

Team Number: 4

Our Names :

- Seif Mahmoud Ahmed
- Yassein Ahmed Mohamed
- Mohamed Ayman
- Ahmed Zedan
- Omar Mahamed

Sales Forecasting and Optimization

• Project Overview:

- The Sales Forecasting and Optimization project aims to predict future sales for a retail or e-commerce
- business by using historical sales data. The project involves data collection, cleaning, exploration, time-series
- forecasting model development, optimization, and deployment. The end goal is to have a model that can
- generate accurate sales predictions to help businesses optimize inventory, marketing, and sales strategies.

Milestone 1: Data Collection, Exploration, and Preprocessing

1. Data Collection:

Acquire a dataset containing historical sales data (e.g., daily or weekly sales data from retail

or e-commerce platforms).

Ensure that the dataset includes relevant features like sales amount, date, promotions,

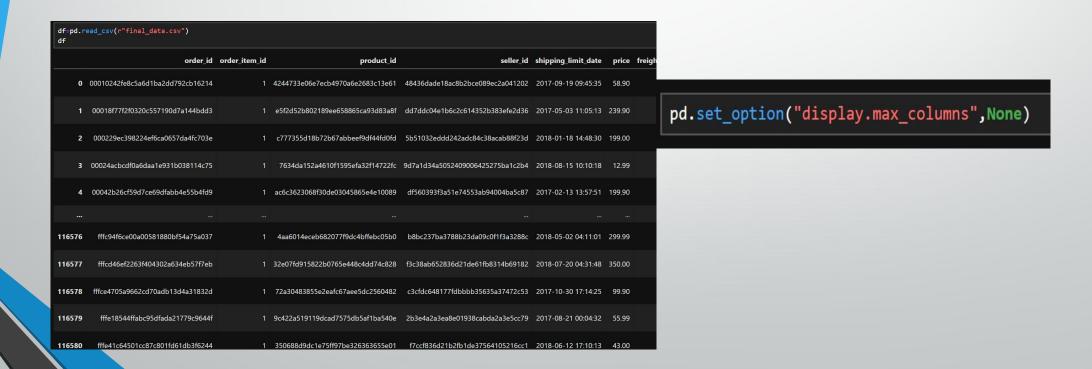
holidays, weather, etc.

1. Data Collection:

Acquire a dataset containing historical sales data (e.g., daily or weekly sales data from retail

or e-commerce platforms).

Ensure that the dataset includes relevant features like sales amount, date, promotions, holidays, weather, etc.



Here is what the data looks like.

```
[68]: df.shape
[68]: (116581, 40)
```

What is Data Types?

```
[73]: df.info()
      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 116581 entries, 0 to 116580
      Data columns (total 40 columns):
          Column
                                          Non-Null Count
                                                           Dtype
           order id
                                          116581 non-null
                                                           object
           order_item_id
                                          116581 non-null
           product_id
                                          116581 non-null
                                                           object
           seller id
                                          116581 non-null
           shipping_limit_date
                                          116581 non-null
                                                           object
           price
                                          116581 non-null
           freight_value
                                          116581 non-null
                                                           float64
                                          116581 non-null
           payment sequential
           payment_type
                                          116581 non-null
                                                           object
           payment_installments
                                          116581 non-null
                                          116581 non-null
       10 payment_value
                                                           float64
           review_id
                                          116581 non-null
                                                           object
       12 review score
                                          116581 non-null
                                                           int64
       13 review comment title
                                          13996 non-null
                                                           object
       14 review_comment_message
                                          49878 non-null
          review creation date
                                          116581 non-null
                                                           object
       16 review_answer_timestamp
                                          116581 non-null
                                                           object
       17 customer id
                                          116581 non-null
                                                           object
       18 order_status
                                          116581 non-null
                                          116581 non-null
                                                           object
       19 order_purchase_timestamp
           order_approved_at
                                          116567 non-null
       21 order delivered carrier date
                                          115368 non-null
                                                           object
       22 order_delivered_customer_date
                                         114066 non-null
       23 order_estimated_delivery_date
                                          116581 non-null
                                                           object
       24
           customer_unique_id
                                          116581 non-null
       25 customer_zip_code_prefix
                                          116581 non-null
       26 customer city
                                          116581 non-null
       27 customer_state
                                          116581 non-null
                                                           object
           product_category_name
                                          116581 non-null
                                                           object
           product_name_lenght
                                                           float64
                                          116581 non-null
          product_description_lenght
                                          116581 non-null
                                                           float64
       31 product_photos_qty
                                          116581 non-null
                                                           float64
       32
           product_weight_g
                                          116580 non-null
                                                           float64
       33
           product_length_cm
                                          116580 non-null
                                                           float64
           product height cm
                                          116580 non-null
                                                           float64
           product_width_cm
                                          116580 non-null
                                                           float64
           seller_zip_code_prefix
                                          116581 non-null
                                                           int64
       37
           seller_city
                                          116581 non-null
          seller_state
                                          116581 non-null
                                                           object
       39 product_category_name_english 116581 non-null
      dtypes: float64(10), int64(6), object(24)
      memory usage: 35.6+ MB
```

NaN Values & Duplicated

```
df.isnull().sum()
order id
order_item_id
product id
seller id
shipping limit date
price
freight_value
payment_sequential
payment_type
payment_installments
payment value
review id
review_score
review_comment_title
                                 102585
review comment message
review_creation_date
review_answer_timestamp
customer id
order_status
                                      0
order_purchase_timestamp
                                      0
order_approved_at
                                     14
order_delivered_carrier_date
                                    1213
order_delivered_customer_date
                                   2515
order_estimated_delivery_date
                                      0
customer_unique_id
customer_zip_code_prefix
                                      0
customer_city
customer_state
product_category_name
product_name_lenght
product description lenght
product_photos_qty
product_weight_g
product_length_cm
product_height_cm
product_width_cm
seller_zip_code_prefix
seller city
seller_state
product_category_name_english
dtype: int64
```

```
| df[df.duplicated()]
| # No Duplicated |
| order_id order_item_id product_id seller_id shipping_limit_date price freight_value payment_sequential payment_type payment_installments payment_value |
| payment_value | payment_type | payment_installments | payment_value | payment_value | payment_value | payment_type | payment_installments | payment_value | payment_value | payment_type | payment_installments | payment_value | payment_value | payment_type | payment_installments | payment_value | payment_value | payment_type | payment_value | payment_value
```

Data Preprocessing:

- Data Preprocessing:
- 1-Handle missing values, remove duplicates, and address any data inconsistencies.
- 2-Engineer time-based features (e.g., day of the week, month, seasonality, promotional
- periods).
- 1-Handle missing values, remove duplicates, and address any data inconsistencies

Handle the Dates

```
df['date of shipping_limit'] = pd.to_datetime(df['shipping_limit_date']).dt.date

df['day shipping_limit']=pd.to_datetime(df['date of shipping_limit']).dt.day

df['month shipping_limit']=pd.to_datetime(df['date of shipping_limit']).dt.month

df['year shipping_limit']=pd.to_datetime(df['date of shipping_limit']).dt.year
```

```
dff day of review creation ] -pd.to_datetime(dff review_creation_date ]).dt.day
dff day of answer for the review creation ] -pd.to_datetime(dff review_creation_date ]).dt.date

dff day of answer for the review creation ] -pd.to_datetime(dff review_creation_date ]).dt.date

dff day of answer for the review creation ] -pd.to_datetime(dff (review_answer_timestamp)).dt.day
dff month of answer for the review creation ] -pd.to_datetime(dff (review_answer_timestamp)).dt.day
dff day of order buy ]-pd.to_datetime(dff (review_answer_timestamp)).dt.day
dff day of approve for the order ]-pd.to_datetime(dff (review_answer_timestamp)).dt.day
dff day of approve for the order ]-pd.to_datetime(dff (review_answer_timestamp)).dt.day
dff day of approve for the order ]-pd.to_datetime(dff (review_answer_timestamp)).dt.day
dff day of deliver for the shipping company ]-pd.to_datetime(dff (review_answer_timestamp)).dt.date

dff day of deliver for the shipping company ]-pd.to_datetime(dff (review_answer_timestamp)).dt.date

dff day of deliver for the shipping company ]-pd.to_datetime(dff (review_answer_timestamp)).dt.date

dff day of deliver for the shipping company ]-pd.to_datetime(dff (review_answer_timestamp)).dt.date

dff day of deliver for customer ]-pd.to_datetime(dff (review_answer_timestamp)).dt.date

dff day of expected to deliver for the customer ]-pd.to_datetime(dff (reder_datetime(dff (reder_datetime(dff (reder_datetime(dff (reder_datetime(dff (reder_datetime(dff (reder_datetime(dff (reder_datetime(dff (reder_datetime(dff (reder_datetime(d
```

Handle the data inconsistencies 1

```
df.loc[
    df['order_delivered_customer_date'] < df['order_purchase_timestamp'],
    'order_delivered_customer_date'
] = pd.NaT

df[df['order_delivered_customer_date'] < df['order_purchase_timestamp']]

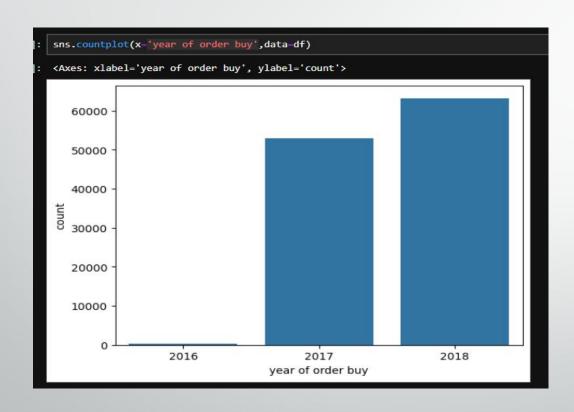
order_id order_item_id product_id seller_id shipping_limit_date price freight_value payment_sequential payment_type payment_installments payment_value re
```

Handle the data inconsistencies 2

Milestone 2: Data Analysis and Visualization

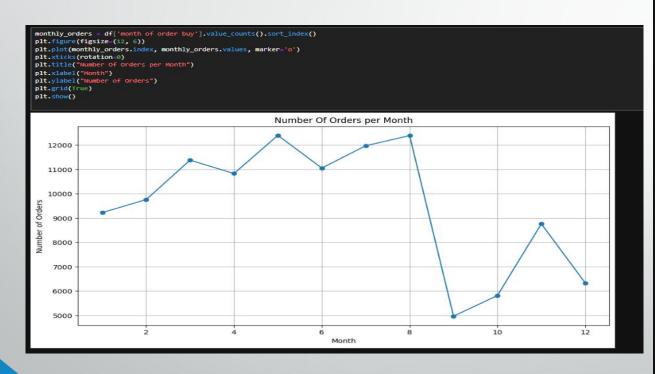
- 1-Data Analysis:
- Perform statistical analysis to identify correlations between sales and other factors such as
- promotions, weather, holidays, and special events.
- Investigate seasonality and long-term trends in sales.
- 2. Data Visualization:
- Create visualizations like line graphs, bar charts, and scatter plots to display sales trends and
- seasonal patterns.
- Develop interactive dashboards (using Plotly or Dash) to allow users to explore trends and
 patterns in the sales data.

Sales Over the Years



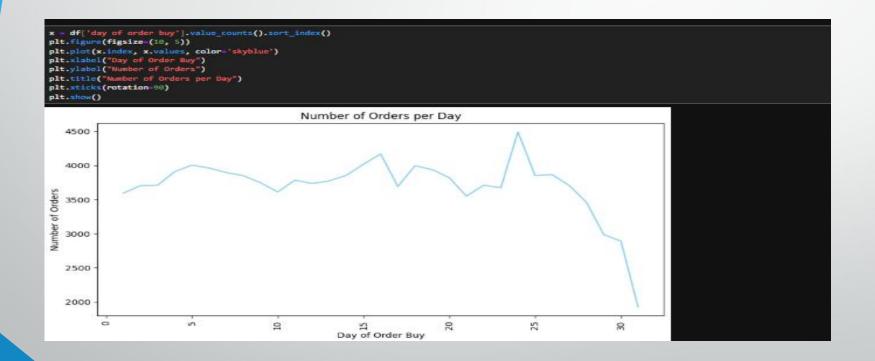
2018 631992017 529942016 388

Sales Over the Months

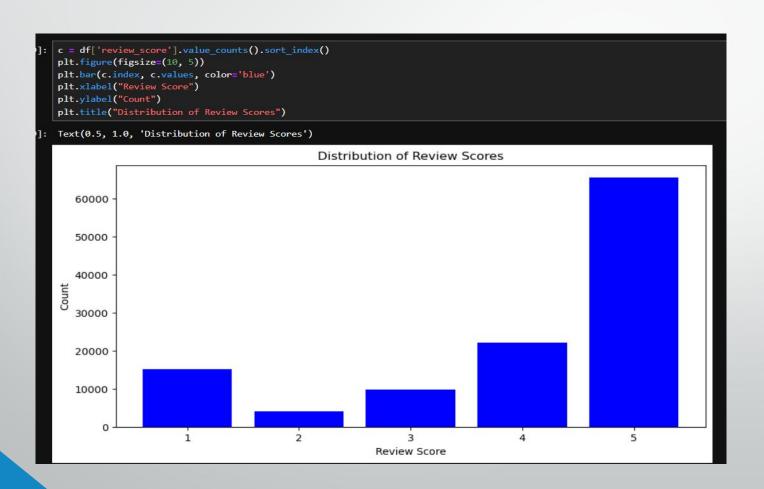


8	12569
5	12517
7	12108
3	11587
6	11128
4	10977
2	9912
1	9392
11	8963
12	6443
10	5926
9	5059

Sales Over the Days



Review Score



Cities with Low Ratings

```
plt.figure(figsize=(12, 6))
plt.bar(grouped_data['customer_city'][:10], grouped_data['count'][:10], color='purple')
plt.xlabel('Customer City')
plt.ylabel('Count of Low Ratings (1 & 2)')
plt.xticks(rotation=45)
plt.title('Top Cities with Low Ratings (1 & 2)')
plt.show()
lowrateC=grouped_data[['customer_city','count']]
                                                      Top Cities with Low Ratings (1 & 2)
   2500
t of Low Ratings (1 & 2)
₫ 1000
                                                                    Customer City
```

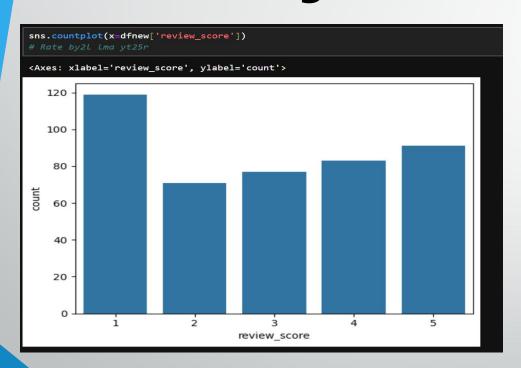
```
filtered_df = df[df['review_score'].isin(x)]
grouped_data = filtered_df.groupby('customer_city').size().reset_index(name='count')
grouped data = grouped data.sort values(by='count', ascending=False)
grouped data.head(10)
     customer city count
         sao paulo 2652
1355 rio de janeiro 1818
 193 belo horizonte 504
           brasilia 424
           salvador 370
       porto alegre 319
          campinas 294
           curitiba 277
            niteroi 224
```

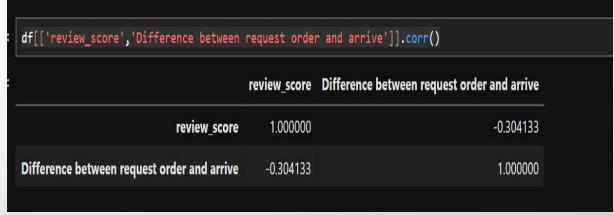
Cities with High Ratings

```
plt.figure(figsize=(12, 6))
plt.bar(grouped_data['customer_city'][:10], grouped_data['count'][:10], color='purple')
plt.xlabel('Customer City')
plt.ylabel('Count of High rates (4 & 5)')
plt.xticks(rotation=45)
plt.title('Top Cities with Low Ratings (4 & 5)')
                                                      Top Cities with Low Ratings (4 & 5)
  14000
   12000
   10000
    8000
of High 1
    6000
    4000
   2000
                                                                   Customer City
```

```
y= [4, 5]
filtered_df = df[df['review_score'].isin(y)]
grouped_data = filtered_df.groupby('customer_city').size().reset_index(name='count')
grouped_data = grouped_data.sort_values(by='count', ascending=False)
grouped_data.head(10)
             customer_city count
3335
                 sao paulo 14256
              rio de janeiro 5688
2922
              belo horizonte 2421
                   brasilia 1829
                   curitiba 1379
                  campinas 1273
2745
                porto alegre 1198
                 quarulhos 1049
                   salvador 979
3180 sao bernardo do campo 897
```

This graph shows how an increase in the number of days between order and delivery affects the review rating





Outliers Outliers were detected in both payment value and weight, and they were dropped to ensure data quality and accuracy in analysis.



```
df = df[df['payment_value'] <= 5000]
```

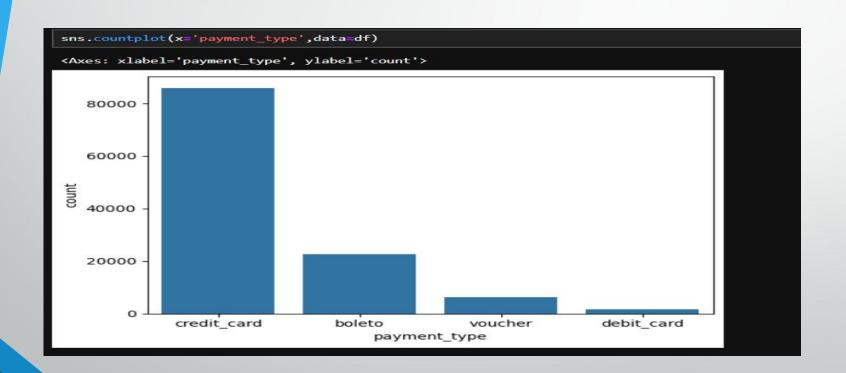
Correlation Analysis Between Freight, Volume, and Weight

```
df[['volume', 'freight_value']].corr()
71]:
                     volume freight value
                   1.000000
                                  0.583976
           volume
      freight_value 0.583976
                                  1.000000
      df[['product_weight_g', 'freight_value']].corr()
74]:
74]:
                        product_weight_g freight_value
      product weight g
                                 1.000000
                                               0.615762
          freight value
                                 0.615762
                                               1.000000
```

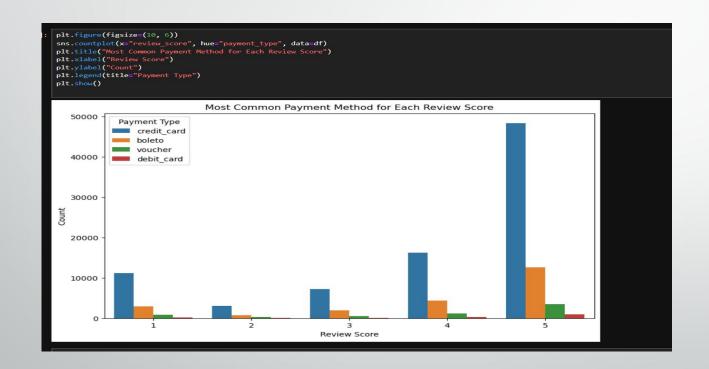
There is a weak correlation between price and both weight and volume

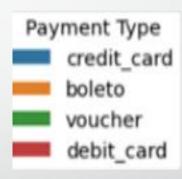
```
df[['product_weight_g', 'price']].corr()
880]:
                          product weight g
                                                price
       product weight g
                                   1.000000
                                            0.343402
                                   0.343402
                   price
                                            1.000000
       df[['volume', 'price']].corr()
879]:
                 volume
                             price
                1.000000
                          1.000000
```

Payment Type



The graph illustrates the relationship between payment methods and product ratings.





The best-selling products.

```
df['product_category_name'].value_counts().head()

product_category_name
cama_mesa_banho 11987
beleza_saude 10025
esporte_lazer 9005
moveis_decoracao 8833
informatica_acessorios 8151
Name: count, dtype: int64
```

The least-selling products.

```
df['product_category_name_english'].value_counts().tail(10)
product_category_name_english
music
                              40
diapers_and_hygiene
flowers
                              33
home comfort 2
                              31
fashion_sport
                              31
arts_and_craftmanship
                              24
la_cuisine
                             16
cds_dvds_musicals
                             14
fashion_childrens_clothes
                              8
security and services
Name: count, dtype: int64
```

Sales across seasons.

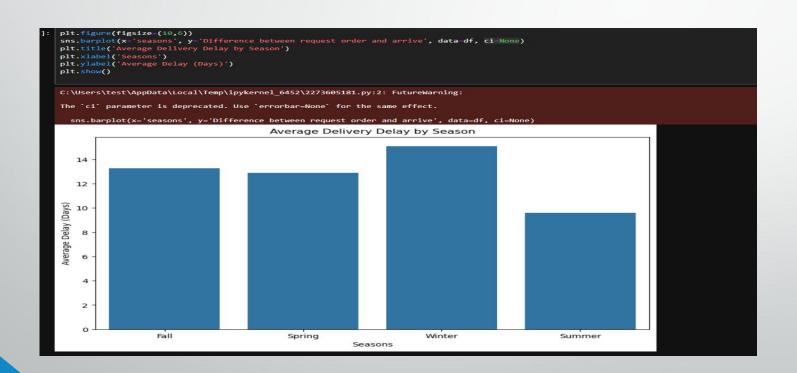


```
condition=[(df['month of order buy'].isin([12,1,2])),(df['month of order buy'].isin([3,4,5])),(df['month of order buy'].isin([6,7,8])),(df['month of order buy']
```



Winter: 12, 1, 2 Spring: 3, 4, 5 Summer: 6, 7, 8 Fall: 9, 10, 11

Order delivery delays across seasons.



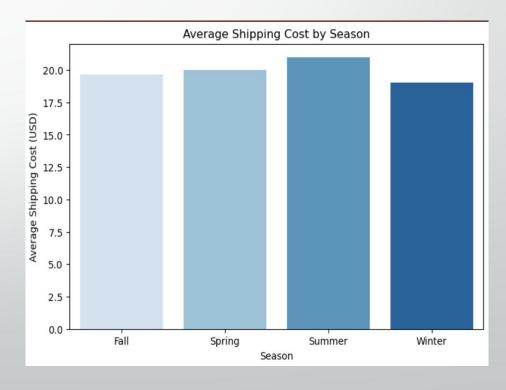
Review ratings across seasons.



Delivery price across seasons.

```
seasonal_shipping_cost = df.groupby("seasons")["freight_value"].mean()
print(seasonal_shipping_cost)

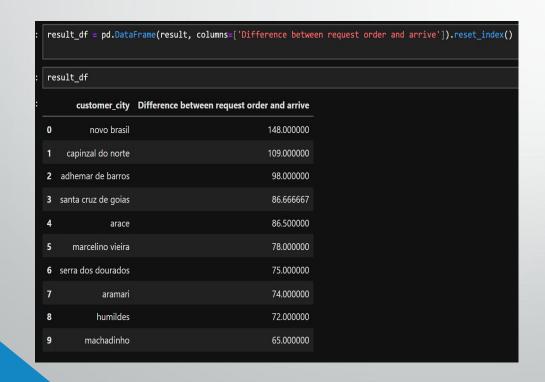
seasons
Fall 19.673781
Spring 19.975340
Summer 20.968218
Winter 19.037837
Name: freight_value, dtype: float64
```

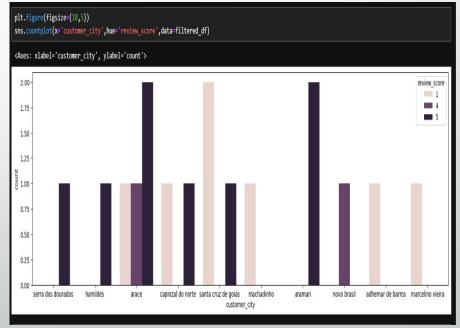


Best-selling and least-selling products across seasons.

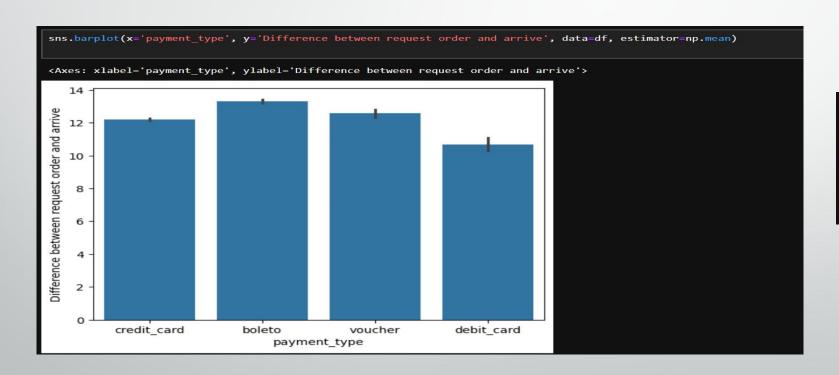
```
items0 = df.groupby("seasons")['product_category_name_english'].agg(lambda x: x.value_counts().idxmax())
items0
seasons
Fall
          bed bath table
Spring
          bed bath table
Summer
          bed_bath_table
          bed_bath_table
Name: product_category_name_english, dtype: object
items1 = df.groupby("seasons")['product_category_name_english'].agg(lambda x: x.value_counts().idxmin())
items1
seasons
Fall
              security_and_services
Spring
                  cds dvds musicals
          fashion_childrens_clothes
Summer
                  cds dvds musicals
Name: product_category_name_english, dtype: object
```

Top 10 cities with the most delivery delays along with their review scores.



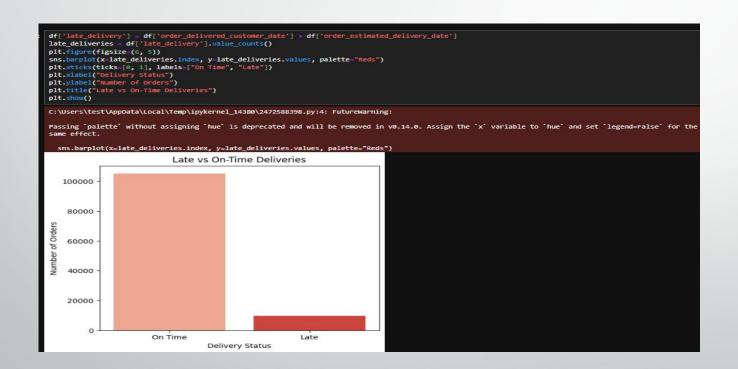


The graph shows which payment method experiences the most delivery delays.



payment_type
credit_card
boleto
voucher
debit_card

Comparison Between Late and On-Time Deliveries



- Milestone 3: Forecasting Model Development and Optimization
- Extract features and target variable from the DataFrame

Build The Model 1-Choose The Features

```
remails the outliers
df=df.query("product_weight_g < 15000")
df=df.query("freight_value < 100")
numerical_cols = [
                                                                                                                                               同个シュー
    "payment_sequential",
"payment_installments",
    "product_description_lenght",
"product_photos_qty",
"product_weight_g",
    "product_length_cm",
"product_height_cm",
categorical_cols = [
    "seller_city",
x_num_df = df[numerical_cols].copy()
valid_rows = (x_num_df >= 0).all(axis=1)
df_valid = df[valid_rows].copy()
x_num_df_valid = x_num_df[valid_rows].copy()
x_num_df_log = np.log1p(x_num_df_valid)
x_catboost = pd.concat([x_num_df_log, df_valid[categorical_cols]], axis=1)
y=df['price']
```

2-Apply Log Transformation to avoid the skewness



3-Split thd data

```
from sklearn.model_selection import train_test_split
xtrain_cat, xtest_cat, ytrain_cat, ytest_cat = train_test_split(x_catboost, y_log, random_state=0, test_size=0.2)
cat features indices = [x catboost.columns.get loc(col) for col in categorical cols]
df.isnull().sum()
order id
                                              0
order item id
product id
seller_id
shipping_limit_date
Difference between request order and arrive
volume
days of answer for review
seasons
late delivery
Length: 75, dtype: int64
```

4-Make RobustScaler

```
from sklearn.preprocessing import RobustScaler
xtrain_num = xtrain_cat[numerical_cols]
xtest_num = xtest_cat[numerical_cols]
xtrain_cat_part = xtrain_cat[categorical_cols]
xtest_cat_part = xtest_cat[categorical_cols]

scaler = RobustScaler()
xtrain_num_scaled = scaler.fit_transform(xtrain_num)
xtest_num_scaled = scaler.transform(xtest_num)

xtrain_num_scaled = pd.DataFrame(xtrain_num_scaled, columns=numerical_cols, index=xtrain_num.index)
xtest_num_scaled = pd.DataFrame(xtest_num_scaled, columns=numerical_cols, index=xtest_num.index)
xtest_num_scaled = pd.Concat([xtrain_num_scaled, xtrain_cat_part], axis=1)
xtest_final = pd.concat([xtrain_num_scaled, xtest_cat_part], axis=1)
```

5-Use the Model "CATBOOSTING"

```
from catboost import CatBoostRegressor
from sklearn.metrics import r2_score

cat_model = CatBoostRegressor(verbose=0)
cat_model.fit(xtrain_final, ytrain_cat, cat_features=cat_features_indices)
y_pred_cat = cat_model.predict(xtest_final)
r2_cat = r2_score(ytest_cat, y_pred_cat)
r2_cat = round(r2_cat, 2) * 100
print("CatBoost Test R2 Score:", r2_cat, "%")

acc_cat = round(r2_score(ytrain_cat, cat_model.predict(xtrain_final)), 2) * 100
print("CatBoost Train R2 Score:", acc_cat, "%")

Loading widget...
CatBoost Test R2 Score: 78.0 %
CatBoost Train R2 Score: 82.0 %
```

• Deployment

Deploy the model using Streamlit

```
# using streamlit
import pickle
pickle.dump(cat_model,open(r"C:\Users\test\DEPI Project\Project DEPI.pkl", "wb"))
```

The interface

