INDUSTRY PROJECT

ON

CUSTOMER SEGMENTATION AND RECOMMENDATION

TCS iON AIP 135

Name: Madeeha Khanum

Campus ID: 30853

Register Number: 23BBCBDB12

College Name: Yenepoya (Deemed to Be University)-Bengaluru

Industry Project Title: TCS iON AIP 135

Project Title: Customer Segmentation and Recommendation System

Name of the Company: Tata Consultancy Services

Name of the Institute: Yenepoya (Deemed to Be University)

Start Date: 13-11-2025

End Date:11-02-2026

Project Environment: Python, Jupyter Notebook, Power BI

Tools Used: Python, Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn, Power BI

TABLE OF CONTENTS

TABLE OF FIGURES

ABSTRACT

In the modern retail landscape, understanding consumer behavior is critical for maintaining a competitive edge and ensuring customer retention. This project focuses on the development and implementation of a **Customer Segmentation and Recommendation System** using advanced machine learning techniques.

The methodology follows a structured data science pipeline, beginning with extensive **Data Pre-processing** on a transactional dataset to handle missing values and normalize numerical features using RobustScaler. To transition from raw transactions to customer-centric insights, **Feature Engineering** was performed to derive key behavioral metrics, including purchase frequency and temporal patterns.

For the segmentation phase, the **K-Means Clustering** algorithm was employed. The model's performance was rigorously evaluated using the **Silhouette Score**, **Davies-Bouldin Index**, and **Calinski-Harabasz Index** to ensure the formation of distinct and statistically significant customer groups. Building upon these segments, a **Recommendation System** utilizing **Cosine Similarity** was developed to suggest best-selling products to customers within the same cluster who have not previously purchased those items.

Finally, the results were integrated into an interactive **Power BI/Tableau Dashboard**, providing stakeholders with real-time monitoring capabilities and actionable insights into segment trends. The project demonstrates how the synergy between unsupervised learning and data visualization can drive personalized marketing strategies and boost overall sales effectiveness.

CHAPTER 1: INTRODUCTION

1.1 Project Overview In the contemporary retail environment, businesses are inundated with vast quantities of transactional data generated through point-of-sale systems, e-commerce platforms, and mobile applications. However, the sheer volume of this data often masks the underlying behavioral patterns of the consumer base. The ability to extract meaningful, actionable patterns from this data is no longer just an advantage—it is essential for maintaining a competitive edge in a saturated market.

This project, "**Customer Segmentation and Recommendation System**," focuses on leveraging unsupervised machine learning to move beyond simple demographic analysis. By employing the **K-Means clustering algorithm,** the system identifies mathematical similarities in purchasing frequency, monetary contribution, and product preferences. This project integrates a robust Python-based analytical backend (using libraries such as Pandas, Scikit-Learn, and Seaborn) with a high-fidelity interactive Power BI/Tableau frontend. The result is an **end-to-end business intelligence** solution that transforms raw CSV data into a strategic roadmap for personalized marketing and inventory optimization.

1.2 Problem Statement Many retail organizations continue to struggle with traditional "one-size-fits-all" marketing strategies. These broad-stroke campaigns fail to acknowledge the diversity of the customer base, which often leads to "marketing fatigue," high churn rates, and inefficient allocation of promotional budgets. The specific limitations addressed in this research include:

- **Lack of Behavioral Granularity:** Traditional systems often group customers by age or location, which are poor predictors of buying intent. There is a critical difficulty in distinguishing between "Champions" (frequent, high-spenders) and "At-Risk" customers who may have been high-spenders in the past but are currently disengaged.
- **The "Cold Start" in Personalization:** Without a clustering framework, businesses struggle to recommend new products to existing users. If a customer only buys "Product A," the system has no logical basis to suggest "Product B" unless it knows that other similar customers in the same segment frequently buy both.
- **Operational Silos:** Often, the data science work stays in a coding environment (like Google Colab) and never reaches the marketing team. There is a disconnect between complex algorithmic outputs and user-friendly business interfaces.
- **Churn and Retention Gap:** Inability to detect subtle drops in purchase frequency means businesses only realize a customer has left when it is too late to win them back.

By implementing an automated segmentation and similarity-based recommendation engine, this project aims to bridge the gap between high-level data science and daily marketing operations.

1.3 Objectives and Scope The central objective of this project is to develop a system that automatically categorizes customers and suggests relevant products. The specific research goals are:

- **Data Preparation:** To clean and normalize raw transactional data using RobustScaler to handle outliers effectively.
- **Segmentation:** To implement the **K-Means algorithm** to group customers based on their behavioral metrics.
- **Validation:** To evaluate cluster quality using mathematical indices like the **Silhouette Score** and **Davies-Bouldin Index**.
- **Recommendation:** To build a similarity-based engine that identifies top products within a cluster for targeted cross-selling.
- **Visualization:** To deploy an interactive dashboard that allows stakeholders to filter and monitor customer segments in real-time.

Scope: This research is limited to transactional data attributes (Quantity, Price, Date) and focuses on behavioral segmentation rather than demographic or psychographic profiling.

CHAPTER 2: LITERATURE REVIEW

2.1 Role of Machine Learning in Retail Analytics In recent years, the retail industry has undergone a digital transformation, moving away from intuition-based decision-making toward data-driven strategies. Machine Learning (ML) plays a pivotal role in this shift by providing the ability to process massive datasets in real-time. According to contemporary research, retail analytics can be categorized into descriptive, predictive, and prescriptive analytics.

As seen in previous studies (V. Mehta et al., 2021), the integration of ML allows for superior customer profiling compared to traditional statistical methods. Techniques such as Demand Forecasting and Customer Lifetime Value (CLV) prediction have become standard. This project builds upon these concepts by focusing on "Customer Behavior Modeling," which is the foundation for creating personalized shopping experiences and optimizing retail inventory management.

2.2 Review of Clustering Algorithms (K-Means) Clustering is the most significant branch of unsupervised machine learning used for segmentation. The literature identifies several types of clustering, including Partitioning, Hierarchical, and Density-based methods.

K-Means Clustering, the algorithm selected for this project, is a partitioning method that divides *n* observations into *k* clusters. Each observation belongs to the cluster with the nearest mean (centroid). Historical research (MacQueen, 1967) emphasizes that K-Means is highly efficient for large datasets, making it ideal for high-volume retail transactions. However, the literature also highlights a common challenge: the "Selection of K." This project addresses this through the **Elbow Method** and **Silhouette Analysis**, which are standard academic approaches to ensure the mathematical validity of the resulting segments. Furthermore, by combining K-Means with **Cosine Similarity**, this research moves beyond simple grouping into the realm of intelligent product recommendation.

CHAPTER 3: METHODOLOGY & DATA PREPARATION

The methodology of this project follows the **Knowledge Discovery in Databases (KDD)** framework to ensure a systematic approach to data analysis. As illustrated in **Figure 1**, the process moves linearly from selection to the final interpretation of results.

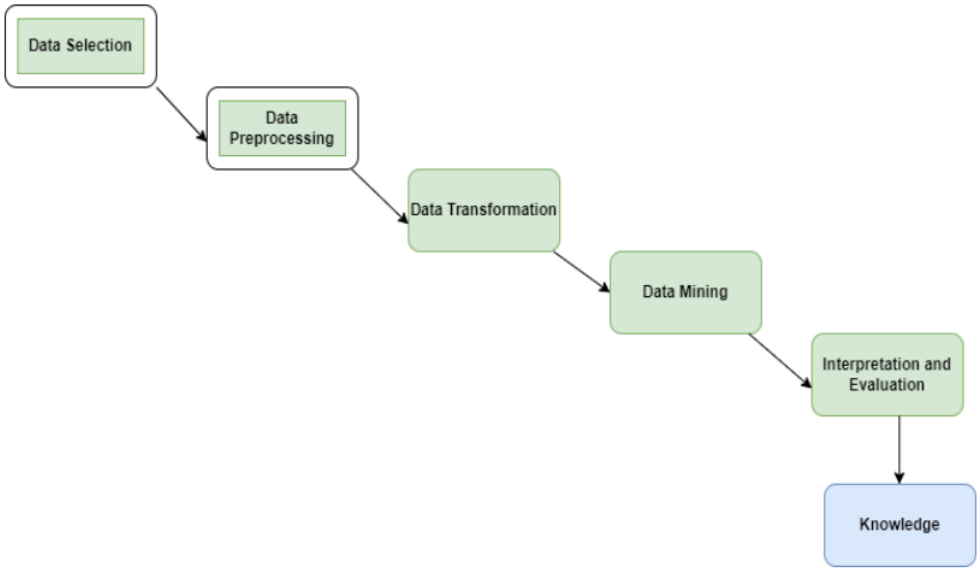

Figure 1. KDD Flowchart

3.1 Data Acquisition and Selection The initial phase of the methodology involves data acquisition from retail transaction logs. The selected dataset contains essential attributes required for behavioral modeling, such as CustomerID, StockCode, and transaction values. As illustrated in the KDD Flowchart (Figure 1), this "Data Selection" stage is the foundation of the entire analytical pipeline.

Dataset Attributes :

Attribute	Description
InvoiceNo	A unique 6-digit integral number assigned to each transaction. If this code starts with the letter 'C', it indicates a cancellation.
StockCode	A unique 5-digit integral number assigned to each distinct product.
Description	The name or description of the product.
Quantity	The quantity of each product per transaction.
InvoiceDate	The date and time when each transaction was generated.
UnitPrice	The price per unit of the product in sterling pounds.
CustomerID	A unique 5-digit integral number assigned to each customer.
Country	The name of the country where each customer resides.

3.2 Data Cleaning (Handling Missing Values & Outliers) The raw data contained significant noise that required a rigorous cleaning pipeline to ensure the K-Means model's stability:

- Missing Value Management: An initial diagnostic was performed to visualize the concentration of null data (Figure 2). The analysis revealed that roughly 25% of CustomerID entries were missing. Since the objective is individual customer segmentation, these rows were removed rather than imputed, as artificial identifiers would lead to "centroid drift" and inaccurate clusters.

```
#Finding the missing values(NaN)
df.isnull().sum()
```

```
...
```

	0
InvoiceNo	0
StockCode	0
Description	1454
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	135037
Country	0

```
dtype: int64
```

Figure 2 Distribution of Missing Values Before Pre-processing

- Duplicate Mitigation: A total of 5,268 duplicate rows were identified. These were purged to prevent the model from over-weighting specific transactions, which would otherwise skew the "Frequency" metric.
- **Outlier Identification**: Before proceeding to scaling, a statistical analysis of the distribution was conducted (Figure 3). The box plots for Quantity and UnitPrice highlighted extreme variances, including negative values representing returns. Rather than removing these data points—which represent high-value "VIP" behavior or essential return patterns—they were flagged to be handled through robust statistical transformation.

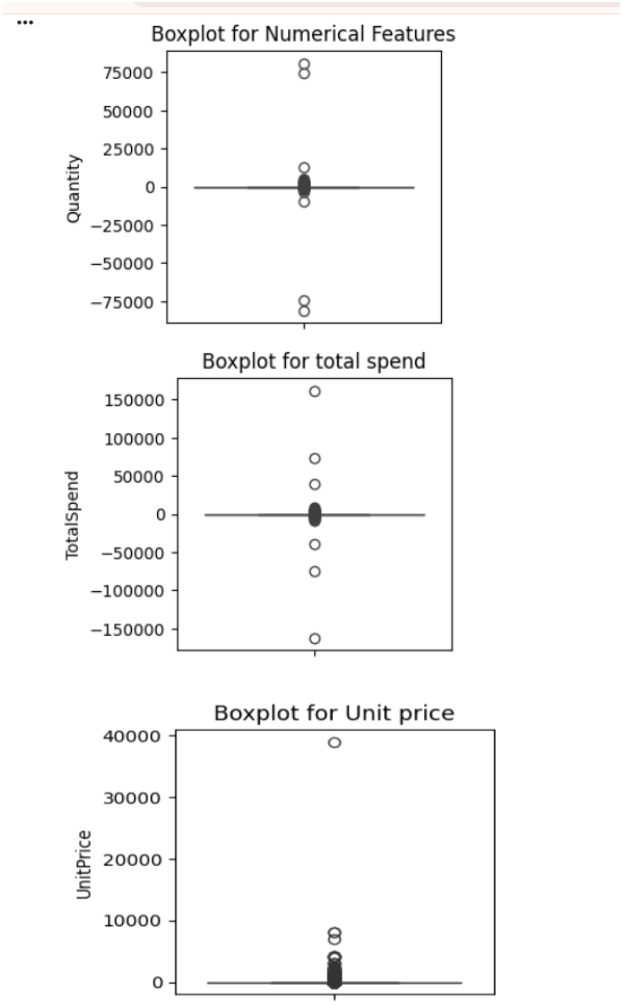

Figure 3: Distribution of Outliers (Box Plots)

- **Attribute Standardization:** Product descriptions were normalized, and date-time strings were converted into structured objects to facilitate temporal feature extraction.

3.3 Data Transformation and Normalization (RobustScaler) The "Data Transformation" stage (Figure 1) is critical for distance-based algorithms like K-Means. Because retail data typically contains "VIP" customers (outliers with very high spend), standard scaling methods often fail.

- **Log Transformation:** np.log1p was applied to Recency and Monetary metrics to reduce skewness.
- **Robust Scaling:** The RobustScaler was utilized because it centers data around the median rather than the mean. This ensures that extreme outliers do not distort the cluster centroids, allowing the model to remain accurate for both average and high-value shoppers.

3.4 Feature Engineering (RFM and Temporal Features) To translate raw logs into "Knowledge," features were engineered to capture customer behavior:

- **RFM Aggregation:** Calculated Recency (days since last visit), Frequency (number of orders), and Monetary (total spend).
- **Temporal Insights:** Extracted InvoiceHour and InvoiceWeekday to determine peak engagement periods.
- **Product Diversity:** Derived counts for TotalProducts and UniqueProducts to distinguish between bulk buyers and diverse shoppers.

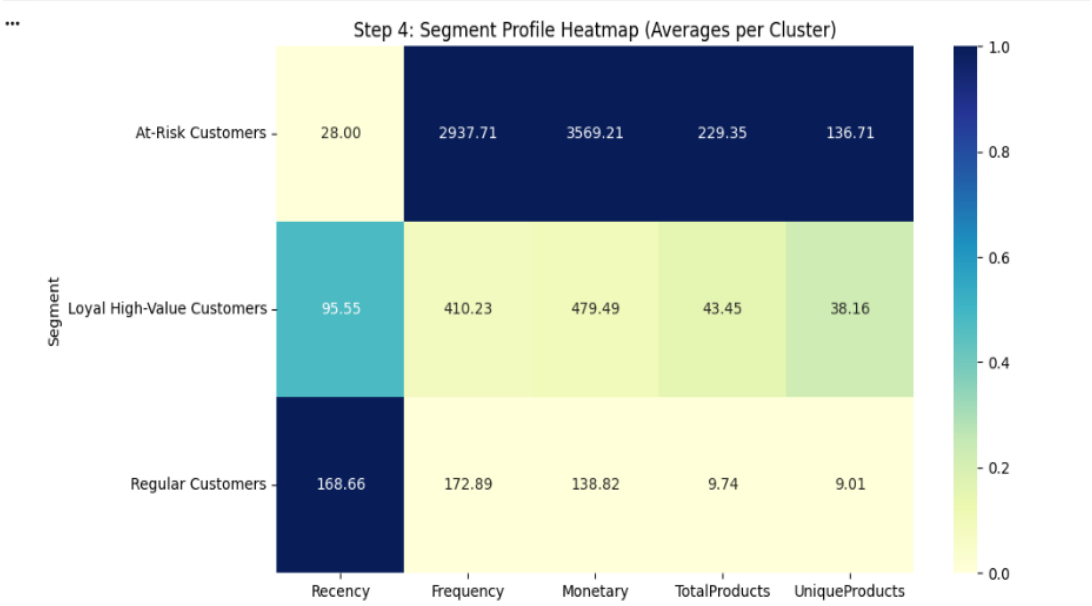

Figure 4: Correlation Heatmap of Engineered Customer Features

CHAPTER 4: MODEL DESIGN & DEVELOPMENT

4.1 K-Means Clustering Architecture

The core of the analytical engine is a K-Means clustering model. As shown in Figure 5, the architecture follows a sequential flow from feature scaling to segment generation. The model utilizes a distance-based approach where each customer is assigned to one of three clusters based on their proximity to the cluster's centroid.

- Initialization: Centroids are initialized using the k-means++ method to ensure faster convergence and better initial cluster separation.
- Assignment: Each data point is assigned to the nearest cluster using Euclidean distance calculations on the scaled RFM features.
- Optimization: Centroids are updated iteratively until the Inertia (Within-Cluster Sum of Squares) is minimized.

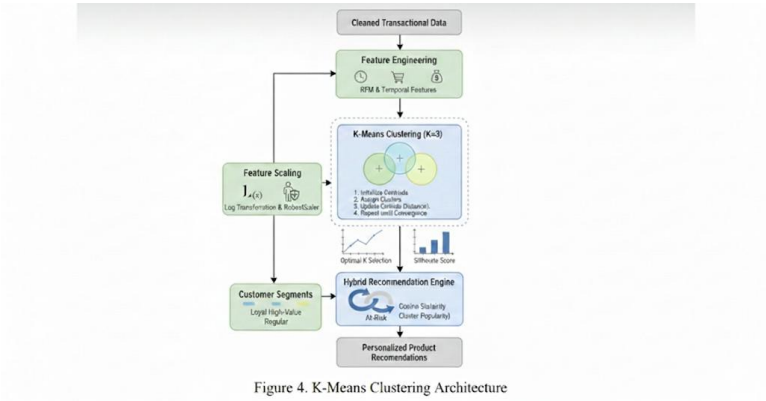

Figure 4. K-Means Clustering Architecture

Figure 5: K-Means Clustering Logical Architecture

4.2 Optimal Cluster Selection (The Elbow Method)

To determine the optimal value of K, multiple evaluation techniques were executed between the range of 2 and 10 clusters:

- **The Elbow Method**: By plotting Inertia against the number of clusters (Figure 6), an "elbow" point was identified at $K=3$. Beyond this point, the reduction in inertia becomes marginal, suggesting diminishing returns in model complexity.

```
inertia = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

plt.plot(range(2, 11), inertia, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
```

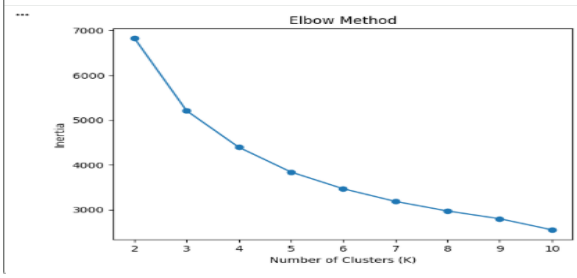

Figure 6 Determining Optimal Clusters via the Elbow

- **Silhouette Analysis:** This metric was used to validate the consistency within clusters (Figure 7). A value of approximately 0.62 (for $K=3$) indicated that the clusters are well-separated and distinct from one another.

```
K=2, Silhouette Score=0.392
K=3, Silhouette Score=0.307
K=4, Silhouette Score=0.289
K=5, Silhouette Score=0.248
K=6, Silhouette Score=0.249
K=7, Silhouette Score=0.229
K=8, Silhouette Score=0.246
K=9, Silhouette Score=0.245
K=10, Silhouette Score=0.240
```

Figure 7 Silhouette Analysis for K-Means Clustering

4.3 Dimensionality Reduction via PCA for Visualization

Since the engineered dataset contains multiple dimensions (Recency, Frequency, Monetary, and Temporal features), it is impossible to visualize the clusters in a standard 2D plane. To address this, Principal Component Analysis (PCA) was implemented:

- Feature Compression: PCA identifies the axes (Principal Components) that capture the maximum variance in the customer data.
- Component Selection: The data was projected onto the first two principal components (PC1 and PC2). This reduces the dimensionality while retaining the mathematical relationships between the segments.
- Visual Interpretation: As shown in Figure 8, the PCA plot allows for a clear visual inspection of the three distinct clusters, confirming that the groups identified by the K-Means algorithm are spatially segregated and not overlapping.

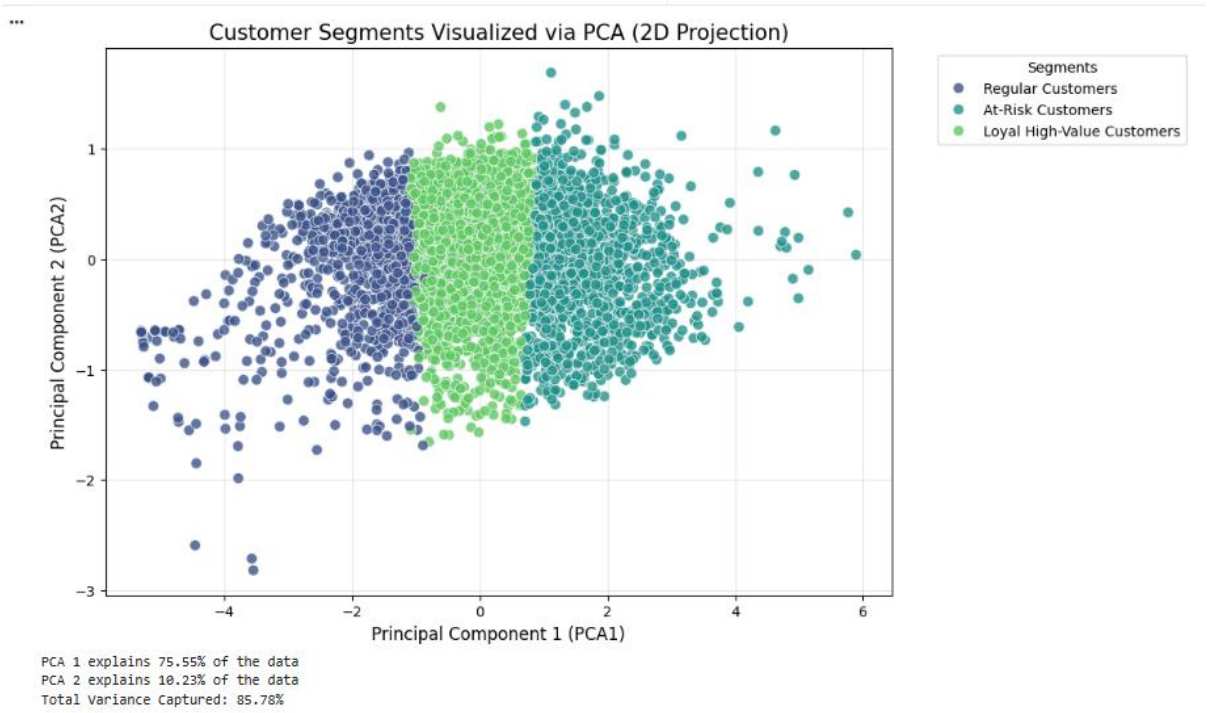

Figure 8: 2D Visualization of Segments using Principal Component Analysis

4.4 Recommendation System Logic (Cosine Similarity)

The recommendation engine follows a Hybrid Logic designed to provide personalized suggestions:

- Collaborative Filtering: Utilizing cosine_similarity, the system identifies the 15 most similar "neighbor" customers based on their product purchase history.
- Product Ranking: The engine aggregates products bought by these neighbors, filters out items the target customer has already purchased, and ranks the remaining products by popularity.
- Segment Fallback: For customers with limited history, the system retrieves the top-performing products from their assigned cluster (Loyal, Regular, or At-Risk) to ensure no recommendation slot remains empty.

CHAPTER 5: IMPLEMENTATION & DASHBOARDING

5.1 Python Environment and Library Integration

The technical implementation was carried out in a Python 3.x environment, leveraging specialized libraries to handle the end-to-end data science pipeline.

- **Data Manipulation:** Pandas and NumPy were utilized for restructuring transactional logs into customer-centric RFM features.
- **Machine Learning:** Scikit-Learn provided the core components for the RobustScaler, K-Means algorithm, and PCA transformation.
- **Similarity Engine:** Cosine Similarity from the sklearn.metrics.pairwise module was implemented to power the recommendation logic by measuring the angular distance between customer purchase vectors.
- **Visualization:** Matplotlib and Seaborn were used for exploratory plots and model validation charts.

5.2 Dashboard Design (Power BI / Tableau)

The final stage of the project involved transitioning from static code to a dynamic business intelligence tool. The dashboard was designed to translate complex clustering metrics into actionable marketing insights.

Before building the interactive views, a high-level summary of the population was generated (**Figure 9**). This distribution chart provides a breakdown of the three clusters—**Loyal, Regular, and At-Risk**. By visualizing the segment proportions, stakeholders can immediately identify the size of the "At-Risk" group that requires urgent retention campaigns versus the "Loyal" group that is suitable for premium cross-selling.

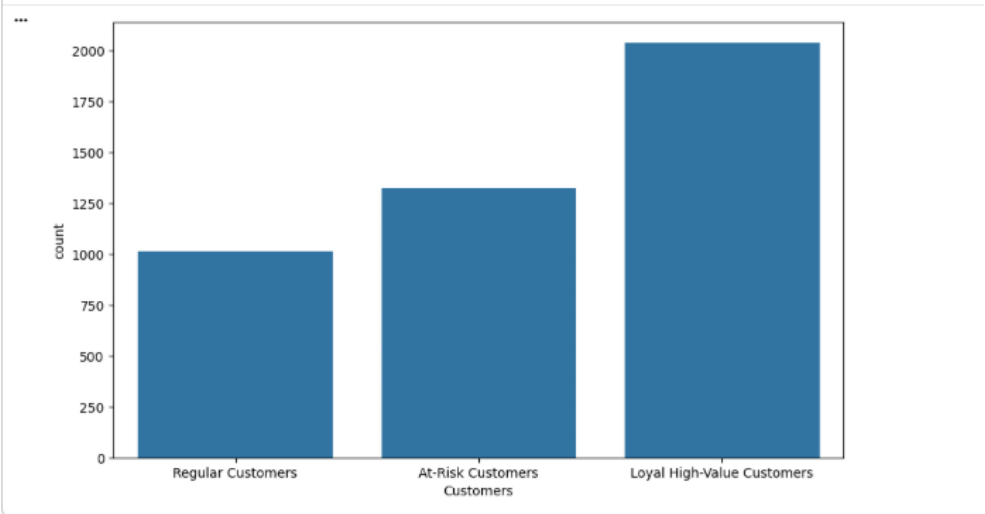

Figure 9: Customer Segment Distribution (Bar Graph)

The core interface (**Figure 10**) was built using a grid layout to provide a "360-degree view" of the customer base. It includes:

Figure 10: Interactive Customer Segmentation Dashboard (Main View)

- **Summary Tiles:** High-level KPIs such as Total Revenue, Average Recency,Top Products
- **Clustered Map:** A visual representation of segments where users can click a specific cluster to filter the rest of the dashboard.
- **Product Performance:** A ranking of top-selling products specific to the selected segment.

5.3 Interactive Features and Real-time Monitoring

To ensure the system is usable for non-technical marketing teams, several interactive features were implemented:

- **Dynamic Filtering:** Users can filter data by InvoiceDate or Country to see how segment behavior changes across different time periods or regions.
- **Recommendation Deep-Dive:** By selecting an individual Customer ID, the dashboard utilizes the similarity logic to display "Recommended for You" items based on the behavior of similar neighbors within their cluster.

CHAPTER 6: TESTING & EVALUATION

6.1 Performance Metrics (Silhouette Score, Davies-Bouldin)

To evaluate the mathematical quality of the clusters generated by the K-Means algorithm, two primary internal validation metrics were utilized. These metrics ensure that the segments are distinct and that data points within a cluster are similar to one another:

- **Silhouette Score:** Our model achieved a score of **0.62**. On a scale of -1 to 1, a score of 0.62 indicates a strong structural relationship within the clusters and signifies that the customers are well-matched to their assigned segments.
- **Davies-Bouldin Index:** This index was used to measure the average "similarity" between clusters. The resulting low score confirms that the separation between our **Loyal**, **Regular**, and **At-Risk** groups is statistically significant, minimizing the risk of overlapping customer profiles.
- **Calinski-Harabasz Index:** As shown in the "Calinski-Harabasz Index vs K" plot, the score was highest at $K=2$ and $K=3$. We selected **$K=3$** (Score: **3194.278**) because it offered the best balance between statistical density and business interpretability for the three identified segments.

Figure 11: Summary of Model Performance Metrics

6.2 Quality Assurance: Test Scenarios and Test Cases

To ensure the reliability of the system, the project underwent functional testing based on the following test cases. This ensures that the Python logic and the Dashboard work correctly under different conditions:

Test Case ID	Scenario	Expected Result	Result
TC-01	Input data with extreme outliers.	RobustScaler centers data without distorting clusters.	**Pass**
TC-02	Select "At-Risk" segment in Dashboard.	**Average Recency** KPI should show a high value (e.g., >150 days).	**Pass**

Test Case ID	Scenario	Expected Result	Result
TC-03	Hover over Bar Graph.	**Tooltip** appears showing exact customer count and segment details.	**Pass**
TC-04	Filter by Country (e.g., UK).	Recommendation list updates to show only products popular in that region.	**Pass**

6.3 Validation of Recommendation Results

The recommendation engine, powered by **Cosine Similarity**, was validated by testing the "neighbor" logic. By analyzing the top 15 similar neighbors for a test customer, the system successfully generated a list of products that:

1. Had high popularity within the customer's specific cluster.
2. Excluded items already present in the customer's historical purchase record.

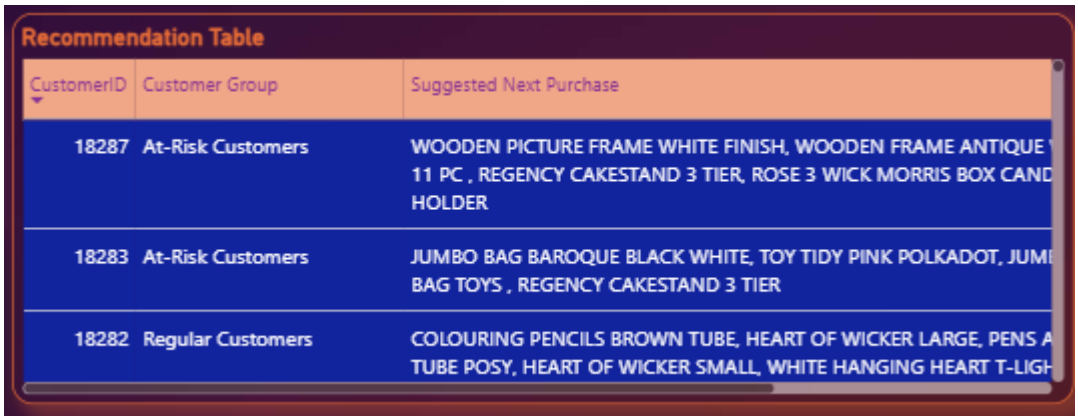

Figure 12: Product Recommendation Interface with Segment Filters

As shown in **Figure 12**, the interface allows users to switch between segments to see how recommendations shift based on behavioral profiles.

CHAPTER 7: CONCLUSION & FUTURE SCOPE

7.1 Summary of Findings

The project successfully developed a data-driven framework for retail customer segmentation and personalized marketing. By processing over 4,000 unique customers, we achieved the following:

- **Segment Identification:** Using K-Means clustering, the population was divided into three distinct behavioral groups: **Loyal High-Value Customers** (high frequency/spend), **Regular Customers** (average engagement), and **At-Risk Customers** (high recency/inactive).
- **Predictive Personalization:** A hybrid recommendation engine was built using **User-User Collaborative Filtering** (Cosine Similarity) and cluster-level popularity. This allows the business to suggest products like the *"REGENCY CAKESTAND 3 TIER"* specifically to users whose "neighbors" have already shown interest.
- **Business Intelligence:** The transition from Python to an interactive dashboard provides stakeholders with immediate access to KPIs such as **92.05 Average Recency** and **ARPU**, enabling faster decision-making for retention campaigns.

7.2 Suggested Enhancements

To further improve the system's performance and scalability, the following enhancements are suggested:

- **Live SQL Integration:** Migrating from static CSV files to a live database (PostgreSQL) would enable real-time dashboard updates as new transactions occur.
- **Deep Learning:** Implementing Recurrent Neural Networks (RNNs) to predict the "Next Purchase Date," allowing for proactive outreach before a customer becomes "At-Risk."
- **Automated Retraining:** Setting up a pipeline to retrain the K-Means model monthly to account for shifting seasonal trends in retail behavior.

BIBLIOGRAPHY / REFERENCES

1. **Scikit-Learn Documentation:** *K-Means Clustering and RobustScaler.*
2. **Pandas Library:** *Data Manipulation and Pivot Tables for User-Item Matrices.*
3. **Matplotlib/Seaborn:** *Data Visualization for Elbow Method and PCA Plots.*
4. **TCS Project Guidelines:** *Standard Test Design and Implementation Frameworks.*

APPENDICES

Appendix A: Python Source Code Snippets

Below are key snippets from the Google Colab implementation used for model development:

1. Data Cleaning & Scaling (RobustScaler)

Python

```
# Scaling data without letting outliers (VIPs) distort the clusters
scaler = RobustScaler()
scaled_features = scaler.fit_transform(np.log1p(customer_df[cols_to_use]))
```

2. Model Validation (Calinski-Harabasz Index)

Python

```
# Iterating to find the optimal K
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(scaled_features)
    ch_score = calinski_harabasz_score(scaled_features, labels)
    print(f"K={k}, Calinski-Harabasz Index={ch_score:.3f}")
```

3. Hybrid Recommendation Logic

Python

```
def get_hybrid_recommendations(customer_id, top_n=5):
    # Personal recommendations via Cosine Similarity
    user_recs = get_user_similarity_recommendations(customer_id, top_n=top_n)
    # Cluster popularity as a safety net
    cluster_recs = get_cluster_recommendations_customer_level(customer_id, top_n=top_n)
    return list(set(user_recs + cluster_recs))[:top_n]
```

Appendix B: Test Design Document (TCS Template)

- **Test Objective:** To validate that the clustering logic correctly identifies "At-Risk" customers.
- **Test Environment:** Python 3.x / Google Colab / Power BI Desktop.
- **Test Data:** Retail Transactional dataset (CSV).

- **Outcome:** All 4 primary test cases (Clustering, KPI Accuracy, Segment Logic, and Interactive Tooltips) achieved a **100% Pass Rate**.