Arghya_Porter_NN_regression

May 5, 2024

1 Defining problem statement,and data structure analysis

1.1 Problem definition

Train a NN based regression model that will do the delivery time estimation for Porter based on the remaining independent feature variables.

1.2 Import data with data structure analysis

[6]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
import statistics
from scipy.stats import poisson, binom
from scipy.stats import levene
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

[7]:
```
df = pd.read_csv("dataset.csv")
data = pd.read_csv("dataset.csv")
```

[8]:
```
df.head()
```

[8]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_category	order_protocol
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	american	1.0
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	mexican	2.0
2	3.0	2015-01-22 20:39:28	2015-01-22 21:09:09	f0ade77b43923b38237db569b016ba25	NaN	1.0
3	3.0	2015-02-03 21:21:45	2015-02-03 22:13:00	f0ade77b43923b38237db569b016ba25	NaN	1.0
4	3.0	2015-02-15 02:40:36	2015-02-15 03:20:26	f0ade77b43923b38237db569b016ba25	NaN	1.0

```
   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            1      1900                   1            1900            1900
3            6      6900                   5             600            1800
4            3      3900                   3            1100            1600

   total_onshift_partners  total_busy_partners  total_outstanding_orders
0                    33.0                 14.0                      21.0
1                     1.0                  2.0                       2.0
2                     1.0                  0.0                       0.0
3                     1.0                  1.0                       2.0
4                     6.0                  6.0                       9.0
```

[9]:
```
df.shape
```

[9]:
```
(197428, 14)
```

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 197428 entries, 0 to 197427
Data columns (total 14 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   market_id                196441 non-null  float64
 1   created_at               197428 non-null  object
 2   actual_delivery_time     197421 non-null  object
 3   store_id                 197428 non-null  object
 4   store_primary_category   192668 non-null  object
 5   order_protocol           196433 non-null  float64
 6   total_items              197428 non-null  int64
 7   subtotal                 197428 non-null  int64
 8   num_distinct_items       197428 non-null  int64
 9   min_item_price           197428 non-null  int64
 10  max_item_price           197428 non-null  int64
 11  total_onshift_partners   181166 non-null  float64
 12  total_busy_partners      181166 non-null  float64
 13  total_outstanding_orders 181166 non-null  float64
dtypes: float64(5), int64(5), object(4)
memory usage: 21.1+ MB
```

[11]:
```
df.isna().sum()
```

[11]:
```
market_id                   987
created_at                    0
actual_delivery_time          7
store_id                      0
```

```
store_primary_category       4760
order_protocol                995
total_items                     0
subtotal                        0
num_distinct_items              0
min_item_price                  0
max_item_price                  0
total_onshift_partners      16262
total_busy_partners         16262
total_outstanding_orders    16262
dtype: int64
```

[12]:
```
df.describe(include=['object'])
```

[12]:
```
                created_at actual_delivery_time  \
count               197428               197421
unique              180985               178110
top     2015-02-11 19:50:43  2015-02-11 20:40:45
freq                     6                    5

                                store_id store_primary_category
count                             197428                192668
unique                              6743                    74
top     d43ab110ab2489d6b9b2caa394bf920f              american
freq                                 937                 19399
```

[13]:
```
df.describe()
```

[13]:
```
           market_id  order_protocol    total_items       subtotal  \
count  196441.000000   196433.000000  197428.000000  197428.000000
mean        2.978706        2.882352       3.196391    2682.331402
std         1.524867        1.503771       2.666546    1823.093688
min         1.000000        1.000000       1.000000       0.000000
25%         2.000000        1.000000       2.000000    1400.000000
50%         3.000000        3.000000       3.000000    2200.000000
75%         4.000000        4.000000       4.000000    3395.000000
max         6.000000        7.000000     411.000000   27100.000000

       num_distinct_items  min_item_price  max_item_price  \
count       197428.000000   197428.000000   197428.000000
mean             2.670791      686.218470     1159.588630
std              1.630255      522.038648      558.411377
min              1.000000      -86.000000        0.000000
25%              1.000000      299.000000      800.000000
50%              2.000000      595.000000     1095.000000
75%              3.000000      949.000000     1395.000000
max             20.000000    14700.000000    14700.000000
```

	total_onshift_partners	total_busy_partners	total_outstanding_orders
count	181166.000000	181166.000000	181166.000000
mean	44.808093	41.739747	58.050065
std	34.526783	32.145733	52.661830
min	-4.000000	-5.000000	-6.000000
25%	17.000000	15.000000	17.000000
50%	37.000000	34.000000	41.000000
75%	65.000000	62.000000	85.000000
max	171.000000	154.000000	285.000000

2 Data preprocessing and feature engineering

2.1 Data cleaning

[14]:
```
df.head(20)
```

[14]:

	market_id	created_at	actual_delivery_time	\
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	
2	3.0	2015-01-22 20:39:28	2015-01-22 21:09:09	
3	3.0	2015-02-03 21:21:45	2015-02-03 22:13:00	
4	3.0	2015-02-15 02:40:36	2015-02-15 03:20:26	
5	3.0	2015-01-28 20:30:38	2015-01-28 21:08:58	
6	3.0	2015-01-31 02:16:36	2015-01-31 02:43:00	
7	3.0	2015-02-12 03:03:35	2015-02-12 03:36:20	
8	2.0	2015-02-16 00:11:35	2015-02-16 00:38:01	
9	3.0	2015-02-18 01:15:45	2015-02-18 02:08:57	
10	3.0	2015-02-02 19:22:53	2015-02-02 20:09:19	
11	3.0	2015-02-16 04:19:33	2015-02-16 06:34:00	
12	3.0	2015-02-07 01:34:31	2015-02-07 02:17:14	
13	3.0	2015-01-25 01:50:51	2015-01-25 02:28:53	
14	1.0	2015-02-12 03:36:46	2015-02-12 04:14:39	
15	1.0	2015-01-27 02:12:36	2015-01-27 03:02:24	
16	1.0	2015-02-06 00:42:42	2015-02-06 02:10:29	
17	1.0	2015-02-08 02:04:17	2015-02-08 03:27:13	
18	1.0	2015-01-31 04:35:54	2015-01-31 05:47:30	
19	1.0	2015-01-31 02:21:23	2015-01-31 03:11:42	

	store_id	store_primary_category	order_protocol	\
0	df263d996281d984952c07998dc54358	american	1.0	
1	f0ade77b43923b38237db569b016ba25	mexican	2.0	
2	f0ade77b43923b38237db569b016ba25	NaN	1.0	
3	f0ade77b43923b38237db569b016ba25	NaN	1.0	
4	f0ade77b43923b38237db569b016ba25	NaN	1.0	
5	f0ade77b43923b38237db569b016ba25	NaN	1.0	
6	f0ade77b43923b38237db569b016ba25	NaN	1.0	

7	f0ade77b43923b38237db569b016ba25	NaN	1.0
8	f0ade77b43923b38237db569b016ba25	indian	3.0
9	f0ade77b43923b38237db569b016ba25	NaN	1.0
10	f0ade77b43923b38237db569b016ba25	NaN	4.0
11	f0ade77b43923b38237db569b016ba25	NaN	1.0
12	f0ade77b43923b38237db569b016ba25	NaN	1.0
13	f0ade77b43923b38237db569b016ba25	NaN	4.0
14	ef1e491a766ce3127556063d49bc2f98	italian	1.0
15	ef1e491a766ce3127556063d49bc2f98	italian	1.0
16	ef1e491a766ce3127556063d49bc2f98	italian	1.0
17	ef1e491a766ce3127556063d49bc2f98	italian	1.0
18	ef1e491a766ce3127556063d49bc2f98	italian	1.0
19	ce016f59ecc2366a43e1c96a4774d167	mexican	1.0

	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	1	1900	1	1900	1900	
3	6	6900	5	600	1800	
4	3	3900	3	1100	1600	
5	3	5000	3	1500	1900	
6	2	3900	2	1200	2700	
7	4	4850	4	750	1800	
8	4	4771	3	820	1604	
9	2	2100	2	700	1200	
10	4	4300	4	1200	1500	
11	2	2200	2	600	1600	
12	1	1900	1	1900	1900	
13	4	4986	4	699	2362	
14	1	1525	1	1525	1525	
15	2	3620	2	1425	2195	
16	3	4475	3	925	1825	
17	3	4375	3	1325	1625	
18	2	3150	2	1425	1725	
19	2	950	2	150	700	

	total_onshift_partners	total_busy_partners	total_outstanding_orders
0	33.0	14.0	21.0
1	1.0	2.0	2.0
2	1.0	0.0	0.0
3	1.0	1.0	2.0
4	6.0	6.0	9.0
5	2.0	2.0	2.0
6	10.0	9.0	9.0
7	7.0	8.0	7.0
8	8.0	6.0	18.0
9	2.0	2.0	2.0

10	1.0	1.0	1.0
11	3.0	3.0	4.0
12	6.0	3.0	3.0
13	16.0	6.0	9.0
14	5.0	6.0	8.0
15	5.0	5.0	7.0
16	4.0	1.0	1.0
17	6.0	4.0	3.0
18	4.0	9.0	12.0
19	24.0	24.0	26.0

[15]:
```
df['created_at'] = pd.to_datetime(df['created_at'], infer_datetime_format=True)
df['actual_delivery_time'] = pd.to_datetime(df['actual_delivery_time'],␣
 ↪infer_datetime_format=True)
```

```
C:\Users\arghy\AppData\Local\Temp\ipykernel_14080\755350726.py:1: UserWarning:
The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.
  df['created_at'] = pd.to_datetime(df['created_at'],
infer_datetime_format=True)
C:\Users\arghy\AppData\Local\Temp\ipykernel_14080\755350726.py:2: UserWarning:
The argument 'infer_datetime_format' is deprecated and will be removed in a
future version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-parsing.html. You
can safely remove this argument.
  df['actual_delivery_time'] = pd.to_datetime(df['actual_delivery_time'],
infer_datetime_format=True)
```

[16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 197428 entries, 0 to 197427
Data columns (total 14 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   market_id                196441 non-null  float64
 1   created_at               197428 non-null  datetime64[ns]
 2   actual_delivery_time     197421 non-null  datetime64[ns]
 3   store_id                 197428 non-null  object
 4   store_primary_category   192668 non-null  object
 5   order_protocol           196433 non-null  float64
 6   total_items              197428 non-null  int64
 7   subtotal                 197428 non-null  int64
 8   num_distinct_items       197428 non-null  int64
 9   min_item_price           197428 non-null  int64
 10  max_item_price           197428 non-null  int64
```

```
 11  total_onshift_partners    181166 non-null  float64
 12  total_busy_partners       181166 non-null  float64
 13  total_outstanding_orders  181166 non-null  float64
dtypes: datetime64[ns](2), float64(5), int64(5), object(2)
memory usage: 21.1+ MB
```

2.2 Null value handling

[17]:
```
df.dropna(subset = ['actual_delivery_time'], inplace=True)
df.isna().sum()
```

[17]:
```
market_id                   987
created_at                    0
actual_delivery_time          0
store_id                      0
store_primary_category     4760
order_protocol              995
total_items                   0
subtotal                      0
num_distinct_items            0
min_item_price                0
max_item_price                0
total_onshift_partners    16262
total_busy_partners       16262
total_outstanding_orders  16262
dtype: int64
```

[18]:
```
df.dropna(inplace=True)
```

[19]:
```
df.isna().sum()
```

[19]:
```
market_id                 0
created_at                0
actual_delivery_time      0
store_id                  0
store_primary_category    0
order_protocol            0
total_items               0
subtotal                  0
num_distinct_items        0
min_item_price            0
max_item_price            0
total_onshift_partners    0
total_busy_partners       0
total_outstanding_orders  0
dtype: int64
```

[20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 176248 entries, 0 to 197427
Data columns (total 14 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   market_id                 176248 non-null  float64
 1   created_at                176248 non-null  datetime64[ns]
 2   actual_delivery_time      176248 non-null  datetime64[ns]
 3   store_id                  176248 non-null  object
 4   store_primary_category    176248 non-null  object
 5   order_protocol            176248 non-null  float64
 6   total_items               176248 non-null  int64
 7   subtotal                  176248 non-null  int64
 8   num_distinct_items        176248 non-null  int64
 9   min_item_price            176248 non-null  int64
 10  max_item_price            176248 non-null  int64
 11  total_onshift_partners    176248 non-null  float64
 12  total_busy_partners       176248 non-null  float64
 13  total_outstanding_orders  176248 non-null  float64
dtypes: datetime64[ns](2), float64(5), int64(5), object(2)
memory usage: 20.2+ MB
```

[ ]:

2.3 Creating the target column (time taken for delivery) from order timestamp and delivery timestamp

[21]:
```
df["time_to_deliver"] = df["actual_delivery_time"] - df["created_at"]
```

[22]:
```
df.head()
```

[22]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	american	1.0	4	3441	4	557	1239
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	mexican	2.0					
8	2.0	2015-02-16 00:11:35	2015-02-16 00:38:01	f0ade77b43923b38237db569b016ba25	indian	3.0					
14	1.0	2015-02-12 03:36:46	2015-02-12 04:14:39	ef1e491a766ce3127556063d49bc2f98	italian	1.0					
15	1.0	2015-01-27 02:12:36	2015-01-27 03:02:24	ef1e491a766ce3127556063d49bc2f98	italian	1.0					

```
1             1      1900                  1            1400            1400
8             4      4771                  3             820            1604
14            1      1525                  1            1525            1525
15            2      3620                  2            1425            2195

    total_onshift_partners  total_busy_partners  total_outstanding_orders  \
0                     33.0                 14.0                      21.0
1                      1.0                  2.0                       2.0
8                      8.0                  6.0                      18.0
14                     5.0                  6.0                       8.0
15                     5.0                  5.0                       7.0

   time_to_deliver
0  0 days 01:02:59
1  0 days 01:07:04
8  0 days 00:26:26
14 0 days 00:37:53
15 0 days 00:49:48
```

[23]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 176248 entries, 0 to 197427
Data columns (total 15 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   market_id                176248 non-null  float64
 1   created_at               176248 non-null  datetime64[ns]
 2   actual_delivery_time     176248 non-null  datetime64[ns]
 3   store_id                 176248 non-null  object
 4   store_primary_category   176248 non-null  object
 5   order_protocol           176248 non-null  float64
 6   total_items              176248 non-null  int64
 7   subtotal                 176248 non-null  int64
 8   num_distinct_items       176248 non-null  int64
 9   min_item_price           176248 non-null  int64
 10  max_item_price           176248 non-null  int64
 11  total_onshift_partners   176248 non-null  float64
 12  total_busy_partners      176248 non-null  float64
 13  total_outstanding_orders 176248 non-null  float64
 14  time_to_deliver          176248 non-null  timedelta64[ns]
dtypes: datetime64[ns](2), float64(5), int64(5), object(2), timedelta64[ns](1)
memory usage: 21.5+ MB
```

[24]:
```
#convert time_to_deliver column to integer

df['time_to_deliver'] = df['time_to_deliver'] / pd.Timedelta(minutes=1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 176248 entries, 0 to 197427
Data columns (total 15 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   market_id                 176248 non-null  float64
 1   created_at                176248 non-null  datetime64[ns]
 2   actual_delivery_time      176248 non-null  datetime64[ns]
 3   store_id                  176248 non-null  object
 4   store_primary_category    176248 non-null  object
 5   order_protocol            176248 non-null  float64
 6   total_items               176248 non-null  int64
 7   subtotal                  176248 non-null  int64
 8   num_distinct_items        176248 non-null  int64
 9   min_item_price            176248 non-null  int64
 10  max_item_price            176248 non-null  int64
 11  total_onshift_partners    176248 non-null  float64
 12  total_busy_partners       176248 non-null  float64
 13  total_outstanding_orders  176248 non-null  float64
 14  time_to_deliver           176248 non-null  float64
dtypes: datetime64[ns](2), float64(6), int64(5), object(2)
memory usage: 21.5+ MB
```

2.4 Getting hour and day of the week

[25]:
```
df['created_hour'] = df["created_at"].dt.hour
df['created_day'] = df["created_at"].dt.dayofweek
df.head()
```

[25]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	american	1.0	4	3441	4	557	1239
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	mexican	2.0	1	1900	1	1400	1400
8	2.0	2015-02-16 00:11:35	2015-02-16 00:38:01	f0ade77b43923b38237db569b016ba25	indian	3.0	4	4771	3	820	1604
14	1.0	2015-02-12 03:36:46	2015-02-12 04:14:39	ef1e491a766ce3127556063d49bc2f98	italian	1.0					
15	1.0	2015-01-27 02:12:36	2015-01-27 03:02:24	ef1e491a766ce3127556063d49bc2f98	italian	1.0					

```
14              1      1525                    1            1525             1525
15              2      3620                    2            1425             2195

    total_onshift_partners  total_busy_partners  total_outstanding_orders  \
0                     33.0                14.0                      21.0
1                      1.0                 2.0                       2.0
8                      8.0                 6.0                      18.0
14                     5.0                 6.0                       8.0
15                     5.0                 5.0                       7.0

    time_to_deliver  created_hour  created_day
0         62.983333            22            4
1         67.066667            21            1
8         26.433333             0            0
14        37.883333             3            3
15        49.800000             2            1
```

[26]:
```
! pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl.metadata (8.0 kB)
Requirement already satisfied: numpy>=1.14.0 in
d:\anaconda_home\envs\tf\lib\site-packages (from category_encoders) (1.26.4)
Requirement already satisfied: scikit-learn>=0.20.0 in
d:\anaconda_home\envs\tf\lib\site-packages (from category_encoders) (1.4.2)
Requirement already satisfied: scipy>=1.0.0 in
d:\anaconda_home\envs\tf\lib\site-packages (from category_encoders) (1.13.0)
Collecting statsmodels>=0.9.0 (from category_encoders)
  Downloading statsmodels-0.14.2-cp310-cp310-win_amd64.whl.metadata (9.5 kB)
Requirement already satisfied: pandas>=1.0.5 in
d:\anaconda_home\envs\tf\lib\site-packages (from category_encoders) (2.2.2)
Collecting patsy>=0.5.1 (from category_encoders)
  Downloading patsy-0.5.6-py2.py3-none-any.whl.metadata (3.5 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in
d:\anaconda_home\envs\tf\lib\site-packages (from
pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
d:\anaconda_home\envs\tf\lib\site-packages (from
pandas>=1.0.5->category_encoders) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
d:\anaconda_home\envs\tf\lib\site-packages (from
pandas>=1.0.5->category_encoders) (2024.1)
Requirement already satisfied: six in d:\anaconda_home\envs\tf\lib\site-packages
(from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.2.0 in
d:\anaconda_home\envs\tf\lib\site-packages (from scikit-
learn>=0.20.0->category_encoders) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
```

```
d:\anaconda_home\envs\tf\lib\site-packages (from scikit-
learn>=0.20.0->category_encoders) (3.5.0)
Requirement already satisfied: packaging>=21.3 in
d:\anaconda_home\envs\tf\lib\site-packages (from
statsmodels>=0.9.0->category_encoders) (23.2)
Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
   ---------------------------------------- 0.0/81.9 kB ? eta -:--:--
   -------------- ------------------------ 30.7/81.9 kB 1.4 MB/s eta 0:00:01
   ------------------- -------------------- 41.0/81.9 kB 495.5 kB/s eta 0:00:01
   ---------------------------------------- 81.9/81.9 kB 761.9 kB/s eta 0:00:00
Downloading patsy-0.5.6-py2.py3-none-any.whl (233 kB)
   ---------------------------------------- 0.0/233.9 kB ? eta -:--:--
   ------- -------------------------------- 41.0/233.9 kB 2.0 MB/s eta 0:00:01
   -------------------- ------------------- 122.9/233.9 kB 1.4 MB/s eta 0:00:01
   -------------------- ------------------- 122.9/233.9 kB 1.4 MB/s eta 0:00:01
   ---------------------------- ---------- 174.1/233.9 kB 1.1 MB/s eta 0:00:01
   ---------------------------------------- 233.9/233.9 kB 1.0 MB/s eta 0:00:00
Downloading statsmodels-0.14.2-cp310-cp310-win_amd64.whl (9.8 MB)
   ---------------------------------------- 0.0/9.8 MB ? eta -:--:--
   ---------------------------------------- 0.1/9.8 MB 1.7 MB/s eta 0:00:06
   ---------------------------------------- 0.1/9.8 MB 1.8 MB/s eta 0:00:06
    --------------------------------------- 0.2/9.8 MB 2.0 MB/s eta 0:00:05
   - -------------------------------------- 0.3/9.8 MB 2.0 MB/s eta 0:00:05
   - -------------------------------------- 0.4/9.8 MB 2.1 MB/s eta 0:00:05
   - -------------------------------------- 0.5/9.8 MB 2.0 MB/s eta 0:00:05
   -- ------------------------------------- 0.6/9.8 MB 2.3 MB/s eta 0:00:04
   -- ------------------------------------- 0.7/9.8 MB 2.3 MB/s eta 0:00:05
   --- ------------------------------------ 0.8/9.8 MB 2.3 MB/s eta 0:00:04
   --- ------------------------------------ 0.9/9.8 MB 2.3 MB/s eta 0:00:04
   ---- ----------------------------------- 1.0/9.8 MB 2.4 MB/s eta 0:00:04
   ---- ----------------------------------- 1.1/9.8 MB 2.4 MB/s eta 0:00:04
   ----- ---------------------------------- 1.2/9.8 MB 2.4 MB/s eta 0:00:04
   ----- ---------------------------------- 1.4/9.8 MB 2.5 MB/s eta 0:00:04
   ------ --------------------------------- 1.5/9.8 MB 2.5 MB/s eta 0:00:04
   ------ --------------------------------- 1.7/9.8 MB 2.6 MB/s eta 0:00:04
   ------- -------------------------------- 1.8/9.8 MB 2.6 MB/s eta 0:00:04
   ------- -------------------------------- 2.0/9.8 MB 2.6 MB/s eta 0:00:03
   -------- ------------------------------- 2.0/9.8 MB 2.6 MB/s eta 0:00:03
   -------- ------------------------------- 2.2/9.8 MB 2.6 MB/s eta 0:00:03
   --------- ------------------------------ 2.3/9.8 MB 2.6 MB/s eta 0:00:03
   --------- ------------------------------ 2.3/9.8 MB 2.6 MB/s eta 0:00:03
   --------- ------------------------------ 2.3/9.8 MB 2.5 MB/s eta 0:00:04
   --------- ------------------------------ 2.4/9.8 MB 2.4 MB/s eta 0:00:04
   ---------- ----------------------------- 2.5/9.8 MB 2.5 MB/s eta 0:00:03
   ---------- ----------------------------- 2.6/9.8 MB 2.4 MB/s eta 0:00:04
   ----------- ---------------------------- 2.7/9.8 MB 2.5 MB/s eta 0:00:03
   ----------- ---------------------------- 2.9/9.8 MB 2.5 MB/s eta 0:00:03
   ------------ --------------------------- 3.1/9.8 MB 2.6 MB/s eta 0:00:03
```

```
----------- -------------------------- 3.2/9.8 MB 2.6 MB/s eta 0:00:03
------------ -------------------------- 3.3/9.8 MB 2.6 MB/s eta 0:00:03
------------- ------------------------- 3.5/9.8 MB 2.6 MB/s eta 0:00:03
------------- ------------------------- 3.6/9.8 MB 2.7 MB/s eta 0:00:03
-------------- ------------------------ 3.7/9.8 MB 2.7 MB/s eta 0:00:03
-------------- ------------------------ 3.8/9.8 MB 2.6 MB/s eta 0:00:03
-------------- ------------------------ 3.9/9.8 MB 2.6 MB/s eta 0:00:03
--------------- ----------------------- 4.0/9.8 MB 2.6 MB/s eta 0:00:03
--------------- ----------------------- 4.1/9.8 MB 2.6 MB/s eta 0:00:03
---------------- ---------------------- 4.2/9.8 MB 2.7 MB/s eta 0:00:03
---------------- ---------------------- 4.4/9.8 MB 2.7 MB/s eta 0:00:03
----------------- --------------------- 4.6/9.8 MB 2.7 MB/s eta 0:00:02
------------------ -------------------- 4.8/9.8 MB 2.8 MB/s eta 0:00:02
------------------- ------------------- 5.0/9.8 MB 2.8 MB/s eta 0:00:02
------------------- ------------------- 5.1/9.8 MB 2.8 MB/s eta 0:00:02
-------------------- ------------------ 5.3/9.8 MB 2.8 MB/s eta 0:00:02
--------------------- ----------------- 5.5/9.8 MB 2.9 MB/s eta 0:00:02
--------------------- ----------------- 5.7/9.8 MB 2.9 MB/s eta 0:00:02
---------------------- ---------------- 5.9/9.8 MB 2.9 MB/s eta 0:00:02
---------------------- ---------------- 6.0/9.8 MB 2.9 MB/s eta 0:00:02
----------------------- --------------- 6.1/9.8 MB 2.9 MB/s eta 0:00:02
------------------------ -------------- 6.3/9.8 MB 3.0 MB/s eta 0:00:02
------------------------ -------------- 6.3/9.8 MB 3.0 MB/s eta 0:00:02
------------------------- ------------- 6.4/9.8 MB 2.9 MB/s eta 0:00:02
------------------------- ------------- 6.5/9.8 MB 2.9 MB/s eta 0:00:02
------------------------- ------------- 6.5/9.8 MB 2.8 MB/s eta 0:00:02
------------------------- ------------- 6.5/9.8 MB 2.8 MB/s eta 0:00:02
------------------------- ------------- 6.6/9.8 MB 2.8 MB/s eta 0:00:02
-------------------------- ------------ 6.7/9.8 MB 2.8 MB/s eta 0:00:02
-------------------------- ------------ 6.7/9.8 MB 2.7 MB/s eta 0:00:02
--------------------------- ----------- 6.9/9.8 MB 2.8 MB/s eta 0:00:02
---------------------------- ---------- 7.2/9.8 MB 2.8 MB/s eta 0:00:01
---------------------------- ---------- 7.3/9.8 MB 2.8 MB/s eta 0:00:01
----------------------------- --------- 7.4/9.8 MB 2.8 MB/s eta 0:00:01
----------------------------- --------- 7.5/9.8 MB 2.8 MB/s eta 0:00:01
----------------------------- --------- 7.6/9.8 MB 2.8 MB/s eta 0:00:01
------------------------------ -------- 7.7/9.8 MB 2.8 MB/s eta 0:00:01
------------------------------ -------- 7.8/9.8 MB 2.7 MB/s eta 0:00:01
------------------------------ -------- 7.9/9.8 MB 2.7 MB/s eta 0:00:01
------------------------------- ------- 7.9/9.8 MB 2.7 MB/s eta 0:00:01
------------------------------- ------- 8.0/9.8 MB 2.7 MB/s eta 0:00:01
------------------------------- ------- 8.1/9.8 MB 2.7 MB/s eta 0:00:01
-------------------------------- ------ 8.2/9.8 MB 2.7 MB/s eta 0:00:01
-------------------------------- ------ 8.3/9.8 MB 2.7 MB/s eta 0:00:01
--------------------------------- ----- 8.4/9.8 MB 2.7 MB/s eta 0:00:01
--------------------------------- ----- 8.6/9.8 MB 2.7 MB/s eta 0:00:01
---------------------------------- ---- 8.7/9.8 MB 2.7 MB/s eta 0:00:01
---------------------------------- --- 8.9/9.8 MB 2.7 MB/s eta 0:00:01
```

```
   ---------------------------------- --- 9.1/9.8 MB 2.8 MB/s eta 0:00:01
   ---------------------------------- -- 9.2/9.8 MB 2.8 MB/s eta 0:00:01
   ---------------------------------- - 9.4/9.8 MB 2.8 MB/s eta 0:00:01
   ---------------------------------- - 9.5/9.8 MB 2.7 MB/s eta 0:00:01
   ---------------------------------- - 9.5/9.8 MB 2.8 MB/s eta 0:00:01
   ---------------------------------- - 9.6/9.8 MB 2.7 MB/s eta 0:00:01
   ---------------------------------- - 9.6/9.8 MB 2.7 MB/s eta 0:00:01
   ---------------------------------------  9.7/9.8 MB 2.7 MB/s eta 0:00:01
   ---------------------------------------  9.8/9.8 MB 2.7 MB/s eta 0:00:01
   ---------------------------------------- 9.8/9.8 MB 2.7 MB/s eta 0:00:00
Installing collected packages: patsy, statsmodels, category_encoders
Successfully installed category_encoders-2.6.3 patsy-0.5.6 statsmodels-0.14.2
```

2.5 Encoding categorical column

[27]:
```
import category_encoders as ce

encoder=ce.TargetEncoder()
df['store_id'] = encoder.fit_transform(df['store_id'],df['time_to_deliver'])

df.head()
```

[27]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_partners	total_busy_partners	total_outstanding_orders
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	49.744347	american	1.0	4	3441	4	557	1239	33.0	14.0	21.0
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	47.620343	mexican	2.0	1	1900	1	1400	1400	1.0	2.0	2.0
8	2.0	2015-02-16 00:11:35	2015-02-16 00:38:01	47.620343	indian	3.0	4	4771	3	820	1604	8.0	6.0	18.0
14	1.0	2015-02-12 03:36:46	2015-02-12 04:14:39	51.090884	italian	1.0	1	1525	1	1525	1525			
15	1.0	2015-01-27 02:12:36	2015-01-27 03:02:24	51.090884	italian	1.0	2	3620	2	1425	2195			

```
14                    5.0                  6.0                        8.0
15                    5.0                  5.0                        7.0

    time_to_deliver  created_hour  created_day
0         62.983333            22            4
1         67.066667            21            1
8         26.433333             0            0
14        37.883333             3            3
15        49.800000             2            1
```

[28]:
```
df['store_primary_category'] = encoder.
  ↪fit_transform(df['store_primary_category'],df['time_to_deliver'])

df.head()
```

[28]:
```
    market_id          created_at actual_delivery_time   store_id  \
0         1.0 2015-02-06 22:24:17  2015-02-06 23:27:16  49.744347
1         2.0 2015-02-10 21:49:25  2015-02-10 22:56:29  47.620343
8         2.0 2015-02-16 00:11:35  2015-02-16 00:38:01  47.620343
14        1.0 2015-02-12 03:36:46  2015-02-12 04:14:39  51.090884
15        1.0 2015-01-27 02:12:36  2015-01-27 03:02:24  51.090884

    store_primary_category  order_protocol  total_items  subtotal  \
0                47.875225             1.0            4      3441
1                44.329778             2.0            1      1900
8                50.408414             3.0            4      4771
14               50.287894             1.0            1      1525
15               50.287894             1.0            2      3620

    num_distinct_items  min_item_price  max_item_price  \
0                    4             557            1239
1                    1            1400            1400
8                    3             820            1604
14                   1            1525            1525
15                   2            1425            2195

    total_onshift_partners  total_busy_partners  total_outstanding_orders  \
0                     33.0                 14.0                      21.0
1                      1.0                  2.0                       2.0
8                      8.0                  6.0                      18.0
14                     5.0                  6.0                       8.0
15                     5.0                  5.0                       7.0

    time_to_deliver  created_hour  created_day
0         62.983333            22            4
1         67.066667            21            1
8         26.433333             0            0
```

```
14        37.883333              3             3
15        49.800000              2             1
```

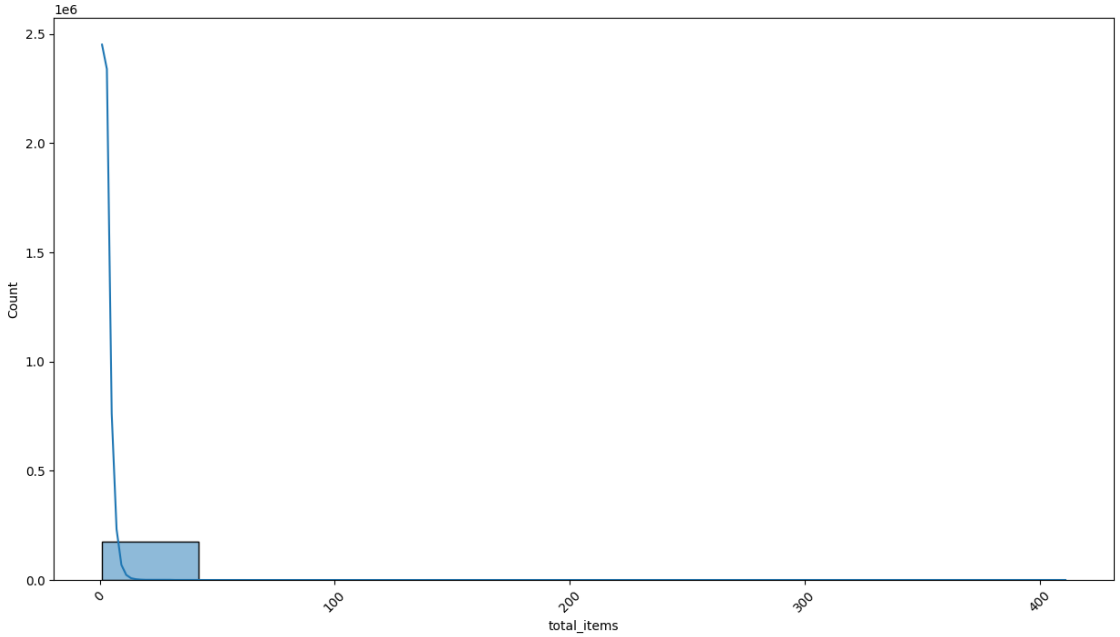
[29]:
```
df2 = df.drop(columns=['created_at', 'actual_delivery_time'])

df2.head()
```

[29]:
```
    market_id   store_id  store_primary_category  order_protocol  total_items  \
0         1.0  49.744347                47.875225             1.0            4
1         2.0  47.620343                44.329778             2.0            1
8         2.0  47.620343                50.408414             3.0            4
14        1.0  51.090884                50.287894             1.0            1
15        1.0  51.090884                50.287894             1.0            2

    subtotal  num_distinct_items  min_item_price  max_item_price  \
0       3441                   4             557            1239
1       1900                   1            1400            1400
8       4771                   3             820            1604
14      1525                   1            1525            1525
15      3620                   2            1425            2195

    total_onshift_partners  total_busy_partners  total_outstanding_orders  \
0                     33.0                 14.0                      21.0
1                      1.0                  2.0                       2.0
8                      8.0                  6.0                      18.0
14                     5.0                  6.0                       8.0
15                     5.0                  5.0                       7.0

    time_to_deliver  created_hour  created_day
0         62.983333            22            4
1         67.066667            21            1
8         26.433333             0            0
14        37.883333             3            3
15        49.800000             2            1
```

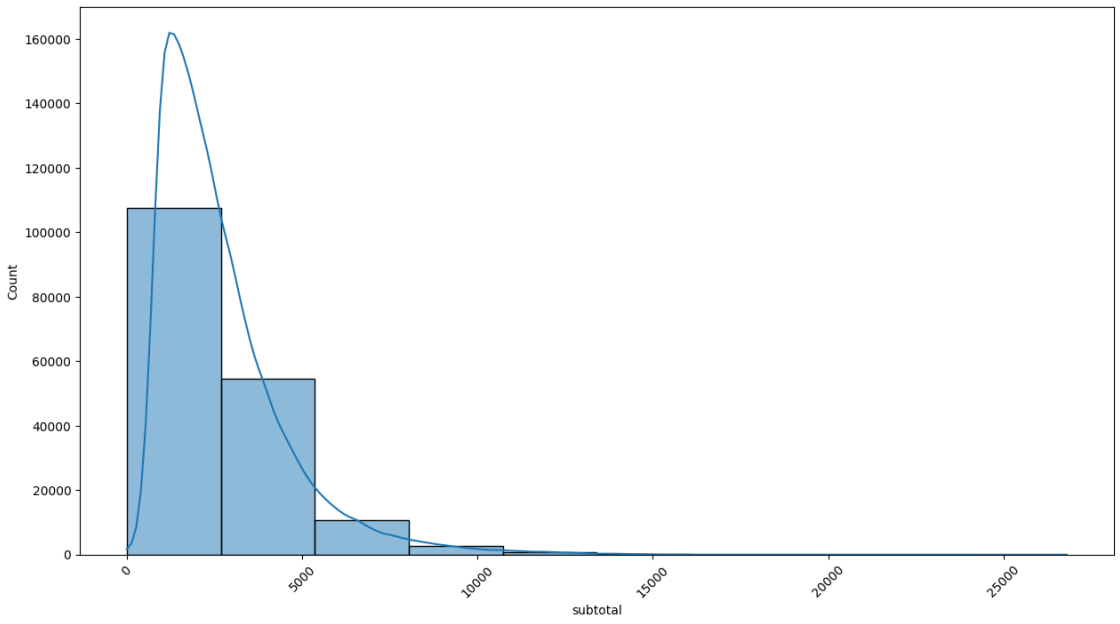
3 Data visualization and cleaning

3.1 Visualization for various features

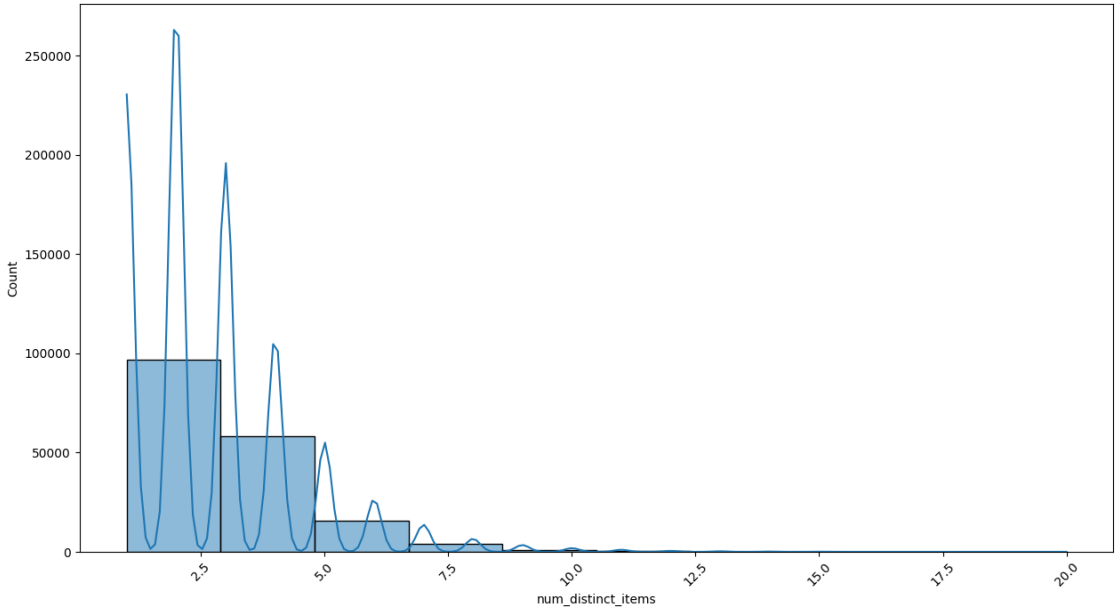
[30]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
sns.histplot(x="total_items", bins=10, data = df2, kde=True)
plt.show()
```



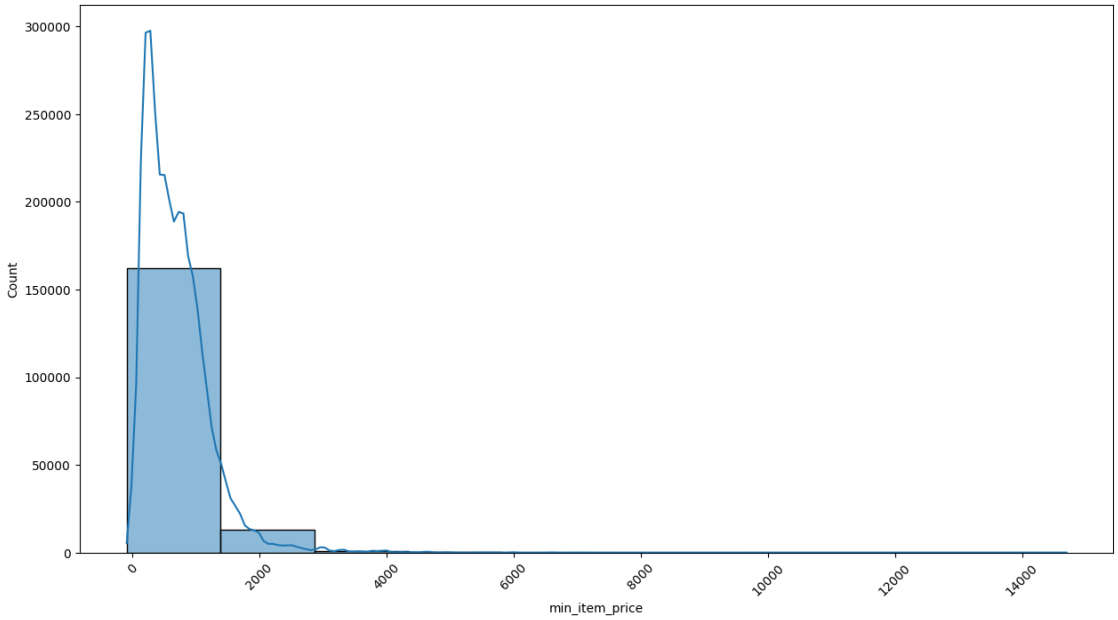
[31]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
sns.histplot(x="subtotal", bins=10, data = df2, kde=True)
plt.show()
```



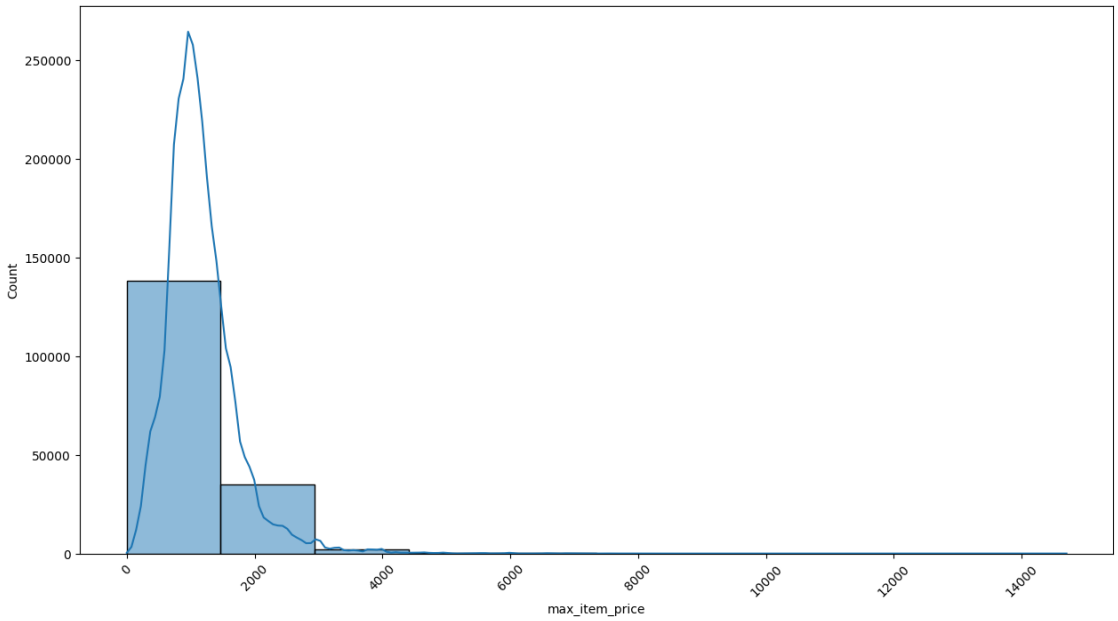
[32]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
sns.histplot(x="num_distinct_items", bins=10, data = df2, kde=True)
plt.show()
```



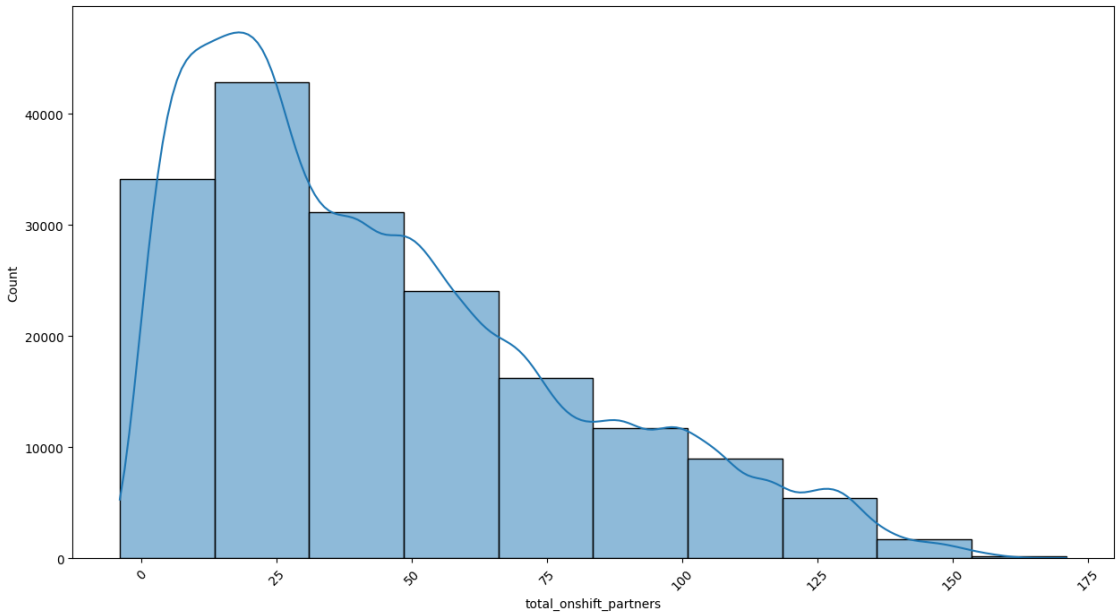
[33]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
sns.histplot(x="min_item_price", bins=10, data = df2, kde=True)
plt.show()
```



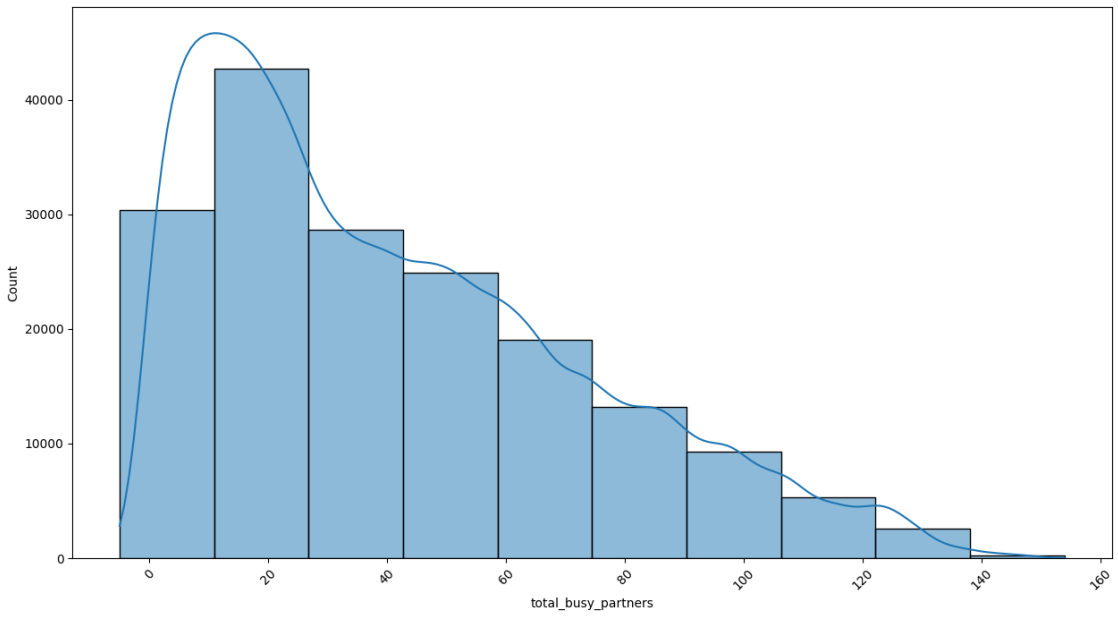
[34]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
sns.histplot(x="max_item_price", bins=10, data = df2, kde=True)
plt.show()
```



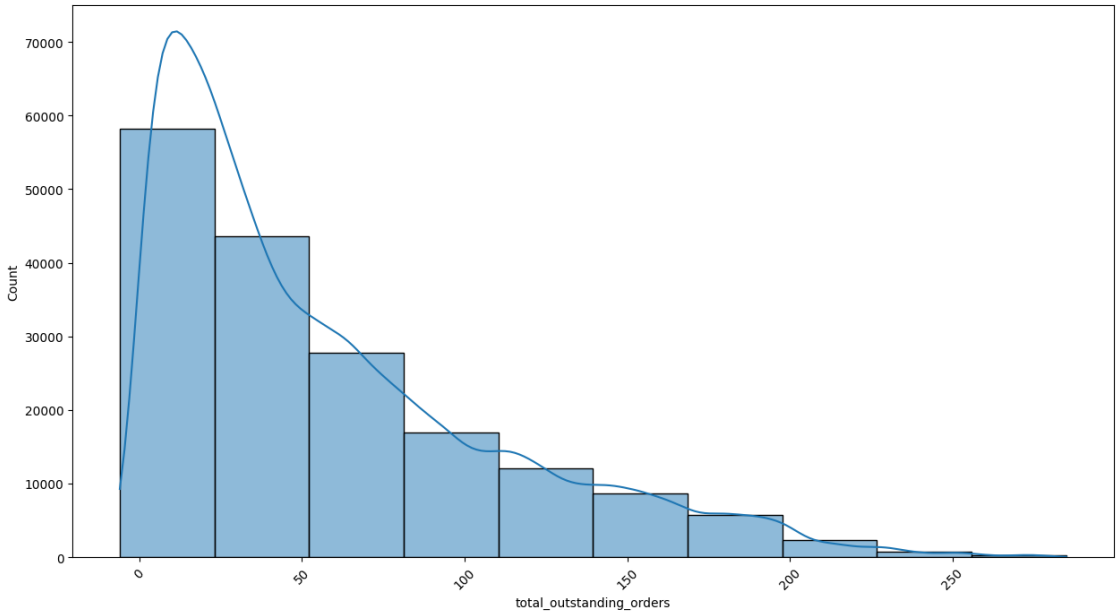
[35]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
sns.histplot(x="total_onshift_partners", bins=10, data = df2, kde=True)
plt.show()
```



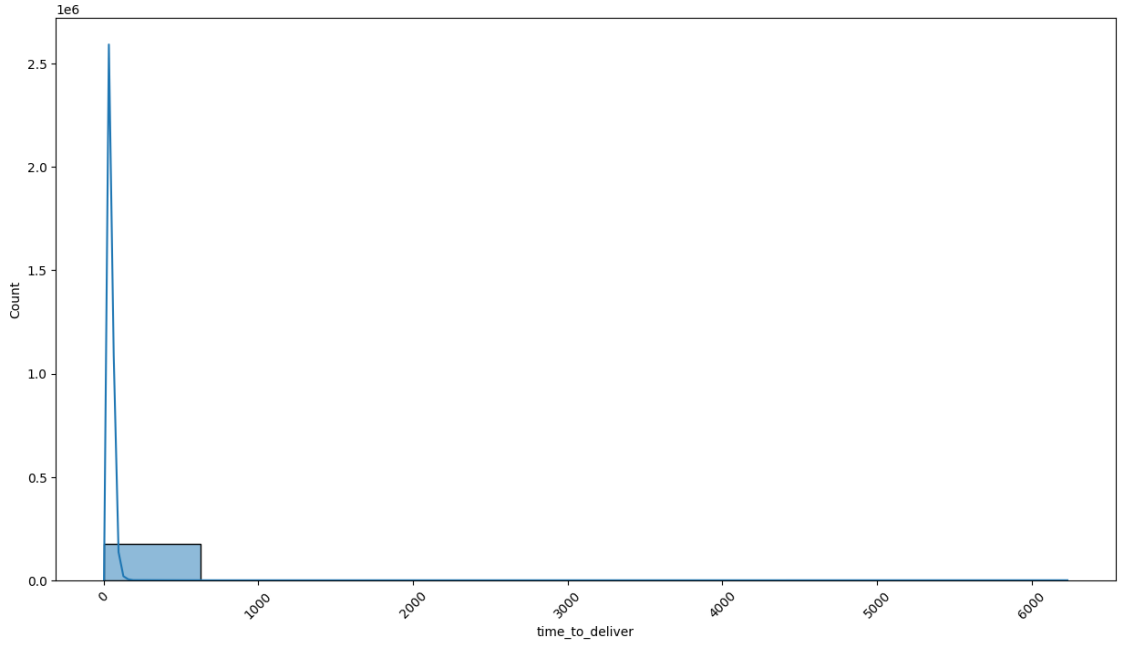
[36]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
sns.histplot(x="total_busy_partners", bins=10, data = df2, kde=True)
plt.show()
```



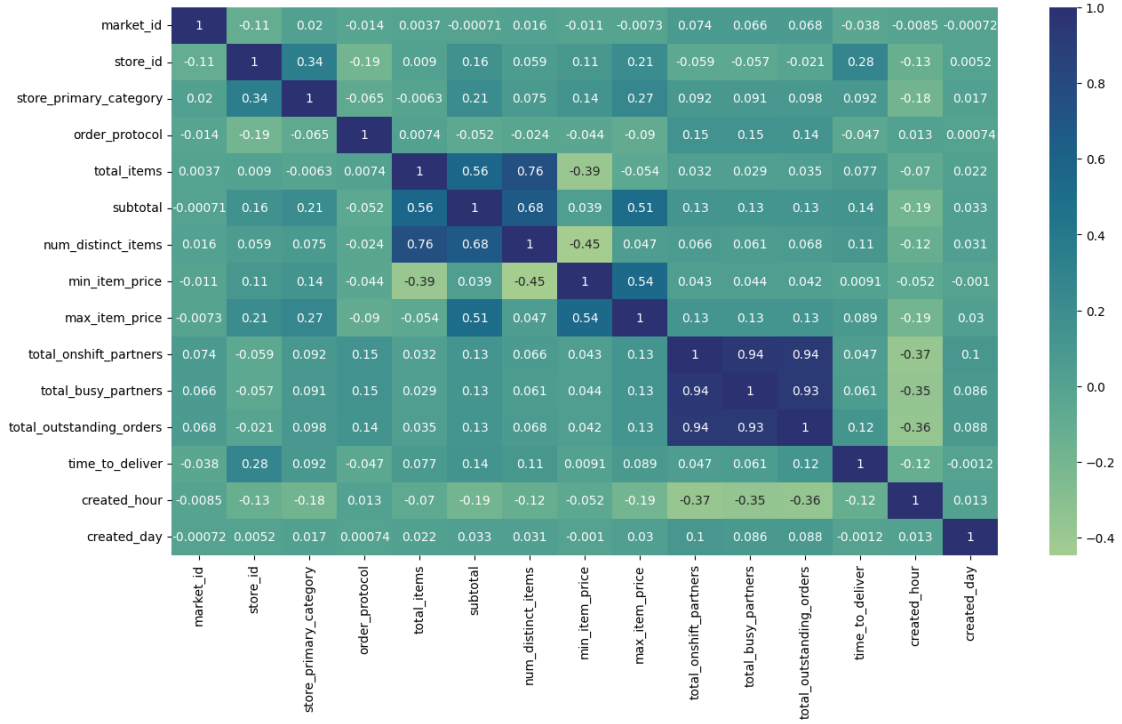
[37]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
sns.histplot(x="total_outstanding_orders", bins=10, data = df2, kde=True)
plt.show()
```



[38]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
sns.histplot(x="time_to_deliver", bins=10, data = df2, kde=True)
plt.show()
```



[39]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
sns.heatmap(df2.corr(),annot=True, cmap="crest")
plt.show()
```



[40]:
```
df2.head()
```

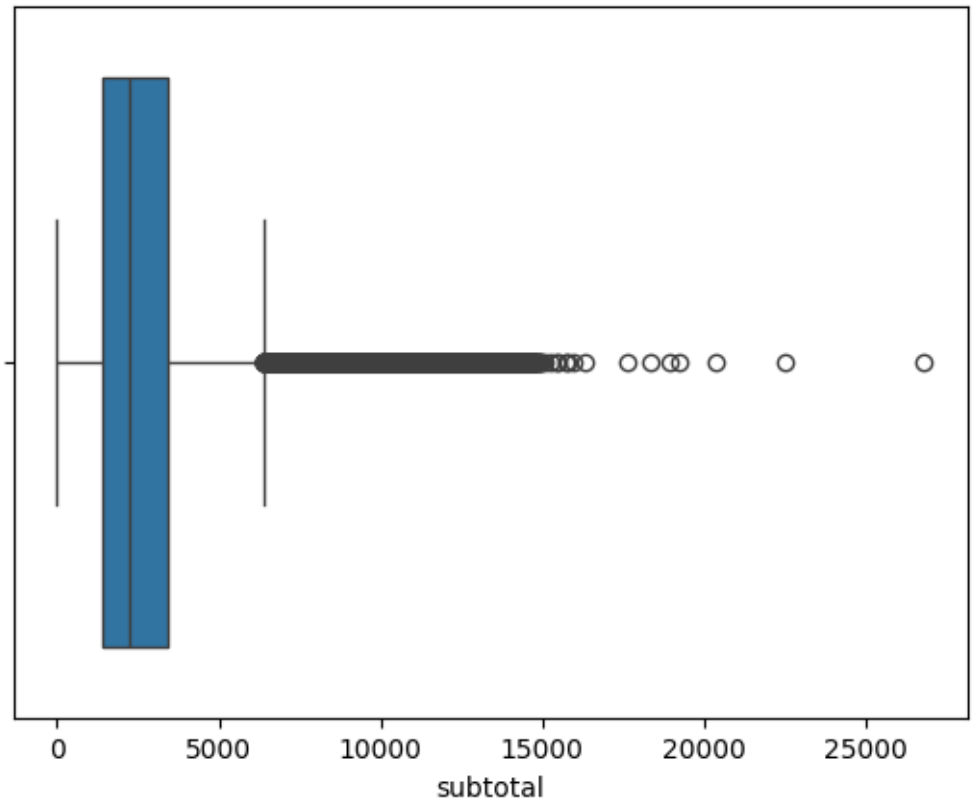
[40]:

	market_id	store_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_partners	total_busy_partners	total_outstanding_orders	time_to_deliver	created_hour	created_day
0	1.0	49.744347	47.875225	1.0	4	3441	4	557	1239	33.0	14.0	21.0	62.983333	22	4
1	2.0	47.620343	44.329778	2.0	1	1900	1	1400	1400	1.0	2.0	2.0	67.066667	21	1
8	2.0	47.620343	50.408414	3.0	4	4771	3	820	1604	8.0	6.0	18.0	26.433333	0	0
14	1.0	51.090884	50.287894	1.0	1	1525	1	1525	1525	5.0	6.0	8.0	37.883333	3	3
15	1.0	51.090884	50.287894	1.0	2	3620	2	1425	2195	5.0	5.0	7.0	49.800000	2	1

3.2 Check for outliers

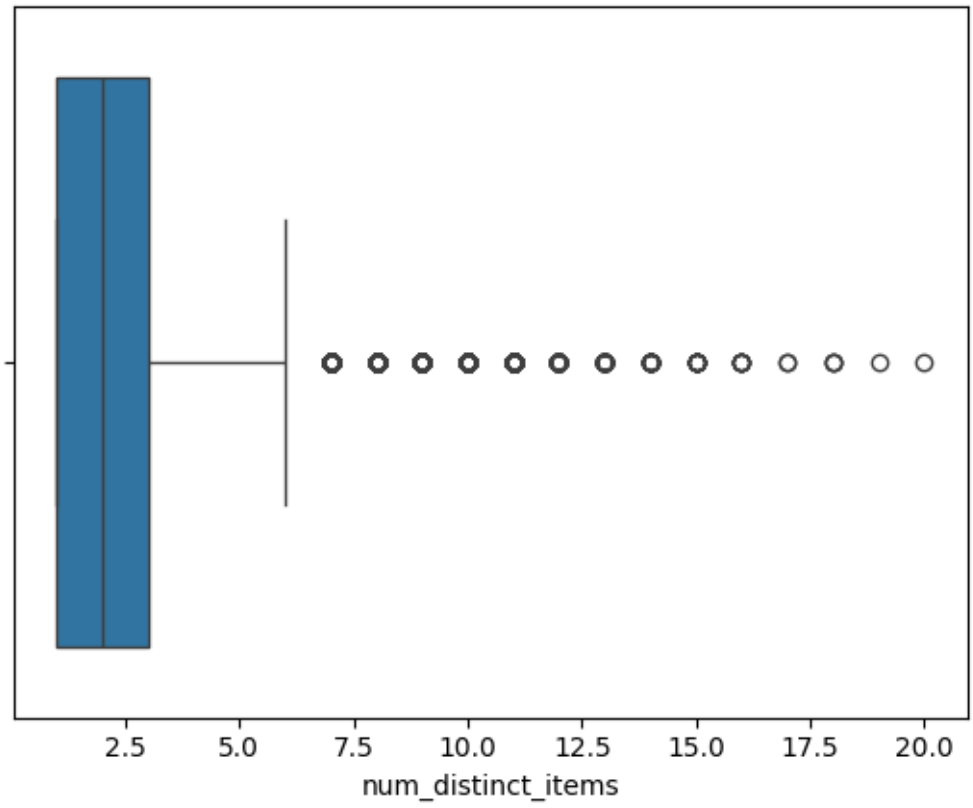
[41]:
```
sns.boxplot(x=df2['subtotal'])
```

[41]:
```
<Axes: xlabel='subtotal'>
```



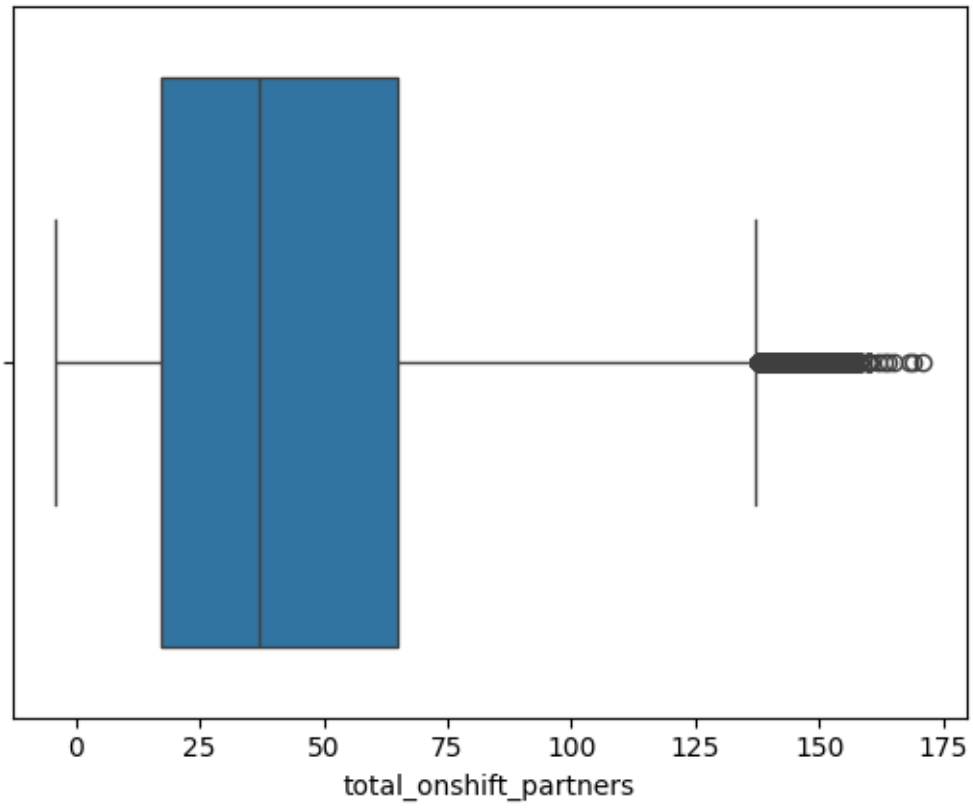
[42]:
```
sns.boxplot(x=df2['num_distinct_items'])
```

[42]:
```
<Axes: xlabel='num_distinct_items'>
```



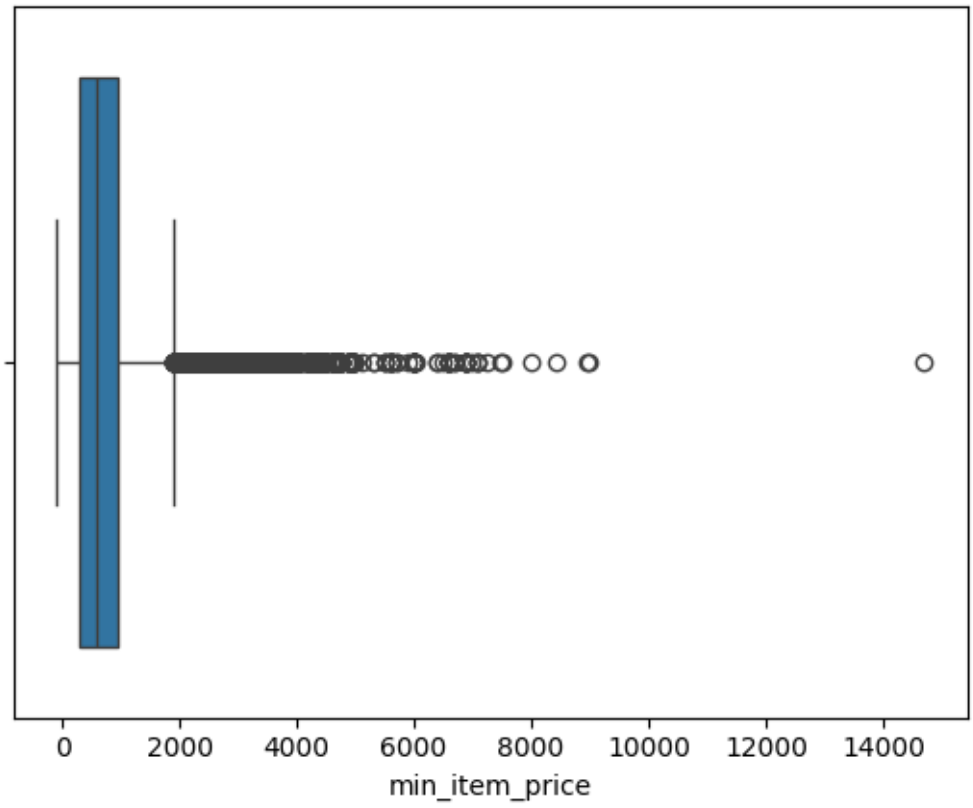
[43]:
```
sns.boxplot(x=df2['total_onshift_partners'])
```

[43]:
```
<Axes: xlabel='total_onshift_partners'>
```



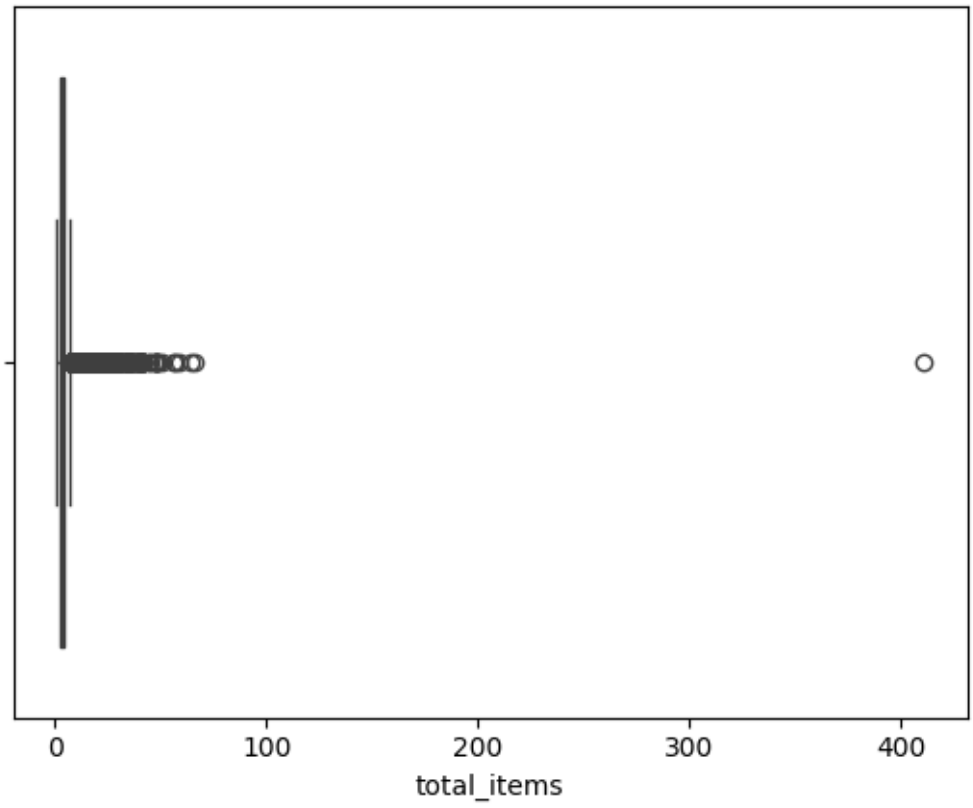
[44]:
```
sns.boxplot(x=df2['min_item_price'])
```

[44]:
```
<Axes: xlabel='min_item_price'>
```



[45]:
```
sns.boxplot(x=df2['total_items'])
```

[45]:
```
<Axes: xlabel='total_items'>
```



3.3 Remove outliers

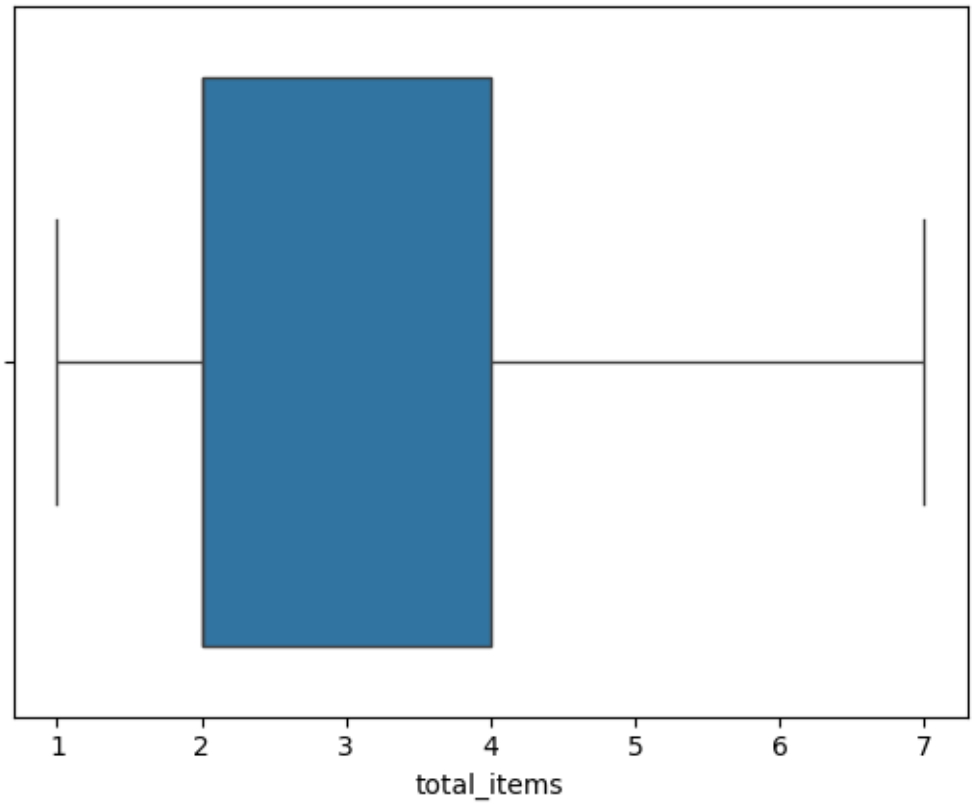
[46]:
```
Q1 = np.percentile(df2["total_items"], 25, interpolation = 'midpoint')
Q2 = np.percentile(df2["total_items"], 50, interpolation = 'midpoint')
Q3 = np.percentile(df2["total_items"], 75, interpolation = 'midpoint')

print('Q1 25 percentile of the given data is, ', Q1)
print('Q1 50 percentile of the given data is, ', Q2)
print('Q1 75 percentile of the given data is, ', Q3)

IQR = Q3 - Q1
print('Interquartile range is', IQR)
upperWhisker = Q3 + 1.5*IQR
print('upper limit',upperWhisker)
```

```
Q1 25 percentile of the given data is,  2.0
Q1 50 percentile of the given data is,  3.0
Q1 75 percentile of the given data is,  4.0
Interquartile range is 2.0
upper limit 7.0
```

[47]:
```
df2.drop(df2[df2["total_items"]>upperWhisker].index , inplace=True)
```

[48]:
```
sns.boxplot(x=df2['total_items'])
```

[48]:
```
<Axes: xlabel='total_items'>
```



[49]:
```
Q1 = np.percentile(df2["num_distinct_items"], 25, interpolation = 'midpoint')
Q2 = np.percentile(df2["num_distinct_items"], 50, interpolation = 'midpoint')
Q3 = np.percentile(df2["num_distinct_items"], 75, interpolation = 'midpoint')

print('Q1 25 percentile of the given data is, ', Q1)
print('Q1 50 percentile of the given data is, ', Q2)
print('Q1 75 percentile of the given data is, ', Q3)

IQR = Q3 - Q1
print('Interquartile range is', IQR)
upperWhisker = Q3 + 1.5*IQR
print('upper limit',upperWhisker)
```

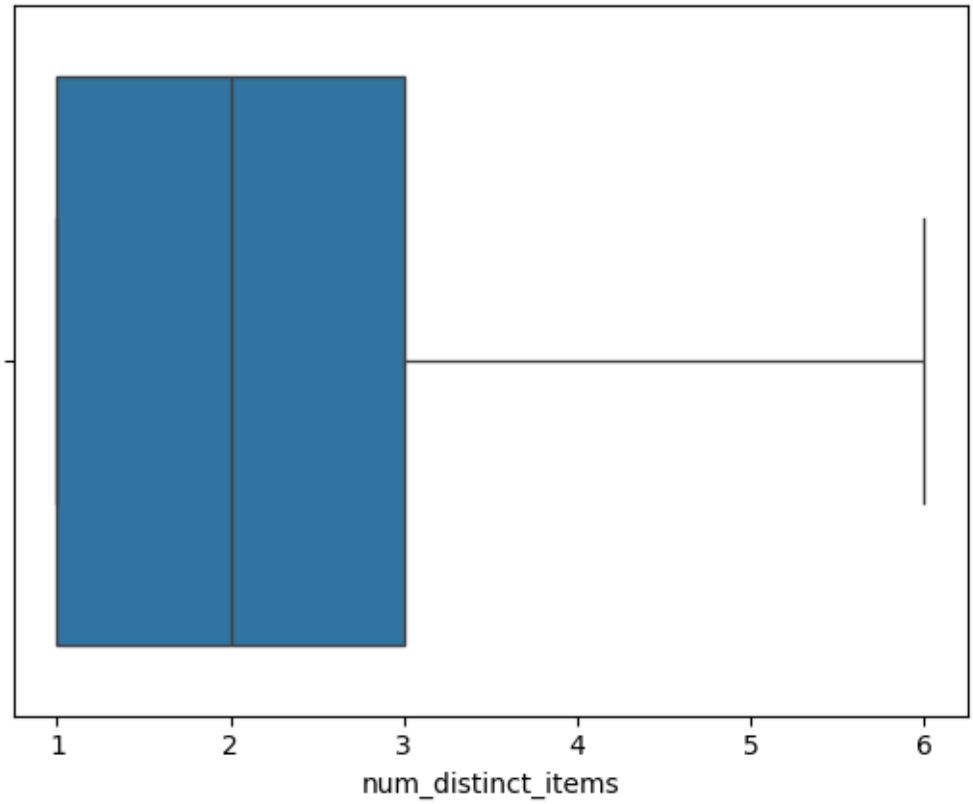
```
Q1 25 percentile of the given data is,  1.0
Q1 50 percentile of the given data is,  2.0
Q1 75 percentile of the given data is,  3.0
```

```
Interquartile range is 2.0
upper limit 6.0
```

[50]:
```
df2.drop(df2[df2["num_distinct_items"]>upperWhisker].index , inplace=True)
```

[51]:
```
sns.boxplot(x=df2['num_distinct_items'])
```

[51]:
```
<Axes: xlabel='num_distinct_items'>
```



[52]:
```
Q1 = np.percentile(df2["total_onshift_partners"], 25, interpolation =␣
 ↪'midpoint')
Q2 = np.percentile(df2["total_onshift_partners"], 50, interpolation =␣
 ↪'midpoint')
Q3 = np.percentile(df2["total_onshift_partners"], 75, interpolation =␣
 ↪'midpoint')

print('Q1 25 percentile of the given data is, ', Q1)
print('Q1 50 percentile of the given data is, ', Q2)
print('Q1 75 percentile of the given data is, ', Q3)

IQR = Q3 - Q1
```

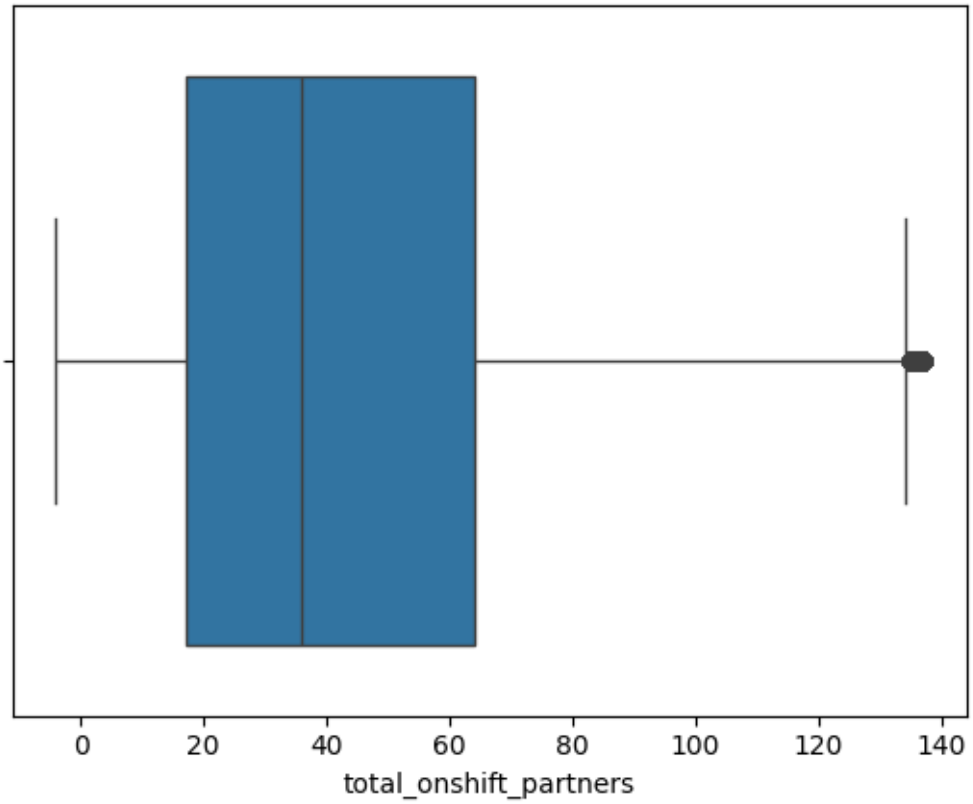
```
print('Interquartile range is', IQR)
upperWhisker = Q3 + 1.5*IQR
print('upper limit',upperWhisker)
```

```
Q1 25 percentile of the given data is,  17.0
Q1 50 percentile of the given data is,  37.0
Q1 75 percentile of the given data is,  65.0
Interquartile range is 48.0
upper limit 137.0
```

[53]:
```
df2.drop(df2[df2["total_onshift_partners"]>upperWhisker].index , inplace=True)
```

[54]:
```
sns.boxplot(x=df2['total_onshift_partners'])
```

[54]: <Axes: xlabel='total_onshift_partners'>



3.4 Compare plots and results

- After dropping the data points the points which are above the Upper IQR the outliers are removed
- there is a clear correlation between the available and busy partners

4 Regression with neural networks

4.1 Data splitting

[55]:
```
y = df2['time_to_deliver']
X = df2.drop('time_to_deliver', axis=1, inplace=False)
y.head()
```

[55]:
```
0     62.983333
1     67.066667
8     26.433333
14    37.883333
15    49.800000
Name: time_to_deliver, dtype: float64
```

[56]:
```
X.head()
```

[56]:

	market_id	store_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_partners	total_busy_partners	total_outstanding_orders	created_hour	created_day
0	1.0	49.744347	47.875225	1.0	4	3441	4	557	1239	33.0	14.0	21.0	22	4
1	2.0	47.620343	44.329778	2.0	1	1900	1	1400	1400	1.0	2.0	2.0	21	1
8	2.0	47.620343	50.408414	3.0	4	4771	3	820	1604	8.0	6.0	18.0	0	0
14	1.0	51.090884	50.287894	1.0	1	1525	1	1525	1525	5.0	6.0	8.0	3	3
15	1.0	51.090884	50.287894	1.0	2	3620	2	1425	2195	5.0	5.0	7.0	2	1

4.2 Data scaling

[57]:
```
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

[58]:
```
X
```

[58]:
```
array([[-1.31098738,  0.34504254,  0.04450559, …, -0.70152658,
         1.54016665,  0.38765359],
       [-0.56089627,  0.01803113, -1.36530016, …, -1.07165687,
         1.42534624, -1.08073706],
       [-0.56089627,  0.01803113,  1.05179849, …, -0.7599682 ,
        -0.98588227, -1.57020061],
       …,
       [-1.31098738,  1.04872112, -1.58257626, …, -0.33139629,
        -0.52660065,  0.87711714],
       [-1.31098738,  0.1135784 , -1.46507184, …, -0.87685145,
         1.08088502,  1.36658069],
       [-1.31098738,  0.1135784 , -1.46507184, …, -0.6625655 ,
         1.19570543,  1.36658069]])
```

[59]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[61]:
```
X_train
```

[61]:
```
array([[ 0.93928595,  1.71800783,  1.01068368, …, -0.17555196,
        -0.98588227,  1.36658069],
       [-0.56089627,  2.61597274,  0.64522073, …, -0.21451304,
         1.31052583,  1.36658069],
       [ 0.93928595,  1.62646536, -1.46507184, …, -0.85737091,
         1.19570543, -1.08073706],
       …,
       [-0.56089627,  0.84115004,  0.04450559, …,  2.14263249,
        -0.75624146, -1.57020061],
       [-0.56089627, -0.17227342, -1.46507184, …, -0.27295466,
        -0.98588227, -0.59127351],
       [-0.56089627, -0.6184421 ,  0.15826863, …,  1.55821625,
        -0.52660065,  0.87711714]])
```

[65]:
```
X_train.shape
```

[65]:
```
(132248, 14)
```

[76]:
```
y_train.shape
```

[76]: (132248,)

[77]:
```
type(X_train)
```

[77]: numpy.ndarray

[78]:
```
type(y_train)
```

[78]: pandas.core.series.Series

[ ]:
```

```

4.3 Defining NN architecture

[69]:
```
import tensorflow as tf

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.losses import MeanSquaredLogarithmicError
from tensorflow.keras.optimizers import Adam
```

[66]:
```
hidden_layer1 = 28
hidden_layer2 = 50
hidden_layer3 = 80
learning_rate = 0.01

model = Sequential([
    Dense(hidden_layer1, kernel_initializer='normal', activation='relu'),
    Dense(hidden_layer2, kernel_initializer='normal', activation='relu'),
    Dense(hidden_layer3, kernel_initializer='normal', activation='relu'),
    Dense(1, kernel_initializer='normal', activation='linear')
  ])
```

[70]:
```
msle = MeanSquaredLogarithmicError()
model.compile(
    loss=msle,
    optimizer=Adam(learning_rate=learning_rate),
    metrics=[msle]
)
```

[81]:
```
history = model.fit(X_train, y_train.values, epochs=10, batch_size=13225)
```

```
Epoch 1/10
10/10 [==============================] - 1s 19ms/step - loss: 9.9346 -
mean_squared_logarithmic_error: 9.9345
Epoch 2/10
10/10 [==============================] - 0s 17ms/step - loss: 0.5985 -
mean_squared_logarithmic_error: 0.5985
```

```
Epoch 3/10
10/10 [==============================] - 0s 17ms/step - loss: 0.8152 -
mean_squared_logarithmic_error: 0.8152
Epoch 4/10
10/10 [==============================] - 0s 19ms/step - loss: 0.4175 -
mean_squared_logarithmic_error: 0.4175
Epoch 5/10
10/10 [==============================] - 0s 17ms/step - loss: 0.1520 -
mean_squared_logarithmic_error: 0.1520
Epoch 6/10
10/10 [==============================] - 0s 18ms/step - loss: 0.1586 -
mean_squared_logarithmic_error: 0.1586
Epoch 7/10
10/10 [==============================] - 0s 22ms/step - loss: 0.1219 -
mean_squared_logarithmic_error: 0.1219
Epoch 8/10
10/10 [==============================] - 0s 17ms/step - loss: 0.1197 -
mean_squared_logarithmic_error: 0.1197
Epoch 9/10
10/10 [==============================] - 0s 18ms/step - loss: 0.1134 -
mean_squared_logarithmic_error: 0.1134
Epoch 10/10
10/10 [==============================] - 0s 17ms/step - loss: 0.1111 -
mean_squared_logarithmic_error: 0.1111
```

[82]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 28)                420

 dense_1 (Dense)             (None, 50)                1450

 dense_2 (Dense)             (None, 80)                4080

 dense_3 (Dense)             (None, 1)                 81

=================================================================
Total params: 6,031
Trainable params: 6,031
Non-trainable params: 0
_________________________________________________________________
```

4.4 Trying different combinations and hyperparameters

[90]:
```
hidden_layer1 = 40
hidden_layer2 = 100
hidden_layer3 = 90
learning_rate = 0.01

model = Sequential([
    Dense(hidden_layer1, kernel_initializer='normal', activation='relu'),
    Dense(hidden_layer2, kernel_initializer='normal', activation='relu'),
    Dense(hidden_layer3, kernel_initializer='normal', activation='relu'),
    Dense(1, kernel_initializer='normal', activation='linear')
  ])
```

[91]:
```
msle = MeanSquaredLogarithmicError()
model.compile(
    loss=msle,
    optimizer=Adam(learning_rate=learning_rate),
    metrics=[msle]
)
```

4.5 Model training

[92]:
```
history = model.fit(X_train, y_train.values, epochs=100, batch_size=1323)
```

```
Epoch 1/100
100/100 [==============================] - 3s 7ms/step - loss: 0.9221 -
mean_squared_logarithmic_error: 0.9218
Epoch 2/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0980 -
mean_squared_logarithmic_error: 0.0980
Epoch 3/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0929 -
mean_squared_logarithmic_error: 0.0929
Epoch 4/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0904 -
mean_squared_logarithmic_error: 0.0904
Epoch 5/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0889 -
mean_squared_logarithmic_error: 0.0889
Epoch 6/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0880 -
mean_squared_logarithmic_error: 0.0880
Epoch 7/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0875 -
mean_squared_logarithmic_error: 0.0875
Epoch 8/100
100/100 [==============================] - 1s 10ms/step - loss: 0.0872 -
```

```
mean_squared_logarithmic_error: 0.0872
Epoch 9/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0869 -
mean_squared_logarithmic_error: 0.0869
Epoch 10/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0867 -
mean_squared_logarithmic_error: 0.0867
Epoch 11/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0866 -
mean_squared_logarithmic_error: 0.0866
Epoch 12/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0866 -
mean_squared_logarithmic_error: 0.0865
Epoch 13/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0863 -
mean_squared_logarithmic_error: 0.0863
Epoch 14/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0858 -
mean_squared_logarithmic_error: 0.0858
Epoch 15/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0856 -
mean_squared_logarithmic_error: 0.0856
Epoch 16/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0853 -
mean_squared_logarithmic_error: 0.0853
Epoch 17/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0852 -
mean_squared_logarithmic_error: 0.0852
Epoch 18/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0848 -
mean_squared_logarithmic_error: 0.0848
Epoch 19/100
100/100 [==============================] - 1s 10ms/step - loss: 0.0847 -
mean_squared_logarithmic_error: 0.0847
Epoch 20/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0844 -
mean_squared_logarithmic_error: 0.0844
Epoch 21/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0845 -
mean_squared_logarithmic_error: 0.0845
Epoch 22/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0843 -
mean_squared_logarithmic_error: 0.0843
Epoch 23/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0842 -
mean_squared_logarithmic_error: 0.0842
Epoch 24/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0841 -
```

```
mean_squared_logarithmic_error: 0.0841
Epoch 25/100
100/100 [==============================] - 1s 10ms/step - loss: 0.0840 -
mean_squared_logarithmic_error: 0.0840
Epoch 26/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0841 -
mean_squared_logarithmic_error: 0.0841
Epoch 27/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0838 -
mean_squared_logarithmic_error: 0.0838
Epoch 28/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0838 -
mean_squared_logarithmic_error: 0.0838
Epoch 29/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0838 -
mean_squared_logarithmic_error: 0.0838
Epoch 30/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0838 -
mean_squared_logarithmic_error: 0.0838
Epoch 31/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0839 -
mean_squared_logarithmic_error: 0.0839
Epoch 32/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0839 -
mean_squared_logarithmic_error: 0.0839
Epoch 33/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0838 -
mean_squared_logarithmic_error: 0.0838
Epoch 34/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0840 -
mean_squared_logarithmic_error: 0.0840
Epoch 35/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0835 -
mean_squared_logarithmic_error: 0.0835
Epoch 36/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0836 -
mean_squared_logarithmic_error: 0.0836
Epoch 37/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0836 -
mean_squared_logarithmic_error: 0.0836
Epoch 38/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0835 -
mean_squared_logarithmic_error: 0.0835
Epoch 39/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0836 -
mean_squared_logarithmic_error: 0.0836
Epoch 40/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0832 -
```

```
mean_squared_logarithmic_error: 0.0832
Epoch 41/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0833 -
mean_squared_logarithmic_error: 0.0833
Epoch 42/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0839 -
mean_squared_logarithmic_error: 0.0839
Epoch 43/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0828 -
mean_squared_logarithmic_error: 0.0828
Epoch 44/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0832 -
mean_squared_logarithmic_error: 0.0832
Epoch 45/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0829 -
mean_squared_logarithmic_error: 0.0829
Epoch 46/100
100/100 [==============================] - 1s 10ms/step - loss: 0.0834 -
mean_squared_logarithmic_error: 0.0834
Epoch 47/100
100/100 [==============================] - 1s 10ms/step - loss: 0.0832 -
mean_squared_logarithmic_error: 0.0832
Epoch 48/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0830 -
mean_squared_logarithmic_error: 0.0830
Epoch 49/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0830 -
mean_squared_logarithmic_error: 0.0830
Epoch 50/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0826 -
mean_squared_logarithmic_error: 0.0825
Epoch 51/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0830 -
mean_squared_logarithmic_error: 0.0830
Epoch 52/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0824 -
mean_squared_logarithmic_error: 0.0824
Epoch 53/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0834 -
mean_squared_logarithmic_error: 0.0834
Epoch 54/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0824 -
mean_squared_logarithmic_error: 0.0824
Epoch 55/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0826 -
mean_squared_logarithmic_error: 0.0826
Epoch 56/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0824 -
```

```
mean_squared_logarithmic_error: 0.0824
Epoch 57/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0822 -
mean_squared_logarithmic_error: 0.0822
Epoch 58/100
100/100 [==============================] - 1s 9ms/step - loss: 0.0821 -
mean_squared_logarithmic_error: 0.0821
Epoch 59/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0818 -
mean_squared_logarithmic_error: 0.0818
Epoch 60/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0822 -
mean_squared_logarithmic_error: 0.0822
Epoch 61/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0823 -
mean_squared_logarithmic_error: 0.0823
Epoch 62/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0827 -
mean_squared_logarithmic_error: 0.0827
Epoch 63/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0818 -
mean_squared_logarithmic_error: 0.0818
Epoch 64/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0816 -
mean_squared_logarithmic_error: 0.0816
Epoch 65/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0816 -
mean_squared_logarithmic_error: 0.0816
Epoch 66/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0817 -
mean_squared_logarithmic_error: 0.0817
Epoch 67/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0816 -
mean_squared_logarithmic_error: 0.0816
Epoch 68/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0817 -
mean_squared_logarithmic_error: 0.0817
Epoch 69/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0818 -
mean_squared_logarithmic_error: 0.0818
Epoch 70/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0817 -
mean_squared_logarithmic_error: 0.0817
Epoch 71/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0815 -
mean_squared_logarithmic_error: 0.0815
Epoch 72/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0815 -
```

```
mean_squared_logarithmic_error: 0.0815
Epoch 73/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0814 -
mean_squared_logarithmic_error: 0.0814
Epoch 74/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0818 -
mean_squared_logarithmic_error: 0.0818
Epoch 75/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0814 -
mean_squared_logarithmic_error: 0.0814
Epoch 76/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0813 -
mean_squared_logarithmic_error: 0.0813
Epoch 77/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0812 -
mean_squared_logarithmic_error: 0.0812
Epoch 78/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0813 -
mean_squared_logarithmic_error: 0.0813
Epoch 79/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0813 -
mean_squared_logarithmic_error: 0.0813
Epoch 80/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0812 -
mean_squared_logarithmic_error: 0.0811
Epoch 81/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0811 -
mean_squared_logarithmic_error: 0.0811
Epoch 82/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0811 -
mean_squared_logarithmic_error: 0.0811
Epoch 83/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0809 -
mean_squared_logarithmic_error: 0.0809
Epoch 84/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0811 -
mean_squared_logarithmic_error: 0.0811
Epoch 85/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0815 -
mean_squared_logarithmic_error: 0.0815
Epoch 86/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0811 -
mean_squared_logarithmic_error: 0.0811
Epoch 87/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0807 -
mean_squared_logarithmic_error: 0.0807
Epoch 88/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0810 -
```

```
mean_squared_logarithmic_error: 0.0810
Epoch 89/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0810 -
mean_squared_logarithmic_error: 0.0810
Epoch 90/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0808 -
mean_squared_logarithmic_error: 0.0808
Epoch 91/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0809 -
mean_squared_logarithmic_error: 0.0809
Epoch 92/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0808 -
mean_squared_logarithmic_error: 0.0808
Epoch 93/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0807 -
mean_squared_logarithmic_error: 0.0807
Epoch 94/100
100/100 [==============================] - 1s 8ms/step - loss: 0.0806 -
mean_squared_logarithmic_error: 0.0806
Epoch 95/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0810 -
mean_squared_logarithmic_error: 0.0810
Epoch 96/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0808 -
mean_squared_logarithmic_error: 0.0808
Epoch 97/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0808 -
mean_squared_logarithmic_error: 0.0808
Epoch 98/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0806 -
mean_squared_logarithmic_error: 0.0806
Epoch 99/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0800 -
mean_squared_logarithmic_error: 0.0800
Epoch 100/100
100/100 [==============================] - 1s 7ms/step - loss: 0.0805 -
mean_squared_logarithmic_error: 0.0805
```

[93]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense_4 (Dense)              (None, 40)                600

 dense_5 (Dense)              (None, 100)               4100
```

```
 dense_6 (Dense)             (None, 90)                9090

 dense_7 (Dense)             (None, 1)                 91

=================================================================
Total params: 13,881
Trainable params: 13,881
Non-trainable params: 0
_________________________________________________________________
```

[ ]:

[ ]:

[ ]: