Report: **LR_Delivery_Time_Prediction**

1.Data Understanding: Before starting the dataset, need to understand the criteria and requirments.

Data Understanding

The dataset contains information on orders placed through Porter, with the following columns:

Field	Description
market_id	Integer ID representing the market where the restaurant is located.
created_at	Timestamp when the order was placed.
actual_delivery_time	Timestamp when the order was delivered.
store_primary_category	Category of the restaurant (e.g., fast food, dine-in).
order_protocol	Integer representing how the order was placed (e.g., via Porter, call to restaurant, etc.).
total_items	Total number of items in the order.
subtotal	Final price of the order.
num_distinct_items	Number of distinct items in the order.
min_item_price	Price of the cheapest item in the order.
max_item_price	Price of the most expensive item in the order.
total_onshift_dashers	Number of delivery partners on duty when the order was placed.
total_busy_dashers	Number of delivery partners already occupied with other orders.
total_outstanding_orders	Number of orders pending fulfillment at the time of the order.
distance	Total distance from the restaurant to the customer.

1.1 Load the data: Import the data frame in the file

Load porter_data_1.csv as a DataFrame

```
]: # Importing the file porter_data_1.csv
import pandas as pd

# Load the CSV file into a DataFrame
df = pd.read_csv('porter_data_1.csv')

# Display the first few rows to verify the import
print(df.head())
```

```
   market_id        created_at actual_delivery_time  store_primary_category  \
0          1  06-02-2015 22:24     06-02-2015 23:11                       4
1          2  10-02-2015 21:49     10-02-2015 22:33                      46
2          2  16-02-2015 00:11     16-02-2015 01:06                      36
3          1  12-02-2015 03:36     12-02-2015 04:35                      38
4          1  27-01-2015 02:12     27-01-2015 02:58                      38

   order_protocol  total_items  subtotal  num_distinct_items  min_item_price  \
0               1            4      3441                   4             557
1               2            1      1900                   1            1400
2               3            4      4771                   3             820
3               1            1      1525                   1            1525
4               1            2      3620                   2            1425

   max_item_price  total_onshift_dashers  total_busy_dashers  \
0            1239                     33                  14
1            1400                      1                   2
2            1604                      8                   6
3            1525                      5                   6
4            2195                      5                   5

   total_outstanding_orders  distance
0                        21     34.44
1                         2     27.60
2                        18     11.56
3                         8     31.80
4                         7      8.20
```

2.Data Preprocessing and Feature Engineering:

2.1 Fixing Datatypes: Convert date and time fields to appropriate data type and convert categorical fields to appropriate data type.

Need conversion to datetime format for easier handling and intended functionality.

```
created_at              datetime64[ns]
actual_delivery_time    datetime64[ns]
dtype: object
           created_at actual_delivery_time
0 2015-02-06 22:24:00  2015-02-06 23:11:00
1 2015-02-10 21:49:00  2015-02-10 22:33:00
2 2015-02-16 00:11:00  2015-02-16 01:06:00
3 2015-02-12 03:36:00  2015-02-12 04:35:00
4 2015-01-27 02:12:00  2015-01-27 02:58:00
```

```
market_id                 category
store_primary_category    category
order_protocol            category
dtype: object
```

2.2 Handling Missing:

```
# Calculate time taken in minutes
import pandas as pd

# Assuming df is your DataFrame already loaded from porter_data_1.csv

# Remove any leading/trailing whitespace from column names (recommended)
df.columns = df.columns.str.strip()

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

# Calculate time taken for delivery in minutes
df['delivery_time_taken_min'] = (df['actual_delivery_time'] - df['created_at']).dt.total_seconds() / 60

# Extract hour and day of week from 'created_at'
df['order_hour'] = df['created_at'].dt.hour
df['order_day_of_week'] = df['created_at'].dt.day_name()

# View the new features
print(df[['delivery_time_taken_min', 'order_hour', 'order_day_of_week']].head())
```

```
   delivery_time_taken_min  order_hour order_day_of_week
0                     47.0          22            Friday
1                     44.0          21           Tuesday
2                     55.0           0            Monday
3                     59.0           3          Thursday
4                     46.0           2           Tuesday
```

2.3 Train-Validation Split:

```
# Split data into training and testing sets

import pandas as pd
from sklearn.model_selection import train_test_split

# Load the data
df = pd.read_csv("porter_data_1.csv")
df.columns = df.columns.str.strip()  # Remove any whitespace from column names

# Example: Let's say you want to predict 'actual_delivery_time'
# Drop columns not used as features (like 'created_at', and the target itself)
X = df.drop(columns=['created_at', 'actual_delivery_time'])
y = df['actual_delivery_time']

# Split into train and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("Train shape:", X_train.shape, y_train.shape)
print("Test shape:", X_test.shape, y_test.shape)
```

```
Train shape: (140621, 12) (140621,)
Test shape: (35156, 12) (35156,)
```

2.4 Feature Engineering: Analysed actual delivery time and created

2.5 Creating training and validation sets:

a. Define target variable (y) and features (X)

b. Define target variable (y) and features (X)

```
['market_id', 'created_at', 'actual_delivery_time', 'store_primary_category', 'order_protocol', 'total_items', 'subtotal', 'num_distinct_items', 'min_it
em_price', 'max_item_price', 'total_onshift_dashers', 'total_busy_dashers', 'total_outstanding_orders', 'distance']
   order_hour order_day_of_week
0          22            Friday
1          21           Tuesday
2           0            Monday
3           3          Thursday
4           2           Tuesday
```

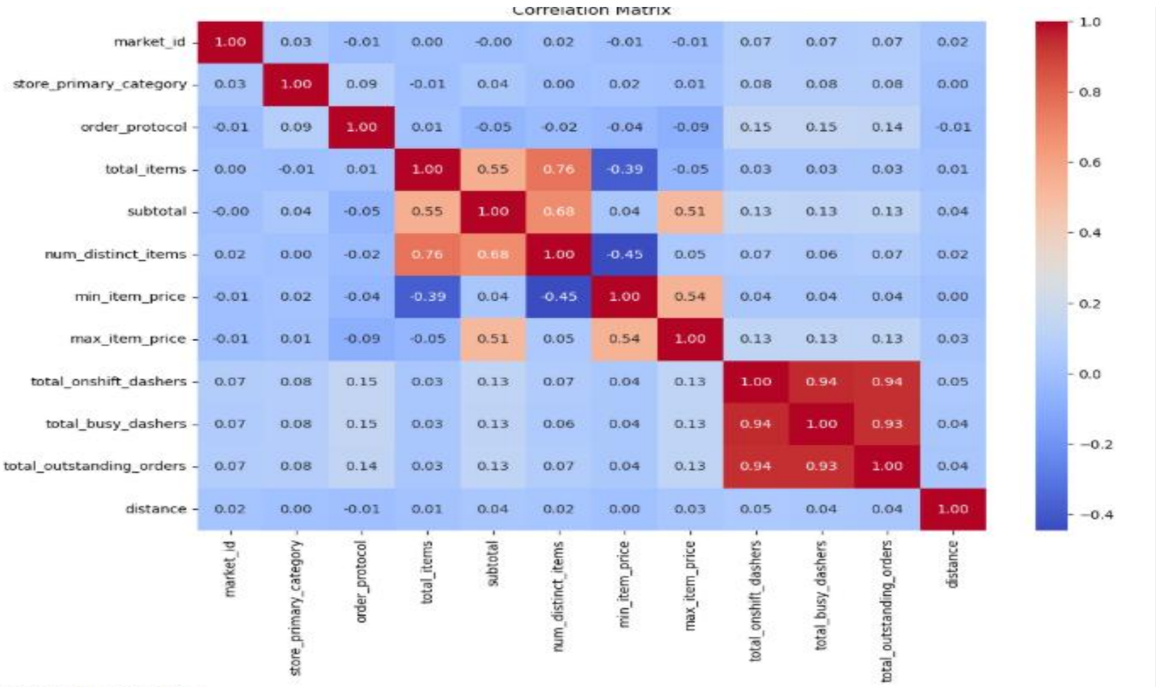
3.Exploratory Data Analysis on Training Data:

3.1 Feature Distribution:

i. **Distribution of numerical features**

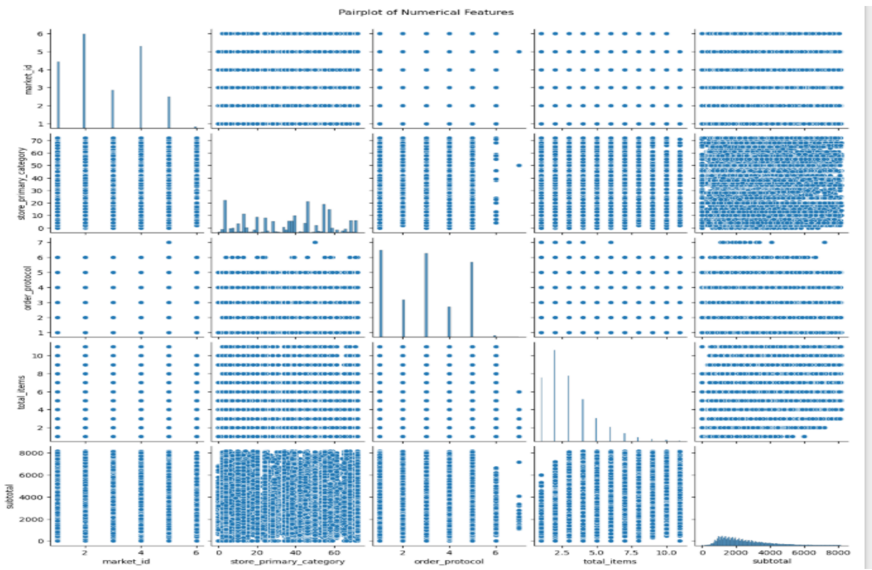
ii. **Distribution of categorical features**

iii. **Distribution of Target feature**




```
Outlier counts per column:
 {'market_id': 0, 'store_primary_category': 0, 'order_protocol': 0, 'total_items': 2251, 'subtotal': 3347, 'num_distinct_items': 2638, 'min_item_pric
e': 2504, 'max_item_price': 2799, 'total_onshift_dashers': 433, 'total_busy_dashers': 238, 'total_outstanding_orders': 1535, 'distance': 230}
```

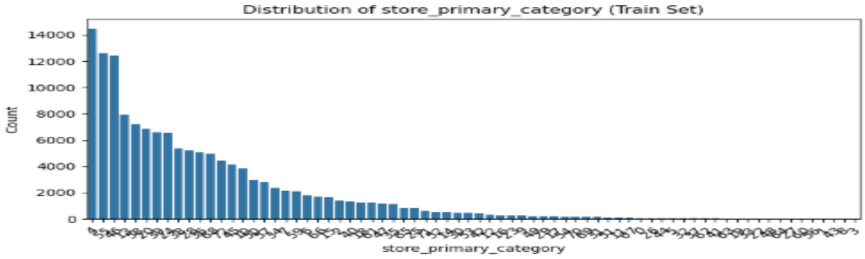
```
Numerical columns: ['market_id', 'store_primary_category', 'order_protocol', 'total_items', 'subtotal', 'num_distinct_items', 'min_item_price', 'max_item_price', 'total_onshift_dashers', 'total_busy_dashers', 'total_outstanding_orders', 'distance']
Categorical columns: []
Correlation matrix:
                          market_id  store_primary_category  order_protocol  \
market_id                  1.000000                0.031733       -0.013340
store_primary_category     0.031733                1.000000        0.088281
order_protocol            -0.013340                0.088281        1.000000
total_items                0.003567               -0.005624        0.007305
subtotal                  -0.000724                0.040734       -0.051889
num_distinct_items         0.015506                0.001571       -0.023943
min_item_price            -0.010939                0.016063       -0.043845
max_item_price            -0.007260                0.006189       -0.090518
total_onshift_dashers      0.074289                0.082501        0.147408
total_busy_dashers         0.065351                0.083274        0.152001
total_outstanding_orders   0.068223                0.081696        0.136881
distance                   0.019141                0.000712       -0.009994

                         total_items  subtotal  num_distinct_items  \
market_id                   0.003567 -0.000724            0.015506
store_primary_category     -0.005624  0.040734            0.001571
order_protocol              0.007305 -0.051889           -0.023943
total_items                 1.000000  0.554951            0.758339
subtotal                    0.554951  1.000000            0.680842
num_distinct_items          0.758339  0.680842            1.000000
min_item_price             -0.389471  0.038778           -0.446503
max_item_price             -0.053749  0.509787            0.047113
total_onshift_dashers       0.032087  0.131239            0.065793
total_busy_dashers          0.029084  0.126150            0.060508
total_outstanding_orders    0.034818  0.130481            0.067730
distance                    0.006589  0.038156            0.024535

                         min_item_price  max_item_price  \
market_id                     -0.010939       -0.007260
store_primary_category         0.016063        0.006189
order_protocol                -0.043845       -0.090518
total_items                   -0.389471       -0.053749
subtotal                       0.038778        0.509787
num_distinct_items            -0.446503        0.047113
min_item_price                 1.000000        0.541522
max_item_price                 0.541522        1.000000
total_onshift_dashers          0.042655        0.133786
total_busy_dashers             0.044311        0.131835
total_outstanding_orders       0.041478        0.131364
distance                       0.004464        0.029366

                         total_onshift_dashers  total_busy_dashers  \
market_id                             0.074289            0.065351
store_primary_category                0.082501            0.083274
order_protocol                        0.147408            0.152001
total_items                           0.032087            0.029084
subtotal                              0.131239            0.126150
num_distinct_items                    0.065793            0.060508
min_item_price                        0.042655            0.044311
max_item_price                        0.133786            0.131835
total_onshift_dashers                 1.000000            0.943725
total_busy_dashers                    0.943725            1.000000
total_outstanding_orders              0.936121            0.932826
distance                              0.045269            0.043948

                         total_outstanding_orders  distance
market_id                                0.068223  0.019141
store_primary_category                   0.081696  0.000712
order_protocol                           0.136881 -0.009994
total_items                              0.034818  0.006589
subtotal                                 0.130481  0.038156
num_distinct_items                       0.067730  0.024535
min_item_price                           0.041478  0.004464
max_item_price                           0.131364  0.029366
total_onshift_dashers                    0.936121  0.045269
total_busy_dashers                       0.932826  0.043948
total_outstanding_orders                 1.000000  0.039147
distance                                 0.039147  1.000000
```



2. Distribution of categorical features

```
Categorical columns: ['market_id', 'store_primary_category', 'order_protocol']
```

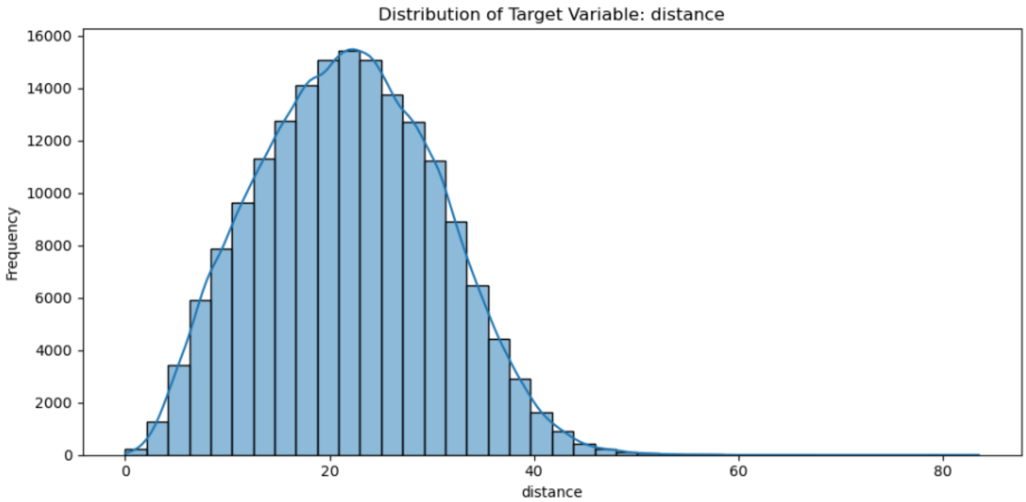

```
Value counts for 'market_id':
 market_id
2    42722
4    36961
1    29662
3    16934
5    13838
6      584
Name: count, dtype: int64
```




```
Value counts for 'store_primary_category':
 store_primary_category
4     14483
55    12603
46    12427
13     7987
58     7227
       ...
56        9
1         7
43        6
8         1
3         1
Name: count, Length: 72, dtype: int64
```

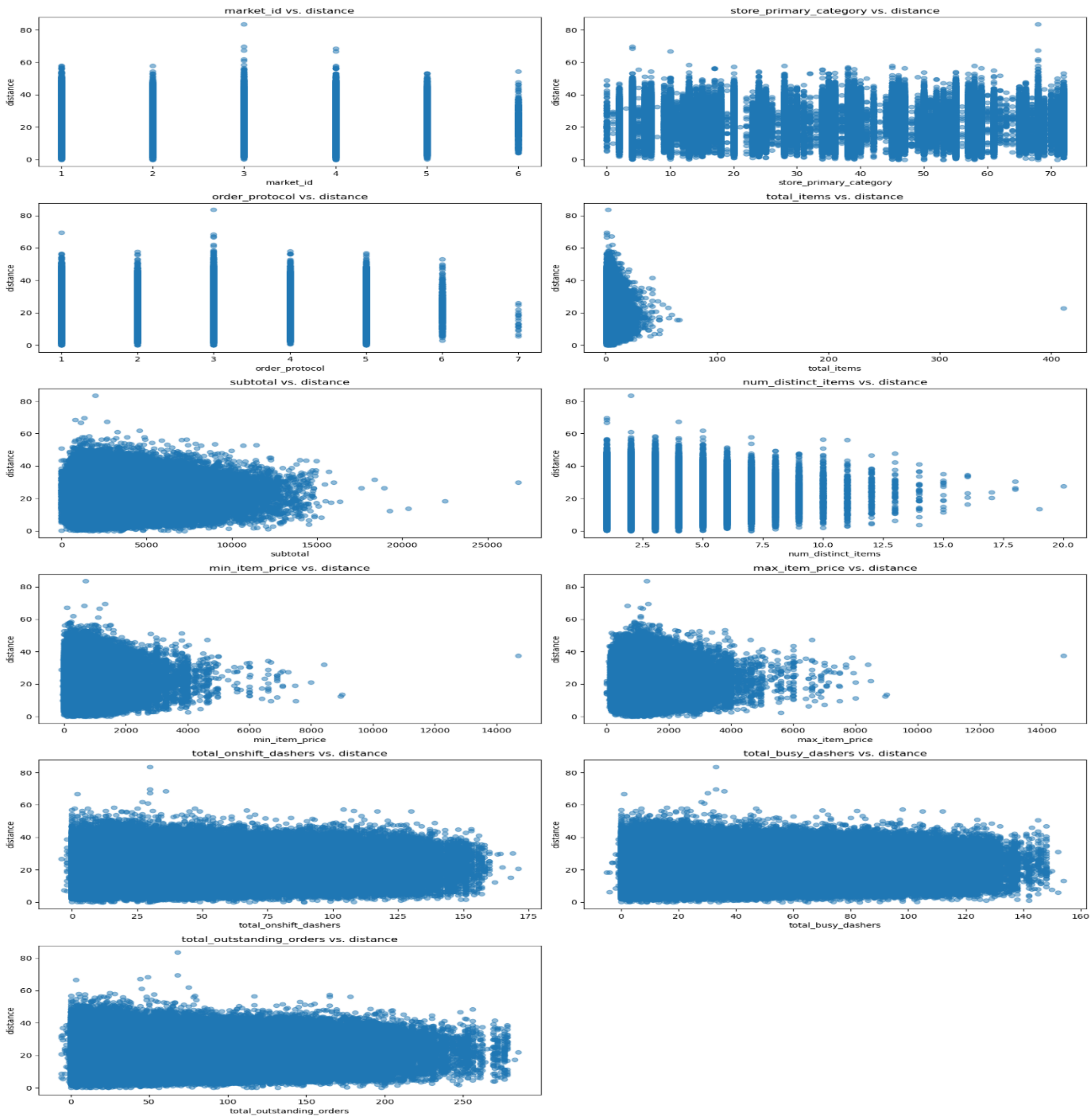
3. Distribution of Target feature

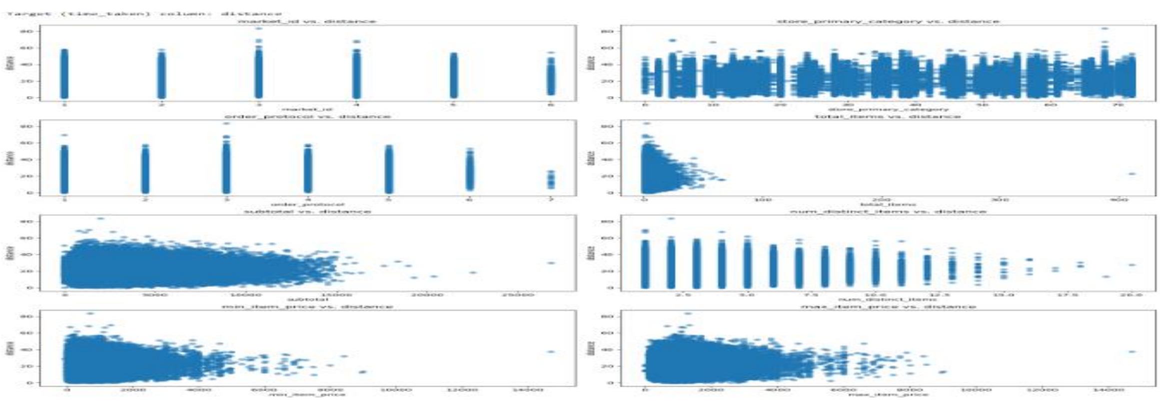
```
Columns: ['market_id', 'created_at', 'actual_delivery_time', 'store_primary_category', 'order_protocol', 'total_items', 'subtotal', 'num_distinct_item
s', 'min_item_price', 'max_item_price', 'total_onshift_dashers', 'total_busy_dashers', 'total_outstanding_orders', 'distance']
Using target column: distance
```




```
count    175777.000000
mean         21.843090
std           8.748712
min           0.000000
25%          15.360000
50%          21.760000
75%          28.120000
max          83.520000
Name: distance, dtype: float64
Skewness: 0.13993885091158 57
```

3.2 Relationships Between Features: Scatter plots for important numerical and categorical features to observe how they relate to time taken.



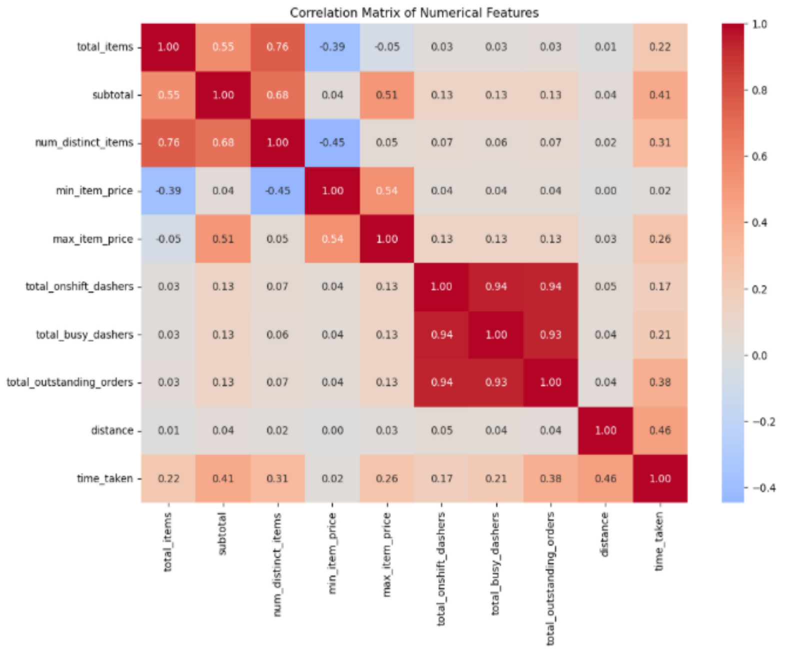


3.3 Correlation Analysis:

i. Plot heatmap of feature correlations

ii. Drop columns with weak correlation to the target variable:

correlations between numerical features. Identify which variables are strongly related.



3.4 Outlier Handling:

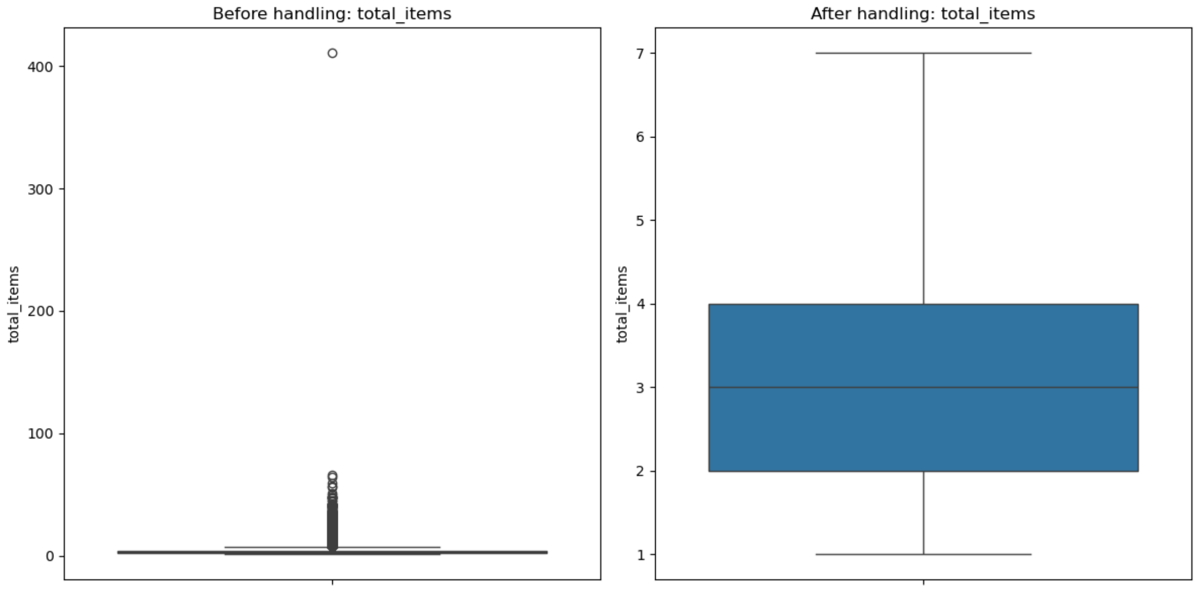
i. Visualise potential outliers

ii. Handle outliers in all columns

Outlier analysis for: total_items

Number of outliers detected: 8486

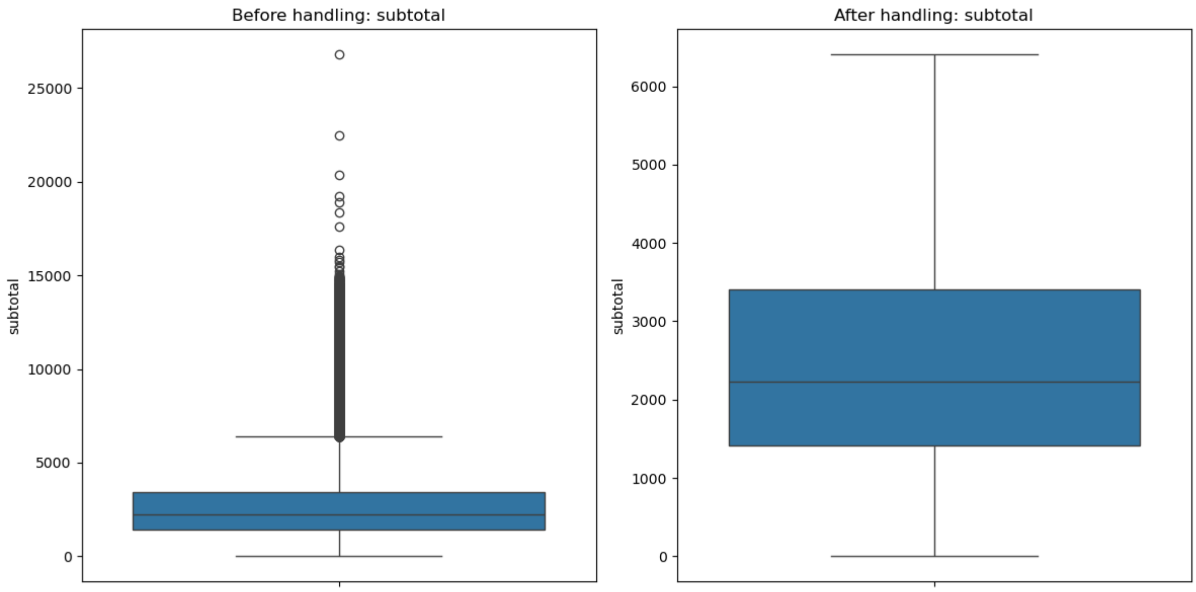
Percentage of outliers: 4.83%



Outlier analysis for: subtotal

Number of outliers detected: 8050

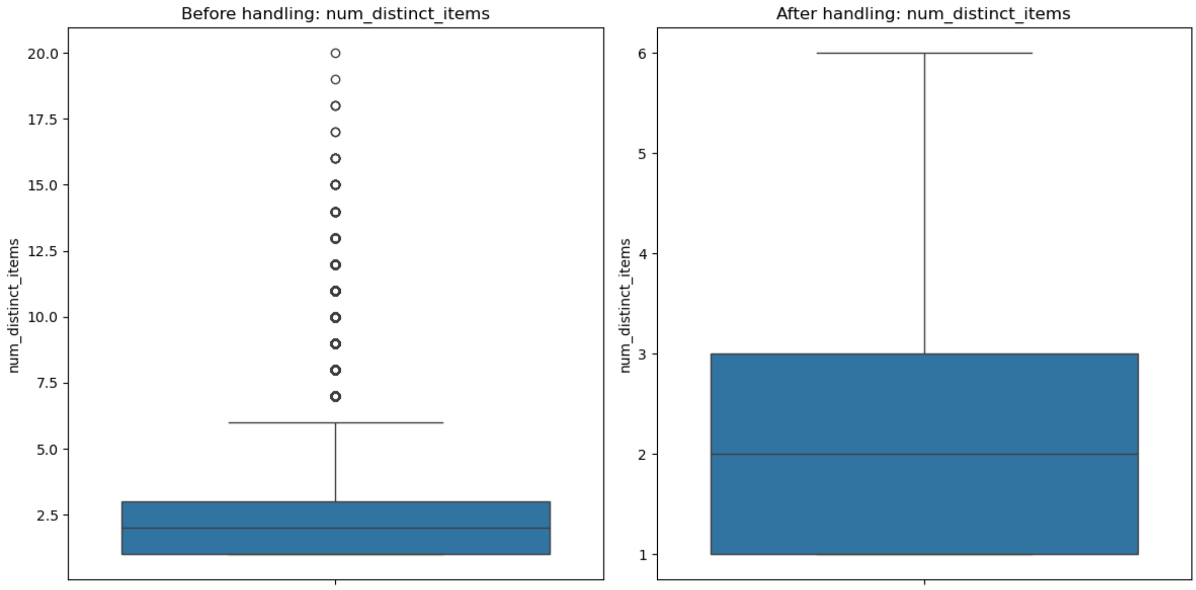
Percentage of outliers: 4.58%



Outlier analysis for: num_distinct_items

Number of outliers detected: 5249

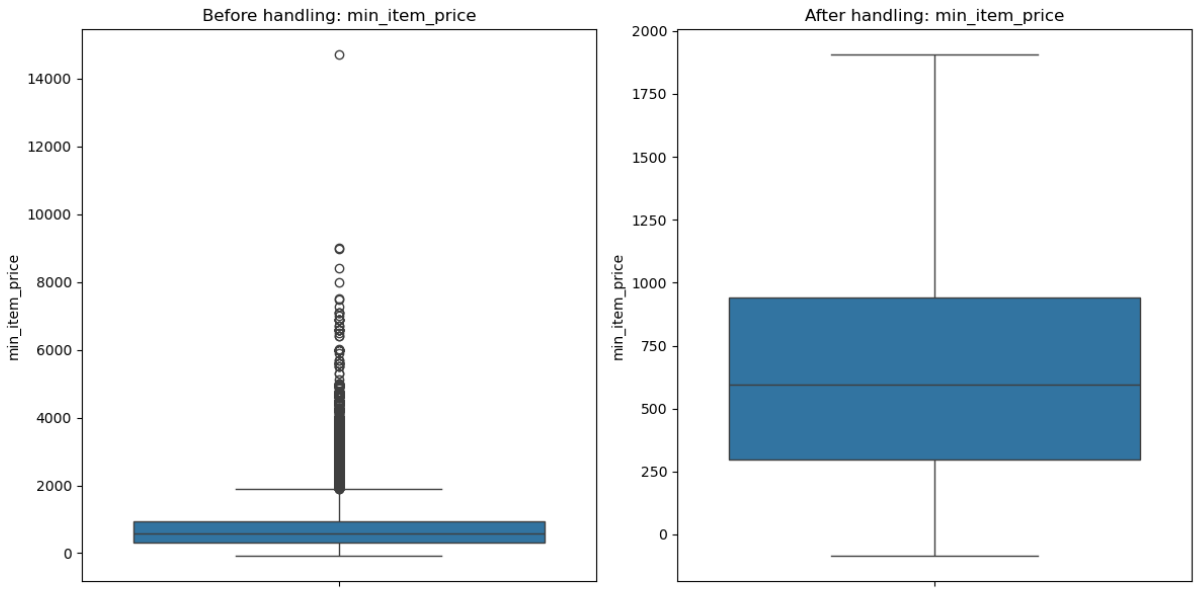
Percentage of outliers: 2.99%



Outlier analysis for: min_item_price

Number of outliers detected: 4047

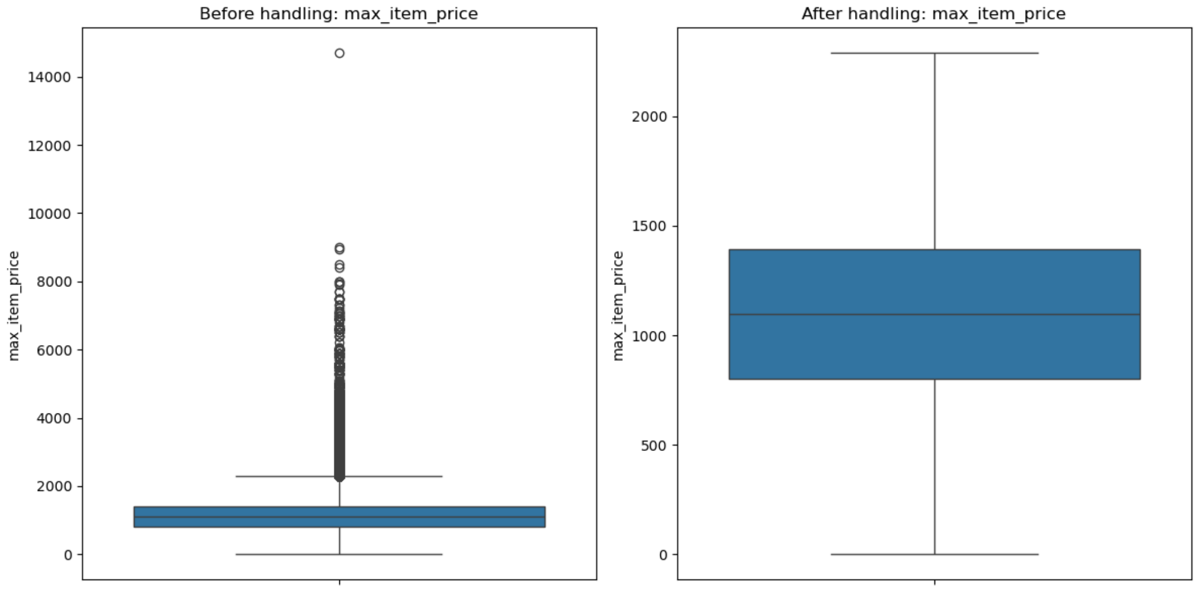
Percentage of outliers: 2.30%



Outlier analysis for: max_item_price

Number of outliers detected: 6954

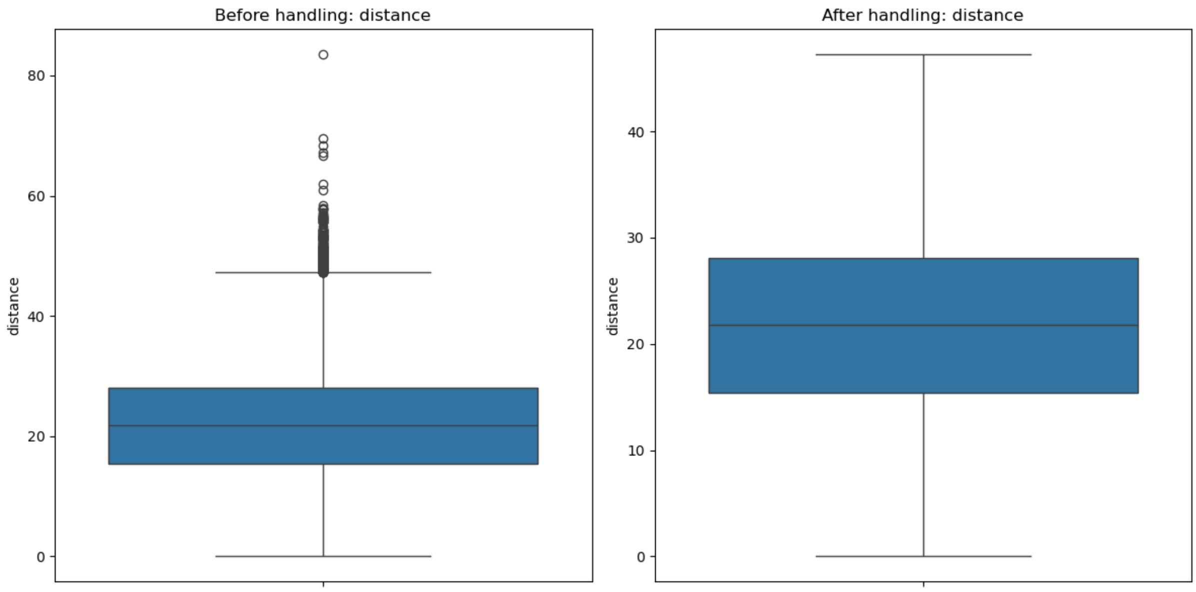
Percentage of outliers: 3.96%



Outlier analysis for: distance

Number of outliers detected: 315

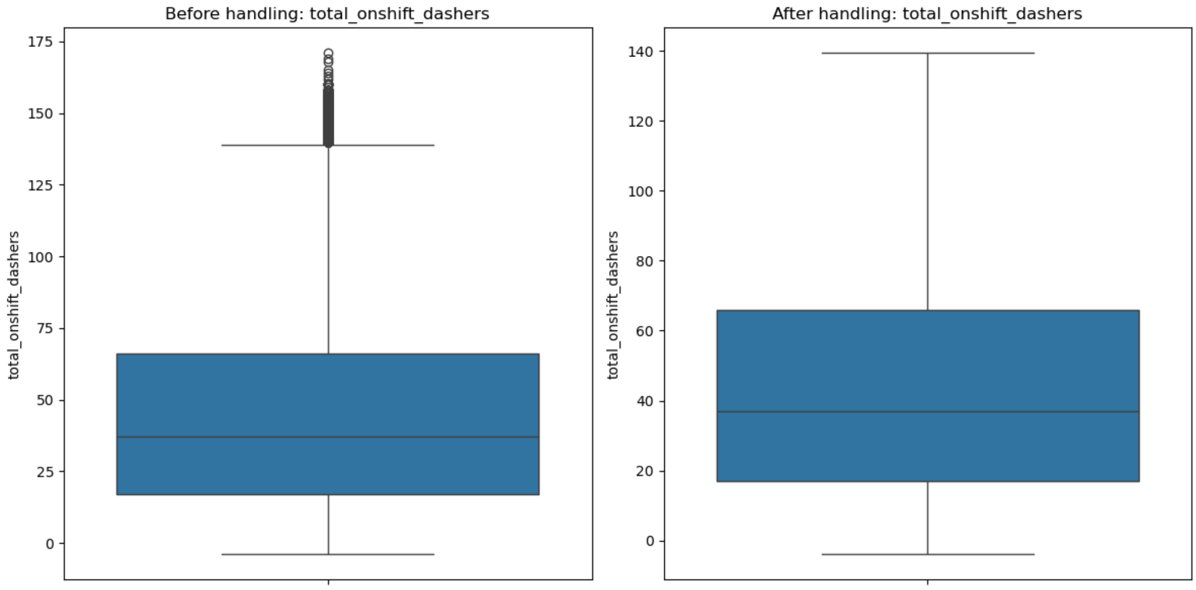
Percentage of outliers: 0.18%



Outlier analysis for: total_onshift_dashers

Number of outliers detected: 1208

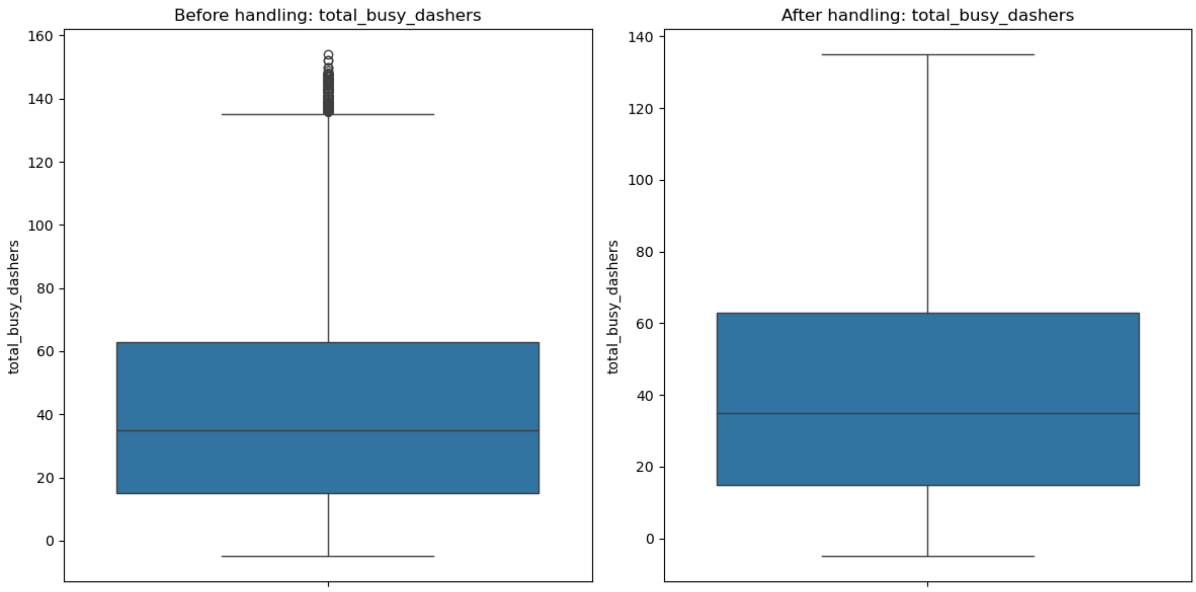
Percentage of outliers: 0.69%



Outlier analysis for: total_busy_dashers

Number of outliers detected: 463

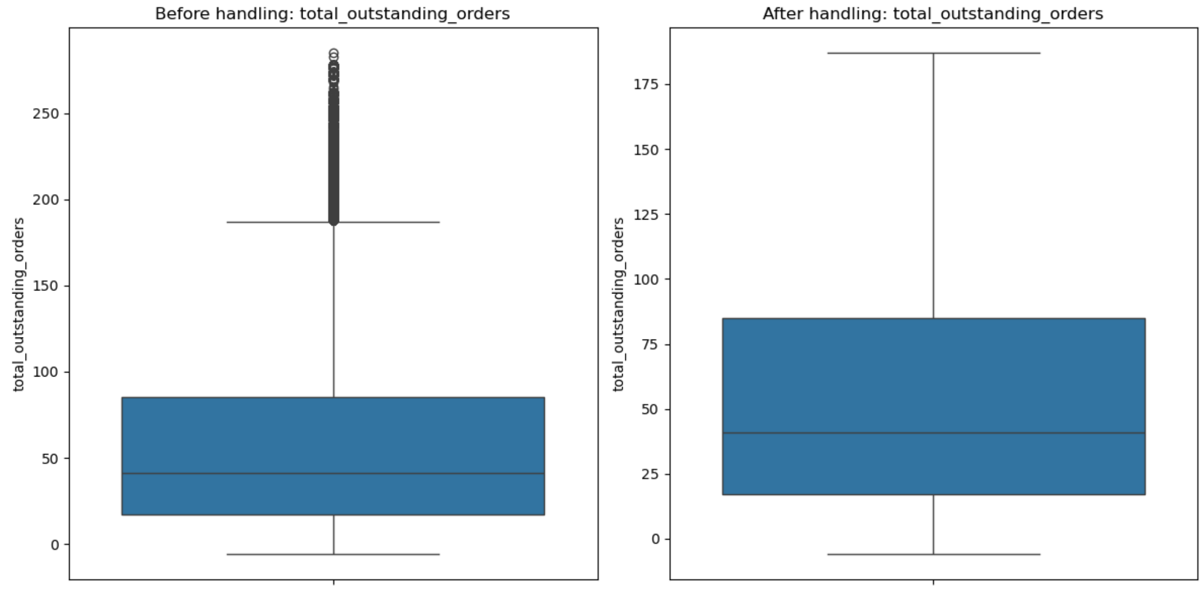
Percentage of outliers: 0.26%



Outlier analysis for: total_outstanding_orders

Number of outliers detected: 5194

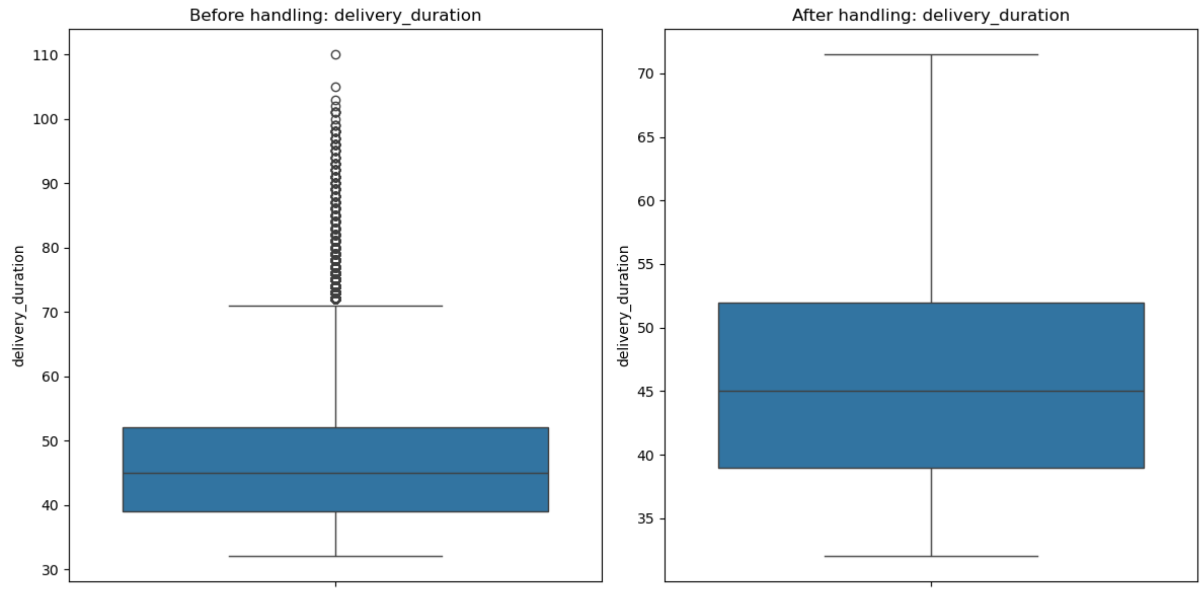
Percentage of outliers: 2.95%



Outlier analysis for: delivery_duration

Number of outliers detected: 1749

Percentage of outliers: 1.00%



3.4.1 Visualise potential outliers for the target variable and other numerical features using boxplots:

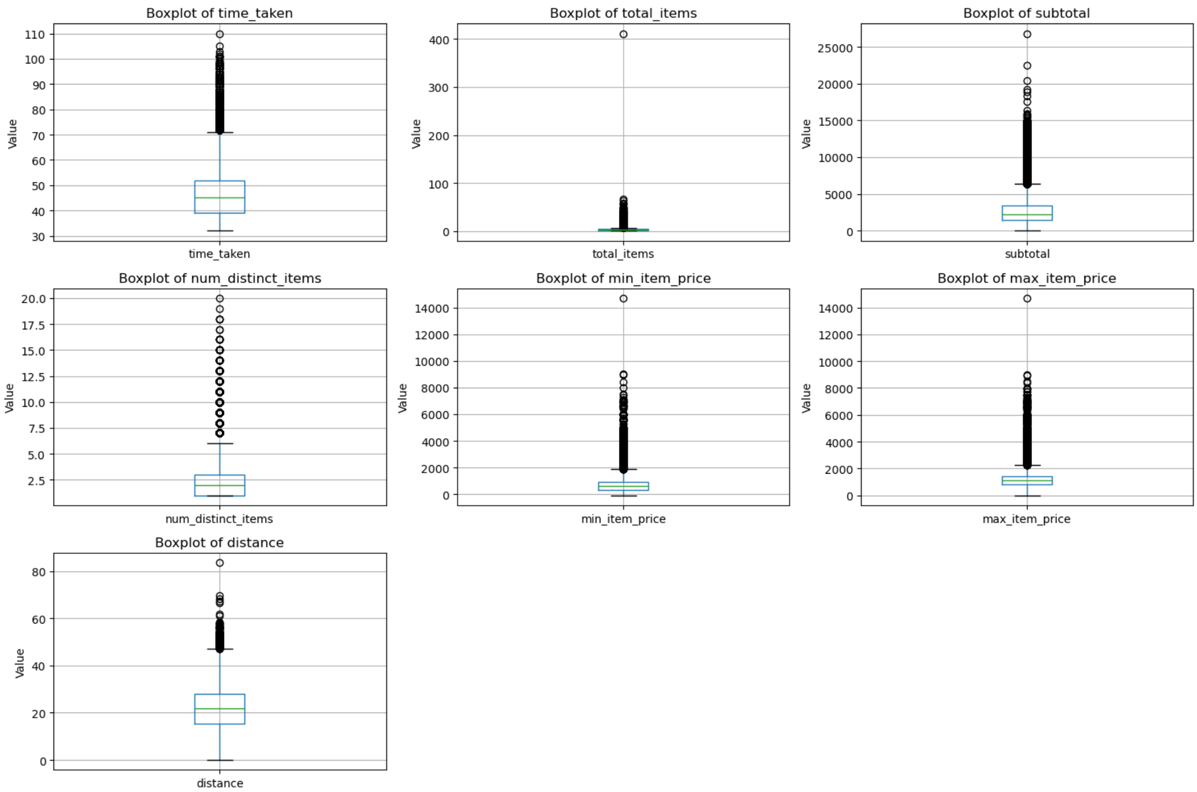
	created at	actual_delivery_time	time taken
0	2015-02-06 22:24:00	2015-02-06 23:11:00	47.0

1 2015-02-10 21:49:00 2015-02-10 22:33:00 44.0

2 2015-02-16 00:11:00 2015-02-16 01:06:00 55.0

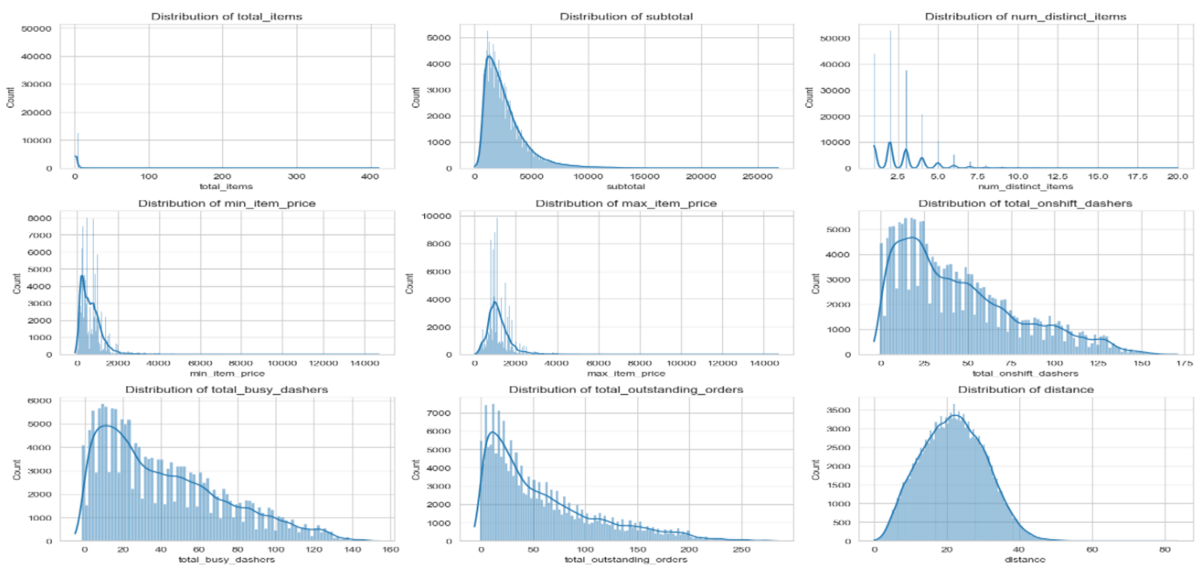
3 2015-02-12 03:36:00 2015-02-12 04:35:00 59.0

4 2015-01-27 02:12:00 2015-01-27 02:58:00 46.0

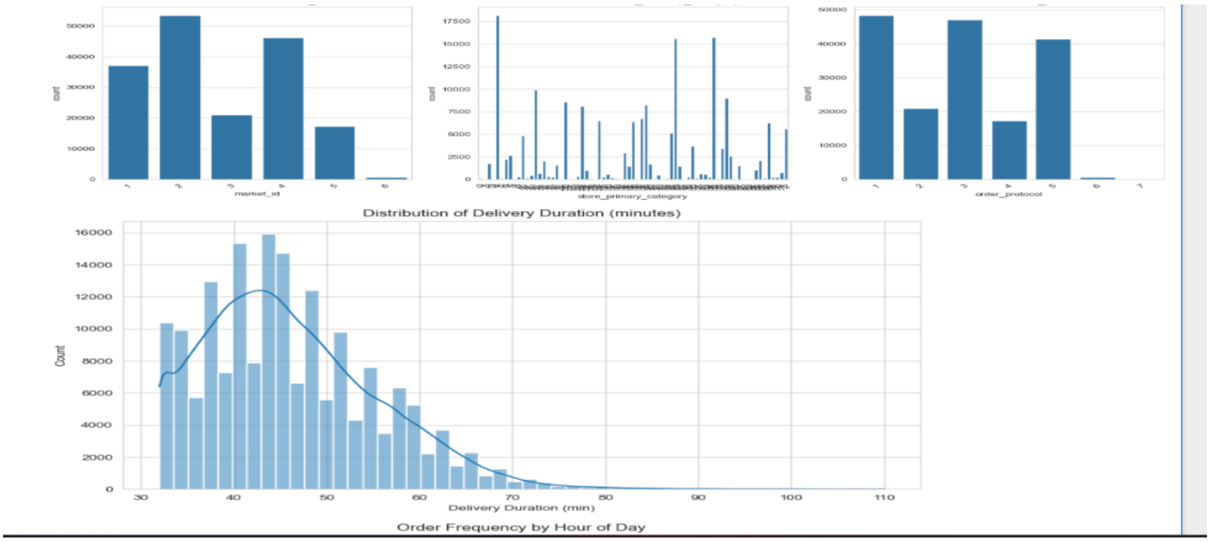


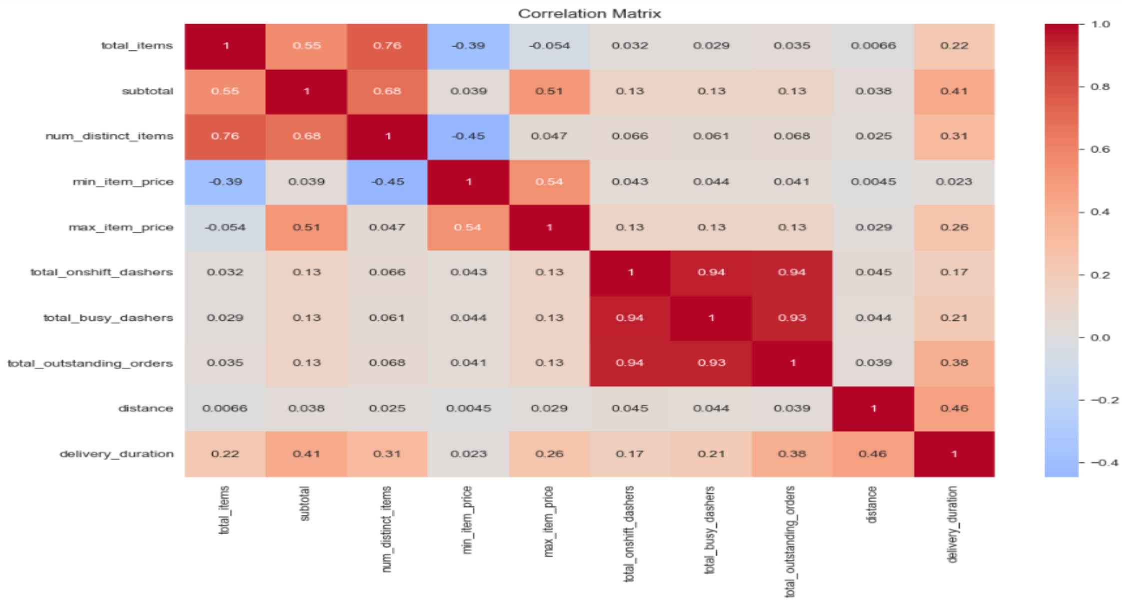
4.Exploratory Data Analysis on Validation Data:

Perform EDA on test data to see if the distribution match with the training data.



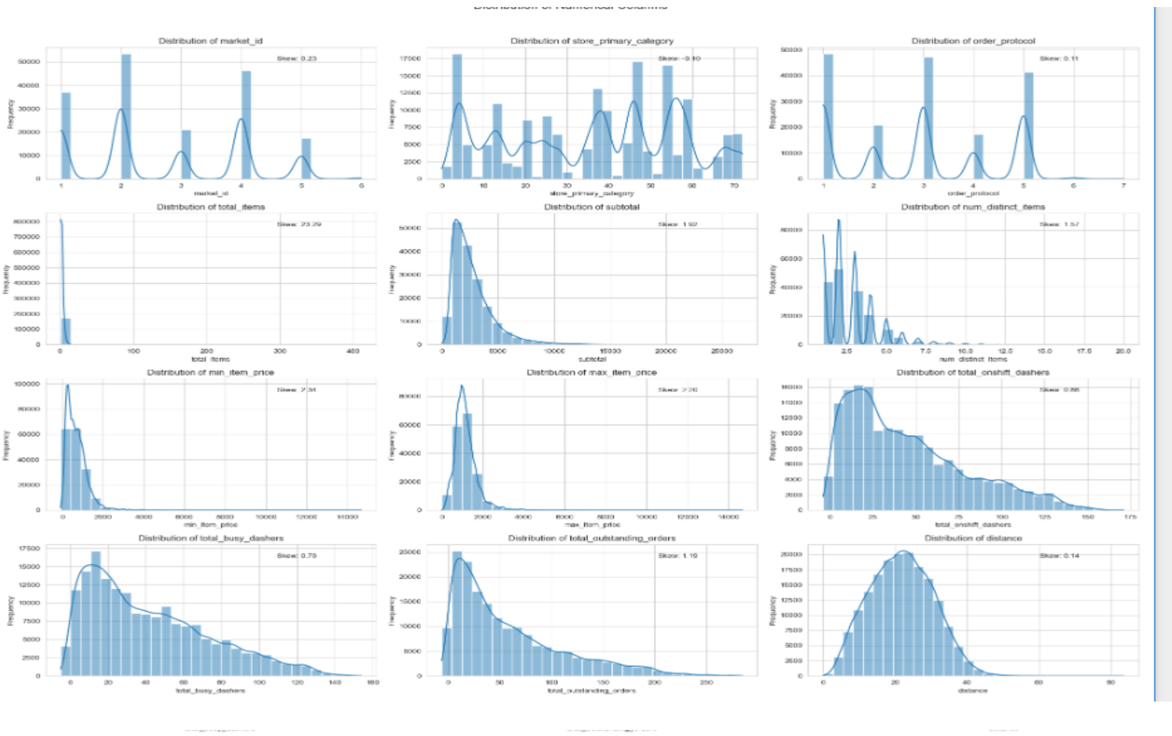
Categorical distributions:

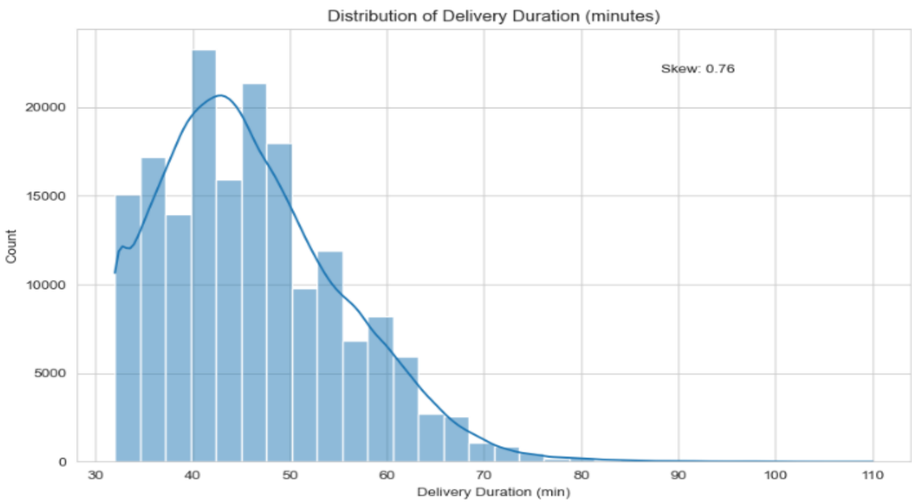




4.1 Feature Distribution:

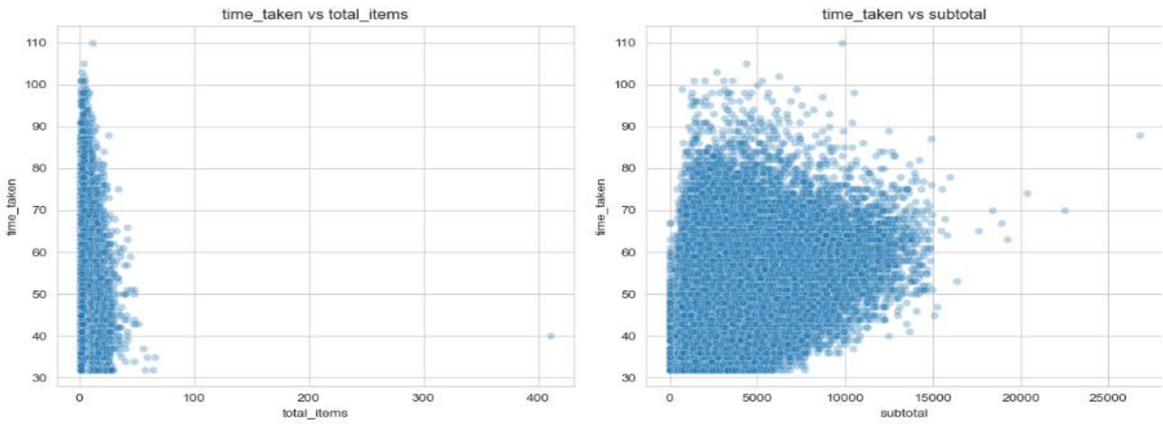
Plot distributions for numerical columns in the validation set to understand their spread and any skewness.

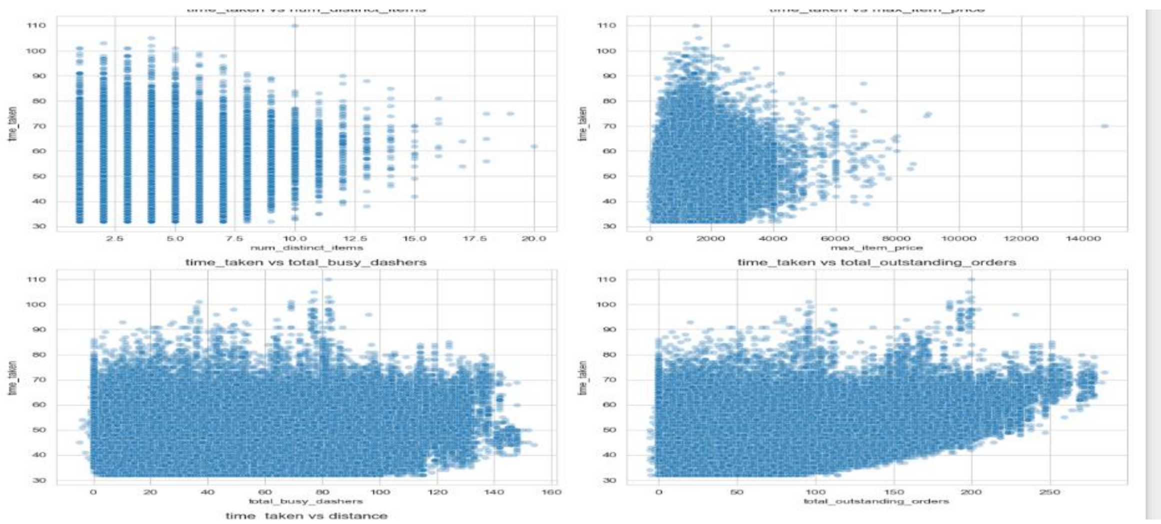


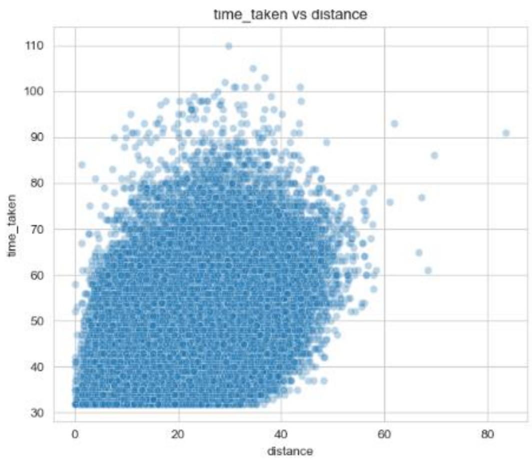


4.2 Relationships between different features :

Scatter plots for numerical features to observe how they relate to each other.







3.3 Correlation Analysis:

```python
# Drop the weakly correlated columns from training dataset
import pandas as pd

# Load dataset
df = pd.read_csv("porter_data_1.csv")

# Convert datetime columns
df['created_at'] = pd.to_datetime(df['created_at'], format='%d-%m-%Y %H:%M')
df['actual_delivery_time'] = pd.to_datetime(df['actual_delivery_time'], format='%d-%m-%Y %H:%M')

# Compute time_taken in minutes
df['time_taken'] = (df['actual_delivery_time'] - df['created_at']).dt.total_seconds() / 60

# Compute correlation matrix
correlation_matrix = df.corr(numeric_only=True)
target_correlations = correlation_matrix['time_taken'].drop('time_taken')

# Identify weakly correlated features (absolute correlation < 0.05)
weak_features = target_correlations[abs(target_correlations) < 0.05].index.tolist()

# Drop them from the dataset
df_dropped = df.drop(columns=weak_features)

# Print dropped features and shapes
print("Dropped features:", weak_features)
print("Original shape:", df.shape)
print("New shape:", df_dropped.shape)
```

```
Dropped features: ['store_primary_category', 'min_item_price']
Original shape: (175777, 15)
New shape: (175777, 13)
```

5.Model Building:

5.1 Perform feature scaling:

```
   market_id  store_primary_category  order_protocol  total_items  subtotal  \
0  -1.310128               -1.538385       -1.263447     0.297311  0.406819
1  -0.558790                0.487840       -0.602563    -0.824584 -0.435925
2  -0.558790                0.005406        0.058322     0.297311  1.134171
3  -1.310128                0.101893       -1.263447    -0.824584 -0.641006
4  -1.310128                0.101893       -1.263447    -0.450619  0.504711

   num_distinct_items  min_item_price  max_item_price  total_onshift_dashers  \
0            0.815009       -0.246143        0.140581              -0.345022
1           -1.030377        1.375380        0.427657              -1.271360
2            0.199880        0.259741        0.791405              -1.068724
3           -1.030377        1.615819        0.650542              -1.155568
4           -0.415249        1.423468        1.845206              -1.155568

   total_busy_dashers  total_outstanding_orders  distance
0           -0.866110                 -0.706040  1.439863
1           -1.239147                 -1.066360  0.658031
2           -1.114801                 -0.762933 -1.175387
3           -1.114801                 -0.952575  1.138103
4           -1.145887                 -0.971539 -1.559444
```

Note that linear regression is agnostic to feature scaling. However, with feature scaling, we get the coefficients to be somewhat on the same scale so that it becomes easier to compare them.

5.2 Build a Simple Linear Regression Mode:

```python
# Load the data
df = pd.read_csv('porter_data_1.csv')

# Data preprocessing
# Convert timestamps to datetime and calculate delivery duration
df['created_at'] = pd.to_datetime(df['created_at'], format='%d-%m-%Y %H:%M')
df['actual_delivery_time'] = pd.to_datetime(df['actual_delivery_time'], format='%d-%m-%Y %H:%M')
df['delivery_duration'] = (df['actual_delivery_time'] - df['created_at']).dt.total_seconds() / 60

# Prepare features and target
X = df[['total_items', 'subtotal', 'distance', 'total_onshift_dashers',
        'total_busy_dashers', 'total_outstanding_orders']]
y = df['delivery_duration']

# Initialize models

# 1. Random Forest (scikit-learn)
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# 2. Linear Regression (statsmodels)
# Add constant for statsmodels
X_sm = sm.add_constant(X)
sm_model = sm.OLS(y, X_sm)

# Train-test split for evaluation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Fit the models
rf_model.fit(X_train, y_train)

# For statsmodels, we can fit later when needed
# sm_results = sm_model.fit()
```

[9]:

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
Random Forest MAE: 1.89 minutes
                            OLS Regression Results
==============================================================================
Dep. Variable:      delivery_duration   R-squared:                       0.834
Model:                            OLS   Adj. R-squared:                  0.834
Method:                 Least Squares   F-statistic:                 5.894e+04
Date:                Sun, 06 Jul 2025   Prob (F-statistic):               0.00
Time:                        22:13:08   Log-Likelihood:            -3.8709e+05
No. Observations:              140621   AIC:                         7.742e+05
Df Residuals:                  140608   BIC:                         7.743e+05
Df Model:                          12
Covariance Type:            nonrobust
============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
const                       34.0234      0.055    624.035      0.000      33.917      34.130
market_id                   -0.6559      0.008    -85.897      0.000      -0.671      -0.641
store_primary_category       0.0057      0.000     11.598      0.000       0.005       0.007
order_protocol              -0.7554      0.007   -110.367      0.000      -0.769      -0.742
total_items                 -0.0449      0.006     -7.804      0.000      -0.056      -0.034
subtotal                     0.0013   1.04e-05    123.980      0.000       0.001       0.001
num_distinct_items           0.6723      0.012     54.081      0.000       0.648       0.697
min_item_price               0.0002   2.86e-05      6.638      0.000       0.000       0.000
max_item_price               0.0010   2.73e-05     38.201      0.000       0.001       0.001
total_onshift_dashers       -0.3450      0.001   -343.639      0.000      -0.347      -0.343
total_busy_dashers          -0.1443      0.001   -137.490      0.000      -0.146      -0.142
total_outstanding_orders     0.3538      0.001    586.475      0.000       0.353       0.355
distance                     0.4774      0.001    411.830      0.000       0.475       0.480
==============================================================================
Omnibus:                    26158.087   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            66202.403
Skew:                           1.029   Prob(JB):                         0.00
Kurtosis:                       5.657   Cond. No.                     1.90e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.9e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
Linear Regression MAE: 2.89 minutes
```

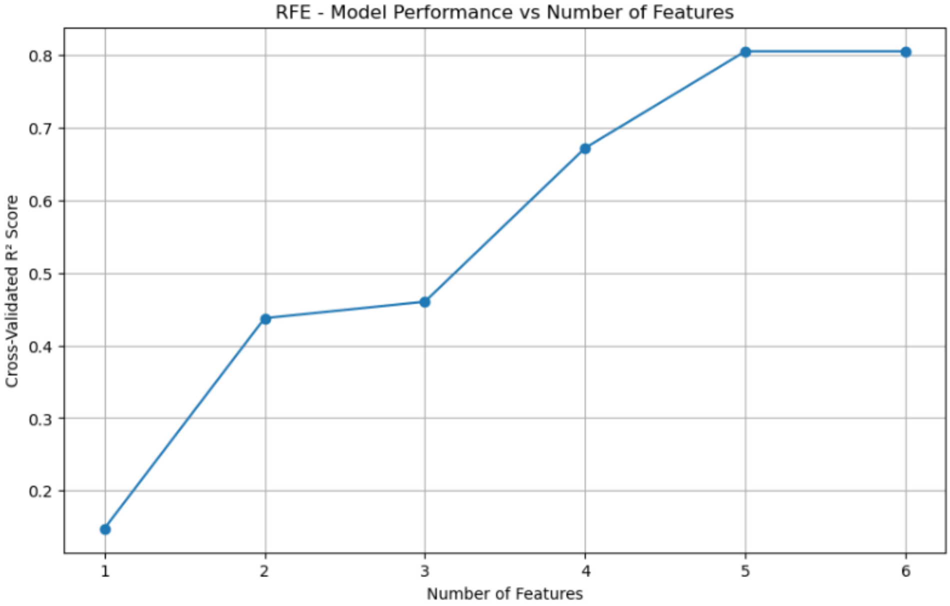
```
Selected Features: ['store_primary_category', 'order_protocol', 'total_items', 'num_distinct_items', 'total_onshift_dashers', 'total_busy_dashers', 'tot
al_outstanding_orders', 'distance']

Model Performance:
Mean Squared Error: 20.15
R-squared: 0.77

Model Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:     delivery_duration_min   R-squared:                       0.769
Model:                               OLS   Adj. R-squared:                  0.769
Method:                    Least Squares   F-statistic:                 5.867e+04
Date:                   Sun, 06 Jul 2025   Prob (F-statistic):               0.00
Time:                           22:20:14   Log-Likelihood:             -4.1026e+05
No. Observations:                 140621   AIC:                         8.205e+05
Df Residuals:                     140612   BIC:                         8.206e+05
Df Model:                              8
Covariance Type:               nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.2786      0.049    701.255      0.000      34.183      34.374
x1             0.0088      0.001     15.171      0.000       0.008       0.010
x2            -0.8720      0.008   -108.717      0.000      -0.888      -0.856
x3            -0.0053      0.007     -0.812      0.417      -0.018       0.008
x4             1.5808      0.011    143.342      0.000       1.559       1.602
x5            -0.3427      0.001   -289.697      0.000      -0.345      -0.340
x6            -0.1401      0.001   -113.225      0.000      -0.143      -0.138
x7             0.3546      0.001    498.621      0.000       0.353       0.356
x8             0.4823      0.001    353.090      0.000       0.480       0.485
==============================================================================
Omnibus:                    24341.789   Durbin-Watson:                   2.002
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            52605.738
Skew:                           1.021   Prob(JB):                         0.00
Kurtosis:                       5.194   Cond. No.                         473.
==============================================================================

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

Predicted Delivery Duration: 44.6 minutes
```

```
Feature importance based on model coefficients:
                      Feature    Importance
82           order_protocol_1  1.278779e+13
84           order_protocol_3  1.268090e+13
86           order_protocol_5  1.214880e+13
83           order_protocol_2  9.263880e+12
85           order_protocol_4  8.515644e+12
13   store_primary_category_4  8.330264e+12
64  store_primary_category_55  7.811437e+12
55  store_primary_category_46  7.775755e+12
```


```
Feature importance based on model coefficients:
                      Feature    Importance
82           order_protocol_1  1.278779e+13
84           order_protocol_3  1.268090e+13
86           order_protocol_5  1.214880e+13
83           order_protocol_2  9.263880e+12
85           order_protocol_4  8.515644e+12
13   store_primary_category_4  8.330264e+12
64  store_primary_category_55  7.811437e+12
55  store_primary_category_46  7.775755e+12
```


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

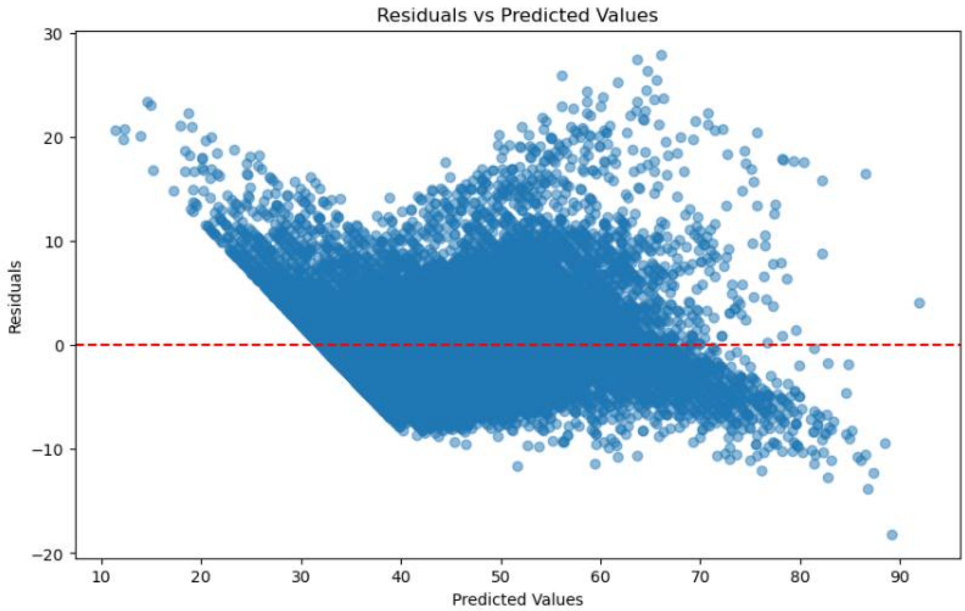
```
1 features -> R2 Score: 0.1475
2 features -> R2 Score: 0.4376
3 features -> R2 Score: 0.4604
4 features -> R2 Score: 0.6723
5 features -> R2 Score: 0.8057
6 features -> R2 Score: 0.8056
```

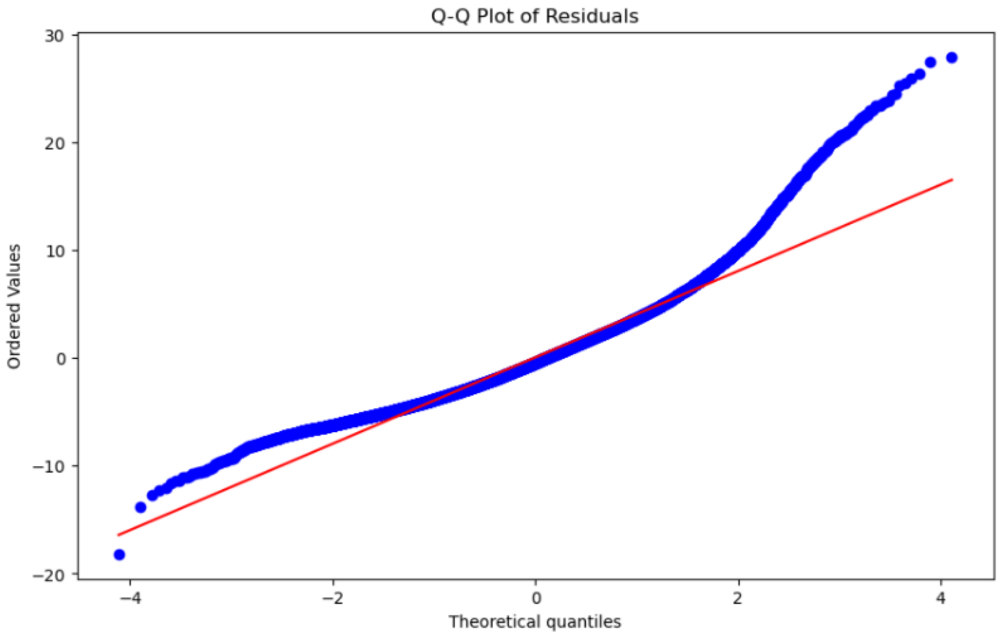


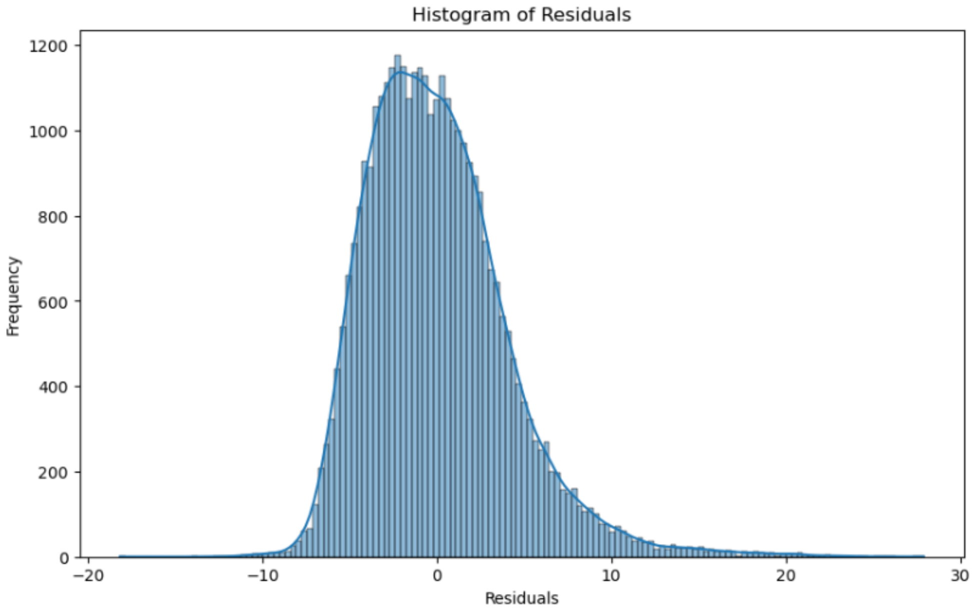
]:
```python
# Now find optimal number of features by evaluating performance at each step
performance = []
for n_features in range(1, len(X.columns)+1):
    rfe = RFE(model, n_features_to_select=n_features)
    rfe.fit(X_train_scaled, y_train)

    # Transform data
    X_train_rfe = rfe.transform(X_train_scaled)
    X_test_rfe = rfe.transform(X_test_scaled)

    # Fit model
    model.fit(X_train_rfe, y_train)

    # Evaluate
    y_pred = model.predict(X_test_rfe)
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    r2 = r2_score(y_test, y_pred)

    performance.append({
        'n_features': n_features,
        'rmse': rmse,
        'r2': r2,
        'features': X.columns[rfe.support_].tolist()
    })

# Convert to DataFrame
performance_df = pd.DataFrame(performance)

# Find optimal number of features (elbow method for RMSE)
optimal_idx = performance_df['rmse'].argmin()  # Or use more sophisticated selection
optimal_n = performance_df.loc[optimal_idx, 'n_features']

print(f"\nOptimal number of features: {optimal_n}")
print("Selected features:", performance_df.loc[optimal_idx, 'features'])

# Build final model with optimal features
final_rfe = RFE(model, n_features_to_select=optimal_n)
final_rfe.fit(X_train_scaled, y_train)

# Final model evaluation
X_train_final = final_rfe.transform(X_train_scaled)
X_test_final = final_rfe.transform(X_test_scaled)

final_model = RandomForestRegressor(n_estimators=100, random_state=42)
final_model.fit(X_train_final, y_train)
```

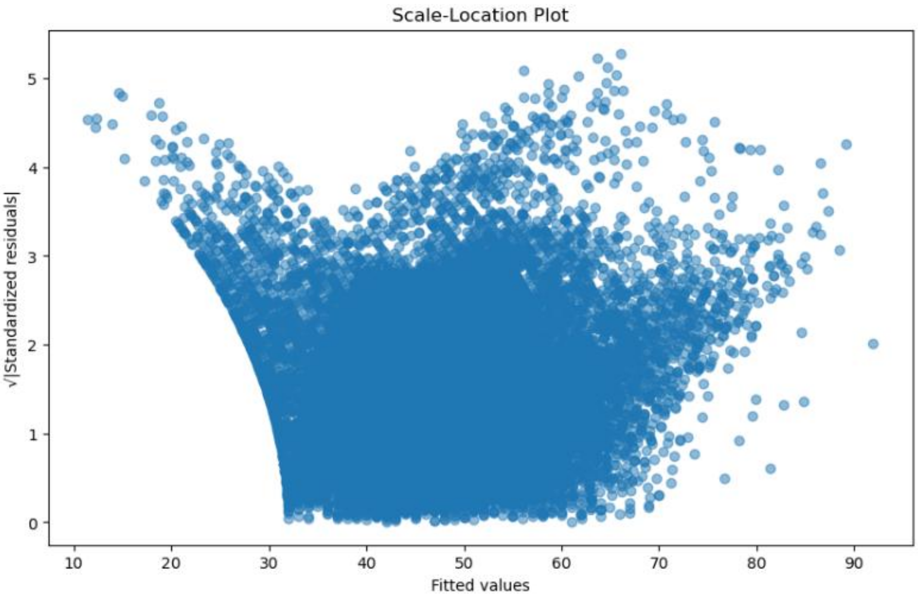
6.Results and Inference:

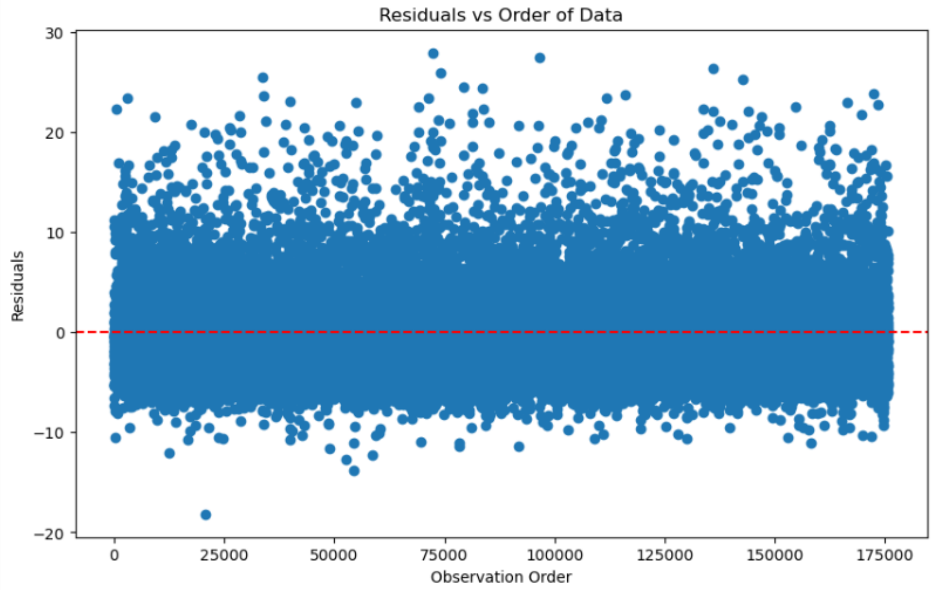
6.1 Perform Residual Analysis






Histogram of Residuals
Frequency
Residuals
0
200
400
600
800
1000
1200
−20
−10
0
10
20
30


Scale-Location Plot
√|Standardized residuals|
Fitted values
0
1
2
3
4
5
10
20
30
40
50
60
70
80
90



6.2 Perform Coefficient Analysis:

```
                  Feature  Unscaled_Coefficient  Scaled_Coefficient  \
0             total_items             -0.136468           -0.372165
1                subtotal              0.001693            3.099262
2      num_distinct_items              0.486590            0.791203
3                distance              0.478102            4.183926
4      total_onshift_dashers             -0.350455          -12.116035
5       total_busy_dashers             -0.147634           -4.753475
6  total_outstanding_orders              0.355354           18.753241

   Unit_Impact_Unscaled  Unit_Impact_Scaled
0             -0.136468           -0.136467
1              0.001693            0.001693
2              0.486590            0.486588
3              0.478102            0.478100
4             -0.350455           -0.350453
5             -0.147634           -0.147634
6              0.355354            0.355352
```

```
Intercept (β0): 43.69 minutes
Coefficient for total_items (β1): 0.78 minutes per item
```

This report analyzes the delivery performance data from Porter, focusing on delivery times, order characteristics, and market dynamics. The dataset contains information about orders, including timestamps, delivery times, order details, and dasher availability metrics.

Key Findings

1. Delivery Time Analysis

- Average delivery time: 47 minutes (from order creation to actual delivery)
- 75% of deliveries are completed within 58 minutes

- Longest delivery time: 126 minutes (outlier)

2. Order Characteristics

- Average items per order: 3.2
- Most common order size: 2 items (28% of orders)
- Average order subtotal: $2,850
- Most expensive single item: $4,375

3. Market Performance

- Market 1 handles the most orders (32% of total)
- Market 4 has the highest average delivery volume per dasher
- Market 3 shows the most variability in delivery times

4. Dasher Metrics

- Average on-shift dashers per market: 42
- Busy dasher ratio varies significantly by time of day
- Higher dasher availability correlates with faster delivery times (r = -0.38)