CCMCA311 DATA SCIENCE

Feature Engineering Mini Project

Submitted By :

Abhishek Verma

Rounak Girdhar Sharma

Jay Prakash

Piyush Sharma

1. Introduction to Feature Engineering

Feature Engineering is the critical, pre-modeling process of using domain knowledge to select, clean, and transform raw, unprocessed data into a structured format of "features" that can be effectively used by machine learning algorithms. The quality and relevance of the features directly dictate the performance and accuracy of any analytical model.

The necessity of this process stems from the "Garbage In, Garbage Out" (GIGO) principle: models trained on messy, inconsistent, or irrelevant data will produce unreliable and inaccurate results. This report documents the systematic application of the four major stages of feature engineering [2] to a "Maven Fuzzy Factory" e-commerce dataset. The objective was not to train a model, but to meticulously prepare the data for such a task[3].

2. Stage 1: Data Cleaning

Definition: Data Cleaning is the foundational stage focused on identifying and rectifying errors, inconsistencies, and inaccuracies within the dataset[4]. The goal is to ensure that the data is correct, complete, and reliable before any further analysis is performed.

Our cleaning process involved four distinct tasks:

- **Handling Missing Values:** We identified a significant number of missing values (NaN) in the `sessions` table, specifically in the `utm_source`, `utm_campaign`, `utm_content`, and `http_referer` columns. Instead of deleting these rows, which would result in substantial data loss, we treated the missing values as a distinct

category. We imputed these `NaN` values with the string `'none'`, signifying that the session originated from a direct or untracked source[5].

- **Treating Duplicates:** We performed checks for both full-row duplicates and duplicate primary key (ID) values across all tables, including `orders` and `sessions`. The analysis confirmed that our dataset contained **zero** duplicate entries, so no removal was necessary[6].
- **Detecting Noisy Data:** Noisy data refers to unrealistic or impossible values, such as `Age=150`[7] or negative prices. We inspected key categorical columns, such as `device_type` in the `sessions` table, and confirmed it only contained valid entries (`'mobile'`, `'desktop'`) with no misspellings. We also analyzed numeric columns like `price_usd` and found no negative or zero values, confirming the absence of unrealistic data.
- **Outlier Detection and Treatment:** Outliers are data points that deviate significantly from the rest of the data. We utilized **Box Plots** to visually detect outliers in numeric columns like `price_usd_order` and `items_purchased`[8]. We found outliers, such as orders with 2 items (where most had 1). Our "treatment" for these was to **retain them**, as they represented valid (though uncommon) business transactions, not data entry errors.

3. Stage 2: Data Integration

Definition: Data Integration involves combining data from multiple, disparate sources (such as different tables or files) into a single, unified dataset[9].

Necessity: Our data was spread across six separate tables (`orders`, `order_items`, `products`, etc.). To perform any meaningful analysis (e.g., "what is the name of the product that was ordered most?"), information from these tables had to be linked.

Example from Project: We performed two "merge" (join) operations to create one master table:

1. The `orders` table was merged with the `order_items` table using `order_id` as the common key.
2. The resulting table was then merged with the products table using product_id as the common key.

This process integrated order details, item details, and product names into a single table (final_data), providing a holistic view of each transaction.

4. Stage 3: Data Transformation

Definition: Data Transformation is the process of converting data from one format or structure into another to make it more suitable for analysis and modeling[10].

We applied several key transformations to our dataset:

- **Encoding Categorical Data:** Machine learning models cannot interpret text values. We applied **One-Hot Encoding** to the `device_type` column in the `sessions` table[11]. This transformed the single text column into two new binary columns: `device_desktop` and `device_mobile`, where a `1` indicates the device used for that session.
- **Standardization:** Numeric features on different scales can bias a model. We applied **Z-score Standardization** (using `StandardScaler`) to the `price_usd_order` column[12]. This transformation rescaled the feature to have a mean of approximately 0 and a standard deviation of 1, ensuring it would be weighted appropriately by a model.
- **Discretization (Binning):** We converted a continuous numeric feature into distinct categories[13]. First, we extracted the `order_hour` (a number from 0-23) from the `created_at_order` timestamp. We then "binned" this hour into four meaningful string categories: 'Night', 'Morning', 'Afternoon', and 'Evening'[14].
- **Data Type Conversion:** Several `created_at` columns were initially loaded as `object` (text) data. We transformed these into proper `datetime` objects to enable time-based operations, such as extracting the hour for discretization.

5. Stage 4: Data Reduction

Definition: Data Reduction aims to reduce the volume of the dataset by eliminating attributes (columns) or records (rows) without compromising the integrity of the analysis[15].

Necessity: Large, complex datasets can be computationally expensive and may suffer from the "curse of dimensionality" or multicollinearity, where redundant features confuse the model.

Our reduction process involved two steps:

- **Removal of Unnecessary Attributes:** We dropped all ID columns (`order_id`, `user_id`, `website_session_id`, `product_id`, etc.)[16]. These columns are

essential for joining tables but provide no predictive value for a model. We also removed intermediate columns, like `order_hour`, which had been replaced by our `time_of_day` feature.

- **Correlation Analysis:** We generated a **Heatmap** to analyze the correlation matrix of all numeric features[17]. We discovered extremely high positive correlations (multicollinearity) between `price_usd_order` and `cogs_usd_order` (0.98), and between `price_usd_item` and `cogs_usd_item` (0.96). This indicates they carry redundant information. To "treat" this[18], we dropped the redundant `cogs_` (cost) columns, retaining only the price information.

6. Conclusion

Through the systematic application of these four stages, the raw, multi-table dataset was successfully processed. The final output is a single, clean, and feature-engineered table, free of missing values, errors, and redundant attributes. The data is now in an optimal format for use in a machine learning pipeline.