Project Report: Retail Store Sales Data Cleaning & Preprocessing

---

1. Project Description & Objectives

1.1. Description

This project focused on the critical first step of the data science pipeline: cleaning and preprocessing a raw retail sales dataset. The primary goal was to transform the raw, incomplete data into a high-quality, structured dataset suitable for both exploratory data analysis (EDA) and building machine learning models. This involved handling missing data, correcting structural issues, treating outliers, and encoding variables to ensure data integrity and consistency.

1.2. Initial Data Challenge

The initial dataset provided was unusable as it contained no column headers, making interpretation and processing impossible. After communication with the support team, a substitute dataset, `retail_store_sales.csv`, was sourced from Kaggle. This report documents the cleaning process applied to this substitute dataset.

1.3. Dataset Overview

The dataset contains **12,575 transactions** with **11 attributes**, capturing details such as customer information, product details, transaction amount, payment method, and location.

Initial Data Snapshot:

- **Total Entries:** 12,575
- **Features:** 11
- **Data Types:** Mix of numeric (`float64`) and categorical (`object`) data.

---

2. Data Quality Report: Initial Assessment

A thorough analysis was conducted to assess the quality of the raw data before any cleaning steps were applied. The following issues were identified:

2.1. Missing Values Analysis

The percentage of missing values for each column was calculated using: `df.isnull().sum() / df.shape[0] * 100`

Column Name	Missing Value %	Severity
Transaction ID	0.00%	☑ None
Customer ID	0.00%	☑ None

Column Name	Missing Value %	Severity
Category	0.00%	☑ None
Item	**9.65%**	⚠ **Moderate**
Price Per Unit	**4.84%**	⚠ **Moderate**
Quantity	**4.80%**	⚠ **Moderate**
Total Spent	**4.80%**	⚠ **Moderate**
Payment Method	0.00%	☑ None
Location	0.00%	☑ None
Transaction Date	0.00%	☑ None
Discount Applied	**33.39%**	✖ **High**

Conclusion: The dataset suffers from significant missing data, particularly in the `Discount Applied` column, requiring a strategic imputation approach.

2.2. Data Types and Inconsistencies

- The `Discount Applied` column was stored as an `object` (string) type, but its content suggests it should be a numerical value (e.g., 0.1 for 10% off). This required conversion.
- The `Transaction Date` is stored as a string and would need conversion to a `datetime` object for time-series analysis (though not a focus of this cleaning phase).

---

3. Phases of Data Cleaning & Preprocessing

Phase 1: Handling Missing Data

Strategy:

1. **For `Item` and `Discount Applied`:**
    - **Technique Applied:** K-Nearest Neighbors (KNN) Imputation.
    - **Justification:** These columns had complex relationships with other features. KNN imputation preserves these relationships by finding similar records (`n_neighbors=8`) to estimate missing values, which is more sophisticated than simple mean/mode imputation.
    - **Action:** `KNNImputer` was used to fill missing values numerically. The `Item` column was first mapped to a numerical key for imputation and then mapped back to its original string values.

2. **For `Price Per Unit`, `Quantity`, and `Total Spent`:**
    - **Technique Applied:** Row Deletion.
    - **Justification:** Imputation for these critical financial columns is illogical and would introduce significant bias. A missing price or quantity cannot be accurately inferred without knowing the specific product and transaction context.

- **Action:** All rows with missing values in these three columns were identified and removed. This resulted in the dataset being reduced from 12,575 to 11,306 records.

Phase 2: Outlier Detection and Treatment

Strategy:

1. **Detection:**
   - **Technique Applied:** Interquartile Range (IQR) method.
   - **Process:** Boxplot visualizations were generated for all numerical columns (`Price Per Unit`, `Quantity`, `Total Spent`) to visually identify potential outliers. The IQR method was then applied programmatically to the `Total Spent` column to detect statistical anomalies.
2. **Treatment:**
   - **Action:** The identified outlier records in the `Total Spent` column were removed from the dataset to prevent them from skewing future analyses and models.
   - **Result:** The final cleaned dataset contains **11,306 records**.

Phase 3: Encoding Categorical Variables

Strategy: Different encoding techniques were applied based on the nature of each categorical variable.

Column	Data Type	Encoding Technique	Justification
`Customer ID`	Nominal	**Label Encoding**	Although nominal, there are many unique IDs (25). Label Encoding is efficient and suitable for tree-based models.
`Category`	Nominal	**Label Encoding**	Many unique categories (8). Efficient for modeling.
`Item`	Nominal	**Label Encoding** (via mapping)	High cardinality (200 unique items). Custom mapping was used for control.
`Payment Method`	Nominal	**One-Hot Encoding**	Only 3 unique categories (Cash, Credit Card, Digital Wallet). Prevents false ordinal relationships.
`Location`	Nominal	**One-Hot Encoding**	Binary category (Online, In-Store). Perfect for One-Hot.

Action:

- `LabelEncoder` from `scikit-learn` was used for `Customer ID` and `Category`.
- A pre-defined mapping dictionary (`items_key`) was used for the `Item` column.
- `OneHotEncoder` from `scikit-learn` was used for `Payment Method` and `Location`, creating new binary columns for each category.

Phase 4: Feature Scaling

Strategy: Scaling was applied to normalize the range of numerical features, which is crucial for distance-based algorithms (e.g., K-Means, SVM, Neural Networks).

Column	Scaling Technique	Justification
`Price Per Unit`	**StandardScaler (Z-score)**	The distribution was not necessarily uniform. Z-score standardization (mean=0, std=1) handles outliers better than Min-Max and is ideal for many algorithms.
`Total Spent`	**StandardScaler (Z-score)**	Same as above. This column had a wide range and potential skew.
`Quantity`	**No Scaling**	The native range is already small and consistent (1-10). Scaling would not provide any benefit and could be omitted for interpretation clarity.

Action: The `StandardScaler` was fit on `Price Per Unit` and `Total Spent`, creating new scaled columns (`Price Per Unit_sc`, `Total Spent_sc`).

---

4. Final Output Datasets

The cleaning process resulted in two distinct, ready-to-use datasets:

1. **`retail_store_sales_clean.csv`**
    - **Purpose:** For **Exploratory Data Analysis (EDA)** and visualization.
    - **Contents:** Contains the cleaned data with original categorical values intact for easy interpretation.
2. **`retail_store_sales_model.csv`**
    - **Purpose:** For **Machine Learning modeling**.
    - **Contents:** Contains only the engineered features:
        - Label Encoded columns (`Customer ID_en`, `Category_en`, `Item_en`).
        - One-Hot Encoded columns (e.g., `Payment Method_Cash`).
        - Scaled numerical features (`Price Per Unit_sc`, `Total Spent_sc`).
        - The target variable or feature `Discount Applied`.

---

5. Tools & Technologies Used

- **Programming Language:** Python
- **Libraries:**
    - `pandas`: For data manipulation, cleaning, and aggregation.
    - `scikit-learn`: For imputation (`KNNImputer`), encoding (`LabelEncoder`, `OneHotEncoder`), and scaling (`StandardScaler`).
    - `plotly.express` (`px`): For generating boxplots for outlier visualization.

---

6. Conclusion

The raw `retail_store_sales.csv` dataset was successfully transformed from a state with significant quality issues (missing data, unencoded categories, unscaled features) into a robust and analysis-ready asset. The process involved:

- **Mitigating missing data** through intelligent imputation and logical row removal.
- **Removing outliers** to ensure model stability.
- **Converting all categorical data** into numerical formats appropriate for machine learning algorithms.
- **Scaling numerical features** to prepare them for algorithms sensitive to feature magnitude.

The resulting datasets, `retail_store_sales_clean.csv` and `retail_store_sales_model.csv`, are now of high quality and are suitable for the next stages of the data science lifecycle: in-depth exploratory analysis and building predictive models.