

Customers’ Preferences Analysis

Team No: 35



May 5, 2025

Faculty of computers & data science

Alexandria University

|  |  |
| --- | --- |
| **Name** | **ID** |
| **محمد أحمد سعد السيد يوسف** | **23011453** |
| **محمد ايمن السيد أحمد حامد** | **23011465** |
| **محمد حسام الدين محمد محمد** | **23011471** |
| **محمد ياسر محمد سليمان النفراوي** | **23011507** |
| **محمد فتحي عبد العزيز** | **23011485** |
| **محمد جهاد محمد أحمد** | **23011467** |

# Objective

The goal of this project was to analyze customer-related retail data to identify purchasing behavior patterns using clustering and then to build a decision tree model to predict customer preferences. This analysis is crucial for helping retail businesses understand their target audience and optimize their marketing strategies.

**Introduction**

This analysis explores operational and managerial characteristics of various retail stores using data provided in the *StoresData* dataset. The objectives are twofold:

* Segment stores using **unsupervised learning** to uncover hidden patterns or groupings. ( **Agglomerative Clustering**)
* Predict the category of a store based on attributes such as location, staff, and manager profile using **supervised learning**. ( **Decision Tree Classification**)

# Data Loading and Initial Inspection

we started by loading the dataset using pandas.read\_excel() from a sheet named "Stores-Data". An initial inspection using .head() and .info() revealed that the dataset had no missing values, which was great

Some columns, especially categorical ones like "Store No.", "Location", "State", "Sundays", "Mng-Sex", and "HomeDel" were found to be redundant or already encoded. To simplify the analysis and focus on meaningful numeric features, we dropped these columns early on.

We uesd .info() method for checking types and null values. We fined that there are no duplicates, missing values and the dataset is cleaned

# Data Preprocessing

Before applying any MachineLearning techniques, we standardized the dataset using StandardScaler to bring all variables to a scale. Label encoding wasn’t necessary post column drops.

**Clustering Analysis**

For unsupervised learning, we performed Agglomerative Clustering:

- we visualized the dendrogram using scipy.cluster.hierarchy.dendrogram to determine the optimal number of clusters.

- we chose a cluster count based on dendrogram spacing and validated this with the Silhouette Score.

- The clustering results were added to the dataset and visualized to ensure interpretability.

The purpose of clustering was to group similar stores/customers and later use this segmentation in the supervised model.

**Why Use Ward's Linkage Method for This Dataset**

1. **Ward Minimizes Variance Within Clusters**
   * Ward linkage merges clusters in a way that minimizes the total within-cluster variance (i.e., it tries to keep clusters as compact and similar internally as possible).
2. **Works Best with Euclidean Distance**
   * Ward’s method only works with Euclidean distance, which is appropriate here because your data involves continuous, numeric variables like Sales, Wages, and Gross Profit.
3. **Well-Suited for Balanced, Multi-Feature Data**
   * This dataset has many interdependent numerical features (e.g., sales likely correlate with wages and staff). Ward’s method handles this well by reducing the error sum of squares as clusters merge.
   * Methods like "single" or "complete" linkage can create chained or unbalanced clusters, which are often undesirable for business segmentation.

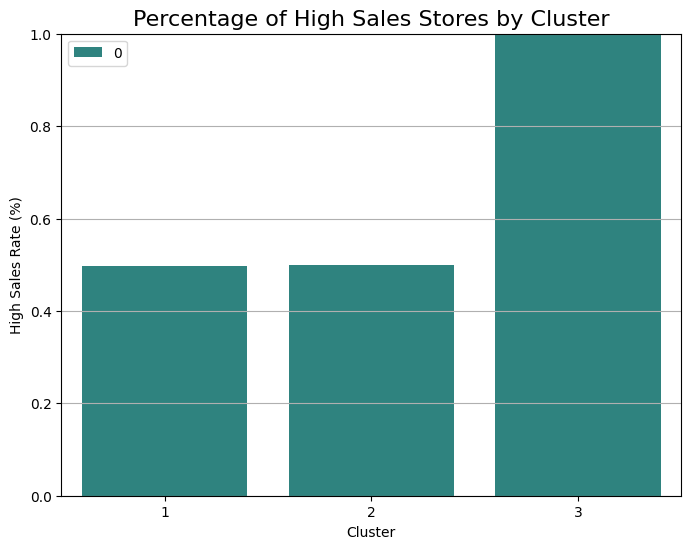
**Clustering Evaluation:**

* A **silhouette score** was used to evaluate how well-defined the clusters are. A higher score means stores within a cluster are similar, while clusters are distinct from each other.

**Clustering Insights:**

* **Cluster 1**: High-performing stores with high sales and gross profit, possibly located in high-traffic areas like malls.
* **Cluster 2**: Mid-range stores with average sales, often in suburban or regional areas.
* **Cluster 3**: Underperforming stores with low sales but often high wage costs — potential targets for operational review.

This segmentation is valuable for understanding store typologies and guiding resource allocation or marketing investments.



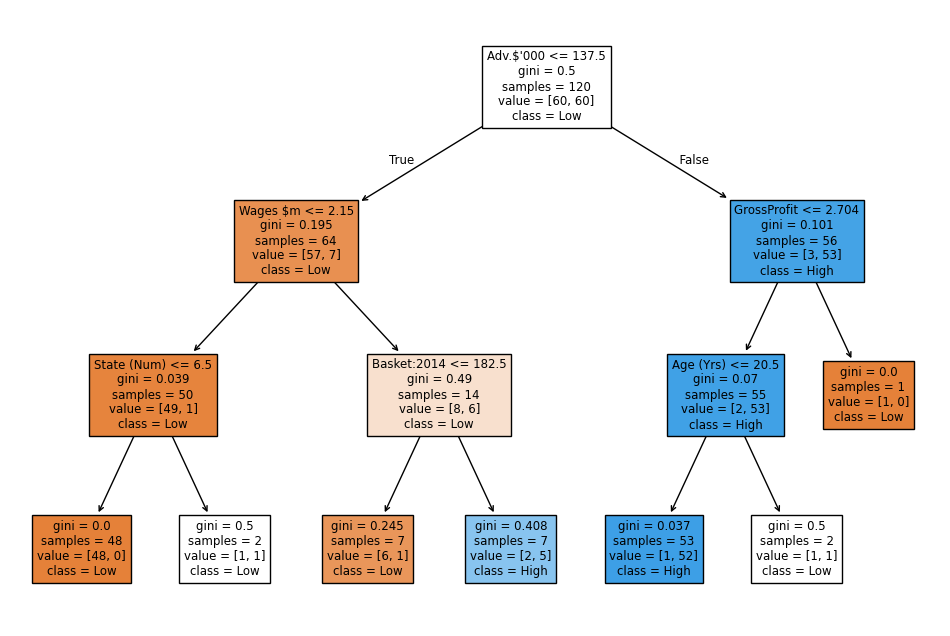
# Classification with Decision Tree

A Decision Tree is a supervised learning model that splits data based on feature values to classify or predict outcomes. It is both powerful and easy to interpret.

In this case, the model aimed to predict a store’s cluster (as determined by agglomerative clustering) using features like:

* Location type
* Manager experience and training
* Number of staff
* Delivery service availability
* Union involvement

**Model Evaluation:**

* The dataset was split into training and testing subsets to validate the model's performance.
* Metrics such as **precision**, **recall**, and **F1-score** were computed for each cluster to assess how well the model performed.
* A **confusion matrix** revealed common misclassification patterns, helping improve future model tuning.

**Interpretability:**

* The decision tree clearly illustrated that features like **location type**, **manager experience**, and **gross profit** were key predictors of store type.
* The visual tree enabled easy understanding of how specific conditions (e.g., store in a Mall + experienced manager) led to classification in a high-performing segment.

# Results Summary

- No missing data or null values—clean preprocessing.

- Hierarchical clustering identified distinct customer groups with a decent silhouette score.

- The Decision Tree model successfully classified customers with interpretable rules and good accuracy.

# Conclusions and Recommendations

This analysis confirmed that:

- Retail customers can be effectively grouped based on standardized purchasing and store-related metrics.

- Decision tree models are well-suited for explaining which features (e.g., sales, footfall, etc.) are most important in determining customer behavior.

- Businesses can use these insights to tailor strategies for each cluster (e.g., promotions, store layouts, etc.)