PURDUE
UNIVERSITY
FORT WAYNE

DEPARTMENT OF MATHEMATICAL SCIENCES
MATH 59800 PROJECT

Project Report - Fall 2022

PREDICTING ACTUAL DELIVERY DURATION FOR DOORDASH

Dariya Mamyrbek,
Saviour Avorgah

Contents

1 Purpose and Background of Study

Today, with the rise of technology, there are many different apps/companies that rely on delivery time prediction, especially companies such as Uber Eats, DoorDash, Grubhub, which deliver food on demand. When a consumer places an order, they're being shown an estimated delivery time. And it's very important that delivery time prediction is accurate because it directly affects customer experience. Since usually food delivery services must deliver within 30-40 minutes, 5-10 minutes in prediction can play a big role and affect customer satisfaction, and hence company's revenue. The goal of the project is to build a model which predicts an estimated time taken for a delivery, which is a total delivery duration in seconds from the time it was created till the delivery time. The model will use data about food delivery as input, which includes time features, store features, order features, and market features.

The dataset was obtained from: `https://platform.stratascratch.com/data-projects/delivery-duration-prediction`. The collection of data is such that The .csv file contains a subset of deliveries received at DoorDash in early 2015 in a subset of the cities. Each row in this file corresponds to one unique delivery. There are a total of 197,429 rows, and 16 columns in the dataset and a detailed description of each variable is presented in Table 3 in the 8, it is important to know that missing data in the dataset was recorded as "N/A".

2 Exploratory Analysis of the Data

2.1 Data Wrangling

In this section, we discuss into details any recoding that was performed on any variable, we also discuss the logic and ideas behind creating new variables and how the problem of outliers was resolved. In total there were 24063 rows with missing data out of 197428 rows in the dataset, so we decided to remove those rows for better model accuracy, since 24063 corresponds to 12 percent of the total number of rows.

Store Primary Category: There were initially seventy-three categories of primary stores. However, we only kept the 30 most frequent store categories. This is because the rest had values below 1000 and so we grouped all of it into another category called "other". We then created dummy variables out of the variable "store primary category", thus; essentially, we made the "store primary category" variable a factored one.

Market ID: For this variable, there were six different market IDs. We created dummy variables from this variable, which means we converted it to a factor.

Order Protocol: For this variable, there were seven different classes. We created dummy variables from this variable, which means we converted it to a factor.

Estimated Order Place Duration: For this variable, the data was distributed in a way that most of them, around 99.9 percent of the data, had values 251 and 461 seconds. So we decided to group them into 9 groups. The range of the values initially was from 0 to 2715 seconds and we also removed an outlier of 2715. In the end we had 9 groups, starting from 0 up to 1800 with 200 seconds interval. We encoded them ordinally, meaning group [0, 200] being integer 1, second group [201, 400] coded to 2, and so on up to 9. Since most of the data falls into groups 2 and 3, we can use different oversampling techniques on other groups. This grouped variable was however removed later because it did not show to be an important variable during our analysis.

Actual Total Delivery Duration: We created a new variable which is a response variable we're predicting - "called actual total delivery duration" as a difference between "actual delivery time" and "created at" variables first converting them from string to the date time, then obtaining value in seconds as a difference between these two variables.

Store ID: This variable was deleted.

Estimated Non Prep Duration: We also created a new variable to estimate time not associated with meal preparation duration which is sum of "estimated store to consumer driving

duration" and "estimated order place duration."

Busy Dashers Ratio: We also created a new variable which shows a proportion of busy dashers by dividing "total busy dashers" by "total onshift dashers."

Subtotal: For this variable, there were some negative values, and because the subtotal is in cents, we deleted the rows where the subtotal was negative.

Max item price: For this variable, there were some negative values, and because the maximum price of an item is in cents and cannot be negative, we deleted the rows where the maximum item price was negative.

Min item price: For this variable, there were some negative values, and because the minimum price of an item is in cents and cannot be negative, we deleted the rows where the minimum item price was negative.

Total Outstanding Orders: For this variable, there were some negative values, and because the number of outstanding orders cannot be negative, we deleted the rows where the total outstanding orders was negative.

Delivary Durations: We removed rows with delivery duration being greater than 15000 seconds and less than 300 seconds, because these values most certainly fall under the outliers category, where 15000 seconds is 4.17 hours and we know that usually it doesn't take more than that to deliver a meal, and 300 seconds is 5 minutes, which taking into account time for Doordash to place an order, restaurant to receive it, a dasher to pick it up, is not possible to take less than 5 minutes.

Total Items: For the total items number the range was from 1 to 411, where 411 is definitely an outlier, and the next maximum values was 66, the distribution of values after 20 constituted of just 0.18 percent of the total number of values, and having more than 20 items in order is quite unusual, so we deleted those rows.

Numumber of Distinct Items: For the number of distinct items in one order, we removed rows with values greater than 13 items, because out of the range from 1 to 17, those values corresponded to just 0.03 percent.

Total Onshift Dashers: Removed rows with negative values, because it's not possible to have a negative number for that variable.

Estimated Order Place Duration: Deleted a row corresponding to 2715 seconds which is 42.25 mins, and it's quite unusual for the order place duration, taking into account 75 percent of the data corresponds to 446 seconds.

Now, putting together our dataset, after deleting the variable Store id, and concating the newly created columns from Store primary category, Market id, and Order protocol, we now have a dataframe with 167559 rows, and 65 columns. Creating dummy variables from Store primary category created 30 columns, Market id created 6 columns, Order protocol created 7 columns, and with the new variables that were created; Actual total delivery duration, Estimated non prep duration, and Busy dashers ratio, we now have 103 columns for now.

2.2 Summary Statistics of Numerical Variables

In the table below, we summarize the statistics of all the numerical variables used in this work.

Variable	Count	Mean	SD	Min	25%	50%	70%	Max
Subtotal	167559	2682.15	1769.76	109	1425	2230	3400	19250
Minimum Item Price	167559	697.38	517.90	1	300	599	950	14700
Maximum Item Price	167559	1167.21	555.62	60	800	1095	1399	14700
Total Outstanding Orders	167559	59.79	52.61	1	19	43	87	285
Est. Store to Consumer Drive Dur.	167559	546.91	218.54	0	385	545	704	2088
Total Items	167559	3.1	2	1	2	3	4	13
Number of Distinct Items	167559	2.6	1.5	1	1	2	3	13
Actual Total Delivery Dur.	167559	2846.5	1090.8	321	2102	2654	3368	14996
Total Busy Dashers	167559	42.9	31.9	0	16	36	63	154
Total Onshift Dashers	167559	46.1	34.3	0	18	38	67	171
Est. Non-Prep Duration	167559	854.4	234.2	153	683	851	1017	3222

Table 1: Summary Statistics of the Predictor Variables.

2.3 Distributions of Variables

In this section, we look at the histogram/bar plots of all predictor variables, we explore the skewness and the normality (whether or not the observations from each predictor variable is bell-shaped) of each one. All figures referenced in this section are in Appendix C here 8.1 for the reader's use. We realized the predictor that tend to follow a normal distribution is; the Estimated Store to Consumer Driving Duration (see figure 9), and it was observed that most of the variables were skewed to the right. We discuss the distributions for each variable in detail below:

Subtotal: For this variable, the distribution is right-skewed, the mean is 2682.14 which is higher than the median of 2230 cents, also the minimum amount here is approximately USD 1.09, and the maximum amount is approximately USD 192.5. This distribution makes sense realistically, and appears to be unimodal. See Figure 15.

Minimum Item Price: For this variable, the distribution is right-skewed, the mean is 697.38

which is higher than the median of 599 cents, also the minimum amount here is approximately USD 6.97, and the maximum amount is approximately USD 147. This distribution makes sense realistically, and appears to be unimodal. See Figure 12.

Maximum Item Price: For this variable, the distribution is right-skewed, the mean is 697.38 which is higher than the median of 599 cents, also the minimum amount here is approximately USD 11.67, and the maximum amount is approximately USD 147. This distribution makes sense realistically, and appears to be unimodal. See Figure 11.

Total Outstanding Orders: For this variable, the distribution is right-skewed, the mean is 60 which is higher than the median of 43 total number of outstanding orders. This variable gives us an idea of how many items (orders) are in queue before a customer places an order, the numbers from the summary statistics makes sense realistically, and the distribution appears to be unimodal. See Figure 19.

Estimated Store to Consumer Driving Duration: For this variable, the distribution appears to be bell-shaped, unimodal and symmetric. As can be seen from the summary statistics, the mean is approximately equal to the median. This in context means that the mean duration for a dasher to move from a store to a customer's location is approximately 9 minutes, whereas the maximum time it takes to get to a customer's location is about 35 minutes and this makes sense realistically. See Figure 9.

Market ID: For the market id variable, there are 6 unique values in range from 1 to 6, each integer representing a city/region in which DoorDash operates. It's possible to conclude here that we have two modes: market id 2 and 4. See Figure 10.

Estimated Order Place Duration: For the estimated order place duration, we grouped them into 9 groups with the interval of 200 seconds, because of the nature of the original dataset as was discussed in section 1. And there are 2 prevailing groups here: group number 2 and 3, which represent time intervals [201,400] and [401,600] seconds. See Figure 8.

Total Items: Total items distribution is right-skewed, the mean at 3.1 and median at 3, so most of the data has values closer to the left side of the data, and this makes sense, because usually people do not order more than, say, 4-5 items per order. The range of the variable is from 1 to 13. See Figure 17.

Number of Distinct Items: Similarly to the total items distribution, the nature of the number of distinct items is similar. It is right-skewed, the mean is at 2.6 and median at 2, so most of the data has values closer to the left side of the data, and this makes sense, because usually people

do not order a large number of distinct items per order. The range of the variable is from 1 to 13 as well. See Figure 13.

Actual Total Delivery Duration: The actual total delivery duration is the response variable, after cleaning data and dealing with outliers, the range for this variable is from 321 to 14996, which is approximately from 5 to 250 minutes. The distribution is right-skewed, the mean is 2846.5 which is higher than the median of 2654 seconds, most of the data lies in the left-hand side of the range, and it makes sense because in most of the times it takes no more than hour for Doordash to deliver. See Figure 20.

Total Busy Dashers: Total busy dashers variable also has a right-skewed distribution, the mean is at 42.9 and median - 36. 75 percent of the data corresponds to 63 dashers. The range is from 0 to 154. See Figure 16.

Total Onshift Dashers: Similarly to the total busy dashers variable, the total onshift dashers variable also has a right-skewed distribution, the mean is at 46.1 and median - 38. 75 percent of the data corresponds to 67 dashers. The range is from 0 to 171. See Figure 18.

Estimated Non Prep Duration: This variable was created to show the time needed for an order to be placed and delivered excluding the time needed for the preparation itself. The distribution of the estimated non-preparation duration is also slightly skewed to the right, with the mean 854.4 seconds and median 851 seconds, the range is from 153 to 3222 seconds, or 2.5 minutes to 53.7 minutes. The 75^{th} percentile corresponds to 16.95 minutes. See Figure 7.

2.4 Correlation and Multicollinearity

It was realized the predictor variables, Subtotal, and Number of Distinct Items are highly positively correlated whiles Total Items is highly negatively correlated to the Price of the minimum item in the order. Additionally, the relationship between Total Items and Subtotal is very strong. It was also observed that, there exists a very weak correlation between the Estimated Store to Consumer Driving Duration and the Price of the minimum item in the order and same with the Estimated Non Prep Duration and the Total Number of Outstanding Orders. For the benefit of the reader, the correlation matrix for the all predictor variables and the response variable is provided here: `https://colab.research.google.com/drive/171yhLQcGUioJXbsRs4lY5E9B2Mgi-Xby#scrollTo=yfSDSnsfraQ7&line=1&uniqifier=1`. Specifically, a plot of the correlations between all the numerical variables used in this data is shown in Appendix E here (8.3).

Also, we discuss the relationship between the response variable, "Actual total delivery duration" and all numerical predictor variables using scatter plots. From the scatter plots listed in Appendix D here 8.2, we can see that in all or most cases, there exists a weak correlation between the numerical variables used as predictors and the response variable.

3 Discussion of Methods Used

In this section, we present a brief description of all the methods that was used in this work, and establish why a method was used. In summary, we will discuss each model we used, why we used them - why are they appropriate for the problem we're solving, some assumptions of each model and the Pros and Cons of each method.

Linear Regression: Linear regression is a supervised machine learning method that finds a linear equation that best describes the correlation of the explanatory variables with the dependent variable. This is achieved by fitting a line to the data using least squares. The line tries to minimize the sum of the squares of the residuals. The residual is the distance between the line and the actual value of the explanatory variable. Linear regression assumes that the relationship between your input and output is linear and will over-fit your data when you have highly correlated input variables. We implemented the use of linear regression in our work because it is simple to implement and easier to interpret the output coefficients.

Lasso Regression: Lasso regression is like linear regression, but it uses a technique "shrinkage" where the coefficients of determination are shrunk towards zero. While linear regression gives you regression coefficients as observed in the dataset, the lasso regression allows you to shrink or regularize these coefficients to avoid overfitting and make them work better on different datasets. Lasso regression penalizes less important features of your dataset and makes their respective coefficients zero, thereby eliminating them. Thus it provides you with the benefit of feature selection and simple model creation. While performing lasso regression, we add a penalizing factor to the least-squares. We implemented lasso regression because our dataset has high dimensionality and high correlation between the dummy variables. Also, lasso regression helps with feature selection.

Ridge Regression: Ridge regression works almost similar to lasso regression. While lasso regression performs L1 regularization, the ridge regression method performs L2 regularization. In ridge regression, the first step is to standardize the variables (both dependent and independent) by subtracting their means and dividing by their standard deviations and the assumptions of ridge regression are the same as that of linear regression. We implemented ridge regression because it helps in feature selection and also because our dataset has high dimensionality and high correlation between the dummy variables.

Stochastic gradient descent (SGD) Regression: Gradient descent is an iterative algorithm,

that starts from a random point on a function and travels down its slope in steps until it reaches the lowest point of that function. This algorithm is useful in cases where the optimal points cannot be found by equating the slope of the function to 0. In SGD, we start with a random point and find a way to update this point with each iteration such that we descend the slope. We find the slope of the objective function with respect to each parameter/feature. In other words, compute the gradient of the function. We then update the gradient function by plugging in the parameter values and calculate the step sizes for each feature as: step size = gradient × "learning rate". Calculate the new parameters as: new params = old params - step size and the we repeat the steps until gradient is almost 0. We know that the goal of regression is to minimize the sum of squared residuals and we also know that a function reaches its minimum value when the slope is equal to 0, hence implementing SGD regression is one way to reach that goal. However, one downside of SGD regression is the amount of computation we make for each iteration of the algorithm and hence gradient descent is slow on huge data. We implemented the SGD regression because it is efficient and well suited for regression problems with a large number of training samples.

K-Nearest Neighbors (KNN) for Regression: The KNN algorithm uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set. In implementing the KNN, first, the distance between the new point and each training point is calculated, the closest k data points are selected (based on the distance) and the average of these data points is the final prediction for the new point. The first step is to calculate the distance between the new point and each training point. There are various methods for calculating this distance, of which the most commonly known methods are – Euclidian, and Manhattan. The second step is to select the k value. This determines the number of neighbors we look at when we assign a value to any new observation. The KNN is easy to implement because it has no assumptions and allows a variety of distance criteria to be choose from. On the other hand, the KNN algorithm becomes very slow when dataset grows, and it sometimes a challenge to select the optimal the optimal number of neighbors. We implemented the KNN regression method because it is easy to implement and it handles non-linearities well, and we observed that the relationships between the predictor variables and the response variable in our dataset is non-linear.

Linear Support Vector Regression (SVR): SVR is a regression algorithm that supports both linear and non-linear regressions. While in simple regression, the idea is to minimize the

error rate, in SVR the idea is to fit the error inside a certain threshold which means, the goal of SVR is to approximate the best value within a given margin called ϵ- tube. A few terminologies exists in the use SVR.

- Hyperplane: It is a separation line between two data classes in a higher dimension than the actual dimension. In SVR it is defined as the line that helps in predicting the target value.
- Kernel: In SVR the regression is performed at a higher dimension. To do that we need a function that should map the data points into its higher dimension. This function is termed as the kernel. Type of kernel used in SVR is Sigmoidal Kernel, Polynomial Kernel, Gaussian Kernel, etc,.
- Boundary Lines: These are the two lines that are drawn around the hyperplane at a distance of ϵ. It is used to create a margin between the data points.
- Support Vector: It is the vector that is used to define the hyperplane or we can say that these are the extreme data points in the dataset which helps in defining the hyperplane. These data points lie close to the boundary. Support Vectors helps in determining the closest match between the data points and the function which is used to represent them.

While the linear SVR is robust to outliers and the decision model can be easily updated, they are not suitable for large datasets. We implemented the linear SVR because it has a faster implementation.

Decision Tree Regression: Decision Tree is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification problems. A tree is built by splitting the source set, constituting the root node of the tree, into subsets—which constitute the successor children. The splitting is based on a set of splitting rules based on classification features. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same values of the target variable, or when splitting no longer adds value to the predictions. [wiki]. Some advantages of decision trees are: they are simple to understand and interpret; require little data preparation; it's possible to validate a model using statistical tests; the cost of using the tree is logarithmic. And the disadvantages include: they are prone to overfitting as tree learners can create over-complex trees that do not generalize the data well; they can be unstable because small variations in the data might result in a completely different tree being generated; they are not good at extrapolation because predictions are piecewise constant approximations.

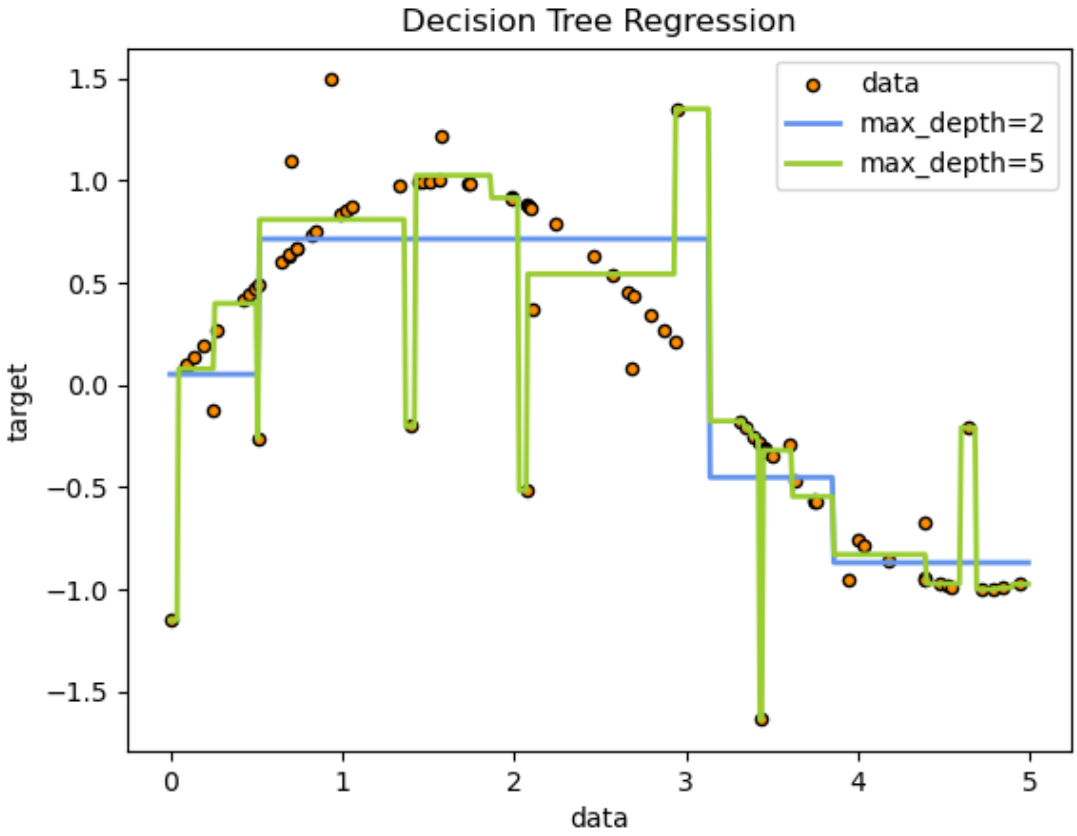

Figure 1: Decision Tree Regressor.

Random Forest Regressor: Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size max features. Individual decision trees usually have high variance and tend to overfit. So having a random subset of a random number of features decreases the variance of the forest estimator by taking an average of the predictions, however, there might be an increase in bias. Random Forests generally provide high accuracy and balance the bias-variance trade-off well, however, they can be computationally intensive for large datasets and are not easily interpretable.

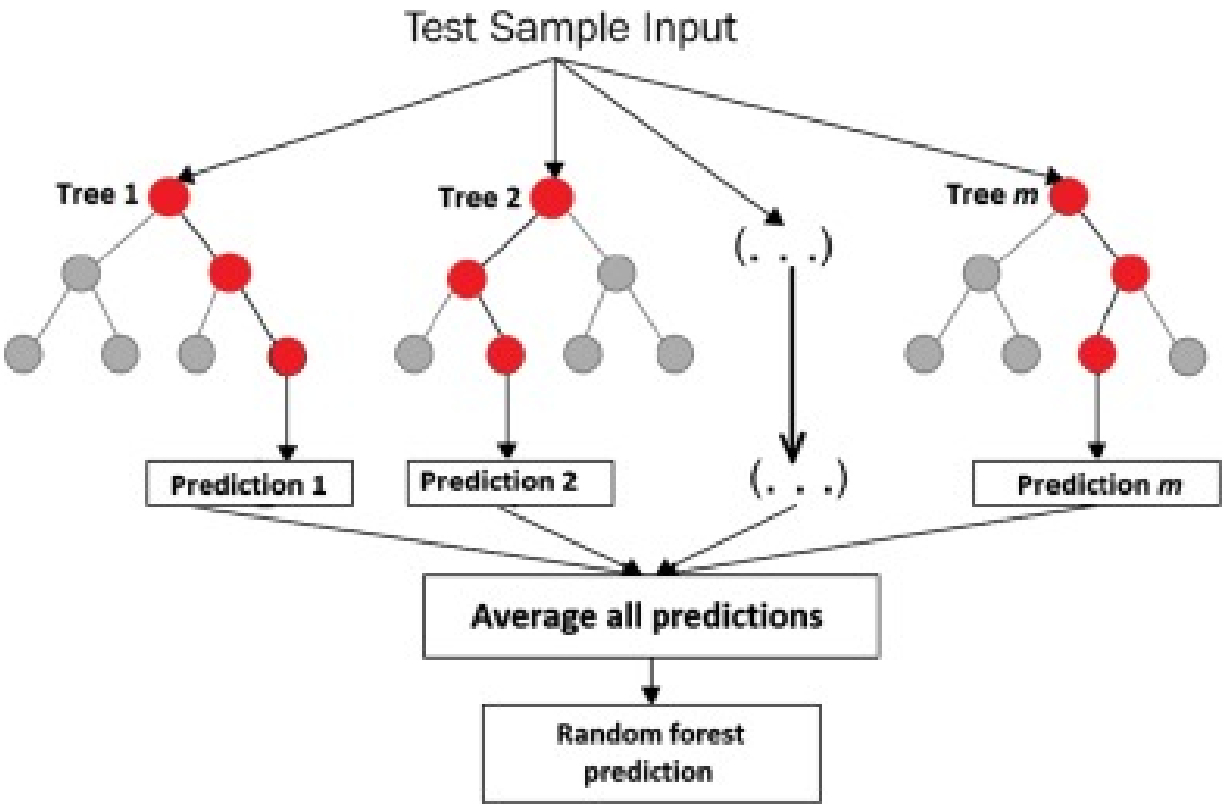

Figure 2: Decision Tree Regressor.

Bagging Regression: is the ensemble learning method that is commonly used to reduce variance within a noisy dataset. In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once. After several data samples are generated, these weak models are then trained independently, and the average of those predictions yield a more accurate estimate. (https://www.ibm.com/cloud/learn/bagging) So, bagging regression introduces randomization into its construction procedure and then makes an ensemble out of it, hence reducing variance of a base estimator. As they provide a way to reduce overfitting, bagging methods work best with strong and complex models. One disadvantage of bagging is that it is difficult to interpret the model. **Adaptive Boosting (AdaBoost):** Boosting is a class of ensemble machine learning algorithms that combine the predictions of many weak learners. AdaBoost is the first designed boosting algorithm with a particular loss function that aims to combine multiple weak classifiers to create one strong classifier. The method is based on the paper by Yoav Freund and Robert Schapire "Experiments with a New Boosting Algorithm". The AdaBoost algorithm uses very short decision trees as weak learners that are added sequentially to the ensemble. Each subsequent model attempts to correct the predictions made by the previous model in the sequence. This is achieved by weighting the training dataset

to give more weight to training samples where the previous model had prediction errors. Boosting technique learns progressively, so Adaboost needs a quality training dataset, which is one of the main disadvantages of using it. AdaBoost is also extremely sensitive to Noisy data and outliers. **Gradient Boosting Regression:** Gradient Boost is a machine learning algorithm that works on the ensemble technique called "boosting". Similar to other boosting models, Gradient Boost sequentially combines many weak learners to form one strong learner. Gradient Boost usually uses decision trees as weak learners. Understanding Gradient Boost requires the following steps: In gradient boosting, weak learners are decision trees.

- Step 1: Builds a base tree with one root node. This is the initial guess for all samples.
- Step 2: Build a tree from the previous tree's errors.
- Step 3: Scale the tree by the learning rate (a value between 0 and 1). This learning rate determines the tree's contribution to prediction.
- Step 4: Combine the new tree with all previous trees to predict the outcome and repeat step 2 until either the maximum number of trees is reached or the new tree no longer improves the fit.

The final predictive model is a combination of all trees.

Extreme Gradient Boosting (XGBoost): XGBoost is based on the gradient boosting algorithm for decision trees. Ensemble training is carried out sequentially, unlike, for example, bagging. At each iteration, the deviations of the predictions of the already trained ensemble on the training set are calculated. The next model to be added to the ensemble will predict these deviations. Thus, by adding the predictions of the new tree to the predictions of the trained ensemble, we can reduce the average deviation of the model, which is the target of the optimization problem. New trees are added to the ensemble until the error decreases, or until one of the "early stopping" rules is met. XGBoost is a more regularized form of Gradient Boosting. XGBoost uses advanced regularization (L1 & L2), which improves model generalization capabilities.

Extremeley randomized trees (Extra tree): Extra trees is an ensemble supervised machine learning method that uses decision trees. The Extra Trees algorithm produces as many decision trees as the Random Forest algorithm, but the samples in each tree are random and there is no permutation. This creates a unique sample data set for each tree. A certain number of features from the total set of features are also randomly selected for each tree. The most important and

unique feature of Extra Trees is the random selection of feature split values. Instead of using Gini or entropy to compute locally optimal values to split the data, the algorithm chooses split values randomly. This makes the tree diversified and uncorrelated.

Light Gradient Boosting Method (LGBM): Light GBM is a gradient-boosting framework that uses a tree-based learning algorithm. Light GBM grows trees vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while other algorithm grows level-wise. It will choose the leaf with large loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm. LightGBM is called "Light" because of its computation power and giving results faster. It takes less memory to run and is able to deal with large amounts of data. However, Light GBM is sensitive to overfitting.

4 Analysis and Comparison of Models

In this section, we present our results from the modeling approaches and analyse the results. In Table 2, we have the names of the models used, and the total number of features used. Firstly, we used all the predictor features and then we used the top thirty features according to the results obtained from variable ranking, and lastly, we used the top twenty features according to the results obtained from variable ranking. The ranking for the full dataset is shown as a plot in Figure 3, and this ranking was based on Gini Index.

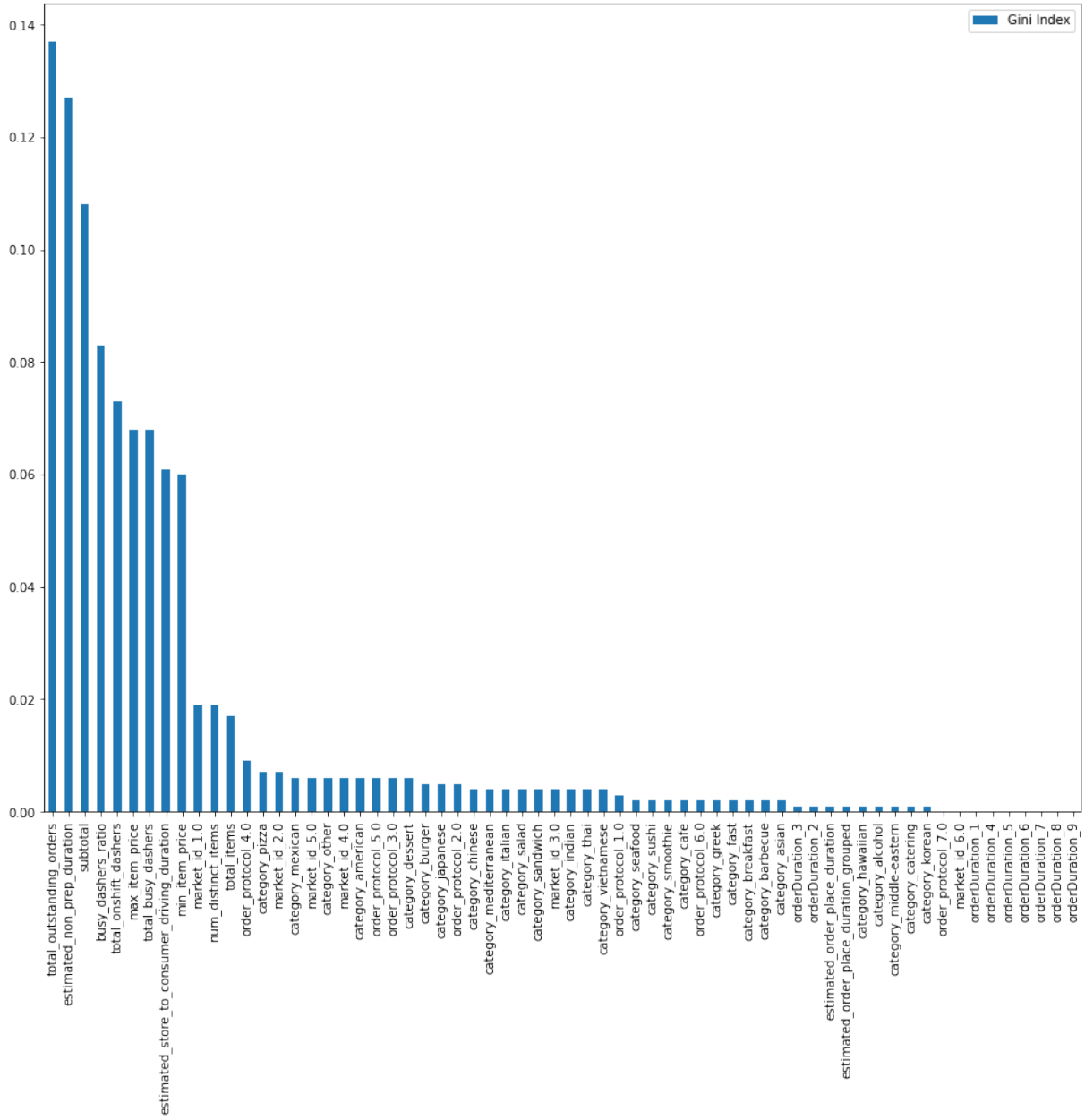

Figure 3: Plot of Variable Importance.

From Table 2, we see that, irrespective of using all the features or the top twenty or the top

thirty features, the RMSE values for the AdaBoost Regressor, K-Neighbors Regressor, Decision Tree Regressor, and the Extra Tree Regressor tends to be very high when compared to the other methods. Overall, we also see that wheter or not we consider all predictor variables, or the top twenty variables or the top thirty variables, the LGBM tends to perform better when compared to the other methods. The LGBM when we use all predictors gives an RMSE value of 904.856 seconds which is approximately 15 minutes. Conversely, if we use the top thirty variables in the LGBM model, our RMSE value is 886.677 seconds which is also approximately 14.8 minutes, and lastly when we use the top twenty features in the LGBM model, the RMSE value is 892.899 seconds which is approximately 14.9 seconds. For visualisation purposes, the plot of RMSE values and model names is shown in Figure 4. Additionally, the plots for the top twenty, and top thirty features are shown in appendix 5, and 6 respectively.

Regression Model	RMSE (Full Dataset)	RMSE (Top 30 Feats.)	RMSE (Top 20 Feats.)
Linear Regression	940.055	931.604	933.792
Ada Boost Regressor	2,554.58	1,402.06	1,439.00
Bagging Regressor	972.65	952.1	961.715
Gradient Boosting Regressor	920.852	914.776	915.807
Random Forest Regressor	927.278	908.355	915.231
Ridge Regression	940.055	931.604	933.792
Lasso Regression	940.053	931.614	933.797
K-Neighbors Regressor	1,088.60	1,088.40	1,088.37
Decision Tree Regressor	1,325.25	1,297.88	1,316.57
Extra Tree Regressor	1,343.75	1,332.56	1,344.46
XGB Regressor	908.549	891.126	897.033
LGBM	904.856	886.677	892.899
MLP	918.391	908.415	910.735

Table 2: Model Names and RMSE Values.

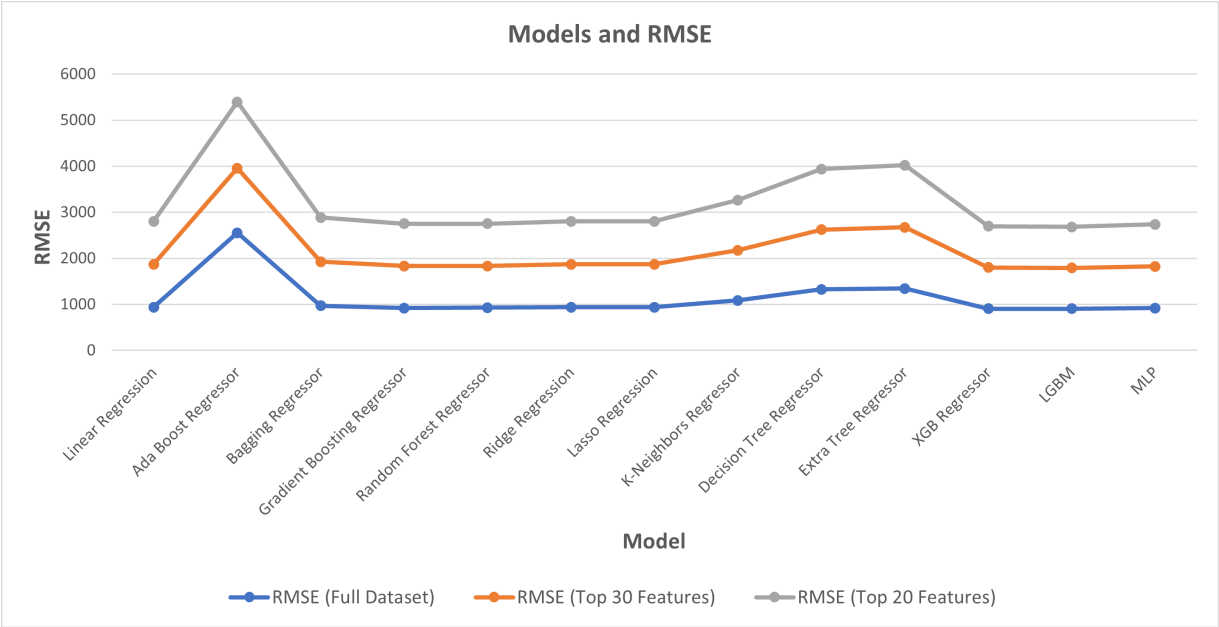

Figure 4: Plot of Models and RMSE Values.

From the results obatined, the XGB Regressor does a relatively similar job as the LGBM. With the XGB Regressor having RMSE values of 15.14 minutes on all predictors, 14.9 minutes on the top thirty features, and 14.95 minutes on the top twenty features, we see that these results are comparable to what we had when we use the LGBM. Generally, we realized that there was little to no difference in the number of predictors used when considering a particular model. For the K-Neighbors Regressor, we noticed that there was absolutely no difference in the number of predictors used. On the other hand, the Ada Boost Regressor shows a significant segregation in using all predictors versus using top twenty or thirty features. Interestingly and unsurprisingly, the Ridge and Lasso regression methods do not show that there any differences between these two methods. In particular, the linear regression surprisingly performed better than some boosting and bagging methods. This results is particularly interesting because when studying the correlation between the predictor variables and the response variable, there was hardly any evidence to establish that there was any sort of linear relationship between each predictor variable and the response variable. In the next section, we shall present the conclusions and some recommendations based on our results and analysis in this section.

5 Conclusions and Recommendations

In this section, we shall present the conclusions made from our analysis and also some recommendations based on the selected model. From the results obtained in section 4, we recommend the use of the LGBM. This model should be used in conjunction with the top twenty features obtained from the ranking of variables. Although using the top thirty features for the LGBM gives us the least the RMSE value, we noticed that the difference in RMSE values when we use top twenty features versus top thirty features is little, a difference of approximately six seconds. Adding ten features to the model only to improve the model by six seconds is not very helpful and hence that is why we are recommending the use of the LGBM with the top twenty features. Another recommendation will be to improve the LGBM by performing Hyper-parameter tuning. Given adequate time, the performance of the model can be improved by adjusting some arguments of the LGMB to achieve optimal performance. In the next section, we will discuss some limitations and implications of our results and analysis.

6 Limitations/Implications

There are several limitations associated with the study. Firstly, talking about limitations associated with the data, variables like order protocols and market id lacked interpretability, and there was no information as to what specific location each market id is associated with or what each protocol id means. Next, is that the variable estimated order place duration had anomalies in a way that most of the data were concentrated only on 2 values, and hence it does not look like natural data and we can assume that there might be a bug during the data collection or a transferring steps. Otherwise, there is an intuition that this variable would be one of the important variables for our model. And lastly, talking about data, the fact that we combined some less frequent restaurant types under the 'Other' category was another limitation.

Moving to the modeling part, hyper-parameter tuning would be another thing to work on further, as some of the models have tens of hyper-parameters, and tuning them using different optimization techniques potentially can improve the results. Moreover, in real-life, companies that predict delivery time have access to supplemental data like geographical, weather, real-time traffic measurements, and map data. Since total delivery time is composed of order place duration time, food preparation time, and store-to-consumer driving duration, using additional data might provide better predictions for the components of our response variable, and hence for the total delivery time duration itself.

7 References

[1] Data Source:
https://platform.stratascratch.com/data-projects/delivery-duration-prediction.

[2] More on Linear Regression in Python:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[3] More on Lasso Regression: https://dataaspirant.com/lasso-regression/

[4] More on Ridge Regression: https://dataaspirant.com/ridge-regression/

[5] More on Stochastic gradient descent (SGD) Regression:
https://scikit-learn.org/stable/modules/sgd.html#:~:text=Stochastic%20Gradient%20Descent%20(SGD)%20is,Vector%20Machines%20and%20Logistic%20Regression

[6] More on K-Nearest Neighbors (KNN) for Regression:
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

[7] More on Linear Support Vector Regression (SVR):
https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc167

[8] More on Bagging Regression:
https://www.numpyninja.com/post/gradient-boost-for-regression-explained

[9] More on Extra Trees Regression:
https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-extra-tree-classificat htm

[10] More on XGBoost:
https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

8 Appendices

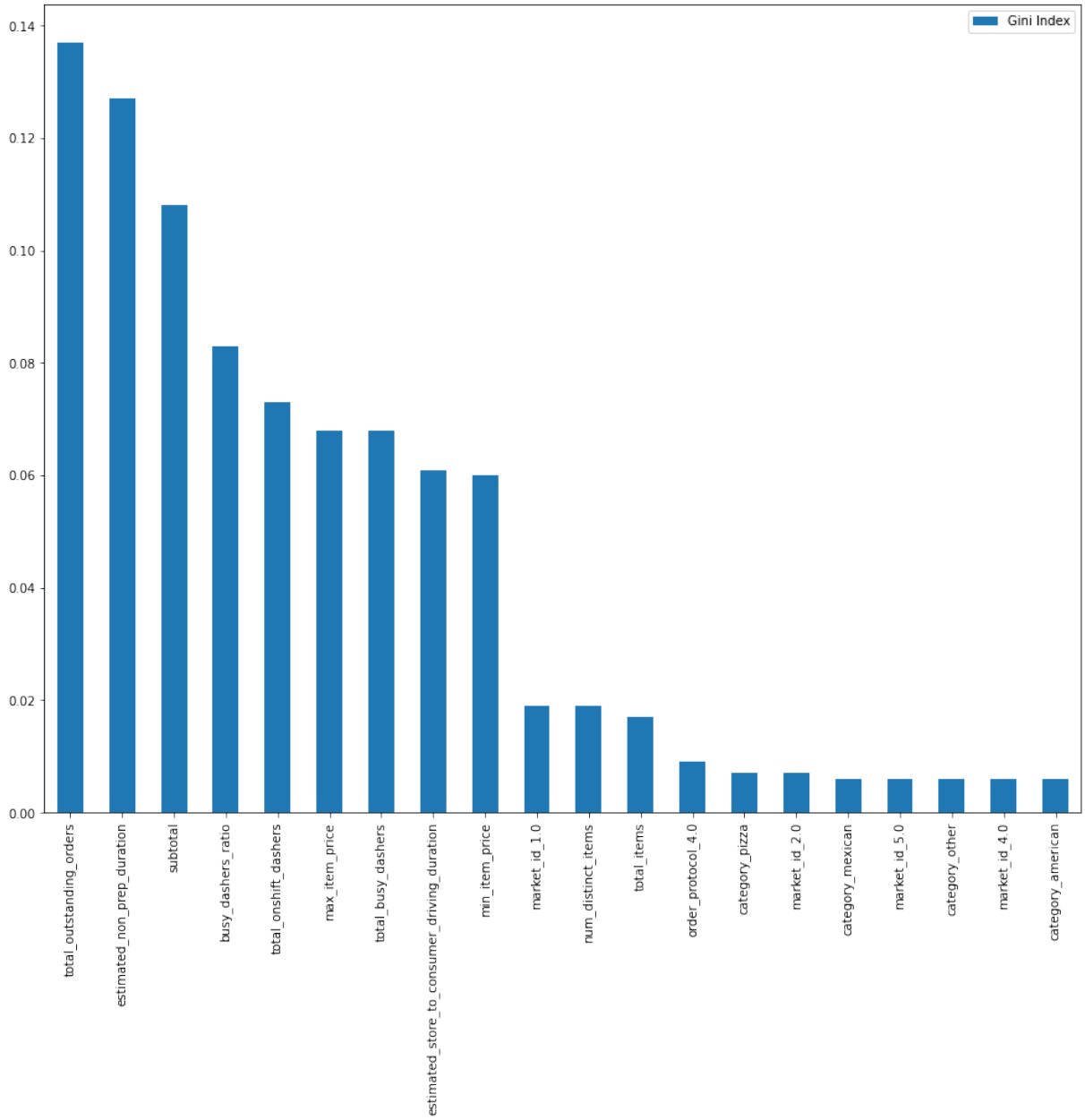

Figure 5: Plot of 20 Top Features.

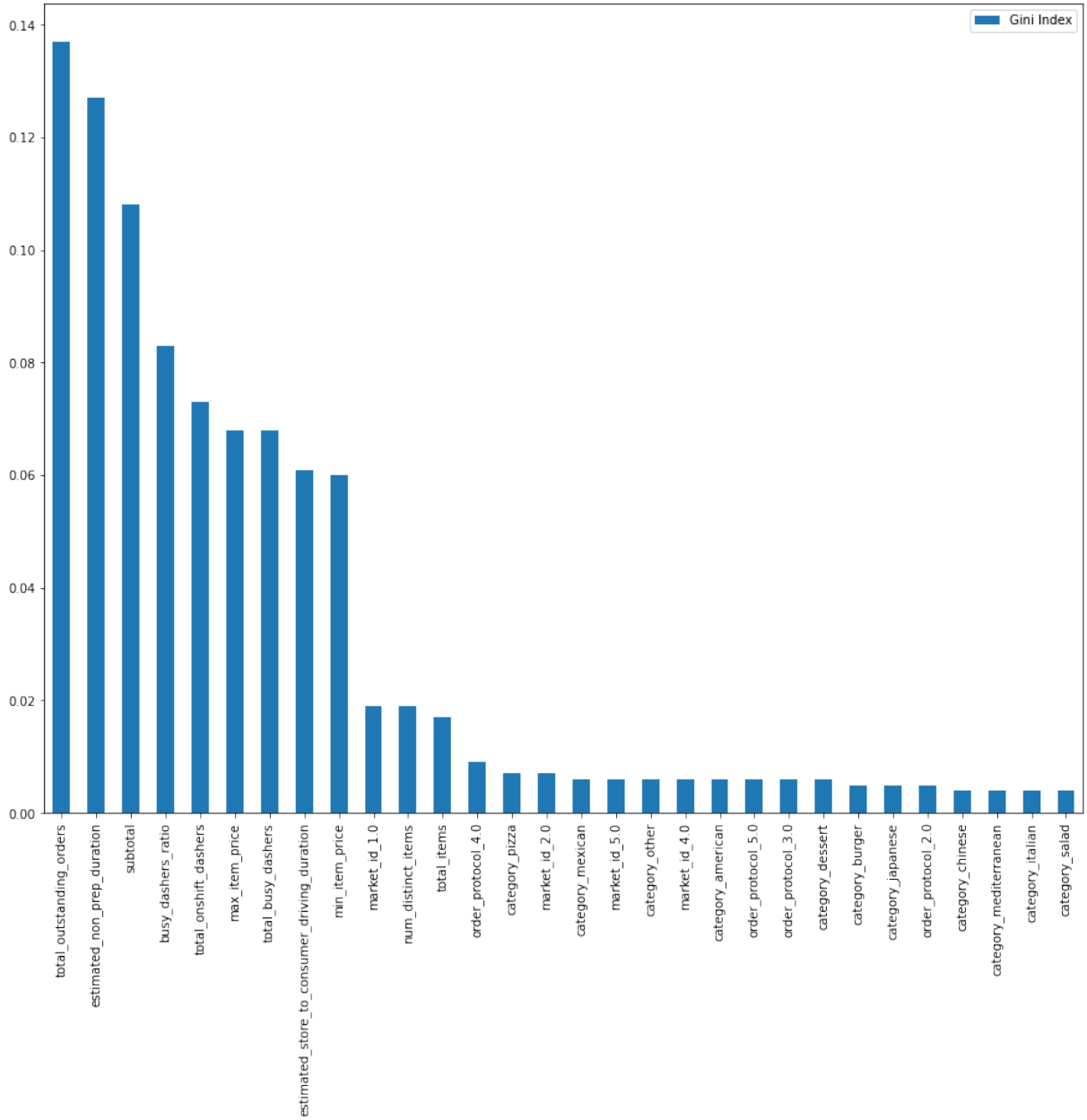

Figure 6: Plot of 30 Top Features.

8.1 A (Plots of Distributions of Variables)

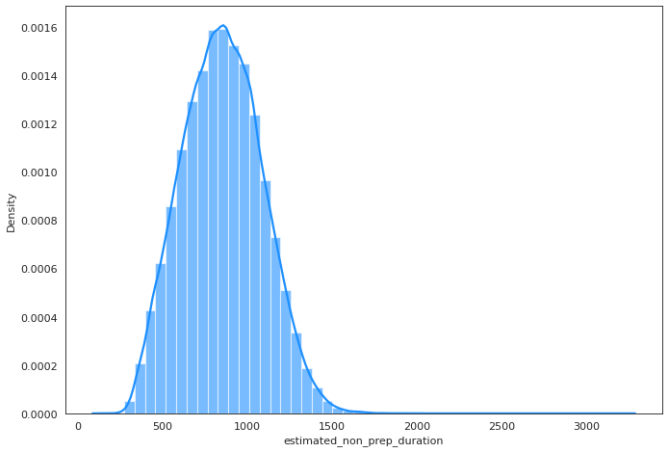

Figure 7: Plot of Estimated Non Prep Duration.

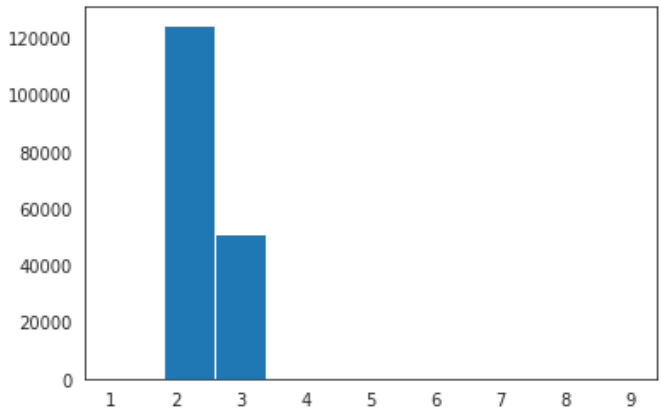

Figure 8: Plot of Estimated Order Placed Duration.

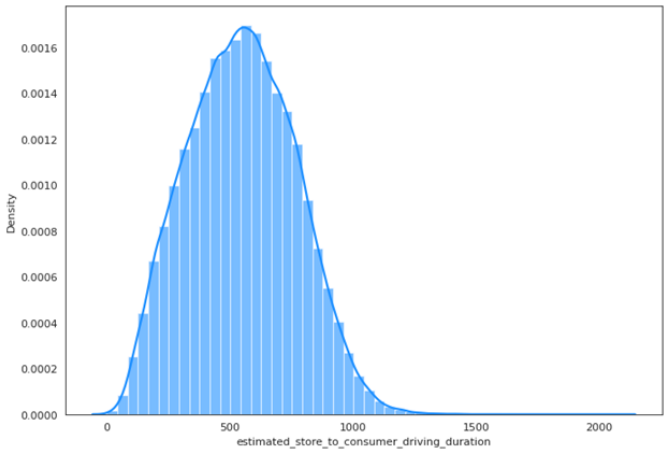

Figure 9: Plot of Estimated Store to Consumer Driving Duration.

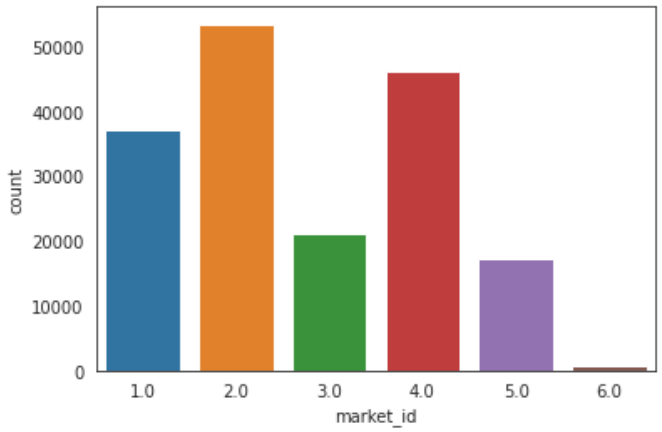

Figure 10: Plot of Market ID.

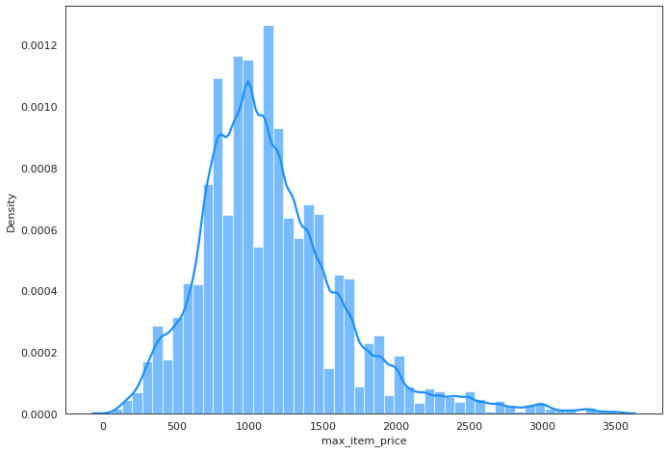

Figure 11: Plot of Maximum Item Price.

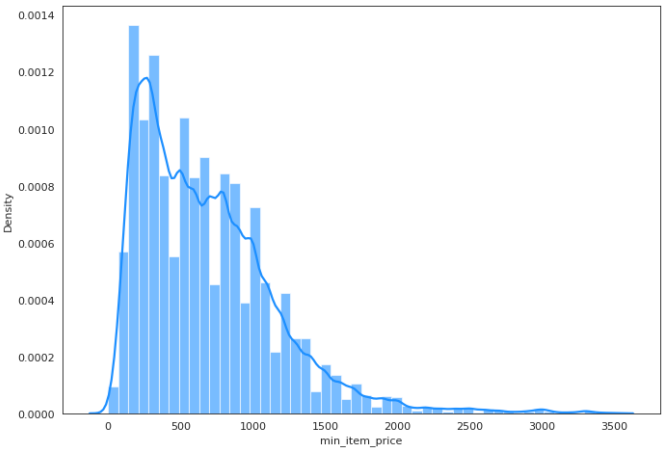

Figure 12: Plot of Minimum Item Price.

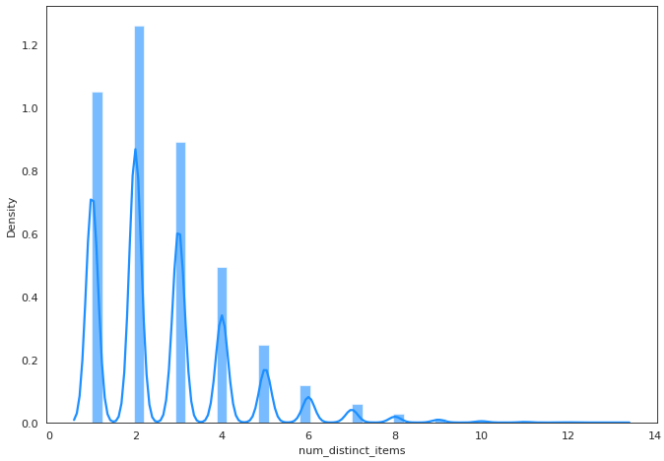

Figure 13: Plot of Number of Distinct Items.

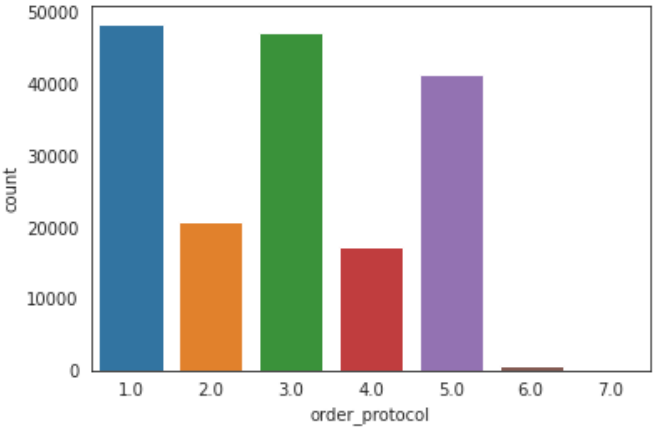

Figure 14: Plot of Order Protocol.

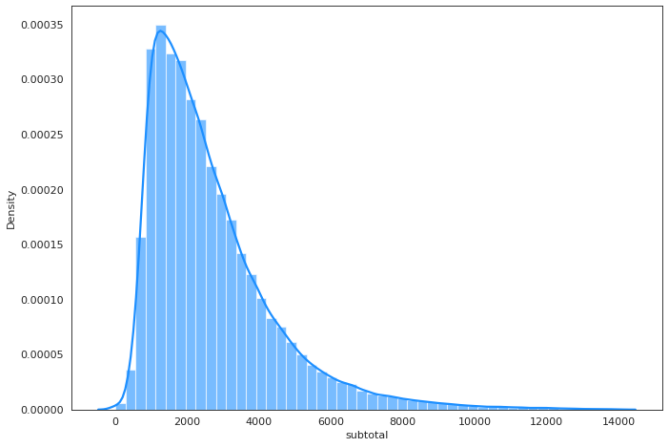

Figure 15: Plot of Subtotal.

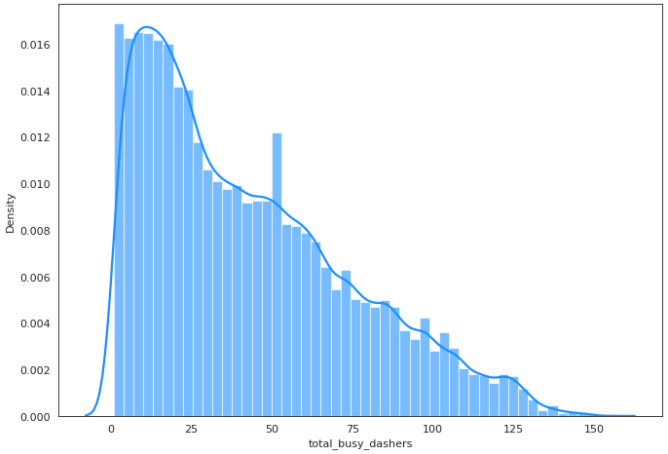

Figure 16: Plot of Total Busy Dashers.

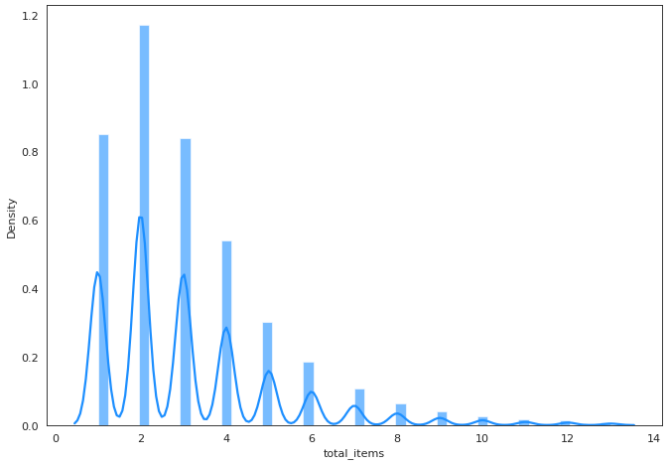

Figure 17: Plot of Total Items.

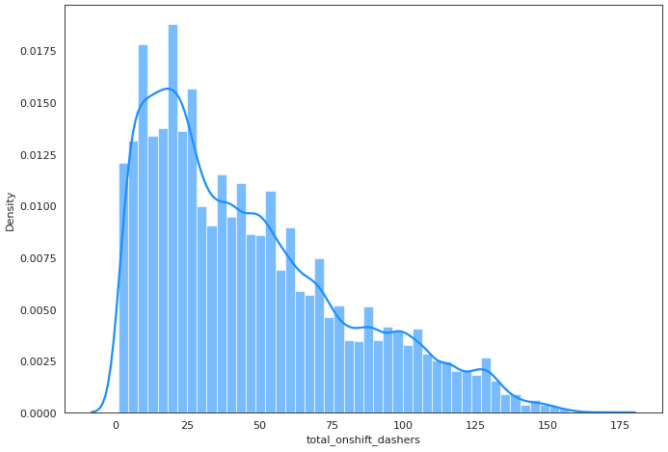

Figure 18: Plot of Total Onshift Dashers.

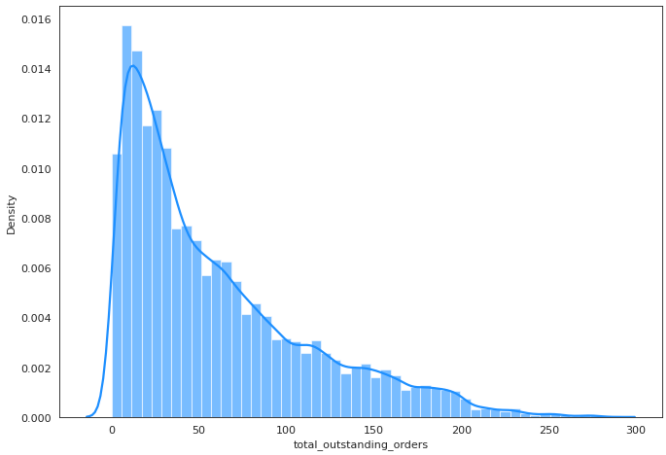

Figure 19: Plot of Total Outstanding Orders.

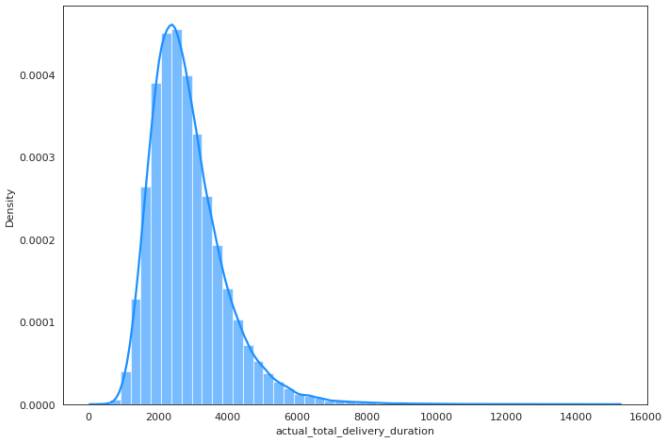

Figure 20: Plot of Actual Total Delivery Duration.

8.2 B (Scatter Plots of Response Versus each Predictor Variable)

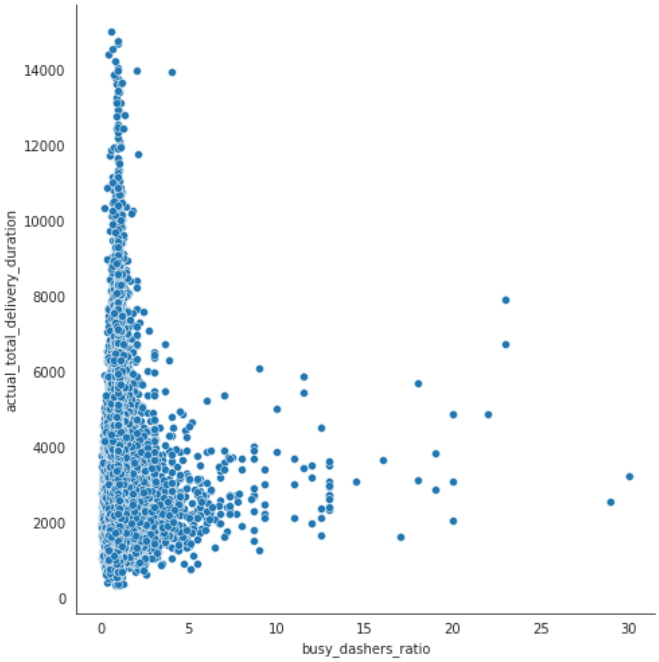

Figure 21: Actual Total Delivery Duration versus Busy Dashers Ratio.

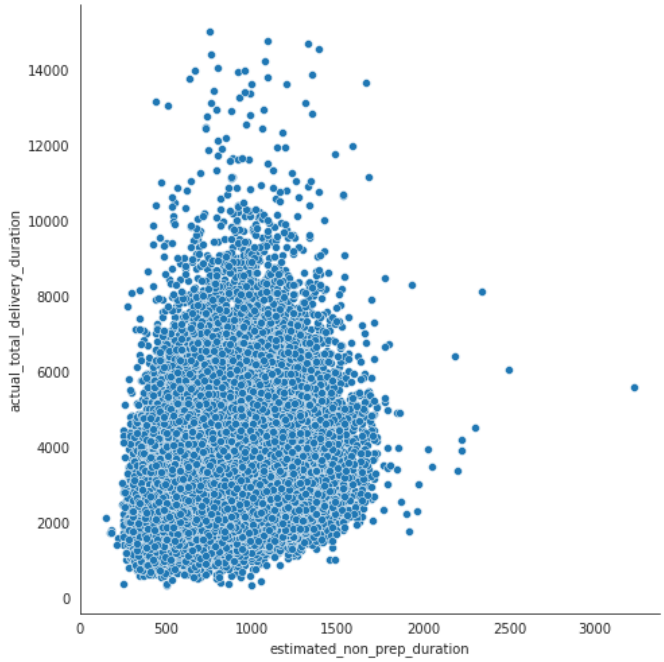

Figure 22: Actual Total Delivery Duration versus Estimated Non-Prep Duration.

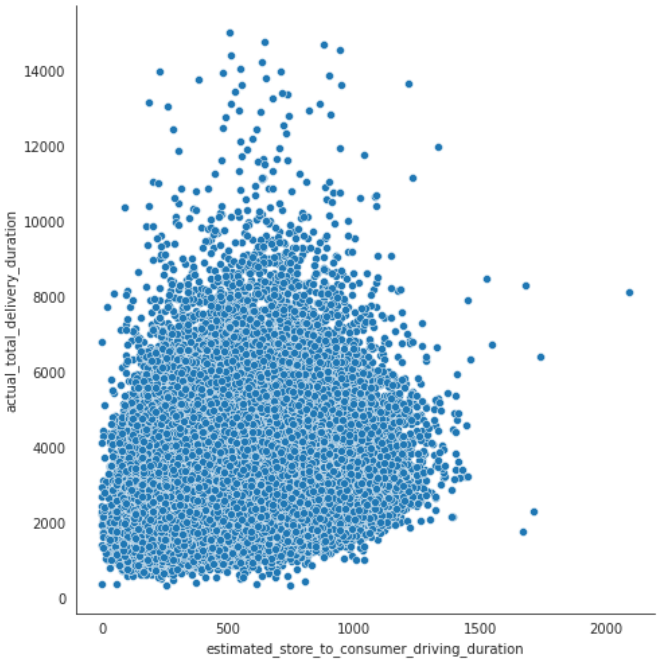

Figure 23: Actual Total Delivery Duration versus Est.Store to Consumer Driving Dur.

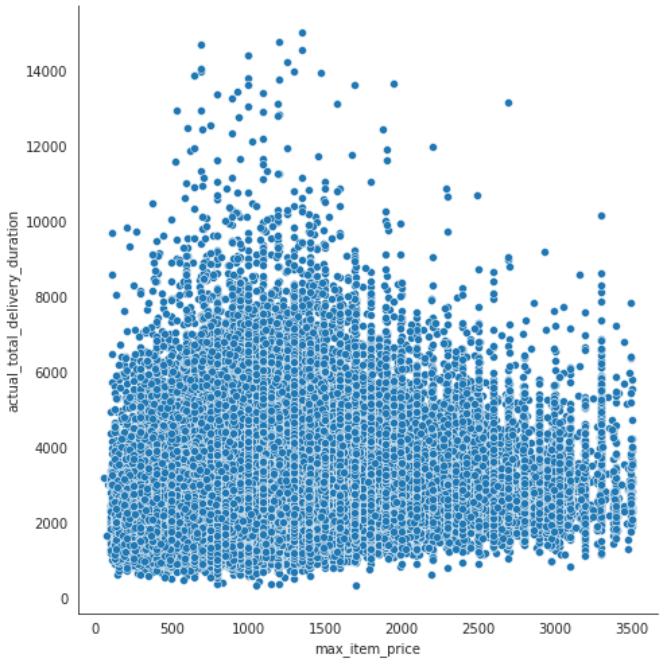

Figure 24: Actual Total Delivery Duration versus Maximum Item Price.

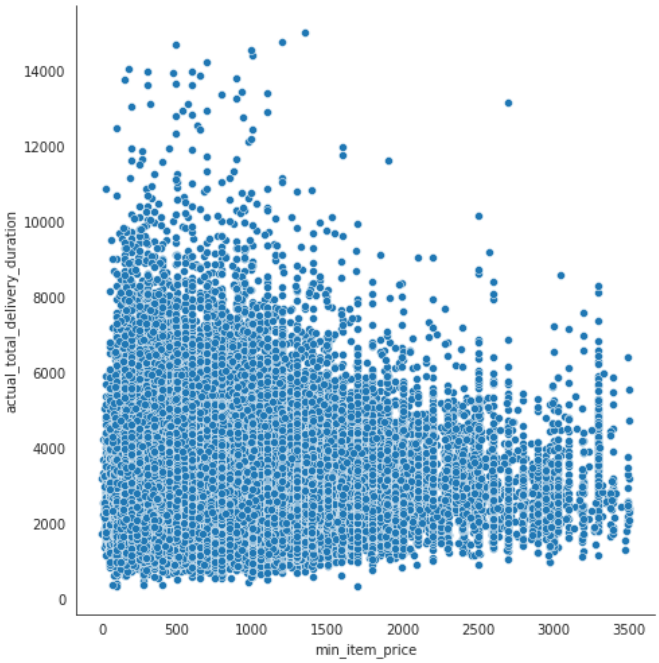

Figure 25: Actual Total Delivery Duration versus Minimum Item Price.

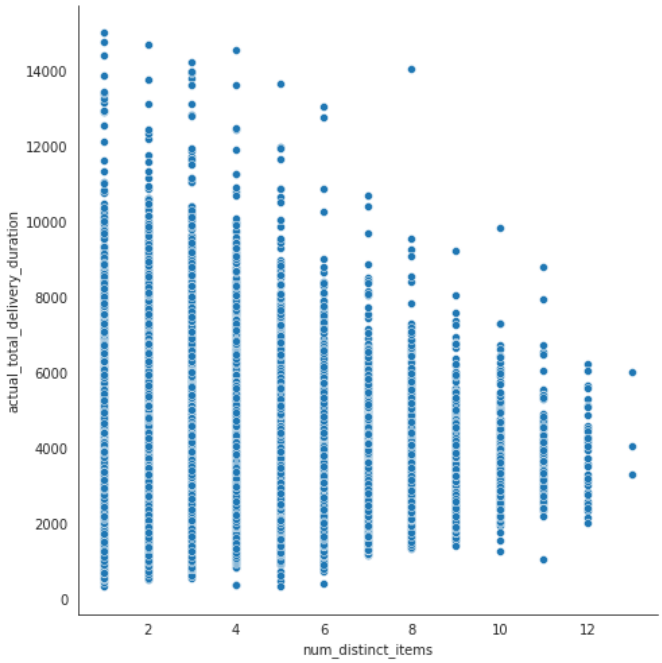

Figure 26: Actual Total Delivery Duration versus Number of Distinct Items.

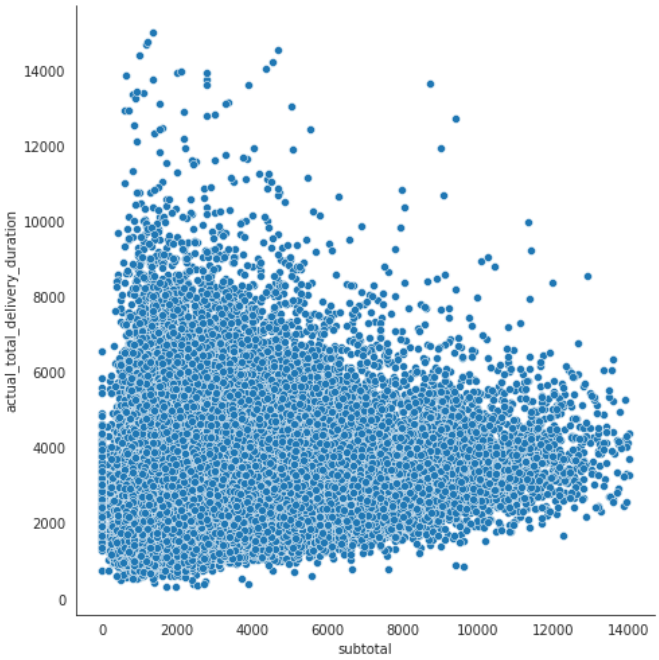

Figure 27: Actual Total Delivery Duration versus Subtotal.

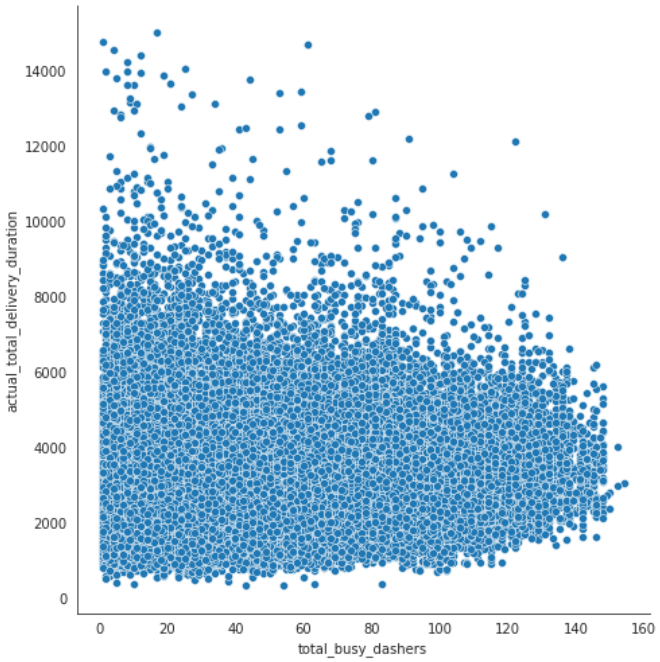

Figure 28: Actual Total Delivery Duration versus Total Busy Dashers.

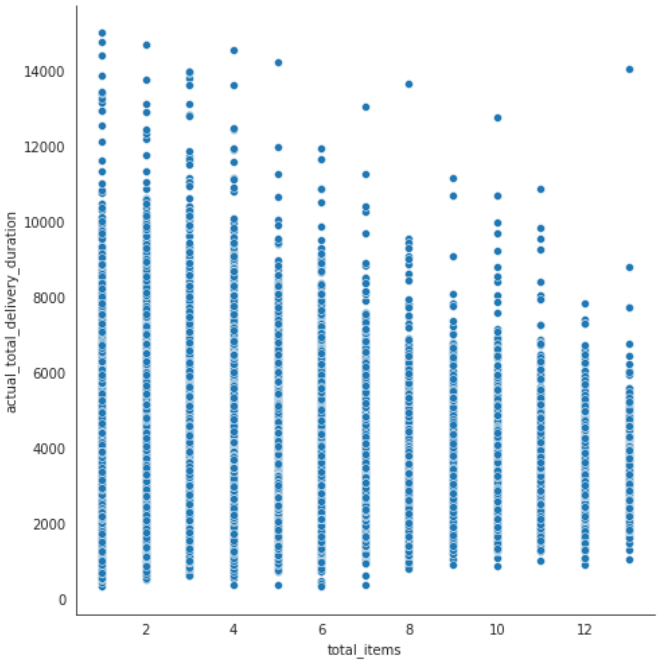

Figure 29: Actual Total Delivery Duration versus Total Number of Items.

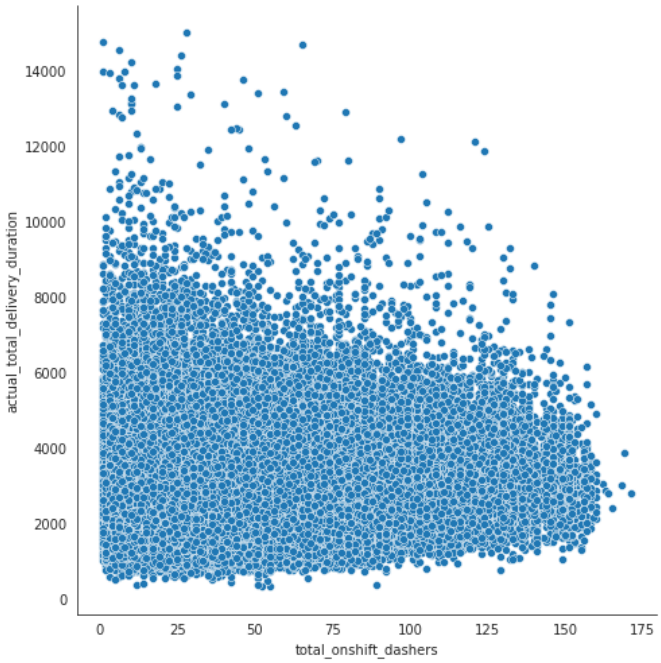

Figure 30: Actual Total Delivery Duration versus Total Onshift Dashers.

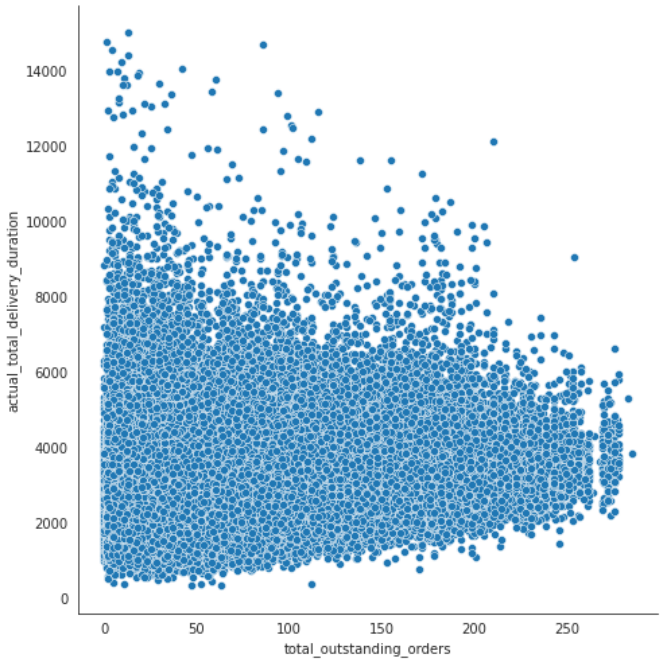

Figure 31: Actual Total Delivery Duration versus Total Outstanding Orders.

8.3 C (Plot of The Correlation Matrix)

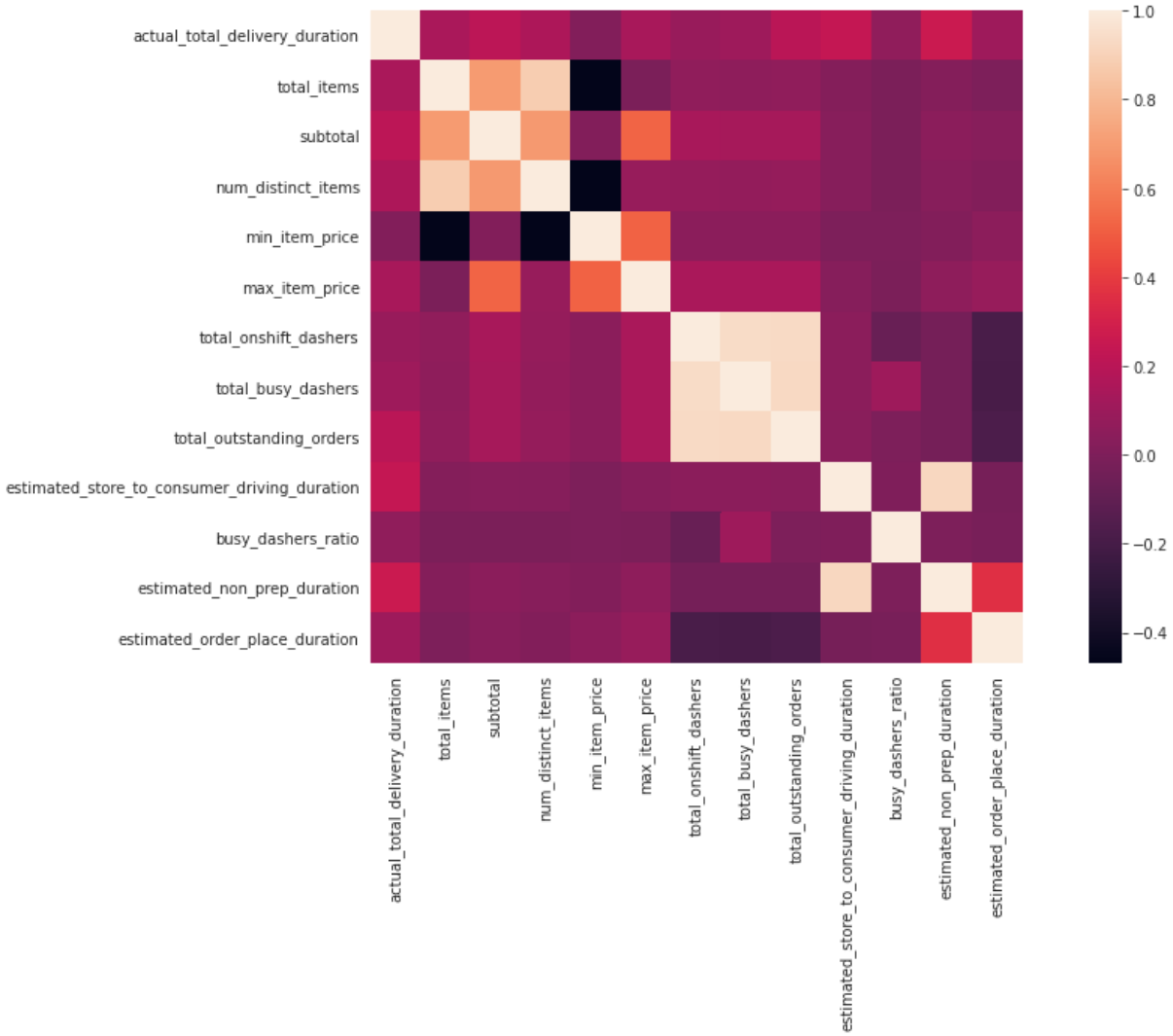

Figure 32: Plot of Correlation Matrix for Numerical Variables.

Variable Name	Description
market id	Represents a city/region in which DoorDash operates, e.g., Los Angeles.
created at	Timestamp in UTC when the order was submitted by the consumer to DoorDash.
actual delivery time	Timestamp in UTC when the order was delivered to the consumer.
store id	A 3 or 4-digit number representing the restaurant the order was submitted for.
store primary category	The Cuisine category of the restaurant. eg; Italian, Asian, Mexican,$\cdots$.
order protocol	The mode through which a store receives orders from DoorDash.
total items	The total number of items in the order.
subtotal	Total value of the order submitted.
num distinct items	Number of distinct items included in the order.
min item price	Price of the item with the least cost in the order.
max item price	Price of the item with the highest cost in the order.
total onshift dashers	Number of available dashers who are within 10 miles of the store at the time of order creation.
total busy dashers	Subset of above "Total onshift dashers" who are currently working on an order.
total outstanding orders	Number of orders within 10 miles of this order that are currently being processed.
estimated order place duration	Estimated time for the restaurant to receive the order from DoorDash.
estimated store to consumer driving duration	Estimated travel time between store and consumer.

Table 3: Description of the Variables.