In [ ]:
```
'''
Create a Linear Regression Model using Python/R to predict home prices using Boston
Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset conta
information about various houses in Boston through different parameters. There are
and 14 feature variables in this dataset.
The objective is to predict the value of prices of the house using the given featur
'''
```

In [19]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error

from sklearn.datasets import load_boston
boston = load_boston()
```

In [3]:
```
boston.data.shape
```

Out[3]:
```
(506, 13)
```

In [4]:
```
data.describe()
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

In [33]:
```
x = boston.data

y = boston.target

from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2, random_state =

print("xtrain shape : ", xtrain.shape)
print("xtest shape : ", xtest.shape)
print("ytrain shape : ", ytrain.shape)
print("ytest shape : ", ytest.shape)
```

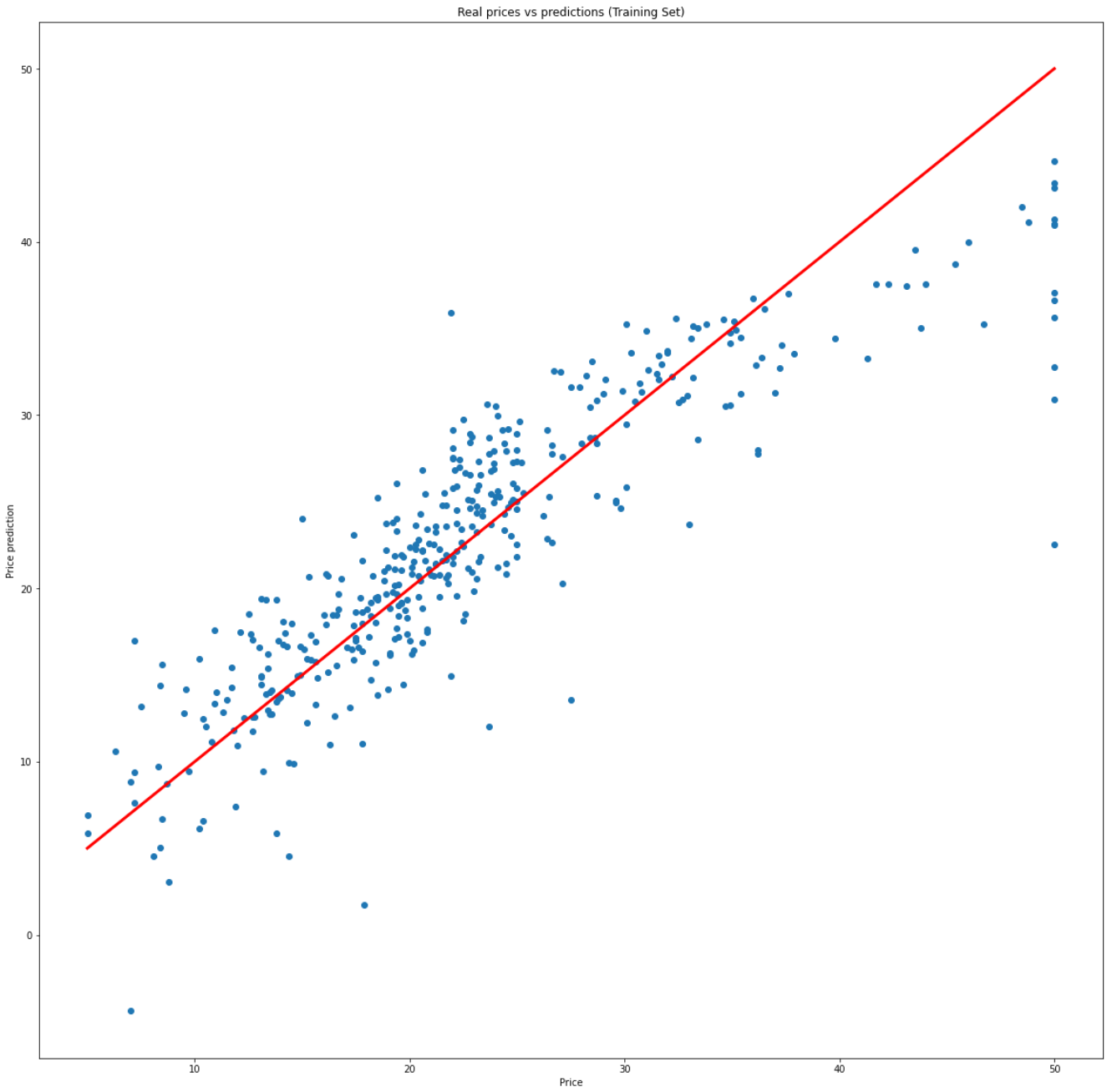
```
xtrain shape :  (404, 13)
xtest shape :  (102, 13)
ytrain shape :  (404,)
ytest shape :  (102,)
```

In [17]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain, ytrain)

y_pred = regressor.predict(xtest)

print('predicted response:', y_pred)
```

```
predicted response: [24.88963777 23.72141085 29.36499868 12.12238621 21.44382254 19.
2834443
 20.49647539 21.36099298 18.8967118  19.9280658   5.12703513 16.3867396
 17.07776485  5.59375659 39.99636726 32.49654668 22.45798809 36.85192327
 30.86401089 23.15140009 24.77495789 24.67187756 20.59543752 30.35369168
 22.41940736 10.23266565 17.64816865 18.27419652 35.53362541 20.96084724
 18.30413012 17.79262072 19.96561663 24.06127231 29.10204874 19.27774123
 11.15536648 24.57560579 17.5862644  15.49454112 26.20577527 20.86304693
 22.31460516 15.60710156 23.00363104 25.17247952 20.11459464 22.90256276
 10.0380507  24.28515123 20.94127711 17.35258791 24.52235405 29.95143046
 13.42695877 21.72673066 20.7897053  15.49668805 13.98982601 22.18377874
 17.73047814 21.58869165 32.90522136 31.11235671 17.73252635 32.76358681
 18.7124637  19.78693475 19.02958927 22.89825374 22.96041622 24.02555703
 30.72859326 28.83142691 25.89957059  5.23251817 36.72183202 23.77267249
 27.26856352 19.29492159 28.62304496 19.17978838 18.97185995 37.82397662
 39.22012647 23.71261106 24.93076217 15.88545417 26.09845751 16.68819641
 15.83515991 13.10775597 24.71583588 31.25165267 22.16640989 20.25087212
  0.59025319 25.44217132 15.57178328 17.93719475 25.30588844 22.3732326 ]
```

In [13]:
```
plt.figure(figsize=(20,20))
plt.scatter(ytrain, regressor.predict(xtrain))
plt.plot([ytrain.min(),ytrain.max()],[ytrain.min(),ytrain.max()], color='red', linew
plt.xlabel("Price")
plt.ylabel("Price prediction")
plt.title("Real prices vs predictions (Training Set)")
```

Out[13]:
```
Text(0.5, 1.0, 'Real prices vs predictions (Training Set)')
```

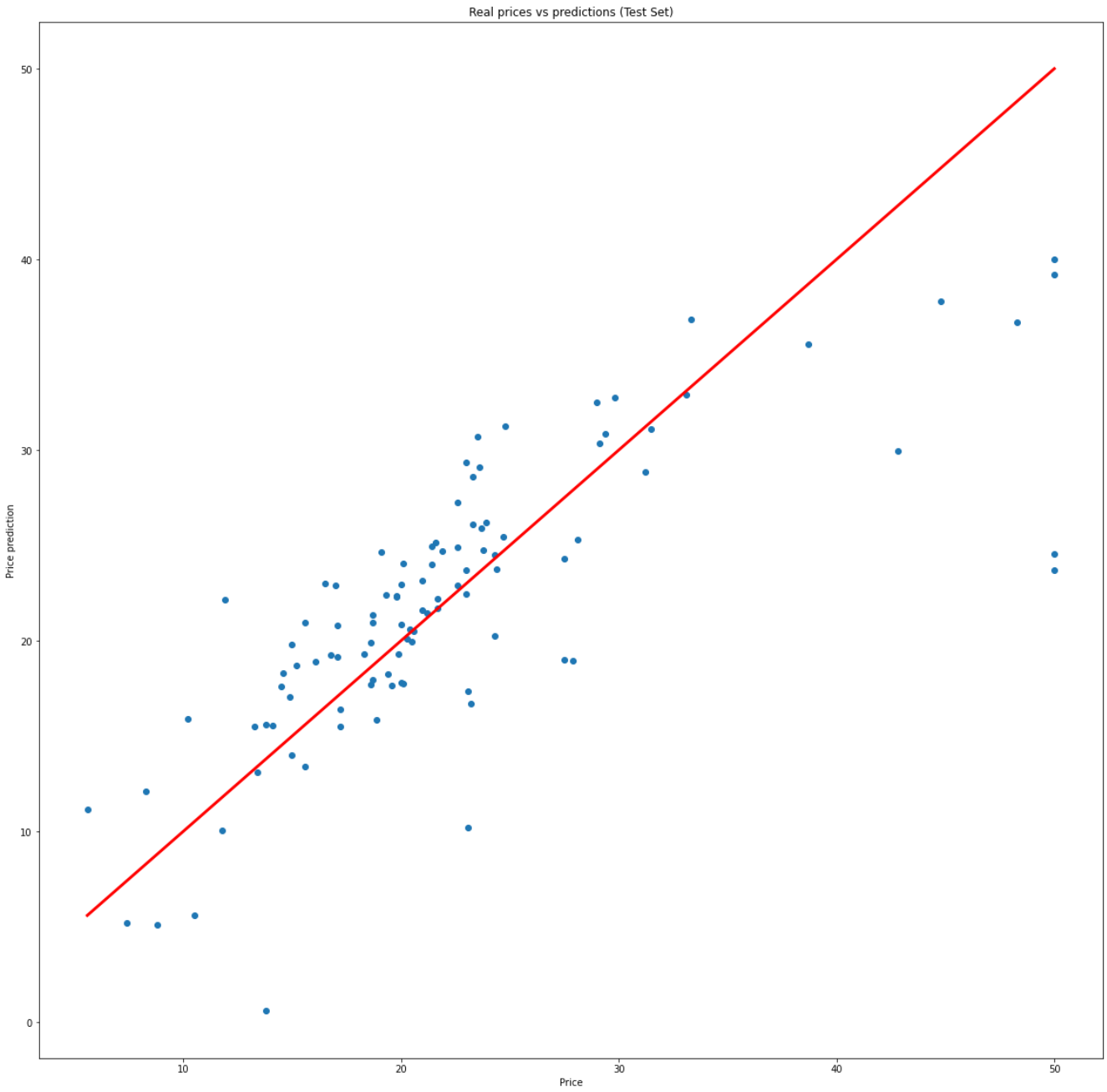


In [12]:

```
plt.figure(figsize=(20,20))
plt.scatter(ytest, y_pred)
plt.plot([ytest.min(),ytest.max()],[ytest.min(),ytest.max()], color='red', linewidth
plt.xlabel("Price")
plt.ylabel("Price prediction")
plt.title("Real prices vs predictions (Test Set)")
```

Out[12]:

```
Text(0.5, 1.0, 'Real prices vs predictions (Test Set)')
```



In [22]:
```
print('Mean Square Error: ', mean_squared_error(ytest, y_pred))
```

```
Mean Square Error:  33.448979997676474
```

In [23]:
```
print('Sqaure Root of Mean Square Error: ', np.sqrt(mean_squared_error(ytest, y_pred
```

```
Sqaure Root of Mean Square Error:  5.78350931508513
```

In [14]:
```
print("Accuracy:", regressor.score(xtest, ytest))
```

```
Accuracy: 0.5892223849182514
```