Cairo University

Faculty of Engineering

Data Mining, Big Data and Analytics
Project
"Instacart Market Basket Order Prediction"

Submitted by

Aya Abdul Wahab	**1170127**
Farah Amr Ayad	**1170455**
Hana Amr Mahmoud	**1170510**
Youmna Khaled Sayed	**1170499**

Submitted to

Eng. Mostafa Hazem

1. Problem Formulation

Instacart is a grocery shopping service. Users order their groceries through an app. The Instacart Market Basket Analysis dataset was engineered for a specific application: to try to predict which items a customer would order again in the future.

This is a classification problem. Our goal is to predict whether a customer will reorder a product based on his prior purchases.
Using the dataset mentioned below, we will be able to predict the next order of the customer based on his behaviors.

2. Project Pipeline

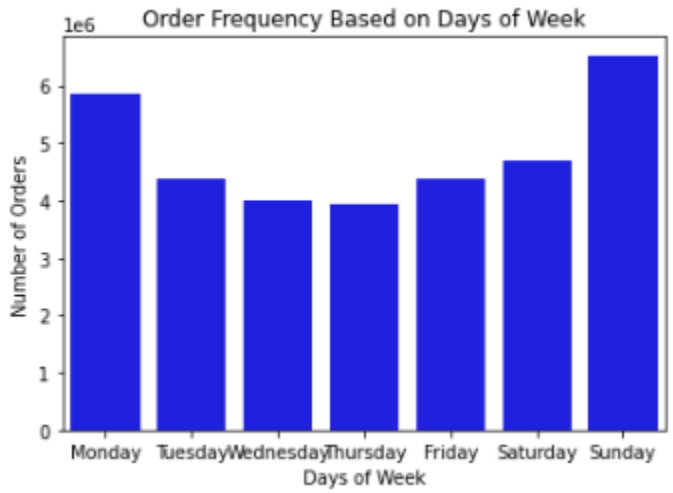
After data collection, we will apply the following flow:

1. **Data Cleaning**
    - In the orders csv file, The Nan days since prior order were replaced **by the mean of the column as the Nan determines that these orders are being ordered by this specific customer for the first time.**
    - In the order_products csv file, There are 3 products with Nan order_id: 3 products that were never ordered, so they were dropped from the dataframe.
    - Finding duplicates and removing them
        - ★ In our dataset luckily we don't have any duplicates, but we added the check as we didn't know beforehand.
    - We investigate inconsistencies and Find outliers and deal with them
2. **Data Visualization and Exploratory Data Analysis**
    - To understand data and know if we need to add more features.
3. **Feature Selection/Engineering**
    - Extracting new features by joining between different tables (After running the models on the original features, the performance was very bad, that's why we decided we should go and find new features)
    - Visualize correlations among different features using correlation matrix and plotting heat map.
    - Removing outliers and Nan values.
    - Data scaling.
    - Merge features with y_train (**order_products_train**), to know if reordered or not.
      Note: Not all products were found, so a lot of nan values were found and were filled with zeros.
4. **Split the processed data**
    - In an 80:20 ratio, split the data into training sets and validation sets.

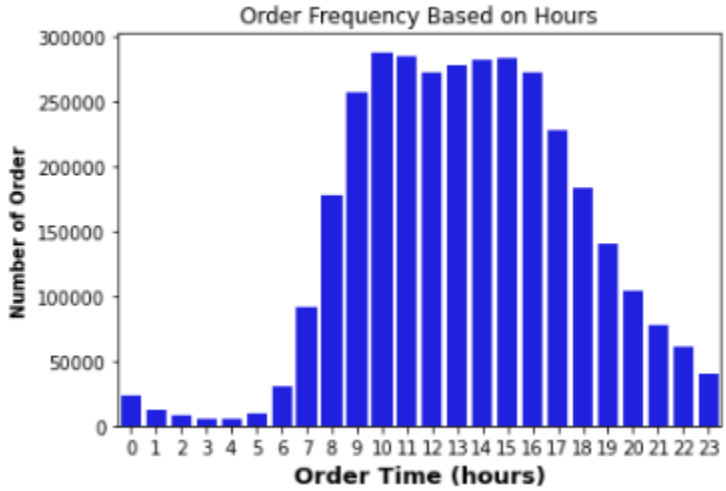
5. **Select and train models**
    - Train different models on the train set
6. **Validate the model**
    - Test the model on the validation set and determine the evaluation metrics that will be used to assess the different models.
7. **Model evaluation**
    - Select the best model based on the following evaluation metrics: Accuracy, Precision, Recall, Log Loss and Auc Roc.
    - We noticed, Precision and Recall are very low, due to **biased data** (a lot of reordered=0), because already the cart size is very small.
    - To deal with this, we did oversampling and undersampling.
      Random oversampling involves randomly selecting examples from the minority class, with replacement, and adding them to the training dataset.
      Random undersampling involves randomly selecting examples from the majority class and deleting them from the training dataset. This can balance the class distribution but does not provide any additional information to the model. An improvement on duplicating examples from the minority class is to synthesize new examples from the minority class. This is a type of data augmentation for tabular data and can be very effective using Synthetic Minority Oversampling TEchnique, or **SMOTE** for short.
    - After doing so, the Precision and Recall went very high.

Data Visualization & Extracting Insights From Data

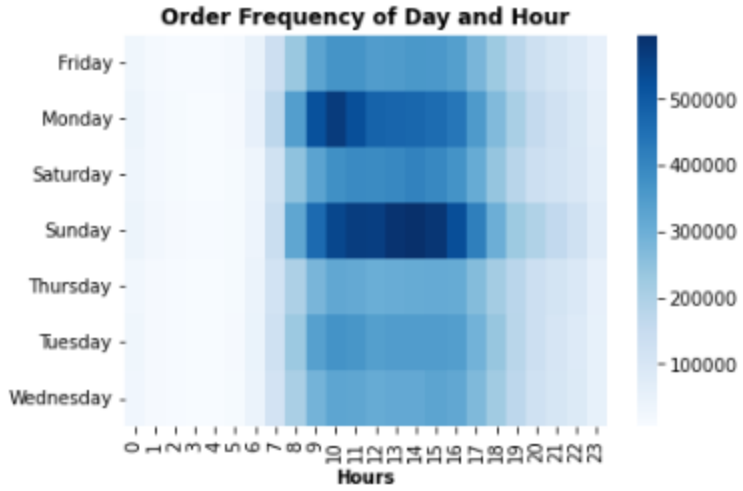
- First, In order to understand the data to be able to extract useful features from it, we need some data visualizations.



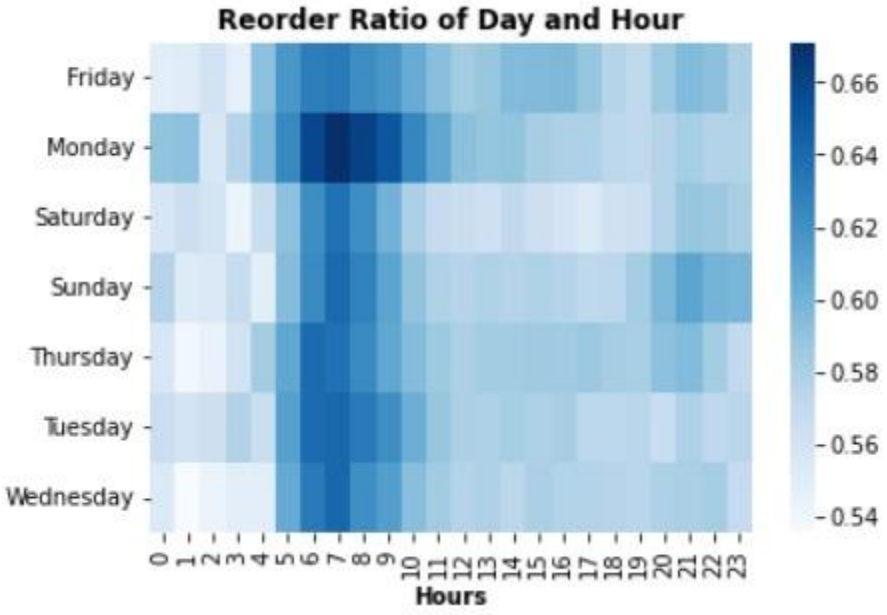
Insight 1: The frequency of orders is higher at the beginning of the week (Sundays and Mondays).



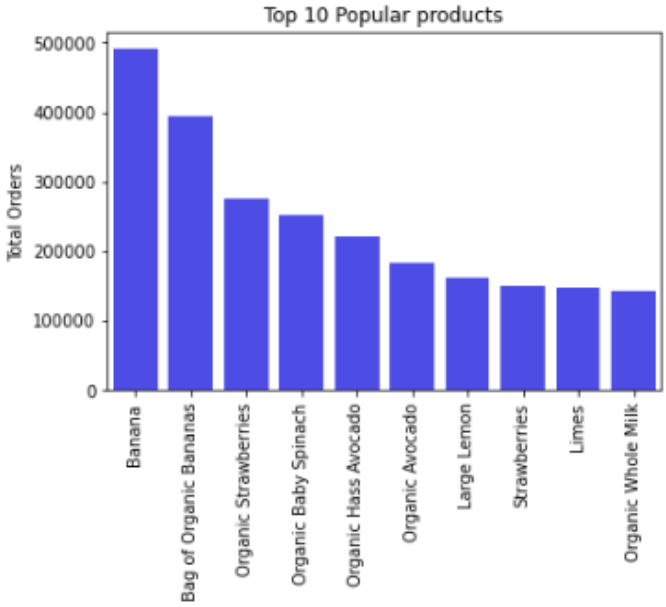
Insight 2: The frequency of orders is higher in the middle of the day (10 am to 4 pm) and is very low from midnight to 5 am.



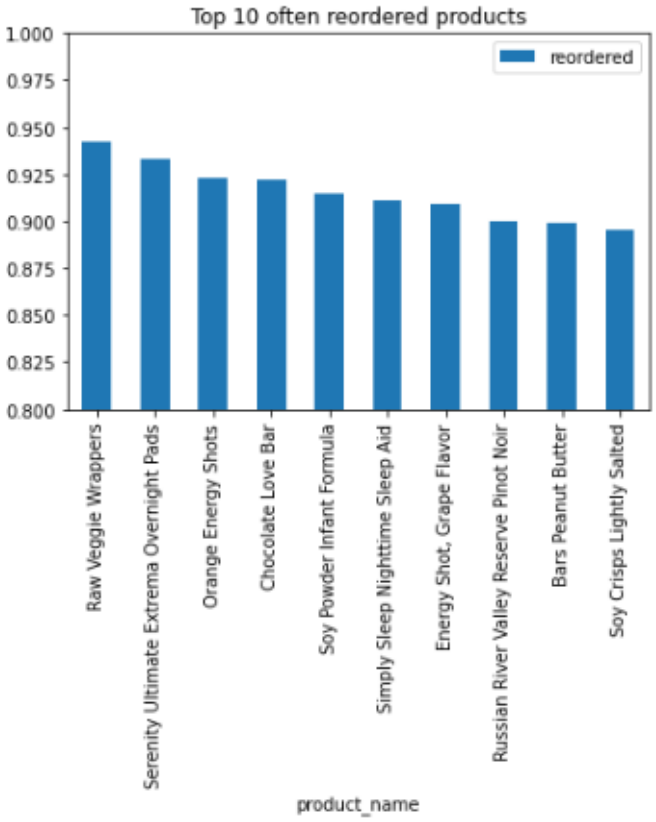
Insight 3: Another representation of the previous 2 graphs, which also shows that on Sundays mornings , the frequency of orders is the highest.



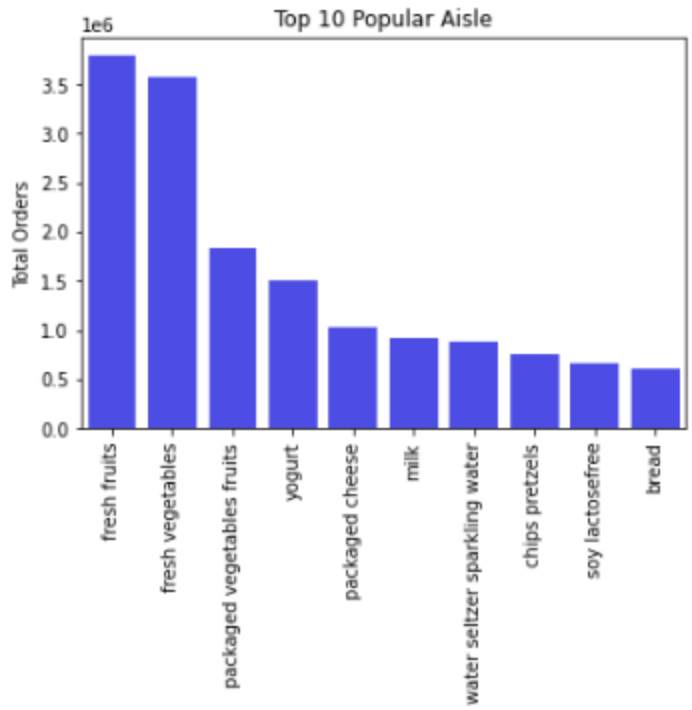
Insight 4: This graph shows, for each day and for each hour, the average reorder ratio for all users.



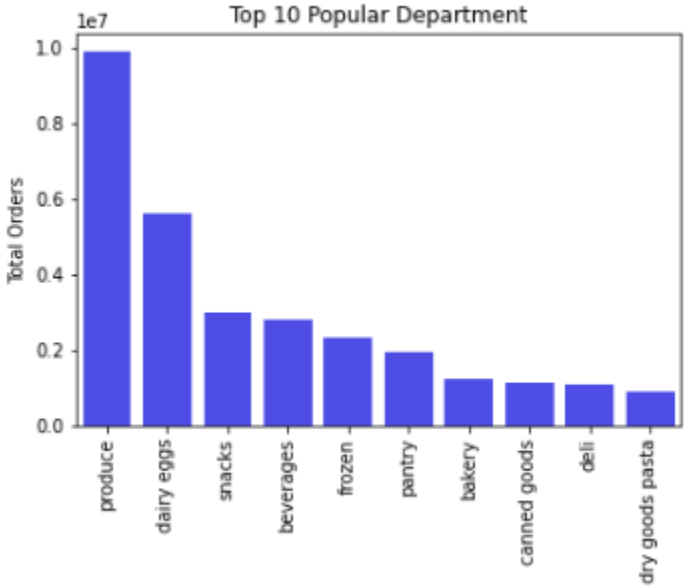
Insight 5: Bananas and organic bananas are the most popular products.



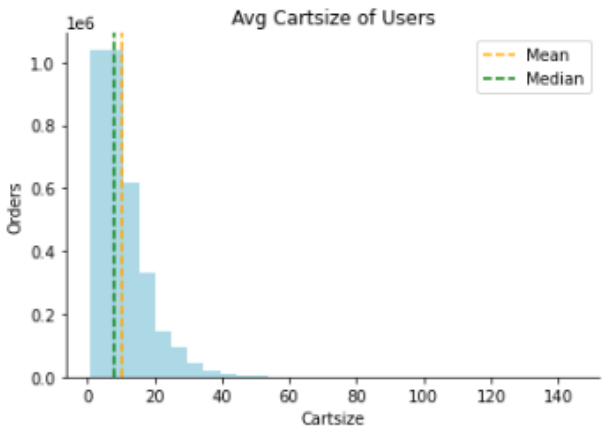
Insight 6: Raw veggies wrappers are the most reordered product



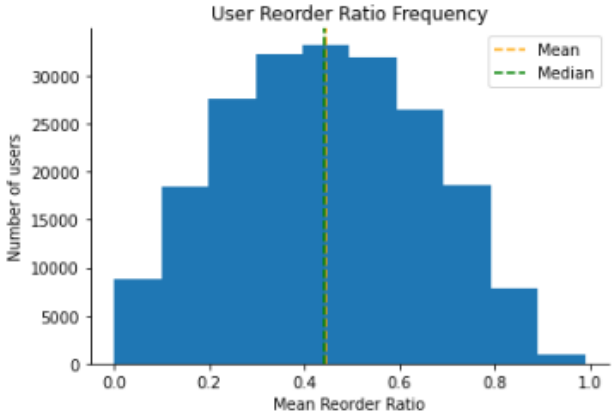
Insight 7: Fresh fruits and vegetables are the most popular aisle



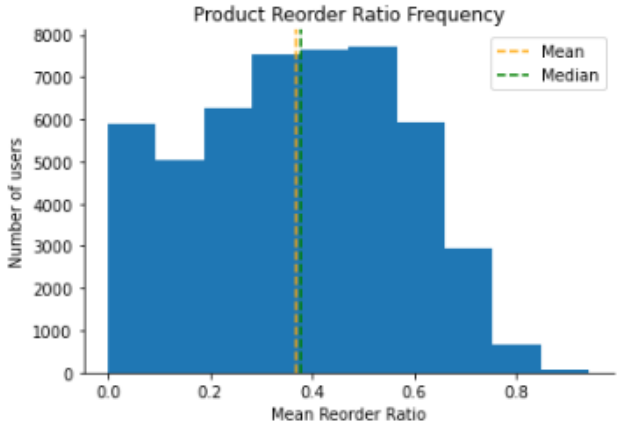
Insight 8: Dry goods pasta, deli, canned goods and bakery departments are the least popular.



Insight 9: This graph shows that the average cart size of users is 10.

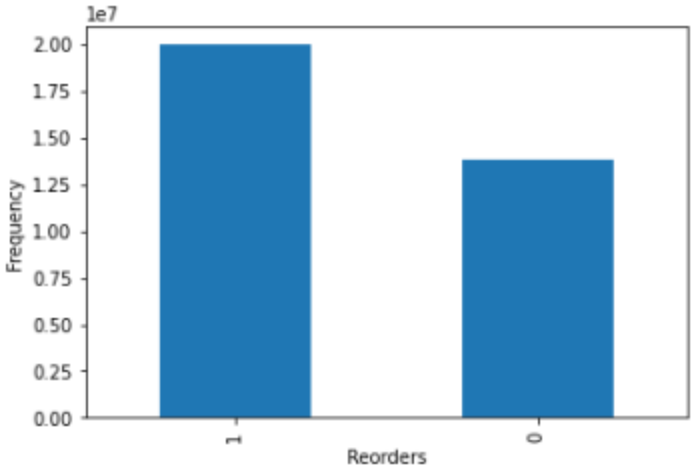


Insight 10: Most of the people have a mean reorder ratio of approx. 0.4, very few who always reorder and a good amount reorder less frequently.

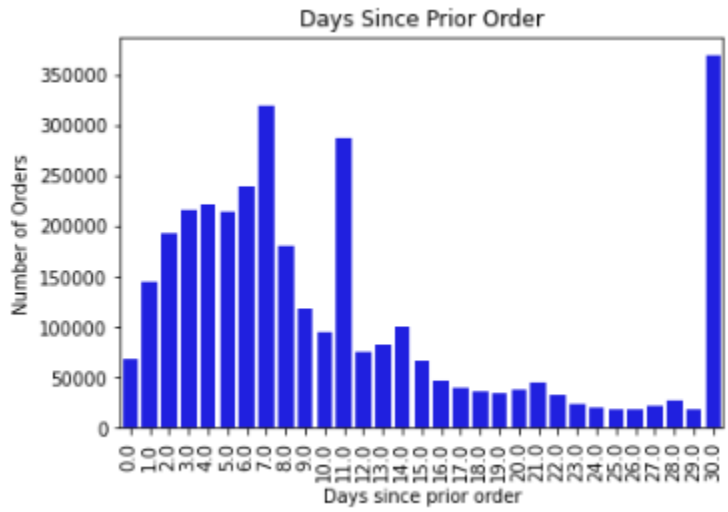


Insight 11: Most of the products have a mean reorder ratio from 0.3 to 0.4, very few get reordered a lot and a good amount of products do not get reordered.

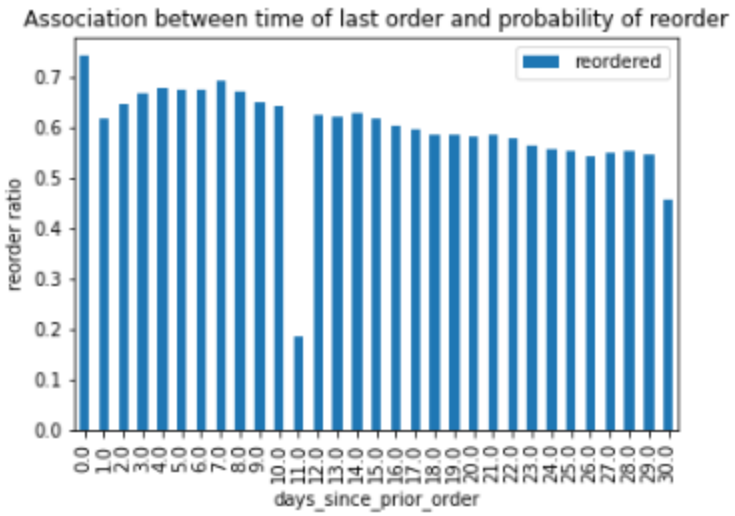
From the last 2 graphs, we saw we can add the features **u_reordered_ratio** and **p_reordered_ratio**



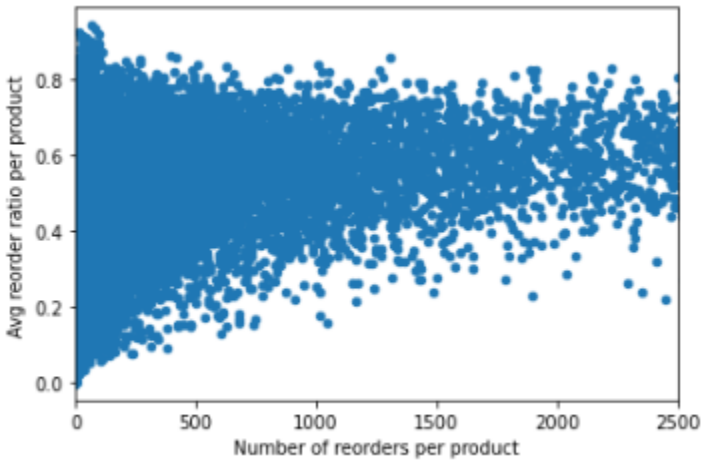
Insight 12: This graph shows that the frequency of products who get reordered is higher than those who do not get reordered.



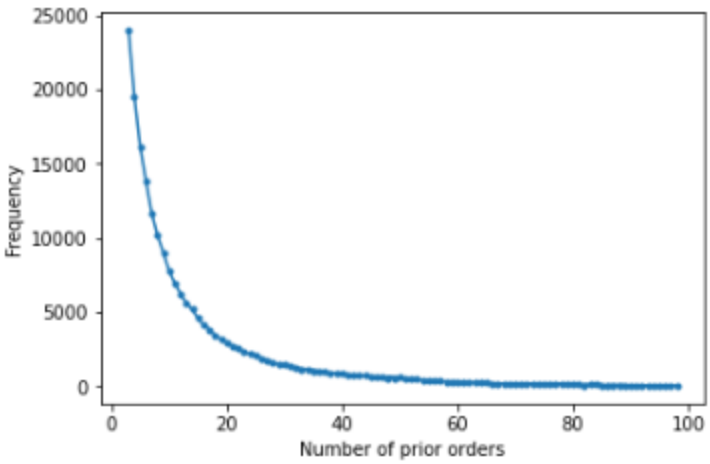
Insight 13: As the number of days increases, the number of orders decreases, until 30 days have passed, the number of orders starts to increase again.



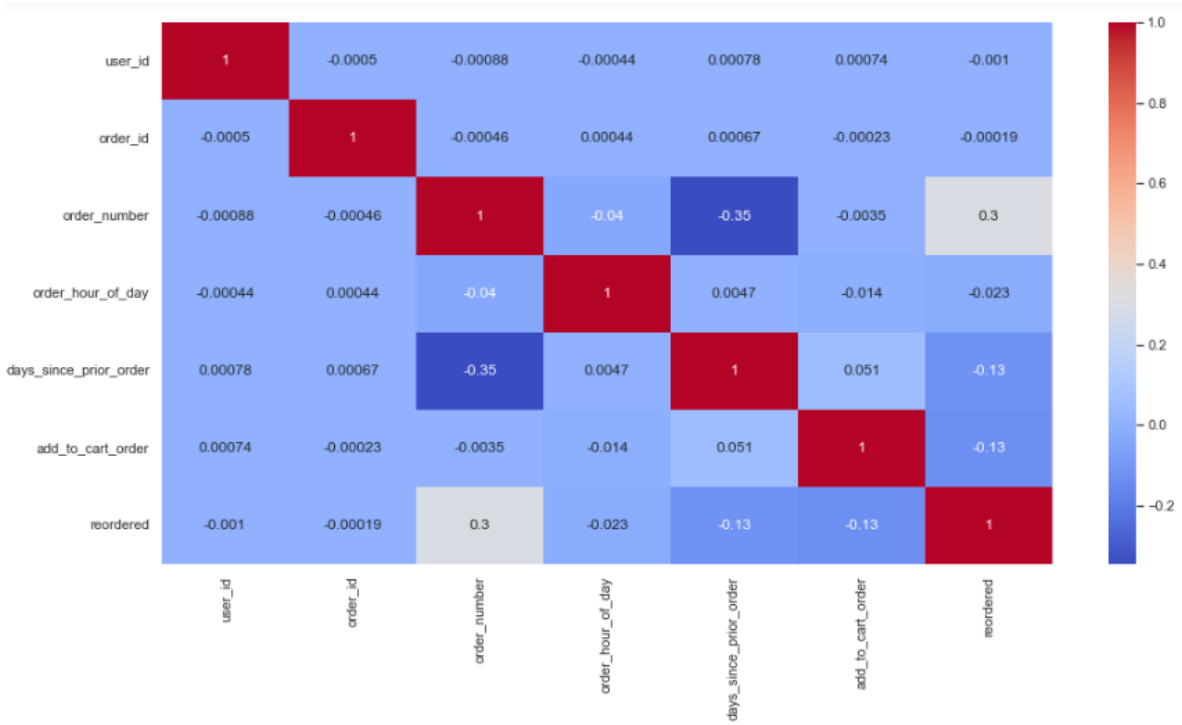
Insight 14: This graph is similar to the previous one but only with respect to the reorder ratio instead.



Insight 15: This graph shows the Association between product number of orders and probability of its reorder.



Insight 16: This graph shows the Number of prior orders per user.

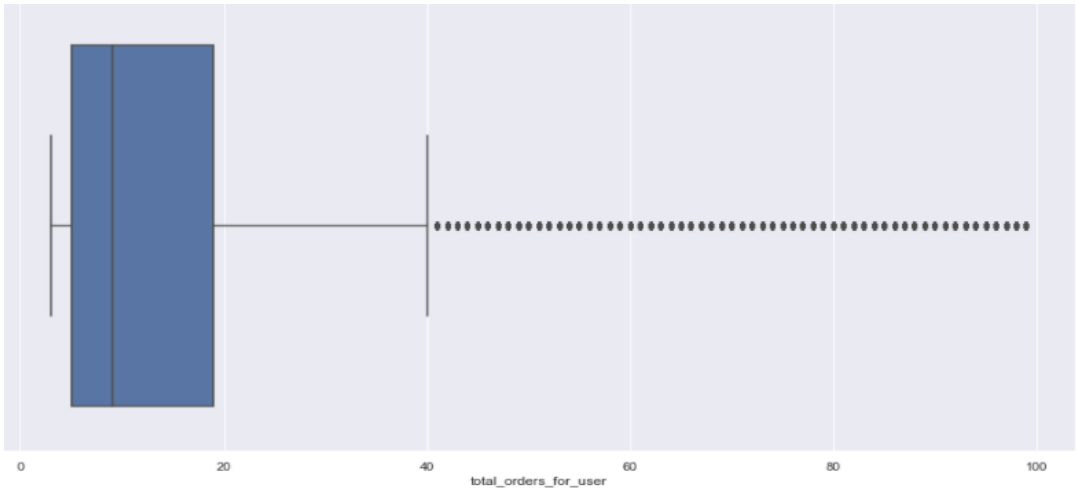


Insight 17: From the correlation matrix, we see that there is a slight correlation between reordered and order_number, which made us think of adding the feature **total_orders_for_user**. Also, there is a slight correlation between reordered and days_since_prior_order and add_to_cart_order , which made us think of adding the features **user_avg_days_since_prior_order** and **adding_product_to_cart_time** (user avg add to cart order for a certain product).

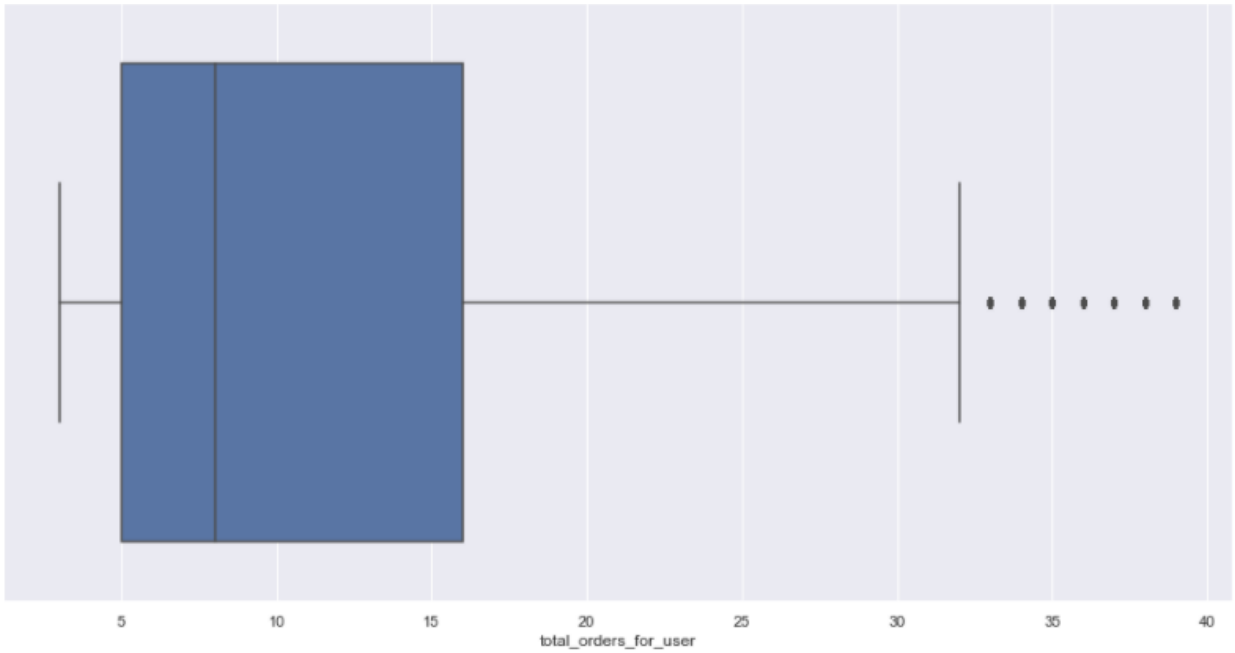
Extra Features:

From the previous data visualization, some new features were deduced such as:

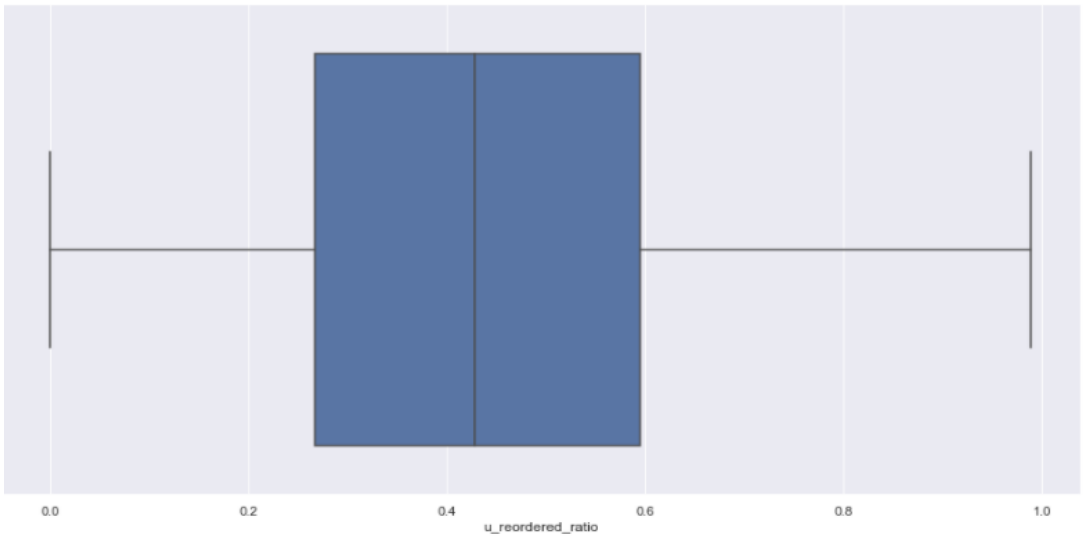
- Total_orders for users: how many orders were purchased by a certain user.



From the previous boxplot, it's clear that there are a lot of outliers that needs to be removed by limiting the points to those smaller than the Q3 + 1.5 IQR and greater than Q1 -1.5 IQR
We removed any customer that made total number of orders > 40

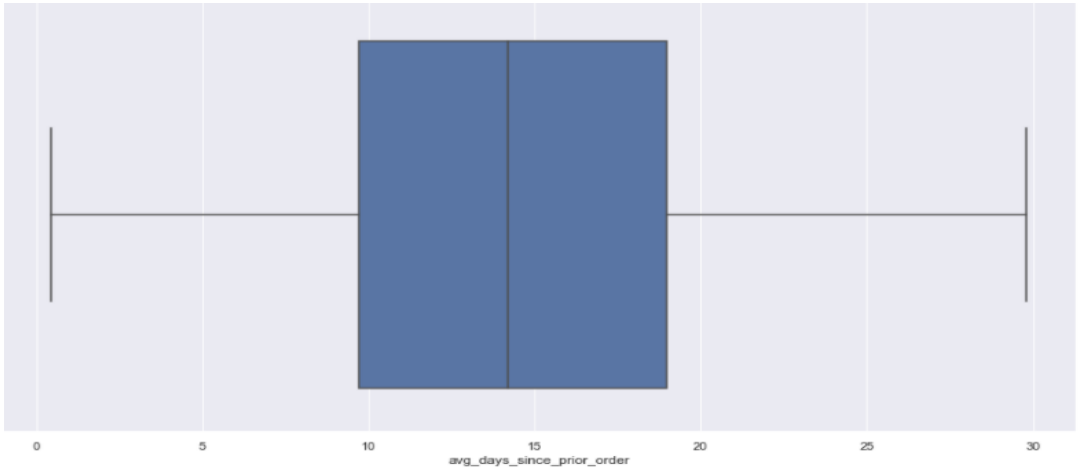


- U_reordered_ratio: Calculated the average of reorders made by each user



No outliers were found

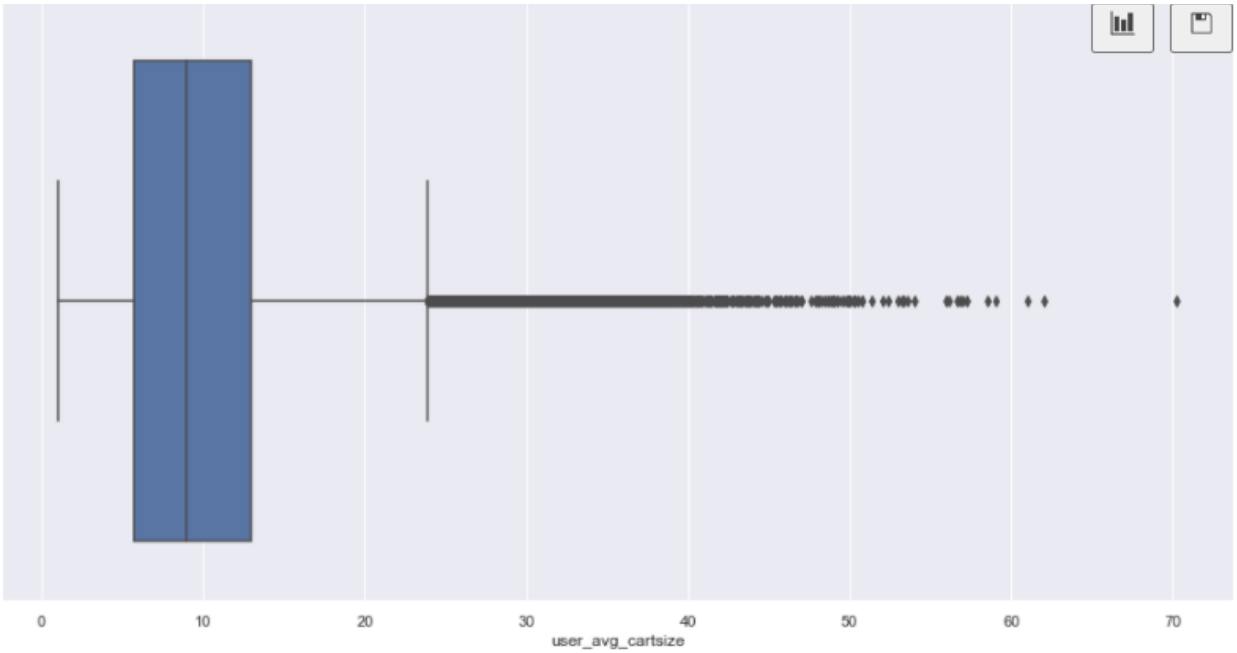
- Avg_days_since_prior_order: shows the frequency of ordering of each user by calculating how many days in average does the user waits to reorder



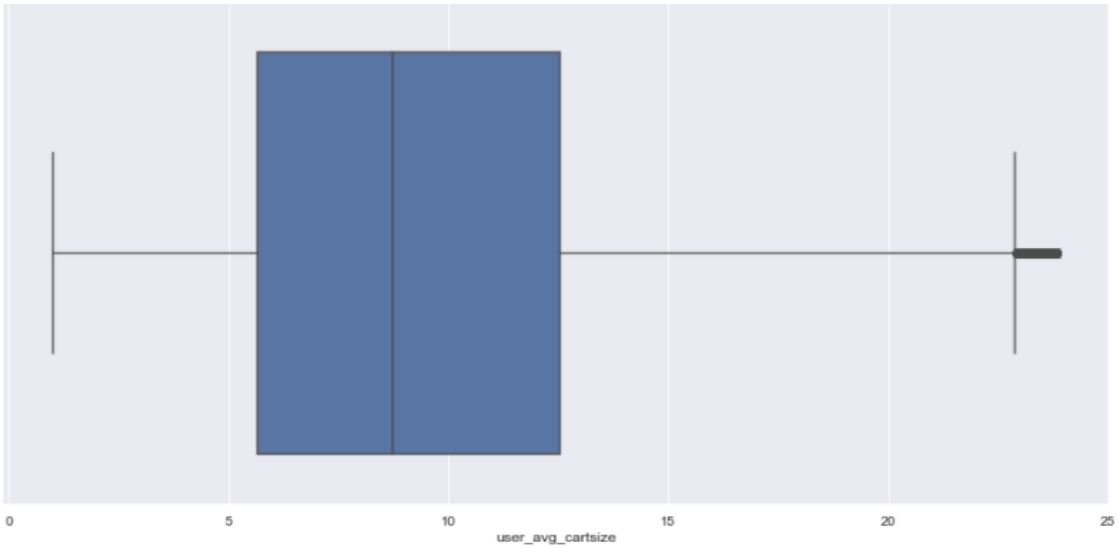
No outliers were found

Extra user features:

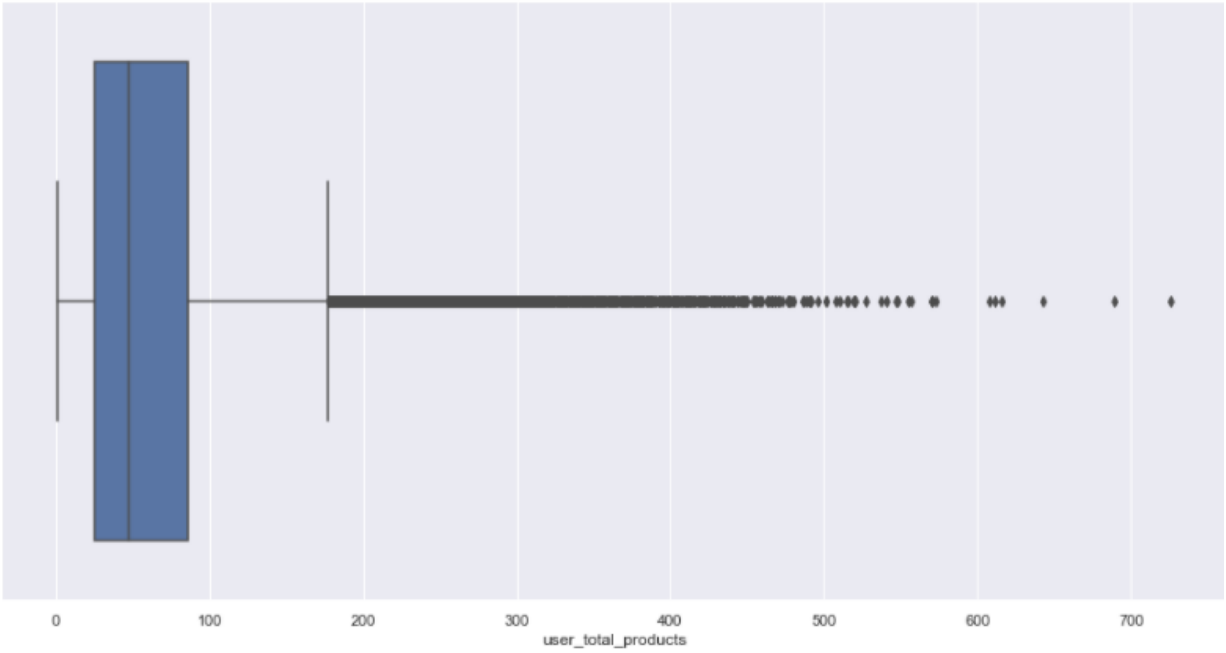
- User_avg_cartsize: average cart size for each user



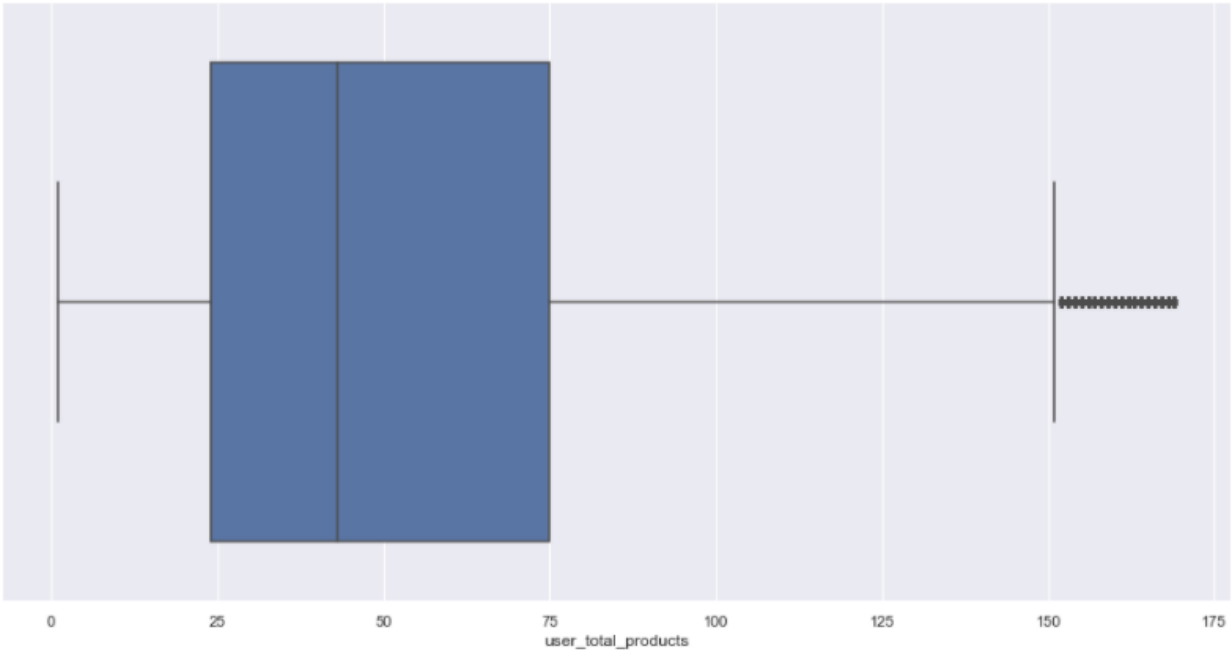
We removed any customer that had average cart size > 25



- User_total_products: Total number of unique products bought by user x

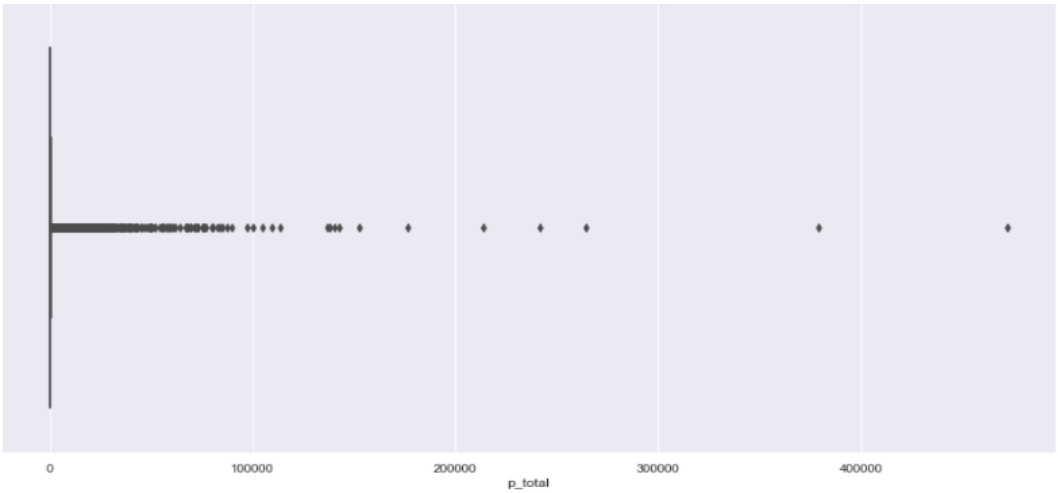


We removed any customer who bought number of unique products > 175

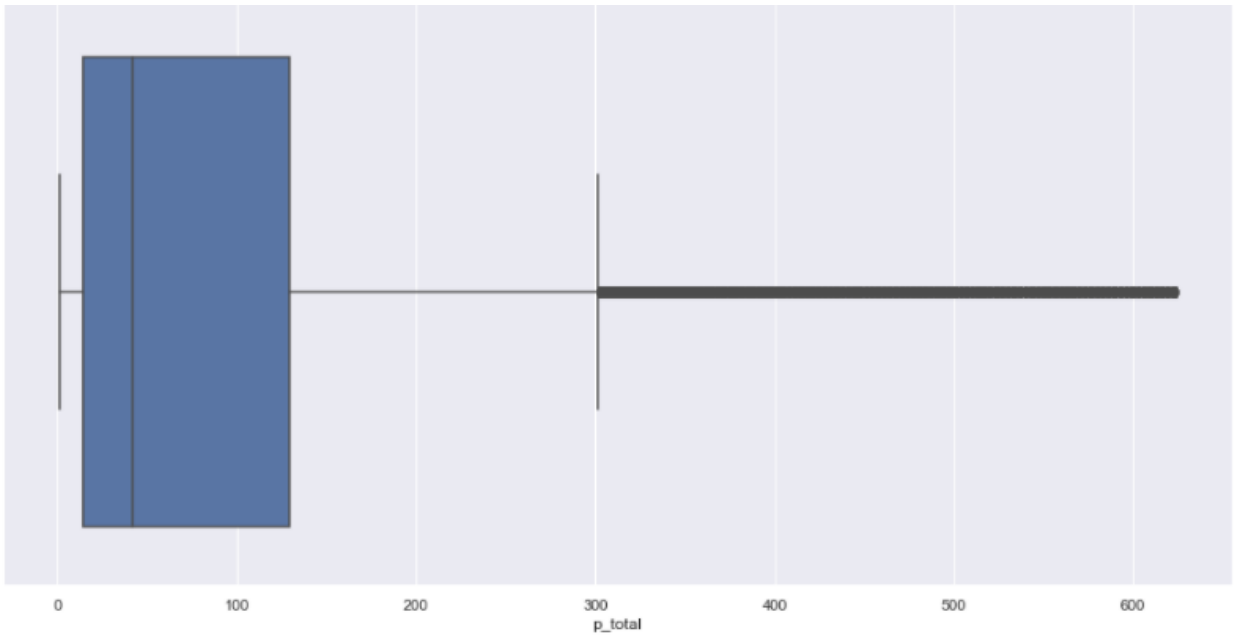


Extra user features:

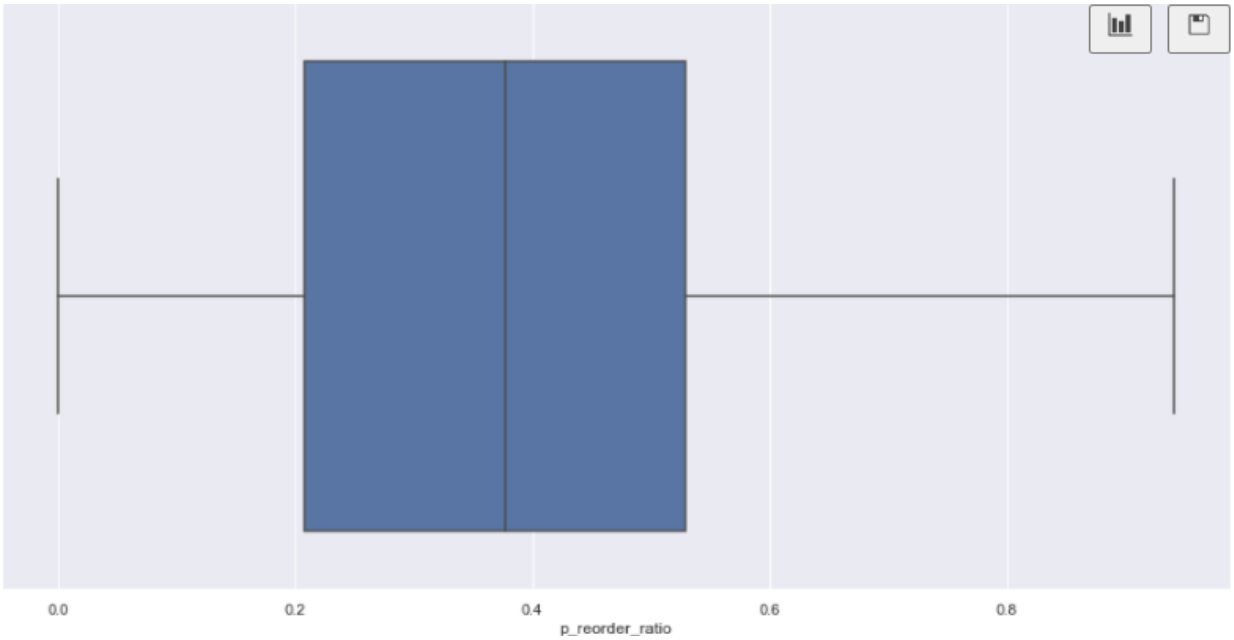
- P_total: total number of orders for each product



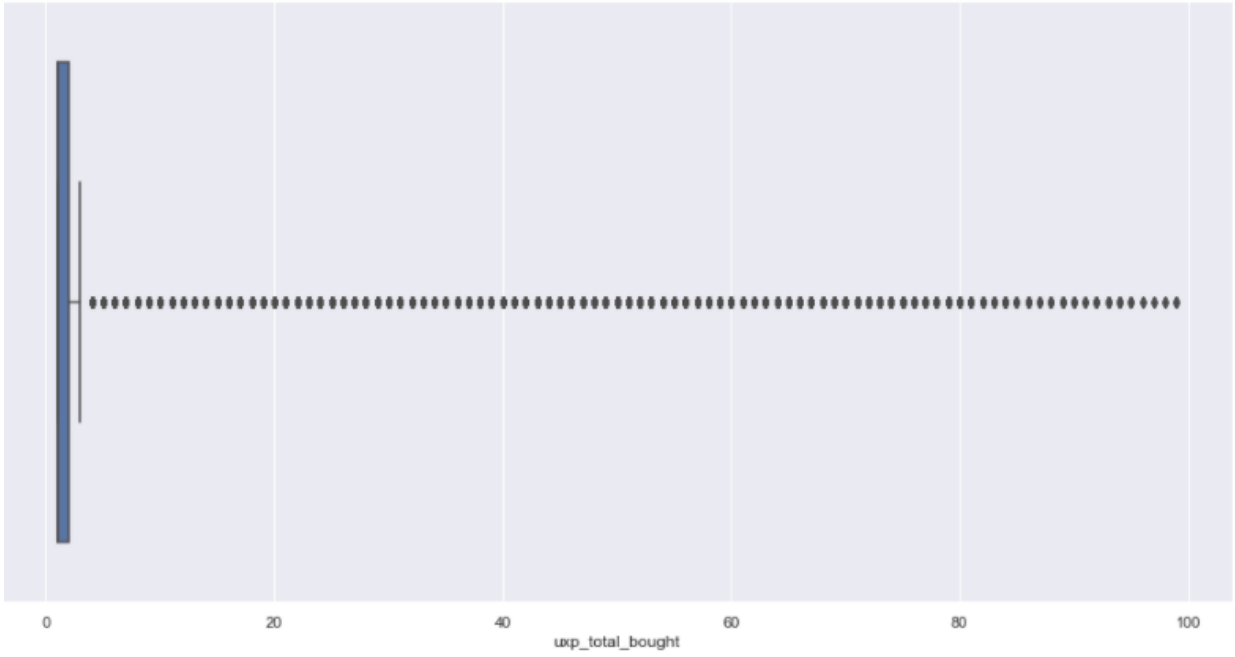
We removed any product that was bought > 650 times



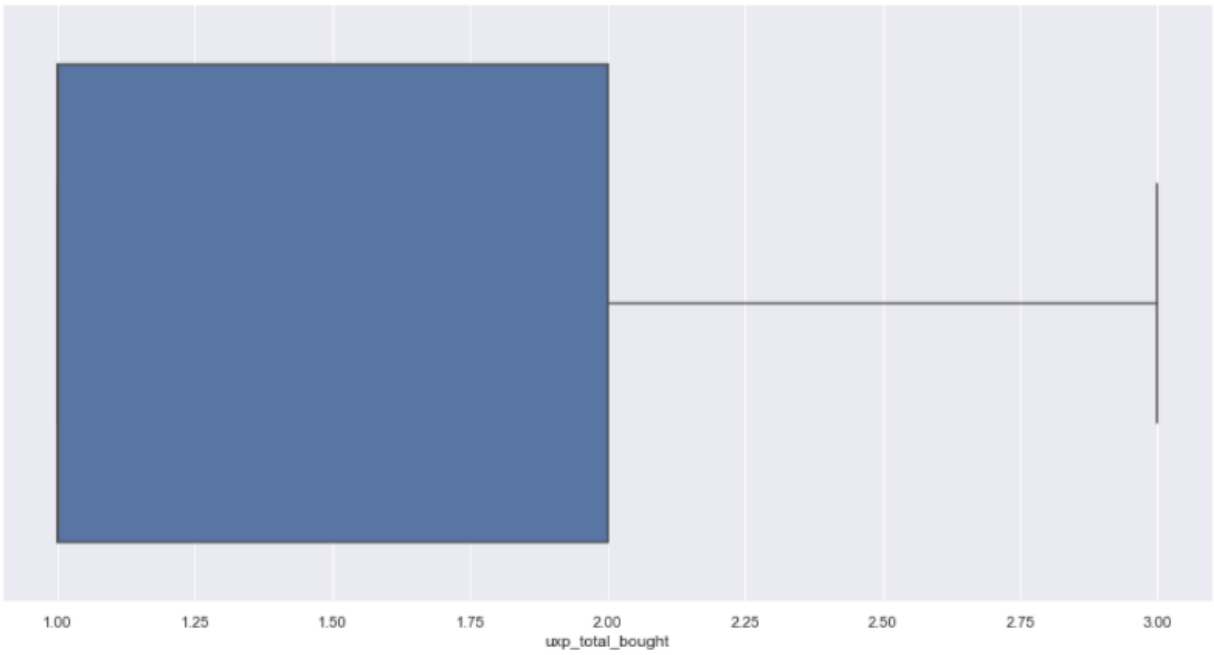
- P_reorder_ratio: average reorder ratio per product



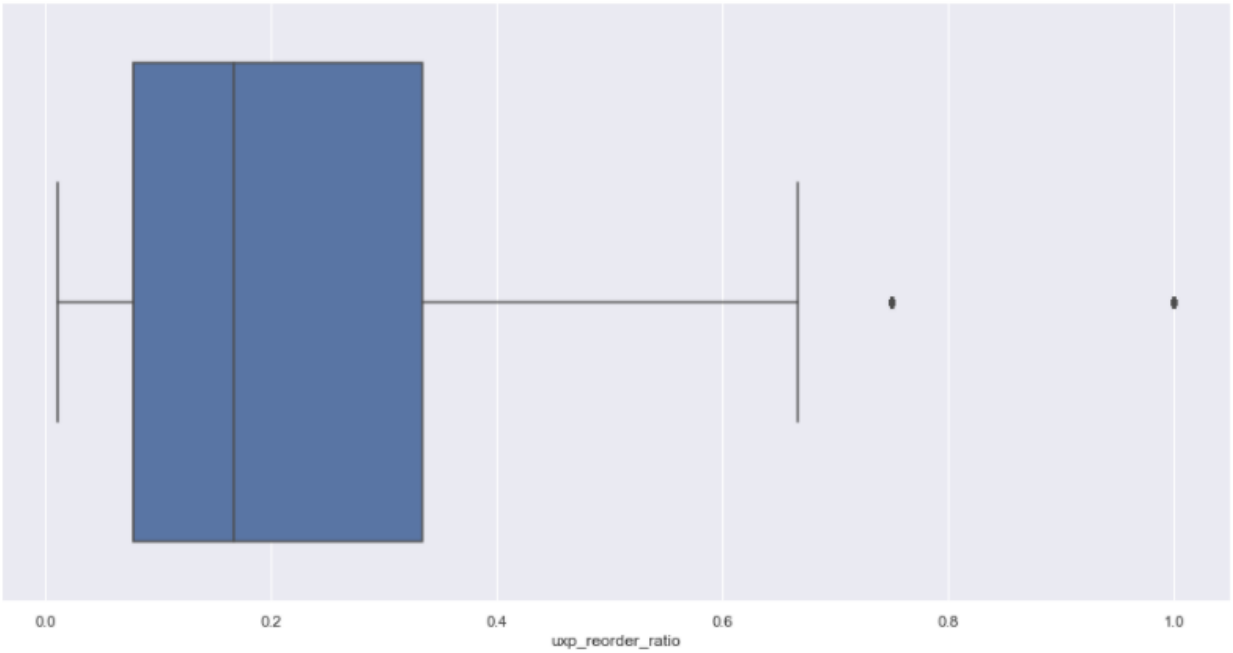
- Uxp_total_bought: number of times user X bought product Y



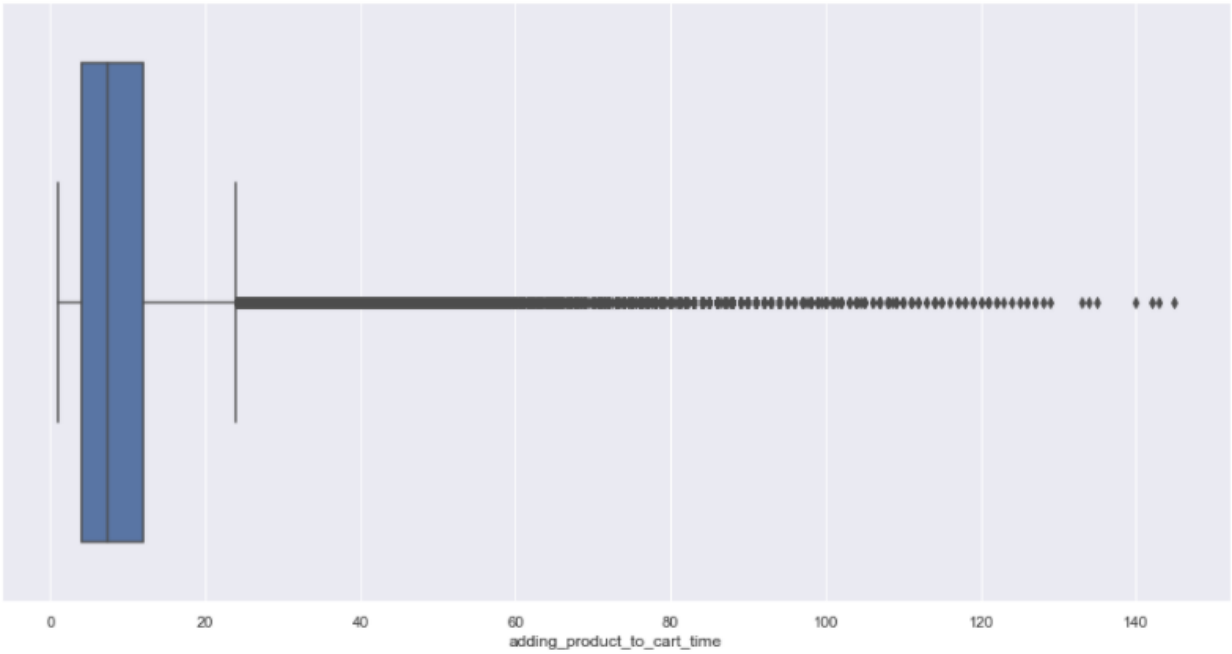
We removed any customer-product combinations that was repeated > 3 times



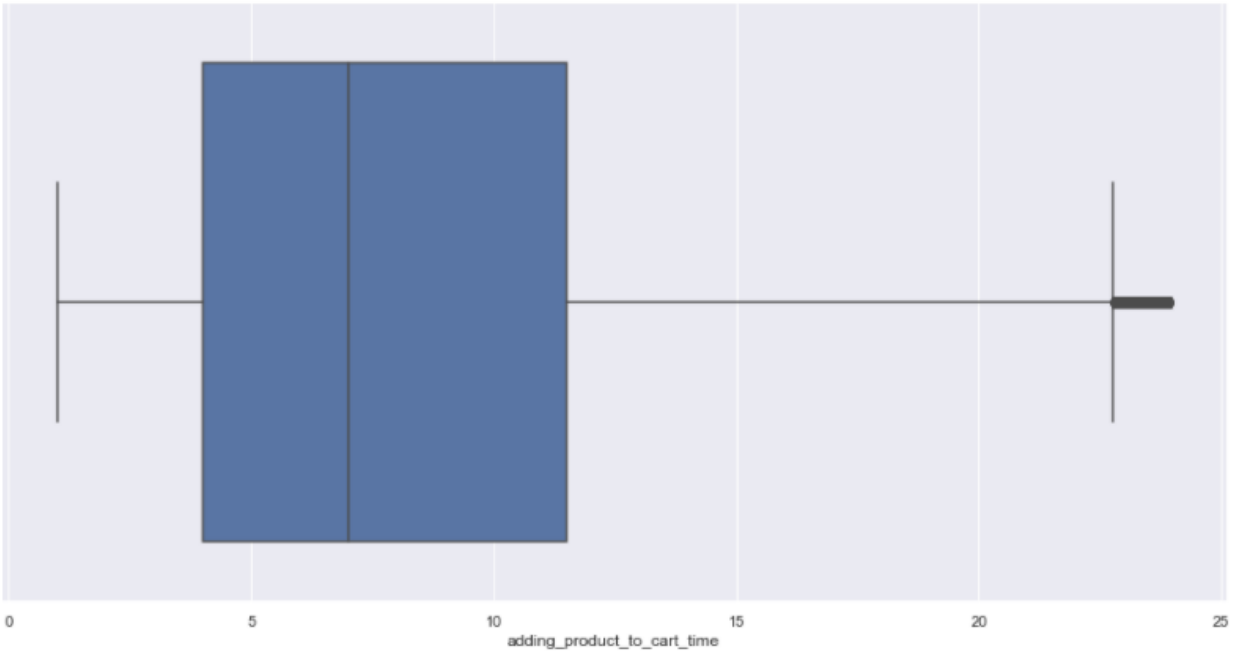
- Uxp_reorder_ratio: reorder ratio for user x on product Y which equals $(uxp\ total\ bought)/(total\ orders\ for\ user\ -\ first\ time\ user\ x\ bought\ product\ y)$



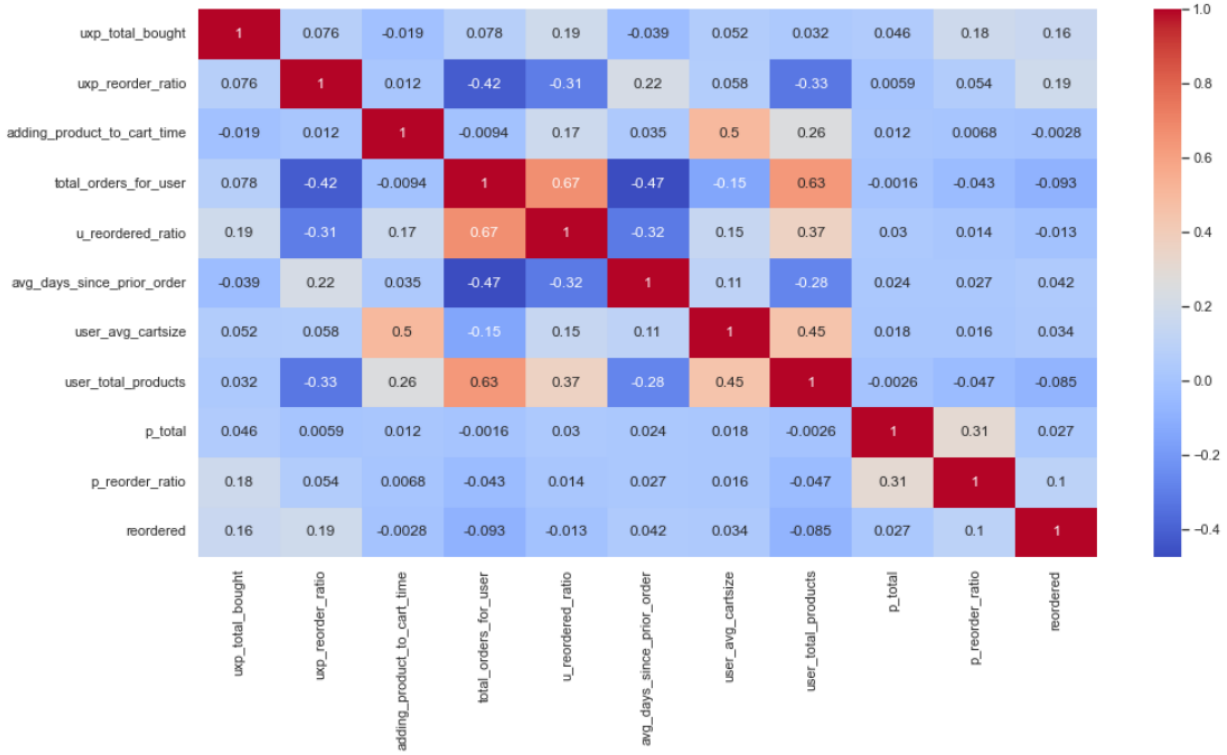
- adding _product_to_cart_time: average add to cart time for user X on product Y (1st, 2nd, 3rd, …etc item to be put into the cart)



We removed any customer-product combinations where the product was added to the cart > 25th item



- The correlation between the new features



Data Splitting

- It is critical to divide the dataset into training and validation sets since we do not have the truth values of the test set. By segmenting the data, we may assess how effectively a model responds to new data. This enables us to assess the model's performance accurately. Our dataset was divided in a 80:20 ratio.

Models Trained

- Feature vector in the following models is
    1. Uxp_total_bought
    2. Uxp_reorder_ratio
    3. Adding_product_to_cart_time
    4. Total_orders_for_user
    5. U_reordered_ratio
    6. Avg_days_since_prior_order
    7. P_total
    8. P_reorder_ratio
    9. User_avg_cartsize
    10. User_total_products

- The models we used are: Logistic Regression, KNN (K=3), Decision Trees, Random Forests, Adaboost, Gradient Boosting, and Gaussian NB

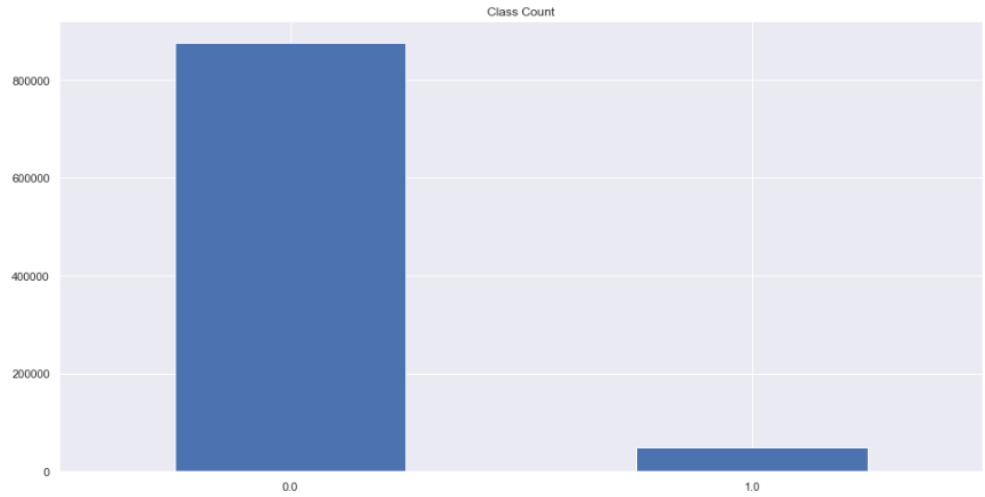
Results and Evaluation

- We have 4 trials for each model, one with the biased data, with undersampling, with oversampling and finally with SMOTE.

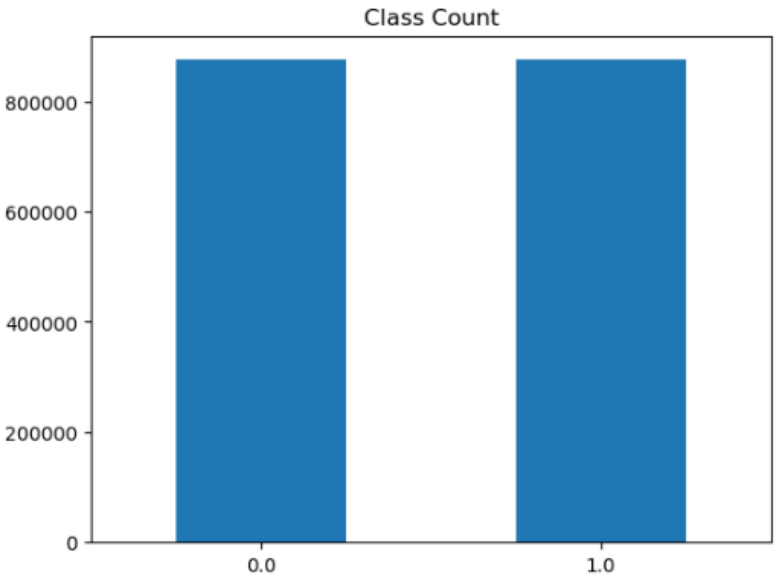
Biased Data

Classifier	Accuracy	Precision	Recall	F1Score	ROC AUC	Log Loss
LogisticRegression	94.596	2.503	52.621	4.779	73.662	0.188
KNeighborsClassifier	93.815	2.005	11.026	3.392	52.832	1.575
DecisionTreeClassifier	89.919	16.386	13.786	14.974	54.472	3.482
RandomForestClassifier	94.579	3.191	49.536	5.996	72.136	0.249
AdaBoostClassifier	94.586	0.249	56.818	0.496	75.706	0.663
GradientBoostingClassifier	94.603	2.284	54.524	4.384	74.609	0.181
GaussianNB	92.098	20.445	23.569	21.896	59.523	0.233

This shows the biased data we figured out later



For the upcoming 3 trials, this problem is solved as shown below



OverSampling

Classifier	Accuracy	Precision	Recall	F1Score	ROC AUC	Log Loss
LogisticRegression	66.888	64.062	67.958	65.953	66.945	0.607
KNeighborsClassifier	94.251	100.000	89.699	94.570	94.849	1.058
DecisionTreeClassifier	96.974	100.000	94.300	97.066	97.150	1.045
RandomForestClassifier	99.605	100.000	99.218	99.607	99.609	0.048
AdaBoostClassifier	70.523	70.958	70.396	70.676	70.524	0.687
GradientBoostingClassifier	71.001	70.810	71.131	70.970	71.001	0.562
GaussianNB	69.048	67.053	69.894	68.444	69.080	0.678

UnderSampling

Classifier	Accuracy	Precision	Recall	F1Score	ROC AUC	Log Loss
LogisticRegression	68.175	66.462	68.258	67.348	68.179	0.594
KNeighborsClassifier	56.125	55.652	55.568	55.610	56.119	4.500
DecisionTreeClassifier	61.159	61.692	60.461	61.070	61.165	13.415
RandomForestClassifier	70.476	69.830	70.219	70.024	70.472	0.571
AdaBoostClassifier	70.770	72.098	69.736	70.897	70.792	0.687
GradientBoostingClassifier	70.785	70.939	70.210	70.572	70.784	0.560
GaussianNB	69.605	68.166	69.640	68.895	69.606	0.666

SMOTE

Classifier	Accuracy	Precision	Recall	F1Score	ROC AUC	Log Loss
LogisticRegression	69.280	68.348	69.699	69.017	69.288	0.602
KNeighborsClassifier	88.965	99.470	82.218	90.025	90.772	1.868
DecisionTreeClassifier	93.839	94.500	93.280	93.886	93.847	2.128
RandomForestClassifier	96.677	94.220	99.099	96.598	96.788	0.124
AdaBoostClassifier	83.675	80.322	86.131	83.125	83.827	0.680
GradientBoostingClassifier	88.732	85.957	91.035	88.423	88.852	0.301
GaussianNB	68.681	71.766	67.644	69.644	68.750	0.668

From the above statistics, we concluded that the best classifier is Random Forest with SMOTE (since this is the most reliable method which uses data augmentation. Random forest→ is a technique used in modeling predictions and behavior analysis and is built on decision trees. It contains many decision trees that represent a distinct instance of the classification of data input into the random forest. The random forest technique takes consideration of the instances individually, taking the one with the majority of votes as the selected prediction

Classifier	Accuracy	Precision	Recall	F1 Score	ROC AUC	Log Loss
RandomForest	96.677	94.220	99.099	96.598	96.788	0.124

Unsuccessful Trials that were not included

- We first tried the original features, but the performance was not the best at all (accuracy in range of 60s).
- Then for every upcoming trial we will mention, the performance metrics were not the best as well:
    - 8 features
        - user_product_total_orders
        - product_total_orders
        - product_avg_add_to_cart_order
        - user_total_orders
        - user_avg_cartsize
        - user_total_products
        - user_avg_days_since_prior_order
        - User_product_avg_add_to_cart_order
    - 9 features
        - uxp_total_bought
        - uxp_reorder_ratio
        - adding_product_to_cart_time
        - user_product_order_freq
        - total_orders_for_user
        - u_reordered_ratio
        - avg_days_since_prior_order
        - p_total
        - P_reorder_ratio
    - Then we added those 2 features
        - added user_avg_cartsize
        - User_total_products
- And finally we decided to go with the 10 features mentioned before.