House Price Prediction

In [1]:
```python
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.datasets
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn import metrics
```

importing the boston house price data

In [2]:
```python
data = pd.read_csv(r"C:\datasets\archive\HousingData.csv")
```

In [3]:
```python
data
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	NaN	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	NaN	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

In [4]:
```python
data.head()
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	NaN	36.2

In [5]:
```python
data.shape
```

Out[5]:
```
(506, 14)
```

In [6]:
```python
# checking for missing values
data.isnull().sum()
```

Out[6]:
```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

In [7]:
```python
# statistical measures of the dataset
data.describe()
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
count	486.000000	486.000000	486.000000	486.000000	506.000000	506.000000	486.000000	506.000000	506.000000	506.000000	506.000000	506.000000	486.000000	506.000000
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519	3.795043	9.549407	408.237154	18.455534	356.674032	12.715432	22.532806
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	27.999513	2.105710	8.707259	168.537116	2.164946	91.294864	7.155871	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	45.175000	2.100175	4.000000	279.000000	17.400000	375.377500	7.125000	17.025000
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	76.800000	3.207450	5.000000	330.000000	19.050000	391.440000	11.430000	21.200000
75%	3.560263	12.500000	18.100000	0.000000	0.624000	6.623500	93.975000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

understanding the correlation between various features in the dataset

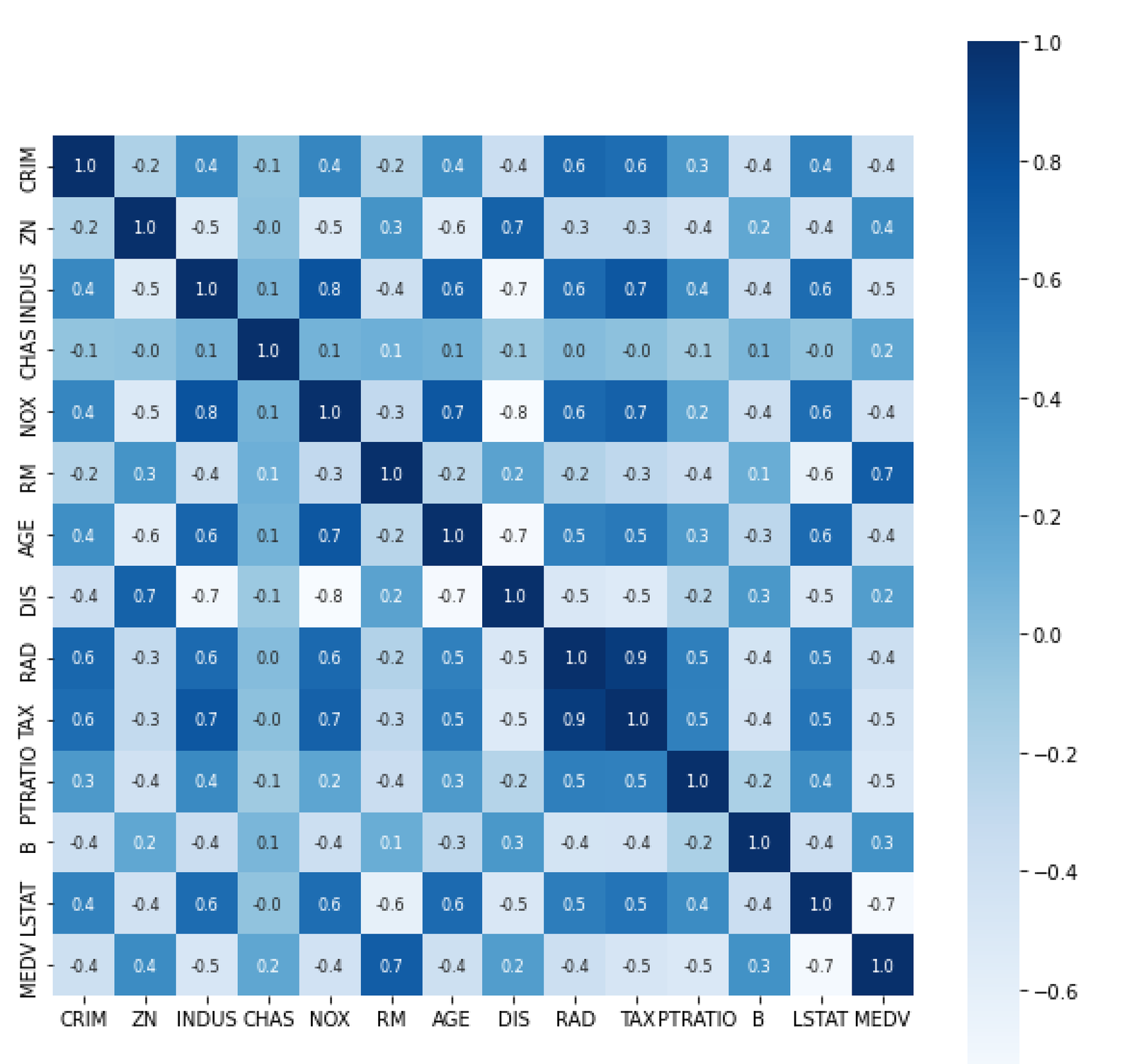
positive negative

negatve

In [8]:
```python
correlation=data.corr()
```

In [9]:
```python
# constructing heat map to understand corrrelation
plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot_kws={'size':8}, cmap='Blues')
```

Out[9]:
```
<AxesSubplot:>
```



splitting the data and target

In [10]:
```python
X=data.drop(['MEDV'], axis=1)
Y=data['MEDV']
```

In [11]:
```python
print(X)
print(Y)
```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  \
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93   0.0  0.573  6.030   NaN  2.5050    1  273

     PTRATIO       B  LSTAT
0       15.3  396.90   4.98
1       17.8  396.90   9.14
2       17.8  392.83   4.03
3       18.7  394.63   2.94
4       18.7  396.90    NaN
..       ...     ...    ...
501     21.0  391.99    NaN
502     21.0  396.90   9.08
503     21.0  396.90   5.64
504     21.0  393.45   6.48
505     21.0  396.90   7.88

[506 rows x 13 columns]
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

splitting the data into training data and test data

In [12]:
```python
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2,random_state=2)
```

In [13]:
```python
print(X.shape,X_train.shape,X_test.shape)
```

```
(506, 13) (404, 13) (102, 13)
```

model training

XG boost Regressor

In [14]:
```python
# loading the model
model= XGBRegressor()
```

In [15]:
```python
# training the model with X_train
model.fit(X_train,Y_train)
```

Out[15]:
```
XGBRegressor
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

evaluation

prediction on training data

In [16]:
```python
# accuracy for prediction on trai ning data
training_data_prediction = model.predict(X_train)
```

In [17]:
```python
training_data_prediction
```

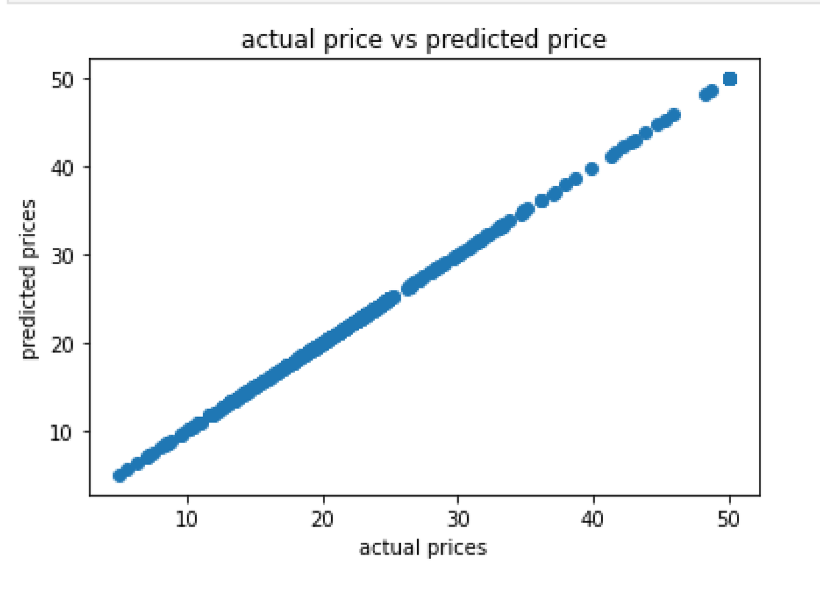
Out[17]:
```
array([23.128431 , 20.994154 , 20.101261 , 34.67768  , 13.903945 ,
       13.489747 , 21.99753  , 15.193501 , 10.899377 , 22.690107 ,
       13.813943 ,  5.614215 , 29.804506 , 49.987877 , 34.888184 ,
       20.615887 , 23.385252 , 19.219105 , 32.693237 , 19.607283 ,
       26.987926 ,  8.400735 , 46.006306 , 21.70009  , 27.08245  ,
       19.38461  , 19.299566 , 24.807394 , 22.600876 , 31.715944 ,
       18.551641 ,  8.707558 , 17.408579 , 23.699173 , 13.299077 ,
       10.500352 , 12.715582 , 25.006472 , 19.694242 , 14.893264 ,
       24.218592 , 24.984715 , 14.913808 , 17.005949 , 15.597038 ,
       12.702356 , 24.503147 , 15.003218 , 49.999443 , 17.526314 ,
       21.187368 , 32.001503 , 15.596226 , 22.899336 , 19.307703 ,
       18.713356 , 23.291584 , 37.196495 , 30.100304 , 33.104607 ,
       20.99445  , 49.98314  , 13.39885  ,  4.9928923, 16.48852  ,
        8.403917 , 28.697334 , 19.492647 , 20.58597  , 45.39965  ,
       39.802162 , 33.39493  , 19.806774 , 33.40387  , 25.294323 ,
       49.999493 , 12.515287 , 17.44132  , 18.611597 , 22.601511 ,
       50.002136 , 23.804588 , 23.334707 , 23.10605  , 41.709064 ,
       16.125729 , 31.617798 , 36.096508 ,  6.997423 , 20.380814 ,
       19.996199 , 12.029594 , 24.961348 , 49.988277 , 37.897827 ,
       23.098347 , 41.294544 , 17.60291  , 16.302446 , 30.032356 ,
       22.907495 , 19.810986 , 17.09506  , 18.901705 , 18.948738 ,
       22.578093 , 23.171654 , 33.21098  , 15.005704 , 11.70465  ,
       18.800985 , 20.792173 , 17.985819 , 19.648312 , 50.004223 ,
       17.206322 , 16.416636 , 17.507555 , 14.60386  , 33.103535 ,
       14.498311 , 43.815643 , 34.948956 , 20.40538  , 14.605213 ,
        8.092749 , 11.785861 , 11.831506 , 18.693441 ,  6.3084145,
       23.953945 , 13.0773945, 19.594473 , 49.998096 , 22.31341  ,
       18.905788 , 31.195295 , 20.697218 , 32.203373 , 36.18067  ,
       14.214334 , 15.695019 , 49.998615 , 20.405207 , 16.200977 ,
       13.408519 , 49.99794  , 31.600187 , 12.290271 , 19.216589 ,
       29.799414 , 31.501423 , 22.813274 , 10.191812 , 24.096865 ,
       23.714815 , 22.007042 , 13.803389 , 28.411673 , 33.18088  ,
       13.10673  , 18.995567 , 26.598248 , 36.971924 , 30.79777  ,
       22.779976 , 10.211277 , 22.200876 , 24.467127 , 36.19966  ,
       23.094261 , 20.11149  , 19.48739  , 10.796084 , 22.66935  ,
       19.488937 , 20.105448 ,  9.614282 , 42.789986 , 48.796795 ,
       13.078567 , 20.304855 , 24.783684 , 14.0974865, 21.697916 ,
       22.20561  , 32.999634 , 21.11631  , 25.009466 , 19.109894 ,
       32.405125 , 13.601782 , 15.092909 , 23.06247  , 27.497938 ,
       19.375496 , 26.495235 , 27.498268 , 28.697725 , 21.232346 ,
       18.695868 , 26.731031 , 14.007644 , 21.689535 , 18.388357 ,
       43.089417 , 29.0748   , 20.285393 , 23.710909 , 18.284605 ,
       17.209354 , 18.319569 , 24.40191  , 26.391329 , 19.077065 ,
       13.293503 , 22.174158 , 22.19947  ,  8.555694 , 18.90311  ,
       21.797913 , 19.330915 , 18.195646 ,  7.509877 , 22.402914 ,
       20.010864 , 14.403046 , 22.502153 , 28.514729 , 21.587112 ,
       13.803478 , 20.499187 , 21.898447 , 23.103111 , 50.006916 ,
       16.22731  , 30.301104 , 50.017963 , 17.784174 , 19.056034 ,
       10.387393 , 20.391016 , 16.50506  , 17.192429 , 16.702799 ,
       19.511196 , 30.51736  , 28.99166  , 19.55188  , 23.183167 ,
       24.382183 ,  9.504991 , 23.899569 , 49.989056 , 21.17416  ,
       22.604053 , 19.994152 , 13.396168 , 19.984293 , 17.110525 ,
       12.7490635, 22.997908 , 15.223642 , 20.594662 , 26.237635 ,
       18.111963 , 24.099932 , 14.086146 , 21.697147 , 20.083914 ,
       25.014418 , 27.89823  , 22.931677 , 18.497055 , 22.178623 ,
       24.003244 , 14.795677 , 19.887085 , 24.404215 , 17.7806   ,
       24.589611 , 31.975996 , 17.80095  , 23.331669 , 16.110304 ,
       13.005892 , 10.997909 , 24.29056  , 15.575491 , 35.209496 ,
       19.619333 , 42.29822  ,  8.792996 , 24.402912 , 14.126401 ,
       15.379655 , 17.305126 , 22.120369 , 23.094246 , 44.790134 ,
       17.80082  , 31.505554 , 22.814024 , 16.8487   , 23.912342 ,
       12.096439 , 38.687733 , 21.384914 , 16.006336 , 23.926025 ,
       11.9002285, 24.975077 ,  7.1953726, 24.699255 , 18.193438 ,
       22.484354 , 23.043955 , 24.287437 , 17.10062  , 17.798908 ,
       13.511288 , 27.066021 , 13.304795 , 21.904535 , 20.021526 ,
       15.383979 , 16.599194 , 22.294048 , 24.703049 , 21.40998  ,
       22.914837 , 29.597427 , 21.887817 , 19.887808 , 29.605515 ,
       23.405313 , 13.791948 , 24.459793 , 11.904582 ,  7.2066965,
       20.496056 ,  9.691774 , 48.30093  , 25.18632  , 11.695794 ,
       17.403475 , 14.494502 , 28.586044 , 19.38797  , 22.472443 ,
        7.019776 , 20.598795 , 22.977278 , 19.691525 , 23.683409 ,
       25.019066 , 27.948833 , 13.3966675, 14.509144 , 20.314003 ,
       19.30595  , 24.096642 , 14.891865 , 26.387436 , 33.29164  ,
       23.610031 , 24.587814 , 18.49297  , 20.89287  , 10.400038 ,
       23.286556 , 13.100235 , 24.700619 , 22.607405 , 20.509108 ,
       16.808247 , 10.204005 , 33.829002 , 18.600655 , 50.000328 ,
       23.78203  , 23.904655 , 21.202484 , 18.794209 ,  8.508018 ,
       21.519442 , 23.20152  , 21.031935 , 16.602638 , 28.08534  ,
       21.196228 , 28.3923   , 14.283643 , 49.998447 , 31.000347 ,
       24.993149 , 21.438381 , 18.98385  , 28.999262 , 15.206557 ,
       22.78243  , 21.792433 , 19.915585 , 23.790918 ], dtype=float32)
```

In [19]:
```python
# r squared error
score_1= metrics.r2_score(Y_train,training_data_prediction)

#mean absolute error

score_2=metrics.mean_absolute_error(Y_train,training_data_prediction)

print("R squared error: ",score_1)
print("Mean Absolute error:",score_2)
```

```
R squared error:  0.9999970506674762
Mean Absolute error: 0.011185854968458139
```

visualizing the actual prices and the prediced prices

In [25]:
```python
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("actual prices")
plt.ylabel("predicted prices")
plt.title("actual price vs predicted price")
plt.show()
```



prediction on test data

In [26]:
```python
# accuracy for prediction on testdata
test_data_predictions= model.predict(X_test)
```

In [27]:
```python
test_data_predictions
```

Out[27]:
```
array([19.683561 , 21.546743 , 32.775703 , 29.274933 ,  7.4516616,
       16.32632  , 23.555683 , 29.930449 , 26.473228 , 20.907225 ,
       26.906319 , 24.167038 , 20.561563 , 22.171478 , 35.403473 ,
       22.177969 , 18.582481 ,  8.858419 ,  9.934703 , 15.65133  ,
       22.326714 , 20.802525 , 36.469894 , 18.383966 , 13.91361  ,
       18.060265 , 48.05177  , 31.959373 , 32.904366 , 21.638412 ,
       15.139871 , 20.102669 , 31.079195 , 24.137821 , 10.84314  ,
       17.316162 ,  6.1782107, 20.801908 , 23.012804 , 21.472626 ,
       26.358389 , 13.326545 , 31.690912 ,  7.0663924, 19.263102 ,
       12.758626 , 38.305126 , 15.226805 , 32.26505  , 13.661363 ,
       30.082272 , 29.561243 ,  7.07074  , 36.405067 , 31.714586 ,
       19.428785 , 20.51014  , 20.352552 , 15.988234 , 21.70389  ,
       19.927158 , 15.8155775, 18.836674 , 30.913027 , 34.26503  ,
       25.256977 , 49.395504 , 26.036608 ,  9.516477 , 22.093874 ,
       16.411308 ,  6.7952156, 17.288977 , 17.47211  , 25.500889 ,
       24.596329 , 21.39696  , 21.36873  , 19.810547 , 23.586697 ,
       34.50355  , 17.120983 , 19.01802  , 31.073286 , 38.10575  ,
       36.67638  , 17.11905  , 24.829042 , 29.10352  , 17.883942 ,
       21.025583 , 20.179815 , 11.981259 , 33.944946 , 37.704678 ,
       10.878566 , 38.495247 , 33.655293 , 21.673916 , 17.088932 ,
       28.578854 , 23.037731 ], dtype=float32)
```

In [29]:
```python
# r squared error
score_1= metrics.r2_score(Y_test,test_data_predictions)

#mean absolute error

score_2=metrics.mean_absolute_error(Y_test,test_data_predictions)

print("R squared error: ",score_1)
print("Mean Absolute error:",score_2)
```

```
R squared error:  0.8361097215940602
Mean Absolute error: 2.466887313244389
```

In [30]:
```python
'--allow-chromium-download'
```

Out[30]:
```
'--allow-chromium-download'
```

In [ ]: