ANALYST REPORT

PERQARA ASSESSMENT

ALFITO PUTRA FAJAR PRATAMA

DATE: 29 AUGUST, 2024

PERQARA ASSESMENT

TABLE OF CONTENT

Project Introduction	2
Customer Segmentation	3-18
Clasification	19-22
Advice for Compeny	23

INTRODUCTION

Background

Through customer data analysis, companies can understand consumer behavior and preferences, which in turn can help in designing more effective business strategies. This project aims to segment customer behavior based on their historical data, such as purchase frequency, number of items purchased, total price spent, total payment value, and average review score.

Purpose of the Report

The project was conducted as part of a technical test for a data science position at the company Perqara. Using clustering techniques and other advanced analytics, the project sought to identify distinct customer segments, which the company could use to improve service, personalize offers, and optimize marketing strategies. A classification model was also created to categorize new customers into appropriate clusters.

Scope of the Report

This report describes the steps taken during the analysis process, from data preparation to predictive modeling. Basic and advanced analyses performed include descriptive statistics, outlier detection, clustering and classification according to the available data.

CUSTOMER SEGMENTATION

The first project is to create customer segmentation that aims to see customer purchasing behavior. Customer behavior that will be identified is, how many orders have been made, the number of products purchased, the amount of payment, and the review score of each customer. This project is made through the process of Data Preparation, Analysis and Result.

DATA PREPARATION

To create customer segmentation, the data goes through the Data Preparation process first. The stages carried out are data encharment, preprocessing, and data exploration.

Data Encharment

This stage is the stage of combining and selecting variables that will be used in the analysis. Order data and customer profiles are combined into one dataset using the following syntax.

```
order_merged_df = pd.merge(orders_df, order_items_df, on='order_id', how='left')
order_merged_df = pd.merge(order_merged_df, order_payments_df, on='order_id', how='left')
order_merged_df = pd.merge(order_merged_df, order_reviews_df, on='order_id', how='left')
customer_order_merged_df = pd.merge(order_merged_df, customer_df, on='customer_id', how='left')
```

Next, the variable is selected using the following syntax.

```
customer_order = customer_order_merged_df[['order_id', 'customer_id','customer_unique_id', 'order_status',
                                           'order_item_id', 'product_id', 'price', 'payment_type', 'payment_value',
                                           'review_score', 'review_id','customer_city', 'customer_state']]
```

These variables are variables that have contributed to this project.

PREPROCESSING

Check for Missing Values

Missing values are data that is not recorded or not available in a dataset. This can happen for many reasons, such as:

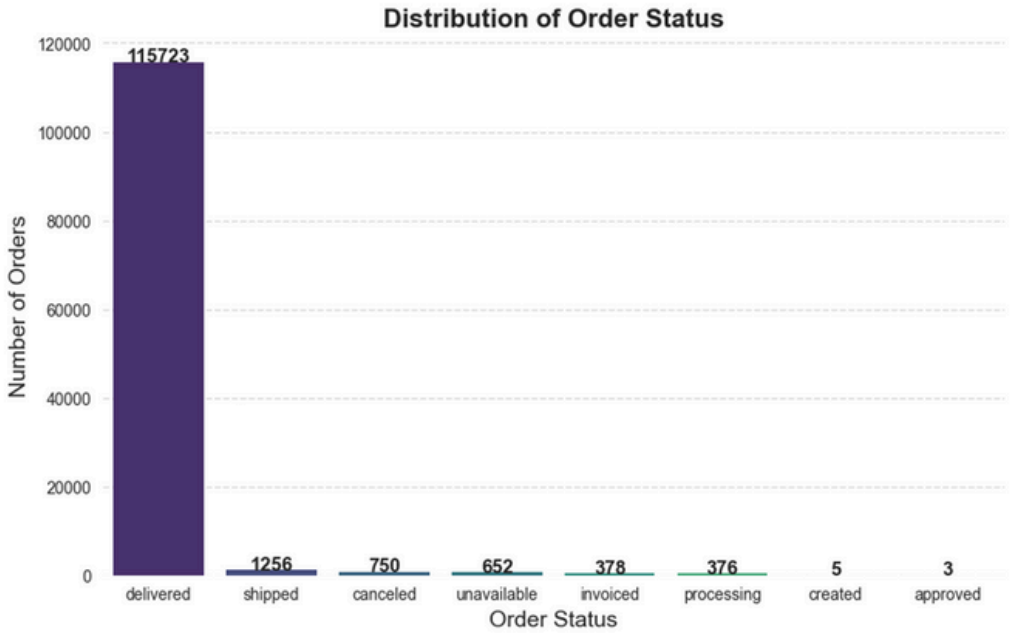
- Data input errors: Data is not entered correctly during data collection.
- Unavailable data: Data that should exist, but cannot be obtained.
- Change in data format: A change in data format may lead to missing values.

Missing values are checked using the following syntax.

```python
# Check for missing values
def check_missing_values(data):
    if data.isnull().values.any() == False:
        print('Data does not contain missing values')
    else:
        for column in data.columns:
            if data[column].isnull().values.any():
                print(f"column '{column}' contains a null value.")
```

```
check_missing_values(customer_order)
✓ 0.3s
column 'order_item_id' contains a null value.
column 'product_id' contains a null value.
column 'price' contains a null value.
column 'payment_type' contains a null value.
column 'payment_value' contains a null value.
column 'review_score' contains a null value.
column 'review_id' contains a null value.
```

There are missing values in the data that need to be investigated.



The causes of missing values are as follows:

- Canceled - 157 Orders: Most of the orders that do not have an approval date are canceled orders. This indicates that when orders are canceled, often the approval process is not continued, so the order_approved_at field remains blank.
- Delivered - 15 Orders: Some orders have been shipped but do not have an approval date. This indicates a possible anomaly or error in the data capture process, where orders were successfully shipped without going through the recorded approval process.
- Created - 5 Orders: Some orders have only been created but not approved and have not been canceled. This could mean the order is still in process or is having issues in the approval section.

Orders that have not gone through the transaction process are removed in the analysis process and the missing value check is performed again.

```
check_missing_values(customer_order)
✓ 0.3s
column 'order_item_id' contains a null value.
column 'product_id' contains a null value.
column 'price' contains a null value.
column 'payment_type' contains a null value.
column 'payment_value' contains a null value.
column 'review_score' contains a null value.
column 'review_id' contains a null value.
```

There are still missing values in the data. The order_item_id variable, which indicates the order ID of the item ordered by the customer, has missing values and needs to be re-identified to determine why this occurred.

```
# customer data that does not have an order_item_id
customer_order[customer_order['order_item_id'].isnull()]
```

✓ 0.0s Python

order_item_id	product_id	price	payment_type	payment_value	review_score	review_id	customer_city	customer_state
NaN	NaN	NaN	boleto	77.73	1.0	b399a4b417fc794a814ef957423bd67a	porto alegre	RS
NaN	NaN	NaN	boleto	73.04	1.0	17bc0bc3207616ee37afd4a39b10054a	aracatuba	SP
NaN	NaN	NaN	credit_card	76.19	1.0	5a4f0fe15c7a914e40cabe9a06d81513	florianopolis	SC

```
# customer data that has no payment_value
customer_order[customer_order['payment_value'].isnull()]
```

✓ 0.0s Python

	product_id	price	payment_type	payment_value	review_score	review_id	customer_city	customer_state
7a4c5ee34285d1e4619a96b4	44.99	NaN	NaN	1.0	6916ca4502d6d3bfd39818759d55d536	sao joaquim da barra	SP	
7a4c5ee34285d1e4619a96b4	44.99	NaN	NaN	1.0	6916ca4502d6d3bfd39818759d55d536	sao joaquim da barra	SP	
7a4c5ee34285d1e4619a96b4	44.99	NaN	NaN	1.0	6916ca4502d6d3bfd39818759d55d536	sao joaquim da barra	SP	

Order payments that are not recorded in the data, even if the transaction is successful, will be deleted as they do not provide enough information for the analysis process. Similarly, order items that are not recorded in the data, even if the order transaction is successful, will also be deleted for the same reason. In addition, data with missing values in the payment_value variable will be checked.

The missing values should be checked again to determine whether there are still any missing values in the data.

```
# customer data that does not have an order_item_id
customer_order[customer_order['review_id'].isnull()]
```

✓ 0.0s Python

order_item_id	product_id	price	payment_type	payment_value	review_score	review_id	customer_city	customer_state
1.0	638bbb2a5e4f360b71f332ddfebfd672	1299.00	credit_card	1376.45	NaN	NaN	rio de janeiro	RJ
1.0	ee0c1cf2fbeae95205b4aa506f1469f0	53.99	boleto	69.12	NaN	NaN	paracatu	MG
1.0	ee406bf28024d97771c4b1e8b7e8e219	144.99	boleto	162.25	NaN	NaN	rio de janeiro	RJ
1.0	41171e11f920c47deb1809edda2bf09d	26.00	credit_card	41.10	NaN	NaN	lages	SC
1.0	040f34fe061b6024771f641fe922e782	69.00	boleto	87.58	NaN	NaN	santa maria	RS
...	...	...	...	...	...	...	...	...
1.0	8466762a9393a2b2b1de73f3cf6081a3	79.90	credit_card	91.96	NaN	NaN	santos	SP
1.0	4deb009c36a910076a023947a7929201	53.89	boleto	138.30	NaN	NaN	sao paulo	SP
2.0	4deb009c36a910076a023947a7929201	53.89	boleto	138.30	NaN	NaN	sao paulo	SP

There are orders that did not review so the review score does not exist. This data will be saved, the review score will be amputated.

FEATURE ENGINEERING

At this stage, data aggregation is carried out to gain insight into consumer purchasing behavior. Data is aggregated and processed based on customer_unique_id to generate some important variables that can be used for further analysis.

Syntax:

```
# Calculate purchase frequency and total purchase value per customer
customer_behavior = customer_order_merged_df.groupby('customer_unique_id').agg({
    'order_id': 'nunique',
    'product_id': 'count',
    'price': 'sum',
    'payment_value': 'sum'
}).reset_index()

customer_behavior.rename(columns={
    'order_id': 'purchase_frequency',
    'product_id': 'total_items_bought',
    'payment_value': 'total_payment_value',
    'price': 'total_price'
}, inplace=True)

customer_behavior.head()
```

Result:

	customer_unique_id	purchase_frequency	total_items_bought	total_price	total_payment_value
0	0000366f3b9a7992bf8c76cfdf3221e2	1	1	129.90	141.90
1	0000b849f77a49e4a4ce2b2a4ca5be3f	1	1	18.90	27.19
2	0000f46a3911fa3c0805444483337064	1	1	69.00	86.22
3	0000f6ccb0745a6a4b88665a16c9f078	1	1	25.99	43.62
4	0004aac84e0df4da2b147fca70cf8255	1	1	180.00	196.89

1. Purchase Frequency (purchase_frequency): Measures how often a customer makes a purchase. It is calculated as the unique number of order_ids associated with each customer.
2. Total Items Purchased (total_items_bought): Calculates the total quantity of products purchased by each customer. This is obtained by counting the total product_ids purchased by the customer.
3. Payment Amount (total_payment_value): Provides information about the total payment value made by each customer, which is the accumulated total of payment_value.
4. Price of Goods Purchased (total_price): Presents the total price value of goods purchased by consumers, which is calculated from the total accumulated price.

Data aggregation is done again to get the average review of each customer

```
review_behavior = customer_order.groupby('customer_unique_id').agg({
    'review_score': 'mean',
}).reset_index()

review_behavior.rename(columns={
    'review_score': 'avg_review_score',
}, inplace=True)

customer_behavior = pd.merge(customer_behavior, review_behavior, on='customer_unique_id', how='left')
customer_behavior.head()
```

✓ 0.4s

	customer_unique_id	purchase_frequency	total_items_bought	total_price	total_payment_value	avg_review_score
0	0000366f3b9a7992bf8c76cfdf3221e2	1	1	129.90	141.90	5.0
1	0000b849f77a49e4a4ce2b2a4ca5be3f	1	1	18.90	27.19	4.0
2	0000f46a3911fa3c0805444483337064	1	1	69.00	86.22	3.0
3	0000f6ccb0745a6a4b88665a16c9f078	1	1	25.99	43.62	4.0
4	0004aac84e0df4da2b147fca70cf8255	1	1	180.00	196.89	5.0

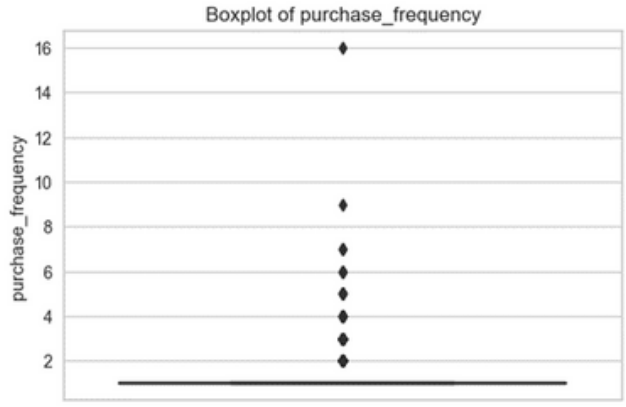
Customers who have never given a review score are imputed with the average score of each customer.

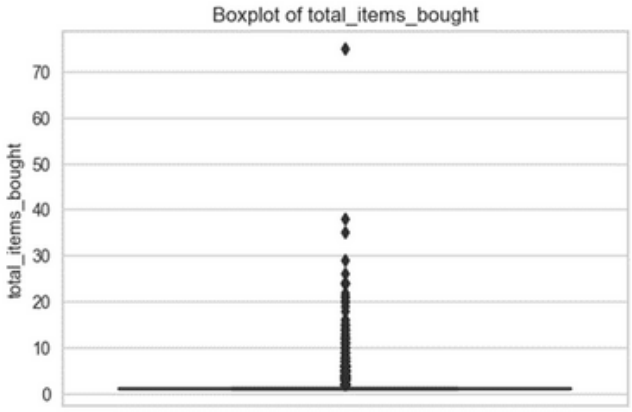
DATA EXPLORATION

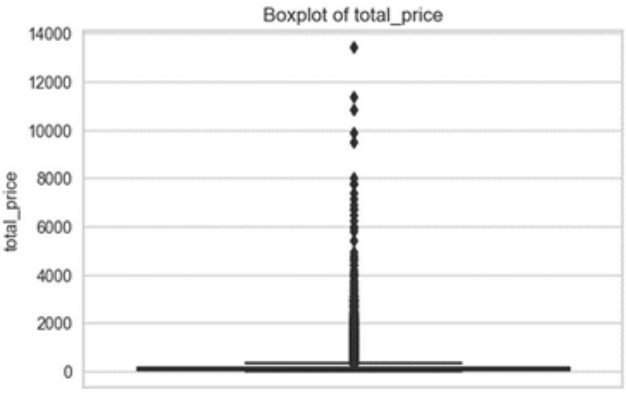
Descriptive statistics of the data:

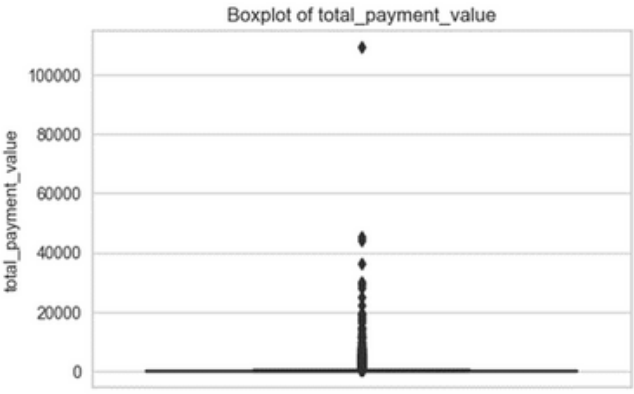
	purchase_frequency	total_items_bought	total_price	total_payment_value	avg_review_score
count	94982.000000	94982.000000	94982.000000	94982.000000	94982.000000
mean	1.033859	1.239498	149.183331	213.344564	4.113933
std	0.210812	0.851993	248.114858	643.785071	1.311557
min	1.000000	1.000000	0.850000	9.590000	1.000000
25%	1.000000	1.000000	48.900000	63.990000	4.000000
50%	1.000000	1.000000	89.900000	113.290000	5.000000
75%	1.000000	1.000000	159.900000	203.080000	5.000000
max	16.000000	75.000000	13440.000000	109312.640000	5.000000

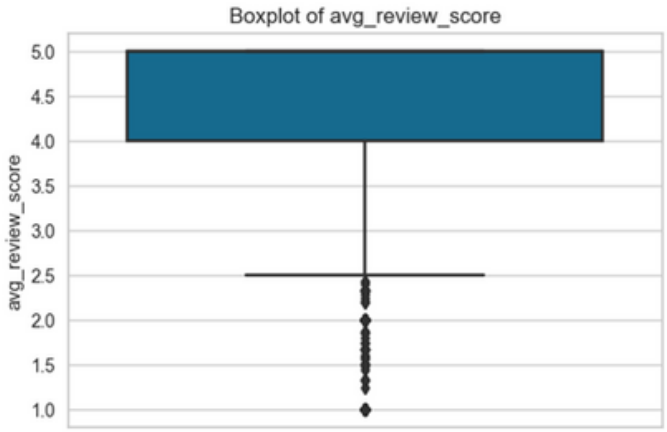
Boxplot:


Boxplot of purchase_frequency
purchase_frequency
16
14
12
10
8
6
4
2


Boxplot of total_items_bought
total_items_bought
70
60
50
40
30
20
10
0


Boxplot of total_price
total_price
14000
12000
10000
8000
6000
4000
2000
0


Boxplot of total_payment_value
total_payment_value
100000
80000
60000
40000
20000
0


Boxplot of avg_review_score
avg_review_score
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0

- **purchase_frequency**: Most of the purchase_frequency data is in the low range, with the majority of customers making 1-2 purchases. There are some significant outliers where some customers make up to 16 purchases, which is far beyond the average value. The interquartile range (IQR) is very narrow, indicating most customers have a low purchase frequency.

- **total_items_boughttotal_items_bought**: As with purchase frequency, most customers purchase relatively small quantities of items. There are outliers that show some customers purchasing much larger quantities of items, up to more than 70 items. This indicates very high variation among customers, with some customers behaving very differently (perhaps a larger customer segment).

- **total_payment_value**: Most customers spend relatively small amounts, with a low IQR. However, there are some very high outliers where some customers spend upwards of 14,000 (price units). These outliers indicate that there are customers who make very high value purchases, which may be the high value customer segment.

- **total_price:** The majority of total_payment_value values are concentrated in a small range. However, there are outliers in total_payment_value that are very high, meaning that there are some transactions with much higher payment amounts than other transactions. This may indicate that there are certain customer segments or transactions that have very high payment values, perhaps due to bulk purchases or the purchase of expensive products.

- **avg_review_score**: The median of the average review score is above 4. This indicates that in general, customers leave moderately positive reviews. The box is quite high, indicating that there is considerable variation in the average review score between customers. This means that there are some customers who give very high scores, but the presence of many outliers below the lower quartile also indicates that there are some customers who consistently give low scores. The large variation and the presence of outliers indicate that there may be several groups of customers with different characteristics in giving reviews.

CLUSTERIZATION

To get customer segmentation, clustering techniques are carried out, this aims to get groups of customer groups based on their characteristics. The clustering technique used is k-means. K-Means can group customers based on similarities in purchasing behavior and valuation, so that we can identify different customer segments. The steps of applying k-means are data standardization, determining the optimal number of clusters and evaluating the results.

Data standardization

Because the variables used have different scales, it is important to standardize the data so that all variables have the same influence in the clustering process.

```
# Data standardization
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_cluster)
data_scaled
```

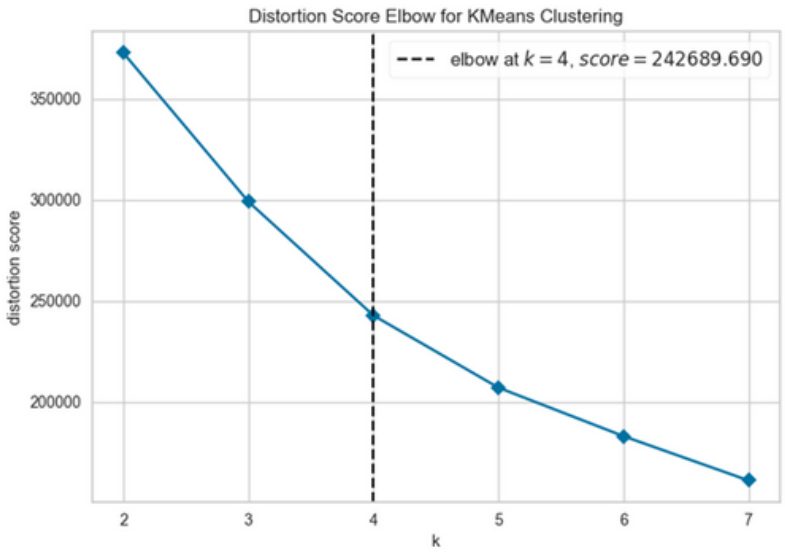
✓ 0.1s

```
array([[-0.16061351, -0.28110485, -0.07771978, -0.11097639,  0.67558767],
       [-0.16061351, -0.28110485, -0.52509559, -0.28915791, -0.08686864],
       [-0.16061351, -0.28110485, -0.32317191, -0.19746533, -0.84932495],
       ...,
       [-0.16061351, -0.28110485, -0.23893629, -0.15670617,  0.67558767],
       [-0.16061351, -0.28110485, -0.13777293, -0.12372916,  0.67558767],
       [-0.16061351, -0.28110485, -0.37157717, -0.22023703,  0.67558767]])
```

This technique is done so that all data is in the range of -1 to 1

Determining the optimal cluster

To determine the optimal number of clusters, the elbow method was used. This technique is based on the within-cluster sum of squares (WCSS) value. WCSS is the sum of squares of the distance between each data and its cluster centroid. Clusters before decreasing WCSS are very significant with the addition of each cluster being the most optimal cluster. In this analysis, the KElbowVisualizer package is used to display the elbow method.



According to the results of the elbow method, the optimal cluster that can be formed is 4 clusters. Furthermore, cluster formation is made using k-means.

```
from sklearn.metrics import silhouette_score
optimal_clusters = 4
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
data_cluster['cluster'] = kmeans.fit_predict(data_scaled)

silhouette_avg = silhouette_score(data_scaled, data_cluster['cluster'])
print('Silhouette Score: ', silhouette_avg)
```

```
Silhouette Score:  0.5734073045711178
```

Silhouette score is a metric used to measure how well objects in a cluster are grouped. Silhouette values range from -1 to 1:

- Values close to 1 indicate that objects fit very well into their cluster and are far away from neighboring clusters.
- A value close to 0 indicates that the object is on the border between two clusters so that it could belong to either cluster.
- A value close to -1 indicates that the object may fit better in an adjacent cluster, rather than their current cluster.

A Silhouette score of 0.573 shows that customer segmentation is quite good.

The segmentation results can be interpreted in the average cluster data table as follows.

cluster	purchase_frequency	total_items_bought	total_price	total_payment_value	avg_review_score
0	1.000000	1.242289	132.237721	196.545595	1.892593
1	1.000000	1.129269	122.531201	158.841809	4.741956
2	2.113942	2.800425	258.968507	388.587134	4.165455
3	1.043956	3.392372	1384.976639	2601.138571	3.839370

Based on the customer segmentation results, four customer clusters were identified based on several variables: **purchase frequency, total items bought, total price, total payment value, and average review score.** The following are interpretations and suggestions based on the values of each cluster:

Cluster 0

- **Purchase Frequency:** 1.00
- **Total Items Bought:** 1.24
- **Total Price:** 132.24
- **Total Payment Value:** 196.55
- **Avg Review Score:** 1.89

Interpretation:
Customers in this cluster tend to buy with low frequency (once only) and buy only a few items. Their purchase value (total price) and payment value are relatively low. Their average reviews are very low, indicating dissatisfaction with the product or service

Suggestion:
Focus on improving the customer experience for this cluster by improving product or service quality and addressing issues that cause dissatisfaction. Consider loyalty or discount programs to encourage them to shop more often or provide better product recommendations.

Cluster 1

- **Purchase Frequency**: 1.00
- **Total Items Bought**: 1.13
- **Total Price**: 122.53
- **Total Payment Value:** 158.84
- **Avg Review Score:** 4.74

Interpretation:
This cluster is similar to Cluster 0 regarding purchase frequency and number of items purchased but with excellent reviews on average. These customers buy little, yet they are very satisfied with the product or service.

Suggestion:
Maintain the quality of the product or service that has been provided to this cluster. Since they are satisfied, they can become potential customers for referral programs or exclusive promotions.

Cluster 2

- **Purchase Frequency:** 2.11
- **Total Items Bought:** 2.80
- **Total Price:** 258.97
- **Total Payment Value:** 388.59
- **Avg Review Score:** 4.17

Interpretation:
Customers in this cluster have a higher purchase frequency and tend to buy more items with a higher total purchase and payment value. They also leave fairly good reviews.

Suggestion:
These customers are active customers who are quite satisfied. You can offer buying packages or bundling to increase the number of items they buy. Loyalty programs that award points for every purchase or review can also increase their frequency and total purchases.

Cluster 3

- **Purchase Frequency:** 1.04
- **Total Items Bought:** 3.39
- **Total Price:** 1384.98
- **Total Payment Value:** 2601.14
- **Avg Review Score:** 3.84

Interpretation:
While the frequency of purchase is about the same as the other clusters (once only), customers in this cluster tend to make purchases in large quantities and with a very high total value. Their reviews are quite positive, although not as high as in clusters 1 and 2.

Suggestion:
This cluster has the potential to become high-value customers. Consider providing VIP services or exclusive offers to keep them loyal. Identify the factors that cause them to only shop once and look for ways to make them repeat customers.

CUSTOMER CLASSIFICATION

This classification project aims to develop a predictive model that can automatically group new customers into pre-identified clusters based on their purchase behavior and characteristics, such as purchase frequency, number of items purchased, total transaction value, item price and average review score. By utilizing the clustered customer data, this model enables companies to implement more personalized marketing strategies, improve customer service, and optimize loyalty and retention programs efficiently. The project also serves as a tool for more informed decision-making, improving customer experience and driving business growth.

PREPROCESSING

Splitting Data

Using previous customer behavior data, the data is divided into training data and testing data with customer clusters being the target variable.

```
X = customer_behavior.drop(columns=['customer_unique_id', 'cluster'], axis=1)
y = customer_behavior['cluster']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Size of the training data:", X_train.shape, y_train.shape)
print("Size of the testing data:", X_test.shape, y_test.shape)
```

```
Size of the training data: (75985, 5) (75985,)
Size of the testing data: (18997, 5) (18997,)
```

Undersampling

There is an unbalanced number of customers in the cluster, an unbalanced class can lead to classification results that tend to predict the majority class. The undersampling technique is performed by randomly removing 50% of the data from the majority class.

Syntax:

```
#  RandomUnderSampler for class 1 and 0 by 50%
rus = RandomUnderSampler(sampling_strategy={0: int(y_train.value_counts()[0] * 0.5), 1: int(y_train.value_counts()[1] * 0.5)},
                         random_state=42)

X_train_resampled, y_train_resampled = rus.fit_resample(X_train, y_train)
```

Result:

Class distribution before and after undersampling technique on training data

Cluster	Before	After
0	15877	28328
1	56657	7938
2	2224	2224
3	1227	1227

Training and Testing Model

The classification methods used are logistic regression, random forest, and Support Vector Machine (SVM). When this model will be trained and predict test data that has been undersampled to see its performance.

Syntax:

```
# Model initialization
logreg = LogisticRegression(random_state=42)
rf = RandomForestClassifier(random_state=42)
svm = SVC(random_state=42)

logreg.fit(X_train_resampled, y_train_resampled)
rf.fit(X_train_resampled, y_train_resampled)
svm.fit(X_train_resampled, y_train_resampled)

y_pred_logreg = logreg.predict(X_test)
y_pred_rf = rf.predict(X_test)
y_pred_svm = svm.predict(X_test)

accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
```

Result:

Accuracy results for each model:

- Logistic Regression: 90%
- Random Forest: 99%
- SVM: 76%

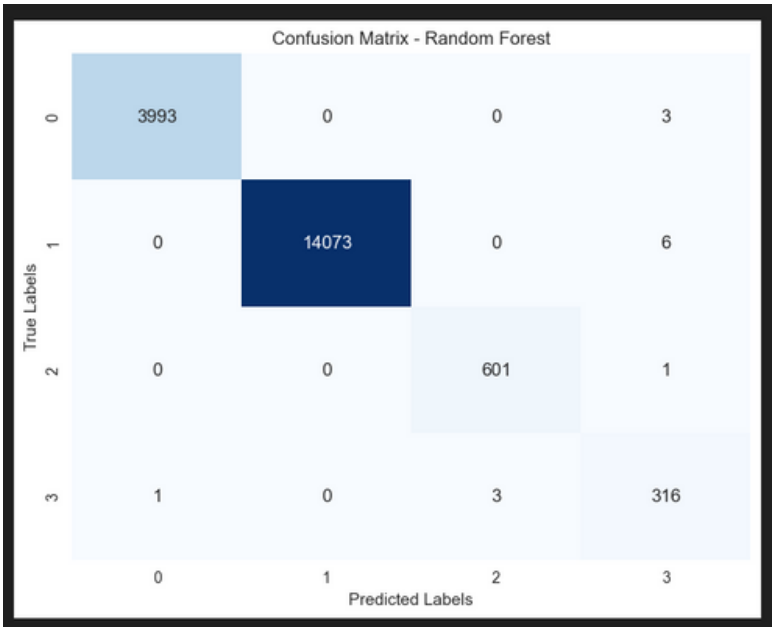
The highest accuracy results are obtained using the Random Forest method. Furthermore, the model will be evaluated for performance using confusion matrix.

Syntax:

```
print(classification_report(y_test, y_pred_rf))
cm = confusion_matrix(y_test, y_pred_rf)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix - Random Forest')
plt.show()
```

Result:



	precision	recall	f1-score	support
0	1.00	1.00	1.00	3996
1	1.00	1.00	1.00	14079
2	1.00	1.00	1.00	602
3	0.97	0.99	0.98	320
accuracy			1.00	18997
macro avg	0.99	1.00	0.99	18997
weighted avg	1.00	1.00	1.00	18997

The confusion matrix results display the model's prediction error. The model has a very small prediction error where cluster 0 has 3 errors, cluster 1 6 errors, cluster 2 1 error, and 3 4 errors. This indicates that the classification model can predict the model very well. The F1-Score results also show the goodness of the model where the macro f1-score gets a value of 99% and the weighted f1-score with a value of 100%, which means the model is very good at predicting all clusters.

GENERAL ADVICE FOR THE COMPANY

1	**Offer Personalization:** Use these segmentation results to offer products, services, or promotions that match the needs and characteristics of each cluster.
2	**Loyalty Management:** Customers with high purchase frequency and transaction value should be given special attention, such as exclusive loyalty programs, special discounts, or priority customer service.
3	**Service Quality Improvement:** For clusters with low reviews, further analysis should be conducted to identify problems and improve service or product quality.
4	**Targeted Marketing Programs:** Focus marketing campaigns on clusters that have the potential to increase purchase value.

PERQARA ASSESMENT

THANK YOU

Alfito Putra Fajar Pratama

+6281546274742

alfito.pfp@gmail.com

alfitoptr