BDM3303 SEC 401

Olist E-commerce

Presentation by

6410767 Jiratchaya Wanitchakorn
6413036 Sarasin Tangtrongharuthai
6420055 Myat Thu Thu Kyaw
6511338 Su Myat Noe
6530298 Kaung Khant Paing

Table of Content

Data Discovery

Dataset : Brazilian E-Commerce Public Dataset by Olist

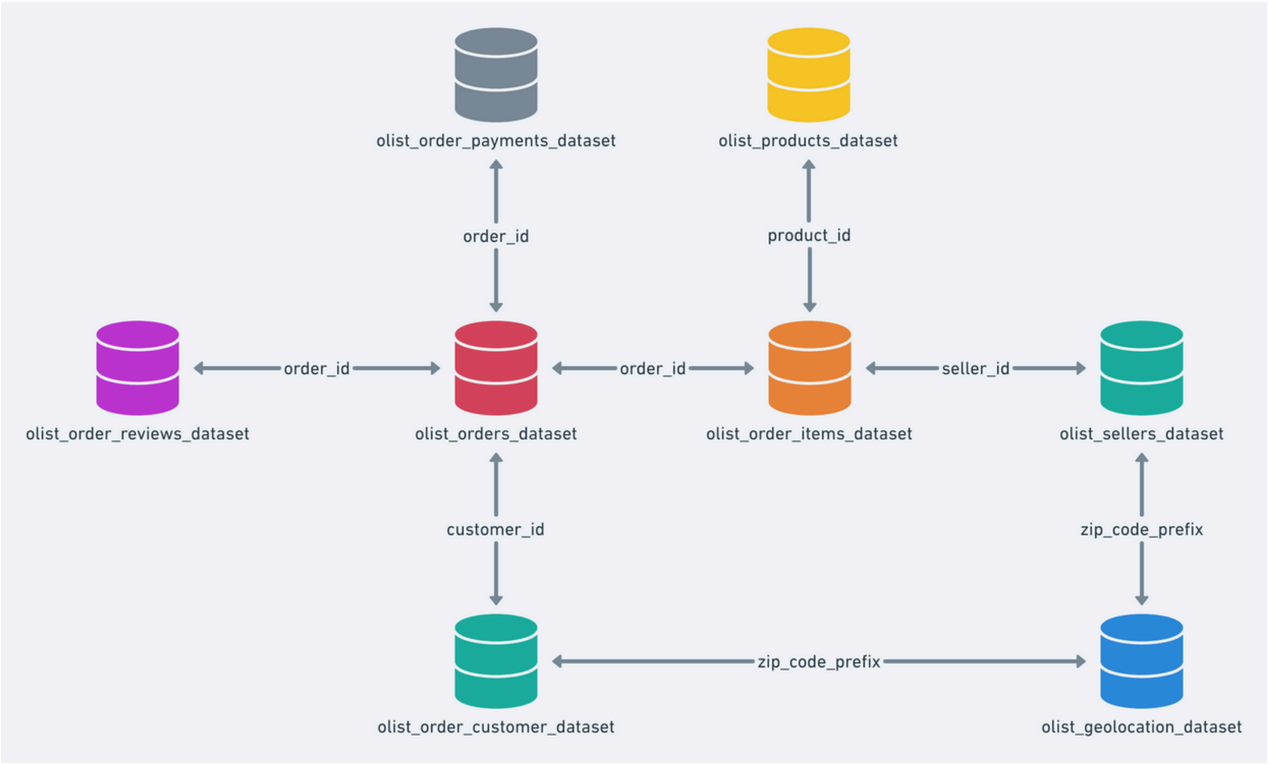
Source: Kaggle
Date : 2016-2018
Link :
https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

- payments
- orders
- reviews
- customers
- order items
- products
- sellers
- geolocations

- An order might have multiple items.
- Each item might be fulfilled by a distinct seller.

Brazilian E-Commerce Public Dataset by Olist

100,000 Orders with product, customer and reviews info



Data Profiling

olist_order_items	
order_id	string
order_item_id	Int64
product_id	string
seller_id	string
price	Float64
freight_value	Float64
shipping_date_limit	datetime64

olist_orders	
order_id	string
customer_id	string
order_status	string
purchase_time	datetime64
approve_time	datetime64
delivered_carrier_date	datetime64
delivered_customer_date	datetime64
estimated_date	datetime64

olist_products	
product_id	string
product_category_name	string
product_name_length	Int64
product_description_length	Int64
product_photos_qty	Int64
product_weight_g	Int64
product_length_cm	Int64
product_height_cm	Int64
prouct_width_cm	Int64

product_category_name_trans	
product_category_name	string
product_catergory_name english	string

olist_order_reviews	
review_id	string
order_id	string
review_score	Int64
review_comment_title	string
review_comment_message	string
review_creation_date	datetime64
review_answer_timestamp	datetime64

Data Profiling

olist_customers	
customer_id	string
customer_unique_id	string
customer_zip_code_prefix	Int64
customer_city	string
customer_state	string

olist_payments	
order_id	string
payment_sequential	Int64
payment_type	string
payment_installments	Int64
payment_value	Float64

olist_sellers	
seller_id	string
seller_zip_code_prefix	Int64
seller_city	string
seller_state	string

olist_geolocation	
geolocation_zip_code_prefix	Int64
geolocation_lat	Float64
geolocation_lng	Float64
geolocation_city	string
geolocation_state	string

Data Profiling

01. Item

Name :
Size : ~99441 rows , 5 columns
Features : 2 categorical ,1 numeric
Meta : 3 text
Missing data : none

02. Order

Name : olist_orders_dataset
Size : ~99441 rows , 8 columns
Features : 1 categorical , 5 numeric
Meta :2 text
Missing data : 4908 (0.8%) in features

03. Product

Name : olist_products_dataset
Size : ~ 32951 rows , 9 columns
Features : 1 categorical , 7 numeric
Meta : 1 text
Missing data : 2448 (0.9%) in features

04. Category

Name :
product_category_name_translation
Size : ~ 72 rows , 2 columns
Meta : 2 text
Missing data : none

05. Review

Name : olist_order_reviews_dataset
Size : ~ 99224 rows , 7 columns
Features : 3 numeric
Meta : 4 text
Missing data : 36% missing data

06. Customer

Name : olist_customers_dataset
Size : ~99441 rows , 5 columns
Features : 2 categorical ,1 numeric
Meta : 2 text
Missing data : none

07. Payment

Name : olist_order_payments_dataset
Size : ~ 103886 rows , 5 columns
Features : 1 categorical , 3 numeric
Meta : 1 text
Missing data : none

08. Seller

Name : olist_orders_dataset
Size : ~ 3095 rows , 4 columns
Features : 1 categorical , 1 numeric
Meta : 2 text
Missing data : none

09. Geolocation

Name : olist_geolocation_dataset
Size : ~ 1000163 rows , 5 columns
Features : 1 categorical , 3 numeric
Meta : 1 text
Missing data : none

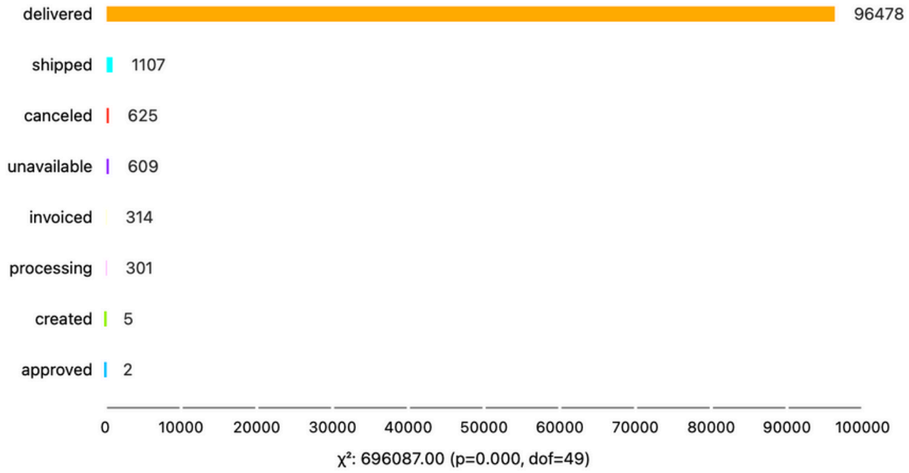
olist_order_dataset

01. Item

Name :
Size : ~99441 rows , 5 columns
Features : 2 categorical ,1 numeric
Meta : 3 text
Missing data : none

olist_orders	
order_id	string
customer_id	string
order_status	string
purchase_time	datetime64
approve_time	datetime64
delivered_carrier_date	datetime64
delivered_customer_date	datetime64
estimated_date	datetime64

Name	Distribution	Mean	Mode	Median	Dispersion	Min.	Max.	Missing
order_purchase_timestamp		2017-12-31 08:43:12.776...	2017-11-20 10:59:08	2018-01-18 23:04:36	~2 years	2016-09-04 21:15:19	2018-10-17 17:30:18	0 (0 %)
order_approved_at		2017-12-31 18:35:24.09...		2018-01-19 11:36:13	~2 years	2016-09-15 12:16:38	2018-09-03 17:40:06	160 (0 %)
order_delivered_carrier_date		2018-01-04 21:49:48.138...		2018-01-24 16:10:58	~23 months	2016-10-08 10:34:01	2018-09-11 19:48:28	1783 (2 %)
order_delivered_customer_date		2018-01-14 12:09:19.035...		2018-02-02 19:28:10.500...	~2 years	2016-10-11 13:46:32	2018-10-17 13:22:46	2965 (3 %)
order_estimated_delivery_date		2018-01-24 03:08:37.73...	2017-12-20 00:00:00	2018-02-15 00:00:00	~2 years	2016-09-30 00:00:00	2018-11-12 00:00:00	0 (0 %)



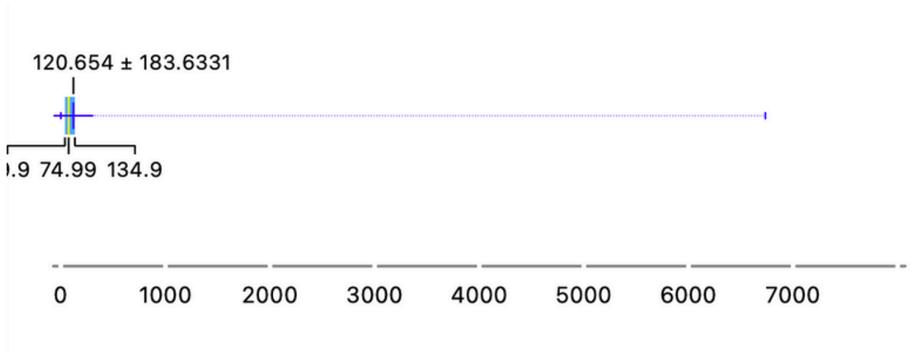
Most of the orders are already **delivered**.

There is **missing values** in some date because some of the **order are not delivered.**

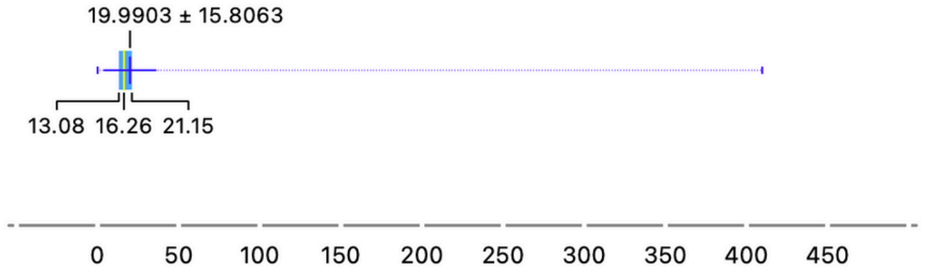
olist_order_item_dataset

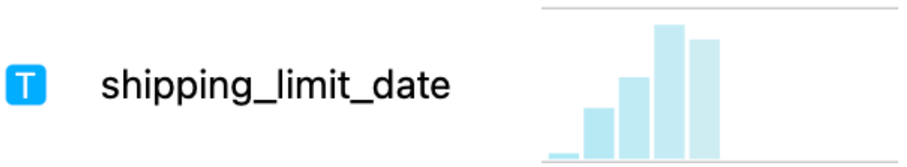
02. Order

Name : olist_orders_dataset
Size : ~99441 rows , 8 columns
Features : 1 categorical , 5 numeric
Meta :2 text
Missing data : 4908 (0.8%) in features



olist_order_items	
order_id	string
order_item_id	Int64
product_id	string
seller_id	string
price	Float64
freight_value	Float64
shipping_date_limit	datetime64





Price of Items

- Average : 120
- Range : 0.85 - 6735
- Mode : 60

Freight Value of Items

- Average : 20
- Range : 0 - 410
- Mode : 15

Shipping Limit Date

- From : 2016-09-19
- To : 2020-04-09
- Dispersion : ~3 years

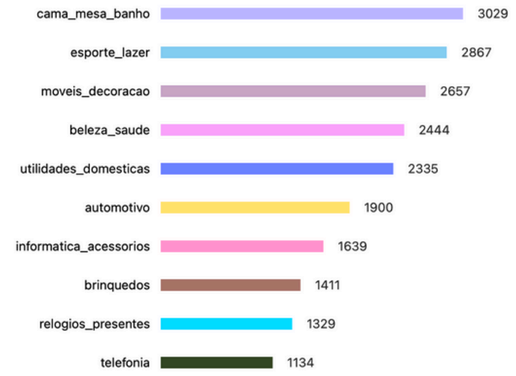
olist_products_dataset

03. Product

Name : olist_products_dataset
Size : ~ 32951 rows , 9 columns
Features : 1 categorical , 7 numeric
Meta : 1 text
Missing data : 2448 (0.9%) in features

olist_products	
product_id	string
product_category_name	string
product_name_length	Int64
product_description_length	Int64
product_photos_qty	Int64
product_weight_g	Int64
product_length_cm	Int64
product_height_cm	Int64
prouct_width_cm	Int64

Name	Distribution	Mean	Mode	Median	Dispersion	Min.	Max.	Missing
product_name_lenght		48.4769	60	51	0.21135	5	76	610 (2 %)
product_description_lenght		771.495		595	0.823214	4	3992	610 (2 %)
product_photos_qty		2.18899	1	1	0.793399	1	20	610 (2 %)
product_weight_g		2276.47	200	700	1.88097	0.00	40425	2 (0 %)
product_length_cm		30.8151	16	25	0.548894	7	105	2 (0 %)
product_height_cm		16.9377	10	13	0.805149	2	105	2 (0 %)
product_width_cm		23.1967	11	20	0.520714	6	118	2 (0 %)



dataset include details information of products.

Bed Bath Table is the **most popular product category** on Olist based on the number of orders made.

product_category_name_translation

04. Category

Name :
product_category_name_translation
Size : ~ 72 rows , 2 columns
Meta : 2 text
Missing data : none

product_category_name_trans	
product_category_name	string
product_catergory_name english	string

Product Category data include only two columns.

Since products category name are in Portuguese it maps to the English translation of the category name.

	X.0	X.1
1	product_category_name	product_category_name_english
2	beleza_saude	health_beauty
3	informatica_acessorios	computers_accessories
4	automotivo	auto
5	cama_mesa_banho	bed_bath_table
6	moveis_decoracao	furniture_decor
7	esporte_lazer	sports_leisure
8	perfumaria	perfumery
9	utilidades_domesticas	housewares
10	telefonia	telephony
11	relogios_presentes	watches_gifts
12	alimentos_bebidas	food_drink
13	bebes	baby
14	papelaria	stationery
15	tablets_impressao_imagem	tablets_printing_image
16	brinquedos	toys
17	telefonia_fixa	fixed_telephony
18	ferramentas_jardim	garden_tools
19	fashion_bolsas_e_acessorios	fashion_bags_accessories
20	eletroportateis	small_appliances

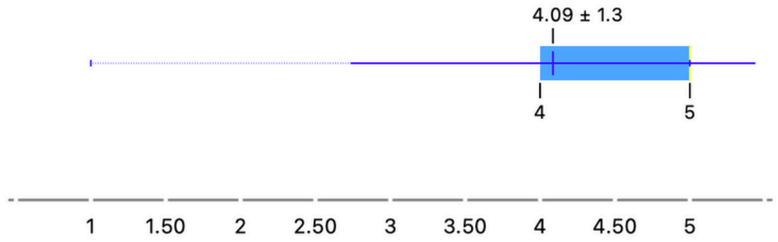
olist_order_reviews_dataset

05. Review

Name : olist_order_reviews_dataset
Size : ~ 99224 rows , 7 columns
Features : 3 numeric
Meta : 4 text
Missing data : 36% missing data

Name	Distribution	Mean	Mode	Median	Dispersion	Min.	Max.	Missing
review_score		4.09	5	5	0.33	1	5	0 (0 %)
review_creation_date		2018-01-12 20:49:23.94...	2017-12-19 00:00:00	2018-02-02 00:00:00	~22 months	2016-10-02 00:00:00	2018-08-31 00:00:00	0 (0 %)
review_answer_timestamp		2018-01-16 00:23:56.97...	2017-06-15 23:21:05	2018-02-04 22:41:47.500...	~2 years	2016-10-07 18:32:28	2018-10-29 12:27:35	0 (0 %)

olist_order_reviews	
review_id	string
order_id	string
review_score	Int64
review_comment_title	string
review_comment_message	string
review_creation_date	datetime64
review_answer_timestamp	datetime64



dataset include details information of reviews.

review comment title and review comment message has missing values because not every order has the text reviews.

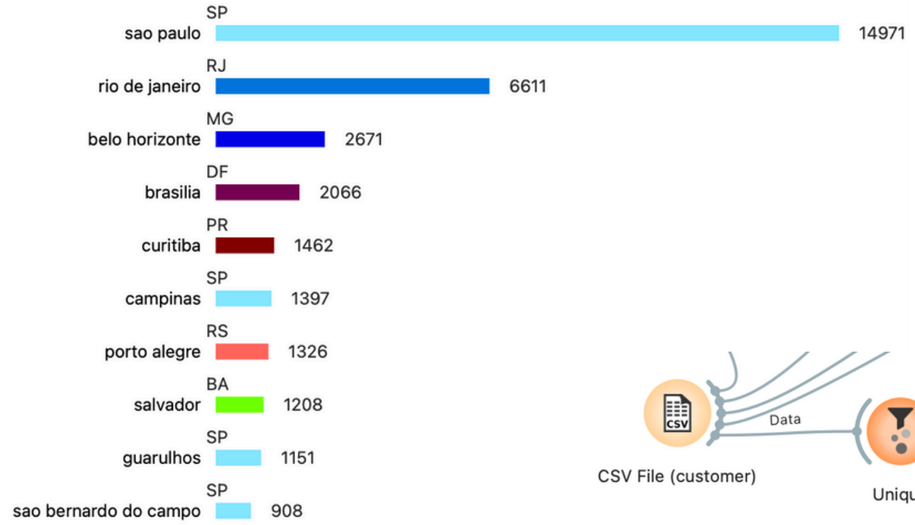
Review Score Range from 1 - 5.
Mean : 4.09
interquartile range: 4-5
indicating most of the customers who give feedback have high satisfaction.

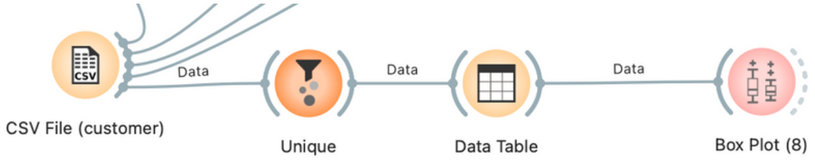
olist_customers_dataset

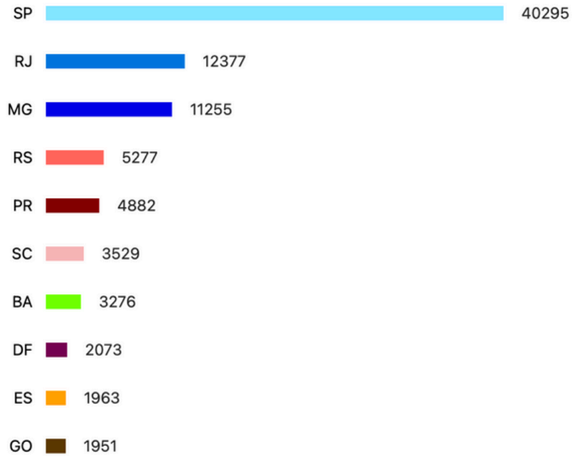
06. Customer

Name : olist_customers_dataset
Size : ~99441 rows , 5 columns
Features : 2 categorical ,1 numeric
Meta : 2 text
Missing data : none

olist_customers	
customer_id	string
customer_unique_id	string
customer_zip_code_prefix	Int64
customer_city	string
customer_state	string







customer id is generated for a single order and customer unique id is to identify a customer entity.

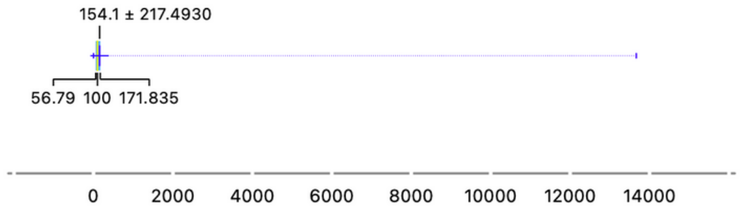
Most of the customer comes from **SP (São Paulo) State.**

Most of the customer comes from **SP (São Paulo) City.**

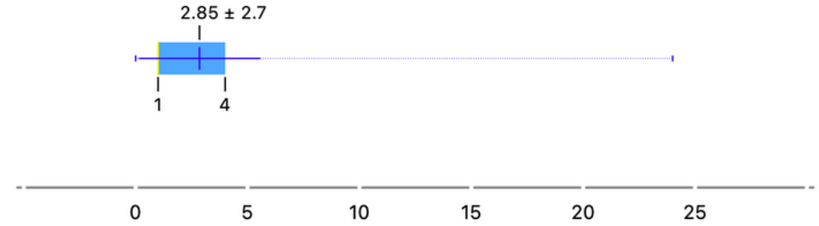
olist_order_payment_dataset

07. Payment

Name : olist_order_payments_dataset
Size : ~ 103886 rows , 5 columns
Features : 1 categorical , 3 numeric
Meta : 1 text
Missing data : none

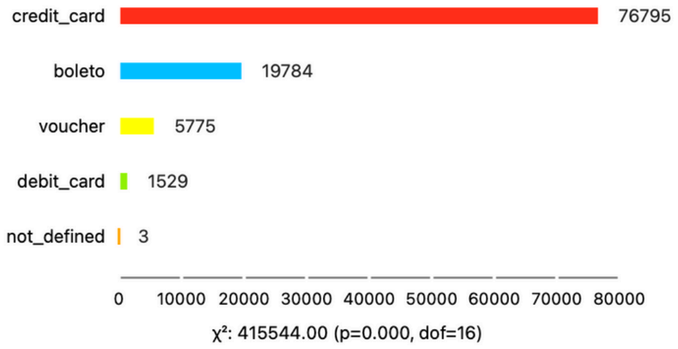


Payment Value



Payment Installments

olist_payments	
order_id	string
payment_sequential	Int64
payment_type	string
payment_installments	Int64
payment_value	Float64



Payment Methods

Mean payment value for each order is **154** with a wide **range 0 to 13664.**

Average number of payment installments made for each order is around **3 times** with the **range from 1 to 24.**

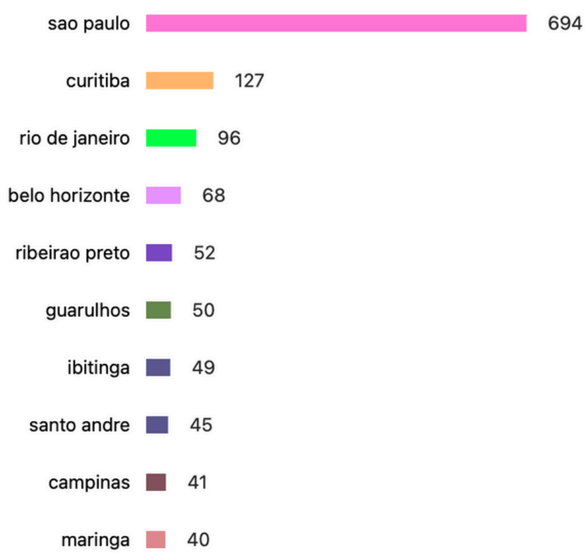
Most of the payment is made with **Credit Card.**

olist_sellers_dataset

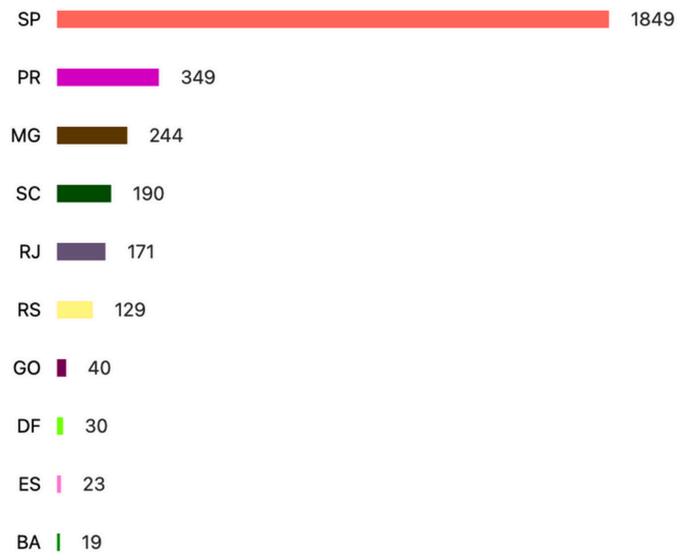
08. Seller

Name : olist_orders_dataset
Size : ~ 3095 rows , 4 columns
Features : 1 categorical , 1 numeric
Meta : 2 text
Missing data : none

olist_sellers	
seller_id	string
seller_zip_code_prefix	Int64
seller_city	string
seller_state	string



Seller City



Seller State

Most of the sellers comes from **SP (São Paulo) State.**

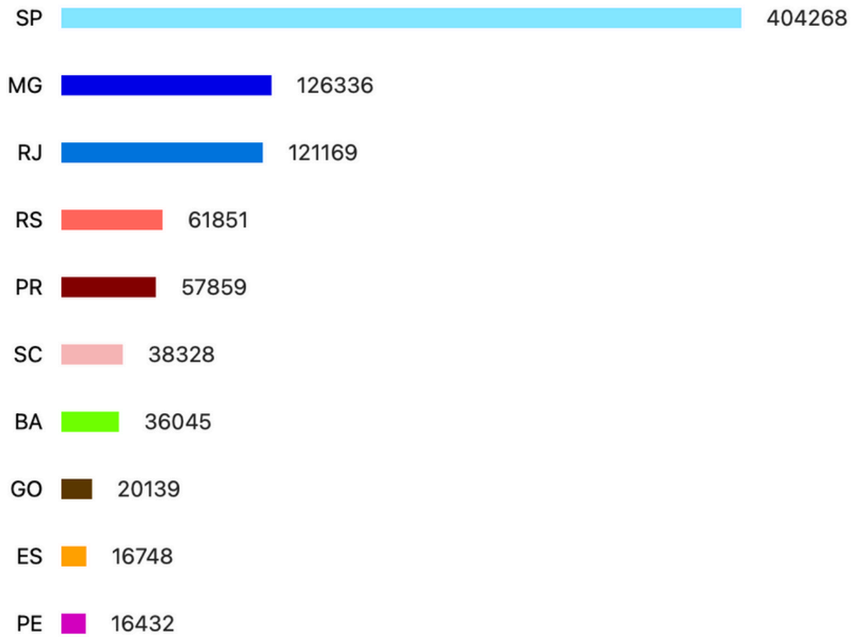
Most of the sellers comes from **SP (São Paulo) City.**

olist_geolocation_dataset

09. Geolocation

Name : olist_geolocation_dataset
Size : ~ 1000163 rows , 5 columns
Features : 1 categorical , 3 numeric
Meta : 1 text
Missing data : none

olist_geolocation	
geolocation_zip_code_prefix	Int64
geolocation_lat	Float64
geolocation_lng	Float64
geolocation_city	string
geolocation_state	string

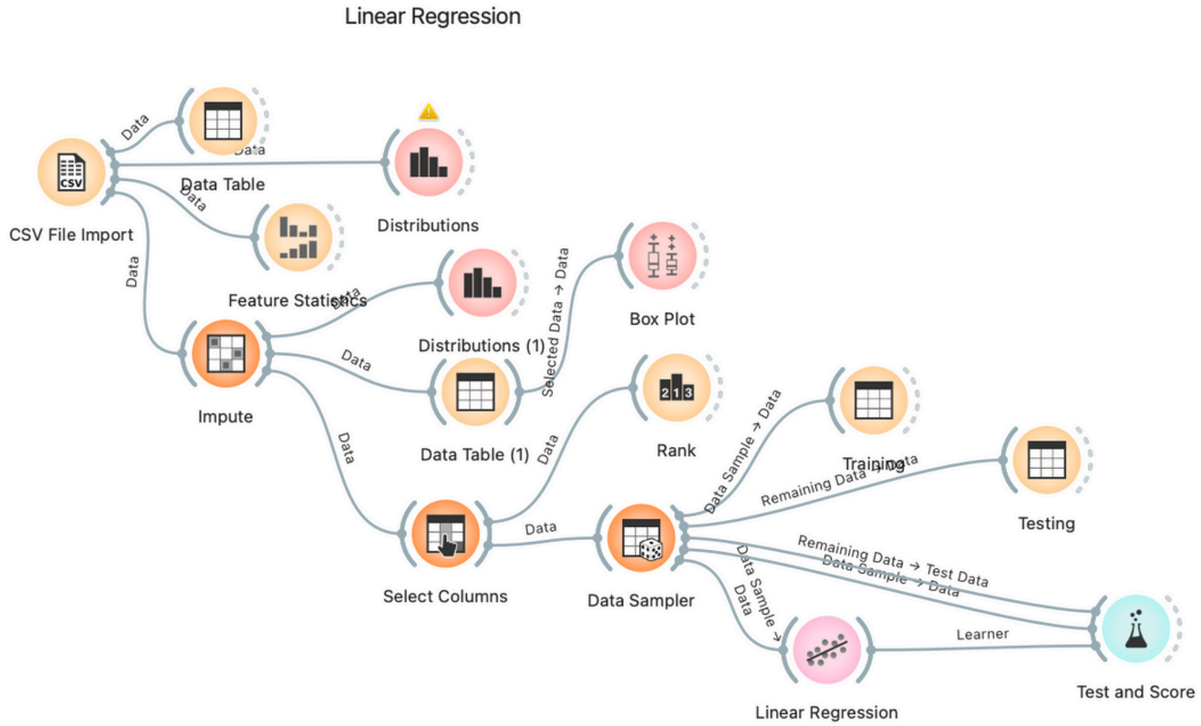


GeoLocation Data helps to identity customers and sellers location using **zip code, llatitude, and longitude.**

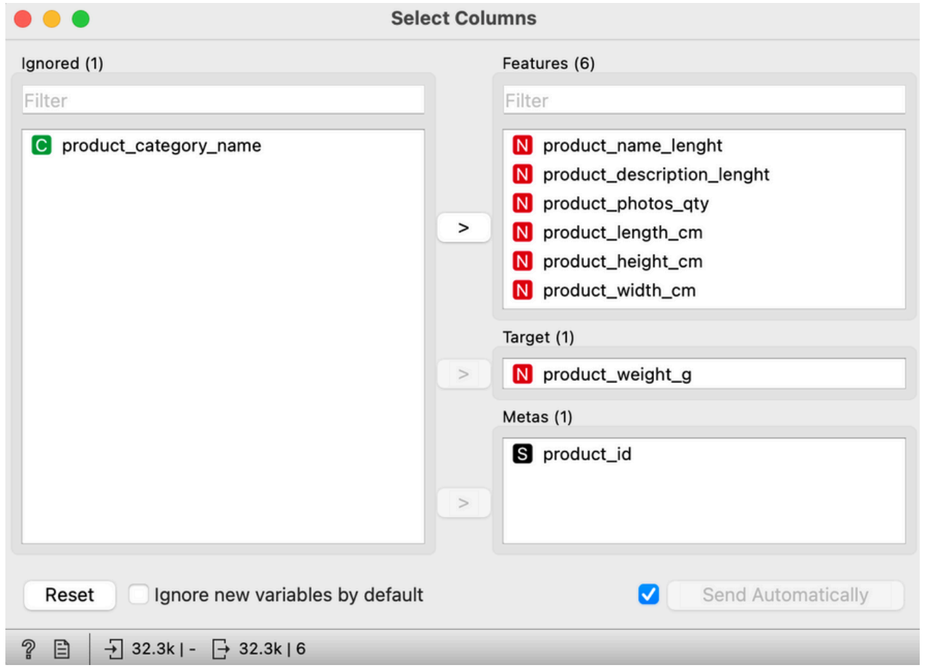
São Paulo (SP) has the most location data because

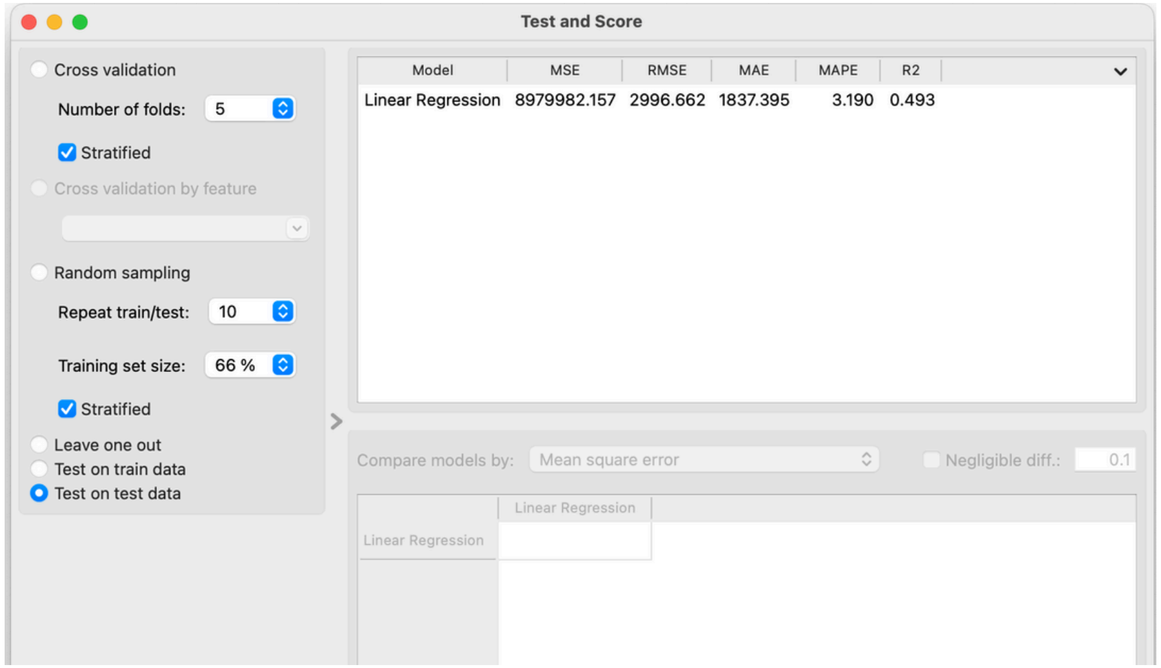
- Brazil's largest economic state
- high population density

Linear Regression on Product dataset



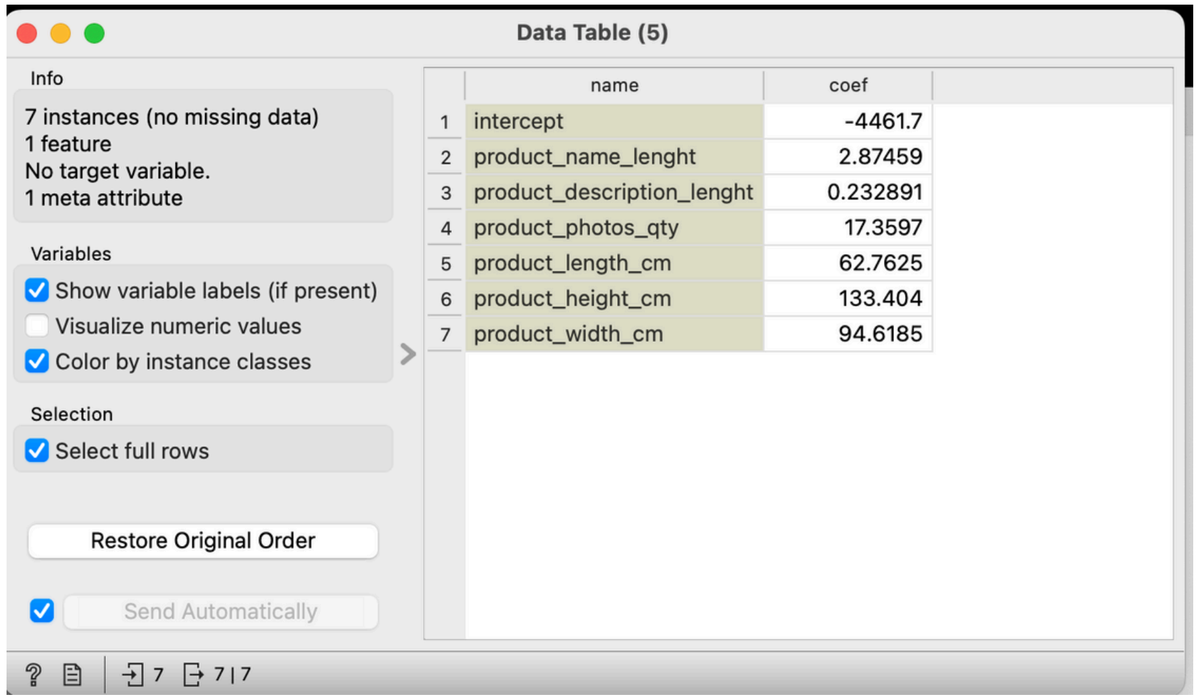
Linear Regression on Product dataset





MAPE = 3.19% and R2 = 49.3% which means the model could explain about of the variance in the data.

Linear Regression on Product dataset



product_weight=−4461.7+(2.87459×product_name_length)+(0.232891×product_description_length)+(17.3597×product_photos_qty)+(62.7625×product_length_cm)+(133.404×product_height_cm)+(94.6185×product_width_cm)

try python for product category

plan for interesing marketing strategy

***know only the highest one**

Step 1 : find name of the product category by idxmax

```
q1=df.groupby('product_category_name')['product_category_name'].size().idxmax()
```

Step 2 : find highest number for product category

```
q2=df.groupby('product_category_name')['product_category_name'].size().max()
```

```
cama_mesa_banho
3029
the category: cama_mesa_banho has the highest product in category = 3029
```

```
print(f'the category: {q1} has the highest product in category = {q2}')
```

***know the first five one**

*easier than command above

```
q1=df.groupby('product_category_name')
['product_category_name'].size().sort_values(
ascending=False)
```

```
product_category_name
cama_mesa_banho                3029
esporte_lazer                  2867
moveis_decoracao               2657
beleza_saude                   2444
utilidades_domesticas          2335
                               ...
```

Top 5 product purchased by category

plan for interesing marketing strategy

We combine the orders, order_items, products, and customers datasets into a single DataFrame (final_df). This allows us to have all the necessary information (orders, products, and customers) in one place.

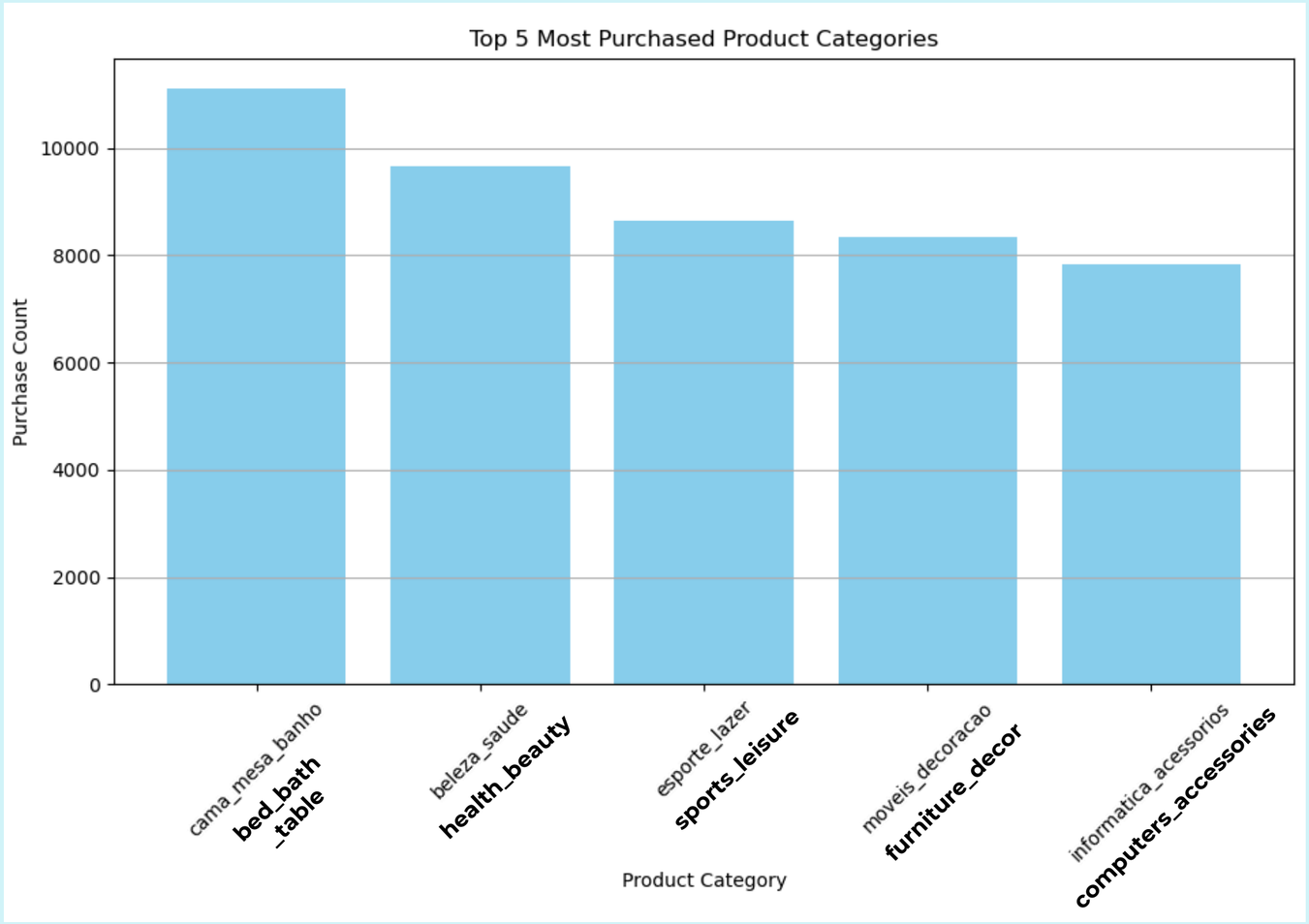
```
merged_orders_items = pd.merge(orders_df, order_items_df, on='order_id', how='left')
final_df = pd.merge(merged_orders_items, products_df, on='product_id', how='left')
final_df = pd.merge(final_df, customers_df, on='customer_id', how='left')

print("Final Merged Data (first 5 rows):\n", final_df.head())
print(final_df.columns)
print("Final DataFrame shape:", final_df.shape)
print(final_df.describe())

most_purchased_products = final_df.groupby('product_category_name')['product_id'].count().reset_index(name='purchase_count')
print("Top 5 most purchased products by category:\n", most_purchased_products.sort_values(by='purchase_count', ascending=False).head())
```

Result

```
Top 5 most purchased products by category:
      product_category_name  purchase_count
13          cama_mesa_banho           11115
11             beleza_saude            9670
32           esporte_lazer             8641
54         moveis_decoracao            8334
44  informatica_acessorios             7827
```

Top 5 product purchased by category

plan for interesing marketing strategy



Interpretation:

The bar chart highlights the top 5 product categories based on purchase counts.

1. cama_mesa_banho	11115
2. beleza_saude	9670
3. esporte_lazer	8641
4. moveis_decoracao	8334
5. informatica_acessorios	7827

Insight:

- The top-selling categories reflect broader consumer behavior: a focus on home improvement, self-care, fitness, and tech adoption.

Top 5 product category by total revenue

plan for interesing marketing strategy

We combine the orders, order_items, products, and customers datasets into a single DataFrame (final_df). This allows us to have all the necessary information (orders, products, and customers) in one place.

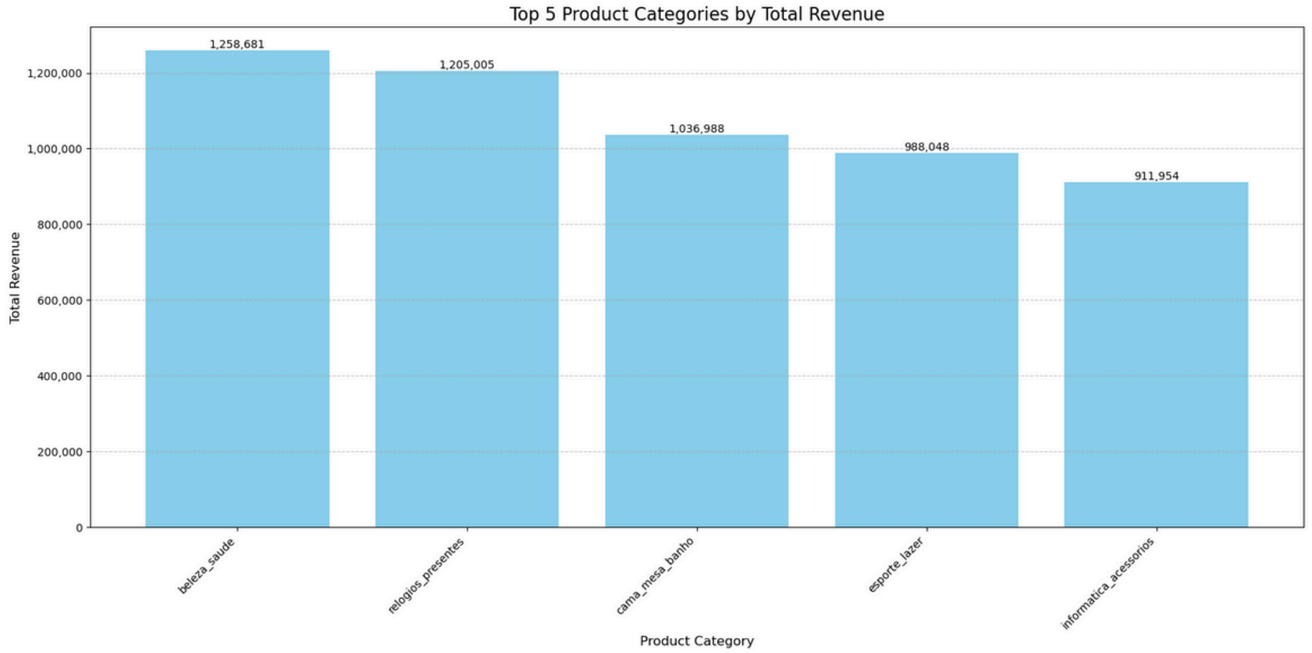
```
merged_orders_items = pd.merge(orders_df, order_items_df, on='order_id', how='left')
final_df = pd.merge(merged_orders_items, products_df, on='product_id', how='left')
final_df = pd.merge(final_df, customers_df, on='customer_id', how='left')

print("Final Merged Data (first 5 rows):\n", final_df.head())
print(final_df.columns)
print("Final DataFrame shape:", final_df.shape)
print(final_df.describe())

revenue_per_category = final_df.groupby('product_category_name')['price'].sum().reset_index(name='total_revenue')

print("Top 5 product categories by total revenue:\n", revenue_per_category.sort_values(by='total_revenue', ascending=False).head())
```

Result

```
[8 rows x 11 columns]
Top 5 product categories by total revenue:
      product_category_name  total_revenue
11            beleza_saude     1258681.34
66      relogios_presentes     1205005.68
13         cama_mesa_banho     1036988.68
32           esporte_lazer      988048.97
44  informatica_acessorios      911954.32
>>>
```

Top 5 product category by total revenue

plan for interesing marketing strategy



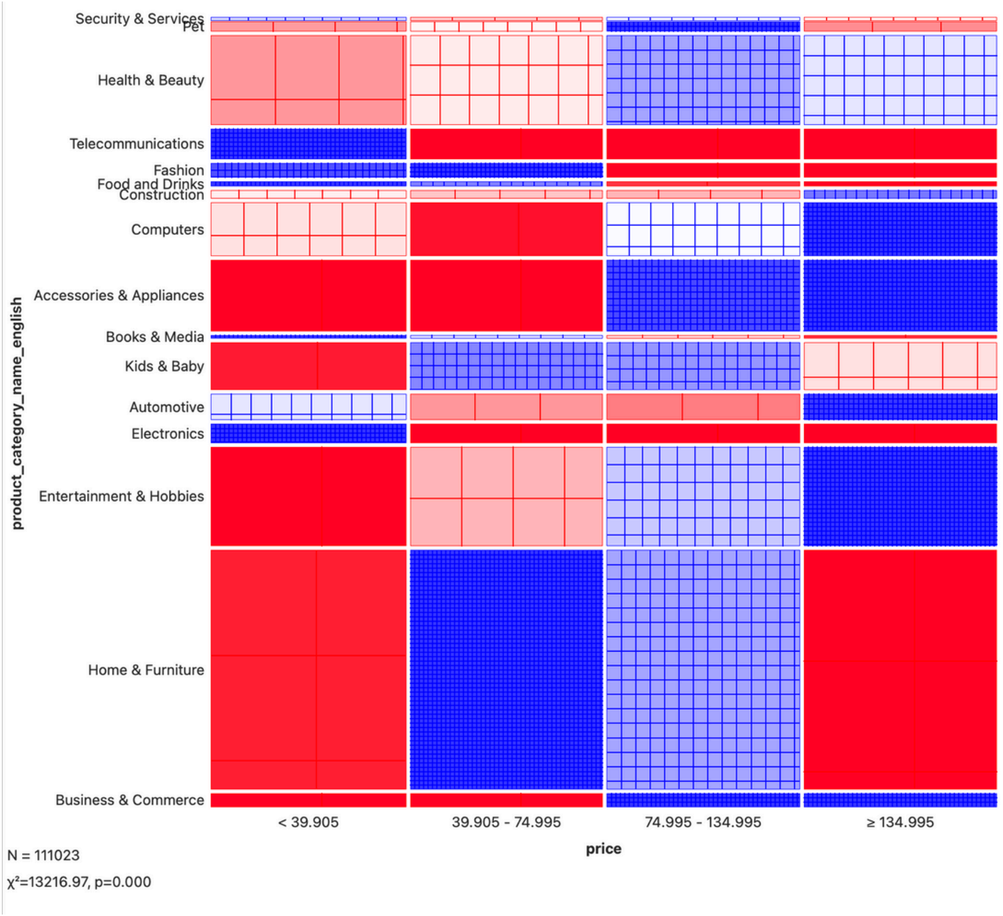
Interpretation:

The analysis of total revenue across product categories reveals that beleza_saude leads with a revenue of 1,258,681.34, closely followed by relogios_presentes at 1,205,005.68. cama_mesa_banho and esporte_lazer generated revenues of 1,036,988.68 and 988,048.97, respectively. Finally, informatica_acessorios contributed 911,954.32.

Insight:

- Beleza Saude leads both in units sold and revenue, indicating high demand and potentially high profit margins in this category.

Product Analysis



Product Category by Sales Revenue

Analysis

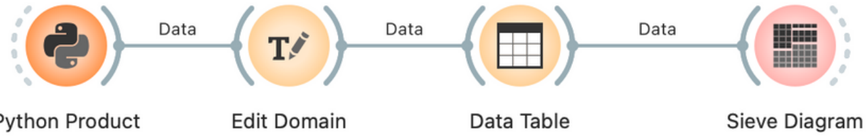
Price Range

- <40
  - Home Appliances (small items)
  - Entertainment and Hobbies
  - Kids and Baby
- 40-75
  - Entertainments and Hobbies
  - Computers
  - Accessories Other appliances
- 75-135
  - Fashion
  - Constructions
  - Food and Drinks
- >135
  - Home and Furniture
  - Electronics
  - Telecommunication

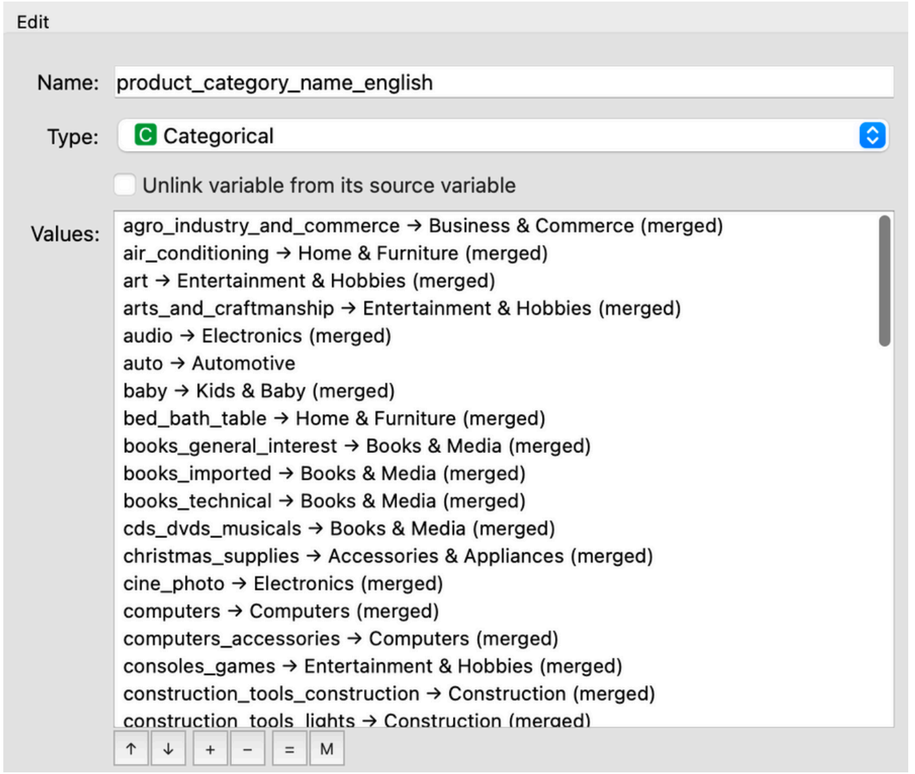
Recommendation

- Product Categories overrepresented (red) in low price (<40) can be said as price sensitive products, should be offered frequent promotion or discount.
- Mid-price to a bit high price products categories can be promoted through proper banding and loyalty program.
- Premium products categories can focus on personalization and customization, warranties services and special shipping.

Product Analysis WorkFlow



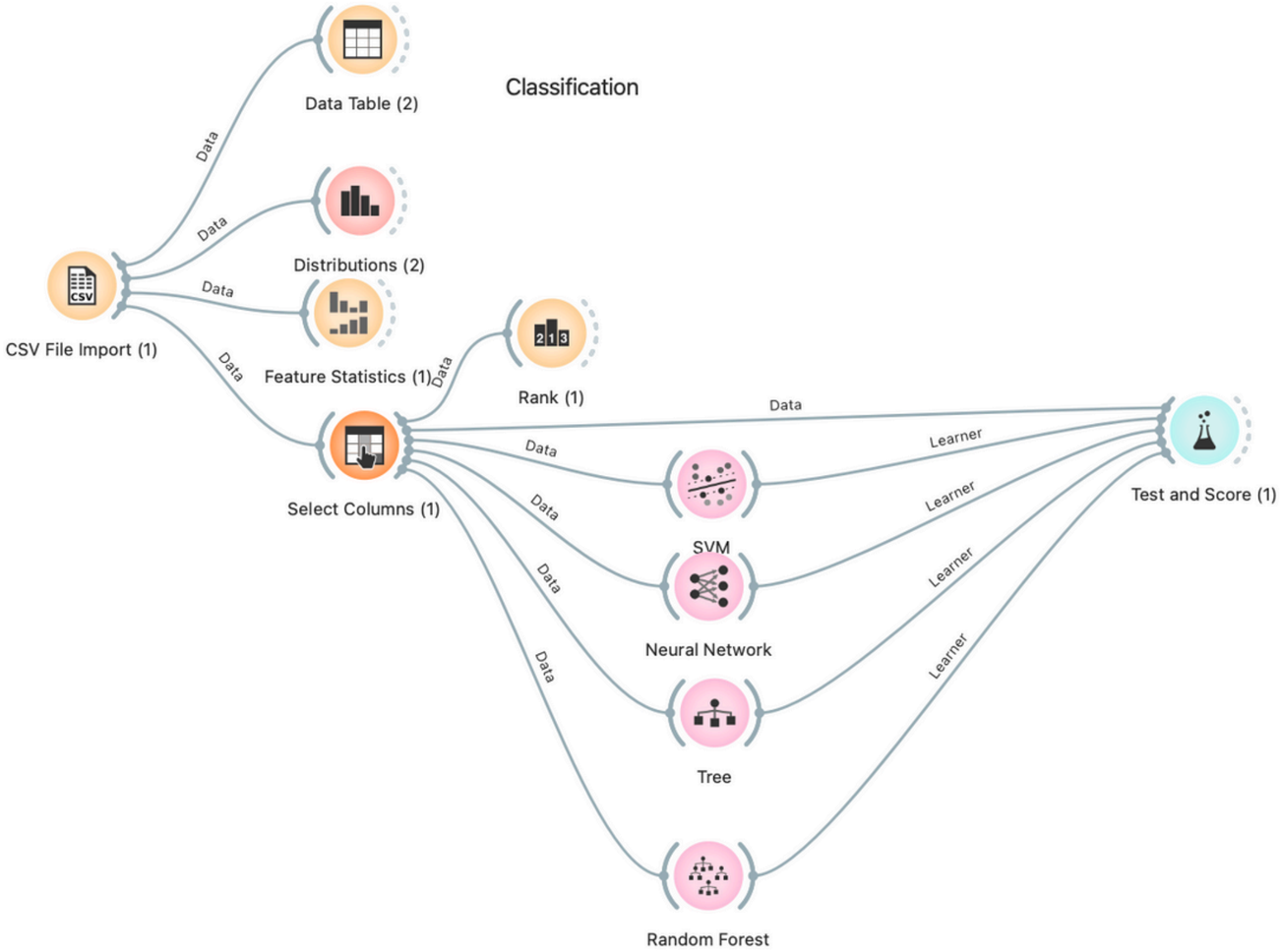

```
import pandas as pd
from Orange.data.pandas_compat import table_from_frame,table_to_frame

product = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_products_dataset.csv')
item = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_order_items_dataset.csv')
category = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/product_category_name_translation.csv')


merged_products = pd.merge(product[['product_id', 'product_category_name']],
category[['product_category_name', 'product_category_name_english']], on='product_category_name',
how='inner')

merged_data = pd.merge(item[['order_id', 'product_id', 'price']], merged_products[['product_id',
'product_category_name_english']], on='product_id', how='inner')

out_data = table_from_frame(merged_data)
```



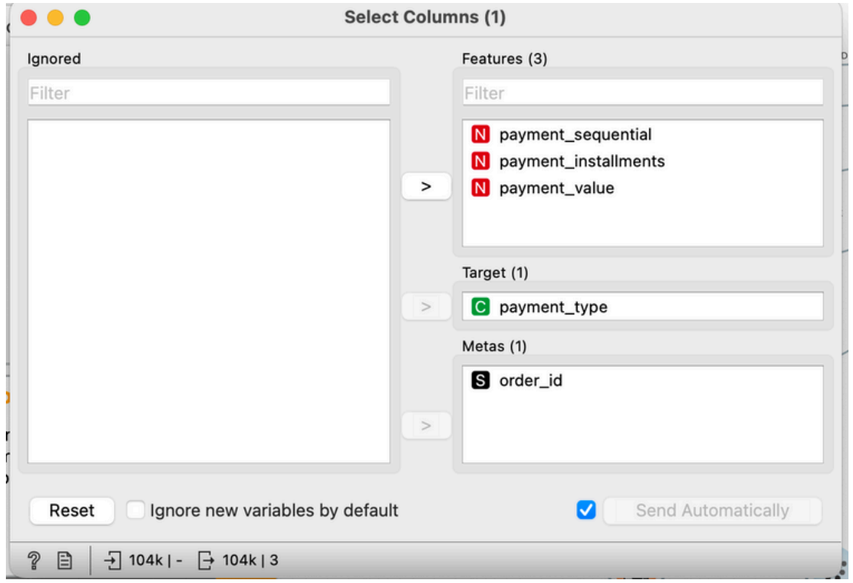
Steps

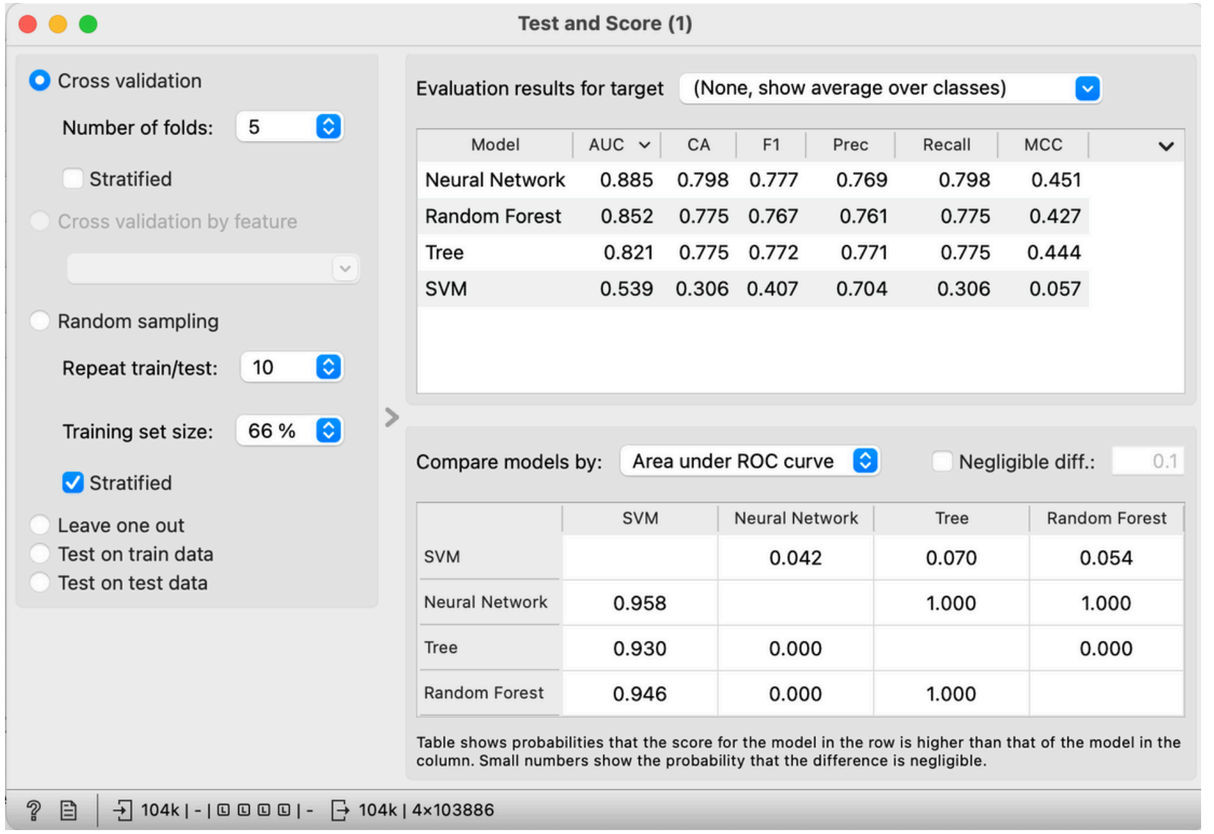
- merge item , product and category dataset
- edit domain >
  - merge similar item categories into one big category

Classification of payment dataset



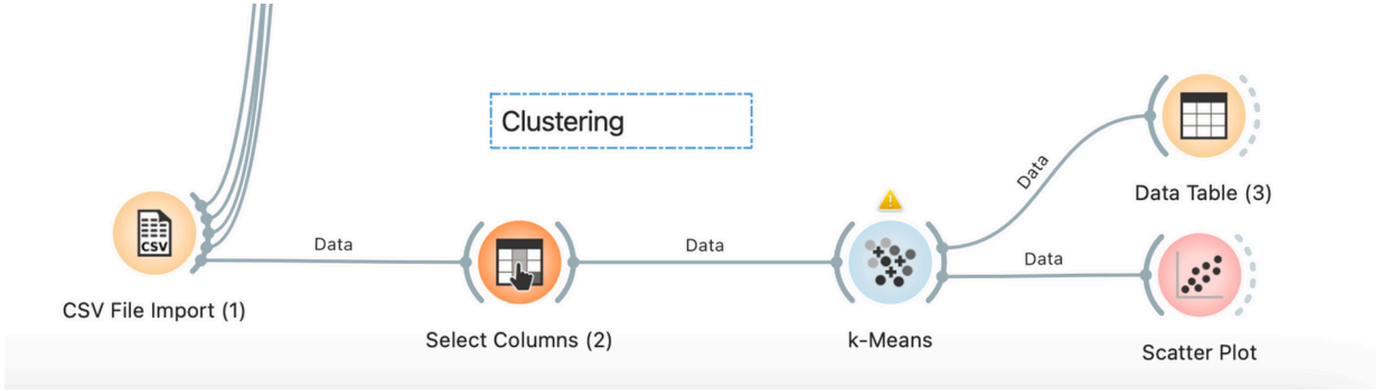
Classification of payment dataset

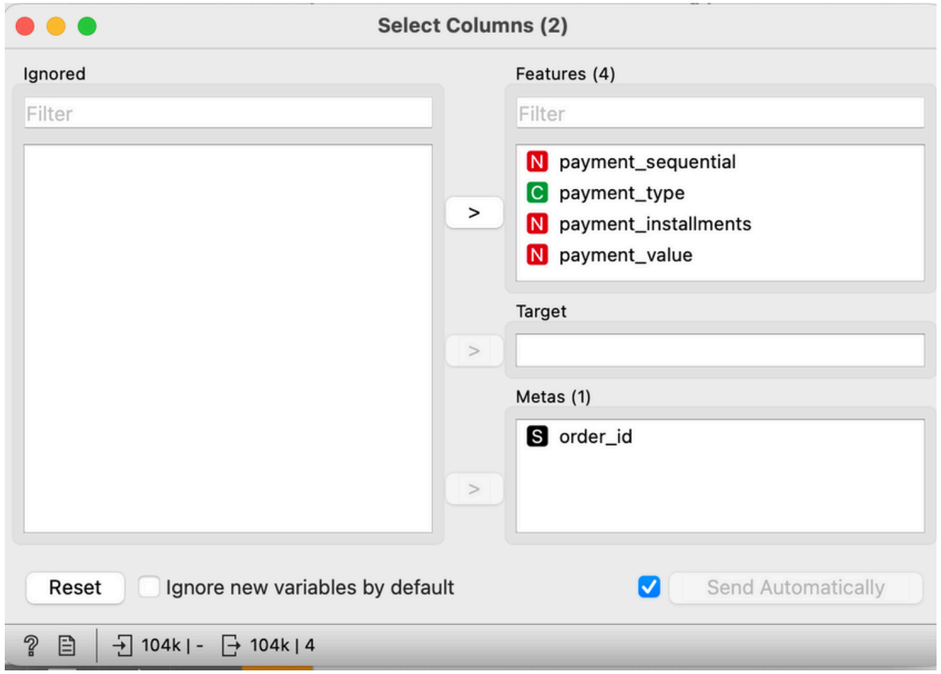


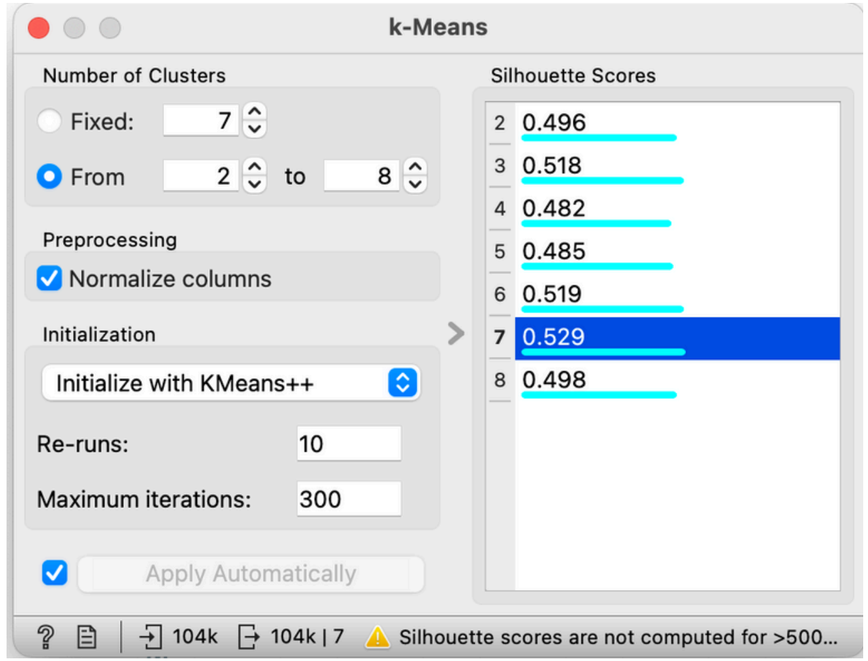


Neural Network is the best among the models with the accuracy of 88.5%.

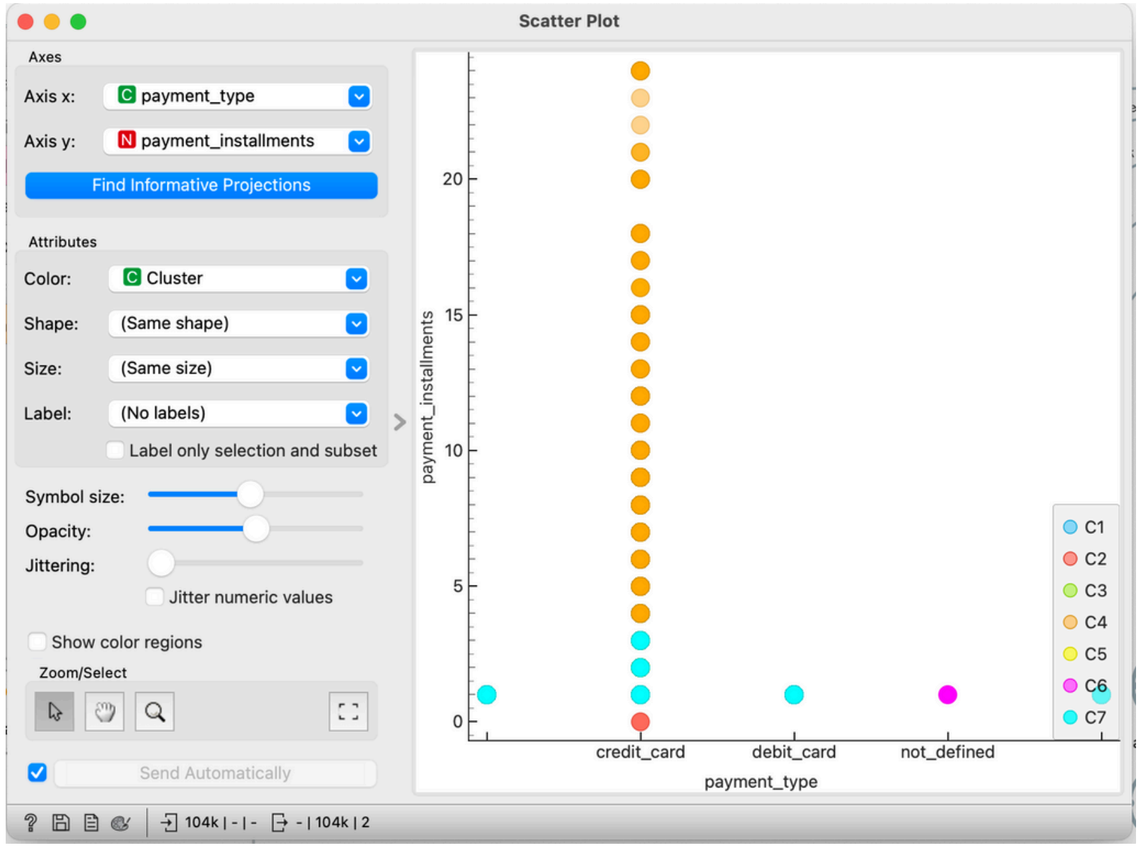
Clustering of payment dataset



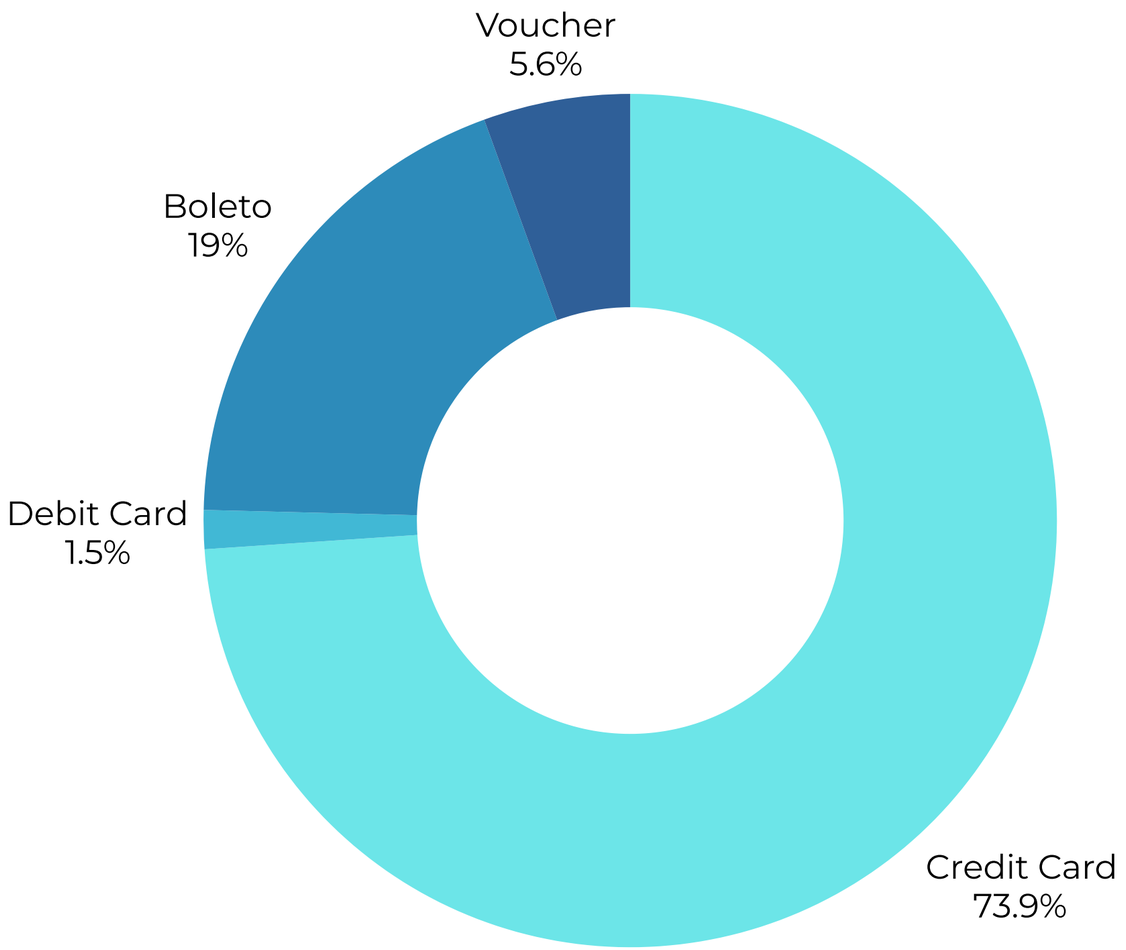




Clustering of payment dataset



try python for payment type



boleto : payment type in Brazil

Option 1 : Use filter statement
*redo 5 time for 5 type of payment type

Step 1 : filter-statement
filter1 = df["column_name"]== "keyword"

Step 2 : apply filter to look data and count how many row that have this word
df.loc[row-range,column-range]
df2=df.loc[filter1,:]
print(df2.count())

Option 2 : Use groupby command
*easier and show all payment type in 1 time

q1=df.groupby("payment_type")["payment_type"].size()
print(q1)

Result in Orange

Option 1 : Filter Statement

Boleto

```
[5 rows x 7 columns]
order_id                19784
payment_sequential      19784
payment_type            19784
payment_installments    19784
payment_value           19784
Unnamed: 5                  0
Unnamed: 6                  0
dtype: int64
```

Credit Card

```
[5 rows x 7 columns]
order_id                76795
payment_sequential      76795
payment_type            76795
payment_installments    76795
payment_value           76795
Unnamed: 5                  0
Unnamed: 6                  1
dtype: int64
```

Debit card

```
[5 rows x 7 columns]
order_id                1529
payment_sequential      1529
payment_type            1529
payment_installments    1529
payment_value           1529
Unnamed: 5                 0
Unnamed: 6                 0
dtype: int64
```

Voucher

```
[5 rows x 7 columns]
order_id                5775
payment_sequential      5775
payment_type            5775
payment_installments    5775
payment_value           5775
Unnamed: 5                 0
Unnamed: 6                 0
dtype: int64
```

Option 2 : Groupby command

```
payment_type
boleto          19784
credit_card     76795
debit_card       1529
not_defined         3
voucher          5775
Name: payment_type, dtype: int64
```

Not defined

```
[3 rows x 7 columns]
order_id                3
payment_sequential      3
payment_type            3
payment_installments    3
payment_value           3
Unnamed: 5              0
Unnamed: 6              0
dtype: int64
```

Correlation between payment_sequential and payment_value from Payment data set

```
Correlation between payment_sequential and payment_value:
                                payment_sequential  payment_value
payment_type
boleto       payment_sequential            1.000000      -0.001686
             payment_value                -0.001686       1.000000
credit_card  payment_sequential            1.000000      -0.004597
             payment_value                -0.004597       1.000000
debit_card   payment_sequential            1.000000      -0.006073
             payment_value                -0.006073       1.000000
not_defined  payment_sequential                 NaN            NaN
             payment_value                      NaN            NaN
voucher      payment_sequential            1.000000      -0.170374
             payment_value                -0.170374       1.000000
>>>
```

```
#Correlation between payment_sequential and payment_value

corr_seq_value=payment.groupby('payment_type')
[['payment_sequential','payment_value']].corr()

print("\nCorrelation between payment_sequential and
payment_value:\n",corr_seq_value)
```

Average payment per status

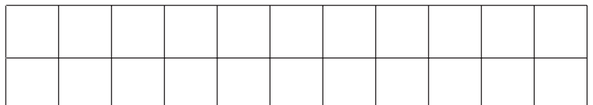
```
Average payment per status:
 order_status
processing      217.536395
canceled        215.746386
invoiced        212.732277
unavailable     194.883683
delivered       153.067428
shipped         151.984528
created         137.620000
approved        120.540000
Name: payment_value, dtype: float64
>>>
```

```
merged_df = pd.merge(payment, orders, on='order_id', how='inner')
avg_payment_per_status = merged_df.groupby('order_status')
['payment_value'].mean().sort_values(ascending=False)
print("\nAverage payment per status:\n",avg_payment_per_status)
```

Distribution on payment type “Delivered”

```
credit_card     74586
boleto          19191
voucher          5493
debit_card       1486
Name: payment_type, dtype: int64
>>>
```

```
delivered_payment_type_distribution = merged_df[merged_df['order_status'] == 'delivered']
['payment_type'].value_counts()
print(delivered_payment_type_distribution)
```



Total payment value per payment type

```
Total payment value per payment type:
 payment_type
boleto          2869361.27
credit_card    12542084.19
debit_card       217989.79
not_defined           0.00
voucher          379436.87
Name: payment_value, dtype: float64
>>>
```

```
total_payment_per_type = payment.groupby('payment_type')['payment_value'].sum()
print("\nTotal payment value per payment type:\n",total_payment_per_type)
```

try python for Customer State

for marketing strategy and inventory plan in each state

*know only the highest one

Way 1 : find highest value and state by decending order

```
q1=df.groupby('customer_state')
['customer_state'].size().sort_values(ascending=False)
```

Way 2 : find highest value and state by decending order and type in sentence

```
state=df.groupby('customer_state')['customer_state'].size().idxmax()
value =df.groupby('customer_state')['customer_state'].size().max()

print(f'the state: {state} has the most customer {value}')
```

Way 1

Console

```
customer_state
SP    41746
RJ    12852
MG    11635
RS     5466
PR     5045
SC     3637
BA     3380
DF     2140
ES     2033
```

Way 2

```
the state: SP has the most customer 41746
```

Visualization for Customer State

Calculate for Percentage

def percentage(part, whole):
return (part / whole) * 100

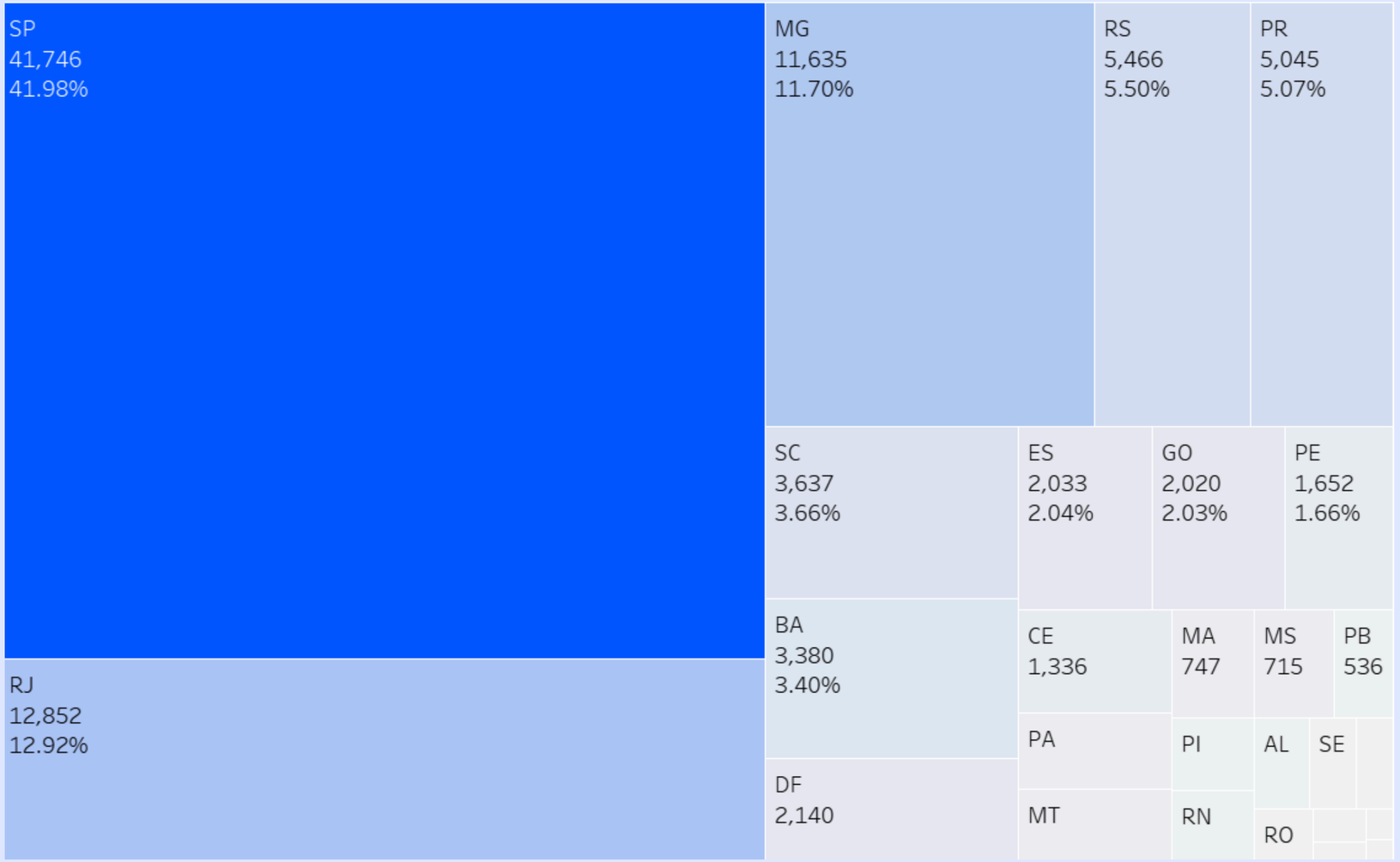
result = percentage(no. of that state, whole)
print(result)

Example : SP
result = percentage(41746, 99441)
print(result)

whole = 99441

SP	41746	41.98%
RJ	12852	12.92%
MG	11635	11.70%
RS	5466	5.50%
PR	5045	5.07%
SC	3637	3.66%
BA	3380	3.40%
DF	2140	2.15%
ES	2033	2.04%
GO	2020	2.03%
PE	1652	1.66%
CE	1336	1.34%
PA	975	0.98%
MT	907	0.91%

MA	747	0.75%
MS	715	0.72%
PB	536	0.54%
PI	495	0.50%
RN	485	0.49%
AL	413	0.42%
SE	350	0.35%
TO	280	0.28%
RO	253	0.25%
AM	148	0.15%
AC	81	0.08%
AP	68	0.07%
RR	46	0.05%



for percentage, using excel for looking easier

Numbers of sellers from each state

```
Numbers of sellers from each state:
 seller_state
AC       1
AM       1
BA      19
CE      13
DF      30
ES      23
GO      40
MA       1
MG     244
MS       5
MT       4
PA       1
PB       6
PE       9
PI       1
PR     349
RJ     171
RN       5
RO       2
RS     129
SC     190
SE       2
SP    1849
Name: seller_id, dtype: int64
>>>
```



```
#How many sellers are there in each state?
No_sellers=seller.groupby('seller_state')['seller_id'].count()
print("\nNumbers of sellers from each state:\n",No_sellers)
```

Average delivery times

plan for interesing marketing strategy

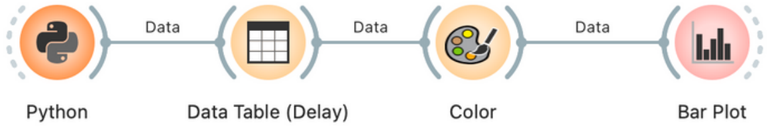
We combine the orders, order_items, products, and customers datasets into a single DataFrame (final_df). This allows us to have all the necessary information (orders, products, and customers) in one place.

```
merged_orders_items = pd.merge(orders_df, order_items_df, on='order_id', how='left')
final_df = pd.merge(merged_orders_items, products_df, on='product_id', how='left')
final_df = pd.merge(final_df, customers_df, on='customer_id', how='left')

print("Final Merged Data (first 5 rows):\n", final_df.head())
print(final_df.columns)
print("Final DataFrame shape:", final_df.shape)
print(final_df.describe())

final_df['order_purchase_timestamp'] = pd.to_datetime(final_df['order_purchase_timestamp'])
final_df['order_delivered_customer_date'] = pd.to_datetime(final_df['order_delivered_customer_date'])

final_df['delivery_time_days'] = (final_df['order_delivered_customer_date'] - final_df['order_purchase_timestamp']).dt.days

average_delivery_time = final_df['delivery_time_days'].mean()
print(f"Average delivery time: {average_delivery_time:.2f} days")
```

Result

```
[8 rows x 11 columns]
Average delivery time: 12.01 days
>>>
```

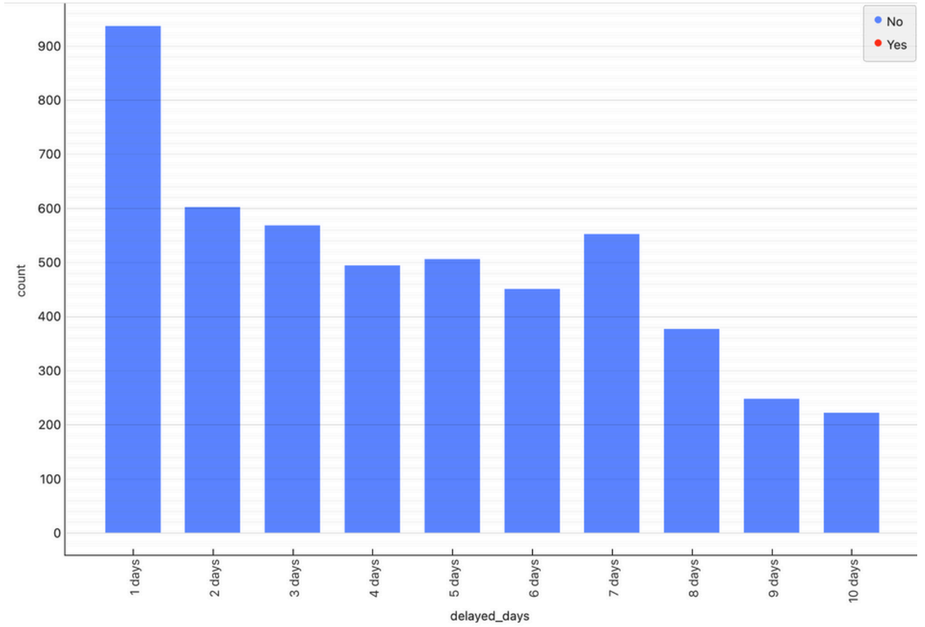
Delayed Date




```
import pandas as pd
from Orange.data.pandas_compat import table_from_frame,table_to_frame

product = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_products_dataset.csv')
item = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_order_items_dataset.csv')
order = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_orders_dataset.csv')

df = pd.merge(left=order,right=pd.merge(left=item,right=product,on="product_id"),on="order_id")

df = df.loc[df["order_delivered_customer_date"] > df["order_estimated_delivery_date"]].copy()
df["delivered_date"] = pd.to_datetime(df["order_delivered_customer_date"]).dt.date
df["estimated_date"] = pd.to_datetime(df["order_estimated_delivery_date"]).dt.date
df = df.loc[:, ["order_id","delivered_date","estimated_date"]]
df["delayed"] = df["delivered_date"] - df["estimated_date"]
df["delayed_days"] = pd.to_timedelta(df["delayed"])
df['delayed'] = pd.to_timedelta(df['delayed']).dt.days
df = df.loc[(df['delayed'] >=1) & (df["delayed"] <= 10)]
dfs = df.groupby(["delayed_days"])["delayed"].count()
df = dfs.reset_index()
df["count"] = df["delayed"]
df = df.loc[:,["delayed_days","count"]]

out_data = table_from_frame(df)



df1 = df.loc[df["delayed"]> 10]["delayed"].count()
print(df1)
```

order delayed more than 10 days = 2298

Delayed Date Count



Steps

- extract product , item and order dataset and merge 3 datasets
- take as order delay if order is not delivered on estimated date
- group by and extract delay date from 1 days to 10 days
- count the delayed days

Insights

- The most delayed days
  - = 1 days
- followed by 2 days , 3 days , 7days etc.
- Delay days greater than 10 days should be reviewed

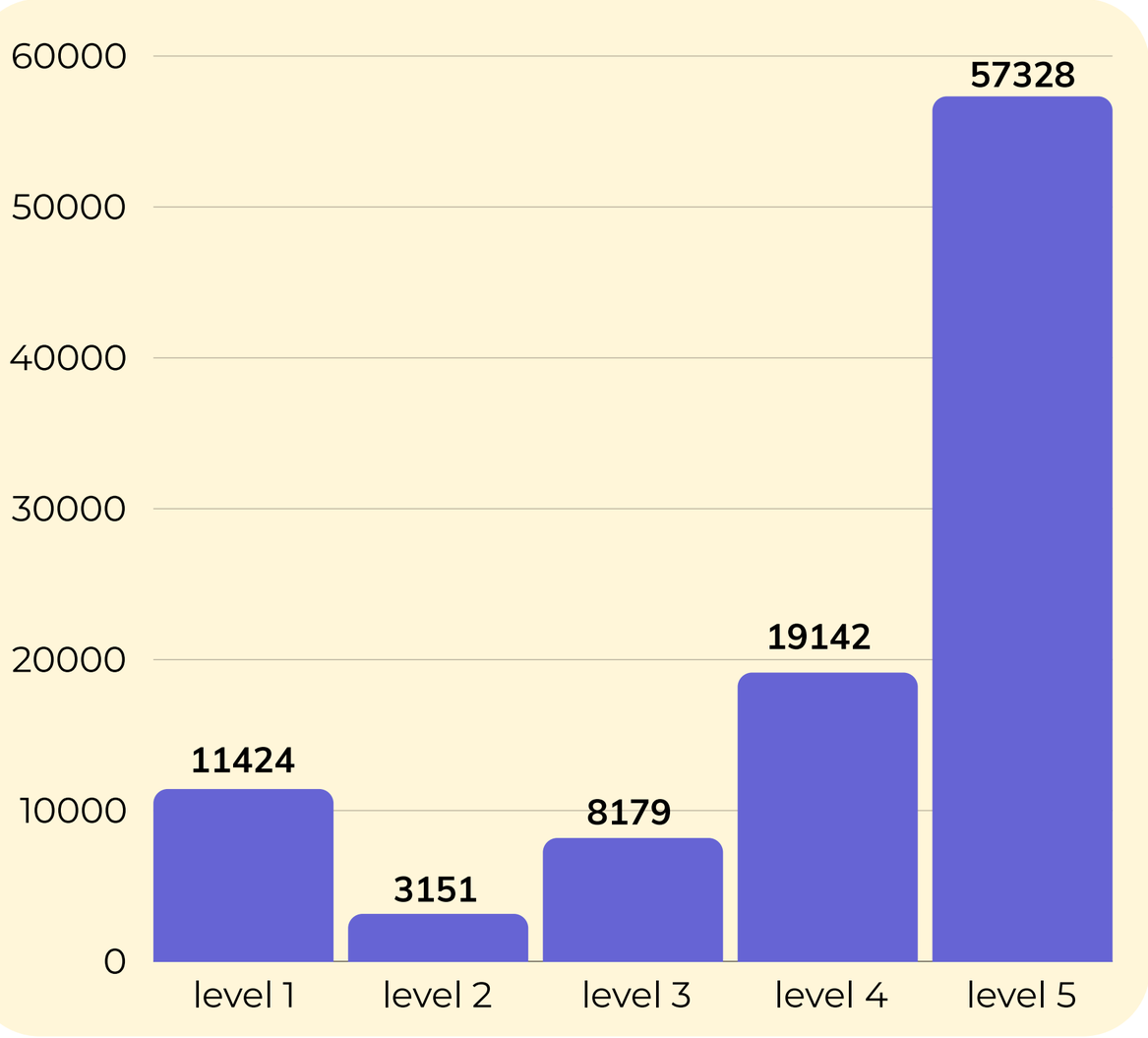
try python for Order Review

Find the satisfy level of the customer from order review to know that **‘Do we really need to develop our own services?’** (*from 99224 reviews)

```
q1=df.groupby('review_score')
['review_score'].size().sort_values(ascending=False)
print(q1)
```

Normally, customer satisfy in our serveice but we still have ***some customer that feel not satisfy*** in our service in some part, we should improve ourself in that field.

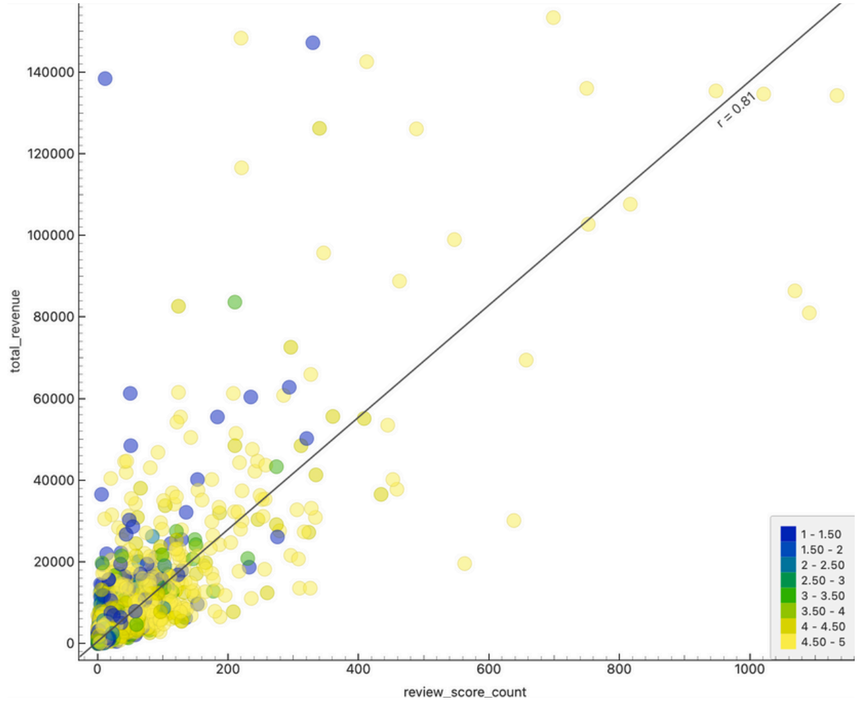
review_score	
1	11424
2	3151
3	8179
4	19142
5	57328



Review and Sales


```python
import pandas as pd
from Orange.data.pandas_compat import table_from_frame,table_to_frame

item = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_order_items_dataset.csv')
review = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_order_reviews_dataset.csv')
payment = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_order_payments_dataset.csv')

merged_data = pd.merge(review[['order_id', 'review_score']], payment[['order_id', 'payment_value']],
on='order_id', how='inner')

merged_sellers = pd.merge(merged_data, item[['order_id', 'seller_id']], on='order_id', how='inner')

result = merged_sellers.groupby(['seller_id','review_score']).agg(
    review_score_count =('review_score', 'count'),
    num_orders=('order_id', 'nunique'),
    total_revenue=('payment_value', 'sum')).reset_index()

out_data = table_from_frame(result)
```

Sales Revenue and review count



Features (3)

Filter

- review_score
- review_score_count
- num_orders

Target (1)

- total_revenue

Metas (1)

- seller_id

Model	MSE	RMSE	MAE	MAPE	R2
Linear Regression	20270...	4502.3...	1332....	2.581	0.650

- Linear regression model can only explain 65% of the total revenue.
- Indicating other factors also impact on total revenue generated by each sellers other than review.

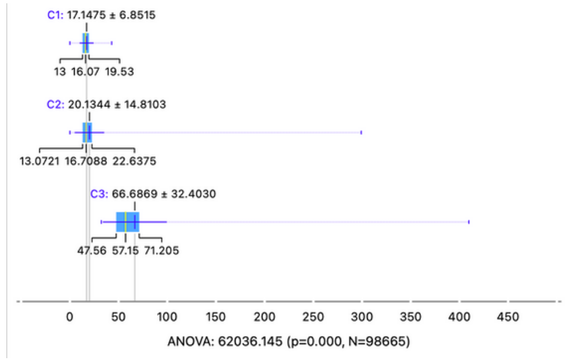
Steps

- merge item, review, payment
- group by seller , review score
- review score count
- number of order > count order_id
- total revenue > sum payment value

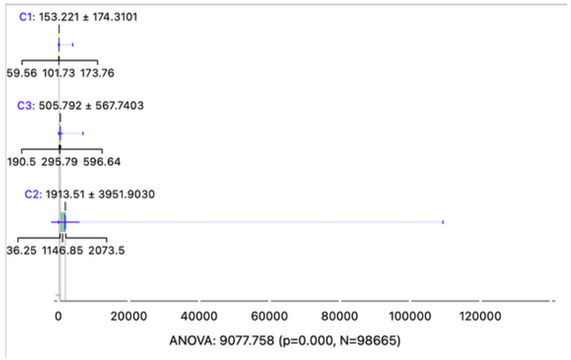
Insights

- according to the plot
- Blue and Green (score: 1-3)
  - 0 - 60,000 sales revenue
- Yellow (score: 3-5)
  - more sales 0- 150,000

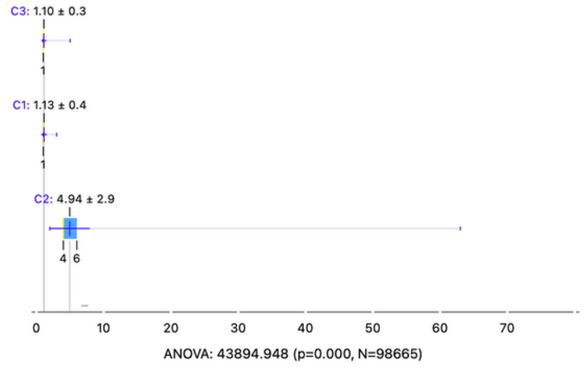
Clustering Sales, Qty and Shipping



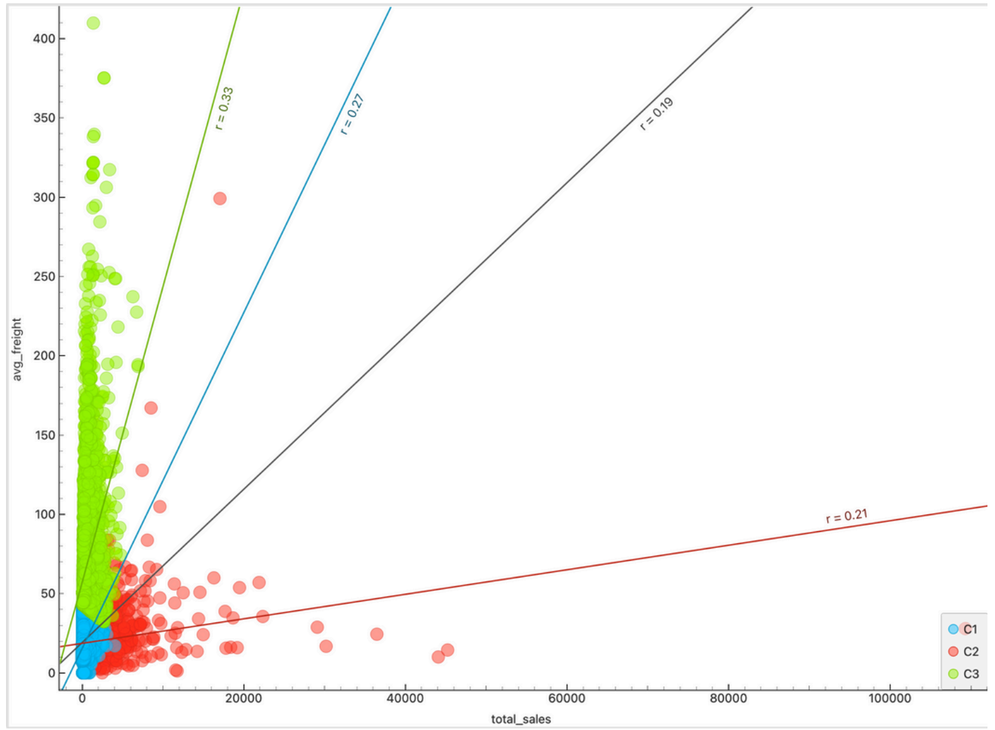
Average shipping by Clusters



Total Sales by Clusters



Number of Items by Clusters



Cluster C1 (Blue)

- r = 27%
- small total sales
- small shipping cost
- can be ordinary size products with free or low shipping fees
- small number of items

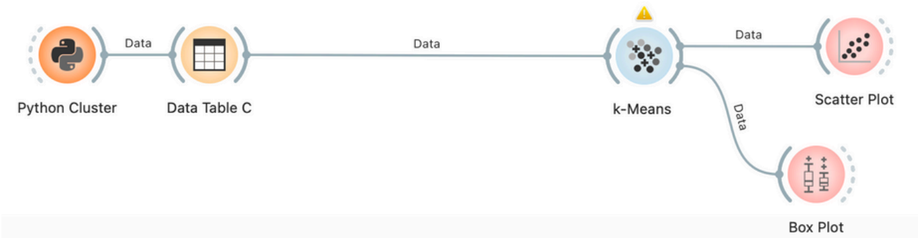
Cluster C2 (Red)

- r= 21%
- high total sales
- small shipping cost
- can be small lightweight products with high cost
- large number of items

Cluster C3 (Green)

- r = 33%
- small total sales
- high shipping cost
- moderate cost with large products > high shipping fees
- small number of items

Clustering Shipping Workflow




```python
import pandas as pd
from Orange.data.pandas_compat import table_from_frame,table_to_frame

product = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_products_dataset.csv')
item = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_order_items_dataset.csv')
order = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_orders_dataset.csv')
payment = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_order_payments_dataset.csv')


merged_data = pd.merge(order[['order_id', 'order_purchase_timestamp']], item[['order_id', 'price',
'freight_value']], on='order_id', how='inner')

merged_data = pd.merge(merged_data, payment[['order_id', 'payment_value']], on='order_id', how='inner')

df = merged_data.groupby('order_id').agg(
    total_sales=('payment_value', 'sum'),
    num_items=('price', 'count'),
    avg_freight=('freight_value', 'mean')
).reset_index()

out_data = table_from_frame(df)
```

Dataset Used

- order
- Item
- payment

Column Used

- order_purchase_timestamp
- price
- freight_value
- payment_value

Data Manipulation

- merge 3 dataset
- group by order_id
- Sum of Payment Value
- Count of Price to get number of items
- Average Fright Value

Annual Revenue:

plan for interesing marketing strategy

We combine the orders, order_items, products, and customers datasets into a single DataFrame (final_df). This allows us to have all the necessary information (orders, products, and customers) in one place.

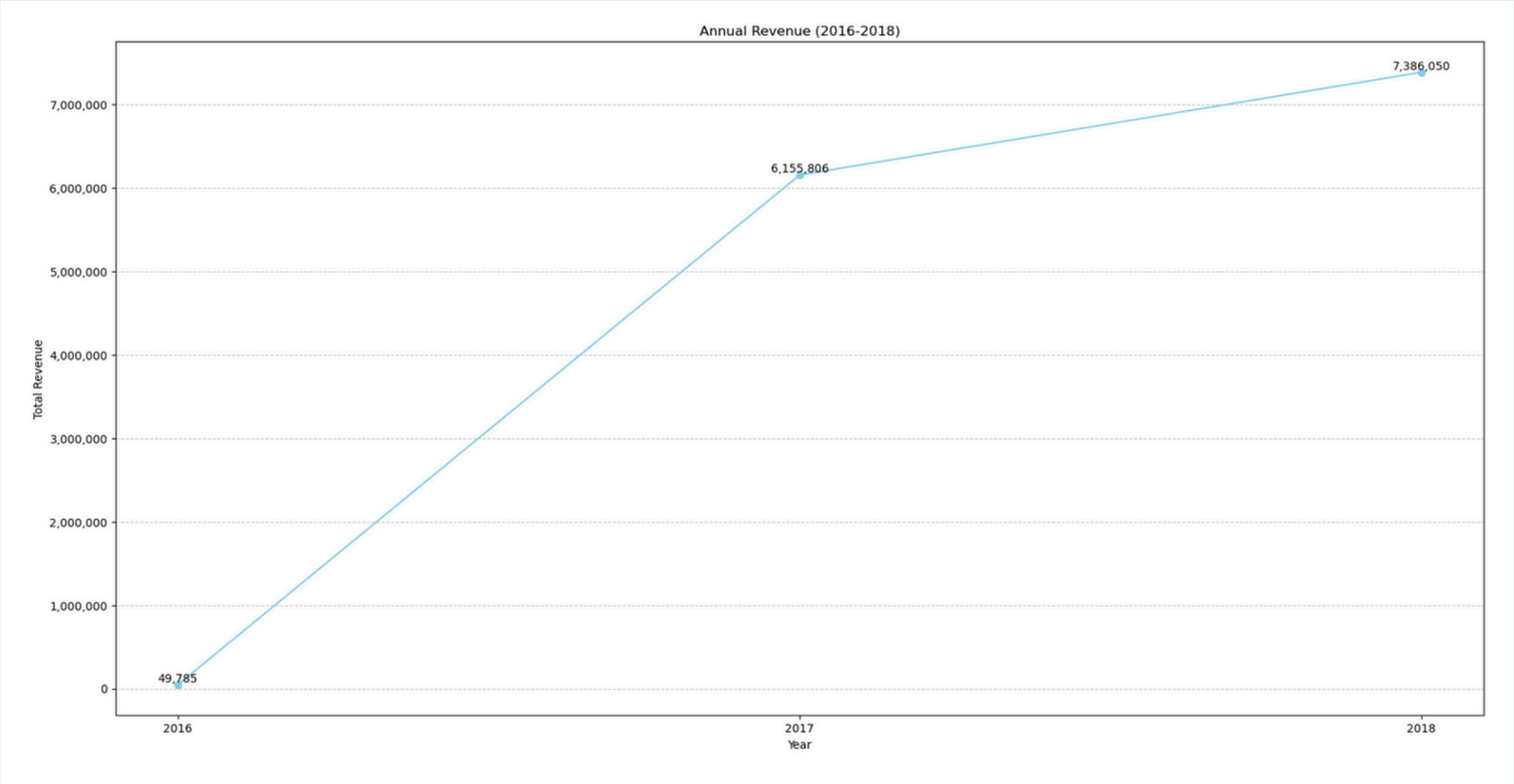
```
final_df['order_purchase_timestamp'] = pd.to_datetime(final_df['order_purchase_timestamp'])
final_df['order_delivered_customer_date'] = pd.to_datetime(final_df['order_delivered_customer_date'])

final_df['delivery_time_days'] = (final_df['order_delivered_customer_date'] - final_df['order_purchase_timestamp']).dt.days

final_df['year'] = final_df['order_purchase_timestamp'].dt.year

annual_revenue = final_df.groupby('year')['price'].sum().reset_index(name='total_revenue')

print("Annual Revenue:\n", annual_revenue)
```

Result

```
[8 rows x 11 columns]
Annual Revenue:
    year  total_revenue
0  2016        49785.92
1  2017      6155806.98
2  2018      7386050.80
>>>
```

Annual Revenue:

plan for interesing marketing strategy



Interpretation:

There is a significant difference in revenue between 2016 and the following years, 2017 and 2018. In 2016, the total revenue was 49,785.92, covering only September to December during the initial phase of operations. In contrast, revenues for 2017 and 2018 were much higher, at 6,155,806.98 and 7,386,050.80, respectively. This growth reflects a full year of operations and an expanding customer base.

Insight:

- The revenue trends show positive growth, but also indicate that Olist is reaching a more mature phase. To keep growing, the company will need to innovate, either by expanding its product lines, entering new markets, or improving customer retention and operational efficiency.

Annual Revenue for 2017

plan for interesing marketing strategy

We combine the orders, order_items, products, and customers datasets into a single DataFrame (final_df). This allows us to have all the necessary information (orders, products, and customers) in one place.

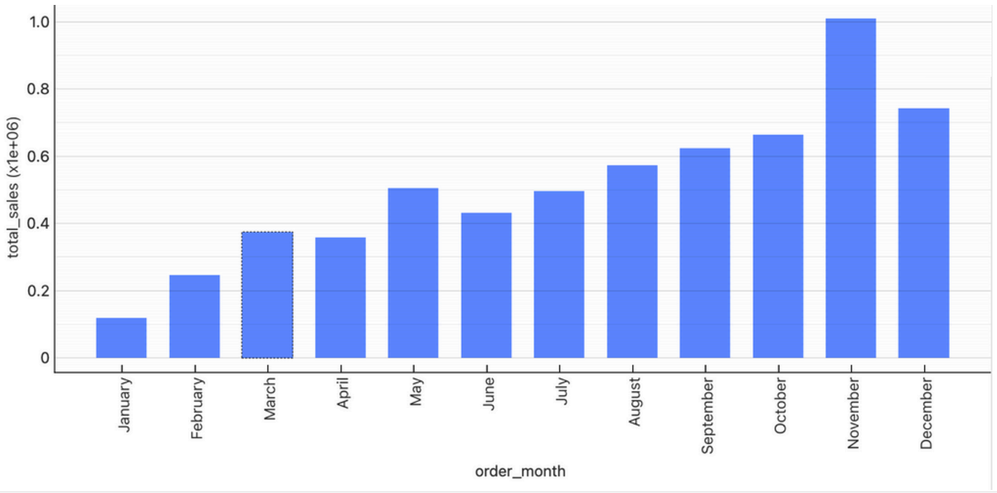
```
final_df['order_purchase_timestamp'] = pd.to_datetime(final_df['order_purchase_timestamp'])
final_df['order_delivered_customer_date'] = pd.to_datetime(final_df['order_delivered_customer_date'])

final_df['delivery_time_days'] = (final_df['order_delivered_customer_date'] - final_df['order_purchase_timestamp']).dt.days

final_df['year'] = final_df['order_purchase_timestamp'].dt.year

annual_revenue = final_df.groupby('year')['price'].sum().reset_index(name='total_revenue')

annual_revenue_2017 = annual_revenue[annual_revenue['year'] == 2017]

print("Annual Revenue for 2017:\n", annual_revenue_2017)
```

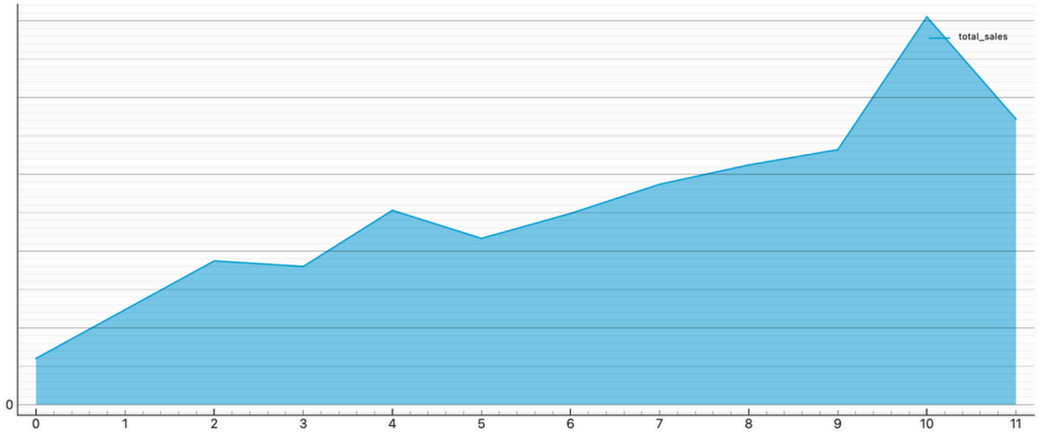
Result

```
Annual Revenue for 2017:
     year  total_revenue
1  2017      6155806.98
>>>
```

Sales Trend Analysis - Year 2017

Monthly Total Sales Trend for the Year 2017





Analysis

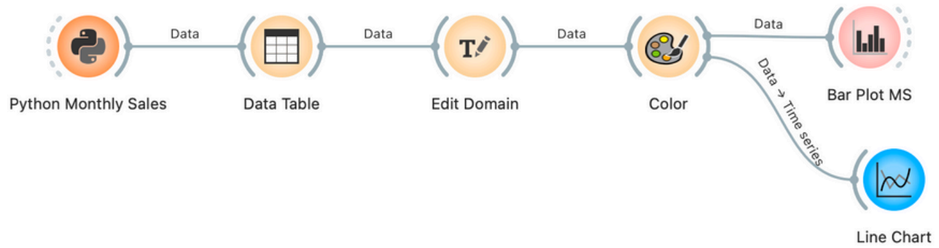
According to the data

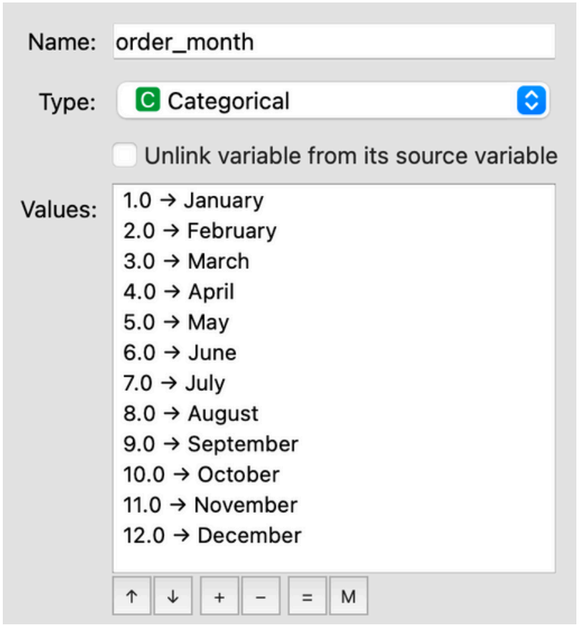
- Highest Sales Volume occurs in November
- Lowest Sales Volume occurs in January

Recommendation

- For High Sales Season
  - Increase Sever Load Capacity
  - Increase Shipping Efficiency
- For Low Sales Season
  - Clearance Sales
  - Referral Discount
  - Free Shipping Period

Sales Trend WorkFlow





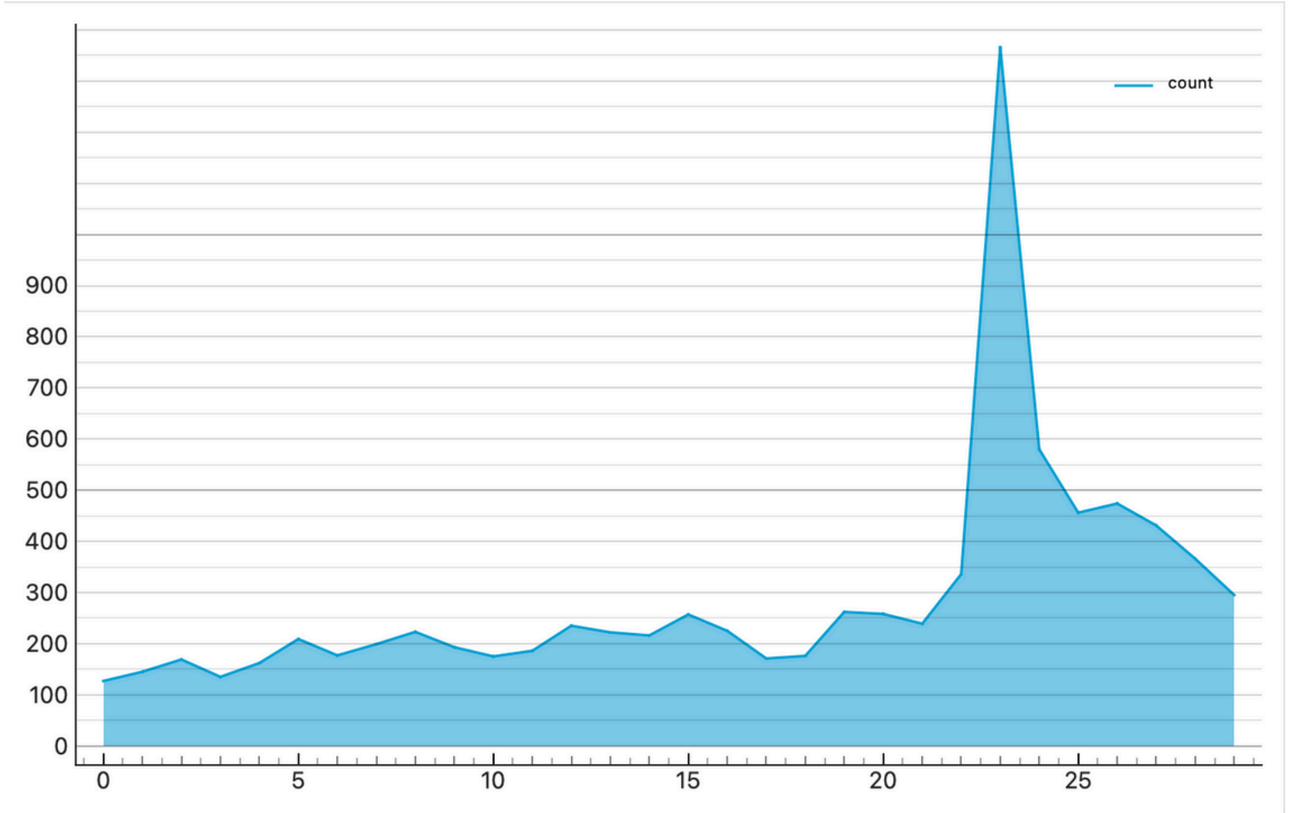

```python
import pandas as pd
from Orange.data.pandas_compat import table_from_frame,table_to_frame

item = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_order_items_dataset.csv')
order = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/olist_orders_dataset.csv')

sale_by_year = pd.merge(order,item,on="order_id")

sale_by_year["order_year"] = pd.to_datetime(sale_by_year["order_purchase_timestamp"]).dt.year
sale_by_year

sale_monthly_2017 = sale_by_year.loc[sale_by_year["order_year"] == 2017]

sale_monthly_2017 = sale_monthly_2017.copy()
sale_monthly_2017["order_month"] = pd.to_datetime(sale_monthly_2017["order_purchase_timestamp"]).dt.month
sale_monthly_2017 = sale_monthly_2017.groupby("order_month")["price"].sum().reset_index()
sale_monthly_2017["total_sales"]=sale_monthly_2017["price"]
sale_monthly_2017 = sale_monthly_2017.loc[:,["order_month","total_sales"]]

out_data = table_from_frame(sale_monthly_2017)
```

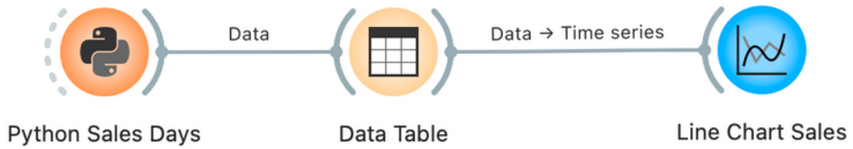
Steps

- merge item and order dataset
- extract 2017 sales
- group by month
- sum price to get total sales

Sales Trend Analysis - Nov 2017

Daily Total Sales Trend For November 2017






```
import pandas as pd
from Orange.data.pandas_compat import table_from_frame,table_to_frame

item = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/
olist_order_items_dataset.csv')
order = pd.read_csv('/Users/myatthuthukyaw/Downloads/ecom/
olist_orders_dataset.csv')

sale_by_year = pd.merge(order,item,on="order_id")

sale_by_year["order_year"] =
pd.to_datetime(sale_by_year["order_purchase_timestamp"]).dt.year
sale_by_year["order_month"] =
pd.to_datetime(sale_by_year["order_purchase_timestamp"]).dt.month
sale_by_year["order_day"] =
pd.to_datetime(sale_by_year["order_purchase_timestamp"]).dt.day

sale_monthly_2017 = sale_by_year.loc[sale_by_year["order_year"] == 2017]
sale_monthly_2017 =
sale_monthly_2017[sale_monthly_2017["order_month"]==11].groupby(["order_day"])
.size().reset_index(name="count")

out_data = table_from_frame(sale_monthly_2017)
```

Analysis

According to the data

- Highest Sales Volume occurs in November

- In November 2017
  Highest Sales Volume is made in
  - 24 November , 2017
  - which is **Black Friday Sales**
  - **with highest number of orders**
  - **1366 orders**

Most Frequent Order day of the week

```
Most Frequent Order Day of the Week:
 Monday        16196
Tuesday        15963
Wednesday      15552
Thursday       14761
Friday         14122
Sunday         11960
Saturday       10887
Name: order_day_of_week, dtype: int64
>>>
```

```
orders['order_purchase_timestamp'] = pd.to_datetime(orders['order_purchase_timestamp'])
orders['order_day_of_week'] = orders['order_purchase_timestamp'].dt.day_name()
order_day_distribution = orders['order_day_of_week'].value_counts()

print("\nMost Frequent Order Day of the Week:\n",order_day_distribution)
```

Business Insights and Recommendation

Most Customers and Seller comes from **SP (São Paulo) State.** Most used payment methods - **credit card**. Peak Sales Season is **Black Friday in November**. Least Sales in January. Most popular product category **Beauty & Health, Home and Furniture , and Sports & Entertainment.**

01. Regional Sales

The firm should expand their market on **underrepresented regions.** Since most of the sales are densely populated in **São Paulo State.**

02. Payment Marketing

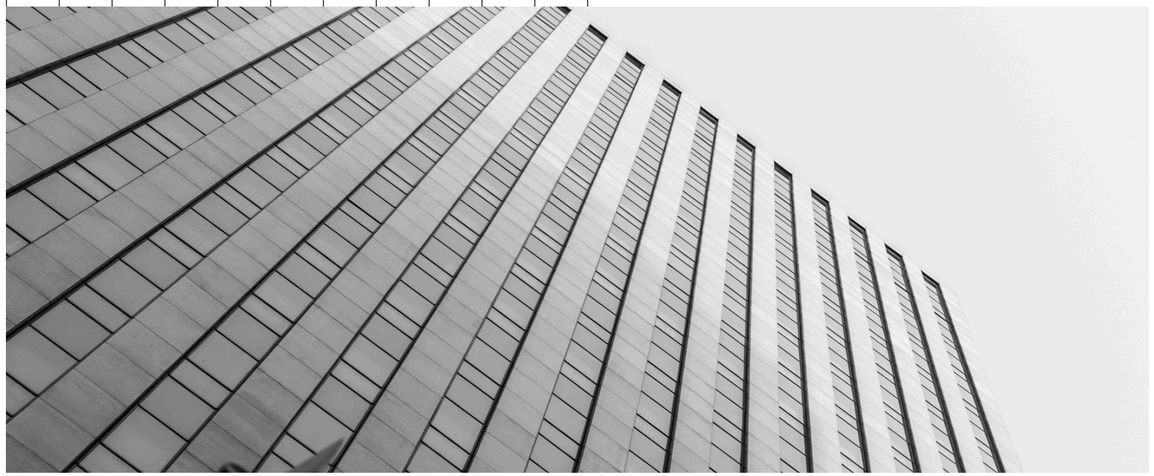
The analysis showed that **credit card** is most used payment method. The firm can attract more customers on high value products by providing **no-interest installment payments.**

03. Seasonal Sales

Since the firm gain highest sales volumes in **November** prepare and extend the sales period like **Pre-Black Friday Sales**. For the lowest sales season like January, promote through **New Year clearance sales** and **free shipping** to attract more customers.

04. Shipping Optimization

Since, average shipping periods is **12 days**, the firm should adjust to optimize for the faster shipping days to gain better customer satisfaction. Also, some pattern shows **high shipping cost** with low sales volumes, the firm should adjust that to elevate the sales.



Limitation for Further Research

Model Optimization

In this study, model **explain only around 50-65%** of the target values, which is not a good model. That is needed to review and **optimize accuracy** in further research.

Deeper Analysis Scope

For further research, deeper analysis on customer can be performed to find the patterns of customer behavior. Further suggested topics are **Customer Life Time Values, RFM Analysis** and **Customer Churn Prediction**.

Data Visualization

There is significant **limitation** in data visualization since the research is carried out under **low code environment.** Further research can be done with other coding tools or visualization tools to **customize the visualization.**

Thank You

For Your Attention

BDM3303 Data Mining
SEC 401