Machine Learning & Data Science

C1S1

Group Members:

Xinqing Liu 5591405
Jiahui Zeng 2232824
Yudie Shi 5517759
Yuxin Yang 5545226
Chenyi Wu 5520790
Zijun Wu 5555118

PART 1: Introduction

Background information

&

Data cleaning

PART 1 Introduction

predictive analytics: forecasting customer review

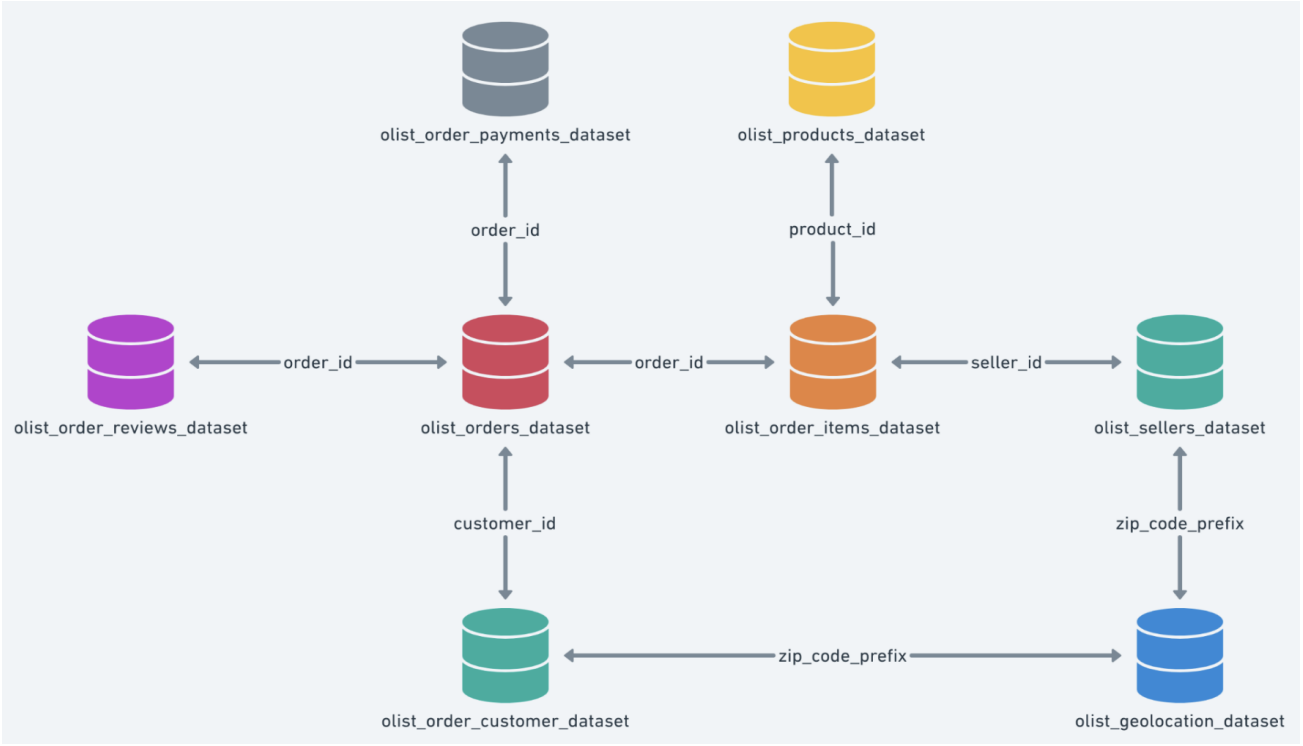
identify those predisposed to leaving high ratings

launch targeted email campaigns

encouraging them to share their positive experiences

A surge in favorable reviews, bolstering the brand's credibility and driving future sales.

PART 1 Data Cleaning



- **Raw data :** 8 charts related to review score directly

- **Methods of merge:** Using "Outer" to merge all charts in case of missing values

- **Methods of handling duplicated & missing values:**

  As we have enough data (above 100000 samples), we drop rows with missing & duplicated values directly

- **Data cleaning process:**

Drop irrelevant columns of each chart ⇨ Clean the data of charts one by one ⇨ When finished generating features, test data again

- **Missing value and Duplicated value Statistics:**

Finally, we dropped 15371 missing values and 0 duplicated values

PART 2: Featuring Engineer & EDA

Generate features

&

Descriptive analysis, Correlation analysis, Scaling data

PART 2 Featuring Engineer | Drop obvious irrelevant features

Delete **9** features that are irrelevant to the predicted value:

review_comment_message

product_weight_g

product_length_cm

product_width_cm

product_height_cm

seller_city

payment_sequential

review_comment_title

review_comment_message

PART 2 Featuring Engineer | Generate new features

- **1. Distance: Measuring the distance between sellers and buyers**

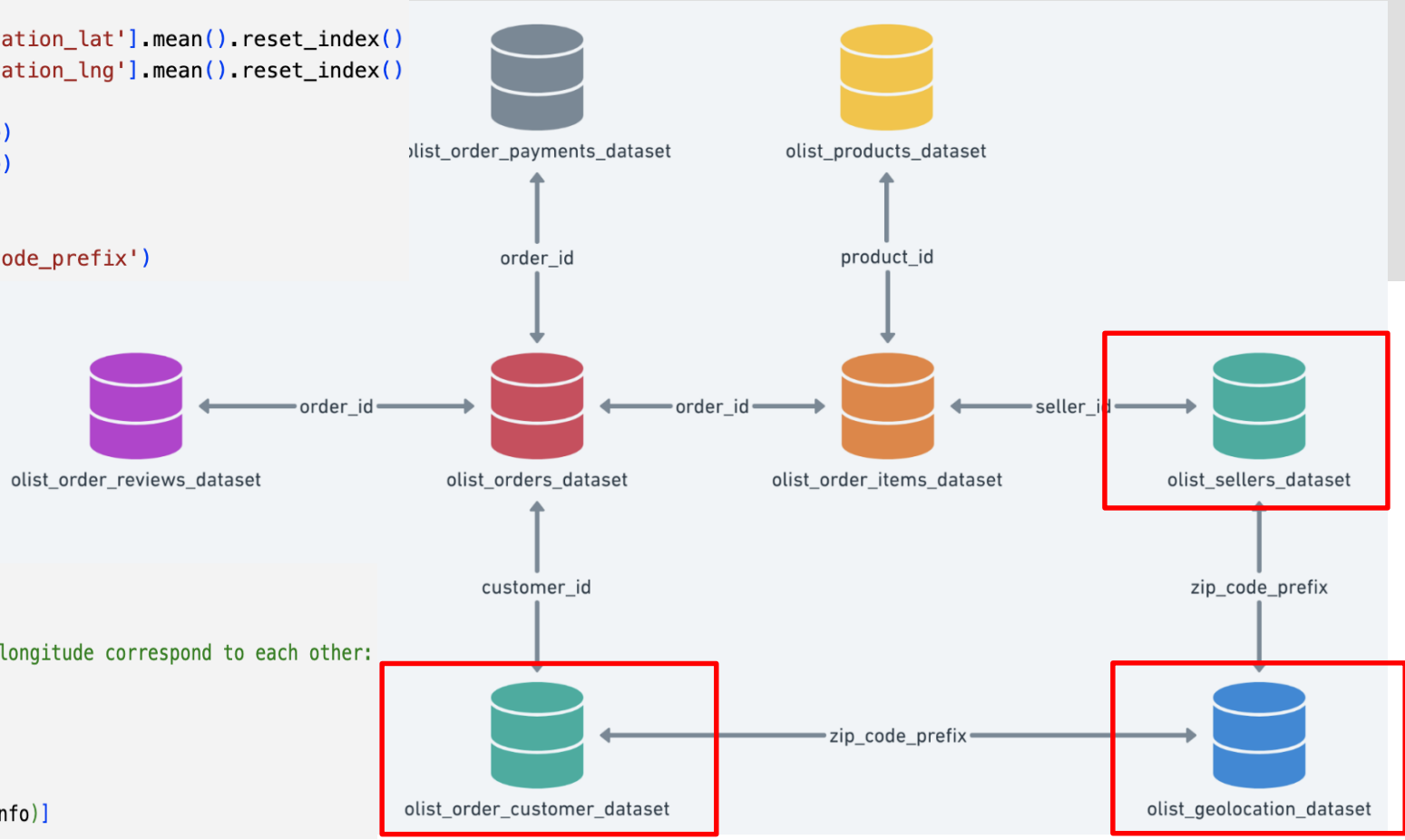
Assumption: Customers may be more inclined to trust local businesses and give them higher score

Use zip code to group and calculate sellers and buyers' average value of latitude and longitude:

```
# calculate mean_lat and mean_lng based on geolocation_zip_code_prefix

mean_lat = geolocation.groupby('geolocation_zip_code_prefix')['geolocation_lat'].mean().reset_index()
mean_lng = geolocation.groupby('geolocation_zip_code_prefix')['geolocation_lng'].mean().reset_index()

mean_lat.rename(columns={'geolocation_lat': 'mean_lat'}, inplace=True)
mean_lng.rename(columns={'geolocation_lng': 'mean_lng'}, inplace=True)

#creat new dataframe to put mean_let and mean_lng
geolocation_mean = pd.merge(mean_lat, mean_lng, on='geolocation_zip_code_prefix')
```

Kostya Esmukov (2023) proposed a method to calculate distance by using *geodesic function* :

```
from geopy.distance import geodesic

# Using tuple to get data by using zip function to make sure that the latitude and longitude correspond to each other:
customer_info = list(zip(df['customer_lat'], df['customer_lng']))
seller_info = list(zip(df['seller_lat'], df['seller_lng']))

# calculate distance between sellers and buyers
df['distance'] = [geodesic(c, s).kilometers for c, s in zip(customer_info, seller_info)]
```

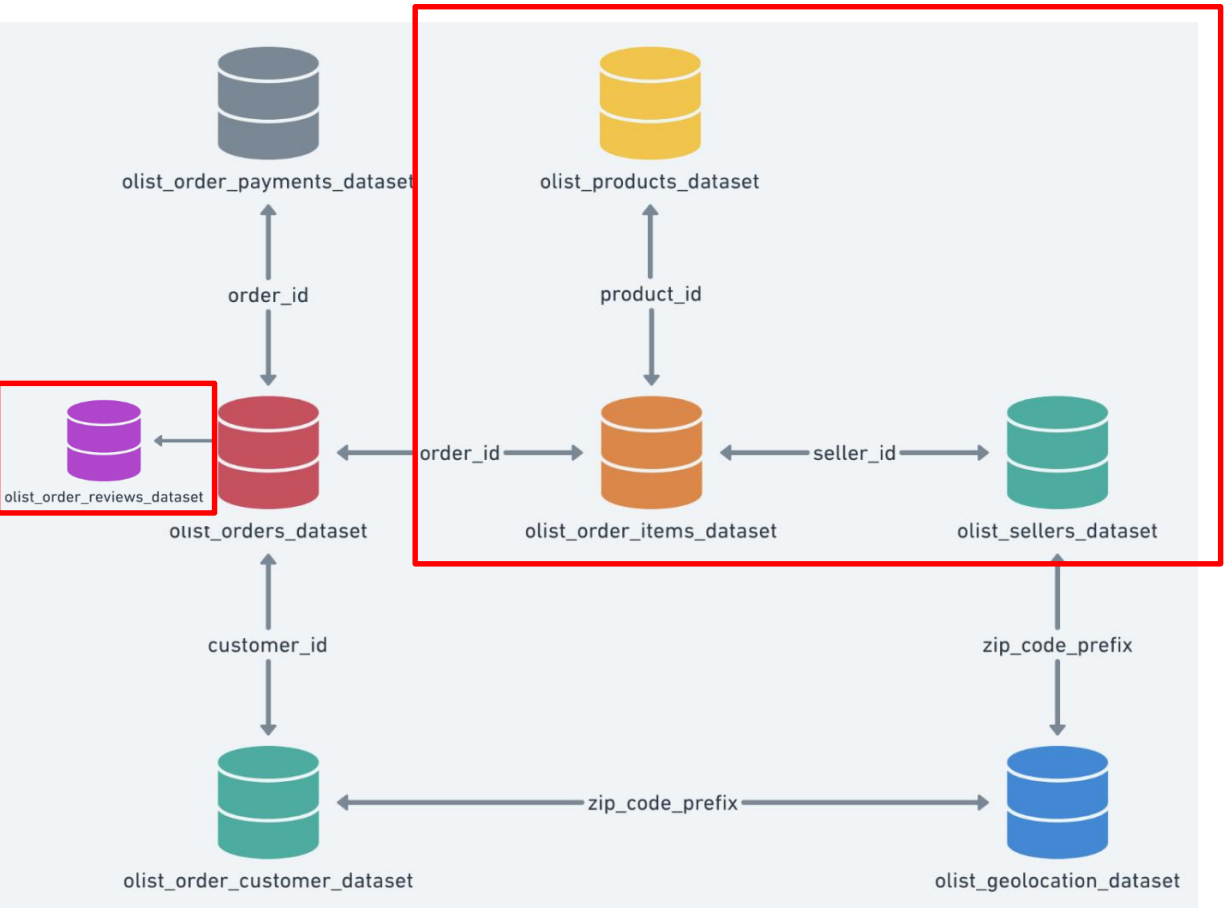


PART 2 Featuring Engineer | Generate new features

- **2 & 3. Review scores from sellers & products: Review scores of the same sellers and products**

Assumption: Characteristics of sellers and Categories of products may affect review score

- **4. Numbers of products sold:** **Assumption: Customers tend to give higher score to sellers who sold more products**



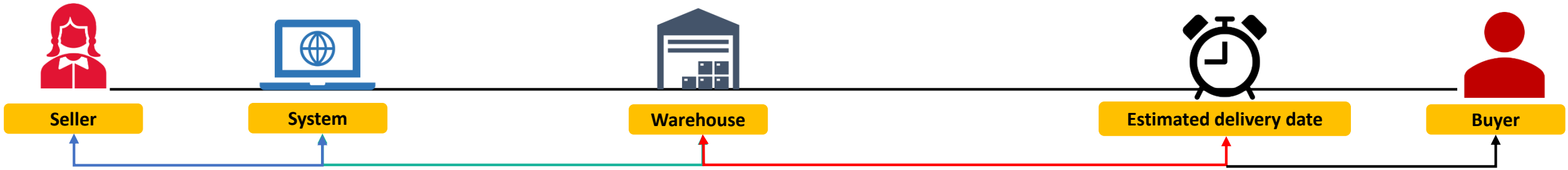
- Aggregate review scores based on seller id and product id, and then use mean value to represent

- Count order id based on seller id and then use count value to aggregate, then we can find how many products sellers have sold

```
# Calculate seller's ratings and product sold counts by using groupby and aggregate function:

seller_grouped = df.groupby('seller_id')[['review_score', 'order_id']].agg({'review_score': 'mean', 'order_id': 'count'})
seller_grouped.rename(columns={'review_score': 'seller_rating', 'order_id': 'product_sold_by_seller'}, inplace=True)

# Calculate products' ratings using groupby and aggregate function:

product_grouped = df.groupby('product_id')['review_score'].mean().reset_index()
product_grouped.rename(columns={'review_score': 'product_rating'}, inplace=True)
```

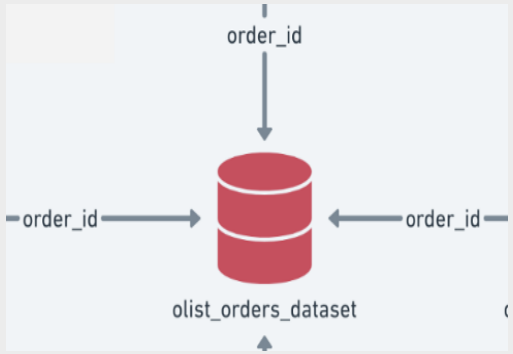
PART 2 Featuring Engineer | Generate new features

Convert types of 'order purchase timestamp', 'order approved at',
'order delivered carrier date', 'order delivered customer date', 'order estimated delivery date' into datetime and use days to calculate



- **5. Response time:** Period between order submission and order confirmation
- **6. Order dispatch time:** Interval between order confirmed and products shipped out from the warehouse
- **7. Shipping time:** Interval between products shipped out and delivered
- **8. Lead time:** Interval between estimated and actual time
- **9. Order delivery time:** Total intervals of buying products

Assumption: Customers hope products can be delivered as quickly as possible, thus 5 features above related to score




```
# Response time of system,:
df["Response_time"] =(df['order_approved_at']-df['order_purchase_timestamp']).dt.days

# Order dispatch time
df["Order_dispatch_time"] =(df['order_delivered_carrier_date']-df['order_approved_at']).dt.days

# Shipping time
df["Shipping_time"] =(df['order_delivered_customer_date']-df['order_delivered_carrier_date']).dt.days

# Order delivery time
df["Order_delivery_time"] =(df['order_delivered_customer_date']-df['order_purchase_timestamp']).dt.days

# Lead time
df["Lead_time"] =(df['order_estimated_delivery_date']-df['order_delivered_customer_date']).dt.days
```

PART 2 Featuring Engineer | Generate new features

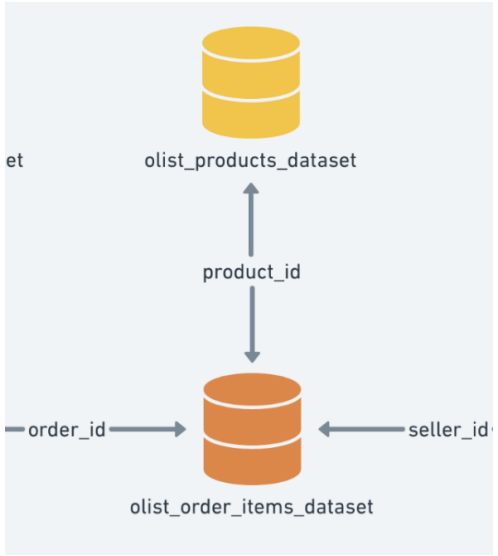
- **10 & 11 Aggregate price and freight value: One Order may conclude many products with various fee**

Assumption: Cheap price and freight fee may result in higher score

- **12 & 13 & 14 Aggregate product_name_lenght,product_description_lenght,product_photos_qty: One Order may conclude many products with various descriptive characteristic**

Assumption: Precise description and efficient display of products may lead higher score

- Aggregate price and freight value based on order id, and then use sum function to represent
- Aggregate products' name length, description length and photos qty based on order id, and then use mean value to represent



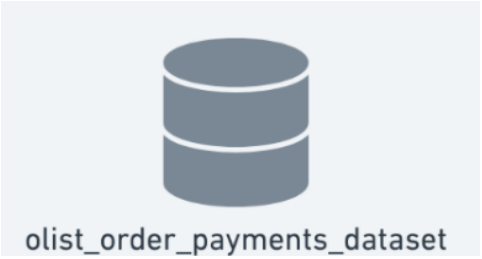

```
## Aggregate price and freight_value by using sum value

price_grouped = df.groupby('order_id')['price'].sum().reset_index()
price_grouped.rename(columns={'price': 'Aggregate_price'}, inplace=True)

df = df.merge(price_grouped, on='order_id', how='left')
df = df.drop('price',axis = 1)

freight_value_grouped = df.groupby('order_id')['freight_value'].sum().reset_index()
freight_value_grouped.rename(columns={'freight_value': 'Aggregate_freight_value'}, inplace=True)
```

```
# Aggregate product_name_lenght,product_description_lenght,product_photos_qty by using mean value

grouped = df.groupby('order_id').agg({'product_name_lenght': 'mean',
                                      'product_description_lenght': 'mean',
                                      'product_photos_qty':'mean'}).reset_index()

grouped.rename(columns={'product_name_lenght': 'Aggregate_product_name_lenght',
                        'product_description_lenght': 'Aggregate_product_description_lenght',
                        'product_photos_qty':'Aggregate_product_photos_qty'}, inplace=True)
```

PART 2 Featuring Engineer | Generate new features

- **15. Use Discount: Generate categorical variable by using ”Payment method”**

Assumption: Customers use ‘voucher’ means having discount, which may encourage customers to give high score



Definition

When customers redeem a voucher, record it as '1'; otherwise, record as '0'

How to set

Step 1: Count the number of voucher and aggregate them according to the order id

Step 2: Number is > = 1: Use voucher, or number is equal to 0: No discount

- **16. Payment speed : Generate categorical variable by using numbers of installment**

Assumption: Customers who have strong purchase power and pay installment quickly tend to more satisfied with products

Use Discount

```
# define a function to check whether voucher is contained
def contains_word(s,word):
  return 1 if word in s.lower() else 0
# word to check
word_to_check = 'voucher'

order_payments['ContainsWord'] = order_payments['payment_type'].apply(contains_word,args=(word_to_check,))

# droup the deplicate order id
df_type = order_payments.groupby(['order_id'],as_index=False)['ContainsWord'].sum()
df_type

# Use Discount or not: 0 = not use; 1 = used
df_type['Use_Discount'] = [1 if value > 0 else 0 for value in df_type['ContainsWord']]
df_type = df_type.drop(['ContainsWord'], axis=1)
```

Payment speed

```
[ ] # If payment_installments < 2, it shows a strong paying ability, so we generate a dummy variable of installment

    df['payment_speed']= [1 if value < 2 else 0 for value in df['payment_installments']]
```

Definition

When customers’ finish payment within 1 period, record it as

‘1’; Otherwise, record as '0'

How to set

Step 1: Aggregate installment based on order id, and then use max value to represent

Step 2: Number is < 2: Strong purchase power

Number is > = 2: Normal purchase power

PART 2 Featuring Engineer | Target variable

Target Variable: Review class

Split aggregated review score of the same orders into 2 levels: High score > 3 & Low score < = 3

Step 1: Aggregate review score based on unique order id, and then use mean value to represent

Step 2: Set condition for Aggregate review score:

if aggregate score > 3: record it as 1 ; When score is < = 3, we record it as 0

```
# generate Aggregate score that from the same order:

review_grouped = df.groupby('order_id')['review_score'].mean().reset_index()
review_grouped.rename(columns={'review_score': 'Aggregate_score'}, inplace=True)

df = df.merge(review_grouped, on='order_id', how='left')

# generate review_class: if aggregate_score is larger than 3, we give it 1, on the other hand, we give it 0
df['review_class']= [1 if value > 3 else 0 for value in df['Aggregate_score']]
df = df.drop(df[['review_score','Aggregate_score']],axis = 1)
```

PART 2 Featuring Engineer | Descriptive Analysis

- **Data description:**

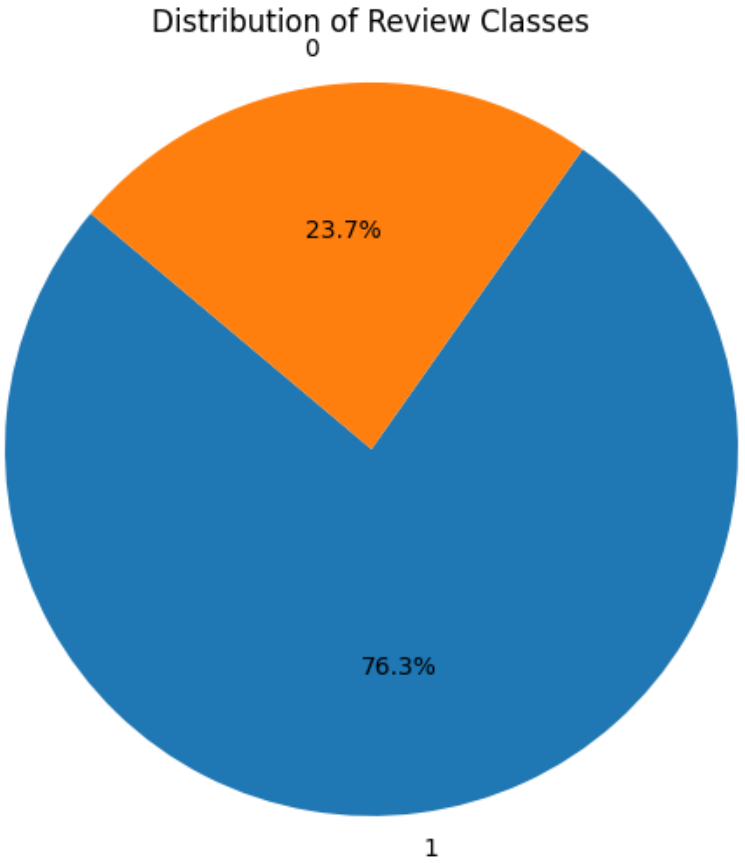
This presentation provides an overview of the descriptive statistics for various numerical variables in our dataset.

Key insights from the descriptive statistics:

1. Prices range widely, from **0.85** to **13440**, with a mean of **180.52**.
2. Product description lengths vary significantly, with a mean of **785.23** characters.
3. Sellers and products receive an average rating of **4 out of 5**, indicating consistent satisfaction among customers.

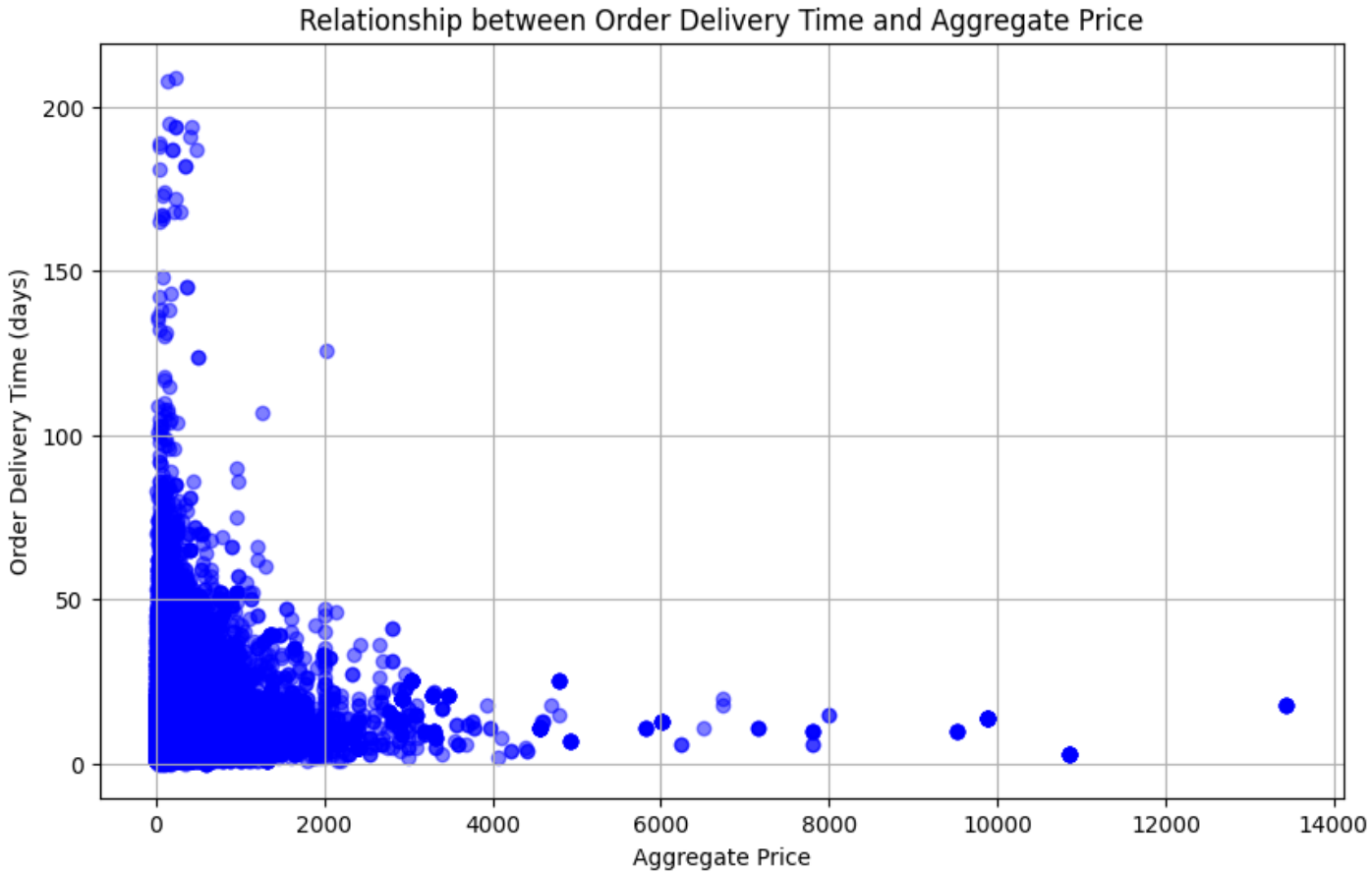
	count	mean	std	min	25%	50%	75%	max
seller_zip_code_prefix	102228.0	24523.062498	27636.309473	1001.00	6429.000000	13720.000000	28470.000000	99730.000000
customer_zip_code_prefix	102228.0	34987.092039	29812.232075	1003.00	11320.000000	24230.000000	58195.000000	99980.000000
distance	102228.0	595.659889	585.641482	0.00	188.559827	432.701098	790.060720	8652.125673
seller_rating	102228.0	4.028432	0.382844	1.00	3.857036	4.053546	4.231939	5.000000
product_sold_by_seller	102228.0	412.912138	542.693616	1.00	56.000000	167.000000	511.000000	1933.000000
product_rating	102228.0	4.044324	0.863000	1.00	3.750000	4.142857	4.666667	5.000000
Response_time	102228.0	0.270151	0.751339	0.00	0.000000	0.000000	0.000000	30.000000
Order_dispatch_time	102228.0	2.342881	3.588787	−172.00	0.000000	1.000000	3.000000	107.000000
Shipping_time	102228.0	8.734505	8.561335	−8.00	4.000000	7.000000	11.000000	205.000000
Order_delivery_time	102228.0	12.006036	9.392568	0.00	6.000000	10.000000	15.000000	209.000000
Lead_time	102228.0	11.080272	10.115455	−189.00	6.000000	12.000000	16.000000	146.000000
Aggregate_price	102228.0	180.521350	394.367887	0.85	49.900000	99.000000	179.800000	13440.000000
Aggregate_freight_value	102228.0	32.354353	52.709818	0.00	14.750000	18.680000	33.420000	1794.960000
Aggregate_product_name_lenght	102228.0	48.791955	9.929973	5.00	42.333333	52.000000	57.000000	76.000000
ggregate_product_description_lenght	102228.0	785.235180	647.780909	4.00	348.000000	600.000000	985.000000	3992.000000
Aggregate_product_photos_qty	102228.0	2.199808	1.697880	1.00	1.000000	1.500000	3.000000	20.000000
payment_value	102228.0	179.680601	271.161606	9.59	65.500000	113.610000	195.560000	13664.080000
payment_installments	102228.0	2.983146	2.798680	0.00	1.000000	2.000000	4.000000	24.000000
Use_Discount	102228.0	0.073649	0.261200	0.00	0.000000	0.000000	0.000000	1.000000
payment_speed	102228.0	0.488340	0.499866	0.00	0.000000	0.000000	1.000000	1.000000
review_class	102228.0	0.763421	0.424984	0.00	1.000000	1.000000	1.000000	1.000000

PART 2 Featuring Engineer | Review Class



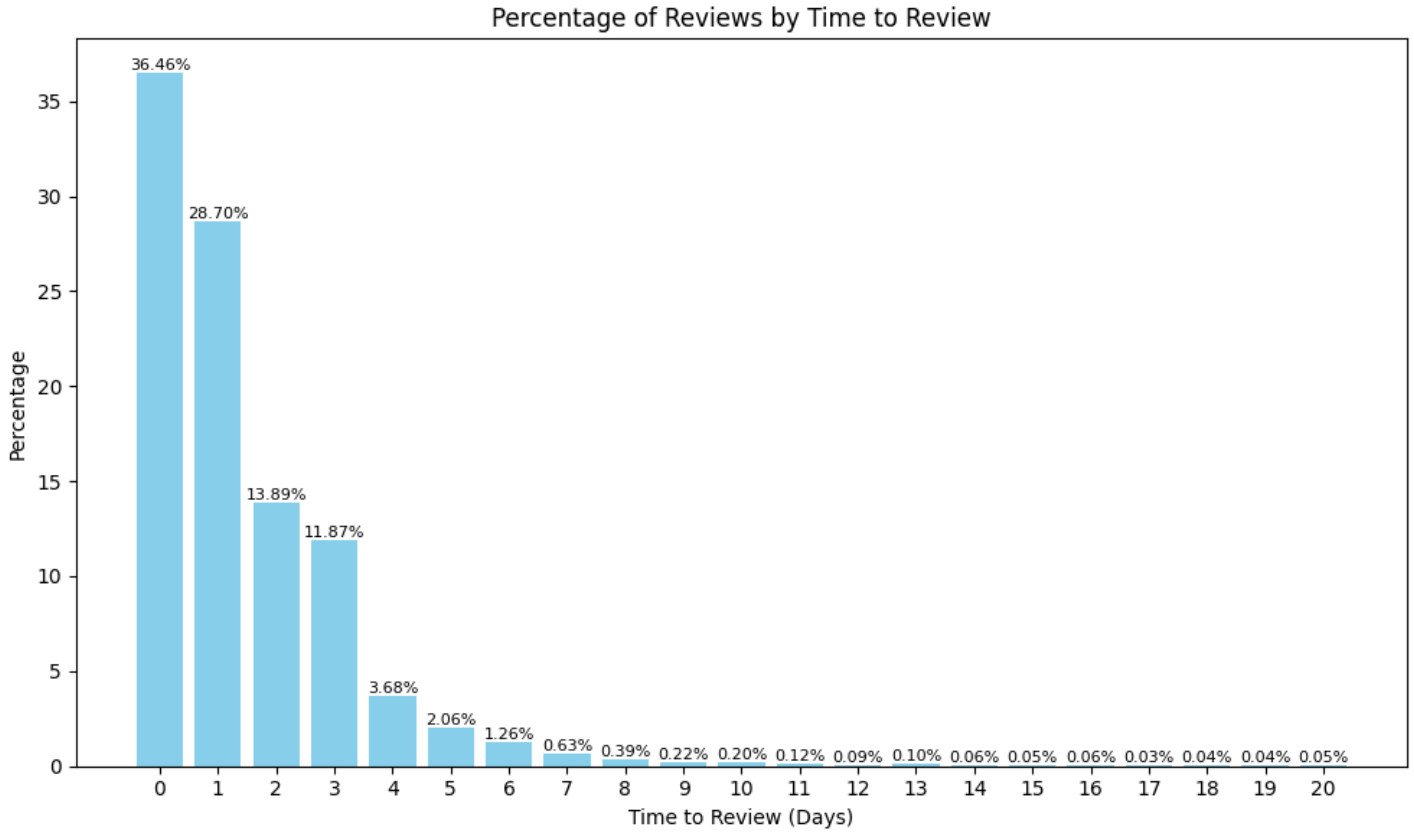
The pie chart shows that the positive class makes up 76.30% of the dataset, while the negative class represents only 23.70%. This suggests that the dataset is unbalanced.

PART 2 Featuring Engineer | Price vs. Order delivery time



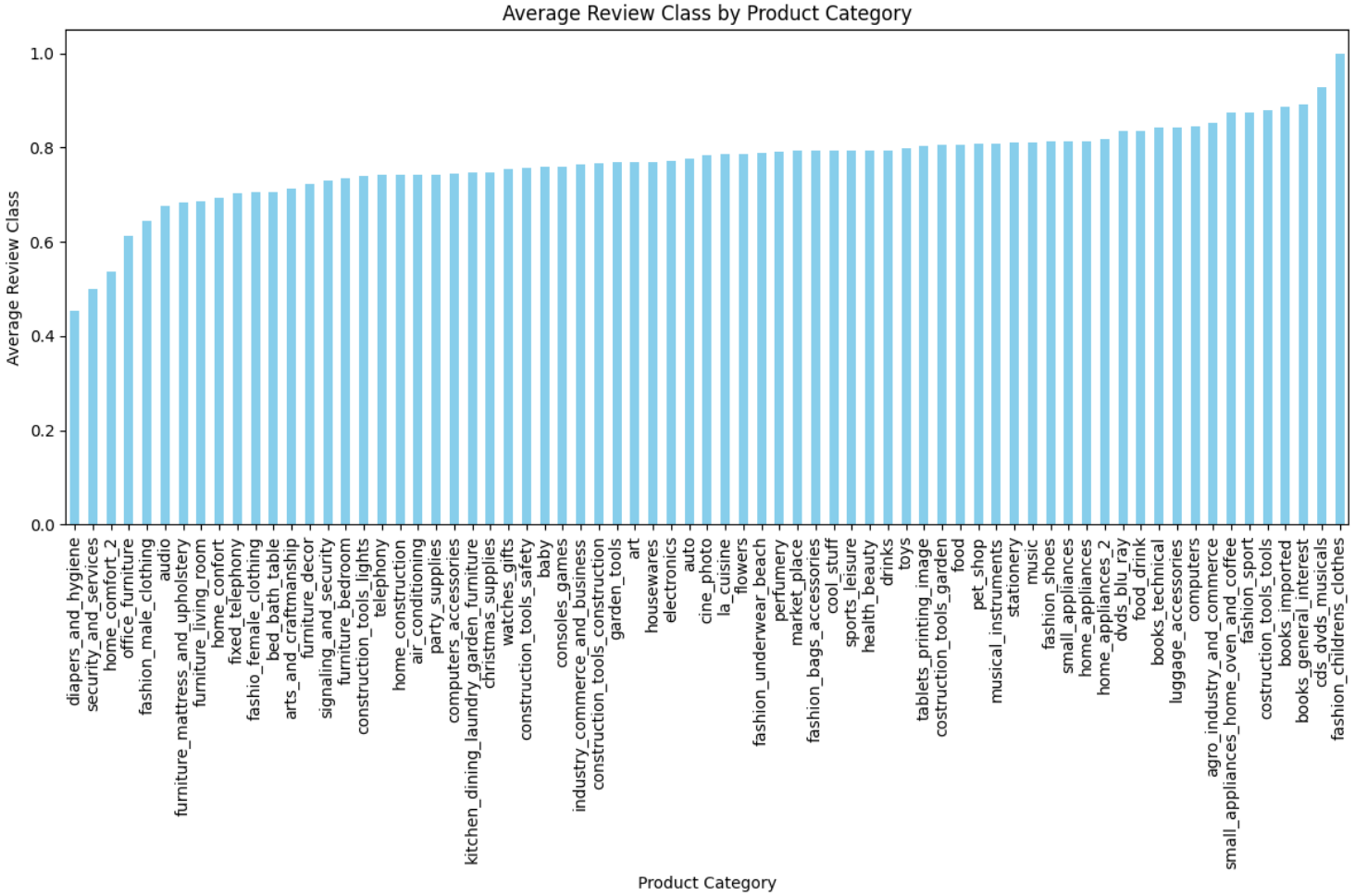
This scatter plot shows that as delivery time increases, customers are more likely to be dissatisfied with the product. However, even with higher product prices, timely delivery doesn't lead to much dissatisfaction.

PART 2 Featuring Engineer | Time for review after invitations



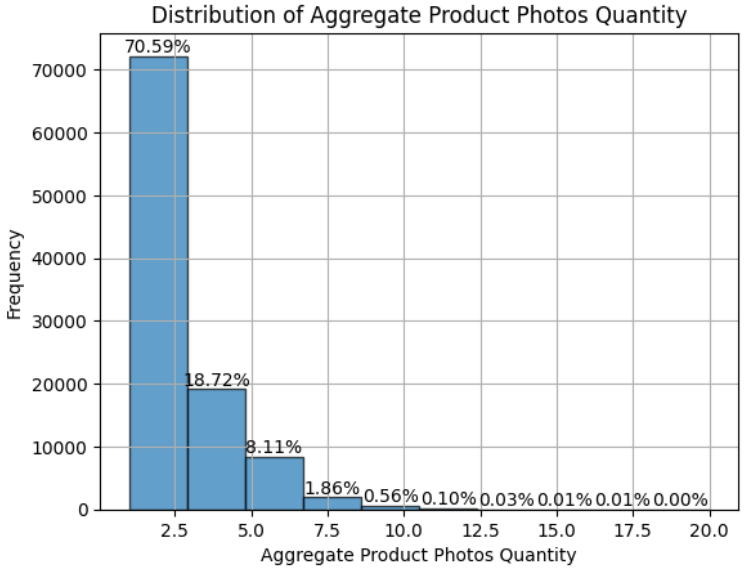
The histogram reveals that 36.46% of users submit reviews within 24 hours of receiving the invitation, while 28.7% do so within one day. This demonstrates the effectiveness of the review invitations in capturing users' interest in providing feedback.

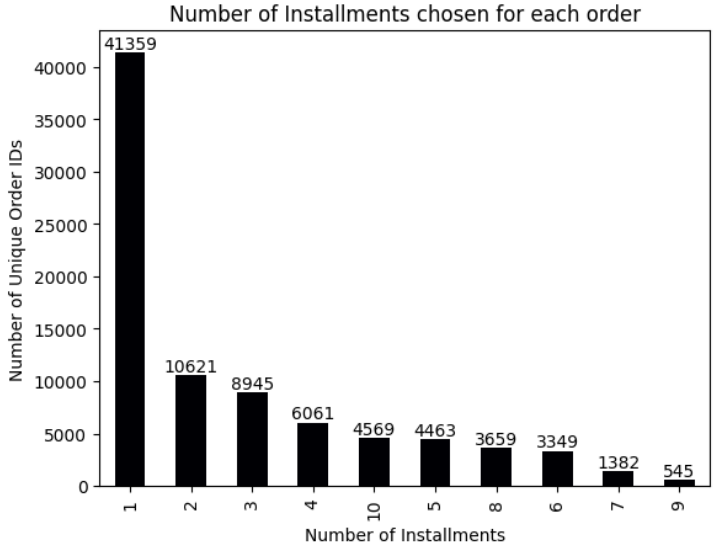
PART 2 Featuring Engineer | Product Category

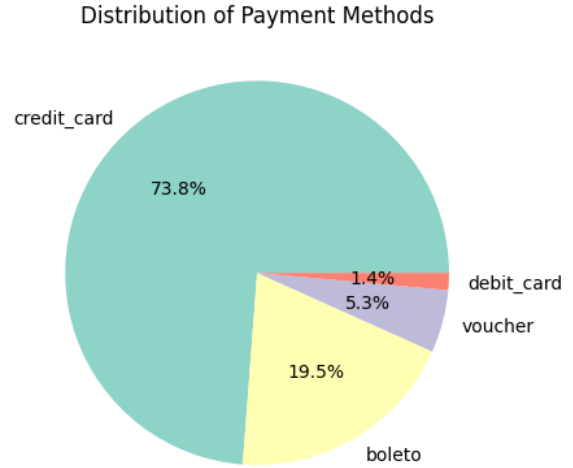


The distribution of review scores across different product categories shows little variation, indicating no significant differences. Therefore, it is concluded that there is a weak correlation between product categories and satisfaction levels.

PART 2 Featuring Engineer | No. of images/installment duration/payment method







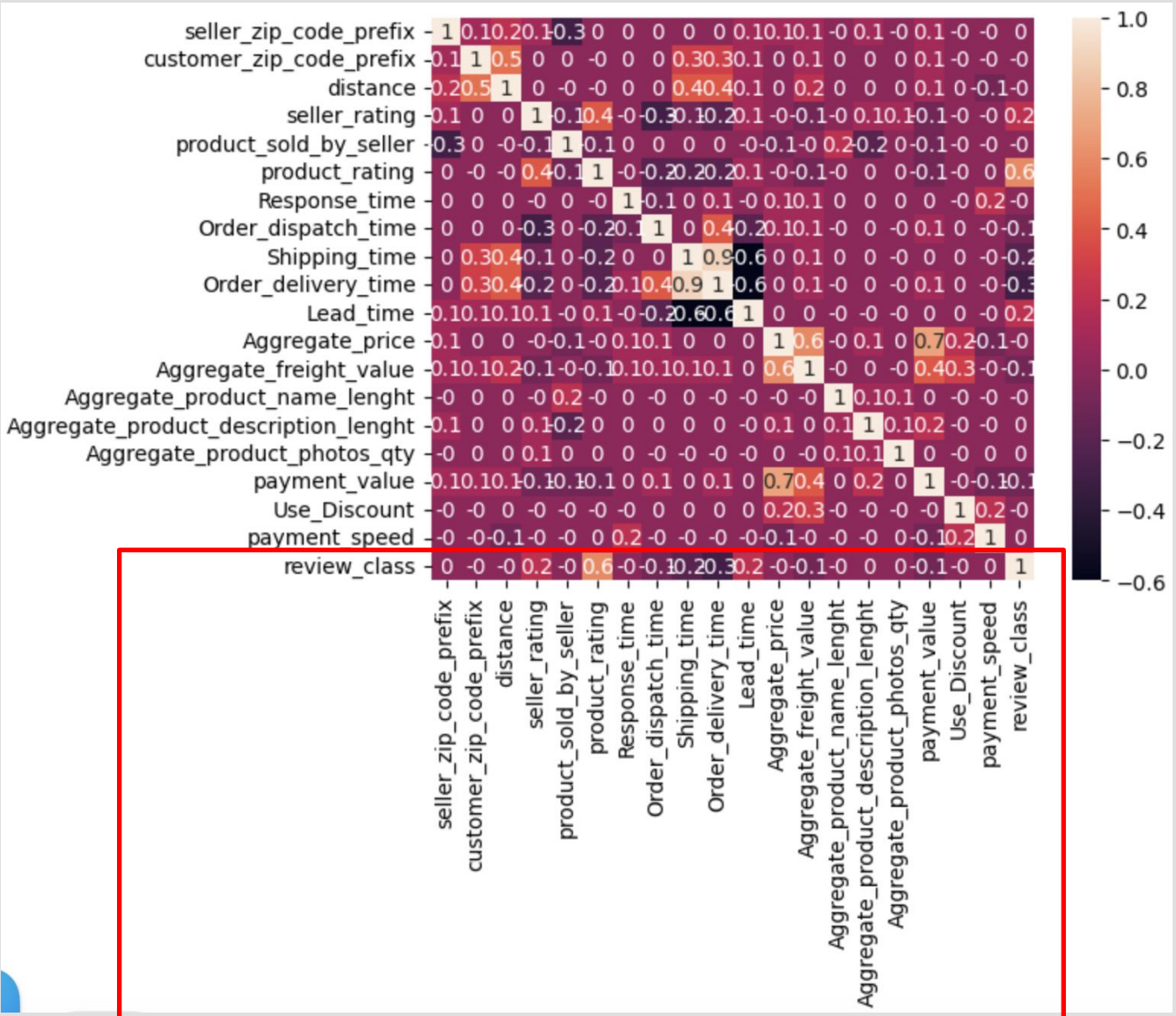
The features of the number of images, installment duration, and payment method show no significant differences across samples, and their distributions are uneven. Therefore, these three features are to be removed.

Additionally, we made the following observations:

1. Nearly half of the products have only one image, suggesting the platform should encourage users to upload more product images to enhance conversion rates.
2. The majority of customers preferred a single payment for each order.
3. Credit cards were the preferred method of payment for the majority of orders.

PART 2 Featuring Engineer | Select features through Correlation analysis

- **Heat map: Using data visualization to display coefficient among features**



Considering autocorrelation:

Order delivery time has high correlation with shipping time

Payment value has high correlation with Aggregate price

Coefficient	Order Delivery time	Shipping time	Aggregate price	Payment value
review class	0.3	0.2	-0.0	-0.1
	Keep	Drop	Drop	Keep

Select features that have obvious effect on review class:

Features	Coefficient
seller rating	0.2
Product rating	0.6
Order dispatch time	- 0.1
Lead time	0.2
Aggregate freight value	- 0.1
Payment value	- 0.1
Order delivery time	- 0.3

PART 2 Featuring Engineer | Scaling data through Robust Scaler method

- **Scaling features which type is float64:**

Purpose : Exclude the effects from extreme values in data set

```
from sklearn.preprocessing import RobustScaler
import pandas as pd

scaler = RobustScaler()

columns_to_scale = df.select_dtypes(include=['float64']).columns

df[columns_to_scale] = scaler.fit_transform(df[columns_to_scale])
```

- **Display clean dataset's first 5 rows:**

	order_id	seller_rating	product_rating	Order_dispatch_time	Order_delivery_time	Lead_time	Aggregate_freight_value	payment_value	review_class
0	00010242fe8c5a6d1ba2dd792cb16214	-0.393450	-0.519481	1.666667	-0.333333	-0.4	-0.288698	-0.318468	1
1	bd31b009e1dbc47fc7c250b1e2cf5440	-0.795521	-1.064935	-0.333333	0.444444	0.0	0.228709	-0.290328	0
2	bb4eb0196897c20281a61f75ce23211c	-0.795521	0.389610	0.333333	0.000000	-0.1	-0.174612	-0.016838	1
3	f9847bf9cc7336c6ba07fe2bdbb6cae1	-0.642533	-0.241969	-0.333333	0.222222	0.0	-0.288163	0.196755	1
4	5c94ad4e194c0e6794688a9d2b9ea94b	0.064125	0.348288	0.000000	2.333333	-2.3	-0.024103	-0.503460	0

Data frame：(102228,9)

PART 3: Model Development

Building Models

&

Evaluation Models' Performance

PART 3 Model Development | Balance the data

- Check the distribution of train set of y

```
[ ] from sklearn.model_selection import train_test_split
    from collections import Counter

    # Check Distribution of 1 and 0 in Y (target value) to make sure our dataset is balanced:

    print("Original Distribution:", Counter(Y_train))
```

```
Original Distribution: Counter({1: 62474, 0: 19308})
```

1 : 0 = 62474 : 19308 not balanced

· The model will pay more attention to categories with a large number,

· resulting in poorer performance in prediction for a smaller number of categories.

- Use the resampled method to do the balance

```
oversampler = RandomOverSampler(random_state=42)
resampled_train_x, resampled_train_y = oversampler.fit_resample(X_train, Y_train)

# Check the distribution:
print("Balanced Distribution:", Counter(resampled_train_y))
```

```
Balanced Distribution: Counter({1: 62474, 0: 62474})
```

Balanced

PART 3 Model Development | Building the models

- Classification Models: due to the business goal and our treatment of Y

- Random Forest Classification

```
# Predictions in trained dataset
rf_classifier = RandomForestClassifier()
rf_classifier.fit(resampled_train_x, resampled_train_y)

# Predictions on test dataset
Y_pred = rf_classifier.predict(X_test)

Y_pred_train = rf_classifier.predict(resampled_train_x)
```

- Logistics Regression

```
# Create Logistics Model
classifier = LogisticRegression()
classifier.fit(resampled_train_x, resampled_train_y)

# Predict test set result
y_pred = classifier.predict(X_test)
y_train_pred = classifier.predict(resampled_train_x)
```

- Support Vector Machine (SVM)

```
from sklearn.svm import SVC
# Create the Model
clf = SVC()
clf.fit(resampled_train_x, resampled_train_y)

# Prediction
predictions = clf.predict(X_test)
Y_pred_train = clf.predict(resampled_train_x)
```

- Voting Classifier—choose Logistics, SVM, Random Forest as individual classifiers

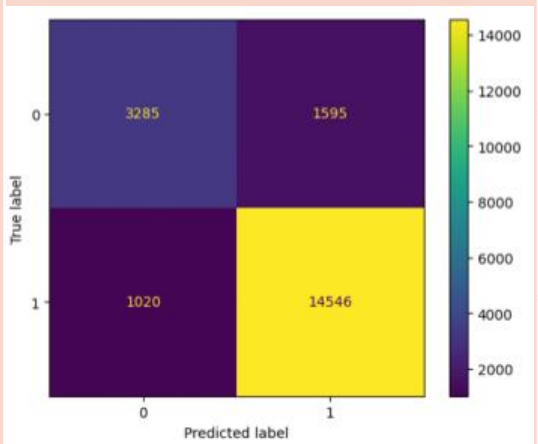
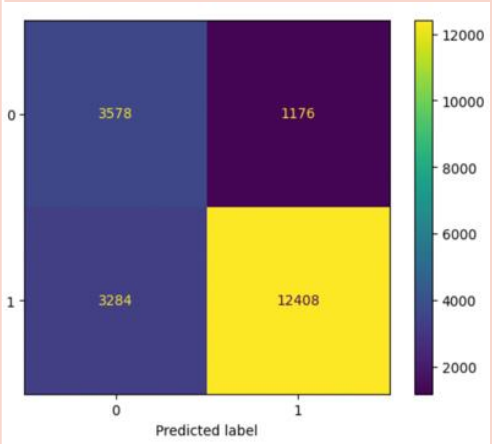
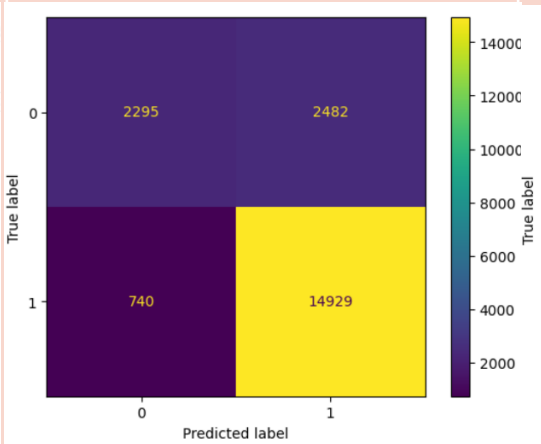
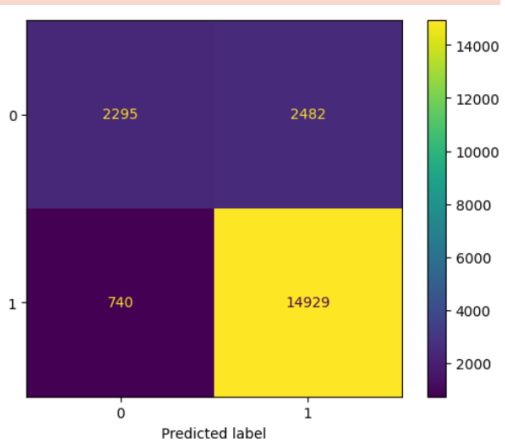
```
# Create three individual classifiers
lr = LogisticRegression()
lr.fit(resampled_train_x, resampled_train_y)
svm = SVC(probability=True, random_state=42)
rf = RandomForestClassifier(n_estimators=50)
rf.fit(resampled_train_x, resampled_train_y)

# Creating VotingClassifier
#——using Logistics, SVM and Random Forest as individual classifiers
voting_clf = VotingClassifier(
    estimators=[('lr', lr), ('svm', svm), ('rf', rf)],
    voting='hard'
)

# train the model
voting_clf.fit(resampled_train_x, resampled_train_y)

# test the model
y_pred_voting = voting_clf.predict(X_test)
```

(Scikit Learn, 2011)

PART 3 Models Development | Evaluation and Comparison

- Index priority in this case: *Precision*
  - to limit the number of false positives
  - A false positive here means those actual low-score customers are predicted to be high-score customers
  - Emails will offend them and may lead to worse reviews
- Secondary Consideration: *Accuracy* — to get more correct predictions

PART 4 Models Development | Evaluation and Comparison

	Random Forest	SVM	Logistics	Voting Classifier
Precision	0.90	0.91	0.86	0.78
Accuracy	0.87	0.78	0.84	0.82
Recall	0.93	0.79	0.95	0.90
F1-score	0.92	0.85	0.90	0.84
Confusion Matrix				

- *Random Forest Model* is selected as the final prediction model in this case

PART 3 Model Development | Optimization

- Default optimal value by Python Library
- Hyper-parameters Optimization

```
rf_classifier = [{'criterion': ['gini', 'entropy'],
                  'max_depth': [3, 5, 7],
                  'min_samples_split': [3, 5, 7],
                  'max_features': ["sqrt", "log2", None]}]
# Training the model on the training set
rf_classifier.fit(resampled_train_x, resampled_train_y)
```

  - The adjusted model did not perform any better than the original
  - Stick to the formal default hyper-parameters value

PART 3 Model Development | Limitations

- Features
  - Some features have a low correlation with Y
  - May explore more features that may be influencers
- F1-score
  - An excessively high F1 score may cause overfitting problems.

  (Manning, Raghavan and Schütze, 2008)

	Random Forest	SVM	Logistics	Voting Classifier
F1-score	0.92	0.85	0.90	0.84

  - May need to use some methods to detect whether there is actual exist an overfitting problem

PART 4: Forecast

Additional indicators selection

&

Further data science project

PART 4 Forecast | Additional indicators for review score prediction model

Return

Returns usually involve a negative experience like receiving damaged products. Therefore, in general, higher return rates are associated with lower customer satisfaction and review score (Vassilyovska, 2021).

Customer service

Customers will contact customer service agents for assistance if they have a question or when they bump into a problem (Morton, 2023)

- Quality of response – communication rating

Some e-commerce platforms request customers to rate their communication with customer service agents, and this rating is a reflection of the customer's satisfaction with the service session.

- response rates
- response times

(Morton, 2023)

PART 4 Forecast | Further data science project

Personalised recommendations

Why

Around **35%** of the Amazon overall revenue comes from product purchases that customers found through recommendations.

How

Make recommendations based on
1. the behaviours of similar users
2. similar products (Mehta, 2023)

Data

Customer demographical characteristics:
age, gender, income level, marital status……
product information:
category, brand, price……

(glood.ai, 2023)

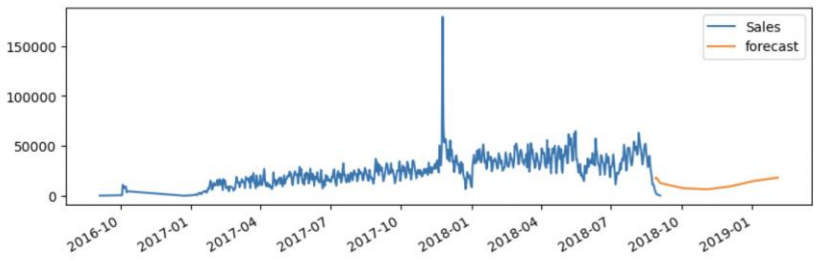
Sales Forecasting

Why

- Identifying early warning signals
- Efficiently allocate resources for future growth and manage its cash flow

How

Forecast based on Time series analysis



(Chakraborty, 2023)

Web traffic monitoring

Why

- Improve website performance by tracking user activity

How

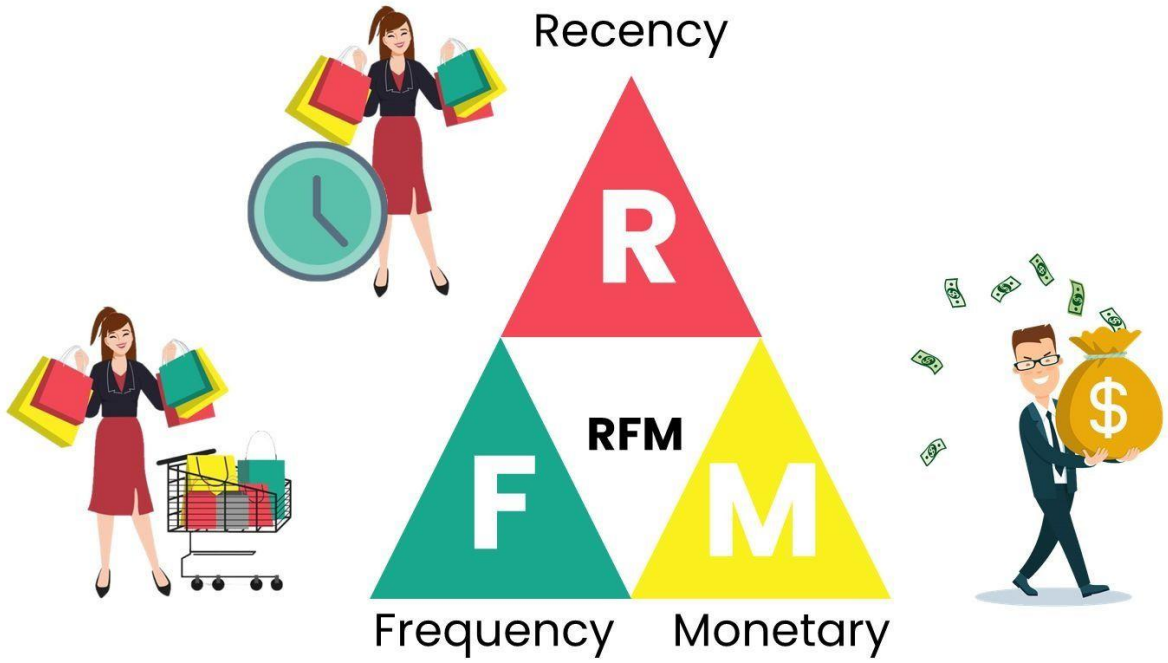
- Collect the number of users that visit a page within a certain period
- Collect the time customers spend on each page and the actions they perform while on a page
- Collect Traffic Sources

(indeed, 2022)

PART 5: Next Project

Customer Segmentation by RFM Analysis

PART 5 Next Project | What is RFM Analysis for Customer Segmentation?

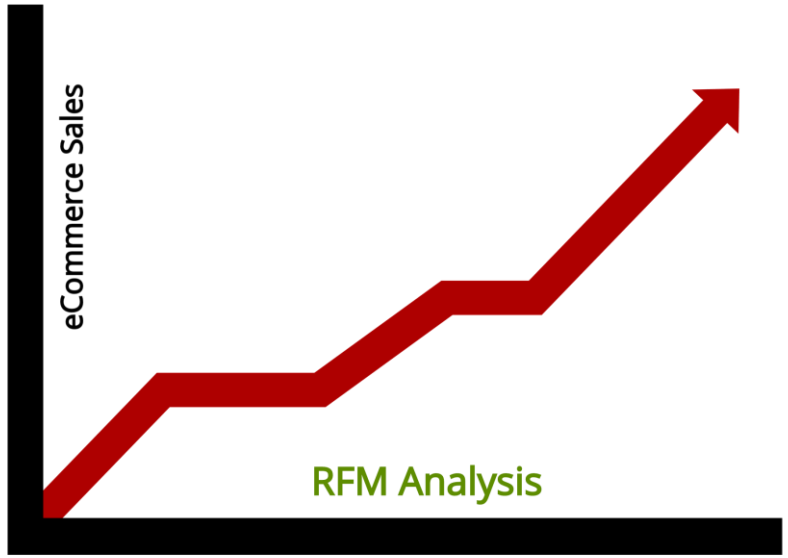


(47Billion, 2021)

RFM ANALYSIS is a method used for analyzing customer value.

This project focuses on doing RFM analysis on a sales dataset and creating a data visualization dashboard showcasing customer segmentation (Fagbenro, 2022).

PART 5 Next Project | Why Use RFM Analysis for Customer Segmentation?

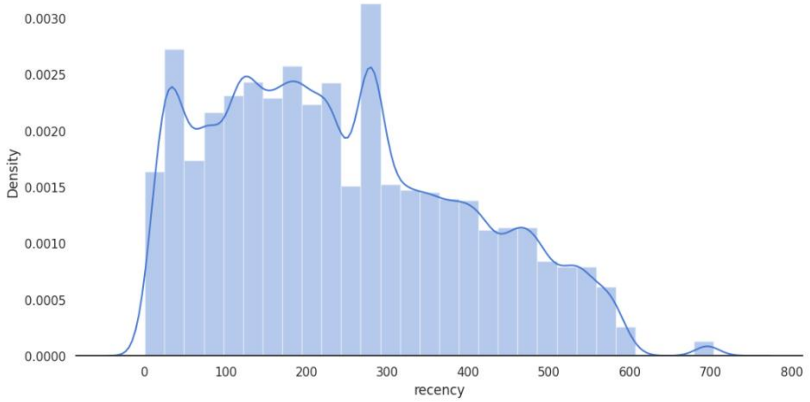


(Singh, 2020)

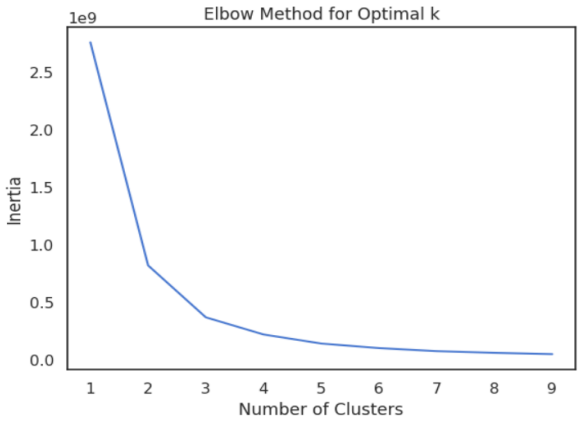
1. Resource Allocation
2. Understand Customer Behaviour
3. Increase Customer Lifetime Value
4. Identify Customer Segments
5. Personalized Marketing Strategies
6. Improve ROI

(Panaitescu, 2023)

PART 5 Next Project | How to do RFM Analysis?



1



2

	count	mean
recencycluster		
0	14743.0	516.283999
1	19613.0	382.171978
2	28142.0	263.975695
3	30600.0	155.705392
4	24503.0	51.995796

3

4

We will use K-means for Cluster Analysis.

1. Data visualization
2. Get elbow points for K-means
3. Get K-means results
4. Create a data visualization dashboard

(Rittman, 2021; Jayheewon, 2023)

Reference List

[1]47Billion (2021) What is RFM (Recency, Frequency, Monetary) Analysis? | HackerNoon. [Accessed 25 February 2024]. Available from: https://hackernoon.com/what-is-rfm-recency-frequency-monetary-analysis.
[2]Chakraborty, S (2023) HPLX_Suraj_Chakraborty_7-9-23_Predictive_Modeling. Available at: https://www.kaggle.com/code/surajchakraborty97/hplx-suraj-chakraborty-7-9-23-predictive-modeling
[3]Esmukov, K (2023) Available at: https://github.com/geopy/geopy?tab=readme-ov-file
[4]Fagbenro, O (2022) Customer Segmentation with RFM Analysis using SQL. *Medium*. [Online]. [Accessed 25 February 2024]. Available from: https://medium.com/@OlorunwaFagbenro/customer-segmentation-with-rfm-analysis-using-sql-b226feadee4e.
[5]glood.ai (2023) 10 Types of Personalized Product Recommendations for Shopify in 2023. Available at: https://glood.ai/blog/10-types-of-product-recommendations-for-shopify
[6]Indeed (2022) What Is Web Traffic Monitoring? Plus Benefits, Tools and Steps. Available at: https://www.indeed.com/career-advice/career-development/web-traffic-monitoring
[7]Jayheewon (2023) RFM: Customer Segmentation by Cluster Analysis. [Accessed 25 February 2024]. Available from: https://kaggle.com/code/jayheewon/rfm-customer-segmentation-by-cluster-analysis.
[8]Manning, C.D., Raghavan, P. & Schütze, H.(2008), Introduction to information retrieval.New York Cambridge University Press.
[9]Mehta, L (2023) How to create personalized product recommendations for your website. Available at: https://abmatic.ai/blog/how-to-create-personalized-product-recommendations-for-website
[10]Morton, H (2023) What Customer Satisfaction indicators should you track?. Available at: https://community.goodays.co/en/blog/what-customer-satisfaction-indicators-should-you-track
[11]Panaitescu, A (2023) A simple guide to RFM Segmentation - Omniconvert Blog. *Omniconvert Ecommerce Growth Blog*. [Online]. [Accessed 25 February 2024]. Available from: https://www.omniconvert.com/blog/rfm-segmentation-guide/.
[12]Rittman, M (2021) RFM Analysis and Customer Segmentation using Looker, dbt and Google BigQuery. *Medium*. [Online]. [Accessed 25 February 2024]. Available from: https://blog.rittmananalytics.com/rfm-analysis-and-customer-segmentation-using-looker-dbt-and-google-bigquery-6f8f97f32ed1.
[13] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[14]Singh, R (2020) How RFM Analysis can Boost e-Commerce Sales -. [Accessed 25 February 2024]. Available from: https://blog.nextbee.com/2020/02/25/how-rfm-analysis-can-boost-e-commerce-sales/.
[15]Vassilyovska, M (2021) How To Measure Customer Satisfaction in the eCommerce Industry. Available at: https://www.kualo.co.uk/blog/how-to-measure-customer-satisfaction-ecommerce

Thank you!

Group Members:

Xinqing Liu 5591405
Jiahui Zeng 2232824
Yudie Shi 5517759
Yuxin Yang 5545226
Chenyi Wu 5520790
Zijun Wu 5555118