Machine Learning Data Science Project

Xiaotong Mu

January 25, 2021

1 Introduction

Delivery time prediction has long been a part of city logistics, It is important for customer satisfaction that the initial prediction is highly accurate and that any delays are communicated effectively.

In this report, I'll discuss my roadmap of building the model that takes data about food delivery as input to predict the total delivery duration seconds.

Some other things to note:

- Drivers can be assumed to arrive at the supplier at exactly the dictated pick-up time. In reality this isn't always true, but with a sufficient number of drivers this can be kept to a minimal amount.
- Food should not be delivered too late as the customer will be left waiting and angry and not too early as it will have to sit out before the customer is ready to eat it.

2 Analyze, identify patterns, and explore the data

As an input to our model, we have five categories of data:Time Features, Store Features, Order Features, Market Features, Predictions From Other Models

Other than the existing features, I generated 6 additional features from the given data to improve model performance.

1. Total available dashers: get the number of available(free) dashers who are within 10 miles of the store at the time of order creation.
2. Average value: average value of each order
3. Order submission day of week
4. Order submission hour
5. Order submission month
6. Estimated delivery duration: Sum of estimated order place duration store to consumer driving duration

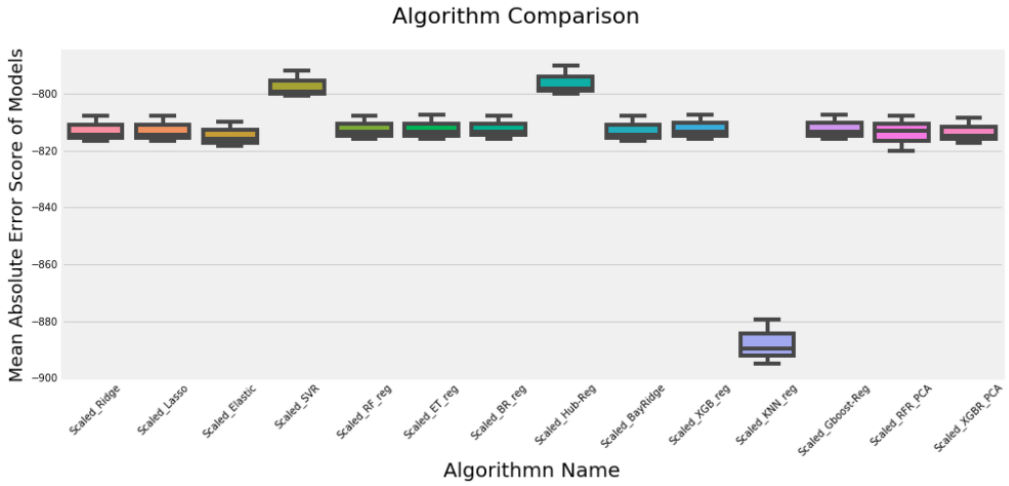
3 Model, predict and solve the problem

3.1 Training

After adding new features, removing redundant features and converting features into suitable form for modeling, now we are ready to train model and predict the required solution.The desired output of the model should be a numeric value, and due to the one hot encoding, we're left with a fairly large number of binary variables which means it's likely that many feature are important. With these three criteria, I narrow down my choice of models to a few. These include:

- Ridge Regression, Lasso Regression, Elastic Net Regression, SVR, Random Forest Regressor, Extra Trees Regressor, Bagging Regressor, Huber Regressor, Bayesian Ridge, XGBRegressor, Decision Tree Regressor, KNeighbors Regressor, Gradient Boosting Regressor, Random Forest Regressor and XGBRegressor

Figure 1: Mean Absolute Error Score of Models



I created the pipeline with the models to do a first evaluate of their power to this problem.

3.2 Performance Metrics

A common metric that we can use to get a quick idea of our performance is the Mean Absolute Error (MAE). This tells us the average of the difference between our delivery time estimates and the actual delivery time. It gives an idea of how wrong the predictions were.

Going further, we can define two more domain-specific metrics:

- Basically on-time ($\leq$ 5 mins late)
- A little late ($\leq$ 10 mins late)

These two metrics tell us how often we'll upset users. If we're 15 minutes late or 25 minutes late, it doesn't really matter to the user, we're just really late. MAE doesn't realise this, and will count one worse than the other but in terms of the business, it's effectively the same thing.

3.3 Report, and present the problem solving steps and final solution

After performing the typical processing and running the top 5 models we see the following:

Figure 2: After performing the typical processing and running the models

	% Greater than 5 mins	% Greater than 10 mins	R2 Score
SVR	33.643156	17.786501	17.677469
RandomForestRegressor	42.535140	25.250095	-2.412415
HuberRegressor	36.018741	19.853109	0.189443
XGBRegressor	41.509434	24.069900	0.294963
GradientBoostingRegressor	43.186020	25.941497	0.247152

Out of the five models, SVR model has the best performance.

Also, instead of train a model to predict the entire delivery time, I would suggest to break the delivery time up to the three main components discussed above, this way we can create specialty models which will have superior ability over one generic model.Combine the models together with the following formula:

Delivery Time = Pick-up Time + Point to Point Time + Drop-off Time + Hyperparamater

The additional hyperparameter can help us to minimise the final error metrics after calculating the combined result, for example we see in the data such as the pick-up time always been set 5 minutes earlier to meet the expectations of drivers.

4 Making Better Estimates

Food Delivery can be break down into three components, and the following features would improve model performance if included in the training set:

1. Pick-up the food from the supplier:

    - Supplier Feedback: how busy the supplier is at the time of pick-up, how many people are dine in: contact the restaurant on the day of the pick-up to check food will be ready on time. The response to these messages would be a useful flag to understand the delivery timing. Also, if the supplier is really busy at that time, it has a higher chance to mess up the order,if there is something missing from the order, drivers must wait for the extra food to be prepared
      Moving back to the business world, it is possible to discuss with the suppliers and identify the problems which can reduce the time spent at these suppliers and by reducing the variation, improve the overall performance of the model.
    - restaurant location(postcode or the latitude/longitude): postcode capture high-level generic information due to their specific structure, it can help us to know the parking availability at the destination

2. Drive from the supplier to the customer

    - Estimated travel time between store and consumer (existed feature): We can also use Google Maps API with the known pick-up and drop-off points. The results returned include parameters taking into account traffic and worse-case-scenarios which we can be used as input to the model.

3. Drop-off the food to the customer

    - Customer's location (address, post code):Once the driver arrives at the location, there are potential issues like building security issue and parking issues. Postal code is a great indicator as it tells us if we can expect a business district or a residential area. Also, if we have Level >30 it's quite obvious the delivery is in a big tower block and will likely have a complicated security process.

One simple way to avoid late deliveries is to take whatever value the model outputs and increase it to make sure that we're early enough. This is not a great solution because being too early also has a problem in that the food could go bad or be unprepared. However, it does bring up the point that we should consider the distribution of 'lateness' and 'earliness'.Another thing to consider is how the model is presented to users. I would recommend to present the consumer a delivery windows instead of a specific delivery time.