INFO 371 - Final Paper

Paul Garces, Saimanasvi Charugundla, Ryan Nishi, Rishita Reddy

Question

Our project centers around the question: **What factors influence the DoorDash delivery drop-off time (ETA), and how accurately can we predict this?** We want to know what specific variables, such as order size, restaurant type, or the number of available drivers, have the most impact on the time it takes for a DoorDash delivery to reach the customer. Once we identify the key factors, we aim to build models to predict the drop-off times with high accuracy.

With this project, we're particularly interested in predicting the actual time from when the customer places an order to when the delivery reaches their door. By understanding the influence of each factor, we can see which variables are most crucial for predicting delivery times.

Motivation

Our motivation for choosing this question stems from the high stakes involved in food delivery, where timing is critical for both customers and businesses. DoorDash, being one of the largest food delivery platforms, relies on accurate ETA predictions to ensure that both customers and drivers have the best possible experience. Predicting delivery times accurately allows the platform to manage customer expectations better, allocate drivers more efficiently, and streamline restaurant operations during peak hours.

For customers, an accurate ETA can mean the difference between a pleasant experience and frustration due to late deliveries. For DoorDash, inaccurate predictions could lead to higher customer complaints, loss of trust, and inefficient use of drivers. Additionally, optimizing delivery times can also mean fewer idle drivers, better route management, and improved order batching, which saves costs and improves overall delivery speed.

In this project, we hope to dive deep into the various factors that could influence delivery times—whether it's the time of day, the number of dashers available, the type of food being delivered, or the size of the order. By building a model to predict ETA, we aim to highlight areas where DoorDash can improve its services and provide customers with more reliable delivery times. Through this, we not only gain a better understanding of the complexities of last-mile delivery but also create a tool that can be used to improve efficiency in real-world applications.

Data Description

The DoorDash ETA Prediction dataset contains DoorDash delivery data from early 2015, focusing on the time it takes to complete deliveries in various cities, and was collected from Kaggle. Before using our actual data, we converted the 'created_at' and 'actual_delivery_time' columns to datetime formats and calculated the `delivery_duration` variable by subtracting the ‘created_at’ from the ‘actual_delivery_time’, which in turn created our target variable (`deliver_duration`), which represents the total time in seconds from order placement to delivery.

The dataset includes time-based features such as `market_id` (representing different cities or regions), along with timestamps for when orders are created and completed. Features include `store_id`, identifying the restaurant, and `store_primary_category`, which indicates the type of cuisine (e.g., Italian, Asian). Additionally, the `order_protocol` feature captures the method by which restaurants receive orders from DoorDash. The dataset also contains order-specific information like the total number of items in an order, `subtotal` (total order value in cents), and details about item prices.

Market features provide insight into the operational environment at the time of the order, such as the number of available dashers (`total_onshift_dashers`), the number currently working on deliveries (`total_busy_dashers`), and the number of orders being processed nearby (`total_outstanding_orders`). The dataset also includes predictions from other models, such as estimated times for order placement (`estimated_order_place_duration`) and the time to drive from the store to the customer (`estimated_store_to_consumer_driving_duration`).

We cleaned the dataset by dropping rows with missing values for key features like created_at, actual_delivery_time, total_onshift_dashers, total_busy_dashers, and total_outstanding_orders. This reduced the dataset to 180,677 rows. This decision was made because the dataset remained sufficiently large enough after dropping these rows, thereby maintaining the representativeness and quality of the dataset. After handling missing values, we split the dataset into training and testing sets using an 80/20 split ratio.

Previous Work

Various people have used the dataset to predict the estimated delivery times of DoorDash orders, as we aim to do. The most common machine learning algorithm that has been used to do this task has been linear regression since it’s relatively straightforward in helping understand the relationship between the features and the target variable/outcome in the dataset. Users like Elliot (2021) have found the most success using these methods. Mu (2021) is an example of a user who has applied more advanced algorithms such as Extreme Gradient Boosting, Light Gradient Boosting, Ridge Regression, Multi-layer Perceptron Regressor, and some Deep Learning algorithms.

Additionally, users who have worked with the dataset have performed feature selection and scaling, by checking for collinearity and multi-collinearity between variables, Random Forest Regressors, and PCA. These algorithms and techniques are used to help identify and find the most important features to create better predictions and a more efficient model.

One last previous work of interest is actually from DoorDash themselves where Lu (2021) focused more on the problem of “long-tail events” which are large deviations from the average that occur with some regularity. These occurrences are too regular to be called outliers, so there is some potential worth in trying to predict them.

Descriptive Statistics

The dataset contains DoorDash delivery data from early 2015, encompassing various cities and features related to orders, dasher availability, and delivery predictions. Missing values are notable in several columns, including market_id (987), store_primary_category (4,760), order_protocol (995), and dasher-related features like total_onshift_dashers, total_busy_dashers, and total_outstanding_orders (16,262 each). Additionally, estimated_store_to_consumer_driving_duration has 526 missing values.

Order-specific features like total_items range from 1 to 411, with an average of 3.2 items, while subtotal ranges from 0 to 27,100 cents, averaging 2,682 cents. Outliers are present in min_item_price and max_item_price, including negative values that suggest data entry errors. Market-specific variables, such as total_onshift_dashers (average 44.8) and total_busy_dashers (average 41.7), also show negative values, likely due to temporary tracking inaccuracies. Similarly, total_outstanding_orders averages 58 and is right-skewed, reflecting instances of high demand. Time-based features reveal that estimated_order_place_duration averages 309 seconds, while estimated_store_to_consumer_driving_duration averages 545 seconds, both with right-skewed distributions.

Most features show right-skewed distributions, with notable outliers indicating variability in order size, dasher workload, and delivery times. Negative values in dasher-related features and min_item_price highlight potential data inaccuracies requiring cleaning. Outliers in time-based variables suggest delays from factors such as restaurant inefficiencies or traffic conditions, impacting delivery performance.

Ridge Regression

For our model, we used Ridge Regression to predict the delivery duration. Ridge Regression is a regularized linear regression technique that helps mitigate the problem of multicollinearity and overfitting by adding a penalty term to the regression equation.

We began by selecting relevant features from our dataset, including total_items, subtotal, num_distinct_items, min_item_price, max_item_price, total_onshift_dashers, total_busy_dashers, total_outstanding_orders, estimated_order_place_duration, and estimated_store_to_consumer_driving_duration. These features were chosen as they provide insight into the various aspects of a DoorDash delivery that could impact delivery time.

We then trained a Ridge Regression model using RidgeCV, which performs cross-validation to select the optimal regularization parameter (alpha). The values of alpha tested were [0.1, 1.0, 10.0, 100.0], and the optimal alpha value chosen by cross-validation was 100.0, indicating a high level of regularization.

Our results showed a training Mean Squared Error (MSE) of 989,457.92 and a test MSE of 965,861.93. The similarity between the training and test MSE suggests that the model generalizes well to unseen data without overfitting or underfitting. This outcome indicates that dropping rows with missing values allowed the model to learn more effectively from cleaner data.

With an optimal alpha value of 100.0, the model remains heavily regularized, which helps prevent overfitting by penalizing large coefficients and ensuring a more robust fit. However, with a Root Mean Squared Error (RMSE) of 982.71 seconds, around 16 minutes and 23 seconds, there is still room for improvement.

To further improve model performance, we experimented with a simplified version of the Ridge Regression model, using only the top three relevant features: num_distinct_items, total_onshift_dashers, and total_outstanding_orders. These features were selected based on their importance in the original model, as they had the largest absolute coefficients, indicating their strong predictive power for the delivery duration.

The results from the simplified model showed an RMSE of 1,027.52 seconds, around 17 minutes, which is higher than the RMSE from the original model that used all the features (approximately 16 minutes). This indicates that the simplified model with only the top three features was less effective at accurately predicting the delivery duration compared to the original model.

Based on this comparison, we concluded that using all the original features yielded better predictive performance. While simplifying the model reduced its complexity, it also resulted in a loss of predictive power. The additional features used in the original model likely captured important aspects of the delivery process that were not fully represented by just the top three features.

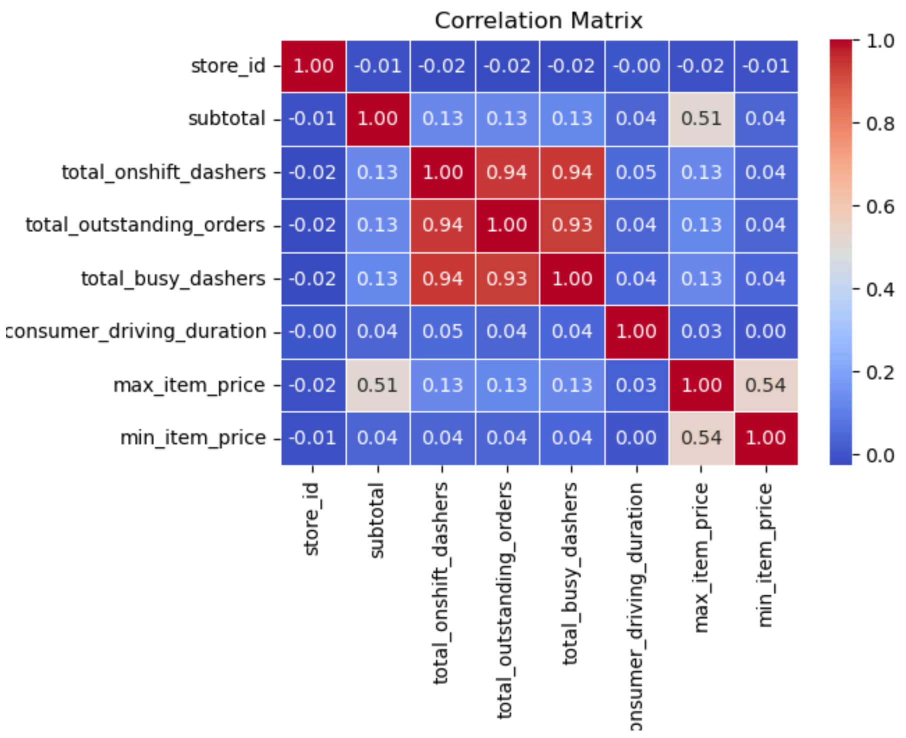
Therefore, while the simplified model is more interpretable and less complex, the original model provides better accuracy and generalizability for predicting DoorDash delivery times. Future work could involve experimenting with other models, such as neural networks, to capture more complex relationships within the data and further improve predictive accuracy.

KNN

We utilized the K-Nearest Neighbors (KNN) algorithm as a baseline model to predict delivery durations. KNN is a non-parametric method that makes predictions based on the average of the k-nearest points in the feature space. We selected this model because it provides an intuitive approach to understanding the relationships in the data without assuming any specific form for those relationships.

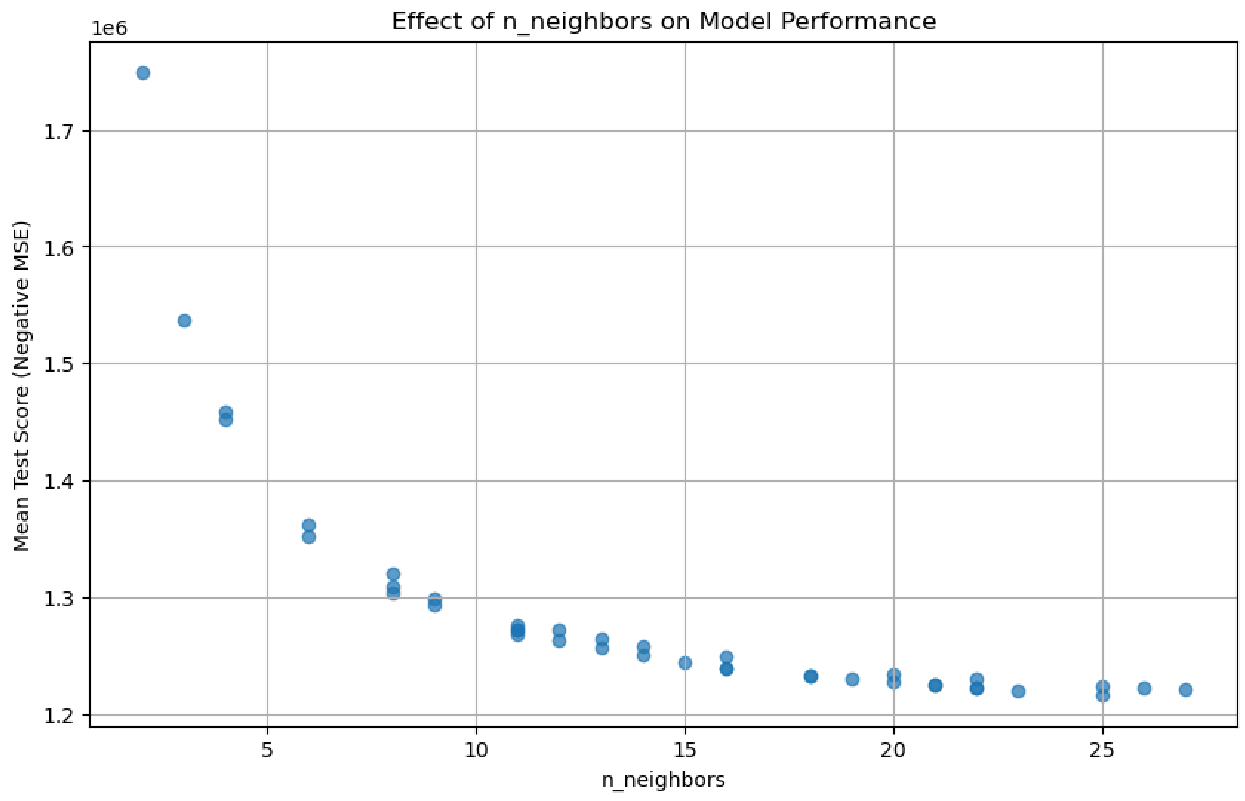
To begin, we prepared our dataset by selecting relevant features such as subtotal, total_items, num_distinct_items, min_item_price, max_item_price, total_onshift_dashers, total_busy_dashers, total_outstanding_orders, estimated_order_place_duration, and estimated_store_to_consumer_driving_duration.

Our initial KNN model used 5 neighbors, and after fitting the model, we achieved a Mean Squared Error (MSE) of 8,140,982.78 and a Root Mean Squared Error (RMSE) of 2,853.24 on the test set. The feature importance was then assessed by fitting a Random Forest Regressor, which identified estimated_store_to_consumer_driving_duration, total_outstanding_orders, subtotal, and total_onshift_dashers as the most influential features. We further filtered the features with a correlation matrix to retain only the most important ones, which allowed us to focus on a reduced set of variables, potentially improving model performance.



After filtering the features, we trained a new KNN model with 5 neighbors and observed a slight change in the model performance, with an MSE of 8,228,234.44 and an RMSE of 2,853.24. This indicated that the feature reduction did not significantly impact model performance, suggesting that the remaining features captured the primary factors influencing delivery duration.

To further improve the model, we performed hyperparameter tuning using RandomizedSearchCV to optimize key parameters, such as the number of neighbors (n_neighbors), weighting strategy (weights), and distance metric (metric). After 40 iterations of random sampling, the best parameters were identified as n_neighbors = 19, weights = 'uniform', metric = 'minkowski', and p = 3. With these optimized parameters, we trained the KNN model again and achieved an improved MSE of 8,065,579.07 and an RMSE of 2,853.24.



The results from the optimized KNN model indicated that while there was some improvement in the Mean Squared Error, the overall model performance did not drastically change. This suggests that the KNN algorithm may have limitations in capturing the complexities of the data, and exploring other algorithms, such as neural networks or ensemble methods, may yield better results in predicting delivery durations more accurately.

KerasRegressor

As for constructing a neural network architecture that could predict DoorDash delivery times, we began by using the MLPRegressor, however, due to the extensive search for the optimal hyperparameters, we switched to the KerasRegressor as it provides the flexibility needed to tune these hyperparameters. To begin with, we began by constructing a single hidden layer neural network, with different neuron values for each model (7, 5, and 3 respectively) and all 9 features, however, we kept the rest of the parameters such as the activation function (ReLu), number of batches (32), number of epochs (75), scoring (mean squared error), and optimizer (Adam) the same throughout. The values of the neurons in each of the different single-layer architectures were decided after doing research which can be found in the references. The test MSE for the model using 7 neurons was 1067306.11 and the test RMSE was 1033.11 seconds (appx. 17 min. 13 sec.). The test MSE for the model using 5 neurons was 1077688.10 and the test RMSE was 1038.12 seconds (appx. 17 min. 18 sec.). Finally, the test MSE for the model using 3 neurons was 1081025.29 and the test RMSE was 1039.72 seconds (appx. 17 min. 20 sec.).

After working with single hidden-layer neural networks, we decided to create a function that optimized the different hyperparameters such as the number of hidden layers, neurons, batches, and epochs, as well as using all 9 of the features. To do this, we used the RandomizedSearchCV function with a cross-validation value of 3 to test the different combinations of these hyperparameters. Additionally, we kept the optimizer, activation function, and scoring the same throughout this whole process (Adam, ReLU, and mean squared error respectively). Furthermore, once the program finished running, it would output the weights of the features to the first layer since this is where we can see the raw weights before any combinations are made. Once this program finished running, the optimized architecture resulted in a two-hidden-layer neural network with 128 neurons in each layer, 32 batches, and 75 epochs which resulted in a test RMSE of 1011.79 seconds (appx. 16 min. 52 sec.). The most important features from the 9 total in this model resulted in all of them besides `min_item_price` and `total_items`, as these were below the threshold weight of 25 that I selected.

Finally, we used the same function as mentioned above but with the newly found 7 features to tune the hyperparameters. With this new model, we found that the optimal architecture was a three-hidden-layer network with 64 neurons in each layer, a batch size of 64, and an epoch size of 75. The test RMSE resulted in 1014.70 seconds (appx. 16 min and 55 sec.) which was 3 seconds off/later than the model that used all 9 features and two layers. Therefore, we would select the initial optimized model that used all 9 features with two hidden layers, 128 neurons in each layer, 32 batches, and 75 epochs due to its small but better RMSE value.

Conclusion

As for our conclusion to our questions, we found that the best model for predicting the time taken for a DoorDash order to be delivered was the Ridge Regression model with all 9 features and an alpha value of 100.0, which results in a test RMSE of 982.71 (appx. 16 min. 23 sec.). The first optimized neural network architecture with all 9 features and 2 hidden layers was a close second, while the KNN model was the worst model out of all the models run.

Work Distribution

As for how we distributed the work, we decided our tasks based on how comfortable we felt with doing what was required and split all the work up equitably among all four group members. Regarding the models that we ran, Ryan performed the KNN models, Saimanasvi the Ridge Regression models, and Paul performed the Neural Networks models which meant that each member was able to discuss their results and process.

Future Directions

If given an additional quarter to work on the project, we would definitely focus on tuning the hyperparameters of each of our models which is one of the limitations we faced this quarter. For example, in the Ridge Regression model we could test a wider of range of alpha values, in the KNN model we could test a more extensive neighbors value, and as for the Neural Network model, we could test a wider range of values for the hidden layer size, neurons per layer, and optimizers, among others. Given the limited computational capabilities of our computers, being able to test a wide range of hyperparameters for the models was extremely time consuming and required a lot of power from our computers.

Beyond hyperparameter optimization, we would also want to expand the dataset by bringing in and training and testing with external variables such as weather conditions, traffic patterns, and peak order times, all of which are critical factors influencing delivery durations. These additional features would make the models more comprehensive and reflective of real-world conditions.

Moreover, the development of real-time data pipelines for live predictions would be a key focus. This would involve implementing efficient preprocessing steps, integrating continuous data streams, and deploying models to handle demands. Finally, we would prioritize evaluating model fairness by examining potential biases and ensuring equitable performance across different customer demographics, markets, and order types.

These enhancements would not only hopefully improve our model accuracy but also provide more insights for practical deployment. By addressing these limitations, the project could achieve its full potential in predicting delivery times with greater precision and reliability.

Works Cited

Dawn Lu, Pratik Parekh. “Improving Eta Prediction Accuracy for Long-Tail Events.” *DoorDash*, 26 Nov. 2024, careersatdoordash.com/blog/improving-eta-prediction-accuracy-for-long-tail-events/.

Dutta, Sanjay. “Number of Neurons per Hidden Layer in Neural Networks: A Guide.” *Medium*, Medium, 5 June 2024, medium.com/%40sanjay_dutta/number-of-neurons-per-hidden-layer-in-neural-networks-a-guide-106fea04fbfe#:~:text=The%20“Stretch%20Pants”%20Approach,down%20to%20the%20optimal%20size.

ES1927ES192717311 gold badge11 silver badge55 bronze badges, and Mihai Alexandru-IonutMihai Alexandru-Ionut 48.3k1313 gold badges105105 silver badges132132 bronze badges. “How to Create a Neural Network for Regression?” *Stack Overflow*, 1 Apr. 1963, stackoverflow.com/questions/49008074/how-to-create-a-neural-network-for-regression.

Karpathy, Andrej. “A Recipe for Training Neural Networks.” *A Recipe for Training Neural Networks*, 25 Apr. 2019, karpathy.github.io/2019/04/25/recipe/.

Karthikvadlamani. “Karthikvadlamani/Doordash-Delivery-Predictions: Prediction of Delivery Times for Doordash Deliveries. Performed Feature Engineering (Creation, Encoding), Feature Selection Using (Multi)Collinearity Analysis, Gini Importance and PCA. Applied 6 ML Models to Perform Regression Analysis on Delivery Time Prediction.” *GitHub*, github.com/karthikvadlamani/doordash-delivery-predictions. Accessed 8 Dec. 2024.

Krishnan, Sandhya. “How Do Determine the Number of Layers and Neurons in the Hidden Layer?” *Medium*, Geek Culture, 6 Aug. 2022, medium.com/geekculture/introduction-to-neural-network-2f8b8221fbd3#:~:text=Number%20of%20Neurons%20In%20Input,as%20a%20regressor%20or%20classifier.

Mannerow. “Mannerow/Doordash-Duration-Prediction: An End-to-End Machine Learning Project Predicting Doordash Delivery Durations, Utilizing MLOps Principles and Best Practices.” *GitHub*, github.com/Mannerow/doordash-duration-prediction. Accessed 8 Dec. 2024.

pr338pr3389, et al. “How Big Should Batch Size and Number of Epochs Be When Fitting a Model?” *Stack Overflow*, 1 Mar. 1961, stackoverflow.com/questions/35050753/how-big-should-batch-size-and-number-of-epochs-be-when-fitting-a-model.

Rob HyndmanRob Hyndman 58.2k2929 gold badges148148 silver badges199199 bronze badges, et al. “How to Choose the Number of Hidden Layers and Nodes in a Feedforward Neural Network?” *Cross Validated*, 1 Sept. 1955, stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw.

Sachdev, Harpreet Singh. “Choosing Number of Hidden Layers and Number of Hidden Neurons in Neural Networks.” *LinkedIn*, 20 Apr. 2021, www.linkedin.com/pulse/choosing-number-hidden-layers-neurons-neural-networks-sachdev/.

Suryaa, Dharun. “Doordash Eta Prediction.” *Kaggle*, 18 June 2024, www.kaggle.com/datasets/dharun4772/doordash-eta-prediction.