UNIVERSITI MALAYA

Faculty of Computer Science and Information Technology

WIE3007

Data Mining & Warehousing

Individual Assignment 1

Leverage Featuretools To Perform Automated Feature Engineering On E-Commerce Data

Lecturer:

Prof. Dr. Teh Ying Wah

Name: JANICE CHONG SEE WAI

Matric Number: S2132420

Semester 1 2023/2024

Table of Contents

Link to GitHub: https://github.com/Janice-Chong/featureTools.git

1.0 Dataset Examination

1.1 About the Dataset

The dataset, 'Brazilian E-Commerce Public Dataset', is obtained from Kaggle.
It is a publicly available dataset from Brazil's e-commerce sector, encompassing information on 100,000 orders that span between 2016 and 2018, originating from multiple marketplaces situated within Brazil, across 27 states. This dataset is rich in various dimensions, including customer data, order data, payment information, product details, and seller information.

Link to dataset: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

1.2 Identify Possible Entities

a. Order table

order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date
e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	2017-10-02 11:07:15	2017-10-04 19:55:00
53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	2018-07-26 03:24:27	2018-07-26 14:31:00
47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49	2018-08-08 08:55:23	2018-08-08 13:50:00
949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	2017-11-18 19:45:59	2017-11-22 13:39:59
ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39	2018-02-13 22:20:29	2018-02-14 19:46:34

order_delivered_customer_date	order_estimated_delivery_date	product_id	seller_id	shipping_limit_date	price
2017-10-10 21:25:13	2017-10-18 00:00:00	87285b34884572647811a353c7ac498a	3504c0cb71d7fa48d967e0e4c94d59d9	2017-10-06 11:07:15	29.99
2018-08-07 15:27:45	2018-08-13 00:00:00	595fac2a385ac33a80bd5114aec74eb8	289cdb325fb7e7f891c38608bf9e0962	2018-07-30 03:24:27	118.70
2018-08-17 18:06:29	2018-09-04 00:00:00	aa4383b373c6aca5d8797843e5594415	4869f7a5dfa277a7dca6462dcf3b52b2	2018-08-13 08:55:23	159.90
2017-12-02 00:28:42	2017-12-15 00:00:00	d0b61bfb1de832b15ba9d266ca96e5b0	66922902710d126a0e7d26b0e3805106	2017-11-23 19:45:59	45.00
2018-02-16 18:17:02	2018-02-26 00:00:00	65266b2da20d04dbe00c5c2d3bb7859e	2c9e548be18521d1c43cde1c582c6de8	2018-02-19 20:31:37	19.90

freight_value	payment_id
8.72	8ce37bad-7838-4a1d-a954-611b5a6481b4
22.76	102aec25-fdb0-498a-8930-48eb50cbc0ce
19.22	f49030d6-53fe-48d6-b85e-93351d439ab8
27.20	ec1853bd-71d0-402c-a0cf-8491bd977261
8.72	75d27dac-6464-4e4f-b3f2-d3dee2040d38

b. Product Table

	product_id	product_category_name	product_name_lenght	product_description_lenght	product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_cm
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	287.0	1.0	225.0	16.0	10.0	14.0
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	276.0	1.0	1000.0	30.0	18.0	20.0
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	250.0	1.0	154.0	18.0	9.0	15.0
3	cef67bcfe19066a932b7673e239eb23d	bebes	27.0	261.0	1.0	371.0	26.0	4.0	26.0
4	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	37.0	402.0	4.0	625.0	20.0	17.0	13.0

c. Customer Table

	customer_id	customer_zip_code_prefix	customer_city	customer_state
0	06b8999e2fba1a1fbc88172c00ba8bc7	14409	franca	SP
1	18955e83d337fd6b2def6b18a428ac77	9790	sao bernardo do campo	SP
2	4e7b3e00288586ebd08712fdd0374a03	1151	sao paulo	SP
3	b2b6027bc5c5109e529d4dc6358b12c3	8775	mogi das cruzes	SP
4	4f2d8ab171c80ec8364f7c12e35b23ad	13056	campinas	SP

d. Payment Table

	payment_type	payment_installments	payment_value	payment_id
0	credit_card	8	99.33	5d835bad-3a14-4fba-a27d-f95743a816fd
1	credit_card	1	24.39	8c28d9cb-2b3a-4747-b65a-91383dba567e
2	credit_card	1	65.71	caa1a7b8-2853-4d27-9635-a9245cb6cbc0
3	credit_card	8	107.78	4763f5ae-b018-4608-87c0-7cd75eb443b3
4	credit_card	2	128.45	a513debc-f26e-44b1-91c0-655a51e2660b

e. Seller Table

	seller_id	seller_zip_code_prefix	seller_city	seller_state
0	3442f8959a84dea7ee197c632cb2df15	13023	campinas	SP
1	d1b65fc7debc3361ea86b5f14c68d2e2	13844	mogi guacu	SP
2	ce3ad9de960102d0677a81f5d0bb7b2d	20031	rio de janeiro	RJ
3	c0f3eea2e14555b6faeea3dd58c1b1c3	4195	sao paulo	SP
4	51a04a8a6bdcb23deccc82b0b80742cf	12914	braganca paulista	SP

1.3 Relationships

a. Order has many-to-one relationship with Product
   - A single order can be associated with one and only one product
   - A product can consist of one or many orders

b. Order has many-to-one relationship with Customer
   - A single order can exclusively belong to one customer.
   - A customer can have one or many orders

c. Order has one-to-one relationship with Payment
   - A single order can have one and only one payment details
   - A payment can only belong to one specific order

d. Order has many-to-one relationship with Seller

- A single order can only be associated to one and only one seller
- A seller can have one or many orders

e. Product has one-to-many relationship with Order
- A product can consist of one or many orders
- An order can consist of one and only one product

f. Customer has one-to-many relationship with Order
- A customer can have one or many order
- An order can exclusively belong to one customer

g. Payment has one-to-one relationship with Order
- A single payment can only belong to one specific order
- An order can have one and only one payment

h. Seller has one-to-many relationship with Order
- A seller can have one or many orders
- An order can only be associated to one and only one order

1.4 Hierarchical Structure

a. Order

Order Information > Order Cost

Order Information	Order Cost
• order_status • order_purchase_timestamp • order_approved_at • order_delivered_carrier_date • order_delivered_customer_date • order_estimated_delivery_date • shipping_limit_date	• price • freight_value

b. Product

Product Information > Product Details > Product Dimensions

Product Information	Product Details	Production Dimensions
• product_category_name	• product_name_length • product_photos_qty	• product_weight_g • product_length_cm • product_height_cm • product_width_cm

c. Customer

customer_state > customer_city > customer_zip_code_prefix

d. Payment

payment_type > payment_installments > payment_value

e. Seller

seller_state > seller_city > seller_zip_code_prefix

2.0 FeatureTools

2.1 Data Cleaning and Preparation

```
# Function to generate unique IDs using UUID
def generate_unique_id():
    return str(uuid.uuid4())

# Add a unique_id column to DataFrame
order_payments_df['payment_id'] = [generate_unique_id() for _ in range(len(order_payments_df))]
```

Payment does not have a unique ID (Primary Key). Thus, here, we will generate a unique ID for 'payment_df'.

```
# Merge 'payment_id' from order_payments_df to orders_df based on 'order_id'
orders_df = pd.merge(orders_df, order_payments_df[['order_id', 'payment_id']], on='order_id', how='inner')
```

Add foreign key, 'payment_id' into the fact table, 'orders_df'.

```
customer_df = customer_df.drop_duplicates(subset=['customer_id'])
order_items_prod_df = order_items_df.drop_duplicates(subset=['product_id'])
order_payments_df = order_payments_df.drop_duplicates(subset=['payment_id'])
orders_df = orders_df.drop_duplicates(subset=['order_id'])
products_df = products_df.drop_duplicates(subset=['product_id'])
```

Remove duplicated rows.

```
customer_df = customer_df.dropna()
order_payments_df = order_payments_df.dropna()
orders_df = orders_df.dropna()
products_df = products_df.dropna()
seller_df = seller_df.dropna()
```

Remove null values.

2.2 Create and Define Entity Set

```
es = ft.EntitySet(id= 'ecommerce_data')
```

Create EntitySet object named 'ecommerce_data' using Featuretools (ft).

```
# Define entities

es.add_dataframe(dataframe_name='customer_en', dataframe=customer_df, index='customer_id')
es.add_dataframe(dataframe_name='order_payments_en', dataframe=order_payments_df, index='payment_id')
es.add_dataframe(dataframe_name='orders_en', dataframe=orders_df, index='order_id')
es.add_dataframe(dataframe_name='products_en', dataframe=products_df, index='product_id')
es.add_dataframe(dataframe_name='seller_en', dataframe=seller_df, index='seller_id')
```

Add multiple dataframes *(customer_df, order_payments_df, orders_df, products_df, seller_df)* to the EntitySet 'ecommerce_data'. Each dataframe represents a specific entity *(customer_en, order_payments_ en, orders_ en, products_ en, seller_ en)* in the dataset and associate each dataframe with a unique index column *(customer_id, order_payments_ id, orders_ id, products_ id, seller_ id)*.

2.3 Define and Create Relationships Between Entities

```
# Define relationships
relationships = [('customer_en', 'customer_id', 'orders_en', 'customer_id'),
                 ('order_payments_en', 'payment_id', 'orders_en', 'payment_id'),
                 ('products_en', 'product_id', 'orders_en','product_id'),
                 ('seller_en', 'seller_id', 'orders_en', 'seller_id')]

# Add relationships to EntitySet
for relationship in relationships:
    es.add_relationship(relationship[0], relationship[1], relationship[2], relationship[3])
```

Defined a list of relationships in the 'relationships' variable. Each relationship is a tuple of four elements, representing the two entities involved and the respective index columns that establish the relationship. For example:

('customer_en', 'customer_id', 'orders_en', 'customer_id')

Entity involved: 'customer_en' and 'orders_en'

Index column: 'customer_id'

Then, iterate through the list of relationships and add them to the EntitySet using the 'es.add_relationship()' method. This step establishes the connections between the entities in the EntitySet.

2.4 Perform Deep Feature Synthesis (DFS)

```
feature_matrix, feature_defs = ft.dfs(entityset=es,
                                      target_dataframe_name='orders_en')
```

Generate a feature matrix, 'feature_matrix' variable, and a set of feature definitions, 'feature_defs', from the EntitySet (es). Target dataframe is the Fact Table, 'orders_en'.

2.5 Save as CSV File

```
feature_matrix.to_csv('feature_matrix2.csv', index=False)
```

Save the 'feature_matrix' as a CSV file called feature_matrix2.csv using '.to_csv()'.

2.6 New Features Generated

```
for column in feature_matrix.columns:
    print(column)
```

```
order_status
price
freight_value
DAY(order_approved_at)
DAY(order_delivered_carrier_date)
DAY(order_delivered_customer_date)
DAY(order_estimated_delivery_date)
DAY(order_purchase_timestamp)
DAY(shipping_limit_date)
MONTH(order_approved_at)
MONTH(order_delivered_carrier_date)
MONTH(order_delivered_customer_date)
MONTH(order_estimated_delivery_date)
MONTH(order_purchase_timestamp)
MONTH(shipping_limit_date)
WEEKDAY(order_approved_at)
WEEKDAY(order_delivered_carrier_date)
WEEKDAY(order_delivered_customer_date)
WEEKDAY(order_estimated_delivery_date)
WEEKDAY(order_purchase_timestamp)
WEEKDAY(shipping_limit_date)
YEAR(order_approved_at)
YEAR(order_delivered_carrier_date)
YEAR(order_delivered_customer_date)
YEAR(order_estimated_delivery_date)
YEAR(order_purchase_timestamp)
YEAR(shipping_limit_date)
customer_en.customer_zip_code_prefix
customer_en.customer_city
customer_en.customer_state
order_payments_en.payment_type
order_payments_en.payment_installments
order_payments_en.payment_value
products_en.product_category_name
products_en.product_name_lenght
products_en.product_description_lenght
products_en.product_photos_qty
products_en.product_weight_g
products_en.product_length_cm
products_en.product_height_cm
products_en.product_width_cm
seller_en.seller_zip_code_prefix
seller_en.seller_city
seller_en.seller_state
customer_en.COUNT(orders_en)
customer_en.MAX(orders_en.freight_value)
customer_en.MAX(orders_en.price)
customer_en.MEAN(orders_en.freight_value)
customer_en.MEAN(orders_en.price)
customer_en.MIN(orders_en.freight_value)
customer_en.MIN(orders_en.price)
customer_en.MODE(orders_en.order_status)
customer_en.NUM_UNIQUE(orders_en.order_status)
customer_en.SKEW(orders_en.freight_value)
customer_en.SKEW(orders_en.price)
customer_en.STD(orders_en.freight_value)
customer_en.STD(orders_en.price)
customer_en.SUM(orders_en.freight_value)
customer_en.SUM(orders_en.price)
order_payments_en.COUNT(orders_en)
order_payments_en.MAX(orders_en.freight_value)
order_payments_en.MAX(orders_en.price)
order_payments_en.MEAN(orders_en.freight_value)
order_payments_en.MEAN(orders_en.price)
order_payments_en.MIN(orders_en.freight_value)
order_payments_en.MIN(orders_en.price)
order_payments_en.MODE(orders_en.order_status)
order_payments_en.NUM_UNIQUE(orders_en.order_status)
order_payments_en.SKEW(orders_en.freight_value)
order_payments_en.SKEW(orders_en.price)
order_payments_en.STD(orders_en.freight_value)
order_payments_en.STD(orders_en.price)
order_payments_en.SUM(orders_en.freight_value)
order_payments_en.SUM(orders_en.price)
products_en.COUNT(orders_en)
products_en.MAX(orders_en.freight_value)
products_en.MAX(orders_en.price)
products_en.MEAN(orders_en.freight_value)
products_en.MEAN(orders_en.price)
products_en.MIN(orders_en.freight_value)
products_en.MIN(orders_en.price)
products_en.MODE(orders_en.order_status)
products_en.NUM_UNIQUE(orders_en.order_status)
products_en.SKEW(orders_en.freight_value)
products_en.SUM(orders_en.price)
seller_en.COUNT(orders_en)
seller_en.MAX(orders_en.freight_value)
seller_en.MAX(orders_en.price)
seller_en.MEAN(orders_en.freight_value)
seller_en.MEAN(orders_en.price)
seller_en.MIN(orders_en.freight_value)
seller_en.MIN(orders_en.price)
seller_en.MODE(orders_en.order_status)
seller_en.NUM_UNIQUE(orders_en.order_status)
seller_en.SKEW(orders_en.freight_value)
seller_en.SKEW(orders_en.price)
seller_en.STD(orders_en.freight_value)
seller_en.STD(orders_en.price)
seller_en.SUM(orders_en.freight_value)
seller_en.SUM(orders_en.price)
```

3.0 Business Objectives

i. To determine the top 5 states with a high concentration of sellers and customers

ii. To analyse the trends in the number of orders over time to identify patterns

iii. To determine the month with the highest demand

iv. To analyse the distribution of payment methods used by customers

v. To determine the number of orders customers placed in a single order

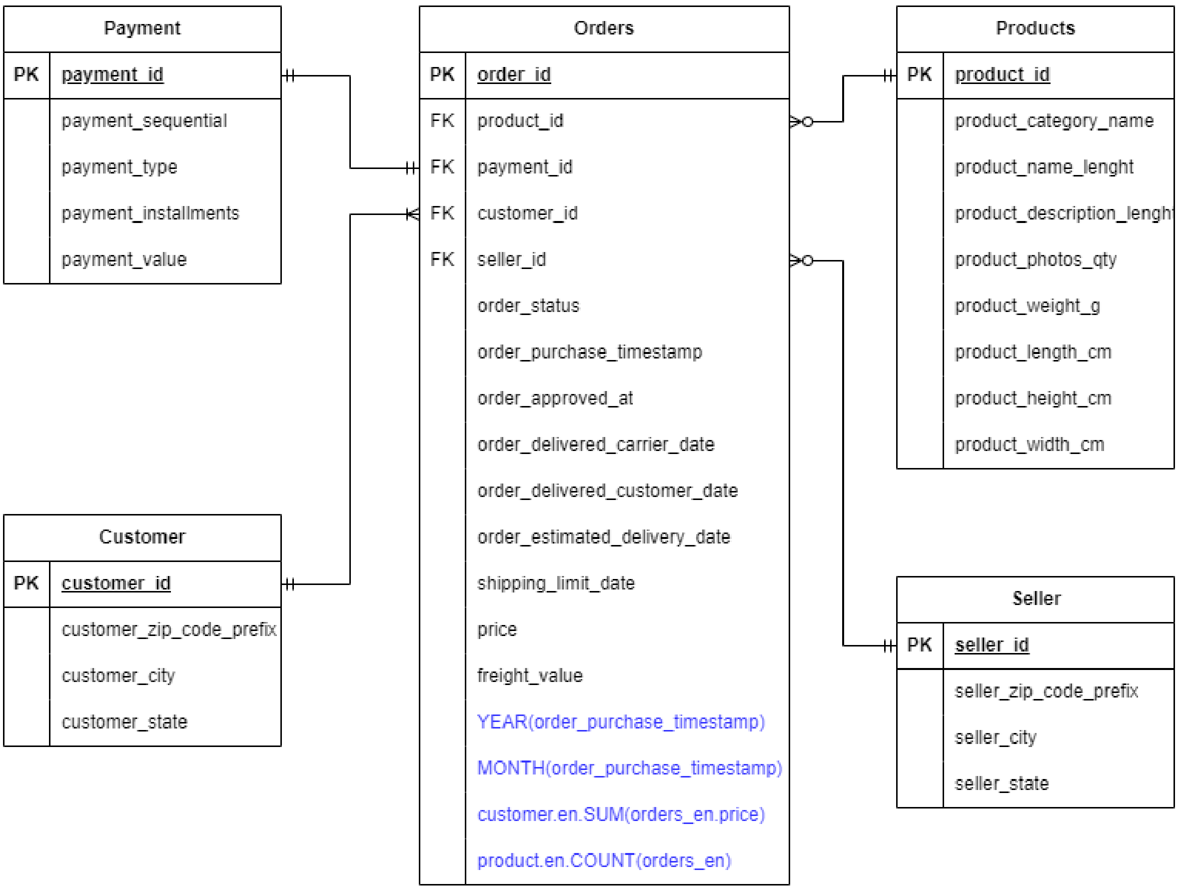
vi. To determine the state which brings the largest sales

4.0 Data Model

4.1 Star Schema

The fact table (i.e. Orders) sits at the center, connected to 4 dimension tables (i.e. Customer, Products, Payment, Seller) representing different aspects of the business. The foreign keys (i.e. product_id, payment_id, customer_id, seller_id) in the fact table connect to the primary keys in each dimension table, establishing relationships and enabling comprehensive data analysis across various dimensions.

This optimised data model supports querying and analysis that aligns with the objectives outlined earlier. It simplifies complex queries, allowing for efficient extraction of insights related to seller and customer demographics, order trends over time, payment methods, product preferences, and regional sales performance.



***Attributes in blue are the new feature generated by FeatureTools and are also used for insights.*

5.0 Data Dictionary

5.1 Fact Table: Orders

Attribute	Data Type	Constraints	Description
order_id	Integer	Primary Key, Not null	Unique identifier for each order.
product_id	String	Foreign Key, Not null	Foreign key linking to the product placed.
payment_id	String	Foreign Key, Not null	Foreign key linking to the payment method used to pay the order.
customer_id	String	Foreign Key, Not null	Foreign key linking to the customer who placed the order.
seller_id	String	Foreign Key, Not null	Foreign key linking to the seller of the products of the order placed.
order_status	String	Not null	Status of the order, indicating its progress or completion.
order_purchase_timestamp	Datetime	Not null	Timestamp when the customer initially purchased the order.
order_approved_at	Datetime	Not null	Timestamp when the order was approved for processing.
order_delivered_carrier_date	Datetime	Not null	Timestamp when the order was handed over to the carrier for delivery.
order_delivered_customer_date	Datetime	Not null	Timestamp when the order was successfully delivered to the customer.
order_estimated_delivery_date	Datetime	Not null	Estimated delivery date for the order, managing customer expectations.
shipping_limit_date	Datetime	Not null	Timestamp indicating the shipping limit date for the order item.
price	Float	Not null	The price of the order item.

freight_value	Float	Not null	The cost of shipping (freight) for the order item.
YEAR(order_purchase_timestamp)	Integer	Not null	The year in which the order purchase timestamp falls.
MONTH(order_purchase_timestamp)	Integer	Not null	The month in which the order purchase timestamp falls.
customer.en.SUM(orders_en.price)	Float	Not null	Total sum of the prices of orders for various customers.
product.en.COUNT(orders_en)	Integer	Not null	Count of products in each order.

5.2 Dimension Table: Customer

Attribute	Data Type	Constraints	Description
customer_id	String	Primary Key, Not null	Unique identifier for each customer.
customer_zip_code_prefix	Integer	Not null	Numeric code that represents the postal code prefix associated with the customer's address.
customer_city	String	Not null	Name of the city where the customer resides.
customer_state	String	Not null	Name of the state where the customer is located.

5.3 Dimension Table: Products

Attribute	Data Type	Constraints	Description
product_id	String	Primary Key, Not null	Unique identifier for each product.
product_category_name	String	Not null	Name of the product category to which the product belongs.
product_name_length	Float	Not null	Length of the product name.
product_description_length	Float	Not null	Length of the product description.
product_photos_qty	Float	Not null	Quantity of photos available for the product.
product_weight_g	Float	Not null	Weight of the product in grams.
product_length_cm	Float	Not null	Length of the product in centimetres.
product_height_cm	Float	Not null	Height of the product in centimeters.

product_width_cm	Float	Not null	Width of the product in centimeters..

5.4 Dimension Table: Payment

Attribute	Data Type	Constraints	Description
payment_id	String	Primary Key, Not null	Unique identifier for each payment.
payment_sequential	Integer	Not null	Sequential number indicating the order of payment within the same order.
payment_type	String	Not null	Type of payment method used for the order.
payment_installments	Integer	Not null	The number of instalments or payments made for the order.
payment_value	Float	Not null	The value or amount of the payment.

5.5 Dimension Table: Seller

Attribute	Data Type	Constraints	Description
seller_id	String	Primary Key, Not null	Unique identifier for each seller.
seller_zip_code_prefix	Integer	Not null	Numeric code that represents the postal code prefix associated with the seller's address.
seller_city	String	Not null	Name of the city where the seller resides.
seller_state	String	Not null	Name of the state where the seller is located.

6.0 Insights

6.1 Insights 1: Demographics of Sellers and Customers

Objective:

- To determine the top 5 states with a high concentration of sellers and customers

Features used:

- seller_en.seller_state
- customer_en.customer_state

Outcome:

- **<u>São Paulo has the highest concentration of sellers (75%) and customers (42%)</u>** *(refer to Fig 1 and Fig 2)*, making it a significant hub for e-commerce activity. Santa Catarina has a relatively higher concentration of sellers (3.7%), indicating a potential area of interest for expanding seller engagement. On the other hand, Rio Grande do Sul stands out with a higher concentration of customers (5.5%), suggesting a potential market focus for customer-oriented strategies.

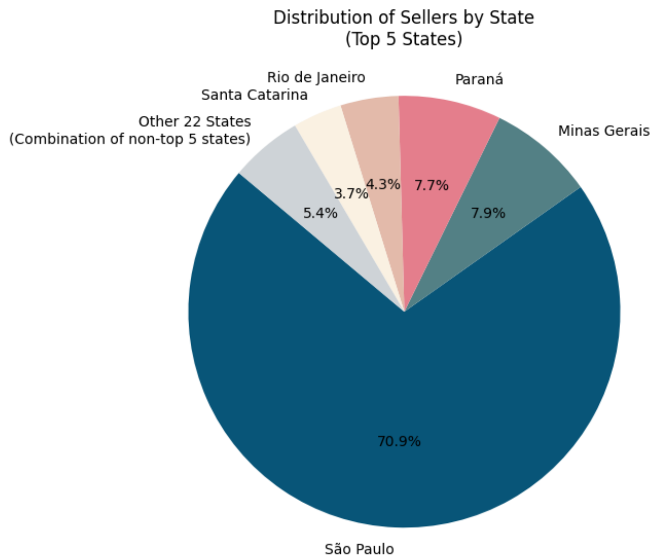


Figure 1: Top 5 states with the greatest number of sellers

Link to GitHub: https://github.com/Janice-Chong/featureTools.git

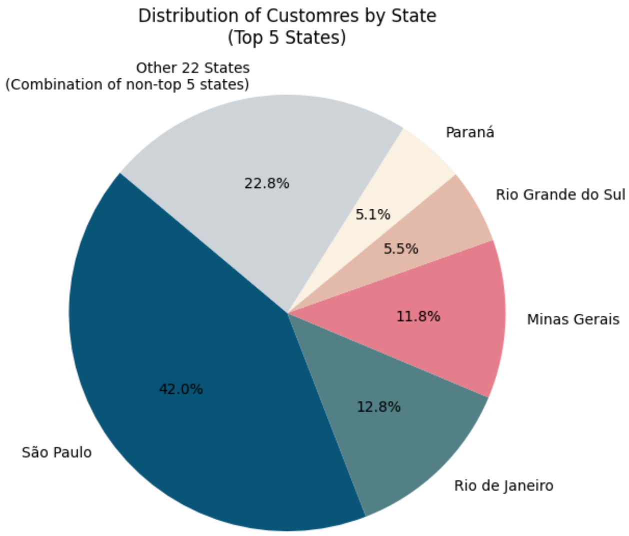


Figure 2: Top 5 states with the greatest number of customers

Python Code

```
# Count the number of sellers in each state
seller_count = df['seller_en.seller_state'].value_counts()
top5_seller_count = seller_count.head(5)
```

```
# Count the number of customers in each state
customer_counts = df['customer_en.customer_state'].value_counts()
top5_customer_counts = customer_counts.head(5)
```

To count the occurrences of unique values in the 'seller_state' / 'customer_state' column of the DataFrame 'df'. It tallies how many sellers/customers are based in each state and provides a count for each state. 'seller_count.head(5)' / 'customer_count.head(5)' is to obtain the top 5 states with the most number of sellers/customers.

```
# Sum the counts of states that are not in the top 5
others_count = seller_count[~seller_count.index.isin(top5_seller_count.index)].sum()
```

```
# Sum the counts of customer that are not in the top 5
others_count = customer_counts[~customer_counts.index.isin(top5_customer_counts.index)].sum()
```

Variable 'others_count' stores the summation of sellers/customers from the other states that are not the top 5 states.

6.2 Insights 2: Order Trends Over Time

Objective:

- To analyse the trends in the number of orders over time to identify patterns

Features used:

- YEAR(order_purchase_timestamp)
- MONTH(order_purchase_timestamp)

Outcome:

- **Monthly orders increase over time, from 2016 to 2018** *(refer to Fig 3).* This suggests that the business experienced a positive trend in customer orders over this three-year period. Such insights are valuable as this may indicate the need to scale up operations, optimise logistics, or plan for increased customer demand. Additionally, this information could be used for forecasting and making informed business decisions.

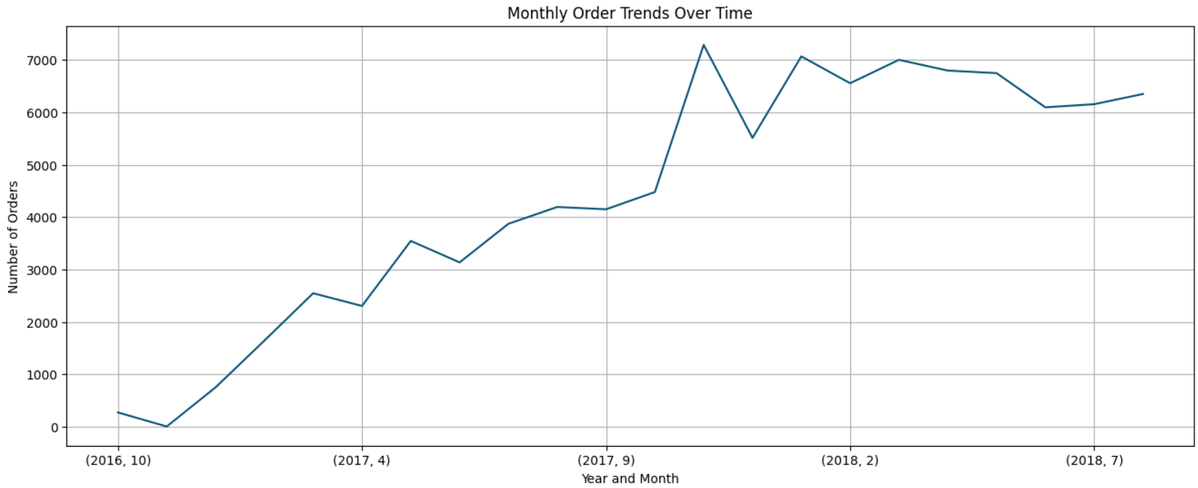


Figure 3: Trends of orders monthly from the year 2016 to 2018

Python Code

```
# Group data by month and year, then count orders
monthly_order_counts = df.groupby(['YEAR(order_purchase_timestamp)', 'MONTH(order_purchase_timestamp)']).size()
```

'monthly_order_counts' provides a time-series representation of the number of orders made each month, with the data organized by both year and month. It is obtained by grouping the original dataset 'df' by both the year and month of the 'order_purchase_timestamp' column. The '.size()' function is used to count the number of occurrences in each group.

6.3 Insights 3: Busiest Month of Each Year

Objective:

- To determine the month with the highest demand

Features used:

- YEAR(order_purchase_timestamp)
- MONTH(order_purchase_timestamp)

Outcome:

- **The busiest months in the years 2016, 2017, and 2018 are October, November, and January respectively** (*refer to Fig 4*). Thus, we can say that the period between October to January may be the busiest month annually. This can possibly be due to seasonal trends. For example, in October 2016, there might have been a seasonal increase in sales due to the approaching holiday season, leading to higher consumer spending. In November 2017, the sales might have been boosted by Black Friday sales, which are known for their high discounts and increased shopping activity. In January 2018, post-holiday clearance sales and New Year promotions might have contributed to increased sales.

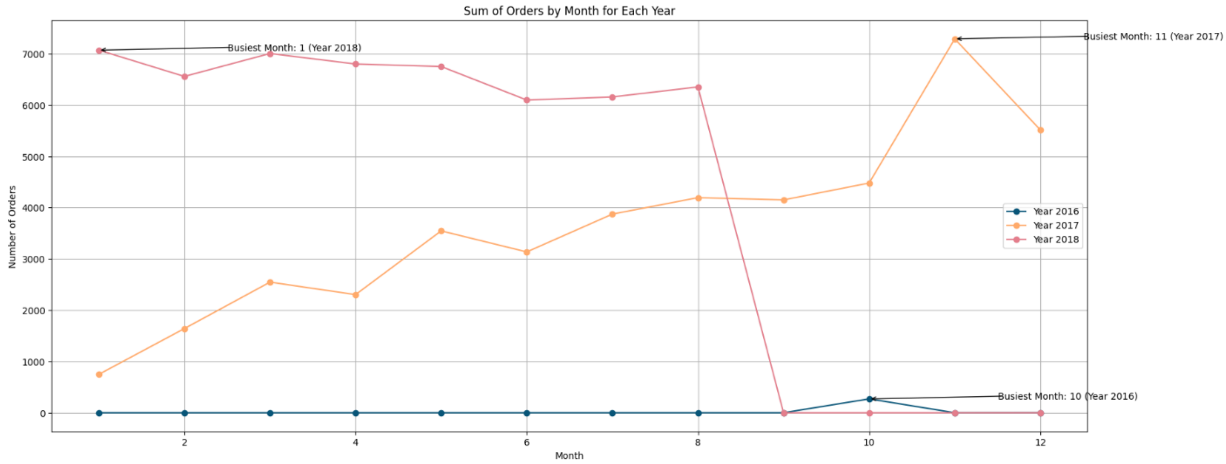


Figure 4: Busiest month of the year 2016, 2017 and 2018

Python Code

```
# Group data by year and count the number of orders in each month
yearly_month_counts =
    df.groupby('YEAR(order_purchase_timestamp)')['MONTH(order_purchase_timestamp)'].value_counts().unstack(fill_value=0)
```

Calculates the count of orders made for each combination of year and month using the 'order_purchase_timestamp' column in the DataFrame 'df'. It groups the data first by the year and then by the month, counting the number of orders for each unique year-month pair.

6.4 Insights 4: Payment Method Analysis

Objective:

- To analyse the distribution of payment methods used by customers

Features used:

- order_payments_en.payment_type

Outcome:

- **Credit cards are the most preferred payment method among customers** (*refer to Fig 5*). Possibly due to customers finding it secure, and easy to use. This insight is significant as it may indicate that they should continue to support and possibly incentivise the use of credit cards. Understanding customer payment preferences can also help tailor marketing strategies and partnerships with financial institutions.

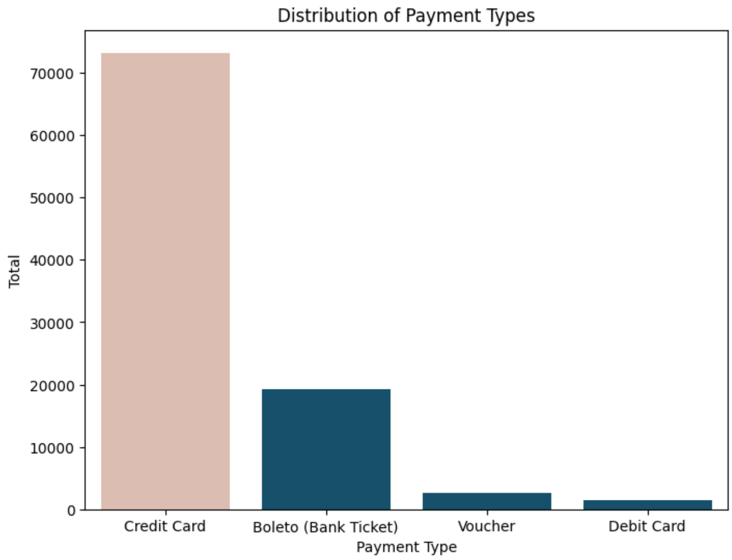


Figure 5: Customers' most preferred payment option

Python Code

```python
# Count the occurrences of each payment type
payment_type_counts = df['order_payments_en.payment_type'].value_counts()
```

Counting the occurrences of each unique payment type in the 'order_payments_en.payment_type' column of the original dataset 'df'. It shows how many orders are made using each payment method, helping to understand the popularity of different payment options.

6.5 Insights 5: Number of Products Per Order

Objective:

- To determine the number of orders customers placed in a single order.

Features used:

- products_en.COUNT(orders_en)

Outcome:

- **Most people buy only 1 to 2 products per order** (*refer to Fig 6*). Single-product orders may indicate a missed opportunity for cross-selling. Cross-selling involves recommending related products that complement the one the customer is purchasing. If a customer is buying a smartphone, for instance, the algorithm can recommend phone cases, screen protectors, or headphones. By showing tailored product suggestions, the chances of customers adding more items to their cart increases. Thus, boosting sales and customer engagement effectively.

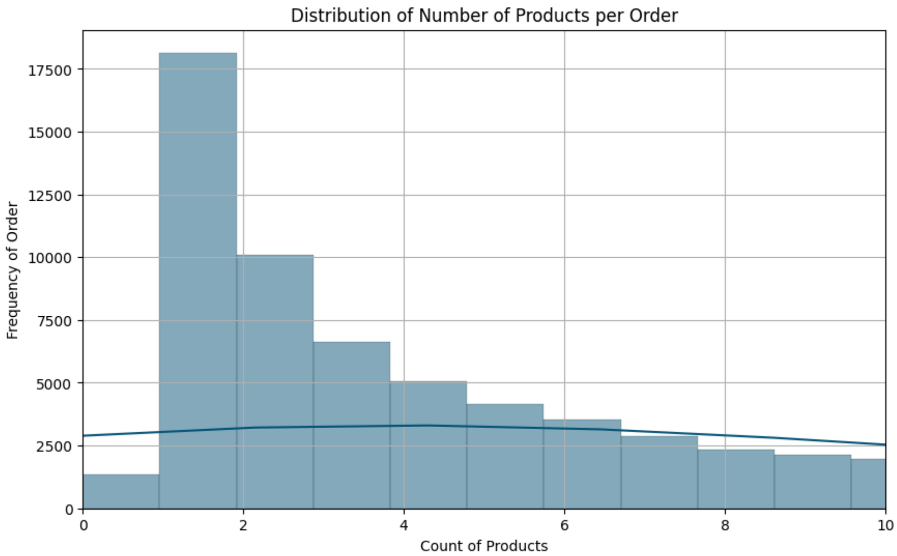


Figure 6: Number of products placed in a single order

Python Code

```python
# Plot a histogram of the count of order items
plt.figure(figsize=(10, 6))

sns.histplot(df['products_en.COUNT(orders_en)'], kde=True, color='#085578')
# Set the x-axis limits
plt.xlim(0, 10)
plt.title('Distribution of Number of Products per Order')
plt.xlabel('Count of Products')
plt.ylabel('Frequency of Order')
plt.grid(True)
plt.show()
```

By plotting a histogram, we can see the distribution of the number of products ordered by the customers in a single order. A histogram helps understand the typical order size and how often customers order a certain number of products.

6.6 Insights 6: Which State is the Biggest Spender?

Objective:

- To determine the state which brings the largest sales.

Features used:

- customer_en.customer_state
- customer_en.SUM(orders_en.price

Outcome:

- **São Paulo state is the biggest spender** (*refer to Fig 7*). São Paulo is a major urban and economic centre in Brazil. Consequently, higher spending is expected due to the state's economic strength. Urban areas typically have higher purchasing power and more access to a variety of goods and services, which can result in greater spending.

Link to GitHub: https://github.com/Janice-Chong/featureTools.git

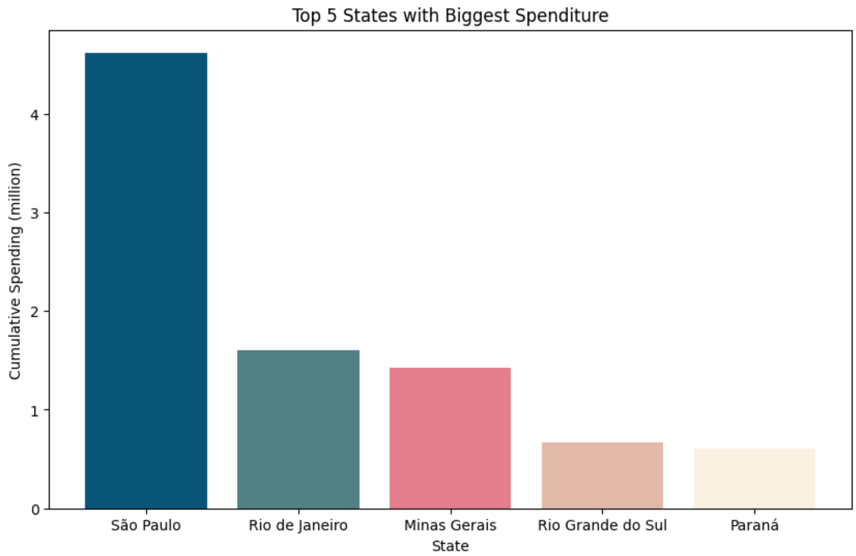


Figure 7: Top 5 states with the highest amount spent by customers

Python Code

```
# Group the data by state and calculate cumulative spending per state
state_spending = df.groupby('customer_en.customer_state')['customer_en.SUM(orders_en.price)'].sum().reset_index()
```

Grouping the data by the customers' states and calculate the cumulative spending per state by summing up the 'customer_en.SUM(orders_en.price)' column for each state.

```
# Sort the data by cumulative spending in descending order
state_spending = state_spending.sort_values(by='customer_en.SUM(orders_en.price)', ascending=False)
```

Sort the cumulative spending in descending order so that the state with the highest spending appears first using the '.sort_values(by='customer_en.SUM(orders_en.price)', ascending=False)'.

```
top5_state = state_spending.head(5)
```

Get the top 5 states with the highest spending using '.head(5)'

7.0 Reflection

Featuretools played a pivotal role in automatically generating insightful features from our dataset, thereby augmenting our data modelling efforts. In essence, Featuretools empowered us to create new features based on the existing data, automating the often-time-consuming feature engineering process. This not only expedited our data modelling endeavours but also significantly enhanced the accuracy and effectiveness of our data models.

One of the standout features of Featuretools is its capability to model complex relationships among entities within our dataset. This was instrumental in our data modelling approach, enabling us to define and leverage entities and their relationships. These relationships proved to be essential for aggregating data across different dataframes, an invaluable process when dealing with multi-source data.

For instance, we harnessed Featuretools to engineer features that offered profound insights into customer behaviour, such as calculating the count of orders. These features were instrumental in helping us understand customer preferences and loyalty, factors critical to our analytical needs.

Having harnessed Featuretools to generate these features, we proceeded to utilise various data visualisation techniques to extract actionable insights from our enriched dataset. The visualisations included pie charts, enabling us to effectively depict the distribution of sellers' and customers' locations. Moreover, line graphs were employed to uncover trends and patterns in customer order patterns, shedding light on the temporal aspects of our data.

Histograms further allowed us to gain an understanding of the distribution of the number of products within each order. Additionally, bar plots provided insights into the preferred payment types, as well as the cumulative spending of customers categorised by their respective states. These visualisations significantly contributed to our analytical needs and data-driven decision-making.

The seamless integration of Featuretools into our data modelling pipeline was instrumental in augmenting our data warehousing efforts. By generating valuable features and enhancing our data visualisation tools, Featuretools empowered us to delve deeper into the complexities of our e-commerce dataset. This newfound understanding lays the foundation for data-driven decision-making and offers valuable insights that can steer future enhancements and improvements in our e-commerce platform.

8.0 References

Alteryx Inc. (2020). *What is Featuretools?* Alteryx. https://featuretools.alteryx.com/en/stable/#:~:text=Featuretools%20is%20a%20framework%20to,feature%20matrices%20for%20machine%20learning.

Joshi, P. (2022). *A Hands-On Guide to Automated Feature Engineering using Featuretools in Python*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/08/guide-automated-feature-engineering-featuretools-python/

OLIST. (2021). *Brazilian E-Commerce Public Dataset by Olist*. Kaggle. https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce