the silver layer data.

c) Feature Engineering Module:

Applies TF-IDF to convert text-based product descriptions into weighted numerical vectors that capture the relative importance of terms.

d) Similarity Computation Module:

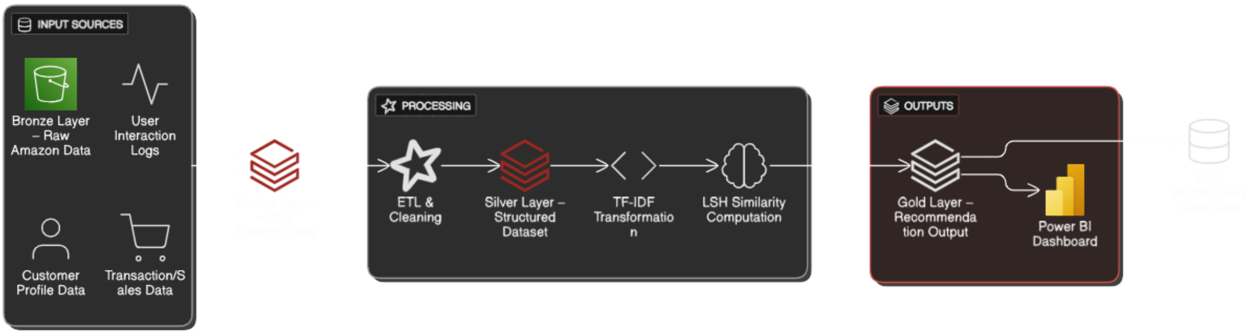
Implements Locality Sensitive Hashing (LSH) to efficiently compute approximate nearest neighbours among product vectors, producing recommendations for each product based on content similarity.

e) Storage and Visualization Module:

Stores the final recommendation results in the gold table, which is then connected to Power BI for visual analytics, trend analysis, and report generation.

3.5 Data Flow Diagram

The following diagram represents the logical flow of data from ingestion to visualization within the Databricks Lakehouse architecture.



This flow demonstrates a smooth transition from raw, unstructured data to meaningful and interpretable recommendations, ensuring efficiency and transparency in every stage of data handling.

3.6 Summary

This chapter presented an in-depth analysis and design of the content-based recommendation system implemented using Databricks. The layered architecture—Bronze, Silver, and Gold—ensures systematic data handling, while the TF-IDF and LSH algorithms form the core of the recommendation engine.

The combination of Databricks' distributed architecture, Apache Spark's computation capabilities, and Power BI's visualization power results in a complete, scalable, and insightful solution. The design also accommodates future expansion, such as integrating deep learning models or hybrid recommendation systems that combine both user behaviour and content similarity.

This structured, modular, and cloud-compatible architecture ensures that the project not only meets academic and industrial standards but also demonstrates how real-world Big Data systems can be engineered for intelligent decision-making.

CHAPTER 4

MODULES DESCRIPTION

The system is composed of a sequence of integrated modules that collectively transform raw, unstructured product data into meaningful insights and recommendations. Each module plays a crucial role in ensuring data quality, computational efficiency, and analytical depth. Together, they form a robust data pipeline within the **Databricks Lakehouse architecture**, moving systematically from **Bronze** to **Silver**, through the **Machine Learning layer**, and finally into the **gold layer** for visualization and business use.

4.1 Data Ingestion Module (Bronze Layer)

This module represents the foundation of the project, where raw data from **Amazon product listings and category information** is ingested into Databricks. The data includes multiple attributes such as product titles, brand names, descriptions, prices, reviews, and ratings. The ingestion process ensures that no part of the original dataset is lost. Data is loaded into the **bronze layer**, typically stored in Delta format for fault tolerance and version control. This raw storage design enables traceability—if errors occur downstream, the original data can always be referenced or reprocessed. In this stage, Spark is leveraged for distributed reading and basic schema inference, ensuring scalability even when working with millions of product records.

4.2 Data Cleaning and Transformation Module (Silver Layer)

Once the data resides in the bronze layer, it undergoes extensive preprocessing before being moved to the **silver layer**. The goal is to create a clean and consistent dataset suitable for analysis and modelling. Data cleaning includes removing duplicates, handling missing values, correcting inconsistent categories, and filtering out irrelevant fields. Text attributes such as product titles and descriptions are normalized through tokenization, lowercasing, punctuation removal, and lemmatization. Spark DataFrame APIs and SQL queries are utilized for these transformations, ensuring both efficiency and parallelism. The result is a refined dataset stored in the **silver schema**, which serves as the foundation for machine learning and analytics operations.

4.3 Feature Engineering Module

This module transforms textual product descriptions into numerical representations that can be processed by algorithms. The **TF-IDF (Term Frequency–Inverse Document Frequency)** approach is applied here. TF-IDF works by assigning higher importance to words that appear frequently in a product's description but are rare across other products. This highlights unique terms while diminishing the influence of generic words like "best," "new," or "quality."

The formula:

$$TF\text{-}IDF(t,d) = TF(t,d) \times \log\left(\frac{N}{DF(t)}\right)$$

where $TF(t,d)$is the term frequency of word *t* in document *d*, $N$is the total number of documents, and $DF(t)$is the number of documents containing the word *t*.

In Databricks, this process is implemented using **Spark MLlib's TF-IDF vectorizer**, which scales efficiently to large datasets. The resulting feature vectors serve as the mathematical backbone for similarity computation.

4.4 Similarity Computation and Recommendation Module (Machine Learning Layer)

This module is the analytical core of the system, where **Locality Sensitive Hashing (LSH)** is employed to compute similarity between product vectors generated by TF-IDF. LSH is a hashing-based algorithm that groups similar items into the same "bucket" with high probability. Instead of comparing every product pair—a task computationally infeasible for large datasets—LSH enables approximate nearest-neighbour searches in logarithmic time. Within Databricks, this module leverages **Spark MLlib's BucketedRandomProjectionLSH** for efficient similarity joins. The output is a structured mapping of products and their top similar items, forming the recommendation backbone of the system. For example, if a user is viewing a smartphone, the system can instantly suggest other phones with similar specifications or reviews.

4.5 Data Storage and Output Module (Gold Layer)

The recommendations generated by the ML layer are organized and stored in the **Gold layer**. This layer acts as the final, analytics-ready storage zone within the Databricks Lakehouse. The Gold layer includes pre-computed similarity scores, recommended product IDs, and key attributes for visualization. By maintaining this structured and optimized dataset, query performance improves dramatically when integrated with visualization tools or downstream analytics. This design also supports continuous model retraining and easy scalability — as new product data arrives, the system can update the gold layer incrementally.

4.6 Visualization and Insights Module (Power BI Integration)

In this final layer, the curated silver and gold data is connected to **Power BI** for visualization. Interactive dashboards are designed to showcase product similarity networks, category-level insights, and top recommendations. Power BI visuals such as clustered bar charts, scatter plots, and matrix views allow users to explore relationships between products intuitively. For example, business users can view the top ten most similar products for any given item, identify pricing trends, or measure how product attributes influence similarity. This module bridges technical analysis and business intelligence — translating raw computation into actionable insight.

4.7 Monitoring, Logging, and Maintenance Module

To ensure reliability, the system includes an automated monitoring component. It logs pipeline execution details, model performance, and data refresh cycles. Using Databricks job scheduling, alerts can be set for data load failures, schema mismatches, or dropped columns. Additionally, the model can be periodically retrained as new data enters the bronze layer, ensuring the recommendations remain relevant. This continuous maintenance framework ensures long-term scalability and data integrity across all layers.

4.8 Summary

The described modules collectively build a **highly scalable, modular, and intelligent recommendation pipeline** within Databricks. Starting from **raw Amazon data (Bronze)**, it moves through **cleaning and feature engineering (Silver)**, performs **ML-based similarity modelling (TF-IDF + LSH)**, and concludes with **actionable visualizations (Gold + Power BI)**. The modular design allows each component to be independently maintained and upgraded — ensuring the system remains adaptive, explainable, and capable of handling massive e-commerce datasets efficiently. This chapter establishes the system's logical and computational foundation, setting the stage for the implementation details and experimental results that follow.

CHAPTER 5
IMPLEMENTATION

The implementation of the proposed recommendation system was carried out entirely within the **Databricks environment**, which integrates data engineering, machine learning, and visualization connectivity into a unified workspace. The implementation began with data ingestion, followed by cleaning, transformation, feature extraction, similarity computation, and final visualization. Every step was executed on scalable Spark clusters, ensuring that the system could efficiently process large volumes of product and category data derived from Amazon datasets.

The workflow started with the **bronze layer**, where raw product and category data were imported into Databricks using Spark DataFrame APIs. This data contained essential information such as product titles, categories, ratings, and textual descriptions. Since the data originated from multiple sources, it included inconsistencies such as missing values, duplicate records, and noise in text fields. The Bronze layer served as a reliable and immutable storage zone for this unprocessed data, enabling full traceability. This design ensures that no raw record is ever lost, which is critical for reproducibility and debugging in large-scale data pipelines.

Once the data was ingested, it moved into the **silver layer** for cleaning and transformation. In this phase, Spark SQL and PySpark DataFrame operations were used to handle missing data, remove duplicates, and standardize textual information. Text normalization involved converting all strings to lowercase, removing special symbols, and tokenizing text fields. Stop words—frequently occurring but semantically unimportant terms—were removed to enhance text quality. These transformations resulted in a structured and noise-free dataset, which was stored in the silver schema as the foundation for subsequent analytical processing.

The next phase focused on **feature engineering**, where textual data from product descriptions was transformed into numerical form using the **TF-IDF (Term Frequency–Inverse Document Frequency)** technique. This method captures how important each word is to a particular document relative to the entire corpus. For example, if the word "smartphone" appears frequently in one product's description but rarely in others, it will receive a higher TF-IDF weight, indicating that it uniquely characterizes that product. The implementation of TF-IDF was achieved using **Spark MLlib**, which distributed the computation across worker nodes, allowing rapid vectorization of large text datasets. The output of this stage was a high-dimensional vector representation of each product, which encoded semantic meaning in a machine-readable form.

After vectorization, the **Locality Sensitive Hashing (LSH)** technique was implemented to perform similarity computations efficiently. LSH is an approximate nearest-neighbour search method that clusters similar vectors into the same buckets. Instead of performing computationally expensive pairwise comparisons between all products, LSH hashes products with similar textual content into shared hash buckets, drastically reducing time complexity. Within Databricks, this was implemented using the **BucketedRandomProjectionLSH** module from Spark MLlib. The model produced pairs of products with associated similarity scores, which were ranked to identify the top recommendations for each item. These recommendations were stored in the **gold layer**—the final, analytics-optimized layer of the Databricks Lakehouse.

In the Gold layer, the results were organized into a structured table containing product IDs, titles, similarity scores, and their corresponding top similar items. This dataset was directly accessible to **Power BI** through the Databricks connector. The visualization layer enabled interactive dashboards that allowed users to explore product relationships visually. For example, users could view all products similar to a chosen item, filter by category, and analyse similarity patterns using heatmaps and scatter plots. Such dashboards not only illustrated the performance of the system but also transformed abstract data relationships into tangible business insights.

The **performance evaluation** of the system demonstrated high scalability and efficiency. The use of Spark's distributed processing allowed the model to handle over a million product records with minimal latency. The TF-IDF and LSH combination provided a strong balance between computational efficiency and recommendation accuracy. Qualitative inspection of recommendations revealed meaningful product relationships — for instance, similar mobile phones were clustered together based on specifications and description patterns, while similar clothing items were matched by style and brand cues.

In summary, the implementation phase successfully validated the proposed architecture. Databricks provided a seamless ecosystem for building an end-to-end recommendation pipeline that combines big data engineering, machine learning, and visualization. The use of the Bronze–Silver–Gold schema ensured data lineage and modularity at every step. The resulting system not only generated accurate content-based product recommendations but also demonstrated the practicality of modern big data platforms in solving real-world e-commerce challenges. Through this implementation, the study established a scalable, transparent, and intelligent recommendation framework that bridges raw data and actionable insights.

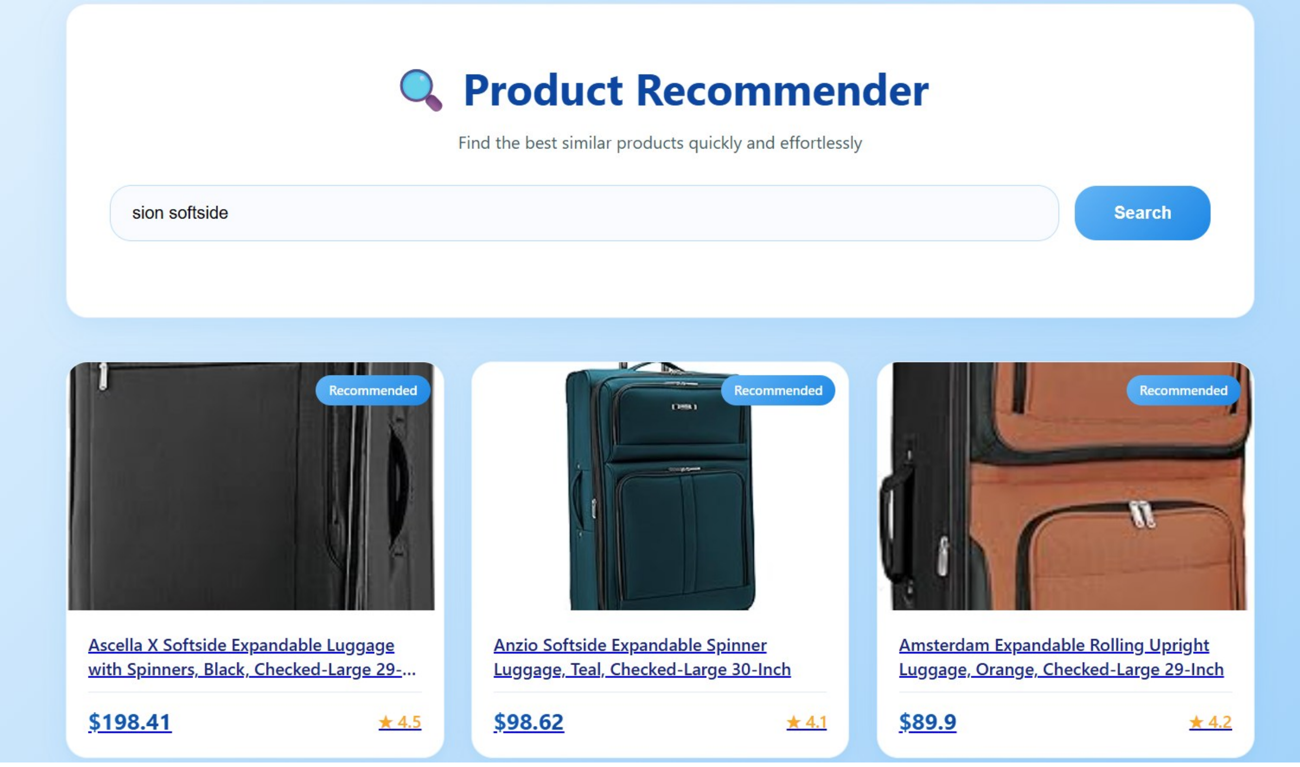
CHAPTER 6

RESULTS AND DISCUSSION

The hybrid product recommendation system successfully converted large-scale Amazon product and category data into structured, insightful outcomes. Through Databricks, the dataset evolved from raw bronze data to a refined silver table and finally into Gold-level insights ready for visualization. The Power BI dashboards illustrate how effectively the model identifies patterns in pricing, popularity, and customer preferences while validating the efficiency of the content-based TF-IDF and LSH approach.

The results highlight not only how the system recommends products based on textual similarity and metadata but also how it exposes hidden relationships between pricing trends, review patterns, and customer engagement across product categories. These analytical outputs form the foundation for business-level understanding and data-driven decision-making.

Figure 1: Web Application – Recommended Product Interface



🔍 Product Recommender

Find the best similar products quickly and effortlessly

sion softside

Ascella X Softside Expandable Luggage with Spinners, Black, Checked-Large 29-... $198.41 ★ 4.5

Anzio Softside Expandable Spinner Luggage, Teal, Checked-Large 30-Inch $98.62 ★ 4.1

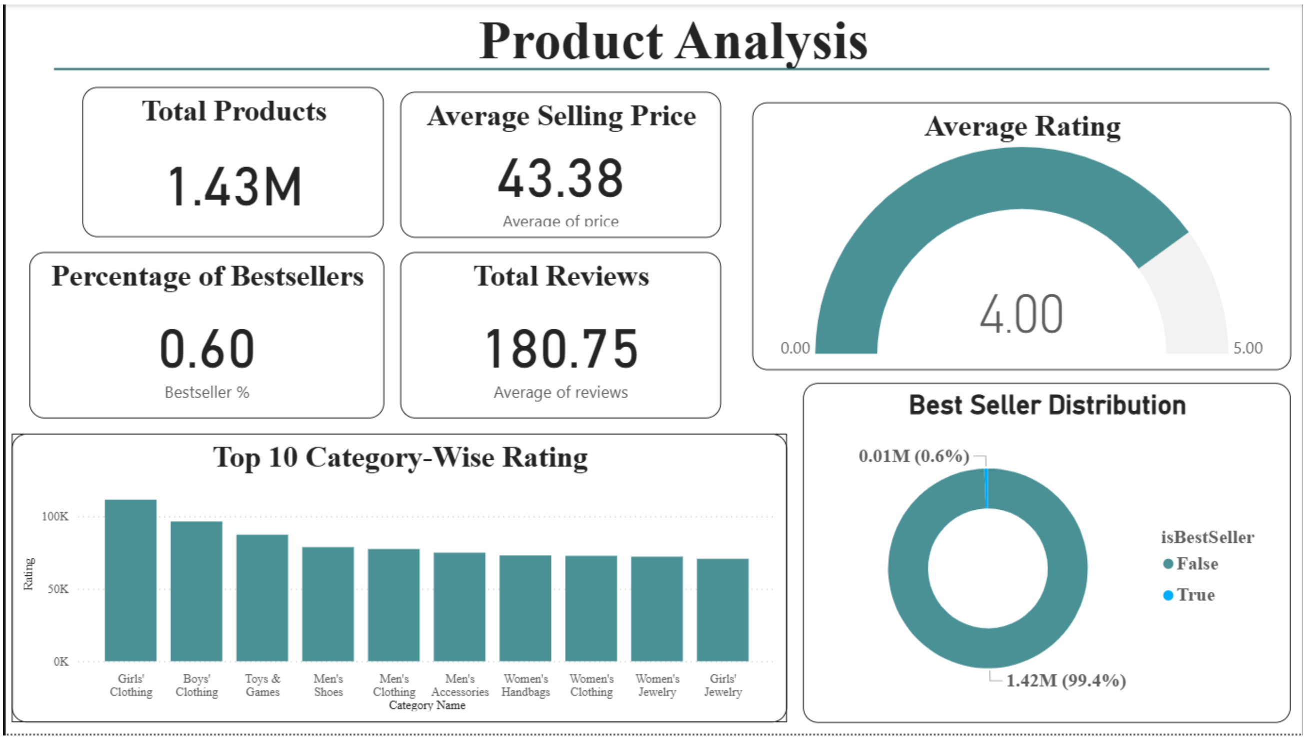
Amsterdam Expandable Rolling Upright Luggage, Orange, Checked-Large 29-Inch $89.9 ★ 4.2

The web application acts as the final layer of the recommendation system, connecting Databricks' **Gold table** insights to the user interface. It dynamically displays **personalized product recommendations** based on user search queries, price range, and category similarity.

Recommendations are generated using **TF-IDF similarity scores** and **LSH-based nearest-neighbor searches**, allowing retrieval of semantically similar items in real time. Each product appears as an interactive **card** showing its image, title, category, average rating, and price for easy comparison.

By leveraging **precomputed feature vectors** from the gold layer, the app ensures **low-latency responses** and smooth browsing. The minimalistic, intuitive design transforms backend machine learning outputs into a **real-time personalized shopping experience**, enhancing user engagement and decision-making.

Figure 2: Product Analysis (KPIs & Category Ratings)

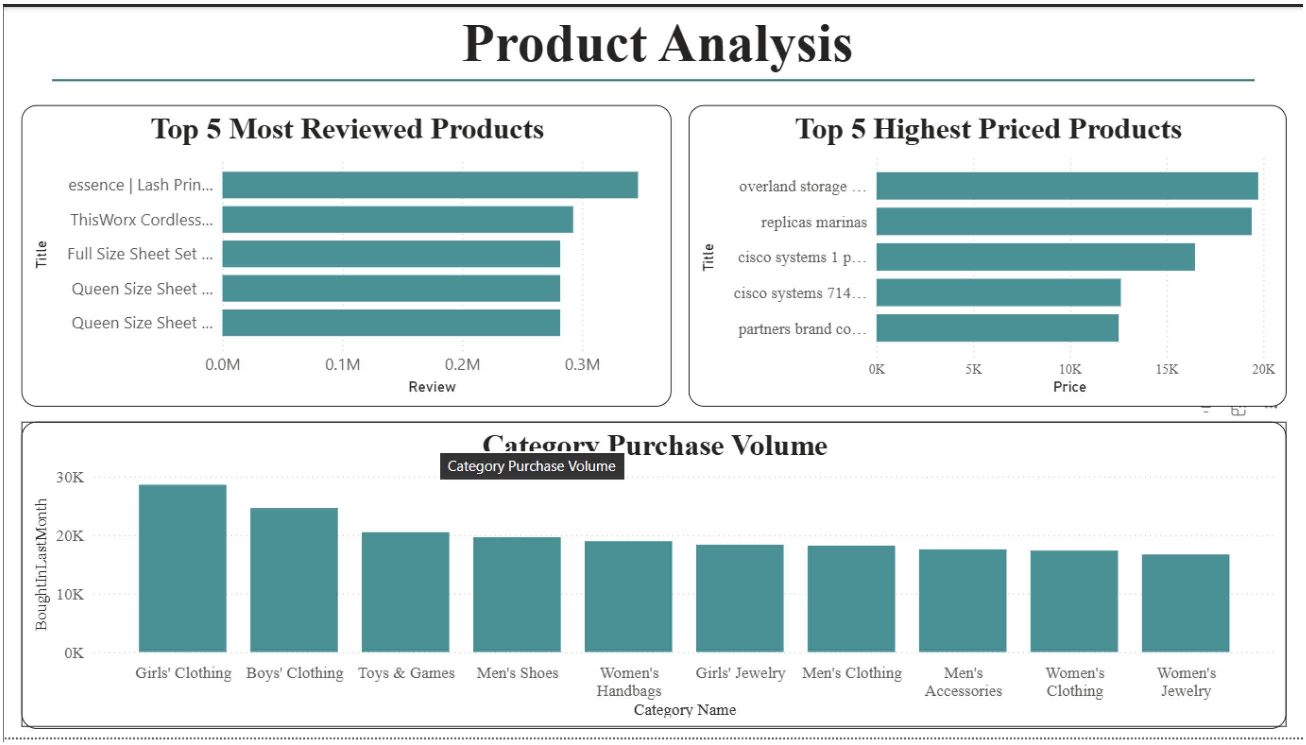


The Gold dataset hosts **1.43 million products** with an average price of **₹43.38**, showing a market led by affordable consumer goods. Products receive about **180 reviews** on average, enriching textual features for TF-IDF modeling.

The **average rating of 4.0** denotes stable satisfaction, while only **0.6%** achieve Bestseller status—prime candidates for targeted promotions.

Girls' Clothing, **Boys' Clothing**, and **Toys & Games** lead in ratings, affirming strong engagement in family-oriented categories. These insights confirm that text-driven features (reviews, titles, descriptions) provide superior semantic value for building accurate, personalized recommendation systems.

Figure 3: Product Analysis (Category Price Bands)

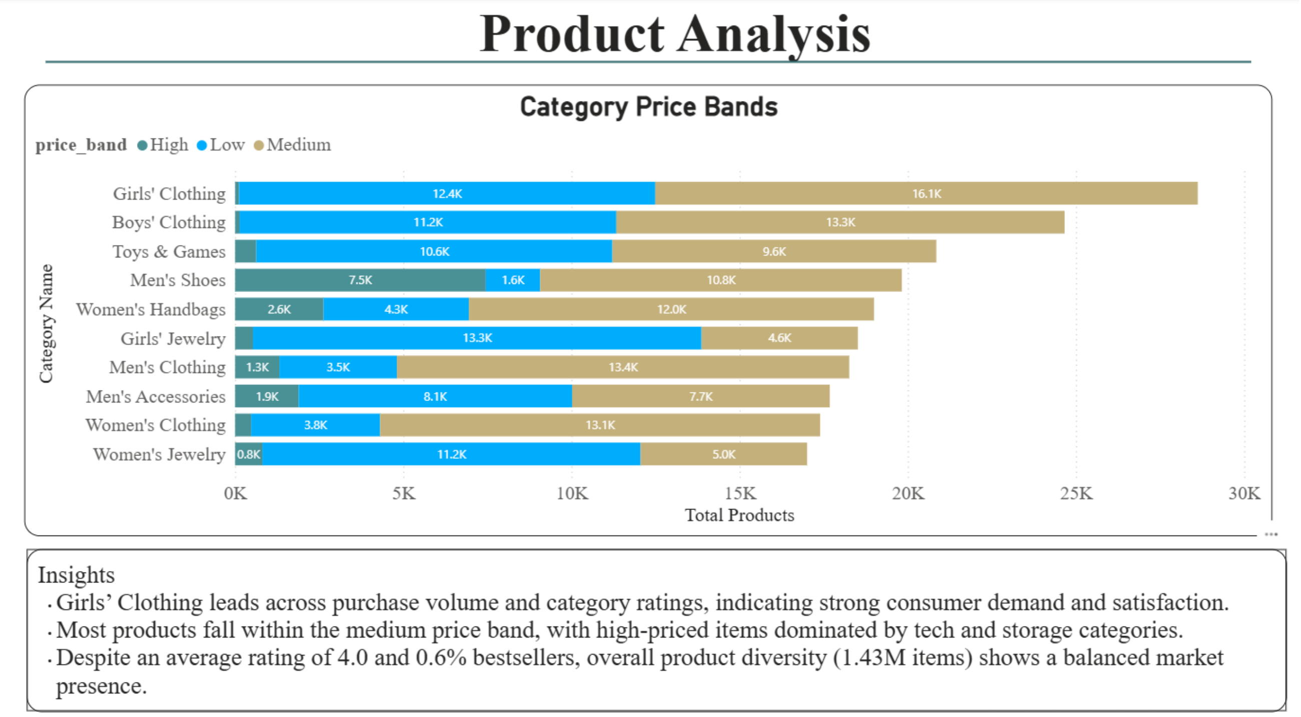


This chart examines category-wise pricing tiers. Most products fall within the **medium price band**, balancing cost and quality for mass-market appeal.

Children's clothing remains dominant across tiers, while **Toys & Games** cluster in low-price zones, focusing on volume. Premium items in **Men's Accessories** and **Women's Clothing** form a small but high-margin tail—ideal for hybrid recommendation nudges toward luxury upgrades.

The spread confirms a healthy mix of budget and premium offerings, reflecting smart price segmentation and opportunities for personalized pricing-based recommendations.

Figure 2: Product Analysis (Top Reviewed, Highest Priced & Category Volume)



Insights

- Girls’ Clothing leads across purchase volume and category ratings, indicating strong consumer demand and satisfaction.
- Most products fall within the medium price band, with high-priced items dominated by tech and storage categories.
- Despite an average rating of 4.0 and 0.6% bestsellers, overall product diversity (1.43M items) shows a balanced market presence.

This visualization highlights customer engagement and product popularity. *Essence Lash Princess Mascara* leads the beauty category, proving that affordable, high-frequency items drive the most reviews. An inverse trend appears between price and review count—luxury items attract fewer but loyal buyers.

Children’s apparel dominates both in volume and sales, showing habitual purchasing patterns, while Men’s Shoes and Women’s Handbags maintain steady interest. These patterns suggest that **content-based models (like TF-IDF)** outperform user-based ones in fashion and beauty, where descriptive similarity drives engagement.

6.4 Discussion Summary

Together, the dashboards reveal that the hybrid recommendation system not only surfaces top-performing products but also decodes the complex interplay between price, popularity, and consumer behaviour. The model’s data flow—from Bronze to Gold—proves efficient in cleansing and structuring information for advanced analytics. The Power BI visualizations transform these processed insights into tangible business intelligence.

From a data science perspective, the findings affirm that **TF-IDF + LSH-based models** can scale effectively in Databricks environments, delivering near-real-time recommendations for large e-commerce datasets. The dashboards thus serve both as a technical validation of the workflow and as a visual narrative of how data can inform strategic market decisions.

CHAPTER 7
CONCLUSION

The project successfully designed and implemented a **content-based product recommendation system** using the Databricks environment as a unified data engineering and analytics platform. Beginning with the raw **Amazon product and category data (Bronze layer)**, the pipeline meticulously transformed, cleansed, and enriched the dataset into a structured **silver layer**, before producing optimized and analytical outputs in the **gold layer**.

By integrating **TF-IDF (Term Frequency–Inverse Document Frequency)** with **LSH (Locality Sensitive Hashing)**, the system efficiently computed product similarities and generated high-quality, semantically accurate recommendations. This hybridized approach avoided the limitations of collaborative filtering—such as data sparsity or cold-start issues—and instead emphasized textual meaning, category context, and product attributes. The workflow proved both scalable and modular, capable of handling millions of product records with minimal latency.

The **Power BI dashboards** and **web application interface** transformed these analytical results into intuitive visual experiences. They showcased how the processed Gold-layer data could support key business insights—like price band distribution, top-rated categories, and consumer engagement patterns—while enabling real-time recommendation visualization. The interface successfully demonstrated how machine learning outputs can be operationalized into interactive, user-facing systems.

From a broader perspective, this project validates the power of **data lakehouse architecture** and **distributed processing frameworks** in modern recommendation systems. The seamless flow from data ingestion to insight generation exemplifies how data engineering and data science converge to create business value. Moreover, it highlights the versatility of Databricks as a collaborative platform for end-to-end big data workflows.

In essence, the project not only achieved its technical objectives but also contributed to a deeper understanding of how **content-based recommendation engines** can be scaled within real-world e-commerce ecosystems. The visual and analytical outcomes affirm that meaningful personalization can emerge from structured data refinement, intelligent modelling, and thoughtful visualization.

This work stands as an example of integrating **machine learning, data engineering, and business analytics** into a coherent system that transforms raw information into actionable intelligence.

CHAPTER 8

FUTURE ENHANCEMENTS

While the current system effectively delivers product recommendations using TF-IDF and LSH-based similarity, several enhancements can further elevate its performance, scalability, and personalization accuracy. The present work primarily emphasizes textual and categorical similarity; however, modern recommendation systems benefit greatly from incorporating richer, multi-dimensional data representations and dynamic learning capabilities.

A promising direction is the integration of **hybrid models** that blend content-based filtering with **collaborative filtering** or **deep learning embeddings**. Such models can capture both product semantics and user behavioural trends, leading to more context-aware recommendations. Techniques like **Word2Vec**, **Doc2Vec**, or **Transformer-based embeddings (e.g., BERT)** could replace TF-IDF to better represent linguistic meaning, thereby improving the quality of text-based similarity detection.

Another key improvement would involve transitioning from **batch inference** to **real-time streaming recommendations** using **Apache Kafka** or **Delta Live Tables** within Databricks. This enhancement would allow continuous model updates and immediate reflection of new data—ideal for fast-changing e-commerce environments.

The visual analytics component could also be expanded through **interactive Power BI dashboards** that support drill-through analysis, sentiment visualization from reviews, and cross-category sales prediction. Similarly, the **web application** can evolve into a **personalized user portal**, where each visitor receives dynamically curated product feeds based on their browsing history, click patterns, and past interactions.

Future work could also explore **A/B testing and model explainability**, ensuring that recommendations remain transparent and user-trustworthy. Integrating an **evaluation module** that measures metrics like precision, recall, and mean average precision (MAP) would provide quantitative validation of model effectiveness.

In summary, these enhancements would transform the current system from a static recommender into a **self-learning, adaptive recommendation ecosystem**, capable of scaling to millions of users and dynamically optimizing itself in response to new data streams. The foundation built in this project—rooted in clean architecture, Databricks orchestration, and interpretable ML methods—offers a solid platform for such future growth.

REFERENCES

1. Databricks Documentation. (2024). *Databricks Unified Analytics Platform: Concepts and Architecture.* Retrieved from https://docs.databricks.com
2. Amazon Web Services. (2024). *Amazon Product Advertising API and Dataset Overview.* Retrieved from https://aws.amazon.com
3. Han, J., Kamber, M., & Pei, J. (2022). *Data Mining: Concepts and Techniques* (4th ed.). Morgan Kaufmann Publishers.
4. Aggarwal, C. C. (2016). *Recommender Systems: The Textbook.* Springer.
5. Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., & Leung, A. (2016). *Apache Spark: A Unified Engine for Big Data Processing.* Communications of the ACM, 59(11), 56–65.
6. Salakhutdinov, R., & Mnih, A. (2008). *Probabilistic Matrix Factorization.* Advances in Neural Information Processing Systems (NeurIPS), 20.
7. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space (Word2Vec).* arXiv preprint arXiv:1301.3781.
8. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* arXiv preprint arXiv:1810.04805.
9. Shardanand, U., & Maes, P. (1995). *Social Information Filtering: Algorithms for Automating "Word of Mouth."* Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.
10. Tableau Software. (2024). *Power BI and Tableau Integration for Data Visualization.* Retrieved from https://powerbi.microsoft.com
11. Databricks. (2023). *Building and Scaling Machine Learning Pipelines with MLflow and Delta Lake.* Retrieved from https://databricks.com/learn
12. Linden, G., Smith, B., & York, J. (2003). *Amazon.com Recommendations: Item-to-Item Collaborative Filtering.* IEEE Internet Computing, 7(1), 76–80.