**Report on DoorDash Data Science, Machine Learning Project**

**High Level Summary**

In this project I used machine learning modeling to predict the expected delivery duration in seconds. The whole process includes data preprocessing and cleaning, some feature engineering, modeling, evaluation of prediction on a holdout set called validation set, and finally generating predictions for the given test set. Validation set is randomly selected from historical data for which the generated prediction is evaluated based on Mean Absolute Error (MAE) that shows average magnitude of error.

Preprocessing and data cleaning step in addition to dealing with missing values, involves removing some data points based on anomalous values found in target value (delivery duration) as well as some key features like total items. At the end this proves to have a very critical impact on improving MAE by close to 10%. In addition to using the list of provided features, I also added three features, one that is derived from two given features and two that are time related features. A side analysis I did with modeling without using these additional features shows that engineering these features helps to improve the overall MAE by over 5%. A comprehensive exploratory analysis is also included that shows how different features are correlated and related to the target value.

In modeling I used an ensemble model that is average of predictions generated by a three-layer dense neural network model as well as a XGBoost regressor model that are trained on the provided historical data. Evaluation of both models showed very close results on validation set, that prompted me to come up with an ensemble model based on the average of their predictions in order to use strength of both models meanwhile alleviating some of their weaknesses. Generated prediction on validation set based on this ensemble model results in MAE of about 636 that is lower than MAE obtained by each one of those models. That means, with delivery duration of average of 2860 seconds (47 mins), the generated prediction has average magnitude of error of about 10 minutes that gives a margin of error of about 22%. Finally, after some preprocessing, by using the ensemble model, predictions are generated for the test set and stored into a csv file called “yahya\_predicted\_data.csv”.

**Preprocessing and Feature Engineering**

Studying data statistics as well as exploratory analysis showed some existing anomalous values for the target value and some features. This analysis showed that most of the data have target values of lower than 30k, with few data points going beyond that which is a delay of over 8 hours being very unrealistic for food delivery time. Additionally, there are some anomalies for some features like “max\_item\_price” with one value being over 14k$, and “total\_item” with one data point having value of more than 200 items. Although, these data points are very few, using them in modeling, may push the model to stretch further in order to also fit these anomalous cases that can result in deteriorating effect on its performance. That is why, I removed all these outliers from historical data being use for training. A side analysis showed that removing these outliers can result in improvement of close to 10% in overall MAE.

In addition to the features provided in historic data I added three more features to improve model performance. One is the “total\_not\_busy\_dashers” that is the number of on-shift dashers that are not busy with orders. Considering the provided information that total number of busy dashers is a subset of on-shift dashers, this derived feature that is the subtraction of total number of busy dashers from on-shift dashers is clipped to the lower value of zero, after observing that there are some data points for which it goes to negative. In other words, the new feature of total not busy dashers is not only a difference between two given features, but it also has that given information embedded in it. After, adding this feature, correlation analysis shows that it has a higher correlation to the target value than the on-shift dashers that is removed from features since its information is inherently given in other two features.

I also added two time-based features, both derived from the “created\_at”, one showing the hour and the other one showing the day of the week at which the order was created. I used box plots to visualize how categorical and ordinal features are related to the target value. For example, for hour that is an ordinal feature it shows how delivery duration is distributed on different hours, and it can be seen that it is different for some hours like 14 and 2. This indicates a pattern that can be learned by the machine learning model and can help with prediction. Both hour and day of the week show some differences in distribution of target value, but this relationship is much more strongly seen for hour.

A quick side analysis shows that adding these features can reduce MAE from about 665-670 to MAE of 630-635 that is improvement of over 5% in the overall accuracy. Among the given features, I decided to not use “store\_id” because with over 6000 store data existing in the dataset, it would create a very large number of features after one-hot encoding, that would consequently slow down the training process. A side analysis I did showed that adding this feature had a very insignificant impact on overall MAE and that is why I decided not to use it in modeling.

**How to Evaluate Performance of the Designed Model**

To evaluate performance of the designed model, I used mean absolute error (MAE) as metric that measures average magnitude of errors in prediction without considering their direction. It is chosen because it gives a measure for error in the scale of data. In order to see if the selected model generalizes well and does not overfit, 20% of the historic data is held out as validation set on which model that is trained on the rest of the data is evaluated.

It should be noted that even though this holdout set is called validation set, in fact it is a test set on which the final model that has been trained completely independent from it gets evaluated. Moreover, since the dataset is large enough, we can assume that the randomly selected validation set is statistically representative of the whole dataset, and that is why there was no need to use cross validation for the final evaluation. Based on all this, I can conclude that the evaluation of the final trained model on the randomly selected validation set can be a good indicator of how well this model will generalize, and its obtained MAE on this set will be close to its performance on other test sets provided that the selected test set has similar statistical characteristics as the training data.

As a result, the MAE of approximately 636 seconds that algorithm obtained on the validation set, that is 22% of the target value with average of 2860, will be close to its performance on the given test set and can be used to compare its performance with any other algorithm that is trained on this data.

**Modeling Approach**

For Model selection, I used 80% of the data that is randomly selected from the whole historic data. First of all, I started with a dense neural network and tried to come up with a structure that almost overfits. For that, I used validation ratio of 0.2 while training, meaning 20% of the training data will be used to validate the trained model measuring validation loss at each epoch. In order to stop when there is no significant drop in validation loss for some consecutive epochs, I used early stopping callback in Keras. I came up with a 3-layer network with 512 neurons at the first layer and trained it that was stopped at epoch 29 and according to the smoothened historic graph started overfitting from epoch 10 with validation loss going over the training loss. To mitigate this overfitting problem, I used the same network but with dropout layers having rate of 0.1 after each layer in the network and trained it with monitoring validation loss at each epoch until it stopped at epoch 37 when it had no significant improvement in the validation loss for 5 consecutive epochs. Observing the graph of historic results over these epochs we can see that the model with dropout layers no longer overfits, and unlike the first model it has been able to go below MAE of 650 for validation loss. After finding the best epoch on which the lowest validation loss is achieved, I had my dense neural network ready with all selected parameters to train on all training set with the selected number of epochs. Then I evaluated the trained model on the holdout set consisting of 20% randomly selected samples from historic data. The evaluation results show that the selected dense neural network model obtains MAE of about 643 on validation set.

Next, I tried XGBoost regressor model for modeling the data. [XGBoost](https://xgboost.ai/) is a decision-tree-based ensemble Machine Learning algorithm that uses a [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting) framework. When it comes to small-to-medium structured/tabular data, decision tree-based algorithms and specially XGBoost algorithm are considered best-in-class right now. In order to choose best hyper-parameters for the XGBoost regressor model I used grid search with a two-fold cross validation (it already takes 39 mins to run with 2 folds) that searches through the given set of parameters and trains model with all combinations, twice on each since we have 2-fold, and selects the combination and parameters with the lowest average loss, and finally trains the selected model on all of training set. In this process the best values for the parameters including learning\_rate, max\_depth, min\_child\_weight and n\_estimators with total of 16 combinations, are selected. To further combat overfitting I chose parameters subsample, and colsample\_bytree to be 0.8. at the end, the trained model is evaluated on the validation set that resulted in MAE of 640.

Since both selected models of XGBoost and dense NN have very close performance on the validation set, I decided to use an ensemble of both models as the main model that takes average of their outputs and delivers that as prediction. The main motivation behind this is that this ensemble enables us to take advantage of both models’ strength and meanwhile alleviate their strength as can be seen from the final MAE that the ensemble model obtained on the validation set, which is about 636 and is below both models’ obtained MAEs.

After model selection the final step is to preprocess and transform the test data and to use the ensemble model in order to generate prediction on that. For that, similar preprocessing and feature engineering steps that were done on the training set, are performed on the test set as well, plus one-hot encoding for categorical and ordinal variables as well as normalization using standard scaler, that was fit on the training data. Finally, after fitting both selected XGBoost and NN models on all of the historic data and generating predictions by both models on the test dataset, ensemble model predictions are generated by taking average of both predictions and persisted to the column “predicted duration”. The final predictions together with the original test set are stored to the file “yahya\_predicted\_data.csv”.