

DoorDash Food Delivery Time Prediction

Prajakta Gujarathi | Data Scientist | 4/6/2020

# Problem Statement:

Before a consumer places an order on DoorDash, we show the expected delivery time. It is very important for DoorDash to get this right, as it has a big impact on consumer experience. Order lateness / underprediction of delivery time is of particular concern as past experiments suggest that underestimating delivery time is roughly twice as costly as overestimating it. Orders that are very early / late are also much worse than those that are only slightly early / late. In this exercise, you will build a model to predict the estimated time taken for a delivery.

Concretely, for a given delivery you must predict the total delivery duration seconds, i.e., the time from

● Start: the time consumer submits the order (`created\_at`) to

● End: when the order will be delivered to the consumer (`actual\_delivery\_time`).

To help with this, we have provided

● historical\_data.csv: table of historical deliveries (your training set)

● data\_to\_predict.csv: data for deliveries that you must predict (label-free test set we will use for evaluation)

● data\_description.txt: description of all columns in historical\_data.csv and details of data\_to\_predict.csv

## Analysis

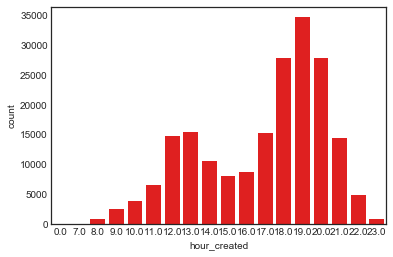
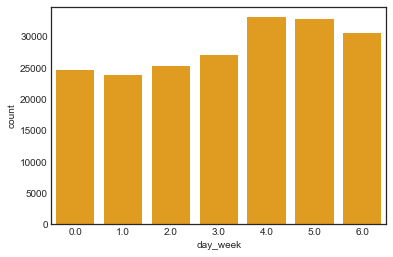
### Exploratory Data Analysis:

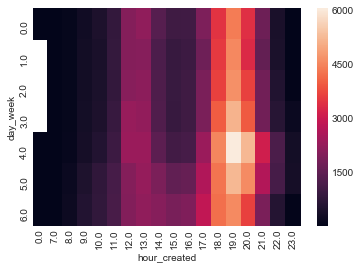
First, I converted order placed and delivery time to PST. This gives me clear idea about hour of the day and day of the week when order is placed.

Derived few features which I thought important in prediction from order created at time like:

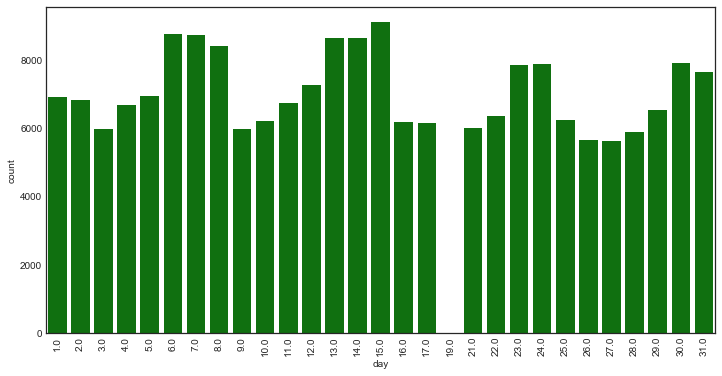
* + Day\_week,
  + Hour,
  + Day\_month

We can see from below chats that most of the order are placed during lunch, and evening after 6Pm – 9Pm and see the peak during weekend since Friday night to Sunday.





Also found some data point missing on 19th of the month.



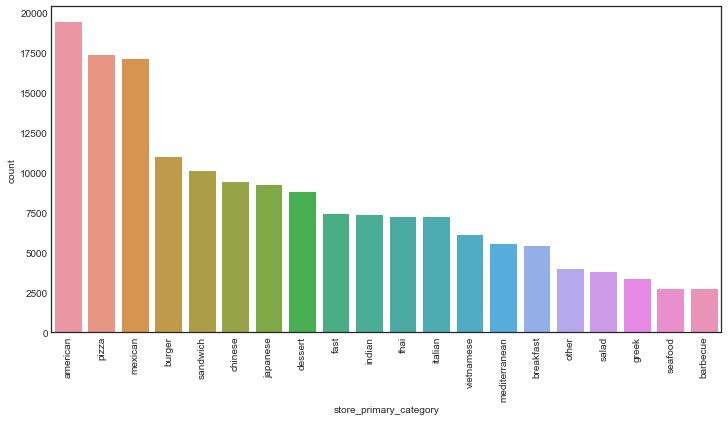
Market Id:

Market 1,2 and 4 has majority of data coming from. Also imputed data for market\_id with help of store\_id.

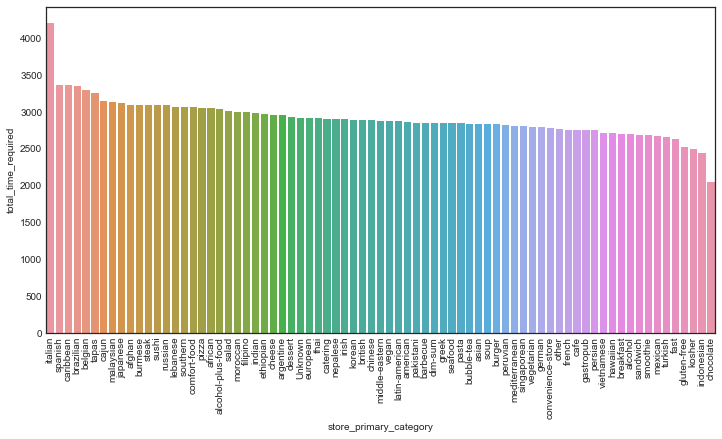
Store\_primary\_category:

Most popular store category came out to be American, pizza, Mexican.

Also, a single store has multiple categories.



Found out that Italian food takes avg longer time to deliver than any other food category. So this feature defiantly has importance in predicting delivery time.



Also found that each category has different avg time to prepare depending on the when the order is created. Created **lookup table(timeReqperCat.csv)** per category which will be used in test set for creating new feature.

* avg\_time\_to\_prepare\_per\_category\_per\_hour

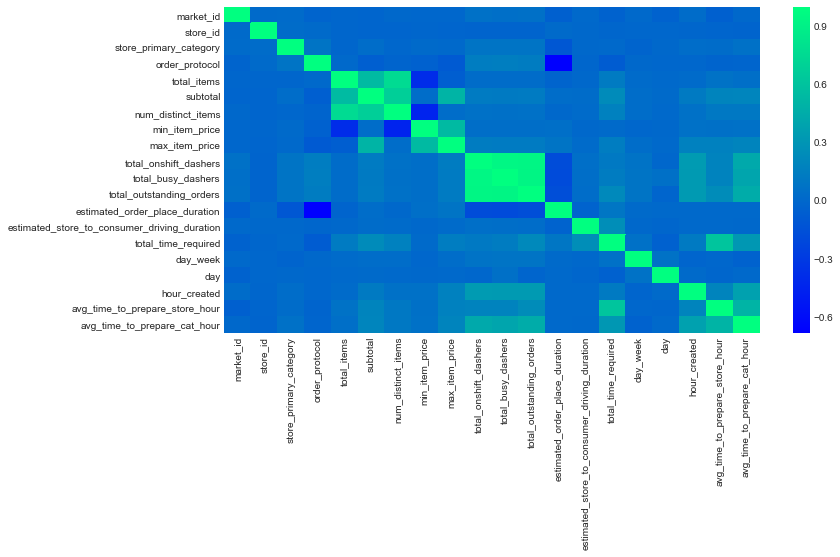
Store\_id:

This seems to be important feature.

Using store\_id created another feature which tell avg time a store takes to prepare order given the hour of the day. This is used as **lookup table(timeReqperstorhour)** which I will be using in creating feature in test data set.

* avg\_time\_to\_prepare\_per\_store\_per\_hour

Here is the correlation between different features we have:



Market features:

I found there are few values in these features that are negative. Assuming that was data entry or some type of error I have converted those features to positive values while training.

Also found like 40k records total\_onshift\_dashers < total\_busy\_dashers value. 40k records is large chunk to neglect it. So, I assume these values are correct and it due to the fact, shift changes or end of the shift that causing total\_onshift\_dashers < total\_busy\_dashers.

# Data cleaning and preprocessing:

Initially there are multiple feature with missing values.

For market feature I could not find any relation with other given features from the data set to impute missing values.

I am using MICE from impyute library to impute values for these features.

After treating missing values, I am handling outliers by using simple technique and some trial and error (using box plot) to remove outliers using IQR(inter quantile range) of selected features.

Added features like:

* avg\_time\_to\_prepare\_per\_category\_per\_hour
* avg\_time\_to\_prepare\_per\_category\_per\_hour

after data cleaning to get more accurate values.

# Model Creation:

**Metrics used to evaluate the model:**

A common metric that we can use to get a quick idea of our performance is the **Mean Absolute Error**. This tells us the average of the difference between our delivery time estimates and the actual delivery time.

MAE takes an average absolute difference between your prediction and observed reality. The metric treats positive and negative error equally. Plus, the measure uses the same scale as the underlying data.

Also, I have created few domain specific Metrics:

Considering 5min or less as on time delivery, and anything above or below 10min as too late/early delivery.

**% More than 5 min late or early**

**% More than 10 min late or early**

**Baseline model:**

Baseline model created using **DummyRegressor has scores:**

mean Absolute error: 710.8795304289841

r^2 values -3.011482348158623e-06

% More than 5 min late or early: 83.42276936026936

% More than 10 min late or early: 54.6875

**I tried different regression model to fit on initial features using sklearn library:**

RandomForestRegressor

r^2 train: 0.8927205443398808

r^2 test: 0.2869827517007354

mean Absolute error: 600.69333

rmse 748.5731047951829

% More than 5 min late or early: 80.80000000000001

% More than 10 min late or early: 46.0

GradientBoostingRegressor

r^2 train: 0.40989386827496066

r^2 test: 0.3195823343136316

mean Absolute error: 585.1551341958791

rmse 731.2602925845564

% More than 5 min late or early: 77.8

% More than 10 min late or early: 43.4

XGBRegressor

r^2 train: 0.41210216039410386

r^2 test: 0.31961737614712316

mean Absolute error: 585.8787283935546

rmse 731.2414622156275

% More than 5 min late or early: 77.0

% More than 10 min late or early: 43.0

LinearRegression

r^2 train: 0.26388757214031955

r^2 test: 0.26649911168021223

mean Absolute error: 610.3051307223869

rmse 759.2495157849355

% More than 5 min late or early: 78.8

% More than 10 min late or early: 47.199999999999996

KNeighborsRegressor

r^2 train: 0.2942858940367634

r^2 test: 0.016778251235537223

mean Absolute error: 699.3098

rmse 879.0426354847642

% More than 5 min late or early: 81.6

% More than 10 min late or early: 52.6

**Further Adding features like**

* avg\_time\_to\_prepare\_per\_category\_per\_hour
* avg\_time\_to\_prepare\_per\_category\_per\_hour

RandomForestRegressor

r^2 train: 0.9256666438077005

r^2 test: 0.5260773304694117

mean Absolute error: 464.9044400000001

rmse 610.2923738989371

% More than 5 min late or early: 63.6

% More than 10 min late or early: 33.4

**GradientBoostingRegressor**

**r^2 train: 0.602819234742102**

**r^2 test: 0.5512936861357172**

**mean Absolute error: 452.24597570520774**

**rmse 593.8343176954317**

**% More than 5 min late or early: 60.199999999999996**

**% More than 10 min late or early: 31.6**

**XGBRegressor**

**r^2 train: 0.5989404535318661**

**r^2 test: 0.5493803720051981**

**mean Absolute error: 453.22021801757813**

**rmse 595.0990457950102**

**% More than 5 min late or early: 59.199999999999996**

**% More than 10 min late or early: 32.0**

LinearRegression

r^2 train: 0.5215933624067564

r^2 test: 0.5461541566937305

mean Absolute error: 455.9922488975281

nomalized rmse 21.97935879471616

% More than 5 min late or early: 60.8

% More than 10 min late or early: 30.4

KNeighborsRegressor

r^2 train: 0.5326080222249927

r^2 test: 0.33442258185887075

mean Absolute error: 565.5776000000001

rmse 723.2417542150066

% More than 5 min late or early: 69.0

% More than 10 min late or early: 39.2

Till now XGBRegressor/ GradientBoostingRegressor are coming out to be winner.

Trying to avoid overfitting by removing non-important features using feature importance of xgboost.

Feature Importance:

Variable: avg\_time\_to\_prepare\_store\_hour Importance: 0.4000000059604645

Variable: estimated\_store\_to\_consumer\_driving\_duration Importance: 0.10000000149011612

Variable: estimated\_order\_place\_duration Importance: 0.05999999865889549

Variable: subtotal Importance: 0.05000000074505806

Variable: total\_outstanding\_orders Importance: 0.05000000074505806

Variable: hour\_created Importance: 0.05000000074505806

Variable: day\_week Importance: 0.03999999910593033

Variable: day Importance: 0.03999999910593033

Variable: num\_distinct\_items Importance: 0.029999999329447746

Variable: total\_onshift\_dashers Importance: 0.029999999329447746

Variable: total\_busy\_dashers Importance: 0.029999999329447746

Variable: market\_id Importance: 0.019999999552965164

Variable: order\_protocol Importance: 0.019999999552965164

Variable: min\_item\_price Importance: 0.019999999552965164

Variable: max\_item\_price Importance: 0.019999999552965164

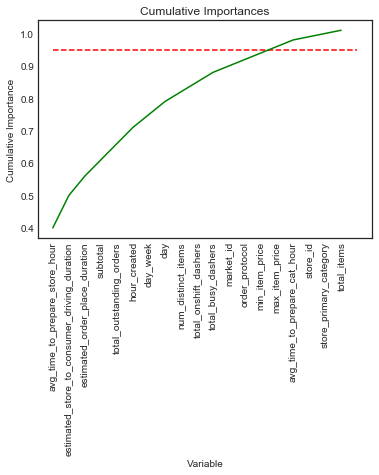
Variable: avg\_time\_to\_prepare\_cat\_hour Importance: 0.019999999552965164

Variable: store\_id Importance: 0.009999999776482582

Variable: store\_primary\_category Importance: 0.009999999776482582

Variable: total\_items Importance: 0.009999999776482582

Number of features for 95% importance: 15



Removing un-important features from training increased score slightly:

GradientBoostingRegressor

mean Absolute error: 450.74334476667855

% More than 5 min late or early: 59.4

% More than 10 min late or early: 31.0

XGBRegressor

mean Absolute error: 452.04122424316404

% More than 5 min late or early: 58.4

% More than 10 min late or early: 31.4

**Model tuning:**

XGBRegressor

mean Absolute error: 446.5844482849121

% More than 5 min late or early: 59.03

% More than 10 min late or early: 26.240000000000002

So, the reduction in 10 min delay from 54% to 26% , in 5 min delay from 83% to 26% from baseline model to tuned model!

Also, MAE is 446 which is like 7.4 mins error rate!

# Improvements:

Following are few Features that could be added for more improvement:

-**Exact location of dasher at the time of order creation**, so in order to determine pick-up time required for dasher.

-**Exact address of store and customer** for getting better understanding of available parking, elevators, gate-code which can be huge factor is delays

-**Distance between pickup location and delivery location**

**Total delivery time = pick-up time + travel time + drop off time**

-Kind of **vehicle dasher** is using (bicycle, car, scooter etc )

- Keeping database of **avg time required for each store per hour of the day**. Few stores might take more time to prepare or busy during certain time of the day than other time or other stores.

# Extra Work:

As a fan of Quantile Regression, I thought this might be problem where we can use Quantile Regression. I did not get enough time to finish of my work. But it might give us better prediction window for such problem statements.