Capstone Project

(Delivery Duration Prediction)

When a consumer places an order on DoorDash, we show the expected time of delivery. It is very important for DoorDash to get this right, as it has a big impact on consumer experience. 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 taken from

Start: the time consumer submits the order (created_at) to

End: when the order will be delivered to the consumer (actual_delivery_time)

Data Description

The attached file historical_data.csv 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. We have added noise to the dataset to obfuscate certain business details. Each column corresponds to a feature as explained below. Note all money (dollar) values given in the data are in cents and all time duration values given are in seconds

The target value to predict here is the total seconds value between created_at and actual_delivery_time.

Columns in historical_data.csv

Time features

market_id: A city/region in which DoorDash operates, e.g., Los Angeles, given in the data as an id

created_at: Timestamp in UTC when the order was submitted by the consumer to DoorDash. (Note this timestamp is in UTC, but in case you need it, the actual timezone of the region was US/Pacific)

actual_delivery_time: Timestamp in UTC when the order was delivered to the consumer

Store features

store_id: an id representing the restaurant the order was submitted for

store_primary_category: cuisine category of the restaurant, e.g., italian, asian

order_protocol: a store can receive orders from DoorDash through many modes. This field represents an id denoting the protocol

Order features

total_items: total number of items in the order

subtotal: total value of the order submitted (in cents)

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 (in cents)

max_item_price: price of the item with the highest cost in the order (in cents)

Market features

DoorDash being a marketplace, we have information on the state of marketplace when the order is placed, that can be used to estimate delivery time. The following features are values at the time of created_at (order submission time):

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.

Predictions from other models

We have predictions from other models for various stages of delivery process that we can use:

estimated_order_place_duration: Estimated time for the restaurant to receive the order from DoorDash (in seconds)

estimated_store_to_consumer_driving_duration: Estimated travel time between store and consumer (in seconds)

Practicalities

Build a model to predict the total delivery duration seconds (as defined above). Feel free to generate additional features from the given data to improve model performance. Explain:

model(s) used,

how you evaluated your model performance on the historical data,

any data processing you performed on the data,

feature engineering choices you made,

other information you would like to share your modeling approach.

We expect the project to take 3-5 hours in total, but feel free to spend as much time as you like on it. Feel free to use any open source packages for the task.

1. What are the key factors affecting delivery duration?

- Perform Exploratory Data Analysis (EDA) to understand the relationships between features and the delivery duration. This includes correlation analysis, feature importance using decision trees or random forests, and visualizations like scatter plots, histograms, and box plots.

2. How to preprocess the data for model training?

- Implement the following preprocessing steps:
    - Convert timestamps to datetime objects and create new time-based features such as hour of day, day of week, etc.
    - Handle missing values through imputation or by removing rows/columns with excessive missing data.
    - Normalize/scale numerical features.
    - Encode categorical features using techniques like one-hot encoding.
    - Split the data into training and testing sets.

3. What is the baseline model performance?

- Create a simple baseline model using Linear Regression. Evaluate the performance using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.

4. Can more complex models improve prediction accuracy?

- Train more sophisticated models such as:
    - Decision Trees and Random Forests
    - Gradient Boosting Machines (e.g., XGBoost, LightGBM)
    - Neural Networks (e.g., using TensorFlow/Keras)
    - Evaluate and compare their performance using cross-validation and metrics mentioned earlier.

5. How to tune hyperparameters for optimal model performance?

- Use grid search or randomized search with cross-validation to find the best hyperparameters for the chosen models.

6. Can feature engineering improve model performance?

- Create new features from existing ones, such as:
    - Time-related features: hour of day, day of week, etc.
    - Interaction features between numerical variables
    - Aggregated features like average order value per store, average delivery time per store, etc.

    - Re-evaluate models with these new features.

7. How to handle imbalanced data if certain delivery times are rare?

- If the dataset is imbalanced, consider techniques such as:
    - Resampling (over-sampling the minority class or under-sampling the majority class)
    - Using algorithms that handle class imbalance well (e.g., XGBoost, LightGBM)
    - Applying cost-sensitive learning where higher penalties are given to misclassifications of the minority class.

8. What is the effect of external factors (e.g., number of dashers, outstanding orders) on delivery duration?

- Analyze the impact of market features on delivery time using feature importance scores and statistical tests to determine if these features significantly affect delivery times.

9. How does the performance of models compare on different subsets of data (e.g., different cities)?

- Train and evaluate models separately on data subsets based on `market_id` to see if performance varies across different regions. Use techniques like domain adaptation if necessary.

10. Can the model be deployed and how to monitor its performance in production?

- Implement the trained model in a production environment. Set up monitoring to track model performance metrics over time, detect data drift, and re-train the model periodically with new data.