Documentation of Brazilian E-Commerce Public Dataset by Olist

The Data:

The dataset used is the **Brazilian E-Commerce Public Dataset by Olist**, containing **100,000+ orders** from **2016 to 2018** across multiple marketplaces in Brazil. It provides a comprehensive view of e-commerce transactions, including:

- **Orders:** Status, timestamps, and fulfillment details.
- **Payments:** Transaction amounts, types, and installment plans.
- **Shipping & Logistics:** Delivery performance, freight costs, and delays.
- **Customers:** Anonymized location data mapped to ZIP codes.
- **Products:** Categories, dimensions, and seller details.
- **Reviews:** Customer ratings and textual feedback.
- **Geolocation:** Mapping ZIP codes to latitude/longitude for spatial analysis.

The dataset was provided by **Olist**, a platform connecting small businesses to marketplaces, facilitating order fulfillment through logistics partners. Additionally, a **Marketing Funnel dataset** is available for analyzing the customer journey from ad exposure to purchase.

Use Case and Business Goals:

The dataset represents Brazilian e-commerce transactions from Olist, covering orders, payments, shipping, customer location, product details, and reviews. The main objectives for analyzing this dataset are:

1. Customer Segmentation – Identifying different types of buyers to optimize marketing strategies.
2. Customer Satisfaction Analysis – Understanding customer reviews and ratings to improve service.
3. Sales and Revenue Insights – Identifying top-selling products, high-revenue categories, and marketplace trends.

Target Description:

Based on the analyses performed, the target variables are:

1. **Product Performance Classification**
    - **Target Variable:** `product_performance` (Hit, Flop, Neutral)
    - **Type:** Categorical (3 classes)
    - **Definition:** A product's success in the market based on sales, reviews, repeat customers, and price.
    - **Goal:** Identify high-performing and underperforming products.

2. **Review Sentiment Analysis**
    - **Target Variable:** `review_sentiment` (Positive, Negative, Neutral)
    - **Type:** Categorical (3 classes)
    - **Definition:** Sentiment classification based on customer reviews.
    - **Goal:** Understand customer feedback and improve service.

3. **Customer Segmentation (Clustering - Unsupervised Learning)**
    - **No explicit target variable.**
    - **Features Used:** Recency, Frequency, Monetary Value
    - **Goal:** Identify different customer segments for targeted marketing.

Each target was either derived from existing data (`review_sentiment`) or engineered (`product_performance`, customer segmentation).

Feature Description:

Below are the most relevant features used in the analysis, categorized by whether they were directly available in the dataset or engineered for analysis.

1. Product Performance Classification

Feature	Type	Description	Source
`total_sales`	Continuous	Total revenue generated by the product.	**Engineered** (`price * units_sold`)
`units_sold`	Continuous	Number of units sold for the product.	**Directly Available**
`avg_review_score`	Continuous	Average customer rating (1-5 scale).	**Directly Available**
`repeat_customers`	Continuous	Number of customers who repurchased the product.	**Engineered** (Count of unique returning customers per product)
`original_price`	Continuous	Initial price of the product.	**Directly Available**

2. Review Sentiment Analysis

Feature	Type	Description	Source
`review_comment_message`	Text	Customer's written feedback about the product.	**Directly Available**
`review_sentiment`	Categorical (Positive, Neutral, Negative)	Sentiment derived from customer review text.	**Engineered** (NLP sentiment analysis)

3. Customer Segmentation (Clustering - RFM Analysis)

Feature	Type	Description	Source
`recency`	Continuous	Days since the last purchase.	**Engineered** (`latest_date - last_purchase_date`)
`frequency`	Continuous	Number of purchases made by the customer.	**Engineered** (Count of orders per customer)
`monetary_value`	Continuous	Total spending of the customer.	**Engineered** (Sum of purchase values per customer)

Model Development:

From this chapter onwards, we focus on the implementation of machine learning models based on the insights gained from Exploratory Data Analysis (EDA). This phase involves defining the problem statement, selecting relevant features, training models, and evaluating their performance. The documentation will also reference the key functions and code used in the implementation.

The model development process includes the following stages:

Preprocessing:

1. **Removed Outliers**

```python
#Removing all the outliers by using IQR that is inter quartile range for better analaysis of data
def remove_outliers_iqr(df):
    Q1 = df.quantile(0.25)  # First quartile
    Q3 = df.quantile(0.75)  # Third quartile
    IQR = Q3 - Q1  # Interquartile range
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[~((df < lower_bound) | (df > upper_bound)).any(axis=1)]
df_numeric = merged_df.select_dtypes(include=['number'])
df_cleaned = remove_outliers_iqr(df_numeric)
```

Interquartile Range (IQR) Method for Outlier Removal:
The IQR method identifies outliers by measuring the spread of the middle 50% of the data. It calculates Q1 (25th percentile) and Q3 (75th percentile) and defines outliers as values lying below Q1 - 1.5 × IQR or above Q3 + 1.5 × IQR.

2. Handling Null Values

```
df.isna().sum()
```

```
order_id                          0
customer_id                       0
order_status                      0
order_purchase_timestamp          0
order_approved_at               177
order_delivered_carrier_date   2086
order_delivered_customer_date  3421
order_estimated_delivery_date     0
customer_unique_id                0
customer_zip_code_prefix          0
customer_city                     0
customer_state                    0
review_id                       997
review_score                    997
review_comment_title         105154
review_comment_message        68898
review_creation_date            997
review_answer_timestamp         997
payment_sequential                3
payment_type                      3
payment_installments              3
payment_value                     3
order_item_id                   833
product_id                      833
seller_id                       833
shipping_limit_date             833
price                           833
freight_value                   833
product_category_name          2542
product_name_lenght            2542
product_description_lenght     2542
product_photos_qty             2542
product_weight_g                853
product_length_cm               853
product_height_cm               853
product_width_cm                853
seller_zip_code_prefix          833
seller_city                     833
seller_state                    833
dtype: int64
```

1. Drop Highly Missing Columns Rows: Columns which have more than 50% missing values (like `review_comment_message`) rows with null values were dropped.
2. Missing values in `order_delivered_customer_date` and `order_delivered_carrier_date` were imputed using the average delivery and carrier time for each `(product_id, customer_city)`, calculated from past orders. The imputed values were derived by adding the respective average time to `order_approved_at`.
3. Missing values in `product_category_name`, `product_name_length`, `product_description_length`, `product_photos_qty`, `product_weight_g`, `product_length_cm`, `product_height_cm`, and `product_width_cm` were replaced with their respective mean values.

3. Insights from DataSet:

A. Major Insights from the Correlation Heatmap:

1. `price` and `payment_value` (0.74) indicate that higher product prices lead to higher payment values.

2. `freight_value` and `price` (0.42) suggest that expensive products tend to have higher shipping costs.

3. `product_weight_g` has a notable correlation with `freight_value` (0.61), meaning heavier products incur higher shipping costs.

B. Monthly Sales Trend:

1. Overall Growth Trend:

- The sales trend shows consistent growth from late 2016 to early 2018, indicating increasing customer demand and business expansion.

2. Peak Sales Period:

- Sales peaked in late 2017 and early 2018, suggesting strong seasonal demand or promotional campaigns during this period.

3. Stable Sales in 2018:

- After the peak, sales remained relatively high and stable from January 2018 to mid-2018.

4. Sudden Drop in September 2018:

- A sharp decline in sales is observed in September 2018, which is due to data unavailability there were only 3 data points that too cancelled orders

C. Best Selling Products:

1. Leading Category - "bed_bath_table"

- This category has the highest number of orders, suggesting strong consumer demand for home essentials.

2. "health_beauty" is the second most popular category

- High sales indicate a growing interest in personal care and wellness products.

3. Moderate Sales in "sports_leisure," "furniture_decor," and "computers_accessories"

- These categories have similar order volumes, showing consistent demand across lifestyle and tech-related products.

4. "housewares" and "telephony" maintain steady sales

- Household essentials and electronic gadgets continue to be relevant among buyers.

D. Payment Insights:

1. Credit Card is the Most Preferred Payment Method

   a. A majority of customers prefer paying via credit cards, making it the dominant mode of payment.

2. "Boleto" is the Second Most Used Payment Method

   a. This suggests a significant portion of customers opt for bank payment slips, a popular method in Brazil.

3. Low Usage of Vouchers and Debit Cards
   a. These payment methods contribute to a smaller fraction of transactions, indicating they are not widely preferred by customers.

4. Negligible "Not Defined" Payments

    a. A very small portion of transactions fall under an undefined category, implying minimal data inconsistencies.

Model Building:

A. Problem Statement 1 -

Sentiment Analysis of Customer Reviews

❖ **Data Preparation & Preprocessing**

- ➢ Extracted relevant columns (`review_comment_title`, `review_comment_message`, `product_category_name`).
- ➢ Cleaned text: Converted to lowercase, removed punctuation, and tokenized.
- ➢ Removed stopwords using NLTK's Portuguese stopword list.
- ➢ Dropped duplicate and missing values.

❖ **Sentiment Analysis using LeIA**

- ➢ Applied `SentimentIntensityAnalyzer()` to classify sentiments as **Positive, Negative, or Neutral**.
- ➢ Used compound sentiment scores to determine polarity.

❖ **Feature Engineering**

- ➢ Extracted the most common **tokens (important words)** using `Counter`.
- ➢ Generated top **bigrams (two-word phrases)** using NLTK's `ngrams()`.
- ➢ Transformed text data into numerical features using **TF-IDF** (Term Frequency-Inverse Document Frequency).

❖ **Model Training & Evaluation**

- ➢ Encoded sentiment labels using `LabelEncoder()`.
- ➢ Split data into training (80%) and testing (20%) sets.
- ➢ Used **TF-IDF vectorization** (max 5000 features) to convert text into numerical format.
- ➢ Trained multiple classifiers:

- ➢ Naïve Bayes, Decision Tree, Random Forest, Logistic Regression, XGBoost, LGBM, SVM, AdaBoost, CatBoost, KNN, Gradient Boosting.
- ➢ Final Model: CatBoostClassifier achieved **90% accuracy** on both training and testing datasets.

❖ **Final Model Deployment**

- ➢ **CatBoostClassifier** was selected as the final model.
- ➢ Saved trained **model, TF-IDF vectorizer, and label encoder** using **Pickle** for future deployment of model.

Key Terms & Their Significance

❖ **LeIA (Lexicon-based Sentiment Analysis Tool)**
- ➢ A Portuguese language-specific sentiment analysis tool used to classify text polarity (Positive, Negative, Neutral).

❖ **TF-IDF (Term Frequency-Inverse Document Frequency)**
- ➢ A feature extraction technique to represent text numerically by emphasizing important words while reducing the weight of common ones.

❖ **Stopwords**
- ➢ Frequently occurring words (like "the", "is") that are removed to improve text processing efficiency.

❖ **SentimentIntensityAnalyzer**
- ➢ A tool from LeIA that assigns scores to text to determine the intensity of emotions in the review.

❖ **Bigrams (n-grams)**
- ➢ Two-word phrases extracted from text to capture common patterns and relationships.

❖ **CatBoostClassifier**
- ➢ A high-performance gradient boosting algorithm that efficiently handles categorical features and achieves high accuracy.

❖ **Label Encoding**

➢ Converts categorical sentiment labels (Positive, Negative, Neutral) into numerical values for machine learning models.

❖ **Pickle (Serialization)**

➢ Saves trained models and preprocessing tools to be reused without retraining.

Results:

	Model	Accuracy	Precision	Recall	F1-Score
0	GaussianNB	0.338682	0.537958	0.338682	0.363443
1	DecisionTreeClassifier	0.845845	0.847047	0.845845	0.845289
2	RandomForestClassifier	0.852722	0.858126	0.852722	0.844919
3	LogisticRegression	0.886533	0.887957	0.886533	0.884159
4	AdaBoostClassifier	0.626934	0.648855	0.626934	0.559243
5	XGBClassifier	0.899713	0.906463	0.899713	0.899361
6	LGBMClassifier	0.885387	0.891308	0.885387	0.884421
7	KNeighborsClassifier	0.503152	0.745484	0.503152	0.486426
8	GradientBoostingClassifier	0.862464	0.873675	0.862464	0.860690
9	SVC	0.903152	0.904255	0.903152	0.902616
10	CatBoostClassifier	0.903152	0.905303	0.903152	0.902568

B. Problem Statement 2 -

Customer Segmentation Analysis

1. Data Preparation & Preprocessing

- **Product Category Translation**: Converted **Portuguese product category names** to English using a dictionary mapping.
- **Outlier Removal**: Used **Interquartile Range (IQR) method** to eliminate extreme values in numerical data, ensuring robust clustering.
- **Total Price Calculation**: Computed **total spending per product** by summing product price and freight value, weighted by the order item quantity.

2. RFM Feature Extraction

- **Recency**: Number of days since the customer's last purchase.
- **Frequency**: Total number of orders placed by the customer.
- **Monetary Value**: Total amount spent by the customer.
- Extracted these features for each **customer_id** from transactional data.

3. Data Standardization

- Used **StandardScaler** to normalize RFM features (`recency`, `frequency`, `monetary_value`).
- Ensured all features were on the same scale to improve K-Means clustering performance.

4. K-Means Clustering for Customer Segmentation

- Applied **K-Means clustering** with **4 clusters**, segmenting customers based on purchasing behavior.
- Cluster assignments were determined using the standardized RFM values.

5. Assigning Customer Segments

- **VIP Buyers** – Frequent shoppers with high spending.
- **Loyal Buyers** – High-spending customers who shop less frequently.
- **Frequent Buyers** – Regular shoppers with moderate spending.
- **Budget Buyers** – Infrequent shoppers with low spending.
- Segments were assigned based on **recency, frequency, and monetary value scores** relative to the mean and standard deviation.
- **Cluster Sizes**: Analyzed the distribution of customers across segments to ensure meaningful segmentation.

6. Results & Business Impact

- Provided **actionable insights for targeted marketing** and **customer retention strategies**.

- Identified **high-value customers** (VIP & Loyal Buyers) for exclusive offers.
- Helped optimize marketing budgets by **focusing on profitable customer segments**.

Key Terms & Their Significance

Interquartile Range (IQR) Method

- A statistical technique used to detect and remove **outliers** in numerical data, ensuring **better clustering accuracy**.

RFM Analysis (Recency, Frequency, Monetary Value)

- A behavioral segmentation technique that categorizes customers based on their **purchase patterns**.

StandardScaler

- A preprocessing method that **normalizes data**, making sure all features have equal influence during clustering.

K-Means Clustering

- An **unsupervised machine learning algorithm** that groups customers into segments based on their purchase behavior.

C. Problem Statement 3 -

Product Performance Classification Model

1. Data Preparation & Feature Engineering

- Converted date columns to datetime format.
- Extracted key time-based features: day_of_week, hour, month, year, delivery_time.
- Computed total price per product including price and freight value.
- Filled missing values for review_score and price using median imputation.

2. Feature Creation & Aggregation

- Computed total sales as `price × order_item_id`.
- Identified repeat customers per product using customer purchase history.

- Aggregated product-level statistics:
    - total_sales, units_sold, avg_review_score, repeat_customers, original_price.

3. Feature Scaling & Normalization

- Applied StandardScaler() to normalize numerical features.

4. Performance Scoring & Product Status Classification

- Defined a performance score as the mean of `total_sales, units_sold, avg_review_score, repeat_customers`.
- Categorized product status using standard deviation and quantiles:
    - Hit: Performance score ≥ 75th percentile.
    - Flop: Performance score ≤ 25th percentile.
    - Neutral: Otherwise.

5. Model Training & Evaluation

- Encoded product status as 1 (Hit), 0 (Neutral), -1 (Flop).
- Split data into training (80%) and testing (20%) sets.
- Trained CatBoostClassifier with class weights to handle imbalanced data.
- Saved trained model and scaler using Pickle.

6. Multi-Model Performance Comparison

- **Traine**d multiple classifiers:
    - GaussianNB, Decision Tree, Random Forest, Logistic Regression, AdaBoost, LGBM, KNN, Gradient Boosting, SVC, CatBoost.
- Evaluated training and testing accuracy for each model.
- Generated a summary table with performance results.

Final Outcome:

- Best performing model selected based on accuracy scores.
- CatBoostClassifier achieved optimal performance for product status classification.

Results::

	Model	Accuracy	Precision	Recall	F1-Score
0	GaussianNB	0.781824	0.803979	0.781824	0.762252
1	DecisionTreeClassifier	0.992717	0.992716	0.992717	0.992715
2	RandomForestClassifier	0.994993	0.994993	0.994993	0.994993
3	LogisticRegression	0.962069	0.964047	0.962069	0.962013
4	AdaBoostClassifier	0.866029	0.886084	0.866029	0.862984
5	LGBMClassifier	0.994993	0.994996	0.994993	0.994992
6	KNeighborsClassifier	0.993021	0.993020	0.993021	0.993020
7	GradientBoostingClassifier	0.991504	0.991511	0.991504	0.991497
8	SVC	0.983159	0.983386	0.983159	0.983071
9	CatBoostClassifier	0.996207	0.996208	0.996207	0.996207