Instacart Recommendation System
Final Report

Lauren Dellon

Table of Contents

1. Problem Statement

Instacart is a grocery ordering and delivery app that aims to make it easy to fill your refrigerator and pantry with your personal favorites when you need them. The Instacart app allows you to browse thousands of products from your favorite stores, from groceries and alcohol to home essentials and more. After selecting the products you want, personal shoppers will pick up those products for you as well as deliver them to you. Kaggle has challenged the data science community to use anonymized data on customer orders over time to predict which previously purchased products will be in a user's next order.

The goal of this project is to develop a model capable of predicting whether an order is reordered or not. The obvious client here is Instacart, as it will be the one profiting from the predictive model. The users of Instacart will also benefit from the project, as the app would be more skilled at recommending products to users. Therefore, this is a classification problem, but it is also a recommendation system.

2. Data Wrangling

The data for this project was supplied by Kaggle. The file containing the aisle names and id's is called aisles.csv. Similarly, the file containing the department names and id's is called departments.csv, and the file containing the product names and id's is called products.csv. Another csv file, order_products__prior.csv contains previous order contents for all customers. Yet another csv file, orders.csv, tells which set (prior, train, test) an order belongs. As there were many csv files to work with, many merges were done to better view the data. Fortunately, there were no missing values to deal with.

3. Exploratory Data Analysis

a. Order Analysis

First, an analysis of the orders.csv file was completed. First, I plotted which days of the week most orders were made. This is seen in Figure 1.

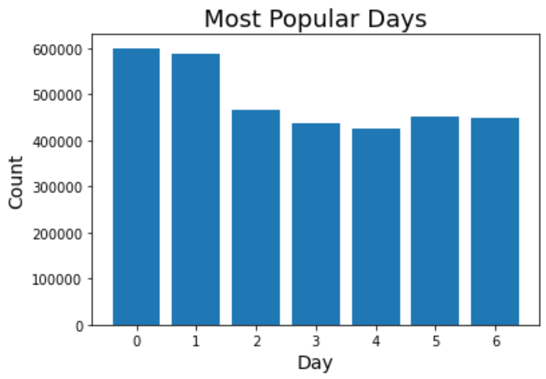


Figure 1: Orders by day of week

As you can see in the figure, days 0 and 1 were the most popular days for orders. It is reasonable to assume that days 0 and 1 correspond to Saturday and Sunday, respectively. Therefore, it is clear that more orders were made on the weekend than on the weekdays. Next, I plotted which hours of the day most orders were made. This can be seen in Figure 2.

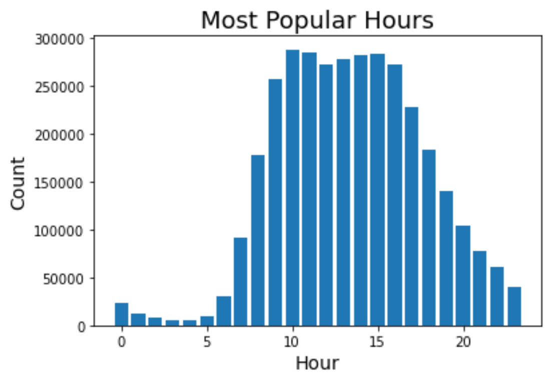


Figure 2: Orders by hour of day

Most orders were made between the hours of 10am and 3pm. Understandably, there were very few orders in the middle of the night (12am to 6am). I also plotted a heat map of the orders for day of week vs. hour of day. This is seen in Figure 3.

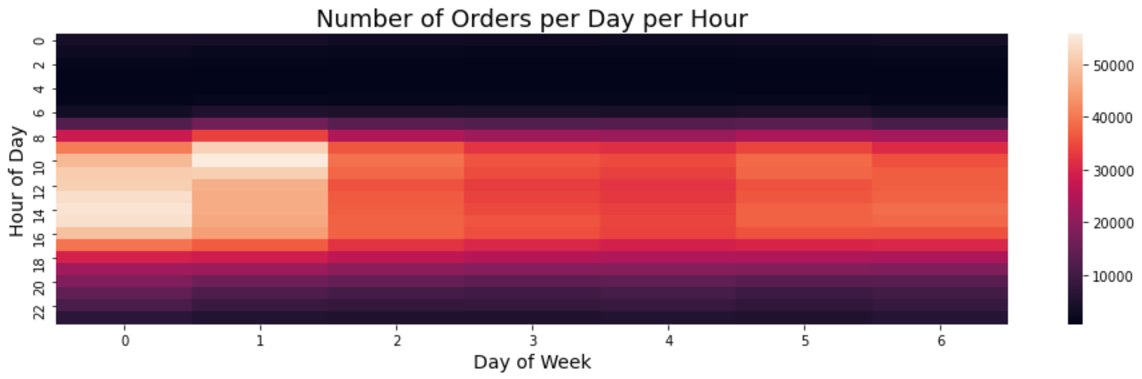


Figure 3: Orders day of week vs. orders hour of day

This heat map shows more clearly that there are very few orders at the beginning of the day (12am to 6am). Then, orders pick up until the end of the night. It is also clear that days 0 and 1 (Saturday and Sunday, respectively) have the most orders. Finally, I plotted the days since prior order, which is seen in Figure 4. The majority of the days since prior order is 30. This is most likely due to any number of days over 30 being assigned to the 30 category. There also appears to be many customers who order weekly, as the bars for 7, 14, and 21 days are local maxima.

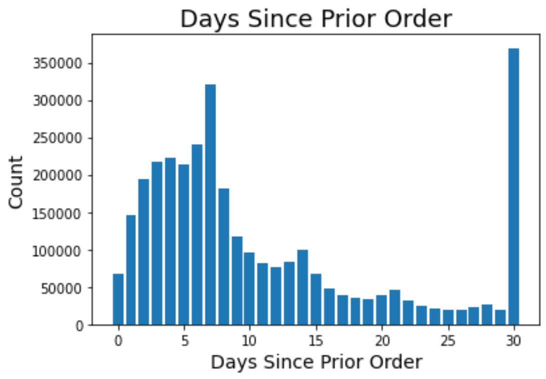


Figure 4: Days since prior order

b. Product and Order Analysis

Next, an analysis of the products and orders combined was conducted. I was able to do a simple inventory check to determine which aisles have the most products and which departments have the most products. These are seen in Figure 5 and 6, respectively.

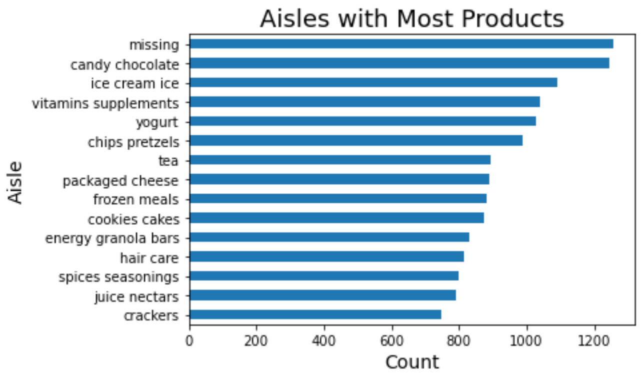


Figure 5: Aisles with most products

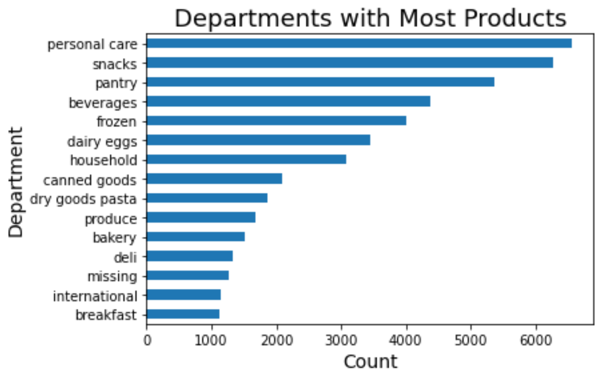


Figure 6: Departments with most products

I also determined the top products ordered, top aisles ordered from, and top departments ordered from. These are seen in Figures 7, 8, and 9, respectively.

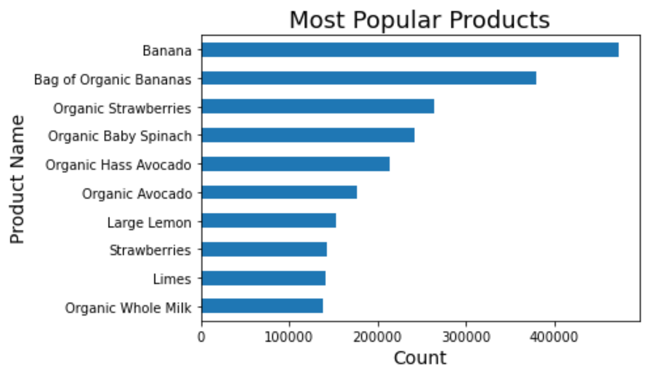


Figure 7: Top products ordered

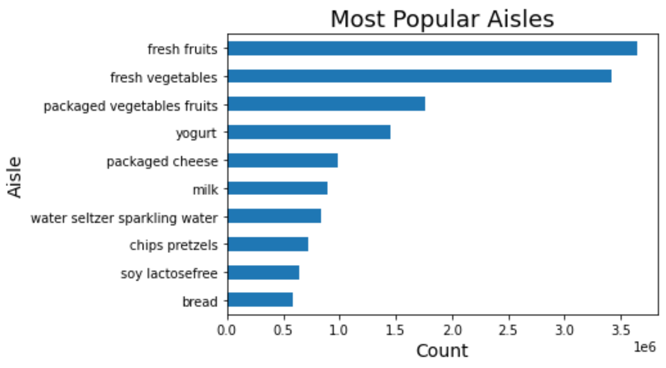


Figure 8: Top aisles ordered from

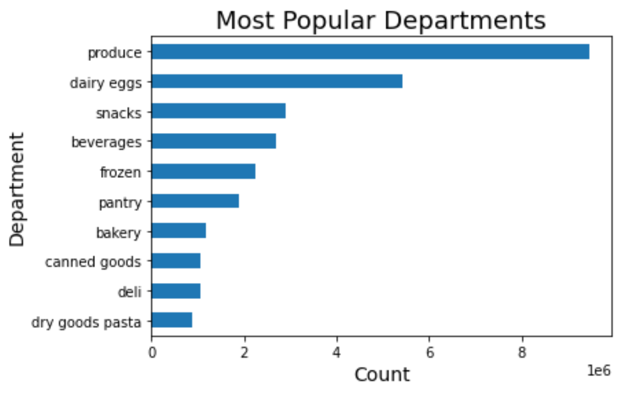


Figure 9: Top departments ordered from

You can see from Figure 7 that most of the top products ordered were actually organic products. The most popular product, however, was regular bananas. The most popular aisles were fresh fruits and vegetables, which makes sense given that most of the top products were fruits and vegetables. Similarly, the most popular department was produce. Next, I analyzed the most popular reordered products, which can be seen in Figure 10.

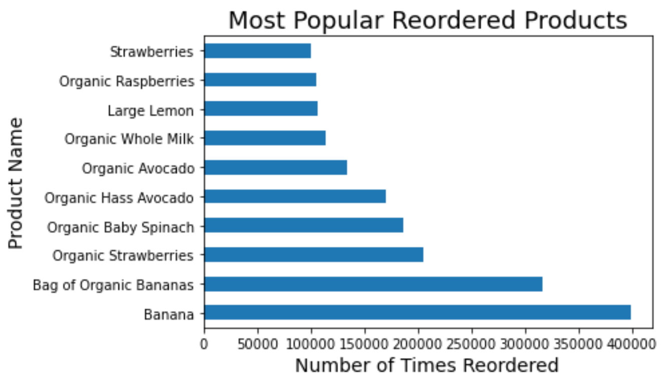


Figure 10: Most popular reordered products

I also calculated that about 59% of the products in the orders were reordered products, while about 12% of the orders had no reordered items. Figure 11 displays a histogram of the number of products in a given order. It has a normal distribution with a long right tail. The count was highest for 5-6 items in an order.

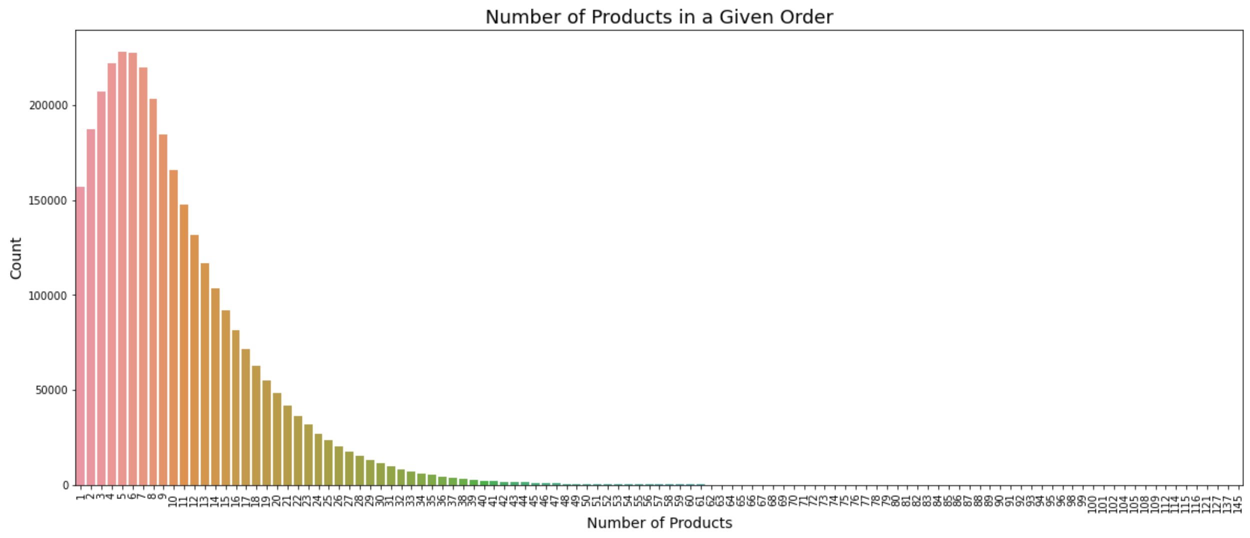


Figure 11: Number of products in a given order

Figure 12 displays a histogram of the number of reordered products in a given order. This is similar to the right side of a normal distribution with a long right tail.

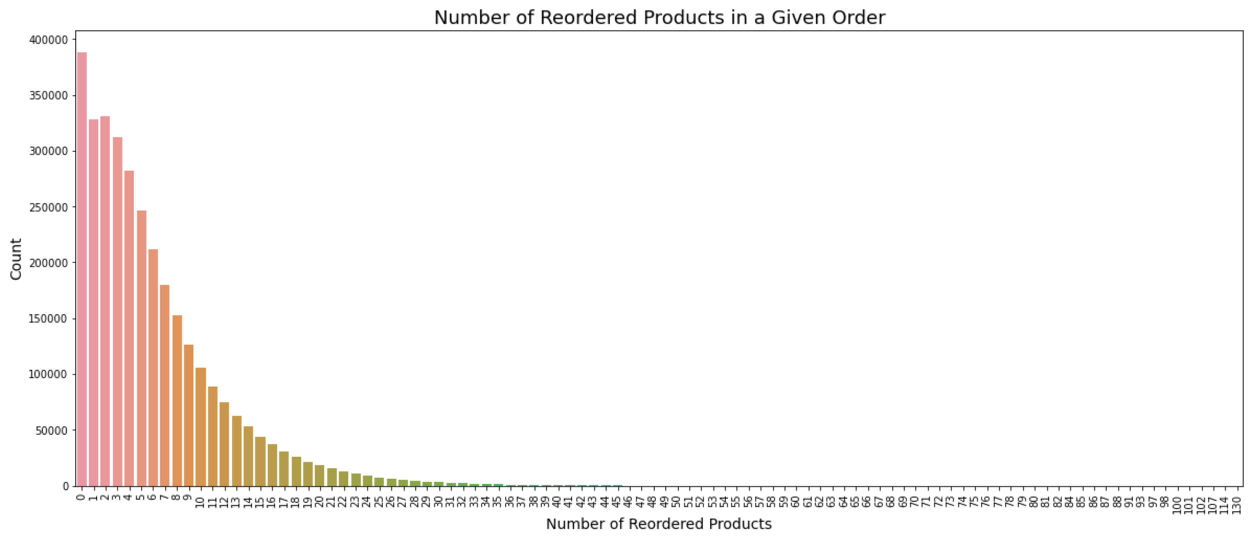


Figure 12: Number of reordered products in a given order

Finally, I completed a user analysis. The average number of prior orders by a customer is 15.59 orders. The minimum is 3 orders, and the maximum is 99 orders. The average number of days between orders for all customers is 15.47, with a minimum of 0 and a maximum of 30. The average basket size for all customers is 9.95 items, with a minimum of 1 and a maximum of 70.25**.**

4. Feature Engineering

a. User Features

The features calculated for each user include the reordered ratio of each user, the total number of orders of each user, the total items purchased by each user, the average days since prior order for each user, and the average basket size of each user.

b. Product Features

The features calculated for each product include the reordered ratio of each product and the number of purchases of each product.

c. Order Features

The features for each order include the order number, aisle ID, department ID, day of week, hour of day, and days since prior order.

5. Preprocessing

First, the categorical variables (aisle ID and department ID) were mean encoded. This means that the original values were replaced by an encoded value representing a probability of the target variable (reordered), conditional on each value of the feature. Next, the data was split into a training set and a testing set. After splitting, I calculated the value counts for y_train and y_test and determined that I had an imbalanced dataset, with a majority of the target variable being 0 (not 1). To fix this issue, I undersampled the majority class. This led to a total number of 0's being 580,297 and a total number of 1's being 580,297 as well. Undersampling was a good choice in this case because it fixed the imbalance and also led to a smaller dataset that was more feasible for modeling. Finally, I scaled the data using StandardScaler.

6. Modeling and Parameters

The goal of this work was to construct a model capable of predicting whether or not a product would be reordered. Therefore, this is a classification problem. I used the following classification models for prediction:

- Logistic Regression
- Gradient Boosting
- Random Forest

Logistic Regression: A simple loop was used to determine the best value for the regularization parameter, C. The best value for C was found to be 1.

Gradient Boosting: The parameters used for the Gradient Boosting model were as follows: loss='deviance', learning_rate=0.1, n_estimators=100, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3.

Random Forest: The parameters used for the Random Forest model were as follows: n_estimators=100, criterion='entropy', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0.

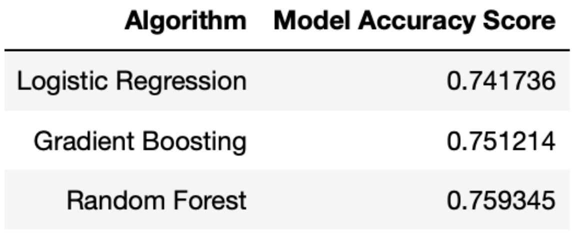
7. Model Selection Metrics

The optimal model was selected based on both primary and secondary performance metrics. The primary performance metric was the model's f1 score, which is calculated from the precision and recall. The secondary performance metrics include the following: accuracy, precision, recall, ROC-AUC train and test scores from 5-fold cross validation, time taken to train the model and make predictions.

8. Modeling Results and Feature Importance

The model accuracy score is seen in Table 1 for each of the 3 models and is plotted in Figure 13.

Table 1: Comparison of Model Accuracy Scores



Algorithm	Model Accuracy Score
Logistic Regression	0.741736
Gradient Boosting	0.751214
Random Forest	0.759345

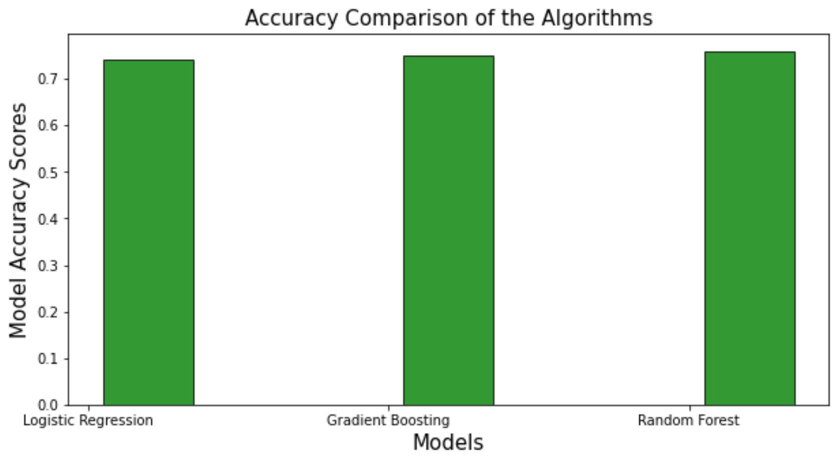
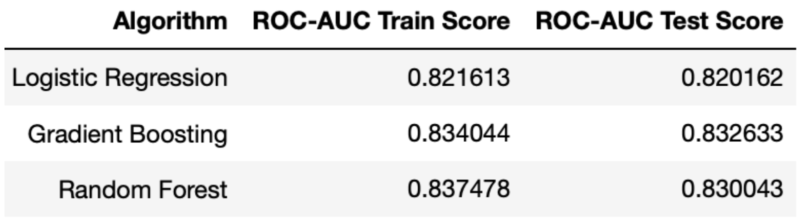


Figure 13: Comparison of Model Accuracy Scores

The model ROC-AUC train and test scores are seen in Table 2 for each of the 3 models and are plotted in Figure 14.

Table 2: Comparison of ROC-AUC Train and Test Scores



Algorithm	ROC-AUC Train Score	ROC-AUC Test Score
Logistic Regression	0.821613	0.820162
Gradient Boosting	0.834044	0.832633
Random Forest	0.837478	0.830043

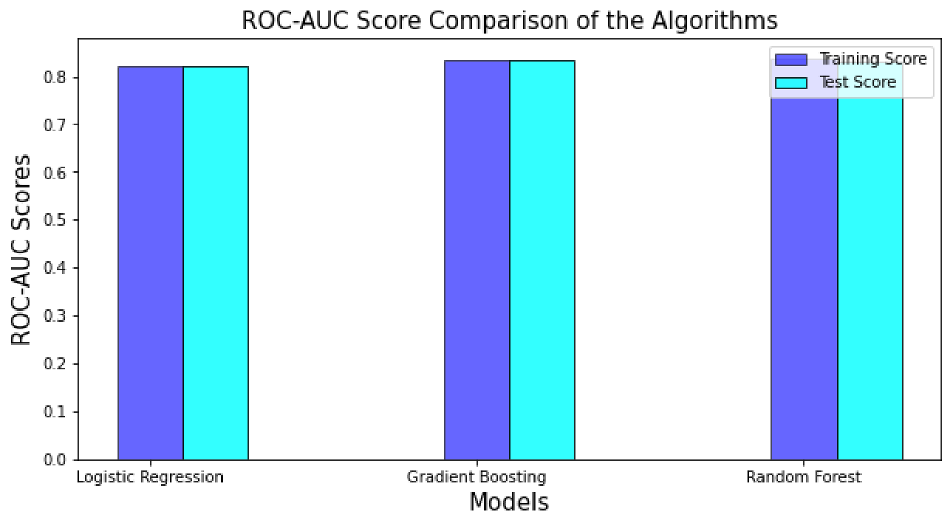


Figure 14: Comparison of ROC-AUC Train and Test Scores

The model precision score is seen in Table 3 for each of the 3 models, the model recall score is seen in Table 4 for each of the 3 models, and the precision and recall scores are plotted in Figure 15.

Table 3: Comparison of Model Precision Scores

Algorithm	Model Precision Score
Logistic Regression	0.694425
Gradient Boosting	0.707842
Random Forest	0.725366

Table 4: Comparison of Model Recall Scores

Algorithm	Model Recall Score
Logistic Regression	0.863403
Gradient Boosting	0.855551
Random Forest	0.834728

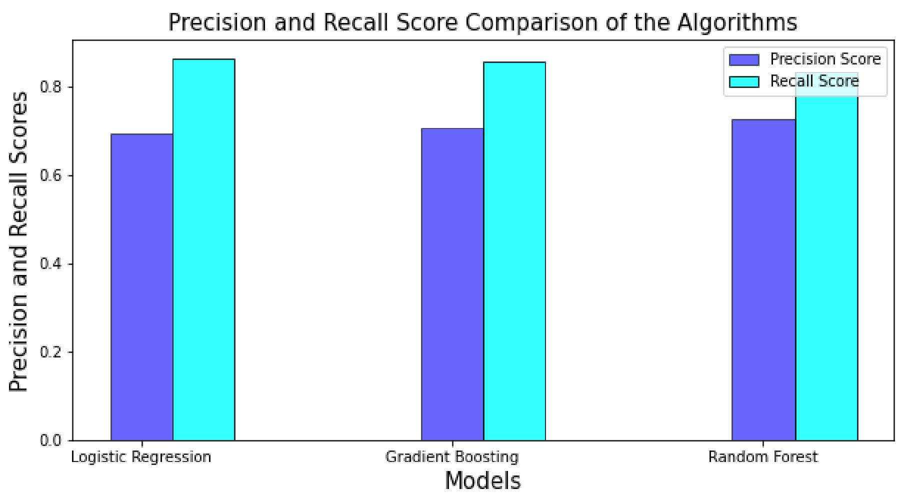


Figure 15: Comparison of Precision and Recall Scores

The model F1 score is seen in Table 5 for each of the 3 models and are plotted in Figure 16. This is the primary performance metric.

Table 5: Comparison of Model F1 Scores

Algorithm	Model F1 Score
Logistic Regression	0.769750
Gradient Boosting	0.774719
Random Forest	0.776214

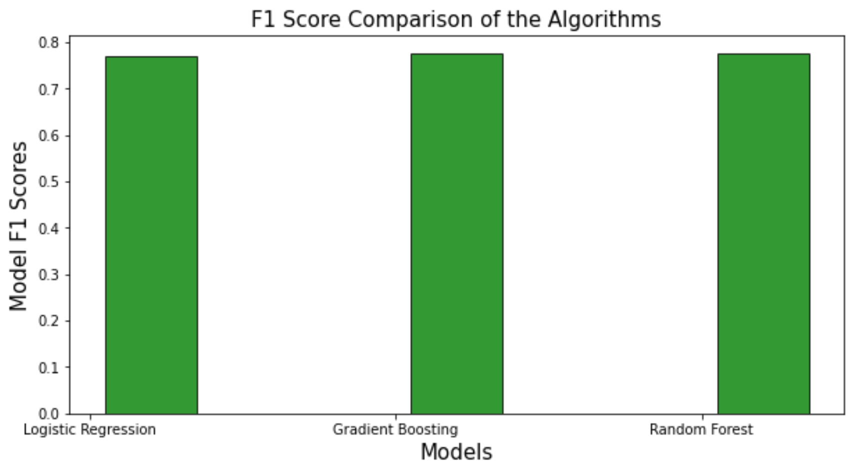


Figure 16: Comparison of F1 Scores

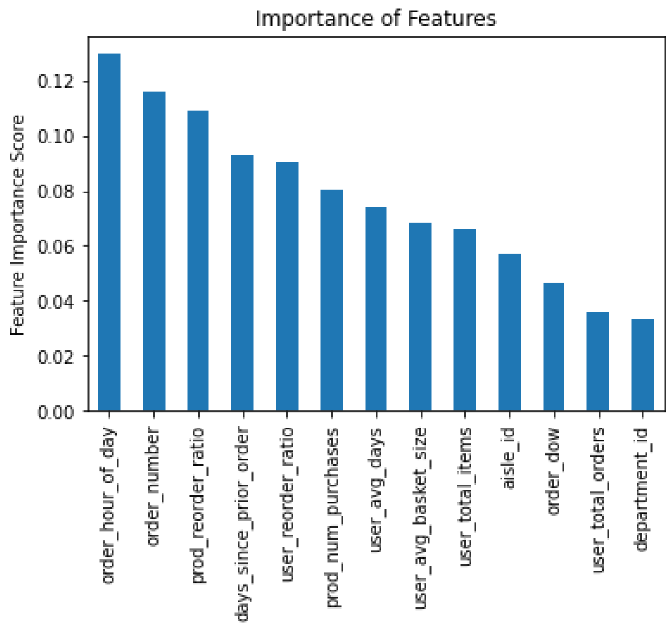
The times taken to train and test the model are seen in Table 6.

Table 6: Comparison of Model Train and Predict Times

Algorithm	Train Time (s)	Predict Time (s)
Logistic Regression	1.542	0.007
Gradient Boosting	154.664	0.560
Random Forest	216.206	10.848

The three models have comparable accuracy, ROC-AUC, precision, and recall scores. However, the Random Forest model is slightly better in terms of accuracy, precision, and f1 score. Additionally, the Random Forest model allows me to easily extract the feature importance. On the other hand, the Random Forest model took the longest amount of time to train and make predictions, but this is a tradeoff I'm willing to accept. Therefore, I conclude that the Random Forest model is the best model.

Using the Random Forest model, I plotted the feature importance for each feature, seen in Figure 17. The three most important features are the hour of day of the order, the order number, and the product reorder ratio. The two least important features are the user's total number of orders and the department ID.



9. Conclusion and Future Work

The goal of this project was to develop a model capable of predicting whether an order is reordered or not. After completing a lengthy exploratory data analysis, features were engineered using the user information and the product information. After downsampling the majority, three different models were created including logistic regression, gradient boosting, and random forest. While the Random Forest model took the longest time to train and test, it had the best F1 score at 0.776. Therefore, I concluded that the Random Forest model is the best model for this project. Finally, I looked at feature importance from the Random Forest model and found three most important features to be the hour of day of the order, the order number and the product reorder ratio.

There are several future work items I would complete. While I created features from the users and the products, there are still more features that could be engineered from the users and products combined. For example, I could have engineered a feature for how many times a user has bought a product. Additionally, there are many models that I did not construct that may have been useful. For example, I could have used KNN, Naïve Bayes, or a more advanced boosting model.