Assignment

Sales and Promotions Analysis

Kusalani Tharunya Thennakoon

Task 01:

1. Load data

1. Sales.csv : sales_df
2. Promotion.csv: promotion_df
3. Supermarkets.csv: supermarkets_df
4. Item.csv: items_df

2. Data Exploration

Inspect the data structure

1. Sales DataFrame:
    - Contains 1,048,575 entries with 11 columns.
    - All columns have non-null values, indicating no missing data.
    - Most columns are of type int64, with 'amount' being float64.
    - 'amount' ranges from -8.28 to 5900.0, with a mean of 1.78.
    - 'time' ranges from 0 to 2359, representing time of day.
    - 'province' is either 1 or 2.
    - 'week' ranges from 1 to 28.
2. Items DataFrame:
    - Contains 927 unique items.
    - Has columns for code, description, type, brand, and size.
3. Promotion DataFrame:
    - Contains 351,372 entries.
    - Includes code, supermarkets, week, feature, display, and province.
    - 'week' ranges from 43 to 104, different from Sales DataFrame.
4. Supermarkets DataFrame:

- Contains 387 supermarkets.
- Includes supermarket_No and postal-code.

Observations:

- There are no missing values or duplicates in any of the datasets
- The Sales DataFrame contains transaction-level data, with each row likely representing a single purchase.
- The Items DataFrame provides details about each product.
- The Promotion DataFrame contains information about promotional activities for specific products in specific supermarkets and weeks.
- The Supermarkets DataFrame provides location information for each supermarket.

3. Data Preprocessing

Procedure

1. Investigate the negative values in the 'amount' column of the Sales DataFrame.
2. The 'time' column in Sales could be converted to a more standard time format.
3. The difference in 'week' ranges between Sales and Promotion DataFrames needs to be addressed.
4. Handle size column in items.csv
5. Outlier detection and removal in sales dataset
6. Encode categorical variables.
7. Merge datasets for a comprehensive analysis, using 'code', 'supermarket', and 'week' as keys.

1. Negative values in the amount column.
   - Negative amounts could represent returns, refunds, or data entry errors.
   - By taking the absolute value, treat all transactions as positive.
2. The 'time' column in Sales could be converted to a more standard time format.
   - The unique values show that the time is stored as integers (e.g., 1100, 1137, 1148).
   - This format is not standard and can be difficult to work with for time-based analyses.

    - The new format (HH:MM:SS) is a standard time format that's easier to work with in pandas.
    - 1440 unique values after conversion show that the dataset captures sales at a minute-level granularity.

3. The difference in 'week' ranges between Sales and Promotion DataFrames needs to be addressed.
    - Original ranges:
        - Sales DataFrame: Weeks 1 to 28,
        - Promotion DataFrame: Weeks 43 to 104.
    - After alignment:
        - Sales DataFrame: Weeks 43 to 70
        - Promotion DataFrame: Weeks 43 to 104
    - The starting week for both datasets is now the same (week 43).
    - Maintained the original 28-week span of the Sales data, just shifted to align with the Promotion data.
    - Note: promotion data for weeks 71-104, but no corresponding sales data. This might represents future planned promotions or there's missing sales data for these weeks.
4. Handle items.csv
    - Clean the 'type' column by extracting numeric values and converting them to integers.
    - The 'brand' column is converted to lowercase for consistency.
    - Size column cleaning:create a new 'size_oz' column
        - Remove entries containing 'KH#' or '%'
        - Extracts numeric values from the size string
        - Convert to uppercase and remove extra spaces
        - Converts sizes in pounds (LB) to ounces.
        - None values are removed
        - Values are rounded to 2 decimal places for consistency.
    - Size Categorization:
        - Small: < 8 oz
        - Medium: 8 - 15.99 oz
        - Large: 16 - 31.99 oz
        - Extra Large: 32 - 63.99 oz
        - Bulk: >= 64 oz
        - Bulk Item Flagging : Create a boolean column 'is_bulk' for items >= 48 oz.

5. Outlier detection and removal
    - Amount Column:
        - Outliers: 3.70% of the data are outliers, which is a significant proportion.
        - Range: The minimum is 0 and the maximum is 5900, which is an extremely wide range.
        - Distribution: The data is highly skewed. The median (50%) is 1.5, while the mean is 1.78, indicating a right-skewed distribution.
        - Extreme values: The max outlier (5900) is far beyond the 75th percentile (2.19), indicating some extreme values.
    - Units Column:
        - Outliers: 15.13% of the data are outliers, which is a very high percentage.
        - Range: The minimum is 1 and the maximum is 100.
        - Distribution: Highly skewed. The 25th, 50th, and 75th percentiles are all 1, but the mean is 1.19.
        - Discrete nature: This column likely represents count data, so it's natural to have integer values.
    - Dropping these outliers too aggressive, and assuming there are some transactions with higher amount will remove. Because of that, used winsorization for outlier handling.

4. Data Normalization

- sales_df, promotion_df, and supermarkets_df originally used different names 'supermarket', 'supermarkets', and 'supermarket_No' to refer to the same concept.
- By renaming them all to 'supermarket_id', the code ensures consistent naming which is essential for accurate merging.
- Standardized text consistency by converting all string columns to uppercase in items and promotions tables.

5. Data Transformation

- After merging the datasets, observed there are some NaN values.
- NaN values found in Merged dataset dataset:

descrption	47264
type	47264
brand	47264
size_oz	47264
size_category	47264
is_bulk	47264
feature	964838
display	964838

- If any item in the descrption column is missing, it is filled with the string 'Unknown Item'.
- Any missing values in the brand column are filled with 'Unknown Brand'.
- The type column is filled with 0 for any missing values, indicating an unknown type with a numerical representation.
- Any missing values in the size_oz column are replaced with the median value of the column. The median is chosen here because it is a good central measure that is less sensitive to extreme values (outliers).
- Missing values in the size_category column are filled with 'Unknown Size' to indicate that the size is not known.
- Any missing values in the is_bulk column are replaced with 'Unknown'. This might indicate whether the item is sold in bulk or not.
- 'feature' and 'display' Columns:

- full_dataset['feature'].isna(), checks if the value is NaN. If NaN, it returns False, meaning no promotion; if not NaN, it returns True, meaning the item has a promotion.
- Missing values in the feature column are filled with 'No Promotion', indicating that the item does not have a promotion.
- Any missing values in the display column are replaced with 'No Display', indicating that the item is not being displayed or promoted on a shelf or other physical displays.

6. Feature Engineering

- day_of_week: converts the day column (assuming it's in a unit of days since a reference date) into a datetime format and extracts the day of the week (0 for Monday, 6 for Sunday).

- is_weekend: checks if the day of the week is a weekend (Saturday or Sunday). It uses isin([5, 6]) to check if the day is either Saturday (5) or Sunday (6).Creates a binary indicator (1 for weekend, 0 for weekday).

- total_sales: multiplies the amount (price per unit) by the units (number of items sold) to calculate the total sales for each transaction.

- hour: converts the time column to a string, transforms it into a datetime format, and extracts the hour (from 0 to 23).

- time_of_day:
    - Morning: 0 AM to 11 AM
    - Afternoon: 11 AM to 4 PM
    - Evening: 4 PM to 8 PM
    - Night: 8 PM to midnight

- items_per_basket: groups the dataset by the basket column (representing each transaction or shopping cart) and sums the units column to calculate how many items are in each basket.

  - basket_value: groups the dataset by the basket and sums the total_sales column to calculate the total value of each basket (total sales amount per transaction).

  - customer_purchase_frequency: groups the dataset by customerId and counts the number of distinct basket entries for each customer. This calculates how frequently each customer makes purchases (i.e., how many transactions they have made).

Task 02:

1. Maze Class

- Parameters:
  - maze (numpy.ndarray): 2D array representing the maze layout (0: wall, 1: path)
  - start_position (tuple): Starting coordinates (row, col)
  - goal_position (tuple): Goal coordinates (row, col)

2. Q-learning algorithm

- Parameters:
  - maze: Reference to Maze object
  - learning_rate: Rate at which new information overrides old (α)
  - discount_factor: Weight of future rewards (γ)
  - exploration_start: Initial exploration rate (ε)
  - exploration_end: Final exploration rate
  - num_episodes: Total number of training episodes

- Key Methods:
  - get_exploration_rate(current_episode): Calculates current exploration rate
  - get_q_value(state, action): Retrieves Q-value for state-action pair
  - get_action(state, current_episode): Selects action using ε-greedy policy
  - update_q_table(state, action, next_state, reward): Updates Q-values