MPhil in Economics and Data Science

Module: D100/D400

Candidate Number (BGN): 3452B

Deadline Date: 19/12/2024

Actual word count:1882

Motivation

This project is made for the D400-D100 combined final assessment. Specifically, I selected a dataset from Kaggle [link], which is made to predict the food delivery time based on relevant features provided.

There are a couple of reasons for why I chose this dataset for my final project:
Firstly, we are required to apply a generalised linear model, and waiting time analysis is one classic area to apply GLM.
Secondly, I am currently working on a project at work to estimate the processing time for JIRA tasks. It would be quite interesting to see how other dataset is constructed and what available attributes should be collected.

Explanatory Data Analysis

Dataset Overview

Before the in-depth analysis, it is worth mentioning that one of the challenges I face is the lack of explanation and documentation about the data collection process. Hence, the definitions and interpretation of columns are based on online study and my personal understanding, which might not be 100% correct.

The dataset consists of 45593 rows and 20 columns, and rows represent one order. The target variable is in column 'Time_taken(min)', the rest of 19 columns are potential predictors.

The table below shows an overview of the predictor columns, including description, feature type and whether there are missing values:

Column	Description	Type	Has missing
ID	Identification of a data entry	Identifier	N
Delivery_person_ID	Identification of a delivery person	Identifier	N
Delivery_person_Age	Age of a delivery person	Numerical	Y
Delivery_person_R	Ratings of a	Numerical	Y

atings	delivery person		
Restaurant_latitude	Latitude of the restaurant	Numerical	N
Restaurant_longitude	Longitude of the restaurant	Numerical	N
Delivery_location_latitude	Latitude of the delivery location	Numerical	N
Delivery_location_longitude	Longitude of the delivery location	Numerical	N
Order_Date	Date when the order was placed	Temporal	N
Time_orderd	Time when the order was placed	Temporal	Y
Time_Order_picked	Time when the order was picked-up	Temporal	N
Weatherconditions	Weather conditions for the day of the delivery	Categorical	Y
Road_traffic_density	The density of the road during the time of delivery	Categorical	Y
Vehicle_condition	Condition of the delivery vehicle	Categorical	N
Type_of_order	Type of cuisine	Categorical	N
Type_of_vehicle	Type of delivery vehicle	Categorical	N
multiple_deliveries	Number of orders to be delivered in one attempt	Categorical	Y
Festival	Whether the day is a festival	Categorical	Y
City	Condition of the city	Categorical	Y

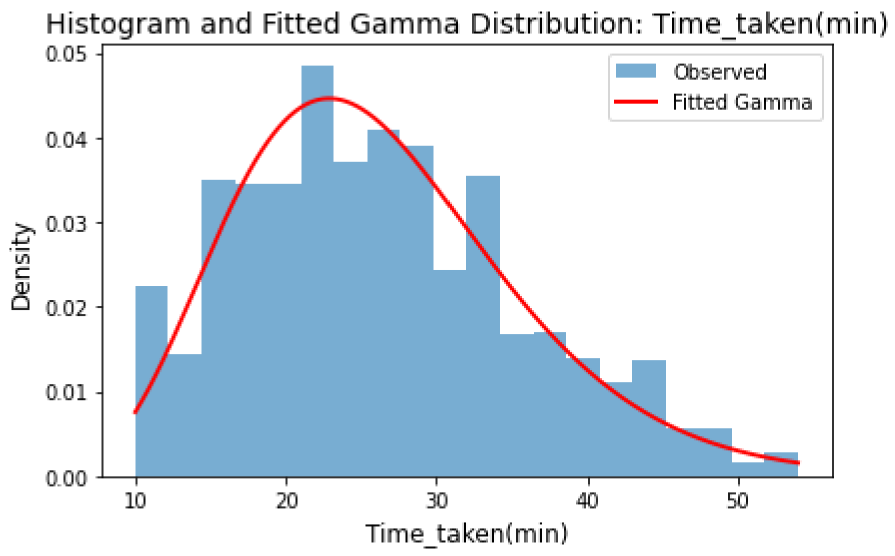
We can see that some features are not fit-for-purpose for modeling directly, hence further feature engineering and transformation would be required, which will be explained in detail in the following sessions.

The raw data was downloaded from the Kaggle webpage and stored as data.csv file under the data/raw folder.

Feature Analysis

Target Variable

Time_taken(min) is a continuous numerical variable ranging from 10 to 54, which is the target of prediction in this project. From its density histogram, we can observe it is a positive, right-skewed distribution, which fits closely with the Gamma distribution. Hence, I choose Gamma distribution in my GLM pipeline.



Numerical Predictors

- ID and Delivery_person_ID

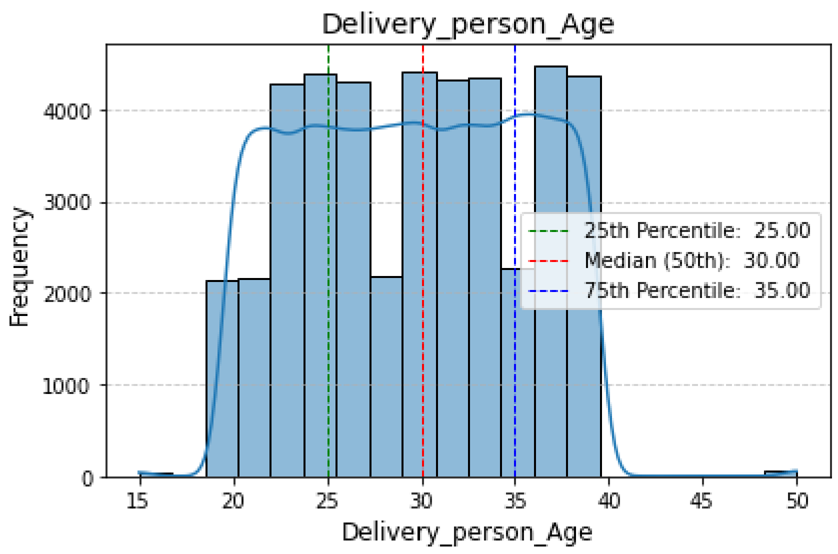
These 2 columns are supposed to serve as unique identification for delivery order and delivery personnel. However, during clean-up, I noticed there are data inconsistency issues for Delvery_person_ID, i.e. for the same delivery personnel, there are several ages for the same person with the same year.

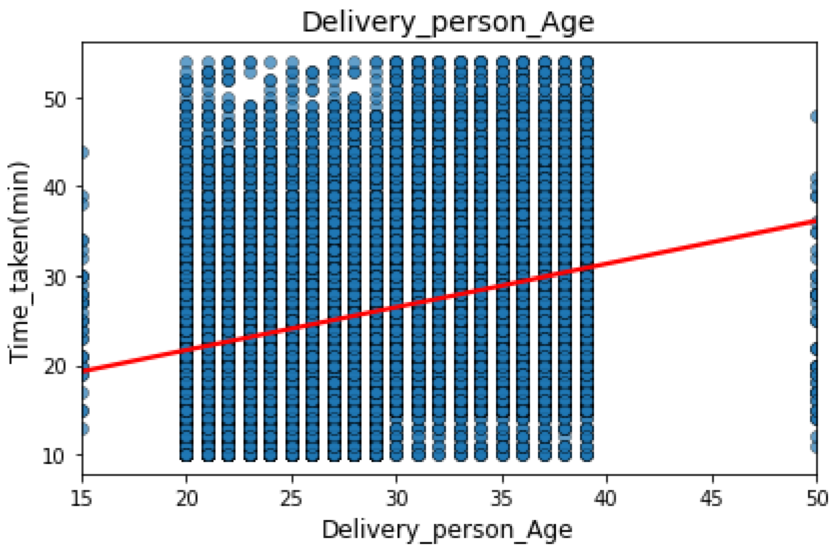
As the data collection process and definition are not clearly stated in the Kaggle website, I chose to drop those columns.

- Delivery_person_Age

This is a numerical feature which roughly follows normal distribution. The delivery person’s age ranges from 15 to 50, with no significant outlier (the data is collected for the Indian food delivery market, where the legal age eligible for work is 14). However, missing values would need to be filled by median.

There is a very weak positive trend between this feature and target variable.

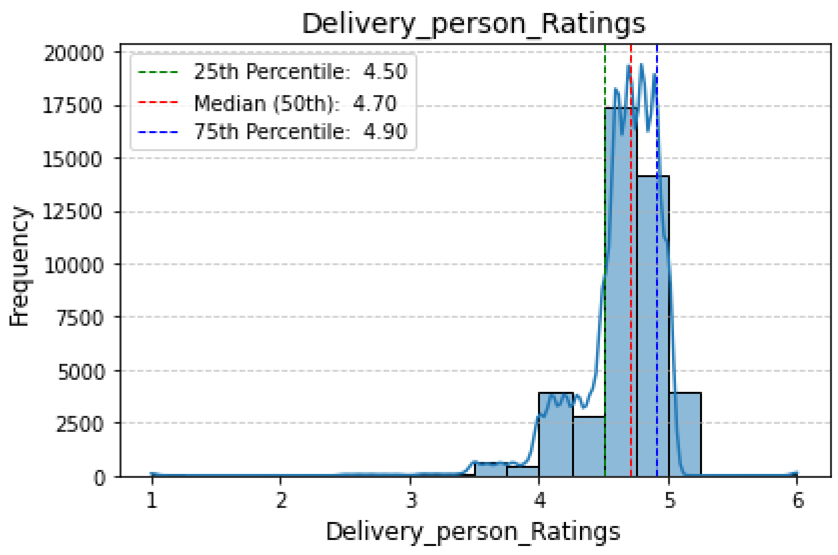


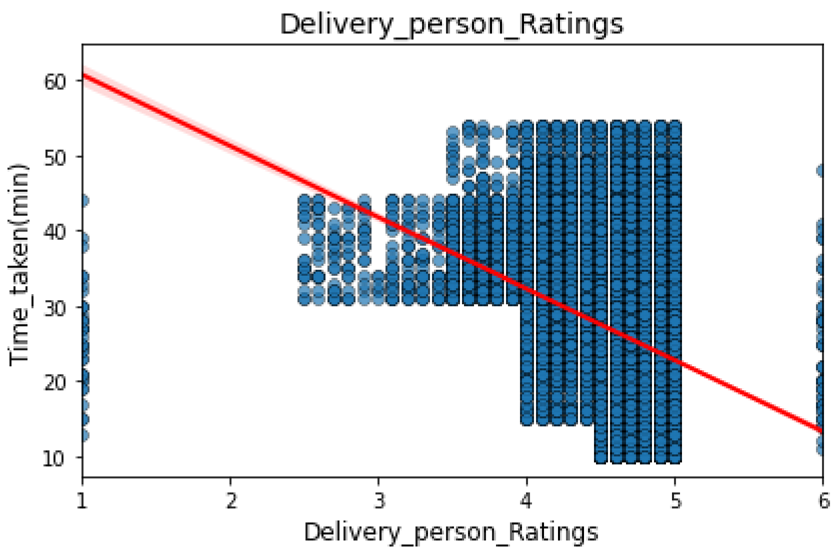


- Delivery_person_Ratings

This is a numerical variable ranging from 1 to 6, with a heavily left-skewed distribution. Hence using median to impute missing values would be a better strategy.
Due to its left-skewed nature, log-transformation is required for GLM.

We can observe a strong negative correlation between this feature and target variable



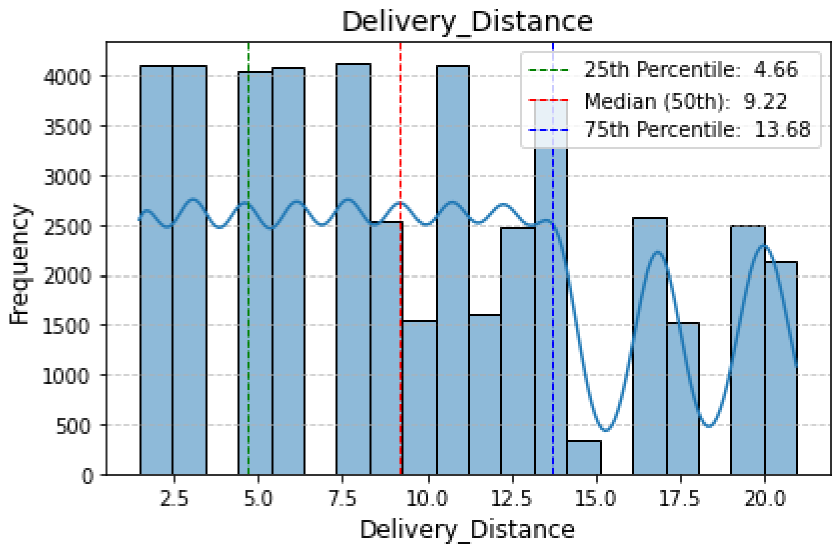


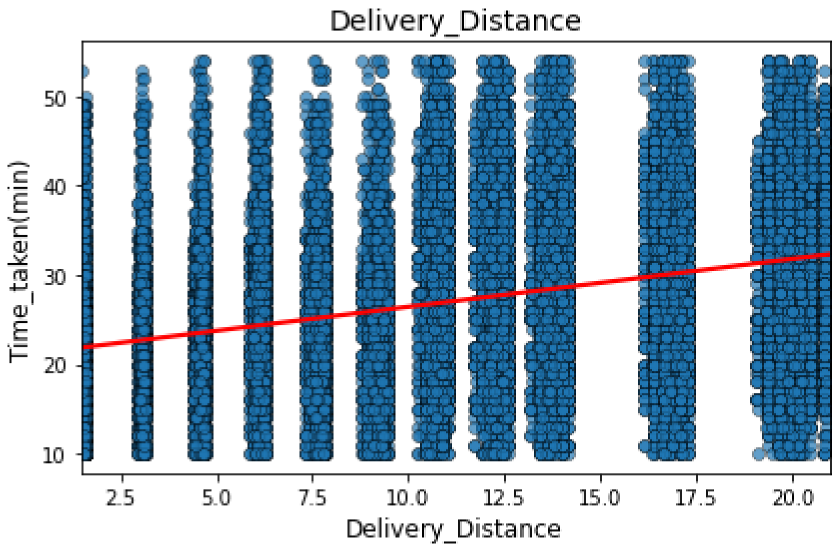
- Latitude and Longitude (Delivery_Distance)

Coordinates for restaurants and delivery locations are provided, however, these values alone won't be fit-for-purpose for the prediction model. Hence, I am doing feature engineering to calculate the distance between restaurant and delivery location for each order for a new feature **Delivery_Distance**.

Before the feature engineering, outlier inspection was done via visualisation. The dataset was based on the Indian food delivery market, so we can identify a couple of data entities with negative longitude and latitude, which are highly likely to be data entry errors, and could be fixed by flipping the sign.

After addressing the outliers, a new feature Delivery_Distance was generated, which is positively correlated with the target variable.



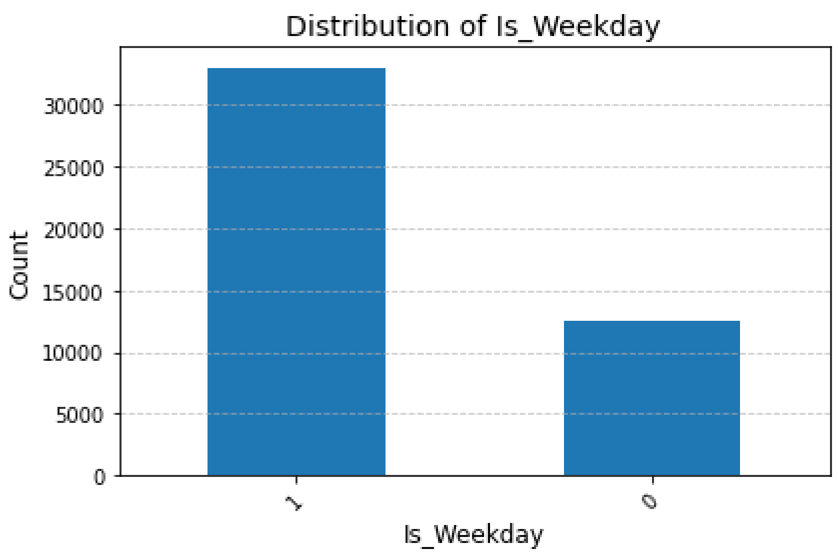


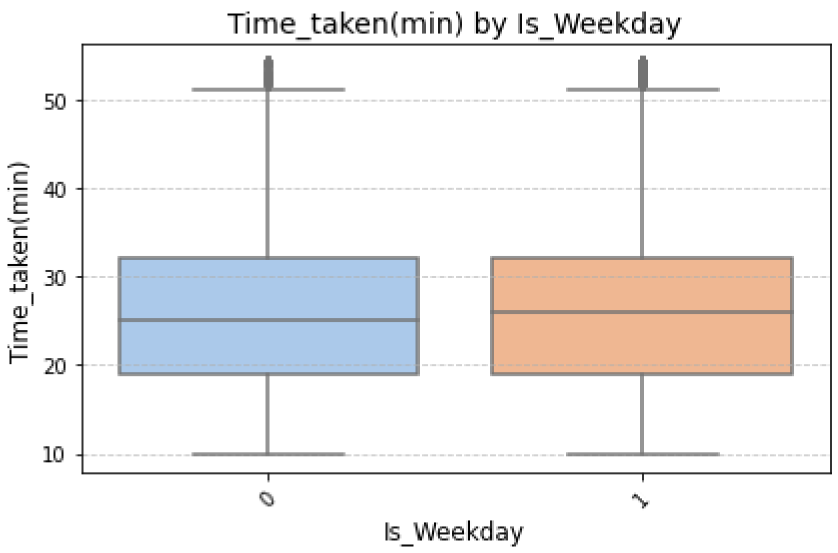
* As these numerical features have various ranges, StandardScaler would need to be applied in the GLM pipeline.

Categorical Predictors

- Order_Date (Is_Weekday)

Since the date of order itself would not provide useful insight for the prediction, I derived a new binary feature to indicate whether the order was made on weekdays or weekends.

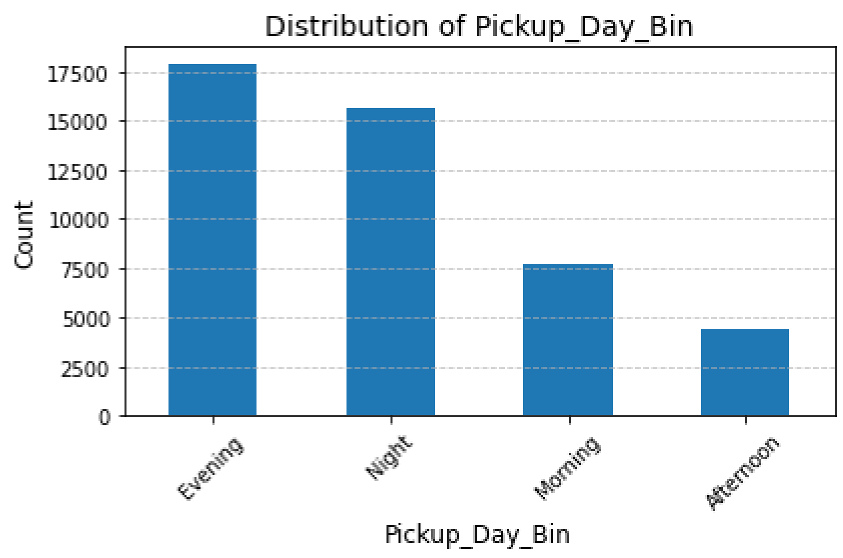


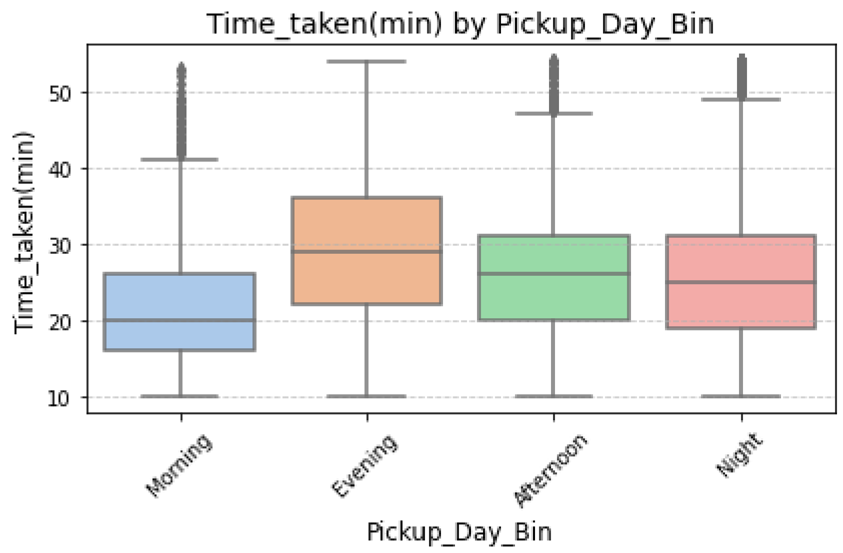


- Time_ordered and Time_order_picked (Pickup_Day_Bin)

Similarly, the timestamp itself would not be directly useful as a feature. A new categorical feature (**Pickup_Day_Bin**) was generated from the order pickup time to indicate whether it was morning, afternoon, evening or night.

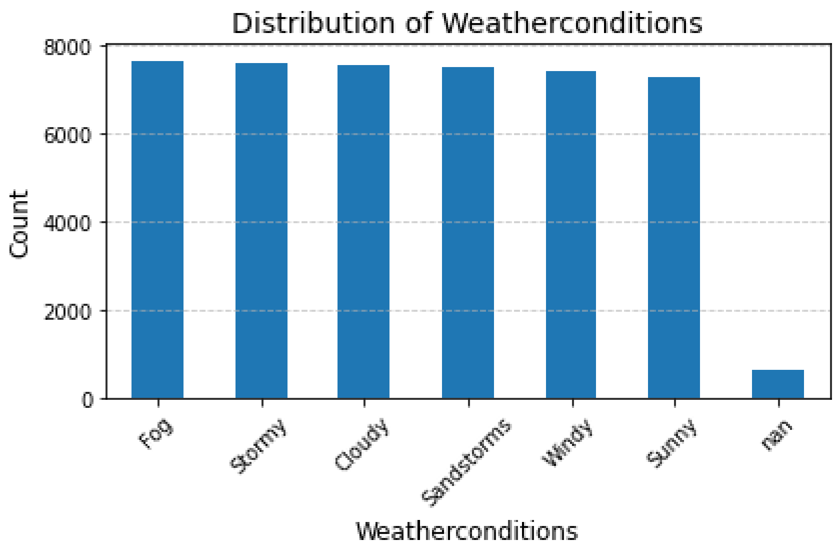
We can see that orders made in the evening normally take longer and have larger variance.

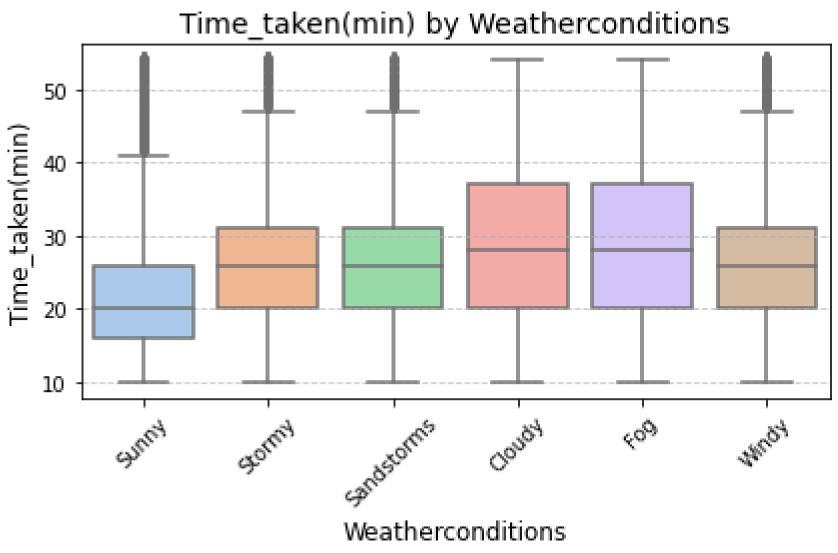




- Weatherconditions

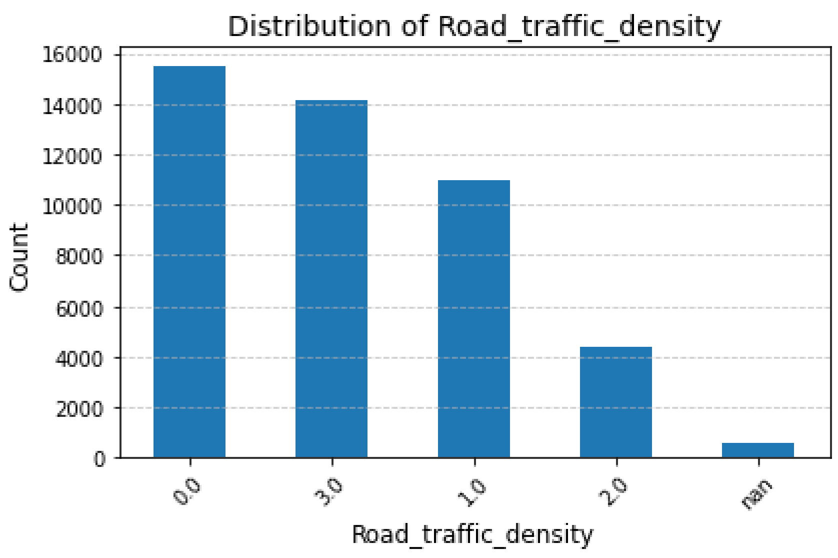
This categorical variable contains a small proportion of missing values which would need to be filled in using mode.
From the boxplot, we can observe delivery time is taking longer and has a larger range under cloudy and foggy weather.

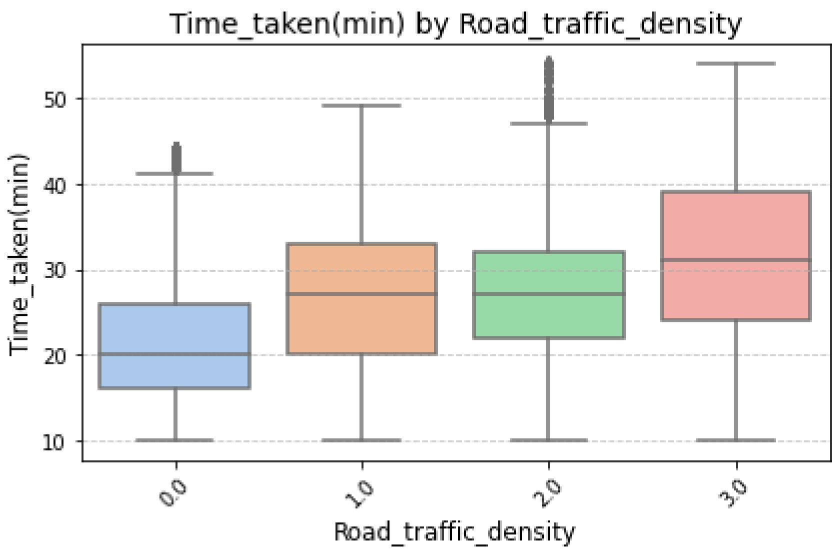




- Road_traffic_density

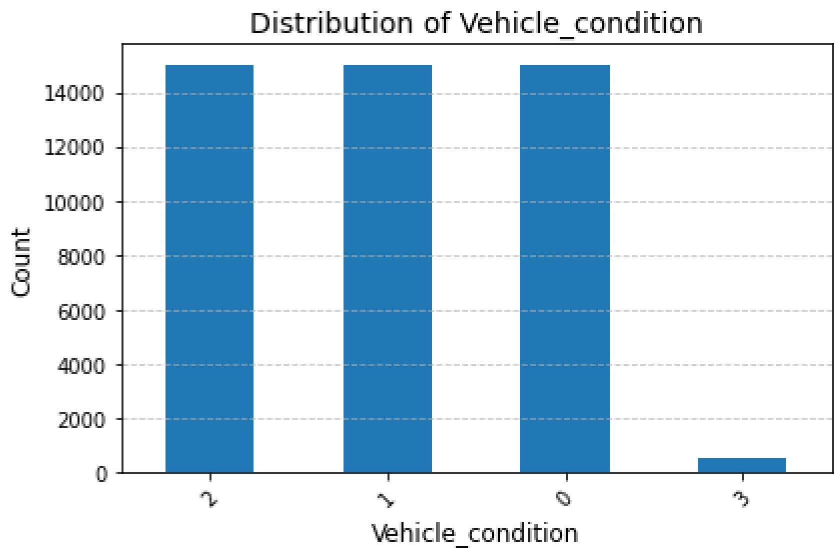
The original values are in string, for the ease of modeling, I convert them to ordinal values with 0 representing low traffic flow while 3 represents heavy traffic jam. The statistical summary in the box plot also aligns with intuition as heavy traffic would lead to longer delivery time on average.

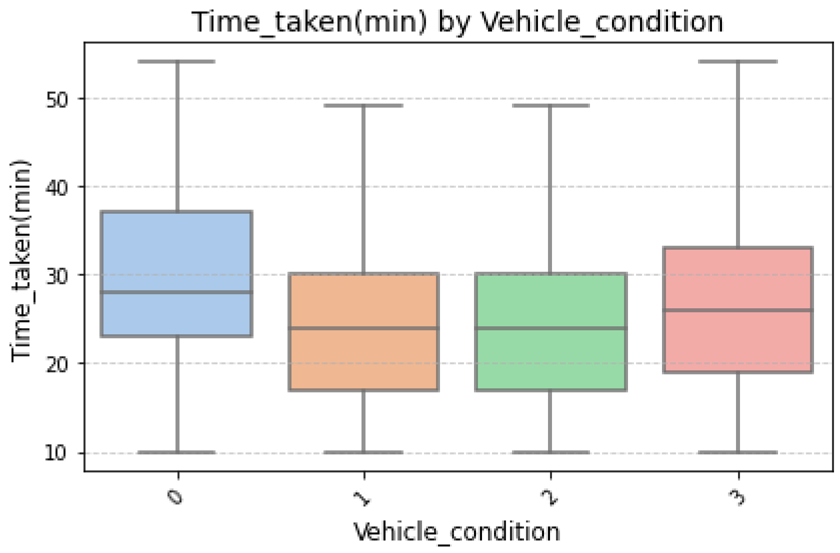




- Vehicle_Condition

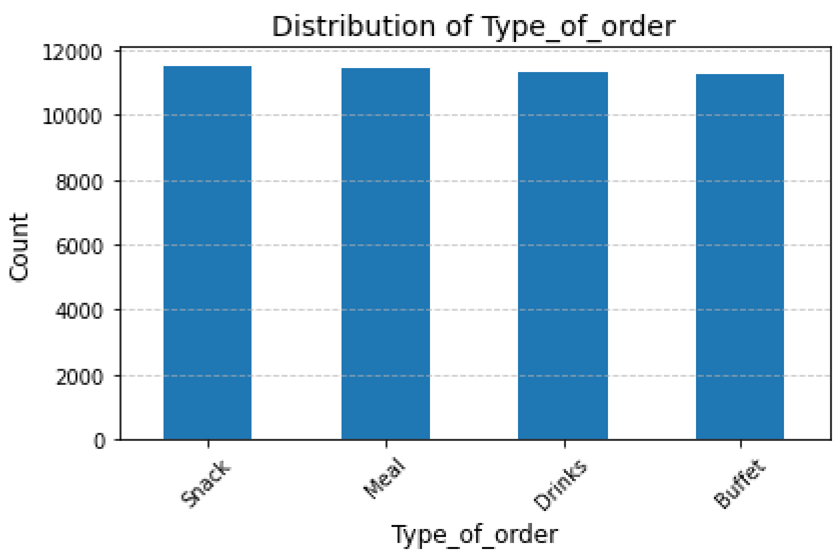
This is an ordinal variable as 0 indicates very poor vehicle quality. Poor quality of vehicles leads to longer average delivery time.

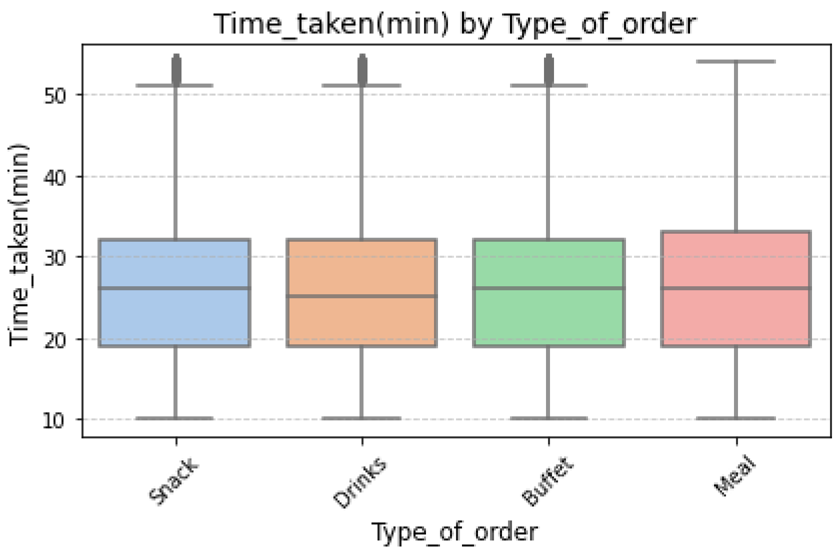




- Type_of_order

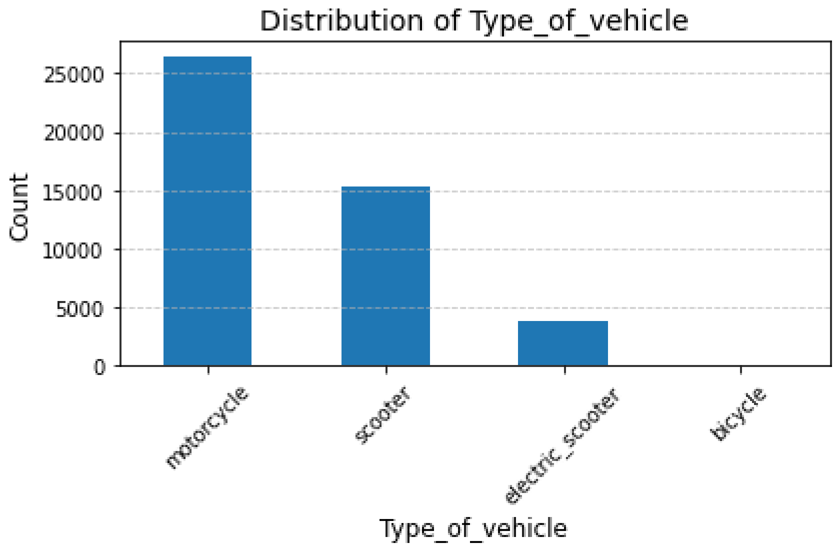
The average delivery time and IQR are not showing significant difference under each cuisine type, hence very likely this feature would have a limiting impact on the prediction.

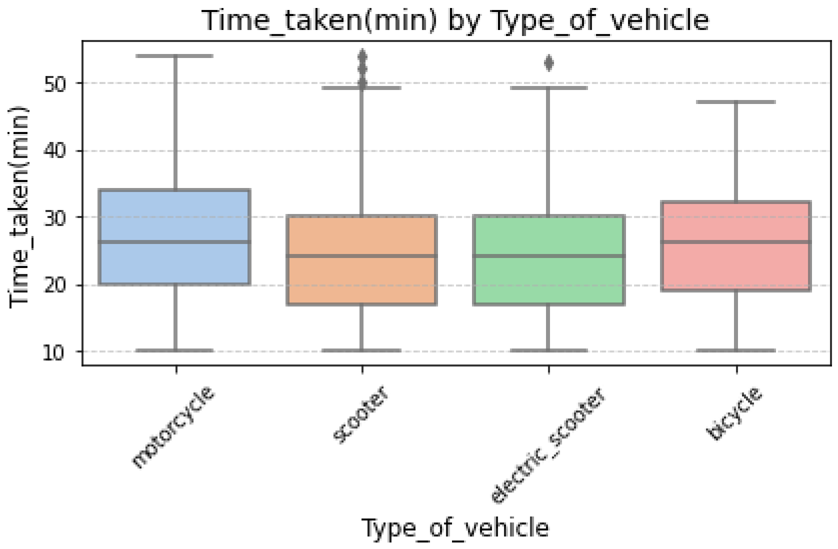




- Type_of_vehicle

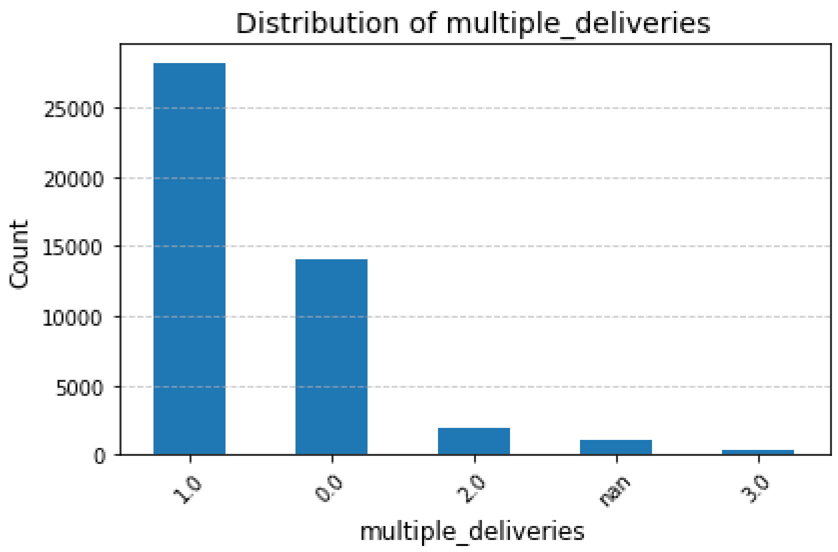
Majority of the orders were made by motorcycles, while bicycles are very rare.

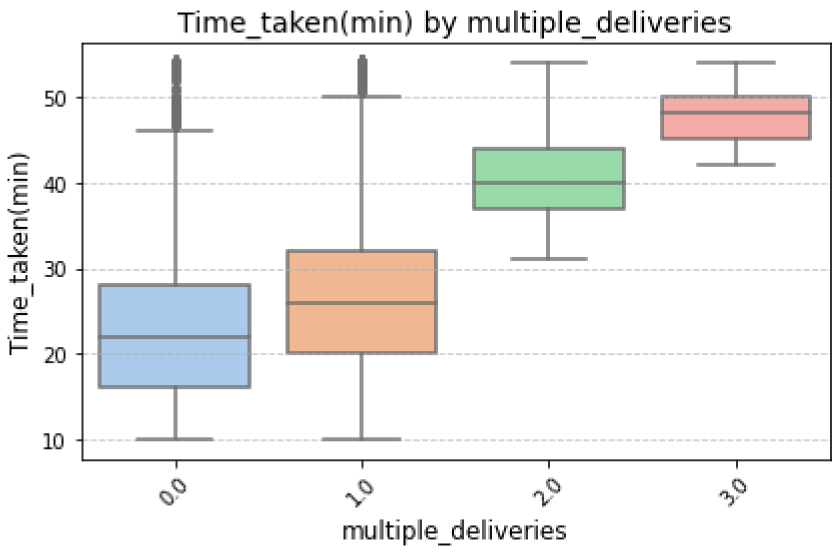




- multiple_deliveries

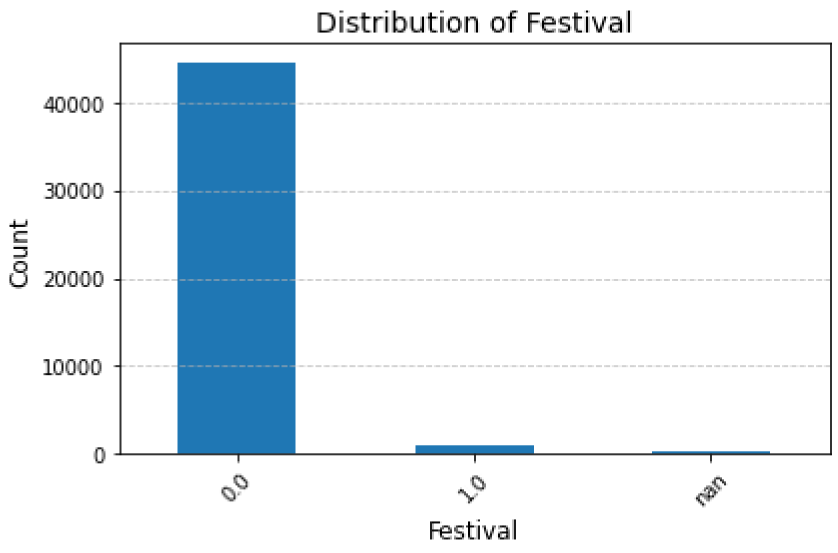
This is by far the feature showing the most significant difference in delivery time under each value and is strongly correlated with the target variable.

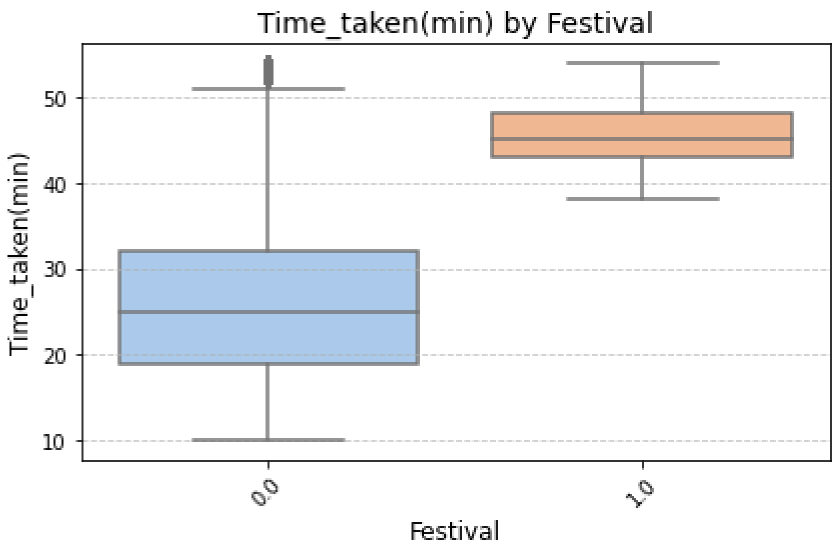




- Festival

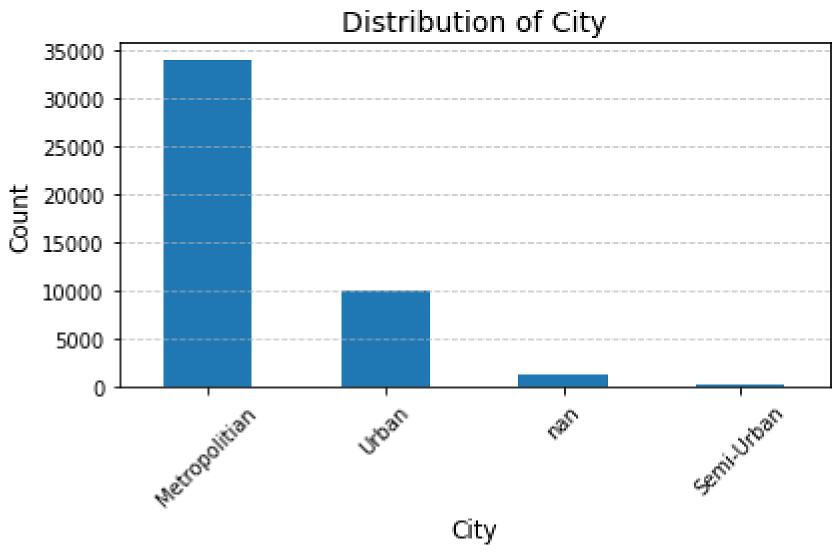
Deliveries made during the festival were significantly taking longer to deliver.

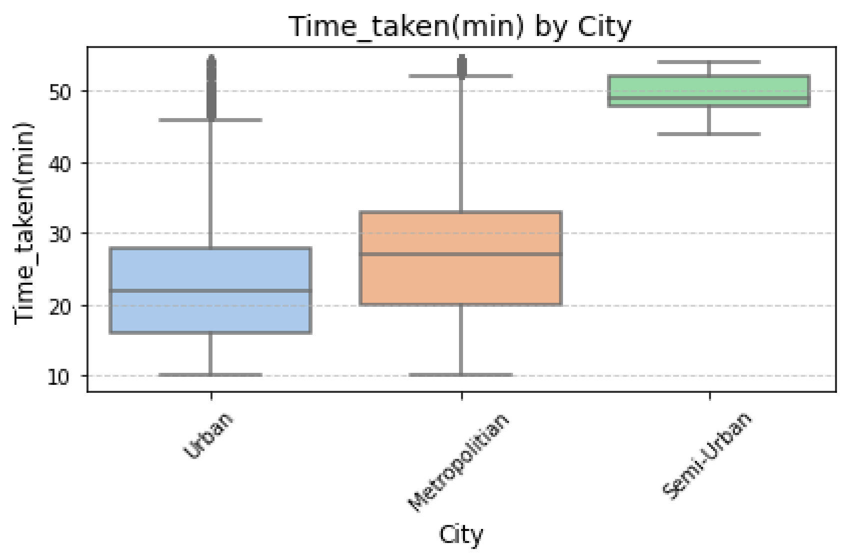




- City

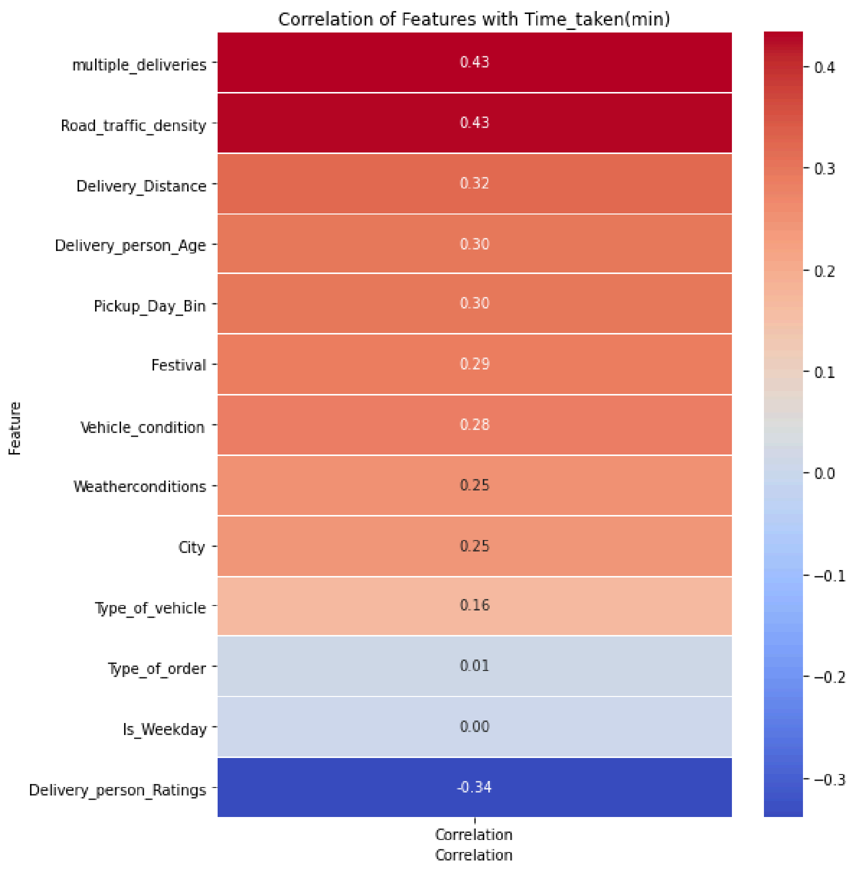
The dominant class is metropolitan.





Correlation

From the heatmap, we can see some features with very strong correlation with the target variables. For features with very low correlation, I am still using them for modeling as there might be interaction effects that can be captured by LGBM.



Cleaning Steps Summary

- Clean up the trailing white space in the text across the entire dataset
- Extract useful values from Time_taken(min) and Weatherconditions columns using regex
- Replace string representations of missing values in the dataframe into np.nan
- Convert specific columns to numeric type
- Convert Order_Date to datetime, and Time_orderd and Time_order_picked to timestamp
- Make sure all values of longitude and latitude are positive
- Impute missing values (this is done in the modeling stage after the train-test split)

Modelling

Model Training

Taking the prepared dataset as the input, train-test split (from sklearn) was performed to separate the dataset into train set (80%) and test set (20%).

Then, I built separate pipelines to handle the imputation, transformation and modeling for GLM and LGBM:

Feature	Missing Value Imputation Strategy	GLM Transformation	LGBM Transformation
Delivery_person_Age	Median	StandardScaler	Not needed
Delivery_person_Ratings	Median	StandardScaler	Not needed
Delivery_Distance	No Missing	Log Transformation StandardScaler	Not needed
Is_Weekday	No Missing	Not needed (already binary)	Not needed
Pickup_Day_Bin	No Missing	One-hot Encoder	Ordinal Encoder
Weatherconditions	Mode	One-hot Encoder	Ordinal Encoder
Road_traffic_density	Mode	Not needed (already ordinal)	Not needed
Vehicle_condition	No Missing	Not needed (already ordinal)	Not needed

Type_of_order	No Missing	One-hot Encoder	Ordinal Encoder
Type_of_vehicle	No Missing	One-hot Encoder	Ordinal Encoder
multiple_deliveries	Mode	Not needed (already ordinal)	Not needed
Festival	Mode	Not needed (already binary)	Not needed
City	Mode	One-hot Encoder	Ordinal Encoder

Hyperparameter Tuning

The baseline model is GLM using default parameters. In order to improve model performance, I fine-tuned hyperparameters using GridSearch.

It is worth mentioning that I am using GammaRegressor from sklearn to perform GLM modeling [Link], and it does not include l1_ratio as parameter as it only supports L2 regularisation.

For GLM, I used GridSearch to find the optimal alpha over set [0.01, 0.1, 1.0, 10.0, 100.0], and the selected alpha is 0.01 (default 1.0)

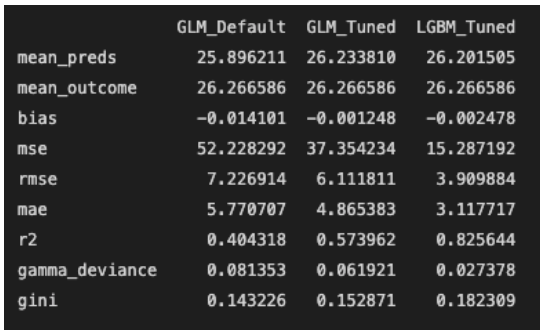
For LGBM, GridSearch is also applied to find the optimal learning_rate (among [0.01, 0.05, 0.1], num_leaves (among [20, 31, 50]) and min_child_weight (among [1,3,5]). The selected optimal combination is learning_rate = 0.01, min_child_weight = 1 and num_leaves = 50.

Model Evaluation and Performance Comparison

For the evaluation metrics, I am mainly using mean square error, R square scores and Gamma Deviance:.

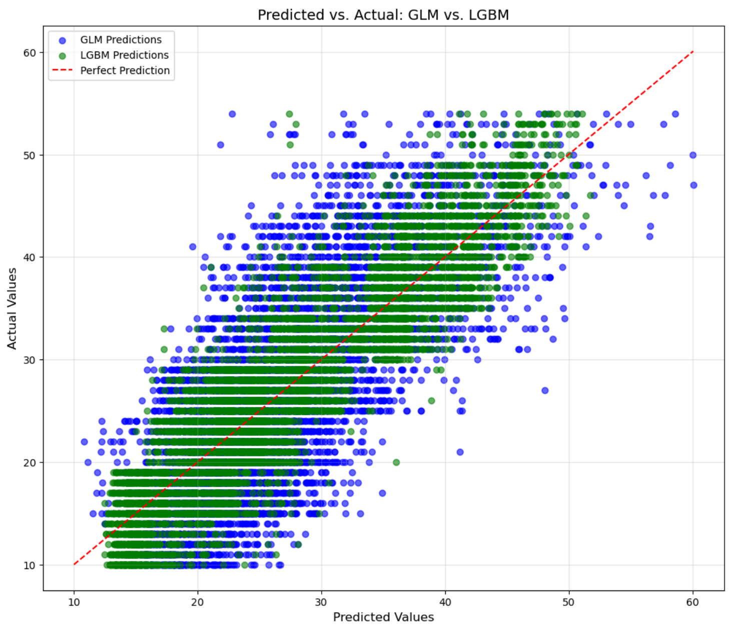
- MSE: lower value suggests more accurate predictions
- R2: higher value suggests the model explains a larger proportion of the variance in the target variable
- Gamma Deviance: lower deviance suggests more accurate predictions

Other evaluation metrics including RMSE, MAE and GINI are also provided:



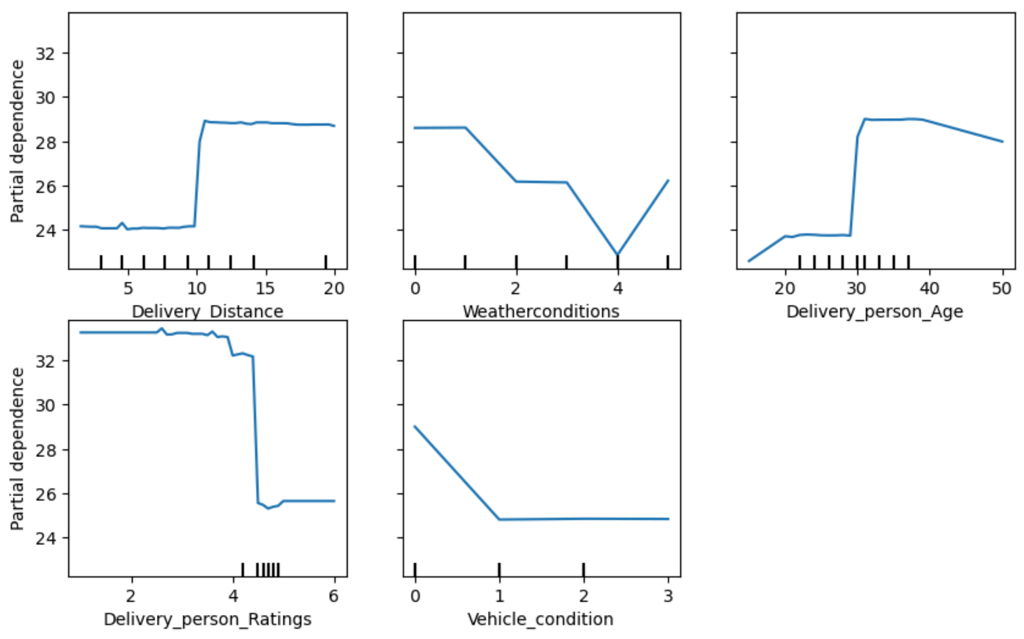
```
              GLM_Default  GLM_Tuned  LGBM_Tuned
mean_preds      25.896211  26.233810   26.201505
mean_outcome    26.266586  26.266586   26.266586
bias            -0.014101  -0.001248   -0.002478
mse             52.228292  37.354234   15.287192
rmse             7.226914   6.111811    3.909884
mae              5.770707   4.865383    3.117717
r2               0.404318   0.573962    0.825644
gamma_deviance   0.081353   0.061921    0.027378
gini             0.143226   0.152871    0.182309
```

To further compare the outcome, the predictions generated from GLM tuned and LGBM tuned were plotted in the same scatter graph. We can see the predictions produced by LGBM tuned are much closer to the actual values.



Hence, it is clear to draw the conclusion that LGBM after hyperparameter tuning performs significantly better.

In order to showcase why LGBM performs better, I have generated the partial dependence plots for the top 5 important features.



The PDP graphs show clear nonlinear patterns for various features, and LGBM, which is a tree-based model, is more superior at handling non-linear relationships.

In addition, LGBM is also better at handling categorical variables and capturing feature interactions, which makes it a better choice for this specific dataset.

Potential Enhancement

Additional Data Collection

Current features such as traffic density and weather are based on daily summary, which are coarse approximations. As food deliveries are made across the day, I would like to get access to more accurate and time-sensitive data in terms of the traffic and weather conditions within the hour of delivery.

I would also get more exposure to numerical features, for example, we are using categorical data for weather conditions. The delivery time is taking much longer and has a large variation under cloudy and foggy days, instead of using categories, I would like to transfer these to numerical features, such as wind speed per hour, road visibility etc.

Additional Feature Engineering

When assessing the relative low accuracy on GLM, we briefly mentioned its weakness in capturing interaction effects. In this project, a lot of features might have interactions, for example, vehicle condition * weather conditions. I think it would be an interesting experiment to engineer a few more features for interaction effects and test its impact on the GLM.

Disclaimer

ChatGPT was leveraged in the project to improve clarity, enhance self-learning and assist the research process. All outputs were carefully reviewed, validated, and adapted to ensure accuracy, relevance, and alignment with project objectives.