FINAL PROJECT REPORT

Tanishk Singh, Kritvirya Singh, Kanit Mann, Umesh Kumar Siyak

(Group 3)

University of Arizona

Course Number: INFO 523_SP2025

Dr. Kunal Arekar

May 2nd, 2025

Introduction

The Brazilian E-commerce Public Dataset by Olist contains information on 100,000 orders from 2016 to 2018 across multiple marketplaces in Brazil. This comprehensive real world dataset that includes order details, product information, customer data, and delivery timestamps, making it ideal for analyzing e-commerce operations.

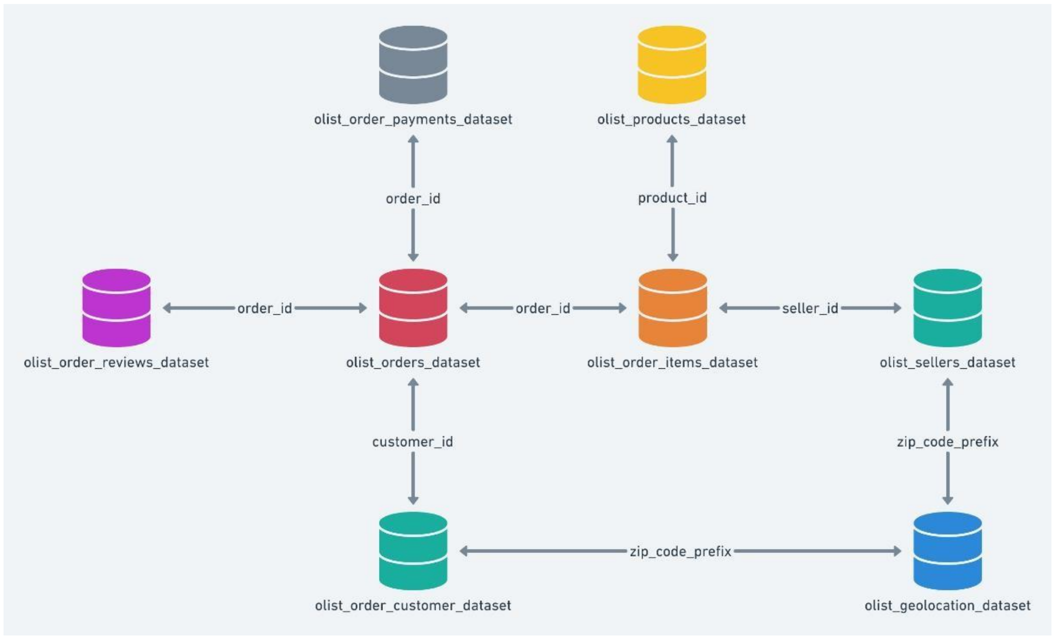


Figure 1 : Entity Diagram for the Dataset

- Customers Dataset

This dataset has information about the customer and its location.

- Geolocation Dataset

This dataset has information Brazilian zip codes and its lat/lng coordinates.

- Order Items Dataset

This dataset includes data about the items purchased within each order.

- Payments Dataset

This dataset includes data about the orders payment options.

- Order Reviews Dataset

This dataset includes data about the reviews made by the customers.

- Order Dataset

This is the core dataset. From each order you might find all other information.

- Products Dataset

This dataset includes data about the products sold by Olist.

- Sellers Dataset

This dataset includes data about the sellers that fulfilled orders made at Olist. Use it to find the seller location and to identify which seller fulfilled each product.

- Category Name Translation

Translates the product_category_name to english.

Question 1: Delivery Performance Analysis: Brazilian E-commerce

What factors most significantly influence delivery performance, and how can delivery performance be improved?

Introduction

The chosen dataset includes comprehensive order details, product information, customer data, and delivery timestamps, making it ideal for analyzing e-commerce operations. Delivery performance is a key area of improvement today in the world of e-commerce and logistics. A big amount of infrastructure and resources are directed towards this. Thus, generating insights around delivery performance became an obvious topic of interest for us. This analysis focuses on identifying the key factors influencing delivery performance and proposing strategies for improvement.

Approach / Methodology

- Data Preparation and Feature Engineering :

The analysis began with calculating delivery performance metrics from the original dataset –

```
orders_df['delivery_time'] =
(pd.to_datetime(orders_df['order_delivered_customer_date']) -

pd.to_datetime(orders_df['order_purchase_timestamp'])).dt.days

orders_df['delivery_delay'] =
(pd.to_datetime(orders_df['order_delivered_customer_date']) -

pd.to_datetime(orders_df['order_estimated_delivery_date'])).dt.days
orders_df['is_late'] = orders_df['delivery_delay'] > 0
```

Figure 2 : Data Preparation

We integrated data from multiple tables, including orders, order items, products, sellers, customers, geolocation, and payments. Key engineered features included shipping distance, product characteristics, seller experience, temporal factors, and payment details.

- Exploratory Analysis and Modelling :

Exploratory data analysis examined relationships between delivery time and potential predictors through visualizations and correlation analysis. We implemented multiple regression models to identify key factors: Linear Regression (baseline), Random Forest (for non-linear relationships), and Gradient Boosting (best performing model for feature importance ranking).

Data Analysis and Results

- Delivery Time Distribution

The average delivery time was approximately 12 days, with a median of 10 days. The distribution showed right-skewness with some extreme outliers exceeding 30 days. About 7% of all deliveries were late (exceeding the estimated delivery date).

Key Factors Influencing Delivery Performance:

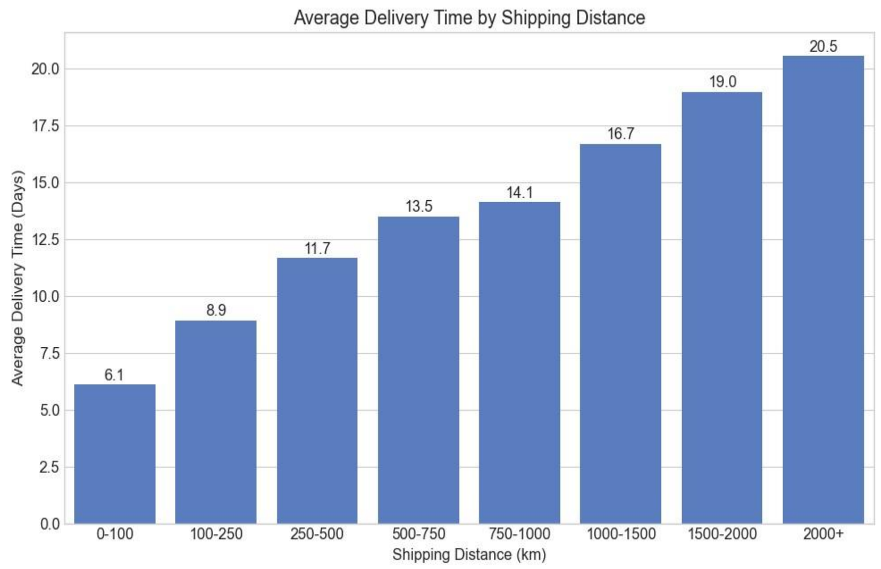


Figure 3: Average delivery time increases significantly with shipping distance

Based on feature importance from the Gradient Boosting model, the most significant factors were:

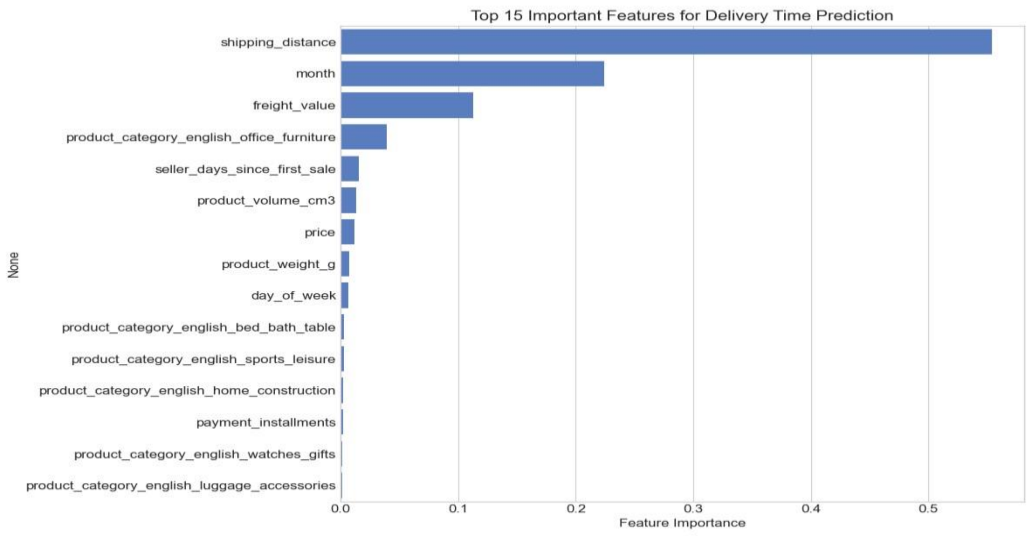


Figure 4: Feature importance ranking shows shipping distance as the dominant factor

1.**Shipping Distance**: Strong positive correlation with delivery time ○
Orders traveling over 1000km took on average 15 days to deliver ○
Orders under 100km were delivered in approximately 8 days

2.**Product Characteristics:**

- Heavier items (>10kg) took ~3 days longer than lightweight items (<500g) ○ Product category: Furniture, large appliances, and construction materials experienced the longest delivery times

3.Customer Location:

- States in northern regions had delivery times ~5 days longer than southeastern states ○ Metropolitan areas received faster deliveries than rural locations

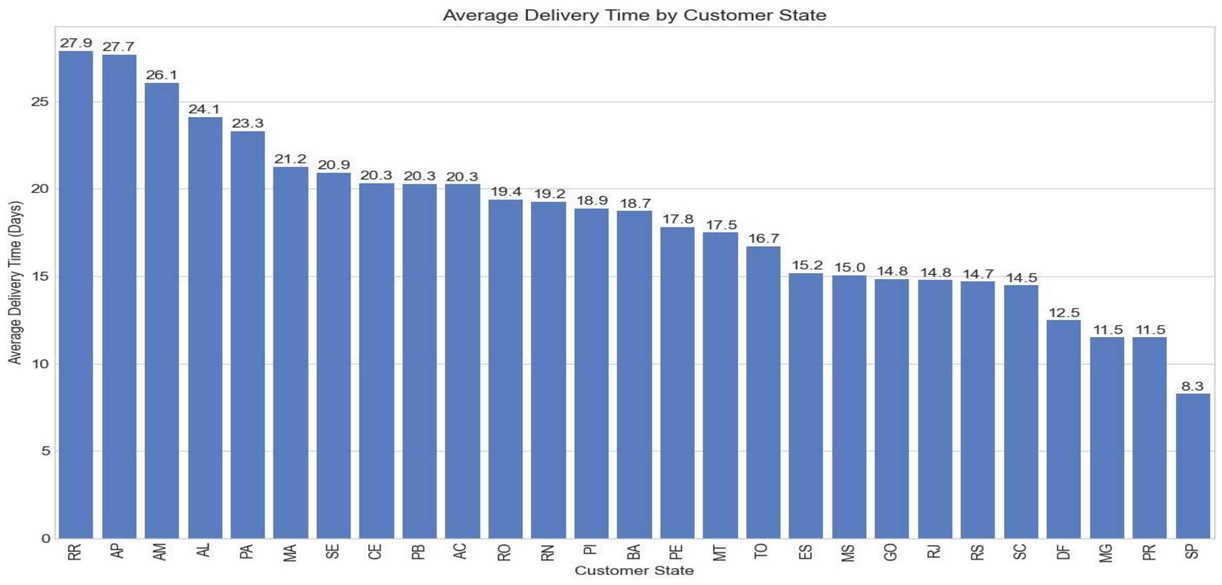


Figure 5: Significant variation in delivery times across Brazilian states

4.Order Timing:

○ Orders placed on weekends took 1-2 days longer than weekday orders
○ Seasonal effects showed slightly longer delivery times during holiday periods

5.Seller Experience:

- o More experienced sellers (>180 days on platform) delivered ~2 days faster than new sellers

The Gradient Boosting model achieved the best performance with an R² of 0.64, compared to 0.59 for Random Forest and 0.51 for Linear Regression. The mean absolute error was approximately 3.2 days.

Model	RMSE	MAE	R^2
Linear Regression	4.56	3.78	0.51
Random Forest	4.12	3.41	0.59
Gradient Boosting	3.87	3.20	0.64

Discussion

The analysis revealed that geographic factors, particularly shipping distance and customer location, are the most influential determinants of delivery performance in Brazilian e-commerce. The strong correlation between distance and delivery time suggests that logistics network optimization should be a primary focus for improvement.

Recommendations for Improving Delivery Performance:

1. **Optimize Distribution Network**:
   - o Establish regional fulfillment centers in northern and northeastern states where delivery times are longest
   - o Implement hub-and-spoke distribution models for remote areas
2. **Enhance Shipping Methods**:
   - o Offer tiered shipping options based on product characteristics ○ Partner with specialized carriers for large and heavy items
3. **Seller Performance Program**:
   - o Develop a mentoring system for new sellers to adopt best practices from experienced sellers
   - o Implement a seller rating system that rewards consistent, on-time deliveries
4. **Order Processing Efficiency**:

    - Optimize weekend staffing to reduce the delay in weekend orders ○ Implement automated order processing systems to reduce handling time
5. **Smart Delivery Estimation**:
    - Develop a more accurate delivery estimation algorithm that accounts for product category, distance, and region
    - Provide more conservative estimates for challenging delivery locations

A combined approach addressing both logistical infrastructure and operational processes would yield the greatest improvements in delivery performance, particularly for northern states and rural areas where delivery times significantly exceed the average.

Question 2: Customer Segmentation using K-Means Clustering

How can we segment customers based on their purchasing behaviours, and what distinctive patterns exist across these segments?

Introduction

This question was chosen as the dataset has rich features such as timestamps, order payment, states, cities, product details etc. which makes this dataset quite useful to perform RMF analysis, segmentation, analyse customer behaviour and plan marketing strategies accordingly. This information can be then implemented by the company to engage the customers based on their clustering profiles.

Approach and Methodology

Our customer segmentation analysis followed a structured approach that combined RFM (Recency, Frequency, Monetary) analysis with additional behavioural metrics:

- Dataset Preparation and Feature Engineering

We created a customer-level dataset from the transaction-level Olist data by aggregating and deriving various relevant features into a final customer dataset and created then below features as required by our RFM and customer behaviour analysis:

1. **Recency**: Days since customer's last purchase (derived from order_purchase_timestamp)
2. **Frequency**: Number of orders placed by each customer (order_count)
3. **Monetary Value**: Total spend per customer and average order value
4. **Product Category Behaviour**: Most frequently purchased category and category diversity
5. **Exploration Indicators**: Binary features indicating repeat customers and category explorers

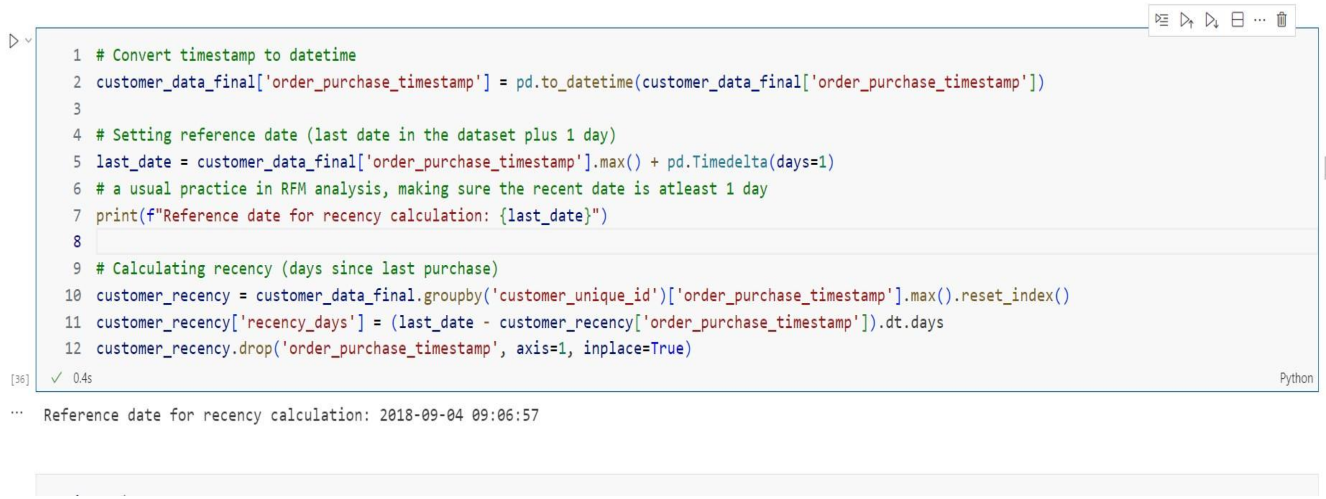
The feature engineering process involved:
- Grouping transactions by customer_unique_id to analyze individual customer behavior
- Calculating recency using the difference between last purchase date and reference date
- Creating binary indicators for repeat purchases and multi-category exploration - Capturing geographic information for regional pattern analysis

- Clustering Methodology

We employed K-means clustering with the following approach:

1. **Feature Selection**:

We created key features that capture different aspects of customer behaviour:
- Total spend (monetary)
- Recency days (recency)



```python
# Convert timestamp to datetime
customer_data_final['order_purchase_timestamp'] = pd.to_datetime(customer_data_final['order_purchase_timestamp'])

# Setting reference date (last date in the dataset plus 1 day)
last_date = customer_data_final['order_purchase_timestamp'].max() + pd.Timedelta(days=1)
# a usual practice in RFM analysis, making sure the recent date is atleast 1 day
print(f"Reference date for recency calculation: {last_date}")

# Calculating recency (days since last purchase)
customer_recency = customer_data_final.groupby('customer_unique_id')['order_purchase_timestamp'].max().reset_index()
customer_recency['recency_days'] = (last_date - customer_recency['order_purchase_timestamp']).dt.days
customer_recency.drop('order_purchase_timestamp', axis=1, inplace=True)
```

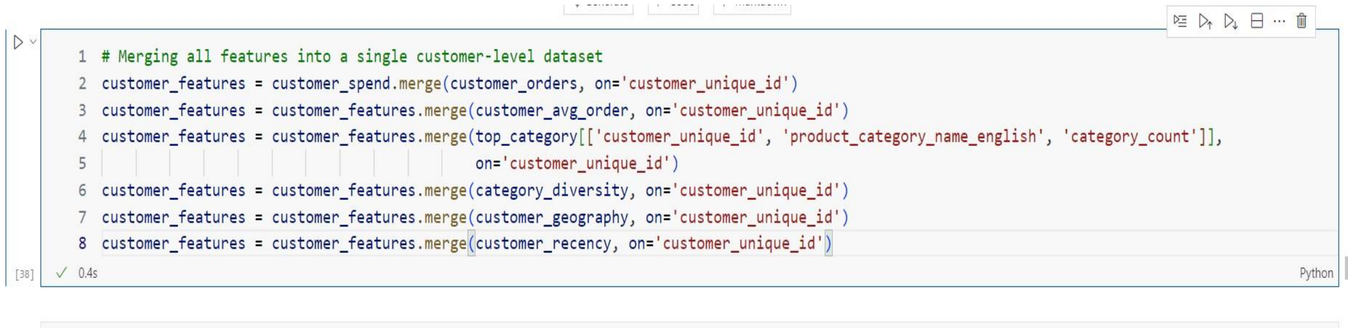
[36] ✓ 0.4s Python

```
Reference date for recency calculation: 2018-09-04 09:06:57
```

Figure 5 : Creating recency feature

- Average order value (monetary) - Repeat customer status (binary)
- Category explorer status (binary)

All the above features were then merged to create a final customer features dataset :



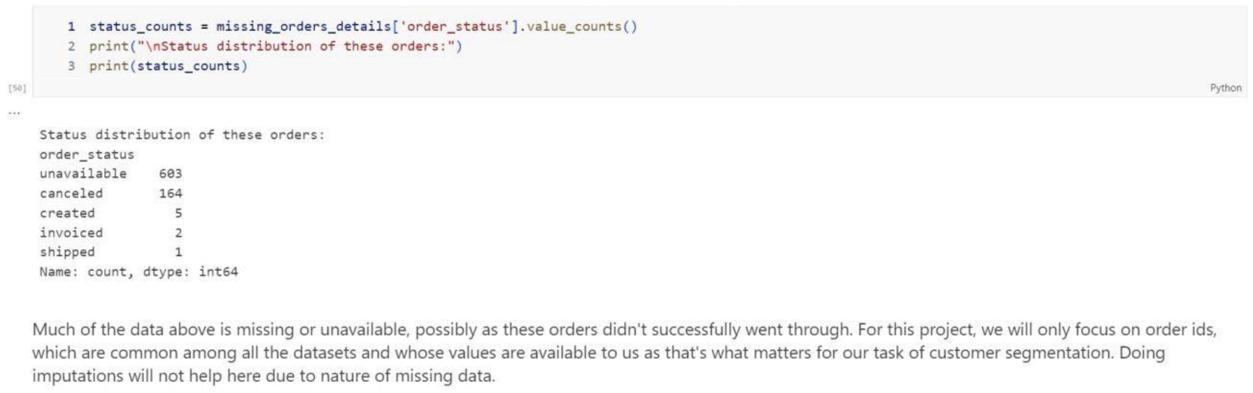
```python
# Merging all features into a single customer-level dataset
customer_features = customer_spend.merge(customer_orders, on='customer_unique_id')
customer_features = customer_features.merge(customer_avg_order, on='customer_unique_id')
customer_features = customer_features.merge(top_category[['customer_unique_id', 'product_category_name_english', 'category_count']],
                                            on='customer_unique_id')
customer_features = customer_features.merge(category_diversity, on='customer_unique_id')
customer_features = customer_features.merge(customer_geography, on='customer_unique_id')
customer_features = customer_features.merge(customer_recency, on='customer_unique_id')
```

[38] ✓ 0.4s Python

Figure 6: Features Dataset

2. **Data Preprocessing**:

- Created synthetic row for one missing order.
- Unavailable data was ignored while joining the datasets as this was from cancelled/unsuccessful orders.



```
status_counts = missing_orders_details['order_status'].value_counts()
print("\nStatus distribution of these orders:")
print(status_counts)
```

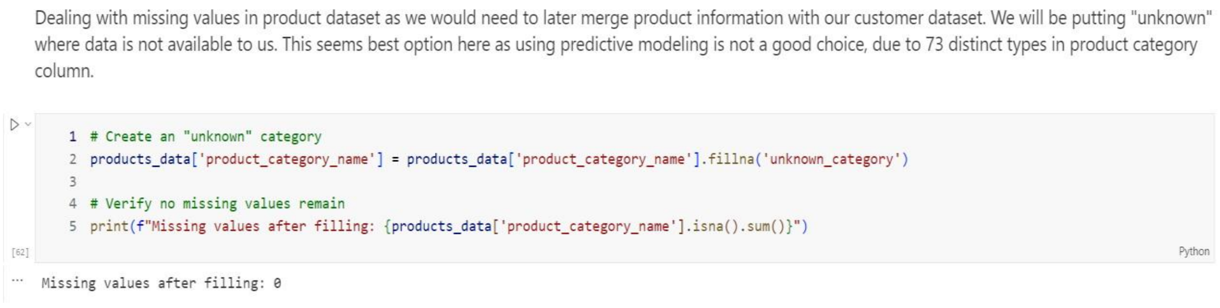
[58]

```
Status distribution of these orders:
order_status
unavailable    603
canceled       164
created          5
invoiced         2
shipped          1
Name: count, dtype: int64
```

Much of the data above is missing or unavailable, possibly as these orders didn't successfully went through. For this project, we will only focus on order ids, which are common among all the datasets and whose values are available to us as that's what matters for our task of customer segmentation. Doing imputations will not help here due to nature of missing data.

Figure 8 : Handling Unavailable Values

- Missing values were replaced by "unknown"



Dealing with missing values in product dataset as we would need to later merge product information with our customer dataset. We will be putting "unknown" where data is not available to us. This seems best option here as using predictive modeling is not a good choice, due to 73 distinct types in product category column.

```
# Create an "unknown" category
products_data['product_category_name'] = products_data['product_category_name'].fillna('unknown_category')

# Verify no missing values remain
print(f"Missing values after filling: {products_data['product_category_name'].isna().sum()}")
```

[62]

```
Missing values after filling: 0
```

Figure 9: Handling Missing Values

- EDA was performed on merged features dataset to identify outliers and scaling needs
- Handled outliers using the IQR method to prevent extreme values from distorting clusters
- Applied standardization to continuous variables to ensure equal weight in clustering
- Created binary encodings for low-variance features like repeat purchases and category exploration

Data Analysis and Results

- K-Means Clustering Code:

```
n_clusters = 5  #usually five clusters are optimum to get
valuable segemntation patterns
```

```
 kmeans = KMeans(n_clusters=n_clusters,
random_state=42, n_init=10) cluster_labels =
kmeans.fit_predict(clustering_data) cluster_labels
```

- Generating Cluster Statistics :

```
    cluster_stats = customer_features.groupby('cluster').agg({
   'total_spend': ['mean', 'median'],
   'order_count': ['mean', 'median'],
   'recency_days': ['mean', 'median'],
   'avg_order_value': ['mean', 'median'],
   'category_diversity': ['mean', 'median'],
   'is_repeat_customer': ['mean'],
   'is_category_explorer': ['mean'],
   'customer_unique_id': 'count'
}).reset_index()
 print("\nCluster
Statistics:") cluster_stats
```

The clustering analysis identified five distinct customer segments, each with unique behavioural patterns and characteristics:

- **Cluster Profiles:**

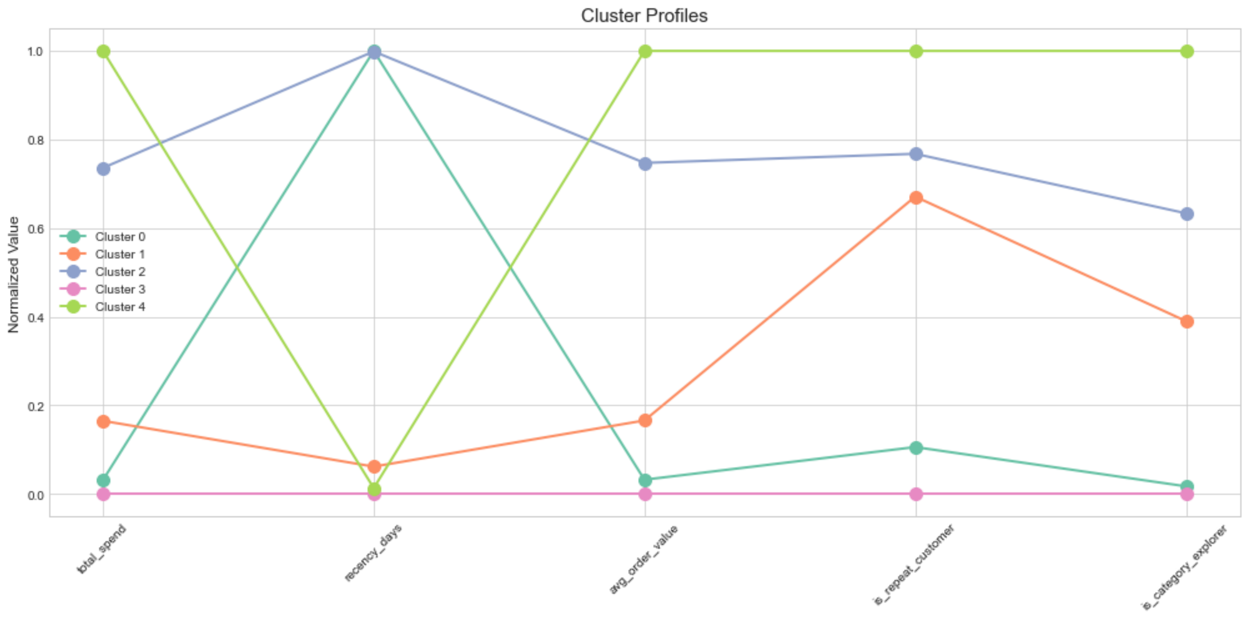


Figure 10: Cluster Profiles

1) Cluster 0: "Dormant Mid-Value Customers" (25.51%)

This segment represents customers who made moderate-value purchases but haven't returned in over a year:

- Average total spend: $92.25
- Average order count: 1.02
- Average recency (days): 423.31
- Average order value: $90.63
- Repeat customers: 2.1%
- Category explorers: 1.1%
- Primary categories: watches & gifts, bed & bath, sports & leisure

2) Cluster 1: "Recent Mid-Value Customers" (20.39%)

These customers made relatively recent purchases with mid-range values:

- Average total spend: $186.69
- Average order count: 1.05
- Average recency (days): 165.94
- Average order value: $180.41
- Repeat customers: 4.6%
- Category explorers: 3.4%
- Primary categories: bed & bath, furniture & decor, health & beauty

3) Cluster 2: "Dormant High-Value Customers" (7.59%)

This segment made significant high-value purchases but hasn't returned in over a year:

- Average total spend: $590.33
- Average order count: 1.06
- Average recency (days): 423.00
- Average order value: $570.61
- Repeat customers: 5.0%
- Category explorers: 4.9%
- Primary categories: bed & bath, furniture & decor, health & beauty

Cluster 3: "Recent Low-Value Customers" (36.07%)

The largest segment, these customers made a single low-value purchase relatively recently:

- Average total spend: $70.08
- Average order count: 1.02

- Average recency (days): 149.05
- Average order value: $69.12
- Repeat customers: 1.6%
- Category explorers: 1.0%
- Primary categories: health & beauty, bed & bath, sports & leisure

1) Cluster 4: "Recent High-Value Explorers" (10.45%) The most valuable and engaged customer segment:

- Average total spend: $776.99
- Average order count: 1.07
- Average recency (days): 152.67
- Average order value: $740.27
- Repeat customers: 6.0
- Category explorers: 7.2%
- Most diverse category preferences, led by health & beauty

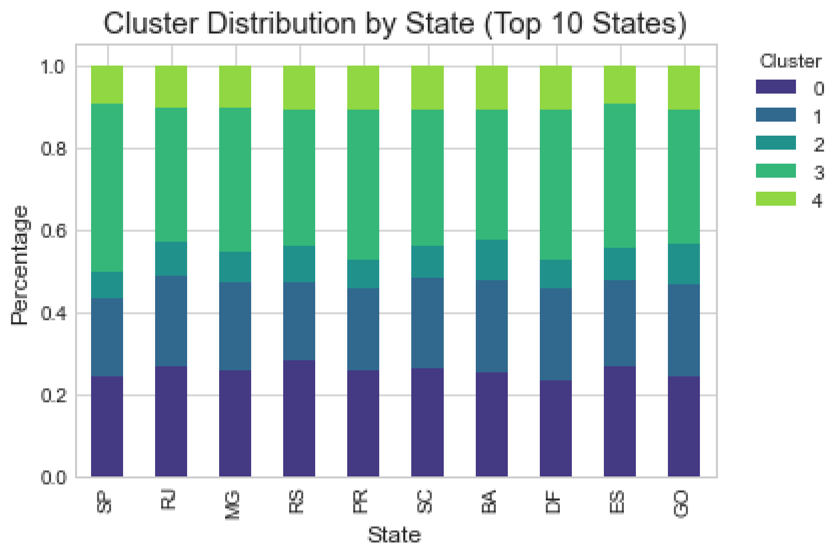


Figure 11: Statewide Cluster Distribution

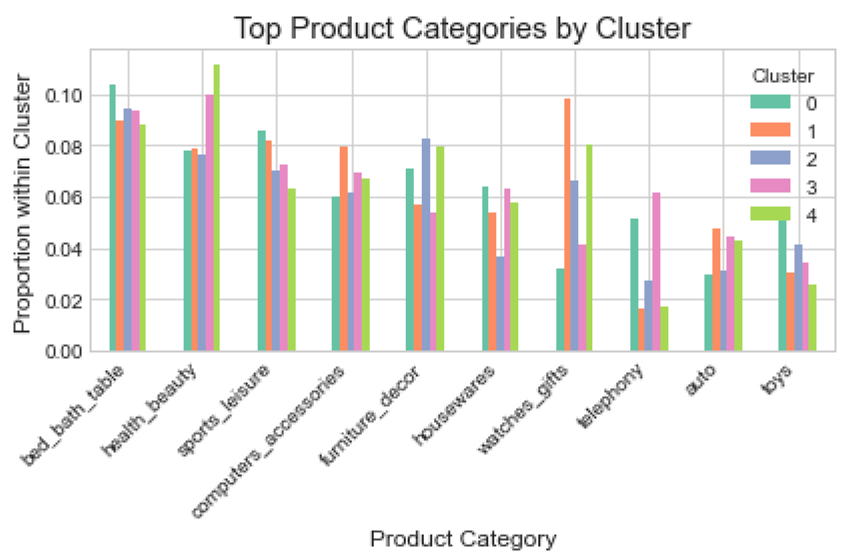


Figure 12: Cluster wise product categories

Discussion

The clustering analysis revealed several key insights about Olist's customer base that have significant implications for marketing strategy:

- **Key Findings**

1) Low Repeat Purchase Rate: Across all segments, the repeat purchase rate is extremely low (1.6%-6%), suggesting a critical need for retention strategies.

2) Recency Patterns: Customers fall into two distinct recency groups - those who haven't purchased in over a year (Clusters 0 and 2) and those who purchased within the last 5-6 months (Clusters 1, 3, and 4).

3) Value Segmentation: Clear differentiation exists between high-value (Clusters 2 and 4), mid-value (Clusters 0 and 1), and low-value customers (Cluster 3).

4) Geographic Concentration: Sao Paulo dominates across all segments, but regional variations exist that provide opportunities for targeted marketing.

5) Category Preferences: While some categories like bed & bath and health & beauty are popular across segments, distinct category preferences emerge within clusters.

Speculations:
In the dataset Sao Paulo dominates across all clusters as it is one of the most populous cities in the world with great architecture and flourishing tourism, which makes it put it perfect for e-commerce growth.

The results from the dataset reflect Brazilian e-commerce market. Health and beauty appear as a constant pattern across all clusters whether high value or less value customers in terms of purchasing power, suggesting health and beauty to be common choice for all. This makes sense given the vibrant demography of Sao Paulo, a city from where the customer orders dominates the dataset.

Q3. Time Series Analysis for Sales Forecasting

Can we develop a reliable time series model to forecast future sales volumes and identify seasonal patterns in the Brazilian e-commerce market?

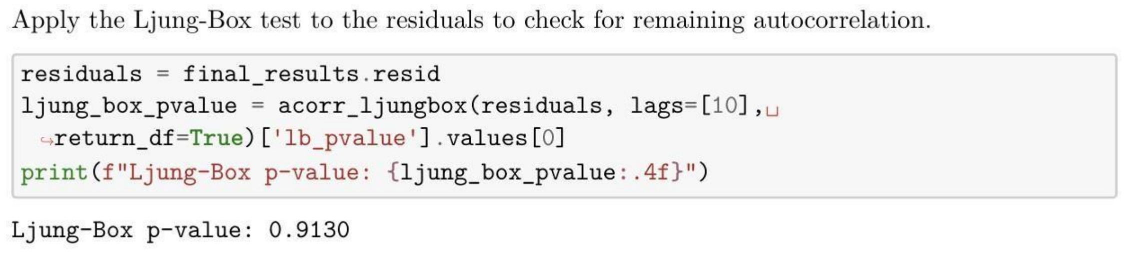
Introduction

For this analysis, we pick olist_orders_dataset and olist_order_items_dataset, using the order timestamps and prices to develop a daily sales time series.
This question interests us because accurate sales forecasting is critical for inventory planning, resource allocation, and customer satisfaction in e-commerce. By identifying seasonal patterns and building predictive models, companies can optimize operations and reduce costs.

Approach / Methodology

The dataset is filtered to include only 'delivered' orders and aggregated daily order counts. This provides a clean and regular time series with which to work. SARIMA modelling is then applied to handle trend and seasonality followed by hyperparameter tuning using AIC (Akaike Information Criterion) to get the best configuration. SARIMA is then compared with Prophet (Facebook Prophet, a modern time series forecasting tool designed to handle seasonality and holidays), while also including the Seasonal Naïve Model as a benchmark, which repeats the sales pattern from the previous week, to provide a baseline for evaluation. The models were evaluated using MAE, RMSE, and MAPE. Ljung-Box test was carried out to check for residuals for autocorrelation.
Data Analysis and Results

Residual Diagnostic:


Apply the Ljung-Box test to the residuals to check for remaining autocorrelation.

```
residuals = final_results.resid
ljung_box_pvalue = acorr_ljungbox(residuals, lags=[10],␣
 ↪return_df=True)['lb_pvalue'].values[0]
print(f"Ljung-Box p-value: {ljung_box_pvalue:.4f}")
```

```
Ljung-Box p-value: 0.9130
```

Figure 13 : Residual Diagnostic

This value indicates, there is no significant auto correlation in residuals. **Sarima Forecast:**

Visualize the SARIMA forecast alongside the actual sales data.

```
plt.figure(figsize=(14, 6))
plt.plot(train.index, train['order_count'], label='Train')
plt.plot(test.index, test['order_count'], label='Test', color='gray')
plt.plot(predicted.index, predicted, label='Forecast', color='green')
plt.fill_between(conf_int.index, conf_int.iloc[:, 0], conf_int.iloc[:, 1],␣
 ↪color='lightgreen', alpha=0.3)
plt.title('Order Count Forecast with Confidence Interval')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

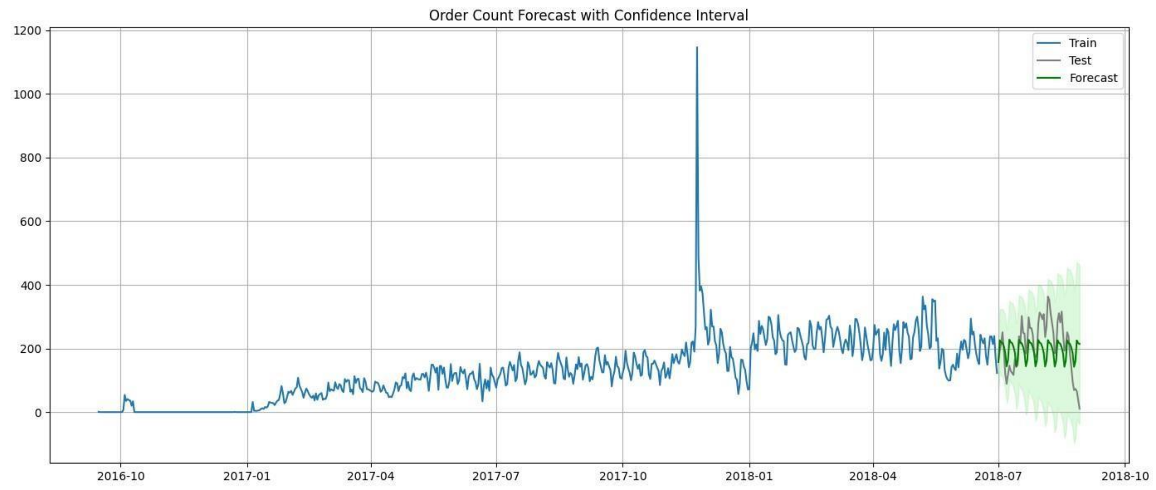


Figure 14 : Forecast Visualisation

Prophet Forecast vs Actual:

```
plt.figure(figsize=(14,6))
plt.plot(train.index, train['order_count'], label='Train')
plt.plot(test.index, test['order_count'], label='Test', color='gray')
plt.plot(forecast_test.index, forecast_test['yhat'], label='Prophet Forecast',␣
 ↪color='purple')
plt.title('Prophet Forecast vs Actual')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figure 15 : Comparing Forecast with Actual

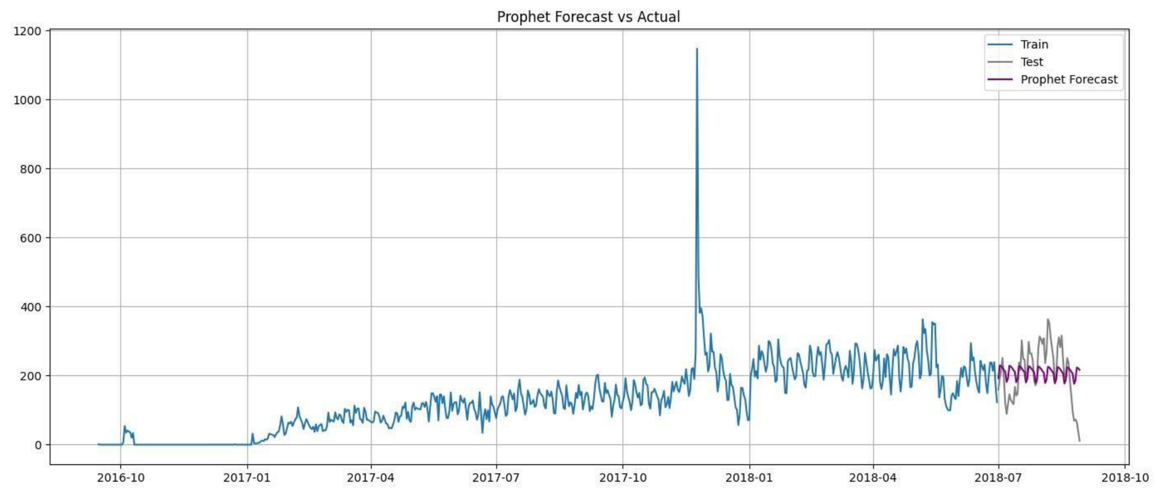


Figure 16: Time-series graph, forecast vs actual

Model	MAE	RMSE	MAPE
SARIMA	63.84%	77.12%	71.21%
Seasonal Naive	58.57%	68.67%	51.57%
Prophet	62.42%	76.73%	73.63%

Discussion

The analysis revealed weekly seasonality and an upward long-term trend in the Brazilian ecommerce sales data. There was also a sharp sales spike near the end of 2017, likely due to a major event or holiday. Both SARIMA and Prophet models were able to capture this pattern, as can be seen in the plots. The residual component displayed mostly random noise, with some extreme outliers during the sales peak period, which suggests that models may have struggled with irregular sales.
However, on being evaluated using error metrics, the seasonal naive benchmark outperformed both SARIMA and Prophet on all three metrics (MAE, RMSE, MAPE). Despite their complexity, the advanced models did not yet meaningfully improve on a simple repeat-last-week strategy. A reason behind this may be that, naïve model fully leverages the strong and stable weekly seasonality seen in the data, while SARIMA and Prophet require further careful parameterization and additional inputs to exceed their performances. Another reason could be the short test window (60 days), which may limit the models' ability to generalize. The residual diagnostics for SARIMA showed no significant autocorrelation, indicating that the model fits the available data well, even if its predictive performance was limited. The Ljung-Box test (residual diagnostic for SARIMA) showed no significant autocorrelation ($p \approx 0.913$), which means the model adequately captured the patterns in training data.
Future work should incorporate holiday effects, and external regressors (like marketing campaigns), and potentially explore machine-learning approaches to improve predictive performance.

References

1. Olist. (2018). Brazilian E-Commerce Public Dataset by Olist. Retrieved from https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

2. Singh, T. (2025). UofA_DataMining_Final_Project: Lets' do it folks! . GitHub. https://github.com/Tanishk-Singh/UofA_DataMining_Final_Project

3. Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd ed). OTexts. Retrieved from https://otexts.com/fpp2/

4. Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. The American Statistician, 72(1), 37–45. DOI: [10.1080/00031305.2017.1380080](https://doi.org/10.1080/00031305.2017.1380080)
(for the Prophet model)

5. Box, G. E. P., & Jenkins, G. M. (1976). Time Series Analysis: Forecasting and Control (Revised ed.). Holden-Day.
(for the SARIMA model methodology)

6. Ljung, G. M., & Box, G. E. P. (1978). On a Measure of Lack of Fit in Time Series Models. Biometrika, 65(2), 297–303.
(for the Ljung-Box residual test)

7. Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and Statistical Modeling with Python. Proceedings of the 9th Python in Science Conference.

8. Facebook Prophet Documentation. Retrieved from https://facebook.github.io/prophet/