Adnaan Shaikh TYA164

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
b_data = pd.read_csv('/content/Boston.csv')
b_data
```

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptra
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	
...	...	...	...	...	...	...	...	...	...	...	...	
501	502	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	
502	503	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	
503	504	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	
504	505	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	
505	506	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	

```
b_data.columns
```

```
Index(['Unnamed: 0', 'crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis',
       'rad', 'tax', 'ptratio', 'black', 'lstat', 'medv'],
      dtype='object')
```

```
b_data.drop('Unnamed: 0', axis=1, inplace=True)
```

```
b_data
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	blac
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.9
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.8
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.6
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9
...	...	...	...	...	...	...	...	...	...	...	...	.
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.9
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.9
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.4

```
b_data.columns = ['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'black', 'lstat', 'prices']
b_data
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	blac
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.9
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.8
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.6
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9
...	...	...	...	...	...	...	...	...	...	...	...	.
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.9
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.9
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.4
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.9

506 rows × 14 columns

```
b_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 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    int64
 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    int64
 9   tax      506 non-null    int64
 10  ptratio  506 non-null    float64
 11  black    506 non-null    float64
 12  lstat    506 non-null    float64
 13  prices   506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

```
b_data.isnull().sum()
```

```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
black      0
lstat      0
prices     0
dtype: int64
```

```
x = b_data[['crim','zn','indus','chas', 'nox', 'rm', 'age', 'dis', 'rad','tax','ptratio','bla
x = x.sum()
print(x)
```

```
crim         0
zn         372
indus        0
chas       471
nox          0
rm           0
age          0
dis          0
rad          0
tax          0
ptratio      0
black        0
lstat        0
```

```
prices       0
dtype: int64
```

```
boston_data= b_data.drop(columns=['zn', 'chas'])
boston_data
```

	crim	indus	nox	rm	age	dis	rad	tax	ptratio	black	lstat	pr
0	0.00632	2.31	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	
1	0.02731	7.07	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	
2	0.02729	7.07	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	
3	0.03237	2.18	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	
4	0.06905	2.18	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	
...	...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	11.93	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	
502	0.04527	11.93	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	
503	0.06076	11.93	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	
504	0.10959	11.93	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	
505	0.04741	11.93	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	

506 rows × 12 columns

```
from sklearn import preprocessing
```

```
b_data
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	blac
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.9

```
std = preprocessing.StandardScaler()
```

3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.6

```
y = b_data['prices']
bos = b_data.drop(['prices'], axis=1)
bos = std.fit_transform(bos)
print(bos[0])
```

```
[-0.41978194  0.28482986 -1.2879095  -0.27259857 -0.14421743  0.41367189
 -0.12001342  0.1402136  -0.98284286 -0.66660821 -1.45900038  0.44105193
 -1.0755623 ]
```

504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.4

```
boston_data = pd.DataFrame(bos)
boston_data['prices'] =y
boston_data.columns = ['crim', 'indus', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'black', 'lstat', 'prices']
```

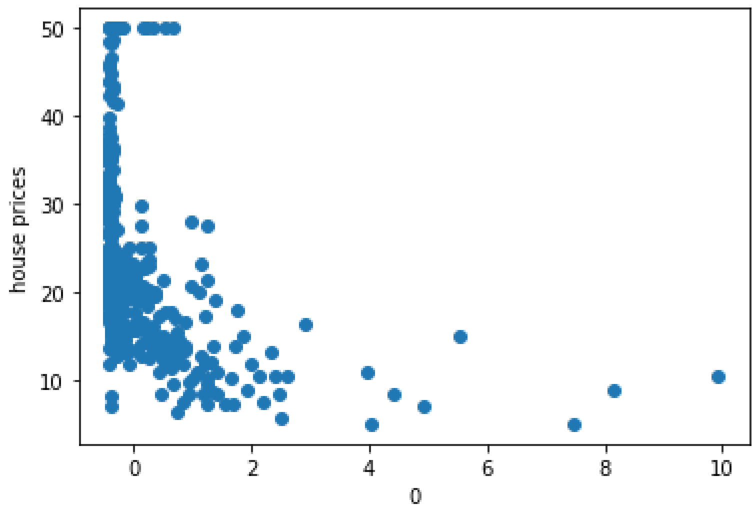
```
b_data
```

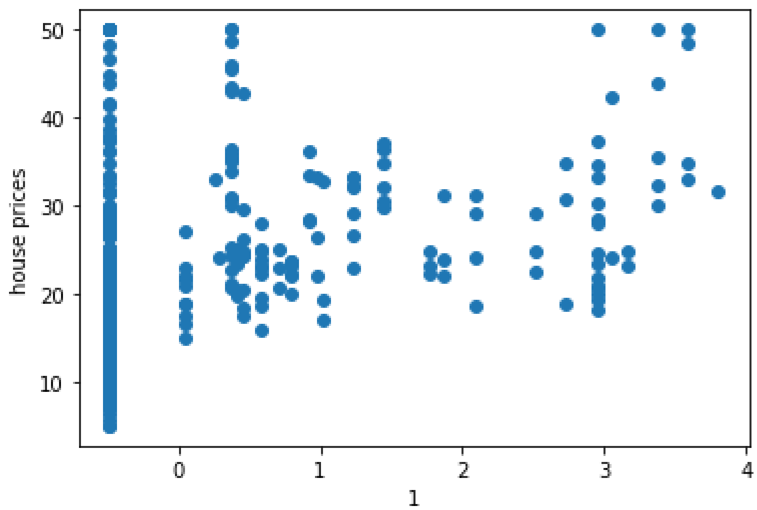
	0	1	2	3	4	5	6	
0	-0.419782	0.284830	-1.287909	-0.272599	-0.144217	0.413672	-0.120013	0.140
1	-0.417339	-0.487722	-0.593381	-0.272599	-0.740262	0.194274	0.367166	0.557
2	-0.417342	-0.487722	-0.593381	-0.272599	-0.740262	1.282714	-0.265812	0.557
3	-0.416750	-0.487722	-1.306878	-0.272599	-0.835284	1.016303	-0.809889	1.077
4	-0.412482	-0.487722	-1.306878	-0.272599	-0.835284	1.228577	-0.511180	1.077
...	...	...	...	...	...	...	...	
501	-0.413229	-0.487722	0.115738	-0.272599	0.158124	0.439316	0.018673	-0.625
502	-0.415249	-0.487722	0.115738	-0.272599	0.158124	-0.234548	0.288933	-0.716
503	-0.413447	-0.487722	0.115738	-0.272599	0.158124	0.984960	0.797449	-0.773
504	-0.407764	-0.487722	0.115738	-0.272599	0.158124	0.725672	0.736996	-0.668
505	-0.415000	-0.487722	0.115738	-0.272599	0.158124	-0.362767	0.434732	-0.613

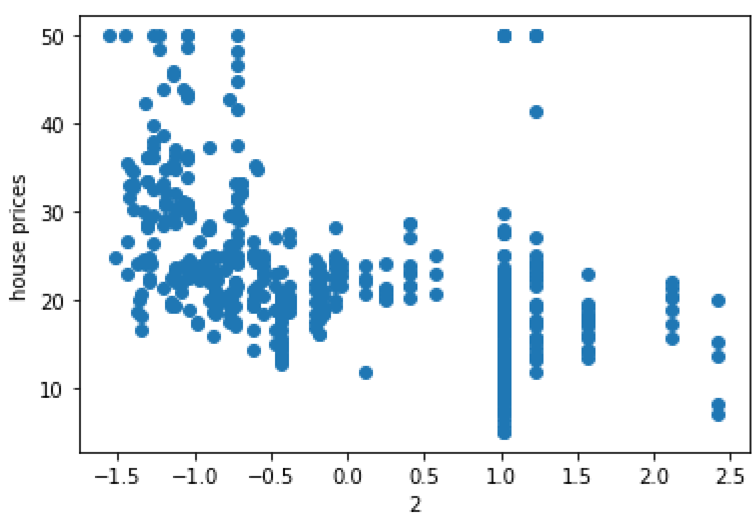
506 rows × 14 columns

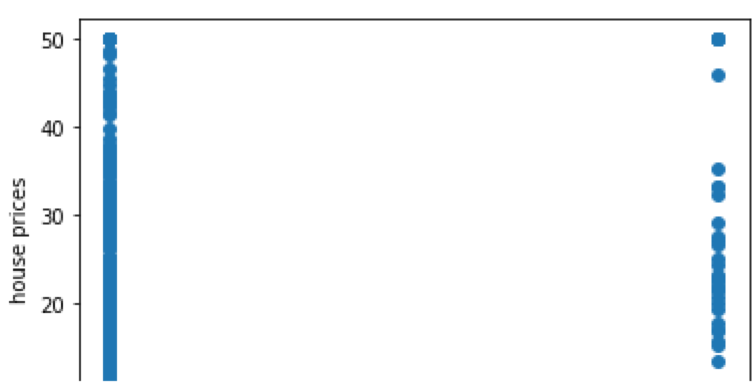
```
for i in b_data.columns:
  plt.scatter(b_data[i], b_data['prices'])
```

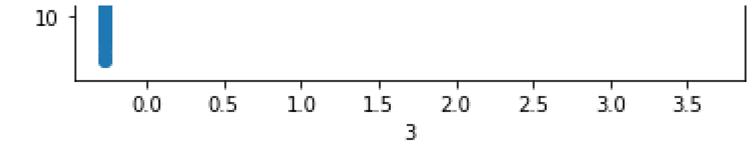
```
    plt.xlabel(i)
    plt.ylabel('house prices')
    plt.show()
```

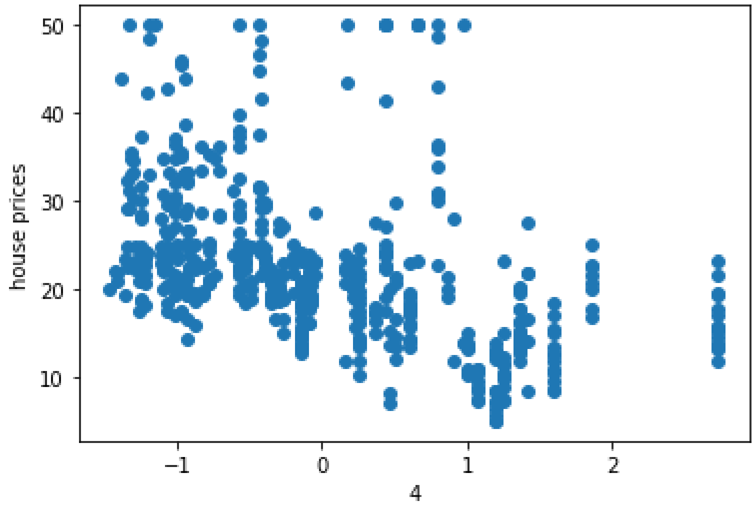

house prices
50
40
30
20
10
0
2
4
6
8
10
0

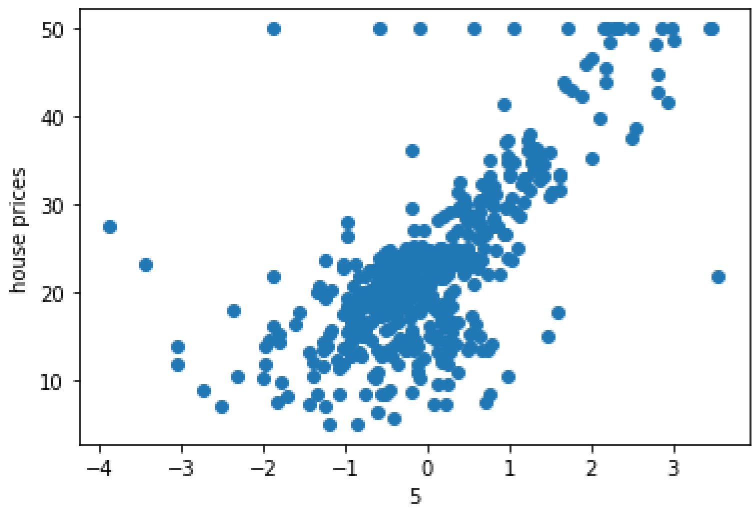

house prices
50
40
30
20
10
0
1
2
3
4
1

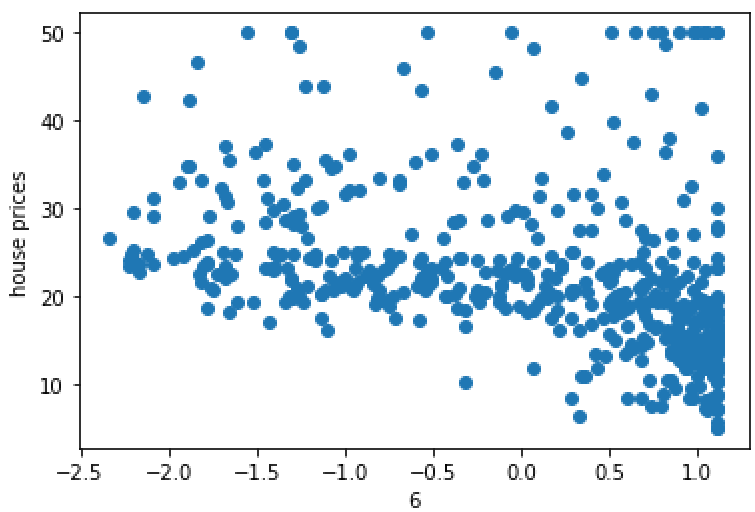

house prices
50
40
30
20
10
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
2

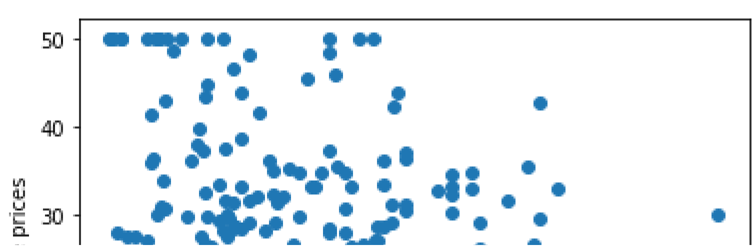

house prices
50
40
30
20

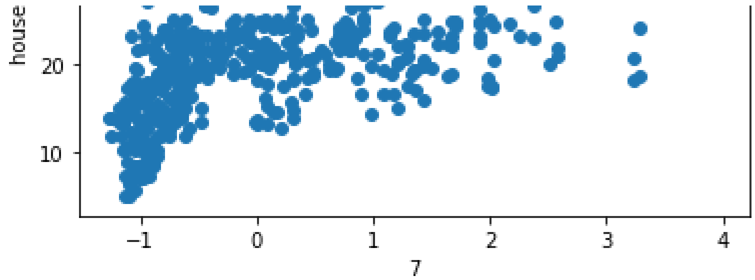

10
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
3

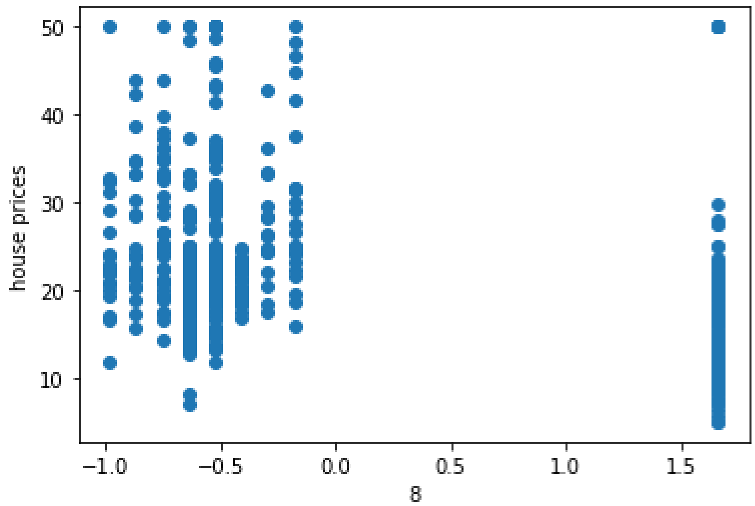

house prices
50
40
30
20
10
−1
0
1
2
4

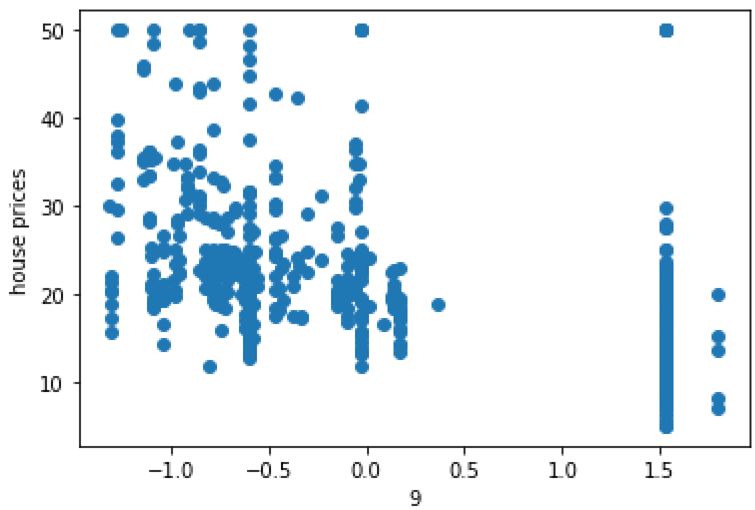

house prices
50
40
30
20
10
−4
−3
−2
−1
0
1
2
3
5

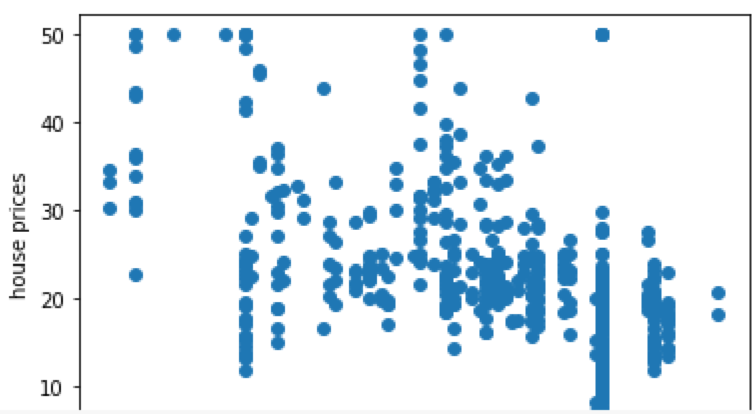

house prices
50
40
30
20
10
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
6


prices
50
40
30










```
from sklearn import linear_model
```

```
Y = b_data['prices']
X = b_data.drop("prices", axis=1)
```

```
X = b_data.drop('prices', axis=1)
X = np.array(X)
X.reshape(1, -1)
```

```
array([[-0.41978194,  0.28482986, -1.2879095 , ...,  1.17646583,
         0.44105193, -0.66905833]])
```

```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(X, Y, test_size=0.2, random_state=4)
```

```
regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
```

```
LinearRegression()
```

```
y_pred = regr.predict(test_x)
```

```
regr.score(test_x, test_y)
```

```
0.7263451459702509
```

This is the testing accuracy calculated as 72.58%

```
regr.score(train_x, train_y)
```

```
0.7415244219726307
```

This is training accuracy calculated as 70.86%

```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(test_y, y_pred))
print('MSE:', metrics.mean_squared_error(test_y, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(test_y, y_pred)))
```

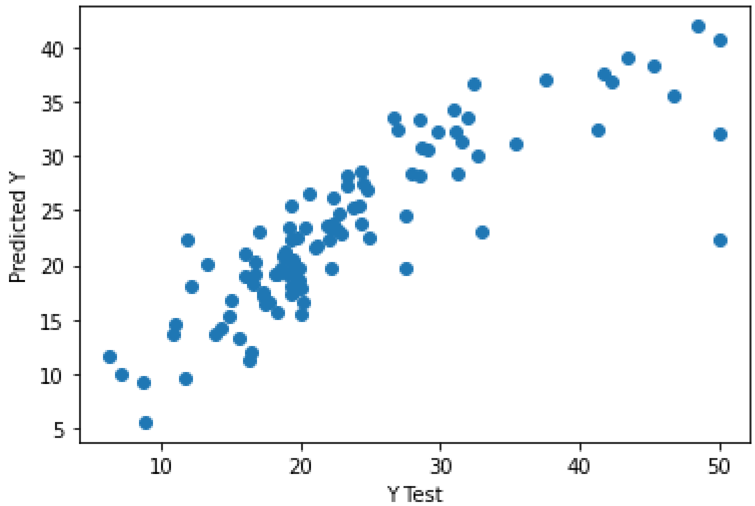
```
MAE: 3.367790983796573
MSE: 25.419587126821853
RMSE: 5.041784121402051
```

Root mean squared error in this case is 4.69

```
plt.scatter(test_y,y_pred)
plt.xlabel('Y Test')
```

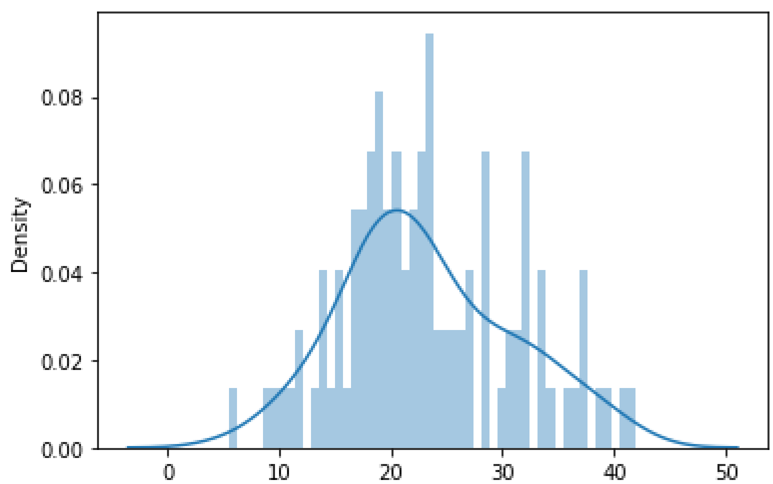
```
plt.ylabel('Predicted Y')
```

```
Text(0, 0.5, 'Predicted Y')
```




```
import seaborn as sns
sns.distplot(y_pred,bins=50)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarnin
  warnings.warn(msg, FutureWarning)
```



Conclusion

The model performance for the training set is 70.86% where as model performance for testing set is 72.58%.

The RMSE that is root mean squared error of the model is 4.69.

Closing

Lets Finally create a file which contain all the independent as well as dependent variables and also the corresponding predicted values along with them.

```
y_train_pred = regr.predict(train_x)
list1 = y_train_pred.tolist() + y_pred.tolist()
```

```
output = boston_data
output['predicted_prices'] = list1
#regr.predict(X)
output.to_csv('/content/output.csv')
```