Porter Case Study

Problem Statement

Porter is India's Largest Marketplace for Intra-City Logistics. Leader in the country's $40 billion intra-city logistics market, Porter strives to improve the lives of 1,50,000+ driver-partners by providing them with consistent earning & independence. Currently, the company has serviced 5+ million customers. Porter works with a wide range of restaurants for delivering their items directly to the people.

Porter has a number of delivery partners available for delivering the food, from various restaurants and wants to get an estimated delivery time that it can provide the customers on the basis of what they are ordering, from where and also the delivery partners.

This dataset has the required data to train a regression model that will do the delivery time estimation, based on all those features

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import os

import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('/Users/bose/Downloads/porter.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	market_id	created_at	actual_delivery_time	store_
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc543!
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba2
2	3.0	2015-01-22 20:39:28	2015-01-22 21:09:09	f0ade77b43923b38237db569b016ba2
3	3.0	2015-02-03 21:21:45	2015-02-03 22:13:00	f0ade77b43923b38237db569b016ba2
4	3.0	2015-02-15 02:40:36	2015-02-15 03:20:26	f0ade77b43923b38237db569b016ba2

In [4]:
```
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
```

In [5]:
```
df.isna().sum()
```

Out[5]:
```
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
```

In [6]:
```
df.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
df.shape
```

Out[7]:
```
(197428, 14)
```

In [8]:
```
df.dropna(inplace=True)
```

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

Out[9]:
```
(176248, 14)
```

In [10]:
```
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  object
 2   actual_delivery_time      176248 non-null  object
 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: float64(5), int64(5), object(4)
memory usage: 20.2+ MB
```

In [11]:
```
 df["order_protocol"] = df.order_protocol.astype("category").cat.codes
df["store_primary_category"] = df.store_primary_category.astype("category
df["market_id"] = df.market_id.astype("category").cat.codes
```

In [12]:
```
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  int8
 1   created_at                176248 non-null  object
 2   actual_delivery_time      176248 non-null  object
 3   store_id                  176248 non-null  object
 4   store_primary_category    176248 non-null  int8
 5   order_protocol            176248 non-null  int8
 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: float64(3), int64(5), int8(3), object(3)
memory usage: 16.6+ MB
```

In [13]:
```
df["created_at"] = pd.to_datetime(df["created_at"])
df["actual_delivery_time"] = pd.to_datetime(df["actual_delivery_time"])
```

In [14]:
```
df["time_taken_mins"] = (df["actual_delivery_time"] - df["created_at"]).d
```

In [15]:
```
df.head()
```

Out[15]:

	market_id	created_at	actual_delivery_time	store_
0	0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc543
1	1	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba
8	1	2015-02-16 00:11:35	2015-02-16 00:38:01	f0ade77b43923b38237db569b016ba
14	0	2015-02-12 03:36:46	2015-02-12 04:14:39	ef1e491a766ce3127556063d49bc2f
15	0	2015-01-27 02:12:36	2015-01-27 03:02:24	ef1e491a766ce3127556063d49bc2f

In [16]:
```
order_volumes = df.groupby('market_id').size()
print("Distribution of Order Volumes Across Different Markets:")
print(order_volumes)
```

```
Distribution of Order Volumes Across Different Markets:
market_id
0    37207
1    53625
2    21119
3    46359
4    17298
5      640
dtype: int64
```

In [17]:
```python
df['order_hour'] = df['created_at'].dt.hour

order_frequency = df['order_hour'].value_counts().sort_index()

peak_hours = order_frequency.idxmax()
peak_order_count = order_frequency.max()
print(order_frequency)
print(f"({peak_hours} - {peak_hours + 1}) hour is the peak time for order
```

```
order_hour
0     11466
1     25734
2     32940
3     23719
4     13254
5      6079
6      1223
7         9
8         2
14       39
15      504
16     1945
17     3071
18     4546
19    12214
20    14014
21    10274
22     7877
23     7338
Name: count, dtype: int64
(2 - 3) hour is the peak time for order placements with 32940 orders.
```

In [18]:
```python
df['order_day_of_week'] = df['created_at'].dt.dayofweek

day_names = {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4:
df['order_day_name'] = df['order_day_of_week'].map(day_names)

order_volumes_by_day = df['order_day_name'].value_counts()
peak_days = order_volumes_by_day.idxmax()
peak_order_count = order_volumes_by_day.max()
print(order_volumes_by_day)
print(f"{peak_days} is the day with the highest order volume with {peak_o
```

```
order_day_name
Saturday     30858
Sunday       29898
Friday       25012
Monday       24202
Thursday     22997
Wednesday    21796
Tuesday      21485
Name: count, dtype: int64
Saturday is the day with the highest order volume with 30858 orders.
```

In [19]:
```
df.head(5)
```

Out[19]:

	market_id	created_at	actual_delivery_time	store_
0	0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc543
1	1	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba
8	1	2015-02-16 00:11:35	2015-02-16 00:38:01	f0ade77b43923b38237db569b016ba
14	0	2015-02-12 03:36:46	2015-02-12 04:14:39	ef1e491a766ce3127556063d49bc2f
15	0	2015-01-27 02:12:36	2015-01-27 03:02:24	ef1e491a766ce3127556063d49bc2f

In [20]:
```
df.describe().transpose()
```

Out[20]:

	count	mean	min	25%
market_id	176248.0	1.743747	0.0	1.0
created_at	176248	2015-02-04 19:35:43.333773824	2015-01-21 15:22:03	2015-01-29 01:37:01.500000
actual_delivery_time	176248	2015-02-04 20:23:29.186373632	2015-01-21 16:16:34	2015-01-29 02:24:29
store_primary_category	176248.0	35.891482	0.0	18.0
order_protocol	176248.0	1.911687	0.0	0.0
total_items	176248.0	3.204592	1.0	2.0
subtotal	176248.0	2696.498939	0.0	1408.0
num_distinct_items	176248.0	2.674589	1.0	1.0
min_item_price	176248.0	684.93773	-86.0	299.0
max_item_price	176248.0	1159.886994	0.0	799.0
total_onshift_partners	176248.0	44.905276	-4.0	17.0
total_busy_partners	176248.0	41.845434	-5.0	15.0
total_outstanding_orders	176248.0	58.2068	-6.0	17.0
time_taken_mins	176248.0	47.76421	1.683333	35.083333
order_hour	176248.0	8.493872	0.0	2.0
order_day_of_week	176248.0	3.221563	0.0	1.0

In [21]:
```
df.store_primary_category.nunique()
```

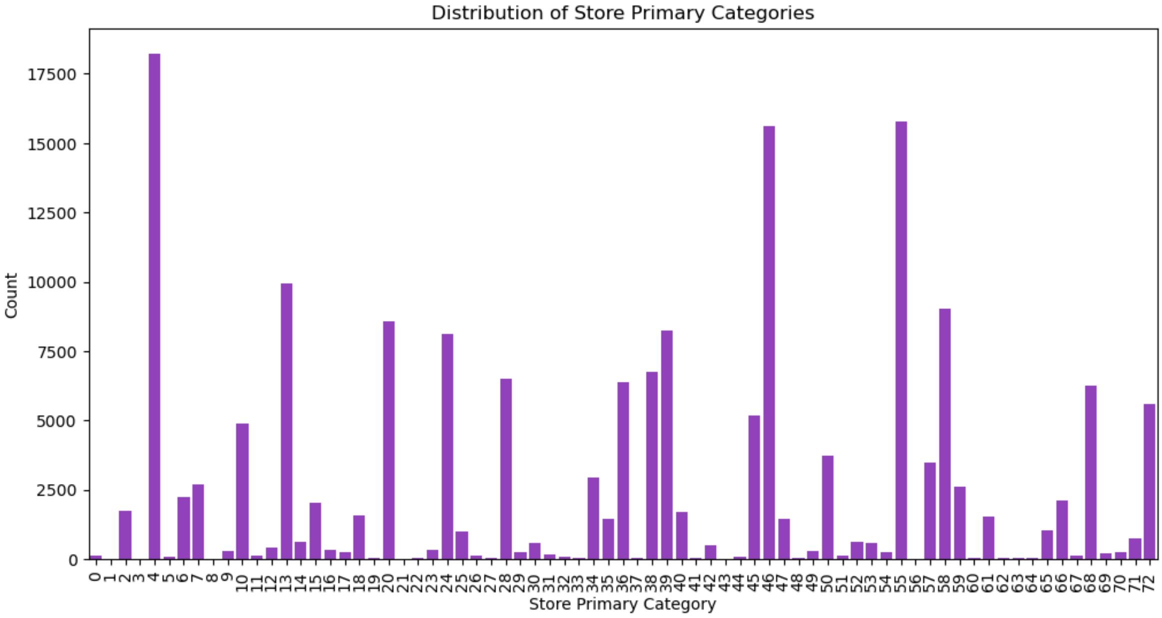
Out[21]: 73

In [22]:
```
df.order_protocol.nunique()
```

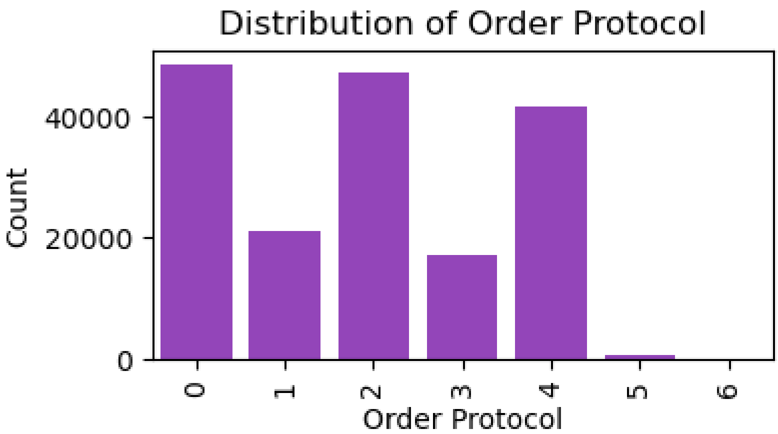
Out[22]: 7

Visual Analysis

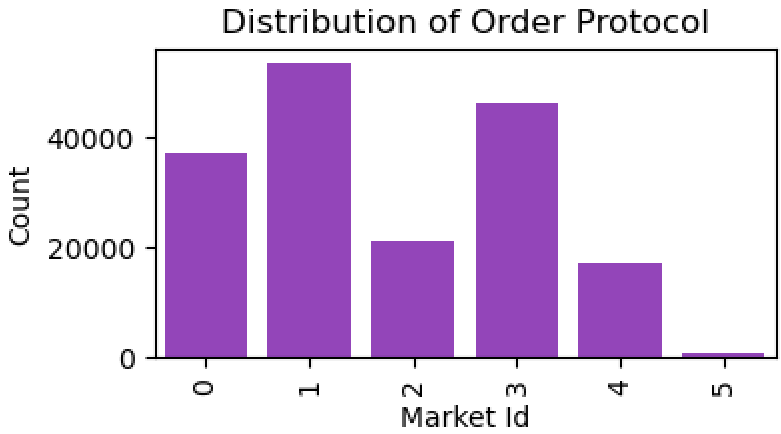
In [23]:
```
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='store_primary_category', color='darkorchid')
plt.xticks(rotation=90)
plt.xlabel('Store Primary Category')
plt.ylabel('Count')
plt.title('Distribution of Store Primary Categories')
plt.show()
```



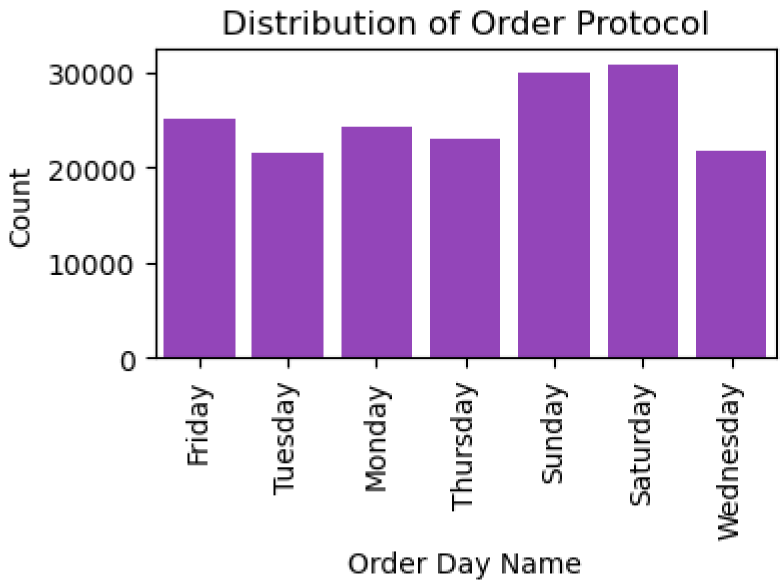
In [24]:
```
plt.figure(figsize=(4, 2))
sns.countplot(data=df, x='order_protocol', color='darkorchid')
plt.xticks(rotation=90)
plt.xlabel('Order Protocol')
plt.ylabel('Count')
plt.title('Distribution of Order Protocol')
plt.show()
```



In [25]:
```
plt.figure(figsize=(4, 2))
sns.countplot(data=df, x='market_id', color='darkorchid')
plt.xticks(rotation=90)
plt.xlabel('Market Id')
plt.ylabel('Count')
plt.title('Distribution of Order Protocol')
plt.show()
```



In [26]:
```
plt.figure(figsize=(4, 2))
sns.countplot(data=df, x='order_day_name', color='darkorchid')
plt.xticks(rotation=90)
plt.xlabel('Order Day Name')
plt.ylabel('Count')
plt.title('Distribution of Order Protocol')
plt.show()
```



In [27]:
```
df.drop(['order_day_name','created_at','actual_delivery_time', 'store_id'
```

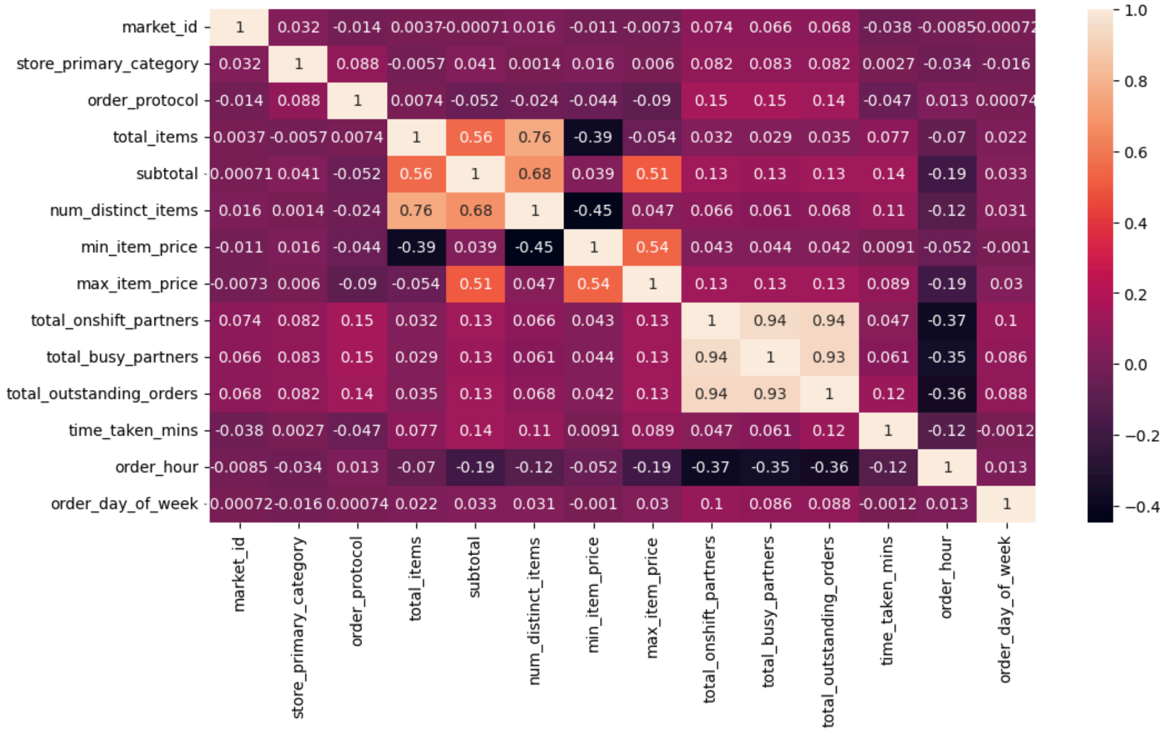
In [28]:
```
df.head()
```

Out[28]:

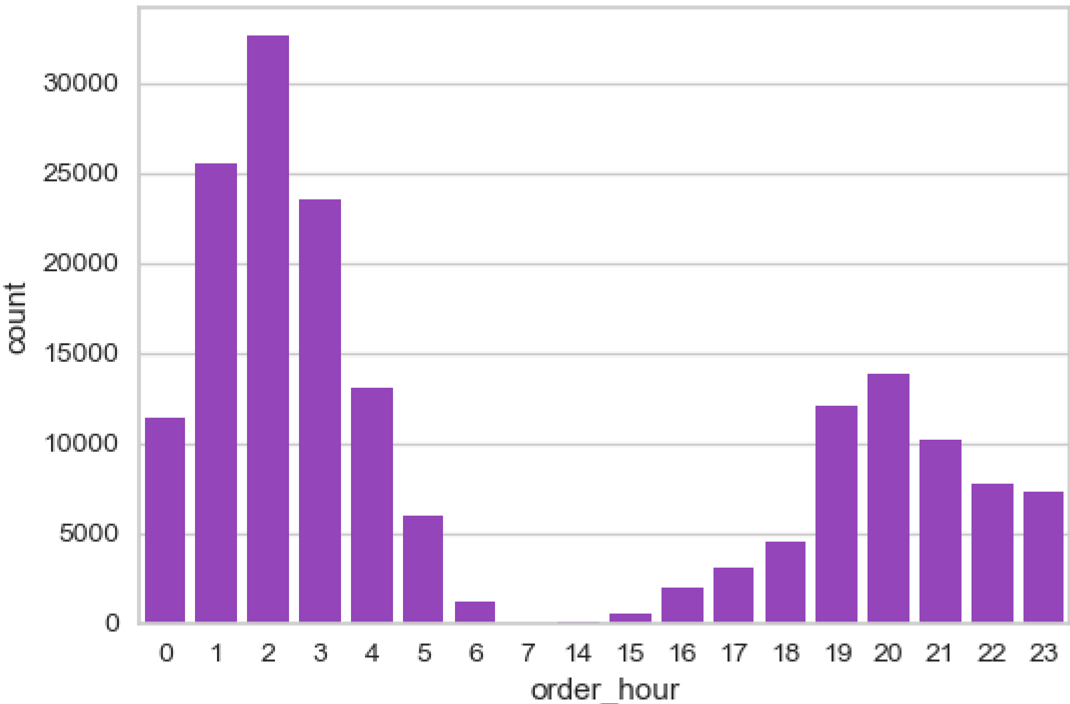
	market_id	store_primary_category	order_protocol	total_items	subtotal	num_
0	0	4	0	4	3441	
1	1	46	1	1	1900	
8	1	36	2	4	4771	
14	0	38	0	1	1525	
15	0	38	0	2	3620	

Correlation Analysis

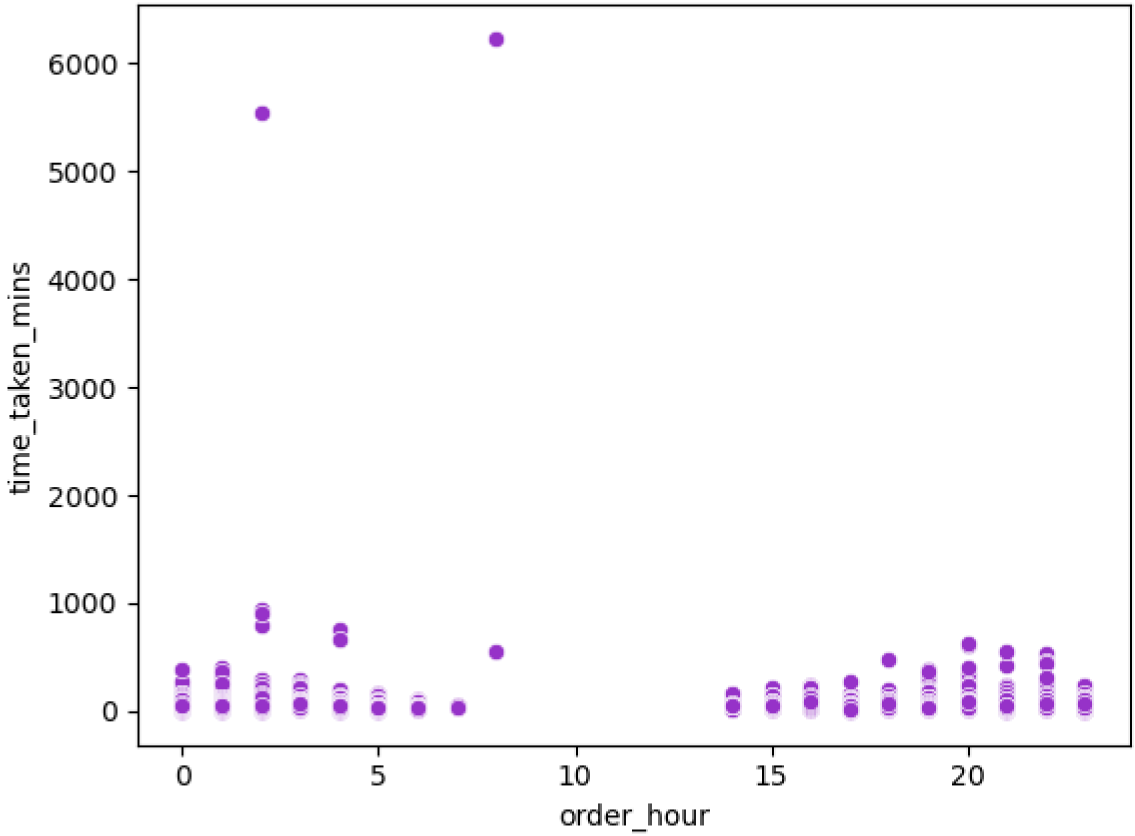
In [29]:
```
plt.figure(figsize=(12, 6))
sns.heatmap(df.corr(), annot=True)
plt.show()
```



In [82]:
```
plt.figure(figsize=(6,4))
sns.countplot(x=df['order_hour'], color='darkorchid')
plt.show()
```

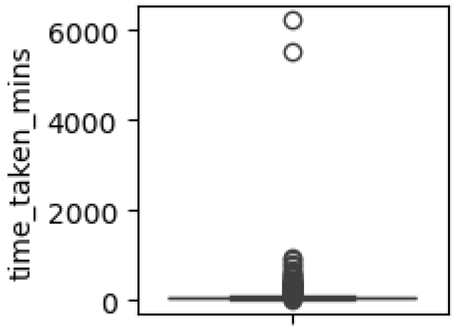


In [31]:
```
sns.scatterplot(x='order_hour',y='time_taken_mins',data=df,color='darkorc
plt.show()
```



Detecting Outliers

In [32]:
```
plt.figure(figsize=(2,2))
sns.boxplot(y='time_taken_mins',data=df)
plt.xticks(rotation=90);
plt.show()
```



In [33]:
```
from sklearn.neighbors import LocalOutlierFactor
import matplotlib.pyplot as plt
model1=LocalOutlierFactor()
df['lof_anomaly_score']=model1.fit_predict(df)
df.lof_anomaly_score.value_counts()
```

Out[33]:
```
lof_anomaly_score
 1    174312
-1      1936
Name: count, dtype: int64
```

In [34]:
```
df.shape
```

Out[34]: (176248, 15)

In [35]:
```
df.drop(df[df['lof_anomaly_score']==-1].index,inplace=True)
```

In [36]:
```
df.shape
```

Out[36]: (174312, 15)

Traiin & Test Split

In [37]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_e
```

In [38]:
```
y = df["time_taken_mins"]
X = df.drop(["time_taken_mins"], axis = 1)
```

In [39]:
```
X.columns
```

Out[39]:
```
Index(['market_id', 'store_primary_category', 'order_protocol', 'total_i
tems',
       'subtotal', 'num_distinct_items', 'min_item_price', 'max_item_pri
ce',
       'total_onshift_partners', 'total_busy_partners',
       'total_outstanding_orders', 'order_hour', 'order_day_of_week',
       'lof_anomaly_score'],
      dtype='object')
```

In [40]:
```
X_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
```

In [41]:
```
X_train.shape
```

Out[41]: (139449, 14)

In [42]:
```
x_test.shape
```

Out[42]: (34863, 14)

In [43]:
```
# Scaling the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(x_test)
```

In [44]:
```
def metrics_evals(y_true,y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = mean_squared_error(y_true, y_pred, squared=False)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)

    return {"MSE":mse,
            "RMSE":rmse,
            "MAE":mae,
            "R2":r2}
```

Random Forest Regressor

In [45]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [46]:
```
regressor = RandomForestRegressor()
regressor.fit(X_train_scaled, y_train)
```

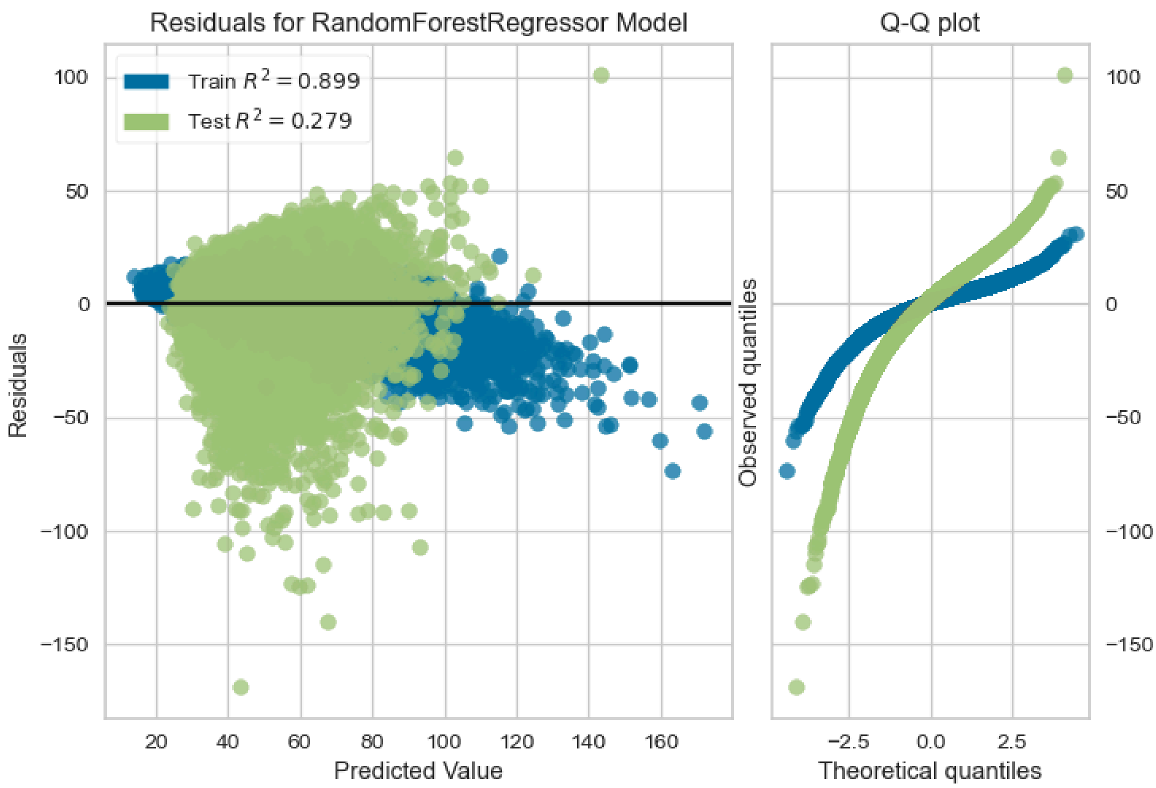
Out[46]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [47]:
```
prediction = regressor.predict(X_test_scaled)
```

In [48]:
```
eval = metrics_evals(y_test, prediction)
eval
```

Out[48]:
```
{'MSE': 226.89873001512498,
 'RMSE': 15.063158035920788,
 'MAE': 11.223980342618887,
 'R2': 0.27912390144128596}
```

In [49]:
```
from yellowbrick.regressor import ResidualsPlot
visualizer = ResidualsPlot(regressor, hist=False, qqplot=True)
visualizer.fit(X_train_scaled, y_train)
visualizer.score(X_test_scaled, y_test)
visualizer.show()
plt.show()
```



Inference -

- Model is overfitting

- It is not fitting he regression line
- We can try the regularizer.

xG Booster Regressor

In [50]:
```
from xgboost import XGBRegressor


xgb = XGBRegressor(n_estimators = 300,
                   n_jobs = -1,
                   random_state = 123)

xgb.fit(X_train_scaled, y_train, eval_set = [(X_test_scaled, y_test)], ve
```

```
[0]	validation_0-rmse:17.00744
[1]	validation_0-rmse:16.57070
[2]	validation_0-rmse:16.21982
[3]	validation_0-rmse:16.01951
[4]	validation_0-rmse:15.88715
[5]	validation_0-rmse:15.76019
[6]	validation_0-rmse:15.59783
[7]	validation_0-rmse:15.52408
[8]	validation_0-rmse:15.45382
[9]	validation_0-rmse:15.39742
[10]	validation_0-rmse:15.35545
[11]	validation_0-rmse:15.28972
[12]	validation_0-rmse:15.23224
[13]	validation_0-rmse:15.20058
[14]	validation_0-rmse:15.15008
[15]	validation_0-rmse:15.11251
[16]	validation_0-rmse:15.07938
[17]	validation_0-rmse:15.06744
[18]	validation_0-rmse:15.04246
[19]	validation_0-rmse:15.01950
[20]	validation_0-rmse:15.01212
[21]	validation_0-rmse:14.98965
[22]	validation_0-rmse:14.98160
[23]	validation_0-rmse:14.97268
[24]	validation_0-rmse:14.96290
[25]	validation_0-rmse:14.94472
[26]	validation_0-rmse:14.93571
[27]	validation_0-rmse:14.92581
[28]	validation_0-rmse:14.92060
[29]	validation_0-rmse:14.91633
[30]	validation_0-rmse:14.91000
[31]	validation_0-rmse:14.89890
[32]	validation_0-rmse:14.87389
[33]	validation_0-rmse:14.86121
[34]	validation_0-rmse:14.85987
[35]	validation_0-rmse:14.85360
[36]	validation_0-rmse:14.84890
[37]	validation_0-rmse:14.84682
[38]	validation_0-rmse:14.83760
[39]	validation_0-rmse:14.82956
[40]	validation_0-rmse:14.82926
[41]	validation_0-rmse:14.82718
[42]	validation_0-rmse:14.82333
[43]	validation_0-rmse:14.82076
[44]	validation_0-rmse:14.81746
[45]	validation_0-rmse:14.79766
[46]	validation_0-rmse:14.79300
[47]	validation_0-rmse:14.79244
[48]	validation_0-rmse:14.79147
[49]	validation_0-rmse:14.79212
[50]	validation_0-rmse:14.79092
[51]	validation_0-rmse:14.78590
[52]	validation_0-rmse:14.78494
[53]	validation_0-rmse:14.78434
[54]	validation_0-rmse:14.78319
[55]	validation_0-rmse:14.77892
[56]	validation_0-rmse:14.77969
[57]	validation_0-rmse:14.78004
[58]	validation_0-rmse:14.78003
[59]	validation_0-rmse:14.77970
```

```
[60]	validation_0-rmse:14.77643
[61]	validation_0-rmse:14.77743
[62]	validation_0-rmse:14.77905
[63]	validation_0-rmse:14.77456
[64]	validation_0-rmse:14.77369
[65]	validation_0-rmse:14.77373
[66]	validation_0-rmse:14.77455
[67]	validation_0-rmse:14.77432
[68]	validation_0-rmse:14.77419
[69]	validation_0-rmse:14.77005
[70]	validation_0-rmse:14.76943
[71]	validation_0-rmse:14.76806
[72]	validation_0-rmse:14.76746
[73]	validation_0-rmse:14.76725
[74]	validation_0-rmse:14.76699
[75]	validation_0-rmse:14.76636
[76]	validation_0-rmse:14.75719
[77]	validation_0-rmse:14.75696
[78]	validation_0-rmse:14.75669
[79]	validation_0-rmse:14.75622
[80]	validation_0-rmse:14.75625
[81]	validation_0-rmse:14.75639
[82]	validation_0-rmse:14.75559
[83]	validation_0-rmse:14.75428
[84]	validation_0-rmse:14.75293
[85]	validation_0-rmse:14.75292
[86]	validation_0-rmse:14.74982
[87]	validation_0-rmse:14.75119
[88]	validation_0-rmse:14.75116
[89]	validation_0-rmse:14.75334
[90]	validation_0-rmse:14.75414
[91]	validation_0-rmse:14.75420
[92]	validation_0-rmse:14.75462
[93]	validation_0-rmse:14.75499
[94]	validation_0-rmse:14.75493
[95]	validation_0-rmse:14.75556
[96]	validation_0-rmse:14.75554
[97]	validation_0-rmse:14.75716
[98]	validation_0-rmse:14.75783
[99]	validation_0-rmse:14.75774
[100]	validation_0-rmse:14.75765
[101]	validation_0-rmse:14.75888
[102]	validation_0-rmse:14.75591
[103]	validation_0-rmse:14.75559
[104]	validation_0-rmse:14.75667
[105]	validation_0-rmse:14.75852
[106]	validation_0-rmse:14.75954
[107]	validation_0-rmse:14.75895
[108]	validation_0-rmse:14.75859
[109]	validation_0-rmse:14.75987
[110]	validation_0-rmse:14.76061
[111]	validation_0-rmse:14.76123
[112]	validation_0-rmse:14.76061
[113]	validation_0-rmse:14.76219
[114]	validation_0-rmse:14.76228
[115]	validation_0-rmse:14.76098
[116]	validation_0-rmse:14.76180
[117]	validation_0-rmse:14.76156
[118]	validation_0-rmse:14.76092
[119]	validation_0-rmse:14.76217
```

```
[120]	validation_0-rmse:14.76080
[121]	validation_0-rmse:14.76216
[122]	validation_0-rmse:14.76099
[123]	validation_0-rmse:14.76231
[124]	validation_0-rmse:14.76273
[125]	validation_0-rmse:14.76617
[126]	validation_0-rmse:14.76815
[127]	validation_0-rmse:14.76700
[128]	validation_0-rmse:14.76352
[129]	validation_0-rmse:14.76288
[130]	validation_0-rmse:14.75950
[131]	validation_0-rmse:14.76196
[132]	validation_0-rmse:14.76248
[133]	validation_0-rmse:14.76543
[134]	validation_0-rmse:14.76534
[135]	validation_0-rmse:14.76626
[136]	validation_0-rmse:14.76656
[137]	validation_0-rmse:14.76663
[138]	validation_0-rmse:14.76846
[139]	validation_0-rmse:14.76755
[140]	validation_0-rmse:14.76805
[141]	validation_0-rmse:14.76847
[142]	validation_0-rmse:14.76876
[143]	validation_0-rmse:14.77143
[144]	validation_0-rmse:14.77058
[145]	validation_0-rmse:14.77183
[146]	validation_0-rmse:14.77312
[147]	validation_0-rmse:14.77294
[148]	validation_0-rmse:14.77320
[149]	validation_0-rmse:14.77526
[150]	validation_0-rmse:14.77536
[151]	validation_0-rmse:14.77653
[152]	validation_0-rmse:14.77709
[153]	validation_0-rmse:14.77643
[154]	validation_0-rmse:14.77704
[155]	validation_0-rmse:14.77501
[156]	validation_0-rmse:14.77606
[157]	validation_0-rmse:14.77764
[158]	validation_0-rmse:14.77870
[159]	validation_0-rmse:14.77820
[160]	validation_0-rmse:14.77837
[161]	validation_0-rmse:14.77954
[162]	validation_0-rmse:14.77695
[163]	validation_0-rmse:14.77894
[164]	validation_0-rmse:14.77749
[165]	validation_0-rmse:14.77583
[166]	validation_0-rmse:14.77474
[167]	validation_0-rmse:14.77487
[168]	validation_0-rmse:14.77526
[169]	validation_0-rmse:14.77336
[170]	validation_0-rmse:14.77304
[171]	validation_0-rmse:14.77332
[172]	validation_0-rmse:14.77352
[173]	validation_0-rmse:14.77396
[174]	validation_0-rmse:14.77263
[175]	validation_0-rmse:14.77351
[176]	validation_0-rmse:14.77481
[177]	validation_0-rmse:14.77549
[178]	validation_0-rmse:14.77568
[179]	validation_0-rmse:14.77568
```

```
[180]	validation_0-rmse:14.77631
[181]	validation_0-rmse:14.77670
[182]	validation_0-rmse:14.77652
[183]	validation_0-rmse:14.77766
[184]	validation_0-rmse:14.77955
[185]	validation_0-rmse:14.77952
[186]	validation_0-rmse:14.78289
[187]	validation_0-rmse:14.78437
[188]	validation_0-rmse:14.78483
[189]	validation_0-rmse:14.78413
[190]	validation_0-rmse:14.78295
[191]	validation_0-rmse:14.78349
[192]	validation_0-rmse:14.78341
[193]	validation_0-rmse:14.78463
[194]	validation_0-rmse:14.78496
[195]	validation_0-rmse:14.78488
[196]	validation_0-rmse:14.78474
[197]	validation_0-rmse:14.78602
[198]	validation_0-rmse:14.78652
[199]	validation_0-rmse:14.78600
[200]	validation_0-rmse:14.78544
[201]	validation_0-rmse:14.78678
[202]	validation_0-rmse:14.78752
[203]	validation_0-rmse:14.78651
[204]	validation_0-rmse:14.78605
[205]	validation_0-rmse:14.78588
[206]	validation_0-rmse:14.78625
[207]	validation_0-rmse:14.78797
[208]	validation_0-rmse:14.78915
[209]	validation_0-rmse:14.79171
[210]	validation_0-rmse:14.78977
[211]	validation_0-rmse:14.79002
[212]	validation_0-rmse:14.79055
[213]	validation_0-rmse:14.79309
[214]	validation_0-rmse:14.79290
[215]	validation_0-rmse:14.79441
[216]	validation_0-rmse:14.79331
[217]	validation_0-rmse:14.79279
[218]	validation_0-rmse:14.79367
[219]	validation_0-rmse:14.79439
[220]	validation_0-rmse:14.79688
[221]	validation_0-rmse:14.79771
[222]	validation_0-rmse:14.79767
[223]	validation_0-rmse:14.79740
[224]	validation_0-rmse:14.79641
[225]	validation_0-rmse:14.79658
[226]	validation_0-rmse:14.79720
[227]	validation_0-rmse:14.79720
[228]	validation_0-rmse:14.79511
[229]	validation_0-rmse:14.79512
[230]	validation_0-rmse:14.79647
[231]	validation_0-rmse:14.79743
[232]	validation_0-rmse:14.79813
[233]	validation_0-rmse:14.79895
[234]	validation_0-rmse:14.79926
[235]	validation_0-rmse:14.79925
[236]	validation_0-rmse:14.79918
[237]	validation_0-rmse:14.79637
[238]	validation_0-rmse:14.79785
[239]	validation_0-rmse:14.79774
```

```
[240]	validation_0-rmse:14.79802
[241]	validation_0-rmse:14.80026
[242]	validation_0-rmse:14.80257
[243]	validation_0-rmse:14.80348
[244]	validation_0-rmse:14.80353
[245]	validation_0-rmse:14.80514
[246]	validation_0-rmse:14.80710
[247]	validation_0-rmse:14.80683
[248]	validation_0-rmse:14.80853
[249]	validation_0-rmse:14.81087
[250]	validation_0-rmse:14.81172
[251]	validation_0-rmse:14.81304
[252]	validation_0-rmse:14.81216
[253]	validation_0-rmse:14.81135
[254]	validation_0-rmse:14.81202
[255]	validation_0-rmse:14.81136
[256]	validation_0-rmse:14.81100
[257]	validation_0-rmse:14.81210
[258]	validation_0-rmse:14.81328
[259]	validation_0-rmse:14.81470
[260]	validation_0-rmse:14.81498
[261]	validation_0-rmse:14.81484
[262]	validation_0-rmse:14.81399
[263]	validation_0-rmse:14.81456
[264]	validation_0-rmse:14.81436
[265]	validation_0-rmse:14.81348
[266]	validation_0-rmse:14.81478
[267]	validation_0-rmse:14.81633
[268]	validation_0-rmse:14.81758
[269]	validation_0-rmse:14.81688
[270]	validation_0-rmse:14.81784
[271]	validation_0-rmse:14.81978
[272]	validation_0-rmse:14.82078
[273]	validation_0-rmse:14.81905
[274]	validation_0-rmse:14.82026
[275]	validation_0-rmse:14.82090
[276]	validation_0-rmse:14.82236
[277]	validation_0-rmse:14.82395
[278]	validation_0-rmse:14.82498
[279]	validation_0-rmse:14.82475
[280]	validation_0-rmse:14.82468
[281]	validation_0-rmse:14.82417
[282]	validation_0-rmse:14.82577
[283]	validation_0-rmse:14.82443
[284]	validation_0-rmse:14.82482
[285]	validation_0-rmse:14.82658
[286]	validation_0-rmse:14.82585
[287]	validation_0-rmse:14.82778
[288]	validation_0-rmse:14.82901
[289]	validation_0-rmse:14.82939
[290]	validation_0-rmse:14.83031
[291]	validation_0-rmse:14.83176
[292]	validation_0-rmse:14.83452
[293]	validation_0-rmse:14.83581
[294]	validation_0-rmse:14.83799
[295]	validation_0-rmse:14.83805
[296]	validation_0-rmse:14.83811
[297]	validation_0-rmse:14.84099
[298]	validation_0-rmse:14.84114
[299]	validation_0-rmse:14.84032
```

Out[50]:

XGBRegressor

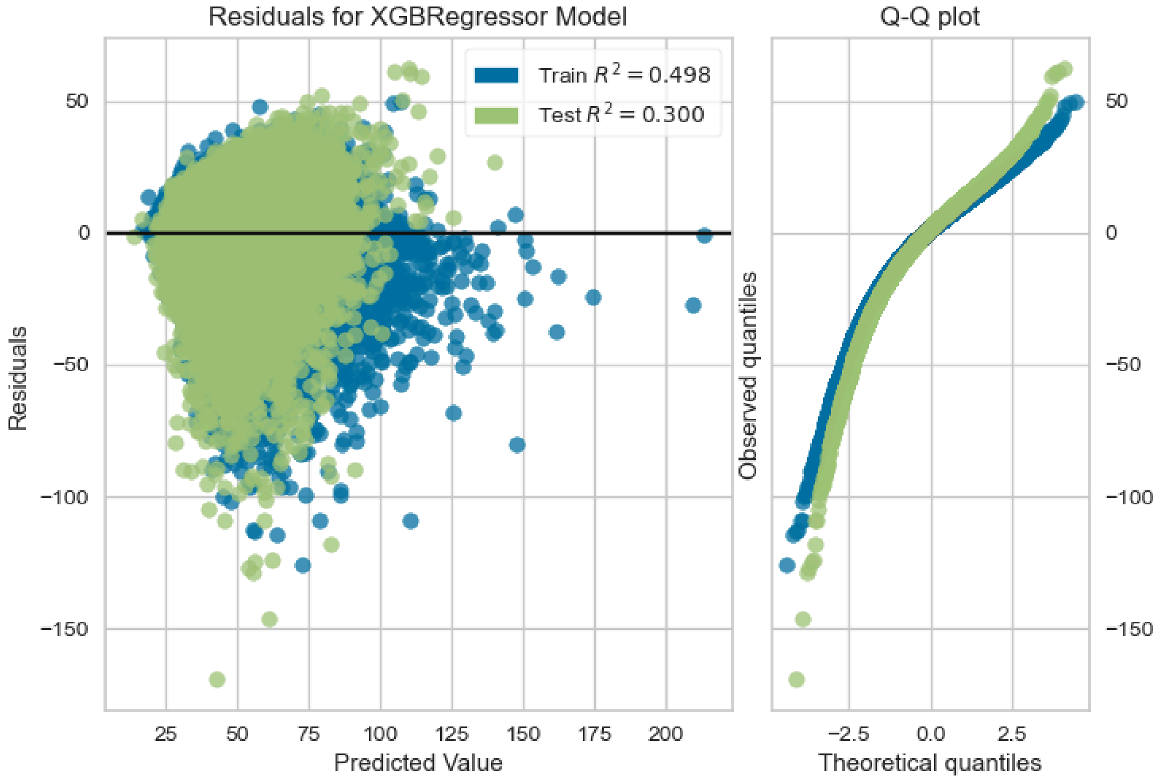
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping
_rounds=None,
             enable_categorical=False, eval_metric=None, featur
e_types=None,
             gamma=None, grow_policy=None, importance_type=Non
e,
             interaction_constraints=None, learning_rate=None,
```

In [51]:
```
prediction = xgb.predict(X_test_scaled)
```

In [52]:
```
print(metrics_evals(y_test, prediction))
```

```
{'MSE': 220.23519870240216, 'RMSE': 14.840323402891265, 'MAE': 10.97097322
3221266, 'R2': 0.3002944935156411}
```

In [53]:
```
visualizer = ResidualsPlot(xgb, hist=False, qqplot=True)
visualizer.fit(X_train_scaled, y_train)
visualizer.score(X_test_scaled, y_test)
visualizer.show()
plt.show()
```



Inference -

- There is a good fitted line without doing any regularization and engineering
- It captures more information than Random Forest
- Train R2 score = 0.498 and Test R2 score = 0.3
- The scores indicates the model is slightly overfitted

In [54]:
```
import os

# Create a function to implement a ModelCheckpoint callback with a specif
def create_model_checkpoint(model_name, save_path="model_experiments"):
  return tf.keras.callbacks.ModelCheckpoint(filepath=os.path.join(save_pa
                                            verbose=0, # only output a li
                                            save_best_only=True) # save o
```

RNA

In [57]:
```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
```

In [58]:
```
import tensorflow as tf
import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Input
from keras import Sequential
```

NN Model

In [59]:
```
tf.random.set_seed(42)
np.random.seed(42)

model = Sequential()
model.add(Input(shape=(X_train.shape[1], )))

model.add(Dense(16, kernel_initializer="normal", activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(512, activation='relu'))

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='linear'))
```

Model Architecture:

- Input Layer: First, we define the input layer to accept the features for the regression problem.
- Hidden Layers: We add three fully connected hidden layers:

    - The first hidden layer consists of 512 units and uses the ReLU activation function.
    - The second hidden layer has 256 units, again using ReLU activation.
    - The third hidden layer has 128 units, still using ReLU activation.
- Output Layer: Since this is a regression task, the output layer consists of a single neuron with a linear activation function to produce continuous values.
- Optimizer: We utilize the Adam optimizer, a widely used adaptive learning rate method that works well for a variety of tasks.
- Loss Function: To optimize the model, we use Mean Squared Error (MSE) as the loss function, which is commonly used in regression problems.
- Metrics: We track the MSE (Mean Squared Error) during training to measure the performance of the model.

In [61]:
```
from tensorflow.keras.optimizers import Adam
adam=Adam(learning_rate=0.01)
model.compile(loss='mse',optimizer=adam,metrics=['mse','mae'])
history=model.fit(X_train_scaled,y_train,epochs=100,batch_size=512,verbos
```

```
Epoch 1/100
218/218 [==============================] - 2s 7ms/step - loss: 330.8075 - 
mse: 330.8075 - mae: 13.3763 - val_loss: 243.3013 - val_mse: 243.3013 - va
l_mae: 11.5957
Epoch 2/100
218/218 [==============================] - 1s 6ms/step - loss: 256.0074 - 
mse: 256.0074 - mae: 11.9746 - val_loss: 238.5121 - val_mse: 238.5121 - va
l_mae: 11.3915
Epoch 3/100
218/218 [==============================] - 1s 6ms/step - loss: 254.9198 - 
mse: 254.9198 - mae: 11.9282 - val_loss: 252.8592 - val_mse: 252.8592 - va
l_mae: 11.4551
Epoch 4/100
218/218 [==============================] - 1s 6ms/step - loss: 252.4400 - 
mse: 252.4400 - mae: 11.8697 - val_loss: 237.4785 - val_mse: 237.4785 - va
l_mae: 11.8283
Epoch 5/100
218/218 [==============================] - 1s 6ms/step - loss: 252.9037 - 
mse: 252.9037 - mae: 11.8821 - val_loss: 241.1079 - val_mse: 241.1079 - va
l_mae: 11.2973
Epoch 6/100
218/218 [==============================] - 1s 6ms/step - loss: 250.3269 - 
mse: 250.3269 - mae: 11.8098 - val_loss: 245.0710 - val_mse: 245.0710 - va
l_mae: 11.3290
Epoch 7/100
218/218 [==============================] - 1s 6ms/step - loss: 251.0205 - 
mse: 251.0205 - mae: 11.8352 - val_loss: 242.3018 - val_mse: 242.3018 - va
l_mae: 11.4666
Epoch 8/100
218/218 [==============================] - 1s 6ms/step - loss: 249.4771 - 
mse: 249.4771 - mae: 11.7812 - val_loss: 231.4435 - val_mse: 231.4435 - va
l_mae: 11.3336
Epoch 9/100
218/218 [==============================] - 1s 6ms/step - loss: 251.1477 - 
mse: 251.1477 - mae: 11.8371 - val_loss: 234.0397 - val_mse: 234.0397 - va
l_mae: 11.5686
Epoch 10/100
218/218 [==============================] - 1s 6ms/step - loss: 249.3399 - 
mse: 249.3399 - mae: 11.7995 - val_loss: 234.8737 - val_mse: 234.8737 - va
l_mae: 11.6737
Epoch 11/100
218/218 [==============================] - 1s 7ms/step - loss: 248.2662 - 
mse: 248.2662 - mae: 11.7616 - val_loss: 232.0869 - val_mse: 232.0869 - va
l_mae: 11.3465
Epoch 12/100
218/218 [==============================] - 1s 6ms/step - loss: 248.1661 - 
mse: 248.1661 - mae: 11.7674 - val_loss: 236.5945 - val_mse: 236.5945 - va
l_mae: 11.4131
Epoch 13/100
218/218 [==============================] - 1s 6ms/step - loss: 249.1016 - 
mse: 249.1016 - mae: 11.7849 - val_loss: 232.3419 - val_mse: 232.3419 - va
l_mae: 11.4289
Epoch 14/100
218/218 [==============================] - 1s 6ms/step - loss: 246.7444 - 
mse: 246.7444 - mae: 11.7271 - val_loss: 235.4498 - val_mse: 235.4498 - va
l_mae: 11.3565
Epoch 15/100
218/218 [==============================] - 1s 6ms/step - loss: 248.7036 - 
mse: 248.7036 - mae: 11.7764 - val_loss: 235.7062 - val_mse: 235.7062 - va
l_mae: 11.7896
```

```
Epoch 16/100
218/218 [==============================] - 1s 6ms/step - loss: 246.8645 -
mse: 246.8645 - mae: 11.7289 - val_loss: 232.6479 - val_mse: 232.6479 - va
l_mae: 11.4417
Epoch 17/100
218/218 [==============================] - 1s 6ms/step - loss: 247.2877 -
mse: 247.2877 - mae: 11.7330 - val_loss: 234.4858 - val_mse: 234.4858 - va
l_mae: 11.2908
Epoch 18/100
218/218 [==============================] - 1s 6ms/step - loss: 246.2801 -
mse: 246.2801 - mae: 11.7191 - val_loss: 242.7359 - val_mse: 242.7359 - va
l_mae: 11.2782
Epoch 19/100
218/218 [==============================] - 1s 6ms/step - loss: 246.8304 -
mse: 246.8304 - mae: 11.7269 - val_loss: 232.7724 - val_mse: 232.7724 - va
l_mae: 11.5777
Epoch 20/100
218/218 [==============================] - 1s 6ms/step - loss: 246.3161 -
mse: 246.3161 - mae: 11.7166 - val_loss: 245.6716 - val_mse: 245.6716 - va
l_mae: 11.3218
Epoch 21/100
218/218 [==============================] - 1s 6ms/step - loss: 244.6115 -
mse: 244.6115 - mae: 11.6649 - val_loss: 233.3516 - val_mse: 233.3516 - va
l_mae: 11.6610
Epoch 22/100
218/218 [==============================] - 1s 6ms/step - loss: 245.7443 -
mse: 245.7443 - mae: 11.7023 - val_loss: 239.7674 - val_mse: 239.7674 - va
l_mae: 11.3305
Epoch 23/100
218/218 [==============================] - 1s 6ms/step - loss: 245.8556 -
mse: 245.8556 - mae: 11.7046 - val_loss: 230.1730 - val_mse: 230.1730 - va
l_mae: 11.3478
Epoch 24/100
218/218 [==============================] - 1s 6ms/step - loss: 243.7188 -
mse: 243.7188 - mae: 11.6422 - val_loss: 233.0534 - val_mse: 233.0534 - va
l_mae: 11.5402
Epoch 25/100
218/218 [==============================] - 1s 6ms/step - loss: 243.1557 -
mse: 243.1557 - mae: 11.6322 - val_loss: 231.5680 - val_mse: 231.5680 - va
l_mae: 11.2464
Epoch 26/100
218/218 [==============================] - 1s 6ms/step - loss: 242.8211 -
mse: 242.8211 - mae: 11.6253 - val_loss: 229.4078 - val_mse: 229.4078 - va
l_mae: 11.2712
Epoch 27/100
218/218 [==============================] - 1s 6ms/step - loss: 241.6351 -
mse: 241.6351 - mae: 11.5996 - val_loss: 227.6474 - val_mse: 227.6474 - va
l_mae: 11.3730
Epoch 28/100
218/218 [==============================] - 1s 6ms/step - loss: 241.2375 -
mse: 241.2375 - mae: 11.5898 - val_loss: 236.1183 - val_mse: 236.1183 - va
l_mae: 11.1826
Epoch 29/100
218/218 [==============================] - 1s 6ms/step - loss: 241.8967 -
mse: 241.8967 - mae: 11.6088 - val_loss: 229.5732 - val_mse: 229.5732 - va
l_mae: 11.3020
Epoch 30/100
218/218 [==============================] - 1s 6ms/step - loss: 240.6681 -
mse: 240.6681 - mae: 11.5722 - val_loss: 227.6795 - val_mse: 227.6795 - va
l_mae: 11.3637
```

```
Epoch 31/100
218/218 [==============================] - 1s 7ms/step - loss: 240.3548 -
mse: 240.3548 - mae: 11.5706 - val_loss: 228.7684 - val_mse: 228.7684 - va
l_mae: 11.1120
Epoch 32/100
218/218 [==============================] - 1s 7ms/step - loss: 241.3904 -
mse: 241.3904 - mae: 11.5963 - val_loss: 228.0444 - val_mse: 228.0444 - va
l_mae: 11.3784
Epoch 33/100
218/218 [==============================] - 1s 6ms/step - loss: 239.5993 -
mse: 239.5993 - mae: 11.5508 - val_loss: 229.6080 - val_mse: 229.6080 - va
l_mae: 11.5130
Epoch 34/100
218/218 [==============================] - 1s 6ms/step - loss: 240.3261 -
mse: 240.3261 - mae: 11.5710 - val_loss: 226.8333 - val_mse: 226.8333 - va
l_mae: 11.2309
Epoch 35/100
218/218 [==============================] - 1s 6ms/step - loss: 238.9737 -
mse: 238.9737 - mae: 11.5435 - val_loss: 230.5467 - val_mse: 230.5467 - va
l_mae: 11.5362
Epoch 36/100
218/218 [==============================] - 1s 7ms/step - loss: 239.5579 -
mse: 239.5579 - mae: 11.5460 - val_loss: 227.6739 - val_mse: 227.6739 - va
l_mae: 11.4528
Epoch 37/100
218/218 [==============================] - 1s 6ms/step - loss: 238.6880 -
mse: 238.6880 - mae: 11.5263 - val_loss: 227.6329 - val_mse: 227.6329 - va
l_mae: 11.1872
Epoch 38/100
218/218 [==============================] - 1s 6ms/step - loss: 239.8139 -
mse: 239.8139 - mae: 11.5517 - val_loss: 227.9006 - val_mse: 227.9006 - va
l_mae: 11.4549
Epoch 39/100
218/218 [==============================] - 1s 7ms/step - loss: 238.1614 -
mse: 238.1614 - mae: 11.5023 - val_loss: 226.6006 - val_mse: 226.6006 - va
l_mae: 11.2634
Epoch 40/100
218/218 [==============================] - 1s 6ms/step - loss: 238.0311 -
mse: 238.0311 - mae: 11.5152 - val_loss: 231.9979 - val_mse: 231.9979 - va
l_mae: 11.1634
Epoch 41/100
218/218 [==============================] - 1s 6ms/step - loss: 238.1833 -
mse: 238.1833 - mae: 11.5078 - val_loss: 227.0870 - val_mse: 227.0870 - va
l_mae: 11.2056
Epoch 42/100
218/218 [==============================] - 1s 7ms/step - loss: 238.8083 -
mse: 238.8083 - mae: 11.5229 - val_loss: 226.7289 - val_mse: 226.7289 - va
l_mae: 11.1756
Epoch 43/100
218/218 [==============================] - 2s 7ms/step - loss: 237.4770 -
mse: 237.4770 - mae: 11.4932 - val_loss: 226.2470 - val_mse: 226.2470 - va
l_mae: 11.2757
Epoch 44/100
218/218 [==============================] - 1s 7ms/step - loss: 237.7442 -
mse: 237.7442 - mae: 11.4975 - val_loss: 227.3918 - val_mse: 227.3918 - va
l_mae: 11.4306
Epoch 45/100
218/218 [==============================] - 1s 7ms/step - loss: 238.1587 -
mse: 238.1587 - mae: 11.5181 - val_loss: 226.8855 - val_mse: 226.8855 - va
l_mae: 11.2932
```

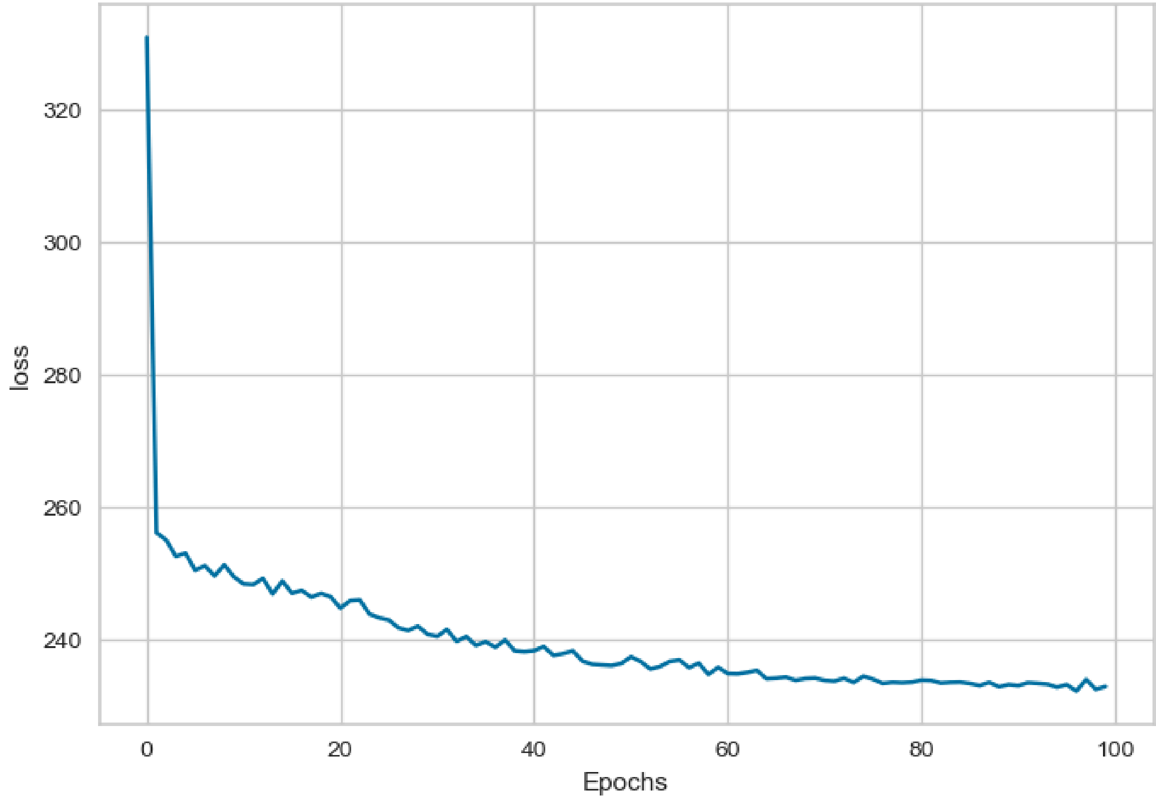
```
Epoch 46/100
218/218 [==============================] - 1s 7ms/step - loss: 236.6061 -
mse: 236.6061 - mae: 11.4785 - val_loss: 227.3983 - val_mse: 227.3983 - va
l_mae: 11.4053
Epoch 47/100
218/218 [==============================] - 1s 7ms/step - loss: 236.1376 -
mse: 236.1376 - mae: 11.4663 - val_loss: 226.9817 - val_mse: 226.9817 - va
l_mae: 11.3570
Epoch 48/100
218/218 [==============================] - 1s 7ms/step - loss: 236.0354 -
mse: 236.0354 - mae: 11.4638 - val_loss: 228.2687 - val_mse: 228.2687 - va
l_mae: 11.5502
Epoch 49/100
218/218 [==============================] - 2s 7ms/step - loss: 235.9444 -
mse: 235.9444 - mae: 11.4700 - val_loss: 227.8789 - val_mse: 227.8789 - va
l_mae: 11.3798
Epoch 50/100
218/218 [==============================] - 1s 7ms/step - loss: 236.2669 -
mse: 236.2669 - mae: 11.4619 - val_loss: 225.4629 - val_mse: 225.4629 - va
l_mae: 11.2555
Epoch 51/100
218/218 [==============================] - 2s 7ms/step - loss: 237.2558 -
mse: 237.2558 - mae: 11.4845 - val_loss: 229.0099 - val_mse: 229.0099 - va
l_mae: 11.5717
Epoch 52/100
218/218 [==============================] - 1s 7ms/step - loss: 236.5538 -
mse: 236.5538 - mae: 11.4747 - val_loss: 228.8131 - val_mse: 228.8131 - va
l_mae: 11.4336
Epoch 53/100
218/218 [==============================] - 2s 7ms/step - loss: 235.4145 -
mse: 235.4145 - mae: 11.4452 - val_loss: 226.5340 - val_mse: 226.5340 - va
l_mae: 11.3034
Epoch 54/100
218/218 [==============================] - 2s 8ms/step - loss: 235.7977 -
mse: 235.7977 - mae: 11.4461 - val_loss: 225.9993 - val_mse: 225.9993 - va
l_mae: 11.3024
Epoch 55/100
218/218 [==============================] - 1s 7ms/step - loss: 236.5634 -
mse: 236.5634 - mae: 11.4716 - val_loss: 225.9704 - val_mse: 225.9704 - va
l_mae: 11.2437
Epoch 56/100
218/218 [==============================] - 2s 7ms/step - loss: 236.7832 -
mse: 236.7832 - mae: 11.4791 - val_loss: 236.0784 - val_mse: 236.0784 - va
l_mae: 11.9693
Epoch 57/100
218/218 [==============================] - 2s 7ms/step - loss: 235.6292 -
mse: 235.6292 - mae: 11.4526 - val_loss: 233.2613 - val_mse: 233.2613 - va
l_mae: 11.1797
Epoch 58/100
218/218 [==============================] - 1s 7ms/step - loss: 236.3059 -
mse: 236.3059 - mae: 11.4559 - val_loss: 225.3824 - val_mse: 225.3824 - va
l_mae: 11.2280
Epoch 59/100
218/218 [==============================] - 1s 7ms/step - loss: 234.5927 -
mse: 234.5927 - mae: 11.4124 - val_loss: 228.6740 - val_mse: 228.6740 - va
l_mae: 11.5226
Epoch 60/100
218/218 [==============================] - 1s 7ms/step - loss: 235.6754 -
mse: 235.6754 - mae: 11.4490 - val_loss: 228.2245 - val_mse: 228.2245 - va
l_mae: 11.4878
```

```
Epoch 61/100
218/218 [==============================] - 1s 7ms/step - loss: 234.7257 -
mse: 234.7257 - mae: 11.4253 - val_loss: 226.7323 - val_mse: 226.7323 - va
l_mae: 11.3891
Epoch 62/100
218/218 [==============================] - 1s 7ms/step - loss: 234.6803 -
mse: 234.6803 - mae: 11.4278 - val_loss: 225.8693 - val_mse: 225.8693 - va
l_mae: 11.3658
Epoch 63/100
218/218 [==============================] - 1s 7ms/step - loss: 234.9016 -
mse: 234.9016 - mae: 11.4288 - val_loss: 226.0366 - val_mse: 226.0366 - va
l_mae: 11.2352
Epoch 64/100
218/218 [==============================] - 2s 8ms/step - loss: 235.2130 -
mse: 235.2130 - mae: 11.4335 - val_loss: 227.5322 - val_mse: 227.5322 - va
l_mae: 11.3410
Epoch 65/100
218/218 [==============================] - 1s 7ms/step - loss: 233.9445 -
mse: 233.9445 - mae: 11.4052 - val_loss: 227.4279 - val_mse: 227.4279 - va
l_mae: 11.3508
Epoch 66/100
218/218 [==============================] - 2s 7ms/step - loss: 234.0288 -
mse: 234.0288 - mae: 11.3973 - val_loss: 226.4180 - val_mse: 226.4180 - va
l_mae: 11.2688
Epoch 67/100
218/218 [==============================] - 2s 7ms/step - loss: 234.1876 -
mse: 234.1876 - mae: 11.4055 - val_loss: 227.9237 - val_mse: 227.9237 - va
l_mae: 11.1929
Epoch 68/100
218/218 [==============================] - 2s 7ms/step - loss: 233.6970 -
mse: 233.6970 - mae: 11.4040 - val_loss: 229.8421 - val_mse: 229.8421 - va
l_mae: 11.6144
Epoch 69/100
218/218 [==============================] - 2s 7ms/step - loss: 233.9760 -
mse: 233.9760 - mae: 11.4171 - val_loss: 226.1088 - val_mse: 226.1088 - va
l_mae: 11.4199
Epoch 70/100
218/218 [==============================] - 2s 7ms/step - loss: 234.0406 -
mse: 234.0406 - mae: 11.4055 - val_loss: 226.7268 - val_mse: 226.7268 - va
l_mae: 11.2948
Epoch 71/100
218/218 [==============================] - 2s 7ms/step - loss: 233.6917 -
mse: 233.6917 - mae: 11.3995 - val_loss: 225.9536 - val_mse: 225.9536 - va
l_mae: 11.3382
Epoch 72/100
218/218 [==============================] - 2s 7ms/step - loss: 233.5790 -
mse: 233.5790 - mae: 11.4014 - val_loss: 225.9818 - val_mse: 225.9818 - va
l_mae: 11.4455
Epoch 73/100
218/218 [==============================] - 2s 7ms/step - loss: 234.0106 -
mse: 234.0106 - mae: 11.4104 - val_loss: 226.3755 - val_mse: 226.3755 - va
l_mae: 11.2100
Epoch 74/100
218/218 [==============================] - 2s 8ms/step - loss: 233.3776 -
mse: 233.3776 - mae: 11.3910 - val_loss: 227.8276 - val_mse: 227.8276 - va
l_mae: 11.4385
Epoch 75/100
218/218 [==============================] - 2s 7ms/step - loss: 234.3293 -
mse: 234.3293 - mae: 11.4144 - val_loss: 227.8163 - val_mse: 227.8163 - va
l_mae: 11.2191
```

```
Epoch 76/100
218/218 [==============================] - 2s 7ms/step - loss: 233.8711 -
mse: 233.8711 - mae: 11.4130 - val_loss: 226.3331 - val_mse: 226.3331 - va
l_mae: 11.2809
Epoch 77/100
218/218 [==============================] - 2s 8ms/step - loss: 233.2346 -
mse: 233.2346 - mae: 11.3876 - val_loss: 226.4231 - val_mse: 226.4231 - va
l_mae: 11.3366
Epoch 78/100
218/218 [==============================] - 2s 7ms/step - loss: 233.4007 -
mse: 233.4007 - mae: 11.3951 - val_loss: 227.5111 - val_mse: 227.5111 - va
l_mae: 11.4095
Epoch 79/100
218/218 [==============================] - 2s 7ms/step - loss: 233.3381 -
mse: 233.3381 - mae: 11.3966 - val_loss: 226.5710 - val_mse: 226.5710 - va
l_mae: 11.3147
Epoch 80/100
218/218 [==============================] - 2s 7ms/step - loss: 233.4202 -
mse: 233.4202 - mae: 11.3915 - val_loss: 229.1625 - val_mse: 229.1625 - va
l_mae: 11.2672
Epoch 81/100
218/218 [==============================] - 2s 7ms/step - loss: 233.7311 -
mse: 233.7311 - mae: 11.4048 - val_loss: 226.9092 - val_mse: 226.9092 - va
l_mae: 11.3215
Epoch 82/100
218/218 [==============================] - 2s 7ms/step - loss: 233.6722 -
mse: 233.6722 - mae: 11.3941 - val_loss: 225.4368 - val_mse: 225.4368 - va
l_mae: 11.1434
Epoch 83/100
218/218 [==============================] - 2s 7ms/step - loss: 233.2939 -
mse: 233.2939 - mae: 11.3896 - val_loss: 227.7446 - val_mse: 227.7446 - va
l_mae: 11.4467
Epoch 84/100
218/218 [==============================] - 2s 8ms/step - loss: 233.3907 -
mse: 233.3907 - mae: 11.4027 - val_loss: 228.0226 - val_mse: 228.0226 - va
l_mae: 11.3633
Epoch 85/100
218/218 [==============================] - 2s 7ms/step - loss: 233.4264 -
mse: 233.4264 - mae: 11.4030 - val_loss: 229.1705 - val_mse: 229.1705 - va
l_mae: 11.2727
Epoch 86/100
218/218 [==============================] - 2s 7ms/step - loss: 233.2217 -
mse: 233.2217 - mae: 11.3931 - val_loss: 226.1839 - val_mse: 226.1839 - va
l_mae: 11.2623
Epoch 87/100
218/218 [==============================] - 2s 7ms/step - loss: 232.9105 -
mse: 232.9105 - mae: 11.3718 - val_loss: 226.9418 - val_mse: 226.9418 - va
l_mae: 11.4284
Epoch 88/100
218/218 [==============================] - 2s 7ms/step - loss: 233.3931 -
mse: 233.3931 - mae: 11.3911 - val_loss: 225.4453 - val_mse: 225.4453 - va
l_mae: 11.2082
Epoch 89/100
218/218 [==============================] - 2s 7ms/step - loss: 232.7178 -
mse: 232.7178 - mae: 11.3798 - val_loss: 226.4494 - val_mse: 226.4494 - va
l_mae: 11.3181
Epoch 90/100
218/218 [==============================] - 2s 7ms/step - loss: 233.0582 -
mse: 233.0582 - mae: 11.3850 - val_loss: 227.7520 - val_mse: 227.7520 - va
l_mae: 11.3284
```

```
Epoch 91/100
218/218 [==============================] - 2s 7ms/step - loss: 232.8994 -
mse: 232.8994 - mae: 11.3900 - val_loss: 225.9706 - val_mse: 225.9706 - va
l_mae: 11.3090
Epoch 92/100
218/218 [==============================] - 2s 8ms/step - loss: 233.3410 -
mse: 233.3410 - mae: 11.3807 - val_loss: 224.7623 - val_mse: 224.7623 - va
l_mae: 11.2777
Epoch 93/100
218/218 [==============================] - 2s 7ms/step - loss: 233.2391 -
mse: 233.2391 - mae: 11.3852 - val_loss: 226.4630 - val_mse: 226.4630 - va
l_mae: 11.2134
Epoch 94/100
218/218 [==============================] - 2s 8ms/step - loss: 233.1083 -
mse: 233.1083 - mae: 11.3776 - val_loss: 225.8687 - val_mse: 225.8687 - va
l_mae: 11.2612
Epoch 95/100
218/218 [==============================] - 2s 7ms/step - loss: 232.6611 -
mse: 232.6611 - mae: 11.3795 - val_loss: 224.9061 - val_mse: 224.9061 - va
l_mae: 11.3174
Epoch 96/100
218/218 [==============================] - 2s 7ms/step - loss: 233.0502 -
mse: 233.0502 - mae: 11.3856 - val_loss: 228.0827 - val_mse: 228.0827 - va
l_mae: 11.4667
Epoch 97/100
218/218 [==============================] - 2s 7ms/step - loss: 232.0630 -
mse: 232.0630 - mae: 11.3684 - val_loss: 226.2964 - val_mse: 226.2964 - va
l_mae: 11.4502
Epoch 98/100
218/218 [==============================] - 2s 8ms/step - loss: 233.8294 -
mse: 233.8294 - mae: 11.4008 - val_loss: 226.8253 - val_mse: 226.8253 - va
l_mae: 11.4552
Epoch 99/100
218/218 [==============================] - 2s 8ms/step - loss: 232.3359 -
mse: 232.3359 - mae: 11.3795 - val_loss: 225.2274 - val_mse: 225.2274 - va
l_mae: 11.2905
Epoch 100/100
218/218 [==============================] - 2s 8ms/step - loss: 232.7648 -
mse: 232.7648 - mae: 11.3796 - val_loss: 227.5540 - val_mse: 227.5540 - va
l_mae: 11.2529
```

In [62]:
```python
def plot_history(history,key):
    plt.plot(history.history[key])
    plt.xlabel("Epochs")
    plt.ylabel(key)
    plt.show()
#plot the history
plot_history(history,'loss')
```



In [63]:
```
prediction = model.predict(X_test_scaled)
```

```
1090/1090 [==============================] - 1s 592us/step
```

In [64]:
```
eval = metrics_evals(y_test, prediction)
eval
```

Out[64]:
```
{'MSE': 236.70689147488216,
 'RMSE': 15.385281650814266,
 'MAE': 11.376727459875644,
 'R2': 0.2479625583757149}
```

In [65]:
```
prediction_train = model.predict(X_train_scaled)
```

```
4358/4358 [==============================] - 3s 603us/step
```

In [66]:
```
evaluation = metrics_evals(y_train,prediction_train)
evaluation
```

Out[66]:
```
{'MSE': 230.75216198642065,
 'RMSE': 15.190528693446474,
 'MAE': 11.286163156888627,
 'R2': 0.2583177275250448}
```

In [67]:
```
model.evaluate(X_test_scaled,y_test)
```

```
1090/1090 [==============================] - 1s 672us/step - loss: 236.707
0 - mse: 236.7070 - mae: 11.3767
```

Out[67]:
```
[236.70701599121094, 236.70701599121094, 11.376723289489746]
```

As the number of epochs increases model tends to perform better

Different Architecture -

In [68]:
```
model = Sequential()
model.add(Input(shape=(X_train.shape[1], )))

model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(256, activation='relu'))

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(1, activation='linear'))
```

In [69]:
```
adam=Adam(learning_rate=0.01)
model.compile(loss='mse',optimizer=adam,metrics=['mse','mae'])
history=model.fit(X_train_scaled,y_train,epochs=50,batch_size=512,verbose
```

```
Epoch 1/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..7646 - mae: 12.9144
273/273 [==============================] - 2s 6ms/step - loss: 302.4351 -
mse: 302.4351 - mae: 12.8917
Epoch 2/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..6545 - mae: 12.0688
273/273 [==============================] - 2s 6ms/step - loss: 260.5866 -
mse: 260.5866 - mae: 12.0681
Epoch 3/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..2566 - mae: 11.9647
273/273 [==============================] - 2s 6ms/step - loss: 256.1387 -
mse: 256.1387 - mae: 11.9629
Epoch 4/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..4629 - mae: 11.9074
273/273 [==============================] - 2s 6ms/step - loss: 254.4493 -
mse: 254.4493 - mae: 11.9066
Epoch 5/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..4129 - mae: 11.8493
273/273 [==============================] - 2s 6ms/step - loss: 251.1922 -
mse: 251.1922 - mae: 11.8436
Epoch 6/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..2170 - mae: 11.8520
273/273 [==============================] - 2s 7ms/step - loss: 252.2060 -
mse: 252.2060 - mae: 11.8548
Epoch 7/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..9304 - mae: 11.8056
273/273 [==============================] - 2s 6ms/step - loss: 250.1177 -
mse: 250.1177 - mae: 11.8081
Epoch 8/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..9741 - mae: 11.7489
273/273 [==============================] - 2s 6ms/step - loss: 247.8414 -
mse: 247.8414 - mae: 11.7476
Epoch 9/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..6253 - mae: 11.7627
273/273 [==============================] - 2s 6ms/step - loss: 248.4618 -
mse: 248.4618 - mae: 11.7600
Epoch 10/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..4218 - mae: 11.7297
273/273 [==============================] - 2s 6ms/step - loss: 247.3134 -
mse: 247.3134 - mae: 11.7290
Epoch 11/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..1705 - mae: 11.6783
273/273 [==============================] - 2s 6ms/step - loss: 245.2284 -
mse: 245.2284 - mae: 11.6811
Epoch 12/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..1259 - mae: 11.6705
273/273 [==============================] - 2s 6ms/step - loss: 245.1549 -
mse: 245.1549 - mae: 11.6699
```

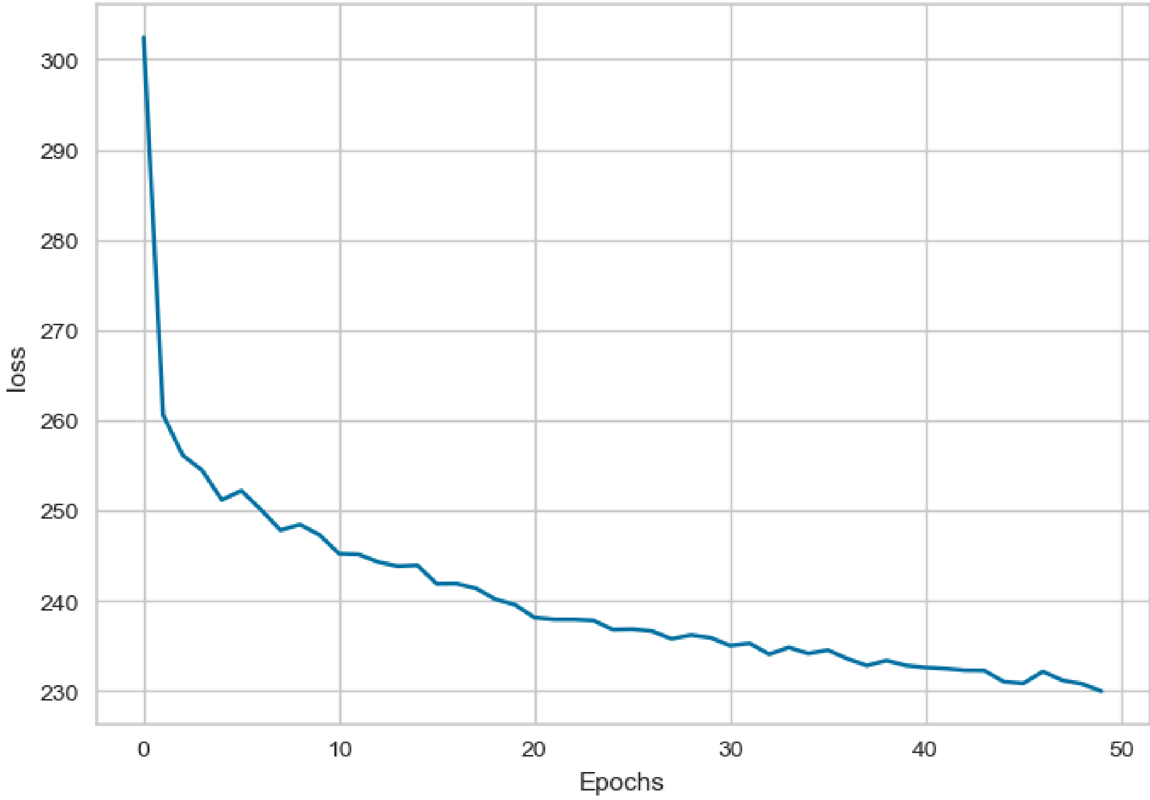
```
Epoch 13/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..2473 - mae: 11.6605
273/273 [==============================] - 2s 6ms/step - loss: 244.3138 -
mse: 244.3138 - mae: 11.6633
Epoch 14/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8183 - mae: 11.6452
273/273 [==============================] - 2s 6ms/step - loss: 243.8456 -
mse: 243.8456 - mae: 11.6464
Epoch 15/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8482 - mae: 11.6412
273/273 [==============================] - 2s 7ms/step - loss: 243.9519 -
mse: 243.9519 - mae: 11.6406
Epoch 16/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8028 - mae: 11.5987
273/273 [==============================] - 2s 6ms/step - loss: 241.8944 -
mse: 241.8944 - mae: 11.5930
Epoch 17/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8673 - mae: 11.5963
273/273 [==============================] - 2s 6ms/step - loss: 241.9223 -
mse: 241.9223 - mae: 11.5979
Epoch 18/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..6404 - mae: 11.6001
273/273 [==============================] - 2s 6ms/step - loss: 241.3987 -
mse: 241.3987 - mae: 11.5943
Epoch 19/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..1751 - mae: 11.5620
273/273 [==============================] - 2s 6ms/step - loss: 240.1914 -
mse: 240.1914 - mae: 11.5630
Epoch 20/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8192 - mae: 11.5494
273/273 [==============================] - 2s 6ms/step - loss: 239.5785 -
mse: 239.5785 - mae: 11.5470
Epoch 21/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..7492 - mae: 11.5022
273/273 [==============================] - 2s 6ms/step - loss: 238.1599 -
mse: 238.1599 - mae: 11.5080
Epoch 22/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..7389 - mae: 11.4938
273/273 [==============================] - 2s 6ms/step - loss: 237.9571 -
mse: 237.9571 - mae: 11.4979
Epoch 23/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..4191 - mae: 11.5018
273/273 [==============================] - 2s 7ms/step - loss: 237.9535 -
mse: 237.9535 - mae: 11.5054
Epoch 24/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8220 - mae: 11.5036
273/273 [==============================] - 2s 7ms/step - loss: 237.8464 -
mse: 237.8464 - mae: 11.5017
```

```
Epoch 25/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8295 - mae: 11.4740
273/273 [==============================] - 2s 7ms/step - loss: 236.8295 -
mse: 236.8295 - mae: 11.4740
Epoch 26/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..1037 - mae: 11.4689
273/273 [==============================] - 2s 7ms/step - loss: 236.8746 -
mse: 236.8746 - mae: 11.4677
Epoch 27/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..6631 - mae: 11.4728
273/273 [==============================] - 2s 7ms/step - loss: 236.6631 -
mse: 236.6631 - mae: 11.4728
Epoch 28/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8956 - mae: 11.4508
273/273 [==============================] - 2s 7ms/step - loss: 235.8080 -
mse: 235.8080 - mae: 11.4504
Epoch 29/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..1008 - mae: 11.4610
273/273 [==============================] - 2s 7ms/step - loss: 236.2276 -
mse: 236.2276 - mae: 11.4631
Epoch 30/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..9307 - mae: 11.4485
273/273 [==============================] - 2s 7ms/step - loss: 235.9307 -
mse: 235.9307 - mae: 11.4485
Epoch 31/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..0534 - mae: 11.4228
273/273 [==============================] - 2s 7ms/step - loss: 235.0534 -
mse: 235.0534 - mae: 11.4228
Epoch 32/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..3039 - mae: 11.4325
273/273 [==============================] - 2s 7ms/step - loss: 235.3039 -
mse: 235.3039 - mae: 11.4325
Epoch 33/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..1439 - mae: 11.3944
273/273 [==============================] - 2s 8ms/step - loss: 234.0998 -
mse: 234.0998 - mae: 11.3981
Epoch 34/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..7163 - mae: 11.4128
273/273 [==============================] - 2s 7ms/step - loss: 234.8505 -
mse: 234.8505 - mae: 11.4160
Epoch 35/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..1796 - mae: 11.4089
273/273 [==============================] - 2s 7ms/step - loss: 234.1796 -
mse: 234.1796 - mae: 11.4089
Epoch 36/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..5584 - mae: 11.4152
273/273 [==============================] - 2s 7ms/step - loss: 234.5584 -
mse: 234.5584 - mae: 11.4152
```

```
Epoch 37/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..6248 - mae: 11.3819
273/273 [==============================] - 2s 7ms/step - loss: 233.6028 -
mse: 233.6028 - mae: 11.3809
Epoch 38/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8120 - mae: 11.3804
273/273 [==============================] - 2s 7ms/step - loss: 232.8465 -
mse: 232.8465 - mae: 11.3807
Epoch 39/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..1131 - mae: 11.3775
273/273 [==============================] - 2s 7ms/step - loss: 233.4019 -
mse: 233.4019 - mae: 11.3794
Epoch 40/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..7923 - mae: 11.3665
273/273 [==============================] - 2s 7ms/step - loss: 232.8520 -
mse: 232.8520 - mae: 11.3696
Epoch 41/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..5788 - mae: 11.3619
273/273 [==============================] - 2s 9ms/step - loss: 232.6300 -
mse: 232.6300 - mae: 11.3631
Epoch 42/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..1792 - mae: 11.3503
273/273 [==============================] - 2s 7ms/step - loss: 232.5130 -
mse: 232.5130 - mae: 11.3556
Epoch 43/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..9996 - mae: 11.3606
273/273 [==============================] - 2s 7ms/step - loss: 232.3167 -
mse: 232.3167 - mae: 11.3646
Epoch 44/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8451 - mae: 11.3481
273/273 [==============================] - 2s 7ms/step - loss: 232.2940 -
mse: 232.2940 - mae: 11.3538
Epoch 45/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8720 - mae: 11.3226
273/273 [==============================] - 2s 7ms/step - loss: 231.0450 -
mse: 231.0450 - mae: 11.3268
Epoch 46/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..8952 - mae: 11.3238
273/273 [==============================] - 2s 7ms/step - loss: 230.8593 -
mse: 230.8593 - mae: 11.3225
Epoch 47/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..1708 - mae: 11.3556
273/273 [==============================] - 2s 7ms/step - loss: 232.1779 -
mse: 232.1779 - mae: 11.3558
Epoch 48/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..2004 - mae: 11.3270
273/273 [==============================] - 2s 8ms/step - loss: 231.2015 -
mse: 231.2015 - mae: 11.3292
```

```
Epoch 49/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..7257 - mae: 11.3227
273/273 [==============================] - 2s 9ms/step - loss: 230.8060 -
mse: 230.8060 - mae: 11.3230
Epoch 50/50
WARNING:tensorflow:Can save best model only with val_loss available, skipp
ing..4019 - mae: 11.2836
273/273 [==============================] - 2s 7ms/step - loss: 230.0005 -
mse: 230.0005 - mae: 11.2957
```

In [70]:
```
def plot_history(history,key):
    plt.plot(history.history[key])
    plt.xlabel("Epochs")
    plt.ylabel(key)
    plt.show()
#plot the history
plot_history(history,'loss')
```



In [71]:
```
prediction = model.predict(X_test_scaled)
```

```
1090/1090 [==============================] - 1s 622us/step
```

In [72]:
```
eval = metrics_evals(y_test, prediction)
eval
```

Out[72]:
```
{'MSE': 229.1251087983177,
 'RMSE': 15.136879097037067,
 'MAE': 11.183416929639728,
 'R2': 0.27205051125071555}
```

In [73]:
```
prediction_train = model.predict(X_train_scaled)
```

```
4358/4358 [==============================] - 3s 615us/step
```

In [74]:
```
evaluation = metrics_evals(y_train,prediction_train)
evaluation
```

Out[74]:
```
{'MSE': 221.1516653933716,
 'RMSE': 14.871168931639893,
 'MAE': 11.033151398899783,
 'R2': 0.28917558848168334}
```

In [75]:
```
model.evaluate(X_test_scaled,y_test)
```

```
1090/1090 [==============================] - 1s 698us/step - loss: 229.125
2 - mse: 229.1252 - mae: 11.1834
```

Out[75]:
```
[229.12518310546875, 229.12518310546875, 11.183415412902832]
```

The model isn't showing significant improvement, this seems to be the optimal underlying model

Leading Questions:

Q) Defining the problem statements and where can this and modifications of this be used?

Ans - The problem at hand involves estimating delivery times for orders accurately and efficiently. The objective is to minimize the time discrepancy between estimated and actual delivery times.

Delivery time estimation problem can be modified and applied to other use cases like * For Delivery Time Estimation in E-commerece * Cab Arrival Time Estimation for Taxi Services

Q) List 3 functions the pandas datetime provides with one line explanation.

1. pd.to_datetime(): Converts a variety of date and time formats (e.g., strings, timestamps) into pandas datetime objects.
2. pd.date_range(): Generates a sequence of dates and times, useful for creating time-series data with a specified frequency.
3. pd.Timestamp(): Creates a specific timestamp object from a string, integer, or datetime-like object for more precise handling of time data.

Q) Short note on datetime, timedelta, time span (period)

- Datetime
  - datetime represents points in time, combining both the date (year, month, day) and time (hour, minute, second)
- Timedelta

  - timedelta represents the difference between two datetime objects, typically expressed in days, seconds, and microseconds.
- Time Span (Period)
  - A Period represents a time span or interval of time, such as a specific month, quarter, or year.

Q) Why do we need to check for outliers in our data?

Checking for outliers is important because they can skew model results, distort statistical analyses, and reduce the accuracy of predictions. Identifying and handling them ensures more reliable, accurate insights and decisions.

Q) Name 3 outlier removal methods?

1. Z-Score Method
2. IQR (InterQuartile Range)
3. Visualization Methods (Boxplots)

Q) What classical machine learning methods can we use for this problem?

- Random Forest
- XGBoost
- Linear Regression

Q) Why is scaling required for neural networks?

1. Improves Convergence Speed
2. Prevents Bias Toward Larger Values
3. Helps Activation Functions Perform Better
4. Stabilizes Training

In summary, scaling is crucial for efficient and effective training, ensuring faster convergence, balanced feature influence, and stable learning.

Q) Briefly explain your choice of optimizer

We are using Adam Optimizer because :

- Adam is efficient for Regression Tasks, as it adapts learning rates for each parameter
- Works Well with Large Datasets
- Reduces Need for Hyperparameter Tuning
- Improves Convergence Speed

Q) Which activation function did you use and why?

In this problem we are using ReLu activation function because:

- Simplicity and Efficiency
- Prevents Vanishing Gradient Problem
- Avoid exploding gradients
- Sparse Activation

Q) Why does a neural network perform well on a large dataset?

Neural networks perform well on large datasets because they can learn complex patterns, generalize better, and reduce overfitting. With more data, the model has diverse examples to train on, improving stability, optimizing effectively, and making more accurate predictions.