Order Delivery Time Prediction

Objectives

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

The key goals are:

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

Data Pipeline

The data pipeline for this assignment will involve the following steps:

1. **Data Loading**
2. **Data Preprocessing and Feature Engineering**
3. **Exploratory Data Analysis**
4. **Model Building**
5. **Model Inference**

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.

Field	Description
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.

Importing Necessary Libraries

```
# Import essential libraries for data manipulation and analysis

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option('display.max_columns', None, 'display.max_rows', None)
# Show all columns in DataFrame
# Set the style for seaborn plots
sns.set(style='whitegrid')
```

1. Loading the data

Load 'porter_data_1.csv' as a DataFrame

```
# Importing the file porter_data_1.csv
df = pd.read_csv('porter_data_1.csv')
```

2. Data Preprocessing and Feature Engineering [15 marks]

2.1 Fixing the Datatypes [5 marks]

The current timestamps are in object format and need conversion to datetime format for easier handling and intended functionality

2.1.1 [2 marks]

Convert date and time fields to appropriate data type

```
df.head()

   market_id           created_at actual_delivery_time  \
0        1.0  2015-02-06 22:24:17  2015-02-06 23:11:17
1        2.0  2015-02-10 21:49:25  2015-02-10 22:33:25
```

```
2        2.0  2015-02-16 00:11:35  2015-02-16 01:06:35
3        1.0  2015-02-12 03:36:46  2015-02-12 04:35:46
4        1.0  2015-01-27 02:12:36  2015-01-27 02:58:36

   store_primary_category  order_protocol  total_items  subtotal  \
0                       4             1.0            4      3441
1                      46             2.0            1      1900
2                      36             3.0            4      4771
3                      38             1.0            1      1525
4                      38             1.0            2      3620

   num_distinct_items  min_item_price  max_item_price
total_onshift_dashers  \
0                   4             557            1239
33.0
1                   1            1400            1400
1.0
2                   3             820            1604
8.0
3                   1            1525            1525
5.0
4                   2            1425            2195
5.0

   total_busy_dashers  total_outstanding_orders  distance
0                14.0                      21.0     34.44
1                 2.0                       2.0     27.60
2                 6.0                      18.0     11.56
3                 6.0                       8.0     31.80
4                 5.0                       7.0      8.20

# Convert 'created_at' and 'actual_delivery_time' columns to datetime
format

df['created_at'] = pd.to_datetime(df['created_at'])
df['actual_delivery_time'] =
pd.to_datetime(df['actual_delivery_time'])

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 14 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   market_id                 175777 non-null  float64
 1   created_at                175777 non-null  datetime64[ns]
 2   actual_delivery_time      175777 non-null  datetime64[ns]
 3   store_primary_category    175777 non-null  int64
 4   order_protocol            175777 non-null  float64
```

```
 5   total_items                175777 non-null  int64
 6   subtotal                   175777 non-null  int64
 7   num_distinct_items         175777 non-null  int64
 8   min_item_price             175777 non-null  int64
 9   max_item_price             175777 non-null  int64
 10  total_onshift_dashers      175777 non-null  float64
 11  total_busy_dashers         175777 non-null  float64
 12  total_outstanding_orders   175777 non-null  float64
 13  distance                   175777 non-null  float64
dtypes: datetime64[ns](2), float64(6), int64(6)
memory usage: 18.8 MB
```

`total_busy_dashers, total_busy_dashers, total_outstanding_orders` should be in int64 not float64, can't be in decimal

```
df[['total_onshift_dashers', 'total_busy_dashers',
'total_outstanding_orders', 'market_id','order_protocol']] =
df[['total_onshift_dashers', 'total_busy_dashers',
'total_outstanding_orders','market_id','order_protocol']].astype('int6
4')

df.head()

   market_id          created_at actual_delivery_time
store_primary_category  \
0          1 2015-02-06 22:24:17  2015-02-06 23:11:17
4
1          2 2015-02-10 21:49:25  2015-02-10 22:33:25
46
2          2 2015-02-16 00:11:35  2015-02-16 01:06:35
36
3          1 2015-02-12 03:36:46  2015-02-12 04:35:46
38
4          1 2015-01-27 02:12:36  2015-01-27 02:58:36
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
```

2.1.2 [3 marks]

Convert categorical fields to appropriate data type

```
df.market_id.value_counts().sort_index()

market_id
1    37115
2    53469
3    21075
4    46222
5    17258
6      638
Name: count, dtype: int64

# Convert categorical features to category type

df[['market_id','store_primary_category','order_protocol']] =
df[['market_id','store_primary_category','order_protocol']].astype('ca
tegory')
```

2.2 Feature Engineering [5 marks]

Calculate the time taken to execute the delivery as well as extract the hour and day at which the order was placed

2.2.1 [2 marks]

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

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

df.head()

  market_id          created_at actual_delivery_time
store_primary_category  \
```

```
0          1 2015-02-06 22:24:17  2015-02-06 23:11:17
4
1          2 2015-02-10 21:49:25  2015-02-10 22:33:25
46
2          2 2015-02-16 00:11:35  2015-02-16 01:06:35
36
3          1 2015-02-12 03:36:46  2015-02-12 04:35:46
38
4          1 2015-01-27 02:12:36  2015-01-27 02:58:36
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  mints_taken
0                        21     34.44         47.0
1                         2     27.60         44.0
2                        18     11.56         55.0
3                         8     31.80         59.0
4                         7      8.20         46.0
```

2.2.2 [3 marks]

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

```
# Extract the hour and day of week from the 'created_at' timestamp

df['created_hour'] = df['created_at'].dt.hour
df['is_weekend'] = df['created_at'].dt.dayofweek.apply(lambda x: 0 if
x < 5 else 1)
df['day_of_week'] = df['created_at'].dt.dayofweek
# Create a categorical feature 'isWeekend'
```

```
df['is_weekend'] = df['is_weekend'].astype('category')

df.head()
```

```
  market_id          created_at actual_delivery_time
store_primary_category  \
0          1 2015-02-06 22:24:17  2015-02-06 23:11:17
4
1          2 2015-02-10 21:49:25  2015-02-10 22:33:25
46
2          2 2015-02-16 00:11:35  2015-02-16 01:06:35
36
3          1 2015-02-12 03:36:46  2015-02-12 04:35:46
38
4          1 2015-01-27 02:12:36  2015-01-27 02:58:36
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  mints_taken  created_hour
is_weekend  \
0                        21     34.44         47.0            22
0
1                         2     27.60         44.0            21
0
2                        18     11.56         55.0             0
0
3                         8     31.80         59.0             3
0
4                         7      8.20         46.0             2
0
```

```
   day_of_week
0            4
1            1
2            0
3            3
4            1
```

There's not much need to drop any column, maybe `actual_delivery_time` bcoz the prediction done on delivery time, best to do that duration from order `created_time`

```python
# Drop unnecessary columns
df = df.drop(columns=['actual_delivery_time','created_at'])

print(df['store_primary_category'].value_counts() ,df['order_protocol'
].value_counts(), df['market_id'].value_counts())
```

```
store_primary_category
4     18183
55    15745
46    15586
13     9915
58     8995
20     8563
39     8232
24     8085
38     6733
28     6495
36     6378
68     6235
72     5570
45     5138
10     4841
50     3714
57     3468
34     2951
7      2672
59     2590
6      2219
66     2103
15     2027
2      1745
40     1693
18     1550
61     1525
47     1457
35     1451
65     1032
25      992
```

```
71       755
14       639
52       625
53       574
30       557
42       490
12       428
23       330
16       310
9        293
49       287
29       259
17       243
70       232
54       230
69       220
31       182
51       139
11       125
67       125
26       110
0        103
44        92
5         70
32        66
62        57
37        55
41        51
63        41
60        30
64        25
19        24
33        24
48        24
22        23
27        22
56        11
1         10
43         9
8          2
3          1
21         1
Name: count, dtype: int64 order_protocol
1    48404
3    47125
5    41415
2    20890
4    17246
6      678
```

```
7        19
Name: count, dtype: int64 market_id
2    53469
4    46222
1    37115
3    21075
5    17258
6      638
Name: count, dtype: int64
```

```
df = pd.get_dummies(columns=['order_protocol','market_id'],
prefix=['order_protocol','market_id'], data=df, drop_first=True)
df.head()
```

```
  store_primary_category  total_items  subtotal  num_distinct_items  \
0                      4            4      3441                   4
1                     46            1      1900                   1
2                     36            4      4771                   3
3                     38            1      1525                   1
4                     38            2      3620                   2

   min_item_price  max_item_price  total_onshift_dashers
total_busy_dashers  \
0             557            1239                     33
14
1            1400            1400                      1
2
2             820            1604                      8
6
3            1525            1525                      5
6
4            1425            2195                      5
5

   total_outstanding_orders  distance  mints_taken  created_hour
is_weekend  \
0                        21     34.44         47.0            22
0
1                         2     27.60         44.0            21
0
2                        18     11.56         55.0             0
0
3                         8     31.80         59.0             3
0
4                         7      8.20         46.0             2
0

   day_of_week  order_protocol_2  order_protocol_3
order_protocol_4  \
0            4             False             False             False
```

```
1              1              True              False              False

2              0             False               True              False

3              3             False              False              False

4              1             False              False              False


   order_protocol_5  order_protocol_6  order_protocol_7  
market_id_2  \
0             False             False             False        False

1             False             False             False         True

2             False             False             False         True

3             False             False             False        False

4             False             False             False        False


   market_id_3  market_id_4  market_id_5  market_id_6
0        False        False        False        False
1        False        False        False        False
2        False        False        False        False
3        False        False        False        False
4        False        False        False        False
```

```
top_categories = 
df['store_primary_category'].value_counts().nlargest(10).index
df['store_primary_category'] = df['store_primary_category'].apply(
    lambda x: x if x in top_categories else 'Other'
)
df = pd.get_dummies(df, columns=['store_primary_category'], 
drop_first=True)
```

2.3 Creating training and validation sets [5 marks]

2.3.1 [2 marks]

Define target and input features

```
df.head()
```

```
   total_items  subtotal  num_distinct_items  min_item_price  
max_item_price  \
0            4      3441                   4             557   
1239   
1            1      1900                   1            1400
```

```
1400
2           4      4771                     3            820
1604
3           1      1525                     1           1525
1525
4           2      3620                     2           1425
2195

   total_onshift_dashers  total_busy_dashers  total_outstanding_orders
\
0                     33                  14                        21

1                      1                   2                         2

2                      8                   6                        18

3                      5                   6                         8

4                      5                   5                         7


   distance  mints_taken  created_hour is_weekend  day_of_week  \
0     34.44         47.0            22          0            4
1     27.60         44.0            21          0            1
2     11.56         55.0             0          0            0
3     31.80         59.0             3          0            3
4      8.20         46.0             2          0            1

   order_protocol_2  order_protocol_3  order_protocol_4
order_protocol_5  \
0             False             False             False
False
1              True             False             False
False
2             False              True             False
False
3             False             False             False
False
4             False             False             False
False

   order_protocol_6  order_protocol_7  market_id_2  market_id_3
market_id_4  \
0             False             False        False        False
False
1             False             False         True        False
False
2             False             False         True        False
False
3             False             False        False        False
```

```
False
4             False             False        False        False
False

   market_id_5  market_id_6  store_primary_category_13  \
0        False        False                      False
1        False        False                      False
2        False        False                      False
3        False        False                      False
4        False        False                      False

   store_primary_category_20  store_primary_category_24  \
0                      False                      False
1                      False                      False
2                      False                      False
3                      False                      False
4                      False                      False

   store_primary_category_28  store_primary_category_38  \
0                      False                      False
1                      False                      False
2                      False                      False
3                      False                       True
4                      False                       True

   store_primary_category_39  store_primary_category_46  \
0                      False                      False
1                      False                       True
2                      False                      False
3                      False                      False
4                      False                      False

   store_primary_category_55  store_primary_category_58  \
0                      False                      False
1                      False                      False
2                      False                      False
3                      False                      False
4                      False                      False

   store_primary_category_Other
0                         False
1                         False
2                          True
3                         False
4                         False

# Define target variable (y) and features (X)
y = df['mints_taken']
X = df.drop(columns=['mints_taken'])
```

```
X.head()

   total_items  subtotal  num_distinct_items  min_item_price
max_item_price  \
0            4      3441                   4             557
1239
1            1      1900                   1            1400
1400
2            4      4771                   3             820
1604
3            1      1525                   1            1525
1525
4            2      3620                   2            1425
2195

   total_onshift_dashers  total_busy_dashers  total_outstanding_orders
\
0                     33                  14                        21

1                      1                   2                         2

2                      8                   6                        18

3                      5                   6                         8

4                      5                   5                         7


   distance  created_hour is_weekend  day_of_week  order_protocol_2  \
0     34.44            22          0            4             False
1     27.60            21          0            1              True
2     11.56             0          0            0             False
3     31.80             3          0            3             False
4      8.20             2          0            1             False

   order_protocol_3  order_protocol_4  order_protocol_5
order_protocol_6  \
0             False             False             False
False
1             False             False             False
False
2              True             False             False
False
3             False             False             False
False
4             False             False             False
False

   order_protocol_7  market_id_2  market_id_3  market_id_4
market_id_5  \
```

```
0              False        False        False        False
False
1              False         True        False        False
False
2              False         True        False        False
False
3              False        False        False        False
False
4              False        False        False        False
False

   market_id_6  store_primary_category_13
store_primary_category_20  \
0        False                      False                      False

1        False                      False                      False

2        False                      False                      False

3        False                      False                      False

4        False                      False                      False


   store_primary_category_24  store_primary_category_28  \
0                      False                      False
1                      False                      False
2                      False                      False
3                      False                      False
4                      False                      False

   store_primary_category_38  store_primary_category_39  \
0                      False                      False
1                      False                      False
2                      False                      False
3                       True                      False
4                       True                      False

   store_primary_category_46  store_primary_category_55  \
0                      False                      False
1                       True                      False
2                      False                      False
3                      False                      False
4                      False                      False

   store_primary_category_58  store_primary_category_Other
0                      False                         False
1                      False                         False
2                      False                          True
```

```
3                       False                         False
4                       False                         False
```

2.3.2 [3 marks]

Split the data into training and test sets

```
# Split data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,
test_size=0.3, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(123043, 33) (52734, 33) (123043,) (52734,)
```

```
X_train.head()
```

```
        total_items  subtotal  num_distinct_items  min_item_price  \
94465             3      4385                   3            1095
100712            4      2772                   3              79
153524            7      1663                   5             139
85660             3      1477                   3             429
100506            1      1134                   1             725

        max_item_price  total_onshift_dashers  total_busy_dashers  \
94465             1395                     84                  63
100712            1349                     22                  16
153524             579                      2                   1
85660              579                     41                  42
100506             725                     60                  61

        total_outstanding_orders  distance  created_hour is_weekend  \
94465                        101     26.72             2          1
100712                        13     22.88             3          1
153524                         0     13.80             5          0
85660                         41      8.76             3          0
100506                        79     12.40            20          0

        day_of_week  order_protocol_2  order_protocol_3
order_protocol_4  \
94465             6             False              True
False
100712            6             False             False
False
153524            1             False             False
False
85660             4             False             False
False
100506            3             False             False
False
```

```
        order_protocol_5  order_protocol_6  order_protocol_7
market_id_2  \
94465              False             False             False
False
100712             False             False             False
False
153524             False             False             False
True
85660               True             False             False
True
100506              True             False             False
True

        market_id_3  market_id_4  market_id_5  market_id_6  \
94465         False         True        False        False
100712         True        False        False        False
153524        False        False        False        False
85660         False        False        False        False
100506        False        False        False        False

        store_primary_category_13  store_primary_category_20  \
94465                       False                       True
100712                      False                      False
153524                      False                      False
85660                       False                      False
100506                      False                      False

        store_primary_category_24  store_primary_category_28  \
94465                       False                      False
100712                      False                      False
153524                      False                      False
85660                       False                      False
100506                      False                      False

        store_primary_category_38  store_primary_category_39  \
94465                       False                      False
100712                      False                      False
153524                      False                      False
85660                       False                      False
100506                      False                      False

        store_primary_category_46  store_primary_category_55  \
94465                       False                      False
100712                       True                      False
153524                       True                      False
85660                       False                      False
100506                      False                      False

        store_primary_category_58  store_primary_category_Other
```

```
94465                          False                          False
100712                         False                          False
153524                         False                          False
85660                          False                          False
100506                          True                          False
```

3. Exploratory Data Analysis on Training Data [20 marks]

1. Analyzing the correlation between variables to identify patterns and relationships
2. Identifying and addressing outliers to ensure the integrity of the analysis
3. Exploring the relationships between variables and examining the distribution of the data for better insights

3.1 Feature Distributions [7 marks]

```
df.columns.to_list()
```

```
['total_items',
 'subtotal',
 'num_distinct_items',
 'min_item_price',
 'max_item_price',
 'total_onshift_dashers',
 'total_busy_dashers',
 'total_outstanding_orders',
 'distance',
 'mints_taken',
 'created_hour',
 'is_weekend',
 'day_of_week',
 'order_protocol_2',
 'order_protocol_3',
 'order_protocol_4',
 'order_protocol_5',
 'order_protocol_6',
 'order_protocol_7',
 'market_id_2',
 'market_id_3',
 'market_id_4',
 'market_id_5',
 'market_id_6',
 'store_primary_category_13',
 'store_primary_category_20',
 'store_primary_category_24',
 'store_primary_category_28',
 'store_primary_category_38',
 'store_primary_category_39',
 'store_primary_category_46',
 'store_primary_category_55',
```

```
 'store_primary_category_58',
 'store_primary_category_Other']
```

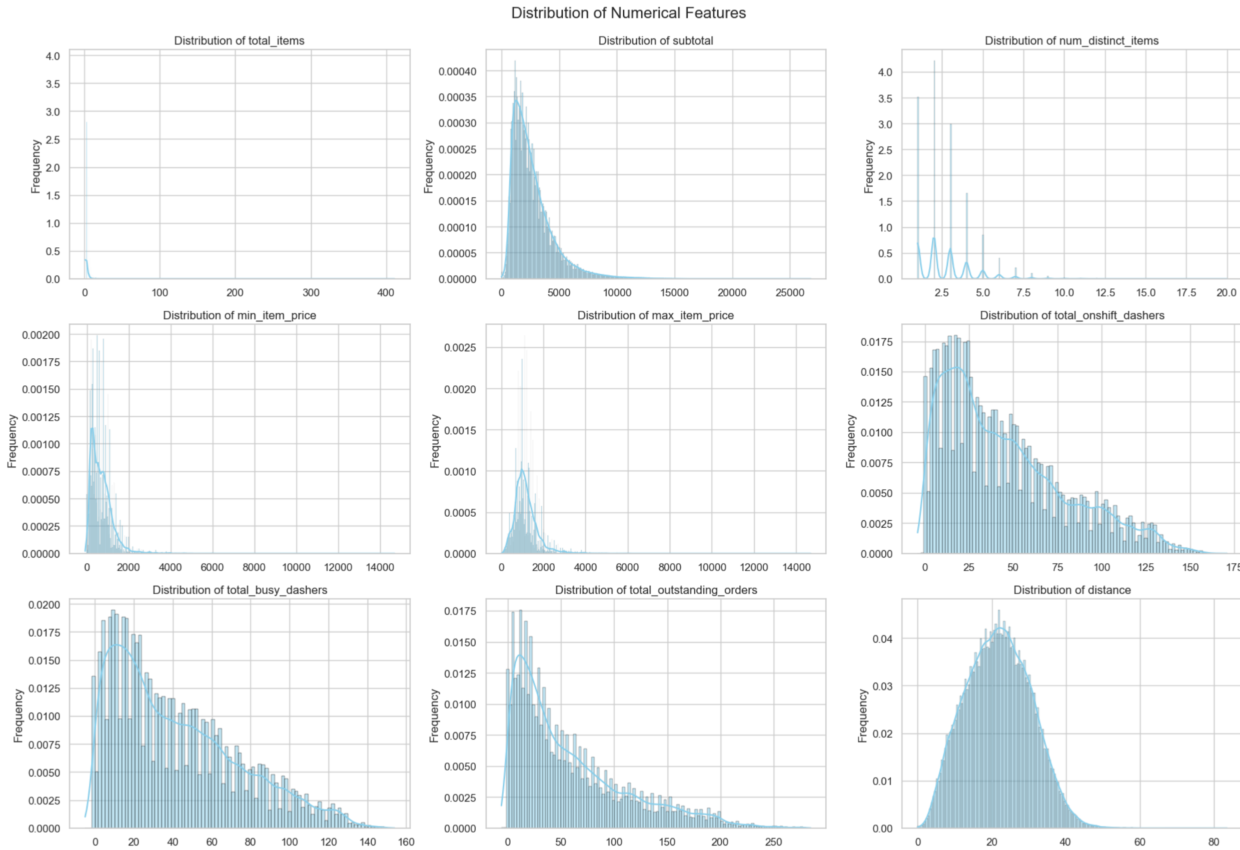
```python
# Define numerical and categorical columns for easy EDA and data
manipulation
numerical =
['total_items','subtotal','num_distinct_items','min_item_price','max_i
tem_price','total_onshift_dashers',

'total_busy_dashers','total_outstanding_orders','distance']

categorical = ['is_weekend','day_of_week','created_hour']

numerical
```

```
['total_items',
 'subtotal',
 'num_distinct_items',
 'min_item_price',
 'max_item_price',
 'total_onshift_dashers',
 'total_busy_dashers',
 'total_outstanding_orders',
 'distance']
```

3.1.1 [3 marks]

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

```python
# Plot distributions for all numerical columns

# Create subplots (3 rows × 3 columns)
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(18, 12))
axes = axes.flatten()  # flatten 2D axes to 1D list

# Plot each histogram
for i, col in enumerate(numerical):
    sns.histplot(df[col], kde=True, ax=axes[i], color='skyblue',
edgecolor='black', stat='density')
    axes[i].set_title(f'Distribution of {col}', fontsize=12)
    axes[i].set_xlabel('')
    axes[i].set_ylabel('Frequency')

# Adjust layout
plt.tight_layout()
plt.suptitle('Distribution of Numerical Features', fontsize=16,
y=1.02)
plt.show()
```



3.1.2 [2 marks]

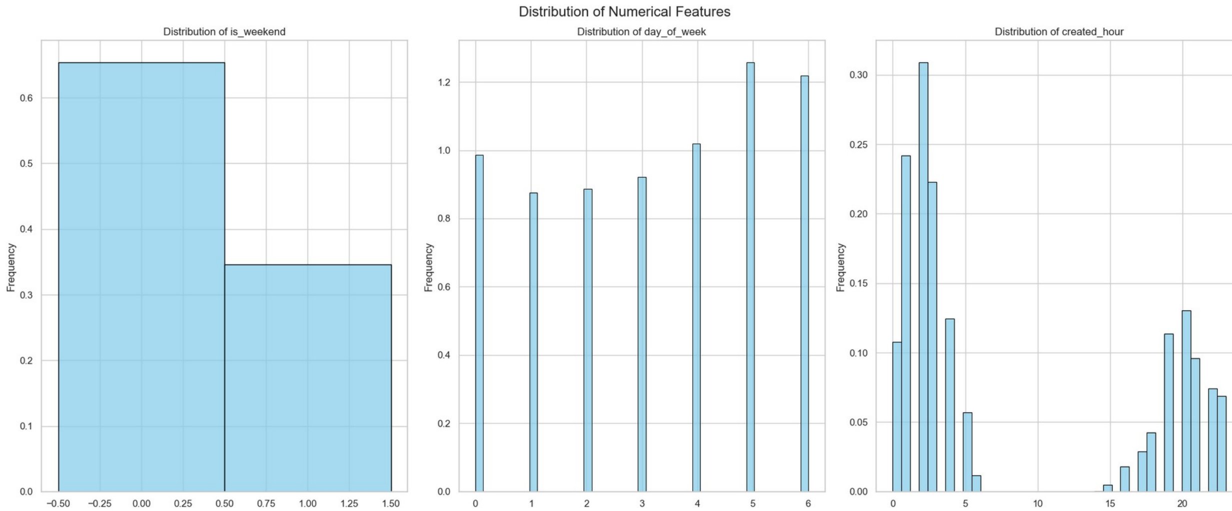
Check the distribution of categorical features

```
df.head()

   total_items  subtotal  num_distinct_items  min_item_price
max_item_price  \
0            4      3441                   4             557
1239
1            1      1900                   1            1400
1400
2            4      4771                   3             820
1604
3            1      1525                   1            1525
1525
4            2      3620                   2            1425
2195

   total_onshift_dashers  total_busy_dashers  total_outstanding_orders
\
0                     33                  14                        21

1                      1                   2                         2
```

```
2                      8                   6                         18

3                      5                   6                          8

4                      5                   5                          7


   distance  mints_taken  created_hour is_weekend  day_of_week  \
0     34.44         47.0            22          0            4
1     27.60         44.0            21          0            1
2     11.56         55.0             0          0            0
3     31.80         59.0             3          0            3
4      8.20         46.0             2          0            1

   order_protocol_2  order_protocol_3  order_protocol_4
order_protocol_5  \
0             False             False             False
False
1              True             False             False
False
2             False              True             False
False
3             False             False             False
False
4             False             False             False
False

   order_protocol_6  order_protocol_7  market_id_2  market_id_3
market_id_4  \
0             False             False        False        False
False
1             False             False         True        False
False
2             False             False         True        False
False
3             False             False        False        False
False
4             False             False        False        False
False

   market_id_5  market_id_6  store_primary_category_13  \
0        False        False                      False
1        False        False                      False
2        False        False                      False
3        False        False                      False
4        False        False                      False

   store_primary_category_20  store_primary_category_24  \
0                      False                      False
1                      False                      False
```

```
2                        False                        False
3                        False                        False
4                        False                        False

   store_primary_category_28  store_primary_category_38  \
0                        False                        False
1                        False                        False
2                        False                        False
3                        False                         True
4                        False                         True

   store_primary_category_39  store_primary_category_46  \
0                        False                        False
1                        False                         True
2                        False                        False
3                        False                        False
4                        False                        False

   store_primary_category_55  store_primary_category_58  \
0                        False                        False
1                        False                        False
2                        False                        False
3                        False                        False
4                        False                        False

   store_primary_category_Other
0                           False
1                           False
2                            True
3                           False
4                           False
```

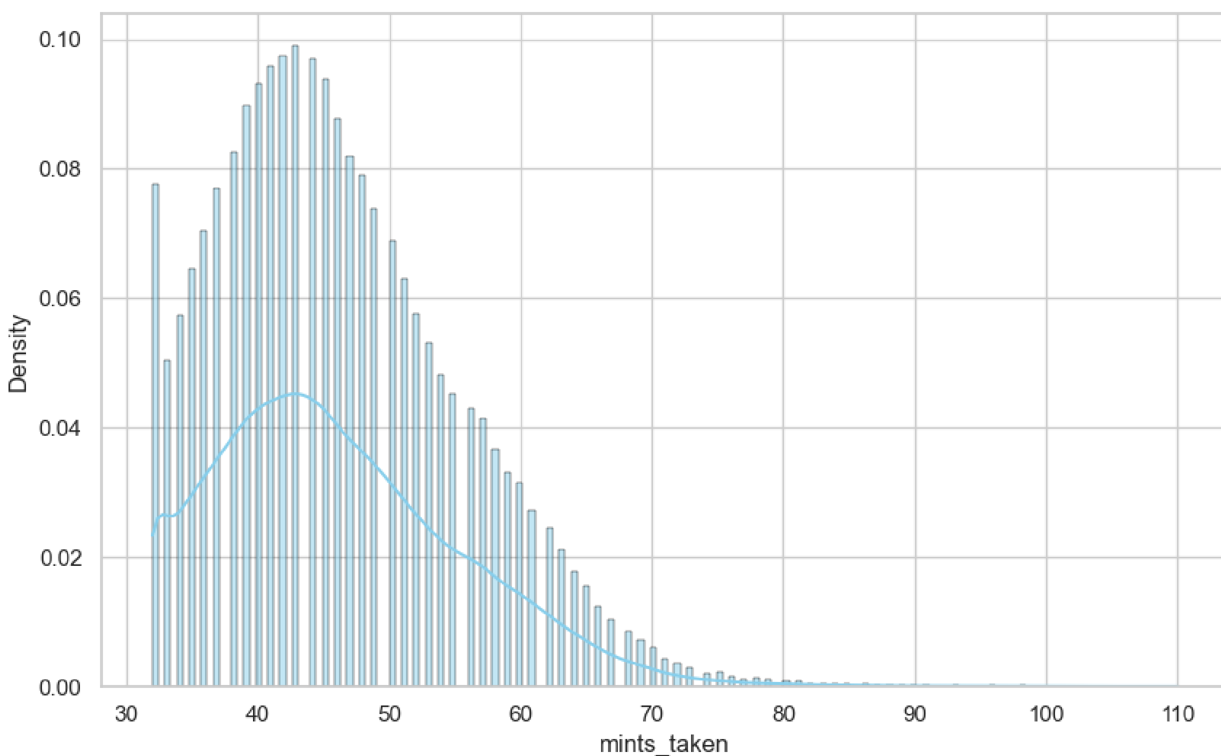
```python
# Distribution of categorical columns

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(20, 8))
axes = axes.flatten()  # flatten 2D axes to 1D list

# Plot each histogram
for i, col in enumerate(categorical):
    sns.histplot(df[col], ax=axes[i], color='skyblue',
edgecolor='black', stat='density')
    axes[i].set_title(f'Distribution of {col}', fontsize=12)
    axes[i].set_xlabel('')
    axes[i].set_ylabel('Frequency')

# Adjust layout
plt.tight_layout()
plt.suptitle('Distribution of Numerical Features', fontsize=16,
y=1.02)
plt.show()
```



3.1.3 [2 mark]

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

```
# Distribution of time_taken

plt.figure(figsize=(10, 6))
sns.histplot(y, kde=True, color='skyblue', edgecolor='black',
stat='density')
plt.show()
```

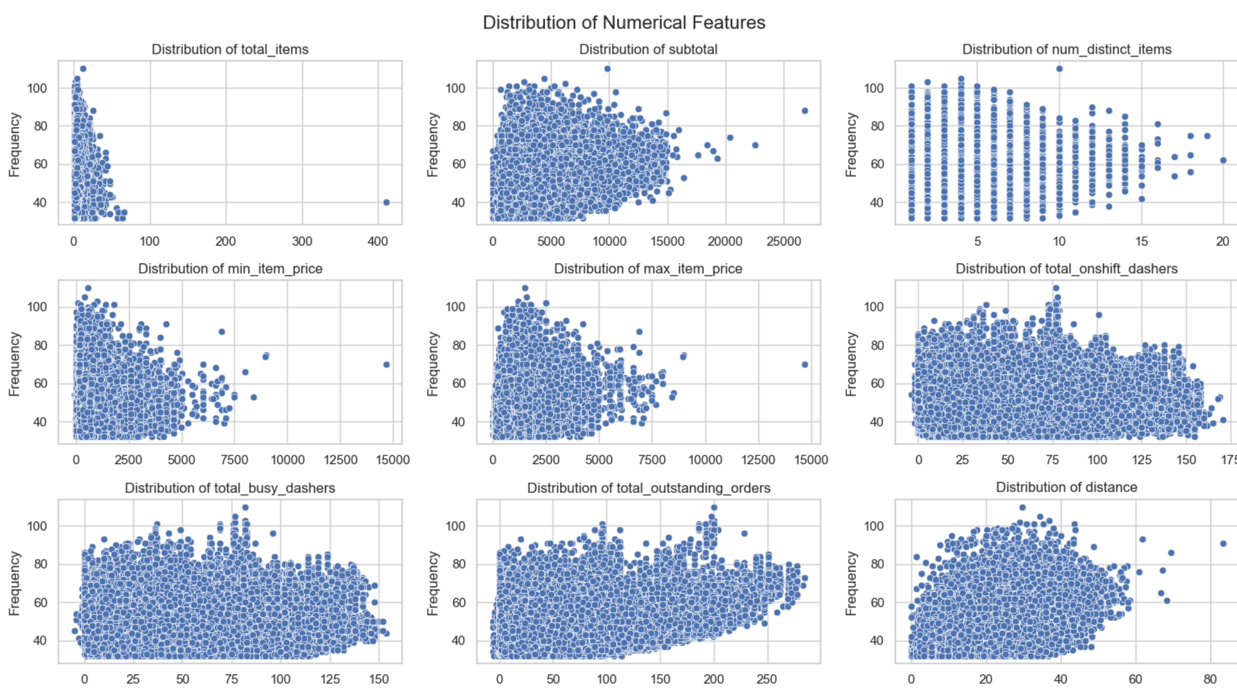


3.2 Relationships Between Features [3 marks]

3.2.1 [3 marks]

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

```
# Scatter plot to visualise the relationship between time_taken and
other features

fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 8))
axes = axes.flatten()  # flatten 2D axes to 1D list

# Plot each histogram
for i, col in enumerate(numerical):
    sns.scatterplot(x=df[col], y=y, ax=axes[i])
    axes[i].set_title(f'Distribution of {col}', fontsize=12)
    axes[i].set_xlabel('')
    axes[i].set_ylabel('Frequency')

# Adjust layout
plt.tight_layout()
plt.suptitle('Distribution of Numerical Features', fontsize=16,
y=1.02)
plt.show()
```

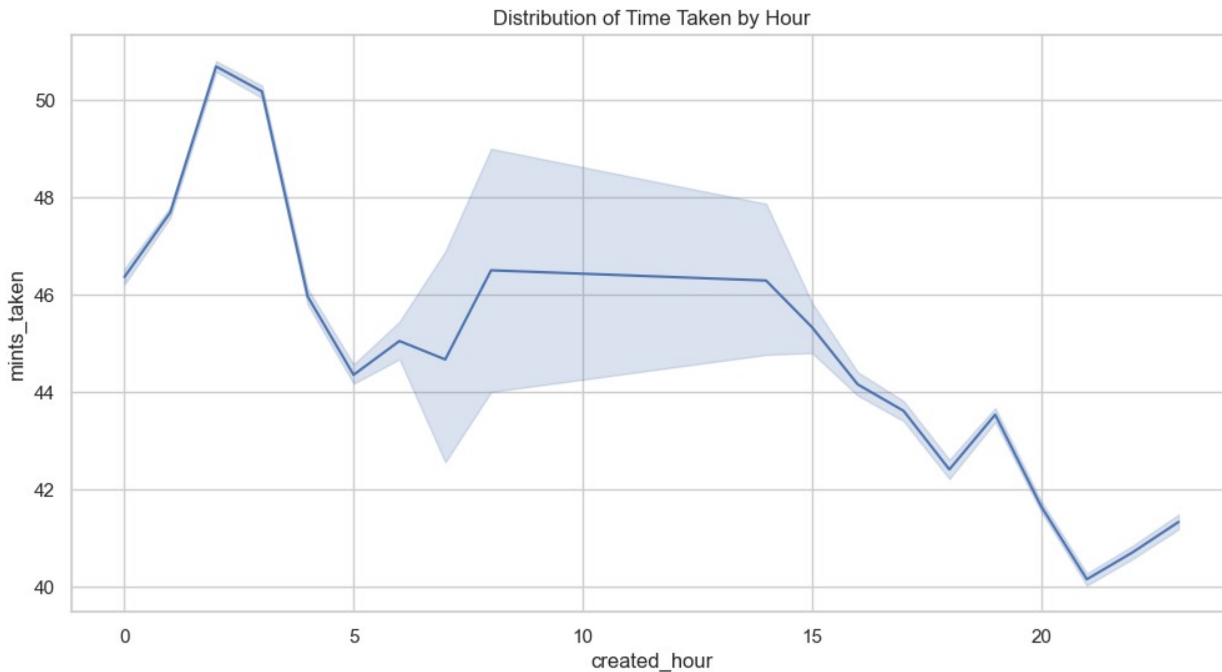



```
df.head()
```

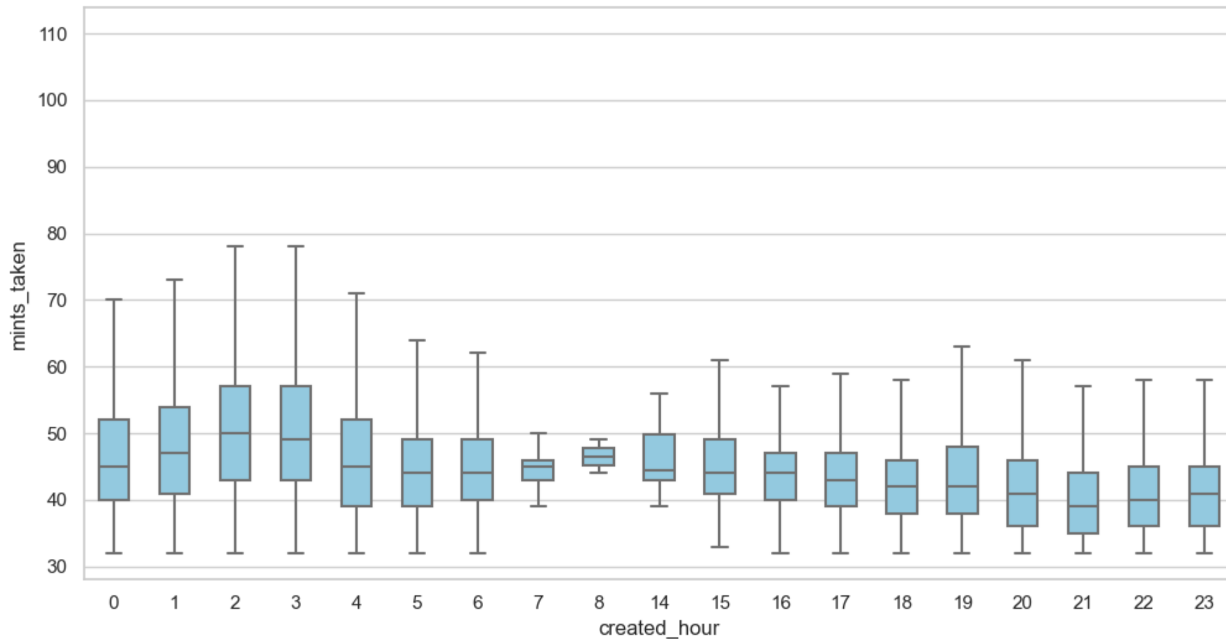
```
   total_items  subtotal  num_distinct_items  min_item_price  
max_item_price  \
0            4      3441                   4             557  
1239
1            1      1900                   1            1400  
1400
2            4      4771                   3             820  
1604
3            1      1525                   1            1525  
1525
4            2      3620                   2            1425  
2195

   total_onshift_dashers  total_busy_dashers  total_outstanding_orders  
\
0                     33                  14                        21  

1                      1                   2                         2  

2                      8                   6                        18  

3                      5                   6                         8  

4                      5                   5                         7  


   distance  mints_taken  created_hour is_weekend  day_of_week  \
0     34.44         47.0            22          0            4   
1     27.60         44.0            21          0            1   
2     11.56         55.0             0          0            0   
3     31.80         59.0             3          0            3   
4      8.20         46.0             2          0            1   

   order_protocol_2  order_protocol_3  order_protocol_4  
order_protocol_5  \
0             False             False             False  
False
1              True             False             False  
False
2             False              True             False  
False
3             False             False             False  
False
4             False             False             False  
False

   order_protocol_6  order_protocol_7  market_id_2  market_id_3  
market_id_4  \
0             False             False        False        False  
False
1             False             False         True        False  
```

```
False
2              False              False         True        False
False
3              False              False        False        False
False
4              False              False        False        False
False

   market_id_5  market_id_6  store_primary_category_13  \
0        False        False                      False
1        False        False                      False
2        False        False                      False
3        False        False                      False
4        False        False                      False

   store_primary_category_20  store_primary_category_24  \
0                      False                      False
1                      False                      False
2                      False                      False
3                      False                      False
4                      False                      False

   store_primary_category_28  store_primary_category_38  \
0                      False                      False
1                      False                      False
2                      False                      False
3                      False                       True
4                      False                       True

   store_primary_category_39  store_primary_category_46  \
0                      False                      False
1                      False                       True
2                      False                      False
3                      False                      False
4                      False                      False

   store_primary_category_55  store_primary_category_58  \
0                      False                      False
1                      False                      False
2                      False                      False
3                      False                      False
4                      False                      False

   store_primary_category_Other
0                         False
1                         False
2                          True
3                         False
4                         False
```

```python
# Show the distribution of time_taken for different hours

plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x='created_hour', y=y )
plt.title('Distribution of Time Taken by Hour')
plt.show()
```




```python
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='created_hour', y=y, color='skyblue',
fliersize=0, linewidth=1.5, width=0.5)
plt.show()
```



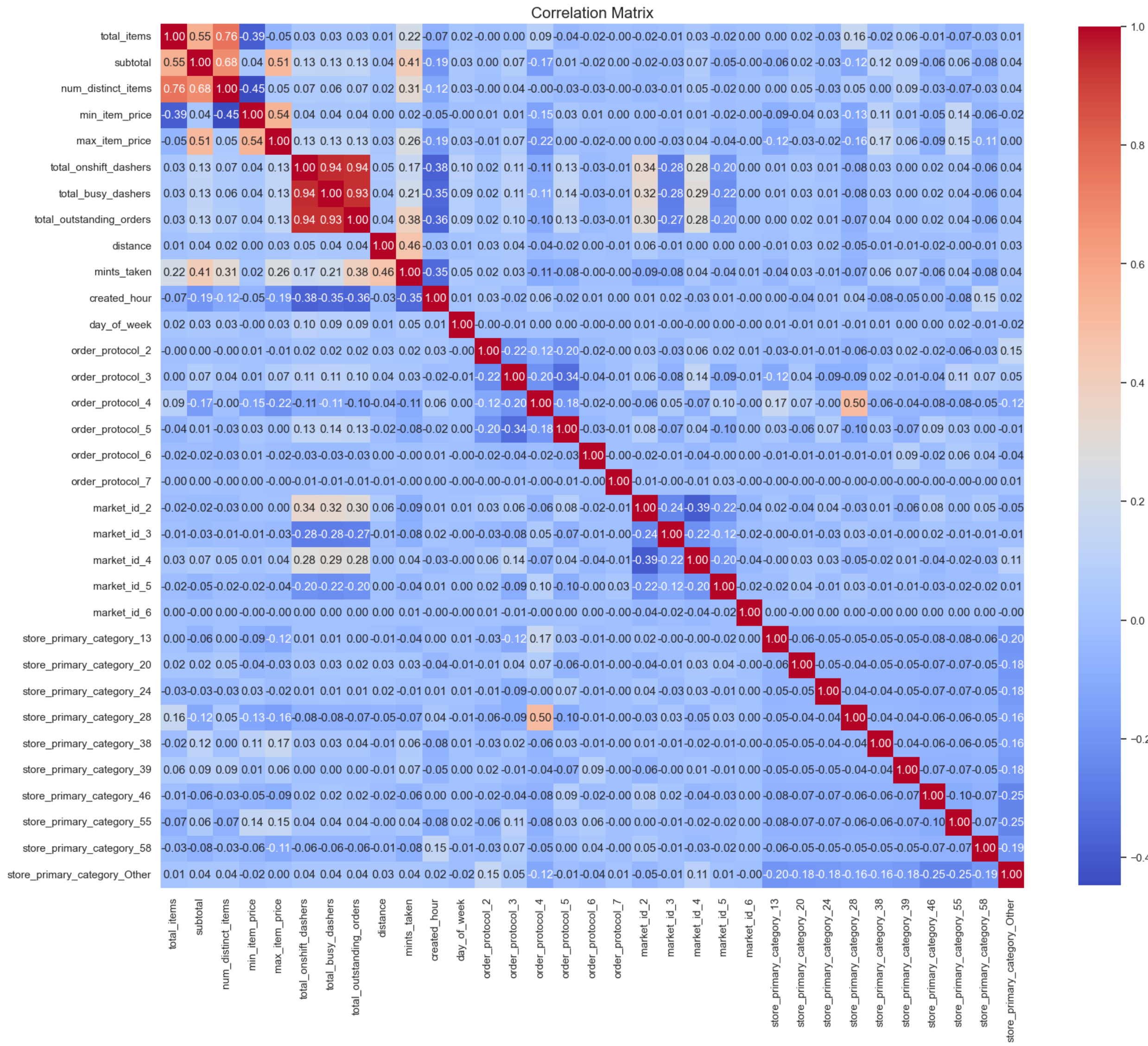
3.3 Correlation Analysis [5 marks]

Check correlations between numerical features to identify which variables are strongly related to `time_taken`

3.3.1 [3 marks]

Plot a heatmap to display correlations

```
# Plot the heatmap of the correlation matrix

corr_matrix = df.corr(numeric_only=True)
target_col = 'mints_taken'

target_corr = corr_matrix[target_col].abs().sort_values()

plt.figure(figsize=(20,20))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm',
square=True, cbar_kws={"shrink": .8})
plt.title('Correlation Matrix', fontsize=16)
plt.show()
```



```
target_corr[target_corr <0.1]

order_protocol_6                 0.001738
order_protocol_7                 0.004744
market_id_6                      0.005360
store_primary_category_24        0.009810
order_protocol_2                 0.019410
min_item_price                   0.022753
store_primary_category_20        0.026218
order_protocol_3                 0.028757
market_id_4                      0.036712
store_primary_category_Other     0.036989
store_primary_category_55        0.041729
store_primary_category_13        0.042540
market_id_5                      0.042567
day_of_week                      0.045878
```

```
store_primary_category_46       0.055209
store_primary_category_38       0.062337
store_primary_category_39       0.069962
store_primary_category_28       0.074015
market_id_3                     0.082070
store_primary_category_58       0.082774
order_protocol_5                0.084833
market_id_2                     0.094109
Name: mints_taken, dtype: float64
```

In correlation usually values less tha 0.4 are considered weak, but that removes too much columns, we are going to remove columns which have correlated values les than 0.1.

3.3.2 [2 marks]

Drop the columns with weak correlations with the target variable

```
# Drop 3-5 weakly correlated columns from training dataset

weak_columns = target_corr[target_corr < 0.1].index.tolist()
print(f"Columns with weak correlation to target: {weak_columns}")
X_train = X_train.drop(columns=weak_columns)
```

```
Columns with weak correlation to target: ['order_protocol_6',
'order_protocol_7', 'market_id_6', 'store_primary_category_24',
'order_protocol_2', 'min_item_price', 'store_primary_category_20',
'order_protocol_3', 'market_id_4', 'store_primary_category_Other',
'store_primary_category_55', 'store_primary_category_13',
'market_id_5', 'day_of_week', 'store_primary_category_46',
'store_primary_category_38', 'store_primary_category_39',
'store_primary_category_28', 'market_id_3',
'store_primary_category_58', 'order_protocol_5', 'market_id_2']
```

```
X_train.columns.to_list()
```

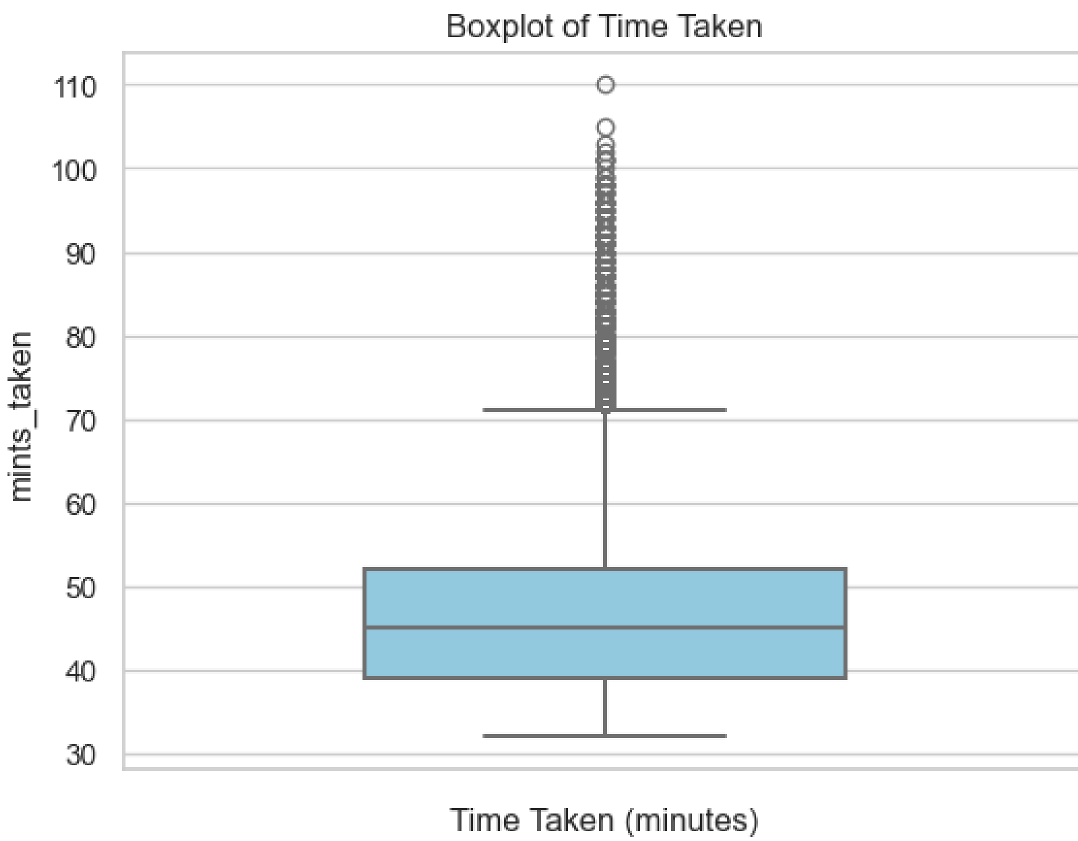
```
['total_items',
 'subtotal',
 'num_distinct_items',
 'max_item_price',
 'total_onshift_dashers',
 'total_busy_dashers',
 'total_outstanding_orders',
 'distance',
 'created_hour',
 'is_weekend',
 'order_protocol_4']
```

3.4 Handling the Outliers [5 marks]

3.4.1 [2 marks]

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

```
# Boxplot for time_taken

sns.boxplot(y=df['mints_taken'], color='skyblue', linewidth=1.5,
width=0.5)
plt.title('Boxplot of Time Taken')
plt.xlabel('Time Taken (minutes)')
plt.show()
```

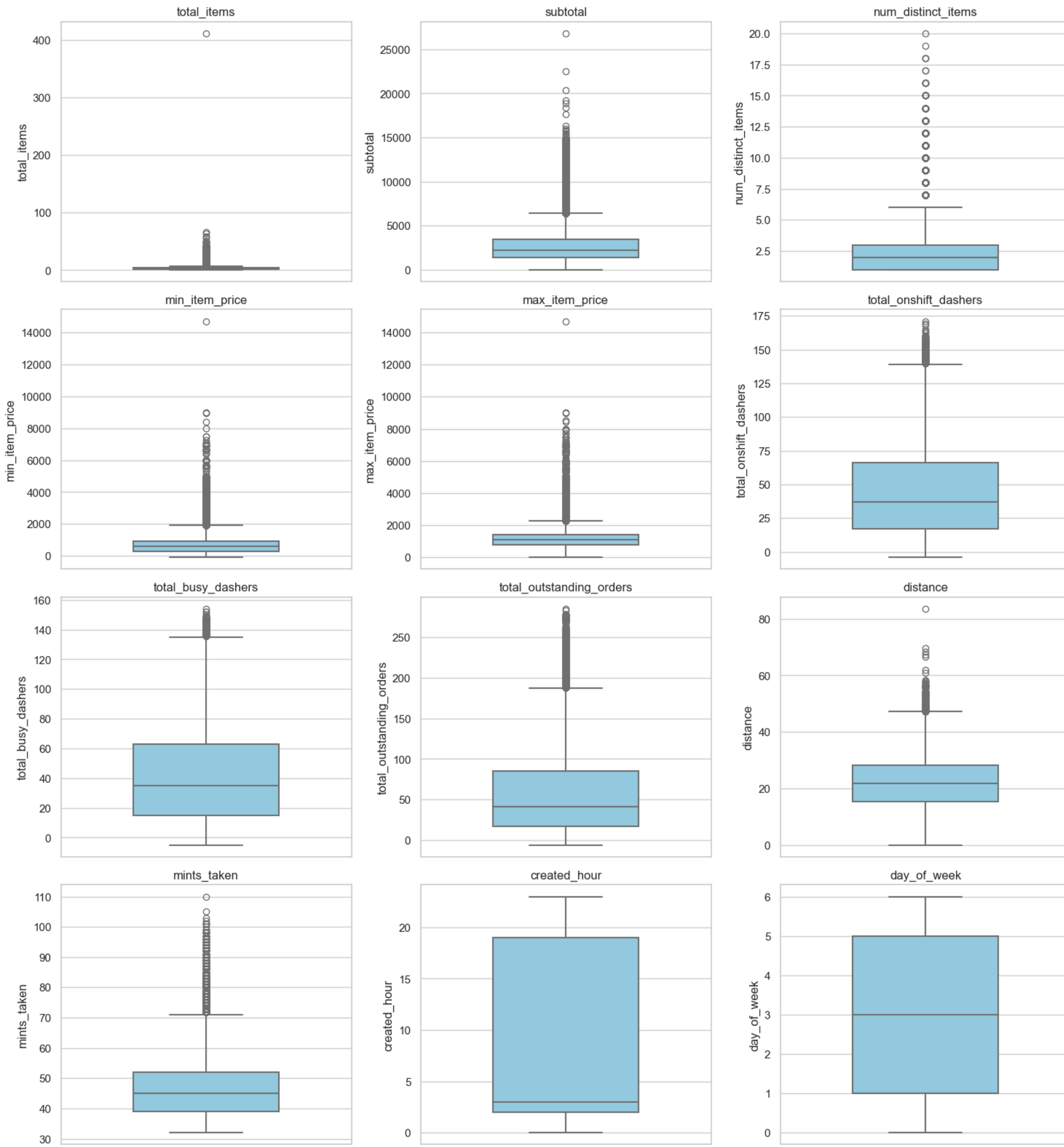


3.4.2 [3 marks]

Handle outliers present in all columns

```
# Handle outliers

numeric_cols = df.select_dtypes(include='number').columns

# Set number of plots
n_cols = 3
```

```python
n_rows = -(-len(numeric_cols) // n_cols)  # Ceiling division

# Set plot size
plt.figure(figsize=(5 * n_cols, 4 * n_rows))

for idx, col in enumerate(numeric_cols, 1):
    plt.subplot(n_rows, n_cols, idx)
    sns.boxplot(y=df[col], color='skyblue', linewidth=1.5, width=0.5)
    plt.title(col)

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

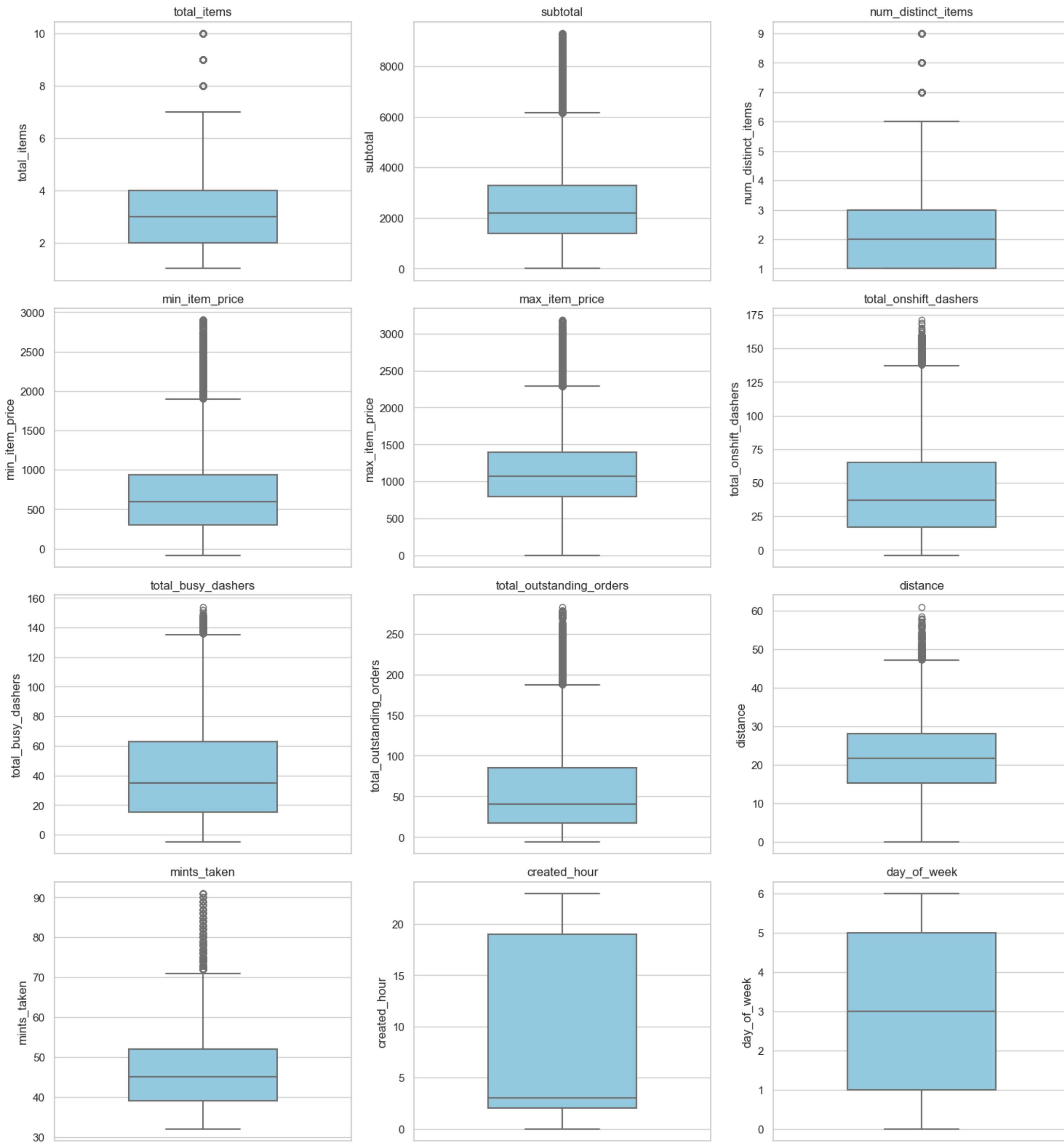



```python
def remove_outliers(df, columns, factor=3):
    df_clean = df.copy()
    for col in columns:
        Q1 = df_clean[col].quantile(0.25)
        Q3 = df_clean[col].quantile(0.75)
        IQR = Q3 - Q1
        lower = Q1 - factor * IQR
        upper = Q3 + factor * IQR
        df_clean = df_clean[(df_clean[col] >= lower) & (df_clean[col]
<= upper)]
```

```
    return df_clean

numeric_cols = df.select_dtypes(include='number').columns.tolist()
df_cleaned = remove_outliers(df, numeric_cols, factor=3)
df_cleaned.shape
```

```
(169730, 34)
```

```
# Set number of plots
n_cols = 3
n_rows = -(-len(numeric_cols) // n_cols)  # Ceiling division

# Set plot size
plt.figure(figsize=(5 * n_cols, 4 * n_rows))

for idx, col in enumerate(numeric_cols, 1):
    plt.subplot(n_rows, n_cols, idx)
    sns.boxplot(y=df_cleaned[col], color='skyblue', linewidth=1.5,
width=0.5)
    plt.title(col)

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



4. Exploratory Data Analysis on Validation Data [optional]

Optionally, perform EDA on test data to see if the distribution match with the training data

```
# Define numerical and categorical columns for easy EDA and data
manipulation
```

4.1 Feature Distributions

4.1.1

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

```
# Plot distributions for all numerical columns
```

4.1.2

Check the distribution of categorical features

```
# Distribution of categorical columns
```

4.1.3

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

```
# Distribution of time_taken
```

4.2 Relationships Between Features

Scatter plots for numerical features to observe how they relate to each other, especially to `time_taken`

```
# Scatter plot to visualise the relationship between time_taken and other features
```

4.3 Drop the columns with weak correlations with the target variable

```
# Drop the weakly correlated columns from training dataset
```

5. Model Building [15 marks]

Import Necessary Libraries

```
# Import libraries
import scipy.stats as stats
import statsmodels.api as sm
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression as LR
from sklearn.metrics import mean_squared_error, r2_score  # Import evaluation metrics
from sklearn.preprocessing import StandardScaler
```

5.1 Feature Scaling [3 marks]

```
df_cleaned[numeric_cols].head()
```

```
   total_items  subtotal  num_distinct_items  min_item_price  
max_item_price  \
0            4      3441                   4             557   
1239   
1            1      1900                   1            1400   
1400   
2            4      4771                   3             820   
1604   
3            1      1525                   1            1525   
1525   
4            2      3620                   2            1425   
2195   

   total_onshift_dashers  total_busy_dashers  total_outstanding_orders  
\
0                     33                  14                        21   

1                      1                   2                         2   

2                      8                   6                        18   

3                      5                   6                         8   

4                      5                   5                         7   


   distance  mints_taken  created_hour  day_of_week  
0     34.44         47.0            22            4  
1     27.60         44.0            21            1  
2     11.56         55.0             0            0  
3     31.80         59.0             3            3  
4      8.20         46.0             2            1  
```

```
numeric_cols
```

```
['total_items',
 'subtotal',
 'num_distinct_items',
 'min_item_price',
 'max_item_price',
 'total_onshift_dashers',
 'total_busy_dashers',
 'total_outstanding_orders',
 'distance',
 'mints_taken',
 'created_hour',
 'day_of_week']
```

```python
print(X_train.columns.to_list())
print(X_test.columns.to_list())
```

```
['total_items', 'subtotal', 'num_distinct_items', 'max_item_price',
'total_onshift_dashers', 'total_busy_dashers',
'total_outstanding_orders', 'distance', 'created_hour', 'is_weekend',
'order_protocol_4']
['total_items', 'subtotal', 'num_distinct_items', 'min_item_price',
'max_item_price', 'total_onshift_dashers', 'total_busy_dashers',
'total_outstanding_orders', 'distance', 'created_hour', 'is_weekend',
'day_of_week', 'order_protocol_2', 'order_protocol_3',
'order_protocol_4', 'order_protocol_5', 'order_protocol_6',
'order_protocol_7', 'market_id_2', 'market_id_3', 'market_id_4',
'market_id_5', 'market_id_6', 'store_primary_category_13',
'store_primary_category_20', 'store_primary_category_24',
'store_primary_category_28', 'store_primary_category_38',
'store_primary_category_39', 'store_primary_category_46',
'store_primary_category_55', 'store_primary_category_58',
'store_primary_category_Other']
```

```python
X_train['is_weekend'] = X_train['is_weekend'].astype(int)
X_train['order_protocol_4'] = X_train['order_protocol_4'].astype(int)

# Apply scaling to the numerical columns
X_test = X_test[X_train.columns]
num_cols = X_train.select_dtypes(include='number').columns.to_list()
scaler = StandardScaler()

X_train_unscaled = X_train.copy()
X_test_unscaled = X_test.copy()

X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols] = scaler.transform(X_test[num_cols])

# X_train = pd.DataFrame(scaler.fit_transform(X_train),
columns=num_cols, index=X_train.index)
# X_test = pd.DataFrame(scaler.transform(X_test), columns=num_cols,
index=X_test.index)

num_cols
```

```
['total_items',
 'subtotal',
 'num_distinct_items',
 'max_item_price',
 'total_onshift_dashers',
 'total_busy_dashers',
 'total_outstanding_orders',
 'distance',
 'created_hour',
```

```
 'is_weekend',
 'order_protocol_4']

X_train.head()

        total_items  subtotal  num_distinct_items  max_item_price  \
94465     -0.075210  0.920885            0.198861        0.420814
100712     0.287056  0.040812            0.198861        0.338664
153524     1.373857 -0.564273            1.427533       -1.036469
85660     -0.075210 -0.665757            0.198861       -1.036469
100506    -0.799744 -0.852902           -1.029812       -0.775729

        total_onshift_dashers  total_busy_dashers
total_outstanding_orders  \
94465                1.130856            0.656539
0.810319
100712              -0.663302           -0.804025                  -
0.857394
153524              -1.242063           -1.270162                  -
1.103761
85660               -0.113480            0.003947                  -
0.326758
100506               0.436343            0.594387
0.393391

        distance  created_hour  is_weekend  order_protocol_4
94465   0.558203     -0.747442    1.376019         -0.330386
100712  0.118915     -0.632199    1.376019         -0.330386
153524 -0.919819     -0.401714   -0.726734         -0.330386
85660  -1.496384     -0.632199   -0.726734         -0.330386
100506 -1.079976      1.326924   -0.726734         -0.330386

# Create/Initialise the model
lr = LR()
lr.fit(X_train, y_train)


LinearRegression()

X_train.dtypes

total_items                 float64
subtotal                    float64
num_distinct_items          float64
max_item_price              float64
total_onshift_dashers       float64
total_busy_dashers          float64
total_outstanding_orders    float64
distance                    float64
created_hour                float64
is_weekend                  float64
```

```
order_protocol_4           float64
dtype: object
```

```
X_train.head()
```

```
        total_items  subtotal  num_distinct_items  max_item_price  \
94465     -0.075210  0.920885            0.198861        0.420814
100712     0.287056  0.040812            0.198861        0.338664
153524     1.373857 -0.564273            1.427533       -1.036469
85660     -0.075210 -0.665757            0.198861       -1.036469
100506    -0.799744 -0.852902           -1.029812       -0.775729

        total_onshift_dashers  total_busy_dashers
total_outstanding_orders  \
94465                1.130856            0.656539
0.810319
100712              -0.663302           -0.804025                 -
0.857394
153524              -1.242063           -1.270162                 -
1.103761
85660               -0.113480            0.003947                 -
0.326758
100506               0.436343            0.594387
0.393391

        distance  created_hour  is_weekend  order_protocol_4
94465   0.558203     -0.747442    1.376019         -0.330386
100712  0.118915     -0.632199    1.376019         -0.330386
153524 -0.919819     -0.401714   -0.726734         -0.330386
85660  -1.496384     -0.632199   -0.726734         -0.330386
100506 -1.079976      1.326924   -0.726734         -0.330386
```

```
X_train['is_weekend'] = X_train['is_weekend'].astype(int)
X_train['order_protocol_4'] = X_train['order_protocol_4'].astype(int)

# Train the model using the training data
X_train = sm.add_constant(X_train)
model =sm.OLS(y_train,X_train).fit()
model.summary()
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results

======================================================================
========
Dep. Variable:            mints_taken   R-squared:
0.862
Model:                            OLS   Adj. R-squared:
0.862
Method:                 Least Squares   F-statistic:
```

```
7.015e+04
Date:                  Sat, 28 Jun 2025   Prob (F-statistic):
0.00
Time:                          06:55:02   Log-Likelihood:            -
3.2718e+05
No. Observations:                123043   AIC:
6.544e+05
Df Residuals:                    123031   BIC:
6.545e+05
Df Model:                            11

Covariance Type:              nonrobust

======================================================================
======================
                              coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
----------------------
const                      45.8191      0.013   3620.129      0.000
45.794      45.844
total_items                -0.1142      0.015     -7.535      0.000
-0.144      -0.085
subtotal                    2.3057      0.018    125.678      0.000
2.270       2.342
num_distinct_items          0.8849      0.018     50.318      0.000
0.850       0.919
max_item_price              0.4399      0.014     32.569      0.000
0.413       0.466
total_onshift_dashers     -12.7285      0.034   -373.735      0.000
-12.795     -12.662
total_busy_dashers         -4.6398      0.033   -140.941      0.000
-4.704      -4.575
total_outstanding_orders    18.3581      0.031    591.864      0.000
18.297      18.419
distance                    4.1576      0.010    421.050      0.000
4.138       4.177
created_hour               -2.2426      0.011   -207.082      0.000
-2.264      -2.221
is_weekend                  1.2660      0.021     60.402      0.000
1.225       1.307
order_protocol_4           -0.1989      0.012    -17.250      0.000
-0.221      -0.176
======================================================================
========
Omnibus:                      36731.307   Durbin-Watson:
2.001
Prob(Omnibus):                    0.000   Jarque-Bera (JB):
144255.998
```

```
Skew:                          1.448   Prob(JB):
0.00
Kurtosis:                      7.444   Cond. No.
7.68
======================================================================
========

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

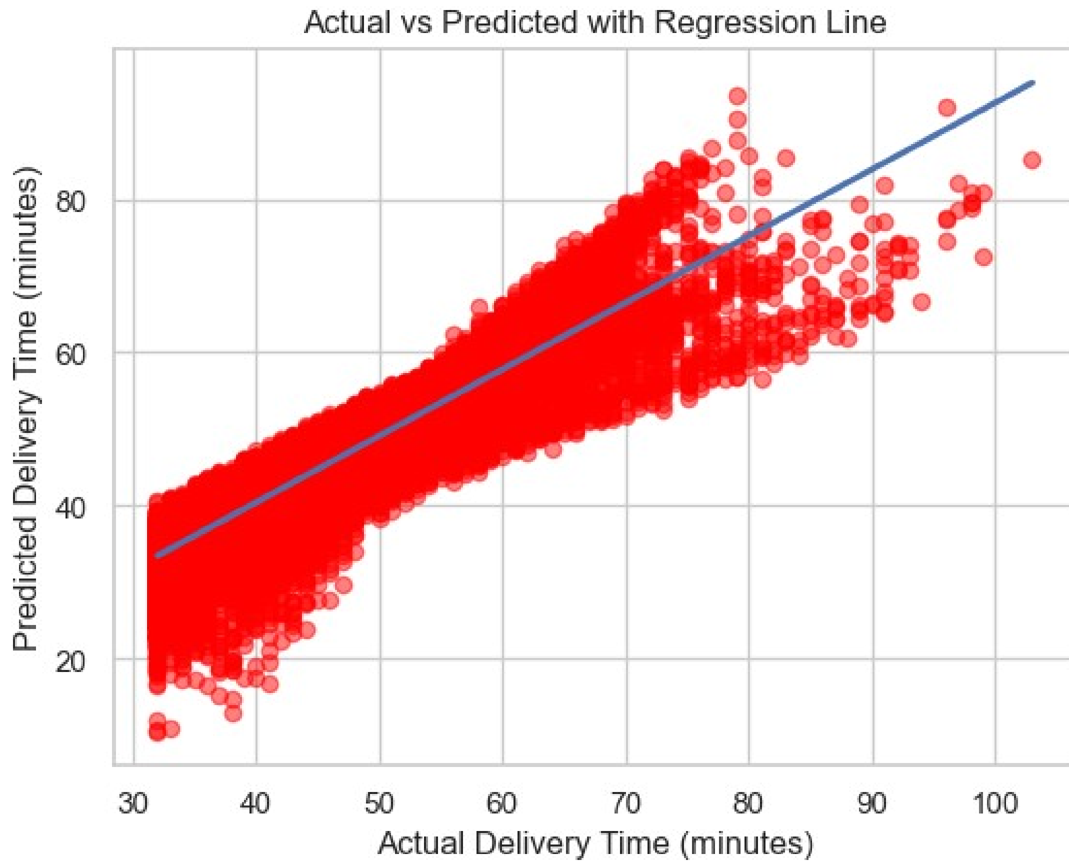
```
# Make predictions
X_test_const = sm.add_constant(X_test, has_constant='add')
y_test_pred =  model.predict(X_test_const)
# y_test_pred

# Find results for evaluation metrics

mse = mean_squared_error(y_test, y_test_pred)
rmse = mse ** 0.5
r2 = r2_score(y_test, y_test_pred)
print(f'The MSE is {mse}, \n RMSE is {rmse},\n R2_score is {r2}')
```

```
The MSE is 12.614605658256282,
 RMSE is 3.5517046130353074,
 R2_score is 0.8555764888025942
```

```
print(y_test.dtypes)
print(y_test_pred.dtypes)
```

```
float64
float64
```

```
y_test_pred = pd.Series(y_test_pred).astype('float64')

sns.regplot(x=y_test, y=y_test_pred, scatter_kws={"alpha":0.5,"color":
"red"})
plt.xlabel("Actual Delivery Time (minutes)")
plt.ylabel("Predicted Delivery Time (minutes)")
plt.title("Actual vs Predicted with Regression Line")
plt.grid(True)
plt.show()
```



Note that we have 12 (depending on how you select features) training features. However, not all of them would be useful. Let's say we want to take the most relevant 8 features.

We will use Recursive Feature Elimination (RFE) here.

For this, you can look at the coefficients / p-values of features from the model summary and perform feature elimination, or you can use the RFE module provided with *scikit-learn*.

5.3 Build the model and fit RFE to select the most important features [7 marks]

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.

```
# Loop through the number of features and test the model
res = []
for i in range(1, X_test.shape[1]+1):

    rfe = RFE(lr, n_features_to_select=i)
    rfe = rfe.fit(X_train,y_train)

    features = X_train.columns[rfe.support_]
```

```
    lr.fit(X_train[features], y_train)

    y_pred = lr.predict(X_train[features])

    mse = mean_squared_error(y_pred=y_pred, y_true=y_train)
    rmse = mse ** 0.5
    r2 = r2_score(y_pred=y_pred, y_true=y_train)

    res.append([i,mse, rmse, r2])

res_df = pd.DataFrame(res, columns=['features', 'MSE', 'RMSE', 'R2'])

print(res_df)
```

```
    features        MSE      RMSE        R2
0          1  73.950613  8.599454  0.148552
1          2  48.815698  6.986823  0.437949
2          3  46.852122  6.844861  0.460557
3          4  28.479874  5.336654  0.672090
4          5  16.880073  4.108537  0.805647
5          6  12.637922  3.554986  0.854490
6          7  12.263652  3.501950  0.858799
7          8  12.116064  3.480814  0.860499
8          9  11.982722  3.461607  0.862034
9         10  11.949522  3.456808  0.862416
10        11  11.944010  3.456011  0.862480
```

```
rfe = RFE(lr, n_features_to_select=10)
rfe = rfe.fit(X_train,y_train)
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

```
[('const', False, 3),
 ('total_items', False, 2),
 ('subtotal', True, 1),
 ('num_distinct_items', True, 1),
 ('max_item_price', True, 1),
 ('total_onshift_dashers', True, 1),
 ('total_busy_dashers', True, 1),
 ('total_outstanding_orders', True, 1),
 ('distance', True, 1),
 ('created_hour', True, 1),
 ('is_weekend', True, 1),
 ('order_protocol_4', True, 1)]
```

```
X_train.columns.to_list()
```

```
['const',
 'total_items',
 'subtotal',
 'num_distinct_items',
 'max_item_price',
```

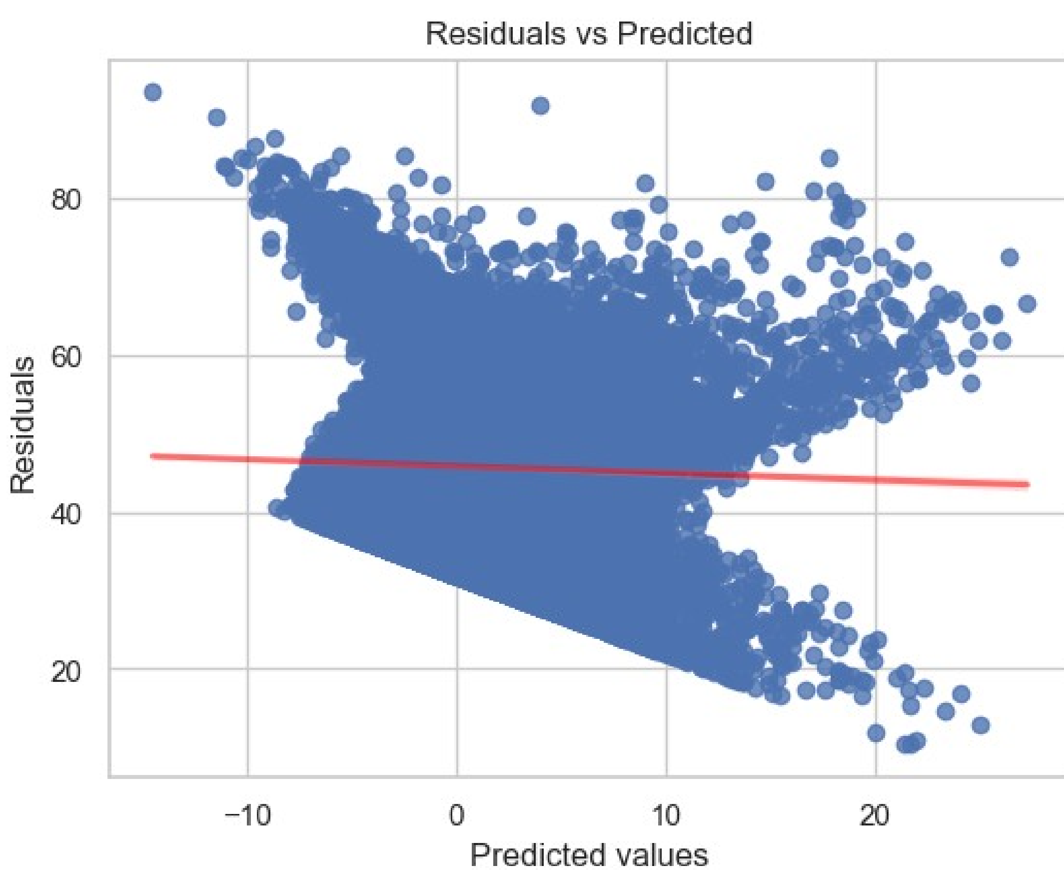
```
 'total_onshift_dashers',
 'total_busy_dashers',
 'total_outstanding_orders',
 'distance',
 'created_hour',
 'is_weekend',
 'order_protocol_4']

# Build the final model with selected number of features

X_train_new = X_train[['subtotal', 'num_distinct_items',
'max_item_price', 'total_onshift_dashers', 'total_busy_dashers',
'total_outstanding_orders', 'distance', 'created_hour', 'is_weekend',
'order_protocol_4']]
X_train_new = sm.add_constant(X_train_new)
model = sm.OLS(y_train,X_train_new).fit()
model.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results

======================================================================
========
Dep. Variable:            mints_taken   R-squared:
0.862
Model:                            OLS   Adj. R-squared:
0.862
Method:                 Least Squares   F-statistic:
7.712e+04
Date:                Sat, 28 Jun 2025   Prob (F-statistic):
0.00
Time:                        06:55:12   Log-Likelihood:            -
3.2721e+05
No. Observations:              123043   AIC:
6.544e+05
Df Residuals:                  123032   BIC:
6.545e+05
Df Model:                          10

Covariance Type:            nonrobust

======================================================================
======================
                               coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
----------------------
const                       45.8227      0.013   3622.129      0.000
45.798      45.847
```

```
subtotal                       2.2748      0.018    127.186      0.000
2.240       2.310
num_distinct_items             0.8207      0.015     53.329      0.000
0.791       0.851
max_item_price                 0.4621      0.013     35.061      0.000
0.436       0.488
total_onshift_dashers        -12.7269      0.034   -373.612      0.000
-12.794     -12.660
total_busy_dashers            -4.6410      0.033   -140.944      0.000
-4.705      -4.576
total_outstanding_orders      18.3578      0.031    591.721      0.000
18.297      18.419
distance                       4.1585      0.010    421.079      0.000
4.139       4.178
created_hour                  -2.2432      0.011   -207.098      0.000
-2.264      -2.222
is_weekend                     1.2661      0.021     60.393      0.000
1.225       1.307
order_protocol_4              -0.2111      0.011    -18.489      0.000
-0.233      -0.189
======================================================================
========
Omnibus:                    36731.693   Durbin-Watson:
2.001
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
144302.228
Skew:                           1.448   Prob(JB):
0.00
Kurtosis:                       7.445   Cond. No.
7.64
======================================================================
========

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

X_test_new = X_test[['subtotal', 'num_distinct_items',
'max_item_price', 'total_onshift_dashers', 'total_busy_dashers',
'total_outstanding_orders', 'distance', 'created_hour', 'is_weekend',
'order_protocol_4']]
X_test_const = sm.add_constant(X_test_new, has_constant='add')
y_test_pred =  model.predict(X_test_const)

residual = y_test - y_test_pred
```

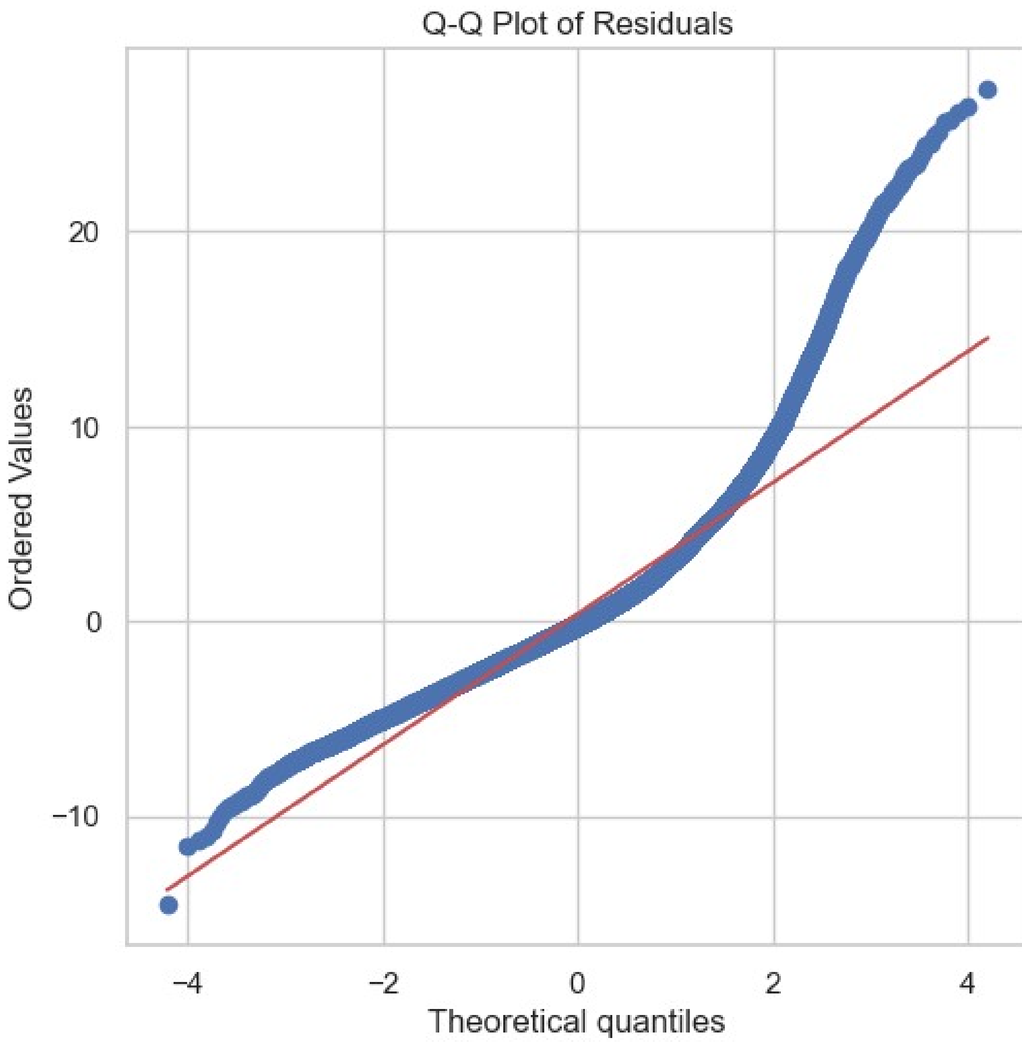
6. Results and Inference [5 marks]

6.1 Perform Residual Analysis [3 marks]

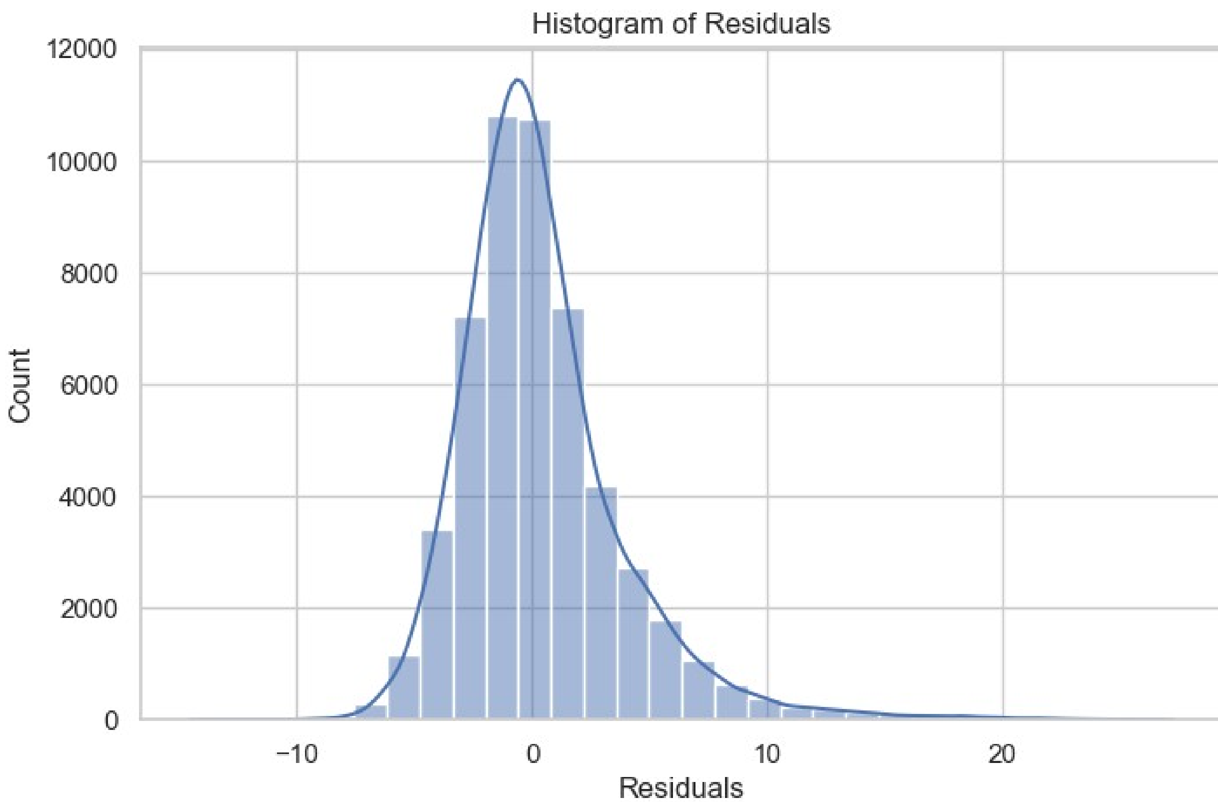
```
print(y_test.dtypes)
print(y_test_pred.dtypes)
print(residual.dtypes)

float64
float64
float64

y_test_pred = pd.Series(y_test_pred).astype('float64')
residual = pd.Series(residual).astype('float64')

# Perform residual analysis using plots like residuals vs predicted
values, Q-Q plot and residual histogram

sns.regplot(x=residual, y=y_test_pred, line_kws={"alpha":0.5,"color":
"red"})
plt.xlabel("Predicted values")
plt.ylabel("Residuals")
plt.title("Residuals vs Predicted")
plt.show()
```



```
plt.figure(figsize=(6, 6))
stats.probplot(residual, dist="norm", plot=plt)
plt.title("Q-Q Plot of Residuals")
plt.show()
```




```
plt.figure(figsize=(8, 5))
sns.histplot(residual, kde=True, bins=30)
plt.xlabel("Residuals")
plt.title("Histogram of Residuals")
plt.show()
```



[Your inferences here:]

6.2 Perform Coefficient Analysis [2 marks]

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.

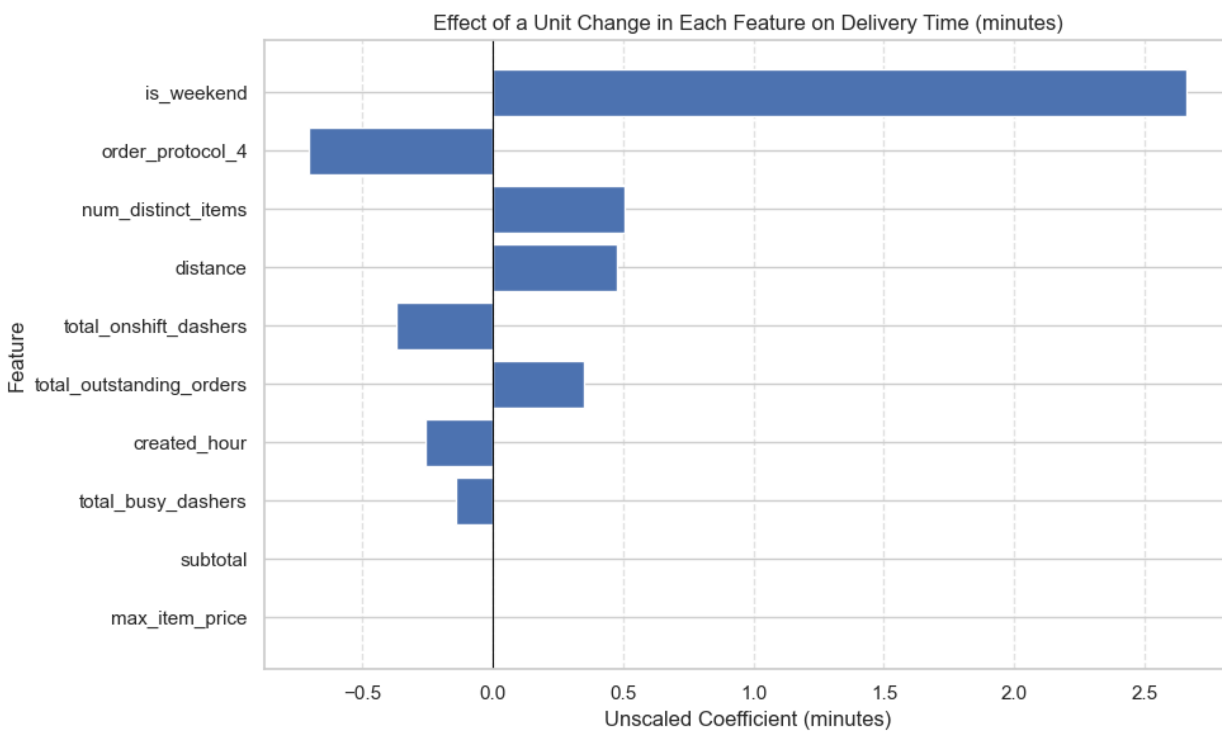
```
num_cols =
X_train_new.select_dtypes(include='number').columns.to_list()

# Compare the scaled vs unscaled features used in the final model

coef_scaled = model.params[num_cols].values
intercept_scaled = model.params[0]

# print(model.params[num_cols])
# print(list(zip(scaler.get_feature_names_out(), scaler.scale_)))

# To unscale: divide by standard scaler's scale
coef_unscaled = (coef_scaled[1:]) / (scaler.scale_[1:])
intercept_unscaled = intercept_scaled - np.sum(coef_scaled[1:] *
scaler.mean_[1:] / scaler.scale_[1:])

C:\Users\user\AppData\Local\Temp\ipykernel_17484\2993582975.py:4:
FutureWarning: Series.__getitem__ treating keys as positions is
```

```
deprecated. In a future version, integer keys will always be treated
as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  intercept_scaled = model.params[0]

coef_df = pd.DataFrame({
    "Feature": X_train_new.columns[1:],
    "Scaled Coef": coef_scaled[1:],
    "Unscaled Coef": coef_unscaled,
    "Mean": scaler.mean_[1:],
    "Std Dev": scaler.scale_[1:]
})

print(coef_df.round(4))

                    Feature  Scaled Coef  Unscaled Coef       Mean
Std Dev
0                  subtotal       2.2748         0.0012  2697.2000
1832.8014
1        num_distinct_items       0.8207         0.5042     2.6763
1.6278
2            max_item_price       0.4621         0.0008  1159.3666
559.9461
3     total_onshift_dashers     -12.7269        -0.3683    44.9215
34.5566
4        total_busy_dashers      -4.6410        -0.1442    41.8730
32.1794
5  total_outstanding_orders      18.3578         0.3479    58.2420
52.7669
6                  distance       4.1585         0.4757    21.8405
8.7414
7              created_hour      -2.2432        -0.2585     8.4858
8.6774
8                is_weekend       1.2661         2.6624     0.3456
0.4756
9          order_protocol_4      -0.2111        -0.7086     0.0984
0.2979
```

Additionally, we can analyse the effect of a unit change in a feature. In other words, because we have scaled the features, a unit change in the features will not translate directly to the model. Use scaled and unscaled coefficients to find how will a unit change in a feature affect the target.

```
# Analyze the effect of a unit change in a feature, say 'total_items'

coef_df["abs_coef"] = coef_df["Unscaled Coef"].abs()
df_sorted = coef_df.sort_values("abs_coef", ascending=True)

# Plot horizontal bar chart
plt.figure(figsize=(10, 6))
plt.barh(df_sorted["Feature"], df_sorted["Unscaled Coef"])
```

```
plt.axvline(0, color='black', linewidth=0.8)
plt.title("Effect of a Unit Change in Each Feature on Delivery Time (minutes)")
plt.xlabel("Unscaled Coefficient (minutes)")
plt.ylabel("Feature")
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



Note: The coefficients on the original scale might differ greatly in magnitude from the scaled coefficients, but they both describe the same relationships between variables.

Interpretation is key: Focus on the direction and magnitude of the coefficients on the original scale to understand the impact of each variable on the response variable in the original units.

Include conclusions in your report document.

Subjective Questions [20 marks]

Answer the following questions only in the notebook. Include the visualisations/methodologies/insights/outcomes from all the above steps in your report.

Subjective Questions based on Assignment

Question 1. [2 marks]

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

Answer:

Yes, the dataset contains several categorical variables.

- weekday or weekend: weekends have longer delivery than weekdays
- Market Id: some market location has higher average delivery time than others
- store_primary_category: had minimal effect on the delivery tiome

---

Question 2. [1 marks]

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

Answer:

having test_size 0.2 means splitting the data set into 80% and 20%; 80% of the goes to training and 20% of the data goes to testing

---

Question 3. [1 marks]

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

Answer:

distance has the highest correlation with delivery time (mints_taken) with value of 0.46

Question 4. [2 marks]

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

Answer:

use IQR method ont the outliers, i took 3 times of the values than normal becoz most of the potential outliers are not outliers

---

Question 5. [2 marks]

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

Answer:

is_weekend, order_protocol_4 and distance are the three top features affecting the delivery tome

---

General Subjective Questions

Question 6. [3 marks]

Explain the linear regression algorithm in detail

Answer:

- Linear regression is a supervised learning algorithm used to model the relationship between a dependent variable (y) and one or more independent variables (X).
- Assume the relation between input and output is linear, constant variance, normality of errors
- Each feature has coefficient which about the relation with the dependent variable.

Question 7. [2 marks]

Explain the difference between simple linear regression and multiple linear regression

Answer:

- simple linear regression has one independent variable, depends on single factor
- multiple linear regression has more than one independent varibale, depends on multiple factor

---

Question 8. [2 marks]

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

Answer:

- also known as loss function, quantifies the difference between predicted values and actual values.
- lower the value the better the model.
- The cost function guides the learning process to adjust coefficients to minimize prediction errors.

Question 9. [2 marks]

Explain the difference between overfitting and underfitting.

Answer:

- under fit: model is too simple to capture the data; high error on training and test set.
- over fit: model is too complex, fits noise in the data, low training error and high testing error

---

Question 10. [3 marks]

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

Answer:

- residual plot tells us about the difference in actual value and predicted value in graphical
- to check assumption: linear relation, constant variance, independence, Normality of errors.
- Random scatter (no patter) is a good fit model.