FINAL PROJECT REPORT

E-COMMERCE DELIVERY PREDICTION

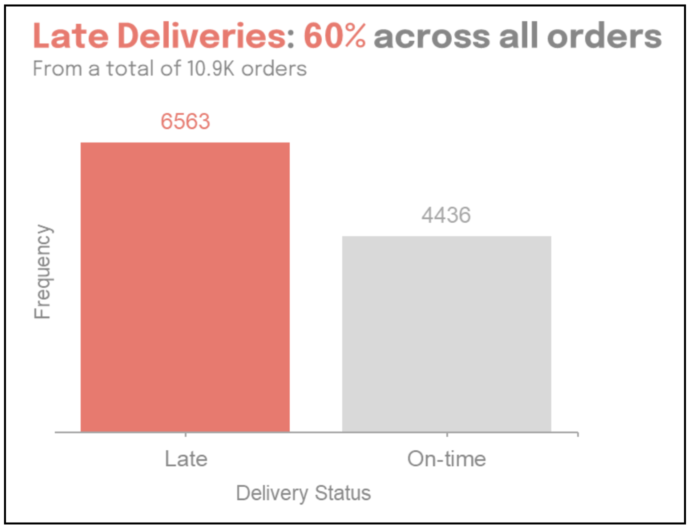
2023

Fourcasters

Rakamin Academy

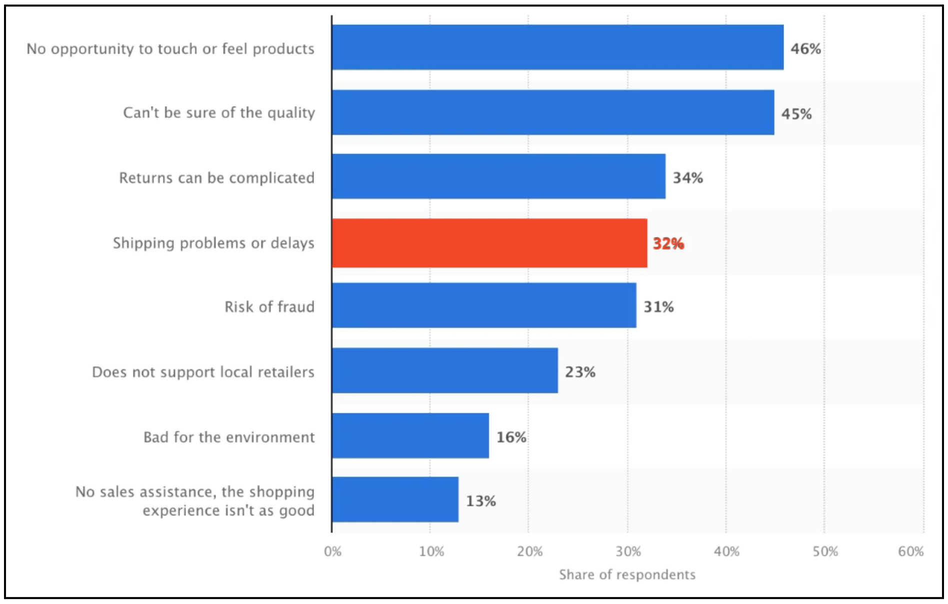
Stage 0: Project Formulation

Problem Statement



Currently, our e-commerce On-Time Delivery Rate is only 40%.

Biggest Drawbacks of E-Commerce Purchases (Worldwide, 2022)
Source: Koen van Gelder via *Statista*



In the world of e-commerce, on-time delivery is essential for customer satisfaction, where 32% of customers complain about late delivery, according to the results of a survey titled "Biggest Drawbacks of E-Commerce Purchases (Worldwide, 2022)" by Koen van Gelder via Statista.

The survey also confirms that late delivery is one of the four major problems faced by e-commerce users.

Project Brief Detail

Background	Concerning high rate of late deliveries that could lead to customer loss
Goal	Improve On-Time Delivery Rate by recommending solutions that address the factors causing delays, so that it can improve the customer shopping experience through our e-commerce.
Objectives	1. Create a machine learning model that can predict whether a delivery will be late or on-time. 2. Identify the factors that cause delivery delays. 3. Recommend solutions and business actions related to the factors that cause delivery delays.
Business Metric	On-Time Delivery Rate - On-time delivery rate is a key performance indicator (KPI) in e-commerce and shipping businesses to evaluate the ability to fulfill orders on schedule.

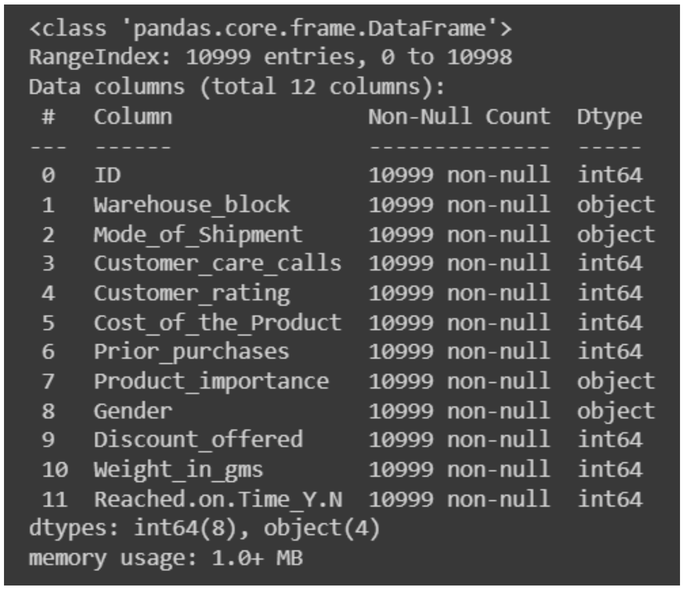
Stage 1: Exploratory Data Analysis (EDA)

Descriptive Analysis

The data contains the following information (taken directly from the source):

- ID: ID Number of Customers.
- Warehouse block: The Company have big Warehouse which is divided in to block such as A,B,C,D,E.
- Mode of shipment:The Company Ships the products in multiple way such as Ship, Flight and Road.
- Customer care calls: The number of calls made from enquiry for enquiry of the shipment.
- Customer rating: The company has rated from every customer. 1 is the lowest (Worst), 5 is the highest (Best).
- Cost of the product: Cost of the Product in US Dollars.
- Prior purchases: The Number of Prior Purchase.
- Product importance: The company has categorized the product in the various parameter such as low, medium, high.
- Gender: Male and Female.
- Discount offered: Discount offered on that specific product.
- Weight in gms: It is the weight in grams.
- Reached on time: It is the target variable, where 1 Indicates that the product has NOT reached on time and 0 indicates it has reached on time.

When we inspected the data attributes in Jupyter Notebook, we got the following information:



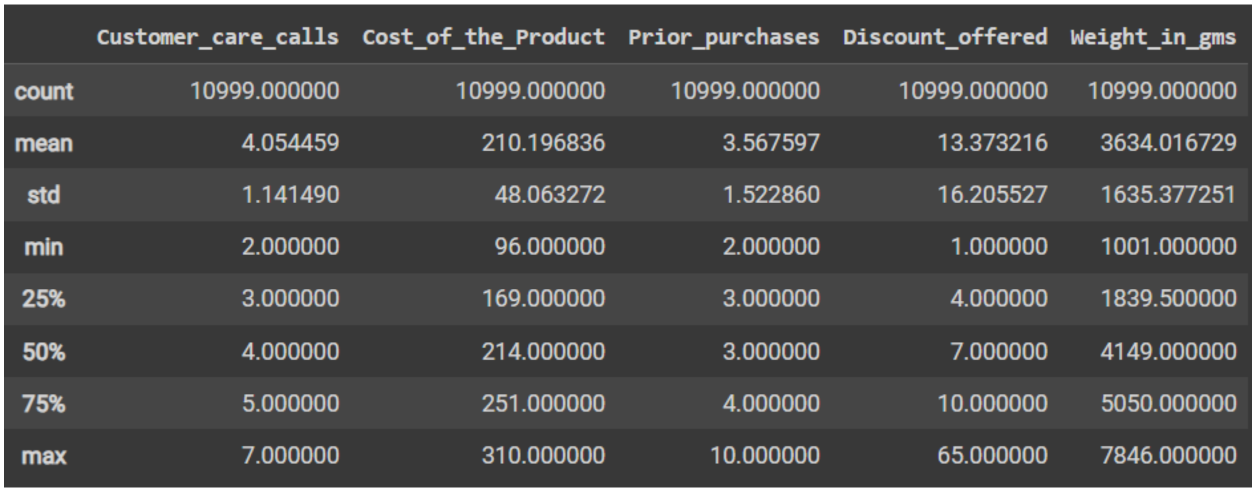
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   ID                   10999 non-null  int64 
 1   Warehouse_block      10999 non-null  object
 2   Mode_of_Shipment     10999 non-null  object
 3   Customer_care_calls  10999 non-null  int64 
 4   Customer_rating      10999 non-null  int64 
 5   Cost_of_the_Product  10999 non-null  int64 
 6   Prior_purchases      10999 non-null  int64 
 7   Product_importance   10999 non-null  object
 8   Gender               10999 non-null  object
 9   Discount_offered     10999 non-null  int64 
 10  Weight_in_gms        10999 non-null  int64 
 11  Reached.on.Time_Y.N  10999 non-null  int64 
dtypes: int64(8), object(4)
memory usage: 1.0+ MB
```

Before the exploration process, we changed several columns/features attribute for visualization purposes, namely:

- ‘ID’ – Changed from int64 to object
- ‘Reached.on.Time_Y.N’ – changed from int64 to object and renamed to ‘Delivery_status’
- ‘Customer_rating’ – changed from int64 to object

Notably, there are no missing values across all columns. Then, we checked for duplicated data and found none

Moving on to summary statistics, we checked the numerical columns/features first:

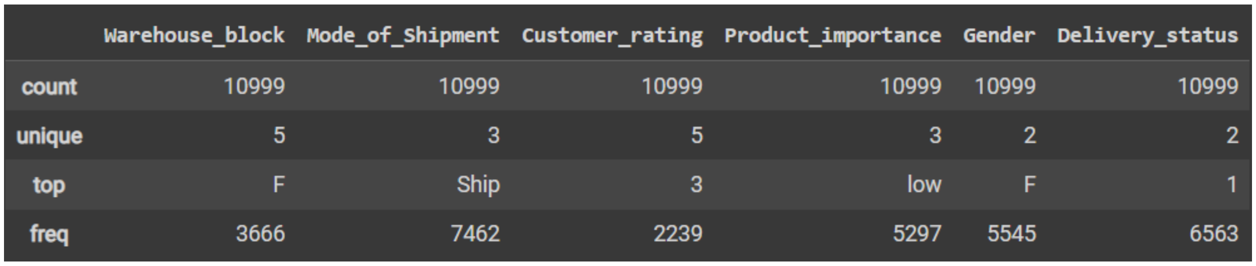


	Customer_care_calls	Cost_of_the_Product	Prior_purchases	Discount_offered	Weight_in_gms
count	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000
mean	4.054459	210.196836	3.567597	13.373216	3634.016729
std	1.141490	48.063272	1.522860	16.205527	1635.377251
min	2.000000	96.000000	2.000000	1.000000	1001.000000
25%	3.000000	169.000000	3.000000	4.000000	1839.500000
50%	4.000000	214.000000	3.000000	7.000000	4149.000000
75%	5.000000	251.000000	4.000000	10.000000	5050.000000
max	7.000000	310.000000	10.000000	65.000000	7846.000000

We found the following:

- The ‘Discount_offered’ variable indicate a right-skewed distribution (Need to confirm later on with histograms), high variance, and potential outlier based on the max and 75% percentile distance
- ‘Prior_purchases’ variable has potential outliers
- ‘Weight_in_gms’ variable indicate a left-skewed distribution
- Other columns looks 'normal' enough based on their numbers

Then, we checked the summary statistics of categorical columns/features:



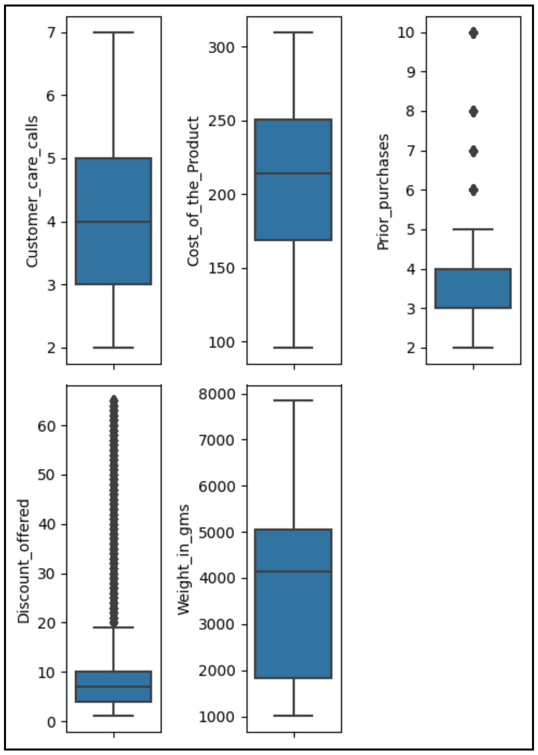
	Warehouse_block	Mode_of_Shipment	Customer_rating	Product_importance	Gender	Delivery_status
count	10999	10999	10999	10999	10999	10999
unique	5	3	5	3	2	2
top	F	Ship	3	low	F	1
freq	3666	7462	2239	5297	5545	6563

We found the following:

- Most products (about 68%) are delivered using the ship method.
- Most deliveries are late (about 60%).
- All columns seem fine based on their numbers (no strange or catchy values).

Univariate Analysis

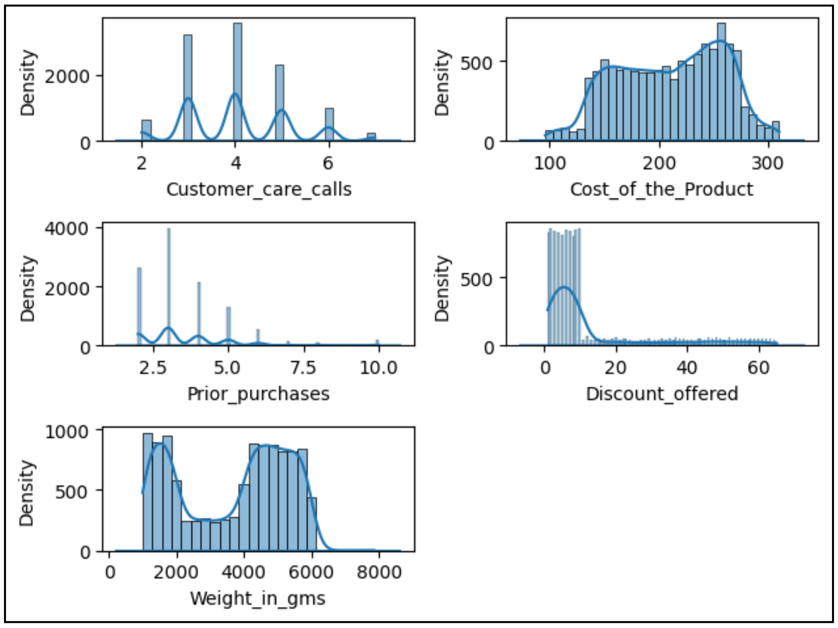
In this section, we excluded the ‘ID’ variable as it has no meaning in the analysis. Just like before, we checked the distribution of numerical columns/features using a box plot first.



We found the following:

- The numerical features that have outliers are ‘Prior_Purchases’ and ‘Dicount_offered’.
- The ‘Discount_offered’ feature has the most outliers.
- No outliers found in other features.

After this, we inspected the distribution shape using a histogram:

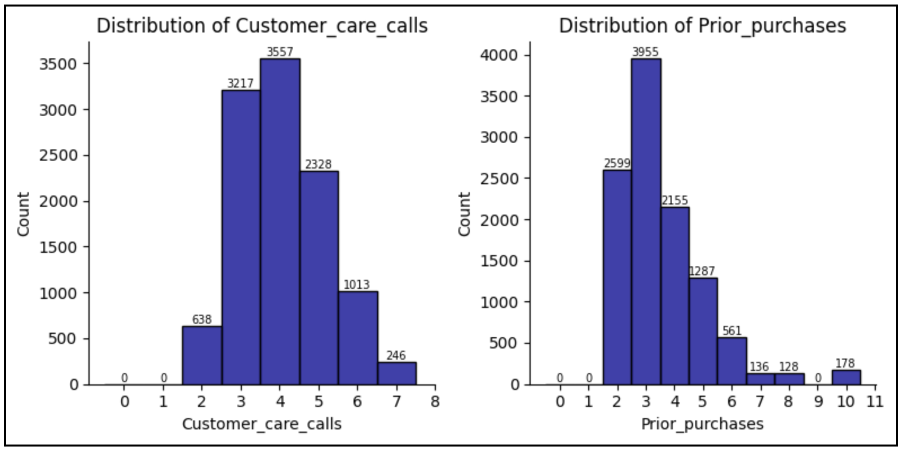


We found the following:

- The ‘Cost_of_the_product’ distribution is approximately normal
- ‘Weight_in_gms’ feature has a bimodal distribution.
- The ‘Discount_offered’ feature has a positively skewed distribution.

After careful consideration, we thought that the following variables contains discrete values and needed their own bins:

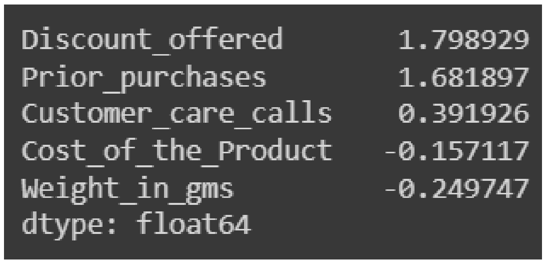
- ‘Customer_care_calls’
- ‘Prior_purchases’



We found the following:

- The ‘Customer_care_calls’ feature has a normal distribution, The highest customer care calls is at 4 calls with 3557 customers.
- The ‘Prior_purchases’ feature has a positively skewed distribution, the highest
- ‘Prior_purchases’ is at 3 prior purchases with 3955 customers.

We further checked the skewness in numbers:

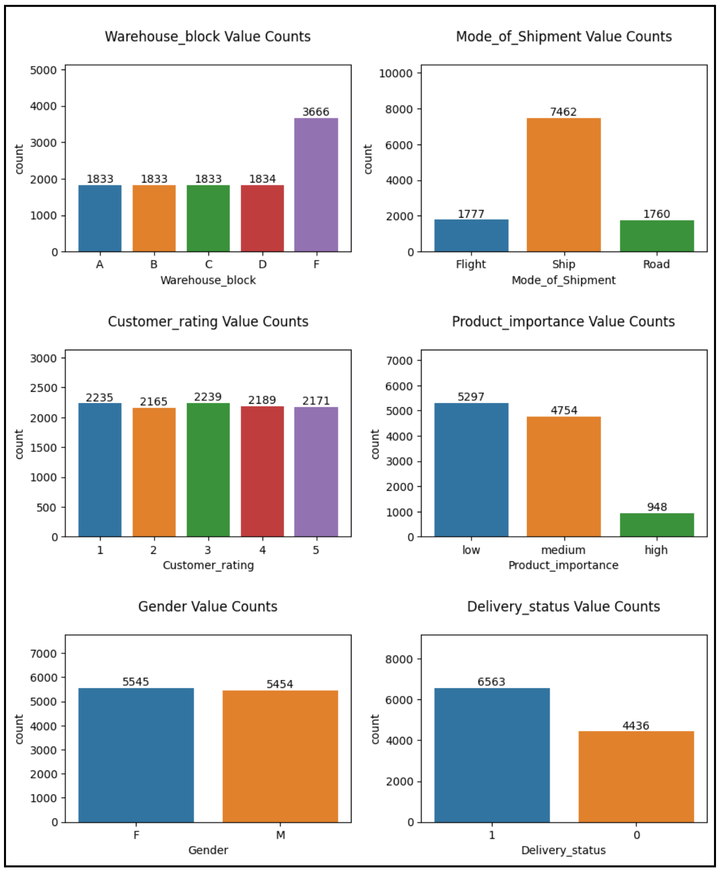


```
Discount_offered        1.798929
Prior_purchases         1.681897
Customer_care_calls     0.391926
Cost_of_the_Product    -0.157117
Weight_in_gms          -0.249747
dtype: float64
```

We can further confirm that the following features have skewed distribution:

- ‘Discount_offered’
- ‘Prior_purchases’

Next, we examined the distribution of categorical columns/features:



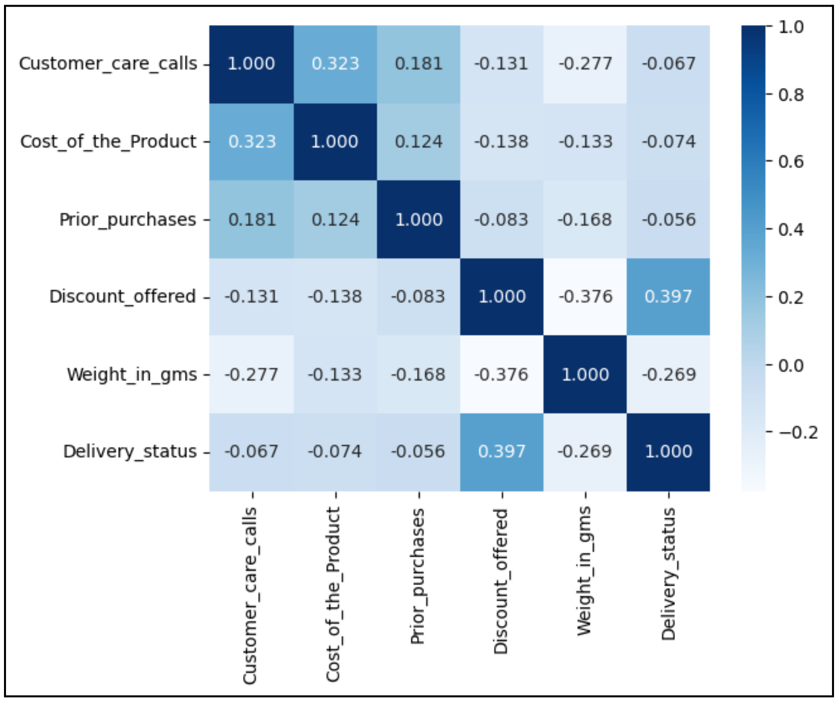
We found out the following:

- Most shipments come from Warehouse_block F.
    - Strangely, warehouse block E is missing from the data
- Based on the 'Mode_of_Shipment', the majority of orders are delivered using the ship method.
- 'Product_importance' shows that the most shipments are "Low", followed by "Medium" and the least are "High".
- There are fewer on time deliveries (4436) than late deliveries (6563) based on the 'Delivery_status' feature.
- The distribution of values inside 'Customer_rating' and 'Gender' is similar (no dominant values).

Multivariate Analysis

<u>Numerical Features Interactions vs. Target</u>

We first checked the correlation heatmap to inspect the interactions between numerical columns/features:



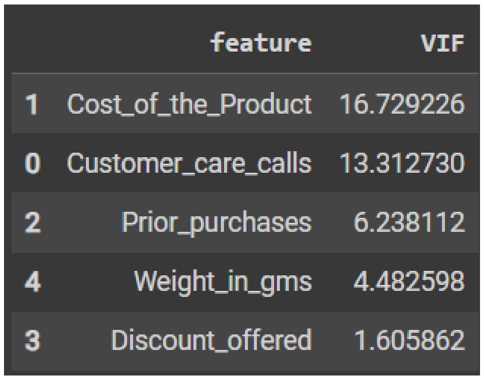
	Customer_care_calls	Cost_of_the_Product	Prior_purchases	Discount_offered	Weight_in_gms	Delivery_status
Customer_care_calls	1.000	0.323	0.181	-0.131	-0.277	-0.067
Cost_of_the_Product	0.323	1.000	0.124	-0.138	-0.133	-0.074
Prior_purchases	0.181	0.124	1.000	-0.083	-0.168	-0.056
Discount_offered	-0.131	-0.138	-0.083	1.000	-0.376	0.397
Weight_in_gms	-0.277	-0.133	-0.168	-0.376	1.000	-0.269
Delivery_status	-0.067	-0.074	-0.056	0.397	-0.269	1.000

We found out the following:

- Discount_offered has a moderate positive relationship with our target (r = 0.4), meaning that higher discount lead to higher probability of late deliveries. This feature may need to be retained for modeling.

- Weight_in_gms has a weak negative relationship with our target (r = -0.27), meaning that the heavier a product is, the higher the probability of late deliveries. This feature may need to be retained for modeling.

As for features vs. features, the correlation doesn't show any strong relationships (correlation above 0.8). We investigated further using VIF



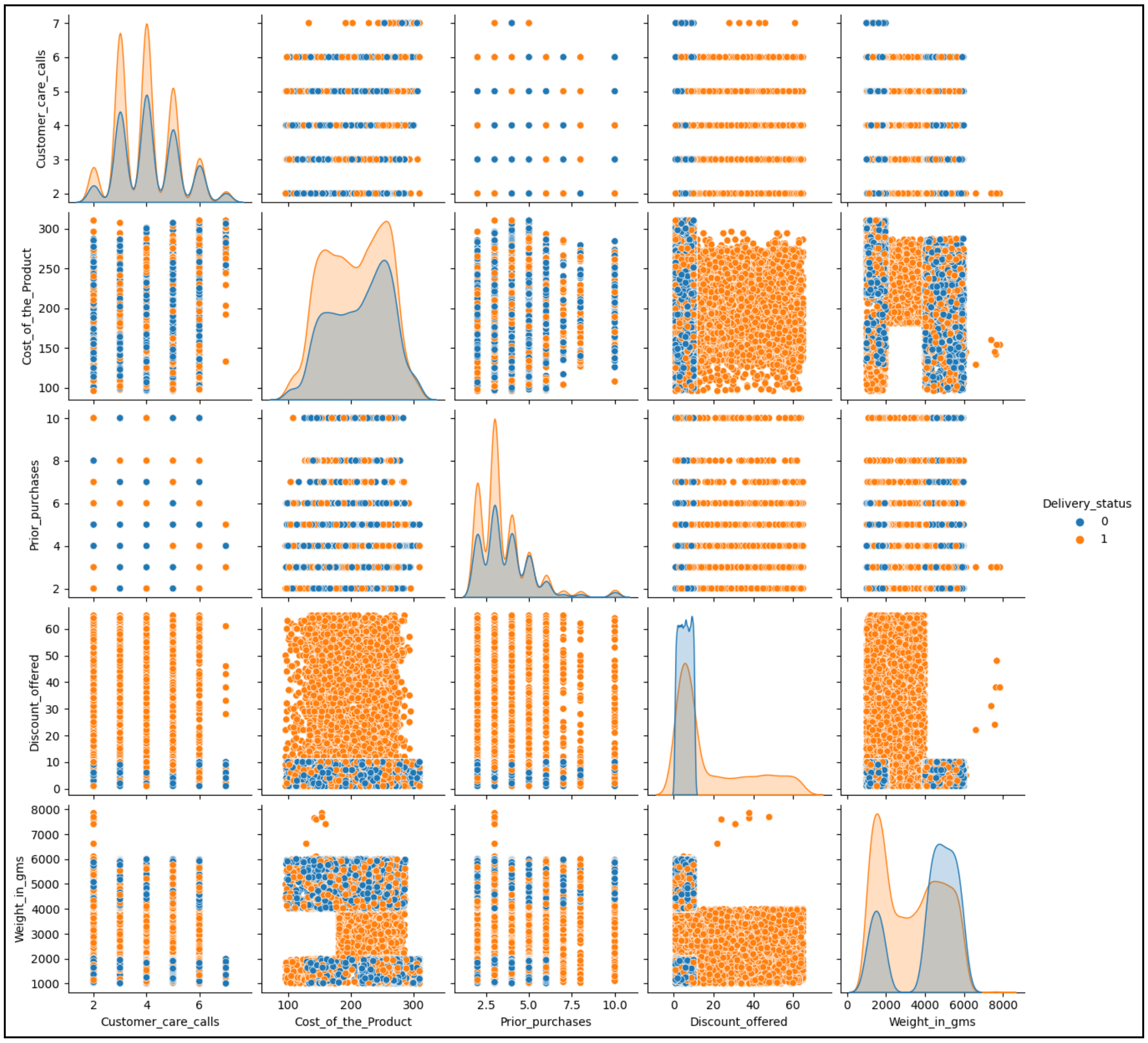
	feature	VIF
1	Cost_of_the_Product	16.729226
0	Customer_care_calls	13.312730
2	Prior_purchases	6.238112
4	Weight_in_gms	4.482598
3	Discount_offered	1.605862

Generally, VIF above 10 means that particular variable can't be used for modeling and the multicollinearity is severe. As we can see from the above results, we have several features that will cause multicollinearity problem, namely:
- 'Cost_of_the_Product'
- 'Customer_care_calls'

To determine which one to remove, we'll compare their strength to the target variable, 'Delivery_status':
- 'Cost_of_the_Product' = -0.074
- 'Customer_care_calls' = -0.067

Because 'Customer_care_calls' have multicollinearity problem and weaker correlation with the target, we'll drop 'Customer_care_calls' later in the preprocessing stage

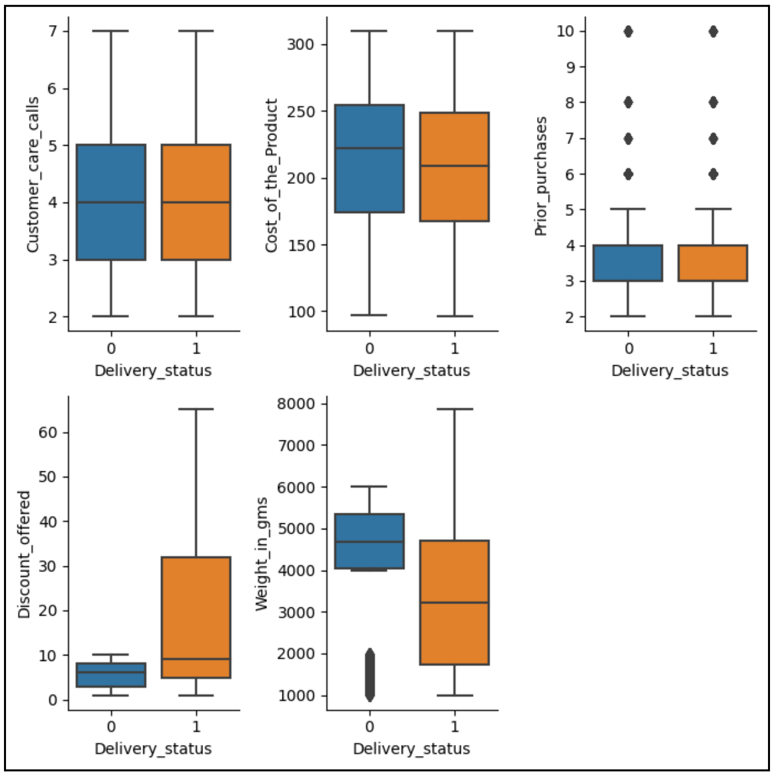
We continued the exploration with a pair plot:



We found out that the late product shipment tend to cluster at:

- Higher amount of discount offered
- Several medium weight range and lower weight range

We further explored to the boxplot of numerical features vs. our target variable, ‘Delivery_status’:



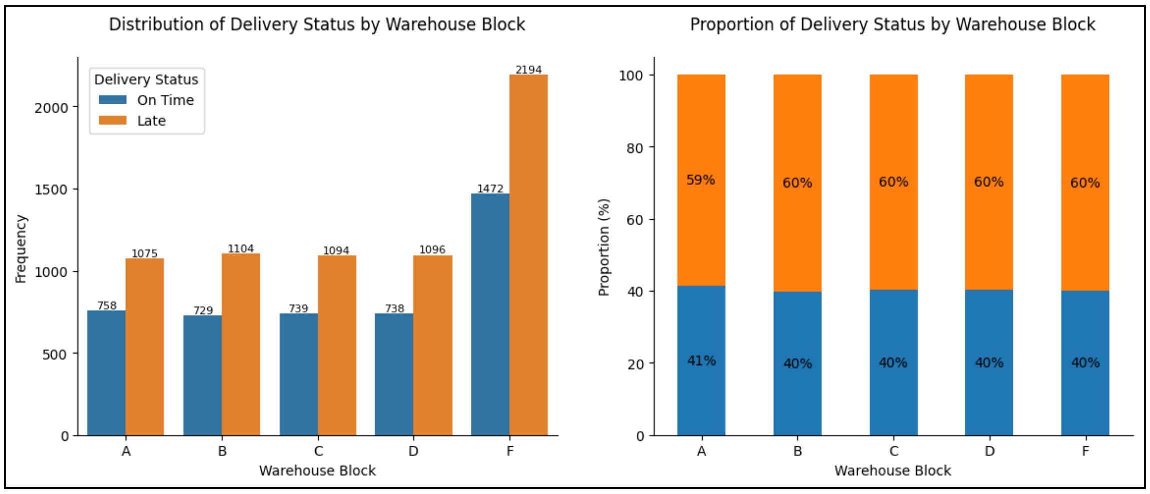
We confirmed further that:

- Late deliveries always happened at a higher amount of discount offered (> 10% of discount)
- Late deliveries always occurred at the weight range of 1000 - 4000 gr and more than 6000 gr
- Other numerical variables doesn't show much difference between on time and late deliveries

Categorical Features Interactions vs. Target

We're comparing different attributes of on-time vs. late deliveries. Therefore, we will base our comparison with frequency and proportion.

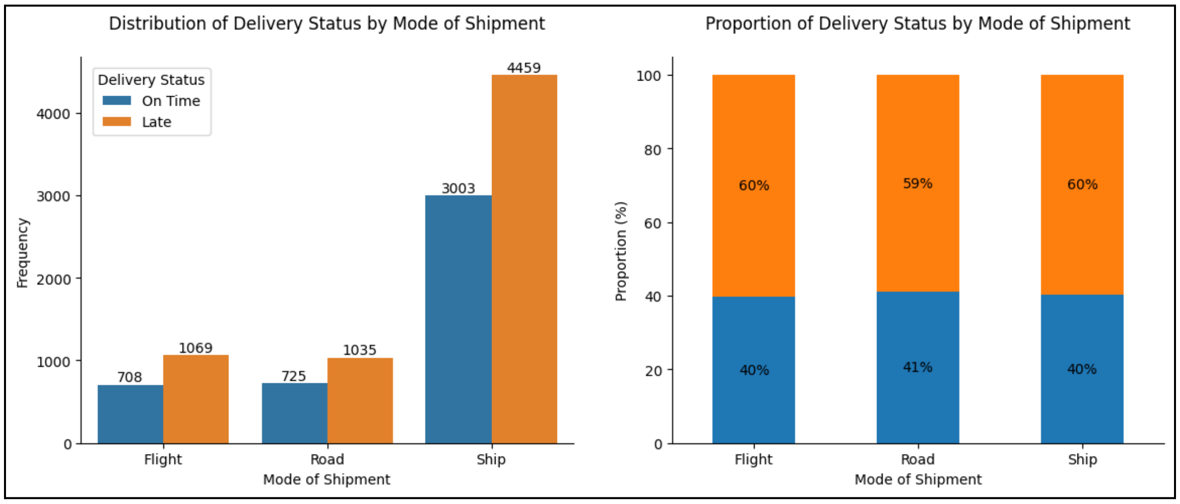
1. Warehouse Block vs. Delivery Status



Observation results:

- All deliveries mostly are late across all warehouse blocks
- When we compare the similar late and on-time delivery proportions for each warehouse block, we can see that the warehouse block itself does not significantly influence whether a shipment will be late or on time.

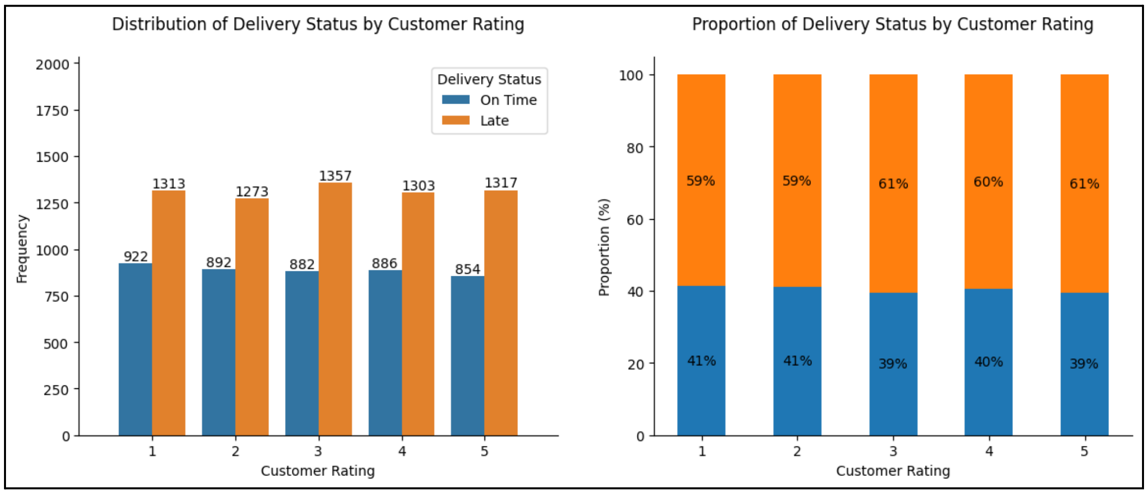
2. Mode of Shipment vs. Delivery Status



Observation results:

- All deliveries mostly are late across all modes of shipment.
- We can see that mode of shipment does not significantly impact delivery lateness, as each mode of shipment shows a similar rate of on time and late deliveries.

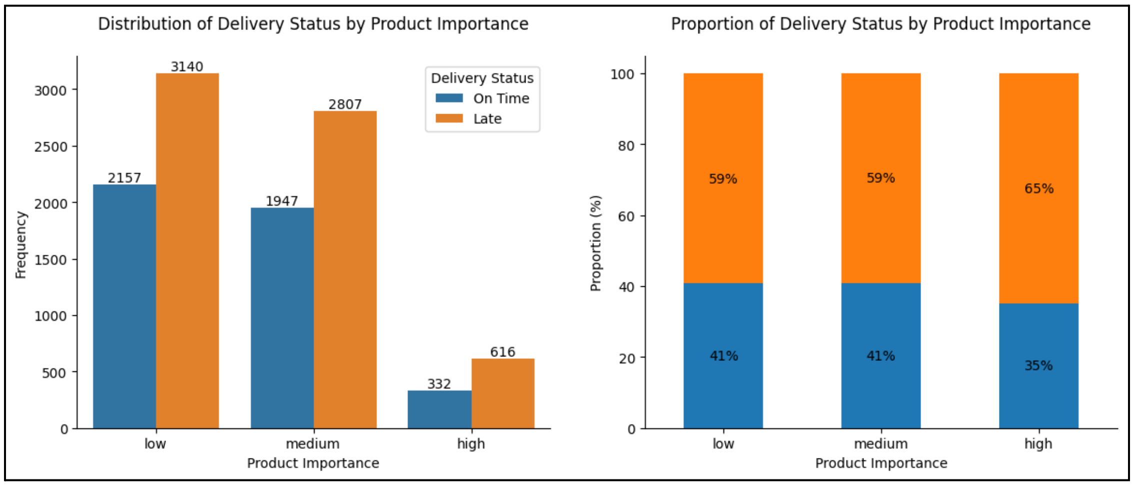
3. <u>Customer Rating vs. Delivery Status</u>



Observation results:

- All deliveries mostly are late across all customer ratings.
- Based on the proportions, we can see that there is not much difference between customer rating of 1 to 5. This means that customer rating doesn't influence the status of a delivery (whether it will be late or not).

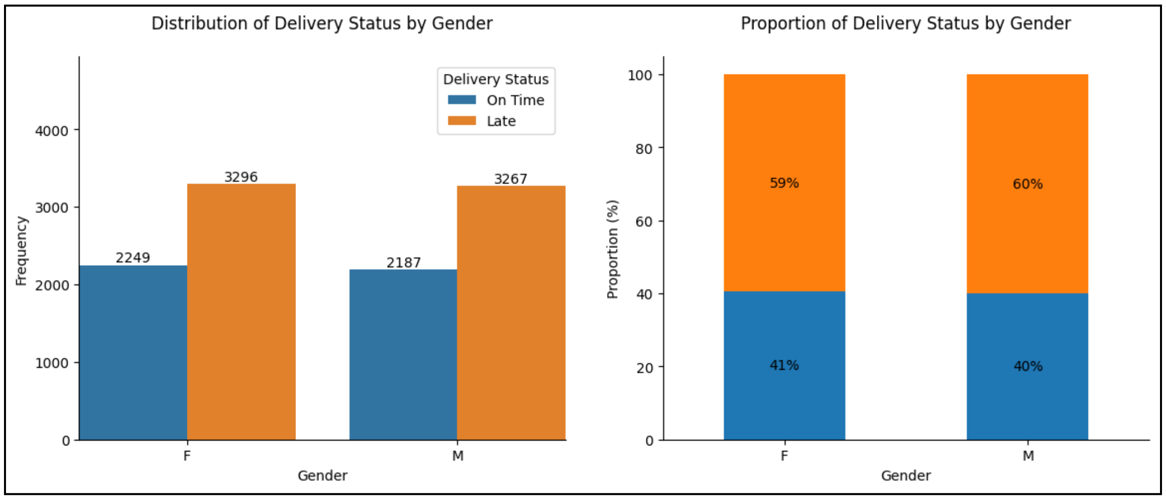
4. <u>Product Importance vs. Delivery Status</u>



Observation results:

- All deliveries mostly are late across all product importance.
- Looking at the proportions, it's visible that products categorized as high importance are more likely to experience late deliveries compared to those categorized as low and medium importance. This indicates that product importance is one of the factors influencing whether a delivery will be late or not.

5. <u>Gender vs. Delivery Status</u>



Observation results:

- All deliveries mostly are late across all genders
- We can see that there's no significant effect of gender to late deliveries because the proportion of late and on time deliveries across all gender is highly similar

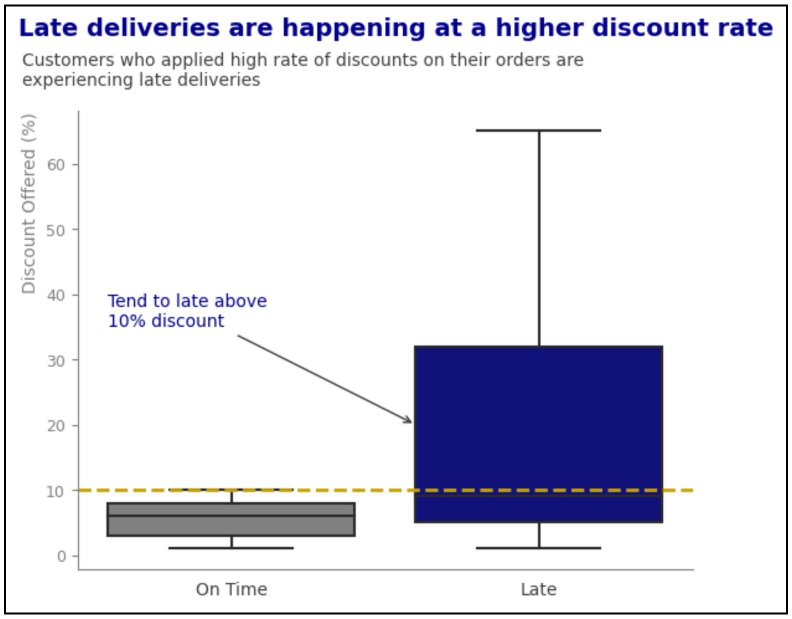
Based on these findings, we have an indication list of good features:

1. 'Discount_offered'
2. 'Weight_in_gms'
3. 'Product_importance'

Business Insights and Recommendations

These insights and recommendations are formulated after the exploratory data analysis (EDA) process above. Let's dive further in

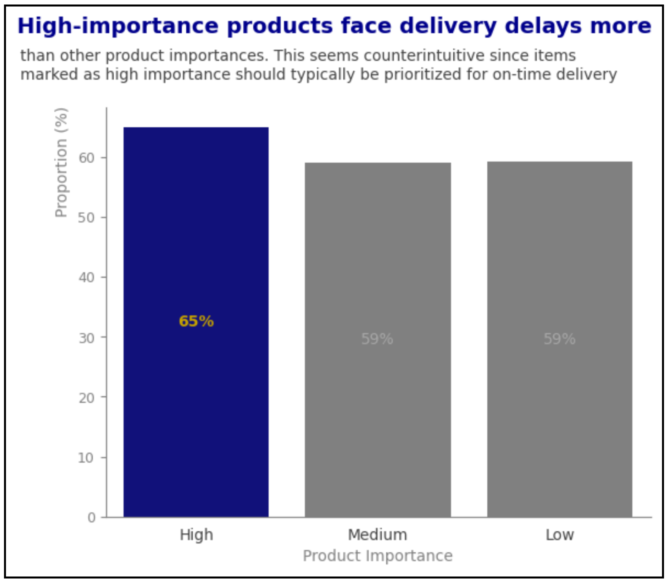
Important Insights



Customers who ordered and applied a discount of more than 10% are experiencing late deliveries. This means that a higher amount of discount is associated with late deliveries. The higher the discount offered, the higher the probability of the delivery status being late. Thus, recommendations regarding the amount of discount offered need to be given.

Here are several assumptions regarding the lateness in higher discounted shipments:

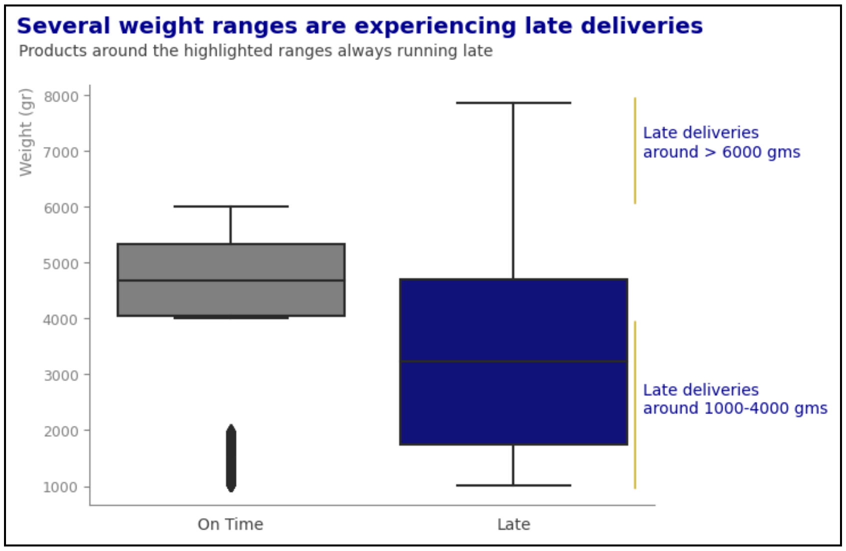
- Resource Allocation: The company may prioritize orders with lower discounts, as they might generate higher profit margins. This could result in fewer resources and attention given to orders with higher discounts, leading to delays in processing and shipment.
- Limited Supply: Orders with higher discounts may be for products that are limited in supply. This could lead to possible inventory shortages and cause delays.



High importance products category tend to experience more late deliveries than other product importance levels. It is important to identify and overcome the causes of this lateness in detail (need more data about the detail about the characteristics of "High" importance goods) to improve the delivery performance of high importance category products

Here are several assumptions regarding the lateness in high product importance:

- Complex Handling: High importance orders may require more complex handling, like specialized packaging or stricter quality checks, which can lead to delays in the shipment process.



Products/goods weighing around mainly 1000 - 4000 gr and some in more than 6000 gr experience more late deliveries than other weight ranges. Further analysis needs to be carried out to identify specific influencing factors to improve heavy product delivery performance.

Nevertheless, the lateness seems to be more concentrated around lighter products' weight that we can see from the center of the distribution.

Note: The late weight range starts from 1000 gr because on-time deliveries at around 1000-2000 gr is considered an outlier, meaning that there's not much on-time deliveries at that weight range

Here are several assumptions regarding the lateness in lighter products' weight:

- Resource Imbalance: Companies may prioritize shipments with weights over 4000 gr because they can generate more revenue. This can lead to resource imbalance, where products under 4000 gr are not adequately resourced.

Business Recommendations

This section provides insights and recommendations based on the current dataset. For more accurate and specific follow-up actions, additional data may be required, depending on the type of factors that cause late deliveries.

1. **Discount Factor**: There is a tendency that the larger the discount given, the more likely the order is to be late. This could be due to several factors, such as high demand for products with large discounts or possible impacts on the delivery process. Interim recommendations that can be made:
    a. Implementing a Discount Threshold Policy (to be simulated later): Simulate the impact of setting a maximum discount threshold (10%) for orders. Test whether limiting discounts to this threshold can reduce late deliveries. This policy can help balance between attracting customers through discounts and ensuring on-time delivery.
    b. Shipping System: Consider optimizing the shipping system/routes for products with large discounts, so that items always arrive on time for products with large discounts.
    c. Continuously Monitor: Monitor late deliveries of products with large discounts specifically and strive to minimize them.

    Note: Limiting discounts may seem counterintuitive. However, here are some of the things we can gain from this policy:
    a. Reduce Late Deliveries: As has been observed, higher discounts are associated with late deliveries. Limiting discounts can help ensure that the company meets its delivery commitments and maintains a better on-time delivery rate, which is essential for customer satisfaction.
    b. Improve Profit Margins: By limiting discounts to 10%, the company can maintain better profit margins on its products. Higher discounts can significantly reduce profits.

2. **Product Importance Influence**: Products categorized as "High" are more likely to be late. This suggests that the "High" category may require special attention to meet delivery schedules. We need additional data on the characteristics of "High" importance products to analyze in more detail. However, there are some interim suggestions that can be made:
    a. Dedicated Team Assignment: Form a dedicated team for the delivery of "high" importance products. This team will have the resources and expertise necessary to ensure on-time delivery for "high" importance items.
    b. Premium Shipping Services: Use shipping companies with a good reputation and track record, especially for "High" importance products. This ensures on-time and fast delivery, even if there is a potential for late delivery.

3. **Weight Factor**: It is seen that some weight ranges (mainly in the 1000-4000 gr range) of products tend to be late. This could be due to the complexity of the process of shipping several items with the mentioned weight ranges or the need for different shipping methods. We need more data on the characteristics of products that have weights in those ranges. Here are some interim recommendations that can be implemented:
    a. Further Analysis: Identify in more detail why products with the aforementioned weights are late. Are there any problems in the delivery process that need to be fixed? Consider adjusting the shipping or logistics process specifically for these products so that they can be shipped more efficiently and on time. Ensure that customers are given realistic delivery estimates for products with certain weights.
    b. Product Packaging: One of the reasons why some items with certain weights are prone to being late can be caused by the packaging of the item itself:
    c. Packaging Strength: Evaluate whether the packaging used for products in that range is strong enough to withstand the weight and handling during shipping. Weak packaging can cause product damage and order delays.
    d. Seal Quality: Make sure the sealing of these packages is secure. Insufficient sealing can cause items to shift during transportation, potentially leading to delays and damage.
    e. Weight Label Accuracy: Make sure the accuracy of the weight label on the package is correct. The wrong weight labeling process can result in the selection of an inappropriate transportation method, leading to late delivery (e.g., items with heavy weight labels are shipped using small modes of transportation, and vice versa).

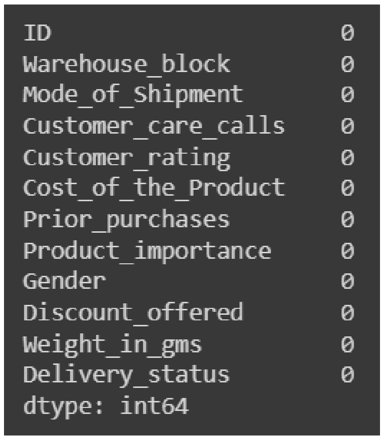
Overall, it is important for businesses to continuously monitor and improve logistics and delivery processes to ensure timely and reliable service to customers. These recommendations should be a starting point for addressing potential problems and optimizing on-time delivery rates.

Stage 2: Data Cleaning and Preprocessing

Data Cleaning

Missing values

There are no missing values in the data and we don't need any cleaning in this part. Here's the detailed number of missing values across all columns:



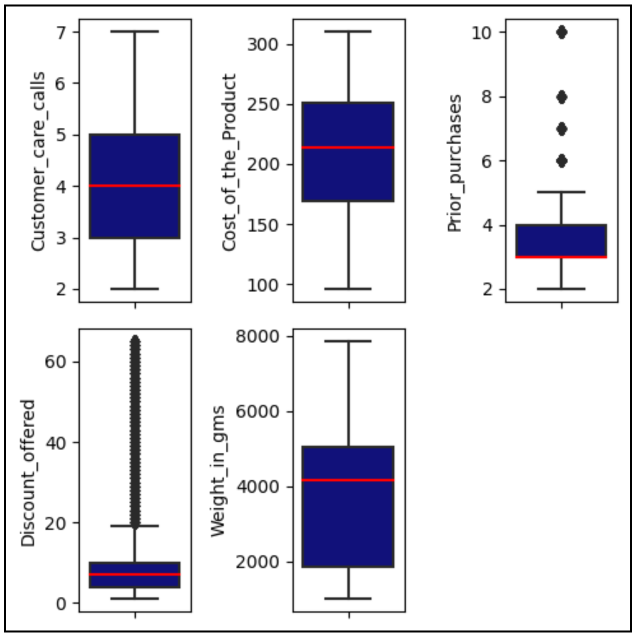
```
ID                     0
Warehouse_block        0
Mode_of_Shipment       0
Customer_care_calls    0
Customer_rating        0
Cost_of_the_Product    0
Prior_purchases        0
Product_importance     0
Gender                 0
Discount_offered       0
Weight_in_gms          0
Delivery_status        0
dtype: int64
```

Duplicated data

There is no duplicated data in this dataset therefore we don't need any cleaning in this part. Here's the result after checking for duplicates:

Outliers

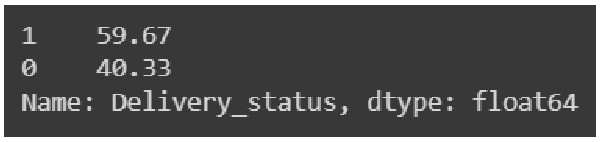
We based our outlier detection with the most common approach: Box plot's IQR



We have several collective outliers in the ‘Discount_offered’ feature but the outlier values are still appropriate (‘Discount_offered’ ranges between 0% to 100%). Hence, we'll not drop any of them because despite outliers, they are still valid observations.

Regarding the ‘Prior_purchases’, we see several outliers but all of them are still appropriate (6, 7, 8, or 10 prior purchases is still a valid number). Therefore, we'll not drop any of them because despite outliers, they're still valid observations.

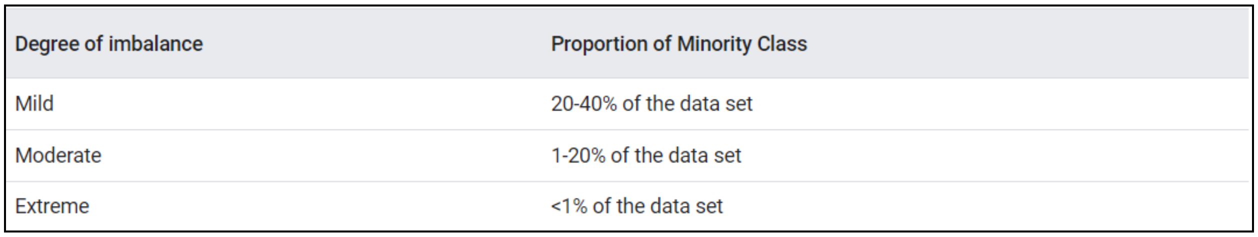
<u>Class imbalance</u>
We inspected the proportion of classes in the target variable, ‘Delivery_status’ and found that the dataset is mildly imbalanced



```
1    59.67
0    40.33
Name: Delivery_status, dtype: float64
```

Note: 1 means late and 0 means on-time

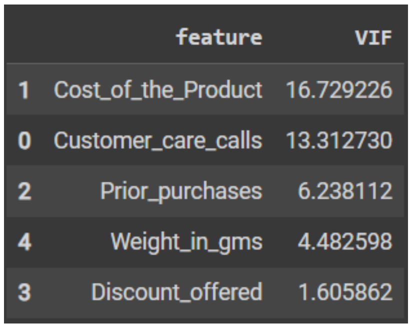
According to Google’s Imbalanced Data Degree, this dataset have a mild class imbalance because our minority class is around 40.33%



Degree of imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

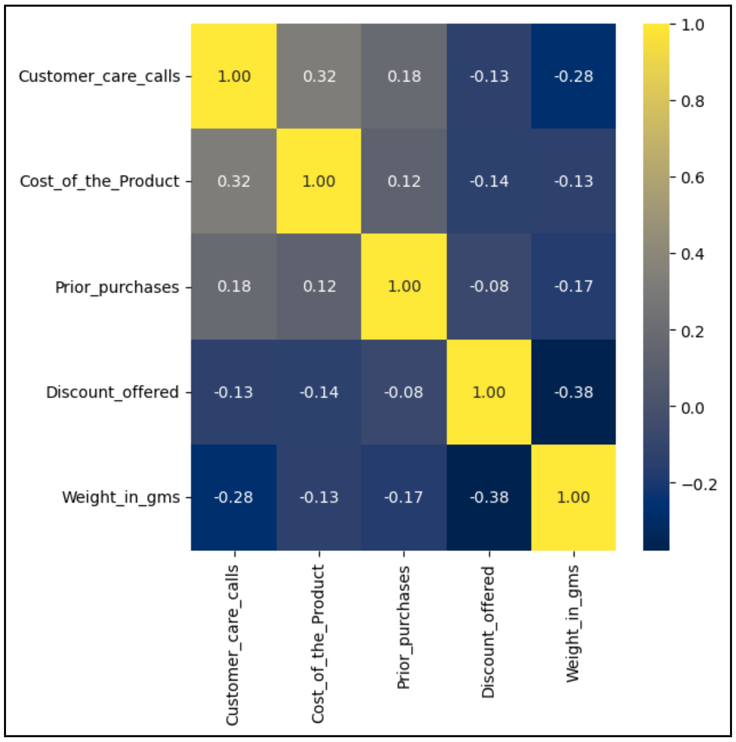
No further action needed unless our dataset is having a moderate/extreme imbalance degree.

<u>Multicollinearity treatment</u>
As we mentioned before in the Exploratory Data Analysis (EDA), there are several features with multicollinearity but we will retain the feature with the most correlation with our target ‘Delivery_status’

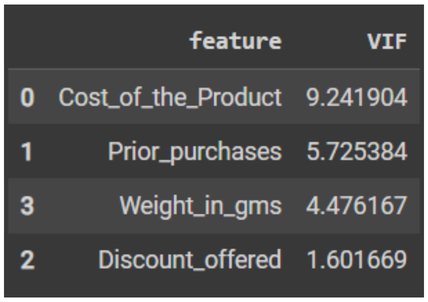


	feature	VIF
1	Cost_of_the_Product	16.729226
0	Customer_care_calls	13.312730
2	Prior_purchases	6.238112
4	Weight_in_gms	4.482598
3	Discount_offered	1.605862

As a reminder, features that could cause multicollinearity are those with VIF > 10, namely 'Cost_of_the_Product' and 'Customer_care_calls'. Let's revisit the correlation heatmap again to see the correlation strength towards our target variable, 'Delivery_status' for these features



Despite the high VIF value of 'Cost_of_the_Product', it has better correlation with the target. Hence, we'll remove 'Customer_care_calls' instead with lower correlation with the target. Here's the VIF after we removed the 'Customer_care_calls' feature:



	feature	VIF
0	Cost_of_the_Product	9.241904
1	Prior_purchases	5.725384
3	Weight_in_gms	4.476167
2	Discount_offered	1.601669

Now there's no more feature that has VIF > 10 and causes further multicollinearity.

Feature Creation/Extraction

<u>Product Cost Class</u>

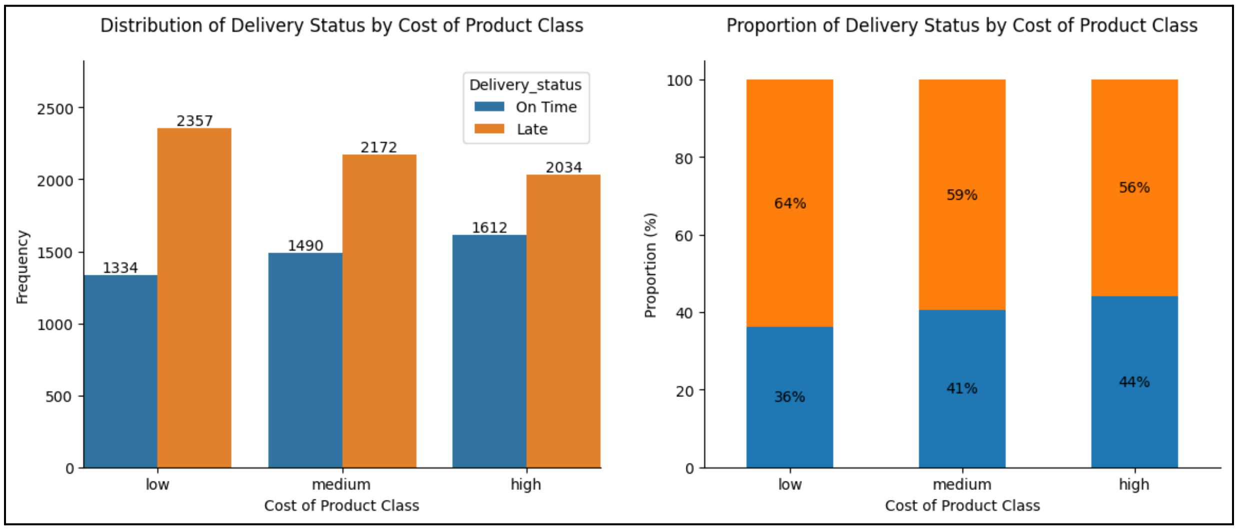
This is a feature that groups several ranges of values in the 'Cost_of_the_Product' feature. We based the grouping by its quantiles. Here's the quantile result:

```
array([ 96., 156., 184., 214., 240., 261., 310.])
```

From the quantile result above, we divided the grouping range by:

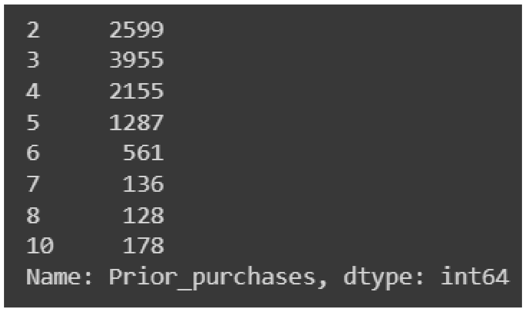
- Low = Orders with product that costs < $185
- Medium = Orders with product that costs between $185 - $241
- High = Orders with product that costs > $241

Let's see if there are any differences in the delivery status rate between the cost classes.



This feature could be a good predictor to 'Delivery_status' because as we can see above, the low, medium, and high product cost class delivery rate varies, where most deliveries are late in low cost class, followed by medium, and lastly high cost class.

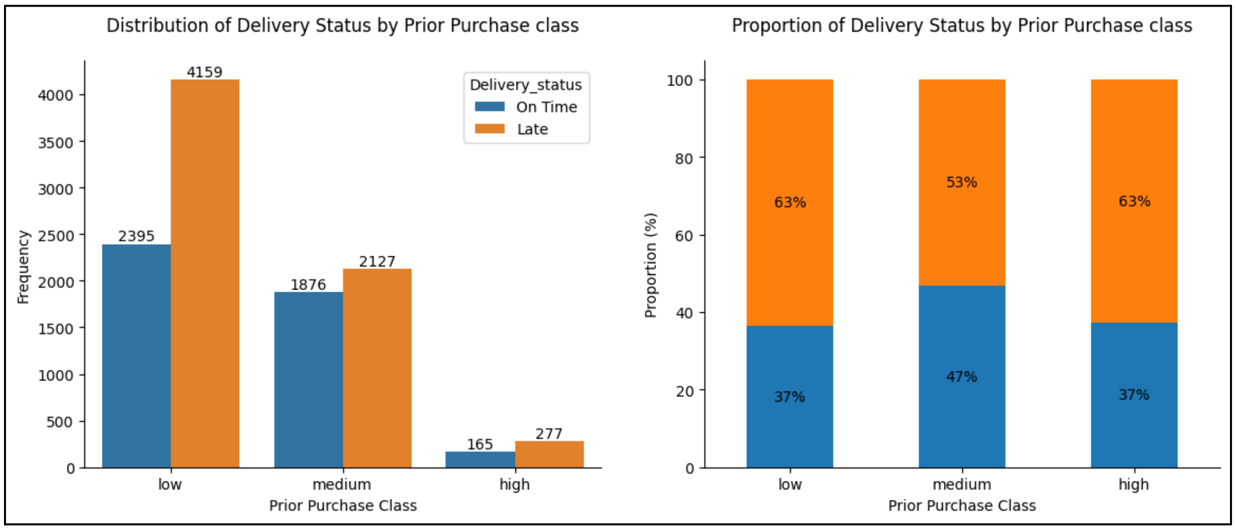
<u>Prior purchase class</u>
This feature is the transformed version of ‘Prior_purchases’ where the values were grouped to low, medium, and high. For the grouping base range, we checked the unique values that it has first.



```
2     2599
3     3955
4     2155
5     1287
6      561
7      136
8      128
10     178
Name: Prior_purchases, dtype: int64
```

We formulated the grouping as:
- low = Prior purchases < 4
- medium = Prior purchases from 4 to 6
- high = Prior purchases more than 6

Let's see if there are any differences in the delivery status rate between the prior purchases classes.



Low and High prior purchases class is suffering more late deliveries than the medium prior purchase class. We can use this feature as a predictor in the model.

Feature Encoding

We used ordinal encoding only because the categorical feature that has a good indication of relationship towards the target are:

1. ‘Product_importance’ (ranging from low to high)
2. ‘pp_class’ (prior purchase class, ranging from low to high)
3. ‘cost_class’ (ranging from low to high)

We have implemented the encoding in these categorical data by creating a custom function called ‘custom_ordinal_encoding’. This function accepts column names, dataset, and categories as its input parameters, which allows us to efficiently and consistently perform ordinal encoding on each column that requires transformation, increasing clarity and reducing the risk of errors in the code, as well as maintaining consistency in the mapping of category values "low," "medium," and "high."

Data Splitting

Here are the variables that we will use in the model:

1. Numeric
    a. ‘Discount_offered’
    b. ‘Weight_in_gms’
2. Categorical: Ordinal
    a. ‘Product_importances’
    b. ‘Cost_class’
    c. ‘Prior_Purchase_class’

We splitted the data with a ratio of 70:30 (train set: test set). We splitted the data in this part because we don’t want to get any data leakage in the model training process, commonly happening in the scaling and transformation process.

Feature Transformation (Normalizing the Skewed Distribution)

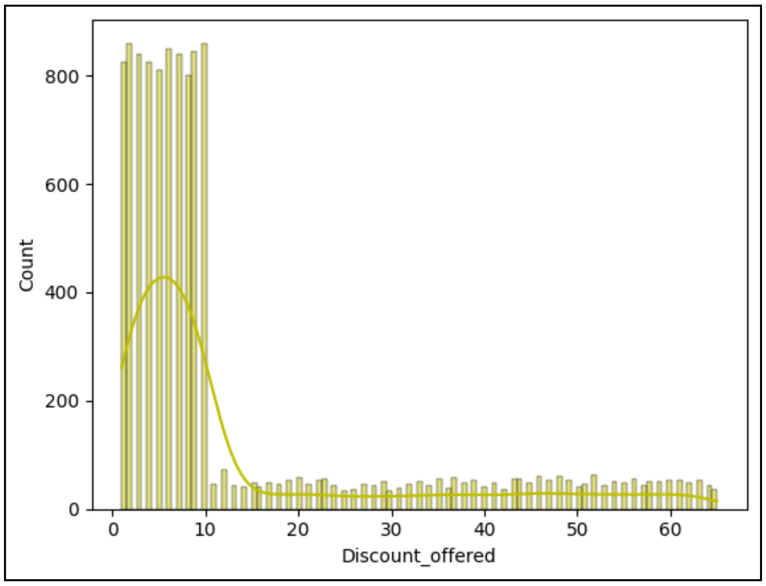
‘Discount_offered’ has a right-skewed distribution problem. We normalized the distribution before stepping into modeling. First, we checked the minimum value that ‘Discount_offered’ has, which is 1, so we knew what are the available methods for normalization.

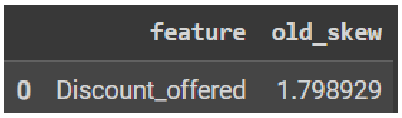
Note: As this first section is only inspecting the initial distribution shape and skewness plus benchmarking all the available transformation methods, we used the main dataset. Further transformation will be made after benchmark on splitted data

We know that it doesn't have 0 or negative values, so we will benchmark the following normalization methods:

1. Square Root
2. Cube Root
3. Log
4. Reciprocal
5. Box Cox
6. Yeo-Johnson

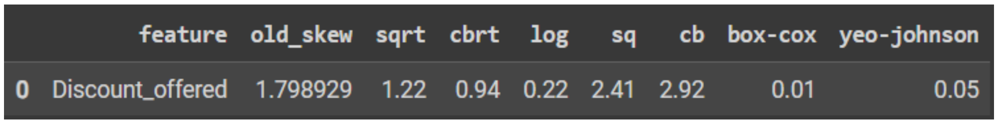
Let's see the shape and the skewness first.





	feature	old_skew
0	Discount_offered	1.798929

Then, we did the benchmark process and got the following result.

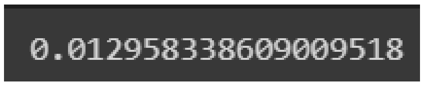


	feature	old_skew	sqrt	cbrt	log	sq	cb	box-cox	yeo-johnson
0	Discount_offered	1.798929	1.22	0.94	0.22	2.41	2.92	0.01	0.05

Box-cox normalization technique is the most effective in reducing skewness for this feature. Then, we fitted the transformer to the train set and transformed it to both the train set and test set.

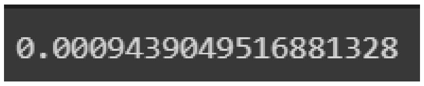
Here are the skewness degrees after we used the box cox transformation technique.

Train Set



```
0.012958338609009518
```

Test Set



```
0.0009439049516881328
```

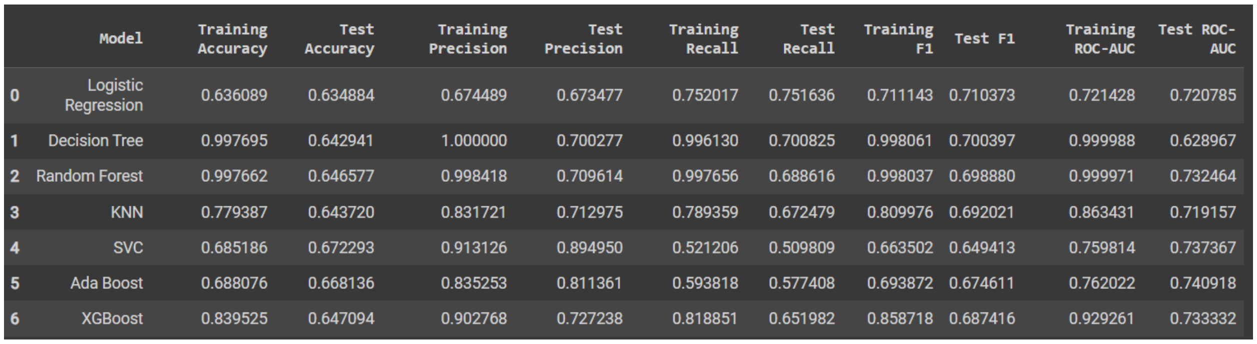
Scaling

We used the StandardScaler as the scaling method because many of the features' initial distribution is approximately normal

Stage 3: Modeling and Evaluation

Initial Evaluation for Model Selection

Recall is the primary evaluation metric for this project because we want to minimize false negatives. False negatives are costly because they represent late deliveries that are predicted to be on time. This can make customers unhappy and less trusting of our company. Here are the results of our initial model evaluation process (cross validation evaluation).



	Model	Training Accuracy	Test Accuracy	Training Precision	Test Precision	Training Recall	Test Recall	Training F1	Test F1	Training ROC-AUC	Test ROC-AUC
0	Logistic Regression	0.636089	0.634884	0.674489	0.673477	0.752017	0.751636	0.711143	0.710373	0.721428	0.720785
1	Decision Tree	0.997695	0.642941	1.000000	0.700277	0.996130	0.700825	0.998061	0.700397	0.999988	0.628967
2	Random Forest	0.997662	0.646577	0.998418	0.709614	0.997656	0.688616	0.998037	0.698880	0.999971	0.732464
3	KNN	0.779387	0.643720	0.831721	0.712975	0.789359	0.672479	0.809976	0.692021	0.863431	0.719157
4	SVC	0.685186	0.672293	0.913126	0.894950	0.521206	0.509809	0.663502	0.649413	0.759814	0.737367
5	Ada Boost	0.688076	0.668136	0.835253	0.811361	0.593818	0.577408	0.693872	0.674611	0.762022	0.740918
6	XGBoost	0.839525	0.647094	0.902768	0.727238	0.818851	0.651982	0.858718	0.687416	0.929261	0.733332

Based on recall as the primary criterion, we selected the following three algorithms for hyperparameter tuning:

1. Logistic regression: This algorithm had the highest recall score in the initial evaluation and the best overall fit.
2. Random forest: This algorithm had a good recall score and a good ROC-AUC score.
3. XGBoost: This algorithm had a slightly lower recall score than random forest, but a slightly better accuracy and AUC. It also overfits less than random forest.

We excluded the following algorithms from hyperparameter tuning:

1. Decision tree: This algorithm had a good recall score, but a worse ROC-AUC score.
2. Support vector machines (SVMs): This algorithm had the lowest initial recall score of all algorithms, despite having the highest accuracy.
3. AdaBoost: This algorithm had a low recall score, despite having the highest ROC-AUC score.
4. k-nearest neighbors (kNN): This algorithm had a good recall score, but a lower AUC than the other algorithms selected for hyperparameter tuning.

Hyperparameter Tuning

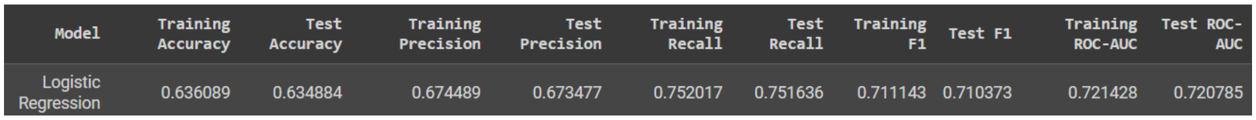
First, we tuned the ROC-AUC because a higher AUC score in a model indicates that the model is better at distinguishing between positive and negative classes. Then, increased the recall for each model using threshold tuning. All of this hyperparameter tuning process was done using the RandomizedSearchCV method because of time and resources constraints.

Logistic regression

For logistic regression, we tuned the following parameters:

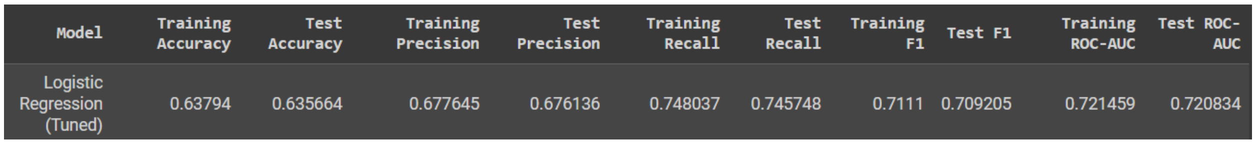
1. penalty
2. C
3. solver

Here's the initial evaluation of our logistic regression model.



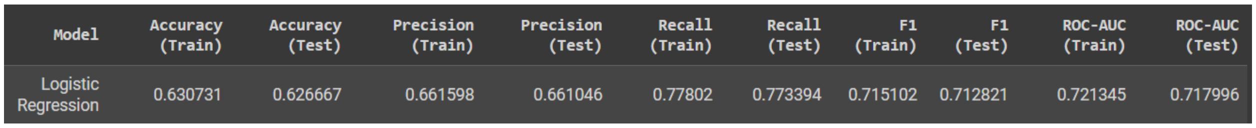
Model	Training Accuracy	Test Accuracy	Training Precision	Test Precision	Training Recall	Test Recall	Training F1	Test F1	Training ROC-AUC	Test ROC-AUC
Logistic Regression	0.636089	0.634884	0.674489	0.673477	0.752017	0.751636	0.711143	0.710373	0.721428	0.720785

And here's how the model performed on the training set (cross validated) after using the best hyperparameters. The model shows a good fit.



Model	Training Accuracy	Test Accuracy	Training Precision	Test Precision	Training Recall	Test Recall	Training F1	Test F1	Training ROC-AUC	Test ROC-AUC
Logistic Regression (Tuned)	0.63794	0.635664	0.677645	0.676136	0.748037	0.745748	0.7111	0.709205	0.721459	0.720834

Then, we tried to tune the threshold to improve the recall and got the following naive train-test evaluation result.



Model	Accuracy (Train)	Accuracy (Test)	Precision (Train)	Precision (Test)	Recall (Train)	Recall (Test)	F1 (Train)	F1 (Test)	ROC-AUC (Train)	ROC-AUC (Test)
Logistic Regression	0.630731	0.626667	0.661598	0.661046	0.77802	0.773394	0.715102	0.712821	0.721345	0.717996

We got an increase of around 2.5% in recall after hyperparameter tuning with the following parameters for logistic regression:

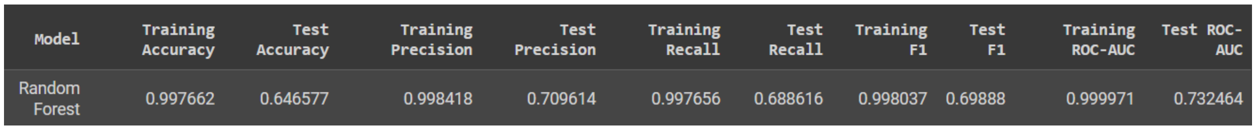
1. solver = 'liblinear'
2. penalty = 'l2'
3. C = 0.016900000000000002
4. threshold = 0.48

Random forest

For random forest, we tuned the following parameters:

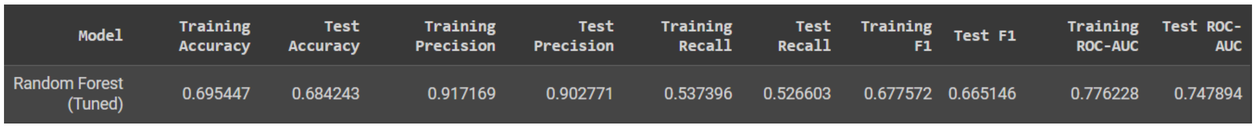
1. n_estimators
2. criterion
3. max_depth
4. min_samples_split
5. min_samples_leaf
6. max_features

Here's the initial evaluation of our random forest model.



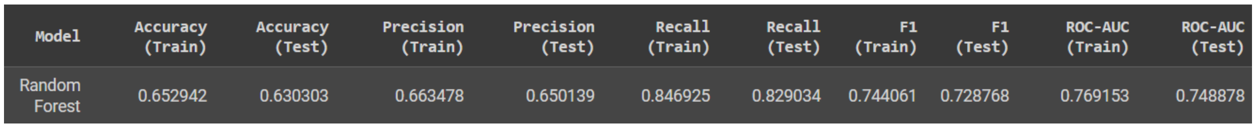
Model	Training Accuracy	Test Accuracy	Training Precision	Test Precision	Training Recall	Test Recall	Training F1	Test F1	Training ROC-AUC	Test ROC-AUC
Random Forest	0.997662	0.646577	0.998418	0.709614	0.997656	0.688616	0.998037	0.69888	0.999971	0.732464

And here's how the model performed on the training set (cross validated) after using the best hyperparameters. The model shows a good fit.



Model	Training Accuracy	Test Accuracy	Training Precision	Test Precision	Training Recall	Test Recall	Training F1	Test F1	Training ROC-AUC	Test ROC-AUC
Random Forest (Tuned)	0.695447	0.684243	0.917169	0.902771	0.537396	0.526603	0.677572	0.665146	0.776228	0.747894

Then, we tried to tune the threshold to improve the recall and got the following naive train-test evaluation result.



Model	Accuracy (Train)	Accuracy (Test)	Precision (Train)	Precision (Test)	Recall (Train)	Recall (Test)	F1 (Train)	F1 (Test)	ROC-AUC (Train)	ROC-AUC (Test)
Random Forest	0.652942	0.630303	0.663478	0.650139	0.846925	0.829034	0.744061	0.728768	0.769153	0.748878

We got an increase of around 14% in recall after hyperparameter tuning with the following parameters for random forest:

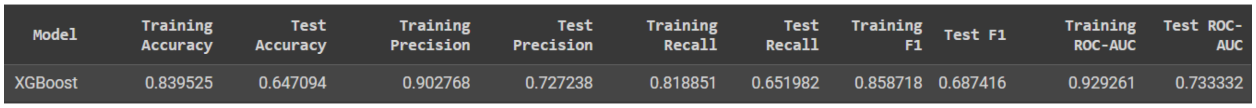
1. n_estimators = 4
2. min_samples_split = 7
3. min_samples_leaf = 6
4. max_depth = 6
5. criterion = 'gini'
6. max_features = 'auto'
7. threshold = 0.419

<u>XGBoost</u>

For XGBoost, we tuned the following parameters:

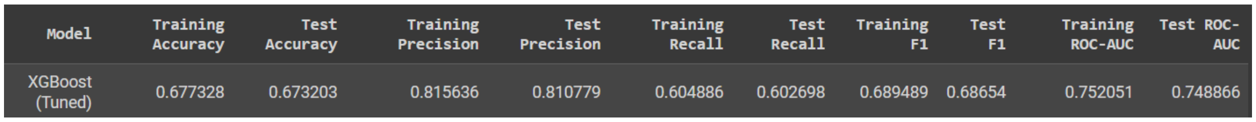
1. max_depth
2. min_child_weight
3. gamma
4. tree_method

Here's the initial evaluation of our XGBoost model.



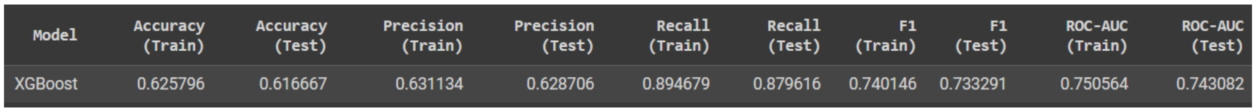
Model	Training Accuracy	Test Accuracy	Training Precision	Test Precision	Training Recall	Test Recall	Training F1	Test F1	Training ROC-AUC	Test ROC-AUC
XGBoost	0.839525	0.647094	0.902768	0.727238	0.818851	0.651982	0.858718	0.687416	0.929261	0.733332

And here's how the model performed on the training set (cross validated) after using the best hyperparameters. The model shows a good fit.



Model	Training Accuracy	Test Accuracy	Training Precision	Test Precision	Training Recall	Test Recall	Training F1	Test F1	Training ROC-AUC	Test ROC-AUC
XGBoost (Tuned)	0.677328	0.673203	0.815636	0.810779	0.604886	0.602698	0.689489	0.68654	0.752051	0.748866

Then, we tried to tune the threshold to improve the recall and got the following naive train-test evaluation result.



Model	Accuracy (Train)	Accuracy (Test)	Precision (Train)	Precision (Test)	Recall (Train)	Recall (Test)	F1 (Train)	F1 (Test)	ROC-AUC (Train)	ROC-AUC (Test)
XGBoost	0.625796	0.616667	0.631134	0.628706	0.894679	0.879616	0.740146	0.733291	0.750564	0.743082

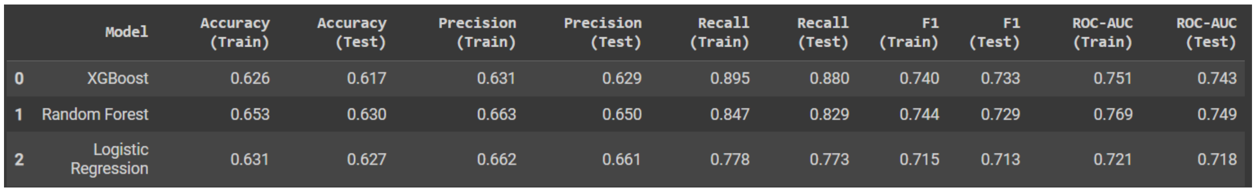
We got an increase of around 23% in recall after hyperparameter tuning with the following parameters for XGBoost:

1. max_depth = 2
2. min_child_weight = 29
3. gamma = 5.0
4. tree_method = 'exact'
5. threshold = 0.447

Final Evaluation for Model Selection

<u>Naive train test evaluation</u>

In this evaluation, we selected the model with the best recall score but not too overoptimistic and still has a decent accuracy and AUC score. Here's the final result.

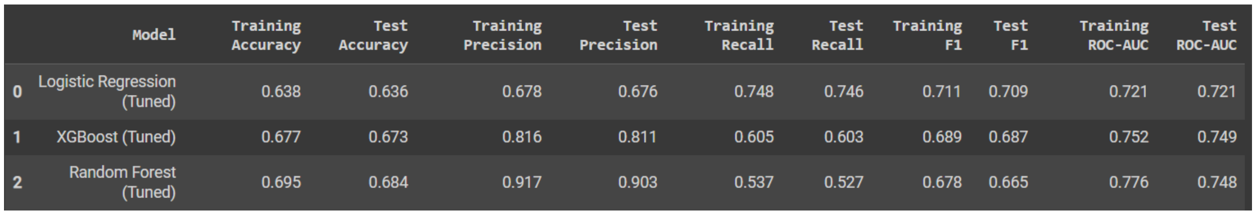


	Model	Accuracy (Train)	Accuracy (Test)	Precision (Train)	Precision (Test)	Recall (Train)	Recall (Test)	F1 (Train)	F1 (Test)	ROC-AUC (Train)	ROC-AUC (Test)
0	XGBoost	0.626	0.617	0.631	0.629	0.895	0.880	0.740	0.733	0.751	0.743
1	Random Forest	0.653	0.630	0.663	0.650	0.847	0.829	0.744	0.729	0.769	0.749
2	Logistic Regression	0.631	0.627	0.662	0.661	0.778	0.773	0.715	0.713	0.721	0.718

<u>Cross validation evaluation</u>

This evaluation aims to see if the model is indeed a good fit (good generalization). It seems that all of the models are indeed a good fit.

Note: We may see that the recall score has dropped significantly. This is because we didn't apply any threshold tuning to them and we just want to see the 'generalizability' of the model itself

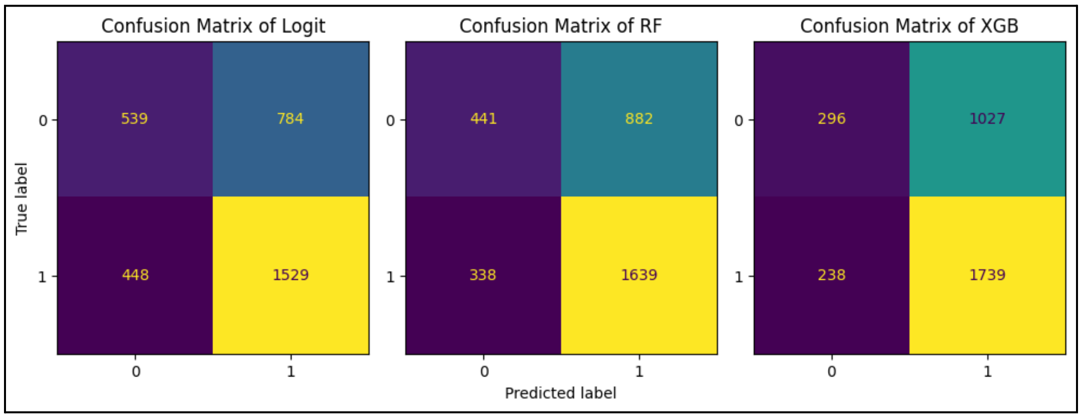


	Model	Training Accuracy	Test Accuracy	Training Precision	Test Precision	Training Recall	Test Recall	Training F1	Test F1	Training ROC-AUC	Test ROC-AUC
0	Logistic Regression (Tuned)	0.638	0.636	0.678	0.676	0.748	0.746	0.711	0.709	0.721	0.721
1	XGBoost (Tuned)	0.677	0.673	0.816	0.811	0.605	0.603	0.689	0.687	0.752	0.749
2	Random Forest (Tuned)	0.695	0.684	0.917	0.903	0.537	0.527	0.678	0.665	0.776	0.748

From both evaluations, we can see that Random Forest is the best model because:

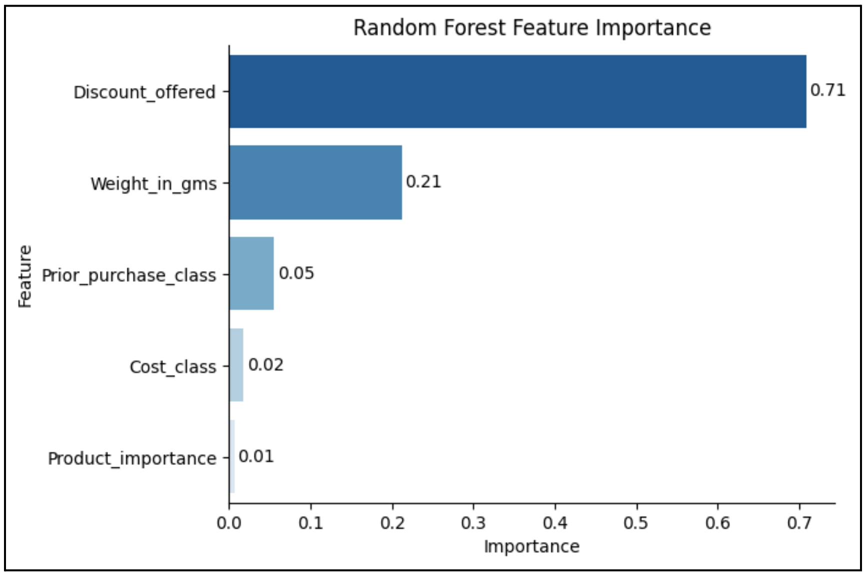
- Has good recall score and not over optimistic
- Has higher accuracy and AUC than other models

<u>Confusion matrix evaluation</u>



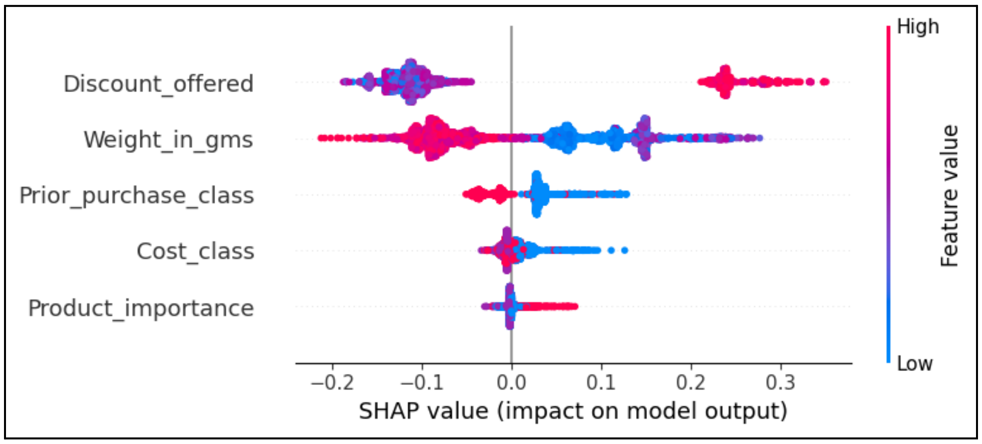
The confusion matrix shows that random forest predicts late deliveries more accurately, but at the cost of increased false positives (on-time deliveries that are predicted late) but not as much as XGBoost. This is not a major problem, because the delivery will still be on time regardless of the late prediction. The core problem is late deliveries that are predicted on time, which can make customers unhappy and less trusting of our company. Random forest does a decent job of avoiding this problem than Logistic Regression although XGBoost does this better but it produces too many False Positives.

Feature Importance



Discount is the most important factor associated with delivery status, followed by weight. Prior purchase, cost of product, and product importance have a minor association with delivery status.

But we don't know the direction of these features towards the delivery status. For that, we used SHAP values.



Remember that a positive output (True / 1) in our dataset indicates a late delivery. Based on the summary plot above, we inferred the following:

- Higher discounts are more likely to be associated with late deliveries. This means that the more discount an order has, the higher the probability that it will be late.
- Lower product weights are more likely to be associated with late deliveries. This means that the lighter the weight of a product, the higher the probability that it will be late.

- Interestingly, people with more prior purchases are slightly less likely to experience late deliveries.
- Lower cost of products tend to be associated with late deliveries.
- Higher product importance tends to be associated with late deliveries.

Based on these insights. We can take the following interim actions before any further analysis:

1. <u>Implementing a Discount Threshold Policy (simulated)</u>
   We propose to simulate the impact of setting a maximum discount threshold for orders. This policy could help balance attracting customers through discounts and ensuring on-time delivery. We will assess whether setting a discount threshold of 10% in this dataset will lead to an increase in on-time delivery rates.

   While this may increase on-time delivery rates, we need to investigate further why higher discounts lead to late deliveries. For this analysis, we would need more relevant data, such as detailed warehouse and logistics data. The dataset in this project is insufficient to identify the root causes of lateness at higher discount rates.

   There is a risk that some customers may lose interest if we limit discounts. We need to clarify that this limit is temporary while we investigate the causes of late deliveries at higher discount rates and develop solutions. This will make us more trustworthy by demonstrating our transparency.

   Because we are reducing discounts, we need to offer something else to customers in their place. This could be loyalty rewards, points, or early access to new products. This way, customers can still maintain their interest, albeit in a different form than discounts.

2. <u>Bundled Products Shipment (simulated)</u>
   One way to reduce the likelihood of late deliveries for lighter products is to bundle them together into heavier packages. For example, we could bundle several small toys together based on their region or address and count them as a single product in the logistics process. Once the bundles arrive at the local shipment warehouse, they can be unpacked.

   This process would not be visible to customers. If we have a mobile app that tracks orders, each customer's order would still appear as a single order, not mixed with other customers' orders. This is because the bundling would only occur during the logistics process until the bundles reach the local shipment warehouse.

   However, we still need to investigate the root causes of why lighter products are more likely to be late. For this, we would need the same dataset as before, including warehouse and logistics data. The dataset in this project is not sufficient to identify the causes of late deliveries for lighter products.

There is one last prescriptive recommendations for those customers who will experience late deliveries:

1. <u>Inform customers of the delay and extend the estimated delivery date.</u>
   The first thing to do is to contact/inform the customer that the delivery will be delayed so that they will not be surprised if the product hasn't been delivered by the estimated delivery date. Then, we can add x days to the estimated delivery date to extend the range.

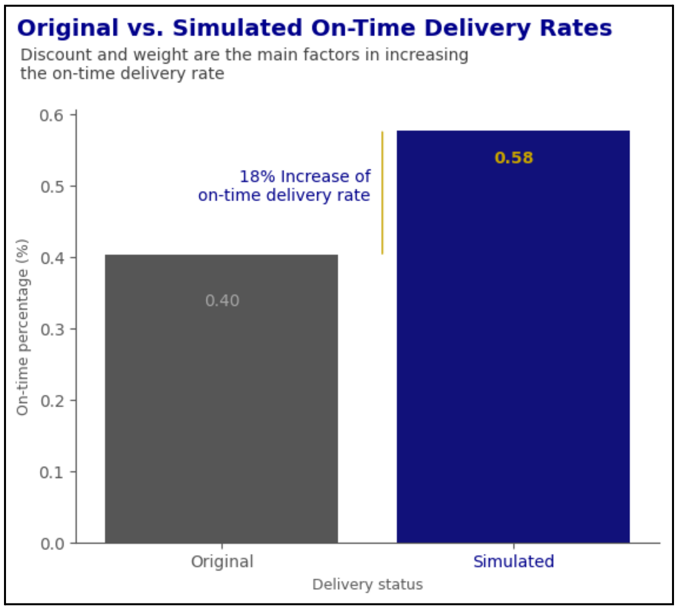
2. <u>Offer incentives to customers who experience late deliveries</u>
   Because higher discounts in this dataset are associated with lateness, we can offer other alternative incentives to them, such as loyalty points, free delivery on the next order, exclusive access to new product/market sections, and many more. Consider offering customers the option to choose the incentive that they prefer. This will give customers more control over the situation and make them more likely to be satisfied with the outcome.

Business Recommendations Simulation

In this process, we tried to find the best range of numbers that the model will predict more on-time deliveries so that we can see the predicted increase in our main metric: on-time delivery rate. We create a new simulation dataset that changes the following features' values:

1. 'Discount_offered'
2. 'Weight_in_gms'
3. (Optional) 'pp_class' (Prior purchase class)

Here's the best scenario that shows the predicted increase in our main metric: on-time delivery rate.



The following are the interim solution/action that we could take based on the simulation:

1. Surprisingly, to better increase our on-time delivery rate we need to put a 5% threshold instead of 10%. After carefully reviewing the visualization of Weight vs. Delivery Status relationship, we can see that there are even some late deliveries in the 5% to 10% range.
2. We need to 'bundle' products with weight less than 5000 gr to be 5000 gr in the logistic process to increase the on-time delivery rate. Therefore, there needs to be an additional shipment identifier for each bundle of the shipment later on.
3. There is one more catch: The more prior purchases that a customer has, the more likely that their delivery will be on-time. This should be further analyzed later on because we're neglecting these new customers' delivery commitment!