Linear Regression Assignment: Parcel Delivery Time Estimation

- Avanindra Singh

Objectives

The objective of this assignment is to build a regression model that predicts the delivery time for orders placed through Porter. The model will use various features such as the items ordered, the restaurant location, the order protocol, and the availability of delivery partners.

The key goals are:

- Predict the delivery time for an order based on multiple input features
- Improve delivery time predictions to optimize operational efficiency
- Understand the key factors influencing delivery time to enhance the model's accuracy

1. Load the data

The data is loaded using Pandas dataframe and has following columns:

```
pdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 14 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   market_id                 175777 non-null  float64
 1   created_at                175777 non-null  object 
 2   actual_delivery_time      175777 non-null  object 
 3   store_primary_category    175777 non-null  int64  
 4   order_protocol            175777 non-null  float64
 5   total_items               175777 non-null  int64  
 6   subtotal                  175777 non-null  int64  
 7   num_distinct_items        175777 non-null  int64  
 8   min_item_price            175777 non-null  int64  
 9   max_item_price            175777 non-null  int64  
 10  total_onshift_dashers     175777 non-null  float64
 11  total_busy_dashers        175777 non-null  float64
 12  total_outstanding_orders  175777 non-null  float64
 13  distance                  175777 non-null  float64
dtypes: float64(6), int64(6), object(2)
memory usage: 18.8+ MB
```

2. Data Preprocessing and Feature Engineering

2.1 Fixing Datatypes

The current timestamps are in columns ‘created_at’ and ‘actual_delivery_time’ are in object format and need conversion to datetime format for easier handling and intended functionality.

Pandas to_datetime() function is used to convert the data type from object to datetime64.

```
pdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 14 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   market_id                 175777 non-null  float64
 1   created_at                175777 non-null  datetime64[ns]
 2   actual_delivery_time      175777 non-null  datetime64[ns]
 3   store_primary_category    175777 non-null  int64
 4   order_protocol            175777 non-null  float64
 5   total_items               175777 non-null  int64
 6   subtotal                  175777 non-null  int64
 7   num_distinct_items        175777 non-null  int64
 8   min_item_price            175777 non-null  int64
 9   max_item_price            175777 non-null  int64
 10  total_onshift_dashers     175777 non-null  float64
 11  total_busy_dashers        175777 non-null  float64
 12  total_outstanding_orders  175777 non-null  float64
 13  distance                  175777 non-null  float64
dtypes: datetime64[ns](2), float64(6), int64(6)
memory usage: 18.8 MB
```

Also, the categorical features ‘market_id’ and ‘order_protocol’ are converted from float to int.

```
pdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 14 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   market_id                 175777 non-null  int64
 1   created_at                175777 non-null  datetime64[ns]
 2   actual_delivery_time      175777 non-null  datetime64[ns]
 3   store_primary_category    175777 non-null  int64
 4   order_protocol            175777 non-null  int64
 5   total_items               175777 non-null  int64
 6   subtotal                  175777 non-null  int64
 7   num_distinct_items        175777 non-null  int64
 8   min_item_price            175777 non-null  int64
 9   max_item_price            175777 non-null  int64
 10  total_onshift_dashers     175777 non-null  float64
 11  total_busy_dashers        175777 non-null  float64
 12  total_outstanding_orders  175777 non-null  float64
 13  distance                  175777 non-null  float64
dtypes: datetime64[ns](2), float64(4), int64(8)
memory usage: 18.8 MB
```

2.2 Feature Engineering

Here we can calculate the time taken to execute the delivery as well as extract the hour and day at which the order was placed. Finally we can drop the unnecessary columns.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 16 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   market_id                 175777 non-null  int64
 1   store_primary_category    175777 non-null  int64
 2   order_protocol            175777 non-null  int64
 3   total_items               175777 non-null  int64
 4   subtotal                  175777 non-null  int64
 5   num_distinct_items        175777 non-null  int64
 6   min_item_price            175777 non-null  int64
 7   max_item_price            175777 non-null  int64
 8   total_onshift_dashers     175777 non-null  float64
 9   total_busy_dashers        175777 non-null  float64
 10  total_outstanding_orders  175777 non-null  float64
 11  distance                  175777 non-null  float64
 12  time_taken                175777 non-null  float64
 13  delivery_hr               175777 non-null  int32
 14  delivery_day              175777 non-null  int32
 15  is_weekend                175777 non-null  int64
dtypes: float64(5), int32(2), int64(9)
memory usage: 20.1 MB
```

2.3 Train-Validation Split

Using Sklearn, we can split the data into training and test sets:

```
train_data, test_data = train_test_split(pdata, train_size=0.7, test_size=0.3, random_state=100)
```

```
print("Training Data: ", train_data.shape)
print("Test Data: ", test_data.shape)
```

```
Training Data:  (123043, 16)
Test Data:  (52734, 16)
```

The data is split into ratio 7:3

3. Exploratory Data Analysis on Training Data

In EDA we analyzed the correlation between variables to identify patterns and relationships and identifying and addressing outliers to ensure the integrity of the analysis.

3.1 Feature Distribution

Here we defined numerical and categorical columns for easy EDA and data manipulation.

Numerical Data:

- total_items
- num_distinct_items
- subtotal
- min_item_price

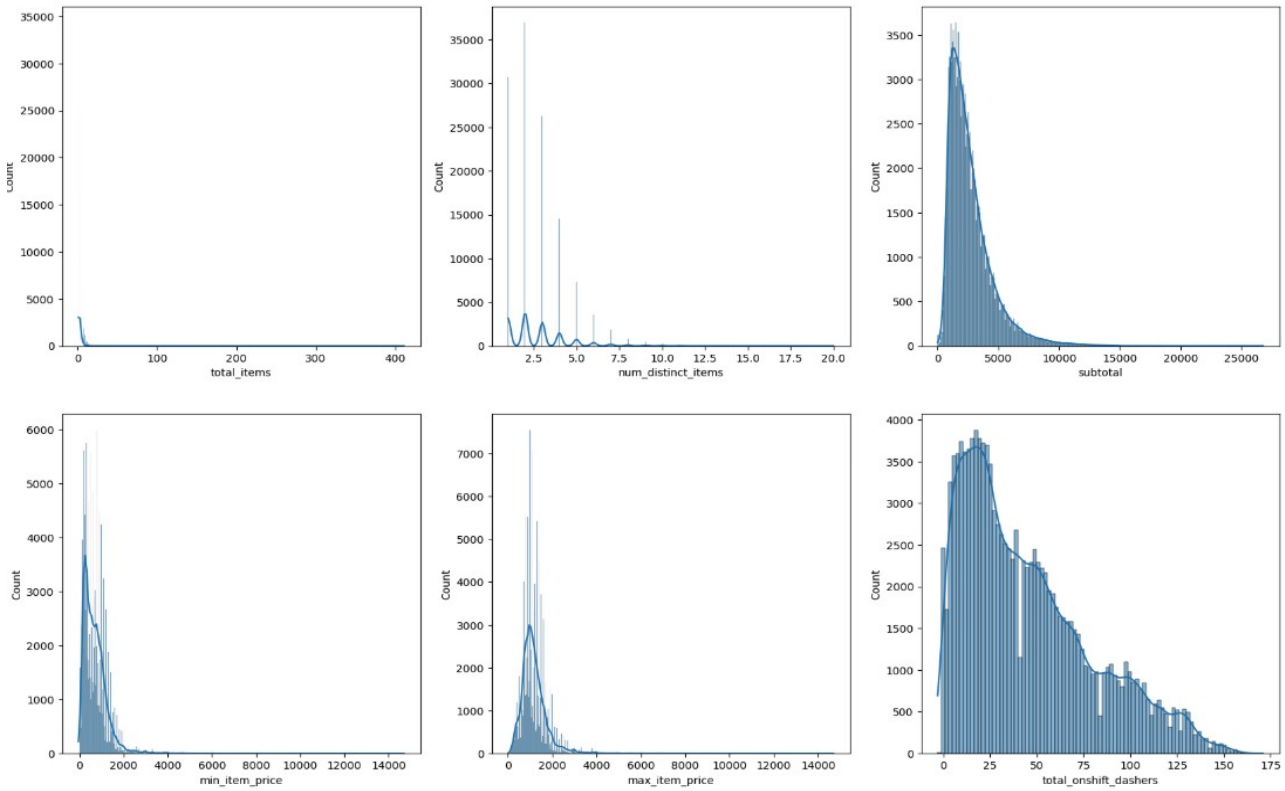
- max_item_price
- total_onshift_dashers
- total_busy_dashers
- total_outstanding_orders
- distance
- time_taken

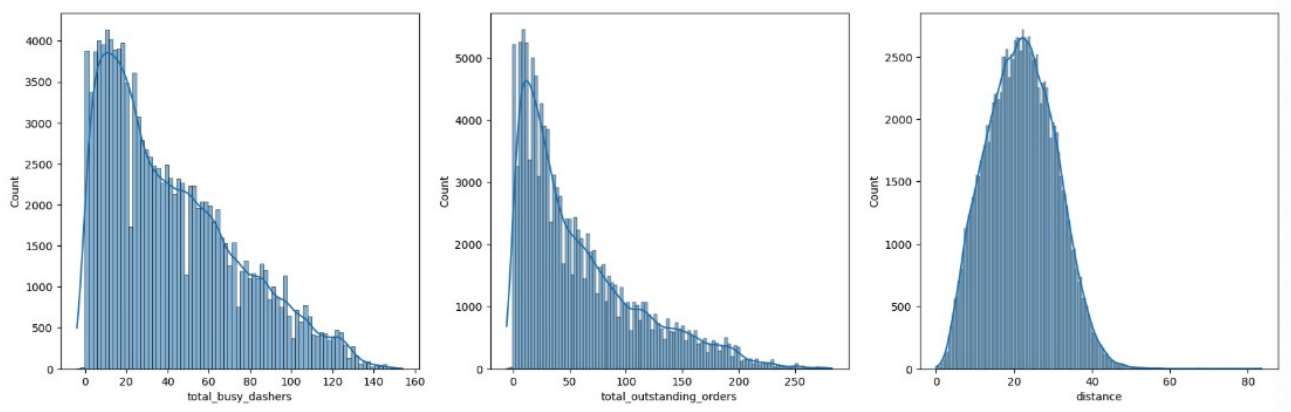
Categorical/Discreet Data:

- market_id
- store_primary_category
- order_protocol
- is_weekend
- delivery_hr
- delivery_day

i. Distribution of numerical features

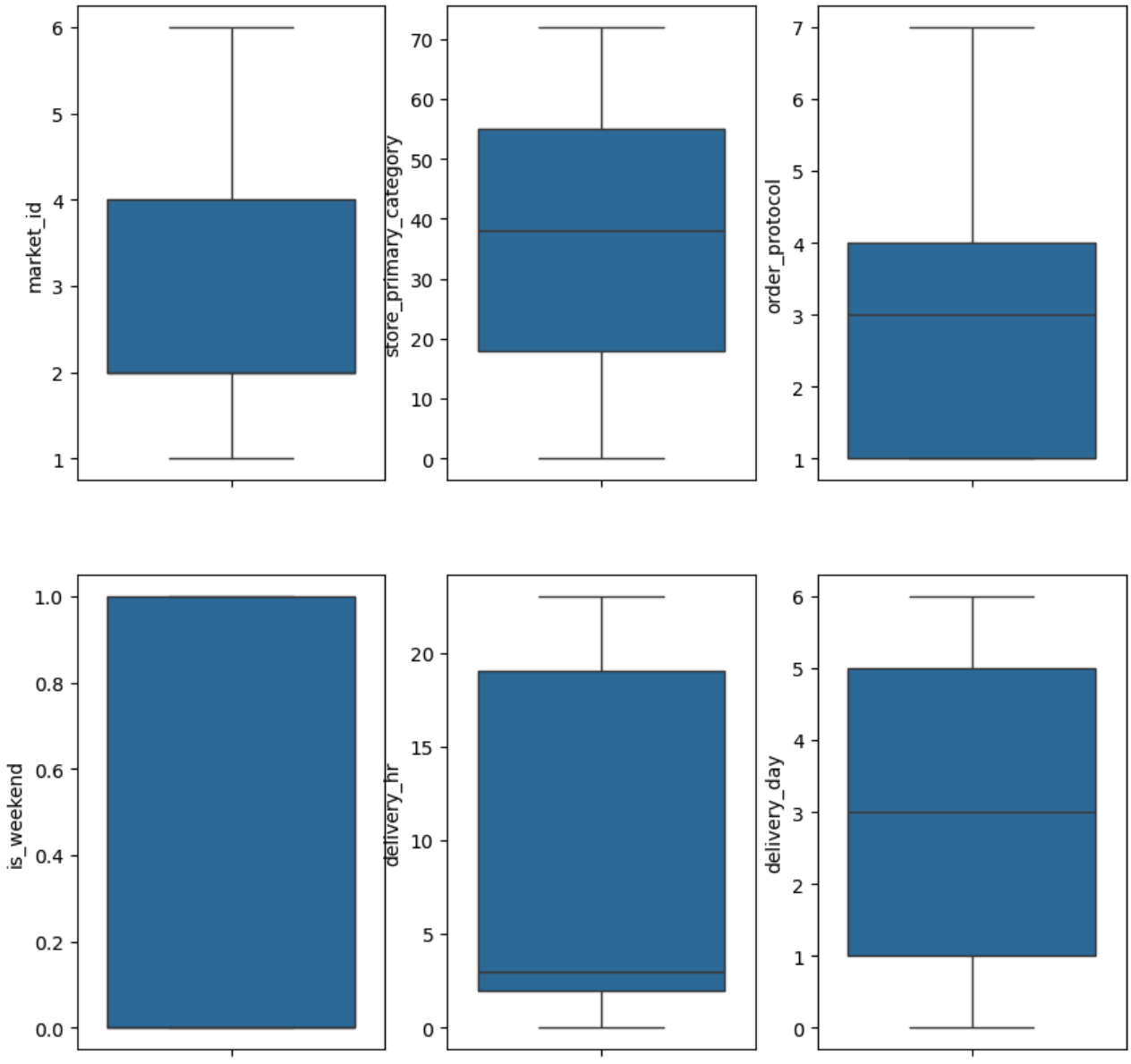
Here we plotted distributions for numerical columns in the training set to understand their spread and any skewness:





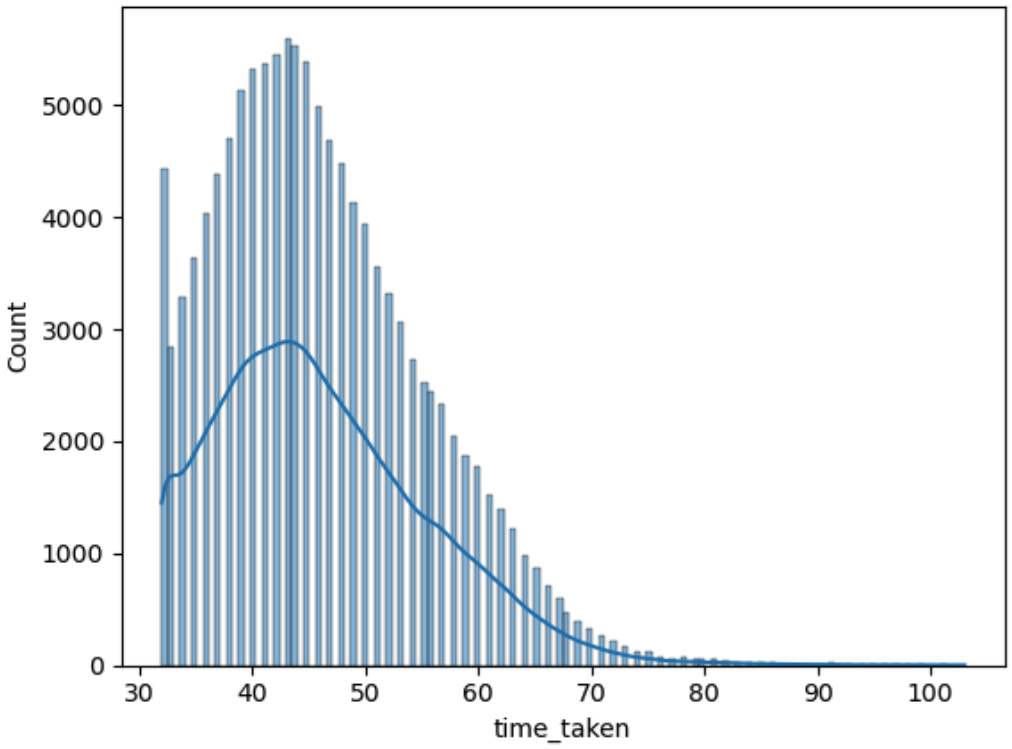
There is slight skewness in data. However, the distance is normally distributed.

ii. Distribution of categorical features



iii. Distribution of Target feature

Here we plot the distribution of the target variable '`time_taken`' to understand its spread and skewness.

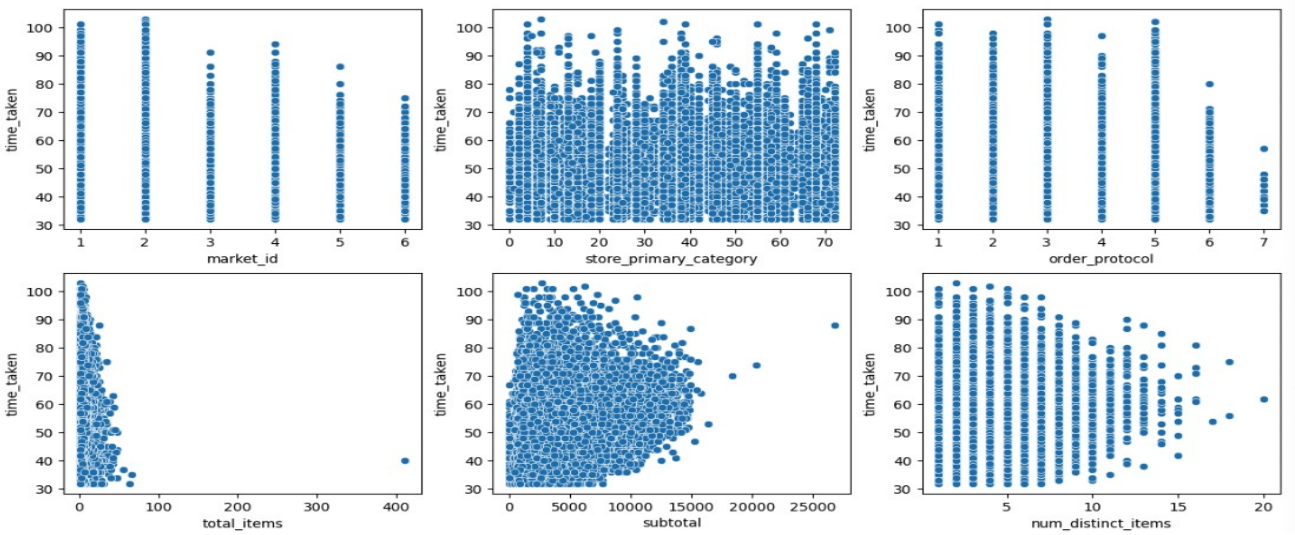


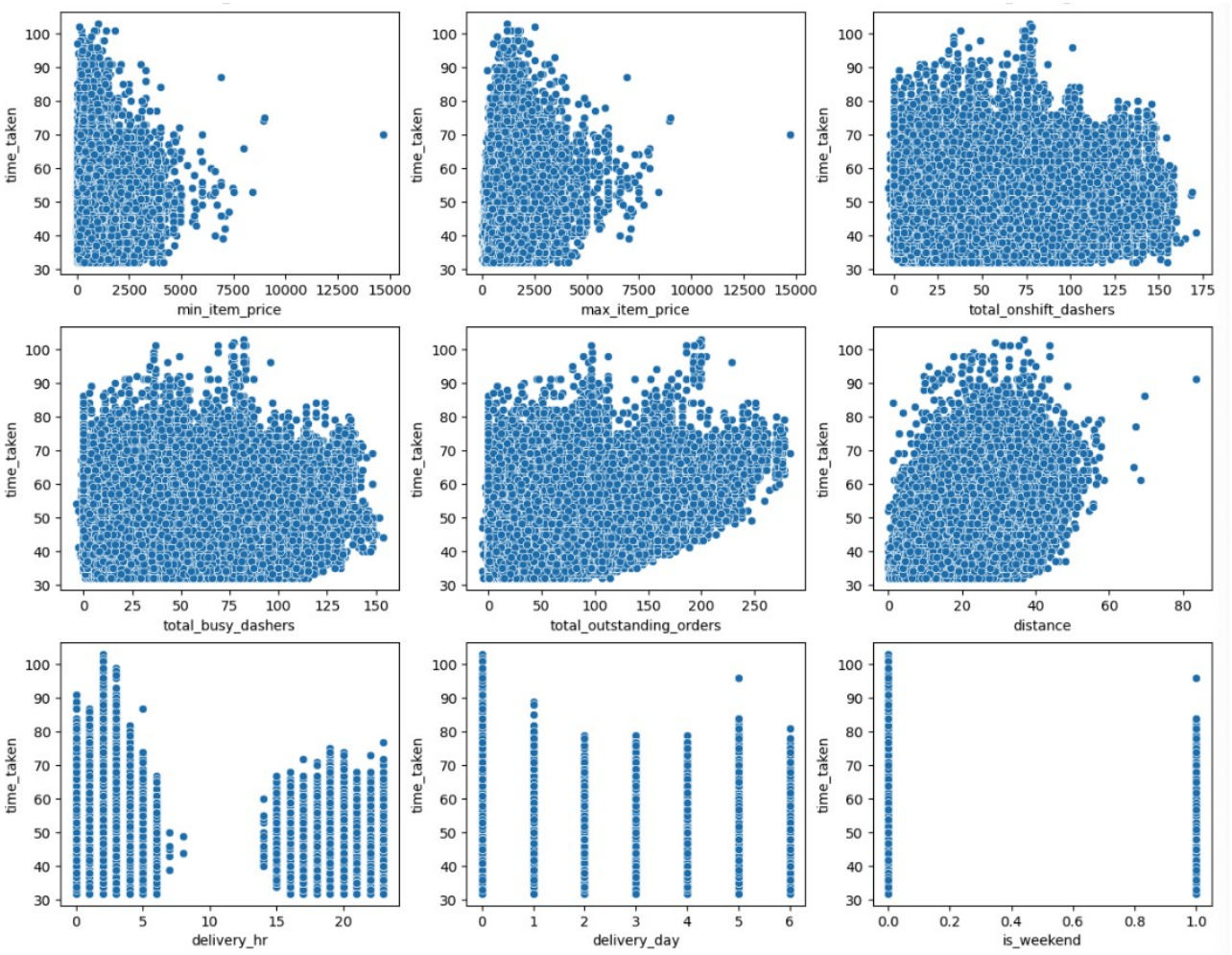
Here, we observe slight skewness in the distribution.

3.2 Relationships Between Features

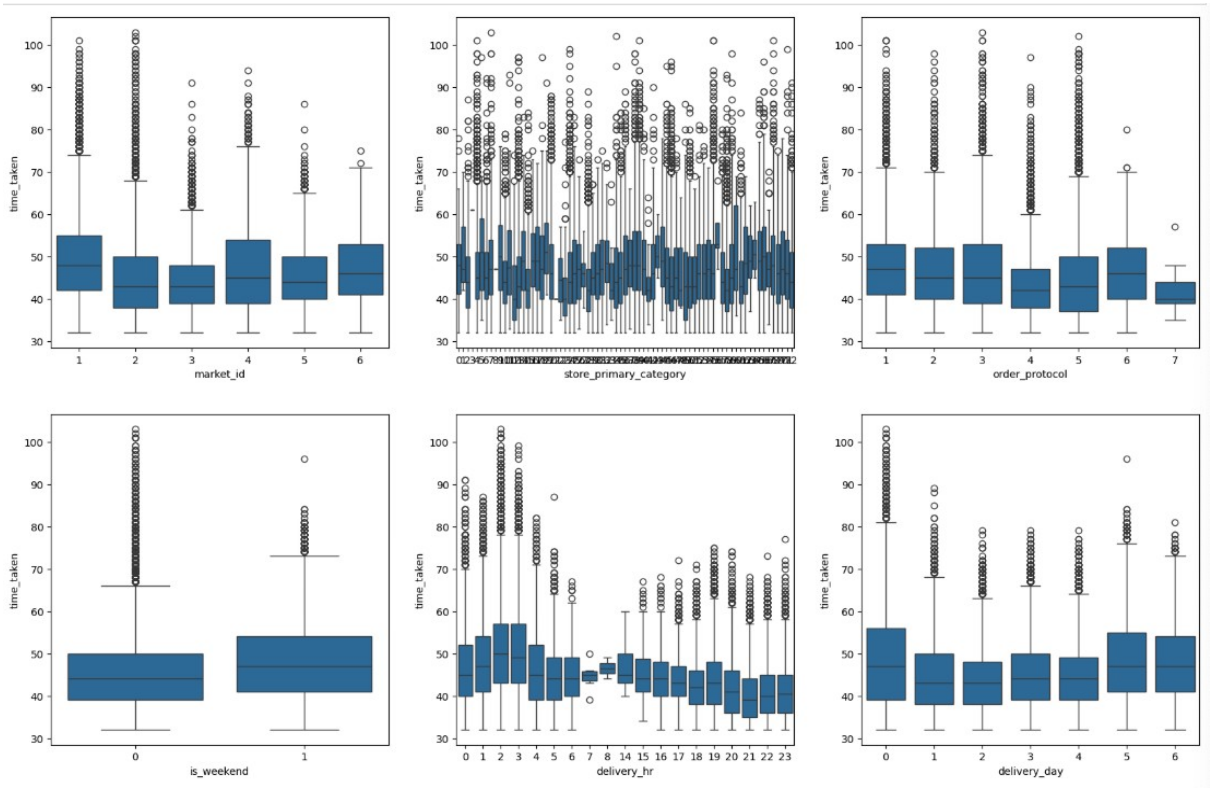
i. Analyse relationships of features with the target variable

Here we visualize Scatter plots for important numerical and categorical features to observe how they relate to `time_taken`:

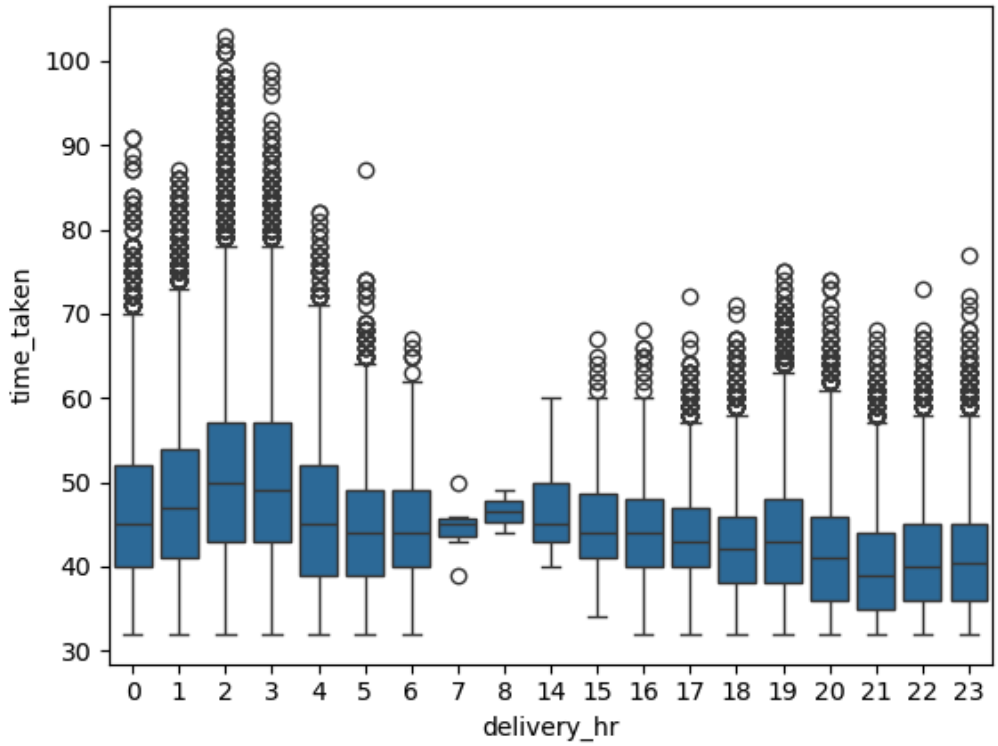




Also, we see the distribution of categorical variable with target variable.



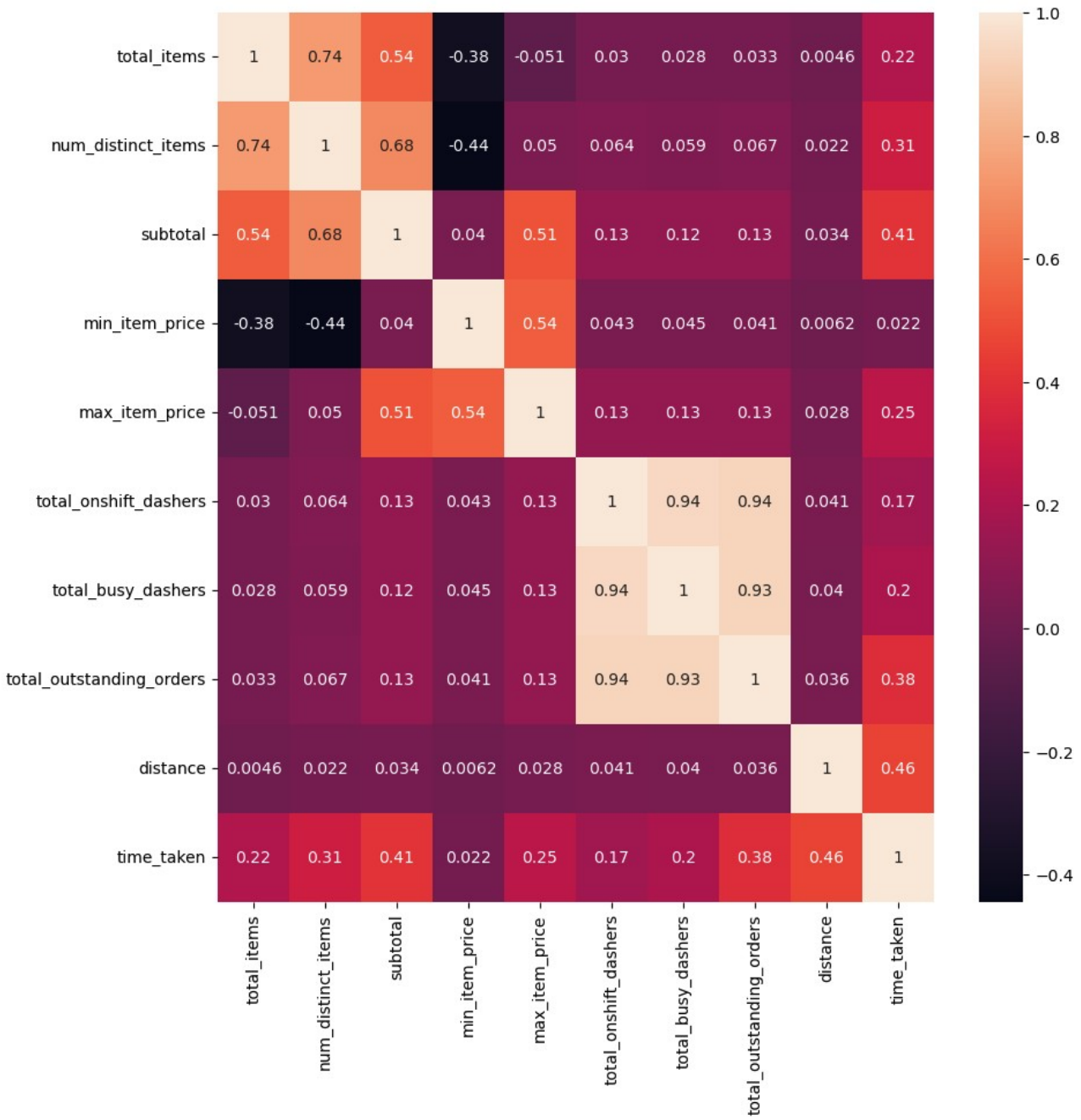
And also the distribution of target variable for different hours:



3.3 Correlation Analysis

i. Plot heatmap of feature correlations

We checked the correlations between numerical features to identify which variables are strongly related to `time_taken`:



Here we observe that following are the most correlated features:

- distance
- subtotal
- total_outstanding_orders
- num_distinct_items

ii. Drop columns with weak correlation to the target variable

Based on the correlation, we can drop following weakly correlated features:

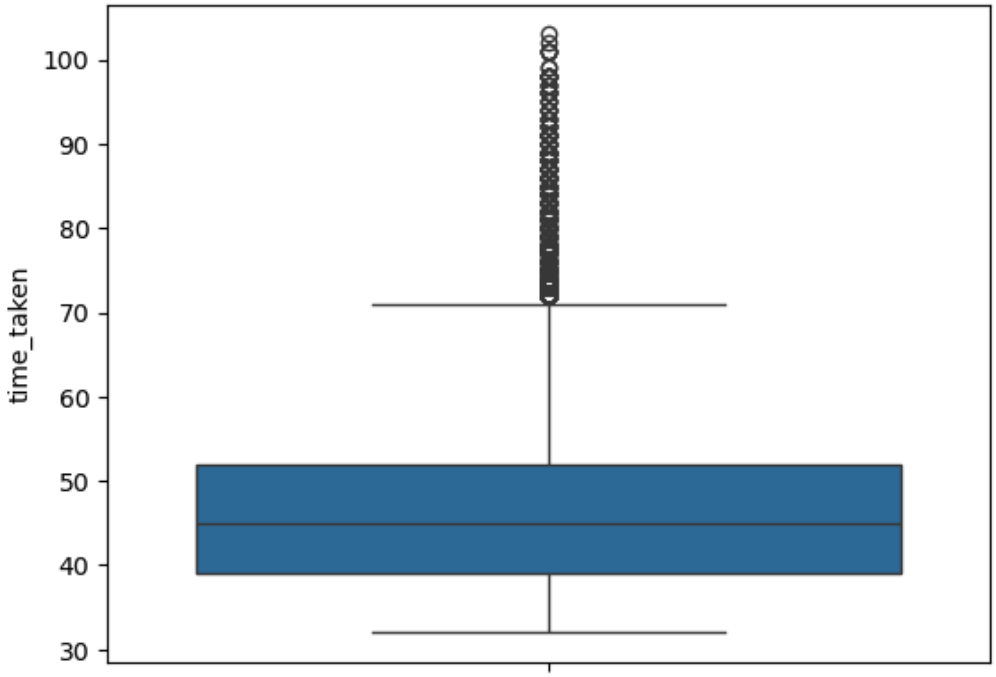
- market_id
- store_primary_category

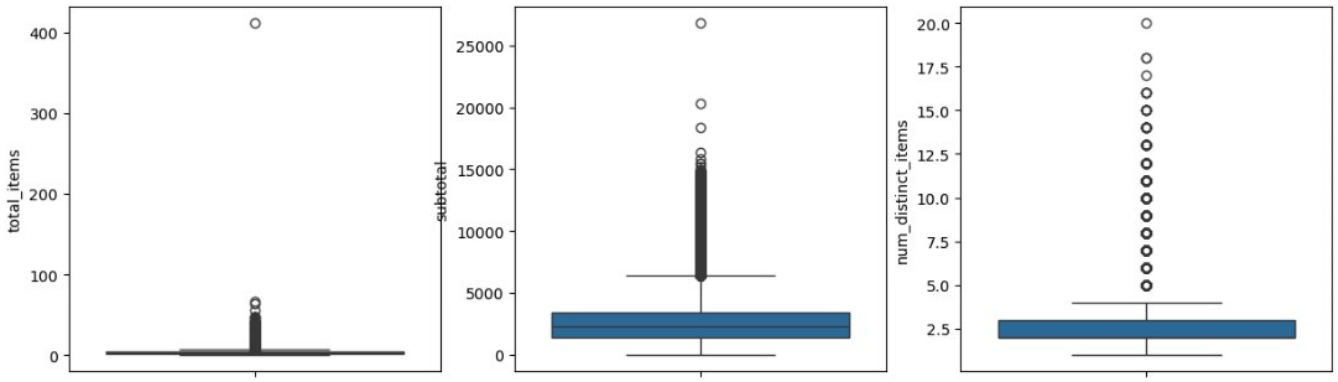
- order_protocol
- min_item_price
- total_onshift_dashers

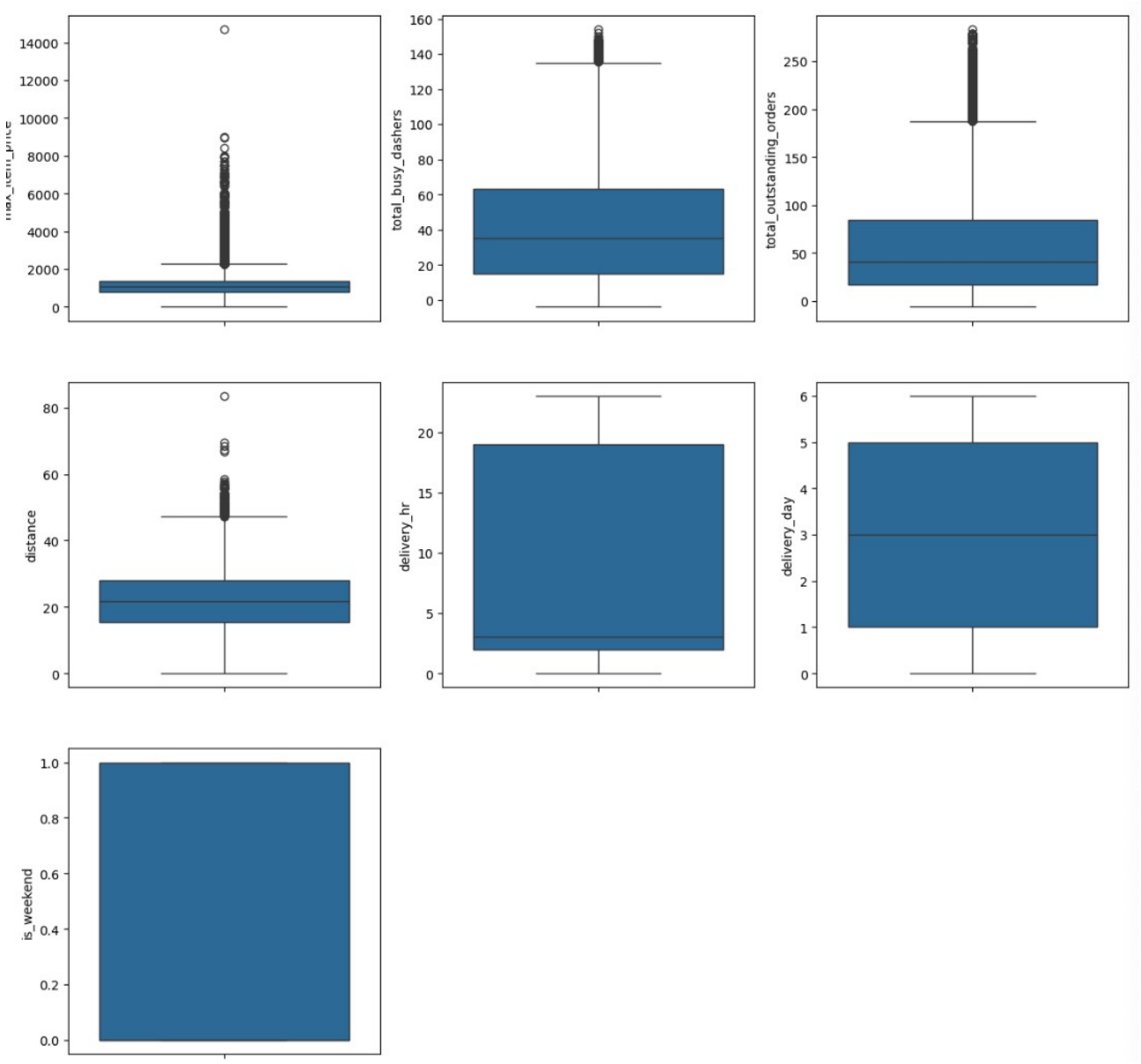
3.4 Outlier Handling

i. Visualise potential outliers

Here we visualized all the feature and target variables for the outliers:

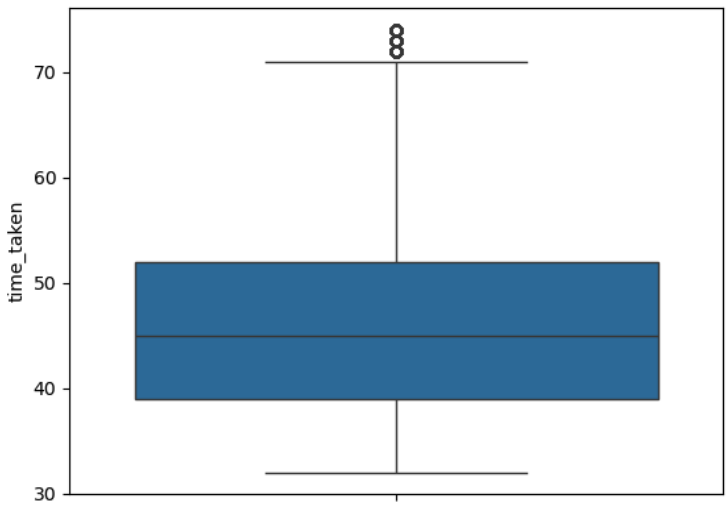


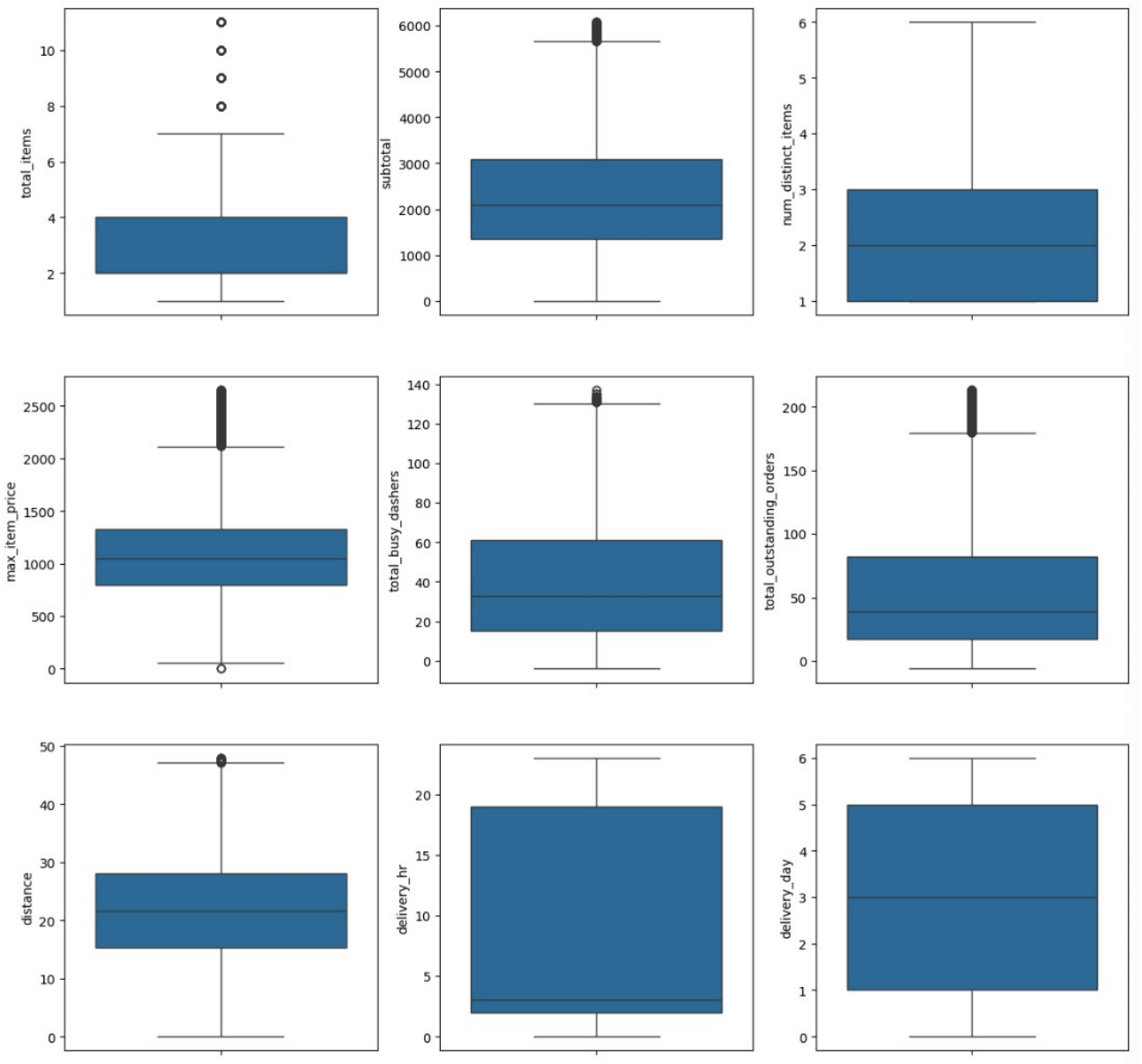


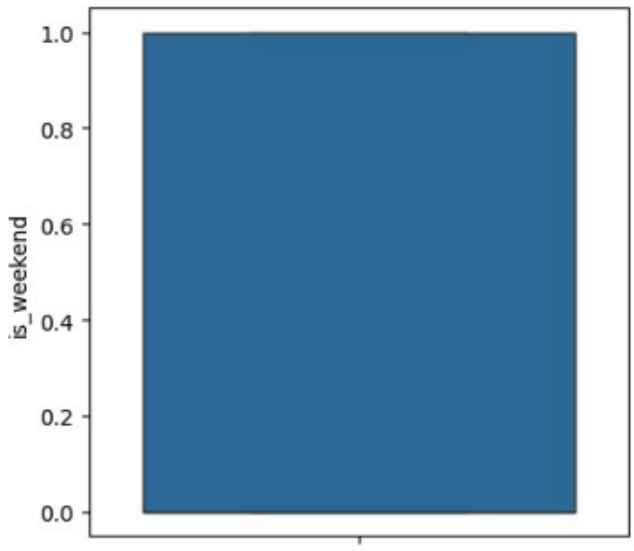


ii. Handle outliers in all columns

Here, we filter all the outliers using z-score method. Following are the plots after handling outliers:



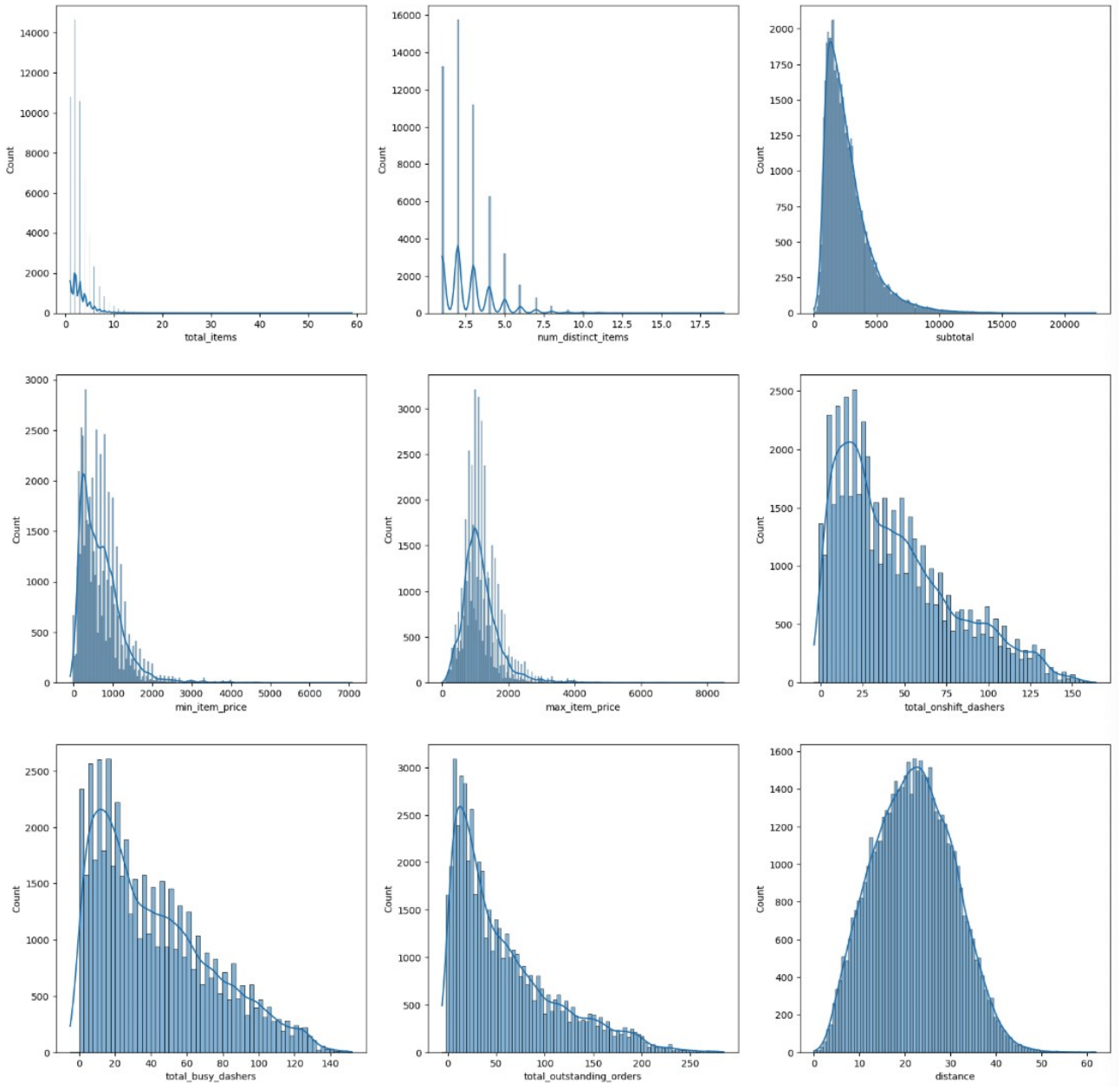

total_items
subtotal
num_distinct_items
max_item_price
total_busy_dashers
total_outstanding_orders
distance
delivery_hr
delivery_day

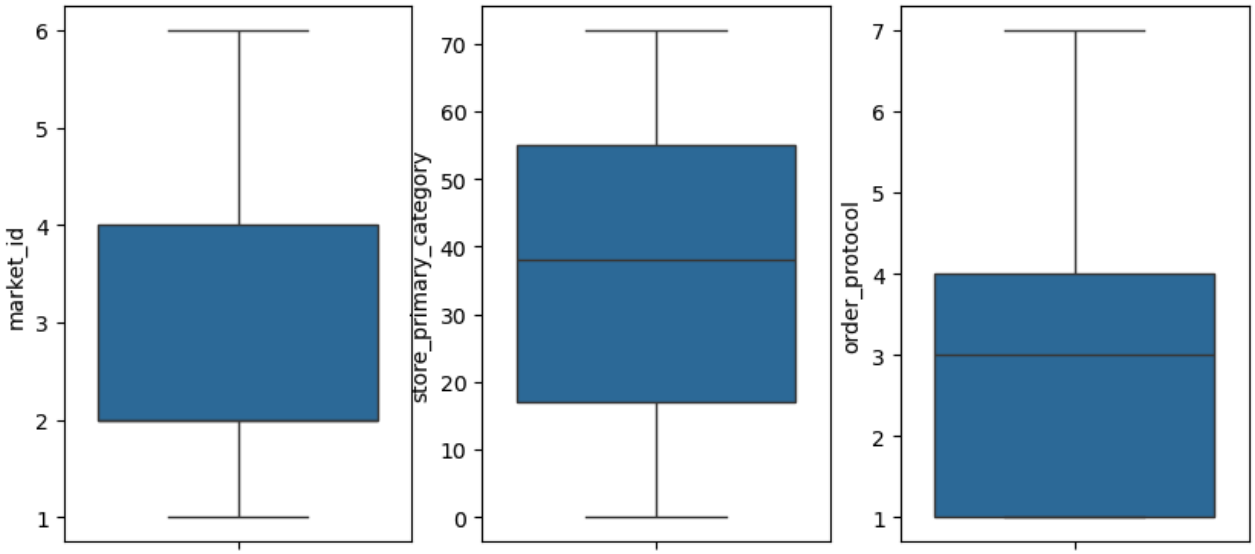

is_weekend

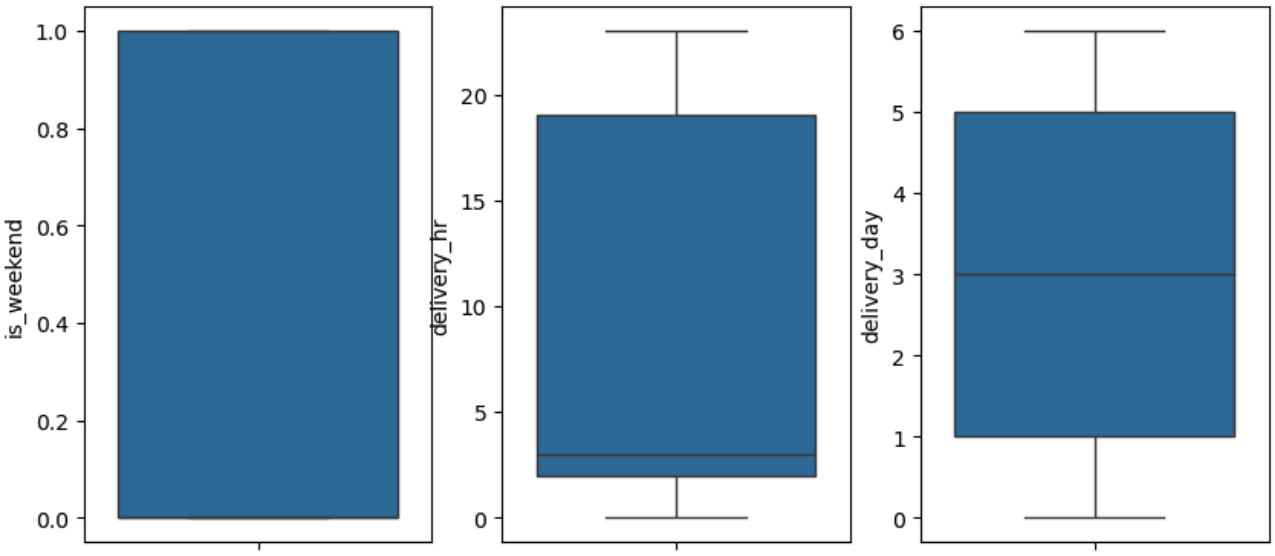
4. Exploratory Data Analysis on Validation Data

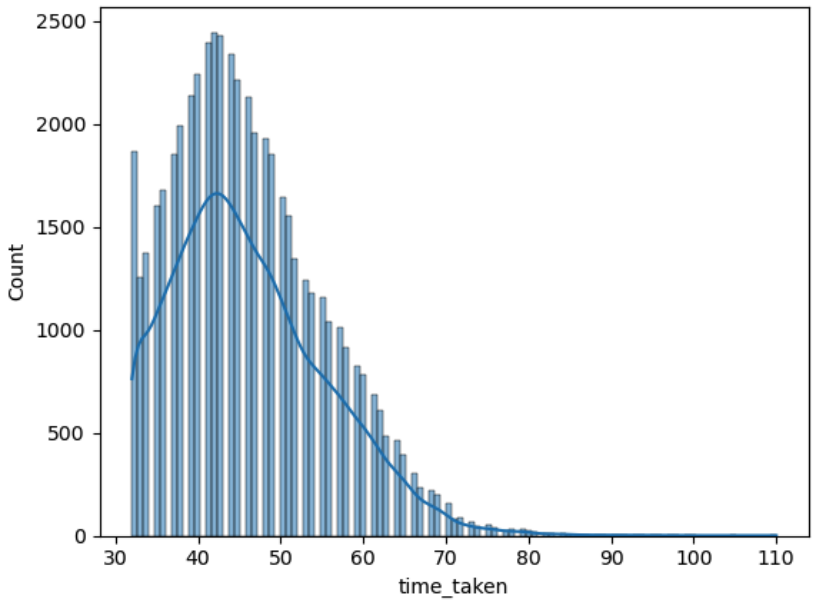
Here, we perform EDA on test data to see if the distribution match with the training data. All the distributions in test data do match with training data.

4.1 Feature Distribution

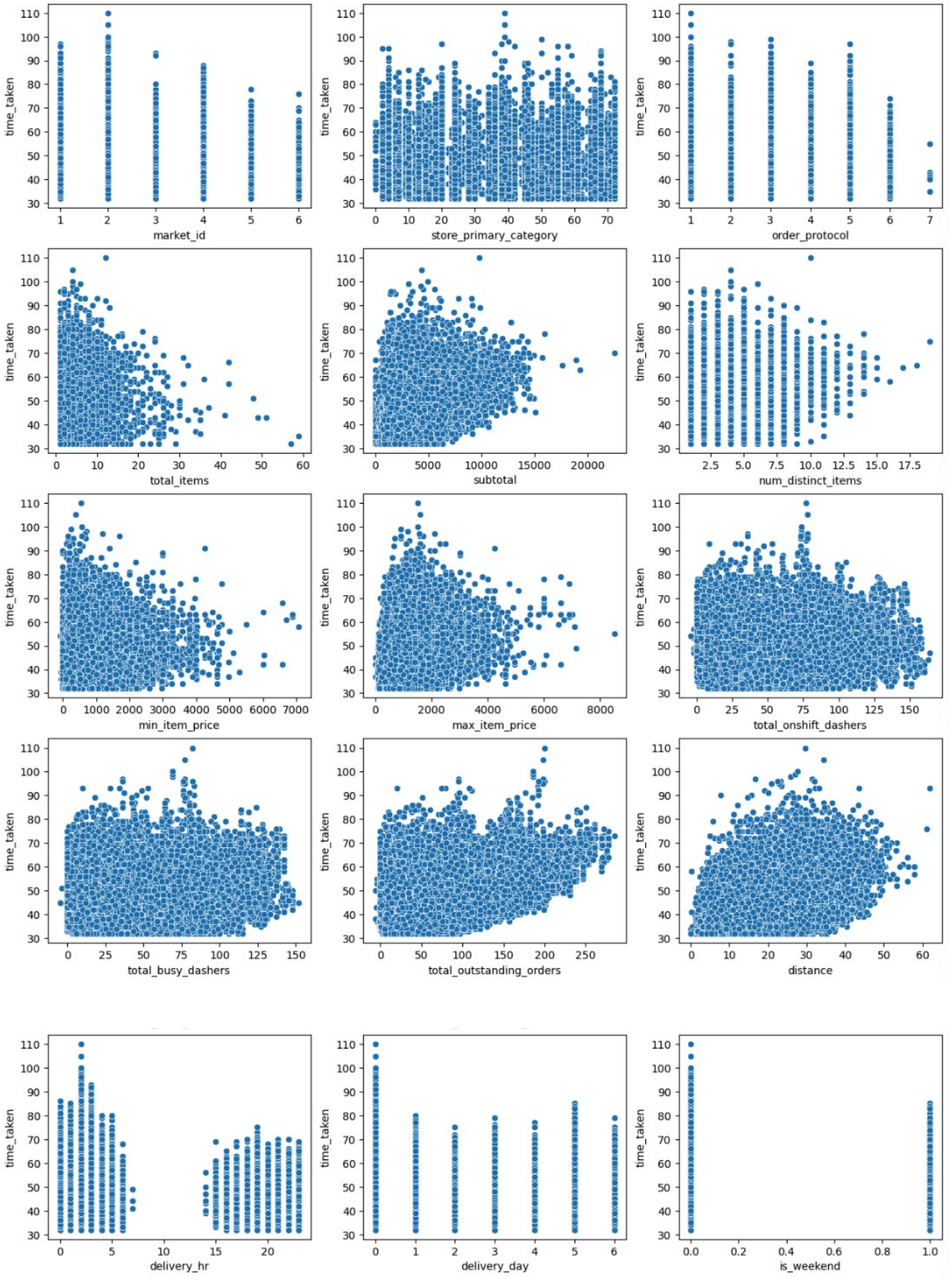



market_id
store_primary_category
order_protocol


is_weekend
delivery_hr
delivery_day


Count
time_taken

4.2 Relationships between different features



5. Model Building

5.1 Perform feature scaling

Here, we scaled all the features using sklearn's MinMaxScaler:

```
train_data_2.describe()
```

	total_items	subtotal	num_distinct_items	max_item_price	total_busy_dashers	total_outstanding_orders	distance	time_taken
count	110758.000000	110758.000000	110758.000000	110758.000000	110758.000000	110758.000000	110758.000000	110758.000000
mean	0.180834	0.384336	0.286249	0.412312	0.315863	0.281128	0.453049	0.312798
std	0.162455	0.203413	0.250432	0.162117	0.219707	0.225805	0.181132	0.201966
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.100000	0.223539	0.000000	0.301509	0.134752	0.105023	0.317765	0.166667
50%	0.100000	0.343210	0.200000	0.394340	0.262411	0.205479	0.452043	0.285714
75%	0.300000	0.506996	0.400000	0.500000	0.460993	0.401826	0.583820	0.428571
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

5.2 Build a Simple Linear Regression Model

We used statsmodels to build the linear regression model:

OLS Regression Results

Dep. Variable:	time_taken	**R-squared:**	0.687
Model:	OLS	**Adj. R-squared:**	0.687
Method:	Least Squares	**F-statistic:**	2.427e+04
Date:	Mon, 30 Jun 2025	**Prob (F-statistic):**	0.00
Time:	17:12:44	**Log-Likelihood:**	84287.
No. Observations:	110758	**AIC:**	-1.686e+05
Df Residuals:	110747	**BIC:**	-1.684e+05
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0400	0.002	24.104	0.000	0.037	0.043
total_items	-0.0137	0.005	-2.985	0.003	-0.023	-0.005
subtotal	0.1878	0.003	59.123	0.000	0.182	0.194
num_distinct_items	0.0615	0.003	22.358	0.000	0.056	0.067
max_item_price	0.0454	0.003	14.340	0.000	0.039	0.052
total_busy_dashers	-1.1742	0.004	-271.617	0.000	-1.183	-1.166
total_outstanding_orders	1.2576	0.004	296.984	0.000	1.249	1.266
distance	0.5288	0.002	281.581	0.000	0.525	0.533
delivery_hr	-0.1102	0.001	-112.437	0.000	-0.112	-0.108
delivery_day	-0.0831	0.002	-49.179	0.000	-0.086	-0.080
is_weekend	0.0917	0.001	75.714	0.000	0.089	0.094

Omnibus:	3025.901	**Durbin-Watson:**	2.004
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6095.521
Skew:	0.185	**Prob(JB):**	0.00
Kurtosis:	4.088	**Cond. No.**	27.3

Notes:

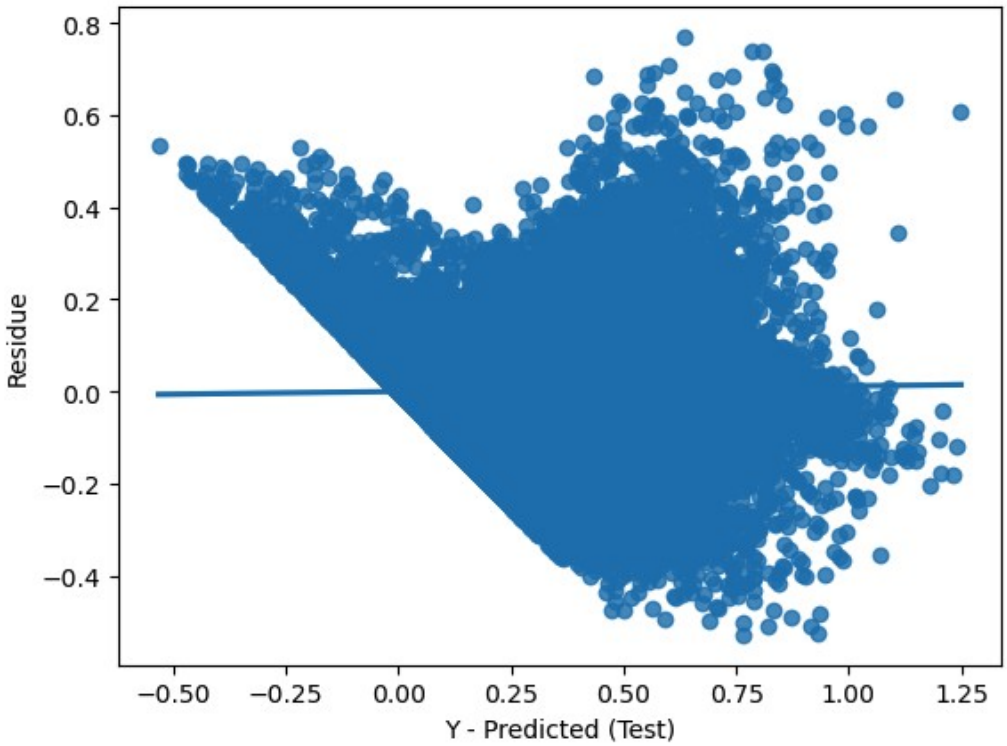
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

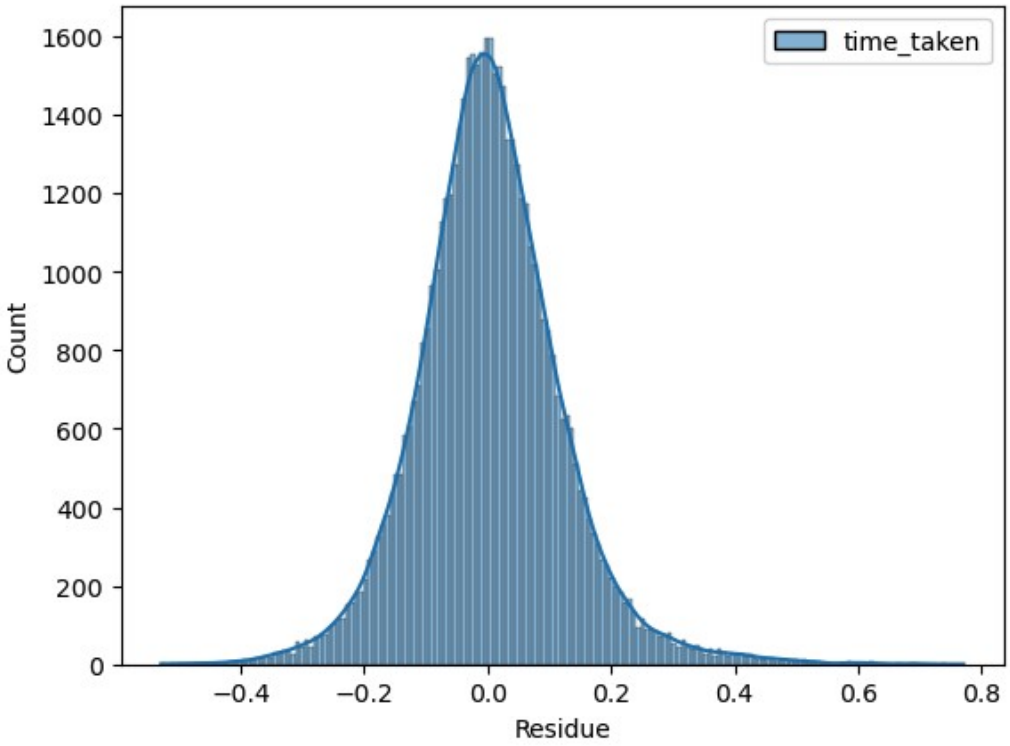
Following are the model evaluation parameters based on **test data**.

```
r2 = r2_score(y_test, y_test_predict)
```

```
r2
```

```
0.7132903144915446
```


Residue
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
Y - Predicted (Test)


time_taken
Count
1600
1400
1200
1000
800
600
400
200
0
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
Residue

5.3 Build the model and fit RFE to select the most important features

Using RFE, we got following most relevant features to construct our model:

- subtotal
- num_distinct_items
- total_busy_dashers
- total_outstanding_orders
- distance
- delivery_hr
- delivery_day
- is_weekend

6. Results and Inference

OLS Regression Results

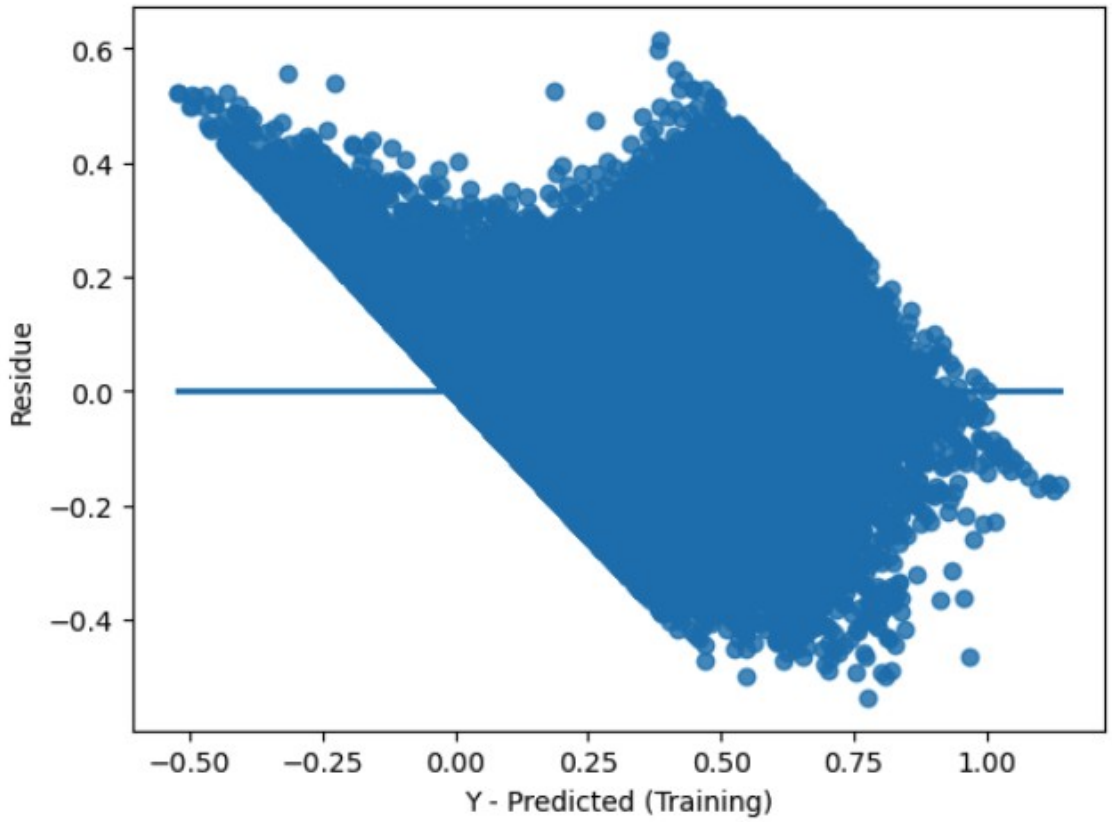
Dep. Variable:	time_taken	**R-squared:**	0.686
Model:	OLS	**Adj. R-squared:**	0.686
Method:	Least Squares	**F-statistic:**	3.022e+04
Date:	Mon, 30 Jun 2025	**Prob (F-statistic):**	0.00
Time:	17:12:47	**Log-Likelihood:**	84131.
No. Observations:	110758	**AIC:**	-1.682e+05
Df Residuals:	110749	**BIC:**	-1.682e+05
Df Model:	8		
Covariance Type:	nonrobust		

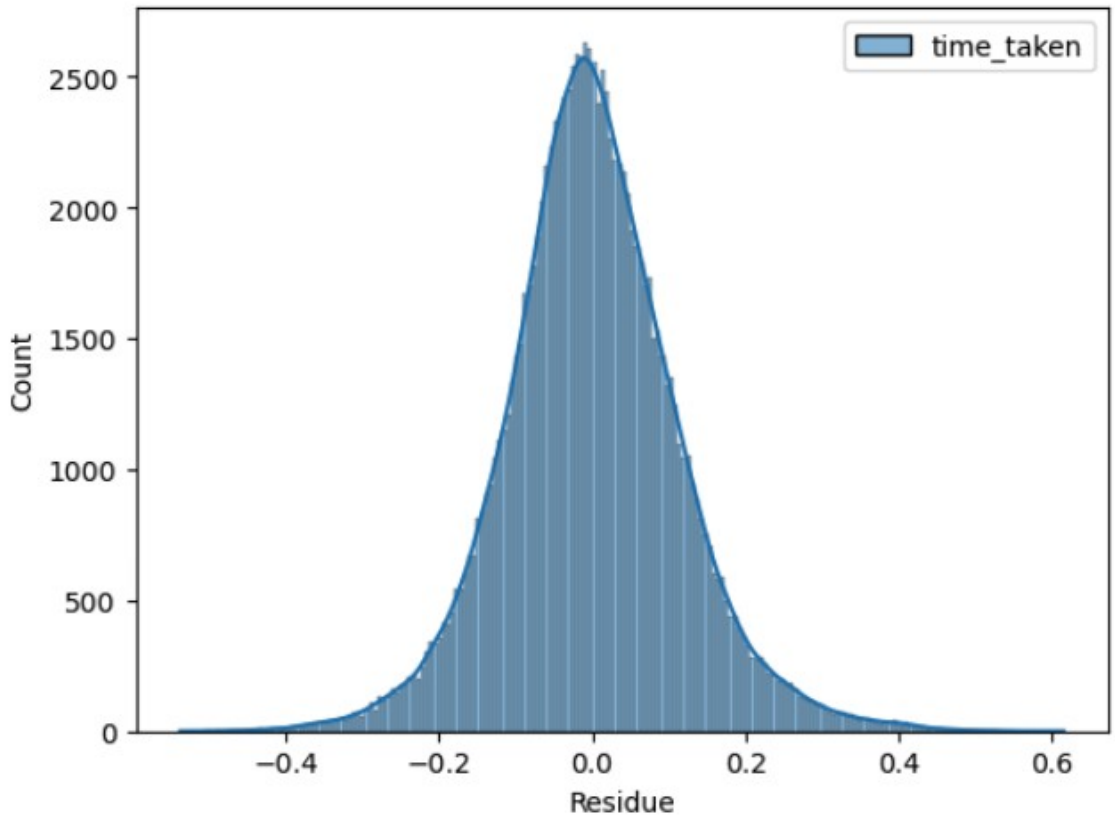
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	0.0513	0.001	35.238	0.000	0.048	0.054
subtotal	0.2167	0.002	98.931	0.000	0.212	0.221
num_distinct_items	0.0395	0.002	22.685	0.000	0.036	0.043
total_busy_dashers	-1.1735	0.004	-271.077	0.000	-1.182	-1.165
total_outstanding_orders	1.2582	0.004	296.709	0.000	1.250	1.266
distance	0.5291	0.002	281.334	0.000	0.525	0.533
delivery_hr	-0.1116	0.001	-114.050	0.000	-0.113	-0.110
delivery_day	-0.0831	0.002	-49.133	0.000	-0.086	-0.080
is_weekend	0.0918	0.001	75.748	0.000	0.089	0.094

Omnibus:	3022.175	**Durbin-Watson:**	2.005
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6066.428
Skew:	0.186	**Prob(JB):**	0.00
Kurtosis:	4.084	**Cond. No.**	26.2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

6.1 Perform Residual Analysis





6.2 Perform Coefficient Analysis

How changes in features affect time_taken?

Positive Impact (increase delivery time):

- Distance (+0.4633): Each additional unit of distance increases delivery time by ~0.46 minutes
- Number of distinct items (+0.3320): More items in an order increase delivery time
- Outstanding orders (+0.2413): Higher order volume in the system increases delivery time
- Weekend delivery (+3.8566): Weekend deliveries take ~3.9 minutes longer
- Order subtotal (+0.0015): Larger orders slightly increase delivery time

Negative Impact (decrease delivery time):

- Busy dashers (-0.3495): More available dashers reduce delivery time
- Delivery hour (-0.2037): Later hours in the day reduce delivery time
- Delivery day (-0.5820): Later days (Thursday, Friday) in some cycle reduce delivery time

Affect of feature scaling on coefficients:

- In linear models, coefficients of unscaled features are not directly comparable due to differing units and scales.
- After scaling, coefficients reflect the relative importance of each feature more meaningfully, since all features are on a comparable scale.
- After scaling:
    - Total outstanding orders (+1.2582) becomes the strongest positive predictor in place of Weekend delivery in case on unscaled feature.
    - Total busy dashers (-1.1735) becomes the strongest negative predictor in place of Delivery day in case of unscaled feature.
- This demonstrates that scaled features offer better explanation of time_taken compared to unscaled features.

Analysis of the effect of a unit change in a feature

Additionally, we can analyse the effect of a unit change in a feature. In other words, because we have scaled the features, a unit change in the features will not translate directly to the model.

- One unit change in total_items causes the time_taken change by: 0.021670952871238 units
- One unit change in subtotal causes the time_taken change by: 3.567235040533004e-05 units

- One unit change in num_distinct_items causes the time_taken change by: 0.043341905742476 units
- One unit change in max_item_price causes the time_taken change by: 8.177718064618112e-05 units
- One unit change in total_busy_dashers causes the time_taken change by: 0.0015369470121445387 units
- One unit change in total_outstanding_orders causes the time_taken change by: 0.0009895412269971687 units
- One unit change in distance causes the time_taken change by: 0.004518547304261467 units
- One unit change in delivery_hr causes the time_taken change by: 0.00942215342227739 units
- One unit change in delivery_day causes the time_taken change by: 0.03611825478539666 units
- One unit change in is_weekend causes the time_taken change by: 0.21670952871237997 units

Scaled features:

Positive Impact (increase delivery time):

- Distance (+0.5291): Each additional unit of distance increases delivery time significantly
- Total outstanding orders (+1.2582): Strongest positive predictor - higher order volume dramatically increases delivery time
- Order subtotal (+0.2167): Larger orders increase delivery time
- Number of distinct items (+0.0395): More items in an order increase delivery time
- Weekend delivery (+0.0918): Weekend deliveries take longer

Negative Impact (decrease delivery time):

- Total busy dashers (-1.1735): Strongest negative predictor - counterintuitively, more busy dashers reduce delivery time
- Delivery hour (-0.1116): Later hours in the day reduce delivery time
- Delivery day (-0.0831): Later days reduce delivery time

Unscaled Features:

Positive Impact (increase delivery time):

- Distance (+0.4633): Each additional unit of distance increases delivery time by ~0.46 minutes

- Number of distinct items (+0.3320): More items in an order increase delivery time
- Outstanding orders (+0.2413): Higher order volume in the system increases delivery time
- Weekend delivery (+3.8566): Weekend deliveries take ~3.9 minutes longer
- Order subtotal (+0.0015): Larger orders slightly increase delivery time

Negative Impact (decrease delivery time):

- Busy dashers (-0.3495): More available dashers reduce delivery time
- Delivery hour (-0.2037): Later hours in the day reduce delivery time
- Delivery day (-0.5820): Later days (Thursday, Friday) in some cycle reduce delivery time

7.Subjective Questions

7.1 Assignment-related subjective questions

Question 1. [2 marks]

Are there any categorical variables in the data? From your analysis of the categorical variables from the dataset, what could you infer about their effect on the dependent variable?

Answer: Following are the categorical data in the dataset:

- market_id
- store_primary_category
- order_protocol
- is_weekend

From the analysis it looks like that on the category is_weekend is statistically significant in affecting the target variable time_taken.

Question 2. [1 marks]

What does test_size = 0.2 refer to during splitting the data into training and test sets?

Answer: test_size = 0.2 denotes that 20% of the total dataset will be split for test data, while 80% will be split for training data.

Question 3. [1 marks]

Looking at the heatmap, which one has the highest correlation with the target variable?

Answer: Looking at the heatmap, it looks like variable distance is the highly correlated feature with time_taken with the correlation of 0.46.

Question 4. [2 marks]

What was your approach to detect the outliers? How did you address them?

Answer: I used box plot to detect outliers visually.

I used Z-Score methodology to filter out outliers that were 2 or 3 deviations away from the mean.

Question 5. [2 marks]

Based on the final model, which are the top 3 features significantly affecting the delivery time?

Answer: Based on the final model (lr_model2_rfe), I found following top three features significantly affecting the delivery time:

- total_busy_dashers
- total_outstanding_orders
- distance

7.2 Topic-related subjective questions

Question 6. [3 marks]

Explain the linear regression algorithm in detail

Answer: Linear regression algorithm is a way of finding best straight line (y = b0 + b1x1 + b2x2 ...) fitting any data. Linear regression algorithm works when:

- There is a linear correlation between the dependend and independent variables.
- The errors (actual - predicted value) are normally distributed with mean=0.
- The errors (actual - predicted value) are independent of each other that is, should be random.
- The errors (actual - predicted value) should have constant variance throughout the feature data values.

If these are true then there will be a good chance that a line can be drawn thought the data that will be able to predict the trend.

The coefficients values are derived using gradient descent method.

Question 7. [2 marks]

Explain the difference between simple linear regression and multiple linear regression

Answer: Simple linear regression deals with only one dependent variable (y = b0 + b1x1) while multiple linear regression deals with seleral dependent variables (y = b0 + b1x1 + b2x2 + ... bnxn).

Simple linear regression can be visualized using 2D plot, while multiple linear regression requires 3D or higher dimention plots for visualization.

Multicollinearity is applicable in multiple linear regression but not in simple linear regression.

Question 8. [2 marks]

What is the role of the cost function in linear regression, and how is it minimized?

Answer: Cost function in linear regression is the average of sum of squares of error terms, where error = actual value - predicted value.

cost_function = 1/n SUM[(y - y_pred)^2]

Cost function is used to measure performance of the model. For better model performance, cost function should be reduced.

Cost function is minimized using gradient descent.

Question 9. [2 marks]

Explain the difference between overfitting and underfitting.

Answer: Overfitting means that the regression model 'learns' the training data and when it is run on test data, it performs poorly because it fails to generalize the general data trend. However the performance is very good on training data.

In overfitting the model is very complex than it is required.

Underfitting means that the regression model could not generalize to the training data itself and hence performs poorly both on training data as well as test data.

In underfitting the model is very simple than it is required.

Question 10. [3 marks]

How do residual plots help in diagnosing a linear regression model?

Answer: Residual plot help us to confirm following three assumptions of linear regression:

The residues (y_actual - y_predicted) are randomly destributed around the 0 value, that is, there is no particular pattern found in residue distribution and residues are not dependent on each other.

The residues are normally distributed with mean=0.

The variance is constant throughout the data (homoscedaticity).

If all the above are true then the quality of model fit is good.