Jagdsh Chand
Report: LR Delivery Time Estimation

Include your visualisations, analysis, results, insights, and outcomes. Explain your methodology and approach to the tasks. Add your conclusions to the sections.

1. Loading the data

“porter_data_1.csv” was loaded into the pandas dataframe and data types of the individual columns was printed.

```
print(porter.dtypes)
[1115]
```

```
market_id                    float64
created_at                    object
actual_delivery_time          object
store_primary_category         int64
order_protocol               float64
total_items                    int64
subtotal                       int64
num_distinct_items             int64
min_item_price                 int64
max_item_price                 int64
total_onshift_dashers        float64
total_busy_dashers           float64
total_outstanding_orders     float64
```

```
porter.describe()
[1116]
```

8 rows × 8 rows × 12 cols — Static Output

	market_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_ons
count	175777.000000	175777.000000	175777.000000	175777.000000	175777.000000	175777.000000	175777.000000	175777.000000	
mean	2.743726	35.887949	2.911752	3.204976	2697.111147	2.675060	684.965433	1160.158616	
std	1.330963	20.728254	1.513128	2.674055	1828.554893	1.625681	519.882924	560.828571	
min	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	-86.000000	0.000000	
25%	2.000000	18.000000	1.000000	2.000000	1412.000000	1.000000	299.000000	799.000000	
50%	2.000000	38.000000	3.000000	3.000000	2224.000000	2.000000	595.000000	1095.000000	
75%	4.000000	55.000000	4.000000	4.000000	3410.000000	3.000000	942.000000	1395.000000	
max	6.000000	72.000000	7.000000	411.000000	26800.000000	20.000000	14700.000000	14700.000000	

2. Data Preprocessing and Feature Engineering

2.1. Fixing the Datatypes

2.1.1. Date and time fields were fixed

```
# Convert 'created_at' and 'actual_delivery_time' columns to datetime format
porter['actual_delivery_time'] = pd.to_datetime(porter['actual_delivery_time'])
porter['created_at'] = pd.to_datetime(porter['created_at'])

print(porter.dtypes)
✓ [6] 38ms
```

```
market_id                      float64
created_at                datetime64[ns]
actual_delivery_time      datetime64[ns]
store_primary_category           int64
order_protocol                 float64
total_items                      int64
subtotal                         int64
num_distinct_items               int64
min_item_price                   int64
max_item_price                   int64
total_onshift_dashers          float64
total_busy_dashers             float64
total_outstanding_orders       float64
```

2.1.2. Convert categorical fields to appropriate data types

The datatype “category” was used for the the categorical columns

```
['market_id', 'store_primary_category', 'order_protocol']
```

```
# Convert categorical features to category type
categorical_cols = ['market_id', 'store_primary_category', 'order_protocol']
for col in categorical_cols:
    porter[col] = porter[col].astype('category')     porter      col
print(porter.dtypes)
✓ [7] < 10 ms
```

```
market_id                    category
created_at                datetime64[ns]
actual_delivery_time      datetime64[ns]
store_primary_category       category
order_protocol               category
total_items                     int64
subtotal                        int64
num_distinct_items              int64
min_item_price                  int64
max_item_price                  int64
total_onshift_dashers         float64
total_busy_dashers            float64
total_outstanding_orders      float64
```

2.2. Feature Engineering

2.2.1. Calculate the time taken using the features `actual_delivery_time` and `created_at`

I populated the time_taken column in minutes for the `actual_delivery_time` and `created_at`

```
# Calculate time taken in minutes
porter['delivery_time_timedelta'] = porter['actual_delivery_time'] - porter['created_at']
porter['time_taken'] = porter['delivery_time_timedelta'].dt.total_seconds() / 60

porter[['created_at', 'actual_delivery_time', 'delivery_time_timedelta', 'time_taken']].head()
```

✓ [74] < 10 ms

5 rows × 4 cols

	created_at	actual_delivery_time	delivery_time_timedelta	time_taken
0	2015-02-06 22:24:17	2015-02-06 23:11:17	0 days 00:47:00	47.0
1	2015-02-10 21:49:25	2015-02-10 22:33:25	0 days 00:44:00	44.0
2	2015-02-16 00:11:35	2015-02-16 01:06:35	0 days 00:55:00	55.0
3	2015-02-12 03:36:46	2015-02-12 04:35:46	0 days 00:59:00	59.0
4	2015-01-27 02:12:36	2015-01-27 02:58:36	0 days 00:46:00	46.0

2.2.2. Extract the hour at which the order was placed and which day of the week it was. Drop the unnecessary columns.

1. I used the lambda function to calculate the isWeekend column
2. I dropped unnecessary columns 'created_at', 'actual_delivery_time', 'day_of_week', 'delivery_time_timedelta'

```
# Extract the hour and day of week from the 'created_at' timestamp
porter_time = porter
porter_time['hour_of_day'] = porter_time['created_at'].dt.hour
porter_time['day_of_week'] = porter_time['created_at'].dt.dayofweek

# Create a categorical feature 'isWeekend'
porter_time['isWeekend'] = porter_time['day_of_week'].apply(lambda x: 1 if x >= 5 else 0)

...

porter_time[['hour_of_day', 'day_of_week', 'isWeekend']].head()     porter_time
```

✓ [139] 33ms

5 rows × 3 cols

	hour_of_day	day_of_week	isWeekend
0	22	4	0
1	21	1	0
2	0	0	0
3	3	3	0
4	2	1	0

2.3. Creating training and validation sets

2.3.1. Define target and input features

I defined the y as time_taken since it is what we are predicting, and the features as variables_all

```
Code | Markdown | SQL

# Define target variable (y) and features (X)

y = porter['time_taken']
variables_all = ['market_id', 'store_primary_category', 'order_protocol', 'subtotal', 'num_distinct_items',
                 'min_item_price',
                 'max_item_price', 'total_onshift_dashers', 'total_busy_dashers', 'total_outstanding_orders',
                 'distance',
                 'hour_of_day', 'isWeekend']
X = porter[variables_all]


✓ [141] < 10 ms
```

2.3.2. Split the data into training and test sets

I split the data into 70% training and 30% test data with the respective rows **123043** and **52734**

```
# Split data into training and testing sets

porter_train, porter_test = train_test_split(porter, train_size=0.7, test_size=0.3, random_state=100)
print(porter_train.shape)
print(porter_test.shape)

✓ [142] < 10 ms

(123043, 15)
(52734, 15)
```

3. Exploratory Data Analysis on Training Data

3.1. Feature Distributions

Defined the categorical ('market_id', 'store_primary_category', 'order_protocol', 'isWeekend') and numerical columns.

```
# Define numerical and categorical columns for easy EDA and data manipulation
categorical_cols = ['market_id', 'store_primary_category', 'order_protocol', 'isWeekend']
numerical_cols = porter_train.drop(columns=categorical_cols).columns.tolist()
print(numerical_cols)

✓ [143] < 10 ms
['total_items', 'subtotal', 'num_distinct_items', 'min_item_price', 'max_item_price', 'total_onshift_dashers', 'total_busy_dashers', 'total_outstanding_orders',
'distance', 'time_taken', 'hour_of_day']
```

3.1.1. Plot distributions for numerical columns in the training set.

Following numerical columns has some notable distributions:

'total_items': mostly less than 100, one outlier above 400

'subtotal': mostly less than 5000, few outliers

'num_distinct_items': 2.5 items distinct stands out, however it varies a lot

'min_item_price': many below 1000 and a huge outlier

'max_item_price': many still less than 1000

'total_onshift_dashers': peak in 20-25 and decreases linearly to 175

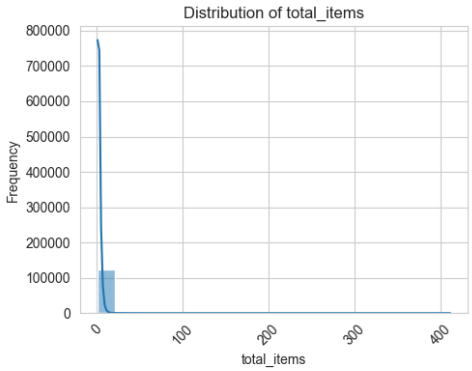
'total_busy_dashers': it is similar to onshift dashers, decreases linearly to 140

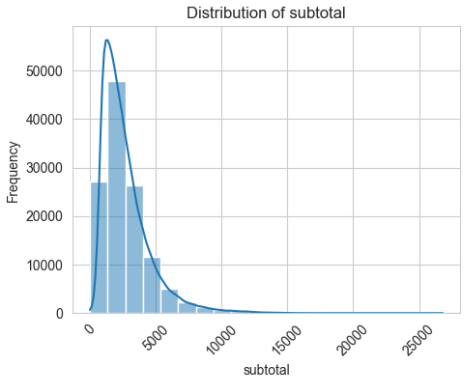
'total_outstanding_orders': more less than 50

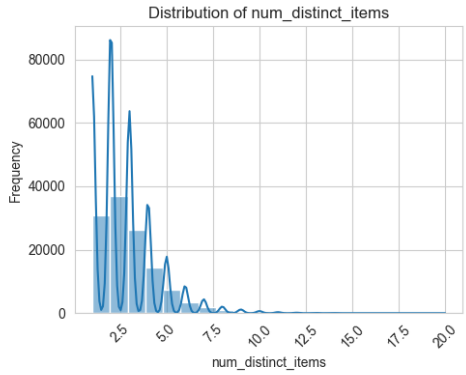
'distance': 20-40 wins the chart

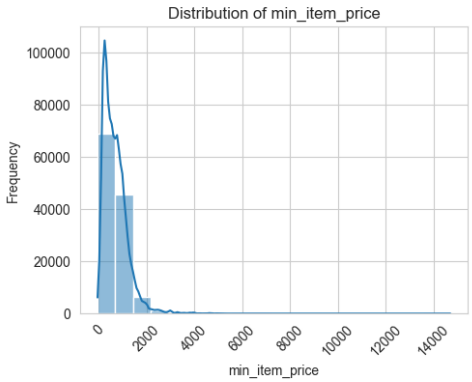
'time_taken': less than 70 , with many in 50

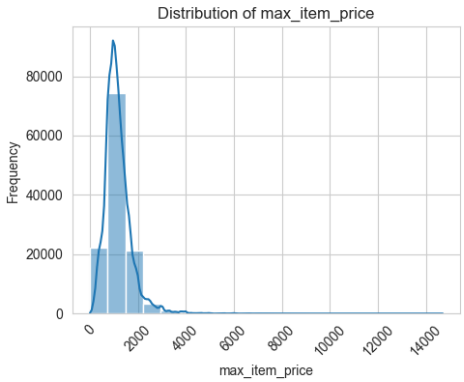
'hour_of_day': as the time increases less orders, peaking midnight

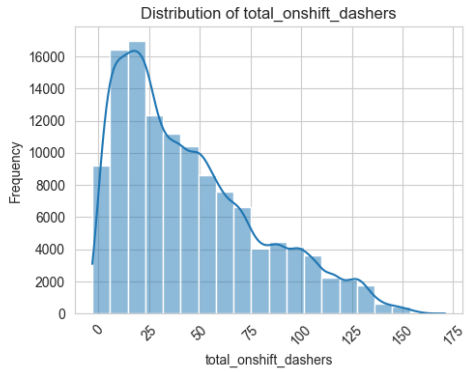

Distribution of total_items
Frequency
total_items

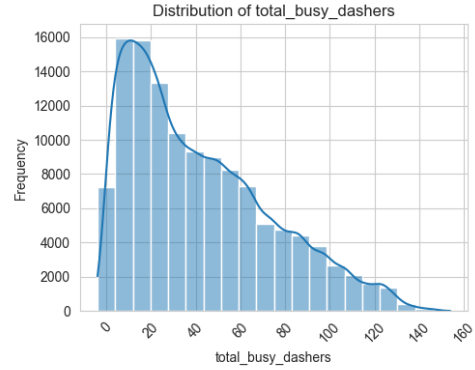

Distribution of subtotal
Frequency
subtotal

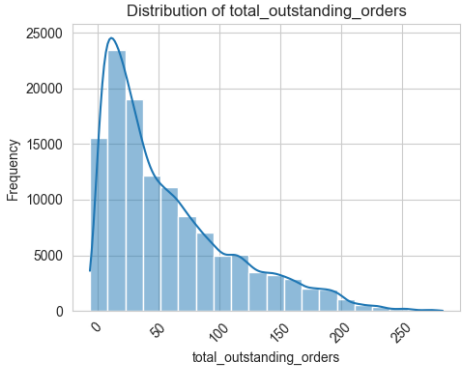

Distribution of num_distinct_items
Frequency
num_distinct_items

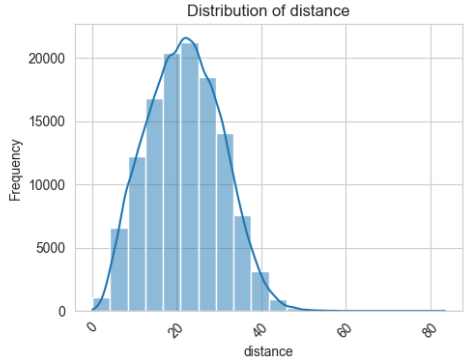

Distribution of min_item_price
Frequency
min_item_price

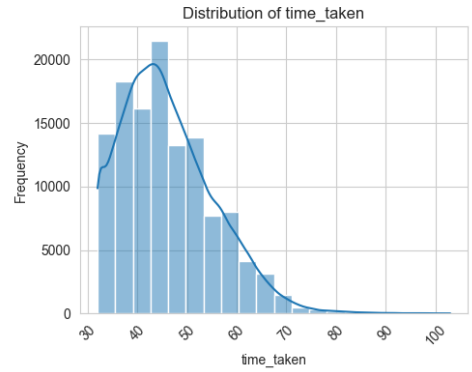

Distribution of max_item_price
Frequency
max_item_price

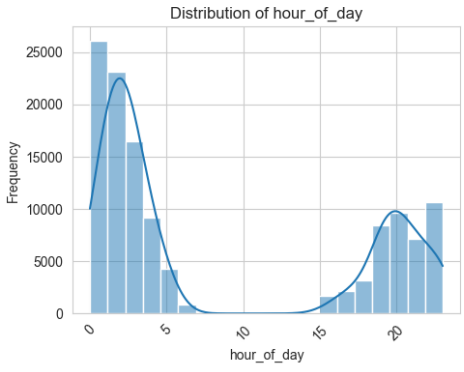

Distribution of total_onshift_dashers
Frequency
total_onshift_dashers


Distribution of total_busy_dashers
Frequency
total_busy_dashers


Distribution of total_outstanding_orders
Frequency
total_outstanding_orders


Distribution of distance
Frequency
distance


Distribution of time_taken
Frequency
time_taken


Distribution of hour_of_day
Frequency
hour_of_day

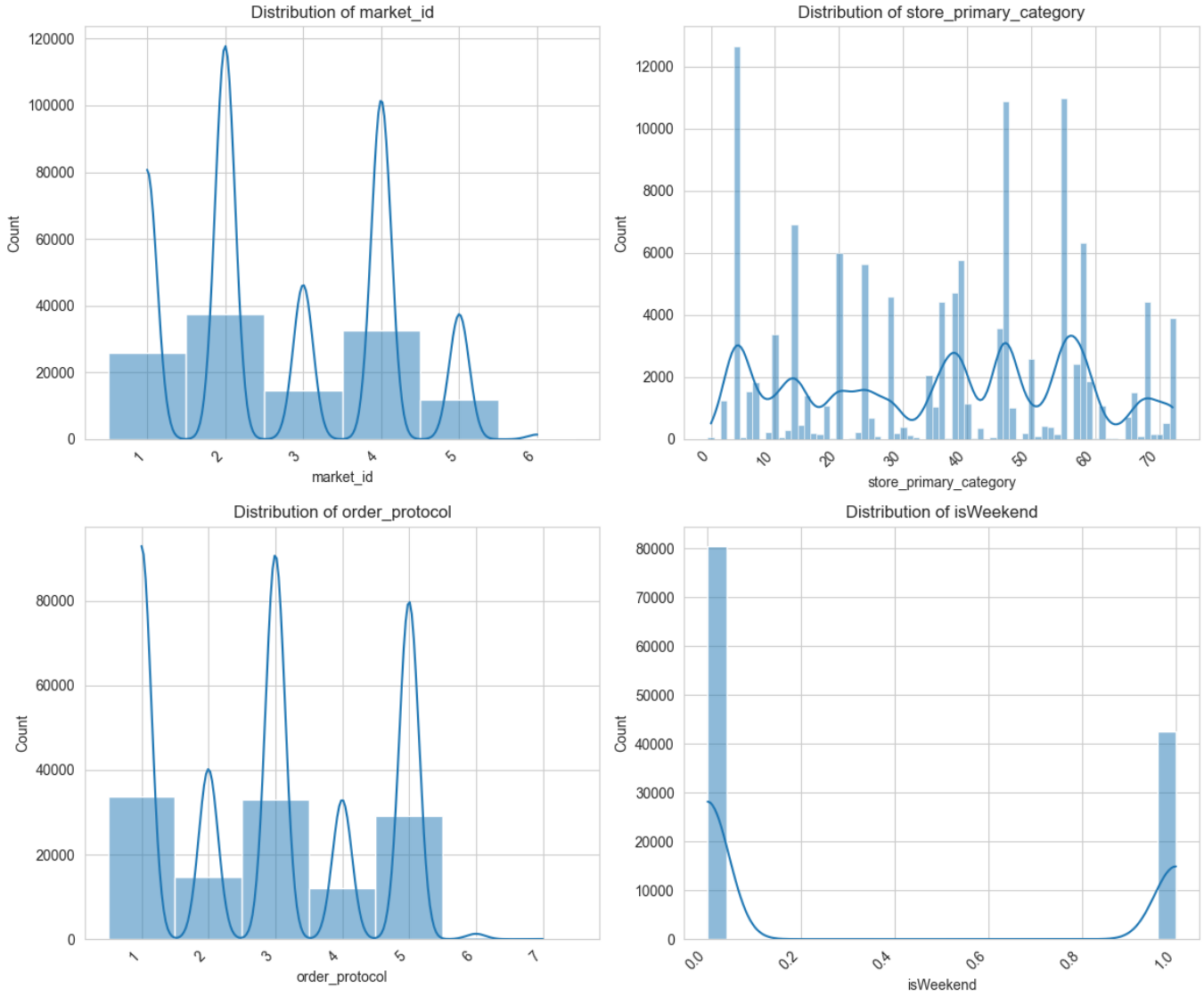
3.1.2. Check the distribution of categorical features

'market_id': market 2, 4,1 are the top three

'store_primary_category': three category (5, 48, 68) makes more than 8000 times

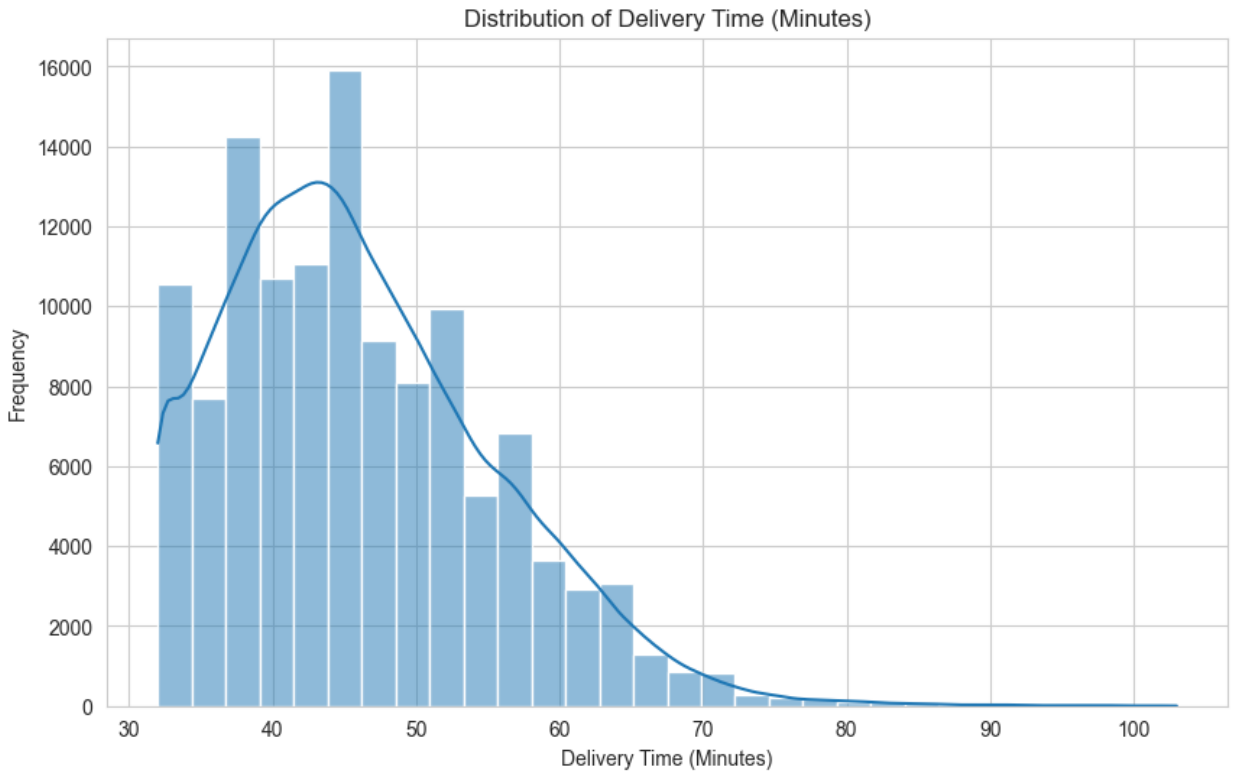
'order_protocol': alternative order protocol (1,3,5) are doing good.

'isWeekend': many in weekdays than in weekends



3.1.3. Visualise the distribution of the target variable to understand its spread and any skewness

'time_taken': less than 70 , with peaking b/w 40-50.



Conclude with results

This analysis of the of the delivery time . Below are the key findings:

1. Majority of transactions involve a relatively small quantity of items.

2. People tend to order a small variety of items, even if the total quantity of items (from total_items) can be higher. The small secondary peak might indicate some bulk or diverse orders

3. Most deliveries are completed within a relatively consistent timeframe, indicating efficiency. The slight skew or minor peak might suggest some faster deliveries.

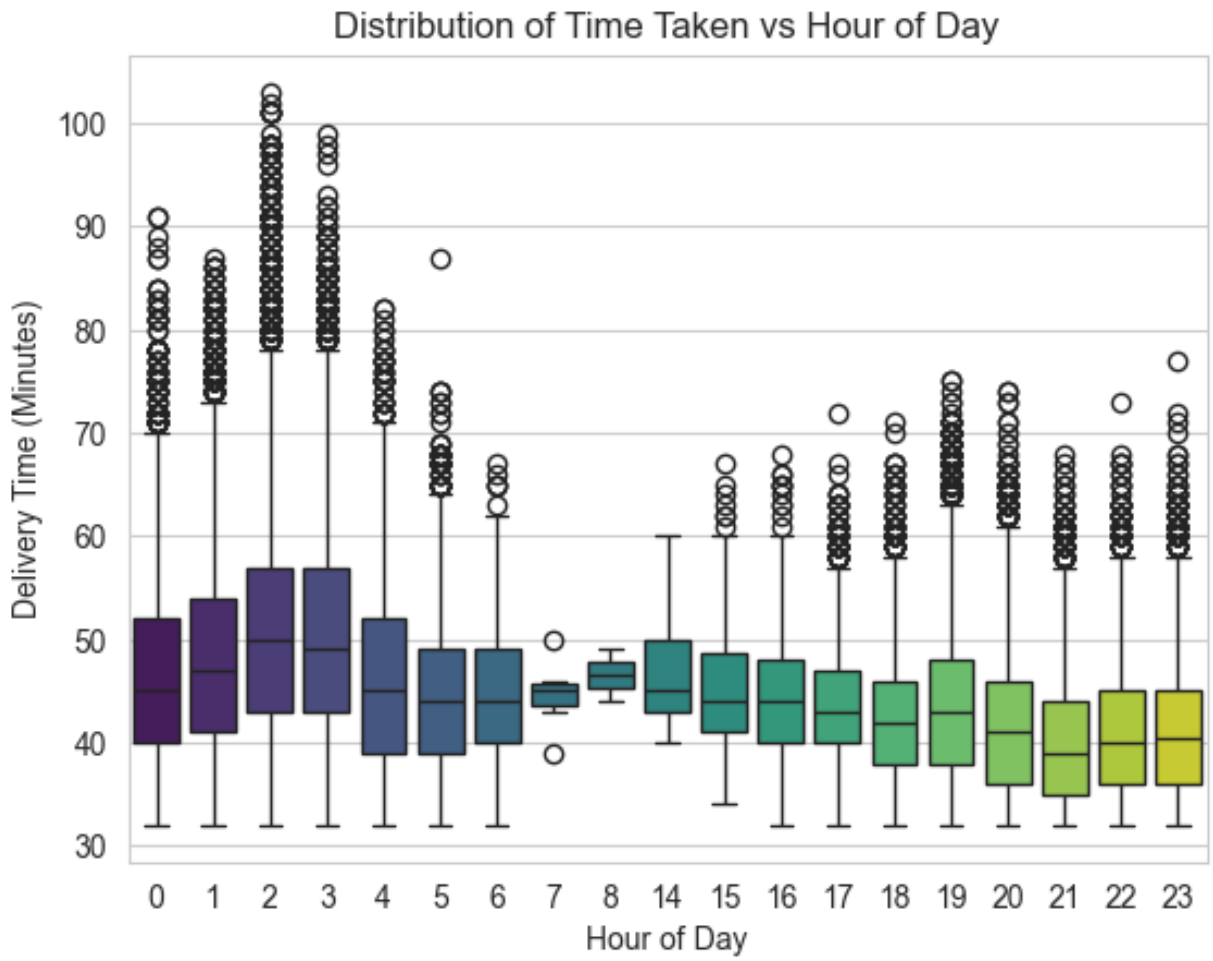
4. Hour of the day distribution strongly suggests typical dinner and late night as the busiest periods for the service. The small early morning peak could be related to breakfast orders.

3.2. Relationships Between Features

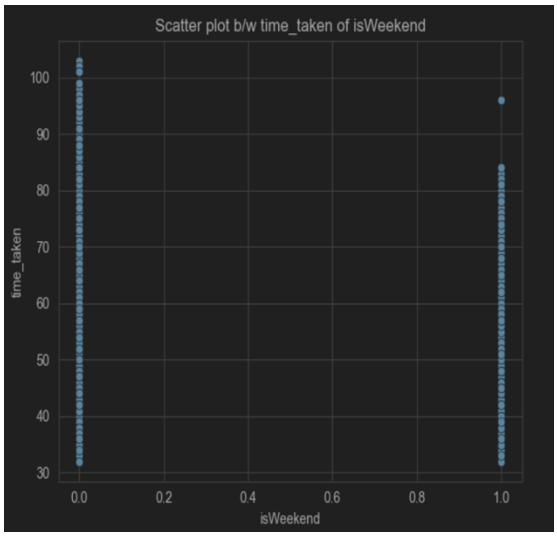
3.2.1. Scatter plots for important numerical and categorical features to observe how they relate to `time_taken`

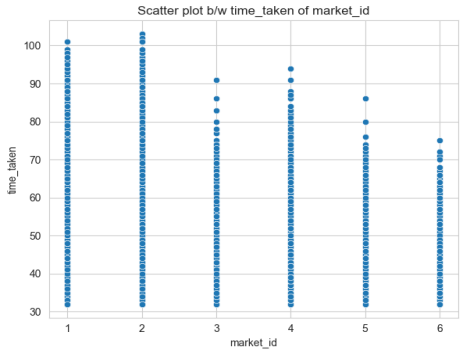
I used scatter plot iterated every variable to plot the scatter plot

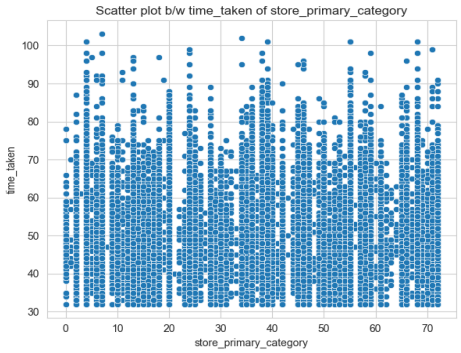
```
n_variables_cols = len(variables_all)
n_cols = 3
n_rows = (n_variables_cols + n_cols - 1) // n_cols
plt.figure(figsize=(n_cols * 6, n_rows * 5))

for i, var in enumerate(variables_all):
    plt.subplot(n_rows, n_cols, i + 1)
    sns.scatterplot(data=porter_train, x=var, y='time_taken')
    plt.title(f'Scatter plot b/w time_taken of {var}')

plt.tight_layout()
plt.show()
```

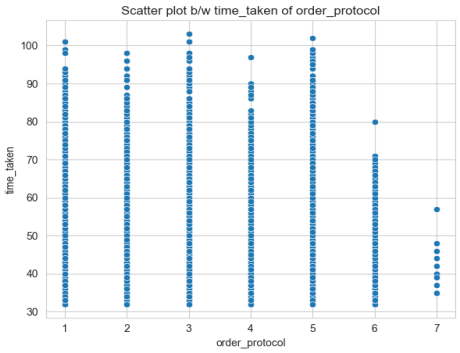


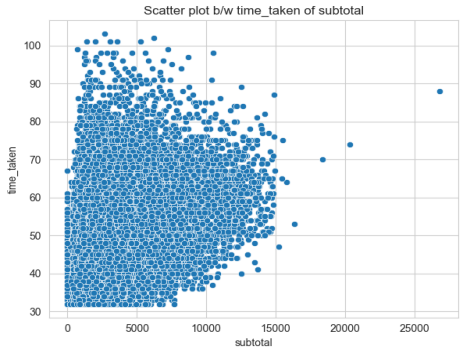
- No much pattern is reconized in store_primary_category, order_protocol, num_distinct_items, hour_of_day, is_weekend since these are categorical, histogram shows more useful info
- As **subtotal** increases the time_taken also increase and there is minimum time_taken (30min) above total of 10000
- Similar to subtotal **total_outstanding_orders** increases time_taken also increase and there is minimum time_taken (30min) above 150 total_outstanding_orders
- As distance increase the time_taken also increases
- **Total_onshit_dasher** and **total_busy_dashers** does not bring any major pattern.
- Time_taken is less in mornings 7-14 and increases especially after midnight 2-4 AM

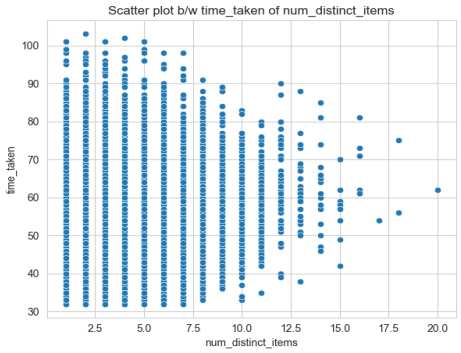


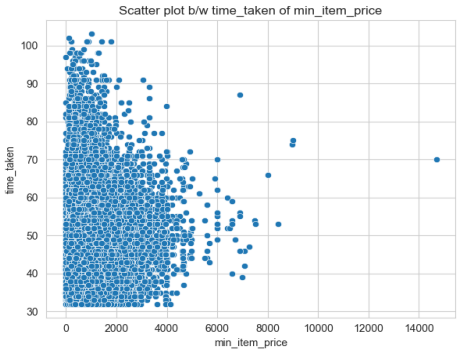

Scatter plot b/w time_taken of market_id
time_taken
market_id

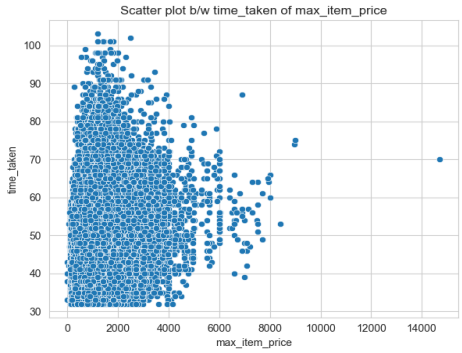

Scatter plot b/w time_taken of store_primary_category
time_taken
store_primary_category

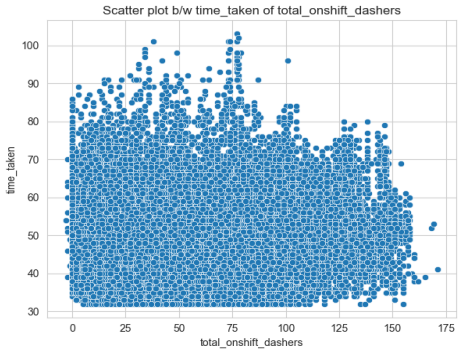

Scatter plot b/w time_taken of order_protocol
time_taken
order_protocol

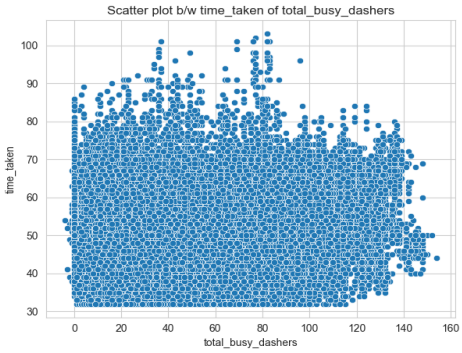

Scatter plot b/w time_taken of subtotal
time_taken
subtotal

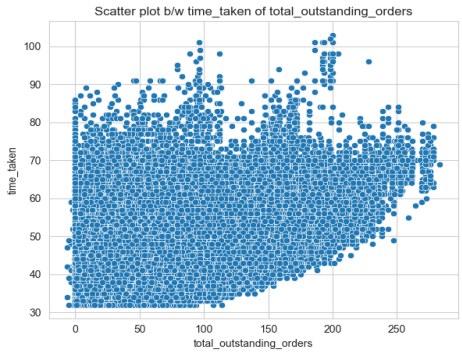

Scatter plot b/w time_taken of num_distinct_items
time_taken
num_distinct_items

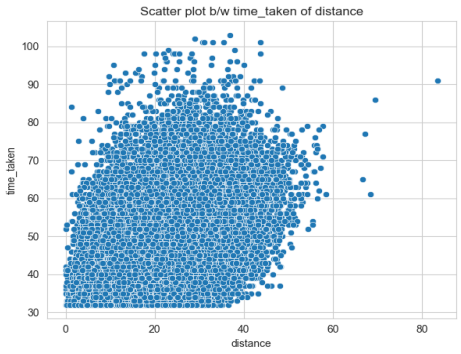

Scatter plot b/w time_taken of min_item_price
time_taken
min_item_price

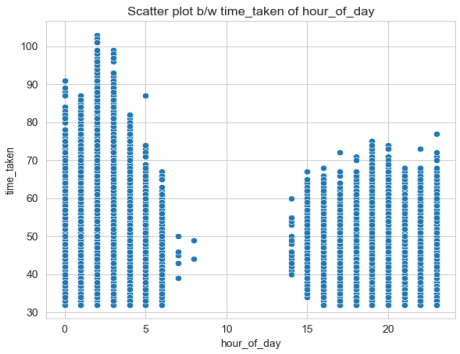

Scatter plot b/w time_taken of max_item_price
time_taken
max_item_price

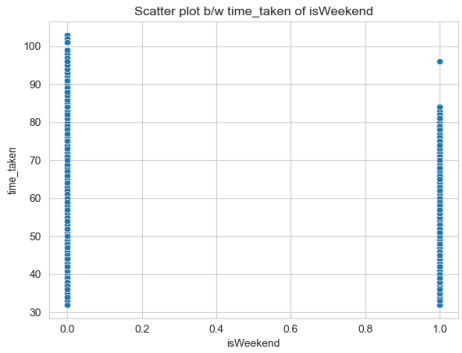

Scatter plot b/w time_taken of total_onshift_dashers
time_taken
total_onshift_dashers


Scatter plot b/w time_taken of total_busy_dashers
time_taken
total_busy_dashers


Scatter plot b/w time_taken of total_outstanding_orders
time_taken
total_outstanding_orders

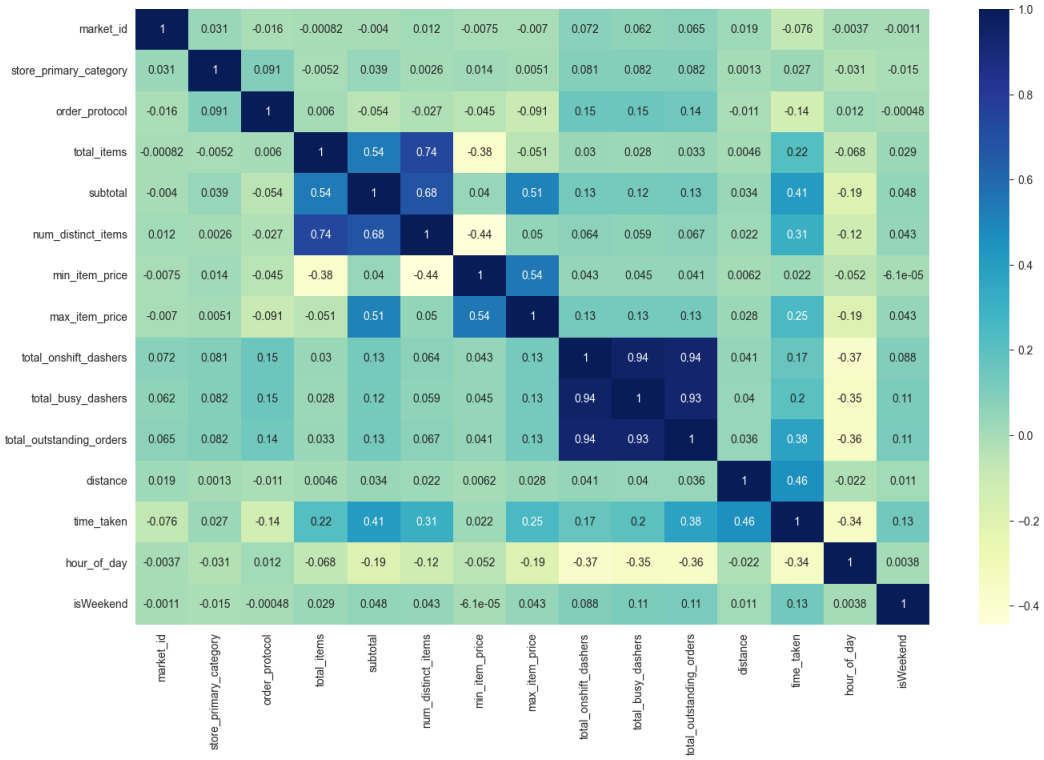

Scatter plot b/w time_taken of distance
time_taken
distance


Scatter plot b/w time_taken of hour_of_day
time_taken
hour_of_day

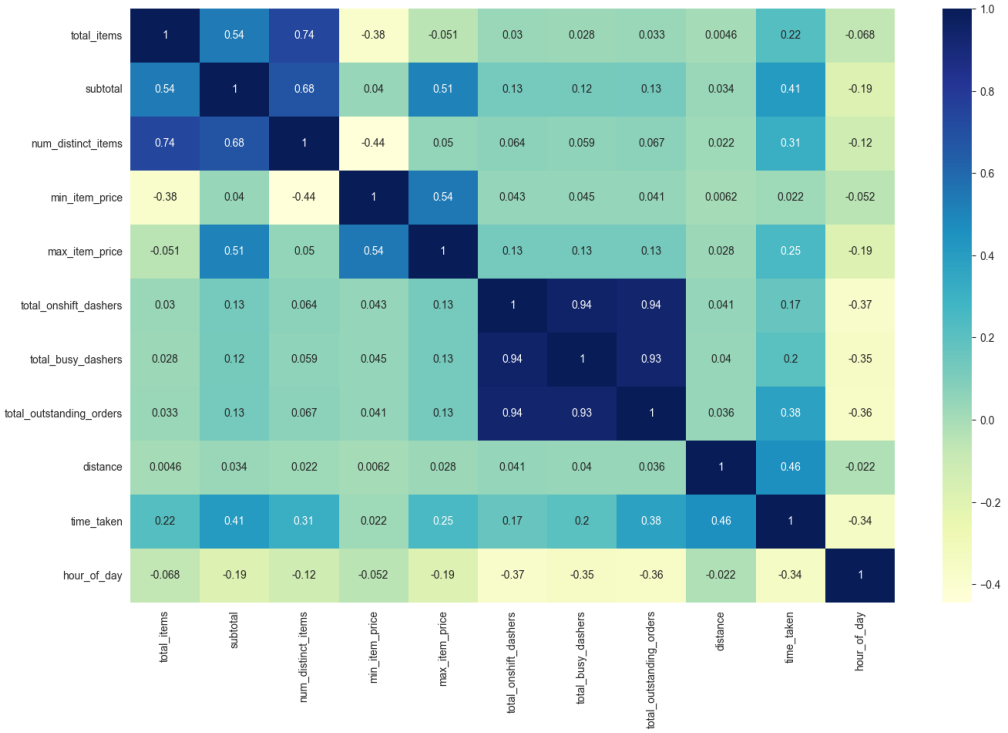

Scatter plot b/w time_taken of isWeekend
time_taken
isWeekend

3.3. Correlation Analysis

3.3.1. Heatmap to display correlation



Heat map correlation with numerical columns:



3.3.2. Drop the columns with target variables

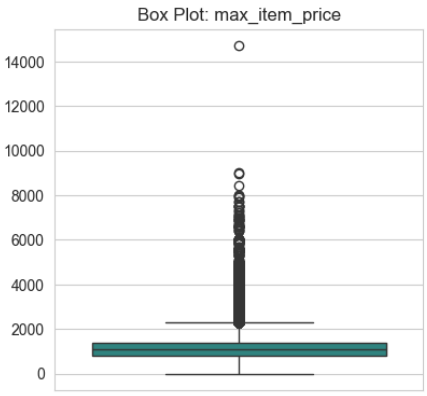
```
time_taken                 1.000000
distance                   0.459712
subtotal                   0.412878
total_outstanding_orders   0.381642
hour_of_day                0.344002
num_distinct_items         0.313384
max_item_price             0.254671
total_items                0.219104
total_busy_dashers         0.202562
total_onshift_dashers      0.166812
order_protocol             0.137906
isWeekend                  0.133896
market_id                  0.075735
store_primary_category     0.027475
min_item_price             0.022281
dtype: float64
```

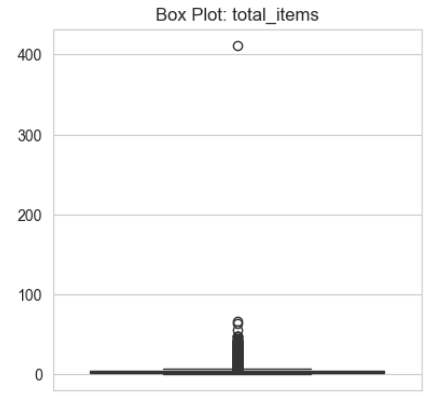
based on the above information dropping the following columns which has weak correlations: ['market_id', 'store_primary_category', 'min_item_price']

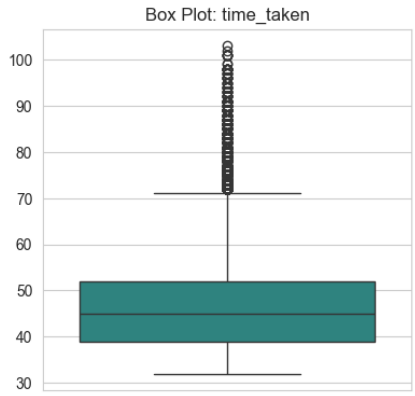
3.4. Handling the Outliers

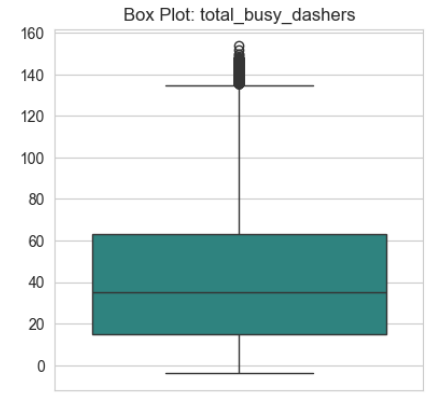
3.4.1. Visualizing the potential outliers

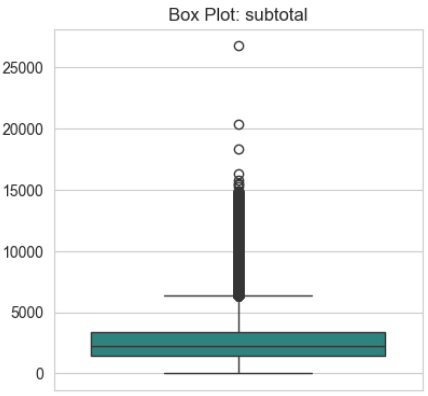
1. time_taken, num_distinct_items, subtotal have visible outliers
2. total_items, max_item_price way out of values which are very clear outlier

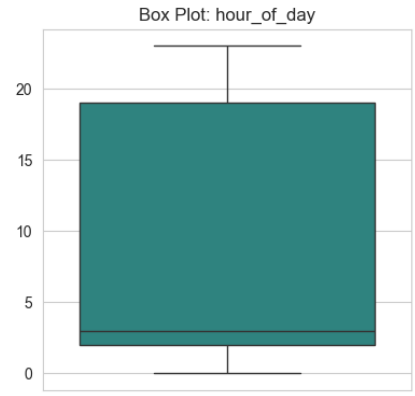

Box Plot: max_item_price
14000
12000
10000
8000
6000
4000
2000
0

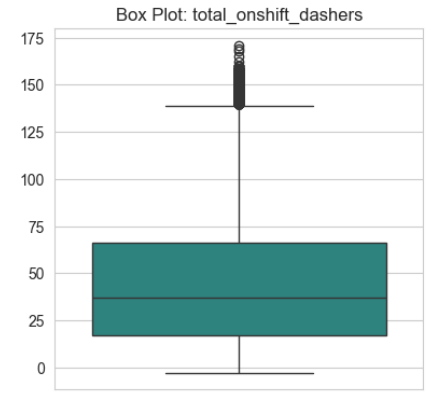

Box Plot: total_items
400
300
200
100
0

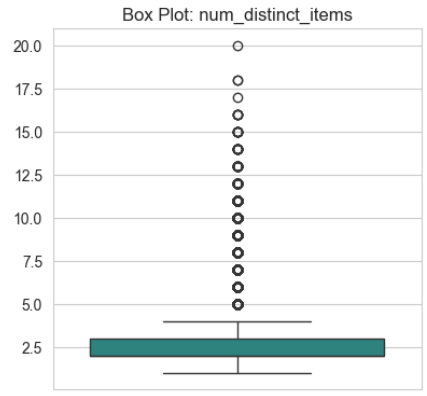

Box Plot: time_taken
100
90
80
70
60
50
40
30

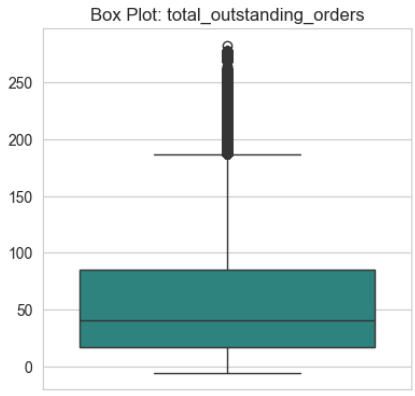

Box Plot: total_busy_dashers
160
140
120
100
80
60
40
20
0

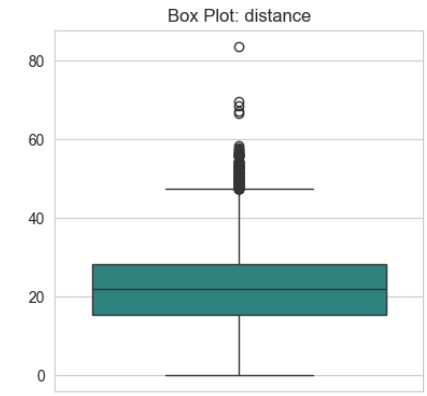

Box Plot: subtotal
25000
20000
15000
10000
5000
0


Box Plot: hour_of_day
20
15
10
5
0


Box Plot: total_onshift_dashers
175
150
125
100
75
50
25
0


Box Plot: num_distinct_items
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5

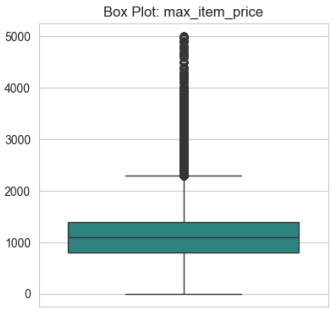

Box Plot: total_outstanding_orders
250
200
150
100
50
0

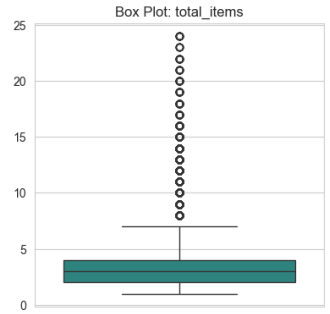

Box Plot: distance
80
60
40
20
0

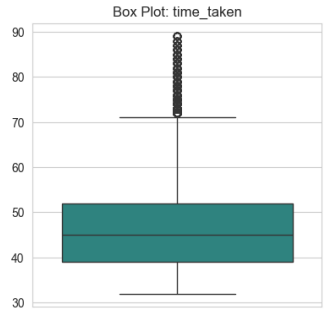
3.4.2. handling Outliers

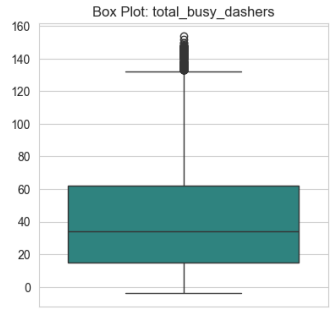
Based on the above info, have take the following condition to eliminate the outlier

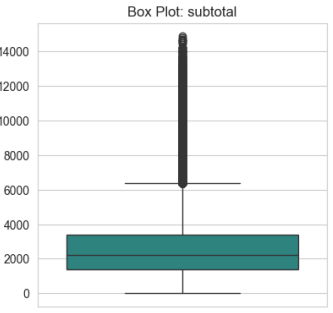
```
(porter_train['distance'] < 60) &
(porter_train['total_items'] < 25) &
(porter_train['max_item_price'] < 5000) &
(porter_train['subtotal'] < 15000) &
(porter_train['total_outstanding_orders'] < 250) &
(porter_train['num_distinct_items'] < 10) &
(porter_train['time_taken'] < 90)
```

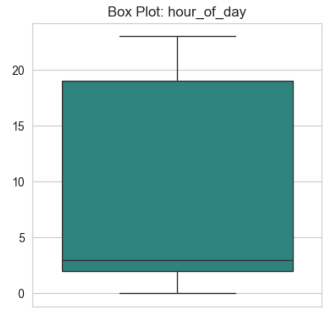


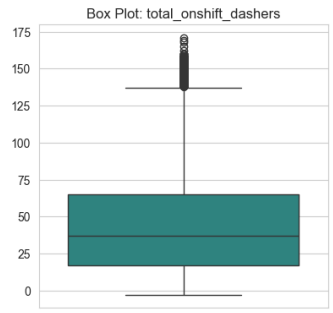


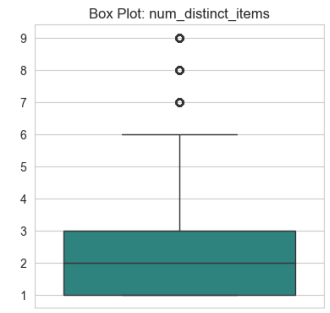


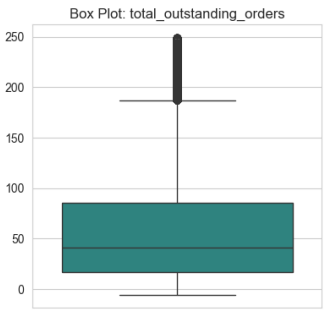


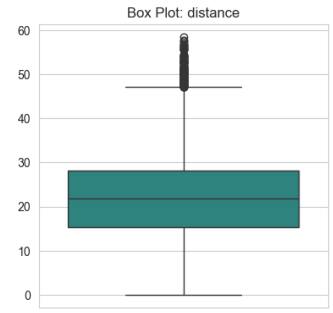












*****Reduced the outlier but not completely removed. I use StandardScalar since there are outliers*****

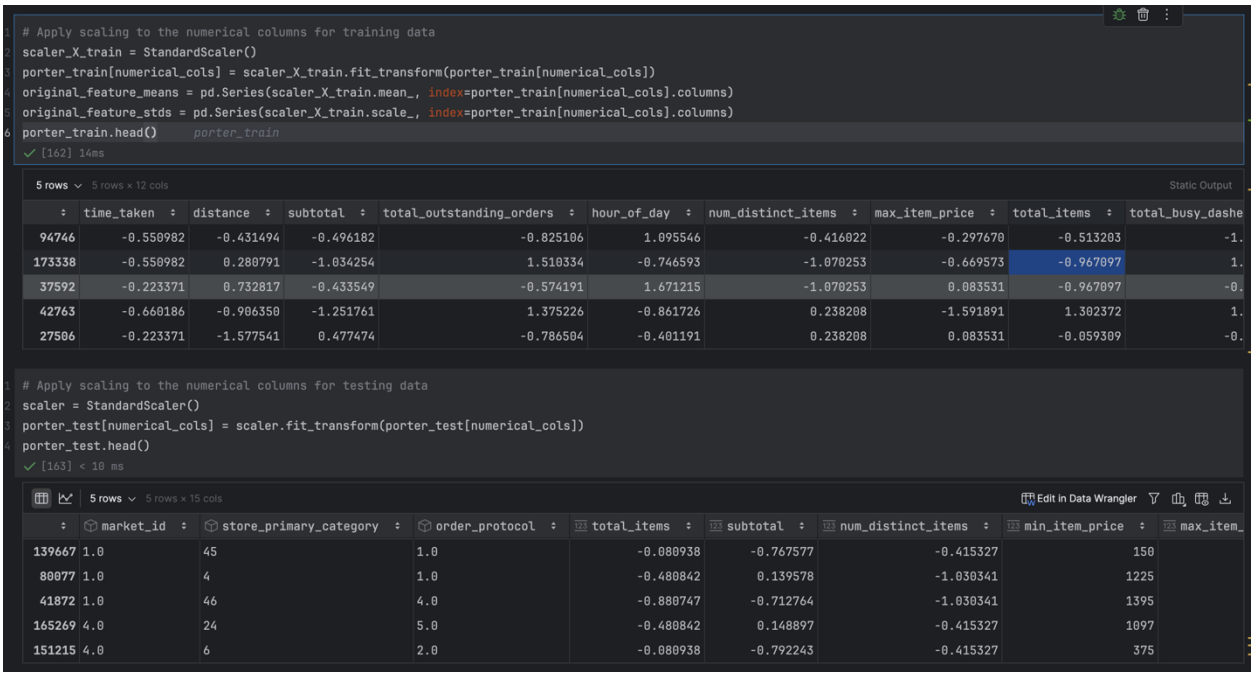
4. Optional

5. Model Building

5.1. Feature Scaling

I used StandardScaler for scaling numerical columns for training and testing data.

```
scaler_X_train = StandardScaler()
porter_train[numerical_cols] =
scaler_X_train.fit_transform(porter_train[numerical_cols])
original_feature_means = pd.Series(scaler_X_train.mean_,
index=porter_train[numerical_cols].columns)
original_feature_stds = pd.Series(scaler_X_train.scale_,
index=porter_train[numerical_cols].columns)
porter_train.head()
```



```
# Apply scaling to the numerical columns for training data
scaler_X_train = StandardScaler()
porter_train[numerical_cols] = scaler_X_train.fit_transform(porter_train[numerical_cols])
original_feature_means = pd.Series(scaler_X_train.mean_, index=porter_train[numerical_cols].columns)
original_feature_stds = pd.Series(scaler_X_train.scale_, index=porter_train[numerical_cols].columns)
porter_train.head()     porter_train
[162] 14ms
```

5 rows · 5 rows × 12 cols — Static Output

	time_taken	distance	subtotal	total_outstanding_orders	hour_of_day	num_distinct_items	max_item_price	total_items	total_busy_dashe
94746	-0.550982	-0.431494	-0.496182	-0.825106	1.095546	-0.416022	-0.297670	-0.513203	-1.
173338	-0.550982	0.280791	-1.034254	1.510334	-0.746593	-1.070253	-0.669573	-0.967097	1.
37592	-0.223371	0.732817	-0.433549	-0.574191	1.671215	-1.070253	0.083531	-0.967097	-0.
42763	-0.660186	-0.906350	-1.251761	1.375226	-0.861726	0.238208	-1.591891	1.302372	1.
27506	-0.223371	-1.577541	0.477474	-0.786504	-0.401191	0.238208	0.083531	-0.059309	-0.

```
# Apply scaling to the numerical columns for testing data
scaler = StandardScaler()
porter_test[numerical_cols] = scaler.fit_transform(porter_test[numerical_cols])
porter_test.head()
[163] < 10 ms
```

5 rows · 5 rows × 15 cols — Edit in Data Wrangler

	market_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_
139667	1.0	45	1.0	-0.080938	-0.767577	-0.415327	150	
80077	1.0	4	1.0	-0.480842	0.139578	-1.030341	1225	
41872	1.0	46	4.0	-0.880747	-0.712764	-1.030341	1395	
165269	4.0	24	5.0	-0.480842	0.148897	-0.415327	1097	
151215	4.0	6	2.0	-0.080938	-0.792243	-0.415327	375	

5.2. Build a linear regression model

I added the constants and run OLS model with summary

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             time_taken   R-squared:                       0.871
Model:                            OLS   Adj. R-squared:                  0.871
Method:                 Least Squares   F-statistic:                 7.461e+04
Date:                Mon, 28 Jul 2025   Prob (F-statistic):               0.00
Time:                        18:26:46   Log-Likelihood:                -48331.
No. Observations:              121943   AIC:                         9.669e+04
Df Residuals:                  121931   BIC:                         9.680e+04
Df Model:                          11
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            0.1621      0.002     67.489      0.000       0.157       0.167
distance                         0.4537      0.001    439.773      0.000       0.452       0.456
subtotal                         0.2455      0.002    125.820      0.000       0.242       0.249
total_outstanding_orders         1.9681      0.003    602.448      0.000       1.962       1.975
hour_of_day                     -0.2362      0.001   -208.042      0.000      -0.238      -0.234
num_distinct_items               0.0873      0.002     43.163      0.000       0.083       0.091
max_item_price                   0.0380      0.001     25.759      0.000       0.035       0.041
total_items                     -0.0103      0.002     -5.065      0.000      -0.014      -0.006
total_busy_dashers              -0.4924      0.003   -143.417      0.000      -0.499      -0.486
total_onshift_dashers           -1.3629      0.004   -384.515      0.000      -1.370      -1.356
order_protocol                  -0.0719      0.001   -103.441      0.000      -0.073      -0.071
isWeekend                        0.1372      0.002     62.681      0.000       0.133       0.142
==============================================================================
Omnibus:                    41937.092   Durbin-Watson:                   2.003
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           181217.440
Skew:                           1.651   Prob(JB):                         0.00
Kurtosis:                       7.976   Cond. No.                         14.5
==============================================================================

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

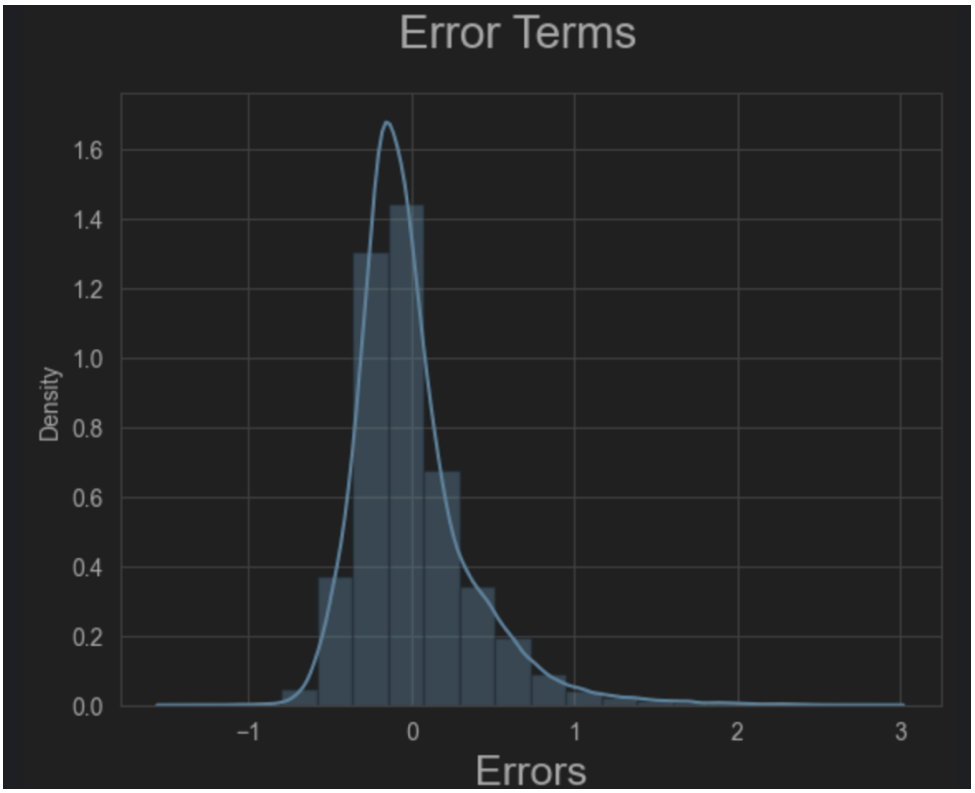
- r-squared is 87.1 which is good
- F-Sta is 7.461e+04 which is high also good
- All the variables have P>|t| as 0.000 which means all the variable contribute to the effect in non randomness manner.
- Certain varaibles has positive and negative coeefficients

Train the model and predict:

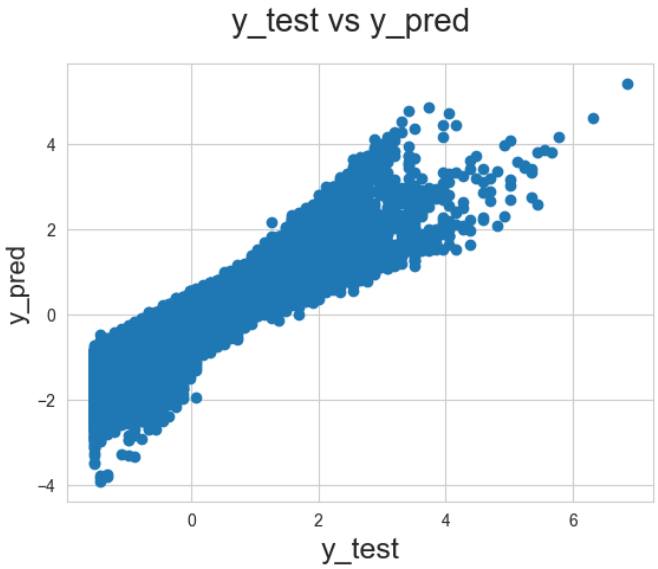
```
# Train the model using the training data
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Evaluation metrics:

- R2 between test data and predicted data: 0.8737992135500233
- Mean absolute error between test data and predicted data: 0.2551228125215107



- Errors meaned near zero,
- Little stretched to the right



- It is scattered around the fitted line and little fan in and fan out model
- It converges the top values

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

By using RFE, I try to find a subset of 9 features that are considered the most impactful predictors of time_taken according to the LinearRegression model's internal metrics

```
# Importing RFE and LinearRegression
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

# Running RFE with the output number of the variable equal to 10
lm = LinearRegression()
lm.fit(X_train, y_train)

rfe = RFE(lm, n_features_to_select=9)
rfe = rfe.fit(X_train, y_train)
print(list(zip(X_train.columns, rfe.support_, rfe.ranking_)))
print(X_train.columns[rfe.support_])
```

Finally I get the following 9 important features and running the OLS :
['distance', 'subtotal', 'total_outstanding_orders', 'hour_of_day', 'num_distinct_items', 'total_busy_dashers', 'total_onshift_dashers', 'order_protocol', 'isWeekend']

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             time_taken   R-squared:                       0.855
Model:                            OLS   Adj. R-squared:                  0.855
Method:                 Least Squares   F-statistic:                 7.985e+04
Date:                Mon, 28 Jul 2025   Prob (F-statistic):               0.00
Time:                        18:26:47   Log-Likelihood:                -55314.
No. Observations:              121943   AIC:                         1.106e+05
Df Residuals:                  121933   BIC:                         1.107e+05
Df Model:                           9
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                        2.766e-16      0.001   2.54e-13      1.000      -0.002       0.002
distance                        0.4556      0.001    417.111      0.000       0.454       0.458
subtotal                        0.2450      0.002    118.546      0.000       0.241       0.249
total_outstanding_orders        1.9854      0.003    574.929      0.000       1.979       1.992
hour_of_day                    -0.2388      0.001   -199.155      0.000      -0.241      -0.236
num_distinct_items              0.0981      0.002     45.852      0.000       0.094       0.102
max_item_price                  0.0515      0.002     32.994      0.000       0.048       0.055
total_items                    -0.0154      0.002     -7.111      0.000      -0.020      -0.011
total_busy_dashers             -0.4997      0.004   -137.665      0.000      -0.507      -0.493
total_onshift_dashers          -1.3866      0.004   -370.173      0.000      -1.394      -1.379
==============================================================================
Omnibus:                    30562.330   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            98793.542
Skew:                           1.273   Prob(JB):                         0.00
Kurtosis:                       6.601   Cond. No.                         7.61
==============================================================================

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

- **Dependent Variable**: time_taken: Confirms that the model is attempting to predict delivery time.

- **R-squared: 0.855:** Very high R-squared value meaning approximately 85.5% of the variance in delivery time can be explained by the 9 independent variables included in your model. This indicates a strong predictive capability.

- **Adjusted R-squared**: 0.855: Identical to the R-squared, which is expected with a large number of observations and a well-chosen set of predictors. It further confirms the strength of the model, indicating that the features are contributing meaningfully.

- **F-statistic:** 7.985e+04 / Prob (F-statistic): 0.00: Large F-statistic and the p-value of 0.00 (effectively zero) mean that the model is highly statistically significant.

VIF of the independent variable to confirm the multicollinearity:

	Features	VIF
8	total_onshift_dashers	11.84
7	total_busy_dashers	11.09
2	total_outstanding_orders	10.06
6	total_items	3.93
4	num_distinct_items	3.86
1	subtotal	3.59
5	max_item_price	2.05
9	order_protocol	1.39
10	isWeekend	1.39
3	hour_of_day	1.21

- **total_onshift_dashers has impact on other variable it is confirmed when it is removed from the model calculation and the r2 reduces to 71.4% from 85.5 %**
- **further when total_busy_dashers is removed r2 score reduces further to 52.5 % from 85.5% so these two variables are important**

Removing total_onshift_dashers:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             time_taken   R-squared:                       0.714
Model:                            OLS   Adj. R-squared:                  0.714
Method:                 Least Squares   F-statistic:                 3.041e+04
Date:                Mon, 28 Jul 2025   Prob (F-statistic):               0.00
Time:                        18:26:47   Log-Likelihood:                -96752.
No. Observations:              121943   AIC:                         1.935e+05
Df Residuals:                  121932   BIC:                         1.936e+05
Df Model:                          10
Covariance Type:            nonrobust
============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
const                        0.1684      0.004     47.139      0.000       0.161       0.175
distance                     0.4478      0.002    291.832      0.000       0.445       0.451
subtotal                     0.2439      0.003     84.002      0.000       0.238       0.250
total_outstanding_orders     1.3797      0.004    321.358      0.000       1.371       1.388
hour_of_day                 -0.1931      0.002   -114.870      0.000      -0.196      -0.190
num_distinct_items           0.0817      0.003     27.157      0.000       0.076       0.088
max_item_price               0.0368      0.002     16.763      0.000       0.033       0.041
total_items                 -0.0047      0.003     -1.562      0.118      -0.011       0.001
total_busy_dashers          -1.2131      0.004   -283.503      0.000      -1.222      -1.205
order_protocol              -0.0797      0.001    -77.125      0.000      -0.082      -0.078
isWeekend                    0.1849      0.003     56.860      0.000       0.178       0.191
==============================================================================
Omnibus:                    12234.451   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            32865.037
Skew:                           0.570   Prob(JB):                         0.00
Kurtosis:                       5.273   Cond. No.                         13.3
==============================================================================

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

Removing total_busy_dashers:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             time_taken   R-squared:                       0.525
Model:                            OLS   Adj. R-squared:                  0.525
Method:                 Least Squares   F-statistic:                 1.498e+04
Date:                Mon, 28 Jul 2025   Prob (F-statistic):               0.00
Time:                        18:26:48   Log-Likelihood:            -1.2762e+05
No. Observations:              121943   AIC:                         2.553e+05
Df Residuals:                  121933   BIC:                         2.554e+05
Df Model:                           9
Covariance Type:            nonrobust
============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
const                        0.2330      0.005     50.739      0.000       0.224       0.242
distance                     0.4397      0.002    222.493      0.000       0.436       0.444
subtotal                     0.2368      0.004     63.321      0.000       0.229       0.244
total_outstanding_orders     0.2600      0.002    119.907      0.000       0.256       0.264
hour_of_day                 -0.1770      0.002    -81.810      0.000      -0.181      -0.173
num_distinct_items           0.0879      0.004     22.669      0.000       0.080       0.095
max_item_price               0.0287      0.003     10.152      0.000       0.023       0.034
total_items               4.663e-05      0.004      0.012      0.990      -0.008       0.008
order_protocol              -0.1008      0.001    -75.914      0.000      -0.103      -0.098
isWeekend                    0.1756      0.004     41.941      0.000       0.167       0.184
==============================================================================
Omnibus:                    10665.257   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            19254.321
Skew:                           0.620   Prob(JB):                         0.00
Kurtosis:                       4.501   Cond. No.                         8.99
==============================================================================

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

5.4. Final Model

Final model was selected with following variables

```
selected_variables = ['distance', 'subtotal', 'total_outstanding_orders',
'hour_of_day','num_distinct_items', 'max_item_price', 'total_items',
'total_busy_dashers', 'total_onshift_dashers']
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             time_taken   R-squared:                       0.855
Model:                            OLS   Adj. R-squared:                  0.855
Method:                 Least Squares   F-statistic:                 7.985e+04
Date:                Mon, 28 Jul 2025   Prob (F-statistic):               0.00
Time:                        18:26:47   Log-Likelihood:                -55314.
No. Observations:              121943   AIC:                         1.106e+05
Df Residuals:                  121933   BIC:                         1.107e+05
Df Model:                           9
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                        2.766e-16      0.001   2.54e-13      1.000      -0.002       0.002
distance                        0.4556      0.001    417.111      0.000       0.454       0.458
subtotal                        0.2450      0.002    118.546      0.000       0.241       0.249
total_outstanding_orders        1.9854      0.003    574.929      0.000       1.979       1.992
hour_of_day                    -0.2388      0.001   -199.155      0.000      -0.241      -0.236
num_distinct_items              0.0981      0.002     45.852      0.000       0.094       0.102
max_item_price                  0.0515      0.002     32.994      0.000       0.048       0.055
total_items                    -0.0154      0.002     -7.111      0.000      -0.020      -0.011
total_busy_dashers             -0.4997      0.004   -137.665      0.000      -0.507      -0.493
total_onshift_dashers          -1.3866      0.004   -370.173      0.000      -1.394      -1.379
==============================================================================
Omnibus:                    30562.330   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            98793.542
Skew:                           1.273   Prob(JB):                         0.00
Kurtosis:                       6.601   Cond. No.                         7.61
==============================================================================

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

Interpretation of Coefficients (coef).

1. **total_outstanding_orders (Coef: 1.9854): Most impactful positive predictor.** A one standard deviation increase in outstanding orders leads to nearly a 2 standard deviation increase in delivery time. Strong evidence of system congestion.
2. **total_onshift_dashers (Coef: -1.3866): Most impactful negative predictor.** A one standard deviation increase in on-shift dashers leads to a 1.39 standard deviation *decrease* in delivery time. Crucial for operational efficiency.
3. **total_busy_dashers (Coef: -0.4997):** Very strong negative impact. A one standard deviation increase in busy dashers reduces delivery time by 0.5 standard deviations.
4. **distance (Coef: 0.4556):** Significant positive impact. A one standard deviation increase in distance adds 0.46 standard deviations to delivery time.

5. **hour_of_day (Coef: -0.2388):** Noticeably larger negative impact than previously seen (-0.1016). This could indicate that transforming hour_of_day or other correlated variables has clarified its linear effect, where later hours correspond to faster delivery times.
6. **subtotal (Coef: 0.2450):** Significant positive impact. Larger subtotals associated with longer times.
7. **num_distinct_items (Coef: 0.0981):** Positive impact. More distinct items slightly increase time.
8. **max_item_price (Coef: 0.0515):** Positive impact. Higher priced items slightly increase time.
9. **total_items (Coef: -0.0154):** Smallest absolute impact. More total items slightly *decrease* delivery time.

6. Performance Residual Analysis

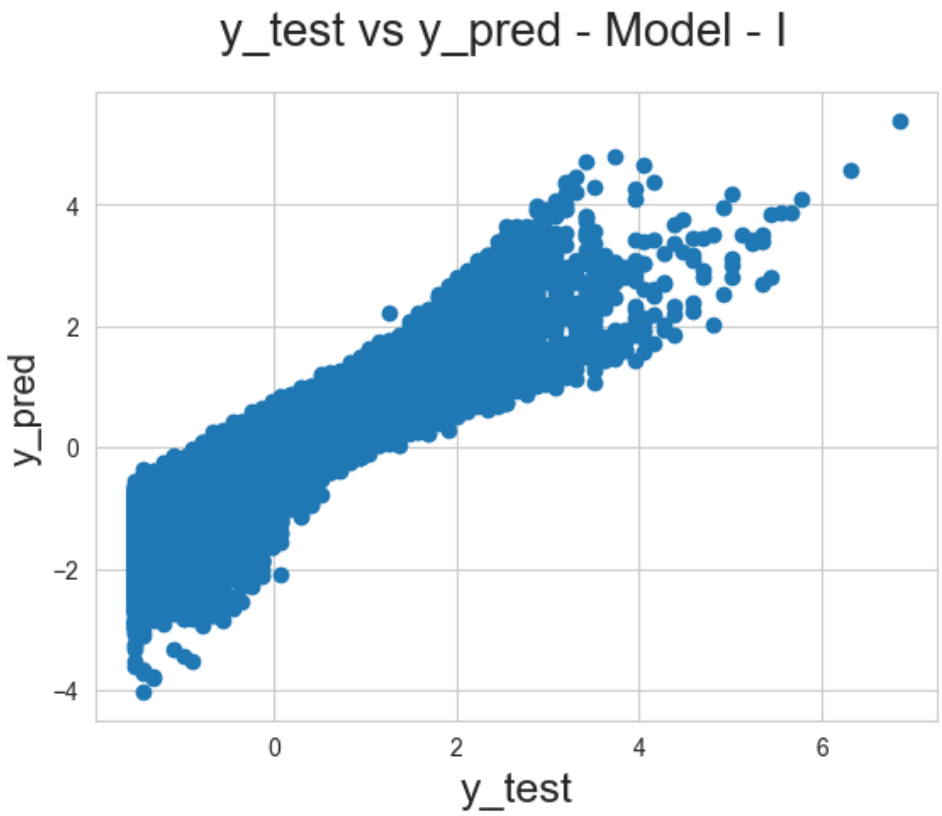
6.1. Performance Residual Analysis

6.1.1. Error Terms

Inference from Residuals Histogram (Error Terms)

- The distribution is cantered very close to zero, which is ideal for regression models, indicating that the model is not systematically over or under-predicting.
- It is nearly bell curve, however it extends to the positive side.the mean error is nearly zero, however little to the negative side.
- **Right skew** (positive skew) in the residuals with the tail extending further to the right (positive errors) than to the left (negative errors). This may tends that model may under-predict more often

6.1.2. Actual vs Predicted



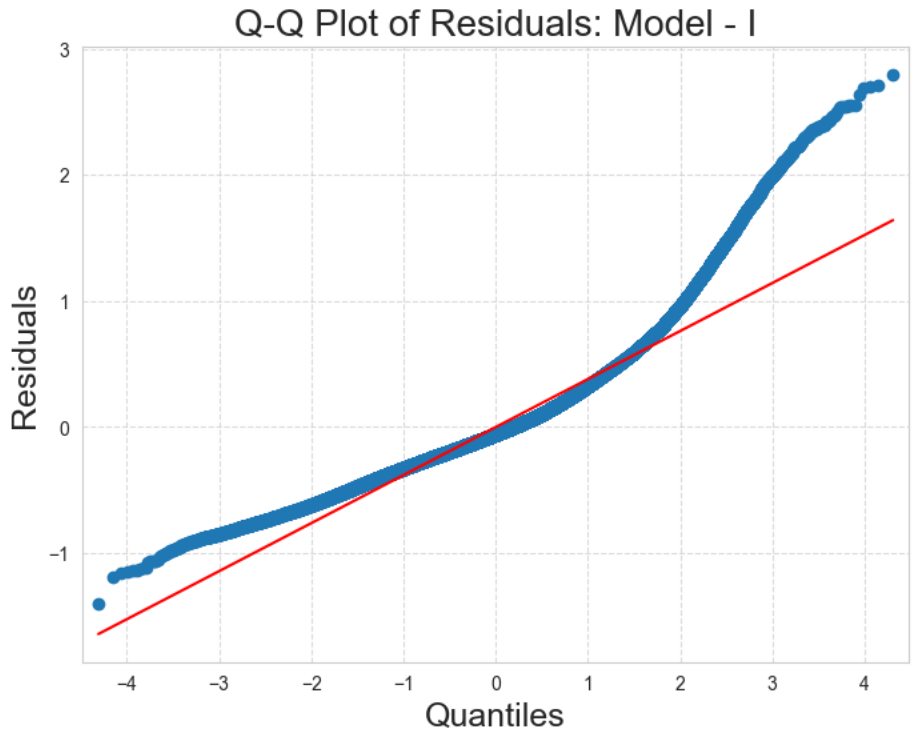
Inference from Plotting y_test and y_pred (Residuals vs. Predicted Values) -

- It clearly says most of the variables
- The residuals are not randomly scattered around the horizontal line.
- The graph is not as scatted when compared to ideal scenario.
- 'total_onshift_dashers'. This is a tarde-off that we are gonna take for prediction
- Heteroscedasticity is present, meaning the variance of the errors is not constant across all levels of the predicted or actual values.

6.1.3. Q_Q Plot

Inference from Q-Q (Quantile-Quantile) plot

- The residuals are not perfectly normally distributed, particularly in the tails. The presence of larger positive residuals (under-predictions) is a concern and aligns with the skew observed in the error distribution plot.
- non Normality of residuals at the start and end of the line.



6.1.4. Variance Inflation Factor (VIF)

- From the Variance Inflation Factor (VIF) it is very clear that "total_onshift_dashers" has more multicollinearity once we remove the r2 comes to 71.4% and once we remove "total_busy_dashers" it comes around 52.5%
- These two fields total_onshift_dashers, total_busy_dashers has high multi collinearity and influences other variables as well.

6.2. Perform coefficient Analysis

6.2.1. Perform coefficient analysis to find how changes in features affect the target. Also, the features were scaled, so interpret the scaled and unscaled coefficients to understand the impact of feature changes on delivery time.

```
7.  # Compare the scaled vs unscaled features used in the final model

    scaled_coefs = lm_final.params.drop('const')
    scaled_intercept = lm_final.params['const']

    unscaled_feature_coefs = {}
    sum_for_unscaled_intercept = 0

    for feature, scaled_b in scaled_coefs.items():
        unscaled_b = scaled_b / original_feature_stds[feature]
        unscaled_feature_coefs[feature] = unscaled_b
        sum_for_unscaled_intercept += (unscaled_b *
    original_feature_means[feature])
        print(f"Coef for {feature} Scaled: {scaled_b:.2f} Unscaled:
    {unscaled_b:.2f}")
```

Feature	Standardized Coefficient	Impact Direction
total_outstanding_orders	+1.98	Positive
total_onshift_dashers	-1.39	Negative
total_busy_dashers	-0.50	Negative
distance	+0.46	Positive
subtotal	+0.25	Positive
hour_of_day	-0.24	Negative
num_distinct_items	+0.10	Positive
max_item_price	+0.05	Positive
total_items	-0.02	Negative

```
Coef for distance Scaled: 0.46 Unscaled: 0.05
Coef for subtotal Scaled: 0.24 Unscaled: 0.00
Coef for total_outstanding_orders Scaled: 1.99 Unscaled: 0.04
Coef for hour_of_day Scaled: -0.24 Unscaled: -0.03
Coef for num_distinct_items Scaled: 0.10 Unscaled: 0.06
Coef for max_item_price Scaled: 0.05 Unscaled: 0.00
Coef for total_items Scaled: -0.02 Unscaled: -0.01
Coef for total_busy_dashers Scaled: -0.50 Unscaled: -0.02
Coef for total_onshift_dashers Scaled: -1.39 Unscaled: -0.04
```

6.3.1. Analyze the effect of a unit change in a feature, say 'total_items'

- For every 1 km increase in distance, time_taken is expected to increase by approximately 0.46 minutes
- For every 1 price increase in subtotal, time_taken is expected to increase by approximately 0.24 minutes
- For every 1 order increase in total_outstanding_orders, time_taken is expected to increase by approximately 1.99 minutes
- For every 1 hour increase in hour_of_day, time_taken is expected to decrease by approximately -0.24 minutes
- For every 1 item increase in num_distinct_items, time_taken is expected to increase by approximately 0.10 minutes
- For every 1 price increase in max_item_price, time_taken is expected to increase by approximately 0.05 minutes
- For every 1 item increase in total_items, time_taken is expected to decrease by approximately -0.02 minutes
- For every 1 dasher increase in total_busy_dashers, time_taken is expected to decrease by approximately -0.50 minutes
- For every 1 dasher increase in total_onshift_dashers, time_taken is expected to decrease by approximately -1.39 minutes

```
For every 1 km increase in distance, time_taken is expected to increase by approximately 0.46 minutes
For every 1 price increase in subtotal, time_taken is expected to increase by approximately 0.24 minutes
For every 1 order increase in total_outstanding_orders, time_taken is expected to increase by approximately 1.99 minutes
For every 1 hour increase in hour_of_day, time_taken is expected to decrease by approximately -0.24 minutes
For every 1 item increase in num_distinct_items, time_taken is expected to increase by approximately 0.10 minutes
For every 1 price increase in max_item_price, time_taken is expected to increase by approximately 0.05 minutes
For every 1 item increase in total_items, time_taken is expected to decrease by approximately -0.02 minutes
For every 1 dasher increase in total_busy_dashers, time_taken is expected to decrease by approximately -0.50 minutes
For every 1 dasher increase in total_onshift_dashers, time_taken is expected to decrease by approximately -1.39 minutes
```

Subjective Questions based on Assignment

Question 1: 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?

- The categrorical columns are 'market_id', 'store_primary_category', 'order_protocol', 'isWeekend' has detectable effect however weak in linear correlation with time_taken. Since they have low correlation with time_taken - 0.076, 0.027, -0.14, 0.13 respectively. And in the Recursive Feature Elimination (RFE) the numerical column have more contribution on the time_taken than any other categorical variables.
- P > |t| values of all the categorical values are 0.000 meaning that non-random effect on the model

Question 2: What does `test_size = 0.2` refer to during splitting the data into training and test sets?

test_size =0.2 means to split 20% of the total data as test data. And keep other 80% as training data.

Question 3:Looking at the heatmap, which one has the highest correlation with the target variable?

- "total_onshift_dashers" and "total_busy_dashers" => 0.94
- "total_onshift_dashers" and "total_outstanding_orders" => 0.94
- "total_outstanding_orders" and "total_busy_dashers" => 0.93
- "num_distinct_items" and "total_items" => 0.74

Question 4:What was your approach to detect the outliers? How did you address them?

> I used boxplot to identify the outliers and I removed the values which are way out of line
> I arrived an optimal threshold in accepting few outliers on individual features, so that I can use StandardScaler

```
    ('distance' < 60)
    ('total_items' < 25)
    ('max_item_price' < 5000)
    ('subtotal' < 15000)
    ('total_outstanding_orders' < 250)
    ('num_distinct_items' < 10)
    ('time_taken' < 90)
```

Question 5: Based on the final model, which are the top 3 features significantly affecting the delivery time?

> The top three significantly affecting delivery time in the decreasing order, first being the most significant
- total_outstanding_orders -> positive correlation
- total_onshift_dashers -> negative correlation
- total_busy_dashers -> negative correlation

Question 6:Explain the linear regression algorithm in detail

> It is supervised predictive analysis model which establishes between one scaler (dependant) variable and one or more independant variables.
> Since it is supervised learning you label the training data and feed the model
> It is a fundamental and most used model for predicting is a continuous variable, e.g. scores of a student, time_taken in delivery.
> It attempts to draw a "best-fit" straight line through the data points.
> There are two main types:
- Simple Linear Regression (SLR): Involves only one independent variable.
- Multiple Linear Regression (MLR): Involves two or more independent variables
> Linear regression guarantes interpolation then extrapolation

> There are few assumptions of OLS Linear Regression that must be satisfied
- Linear relationship between X and Y
- Error terms are normally distributed (not X, Y)
- Error terms are independent of each other
- Error terms have constant variance (homoscedasticity)

Question 7:Explain the difference between simple linear regression and multiple linear regression

> Simple Linear Regression (SLR): Involves only one independent variable while
> the equation for SLR is `Y=Beta0 + Beta1X + constant` since there is only one independe variable, only one X with coefficient
> Main purpose of using this to model is to fit a straight-line relationship between two continuous variables and predict the target based on changes in that single feature.
> Use Cases: When only one factor significantly influences the outcome, or for initial exploratory analysis of pairwise relationships

> Multiple Linear Regression (MLR): Involves two or more independent variables
> MLR equat is similar and goes like `Y=Beta0 + Beta1X1 + Beta2X2 + Beta3X3 + .... + BetanXn + constant` where X1 ... Xn are the multiple independent variables
> Main purpose of using this to model the linear relationship between a dependent variable and several predictors, allowing for a more comprehensive and often more accurate prediction by considering multiple influencing factors simultaneously.
> use cases: predicting house prices based on square footage, number of bedrooms, number of bathrooms, and location. Delivery time prediction model is a classic example of MLR.

Question 8:What is the role of the cost function in linear regression, and how is it minimized?

> The main goal of cost function is to predict the errors between the values predicted and the actual vales. This is created to evaluate the model efficiency. And the process involves minimizing the cost function
> Ways to minimize the cost function (RSS)
- Differentiation: e.g Ordinary Least Squares (OLS)
- Gradient Descent Appxoach: e.g R2

> Find the optimized Beta0, Beta1 and use it in the susequent prediction

> Residual Sum of Squares (RSS) will be minimal for the best fit line, and Total sum of squares (TSS) will be higher
> R2 = 1 - (RSS / TSS)

Question 9:Explain the difference between overfitting and underfitting.

> When we hear Model may ‘overfit’ means that Model fits the train set ‘too well’, doesn’t generalize and does not predict well in unknow dataset.
> It might give right data in the training dataset, while in test data it might not perform well.
> It essentially memorizes the training data instaed of generalizing
> Common cause : Complex model, noisy, insufficient data set, making it too much perfect.

> it is just opposite of overfit, Underfitting means the model is too simple to capture relationships in the training data. It fails to learn from the data adequately.
> When if we have the right data in the training dataset, the model does understand the varies varibles and contribute
> Common cause : Insufficient features, model is simple, Over-regularization

Question 10:How do residual plots help in diagnosing a linear regression model?

> Residual plots are used for diagnosing linear regression models. They visually represent the errors (residuals) of the model, allowing to check whether the underlying assumptions of Ordinary Least Squares (OLS) regression hold true.
Residuals vs. Predicted Values Plot

> A random scatter of points around the horizontal line at zero
> Common Problems (Patterns) to help understand the model

- Curved/Non-linear Pattern (e.g., U-shape, inverted U-shape): Indicates that the relationship between the variables is not linear, and a linear model is likely missing an important curve or trend. The model might be biased
- Funnel/Cone Shape: Indicates heteroscedasticity The spread of the residuals either increases or decreases as the predicted values change.