Task Predicting the House Price

Content

Each record in the database describes a Boston suburb or town. The data was drawn from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970. The attributes are defined as follows (taken from the UCI Machine Learning Repository

CRIM: per capita crime rate by town

ZN: proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS: proportion of non-retail business acres per town

CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) NOX: nitric oxides concentration (parts per 10 million) 1https://archive.ics.uci.edu/ml/datasets/Housing 123 20.2. Load the Dataset 124

RM: average number of rooms per dwelling

AGE: proportion of owner-occupied units built prior to 1940

DIS: weighted distances to five Boston employment centers

RAD: index of accessibility to radial highways

TAX: full-value property-tax rate per $10,000

PTRATIO: pupil-teacher ratio by town 12. B: 1000(Bk−0.63)2 where Bk is the proportion of blacks by town 13. LSTAT: % lower status of the population

MEDV: Median value of owner-occupied homes in $1000s We can see that the input attributes have a mixture of units.

LOADING DATASET

```
import numpy as np
import pandas as pd
from pandas import Series,DataFrame
from sklearn.datasets import load_boston
import seaborn as sns
import matplotlib.pyplot as plt
```

```
boston = load_boston()
print(boston.data)
print(boston.data.shape)
```

```
[[6.3200e-03 1.8000e+01 2.3100e+00 ... 1.5300e+01 3.9690e+02 4.9800e+00]
 [2.7310e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01 3.9690e+02 9.1400e+00]
 [2.7290e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01 3.9283e+02 4.0300e+00]
 ...
```

```
 [6.0760e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9690e+02 5.6400e+00]
 [1.0959e-01 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9345e+02 6.4800e+00]
 [4.7410e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9690e+02 7.8800e+00]]
(506, 13)
```

bos

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90

506 rows × 14 columns

```
print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwi
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
```

```
            - LSTAT    % lower status of the population
            - MEDV     Median value of owner-occupied homes in $1000's

        :Missing Attribute Values: None

        :Creator: Harrison, D. and Rubinfeld, D.L.

    This is a copy of UCI ML housing dataset.
    https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


    This dataset was taken from the StatLib library which is maintained at Carnegie Mellc

    The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
    prices and the demand for clean air', J. Environ. Economics & Management,
    vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
    ...', Wiley, 1980.   N.B. Various transformations are used in the table on
    pages 244-261 of the latter.

    The Boston house-price data has been used in many machine learning papers that addres
    problems.

    .. topic:: References

       - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data anc
       - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceec
```

```
bos.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AC
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.0000(
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.5749(
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.1488(
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.9000(
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.0250(
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.5000(
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.0750(
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.0000(

```
bos.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	

DATA PREPROCESSING

2 0.02729 0.0 7.07 0.0 0.469 7.185 61.1 4.9671 2.0 242.0 17.8 392.83

Checking for missing values

4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	

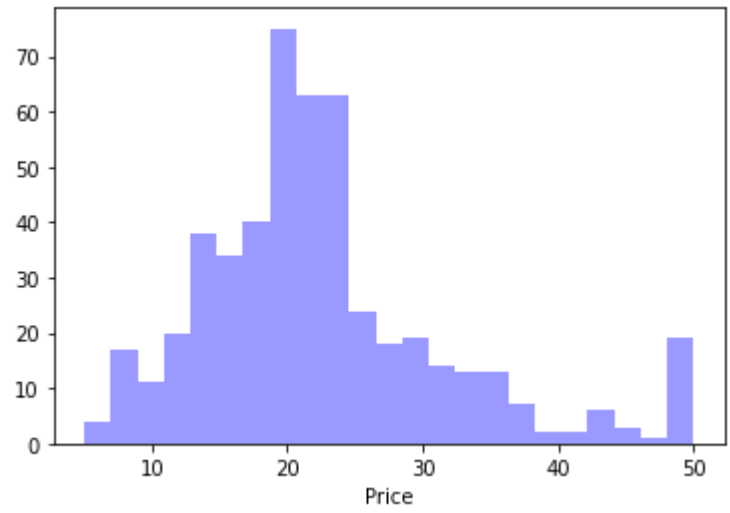
```
bos.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
PRICE      0
dtype: int64
```

VISUALSING DATA

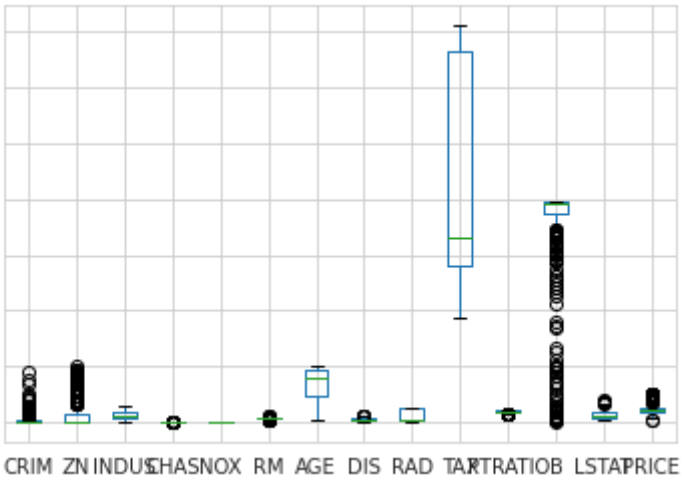
```
sns.distplot(bos.Price,kde=False,color="b")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fc09e74b0d0>
```

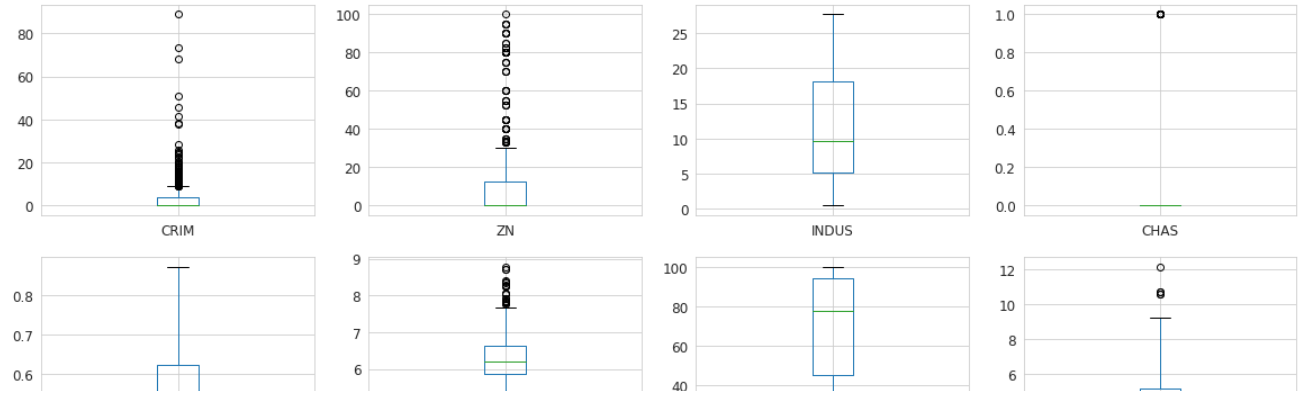



```
bos.boxplot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6cad3990>
```




```
bos.plot(kind='box',subplots=True,layout=(5,4),fontsize=12,figsize=(20,20))
plt.show()
```

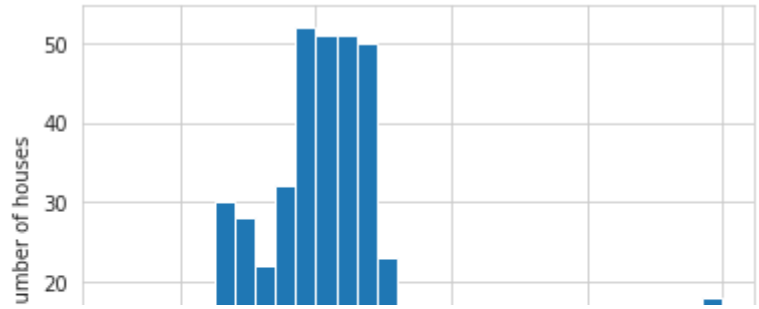



```
bos.describe(include='all')
corr = bos.corr()
corr.style.background_gradient(cmap = 'coolwarm') # let's color code correlation
```

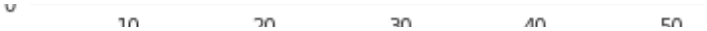
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.3
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.6
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.7
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.0
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.7
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.2
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.7
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.0
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.4
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.5
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.2
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.2
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.4
Price	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.2

```
# Histogram of prices (this is the target of our dataset)
plt.hist(bos['PRICE'],bins=30)
sns.set_style('whitegrid')
#label
plt.xlabel('Price in $1000s')
plt.ylabel('Number of houses')
```

```
Text(0, 0.5, 'Number of houses')
```



Interesting, now let's see a scatter plot of one feature, versus the target. In this case we'll use the housing price versus the number of rooms in the dwelling.




```
# Plot the column at the 5 index (Labeled RM)
plt.scatter(boston.data[:,5],boston.target)
plt.ylabel('Price in $1000s')
plt.xlabel('Number of rooms')
plt.figure()

plt.scatter(boston.data[:,6],boston.target)
plt.xlabel('Age')
plt.ylabel('Price in $1000s')
plt.figure()
```

<Figure size 432x288 with 0 Axes>

Great! Now we can make out a slight trend that price increases along with the number of rooms in that house, which intuitively makes sense! Now let's use scikit learn to see if we can fit the data linearly. Let's try to do the following:

1.) Use pandas to transform the boston dataset into a DataFrame:

2.) Then use seaborn to perform an lmplot on that DataFrame to reproduce the scatter plot with a linear fit line.

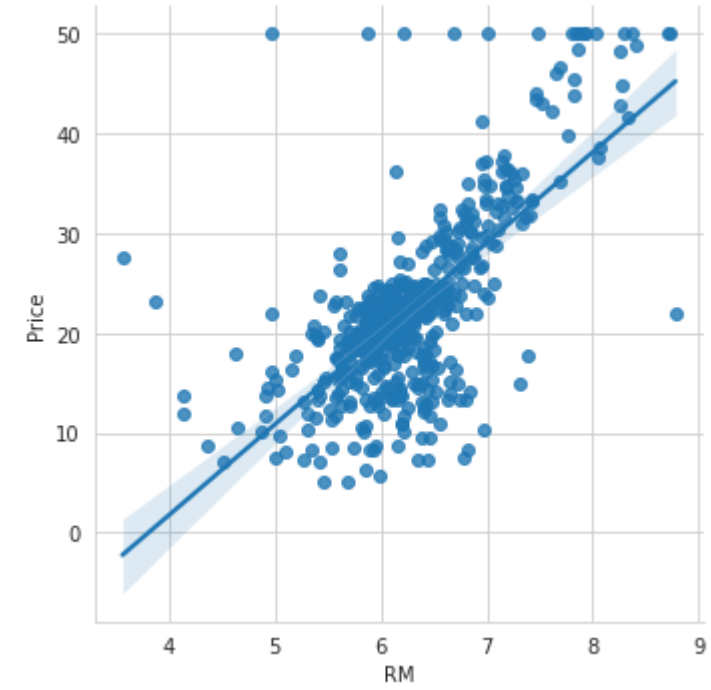
```
bos = DataFrame(boston.data)
bos.columns = boston.feature_names
bos['Price'] = boston.target # our target value is price
```

Now let's add the target of the boston data set, the price. We'll create a new column in our DataFrame

```
# Set price column for target
bos['Price'] = boston.target
```

```
sns.lmplot('RM','Price',data=bos)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<seaborn.axisgrid.FacetGrid at 0x7f1c69f51450>
```



Using Numpy for a Univariate Linear Regression

```
X=bos['RM']
```

```
X=np.vstack(bos['RM'])
Y=bos['Price']

X=np.array([[value,1] for value in X])
Y=bos['Price']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: VisibleDeprecationWar
  """Entry point for launching an IPython kernel.
```

```
import scipy
scipy.linalg.lstsq(X,Y)
```

```
(array([3.6533504]), 29555.781528643478, 1, array([142.24836817]))
```

```
bos.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	

Using scikit learn to implement a multivariate regression

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error

Lreg = LinearRegression()
```

The functions we will be using are:

lreg.fit() which fits a linear model

lreg.predict() which is used to predict Y using the linear model with estimated coefficients

lreg.score() which returns the coefficient of determination (R^2).

We'll start the multi variable regression analysis by seperating our boston dataframe into the data columns and the target columns:

```
#Data Columns
```

```
X_Feature=bos.iloc[:,0:13]

#Target
Y_Target=bos.Price

Lreg.fit(X_Feature,Y_Target)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
print("The estimated intercept coefficient is %.2f"%Lreg.intercept_)
```

```
The estimated intercept coefficient is 36.46
```

```
print("The number of coefficients used are %d"%len(Lreg.coef_))
```

```
The number of coefficients used are 13
```

Next step is to set up a DataFrame showing all the Features and their estimated coefficients obtained form the linear regression.

```
# Set a DataFrame from the Features
coeff_df = DataFrame(bos.columns)
coeff_df.columns = ['Features']

# Set a new column lining up the coefficients from the linear regression
coeff_df["Coefficient Estimate"] = pd.Series(Lreg.coef_)

# Show
coeff_df
```

	Features	Coefficient Estimate
0	CRIM	-0.108011

Just like we initially plotted out, it seems the highest correlation between a feature and a house price was the number of rooms.

Now let's move on to Predicting prices!

4	NOX	-17.766611

Using Training and Validation

6	AGE	0.000692

```
# Grab the output and set as X and Y test and train data sets!
X=np.vstack(bos['RM'])
X_train, X_test, Y_train, Y_test = train_test_split(X,bos.Price,test_size=0.3,random_state
```

9	TAX	-0.012335

```
# Print shapes of the training and testing data sets
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
(354, 1) (152, 1) (354,) (152,)
```

12	LSTAT	-0.524758

Predicting Prices

```
Lreg = LinearRegression()
```

```
Lreg.fit(X_train,Y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Now run a prediction on both the X training set and the testing set.

```
pred_train=Lreg.predict(X_train)
Y_pred=Lreg.predict(X_test)
```

Now we will get the mean square error

```
print("Fit a model X_train, and calculate MSE with Y_train: %.2f"  % np.mean((Y_train - pr
```

```
print("Fit a model X_train, and calculate MSE with X_test and Y_test: %.2f"  %np.mean((Y_t
```

```
Fit a model X_train, and calculate MSE with Y_train: 42.16
Fit a model X_train, and calculate MSE with X_test and Y_test: 47.03
```

```
r2_score(Y_test,Y_pred)
```

```
0.43514364832115193
```

It looks like our mean square error between our training and testing was pretty close. But how do we actually visualize this?

Residual Plots

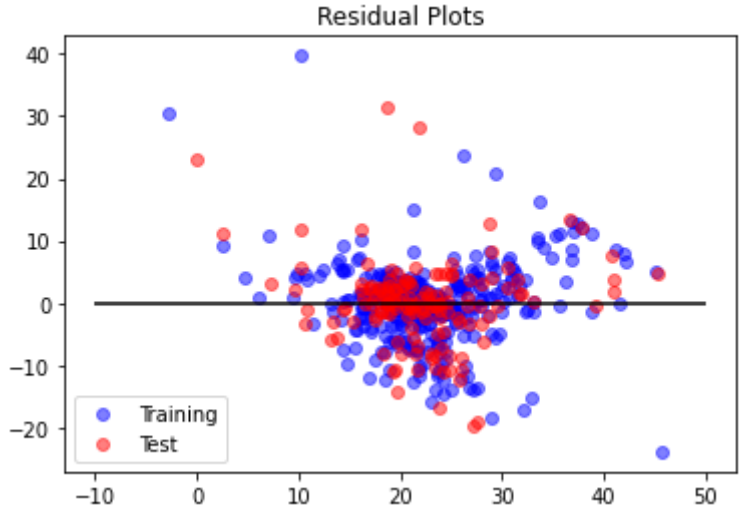
```
# Scatter plot the training data
train = plt.scatter(pred_train,(Y_train-pred_train),c='b',alpha=0.5)

# Scatter plot the testing data
test = plt.scatter(Y_pred,(Y_test-Y_pred),c='r',alpha=0.5)

# Plot a horizontal axis line at 0
plt.hlines(y=0,xmin=-10,xmax=50)

#Labels
plt.legend((train,test),('Training','Test'),loc='lower left')
plt.title('Residual Plots')
```

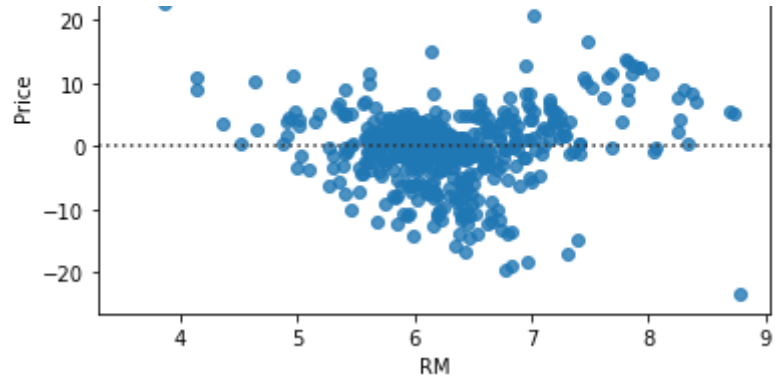
```
Text(0.5, 1.0, 'Residual Plots')
```




```
# Residual plot of all the dataset using seaborn
sns.residplot('RM', 'Price', data = bos)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7ff336b27990>
```

Double-click (or enter) to edit



Colab paid products - Cancel contracts here