EDA + Satisfaction Prediction + Reviews NLP + RFM Analysis + Deployment

Table of Contents

1.0 Introduction

This project is about "Olist", a Brazilian ecommerce store which has information of 100k orders from 2016 to 2018 made at multiple marketplaces in Brazil where its features allow viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers.

Overview of Final Web App

2.0 Data Loading

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```


Read All Files

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/eda-satisfaction-prediction-nlp-rfm-deployment/__results__.html
/kaggle/input/eda-satisfaction-prediction-nlp-rfm-deployment/__notebook_source_
/kaggle/input/eda-satisfaction-prediction-nlp-rfm-deployment/__resultx__.html
/kaggle/input/eda-satisfaction-prediction-nlp-rfm-deployment/__notebook__.ipynb
/kaggle/input/eda-satisfaction-prediction-nlp-rfm-deployment/__output__.json
/kaggle/input/eda-satisfaction-prediction-nlp-rfm-deployment/custom.css
/kaggle/input/brazilian-ecommerce/olist_customers_dataset.csv
/kaggle/input/brazilian-ecommerce/olist_sellers_dataset.csv
/kaggle/input/brazilian-ecommerce/olist_order_reviews_dataset.csv
/kaggle/input/brazilian-ecommerce/olist_order_items_dataset.csv
/kaggle/input/brazilian-ecommerce/olist_products_dataset.csv
/kaggle/input/brazilian-ecommerce/olist_geolocation_dataset.csv
/kaggle/input/brazilian-ecommerce/product_category_name_translation.csv
/kaggle/input/brazilian-ecommerce/olist_orders_dataset.csv
/kaggle/input/brazilian-ecommerce/olist_order_payments_dataset.csv
```

```
customers_df= pd.read_csv('/kaggle/input/brazilian-ecommerce/olist_customers_dataset
geolocation_df= pd.read_csv('/kaggle/input/brazilian-ecommerce/olist_geolocation_dat
items_df= pd.read_csv('/kaggle/input/brazilian-ecommerce/olist_order_items_dataset.c
payments_df= pd.read_csv('/kaggle/input/brazilian-ecommerce/olist_order_payments_dat
reviews_df= pd.read_csv('/kaggle/input/brazilian-ecommerce/olist_order_reviews_datas
orders_df= pd.read_csv('/kaggle/input/brazilian-ecommerce/olist_orders_dataset.csv')
products_df= pd.read_csv('/kaggle/input/brazilian-ecommerce/olist_products_dataset.c
sellers_df= pd.read_csv('/kaggle/input/brazilian-ecommerce/olist_sellers_dataset.csv
category_translation_df= pd.read_csv('/kaggle/input/brazilian-ecommerce/product_cate
```

"Customers" Dataset

```
customers_df.head()
```

	customer_id	customer_unique_id	customer_zip_
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	

"Geolocation" Dataset

```
geolocation_df.head()
```

	geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_ci
0	1037	-23.545621	-46.639292	sao pa
1	1046	-23.546081	-46.644820	sao pa
2	1046	-23.546129	-46.642951	sao pa
3	1041	-23.544392	-46.639499	sao pa
4	1035	-23.541578	-46.641607	sao pa

"Order items" Dataset

```
items_df.head()
```

	order_id	order_item_id	product_id
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089

"Order Payments" Dataset

```
payments_df.head()
```

	order_id	payment_sequential	payment_type	payment_ins
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	

"Order Reviews" Dataset

```
reviews_df.head()
```

	review_id	order_id	review_score
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5

"Orders" Dataset

```
orders_df.head()
```

	order_id	customer_id	order_status	o
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	

"Products" Dataset

```
products_df.head()
```

	product_id	product_category_name	product_name_lenght	pr
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	
3	cef67bcfe19066a932b7673e239eb23d	bebes	27.0	
4	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	37.0	

"Sellers" Dataset

```
sellers_df.head()
```

	seller_id	seller_zip_code_prefix	seller_city	seller_st
0	3442f8959a84dea7ee197c632cb2df15	13023	campinas	
1	d1b65fc7debc3361ea86b5f14c68d2e2	13844	mogi guacu	
2	ce3ad9de960102d0677a81f5d0bb7b2d	20031	rio de janeiro	
3	c0f3eea2e14555b6faeea3dd58c1b1c3	4195	sao paulo	

"Product Category Name Translation" Dataset

```
category_translation_df.head()
```

	product_category_name	product_category_name_english
0	beleza_saude	health_beauty
1	informatica_acessorios	computers_accessories
2	automotivo	auto
3	cama_mesa_banho	bed_bath_table
4	moveis_decoracao	furniture_decor

3.0 Data Cleaning

3.1 Merging All Dataframes

```
df= pd.merge(customers_df, orders_df, on="customer_id", how='inner')
df= df.merge(reviews_df, on="order_id", how='inner')
df= df.merge(items_df, on="order_id", how='inner')
df= df.merge(products_df, on="product_id", how='inner')
df= df.merge(payments_df, on="order_id", how='inner')
df= df.merge(sellers_df, on='seller_id', how='inner')
df= df.merge(category_translation_df, on='product_category_name', how='inner')
df.shape
```

```
(115609, 40)
```

Show All Features

```
df.columns
```

```
Index(['customer_id', 'customer_unique_id', 'customer_zip_code_prefix',
       'customer_city', 'customer_state', 'order_id', 'order_status',
       'order_purchase_timestamp', 'order_approved_at',
       'order_delivered_carrier_date', 'order_delivered_customer_date',
       'order_estimated_delivery_date', 'review_id', 'review_score',
```

```
       'review_comment_title', 'review_comment_message',
       'review_creation_date', 'review_answer_timestamp', 'order_item_id',
       'product_id', 'seller_id', 'shipping_limit_date', 'price',
       'freight_value', 'product_category_name', 'product_name_lenght',
       'product_description_lenght', 'product_photos_qty', 'product_weight_g',
       'product_length_cm', 'product_height_cm', 'product_width_cm',
       'payment_sequential', 'payment_type', 'payment_installments',
       'payment_value', 'seller_zip_code_prefix', 'seller_city',
       'seller_state', 'product_category_name_english'],
      dtype='object')
```

Check duplicates

```
df.duplicated().sum()
```

```
0
```

```
df.describe()
```

	customer_zip_code_prefix	review_score	order_item_id	price	freig
count	115609.000000	115609.000000	115609.000000	115609.000000	1156
mean	35061.537597	4.034409	1.194535	120.619850	
std	29841.671732	1.385584	0.685926	182.653476	
min	1003.000000	1.000000	1.000000	0.850000	
25%	11310.000000	4.000000	1.000000	39.900000	
50%	24241.000000	5.000000	1.000000	74.900000	
75%	58745.000000	5.000000	1.000000	134.900000	
max	99980.000000	5.000000	21.000000	6735.000000	4

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 115609 entries, 0 to 115608
Data columns (total 40 columns):
 #   Column                         Non-Null Count   Dtype 
---  ------                         --------------   ----- 
 0   customer_id                    115609 non-null  object
 1   customer_unique_id             115609 non-null  object
 2   customer_zip_code_prefix       115609 non-null  int64 
 3   customer_city                  115609 non-null  object
 4   customer_state                 115609 non-null  object
 5   order_id                       115609 non-null  object
 6   order_status                   115609 non-null  object
 7   order_purchase_timestamp       115609 non-null  object
 8   order_approved_at              115595 non-null  object
 9   order_delivered_carrier_date   114414 non-null  object
 10  order_delivered_customer_date  113209 non-null  object
 11  order_estimated_delivery_date  115609 non-null  object
```

```
 12  review_id                      115609 non-null  object
 13  review_score                   115609 non-null  int64
 14  review_comment_title           13801 non-null   object
 15  review_comment_message         48906 non-null   object
 16  review_creation_date           115609 non-null  object
 17  review_answer_timestamp        115609 non-null  object
 18  order_item_id                  115609 non-null  int64
 19  product_id                     115609 non-null  object
 20  seller_id                      115609 non-null  object
 21  shipping_limit_date            115609 non-null  object
 22  price                          115609 non-null  float64
 23  freight_value                  115609 non-null  float64
 24  product_category_name          115609 non-null  object
 25  product_name_lenght            115609 non-null  float64
 26  product_description_lenght     115609 non-null  float64
 27  product_photos_qty             115609 non-null  float64
 28  product_weight_g               115608 non-null  float64
 29  product_length_cm              115608 non-null  float64
 30  product_height_cm              115608 non-null  float64
 31  product_width_cm               115608 non-null  float64
 32  payment_sequential             115609 non-null  int64
 33  payment_type                   115609 non-null  object
 34  payment_installments           115609 non-null  int64
 35  payment_value                  115609 non-null  float64
 36  seller_zip_code_prefix         115609 non-null  int64
 37  seller_city                    115609 non-null  object
 38  seller_state                   115609 non-null  object
 39  product_category_name_english  115609 non-null  object
dtypes: float64(10), int64(6), object(24)
memory usage: 36.2+ MB
```

3.2 Handling Missing Values

```
# Number of Missing Values for the first half of features

df.isna().sum()[:20]
```

```
customer_id                         0
customer_unique_id                  0
customer_zip_code_prefix            0
customer_city                       0
customer_state                      0
order_id                            0
order_status                        0
order_purchase_timestamp            0
order_approved_at                  14
order_delivered_carrier_date     1195
order_delivered_customer_date    2400
order_estimated_delivery_date       0
review_id                           0
review_score                        0
review_comment_title           101808
review_comment_message          66703
review_creation_date                0
review_answer_timestamp             0
order_item_id                       0
product_id                          0
dtype: int64
```

Drop All Missing Values in datetime columns

```
df.dropna(subset= ['order_approved_at', 'order_delivered_carrier_date', 'order_deliv
```

Keep " review_comment_message " & " review_comment_title " Features (Will be handled later)

```
# Number of Missing Values for the Second half of features

df.isna().sum()[20:]
```

```
seller_id                       0
shipping_limit_date             0
price                           0
freight_value                   0
product_category_name           0
product_name_lenght             0
product_description_lenght      0
product_photos_qty              0
product_weight_g                1
product_length_cm               1
product_height_cm               1
product_width_cm                1
payment_sequential              0
payment_type                    0
payment_installments            0
payment_value                   0
seller_zip_code_prefix          0
seller_city                     0
seller_state                    0
product_category_name_english   0
dtype: int64
```

Check the missing values

```
df[['product_weight_g', 'product_length_cm', 'product_height_cm', 'product_width_cm'
```

	product_weight_g	product_length_cm	product_height_cm	product_width_cm
27352	NaN	NaN	NaN	NaN

```
# Since all the missing values are in the same raw, we will drop this raw
df.drop(27352, inplace=True)

# Reset Index
df.reset_index(inplace= True, drop= True)
```

3.3 Feature Engineering

Classify Products Categories (71) into 9 main Categories

```
def classify_cat(x):

    if x in ['office_furniture', 'furniture_decor', 'furniture_living_room', 'kitche
        return 'Furniture'

    elif x in ['auto', 'computers_accessories', 'musical_instruments', 'consoles_gam
        return 'Electronics'

    elif x in ['fashio_female_clothing', 'fashion_male_clothing', 'fashion_bags_acce
        return 'Fashion'

    elif x in ['housewares', 'home_confort', 'home_appliances', 'home_appliances_2',
        return 'Home & Garden'

    elif x in ['sports_leisure', 'toys', 'cds_dvds_musicals', 'music', 'dvds_blu_ray
        return 'Entertainment'

    elif x in ['health_beauty', 'perfumery', 'diapers_and_hygiene']:
        return 'Beauty & Health'

    elif x in ['food_drink', 'drinks', 'food']:
        return 'Food & Drinks'

    elif x in ['books_general_interest', 'books_technical', 'books_imported', 'stati
        return 'Books & Stationery'

    elif x in ['construction_tools_construction', 'construction_tools_safety', 'indu
        return 'Industry & Construction'

df['product_category'] = df.product_category_name_english.apply(classify_cat)
```

```
df.product_category.value_counts()
```

```
Electronics               29568
Furniture                 28050
Entertainment             13507
Beauty & Health           13283
Home & Garden             12848
Fashion                    9722
Books & Stationery         3427
Industry & Construction    1633
Food & Drinks              1155
Name: product_category, dtype: int64
```

Combine Width, Height and Length to get Product Volume

```
# Create Volume Column
df['product_vol_cm3'] = df.product_length_cm * df.product_width_cm * df.product_heigl

# Drop Width, Height and Length
df.drop(['product_length_cm', 'product_width_cm', 'product_height_cm'], axis= 1, inp
```

Convert Datetime features from Object to Datetime

```
df['order_purchase_timestamp'] = pd.to_datetime(df['order_purchase_timestamp'])
df['order_delivered_customer_date'] = pd.to_datetime(df['order_delivered_customer_da
df['order_estimated_delivery_date'] = pd.to_datetime(df['order_estimated_delivery_da
df['shipping_limit_date'] = pd.to_datetime(df['shipping_limit_date'])
df['order_delivered_carrier_date'] =pd.to_datetime(df['order_delivered_carrier_date'
```

Extract duration of estimated shipping from purchasing date untill estimated delivery date

```
df['estimated_days'] = (df['order_estimated_delivery_date'].dt.date - df['order_purcl
```

Extract duration of shipping from purchasing date until delivered to customer date

```
df['arrival_days'] = (df['order_delivered_customer_date'].dt.date - df['order_purcha
```

Extract duration of shipping from purchasing carrier delivered date untill delivered to customer

```
df['shipping_days'] = (df['order_delivered_customer_date'].dt.date - df['order_deliv
```

Drop inconsistent dates where "order_delivered_carrier_date" is greater than "order_delivered_customer_date"

```
df.drop((df[['order_delivered_carrier_date', 'order_delivered_customer_date']][df.sh
```

Shipping status from Seller to Carrier

```
# First get seller to carrier duration in days
df['seller_to_carrier_status'] = (df['shipping_limit_date'].dt.date - df['order_deli

# Now calssify the duration into 'OnTime/Early' & 'Late'
df['seller_to_carrier_status'] = df['seller_to_carrier_status'].apply(lambda x : 'On
```

Shipping status from Carrier to Customer

```
# First get difference between estimated delivery date and actual delivery date in d
df['arrival_status'] = (df['order_estimated_delivery_date'].dt.date - df['order_deli
```

```
# Now Classify the duration in 'OnTime/Early' & 'Late'
df['arrival_status'] = df['arrival_status'].apply(lambda x : 'OnTime/Early' if x >=0
```

Show statistics of new Features

```
df[['estimated_days', 'arrival_days', 'shipping_days']].describe()
```

	estimated_days	arrival_days	shipping_days
count	113140.000000	113140.000000	113140.000000
mean	24.469427	12.383578	9.127594
std	8.825504	9.365912	8.567799
min	3.000000	0.000000	0.000000
25%	19.000000	7.000000	4.000000
50%	24.000000	10.000000	7.000000
75%	29.000000	15.000000	12.000000
max	156.000000	208.000000	205.000000

Remove Outliers in both features (More than 60 days)

```
outlier_indices = df[(df.estimated_days > 60) | (df.arrival_days > 60) | (df.shippin

df.drop(outlier_indices, inplace= True)
df.reset_index(inplace= True, drop= True)
```

Rating estimated delivery time

```
def rates(x):

    if x in range(0, 8):
        return 'Very Fast'

    elif x in range(8, 16):
        return 'Fast'

    elif x in range(16, 25):
        return 'Neutral'

    elif x in range(25, 40):
        return 'Slow'

    else:
        return 'Very Slow'
```

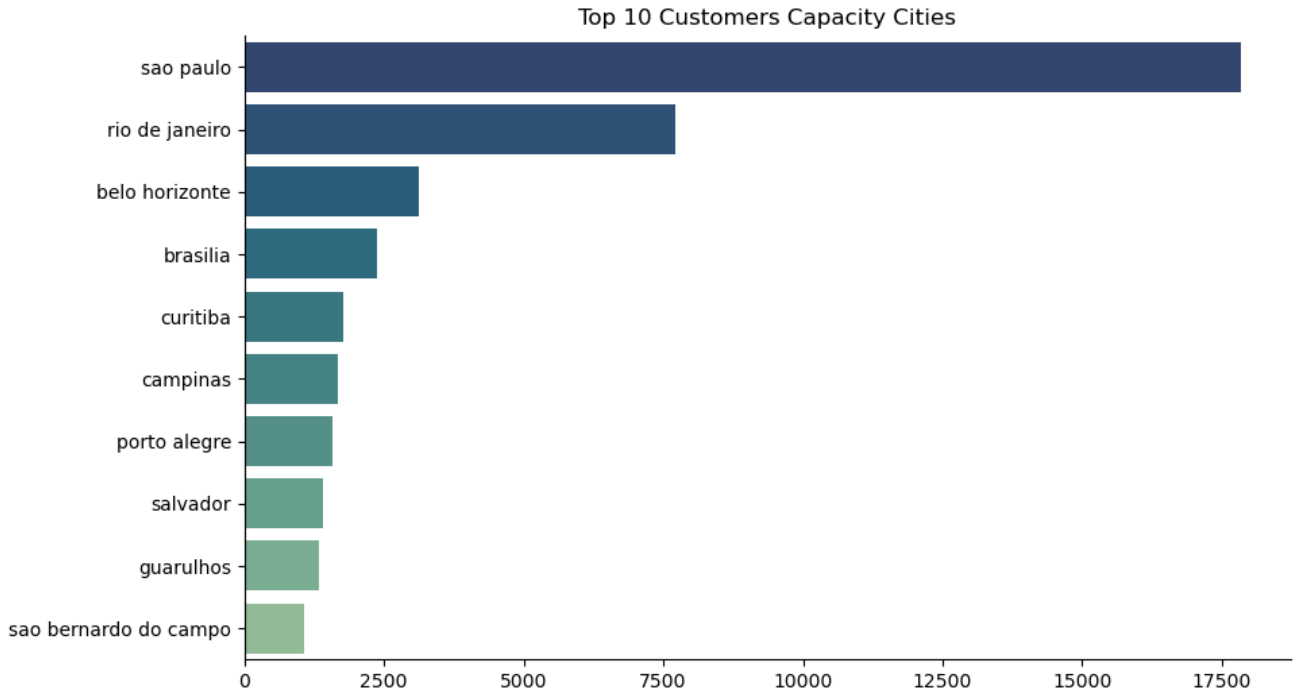
```
df['estimated_delivery_rate'] = df.estimated_days.apply(rates)

df['arrival_delivery_rate'] = df.arrival_days.apply(rates)

df['shipping_delivery_rate'] = df.shipping_days.apply(rates)
```

4.0 Exploratory Data Analysis (EDA)

4.1 Univariate Analysis

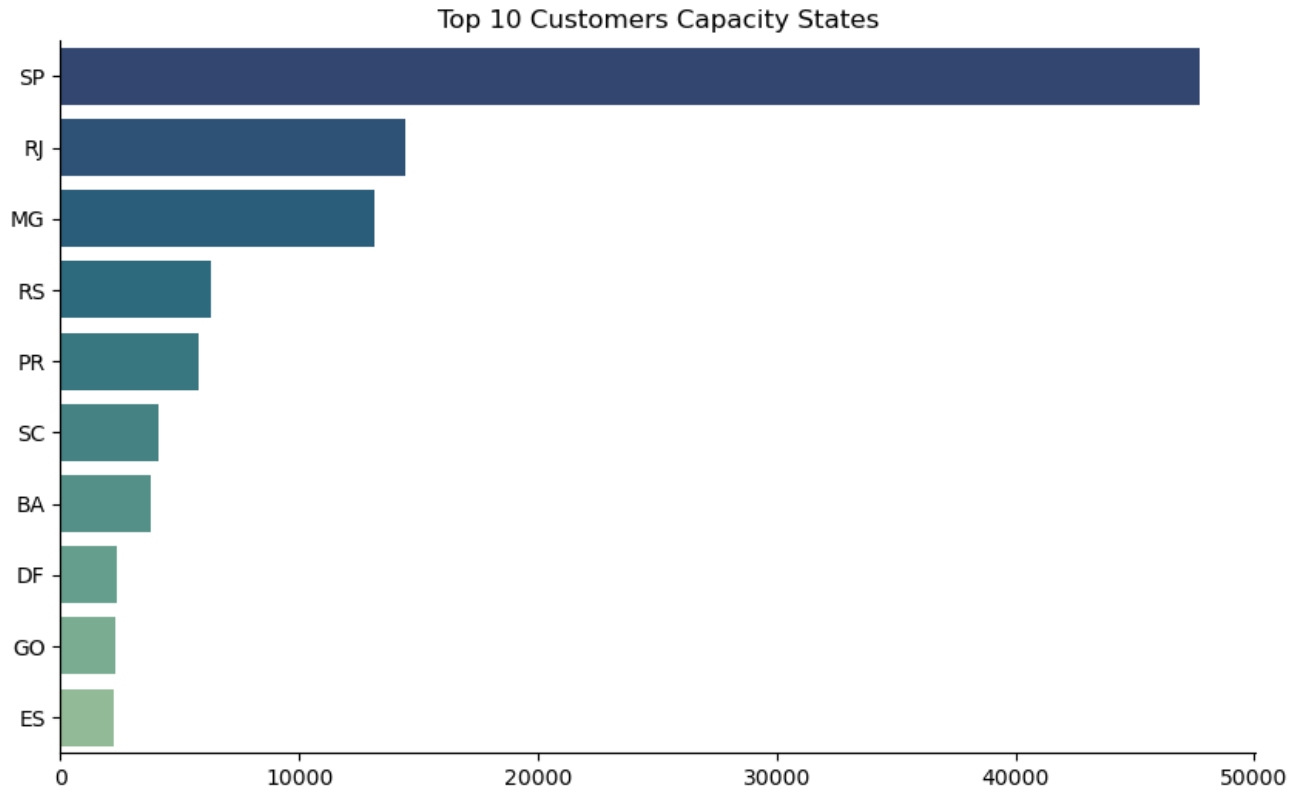
Top 10 Customers Capacity Cities

```
plt.figure(figsize=[10, 6])
sns.barplot(x = df.customer_city.value_counts().values[:10], y = df.customer_city.va
plt.title('Top 10 Customers Capacity Cities')
sns.despine()
```



Top 10 Customers Capacity States

```
plt.figure(figsize=[10, 6])
sns.barplot(x = df.customer_state.value_counts().values[:10], y = df.customer_state.
plt.title('Top 10 Customers Capacity States')
sns.despine()
```



"Order_Status"

```
df.order_status.value_counts()
```

```
delivered    112526
canceled          7
Name: order_status, dtype: int64
```

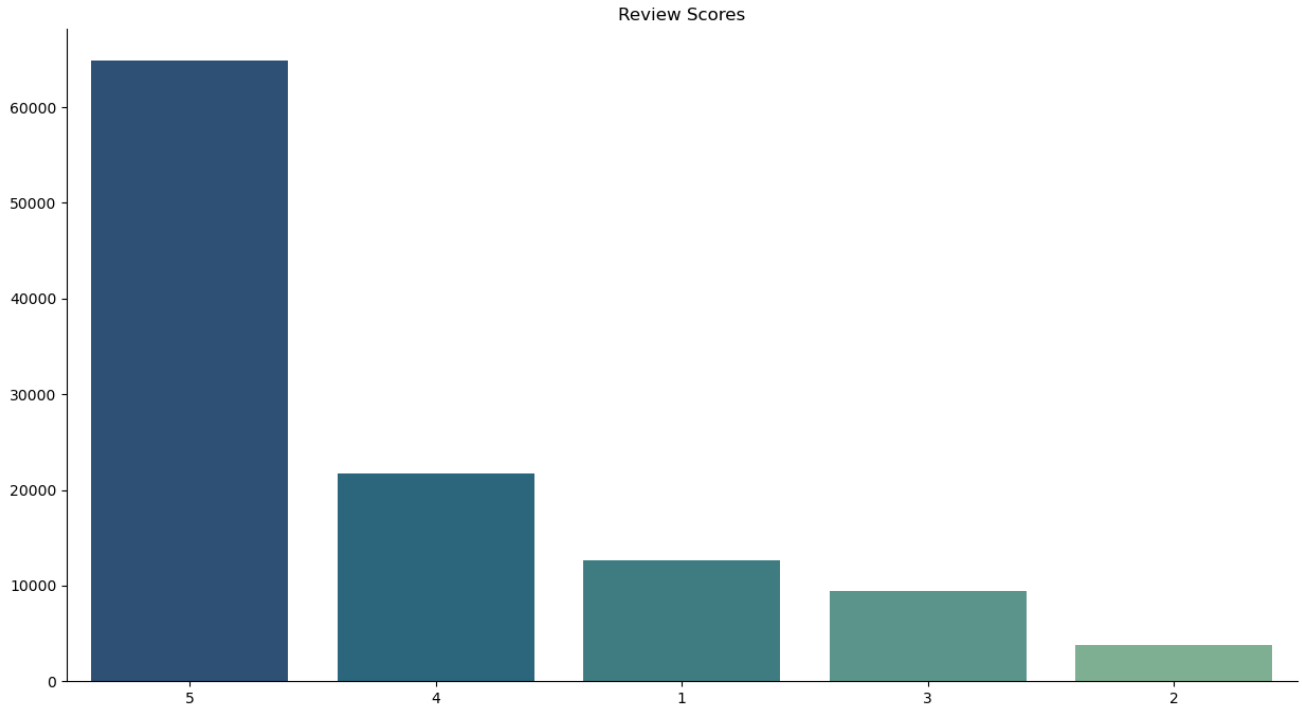
Due to the Extreme imbalance and lack of variance in the feature, we should drop it

```
df.drop('order_status', axis=1, inplace=True)
```

" Review_Score "

```
plt.figure(figsize=[15, 8])
review_score_index = [str(i) for i in df.review_score.value_counts().index]
sns.barplot(x = review_score_index, y= df.review_score.value_counts().values, palett
```

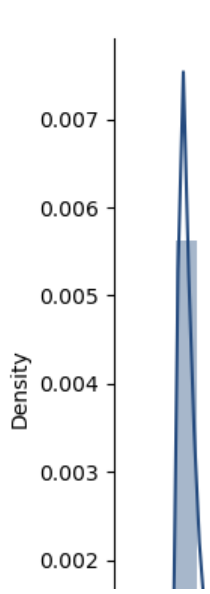
```
plt.title('Review Scores')
sns.despine()
```



" Price "

```
plt.figure(figsize=[10, 6])
sns.set_palette('crest_r')
sns.distplot(x = df.price)
plt.title('Price Distribution')
sns.despine()
```

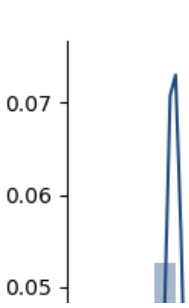
```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  This is separate from the ipykernel package so we can avoid doing imports unt
```



" Freight Value "

```
plt.figure(figsize=[10, 6])
sns.set_palette('crest_r')
sns.distplot(x = df.freight_value)
plt.title('Freight Value Distribution')
sns.despine()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  This is separate from the ipykernel package so we can avoid doing imports unt
```



" Number of orders per each Category "

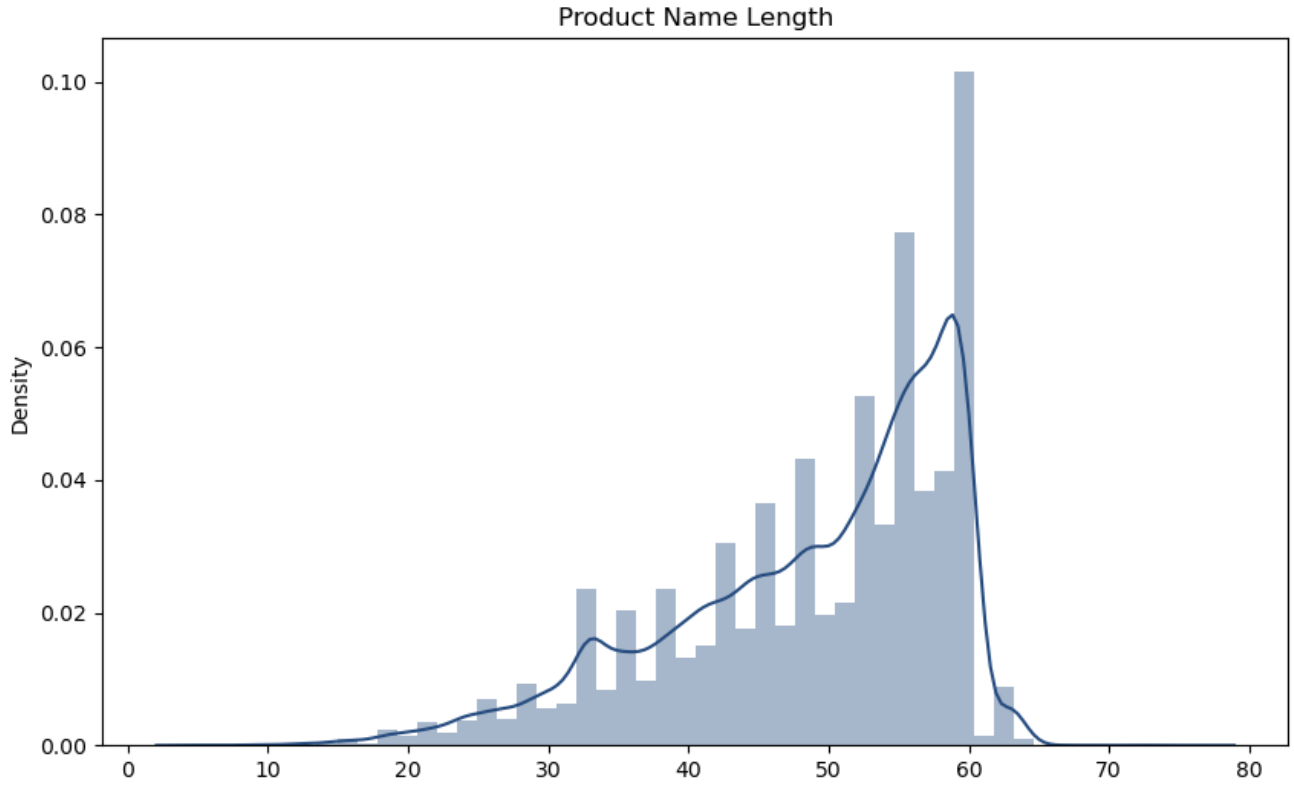
```
plt.figure(figsize=[10, 6])
sns.barplot(x = df.product_category.value_counts().values, y = df.product_category.v
plt.title('Number of orders per each Category')
plt.xticks(rotation = 45)
sns.despine()
```

" Product Name Length "

```
plt.figure(figsize=[10, 6])
sns.distplot(x = df.product_name_lenght)
plt.title('Product Name Length')
df.product_category.value_counts().values
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

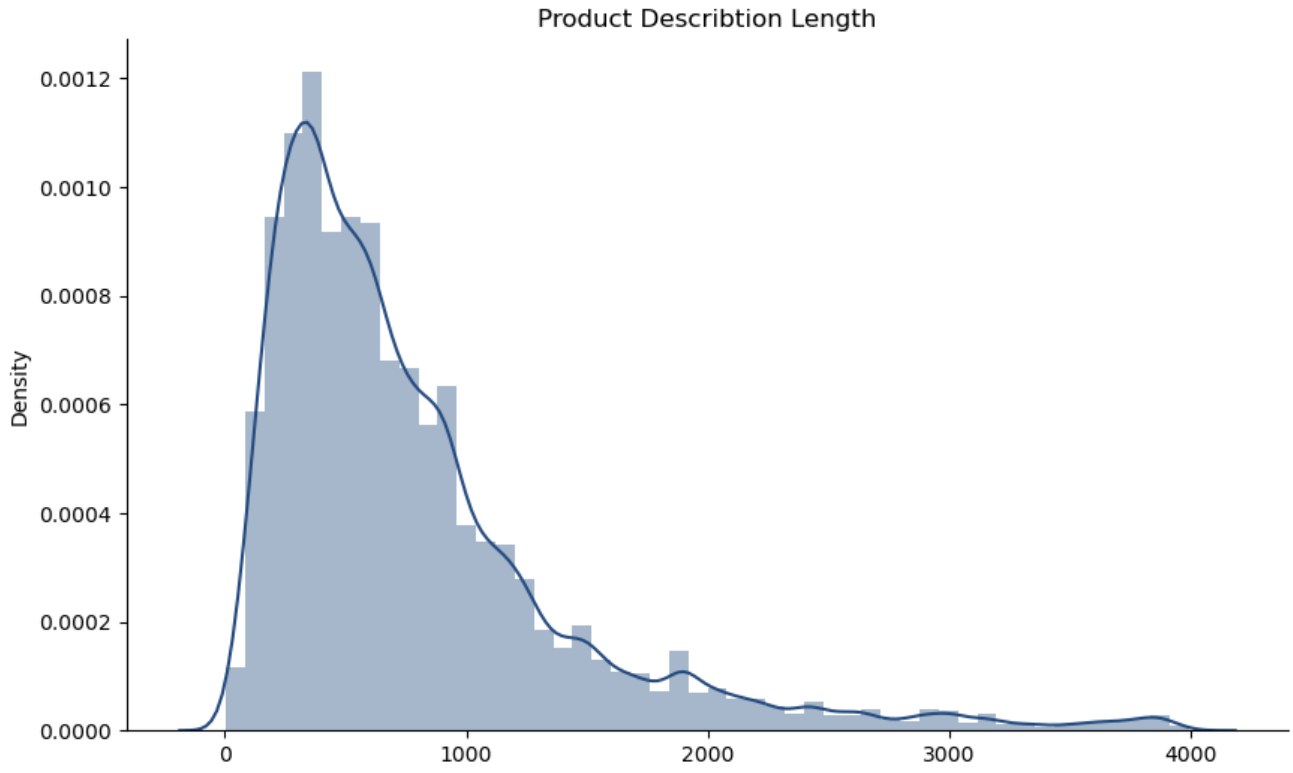
```
array([29441, 27778, 13450, 13219, 12780,  9679,  3410,  1630,  1146])
```



" Product Description Length "

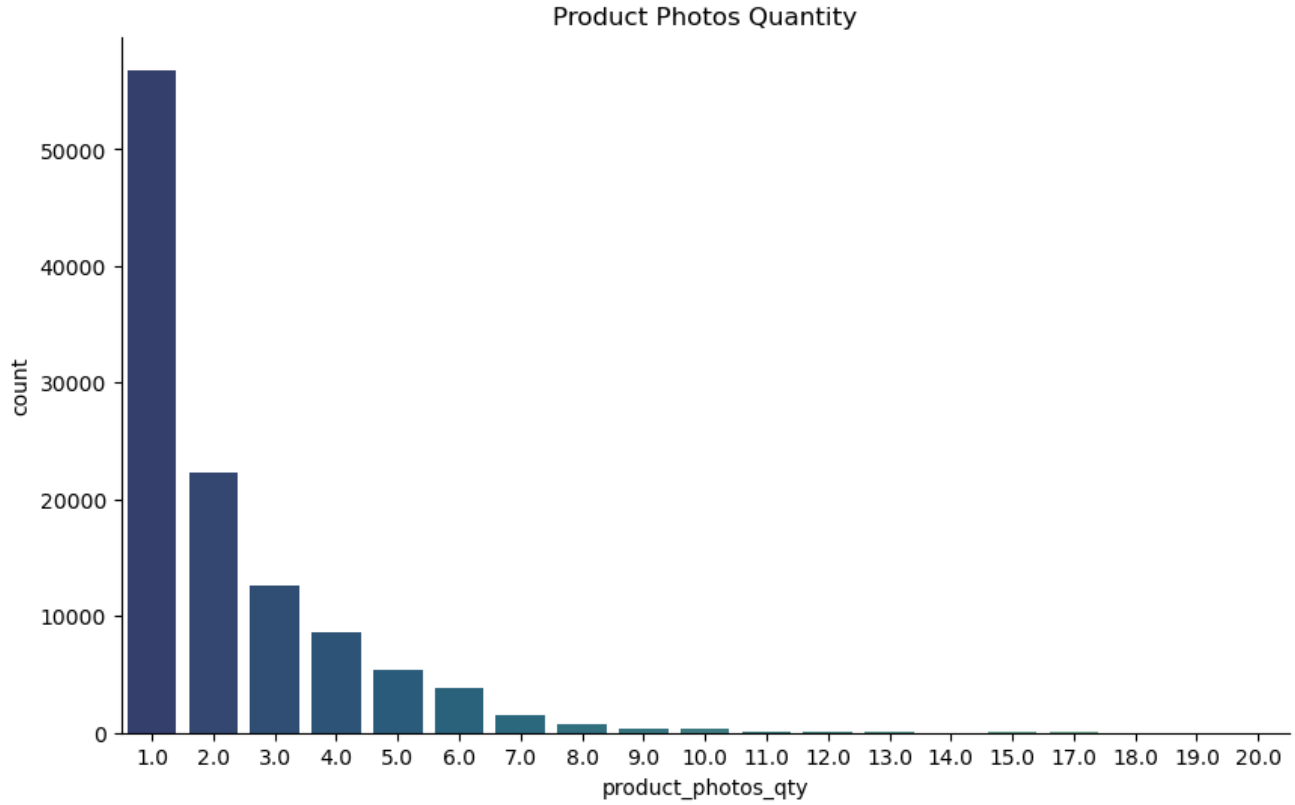
```
plt.figure(figsize=[10, 6])
sns.distplot(x = df.product_description_lenght)
plt.title('Product Describtion Length')
sns.despine()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```



" Product Photos Quantity "

```
plt.figure(figsize=[10, 6])
sns.countplot(x = df.product_photos_qty, palette= 'crest_r')
plt.title('Product Photos Quantity')
sns.despine()
```



" Product Weight "

```
plt.figure(figsize=[10, 6])
sns.distplot(x = df.product_weight_g)
plt.title('Product Weight')
sns.despine()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

" Product Volume "


```
plt.figure(figsize=[10, 6])
sns.distplot(x = df.product_vol_cm3)
plt.title('Product Volume')
sns.despine()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

" Payment Type "

```
plt.figure(figsize=[10, 10])
plt.pie(df.payment_type.value_counts().values, explode=(0.05, 0.05, 0.05, 0.05), lab
```

" Payment Installments "

```
df.payment_installments.value_counts()
```

```
1     56117
2     13093
3     11255
4      7635
10     6574
5      5779
8      4844
6      4417
7      1721
9       685
12      157
15       85
18       38
24       34
11       22
20       19
13       18
14       15
17        7
16        7
21        6
0         3
23        1
22        1
Name: payment_installments, dtype: int64
```

```
df[df.payment_installments == 0]
```

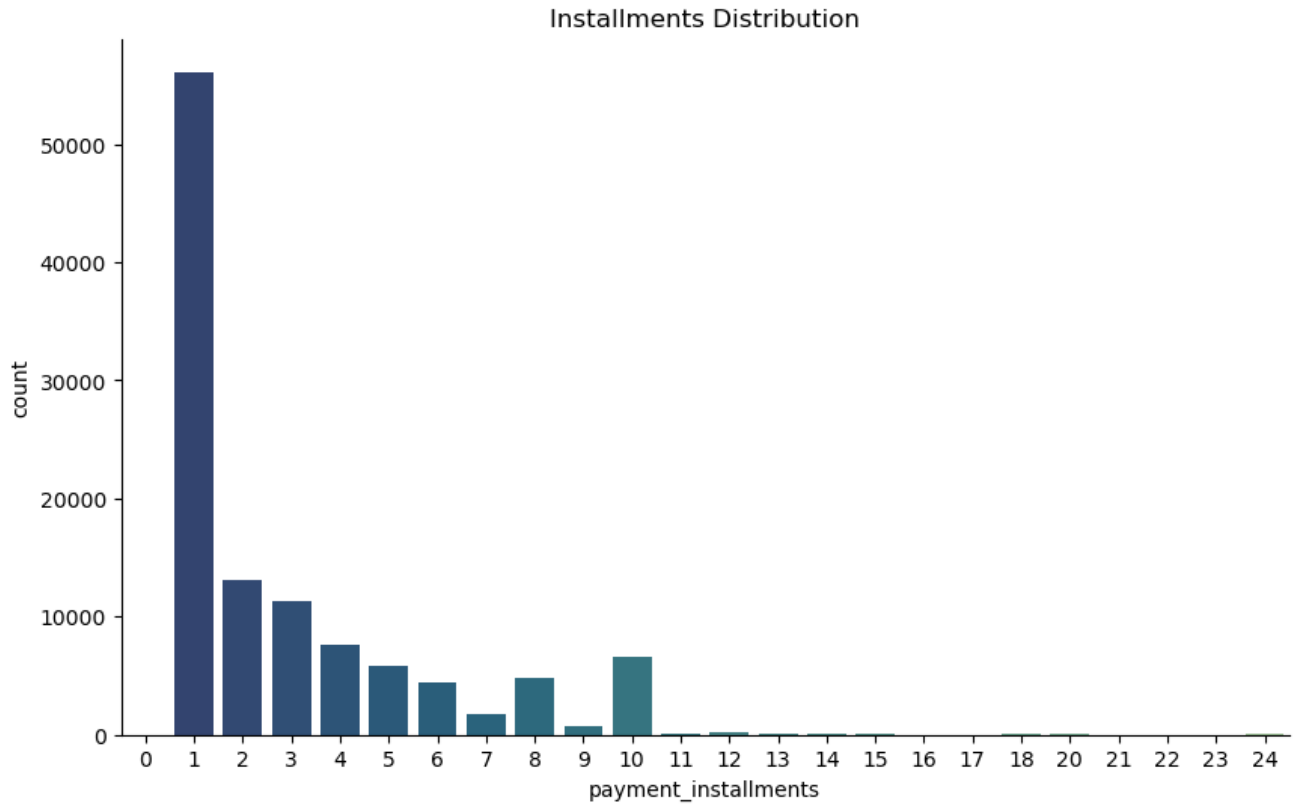
	customer_id	customer_unique_id	customer_zi
28825	48ebb06cf56dba9d009230cc751bb195	9925e1d7dff0d807355599dee04830ab	
28826	48ebb06cf56dba9d009230cc751bb195	9925e1d7dff0d807355599dee04830ab	
96127	5e5794daaa13f73e2f1cdb4114529843	f54cea27c80dc09bfe07b1cf1e01b845	

3 rows × 46 columns

Since no of installments can't be 0, we should drop these raws

```
# Drop indices
df.drop([29113, 29114, 96733], inplace=True)

# Reset Index
df.reset_index(inplace= True, drop= True)
```

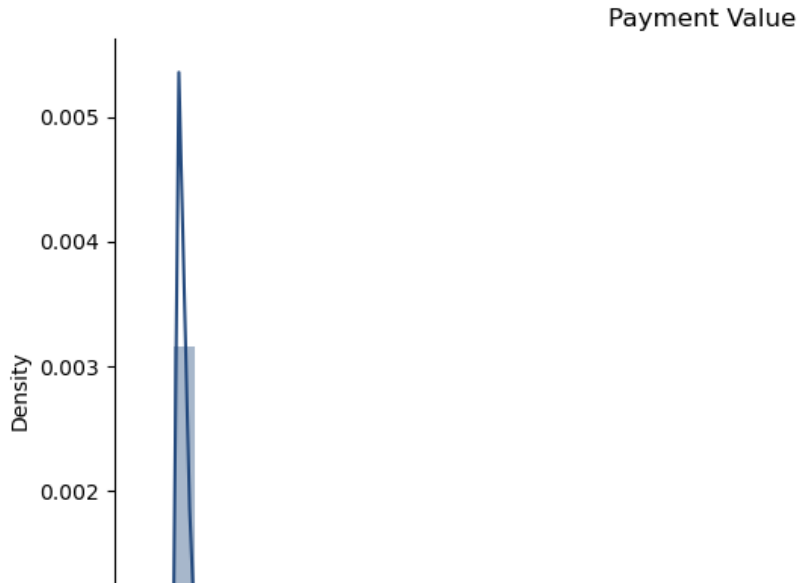
```
plt.figure(figsize=[10, 6])
sns.countplot(x = df.payment_installments, palette= 'crest_r')
plt.title('Installments Distribution')
sns.despine()
```



" Payment Value "

```
plt.figure(figsize=[10, 6])
sns.distplot(x = df.payment_value)
plt.title('Payment Value')
sns.despine()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

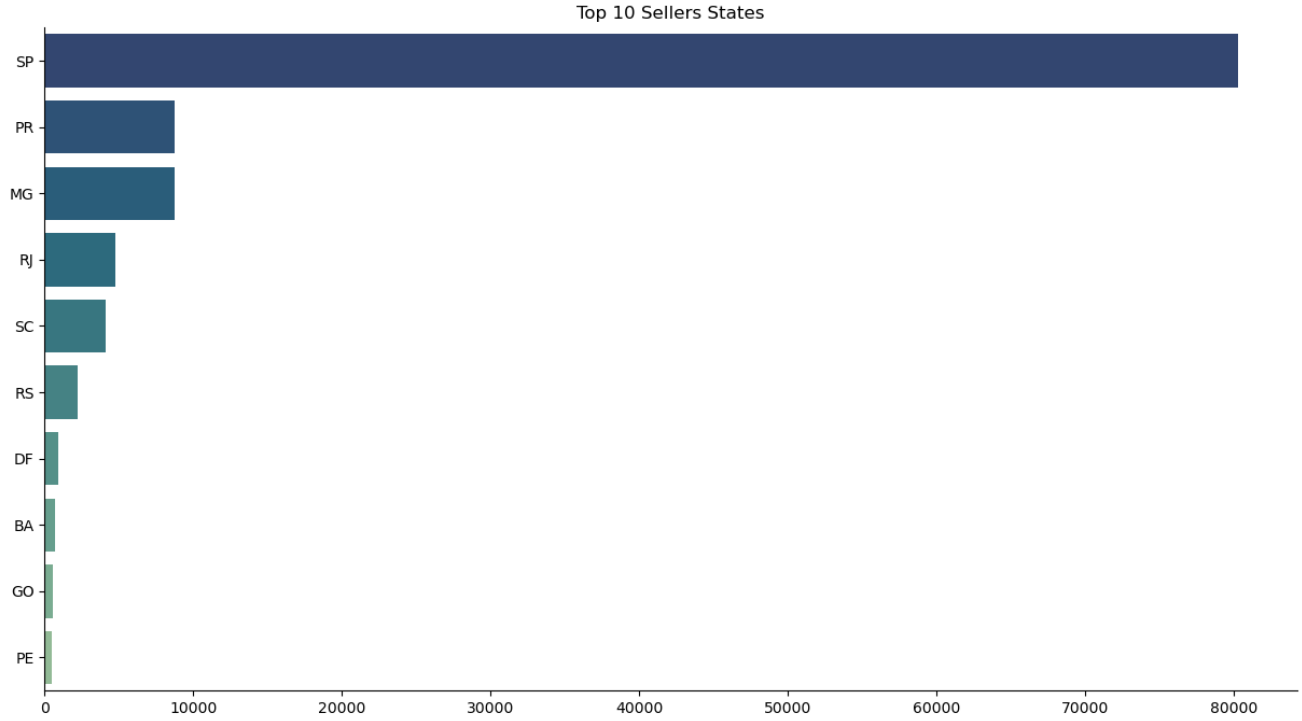


" Top 10 Cities for Sellers "

```
plt.figure(figsize=[10, 6])
sns.barplot(x = df.seller_city.value_counts().values[:10], y= df.seller_city.value_c
plt.title('Top 10 Sellers Cities')
sns.despine()
```

" Top 10 Sellers Capacity States "

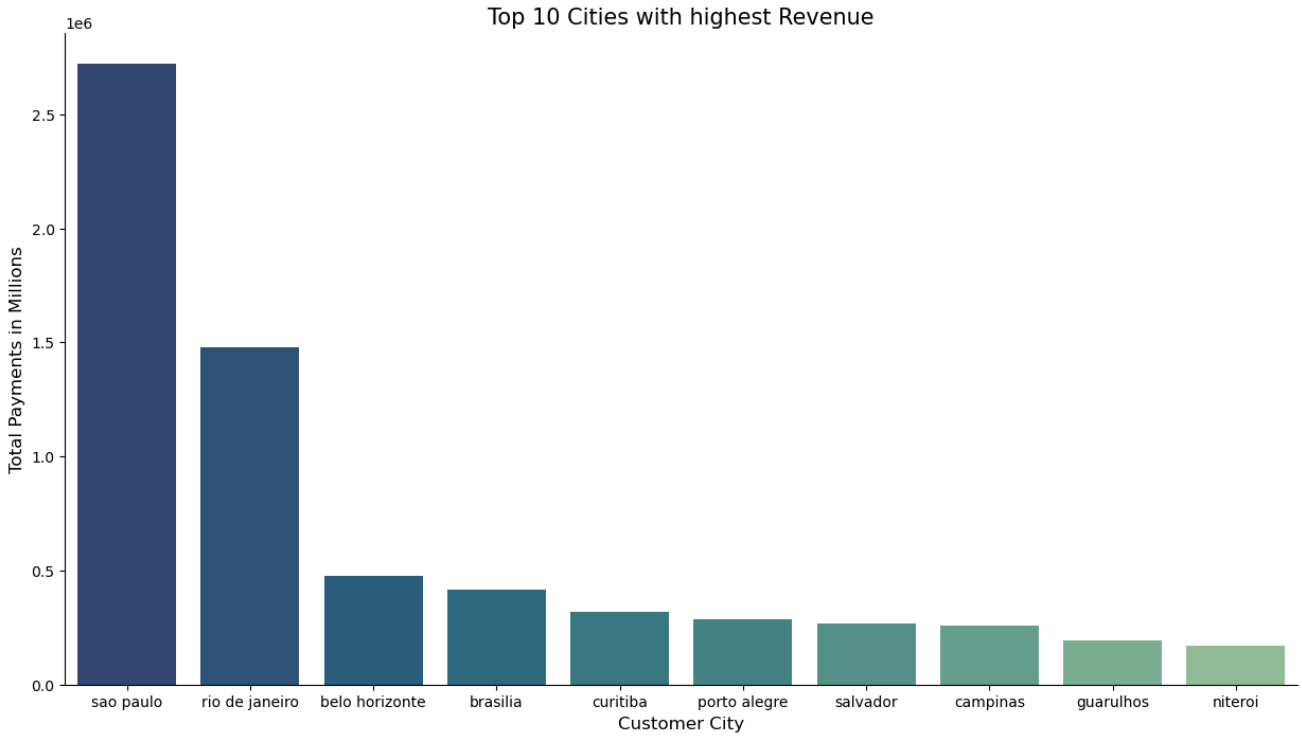
```
plt.figure(figsize=[15, 8])
sns.barplot(x = df.seller_state.value_counts().values[:10], y= df.seller_state.value
plt.title('Top 10 Sellers States')
sns.despine()
```



4.2 Multivariate Analysis

Which Cities have highest Revenue ?

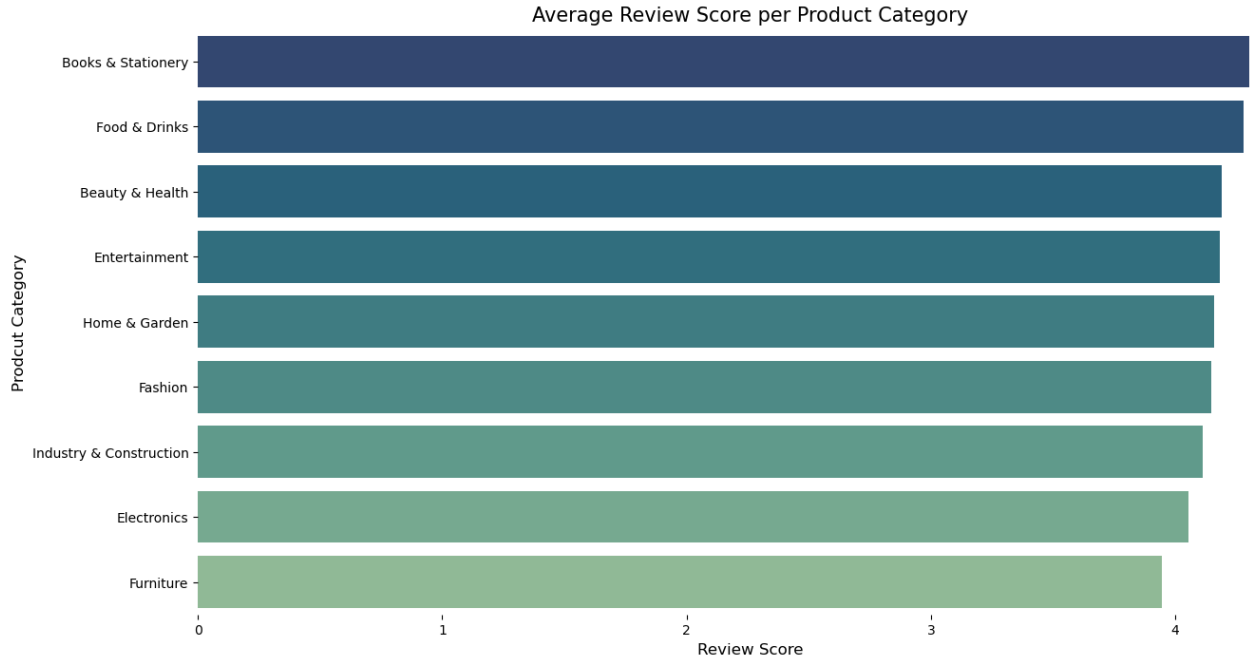
```
# Group customer city by payment value
revenue_per_city = df.groupby('customer_city')[['payment_value']].sum().sort_values(
revenue_per_city.reset_index(inplace=True)

# plot Top 10 cities with highest revenue
plt.figure(figsize=[15, 8])
sns.barplot(x = revenue_per_city.customer_city[:10], y= revenue_per_city.payment_valu
plt.title('Top 10 Cities with highest Revenue', fontsize= 15)
plt.xlabel('Customer City', fontsize= 12)
plt.ylabel('Total Payments in Millions',fontsize= 12)
sns.despine()
```



What is the average review score for each product category ?

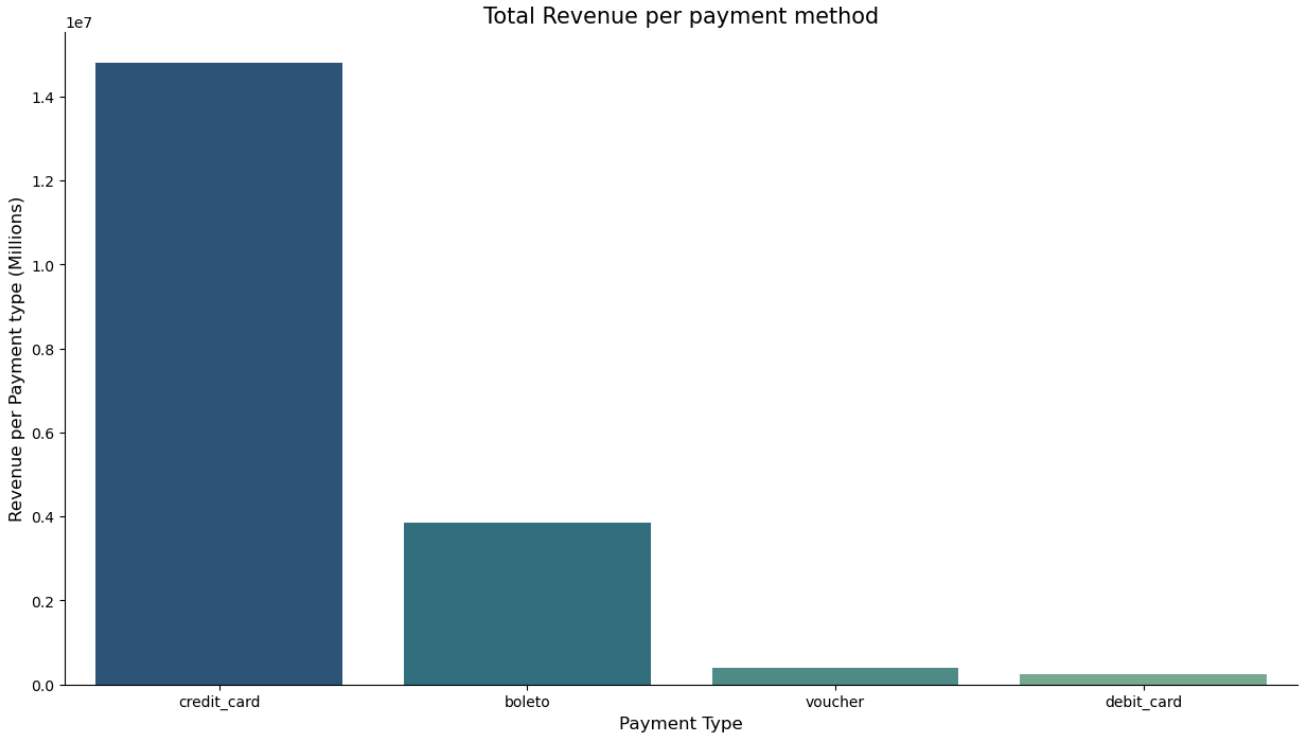
```
# Filter product category with 4.5 or above
review_per_cat = df.groupby('product_category')[['review_score']].mean().sort_values
review_per_cat.reset_index(inplace=True)

# Plot Product Category vs Review Score
plt.figure(figsize=[15, 8])
sns.barplot(x = review_per_cat.review_score, y= review_per_cat.product_category, pal
```

```
plt.title('Average Review Score per Product Category', fontsize= 15)
plt.xlabel('Review Score', fontsize=12)
plt.ylabel('Prodcut Category', fontsize= 12)
ax = plt.gca()
ax.set_frame_on(False);
```

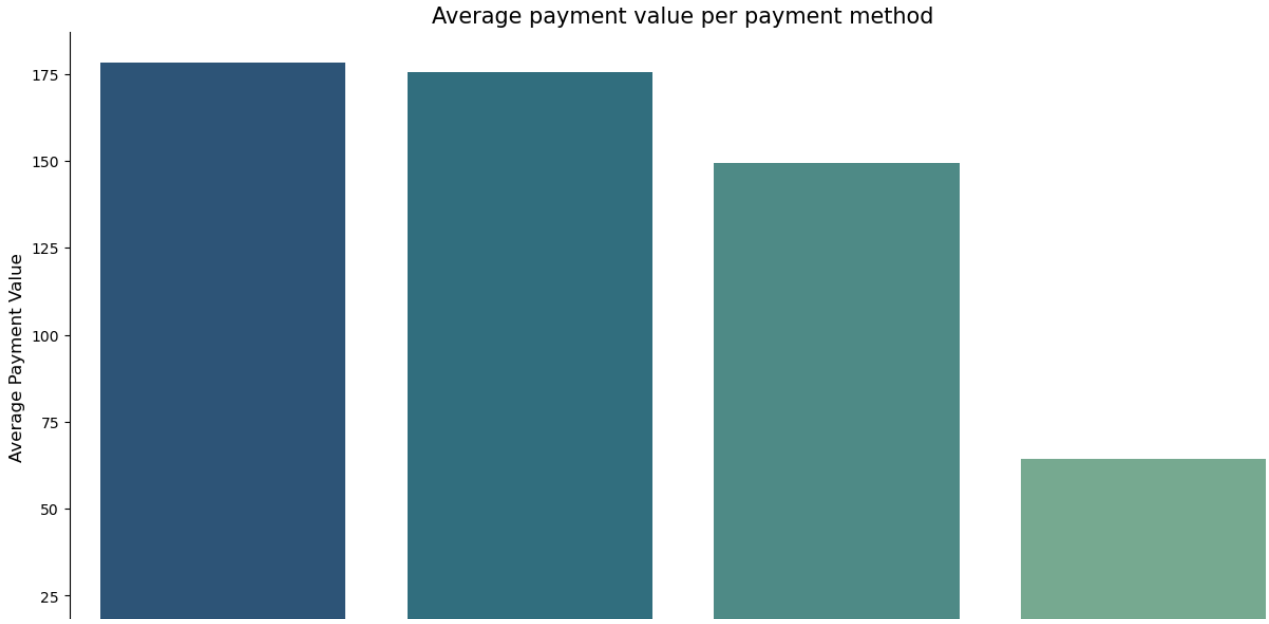


Are customers more likely to make larger payments using certain payment methods ?

```
# Group each payment type by average payment value
payment_methods = df.groupby('payment_type')[['payment_value']].sum().sort_values(by=
payment_methods.reset_index(inplace=True)

# plot Average payments per payment method
plt.figure(figsize=[15, 8])
sns.barplot(x = payment_methods.payment_type, y= payment_methods.payment_value, pale
plt.title('Total Revenue per payment method', fontsize= 15)
plt.xlabel('Payment Type', fontsize= 12)
plt.ylabel('Revenue per Payment type (Millions)', fontsize= 12)
sns.despine()
```




```
# Group each payment type by average payment value
payment_methods = df.groupby('payment_type')[['payment_value']].mean().sort_values(b
payment_methods.reset_index(inplace=True)

# plot Average payments per payment method
plt.figure(figsize=[15, 8])
sns.barplot(x = payment_methods.payment_type, y= payment_methods.payment_value, pale
plt.title('Average payment value per payment method', fontsize= 15)
plt.xlabel('Payment Type', fontsize= 12)
plt.ylabel('Average Payment Value', fontsize= 12)
sns.despine()
```

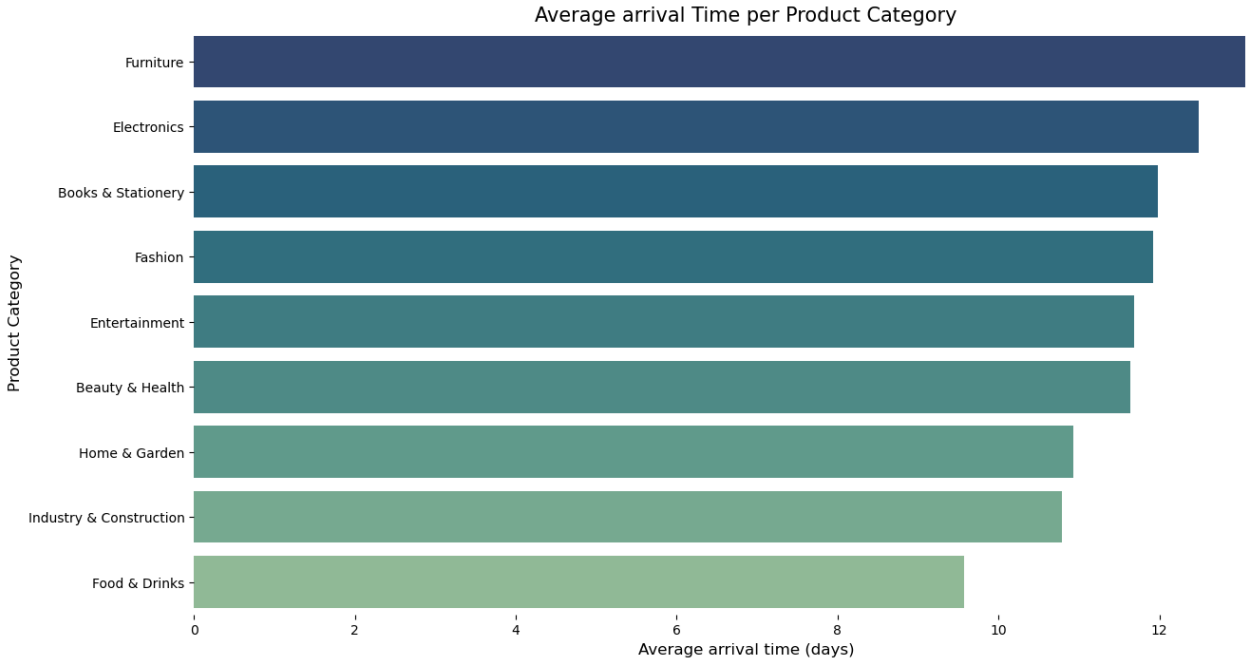


What is the average freight value for each product category ?

```
# Group product category by average freight value
freight_per_cat = df.groupby('product_category')[['freight_value']].mean().sort_valu
freight_per_cat.reset_index(inplace=True)

# plot average freight value per product category
plt.figure(figsize=[15, 8])
sns.barplot(x = freight_per_cat.freight_value, y=  freight_per_cat.product_category,
plt.title('Average Freight Value per Product Category', fontsize= 15)
plt.xlabel('Average Freight Value',fontsize= 12)
plt.ylabel('Product Category', fontsize= 12)
ax = plt.gca()
ax.set_frame_on(False);
```

What is the average shipping time for each product Category ?

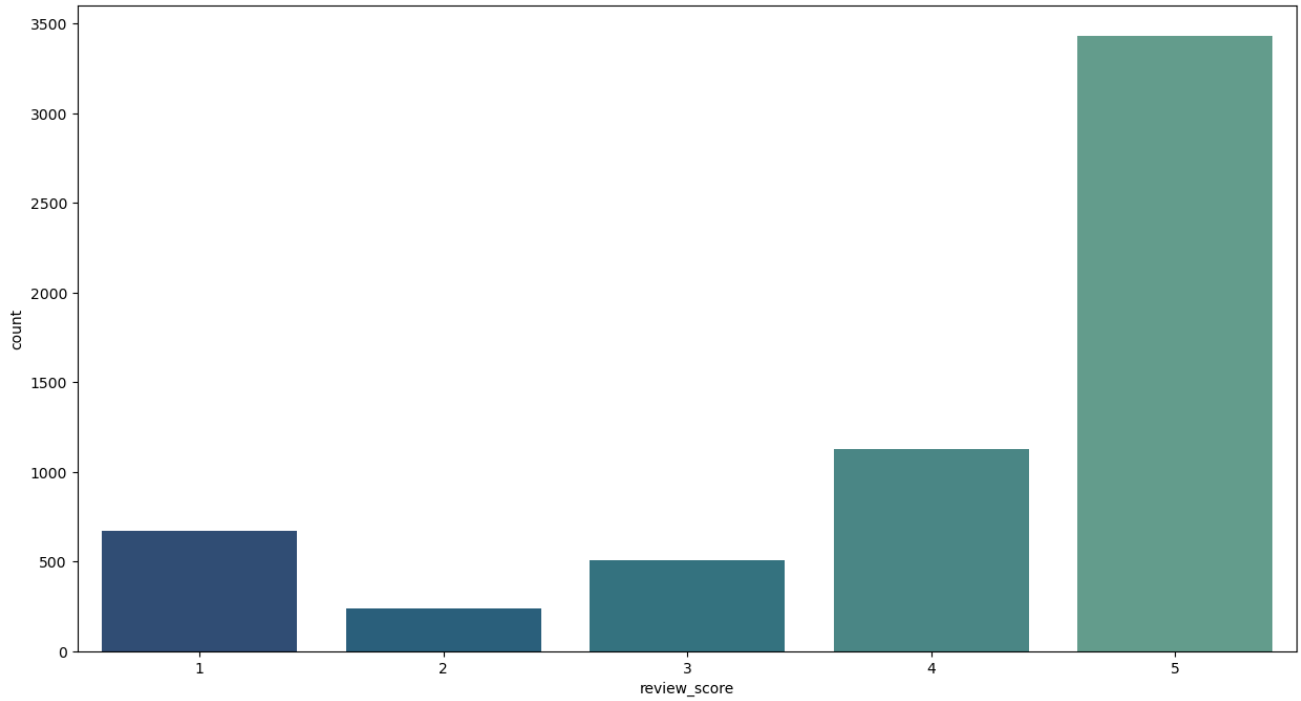
```
# Group product category by average arrival time
ship_per_cat = df.groupby('product_category')[['arrival_days']].mean().sort_values(by
ship_per_cat.reset_index(inplace=True)

# plot average freight value per product category
plt.figure(figsize=[15, 8])
sns.barplot(x = ship_per_cat.arrival_days, y=  ship_per_cat.product_category, palett
plt.title('Average arrival Time per Product Category', fontsize= 15)
plt.xlabel('Average arrival time (days)',fontsize= 12)
plt.ylabel('Product Category', fontsize= 12)
ax = plt.gca()
ax.set_frame_on(False);
```



Are Transactions done with Vouchers lead to high review score ?

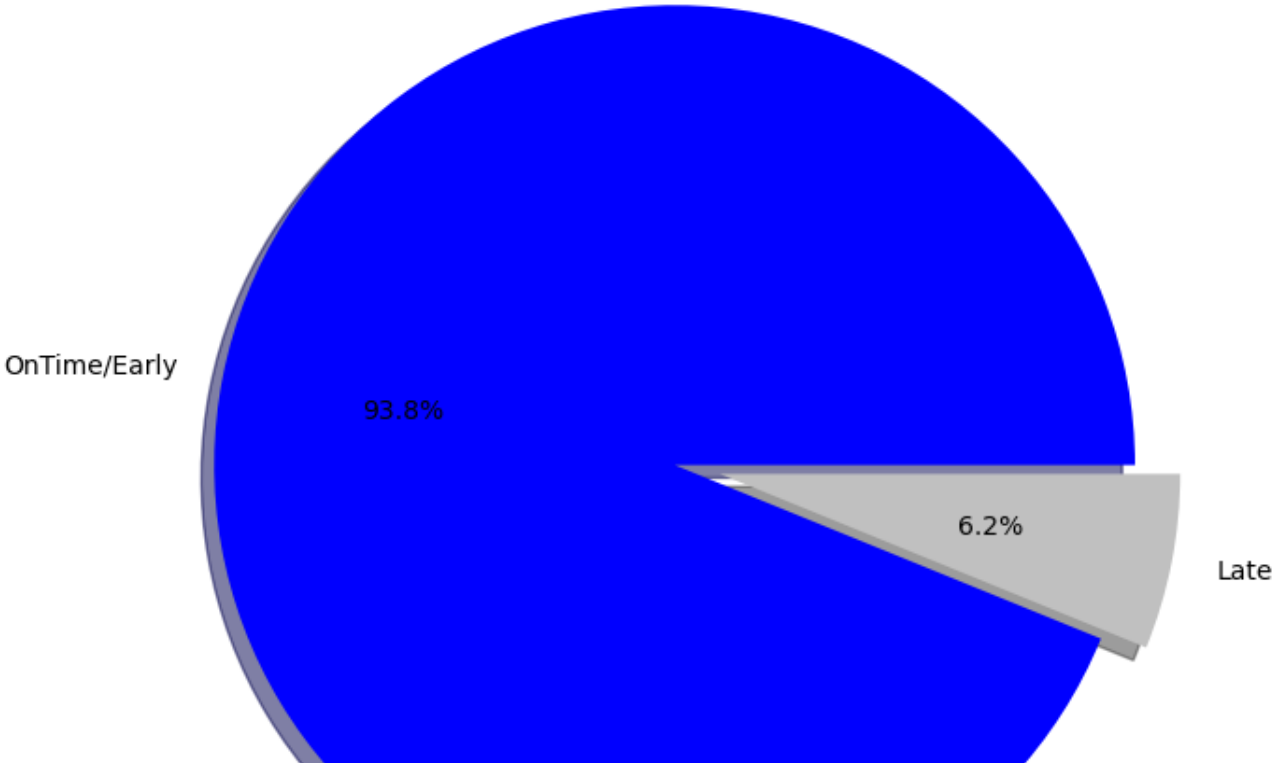
```
plt.figure(figsize=[15, 8])
voucher_trans = df[df.payment_type == 'voucher']
sns.countplot(x= voucher_trans.review_score)
```

```
<AxesSubplot:xlabel='review_score', ylabel='count'>
```



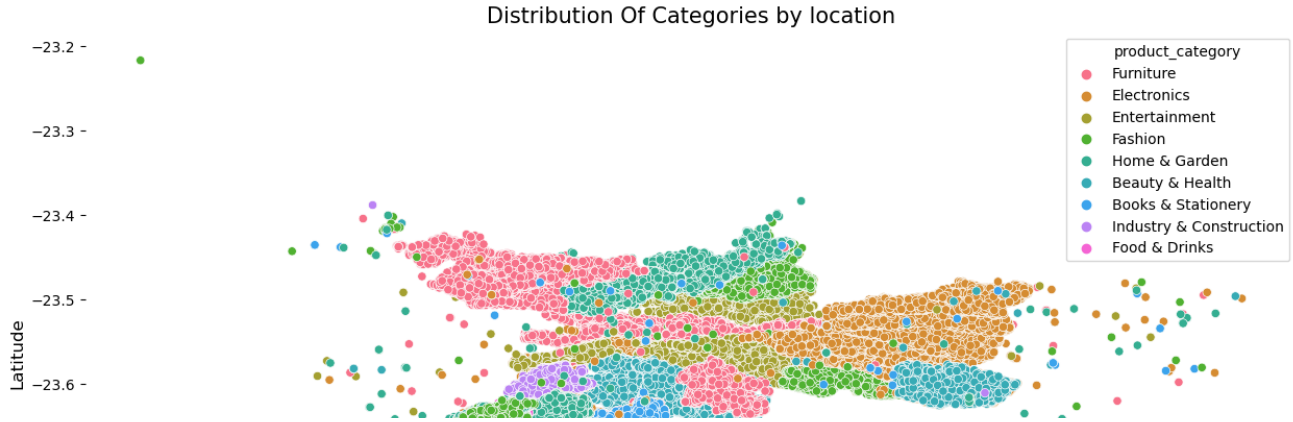
How accurate are the estimated delivery dates provided to customers ?

```
plt.figure(figsize=[30,8])
Values = df.arrival_status.value_counts().values
Labels = df.arrival_status.value_counts().index
plt.pie(Values, explode=(0.05, 0.05), labels= ['OnTime/Early', 'Late'], autopct='%1.
```



Distribution of products categories by location ?

```
plt.figure(figsize=[15, 8])
sns.scatterplot(x = geolocation_df.geolocation_lng, y = geolocation_df.geolocation_l
plt.title('Distribution Of Categories by location', fontsize= 15)
plt.xlabel('Longitude',fontsize= 12)
plt.ylabel('Latitude', fontsize= 12)
ax = plt.gca()
ax.set_frame_on(False);sns.despine()
```



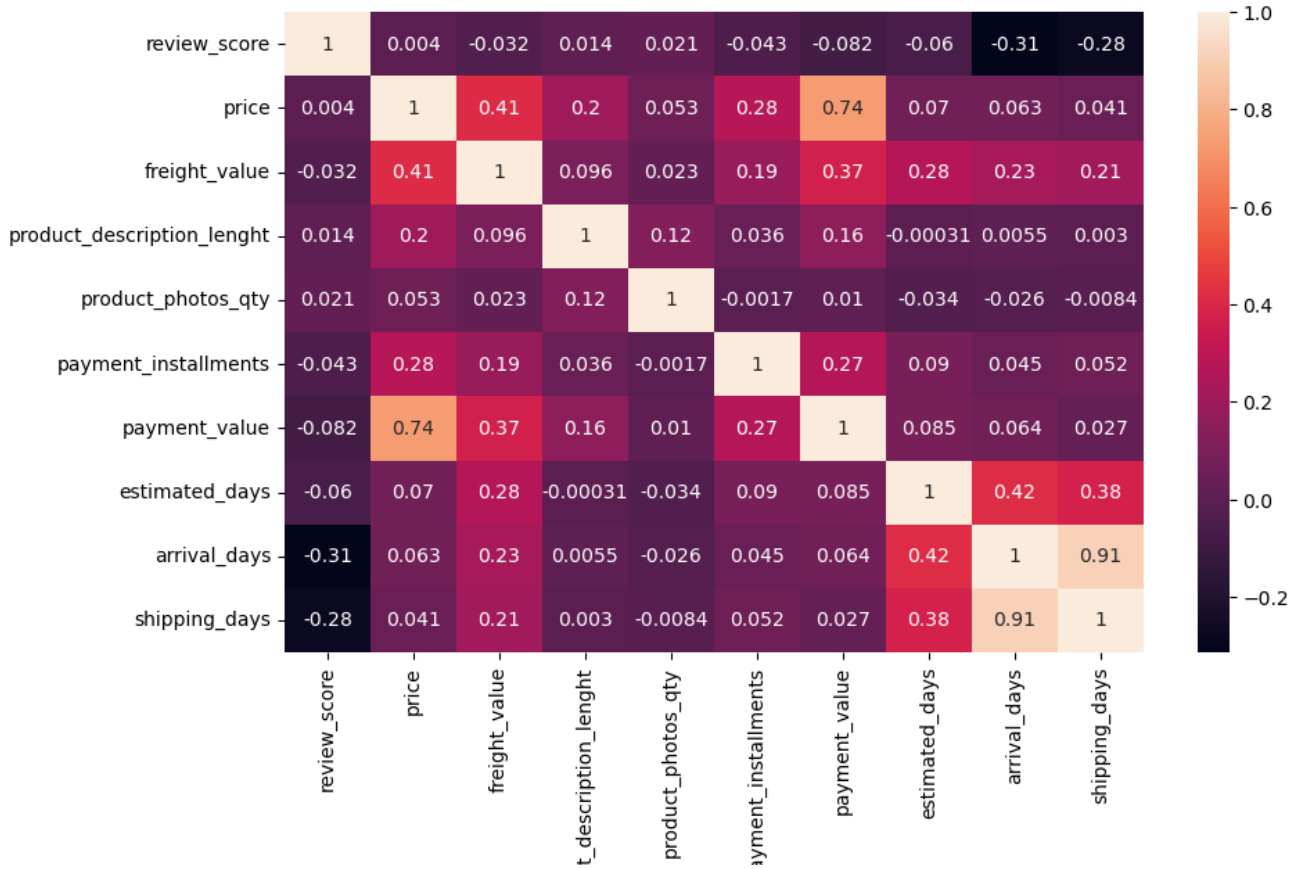

```
# Create copy of DataFrame
df_2 = df.copy()

# Save sample for EDA Deployment
EDA_df = df_2.drop(['customer_id', 'customer_unique_id', 'customer_zip_code_prefix',
EDA_sample = EDA_df.sample(frac= 1)[:10000]
EDA_sample.to_csv('EDA.csv')
```

5.0 Data Preprocessing

5.1 Drop Unneccessary Features

```
# Drop all ids, zip codes, datetimes, review comment and title, product length

df.drop(['customer_id', 'customer_unique_id', 'customer_zip_code_prefix', 'customer_
        'order_approved_at', 'order_delivered_carrier_date', 'order_delivered_custom
        'review_id', 'review_comment_title', 'review_comment_message', 'review_creat
        'order_item_id', 'product_id', 'seller_id', 'seller_zip_code_prefix', 'selle
        'product_category_name_english', 'product_category', 'product_weight_g', 'pr
        'product_vol_cm3'], axis= 1, inplace= True)


# Show Correlation between Features
plt.figure(figsize= [10, 6])
sns.heatmap(df.corr(), annot= True)
```

```
<AxesSubplot:>
```




```
# Remove features with high correlations
df.drop(['shipping_days', 'price'], axis= 1, inplace= True)
```

```
df.head()
```

	review_score	freight_value	product_description_lenght	product_photos_qty
0	4	21.88	1141.0	1.0
1	1	24.90	1141.0	1.0
2	1	24.90	1141.0	1.0
3	3	15.62	1141.0	1.0
4	4	30.59	1141.0	1.0

Convert Review Score from Multiclass to Binary

```
encoded_class = { 1 : 'Not Satisfied',
                  2 : 'Not Satisfied',
                  3 : 'Not Satisfied',
                  4 : 'Satisfied',
                  5 : 'Satisfied'}
```

```
df['review_score'] = df['review_score'].map(encoded_class)
```

Split Data into Input Features & Target Variable

```
X = df.drop('review_score', axis=1)
y = df['review_score']
```

5.2 Handling Categorical Features

Handling Ordinal Features (Label Encoding)

```
labels = {'Very Slow' : 1,
          'Slow' : 2,
          'Neutral' : 3,
          'Fast' : 4,
          'Very Fast' : 5}

X.estimated_delivery_rate = X.estimated_delivery_rate.map(labels)
X.shipping_delivery_rate = X.shipping_delivery_rate.map(labels)
X.arrival_delivery_rate = X.arrival_delivery_rate.map(labels)
```

Handling Nominal Features (One Hot Encoding)

```
X = pd.get_dummies(X, drop_first=True)
```

Split Data into Train & Test

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat
```

5.3 Feature Selection

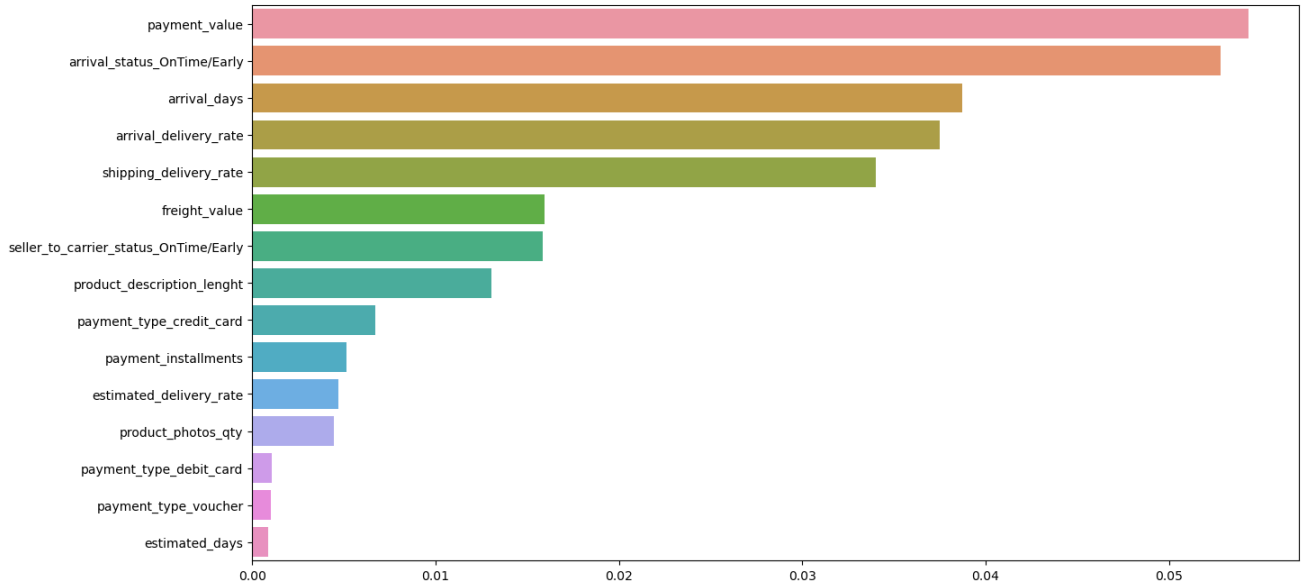
```
from sklearn.feature_selection import mutual_info_classif, SelectKBest
fs = SelectKBest(mutual_info_classif, k= 'all')
fs.fit(x_train, y_train)
x_train_fs = fs.transform(x_train)
x_test_fs = fs.transform(x_test)
```

Plotting Featres as per importance

```
# Get the indices sorted by most important to least important
plt.figure(figsize=[15, 8])
indices = np.argsort(fs.scores_)[::-1]
```

```
# To get your top 10 feature names
features = []
for i in range(15):
    features.append(fs.feature_names_in_[indices[i]])

# Now plot
sns.barplot(x = fs.scores_[indices[range(15)]], y = features)
```

```
<AxesSubplot:>
```



Select best 9 Features

```
from sklearn.feature_selection import mutual_info_classif, SelectKBest
fs = SelectKBest(mutual_info_classif, k= 9)
fs.fit(x_train, y_train)
x_train_fs = fs.transform(x_train)
x_test_fs = fs.transform(x_test)
```

```
x_train_fs = pd.DataFrame(x_train_fs, columns= fs.get_feature_names_out())
x_test_fs = pd.DataFrame(x_test_fs, columns= fs.get_feature_names_out())
```

5.4 Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler(with_mean= False)
```

```
x_train_scaled = sc.fit_transform(x_train_fs)
x_test_scaled = sc.transform(x_test_fs)
```

Convert Array to Dataframe

```
x_train_scaled = pd.DataFrame(x_train_scaled, columns= sc.get_feature_names_out())
x_test_scaled = pd.DataFrame(x_test_scaled, columns= sc.get_feature_names_out())
```

5.5 Handling Imbalance

Check imbalance percentage

```
round((y_train.value_counts() / y_train.shape[0]) * 100, 2)
```

```
Satisfied        76.99
Not Satisfied    23.01
Name: review_score, dtype: float64
```

Use SMOTE for handling imbalance

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state= 42)
x_train_resampled, y_train_resampled = smote.fit_resample(x_train_scaled, y_train)
```

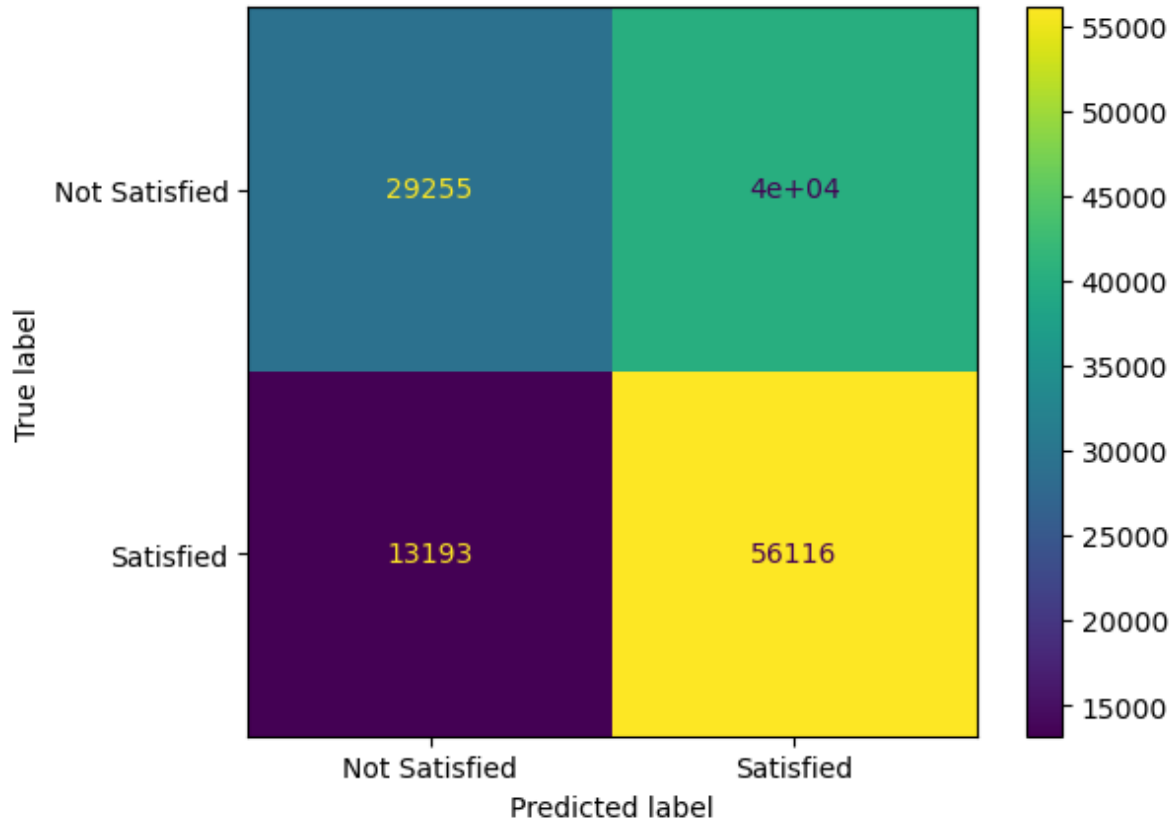
6.0 Modeling

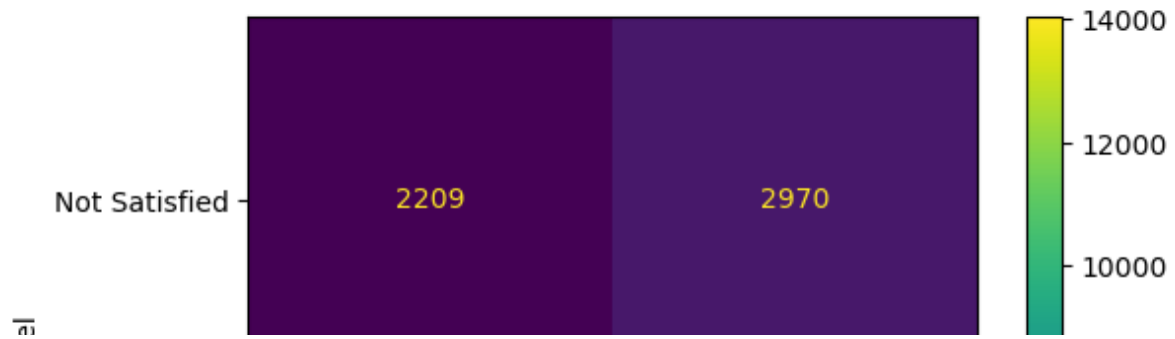
6.1 Apply ML Models

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, plot_confusion_matrix

lr = LogisticRegression()
lr.fit(x_train_resampled, y_train_resampled)

print('Evaluation on Training \n', classification_report(y_train_resampled, lr.predi
print('Evaluation on Testing \n', classification_report(y_test, lr.predict(x_test_sc

plot_confusion_matrix(lr, x_train_resampled, y_train_resampled)
plot_confusion_matrix(lr, x_test_scaled, y_test)
```

```
Evaluation on Training
                precision    recall  f1-score   support

Not Satisfied       0.69      0.42      0.52     69309
    Satisfied       0.58      0.81      0.68     69309

     accuracy                           0.62    138618
    macro avg       0.64      0.62      0.60    138618
 weighted avg       0.64      0.62      0.60    138618

Evaluation on Testing
                precision    recall  f1-score   support

Not Satisfied       0.40      0.43      0.41      5179
    Satisfied       0.83      0.81      0.82     17327

     accuracy                           0.72     22506
    macro avg       0.61      0.62      0.62     22506
 weighted avg       0.73      0.72      0.72     22506

/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7b3266807f50>
```

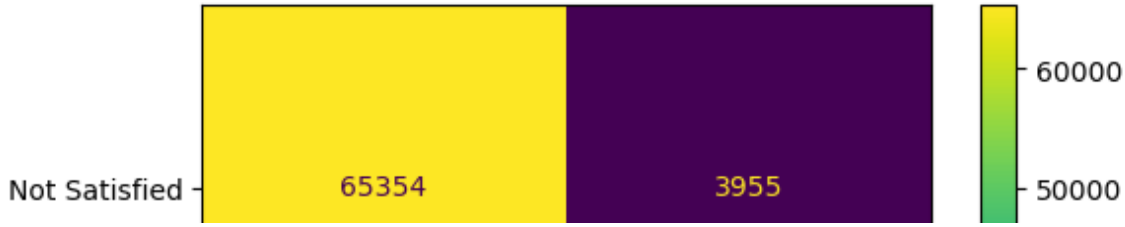




KNN Classifier

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(x_train_resampled, y_train_resampled)

print('Evaluation on Training \n', classification_report(y_train_resampled, knn.pred
print('Evaluation on Testing \n', classification_report(y_test, knn.predict(x_test_s

plot_confusion_matrix(knn,x_train_resampled,y_train_resampled)
plot_confusion_matrix(knn, x_test_scaled, y_test)
```

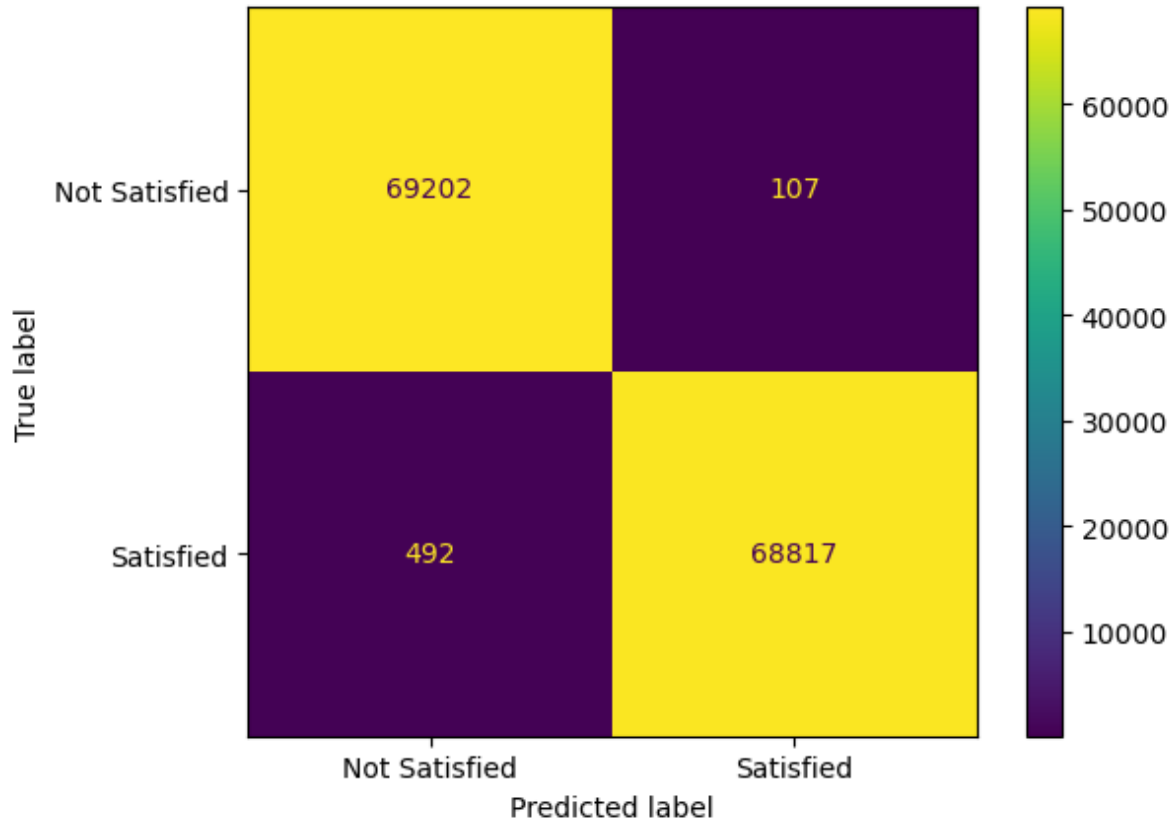
```
Evaluation on Training
                precision    recall  f1-score   support

Not Satisfied       0.82      0.94      0.88     69309
    Satisfied       0.93      0.79      0.86     69309

     accuracy                           0.87    138618
    macro avg       0.88      0.87      0.87    138618
 weighted avg       0.88      0.87      0.87    138618

Evaluation on Testing
                precision    recall  f1-score   support

Not Satisfied       0.37      0.59      0.46      5179
    Satisfied       0.85      0.70      0.77     17327

     accuracy                           0.67     22506
    macro avg       0.61      0.65      0.61     22506
 weighted avg       0.74      0.67      0.70     22506

/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7b32666324d0>
```

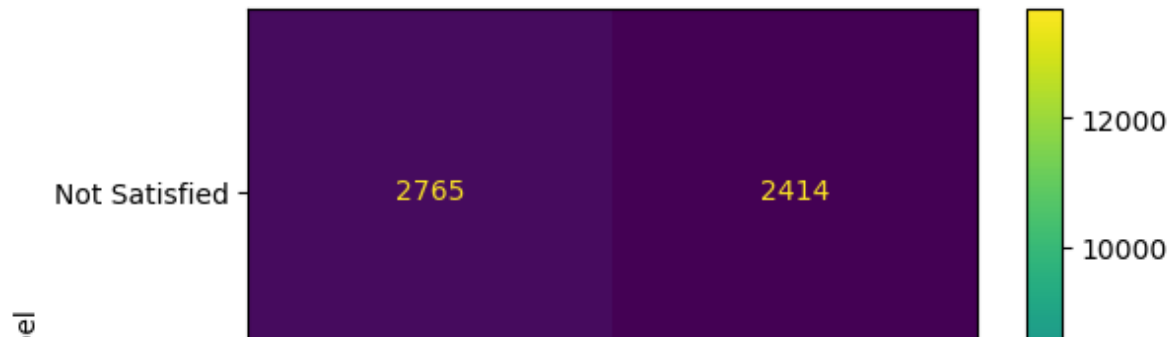


Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train_resampled, y_train_resampled)

print('Evaluation on Training \n', classification_report(y_train_resampled, dt.predi
print('Evaluation on Testing \n', classification_report(y_test, dt.predict(x_test_sc

plot_confusion_matrix(dt,x_train_resampled,y_train_resampled)
plot_confusion_matrix(dt, x_test_scaled, y_test)
```

```
Evaluation on Training
                precision    recall  f1-score   support

Not Satisfied       0.99      1.00      1.00     69309
    Satisfied       1.00      0.99      1.00     69309

     accuracy                           1.00    138618
    macro avg       1.00      1.00      1.00    138618
 weighted avg       1.00      1.00      1.00    138618

Evaluation on Testing
                precision    recall  f1-score   support

Not Satisfied       0.43      0.53      0.48      5179
    Satisfied       0.85      0.79      0.82     17327

     accuracy                           0.73     22506
    macro avg       0.64      0.66      0.65     22506
 weighted avg       0.75      0.73      0.74     22506

/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7b327a235850>
```

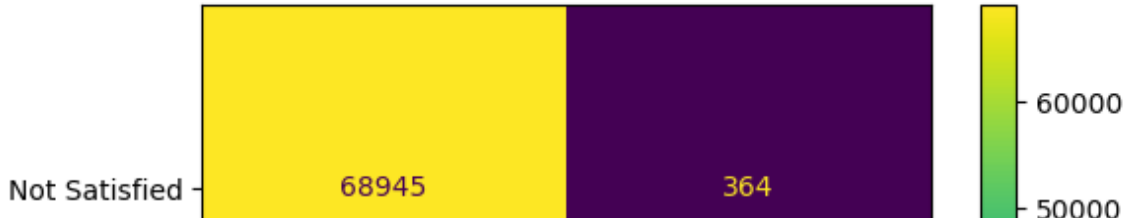




Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(x_train_resampled, y_train_resampled)

print('Evaluation on Training \n', classification_report(y_train_resampled, rf.predi
print('Evaluation on Testing \n', classification_report(y_test, rf.predict(x_test_sc

plot_confusion_matrix(rf, x_train_resampled, y_train_resampled)
plot_confusion_matrix(rf, x_test_scaled, y_test)
```

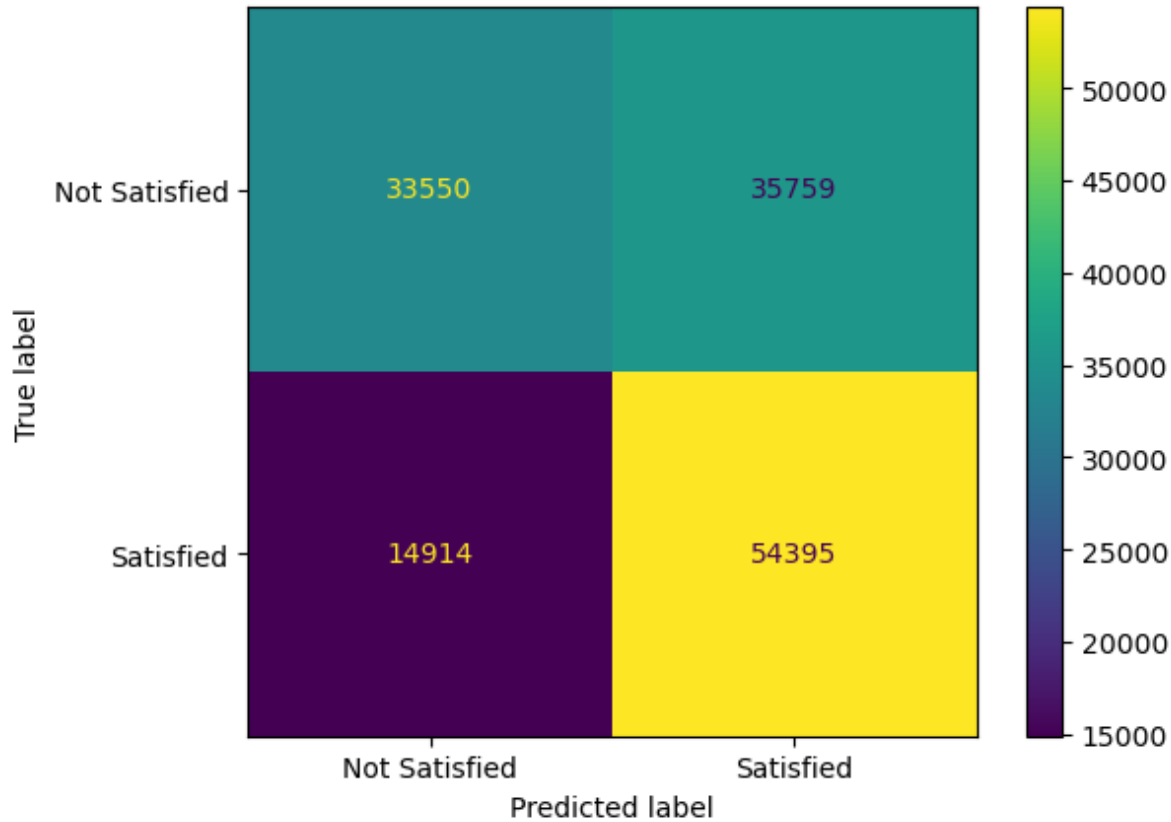
```
Evaluation on Training
                precision    recall  f1-score   support

Not Satisfied       1.00      0.99      1.00     69309
    Satisfied       0.99      1.00      1.00     69309

     accuracy                           1.00    138618
    macro avg       1.00      1.00      1.00    138618
 weighted avg       1.00      1.00      1.00    138618

Evaluation on Testing
                precision    recall  f1-score   support

Not Satisfied       0.58      0.50      0.54      5179
    Satisfied       0.86      0.89      0.87     17327

     accuracy                           0.80     22506
    macro avg       0.72      0.70      0.71     22506
 weighted avg       0.79      0.80      0.80     22506

/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7b32664a6a50>
```



Ada Boost

```
from sklearn.ensemble import AdaBoostClassifier

ad = AdaBoostClassifier()
ad.fit(x_train_resampled, y_train_resampled)

print('Evaluation on Training \n', classification_report(y_train_resampled, ad.predi
print('Evaluation on Testing \n', classification_report(y_test, ad.predict(x_test_sc

plot_confusion_matrix(ad, x_train_resampled, y_train_resampled)
plot_confusion_matrix(ad, x_test_scaled, y_test)
```

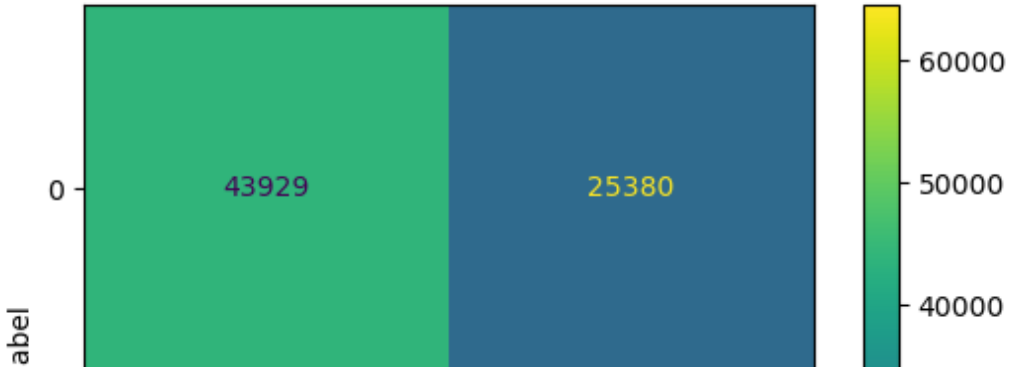
```
Evaluation on Training
                precision    recall  f1-score   support

Not Satisfied       0.69      0.48      0.57     69309
    Satisfied       0.60      0.78      0.68     69309

     accuracy                           0.63    138618
    macro avg       0.65      0.63      0.63    138618
 weighted avg       0.65      0.63      0.63    138618

Evaluation on Testing
                precision    recall  f1-score   support

Not Satisfied       0.39      0.45      0.42      5179
    Satisfied       0.83      0.79      0.81     17327

     accuracy                           0.71     22506
    macro avg       0.61      0.62      0.61     22506
 weighted avg       0.73      0.71      0.72     22506

/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7b32665bd190>
```



XGboost

```
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
y_train_xg = le.fit_transform(y_train_resampled)
y_test_xg = le.fit_transform(y_test)
xg = XGBClassifier()
xg.fit(x_train_resampled, y_train_xg)

print('Evaluation on Training \n', classification_report(y_train_xg, xg.predict(x_tr
print('Evaluation on Testing \n', classification_report(y_test_xg, xg.predict(x_test

plot_confusion_matrix(xg, x_train_resampled, y_train_xg)
plot_confusion_matrix(xg, x_test_scaled, y_test_xg)
```

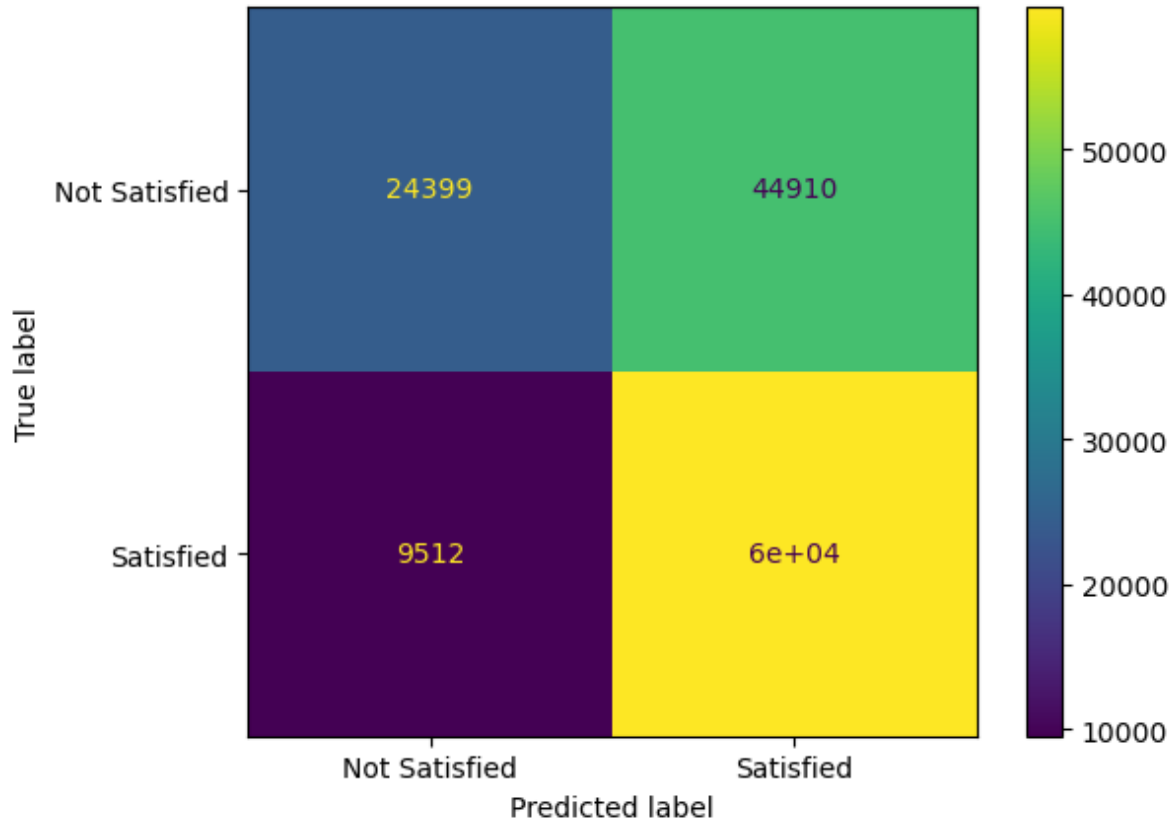
```
Evaluation on Training
              precision    recall  f1-score   support

           0       0.90      0.63      0.74     69309
           1       0.72      0.93      0.81     69309

    accuracy                           0.78    138618
   macro avg       0.81      0.78      0.78    138618
weighted avg       0.81      0.78      0.78    138618

Evaluation on Testing
              precision    recall  f1-score   support

           0       0.55      0.35      0.43      5179
           1       0.83      0.91      0.87     17327

    accuracy                           0.78     22506
   macro avg       0.69      0.63      0.65     22506
weighted avg       0.76      0.78      0.77     22506

/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7b326643c710>
```



Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, plot_confusion_matrix

nb = GaussianNB()
nb.fit(x_train_resampled, y_train_resampled)
y_pred = nb.predict(x_test_scaled)

print('Evaluation on Training \n', classification_report(y_train_resampled, nb.predi
print('Evaluation on Testing \n', classification_report(y_test, nb.predict(x_test_sc

plot_confusion_matrix(nb, x_train_resampled, y_train_resampled)
plot_confusion_matrix(nb, x_test_scaled, y_test)
```

```
Evaluation on Training
                precision    recall  f1-score   support

Not Satisfied       0.72      0.35      0.47     69309
    Satisfied       0.57      0.86      0.69     69309

     accuracy                           0.61    138618
    macro avg       0.65      0.61      0.58    138618
 weighted avg       0.65      0.61      0.58    138618

Evaluation on Testing
                precision    recall  f1-score   support

Not Satisfied       0.44      0.36      0.39      5179
    Satisfied       0.82      0.86      0.84     17327

     accuracy                           0.75     22506
    macro avg       0.63      0.61      0.62     22506
 weighted avg       0.73      0.75      0.74     22506

/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7b32667154d0>
```



LightGBM

```
import lightgbm as ltb

lg = ltb.LGBMClassifier()
lg.fit(x_train_resampled, y_train_resampled)

print('Evaluation on Training \n', classification_report(y_train_resampled, lg.predi
print('Evaluation on Testing \n', classification_report(y_test, lg.predict(x_test_sc

plot_confusion_matrix(lg, x_train_resampled, y_train_resampled)
plot_confusion_matrix(lg, x_test_scaled, y_test)
```

```
Evaluation on Training
                precision    recall  f1-score   support

Not Satisfied       0.88      0.56      0.68     69309
    Satisfied       0.68      0.92      0.78     69309

     accuracy                           0.74    138618
    macro avg       0.78      0.74      0.73    138618
 weighted avg       0.78      0.74      0.73    138618

Evaluation on Testing
                precision    recall  f1-score   support

Not Satisfied       0.55      0.34      0.42      5179
    Satisfied       0.82      0.92      0.87     17327

     accuracy                           0.79     22506
    macro avg       0.69      0.63      0.64     22506
 weighted avg       0.76      0.79      0.76     22506

/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7b32324a6210>
```


```
! pip install catboost
```

```
Requirement already satisfied: catboost in /opt/conda/lib/python3.7/site-packag
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packages (f
Requirement already satisfied: numpy>=1.16.0 in /opt/conda/lib/python3.7/site-p
Requirement already satisfied: pandas>=0.24.0 in /opt/conda/lib/python3.7/site-
Requirement already satisfied: plotly in /opt/conda/lib/python3.7/site-packages
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.7/site-pack
Requirement already satisfied: graphviz in /opt/conda/lib/python3.7/site-packag
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-pa
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-p
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.7/si
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.7/site
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.7/sit
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-pa
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.7/si
Requirement already satisfied: tenacity>=6.2.0 in /opt/conda/lib/python3.7/site
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.7/si
WARNING: Running pip as the 'root' user can result in broken permissions and co
```

CatBoost

```
import catboost as cb

cb = cb.CatBoostClassifier()
cb.fit(x_train_resampled, y_train_resampled)

print('Evaluation on Training \n', classification_report(y_train_resampled, cb.predi
print('Evaluation on Testing \n', classification_report(y_test, cb.predict(x_test_sc

plot_confusion_matrix(cb, x_train_resampled, y_train_resampled)
plot_confusion_matrix(cb, x_test_scaled, y_test)
```

```
Learning rate set to 0.084624
0:      learn: 0.6823809        total: 77ms     remaining: 1m 16s
1:      learn: 0.6727305        total: 100ms    remaining: 50.1s
2:      learn: 0.6640268        total: 120ms    remaining: 39.8s
3:      learn: 0.6579464        total: 137ms    remaining: 34.2s
4:      learn: 0.6520828        total: 156ms    remaining: 31s
5:      learn: 0.6473676        total: 174ms    remaining: 28.9s
6:      learn: 0.6420148        total: 195ms    remaining: 27.6s
7:      learn: 0.6391014        total: 213ms    remaining: 26.4s
8:      learn: 0.6360239        total: 230ms    remaining: 25.3s
9:      learn: 0.6320128        total: 248ms    remaining: 24.6s
10:     learn: 0.6264585        total: 269ms    remaining: 24.2s
11:     learn: 0.6244286        total: 286ms    remaining: 23.5s
12:     learn: 0.6218838        total: 304ms    remaining: 23.1s
13:     learn: 0.6200140        total: 323ms    remaining: 22.7s
14:     learn: 0.6170580        total: 342ms    remaining: 22.5s
15:     learn: 0.6151424        total: 361ms    remaining: 22.2s
16:     learn: 0.6125775        total: 378ms    remaining: 21.8s
17:     learn: 0.6108710        total: 398ms    remaining: 21.7s
18:     learn: 0.6091404        total: 416ms    remaining: 21.5s
19:     learn: 0.6070248        total: 433ms    remaining: 21.2s
20:     learn: 0.6049733        total: 450ms    remaining: 21s
21:     learn: 0.6022157        total: 467ms    remaining: 20.8s
22:     learn: 0.6008552        total: 487ms    remaining: 20.7s
23:     learn: 0.5996607        total: 505ms    remaining: 20.5s
24:     learn: 0.5992381        total: 523ms    remaining: 20.4s
25:     learn: 0.5975544        total: 541ms    remaining: 20.3s
26:     learn: 0.5948252        total: 563ms    remaining: 20.3s
27:     learn: 0.5934518        total: 580ms    remaining: 20.1s
28:     learn: 0.5920970        total: 600ms    remaining: 20.1s
29:     learn: 0.5913211        total: 620ms    remaining: 20s
30:     learn: 0.5905422        total: 636ms    remaining: 19.9s
31:     learn: 0.5898947        total: 652ms    remaining: 19.7s
32:     learn: 0.5887101        total: 672ms    remaining: 19.7s
33:     learn: 0.5872717        total: 687ms    remaining: 19.5s
34:     learn: 0.5834536        total: 706ms    remaining: 19.5s
35:     learn: 0.5831164        total: 726ms    remaining: 19.4s
36:     learn: 0.5827740        total: 743ms    remaining: 19.3s
37:     learn: 0.5816730        total: 760ms    remaining: 19.2s
38:     learn: 0.5802421        total: 776ms    remaining: 19.1s
39:     learn: 0.5799252        total: 793ms    remaining: 19s
40:     learn: 0.5790851        total: 811ms    remaining: 19s
41:     learn: 0.5785281        total: 828ms    remaining: 18.9s
42:     learn: 0.5774336        total: 845ms    remaining: 18.8s
43:     learn: 0.5766158        total: 860ms    remaining: 18.7s
44:     learn: 0.5763857        total: 876ms    remaining: 18.6s
45:     learn: 0.5760014        total: 893ms    remaining: 18.5s
46:     learn: 0.5745487        total: 910ms    remaining: 18.4s
47:     learn: 0.5732138        total: 927ms    remaining: 18.4s
48:     learn: 0.5730449        total: 943ms    remaining: 18.3s
49:     learn: 0.5726410        total: 960ms    remaining: 18.2s
50:     learn: 0.5723817        total: 977ms    remaining: 18.2s
51:     learn: 0.5714139        total: 994ms    remaining: 18.1s
52:     learn: 0.5712144        total: 1.01s    remaining: 18s
53:     learn: 0.5710309        total: 1.03s    remaining: 18s
54:     learn: 0.5705703        total: 1.04s    remaining: 17.9s
55:     learn: 0.5697256        total: 1.06s    remaining: 17.9s
56:     learn: 0.5690120        total: 1.08s    remaining: 17.9s
57:     learn: 0.5688257        total: 1.1s     remaining: 17.8s
58:     learn: 0.5686540        total: 1.11s    remaining: 17.8s
59:     learn: 0.5674617        total: 1.13s    remaining: 17.7s
```

```
60:     learn: 0.5668822        total: 1.15s    remaining: 17.7s
```

6.2 Hyperparameter Tuning

```
64:     learn: 0.5639877        total: 1.21s    remaining: 17.5s
```

XGboost

```
#from sklearn.model_selection import GridSearchCV

#param_grid = {
    #'learning_rate': [0.1, 0.2],
    #'max_depth': [5, 7, 8],
    #'n_estimators': [100, 200]
    #}
#grid_search = GridSearchCV(xg, param_grid= param_grid, cv= 5, scoring= 'f1_macro')
#grid_search.fit(x_train_resampled, y_train_xg)
```

```
79:     learn: 0.5572834        total: 1.49s    remaining: 17.1s
```

```
final_xg_model = XGBClassifier(learning_rate= 0.2, max_depth= 8, n_estimators= 200)
final_xg_model.fit(x_train_resampled, y_train_xg)

print('Evaluation on Training \n', classification_report(y_train_xg, final_xg_model.p
print('Evaluation on Testing \n', classification_report(y_test_xg, final_xg_model.pr
```

```
Evaluation on Training
               precision    recall  f1-score   support

           0       0.95      0.73      0.82     69309
           1       0.78      0.96      0.86     69309

    accuracy                           0.84    138618
   macro avg       0.86      0.84      0.84    138618
weighted avg       0.86      0.84      0.84    138618

Evaluation on Testing
               precision    recall  f1-score   support

           0       0.60      0.38      0.46      5179
           1       0.83      0.93      0.88     17327

    accuracy                           0.80     22506
   macro avg       0.72      0.65      0.67     22506
weighted avg       0.78      0.80      0.78     22506
```

```
#param_grid = {
    #'learning_rate': [0.1, 0.2],
    #'depth': [5, 7, 8],
    #'iterations': [100, 200]}

#grid_search = GridSearchCV(cb, param_grid= param_grid, cv= 5, scoring= 'f1_macro')
#grid_search.fit(x_train_resampled, y_train_resampled)
#grid_search.best_params_
```

```
118:    learn: 0.5431756        total: 2.19s    remaining: 16.2s
```

```
import catboost as cb
final_cb_model = cb.CatBoostClassifier(depth= 7, iterations= 200, learning_rate= 0.2
final_cb_model.fit(x_train_resampled,y_train_resampled)
print('Evaluation on Training \n', classification_report(y_train_resampled, final_cb
print('Evaluation on Training \n', classification_report(y_test, final_cb_model.pred
```

```
0:      learn: 0.6689060        total: 24.6ms   remaining: 4.9s
1:      learn: 0.6508960        total: 46.3ms   remaining: 4.58s
2:      learn: 0.6360345        total: 69ms     remaining: 4.53s
3:      learn: 0.6238844        total: 91.6ms   remaining: 4.49s
4:      learn: 0.6165656        total: 112ms    remaining: 4.38s
5:      learn: 0.6091390        total: 135ms    remaining: 4.37s
6:      learn: 0.6055839        total: 155ms    remaining: 4.27s
7:      learn: 0.6025524        total: 174ms    remaining: 4.19s
8:      learn: 0.6001775        total: 196ms    remaining: 4.17s
9:      learn: 0.5949882        total: 219ms    remaining: 4.17s
10:     learn: 0.5907425        total: 242ms    remaining: 4.16s
11:     learn: 0.5874446        total: 263ms    remaining: 4.12s
12:     learn: 0.5864624        total: 282ms    remaining: 4.05s
13:     learn: 0.5826798        total: 299ms    remaining: 3.98s
14:     learn: 0.5797200        total: 320ms    remaining: 3.95s
15:     learn: 0.5783667        total: 338ms    remaining: 3.89s
16:     learn: 0.5772234        total: 356ms    remaining: 3.83s
17:     learn: 0.5766517        total: 375ms    remaining: 3.79s
18:     learn: 0.5748467        total: 394ms    remaining: 3.76s
19:     learn: 0.5741698        total: 413ms    remaining: 3.72s
20:     learn: 0.5731762        total: 430ms    remaining: 3.67s
21:     learn: 0.5719358        total: 454ms    remaining: 3.67s
22:     learn: 0.5695675        total: 474ms    remaining: 3.64s
23:     learn: 0.5691448        total: 493ms    remaining: 3.62s
24:     learn: 0.5633970        total: 514ms    remaining: 3.6s
25:     learn: 0.5614536        total: 535ms    remaining: 3.58s
26:     learn: 0.5600030        total: 557ms    remaining: 3.56s
27:     learn: 0.5595214        total: 576ms    remaining: 3.54s
28:     learn: 0.5574094        total: 595ms    remaining: 3.51s
29:     learn: 0.5568555        total: 615ms    remaining: 3.48s
30:     learn: 0.5555017        total: 632ms    remaining: 3.45s
31:     learn: 0.5550853        total: 654ms    remaining: 3.43s
32:     learn: 0.5548129        total: 672ms    remaining: 3.4s
33:     learn: 0.5544462        total: 692ms    remaining: 3.38s
34:     learn: 0.5504791        total: 709ms    remaining: 3.34s
35:     learn: 0.5500205        total: 730ms    remaining: 3.33s
36:     learn: 0.5495187        total: 749ms    remaining: 3.3s
37:     learn: 0.5485829        total: 766ms    remaining: 3.27s
38:     learn: 0.5478082        total: 787ms    remaining: 3.25s
39:     learn: 0.5466473        total: 805ms    remaining: 3.22s
40:     learn: 0.5441445        total: 828ms    remaining: 3.21s
41:     learn: 0.5427713        total: 847ms    remaining: 3.19s
42:     learn: 0.5423966        total: 869ms    remaining: 3.17s
43:     learn: 0.5413004        total: 891ms    remaining: 3.16s
44:     learn: 0.5403157        total: 911ms    remaining: 3.14s
45:     learn: 0.5394461        total: 930ms    remaining: 3.11s
46:     learn: 0.5391319        total: 949ms    remaining: 3.09s
47:     learn: 0.5385517        total: 969ms    remaining: 3.07s
48:     learn: 0.5382132        total: 993ms    remaining: 3.06s
49:     learn: 0.5378639        total: 1.01s    remaining: 3.03s
50:     learn: 0.5362751        total: 1.03s    remaining: 3.01s
51:     learn: 0.5359378        total: 1.05s    remaining: 2.98s
52:     learn: 0.5349032        total: 1.07s    remaining: 2.97s
53:     learn: 0.5346606        total: 1.09s    remaining: 2.95s
```

```
54:     learn: 0.5341754        total: 1.11s    remaining: 2.92s
55:     learn: 0.5336243        total: 1.13s    remaining: 2.9s
56:     learn: 0.5322062        total: 1.15s    remaining: 2.87s
186:    learn: 0.5254282        total: 3.39s    remaining: 14.7s
```

RandomForest

```
#param_grid = {
    #'max_depth': [8, 9, 10],
    #'n_estimators': [100, 200]
    #}
#grid_search = GridSearchCV(rf, param_grid= param_grid, cv= 5, scoring= 'f1_macro')
#grid_search.fit(x_train_resampled, y_train_resampled)
#grid_search.best_params_
```

```
199:    learn: 0.5223614        total: 3.63s    remaining: 14.5s
```

```
final_rf_model = RandomForestClassifier(n_estimators= 200, max_depth= 10)
final_rf_model.fit(x_train_resampled,y_train_resampled)
print('Evaluation on Training \n', classification_report(y_train_resampled, final_rf_
print('Evaluation on Training \n', classification_report(y_test, final_rf_model.pred
```

```
Evaluation on Training
                precision    recall  f1-score   support

Not Satisfied       0.79      0.53      0.64     69309
    Satisfied       0.65      0.86      0.74     69309

     accuracy                           0.70    138618
    macro avg       0.72      0.70      0.69    138618
 weighted avg       0.72      0.70      0.69    138618

Evaluation on Training
                precision    recall  f1-score   support

Not Satisfied       0.45      0.42      0.43      5179
    Satisfied       0.83      0.85      0.84     17327

     accuracy                           0.75     22506
    macro avg       0.64      0.63      0.64     22506
 weighted avg       0.74      0.75      0.75     22506
```

```
226:    learn: 0.5170239        total: 4.14s    remaining: 14.1s
```

Select XGboost for pipeline as it provides highest performance on testing

```
230:    learn: 0.5161241        total: 4.21s    remaining: 14s
```

7.0 Pipeline

```
225:    learn: 0.5153824        total: 4.2s     remaining: 13.9s
```

```
df_pipeline = df.copy()
df_pipeline.head()
```

	review_score	freight_value	product_description_lenght	product_photos_qty
0	Satisfied	21.88	1141.0	1.0
1	Not Satisfied	24.90	1141.0	1.0
2	Not Satisfied	24.90	1141.0	1.0

252: Not Satisfied 0.5129219 15.62 total: 4.62s remaining: 13.6s

Encoding Review score to 0 and 1

255: learn: 0.5125978 total: 4.67s remaining: 13.6s

```
encoded_class = { 'Not Satisfied' : 0,
                  'Satisfied' : 1,
                }

df_pipeline['review_score'] = df_pipeline['review_score'].map(encoded_class)
```

263: learn: 0.5114383 total: 4.81s remaining: 13.4s

Split Input Features and Targe Variable

267: learn: 0.5110785 total: 4.89s remaining: 13.4s

```
X = df_pipeline.drop('review_score', axis=1)
y = df_pipeline['review_score']
```

271: learn: 0.5102804 total: 4.96s remaining: 13.3s

Split into Train & Test

275: learn: 0.5099621 total: 5.04s remaining: 13.3s

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_sta
```

277: learn: 0.5096183 total: 5.07s remaining: 13.2s

Prepare Numerical Features

281: learn: 0.5090404 total: 5.14s remaining: 13.1s

```
numeric_columns = x_train.select_dtypes(exclude = 'object').columns
numeric_columns
```

```
Index(['freight_value', 'product_description_lenght', 'product_photos_qty',
       'payment_installments', 'payment_value', 'estimated_days',
       'arrival_days'],
      dtype='object')
```

290: learn: 0.5076169 total: 5.3s remaining: 12.9s

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

numerical_pipeline = Pipeline(steps=[('Handle Missing Values', SimpleImputer(strateg
                                      ('Feature Scaling', StandardScaler(with_mean=Fal
```

299: learn: 0.5064378 total: 5.46s remaining: 12.7s

Prepare Categorical Features

303: learn: 0.5061087 total: 5.53s remaining: 12.7s

```
cat_columns = x_train.select_dtypes(include = 'object').columns
cat_columns
```

```
Index(['payment_type', 'seller_to_carrier_status', 'arrival_status',
       'estimated_delivery_rate', 'arrival_delivery_rate',
       'shipping_delivery_rate'],
      dtype='object')
```

```
313:    learn: 0.5049484        total: 5.71s    remaining: 12.5s
```

```
cat_pipeline = Pipeline(steps=[('Handle Missing Values', SimpleImputer(strategy= 'mos
                               ('OneHot Encoding', OneHotEncoder(drop= 'first')),
                               ('Feature Scaling', StandardScaler(with_mean= False)
```

```
318:    learn: 0.5045412        total: 5.79s    remaining: 12.4s
```

```
from sklearn.compose import ColumnTransformer

preprocessing = ColumnTransformer(transformers=[('Numerical Columns', numerical_pipe
                                                 ('Cat Columns', cat_pipeline, cat_co
preprocessing
```

```
ColumnTransformer(remainder='passthrough',
                  transformers=[('Numerical Columns',
                                 Pipeline(steps=[('Handle Missing Values',

SimpleImputer(strategy='median')),
                                                 ('Feature Scaling',

StandardScaler(with_mean=False))]),
                                 Index(['freight_value',
'product_description_lenght', 'product_photos_qty',
       'payment_installments', 'payment_value', 'estimated_days',
       'arrival_days'],
      dty...
                                ('Cat Columns',
                                 Pipeline(steps=[('Handle Missing Values',

SimpleImputer(strategy='most_frequent')),
                                                 ('OneHot Encoding',

OneHotEncoder(drop='first')),
                                                 ('Feature Scaling',

StandardScaler(with_mean=False))]),
                                 Index(['payment_type',
'seller_to_carrier_status', 'arrival_status',
       'estimated_delivery_rate', 'arrival_delivery_rate',
       'shipping_delivery_rate'],
      dtype='object'))])
```

```
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE

final_pipeline = Pipeline(steps=[('Preprocessing', preprocessing), ('Smote', SMOTE()
                               ('Model', XGBClassifier(learning_rate= 0.2, max_depth
final_pipeline
```

```
Pipeline(steps=[('Preprocessing',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('Numerical Columns',
```

```
                                                  Pipeline(steps=[('Handle '
                                                                   'Missing '
                                                                   'Values',

      SimpleImputer(strategy='median')),
                                                                  ('Feature '
                                                                   'Scaling',

      StandardScaler(with_mean=False))]),
                                                  Index(['freight_value',
      'product_description_lenght', 'product_photos_qty',
             'payment_installments', 'payment_valu...
                                  gamma=None, gpu_id=None, grow_policy=None,
                                  importance_type=None,
                                  interaction_constraints=None,
      learning_rate=0.2,
                                  max_bin=None, max_cat_to_onehot=None,
                                  max_delta_step=None, max_depth=8,
                                  max_leaves=None, min_child_weight=None,
                                  missing=nan, monotone_constraints=None,
                                  n_estimators=200, n_jobs=None,
                                  num_parallel_tree=None, predictor=None,
                                  random_state=None, reg_alpha=None,
                                  reg_lambda=None, ...))])
      391:    learn: 0.4965553        total: 7.08s    remaining: 11s
```

```
final_pipeline.fit(x_train, y_train)
```

```
      Pipeline(steps=[('Preprocessing',
                       ColumnTransformer(remainder='passthrough',
                                         transformers=[('Numerical Columns',
                                                        Pipeline(steps=[('Handle '
                                                                         'Missing '
                                                                         'Values',

      SimpleImputer(strategy='median')),
                                                                        ('Feature '
                                                                         'Scaling',

      StandardScaler(with_mean=False))]),
                                                        Index(['freight_value',
      'product_description_lenght', 'product_photos_qty',
             'payment_installments', 'payment_valu...
                                  enable_categorical=False, eval_metric=None,
                                  gamma=0, gpu_id=-1, grow_policy='depthwise',
                                  importance_type=None,
      interaction_constraints='',
                                  learning_rate=0.2, max_bin=256,
                                  max_cat_to_onehot=4, max_delta_step=0,
                                  max_depth=8, max_leaves=0, min_child_weight=1,
                                  missing=nan, monotone_constraints='()',
                                  n_estimators=200, n_jobs=0,
      num_parallel_tree=1,
                                  predictor='auto', random_state=0, reg_alpha=0,
                                  reg_lambda=1, ...))])
      422:    learn: 0.4936537        total: 7.63s    remaining: 10.4s
```

```
import joblib
joblib.dump(final_pipeline, 'Brazilian Ecommerce Classification.bkl')
```

```
      ['Brazilian Ecommerce Classification.bkl']
```

8.0 NLP For Customer Satisfaction

```
reviews_df.head()
```

	review_id	order_id	review_score
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5

```
# Remove 'review_comment_title' because of high missing values perentage and remove
reviews_df = reviews_df[['review_comment_message', 'review_score']]

reviews_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99224 entries, 0 to 99223
Data columns (total 2 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   review_comment_message  40977 non-null  object
 1   review_score            99224 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 1.5+ MB
```

```
# Drop missing values
reviews_df.dropna(inplace= True)

# Rename columns for ease
reviews_df.rename(columns = {'review_comment_message' : 'comment', 'review_score' :

# Reset index
reviews_df.reset_index(inplace= True, drop= True)
```

```
# Encode scores to be Satisfied or Not Satisfied
encoded_class = { 1 : 'Not Satisfied',
                  2 : 'Not Satisfied',
                  3 : 'Not Satisfied',
                  4 : 'Satisfied',
                  5 : 'Satisfied'}

reviews_df['score'] = reviews_df['score'].map(encoded_class)
```

Text Cleaning & Processing

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

stemmer = PorterStemmer()
corpus = []

for i in range(reviews_df.shape[0]):
    # Remove any special characters or number
    comment = re.sub('[^a-zA-Z]', ' ', reviews_df.comment[i])
    # Lower text
    comment = comment.lower()
    # Remove any spaces before or after text
    comment = comment.strip()
    # Split text for stemming
    comment = comment.split()
    # Stemming words in Portugues
    comment = [stemmer.stem(word) for word in comment if word not in set(stopwords.w
    # Merge stemmed words to be sentences
    comment = ' '.join(comment)

    corpus.append(comment)

corpus
```

```
['recebi bem ant prazo estipulado',
 'parab ns loja lannist adorei comprar internet seguro pr tico parab ns todo
feliz p scoa',
 'aparelho eficient site marca aparelho impresso desinfector chegar outro
nome atualizar marca correta vez aparelho',
 'pouco travando valor ta boa',
 'vendedor confi vel produto ok entrega ant prazo',
 'gostaria saber sempr recebi compra agora decpcion',
 'p ssimo',
 'loja nota',
 'obrigado aten amim dispensada',
 'compra realizada facilment entrega efetuada ant prazo dado produto j come
usado at present problema',
 'rel gio bonito barato',
 'n gostei comprei gato lebr',
 'sempr compro internet entrega ocorr ant prazo combinado acredito prazo m
ximo stark prazo m ximo j esgot ainda n recebi produto',
 'recebi exatament esperava demai encomenda outro vendedor atrasaram chegou
prazo',
 'recomendo',
 'boa',
 't completament apaixonada loja super respon vel confi vel',
 'nada chegar pedido',
 'bom cheiroso',
 'otimo vendedor chegou ate ant prazo adorei produto',
 'processo compra tranquilo eficient',
 'tomara q dure poi pelinho',
 'recebi soment control midea split estilo faltou control remoto ar
```

```
 'condicionado consul',
 'boa',
 'mt lindo',
 'ocorreu tudo contratado sendo entrega realizada ant prazo satisfeita',
 'amei achei lindo delicado adorei',
 'tima loja parceria r pid ssima produto bem embalado qualidad s custo frete meio azedo',
 'recomendo vendedor',
 'produto n chegou prazo estipulado causou transtorno pq programei viagem f ria filho baseado prazo moro bahia cuiab sozinho agora casa est vazia',
 'produto entregu solicitado muita brevidad parab ns',
 'comprei dua unidad s recebi agora fa',
 'produto bom m veio mim n condiz foto an ncio',
 'produto inferior mal acabado',
 'kit mochila patrulha canina lindo netinho vai amar obrigada',
 'maravilha',
 'entrega prazo',
 'cumpriu prometido compra entrega tempo satisfat rio',
 'pedi reembolso resposta at momento',
 'super r pido',
 'satisfeito',
 'estao paraben sempr chega muiiita antecedencia obrigada',
 'ok recomendo',
 'produto chegou pc n conseguiu reconhec porta usb',
 'produto boa qualidad chegou ant prazo prometido',
 'pedido bald pe bloco montar un r cada n entregu vendido entregu targaryen tapet eva n letra pe crian un r entreg',
 'entrega r pida parab ns',
 'comprei tre pacot cinco folha cada papel transfer tecido escuro so recebi doi']
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Use TFIDF Vectorizer to convert text into numbers
tf = TfidfVectorizer()
df_new = tf.fit_transform(corpus).toarray()
```

```
df_new = pd.DataFrame(df_new, columns= tf.get_feature_names_out())
df_new
```

	aa	aaa	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	aaguarda
0	0.0	0.0	0.0	
1	0.0	0.0	0.0	

61?: learn: 0.4790666 total: 11s remaining: 6.83s

Split into Input Features & Target Variable

621: learn: 0.4787506 total: 11.1s remaining: 6.76s

```
X = df_new
y = reviews_df['score']
```

Split data into Train & Test

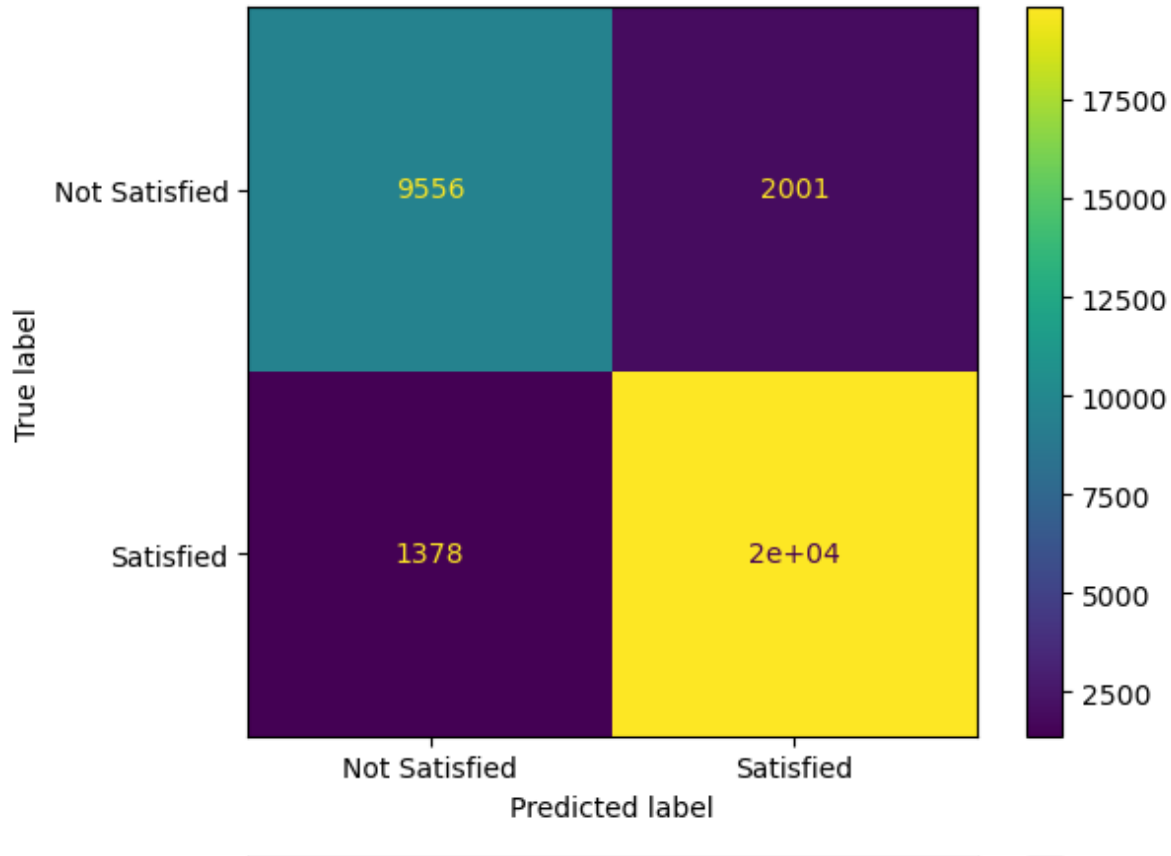
620: learn: 0.4781695 total: 11.2s remaining: 6.69s

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_sta
```

40976 0.0 0.0 0.0

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, plot_confusion_matrix

nb = MultinomialNB()
nb.fit(x_train, y_train)
y_pred = nb.predict(x_test)

print('Evaluation on Training \n', classification_report(y_train, nb.predict(x_train
print('Evaluation on Testing \n', classification_report(y_test, nb.predict(x_test)))

plot_confusion_matrix(nb, x_train, y_train)
plot_confusion_matrix(nb, x_test, y_test)
```

```
Evaluation on Training
                precision    recall  f1-score   support

Not Satisfied       0.87      0.83      0.85     11557
    Satisfied       0.91      0.94      0.92     21224

     accuracy                           0.90     32781
    macro avg       0.89      0.88      0.89     32781
 weighted avg       0.90      0.90      0.90     32781

Evaluation on Testing
                precision    recall  f1-score   support

Not Satisfied       0.85      0.80      0.82      2890
    Satisfied       0.89      0.92      0.91      5306

     accuracy                           0.88      8196
    macro avg       0.87      0.86      0.87      8196
 weighted avg       0.88      0.88      0.88      8196

/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7b32647f28d0>
```



Now we have a reliable model with better results for classifying customers based on reviews, Next we should investigate further reasons for non-satisfaction.

Check Reasons for Non-Satisfaction

```
# Create a separate DataFrame to check reasons of non satisfaction
non_satisfied = reviews_df[reviews_df.score == 'Not Satisfied']

# Reset index
non_satisfied.reset_index(inplace= True, drop= True)
```

Text Cleaning & Processing

```
stemmer = PorterStemmer()
corpus = []

for i in range(non_satisfied.shape[0]):
    comment = re.sub('[^a-zA-Z]', ' ', non_satisfied.comment[i])
    comment = comment.lower()
    comment = comment.strip()
    comment = comment.split()
    comment = [stemmer.stem(word) for word in comment if word not in set(stopwords.w
    comment = ' '.join(comment)

    corpus.append(comment)

corpus
```

```
['gostaria saber sempr recebi compra agora decpcion',
 'p ssimo',
 'n gostei comprei gato lebr',
 'sempr compro internet entrega ocorr ant prazo combinado acredito prazo m
ximo stark prazo m ximo j esgot ainda n recebi produto',
 'nada chegar pedido',
 'recebi soment control midea split estilo faltou control remoto ar
condicionado consul',
 'produto n chegou prazo estipulado causou transtorno pq programei viagem f
ria filho baseado prazo moro bahia cuiab sozinho agora casa est vazia',
 'comprei dua unidad s recebi agora fa',
 'produto bom m veio mim n condiz foto an ncio',
 'produto inferior mal acabado',
 'entrega prazo',
 'pedi reembolso resposta at momento',
 'produto chegou pc n conseguiu reconhec porta usb',
 'pedido bald pe bloco montar un r cada n entregu vendido entregu targaryen
tapet eva n letra pe crian un r entreg',
 'comprei tre pacot cinco folha cada papel transfer tecido escuro so recebi
doi',
 'demor pra entrega',
 'entrega dividida dua n comunicado loja cheguei pensar s haviam enviado part
produto',
 'n funciona n faz sincronismo',
 'gostei',
 'chegou apena pe nota garantia constam dua joia',
 'n recebi produto consta sistema recebi al m pagar caro frete',
 'comprei rel gio unissex enviaram rel gio feminino bem menor especifica es
an ncio',
 'pe n serviu',
 'comprei produto dia fevereiro hoje dia marco n entregu resid ncia n sei
correio dess brasil p ssimo pr pria loja demor postar',
 'fiz compra faz dia n recebi ainda produto precisa melhorar entrega',
 'comprei doi lustr pendent parceira targaryen s enviaram lustr abri reclama
```

```
 'ainda n recebi resposta aguardo solu part loja lannist',
 'faltou produto recebi veio quebrado',
 'aqui est descrevendo entregu s ate agora n recebi',
 'comecei usar agora',
 'comprar produto correto capa interno outro',
 'entrega super rapida',
 'fiz pedido garrafa azeit chegaram dia outra quas hora contato loja dificil
tanto via mail qto telef suspens termin l n compro',
 'n posso aguardando chegada produto comprei logo acont opini formada',
 'comprei doi produto recebi soment mandei doi email loja n nenhum retorno
insatisfeita',
 'n recomendaria loja pretendo voltar comprar nela mercadoria entregu ap s
prazo quas m s ap s compra site lannist costumam chegar ant prazo estranhei',
 'demora entrega detestei atendimento nunca compro baratheon vendia mesmo bem
melhor',
 'produto n chegou segundo rastreio est parado ribeir preto sp',
 'produto deu defeito vez uso',
 'at agora n recebi produto',
 'loja anuncia produto entrega outro',
 'apesar ter sido entregu rapidament substitu ram caixa meteoro caixa born
tentei reclamar lannist sistema estaav ar guitarra acess rio correto',
 'produto entregu uma al problema est faltando pino fixa',
 'marcado dia entrega pedido rastreamento marcava produto j havia sido
entregu data realidad n entregu produto s entregu dia',
820:    learn: 0.4662641        total: 14.7s    remaining: 3.2s
```

Translate sample of non-satisfied comments for comprehension

```
824:    learn: 0.4660485        total: 14.7s    remaining: 3.13s
```

```
# First install deep_translator library
! pip install deep_translator
```

```
Collecting deep_translator
  Downloading deep_translator-1.10.1-py3-none-any.whl (35 kB)
Requirement already satisfied: beautifulsoup4<5.0.0,>=4.9.1 in /opt/conda/lib/p
Requirement already satisfied: requests<3.0.0,>=2.23.0 in /opt/conda/lib/python
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.7/site-p
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.7/site-pa
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/s
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/pytho
Installing collected packages: deep_translator
Successfully installed deep_translator-1.10.1
WARNING: Running pip as the 'root' user can result in broken permissions and co
```

```
842:    learn: 0.4652169        total: 15.1s    remaining: 2.81s
```

```
from deep_translator import GoogleTranslator
import random

non_satisfied_trans = []
random.seed(42)

for sentence in random.sample(corpus, 1000):
    non_satisfied_trans.append(GoogleTranslator(source='portuguese', target='english
```

```
non_satisfied_trans
```

```
['aimda product has not arrived',
 'pel cula gel came another device',
```

```
 "They didn't deliver the deadline by the t day m. It will be scissors to buy
here because the company didn't commit, send an email to reassure",
 'I will not confirm delivery where the product is purchased, it just
arrived, only the purchase code tracking says delivery of complicated goods',
 'apparently good product not yet time to test',
 'thanks',
 "I don't inform any order I will receive it The problem happened I can't
contact due information",
 'delivery delay product arrived defect',
 'only product delivered rope I did not receive palm',
 'I bought two perfumes, only came in a box to help, counterfeit perfume
marked in a forbidden sale box, only for testing',
 "The brush and brush kit did not arrive, it can come separately. Anyway, I
didn't like the shipping method, so it informs the product delivered.",
 "delivered quickly I didn't like the product bad fabric",
 "hello i bought product just thermally surprised i didn't put very cold
water it didn't last i recommend store n product",
 'store can be good but product bad quality',
 "order arrived wrong missing item I paid for two mirrors I just received it
I'm waiting for an urgent solution",
 "I've been waiting for the product for a month, I've already contacted them,
they still haven't solved the problem, partner Lannist, it's horrible",
 'I bought the product there was stock soon I received it informs there was
no stock I am waiting for a response to exchange the product another money
back',
 'doll came pacifier',
 'some brushes are loosening bristles',
 'Lannist delivery time arrives order day',
 "inferior product I didn't like it",
 'I request reversal of credit card payment missing product delivery',
 'delay product delivery',
 "I still haven't received product",
 'little resistant material',
 'I was surprised only the product arrived on request I realized it was
delivered to two carriers so far I have not received another item worried',
 'xxxc',
 "I still haven't received the product, the system is delivered",
 'I received an excellent product, in excellent conditions, before the
expected time, color difference, I did not request an exchange to avoid
inconvenience, delay in having the product',
 "I didn't receive goods inform answer nothing nothing",
 'photo site lannist displayed photo cartridge canon origin m product
received n canon treats cartridge gen rico reposi',
 'product takes approximately one day to be delivered in addition to that
they sent a product another totally divergent model announced flee targaryen
partner',
 'verdad est courier form delayed delivery',
 'Order received exception to delivery date',
 'I bought two shoes and I only received an invoice issued for two items',
 'I heard a problem with delivery not carried out I want a refund',
 'I bought stark finding it delivered via carrier they sent mail to the post
office where I live cross the city to get mail return by bus',
 'table top arrived all crumpled',
 'I bought a product I received a completely different one I must proceed',
```

```
# Apply stemming to the translated text
non_satisfied_final = []

for sent in non_satisfied_trans:
```

```
    for word in sent.split():

        if word not in set(stopwords.words('english')):

            non_satisfied_final.append(stemmer.stem(word))

non_satisfied_final
```

```
['aimda',
 'product',
 'arriv',
 'pel',
 'cula',
 'gel',
 'came',
 'anoth',
 'devic',
 'they',
 'deliv',
 'deadlin',
 'day',
 'm.',
 'It',
 'scissor',
 'buy',
 'compani',
 'commit,',
 'send',
 'email',
 'reassur',
 'I',
 'confirm',
 'deliveri',
 'product',
 'purchased,',
 'arrived,',
 'purchas',
 'code',
 'track',
 'say',
 'deliveri',
 'complic',
 'good',
 'appar',
 'good',
 'product',
 'yet',
 'time',
 'test',
 'thank',
 'I',
 'inform',
 'order',
 'I',
 'receiv',
 'the',
 'problem',
 'happen',
 'I',
```

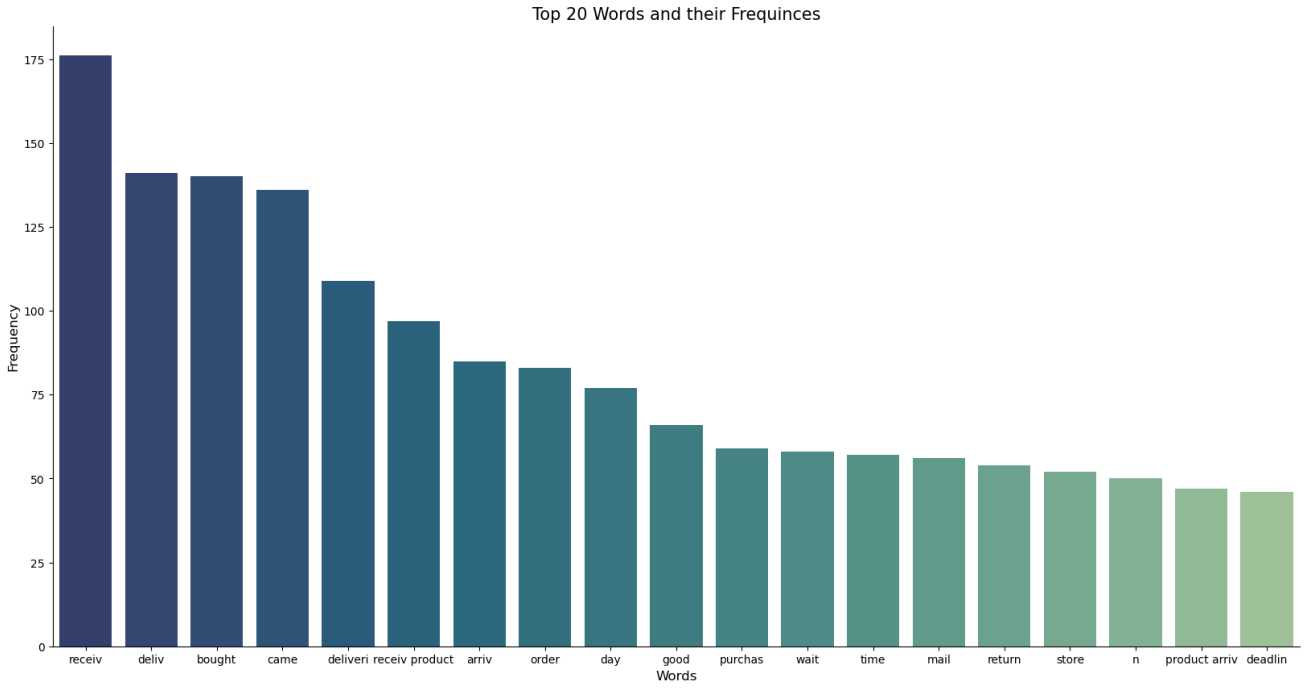
```
        "can't",
        'contact',
        'due',
        'inform',
        'deliveri',
        'delay',
        'product',
    987:    learn: 0.4577641        total: 17.7s    remaining: 215ms
```

```
# Visualize most common words for non-satsifaction
from wordcloud import WordCloud

non_satisfied_final = ' '.join(non_satisfied_final)
non_satisfied_freq = WordCloud(width=1000, height=800, background_color='white').gen

plt.figure(figsize=(15, 10))
plt.imshow(non_satisfied_freq)
plt.axis("off")
```

```
(-0.5, 999.5, 799.5, -0.5)
```


```
# Sort the word frequencies in descending order
non_satisfied_freq = non_satisfied_freq.process_text(non_satisfied_final)
sorted_word_frequencies = sorted(non_satisfied_freq.items(), key= lambda x : x[1], r
```


```
# Plot Words vs Frequency
plt.figure(figsize= [20, 10])
sns.barplot(x = pd.DataFrame(sorted_word_frequencies)[1:20][0], y= pd.DataFrame(sort
plt.title('Top 20 Words and their Frequinces', fontsize= 15)
plt.xlabel('Words', fontsize= 12)
plt.ylabel('Frequency', fontsize= 12)
sns.despine()
```



From the study of non-satisfied customers reviews, we can see that majority of words mentioned are related to shipping issues.

9.0 Customer Segmentation

9.1 Customer Segmentation by RFM Analysis

```
df_2.head()
```

	customer_id	customer_unique_id	customer_zip_c
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	
1	8912fc0c3bbf1e2fbf35819e21706718	9eae34bbd3a474ec5d07949ca7de67c0	
2	8912fc0c3bbf1e2fbf35819e21706718	9eae34bbd3a474ec5d07949ca7de67c0	
3	f0ac8e5a239118859b1734e1087cbb1f	3c799d181c34d51f6d44bbbc563024db	
4	6bc8d08963a135220ed6c6d098831f84	23397e992b09769faf5e66f9e171a241	

5 rows × 46 columns

```
# Get last transaction date to help calculate Recency
max_trans_date = max(df_2.order_purchase_timestamp).date()
max_trans_date
```

```
datetime.date(2018, 8, 29)
```

Create Recency, Frequancy and Monetary Features

```
from datetime import datetime

rfm_table = df_2.groupby('customer_unique_id').agg({'order_purchase_timestamp': lamb
                                                    'product_id': lambda
                                                  'payment_value': lambda
rfm_table
```

customer_unique_id	order_purchase_timestamp	product_id	payment_va
0000366f3b9a7992bf8c76cfdf3221e2	110	1	14

```
# Rename columns
rfm_table.rename(columns={'order_purchase_timestamp':'Recency','product_id':'Frequan
rfm_table
```

customer_unique_id	Recency	Frequancy	Monetary
0000366f3b9a7992bf8c76cfdf3221e2	110	1	141.90
0000b849f77a49e4a4ce2b2a4ca5be3f	113	1	27.19
0000f46a3911fa3c0805444483337064	536	1	86.22
0000f6ccb0745a6a4b88665a16c9f078	320	1	43.62
0004aac84e0df4da2b147fca70cf8255	287	1	196.89
...	...	...	...
fffcf5a5ff07b0908bd4e2dbc735a684	446	2	4134.84
fffea47cd6d3cc0a88bd621562a9d061	261	1	84.58
ffff371b4d645b6ecea244b27531430a	567	1	112.46
ffff5962728ec6157033ef9805bacc48	118	1	133.69
ffffd2657e2aad2907e67c3e9daecbeb	483	1	71.56

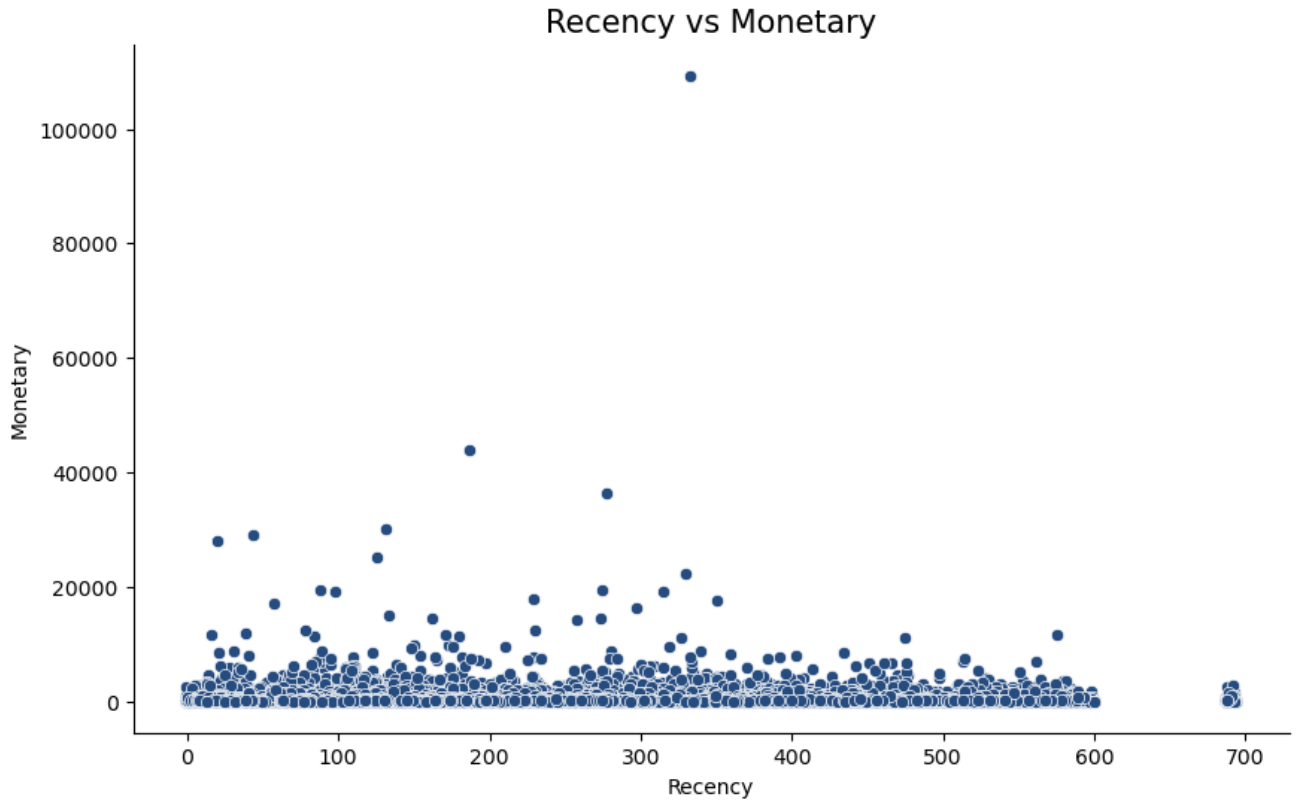
90967 rows × 3 columns

Create Recency, Frequancy and Monetary scores

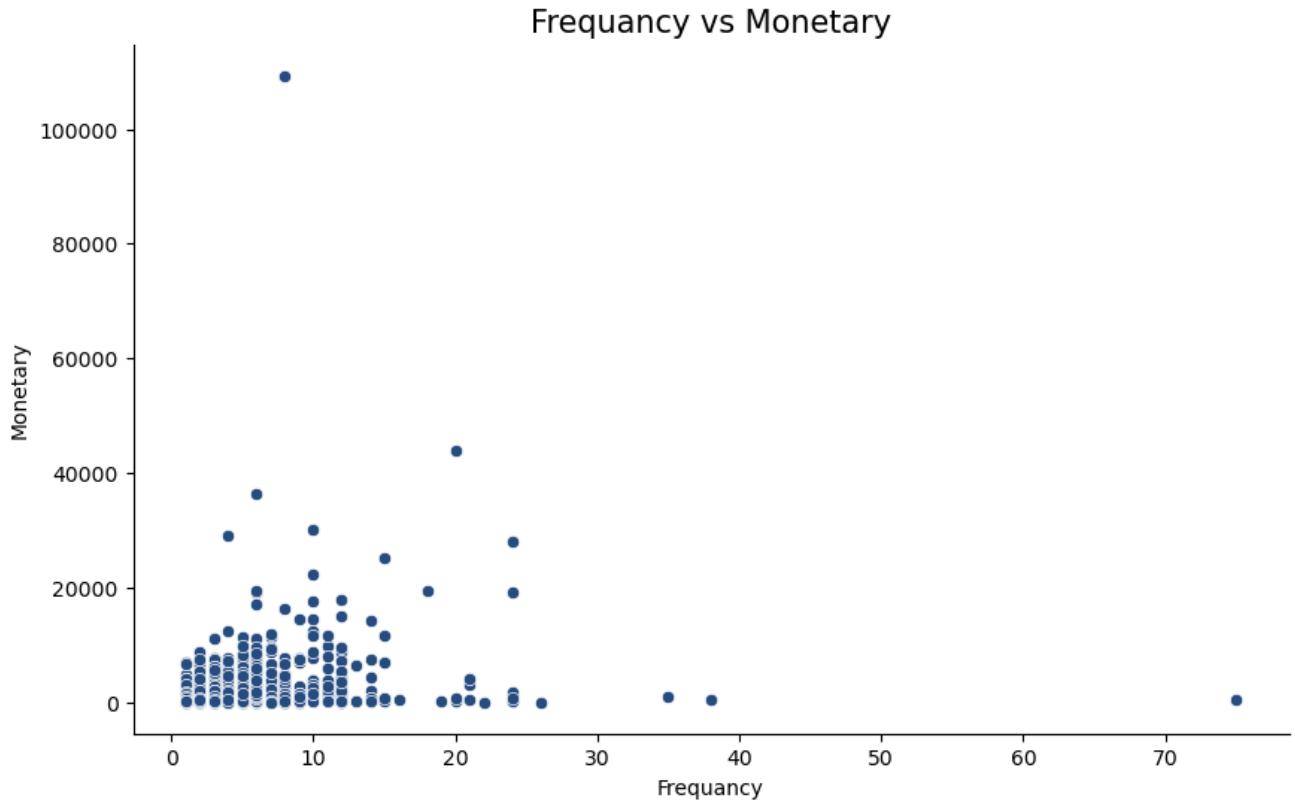
```
rfm_table['r_score'] = pd.qcut(rfm_table['Recency'], 4, ['4','3','2','1'])
rfm_table['f_score'] = pd.qcut(rfm_table['Frequancy'].rank(method= 'first'), 4, ['1'
rfm_table['m_score'] = pd.qcut(rfm_table['Monetary'], 4, ['1','2','3','4'])
rfm_table
```

customer_unique_id	Recency	Frequancy	Monetary	r_score	f_score	m
0000366f3b9a7992bf8c76cfdf3221e2	110	1	141.90	4	1	
0000b849f77a49e4a4ce2b2a4ca5be3f	113	1	27.19	3	1	
0000f46a3911fa3c0805444483337064	536	1	86.22	1	1	
0000f6ccb0745a6a4b88665a16c9f078	320	1	43.62	2	1	
0004aac84e0df4da2b147fca70cf8255	287	1	196.89	2	1	

```
plt.figure(figsize= [10, 6])
sns.scatterplot(x= 'Recency',y= 'Monetary', data=rfm_table)
plt.title('Recency vs Monetary', fontsize= 15)
sns.despine()
```




```
plt.figure(figsize= [10, 6])
sns.scatterplot(x='Frequancy', y='Monetary', data=rfm_table)
plt.title('Frequancy vs Monetary', fontsize= 15)
sns.despine()
```



Calculate RFM Score

```
rfm_table['rfm_score'] = 100 * rfm_table['r_score'].astype(int) + 10 * rfm_table['f_
rfm_table
```

	Recency	Frequancy	Monetary	r_score	f_score	m
0000366f3b9a7992bf8c76cfdf3221e2	110	1	141.90	4	1	

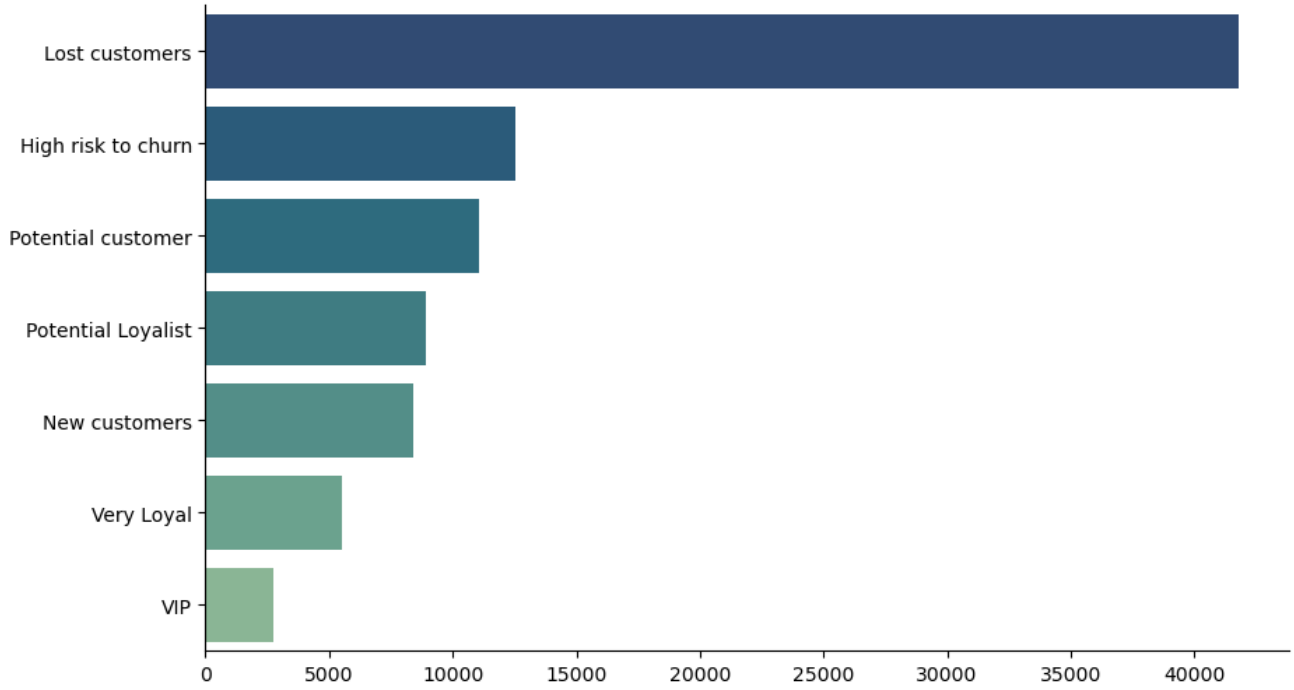
Cluster customers based on RFM Score

```
def customer_segmenation(rfm_score):

  if rfm_score == 444:
    return 'VIP'

  elif  rfm_score >= 433 and rfm_score < 444:
    return 'Very Loyal'

  elif   rfm_score >=421 and rfm_score< 433:
    return 'Potential Loyalist'

  elif rfm_score>=344 and rfm_score < 421:
    return 'New customers'

  elif rfm_score>=323 and rfm_score<344:
    return 'Potential customer'

  elif rfm_score>=224 and rfm_score<311:
    return 'High risk to churn'

  else:
    return 'Lost customers'

rfm_table['customer_segmentation'] = rfm_table['rfm_score'].apply(customer_segmenati

rfm_table
```

customer_unique_id	Recency	Frequancy	Monetary	r_score	f_score	m
0000366f3b9a7992bf8c76cfdf3221e2	110	1	141.90	4	1	
0000b849f77a49e4a4ce2b2a4ca5be3f	113	1	27.19	3	1	
0000f46a3911fa3c0805444483337064	536	1	86.22	1	1	
0000f6ccb0745a6a4b88665a16c9f078	320	1	43.62	2	1	
0004aac84e0df4da2b147fca70cf8255	287	1	196.89	2	1	
...	...	...	...	...	...	...
fffcf5a5ff07b0908bd4e2dbc735a684	446	2	4134.84	1	4	
fffea47cd6d3cc0a88bd621562a9d061	261	1	84.58	2	4	
ffff371b4d645b6ecea244b27531430a	567	1	112.46	1	4	
ffff5962728ec6157033ef9805bacc48	118	1	133.69	3	4	
ffffd2657e2aad2907e67c3e9daecbeb	483	1	71.56	1	4	

90967 rows × 8 columns

```
# Plot frquency of each segment
plt.figure(figsize=[10,6])
sns.barplot(x = rfm_table.customer_segmentation.value_counts().values, y= rfm_table.
sns.despine()
```



Check Outliers

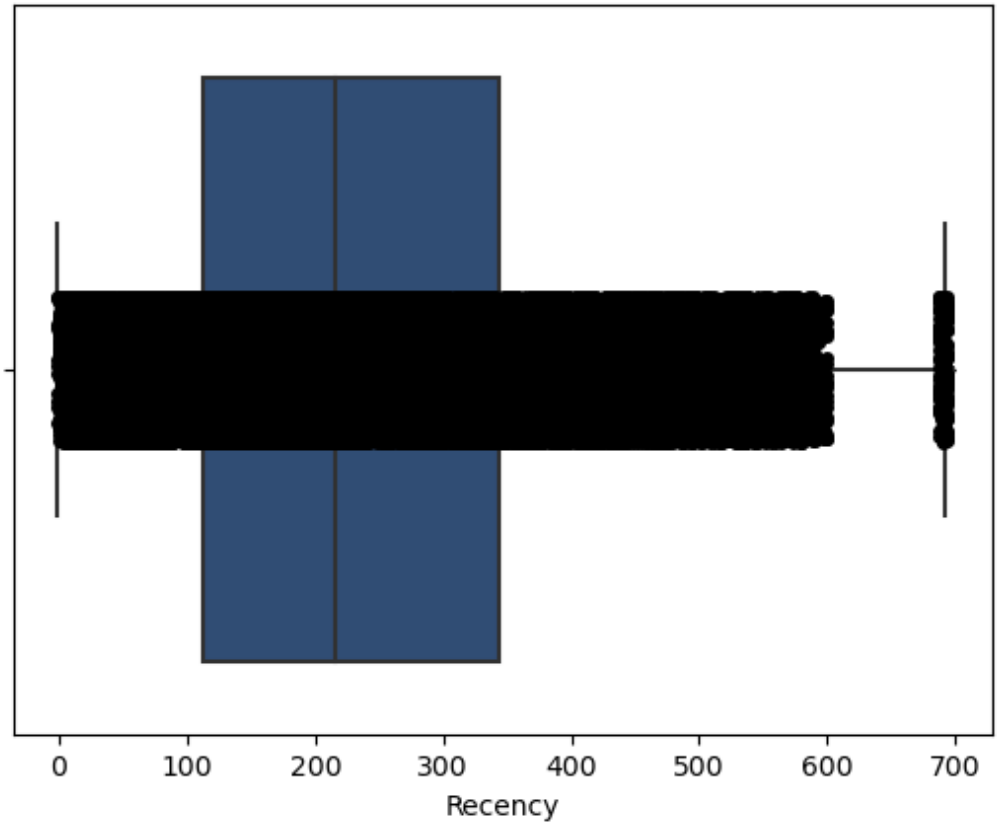
```
rfm_table.describe()
```

	Recency	Frequancy	Monetary	rfm_score
count	90967.000000	90967.000000	90967.000000	90967.000000
mean	235.148197	1.237042	211.874402	277.935768
std	152.224787	0.843657	632.222983	112.463024
min	-1.000000	1.000000	9.590000	111.000000
25%	112.000000	1.000000	63.680000	211.000000
50%	216.000000	1.000000	112.680000	311.000000
75%	344.000000	1.000000	201.990000	411.000000
max	694.000000	75.000000	109312.640000	444.000000

Recency

```
sns.boxplot(x= rfm_table.Recency)
sns.stripplot(x = rfm_table.Recency, color= 'black')
```

```
<AxesSubplot:xlabel='Recency'>
```



Frequancy

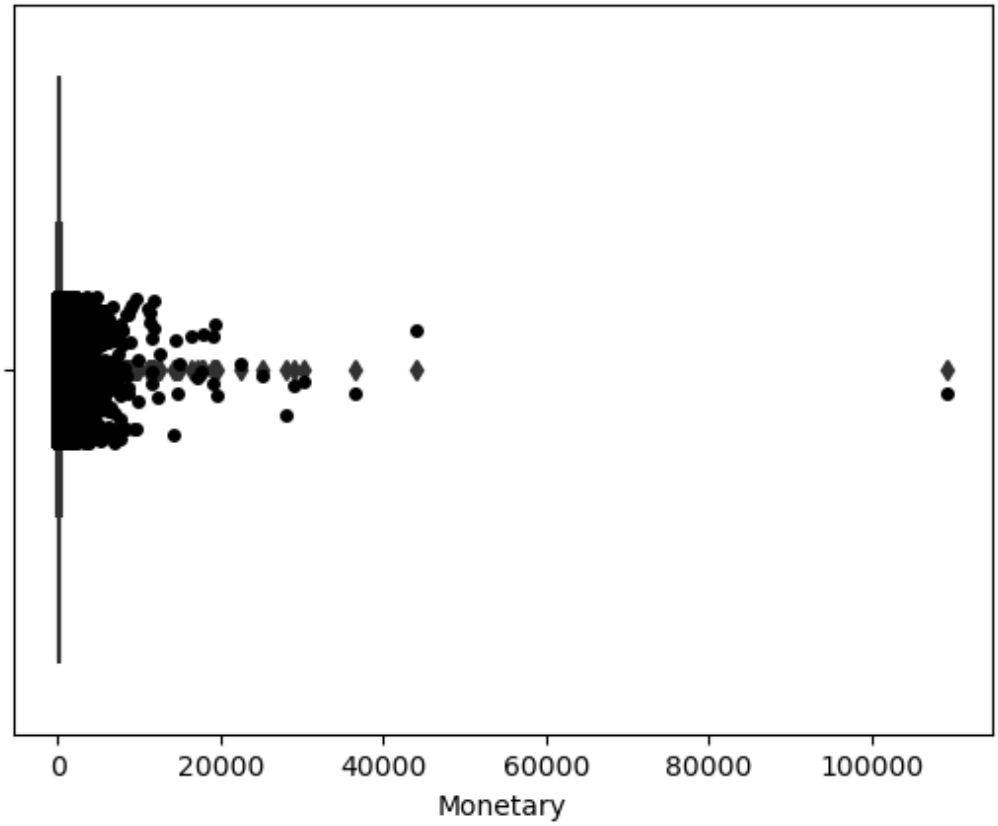
```
sns.boxplot(x= rfm_table.Frequancy)
sns.stripplot(x = rfm_table.Frequancy, color= 'black')
```

```
<AxesSubplot:xlabel='Frequancy'>
```

Monetary

```
sns.boxplot(x= rfm_table.Monetary)
sns.stripplot(x = rfm_table.Monetary, color= 'black')
```

```
<AxesSubplot:xlabel='Monetary'>
```



Remove Extreme 5% of Outliers

```
print('Recency 5% Outliers Limits:', np.percentile(rfm_table.Recency, 5), np.percent
print('Frequancy 5% Outliers Limits:', np.percentile(rfm_table.Frequancy, 5), np.per
print('Monetary 5% Outliers Limits:', np.percentile(rfm_table.Monetary, 5), np.perce
```

```
Recency 5% Outliers Limits: 22.0 517.0
Frequancy 5% Outliers Limits: 1.0 2.0
Monetary 5% Outliers Limits: 32.69 660.3969999999999
```

Remove Outliers for Recency & Monetary (Extreme 5%)

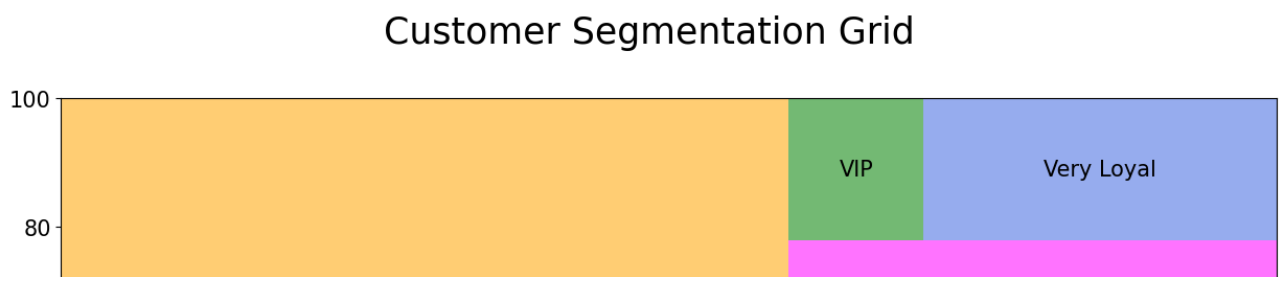
```
for i in [0, 2]:

    outlier_indices = []
    col = rfm_table.columns[i]
    percentile_5 = np.percentile(rfm_table[col], 5)
    percentile_95 = np.percentile(rfm_table[col], 95)
    outlier_indices.append(rfm_table[(rfm_table[col] < percentile_5) | (rfm_table[co

rfm_table.drop(outlier_indices[0][:], inplace= True)
rfm_table.reset_index(inplace= True, drop= True)
```

```
! pip install squarify
```

```
Requirement already satisfied: squarify in /opt/conda/lib/python3.7/site-packag
WARNING: Running pip as the 'root' user can result in broken permissions and co
```

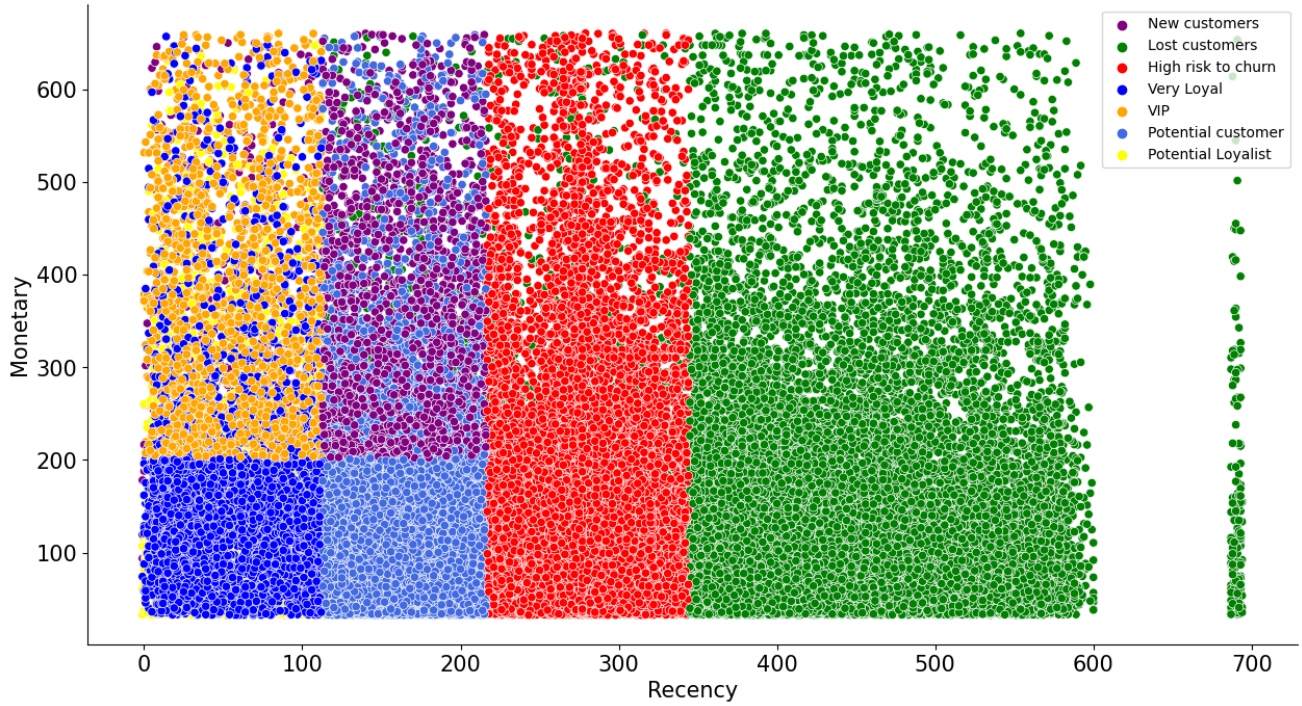
Customer Segmentation Grid

```
import squarify

plt.figure(figsize=[15,8])
plt.rc('font', size=15)

Sizes = rfm_table.groupby('customer_segmentation')[['Monetary']].count()
squarify.plot(sizes= Sizes.values, label= Sizes.index, color=["red", "orange", "blue
plt.suptitle("Customer Segmentation Grid", fontsize=25);
```



Recency & Monetary Plot

```
plt.figure(figsize= [15, 8])
colors = ['purple', 'green', 'red', 'blue', 'orange', 'royalblue', 'yellow']
sns.scatterplot(x= rfm_table.Recency, y= rfm_table.Monetary, hue= rfm_table.customer
plt.legend(prop={'size':10})
sns.despine()
```

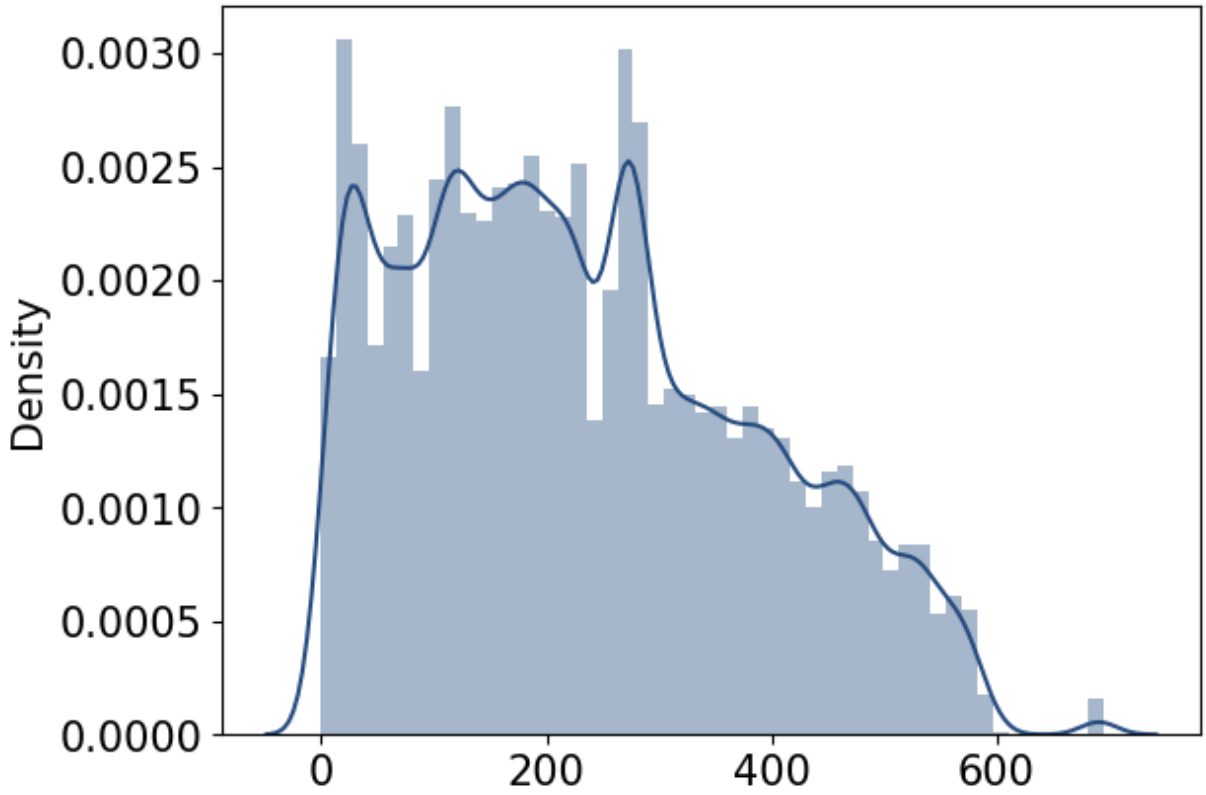


Check Skeweness

```
# Recency
sns.distplot(x= rfm_table.Recency)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
<AxesSubplot:ylabel='Density'>
```

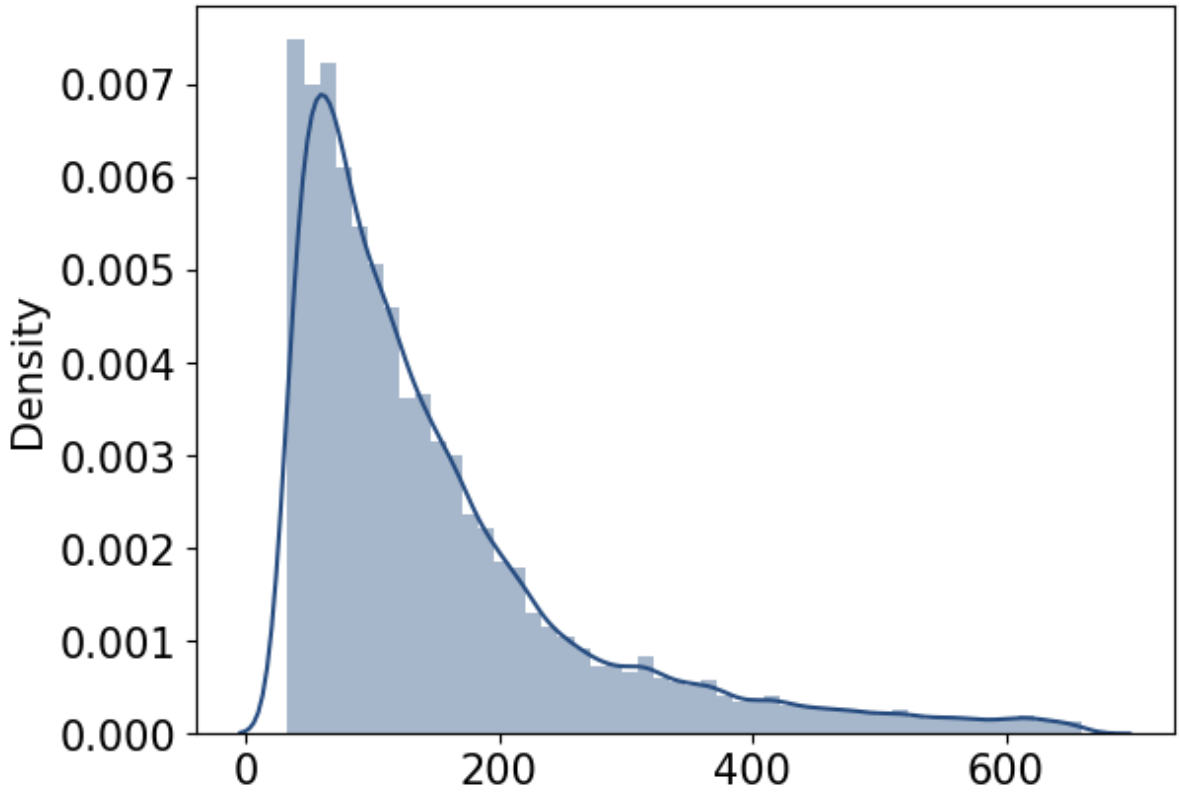



```
# Frequancy
sns.distplot(x= rfm_table.Frequancy)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751


<AxesSubplot:ylabel='Density'>
```


```
# Monetary
sns.distplot(x= rfm_table.Monetary)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751


<AxesSubplot:ylabel='Density'>
```



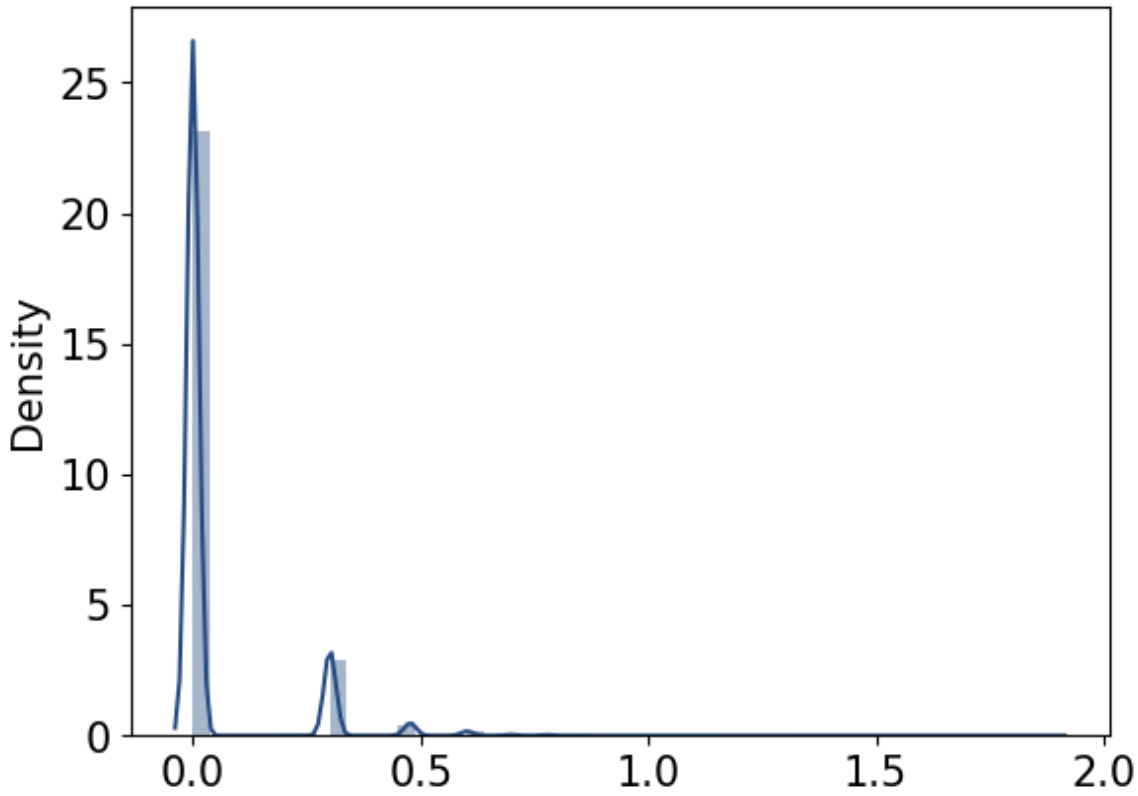
Apply Log function to handle skeweness for Frequancy & Monetary

```
for i in ['Frequancy', 'Monetary']:
    rfm_table[i] = np.log10(rfm_table[i])
```

```
# Frequancy
sns.distplot(x= rfm_table.Frequancy)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

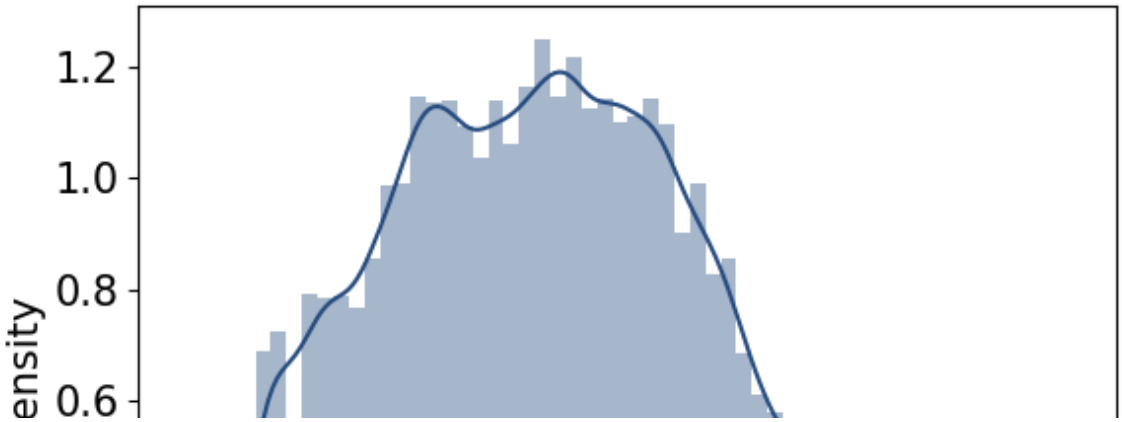
```
<AxesSubplot:ylabel='Density'>
```




```
# Monetary
sns.distplot(x= rfm_table.Monetary)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
<AxesSubplot:ylabel='Density'>
```



9.2 Clustering with K-means

```
df_cluster = df_2[['freight_value', 'price', 'payment_value', 'payment_installments'
df_cluster
```

	freight_value	price	payment_value	payment_installments	payment_seque
0	21.88	124.99	146.87	2	
1	24.90	112.99	275.79	1	
2	24.90	112.99	275.79	1	
3	15.62	124.99	140.61	7	
4	30.59	106.99	137.58	10	
...	...	...	...	...	
112525	39.19	193.00	232.19	10	
112526	37.70	389.00	426.70	8	
112527	21.46	139.00	160.46	3	
112528	26.18	129.00	55.18	2	
112529	26.18	129.00	100.00	1	

112530 rows × 5 columns

Take sample from data (10k)

```
df_sample = df_cluster.sample(frac= 1, random_state= 42)[:10000]
```

Save sample as CSV for deployment

```
df_sample.to_csv('Clustering Sample.csv')
```

```
df_sample.describe()
```

	freight_value	price	payment_value	payment_installments	payment_s
count	10000.000000	10000.000000	10000.000000	10000.000000	1(
mean	19.914930	119.287136	169.829120	2.931900	
std	16.079686	181.080900	240.579562	2.771828	
min	0.000000	0.850000	0.030000	1.000000	
25%	13.070000	39.990000	61.850000	1.000000	
50%	16.320000	75.000000	108.000000	2.000000	
75%	21.190000	130.125000	188.940000	4.000000	
max	338.300000	3999.900000	7274.880000	24.000000	

Drop freight values with zeros

```
df_sample.drop(df_sample[df_sample.freight_value == 0].index, inplace= True)
df_sample.reset_index(inplace= True, drop= True)
```

Take copy for Pipeline

```
cluster_pipeline = df_sample.copy()
```

```
for i in ['freight_value', 'price', 'payment_value', 'payment_installments', 'paymen
    df_sample[i] = np.log10(df_sample[i])
```

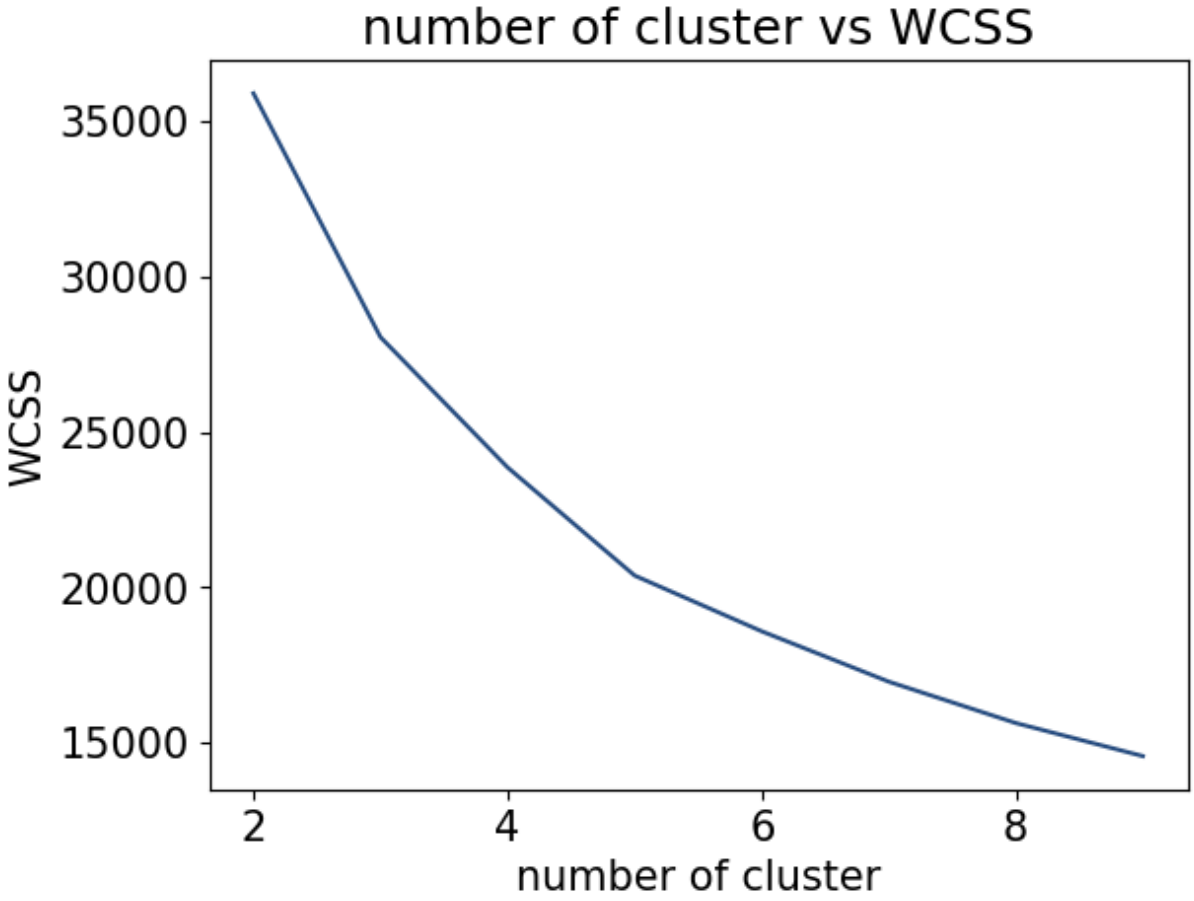
Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler(with_mean= False)
data_scaled = sc.fit_transform(df_sample)
```

Detecting number of clusters uning Elbow Method

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
wcss = []
scores = []

for i in range(2,10):
  kmean = KMeans(n_clusters=i)
  y_pred_kmean = kmean.fit_predict(data_scaled)
  wcss.append(kmean.inertia_)
  scores.append(silhouette_score(data_scaled,y_pred_kmean))

plt.plot(range(2,10),wcss)
plt.title('number of cluster vs WCSS')
plt.xlabel('number of cluster')
plt.ylabel('WCSS')
```

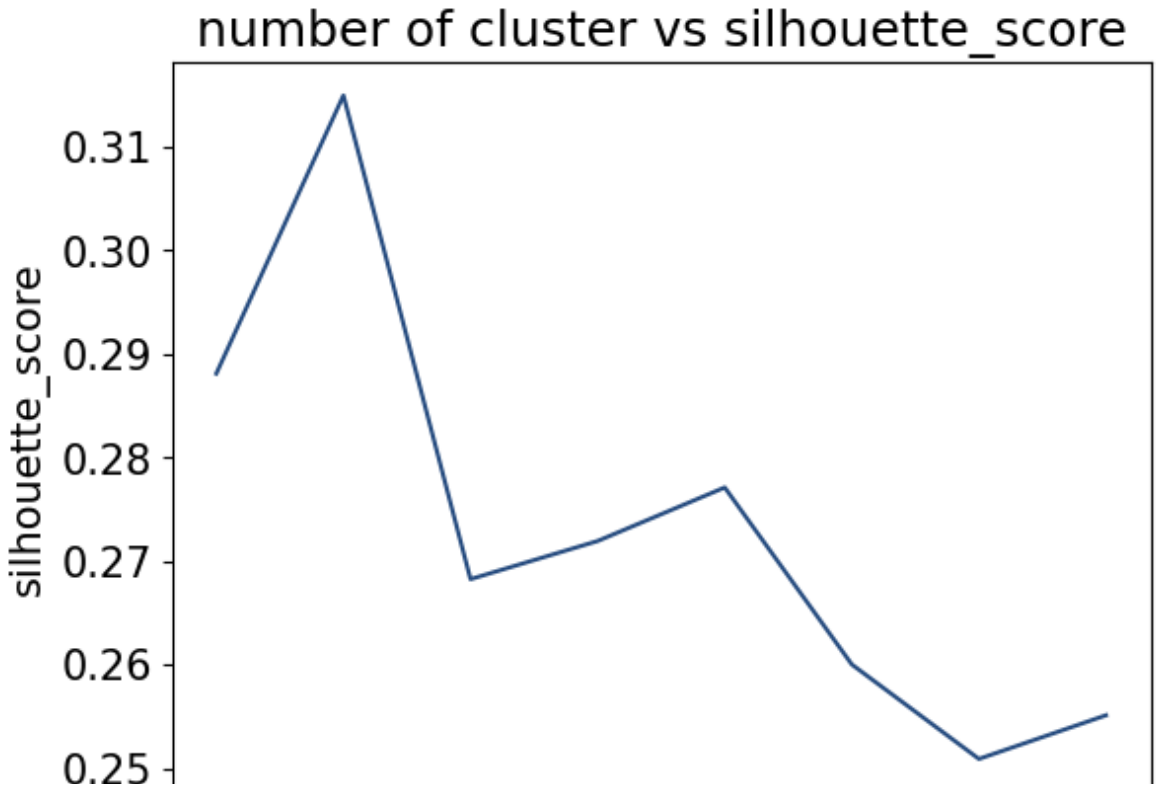
```
Text(0, 0.5, 'WCSS')
```



Detecting number of clusters using Silhouete Score

```
plt.plot(range(2,10),scores)
plt.title('number of cluster vs silhouette_score')
plt.xlabel('number of cluster')
plt.ylabel('silhouette_score')
```

```
Text(0, 0.5, 'silhouette_score')
```



Select number of clusters k= 3

```
from sklearn.cluster import KMeans

kmean = KMeans(n_clusters= 3)
y_pred_kmean = kmean.fit_predict(data_scaled)


# Count of each cluster
len(kmean.labels_[kmean.labels_ == 0]), len(kmean.labels_[kmean.labels_ == 1]), len(
```

```
(4155, 5388, 423)
```

```
# Take another sample of original cluster dataframe to assign kmeans labels
original_cluster_sample = df_cluster.sample(frac= 1, random_state= 42)[:9966]

# Assign cluster label to original cluster sample
original_cluster_sample['cluster_label'] = y_pred_kmean
original_cluster_sample.head()
```

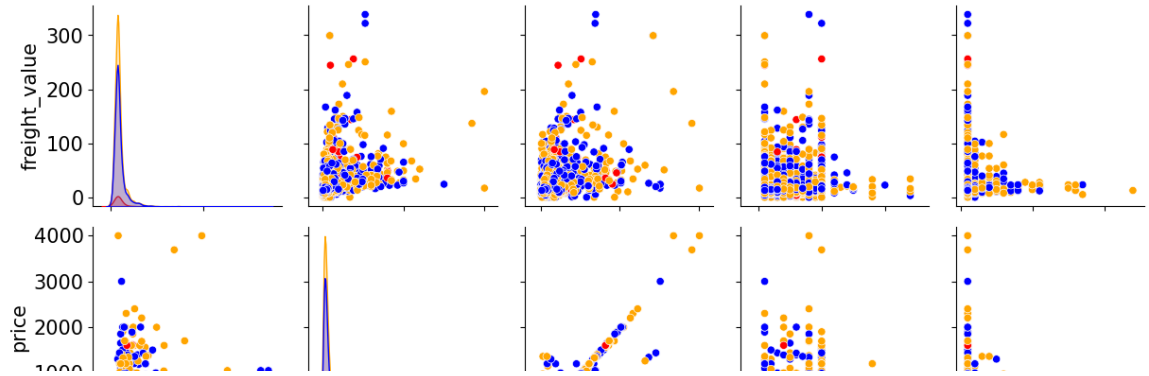
	freight_value	price	payment_value	payment_installments	payment_sequen
20549	18.23	45.0	63.23	1	
96866	7.87	32.5	40.37	1	
51608	12.87	195.9	417.54	1	
20676	22.41	76.0	479.22	2	
98133	15.10	25.9	41.00	1	

```
original_cluster_sample.groupby('cluster_label').describe().T
```

	cluster_label	0	1	2
freight_value	count	4155.000000	5388.000000	423.000000
	mean	20.073249	19.727804	20.604184
	std	16.509072	15.291620	21.042242
	min	0.000000	0.000000	0.000000
	25%	13.075000	12.997500	13.370000
	50%	16.250000	16.335000	16.740000
	75%	21.330000	21.150000	20.800000
	max	338.300000	299.160000	255.920000

```
sns.pairplot(data= original_cluster_sample, hue= 'cluster_label', palette= ['blue',
```

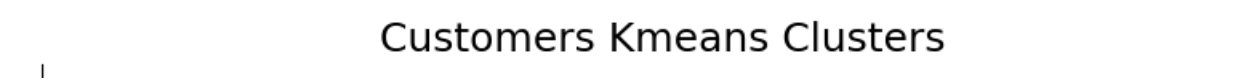
```
<seaborn.axisgrid.PairGrid at 0x7b3209908cd0>
```



As we can see from statistics table and pairplot that clusters have high percentage of overlaping, sow RFM would be better in this case to cluster customers.

Show Kmeans Clusters


```
plt.figure(figsize=[10, 6])
plt.scatter(data_scaled[y_pred_kmean==0,0], data_scaled[y_pred_kmean==0,1], c = 'red
plt.scatter(data_scaled[y_pred_kmean==1,0], data_scaled[y_pred_kmean==1,1], c = 'gre
plt.scatter(data_scaled[y_pred_kmean==2,0], data_scaled[y_pred_kmean==2,1], c = 'blu
plt.scatter(kmean.cluster_centers_[:,0], kmean.cluster_centers_[:,1], c='yellow', s=
plt.title('Customers Kmeans Clusters')
plt.legend()
sns.despine()
```



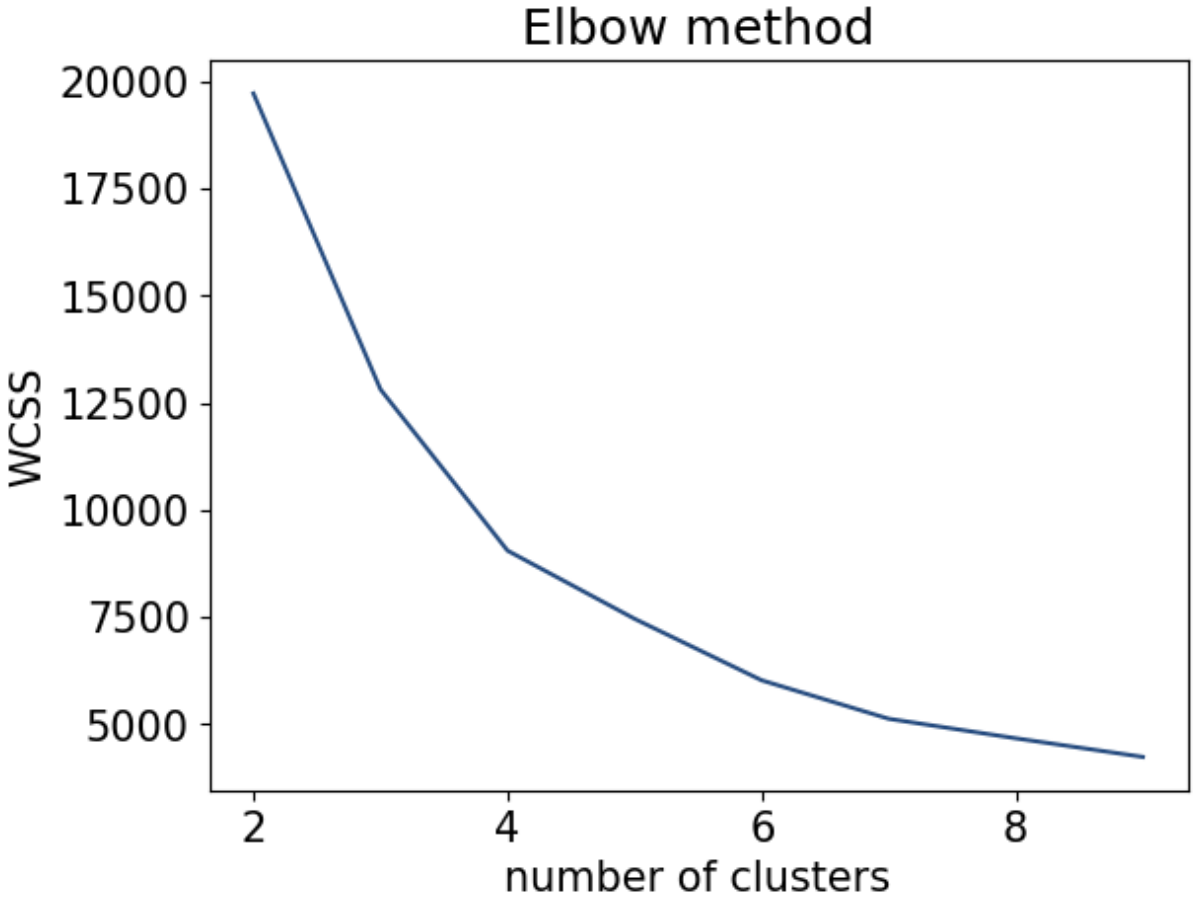
Show Clusters using PCA

```
from sklearn.decomposition import PCA
pca = PCA(n_components= 2)
x_pca = pca.fit_transform(data_scaled)
pca.explained_variance_ratio_
```

```
array([0.45422138, 0.21956351])
```

```
wcss = []
scores = []
for i in range(2,10):
  kmean = KMeans(n_clusters=i)
  y_pred = kmean.fit_predict(x_pca)
  wcss.append(kmean.inertia_)
  scores.append(silhouette_score(x_pca,y_pred))
plt.plot(range(2,10),wcss)
plt.title('Elbow method')
plt.xlabel('number of clusters')
plt.ylabel('WCSS')
```

```
Text(0, 0.5, 'WCSS')
```



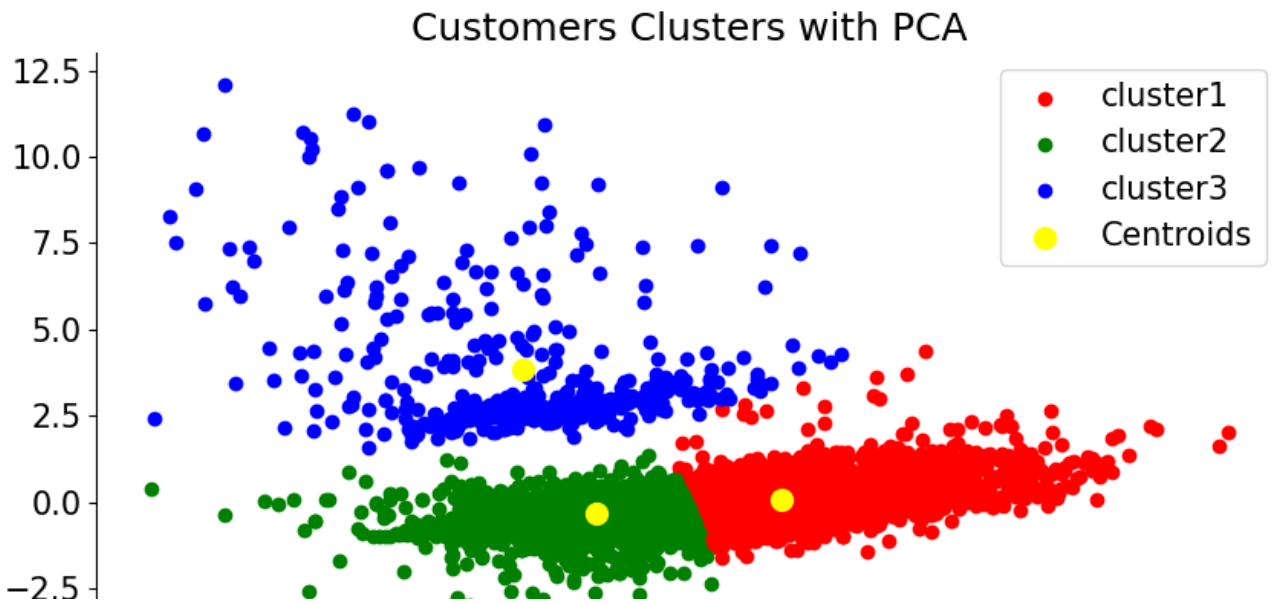

```
plt.plot(range(2,10),scores)
plt.title('silhouette_score')
```

```
plt.xlabel('number of clusters')
plt.ylabel('silhouette_score')
```

```
Text(0, 0.5, 'silhouette_score')
```


```
kmean = KMeans(n_clusters=3)
y_pred_pca = kmean.fit_predict(x_pca)
```

```
plt.figure(figsize=[10, 6])
plt.scatter(x_pca[y_pred_pca==0,0],x_pca[y_pred_pca==0,1],c = 'red',label = 'cluster
plt.scatter(x_pca[y_pred_pca==1,0],x_pca[y_pred_pca==1,1],c = 'green',label = 'clust
plt.scatter(x_pca[y_pred_pca==2,0],x_pca[y_pred_pca==2,1],c = 'blue',label = 'cluste
plt.scatter(kmean.cluster_centers_[:,0],kmean.cluster_centers_[:,1],c='yellow',s=100
plt.title('Customers Clusters with PCA')
plt.legend()
sns.despine()
```



9.3 Pipeline

Prepare Features

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import FunctionTransformer

numerical_pipeline_cluster = Pipeline(steps=[('Feature Scaling', StandardScaler(with_
```

```
from sklearn.compose import ColumnTransformer

preprocessing_cluster = ColumnTransformer(transformers= [('Numerical Columns', numer
                                          remainder= 'passthrough')
preprocessing_cluster
```

```
    ColumnTransformer(remainder='passthrough',
                      transformers=[('Numerical Columns',
                                     Pipeline(steps=[('Feature Scaling',

    StandardScaler(with_mean=False))]),
                                     Index(['freight_value', 'price',
    'payment_value', 'payment_installments',
           'payment_sequential'],
          dtype='object'))])
```

```
final_pipeline_cluster = Pipeline(steps=[('Preprocessing', preprocessing_cluster), (
                                ('Model', KMeans(n_clusters= 3))])
final_pipeline_cluster
```

```
    Pipeline(steps=[('Preprocessing',
                     ColumnTransformer(remainder='passthrough',
                                       transformers=[('Numerical Columns',
                                                      Pipeline(steps=[('Feature '
```

```
                                                                    'Scaling',

      StandardScaler(with_mean=False))]),
                                                        Index(['freight_value',
      'price', 'payment_value', 'payment_installments',
             'payment_sequential'],
            dtype='object'))])),
                     ('Log Transformer', FunctionTransformer(func=<ufunc
      'log10'>)),
                     ('Model', KMeans(n_clusters=3))])


# Fit pipeline to Dataframe
final_pipeline_cluster.fit(cluster_pipeline)

      Pipeline(steps=[('Preprocessing',
                       ColumnTransformer(remainder='passthrough',
                                         transformers=[('Numerical Columns',
                                                        Pipeline(steps=[('Feature '
                                                                         'Scaling',

      StandardScaler(with_mean=False))]),
                                                        Index(['freight_value',
      'price', 'payment_value', 'payment_installments',
             'payment_sequential'],
            dtype='object'))])),
                     ('Log Transformer', FunctionTransformer(func=<ufunc
      'log10'>)),
                     ('Model', KMeans(n_clusters=3))])


# Save model as bkl file
import joblib
joblib.dump(final_pipeline_cluster, 'Brazilian Ecommerce Clustering.bkl')

      ['Brazilian Ecommerce Clustering.bkl']


final_pipeline_cluster

      Pipeline(steps=[('Preprocessing',
                       ColumnTransformer(remainder='passthrough',
                                         transformers=[('Numerical Columns',
                                                        Pipeline(steps=[('Feature '
                                                                         'Scaling',

      StandardScaler(with_mean=False))]),
                                                        Index(['freight_value',
      'price', 'payment_value', 'payment_installments',
             'payment_sequential'],
            dtype='object'))])),
                     ('Log Transformer', FunctionTransformer(func=<ufunc
      'log10'>)),
                     ('Model', KMeans(n_clusters=3))])
```

10.0 Model Deployment

```
model_classification = joblib.load('Brazilian Ecommerce Classification.bkl')
model_clustering = joblib.load('Brazilian Ecommerce Clustering.bkl')
```

Test Classification Model

```
model_classification.predict(pd.DataFrame({'freight_value' :[30], 'product_descripti
```

```
array([1])
```

Test Clustering Model

```
model_clustering.predict(pd.DataFrame({'freight_value' :[10], 'price' :[90], 'paymen
```

```
array([2], dtype=int32)
```

```
# Install neccessary libraries for deployment

! pip install ydata_profiling
! pip install streamlit_pandas_profiling
```

```
Collecting ydata_profiling
  Downloading ydata_profiling-4.1.2-py2.py3-none-any.whl (345 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 345.9/345.9 kB 15.1 MB/s eta 0:00
Requirement already satisfied: statsmodels<0.14,>=0.13.2 in /opt/conda/lib/py
Requirement already satisfied: numpy<1.24,>=1.16.0 in /opt/conda/lib/python3.
Requirement already satisfied: imagehash==4.3.1 in /opt/conda/lib/python3.7/s
Requirement already satisfied: jinja2<3.2,>=2.11.1 in /opt/conda/lib/python3.
Requirement already satisfied: pandas!=1.4.0,<1.6,>1.1 in /opt/conda/lib/pyth
Requirement already satisfied: visions[type_image_path]==0.7.5 in /opt/conda/
Requirement already satisfied: htmlmin==0.1.12 in /opt/conda/lib/python3.7/si
Requirement already satisfied: tqdm<4.65,>=4.48.2 in /opt/conda/lib/python3.7
Requirement already satisfied: matplotlib<3.7,>=3.2 in /opt/conda/lib/python3
Requirement already satisfied: multimethod<1.10,>=1.4 in /opt/conda/lib/pytho
Requirement already satisfied: typeguard<2.14,>=2.13.2 in /opt/conda/lib/pyth
Requirement already satisfied: requests<2.29,>=2.24.0 in /opt/conda/lib/pytho
Requirement already satisfied: phik<0.13,>=0.11.1 in /opt/conda/lib/python3.7
Requirement already satisfied: seaborn<0.13,>=0.10.1 in /opt/conda/lib/python
Requirement already satisfied: scipy<1.10,>=1.4.1 in /opt/conda/lib/python3.7
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in /opt/conda/lib/python3.7
Requirement already satisfied: pydantic<1.11,>=1.8.1 in /opt/conda/lib/python
Requirement already satisfied: PyWavelets in /opt/conda/lib/python3.7/site-pa
Requirement already satisfied: pillow in /opt/conda/lib/python3.7/site-packag
Requirement already satisfied: tangled-up-in-unicode>=0.0.4 in /opt/conda/lib
Requirement already satisfied: attrs>=19.3.0 in /opt/conda/lib/python3.7/site
Requirement already satisfied: networkx>=2.4 in /opt/conda/lib/python3.7/site
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/si
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.7/s
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.7/
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.7/
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.7/si
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-
Requirement already satisfied: joblib>=0.14.1 in /opt/conda/lib/python3.7/sit
```

```
Requirement already satisfied: typing-extensions>=4.2.0 in /opt/conda/lib/pyt
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/pyt
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.7/site-
Requirement already satisfied: patsy>=0.5.2 in /opt/conda/lib/python3.7/site-
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packages
Installing collected packages: ydata_profiling
Successfully installed ydata_profiling-4.1.2
WARNING: Running pip as the 'root' user can result in broken permissions and
Collecting streamlit_pandas_profiling
  Downloading streamlit_pandas_profiling-0.1.3-py3-none-any.whl (259 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 259.8/259.8 kB 11.8 MB/s eta 0:00
Collecting streamlit>=0.63
  Downloading streamlit-1.21.0-py2.py3-none-any.whl (9.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.7/9.7 MB 102.3 MB/s eta 0:00:
Requirement already satisfied: pandas-profiling in /opt/conda/lib/python3.7/s
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-package
Requirement already satisfied: rich>=10.11.0 in /opt/conda/lib/python3.7/site
Requirement already satisfied: tornado>=6.0.3 in /opt/conda/lib/python3.7/sit
Requirement already satisfied: protobuf<4,>=3.12 in /opt/conda/lib/python3.7/
Requirement already satisfied: cachetools>=4.0 in /opt/conda/lib/python3.7/si
```

Deployment with Streamlit

```
%%writefile Brazilian_Ecommerce_Project.py

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import joblib
import streamlit as st
from sklearn.preprocessing import  StandardScaler
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.decomposition import PCA
from ydata_profiling import ProfileReport
from streamlit_pandas_profiling import st_profile_report

# Load Classification and Clustering Pipeline models
model_classification = joblib.load('Brazilian Ecommerce Classification.bkl')
model_clustering = joblib.load('Brazilian Ecommerce Clustering.bkl')

# Create Sidebar to navigate between EDA, Classification and Clustering
sidebar = st.sidebar
mode = sidebar.radio('Mode', ['EDA', 'Classification', 'Clustering'])
st.markdown("<h1 style='text-align: center; color: #ff0000;'></h1>", unsafe_allow_ht

if mode == "EDA":

    def main():

        # Header of Customer Satisfaction Prediction
        html_temp="""
```

```
                    <div style="background-color:#F5F5F5">
                    <h1 style="color:#31333F;text-align:center;"> Customer Satisfact
                    </div>
                """
        # Create sidebar to upload CSV files
        with st.sidebar.header('Upload your CSV data'):
            uploaded_file = st.sidebar.file_uploader('Upload your input csv file')

        if uploaded_file is not None:
            # Read file and Put headers
            EDA_sample = pd.read_csv(uploaded_file, index_col= 0)
            pr = ProfileReport(EDA_sample, explorative=True)
            st.header('**Input DataFrame**')
            st.write(EDA_sample)
            st.write('---')
            st.header('**Pandas Profiling Report**')
            st_profile_report(pr)

        else:
            st.info('Awaiting for CSV file to be uploaded.')

    if __name__ == '__main__':
        main()

if mode == "Classification":

    # Define function to predict classification based on assigned features
    def predict_satisfaction(freight_value, product_description_lenght, product_photo
    estimated_days, arrival_days, arrival_status, seller_to_carrier_status, estimate

        prediction_classification = model_classification.predict(pd.DataFrame({'frei
        return prediction_classification

    def main():

        # Header of Customer Satisfaction Prediction
        html_temp="""
                    <div style="background-color:#F5F5F5">
                    <h1 style="color:#31333F;text-align:center;"> Customer Satisfact
                    </div>
                """
        st.markdown(html_temp,unsafe_allow_html=True)

        # Assign all features with desired data input method
        sidebar.title('Numerical Features')
        product_description_lenght = sidebar.slider('product_description_lenght', 4,
        product_photos_qty = sidebar.slider('product_photos_qty', 1,20,1)
        payment_installments = sidebar.slider('payment_installments', 1,24,1)
        estimated_days = sidebar.slider('estimated_days', 3,60,1)
        arrival_days = sidebar.slider('arrival_days', 0,60,1)
        payment_type = st.selectbox('payment_type', ['credit_card', 'boleto', 'vouch
        arrival_status = st.selectbox('arrival_status', ['OnTime/Early', 'Late'])
        seller_to_carrier_status = st.selectbox('seller_to_carrier_status', ['OnTime
        estimated_delivery_rate = st.selectbox('estimated_delivery_rate', ['Very Slo
        arrival_delivery_rate = st.selectbox('arrival_delivery_rate', ['Very Slow',
        shipping_delivery_rate = st.selectbox('shipping_delivery_rate Date', ['Very
```