Final Report

Modelling & Predictive Analytics for price in E-commerce

By

Aarti Sampat Kadu

Acknowledgements

We would like to extend our heartfelt thanks to all those who supported us in completing our capstone project on “Modelling & Predictive Analytics for Price in E-commerce ". We are grateful for the guidance and support of our faculty advisor, the resources provided by the Kaggle community, the dedication of our team members, the support of our families and friends, and the opportunities provided by our educational institution. We also acknowledge the contributions of other individuals and organizations that may have helped us along the way. Your support has been invaluable in shaping the outcome of our project. Thank you for your belief in our abilities and your encouragement throughout this journey. We certify that the work done by us for conceptualizing and completing this project is original and authentic.

Sincerely,

Aarti Kadu

Table of Contents

Overview

The project revolves around analyzing a Brazilian E-Commerce Public Dataset by Olist, which contains information on 100,000 orders made at the Olist Store from 2016 to 2018. The dataset encompasses various dimensions of orders, including product details, customer information, order status, payment details, and reviews.

The primary objective of our project is to forecast prices for the e-commerce business by leveraging historical purchase information. This involves understanding sales trends, identifying patterns, and recognizing seasonality in the data. Additionally, we aim to utilize predictive models such as machine learning algorithms to enhance the accuracy of price predictions and inform strategic business decisions.

Objectives

The objective of my data science project is to develop a regression model that can accurately predict the prices for the Brazilian e-commerce market, Olist dataset based on various features. The accuracy of the model will be evaluated based on various metrics such as root mean square error (RMSE) and R-squared value. The model will be trained on a dataset that includes prices and tested on a separate dataset to assess its predictive performance.

Additionally, the project aims to provide insights into the factors that influence the price of the products. By analyzing the feature importance of the trained regression model, we can identify the most important factors that affect the amount. This information can be used by Olist Ecommerce to improve the prices and provide more targeted assistance to products. Furthermore, the project can serve as a reference for future studies on price predictions, leading to more informed decisions and better outcomes for small businesses in E-commerce.

Industry Review

Current Practices:

In the realm of e-commerce, current practices reflect a burgeoning landscape driven by technological advancements and changing consumer behaviours. The rise of online shopping has transformed the retail industry, offering convenience and accessibility to consumers while presenting new opportunities and challenges for businesses.

Olist, as a prominent player in the Brazilian e-commerce market, exemplifies the current practices prevalent in the industry. By providing a platform that connects small businesses with customers across Brazil, Olist streamlines the selling process and facilitates transactions in a dynamic marketplace. This approach aligns with broader industry trends focused on leveraging technology to enhance the online shopping experience and expand market reach.

Background Research:

Furthermore, background research in the e-commerce domain has yielded valuable insights into consumer behaviour, market dynamics, and predictive analytics. Publications and studies have delved into various aspects of e-commerce, including sales prediction, customer segmentation, recommendation systems, and logistics optimization. These research endeavours underscore the importance of data-driven decision-making and innovation in driving business success in the competitive e-commerce landscape.

Literature Survey

Publications:

Academic papers and industry reports offer insights into predictive analytics and customer segmentation methodologies. Papers explore techniques like ARIMA, Random Forest, and Gradient Boosting Machines for forecasting sales trends. Industry reports highlight practical applications in e-commerce, including case studies from platforms like Olist.

Applications:

Olist utilizes predictive analytics to forecast sales trends and customer segmentation to personalize marketing strategies. Machine learning algorithms and time series forecasting techniques optimize inventory management and supply chain efficiency. Customer segmentation tailors marketing campaigns and improves customer engagement.

Past and Ongoing Research:

Past research laid the foundation for current practices in sales prediction and customer segmentation. Ongoing research explores emerging trends like deep learning and ensemble methods, addressing challenges such as data sparsity and feature selection. Insights from academia and industry inform Olist's innovative approaches in the Brazilian e-commerce market.

1.1 Dataset and Domain

Dataset: The dataset consists of seven CSV files containing information about products, orders, sellers, customers, payments, reviews, and geolocation data. It includes 100,000 orders made from 2016 to 2018 across various Brazilian marketplaces through Olist Store.

Domain: This dataset falls within the domain of e-commerce and retail, offering insights into product attributes, order fulfilment, customer behaviour, payment methods, and seller performance.

Source: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce?rvi=1

1.2 Data Dictionary

order_status: Status of the order (categorical)

order_purchase_timestamp: Timestamp when the order was purchased

order_approved_at: Timestamp when the order was approved

order_delivered_carrier_date: Timestamp when the order was handed over to the carrier

order_delivered_customer_date: Timestamp when the order was delivered to the customer

order_estimated_delivery_date: Estimated delivery date of the order

shipping_limit_date: Deadline for shipping the order

price: Price of the product

freight_value: Cost of freight for the delivery

payment_sequential: Sequential number of the payment for the order

payment_type: Type of payment used

payment_installments: Number of installments for the payment

payment_value: Value of the payment

review_score: Score given by the customer for the product

review_comment_title: Title of the review comment

review_comment_message: Message of the review comment

review_creation_date: Date when the review was created

review_answer_timestamp: Timestamp when the review was answered

product_category_name: Category of the product

product_name_length: Length of the product name

product_description_length: Length of the product description

product_photos_qty: Quantity of photos for the product

product_weight_g: Weight of the product in grams

product_length_cm: Length of the product in centimetres

product_height_cm: Height of the product in centimetres

product_width_cm: Width of the product in centimetres

customer_zip_code_prefix: Zip code prefix of the customer

customer_city: City of the customer

customer_state: State of the customer

seller_zip_code_prefix: Zip code prefix of the seller

seller_city: City of the seller

seller_state: State of the seller

Attributes / Features / Column names are given below:

Features	Type
Order id	object
customer_id	object
order_status	object
order_purchase_timestamp	object
order_approved_at	object
order_delivered_carrier_date	object
order_delivered_customer_date	object
order_estimated_delivery_date	object
order_item_id	int64
product_id	object
seller_id	object

shipping_limit_date	object
price	float64
freight_value	float64
payment_sequential	int64
payment_type	object
payment_installments	int64
payment_value	float64
review_id	object
review_score	int64
review_comment_title	object
review_comment_message	object
review_creation_date	object

review_answer_timestamp	object
product_category_name	object
product_name_lenght	float64
product_description_lenght	float64
product_photos_qty	float64
product_weight_g	float64
product_length_cm	float64
product_height_cm	float64
product_width_cm	float64
customer_unique_id	object
customer_zip_code_prefix	int64
customer_city	object
customer_state	object

seller_zip_code_prefix	int64
seller_city	object
seller_state	object

1.3 Variable categorization (count of numeric and categorical

1. Number of rows in the Dataset is 117329

2. Number of columns in the Dataset is 39

3. Number of Categorical variables is 23

'order_id', 'customer_id', 'order_status', 'order_purchase_timestamp', 'order_approved_at', 'order_delivered_carrier_date', 'order_delivered_customer_date', 'order_estimated_delivery_date', 'product_id', 'seller_id', 'shipping_limit_date', 'payment_type', 'review_id', 'review_comment_title', 'review_comment_message', 'review_creation_date', 'review_answer_timestamp', 'product_category_name', 'customer_unique_id', 'customer_city', 'customer_state', 'seller_city', 'seller_state'

4. Number of Numeric variables is 16

'order_item_id', 'price', 'freight_value', 'payment_sequential', 'payment_installments', 'payment_value', 'review_score', 'product_name_lenght', 'product_description_lenght', 'product_photos_qty', 'product_weight_g', 'product_length_cm', 'product_height_cm', 'product_width_cm', 'customer_zip_code_prefix', 'seller_zip_code_prefix'

1.4 Pre Processing Data Analysis

Count of missing

- Missing/Null values in the dataset are a total of 181668

Null values

- Missing/Null values in a total of 38 columns. The max percentage is 88.15%

Redundant columns

- In our dataset, all the columns indicate id those Redundant columns in our project that is the columns review_id, order_id, customer_id, customer_unique_id, order_item_id, product_id, seller_id. (**Before and After**)

```
df.isnull().sum()
```

```
order_id                           3820
customer_id                        3987
order_status                       6967
order_purchase_timestamp           6967
order_approved_at                  6967
order_delivered_carrier_date       6967
order_delivered_customer_date      6967
order_estimated_delivery_date      6967
order_item_id                      6967
product_id                         6252
seller_id                          6842
shipping_limit_date                6967
price                              6967
freight_value                      6967
payment_sequential                 3823
payment_type                       3823
payment_installments               3823
payment_value                      3823
review_id                        109997
review_score                     109997
review_comment_title             109997
review_comment_message           109997
review_creation_date             109997
review_answer_timestamp          109997
product_category_name              7880
product_name_lenght                7880
product_description_lenght         7880
product_photos_qty                 7880
product_weight_g                   7880
product_length_cm                  7880
product_height_cm                  7880
product_width_cm                   7880
customer_unique_id                 3987
customer_zip_code_prefix           3987
customer_city                      3987
customer_state                     3987
seller_zip_code_prefix             6842
seller_city                        6842
seller_state                       6842
dtype: int64
```

```
df.isnull().sum()
```

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

Treat the missing values using median and mode for we use median for numerical columns and mode for categorical variables.

Project Justification

Project Statement:

The project aims to develop a machine learning model to predict product prices in an e-commerce setting using a dataset containing various features such as order details, customer information, and product attributes.

Olist, a prominent Brazilian e-commerce platform, aims to optimize its pricing strategy to maximize revenue and customer satisfaction. To achieve this objective, Olist seeks to develop a robust predictive model that accurately forecasts the prices of products listed on its platform.

By leveraging this dataset, the goal is to build a robust predictive model that can accurately estimate the price of products based on their characteristics and other relevant factors.

Complexity Involved:

Data Complexity: The dataset may contain diverse features with varying data types, including categorical, numerical, and text data. Handling and preprocessing these features require careful consideration of encoding techniques, scaling, and dealing with missing values.

Feature Engineering: Extracting meaningful features from raw data and creating new variables that capture important patterns and relationships is crucial for building an effective predictive model.

Model Selection: Choosing the most suitable regression algorithm for predicting prices, such as linear regression, decision trees, or ensemble methods, involves experimentation and evaluation of performance metrics.

Hyperparameter Tuning: Optimizing the hyperparameters of the selected regression model(s) to achieve the best performance requires iterative testing and tuning, adding complexity to the modeling process.

Evaluation Metrics: Assessing the model's performance using appropriate evaluation metrics such as mean absolute error (MAE), root mean square error (RMSE), or R-squared to ensure accurate price predictions.

Project Outcome:

Commercial Value: The developed predictive model can be deployed in e-commerce platforms to assist in dynamic pricing strategies, inventory management, and personalized product recommendations. This can lead to increased sales, improved customer satisfaction, and enhanced competitiveness in the market.

Academic Value: The project contributes to the field of machine learning and predictive modeling by exploring regression techniques in the context of e-commerce pricing dynamics. It provides insights into feature importance, model interpretability, and best practices for building predictive models in real-world applications.

Social Value: Transparent and fair pricing practices benefit consumers by enabling informed purchasing decisions and fostering trust in online shopping platforms. By ensuring that prices are reflective of product attributes and market conditions, the developed model promotes fairness and integrity in e-commerce transactions.

Statistical Summary

We will perform a statistical summary on numerical data only:

	count	mean	std	min	25%	50%	75%	max
price	117329.0	120.524349	182.944843	0.85	39.90	74.90	134.90	6735.00
freight_value	117329.0	20.027514	15.828077	0.00	13.08	16.28	21.18	409.68
payment_sequential	117329.0	1.094452	0.731174	1.00	1.00	1.00	1.00	29.00
payment_installments	117329.0	2.940151	2.775370	0.00	1.00	2.00	4.00	24.00
payment_value	117329.0	172.062565	265.388194	0.00	60.75	108.10	189.06	13664.08
review_score	117329.0	4.031467	1.387927	1.00	4.00	5.00	5.00	5.00
product_name_lenght	115634.0	48.768018	10.033831	5.00	42.00	52.00	57.00	76.00
product_description_lenght	115634.0	785.802861	652.382965	4.00	346.00	600.00	983.00	3992.00
product_photos_qty	115634.0	2.205528	1.717783	1.00	1.00	1.00	3.00	20.00
product_weight_g	117309.0	2110.763062	3785.128931	0.00	300.00	700.00	1800.00	40425.00
product_length_cm	117309.0	30.254456	16.177519	7.00	18.00	25.00	38.00	105.00
product_height_cm	117309.0	16.612476	13.452625	2.00	8.00	13.00	20.00	105.00
product_width_cm	117309.0	23.071452	11.745875	6.00	15.00	20.00	30.00	118.00
customer_zip_code_prefix	117329.0	35060.118112	29849.496175	1003.00	11250.00	24240.00	58770.00	99990.00
seller_zip_code_prefix	117329.0	24450.781955	27582.364358	1001.00	6429.00	13660.00	28035.00	99730.00

Interpretation:

The minimum price is 0.85 and the maximum price is 6735 but the low price may be a reason for offers, coupons many more reasons.

Most of the users pay bills of order by credit card and most of the users go ahead with installments.

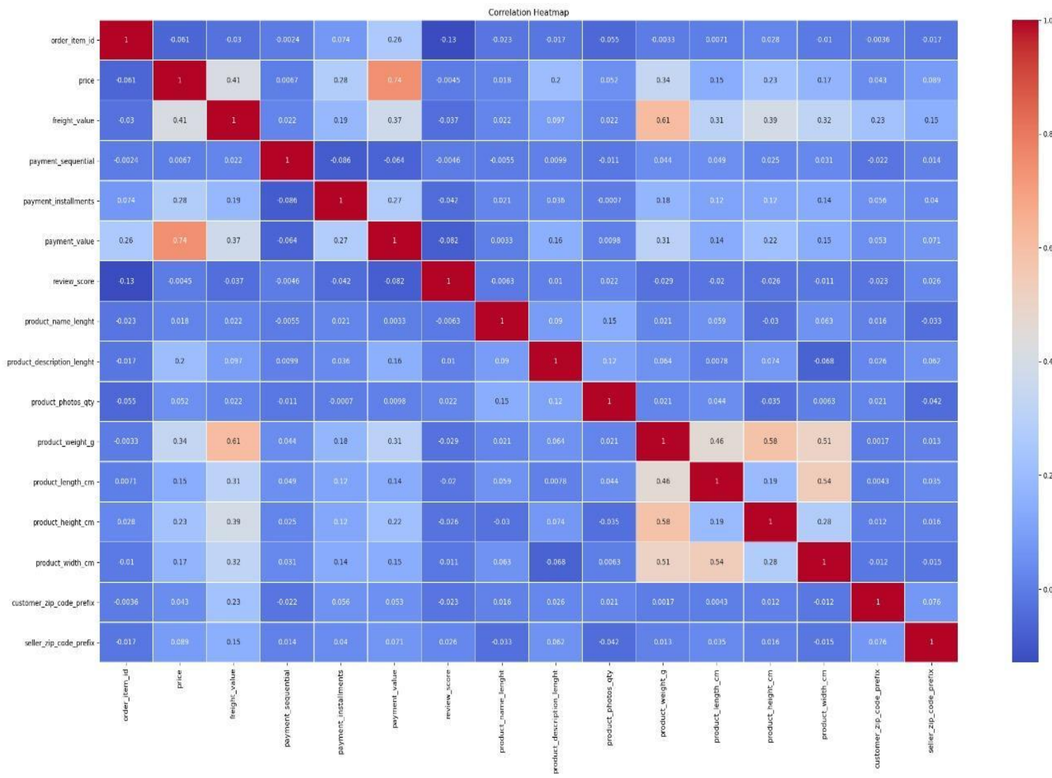
Most of the users are satisfied with using products because more than 50% of users are satisfied with the product.

Most of the Products have high-quality photos, product titles, and clear descriptions of the product.

Only a small number of products which are product weight, height, and width is more.

Data Exploration (EDA)

Relationship between variables:



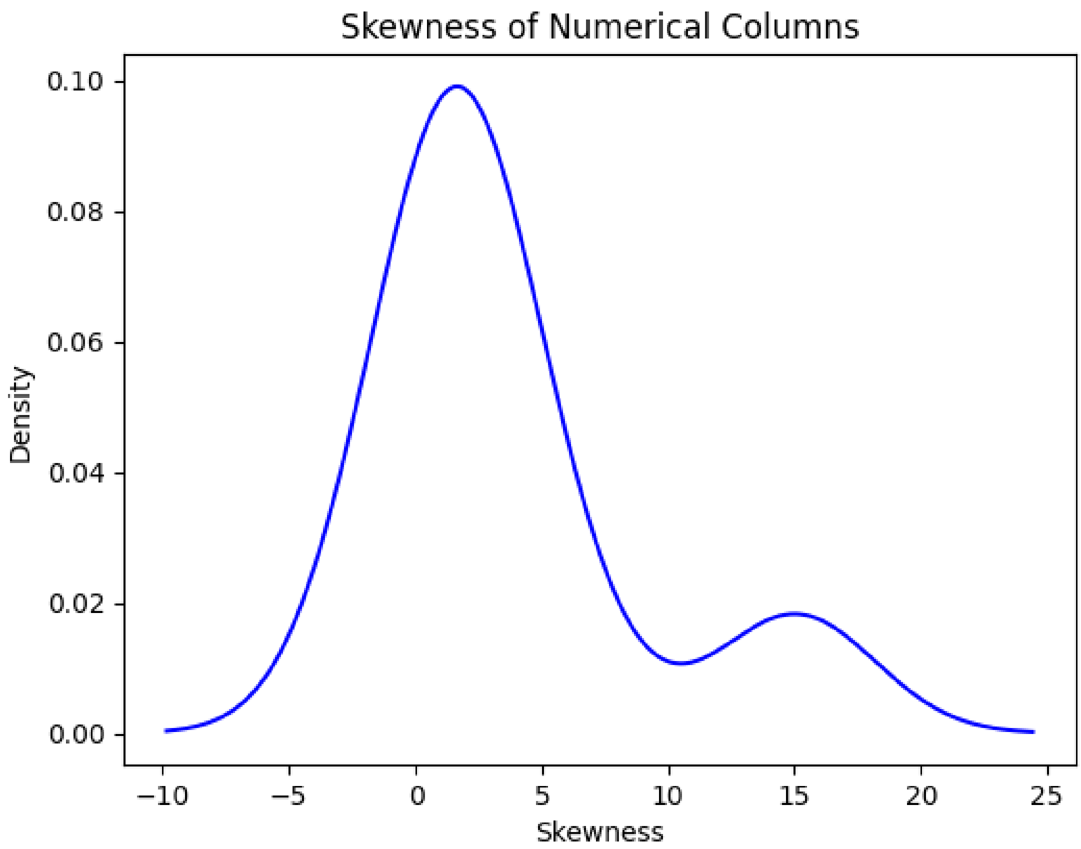
.

Multi-collinearity:

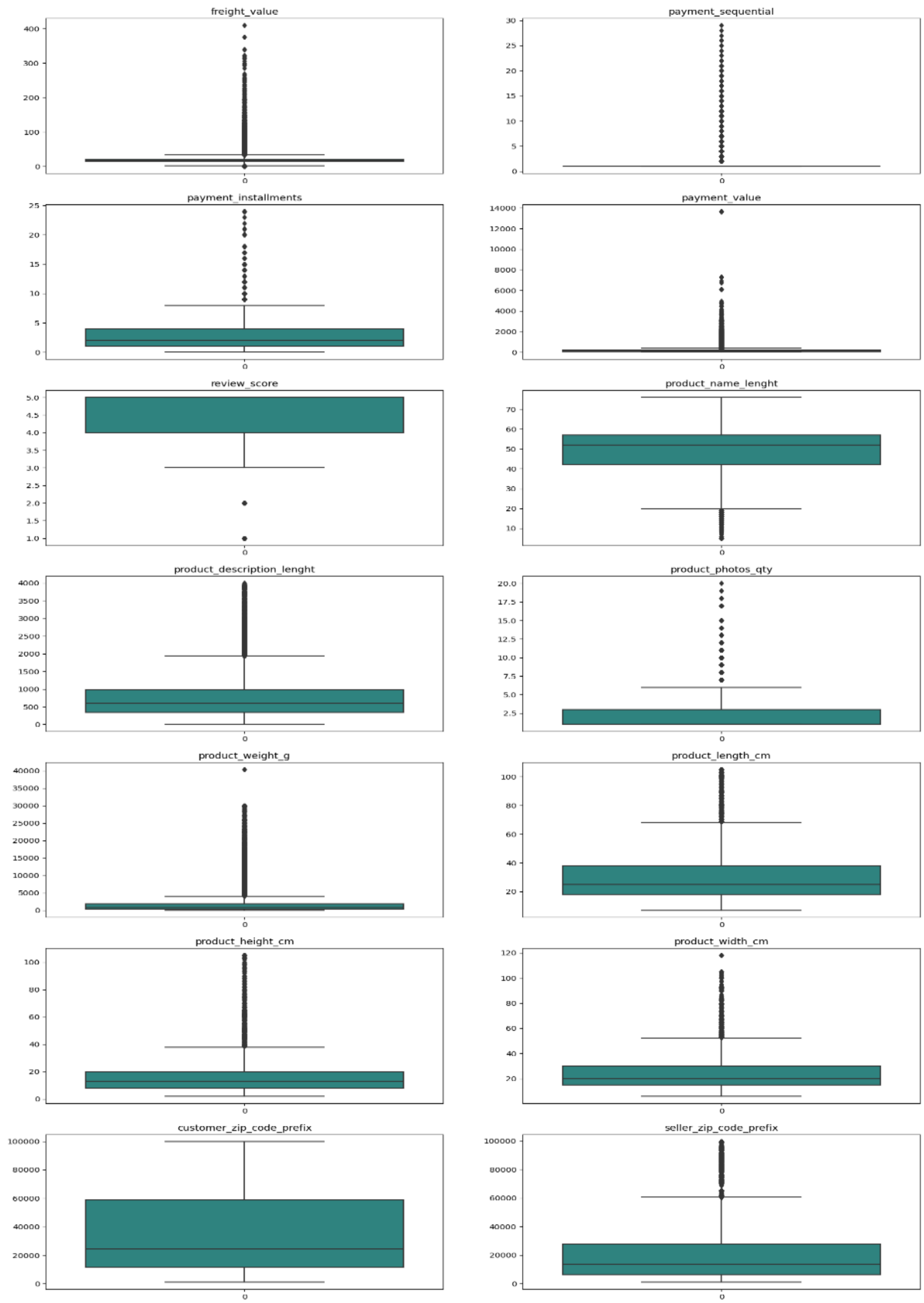
- For correlation we will consider more than 0.50 value for highly correlated features.
- We can see in the Correlation plot of numeric columns some features highly correlated for my model like
- "Product_weight_g" and "freight_value" shows highly positive correlation
- "Product_weight_g" and "Product_height_cm" shows highly positively correlated

- "product_width_cm" and "product_length_cm" shows highly positively correlated
- "product_width_cm" and "product_weight_g" shows highly positively correlated

Distribution of variables: Right Skewed for Numeric



Presence of outliers and their treatment:



Price: Potential outliers exist in the higher price range due to the maximum price of 6735.00, significantly higher than the 75th percentile (134.90) and the mean (120.52).

Freight Value: Possible outliers are observed with the maximum freight value of 409.68, considerably higher than the 75th percentile (21.18) and the mean (20.03).

Payment Sequential: An outlier is indicated by the maximum payment sequential value of 29.00, unusual compared to the rest of the data.

Payment Installments: There are potential outliers as seen in the maximum payment installments value of 24.00, substantially higher than the 75th percentile (4.00) and the mean (2.94).

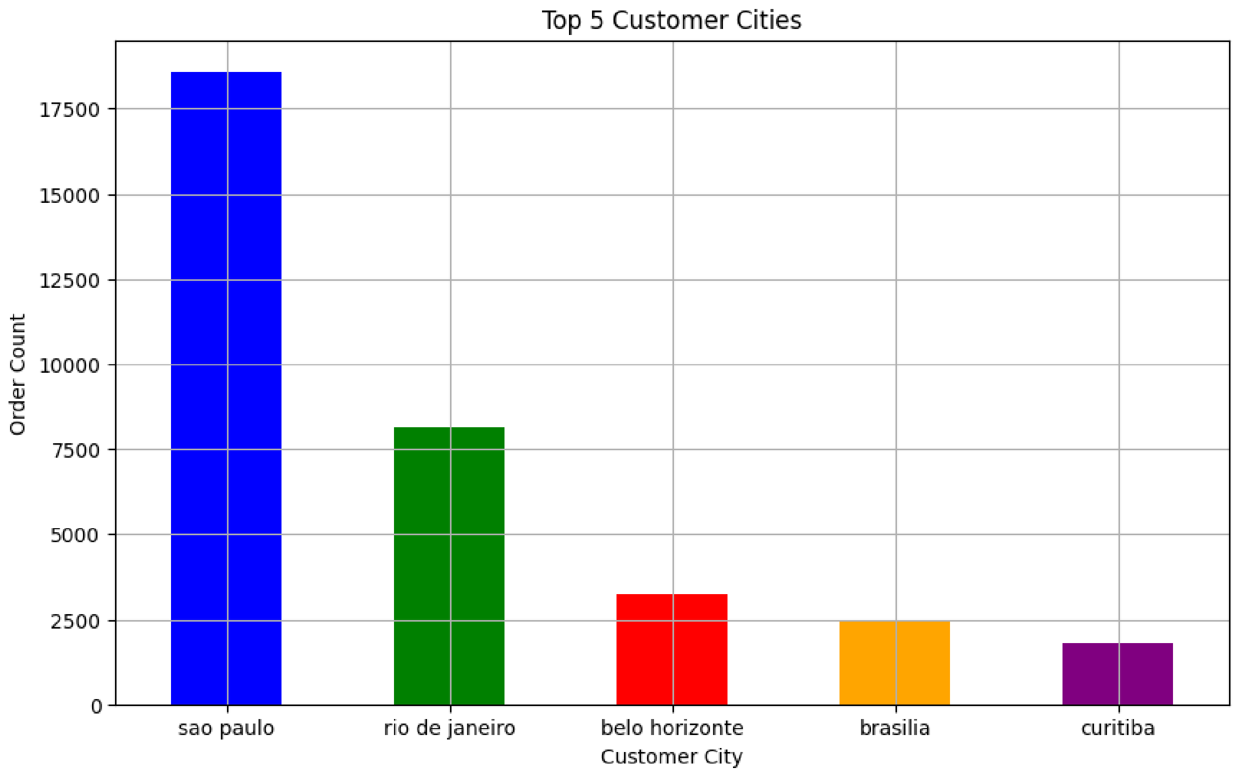
Payment Value: Potential outliers exist in the higher payment range due to the maximum payment value of 13664.08, significantly higher than the 75th percentile (189.06) and the mean (172.06).

Other Columns: Further outlier analysis can be conducted for other numerical columns such as product weight, length, height, width, and zip code prefixes.

We won’t be treating outliers as they are valuable data points for us , since they are orders on Black Day in Brazil they cannot be neglected.

Visualization Analysis:

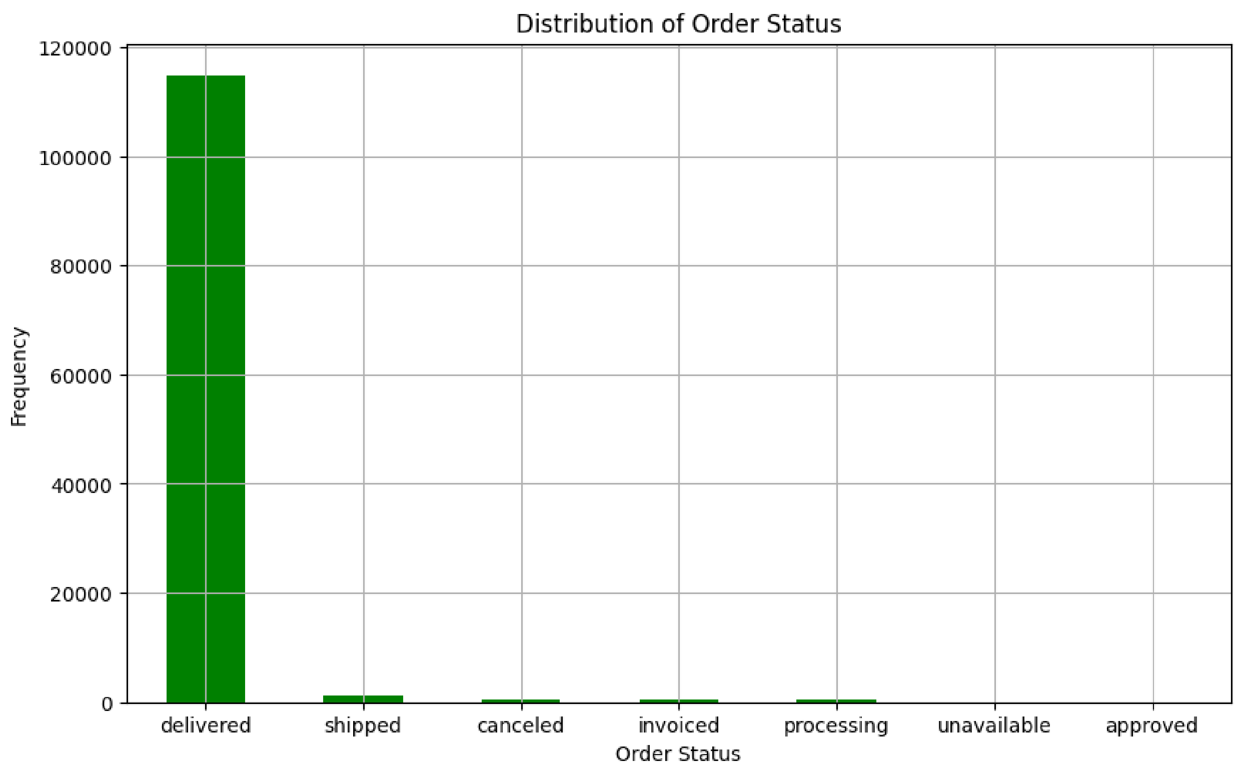
- What are the top Customer Cities?



Interpretation:

The graph shows that São Paulo has the most customers, with an order count of around 17,500. Rio de Janeiro is in second place, with an order count of around 15,000. Belo Horizonte, Brasilia, and Curitiba follow, in that order, with significantly fewer customers.

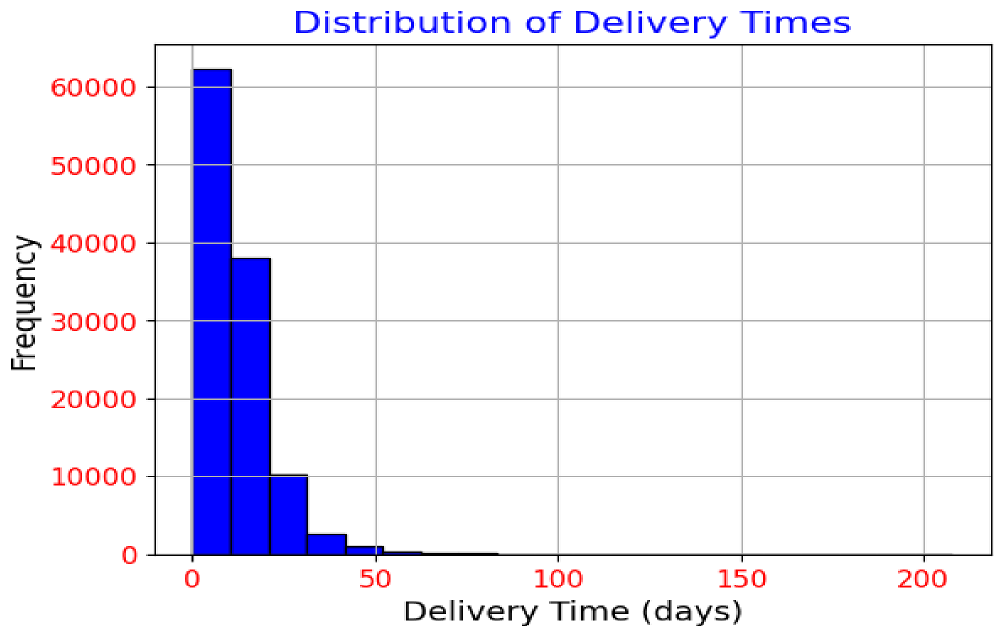
- What is the distribution of order statuses?



Interpretation:

The most frequent order status is "delivered", with a frequency of around 100,000. This is followed by "shipped", with a frequency of around 60,000. The other order statuses, "canceled", "invoiced", "processing", and "unavailable", all have frequencies of less than 20,000.

● What is the distribution of delivery times?

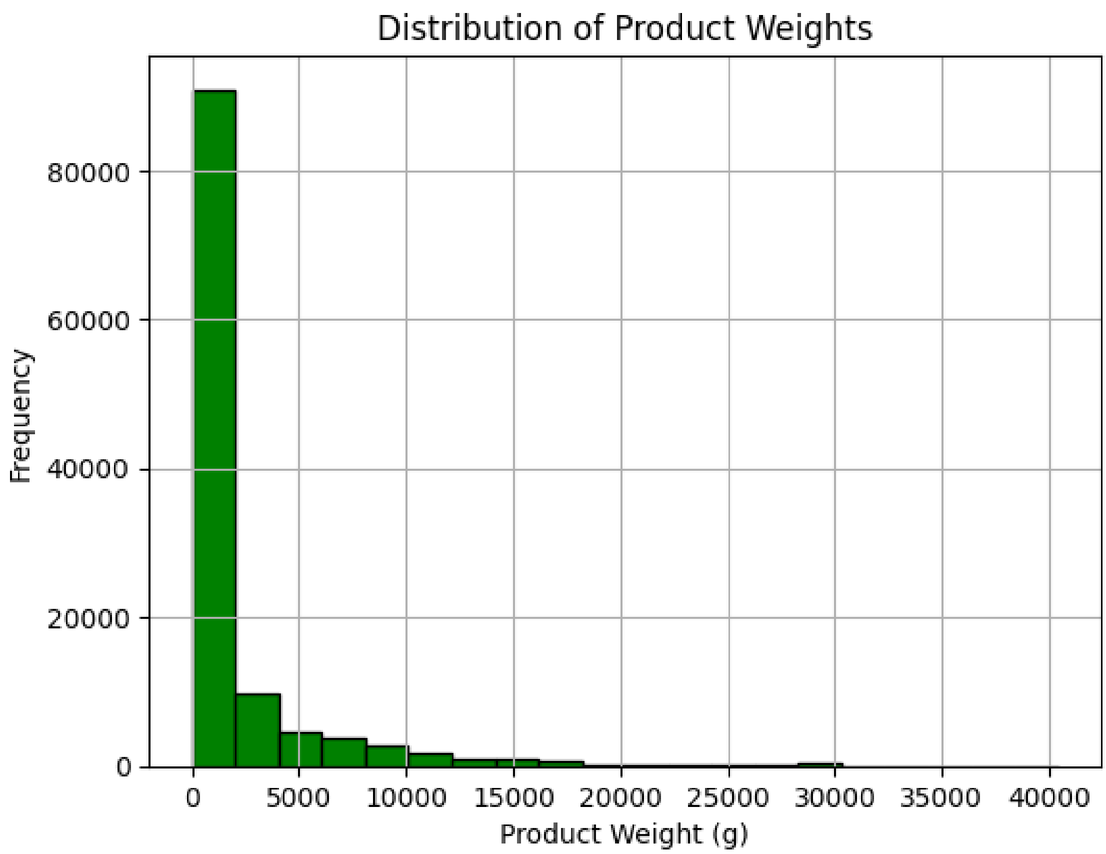


Interpretation:

The x-axis shows the delivery time in days, and the y-axis shows the frequency.

The histogram shows that most deliveries occur between 0 and 50 days. Some deliveries take less than 80 days and deliveries that take more than 50 days, but these are less frequent.

- What is the distribution of product weights (in grams) in the dataset?



Interpretation:

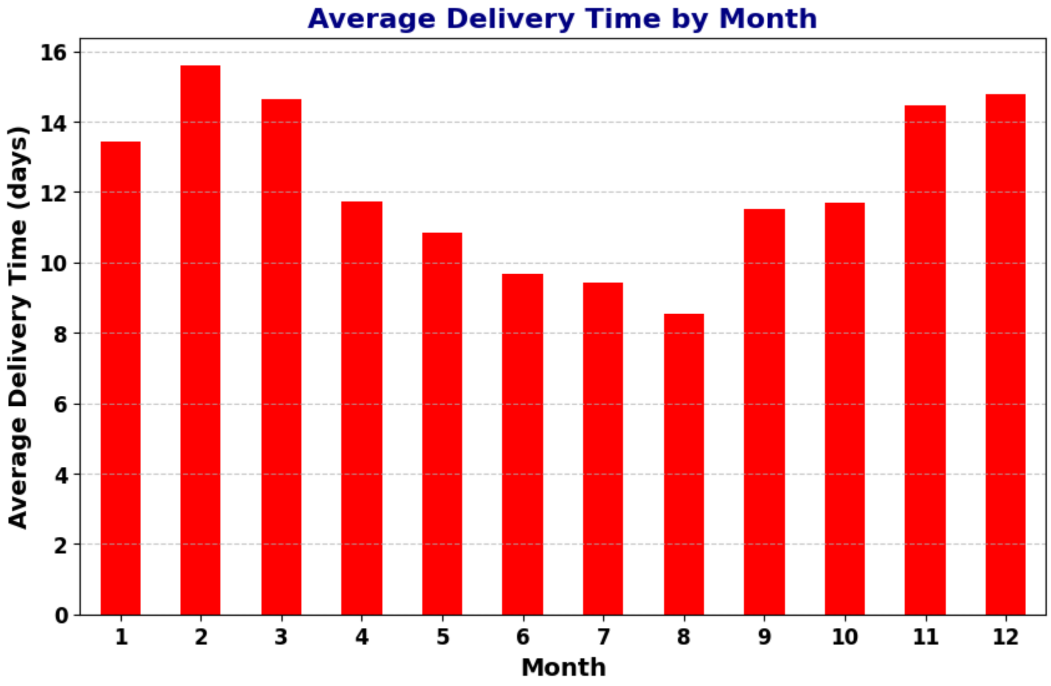
There's a high frequency of products weighing less than 5000 grams.

As the weight increases, the frequency decreases sharply.

No products are shown in the weight range above approximately 15000 grams.

In summary, lighter products are more common, and heavier products are less frequent based on this distribution.

- How does the average delivery time vary over different months?



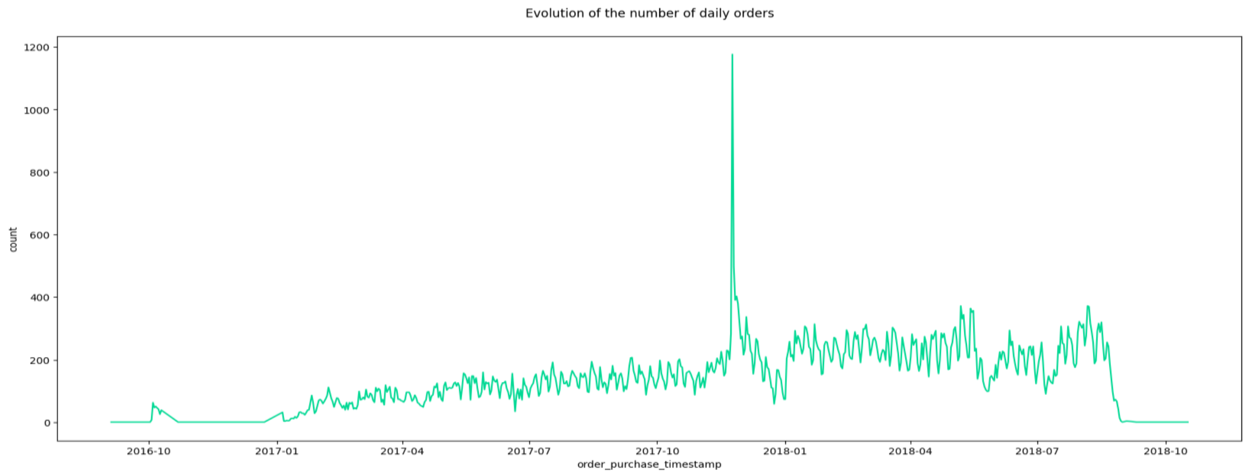
Interpretation:

In months 2 which is February the bar is at highest, indicating longer average delivery times of nearly 16 and around 14 days respectively.

In months 8 and 6, we observe shorter average delivery times as indicated by lower bars.

In summary, this graph shows how the average delivery time varies throughout the year.

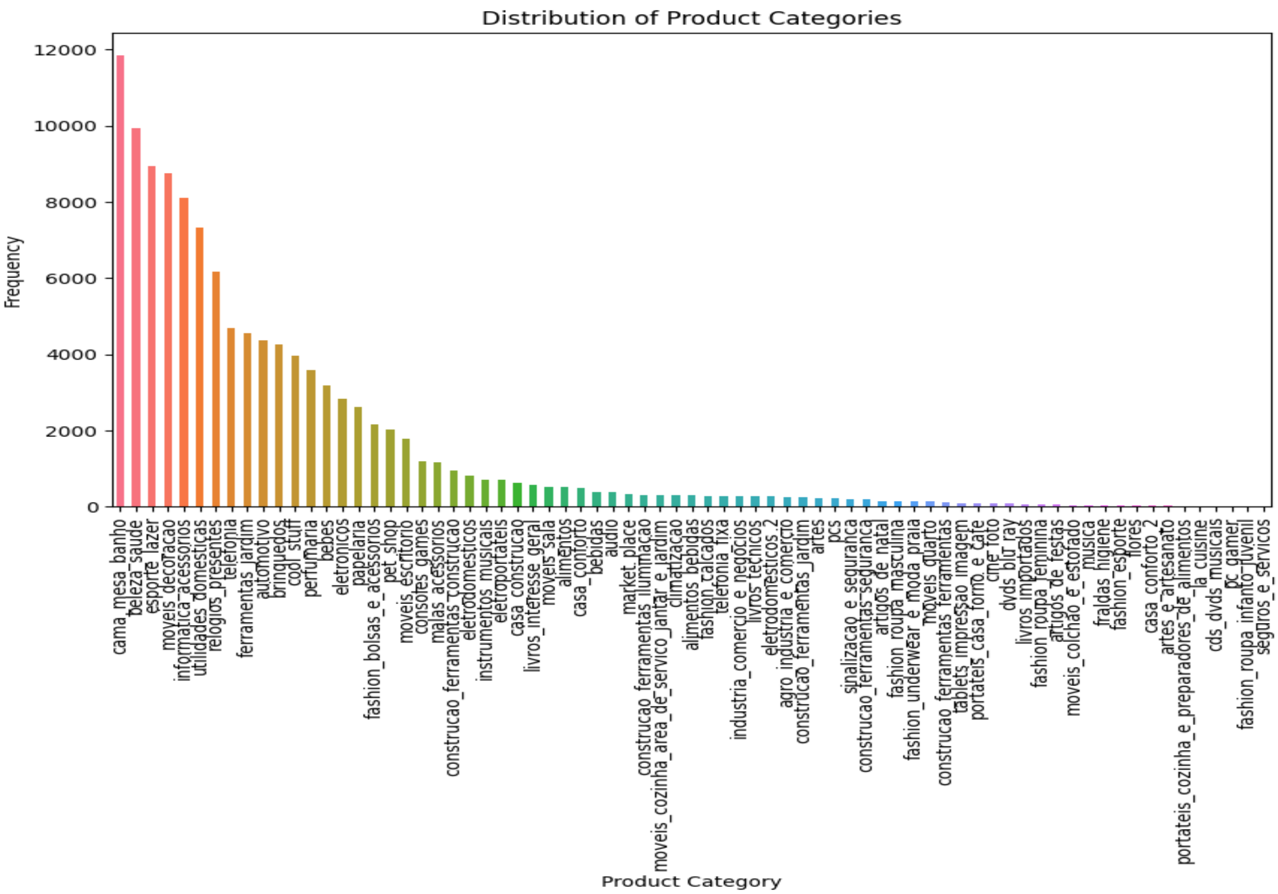
- What could you say about the order purchased?



Interpretation:

We noticed an abnormal peak around November/December 2017 (which may be a peak corresponding to end-of-year gifts) as well as a plateau without orders between October 2016 and January 2017.

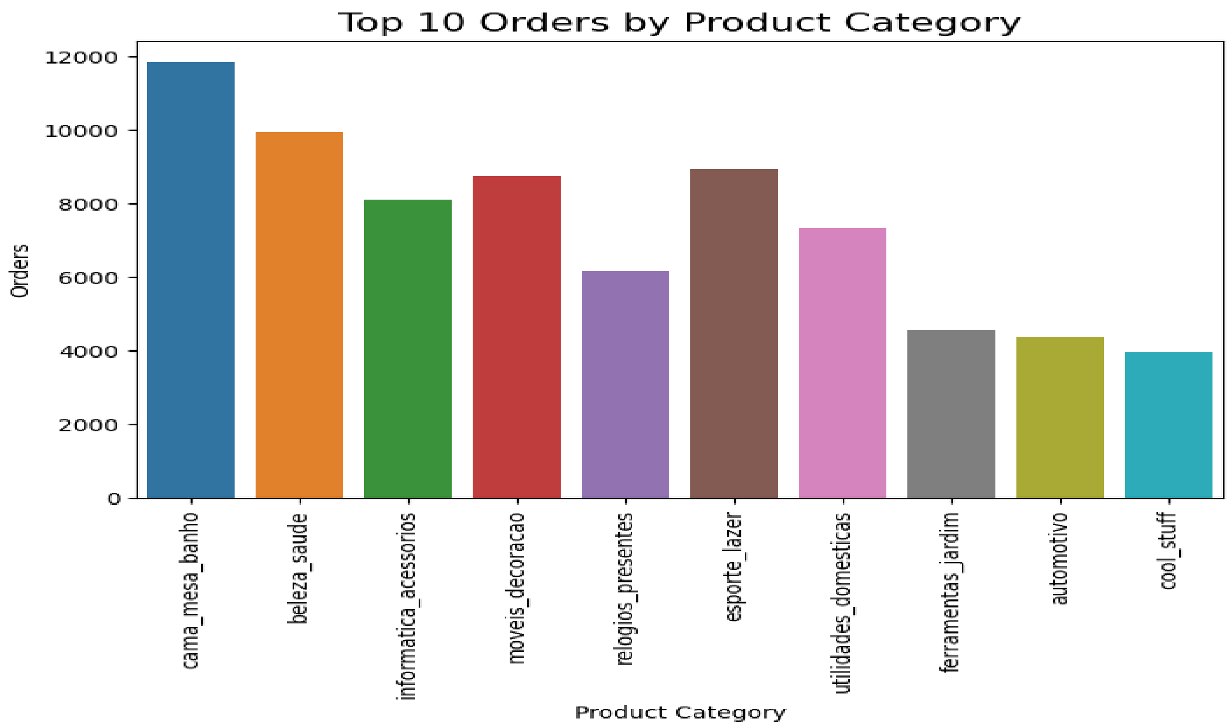
- What is the distribution of product categories?



Interpretation:

- The most frequent product category is **“cama mesa banho”**, which translates to **“bed, table, and bath”**, with a frequency of around 12,000.
- Other product categories with a high frequency include **“beleza e saúde”** (beauty and health), **“esporte lazer”** (sports and leisure), **“moveis e decoração”** (furniture and decor), and **“informatica e acessórios”** (electronics and accessories).
- Some product categories, such as **“industria_comercio e Negocios”** (industry, commerce, and business) and **“livros importados”** (imported books), have a much lower frequency than others.

- What are the top orders by product Category?



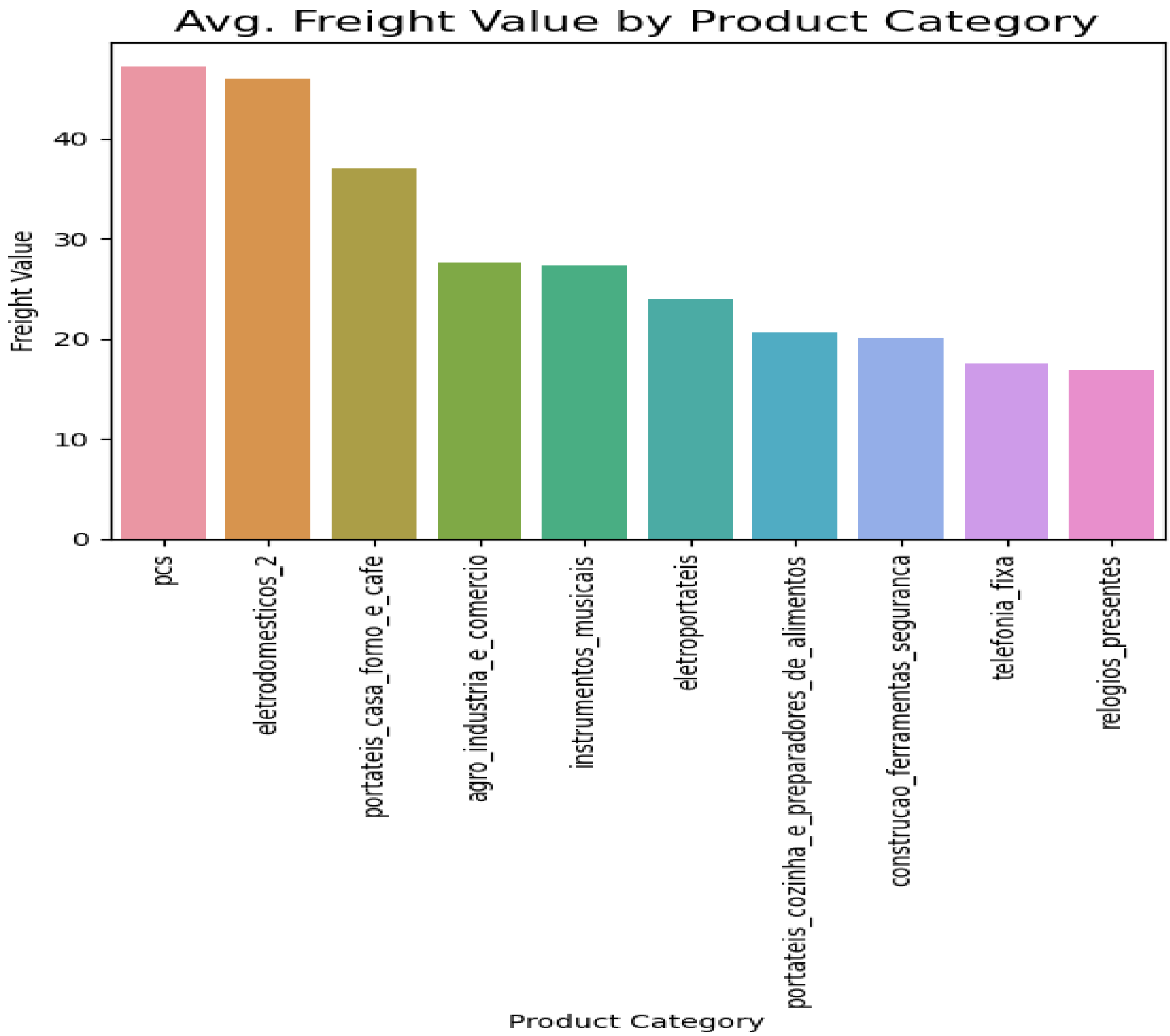
Interpretation:

The graph shows that the most popular product category is cama_mesa_banho, which translates to "bed, table, and bath", with above 11,000 orders.

The second most popular category is beleza_saude, which translates to "beauty and health", with around 10,000 orders.

The third most popular category is esporte_lazer, which translates to "sports and leisure" with over 9,000 orders.

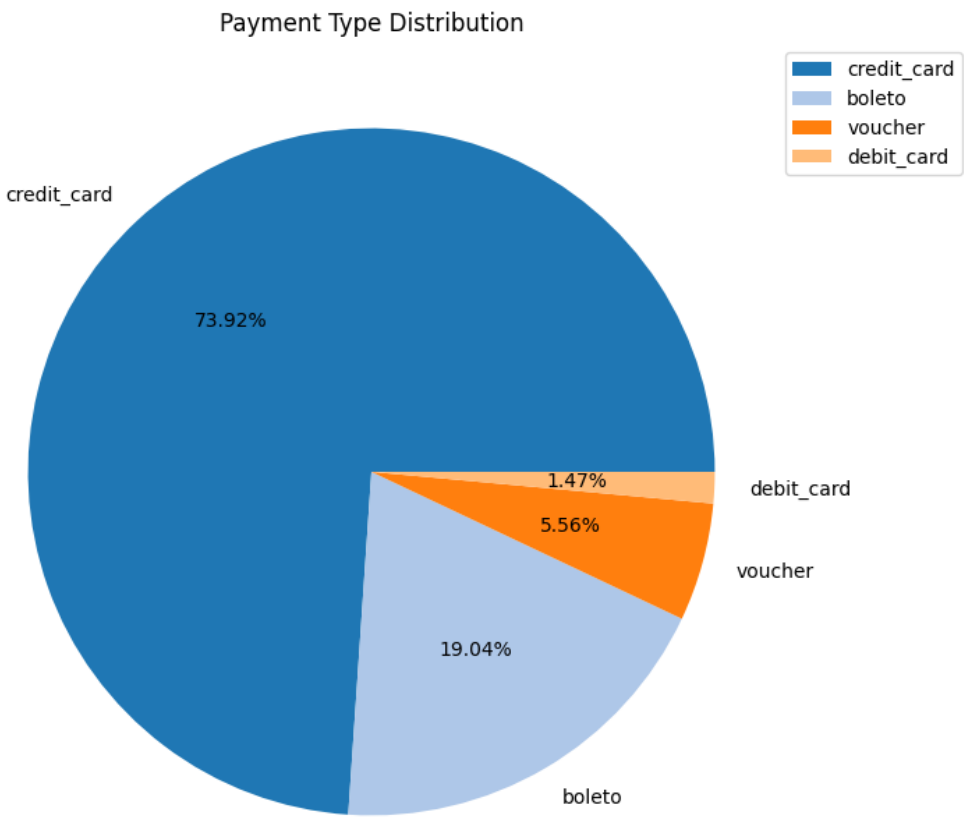
- What is the average Freight value by product category?



Interpretation:

Theproduct category with the highest average freight value is "portateis_casa_forno_e_cafe", which has an average freight value of around 48 pcs.
The product category with the lowest average freight value is "telefonia fix", which has an average freight value of around 17 pcs.
The other product categories have average freight values that fall somewhere in between these two extremes.

- Which is the most used payment type?



Interpretation:

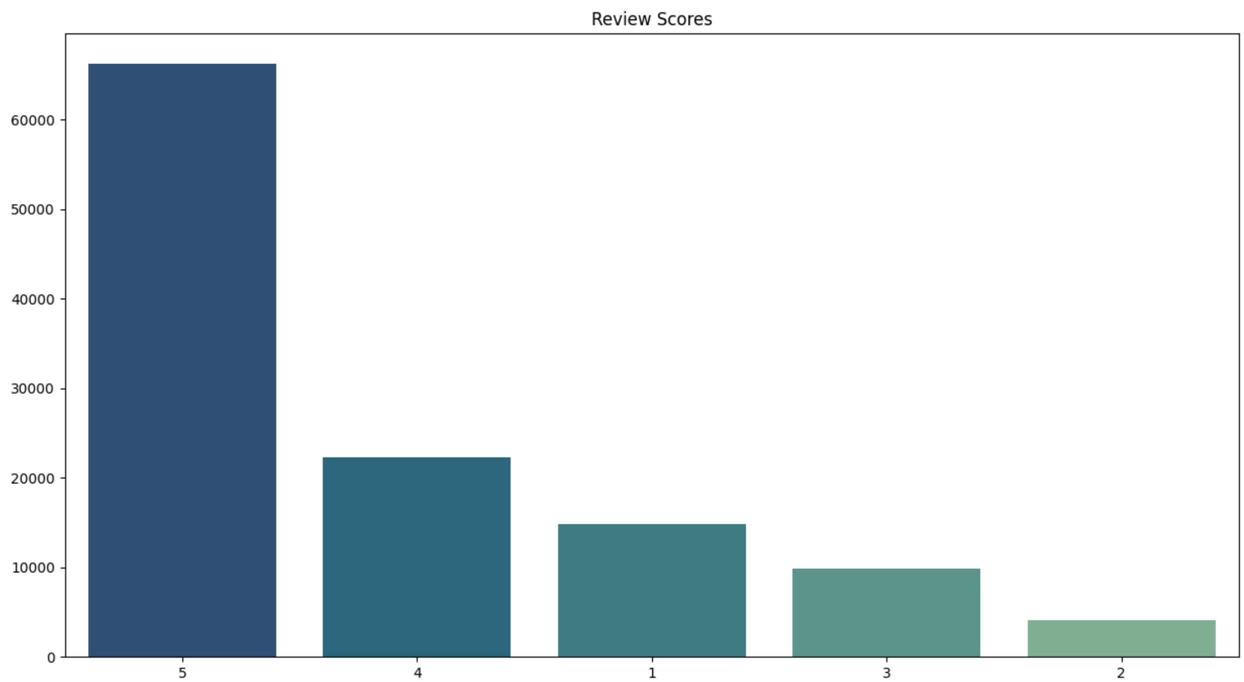
1. Credit Cards: Credit cards are the most popular payment method, accounting for nearly three-quarters (74%) of all payments. They offer convenience, and security, and often come with rewards or cashback programs.

2. Debit Cards: Debit cards come in as the second most popular option. They are linked directly to a user's bank account and allow for easy electronic transactions. Debit cards are widely accepted and provide a convenient way to pay without carrying cash.

3. Vouchers: Vouchers follow closely behind. These can include gift cards, store credits, or promotional vouchers. They allow users to redeem a specific value for goods or services.

4. Boletos: Boletos are a unique type of payment slip commonly used in Brazil. They serve as a popular payment method there. When making a purchase, customers receive a bolero with a barcode, which they can pay at banks, ATMs, or online platforms. It's a widely accepted method for various transactions in Brazil.

- Which review score is more frequent?



Interpretation:

We see review score 5 is the leading followed by 4 then 1 and last 2.

Statistical significance of variables

All variables have a significant association with the Target variable

(We will encounter this in the coming OLS Model summary.)

Class imbalance and its treatment:

The proportion of the most common price values is significantly higher than that of the rarest price values. This indicates an imbalance in the distribution.

Price

59.90	2.211729
69.90	1.794100
49.90	1.736996
89.90	1.379881
99.90	1.293798
	...
346.99	0.000852
290.99	0.000852
181.82	0.000852
465.38	0.000852
789.89	0.000852

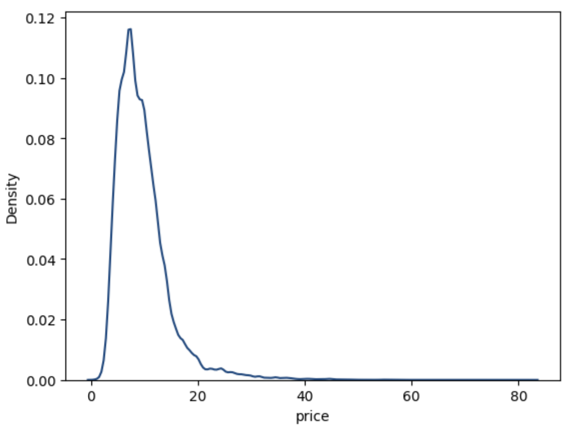
Name: proportion, Length: 5948, type: float64

Feature Engineering

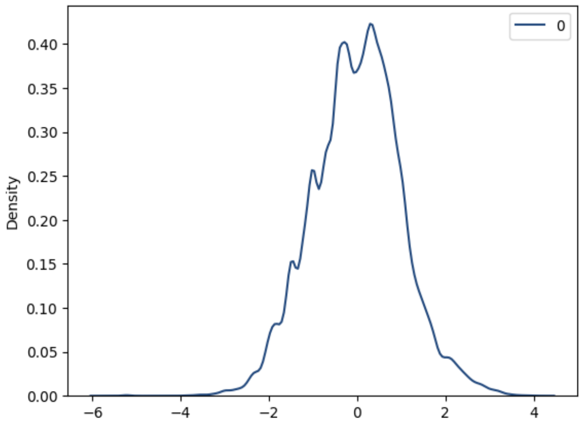
Whether any transformations required?

Yes, Transformation is required because data is not normal so to become normal we used transformation.

Before:



After:



We are getting similar to the normal distribution yeo-Johnson transformation technique so we will go ahead with that transformation technique.

Scaling the data

Yes, some models work only with scale data and it return the best accuracy like KNN which is a distance-based algorithm so here we are doing scaling on our project.

There are so many techniques to scale the data but we will go ahead with StandardScaler and other techniques are MinMaxscaler, and RoubstScaler.

Before Scaling:

	0	1	2	3	4
freight_value	8.72	8.72	8.72	7.78	7.78
payment_sequential	1.00	3.00	2.00	1.00	1.00
payment_installments	1.00	1.00	1.00	3.00	1.00
payment_value	18.12	2.00	18.59	37.77	37.77
review_score	4.00	4.00	4.00	4.00	5.00
product_name_lenght	40.00	40.00	40.00	40.00	40.00
product_description_lenght	268.00	268.00	268.00	268.00	268.00
product_photos_qty	4.00	4.00	4.00	4.00	4.00
product_weight_g	500.00	500.00	500.00	500.00	500.00
product_length_cm	19.00	19.00	19.00	19.00	19.00
product_height_cm	8.00	8.00	8.00	8.00	8.00
product_width_cm	13.00	13.00	13.00	13.00	13.00
customer_zip_code_prefix	3149.00	3149.00	3149.00	3366.00	2290.00
seller_zip_code_prefix	9350.00	9350.00	9350.00	9350.00	9350.00

After Scaling:

	0	1	2	3	4
freight_value	-0.714399	-0.714399	-0.714399	-0.773787	-0.773787
payment_sequential	-0.129180	2.606161	1.238491	-0.129180	-0.129180
payment_installments	-0.699063	-0.699063	-0.699063	0.021564	-0.699063
payment_value	-0.580068	-0.640810	-0.578297	-0.506025	-0.506025
review_score	-0.022672	-0.022672	-0.022672	-0.022672	0.697830
product_name_lenght	-0.884255	-0.884255	-0.884255	-0.884255	-0.884255
product_description_lenght	-0.794899	-0.794899	-0.794899	-0.794899	-0.794899
product_photos_qty	1.058730	1.058730	1.058730	1.058730	1.058730
product_weight_g	-0.425520	-0.425520	-0.425520	-0.425520	-0.425520
product_length_cm	-0.695686	-0.695686	-0.695686	-0.695686	-0.695686
product_height_cm	-0.640215	-0.640215	-0.640215	-0.640215	-0.640215
product_width_cm	-0.857473	-0.857473	-0.857473	-0.857473	-0.857473
customer_zip_code_prefix	-1.069072	-1.069072	-1.069072	-1.061802	-1.097850
seller_zip_code_prefix	-0.547482	-0.547482	-0.547482	-0.547482	-0.547482

Encoding Techniques

We use Encoding to convert categorical data into numeric data because the ML model doesn't understand categorical data.

There are so many techniques in encoding but we will go ahead with cat boost encoding because this will return us the weight according to values in all columns.

Other techniques are label encoding, target encoding, frequency encoding, dummy encoding, and many more which we won't be using.

Before:

	0	1	2	3	4
order_status	delivered	delivered	delivered	delivered	delivered
payment_type	credit_card	voucher	voucher	credit_card	credit_card
review_comment_title	Recomendo	Recomendo	Recomendo	Recomendo	Recomendo
review_comment_message	Não testei o produto ainda, mas ele veio corre...	Não testei o produto ainda, mas ele veio corre...	Não testei o produto ainda, mas ele veio corre...	Deveriam embalar melhor o produto. A caixa vei...	Só achei ela pequena pra seis xícaras ,mais é ...
product_category_name	utilidades_domesticas	utilidades_domesticas	utilidades_domesticas	utilidades_domesticas	utilidades_domesticas
customer_city	sao paulo	sao paulo	sao paulo	sao paulo	sao paulo
customer_state	SP	SP	SP	SP	SP
seller_city	maua	maua	maua	maua	maua
seller_state	SP	SP	SP	SP	SP

After:

	0	1	2	3	4
order_status	119.77	119.77	119.77	119.77	119.77
payment_type	126.25	104.29	104.29	126.25	126.25
review_comment_title	119.14	119.14	119.14	119.14	119.14
review_comment_message	52.62	52.62	52.62	120.52	120.52
product_category_name	90.61	90.61	90.61	90.61	90.61
customer_city	108.14	108.14	108.14	108.14	108.14
customer_state	109.90	109.90	109.90	109.90	109.90
seller_city	59.33	59.33	59.33	59.33	59.33
seller_state	108.65	108.65	108.65	108.65	108.65

Model Building

Assumptions before the MLR model:

In Regression Problem:

Simple linear regression is a linear approach to modeling the relationship between a dependent variable and a single independent variable.
It is commonly used when trying to predict the value of a dependent variable based on the value of a single independent variable.

The assumption to be checked before building a model:

1. T**he target variable (Dependent variable) should be a numeric variable** and here 'Price' is numeric.
   dtype('float64')

2. **Absence of Multicollinearity**: To check the Absence of Multicollinearity we use the VIF(Variance influence factor) method.

	VIF_Value
const	46.266579
product_weight_g	2.605071
freight_value	1.904854
product_width_cm	1.627016
product_height_cm	1.550187
product_length_cm	1.534482
payment_value	1.270091
payment_installments	1.108986
customer_zip_code_prefix	1.091964
product_description_lenght	1.069064
seller_zip_code_prefix	1.046346
product_photos_qty	1.040563
product_name_lenght	1.035777
payment_sequential	1.018099
review_score	1.010452

VIF values are not more than 10 so we conclude that all the columns are useful for my model building.

Feature selection

We will drop ID columns, and select all other columns before building a model.

Dimensionality reduction

Given that the majority of features are categorical, dimensionality reduction techniques like PCA may not be applicable. However, assess the potential for reducing feature space to simplify model complexity if needed.

- Now we will perform one-by-one ML models for that we first require to split the data into training and testing
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= .30, random_state = 10).

  X Train shape is (82130, 23)
  X Test shape is (35199, 23)
  y Train shape is (82130, 1)
  y Test shape is (35199, 1)

Base Model Performance:

Linear Regression (OLS) - full model without transformation:

Dep. Variable:	y	R-squared:	0.472
Model:	OLS	Adj. R-squared:	0.472
Method:	Least Squares	F-statistic:	3190.
Date:	Wed, 06 Mar 2024	Prob (F-statistic):	0.00
Time:	16:54:42	Log-Likelihood:	-90339.
No. Observations:	82130	AIC:	1.807e+05
Df Residuals:	82106	BIC:	1.809e+05
Df Model:	23		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.9965	0.072	-13.826	0.000	-1.138	-0.855
freight_value	0.0154	0.004	4.063	0.000	0.008	0.023
payment_sequential	0.0154	0.003	5.886	0.000	0.010	0.020
payment_installments	0.1594	0.003	53.880	0.000	0.154	0.165
payment_value	0.3284	0.003	101.376	0.000	0.322	0.335
review_score	0.0408	0.003	15.545	0.000	0.036	0.046
product_name_lenght	0.0336	0.003	13.006	0.000	0.029	0.039
product_description_lenght	0.0982	0.003	37.005	0.000	0.093	0.103
product_photos_qty	0.0276	0.003	10.626	0.000	0.022	0.033
product_weight_g	0.0834	0.004	20.067	0.000	0.075	0.092
product_length_cm	0.0577	0.003	18.309	0.000	0.052	0.064
product_height_cm	0.0829	0.003	26.210	0.000	0.077	0.089
product_width_cm	0.0982	0.003	30.436	0.000	0.092	0.105
customer_zip_code_prefix	-0.0040	0.003	-1.304	0.192	-0.010	0.002
seller_zip_code_prefix	0.0721	0.003	24.534	0.000	0.066	0.078
order_status	0.0021	0.000	4.775	0.000	0.001	0.003
payment_type	-0.0009	0.000	-3.104	0.002	-0.002	-0.000
review_comment_title	0.0003	0.000	2.842	0.004	0.000	0.001
review_comment_message	0.0017	7.22e-05	23.164	0.000	0.002	0.002
product_category_name	0.0012	4.55e-05	25.512	0.000	0.001	0.001
customer_city	0.0013	9.03e-05	14.856	0.000	0.001	0.002
customer_state	0.0011	0.000	4.450	0.000	0.001	0.002
seller_city	0.0024	4.2e-05	57.430	0.000	0.002	0.002
seller_state	-0.0009	9.62e-05	-8.947	0.000	-0.001	-0.001

Omnibus:	24319.575	Durbin-Watson:	1.992
Prob(Omnibus):	0.000	Jarque-Bera (JB):	375257.543
Skew:	-1.009	Prob(JB):	0.00
Kurtosis:	13.276	Cond. No.	1.03e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.03e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

The R-squared value obtained from the above model is 0.472 which means that the above model explains 47.2% of the variation in the Target.

Overall F-Test & p-value of the Model:

Ho: All β's are equal to zero (i.e. regression model is not significant)

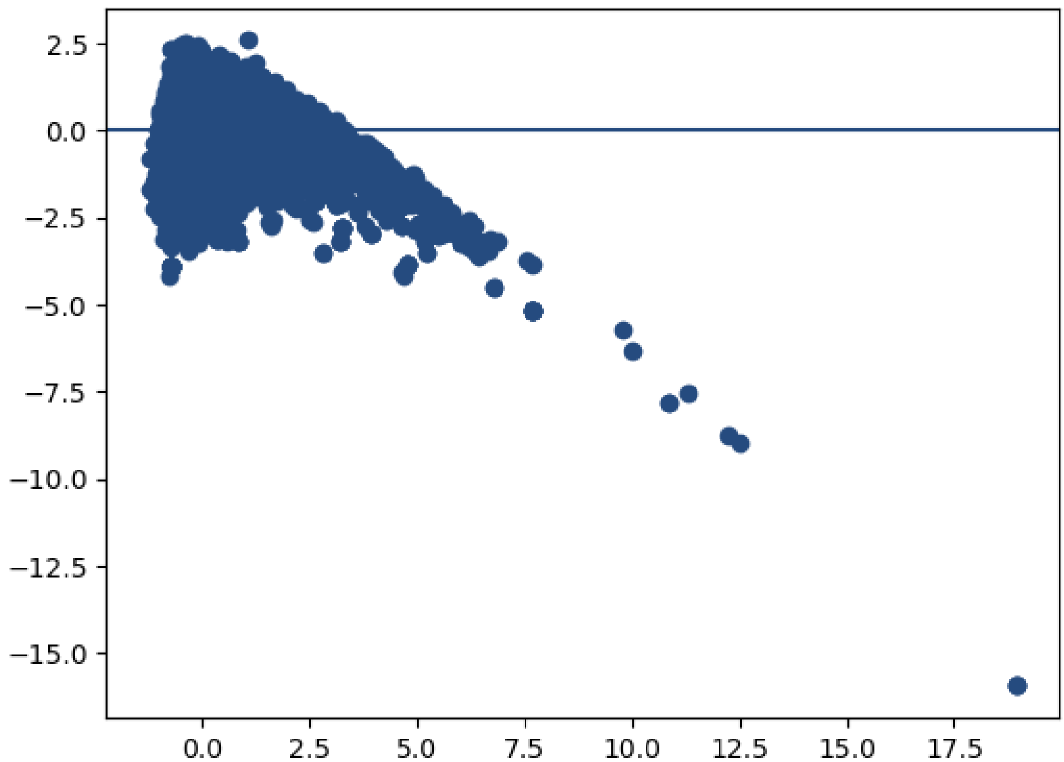
H1: At least one β is not equal to zero (i.e. regression model is significant)

- Prob (F-statistic): 0.00
- As the p-value is less than 0.05, we accept the alternate hypothesis i.e. the regression model is significant.

****

The assumption to be checked after building a model:

1. A **linear relationship** between dependent and independent variables.



2. **Assumption of autocorrelation:** Check Homoscedasticity we use the durbin_watson Test

   H0: The error terms are not auto-correlated

   H1: The error terms are auto-correlated

   durbin_watson(residuals): 1.9919756300068696

   There is autocorrelation. The assumption is violated because the p-value is greater than 0.05

3. **Homoscedasticity assumption: To check Homoscedasticity we use the** Breuschpagan Test :

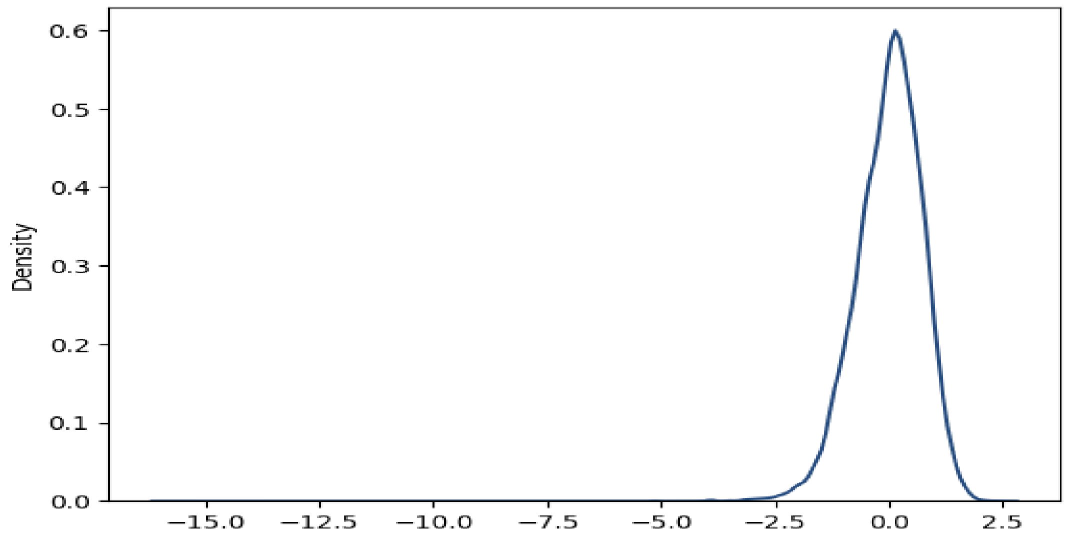
   H0: The error terms are homoskedastic.

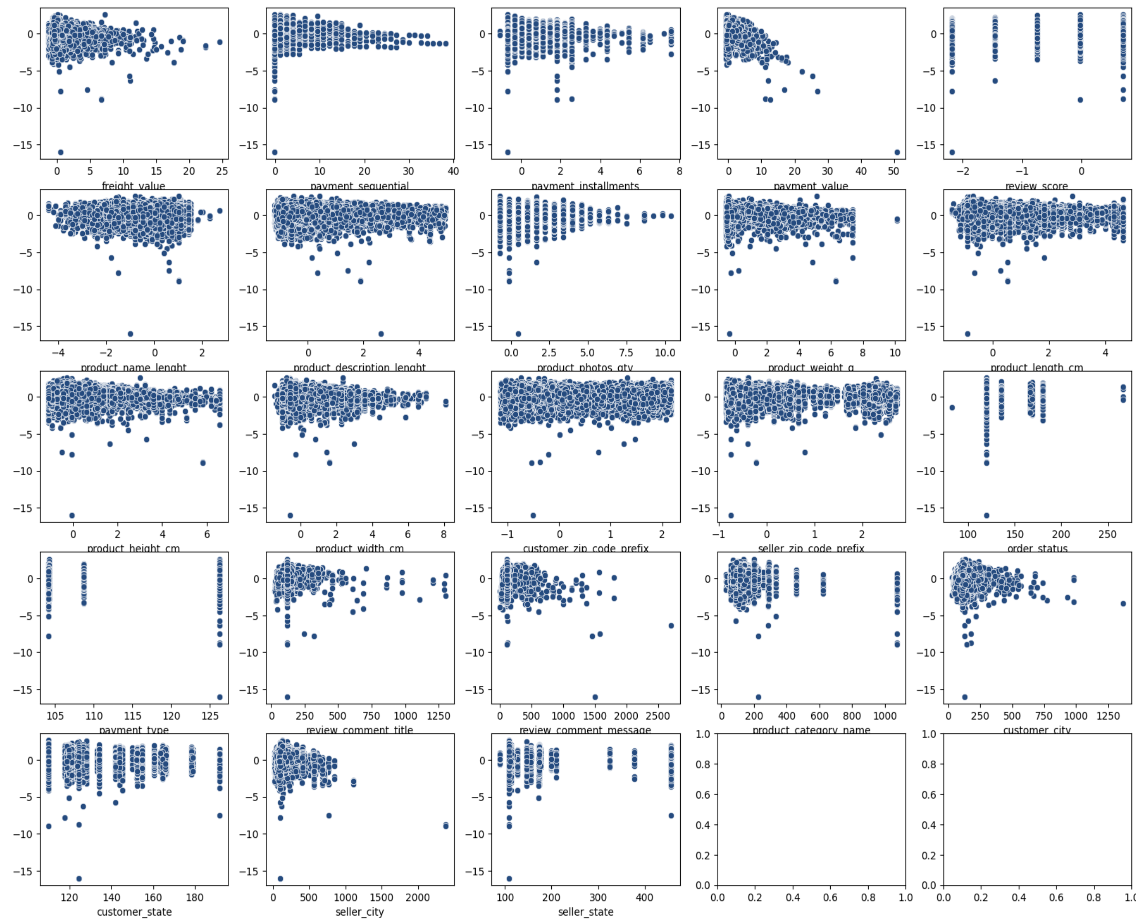
   H1: The error terms are heteroskedastic.

   P_Value=0

   Rejecting H_0: The variance of the errors is not constant across all levels of the independent variables.

4. **Residuals should follow normal distribution**. : To Check normality we use the Shapiro test, QQPLOT, and jarque_bera test





The graph shows us that Residuals follow normality.

H0: Data is normal

H1: Data is not normal

Shapiro test:

ShapiroResult (statistic = 0.9688560366630554, pvalue=0.0)

Jarque_bera test:

P value by Jarque_bera test: 0.0

The p-value of the test is less than 0.05, this implies that the residuals are not normally distributed.

Here, normality tests give contradictory results.

Linear Regression:

We can consider base models such as Linear regression, Decision Tree, and KNN but we will consider only the Linear regression model.

Base Model:

```
lr=LinearRegression()
model_lr = lr.fit(X_train,y_train)
# Linear regression model using sklearn method
model_lr
```

```
▾ LinearRegression
LinearRegression()
```

```
performance(X_train,y_train,model_lr)  # Training R2 and MAE score
```

```
R2 score is :  0.47190320022611276
MAE score is :  0.5658358101353103
```

```
performance(X_test,y_test,model_lr)  # Testing R2 and MAE score
```

```
R2 score is :  0.4394121981755589
MAE score is :  0.5689565480802902
```

We will perform feature selection techniques because these are our important features so we can use techniques such as SFS, RFS, FORWARD, and BACKWARD.

```python
from mlxtend.feature_selection import SequentialFeatureSelector as sfs
```

```python
# FORWARD

lr = LinearRegression()
lr_sfs = sfs(estimator = lr, k_features = 'best', forward = True)

sfs_forward = lr_sfs.fit(X_train,y_train)
forward_feature = list(sfs_forward.k_feature_names_)
forward_feature
```

```
['freight_value',
 'payment_sequential',
 'payment_installments',
 'payment_value',
 'review_score',
 'product_name_lenght',
 'product_description_lenght',
 'product_photos_qty',
 'product_weight_g',
 'product_length_cm',
 'product_height_cm',
 'product_width_cm',
 'seller_zip_code_prefix',
 'order_status',
 'payment_type',
 'review_comment_title',
 'review_comment_message',
 'product_category_name',
 'customer_city',
 'customer_state',
 'seller_city',
 'seller_state']
```

```
# BACKWARD

lr = LinearRegression()
lr_sfs = sfs(estimator = lr, k_features = 'best', forward = False)

sfs_back = lr_sfs.fit(X_train,y_train)
backward_feature = list(sfs_back.k_feature_names_)
backward_feature
```

```
['freight_value',
 'payment_sequential',
 'payment_installments',
 'payment_value',
 'review_score',
 'product_name_lenght',
 'product_description_lenght',
 'product_photos_qty',
 'product_weight_g',
 'product_length_cm',
 'product_height_cm',
 'product_width_cm',
 'seller_zip_code_prefix',
 'order_status',
 'payment_type',
 'review_comment_title',
 'review_comment_message',
 'product_category_name',
 'customer_city',
 'customer_state',
 'seller_city',
 'seller_state']
```

All the feature selection techniques give me the same features so we can go ahead with those features.

But we will check other models to beat the R2 score on test data we will see one by one:

Decision Tree Model:

```
dt = DecisionTreeRegressor()
dt.fit(X_train,y_train)
```

```
DecisionTreeRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,dt)  # Training R2 and MAE score
```

```
R2 score is :  0.9999998212967505
MAE score is :  2.4154145482093616e-06
```

```
performance(X_test,y_test,dt)  # Testing R2 and MAE score
```

```
R2 score is :  0.8674044231368563
MAE score is :  0.12359886707995761
```

Decision Tree gives me an on-train data R2 score is 0.99 and on test data R2 score is 0.86. MAE score on train data was 0.000215 and test data was 0.12 we will apply Hyperparameters to increase model performance.

We will perform Hyperparameters tunning using GridSearchCV to increase model performance so we will try some parameters on the Decision tree model.

```
%%time
param_grid = {
    'max_depth': [10,15,20,25,30],
    'min_samples_leaf': [2,3,4,5,6],
    'min_samples_split': [2,3,4,5,6],
    "max_features"  : [20,24,28,33],
    "max_leaf_nodes" : [2,4,6,8,10]
}

dtr = DecisionTreeRegressor(random_state = 10)

grid_search = GridSearchCV(estimator = dtr, param_grid = param_grid, cv = 3)


grid_search.fit(X_train,y_train)

print(grid_search.best_params_)
```

```
{'max_depth': 10, 'max_features': 24, 'max_leaf_nodes': 10, 'min_samples_leaf': 2, 'min_samples_split': 2}
CPU times: total: 25min 43s
Wall time: 27min 48s
```

We will build a model on hyperparameters which gives me GridSearchCV on the Decision Tree model.

```
dtr = DecisionTreeRegressor(**grid_search.best_params_, random_state=123)
dtr.fit(X_train,y_train)
```

```
DecisionTreeRegressor(max_depth=10, max_features=24, max_leaf_nodes=10,
                      min_samples_leaf=2, random_state=123)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,dtr)  # Training R2 and MAE score
```

```
R2 score is :  0.6897344658043558
MAE score is :  0.39483753549195594
```

```
performance(X_test,y_test,dtr)  # Testing R2 and MAE score
```

```
R2 score is :  0.6927892843288358
MAE score is :  0.3934903290564659
```

Decision Tree gives me an on-train data R2 score is 0.68 and on test data R2 score is 0.69. MAE score on train data was 0.39 and test data was 0.39 we will consider high R2 and low MAE (Mean Absolute Error) scores for "BEST MODEL".

KNN:

KNN is a distance-based model so scaling is required and we will see the performance of that model.

```
knn = KNeighborsRegressor()
knn.fit(X_train,y_train)
```

```
KNeighborsRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,knn)  # Training R2 and MAE score
```

```
R2 score is :  0.6448676091869565
MAE score is :  0.423559168556426
```

```
performance(X_test,y_test,knn)   # Testing R2 and MAE score
```

```
R2 score is :  0.4519204218071049
MAE score is :  0.5317184988836363
```

KNN gives me an on-train data R2 score is 0.64 and on test data R2 score is 0.45. MAE score on train data was 0.42 and test data was 0.53 we will consider high R2 and low MAE (Mean Absolute Error) scores for "BEST MODEL".

Random Forest:

```
rf = RandomForestRegressor()
rf.fit(X_train,y_train)
```

```
RandomForestRegressor()
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

```
performance(X_train,y_train,rf)  # Training R2 and MAE score
```

```
R2 score is :  0.9905598972352645
MAE score is :  0.043515766779606585
```

```
performance(X_test,y_test,rf)  # Testing R2 and MAE score
```

```
R2 score is :  0.933379477234183
MAE score is :  0.11675772574398584
```

Random Forest gives me an on-train data R2 score is 0.99 and a test data R2 score is 0.93. MAE score on train data was 0.04 and test data was 0.11 we will consider high R2 and low MAE (Mean Absolute Error) scores for “BEST MODEL”.

Adaboost:

```
adb = AdaBoostRegressor()
adb.fit(X_train,y_train)
```

```
AdaBoostRegressor()
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

```
performance(X_train,y_train,adb)  # Training R2 and MAE score
```

```
R2 score is :  0.5621682207244609
MAE score is :  0.5397295000993287
```

```
performance(X_test,y_test,adb)   # Testing R2 and MAE score
```

```
R2 score is :  0.555930885998144
MAE score is :  0.5427037117667921
```

AdaBoost gives me an on-train data R2 score is 0.56 and a test data R2 score is 0.55. MAE score on train data was 0.53 and test data was 0.54 we will consider high R2 and low MAE (Mean Absolute Error) scores for "BEST MODEL".

Gradient Boosting:

```
gb = GradientBoostingRegressor()
gb.fit(X_train,y_train)
```

```
GradientBoostingRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,gb)   # Training R2 and MAE score
```

```
R2 score is :  0.8321552931222456
MAE score is :  0.2738398848355144
```

```
performance(X_test,y_test,gb)   # Testing R2 and MAE score
```

```
R2 score is :  0.8284140949720513
MAE score is :  0.2757129883563975
```

Gradient Boost gives me an on-train data R2 score is 0.83 and a test data R2 score is 0.82. MAE score on train data was 0.27 and test data was 0.27 we will consider high R2 and low MAE (Mean Absolute Error) scores for "BEST MODEL".

Ridge:

```
rg = Ridge()
rg.fit(X_train,y_train)
```

```
Ridge()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,rg)  # Training R2 and MAE score
```

```
R2 score is :  0.4719032001965613
MAE score is :  0.5658360887755367
```

```
performance(X_test,y_test,rg)   # Testing R2 and MAE score
```

```
R2 score is :  0.4394130264342683
MAE score is :  0.5689568195536916
```

Ridge gives me an on-train data R2 score is 0.47 and a test data R2 score is 0.43. MAE score on train data was 0.56 and test data was 0.56 we will consider high R2 and low MAE (Mean Absolute Error) scores for "BEST MODEL".

We will check model performance after Hyperparameters tunning because we will do this to increase model performance.

```
%%time

param = {'alpha':[0.01, 0.1, 0.5, 1, 2,3,4]}

rg = Ridge()

rg_cv = GridSearchCV(estimator = rg, param_grid=param, cv = 5, scoring='r2')

rg_cv.fit(X_train,y_train)

rg_cv.best_params_
```

```
CPU times: total: 5.59 s
Wall time: 2.17 s
```

```
{'alpha': 4}
```

```
rg = Ridge(**rg_cv.best_params_)
rg.fit(X_train,y_train)
```

```
Ridge(alpha=4)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,rg)  # Training R2 and MAE score
```

```
R2 score is :  0.4719031997533423
MAE score is :  0.5658369246318171
```

```
performance(X_test,y_test,rg)  # Testing R2 and MAE score
```

```
R2 score is :  0.43941551056320893
MAE score is :  0.5689576339025795
```

After GridSearchCv we are getting the same performance on Ridge gives me an on-train data R2 score is 0.47 and a test data R2 score is 0.43. MAE score on train data was 0.56 and test data was 0.56 we will consider high R2 and low MAE (Mean Absolute Error) scores for "BEST MODEL".

Lasso:

```
ls = Lasso()
ls.fit(X_train,y_train)
```

```
Lasso()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,ls)  # Training R2 and MAE score
```

```
R2 score is :  0.2069429494611159
MAE score is :  0.7016738510076335
```

```
performance(X_test,y_test,ls)  # Testing R2 and MAE score
```

```
R2 score is :  0.20502929603409603
MAE score is :  0.7018316582826365
```

```
We will check model performance after Hyperparameters tunning because we will do this for incr
```

```
%%time

param = {'alpha':[0.001, 0.01, 0.1, 0.5, 1]}

ls = Lasso()

ls_cv = GridSearchCV(estimator=ls, param_grid= param, cv=5, scoring='r2' )

ls_cv.fit(X_train, y_train)

ls_cv.best_params_
```

```
CPU times: total: 6.28 s
Wall time: 2.42 s

{'alpha': 0.001}
```

```
ls = Lasso(**ls_cv.best_params_)
ls.fit(X_train,y_train)
```

```
Lasso(alpha=0.001)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,ls)  # Training R2 and MAE score
```

```
R2 score is :  0.4718920358181825
MAE score is :  0.5658951181727577
```

```
performance(X_test,y_test,ls)  # Testing R2 and MAE score
```

```
R2 score is :  0.4394705284247735
MAE score is :  0.5690367401461727
```

After GridSearchCv we are getting the same performance Lasso gives me an on-train data R2 score is 0.47 and a test data R2 score is 0.43. MAE score on train data was 0.56 and test data was 0.56 my model performance increased after hypertunning we will consider high R2 and low MAE (Mean Absolute Error) scores for “BEST MODEL”.

ElasticNet:

```
en = ElasticNet()
en.fit(X_train,y_train)
```

```
ElasticNet()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,en)  # Training R2 and MAE score
```

```
R2 score is :  0.20992424632204365
MAE score is :  0.6996958785506573
```

```
performance(X_test,y_test,en)   # Testing R2 and MAE score
```

```
R2 score is :  0.20748534539591534
MAE score is :  0.699881819895755
```

Elastic Net gives me an on-train data R2 score is 0.209 and a test data R2 score is 0.207. MAE score on train data was 0.6996 and test data was 0.6998 we will consider high R2 and low MAE (Mean Absolute Error) scores for “BEST MODEL”.

We will check model performance after Hyperparameters tunning because we will do this to increase model performance

```
%%time

param = {'alpha':[0.0001, 0.001, 0.01, 0.1, 0.5, 1],
         'l1_ratio':[0.01,0.1,0.3,0.5,0.7, 0.9, 1]}

eln = ElasticNet()

eln_cv = GridSearchCV(estimator=eln, param_grid= param, cv=5, scoring='r2')

eln_cv.fit(X_train, y_train)

eln_cv.best_params_
```

```
CPU times: total: 1min 5s
Wall time: 22.7 s

{'alpha': 0.01, 'l1_ratio': 0.01}
```

```
eln = ElasticNet(**eln_cv.best_params_)
eln.fit(X_train, y_train)
```

```
ElasticNet(alpha=0.01, l1_ratio=0.01)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,eln)  # Training R2 and MAE score
```

```
R2 score is :  0.47188248172446445
MAE score is :  0.5660725245731264
```

```
performance(X_test,y_test,eln)   # Testing R2 and MAE score
```

```
R2 score is :  0.4400559492700332
MAE score is :  0.5691857370427704
```

After GridSearchCv we are getting the same performance on Elastic Net gives me an on-train data R2 score is 0.4718 and a test data R2 score is 0.44. MAE score on train data was 0.5660 and test data was 0.5691 we will consider high R2 and low MAE (Mean Absolute Error) scores for “BEST MODEL”.

XGBoost:

```
xgb = XGBRegressor()
xgb.fit(X_train,y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
performance(X_train,y_train,xgb)  # Training R2 and MAE score
```

```
R2 score is :  0.9281324653711821
MAE score is :  0.1681816011164085
```

```
performance(X_test,y_test,xgb)   # Testing R2 and MAE score
```

```
R2 score is :  0.9039083598018871
MAE score is :  0.19075903643342831
```

XGBoost gives me an on-train data R2 score is 0.92 and a test data R2 score is 0.90. MAE score on train data was 0.16 and test data was 0.19 we will consider high R2 and low MAE (Mean Absolute Error) scores for “BEST MODEL”.

For the final Business **Conclusion**, we will see the score table of all the models and we will go ahead:

	Model	R2 Train Score	R2 Test Score	MAE Train Score	MAE Test Score
0	Linear Regression	0.471903	0.439412	0.565836	0.568957
1	Decision Tree	0.689734	0.692789	0.394838	0.393490
2	KNN	0.644868	0.451920	0.423559	0.531718
3	Random Forest	0.990560	0.933379	0.043516	0.116758
4	Adaboost	0.562168	0.555931	0.539730	0.542704
5	Gradient Boosting	0.832155	0.828414	0.273840	0.275713
6	Ridge	0.471903	0.439416	0.565837	0.568958
7	Lasso	0.471892	0.439471	0.565895	0.569037
8	Elastic Net	0.471882	0.440056	0.566073	0.569186
9	XGBoost	0.928132	0.903908	0.168182	0.190759

Business Inference:

We did many models such as Linear Regression, Decision Tree, bagging technique Random forest, and boosting techniques GradientBoosting, Adaboost, Xgboost other models are KNN, Ridge, Lasso, Elastic Net.

Our final and best model from all the models is RandomForest because that gives us good scores on test data or unseen data so my BEST model is RandomForest.

Random Forest gives me an on-train data R2 score is 0.99 and a test data R2 score is 0.93. MAE score on train data was 0.04 and test data was 0.11 we will consider high R2 and low MAE (Mean Absolute Error) scores for “BEST MODEL”.

We are getting an R2 score on the train is 0.990560 and on test data, R2 is 0.933379 which is my "BEST" model because that model score is not overfitting, underfitting, and variance so that's why this is my best model.

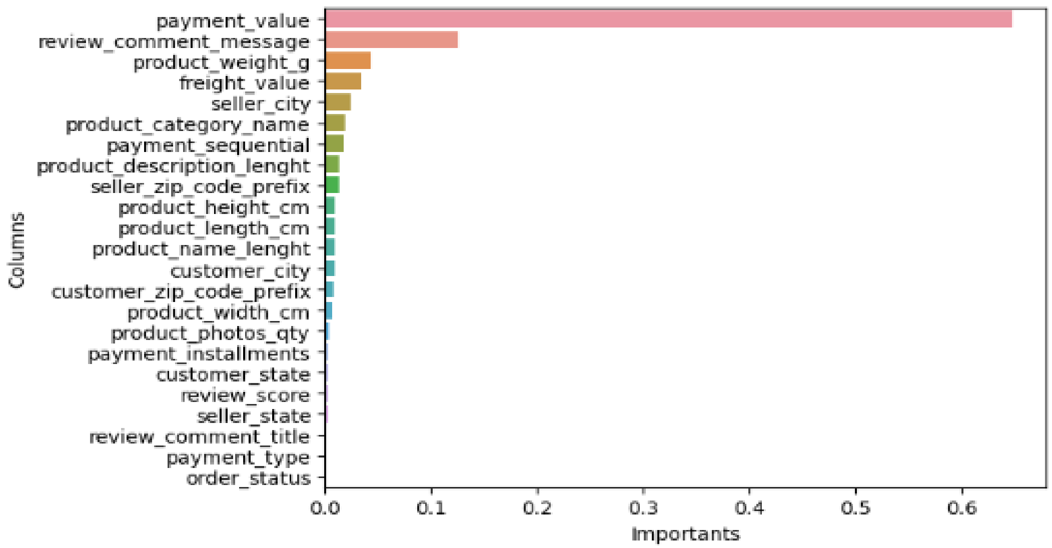
Also, one more measurement in the regression problem which is MAE (Mean Absolute Error) also gave me a low which is the best in comparison to other models.

We will get Important Features that give me my BEST MODEL Random Forest:

```
imp_fea = pd.DataFrame(zip(X_train.columns, feature_importants),columns = ['Columns','Importants'])
imp_fea = imp_fea.sort_values('Importants', ascending = False)
imp_fea
```

	Columns	Importants
3	freight_value	0.647674
17	payment_installments	0.125742
8	seller_zip_code_prefix	0.043024
0	payment_value	0.034029
21	payment_type	0.024153
18	review_score	0.019221
1	review_comment_message	0.016763
6	payment_sequential	0.012738
13	customer_zip_code_prefix	0.012659
10	customer_city	0.009080
9	product_height_cm	0.008698
5	product_category_name	0.008478
19	seller_state	0.008475
12	product_name_lenght	0.007154
11	product_length_cm	0.006380
7	product_description_lenght	0.004236
2	product_weight_g	0.002394
20	review_comment_title	0.002385
4	seller_city	0.002088
22	order_status	0.001687
16	customer_state	0.001482
15	product_photos_qty	0.001120
14	product_width_cm	0.000343

```
sns.barplot(y = 'Columns', x = 'Importants', data = imp_fea)
plt.show()
```



So, the conclusion from the above analysis is, businesses can focus on important features as follows:

payment_value, review_comment_message, product_weight_g, freight_value, seller_city, and many more SO THEY CAN INCREASE SALES OF THE PRODUCT AS WELL AS INCREASE THE REVENUE OF COMPANY.

BUSINESSES MUST FOCUS ON THESE IMPORTANT FEATURES TO INCREASE SALES AND REVENUE.

Implications:

Revenue Optimization: Implementing a predictive pricing model enables Olist to strategically optimize its revenue streams. By accurately forecasting product prices, Olist can capture maximum value from each transaction, ensuring competitiveness while maximizing profitability.

Market Differentiation: The adoption of advanced pricing strategies distinguishes Olist within the Brazilian e-commerce landscape. The ability to dynamically adjust prices in response to market fluctuations and consumer behavior sets Olist apart from competitors, reinforcing its position as a market leader.

Customer Experience Enhancement: Accurate pricing fosters positive customer experiences by aligning perceived value with pricing. Customers are more likely to engage with Olist's platform when they perceive fairness and transparency in pricing, leading to increased satisfaction, loyalty, and repeat purchases.

Operational Efficiency: Automation of pricing processes through predictive modeling streamlines operations at Olist. By reducing manual intervention and optimizing pricing decisions, Olist improves operational efficiency, reduces costs, and enhances overall productivity.

Data-Driven Insights: The project underscores Olist's commitment to data-driven decision-making. By harnessing data analytics and machine learning, Olist gains actionable insights into pricing trends, customer preferences, and market dynamics, enabling proactive and informed decision-making across all business functions.

Sustainable Growth: The successful deployment of the predictive pricing model lays the foundation for sustainable growth and long-term success at Olist. By continuously refining pricing strategies based on data-driven insights, Olist drives revenue growth, enhances market position, and builds a foundation for future expansion and innovation.

Limitations:

Data Quality and Completeness: Inaccurate or incomplete data can impact model performance.

Model Complexity: Advanced models may be difficult to interpret, hindering stakeholder understanding.

Assumption of Stationarity: The Model assumes consistent market dynamics, which may not hold true.

Limited Scope of Variables: Exclusion of important factors like competitor pricing can limit accuracy.

Overfitting: Models may learn noise in data, reducing generalization to new data.

Ethical Considerations: Biased pricing practices may harm customer trust and brand reputation.

Regulatory Compliance: Pricing decisions must adhere to legal requirements, posing compliance challenges.

Market Dynamics: External factors like consumer preferences can influence pricing, impacting model accuracy.

Model Maintenance: Ongoing updates and adjustments are needed to keep the model relevant.

Closing Reflections:

Our project aimed to develop a predictive pricing model to optimize revenue and customer satisfaction for Olist, a leading Brazilian e-commerce platform. Leveraging a comprehensive dataset of order information, customer feedback, and product attributes, we trained regression models to forecast product prices accurately.

Our results demonstrate the potential of machine learning in optimizing pricing strategies for e-commerce platforms. By accurately predicting prices, Olist can set competitive prices while maximizing profitability, enhancing customer satisfaction, and driving revenue growth. These insights can inform pricing decisions and contribute to Olist's competitive advantage in the Brazilian market.

However, it's important to acknowledge the limitations of our approach. Data quality and completeness, model complexity, and external market dynamics pose challenges to the accuracy and effectiveness of the predictive pricing model. Continuous refinement, validation, and adaptation are necessary to address these limitations and improve model performance over time.

Overall, this project underscores the importance of data-driven decision-making in e-commerce and highlights the potential of predictive analytics to drive business success. By embracing innovation and leveraging insights from predictive pricing models, Olist can strengthen its position in the market, enhance customer experiences, and achieve sustainable growth in the long term.