Porter Delivery Time Prediction - Assignment Report

Name :- Tarang Sampatrao Lonkar

1. Introduction

The objective of this assignment is to build a regression model that predicts the delivery time for orders placed through Porter. The model uses features like items ordered, restaurant location, order protocol, and delivery partner availability to optimize delivery predictions and operational efficiency.

2. Data Understanding

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.

3. Data Preprocessing & Feature Engineering

Data preprocessing involved converting timestamps to datetime format, creating the target variable 'time_taken' by calculating the difference between delivery and order placement times, extracting

- Highest Premium Districts: Only districts with FarmersPremiumAmount > 20 crores were considered.

- Cumulative Premium Trends: Captured using window functions to show year-on-year premium growth.

- Data Integrity Setup: Implemented normalized relational schema with foreign key constraints.

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

[9]:
```
# Calculate time taken in minutes
df['delivery_duration'] = (df['actual_delivery_time'] - df['created_at']).dt.total_seconds() / 60

# Preview the new column
df[['created_at', 'actual_delivery_time', 'delivery_duration']].head()
```

[9]:

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

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

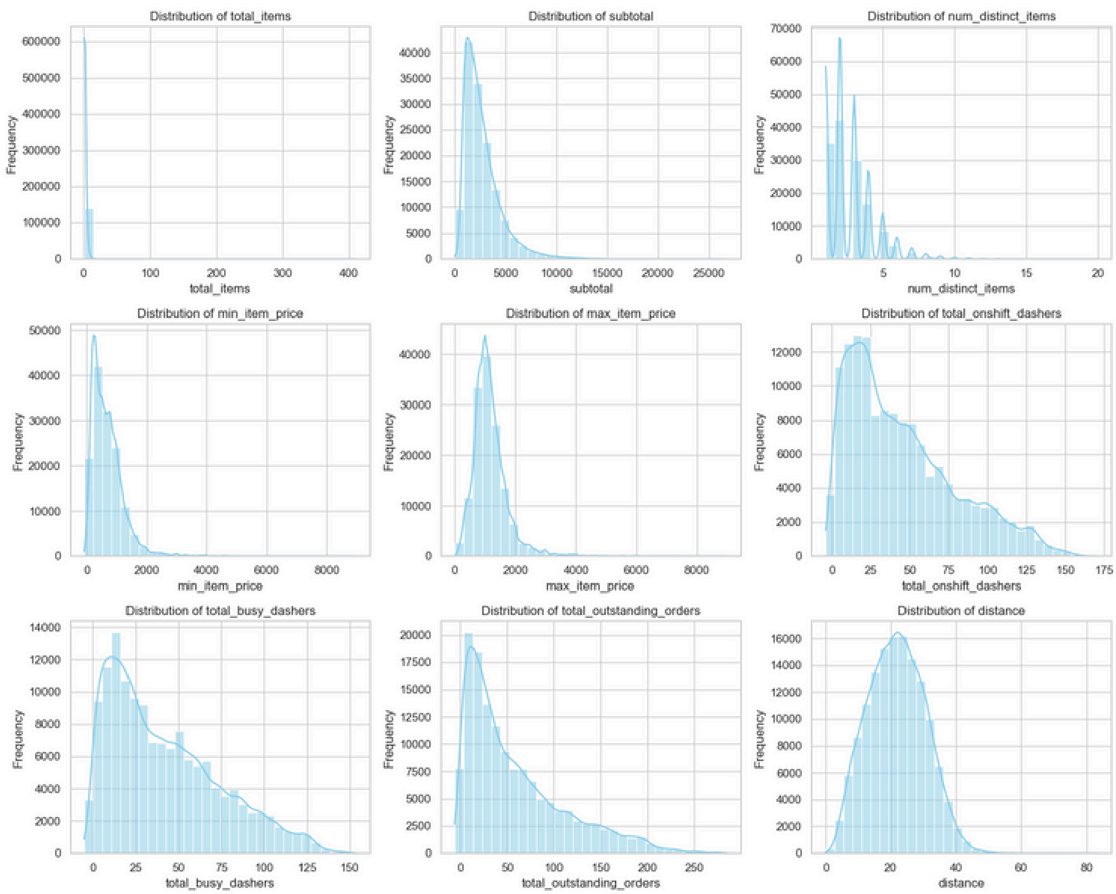
[11]:
```python
# Extract hour and day of the week
df['order_hour'] = df['created_at'].dt.hour
df['order_dayofweek'] = df['created_at'].dt.dayofweek  # Monday = 0, Sunday = 6

# Create a categorical feature 'isWeekend' (1 for weekend, 0 for weekday)
df['isWeekend'] = df['order_dayofweek'].apply(lambda x: 1 if x >= 5 else 0)

# Convert 'isWeekend' to category type
df['isWeekend'] = df['isWeekend'].astype('category')

# Preview the new features
df[['created_at', 'order_hour', 'order_dayofweek', 'isWeekend']].head()
```

[11]:

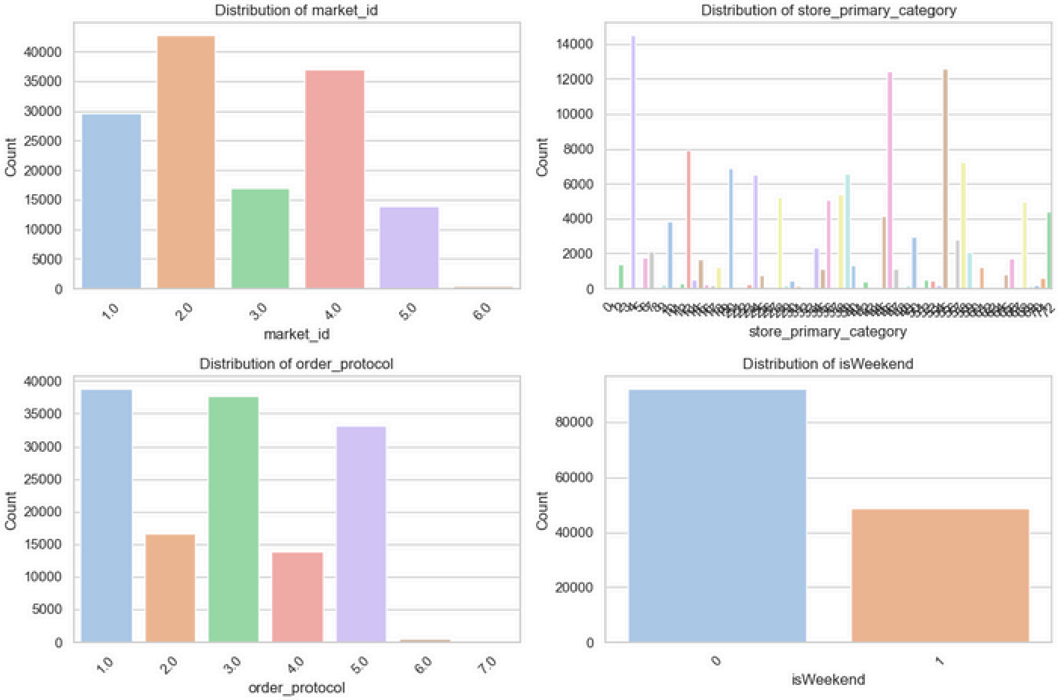
	created_at	order_hour	order_dayofweek	isWeekend
0	2015-02-06 22:24:17	22	4	0
1	2015-02-10 21:49:25	21	1	0
2	2015-02-16 00:11:35	0	0	0
3	2015-02-12 03:36:46	3	3	0
4	2015-01-27 02:12:36	2	1	0

4. Exploratory Data Analysis

We begin by understanding the distribution of the numerical variables in the dataset.

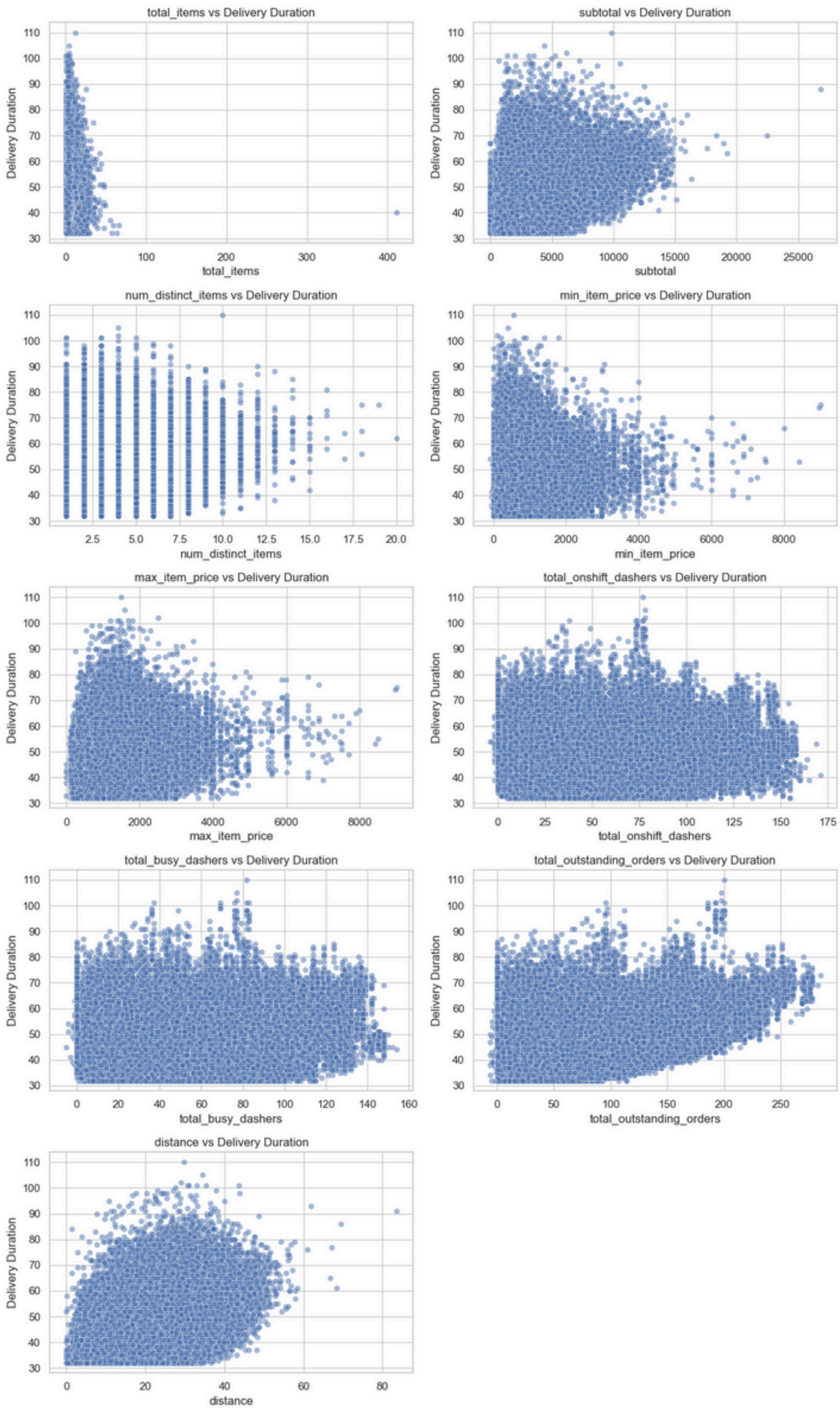


Check the distribution of categorical features.

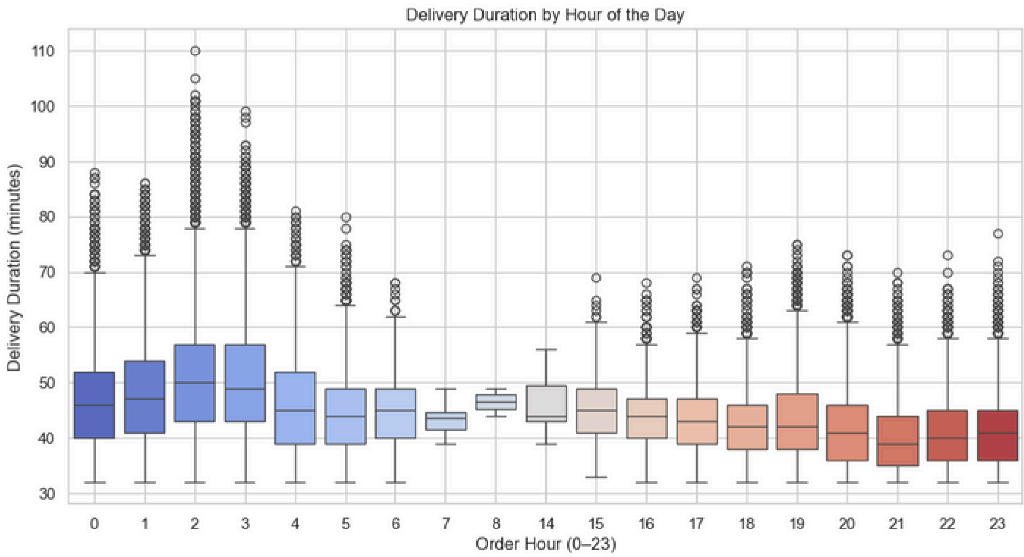


Visualise the distribution of the target variable to understand its spread and any skewness

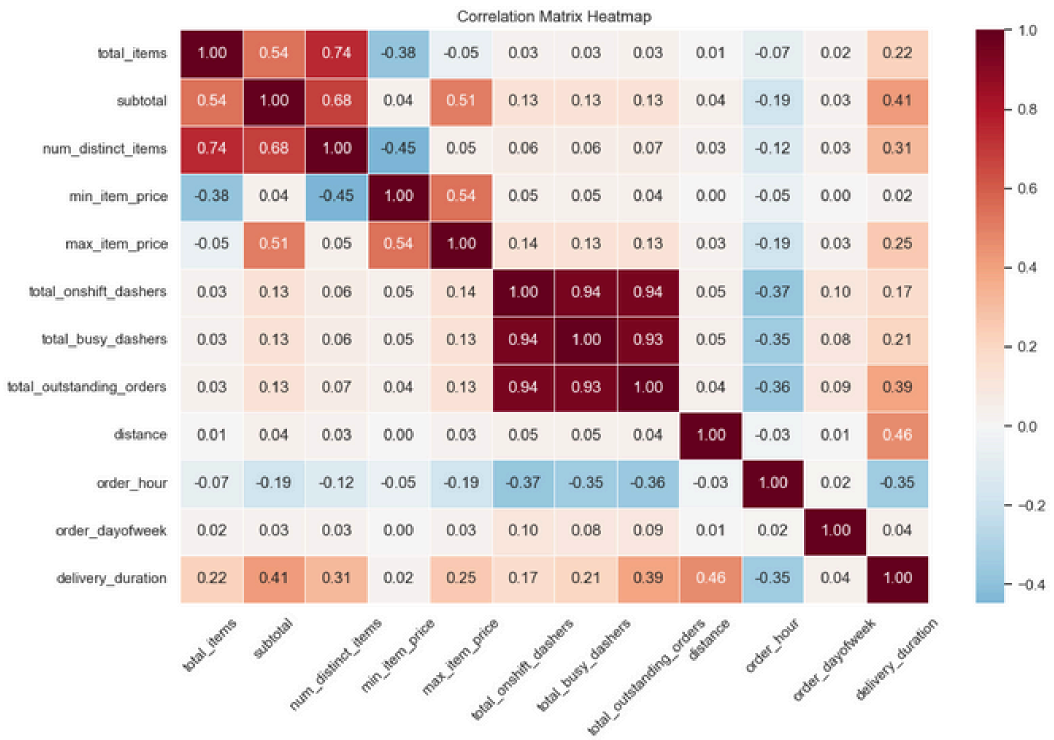
Scatter plots for important numerical and categorical features to observe how they relate to time_taken .



Show the distribution of time_taken for different hours

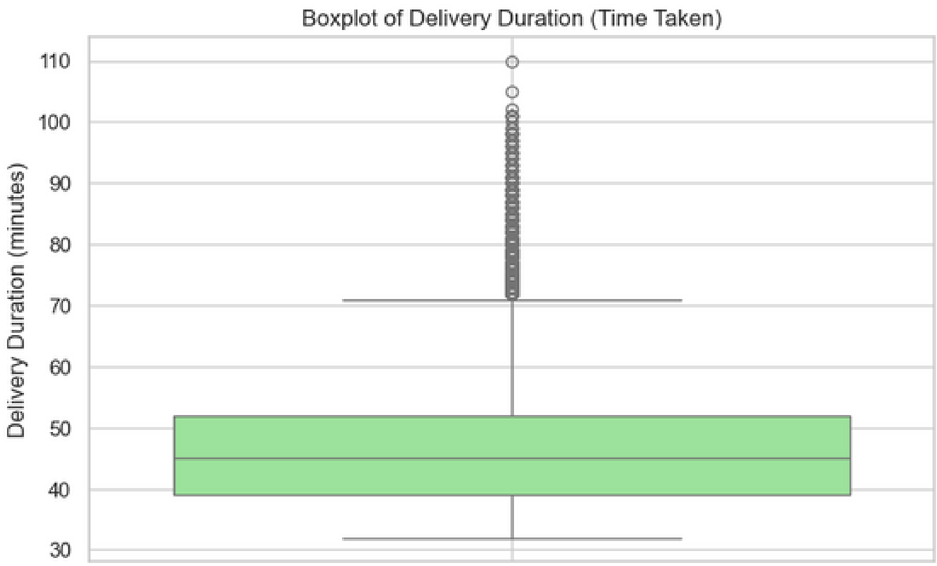


Plot a heatmap to display correlations

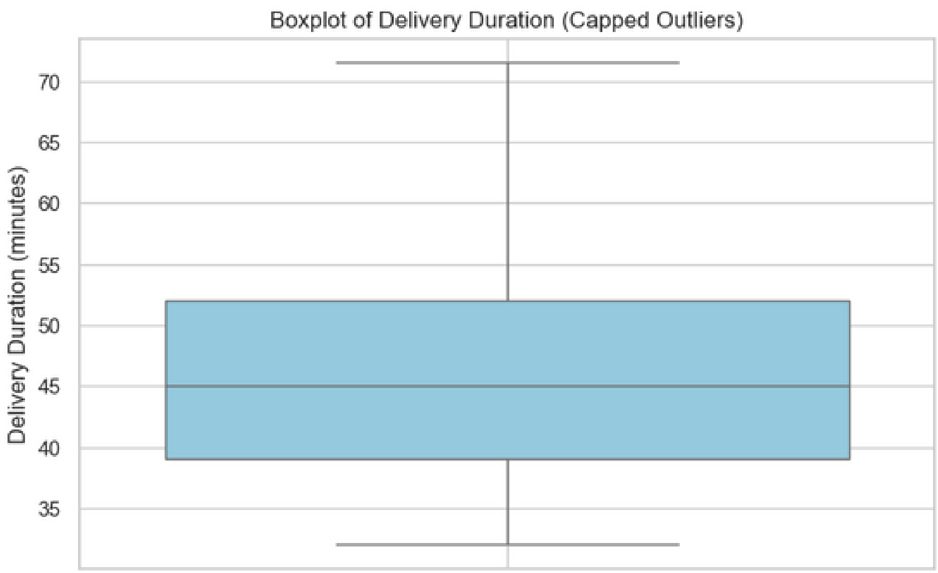


5. Handling the Outliers

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



Handle outliers present in all columns



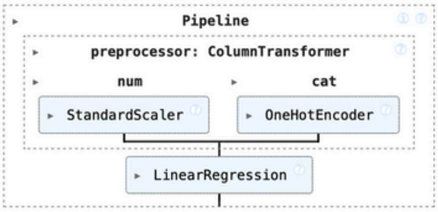
6. Model Building

You can choose from the libraries statsmodels and scikit-learn to build the model.

```
[103]: # Create/Initialise the model
from sklearn.linear_model import LinearRegression

# Initialize the Linear Regression model
lr_model = LinearRegression()
```

```
[105]: # Train the model using the training data

from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression

# Create a pipeline with preprocessing and model
model = Pipeline(steps=[
    ('preprocessor', preprocessor),     # includes scaling + encoding
    ('regressor', LinearRegression())
])

# Train the model
model.fit(X_train, y_train)
```

[105]:



Feature selection with RFE

For RFE, we will start with all features and use the RFE method to recursively reduce the number of features one-by-one.
After analysing the results of these iterations, we select the one that has a good balance between performance and number of features.

RFE Model Performance:
MAE : 3.86
RMSE: 4.96
R^2 : 0.72
3 features - R^2 (CV): 0.468
5 features - R^2 (CV): 0.673
7 features - R^2 (CV): 0.819
9 features - R^2 (CV): 0.843