1. Supervised Learning - Linear Regression

February 26, 2017

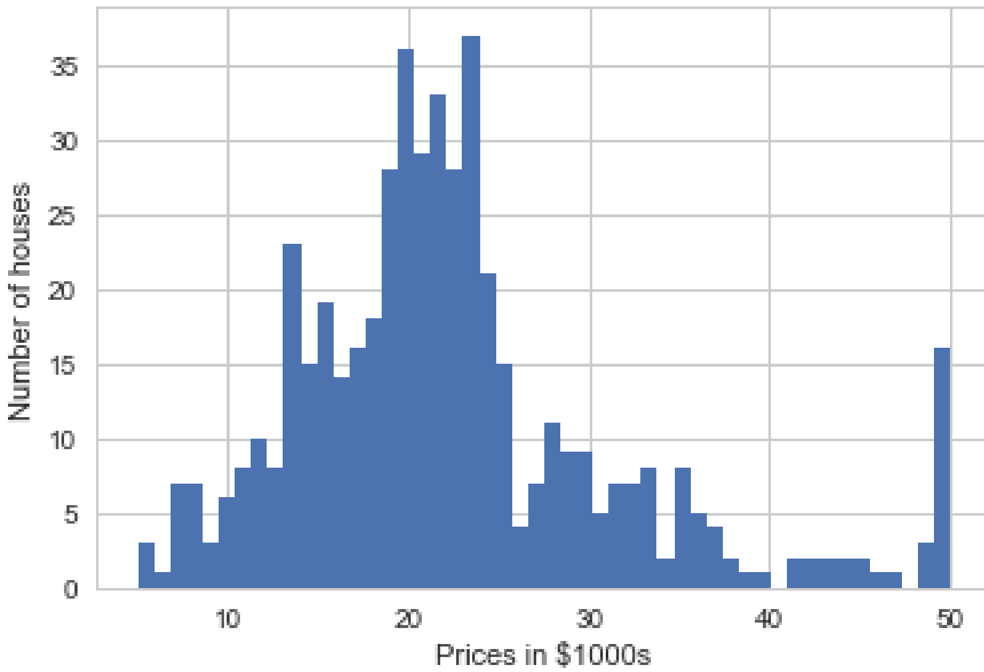
```
In [109]: import numpy as np
          import pandas as pd
          from pandas import Series, DataFrame

In [110]: import matplotlib.pyplot as plt
          import seaborn as sns

          %matplotlib inline
          sns.set_style('whitegrid')

In [111]: from sklearn.datasets import load_boston #load dataset boston

In [112]: boston = load_boston()

In [113]: print boston.DESCR
```

```
Boston House Prices dataset
===========================

Notes
------
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :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 other
        - 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.
http://archive.ics.uci.edu/ml/datasets/Housing


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

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 addr
problems.

**References**

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data a
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proce
   - many more! (see http://archive.ics.uci.edu/ml/datasets/Housing)
```

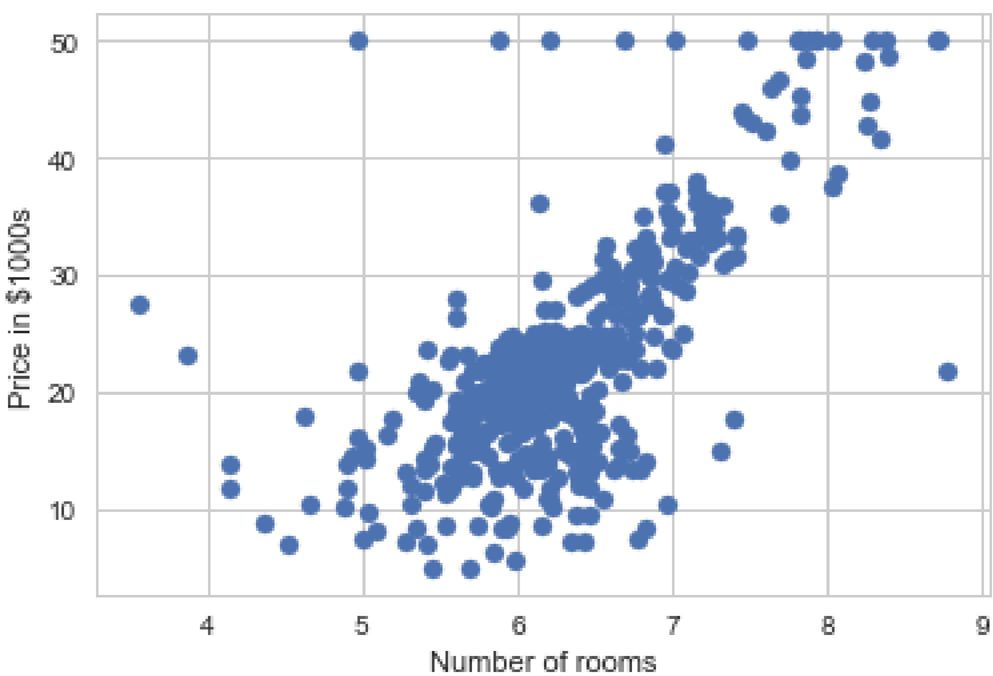
In [114]:
```
plt.hist(boston.target,bins=50)
plt.xlabel('Prices in $1000s')
plt.ylabel('Number of houses')
```

Out[114]:
```
<matplotlib.text.Text at 0x11ce67550>
```




```
In [115]: plt.scatter(boston.data[:,5],boston.target)

          plt.ylabel('Price in $1000s')
          plt.xlabel('Number of rooms')
```

```
Out[115]: <matplotlib.text.Text at 0x11cf3d990>
```



```
In [116]: boston_df = DataFrame(boston.data)

          boston_df.columns = boston.feature_names

          boston_df.head()
```

```
Out[116]:       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  1.0  296.0
          1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
          2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
          3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
          4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

             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   5.33
```

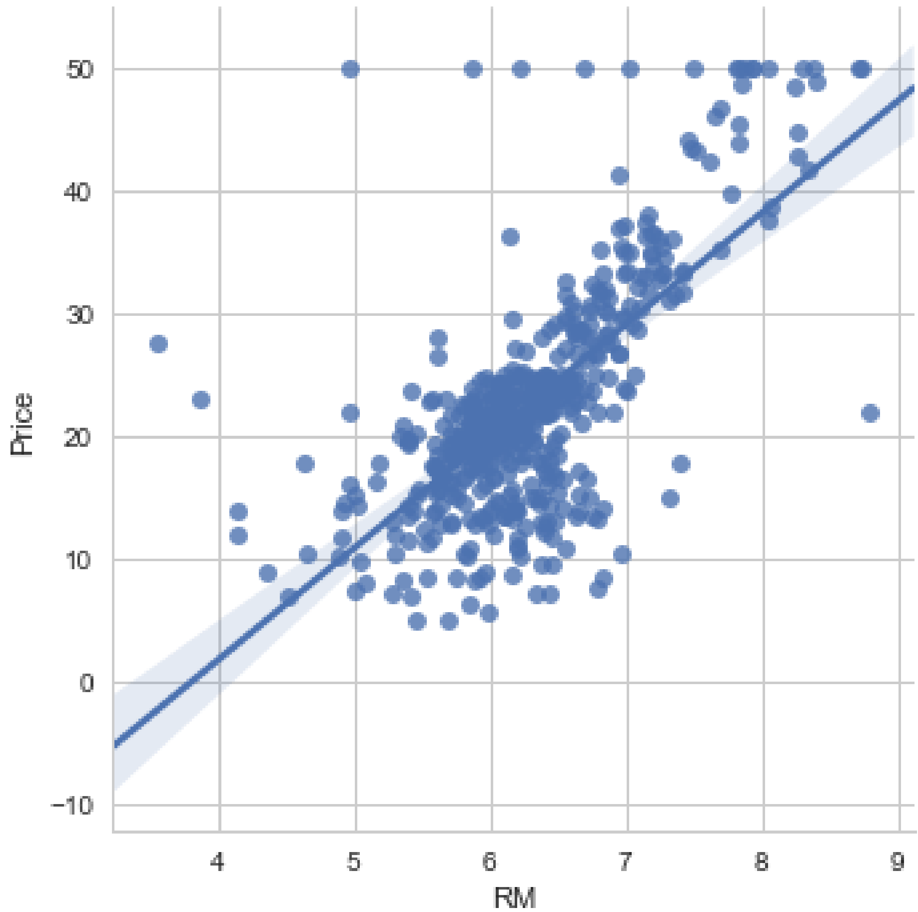
```
In [117]: boston_df['Price'] = boston.target
```

```
In [118]: boston_df.head()
```

```
Out[118]:       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  1.0  296.0
          1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
          2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
          3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
          4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

             PTRATIO       B  LSTAT  Price
          0     15.3  396.90   4.98   24.0
          1     17.8  396.90   9.14   21.6
          2     17.8  392.83   4.03   34.7
          3     18.7  394.63   2.94   33.4
          4     18.7  396.90   5.33   36.2
```

```
In [119]: sns.lmplot('RM','Price',data=boston_df)
```

```
Out[119]: <seaborn.axisgrid.FacetGrid at 0x11de2d250>
```




```
In [120]: X= boston_df.RM

In [121]: X.head()

Out[121]: 0    6.575
          1    6.421
          2    7.185
          3    6.998
          4    7.147
          Name: RM, dtype: float64

In [122]: X = np.vstack(boston_df.RM)
          X.shape

Out[122]: (506, 1)

In [123]: Y = boston_df.Price
          Y.head()
```

Out[123]:
```
0    24.0
1    21.6
2    34.7
3    33.4
4    36.2
Name: Price, dtype: float64
```

In [124]:
```
# array in the form of [X 1]
X = np.array([[value,1] for value in X])
```

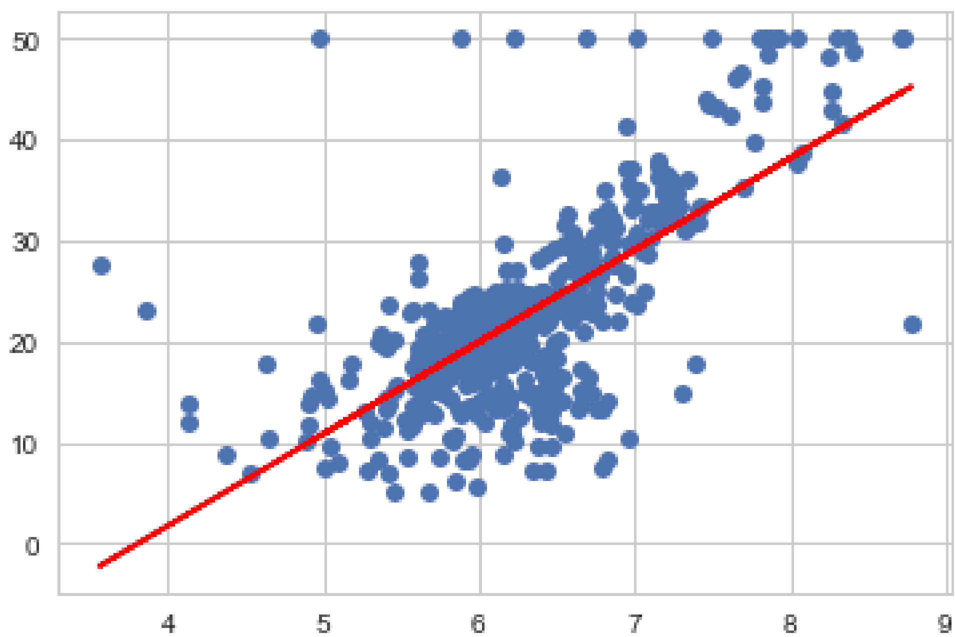
In [125]:
```
X
```

Out[125]:
```
array([[array([ 6.575]), 1],
       [array([ 6.421]), 1],
       [array([ 7.185]), 1],
       ...,
       [array([ 6.976]), 1],
       [array([ 6.794]), 1],
       [array([ 6.03]), 1]], dtype=object)
```

In [126]:
```
m , b = np.linalg.lstsq(X,Y)[0]
```

In [145]:
```
plt.plot(boston_df.RM,boston_df.Price,'o')

x = boston_df.RM

plt.plot(x, m*x+b,'r',label = 'Best Fit Line')
```

Out[145]:
```
[<matplotlib.lines.Line2D at 0x11e4a2450>]
```



In [128]:
```
result = np.linalg.lstsq(X,Y)

error_total = result[1]

rmse = np.sqrt(error_total/len(X))

print 'The root mean square error was %.2f' %rmse
```

```
The root mean square error was 6.60
```

In [129]:
```
import sklearn
from sklearn.linear_model import LinearRegression
```

In [130]:
```
lreg = LinearRegression()
```

In [131]:
```
X_multi = boston_df.drop('Price',1)

Y_target = boston_df.Price
```

In [132]:
```
lreg.fit(X_multi,Y_target)
```

Out[132]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Fal
```

In [133]:
```
print ' The estimated intercept coefficient is %.2f' %lreg.intercept_

print ' The nuber of the coefficient used was %d' %len(lreg.coef_)
```

```
 The estimated intercept coefficient is 36.49
 The nuber of the coefficient used was 13
```

In [134]:
```
coeff_df = DataFrame(boston_df.columns)
coeff_df.columns = ['Features']

coeff_df['Coefficient Estimate'] = Series(lreg.coef_)

coeff_df
```

Out[134]:

	Features	Coefficient Estimate
0	CRIM	-0.107171
1	ZN	0.046395
2	INDUS	0.020860
3	CHAS	2.688561
4	NOX	-17.795759
5	RM	3.804752
6	AGE	0.000751

```
7       DIS              -1.475759
8       RAD               0.305655
9       TAX              -0.012329
10  PTRATIO              -0.953464
11        B               0.009393
12    LSTAT              -0.525467
13    Price                    NaN
```

In [135]:
```
import sklearn
#from sklearn.model_selection import train_test_split
```

In [136]:
```
# Grab the output and set as X and Y test and train data sets!

X_train, X_test, Y_train, Y_test = sklearn.model_selection.train_test_spl
```

In [137]:
```
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
((379, 2), (127, 2), (379,), (127,))
```

In [138]:
```
lreg = LinearRegression()

lreg.fit(X_train,Y_train)
```

Out[138]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Fal
```

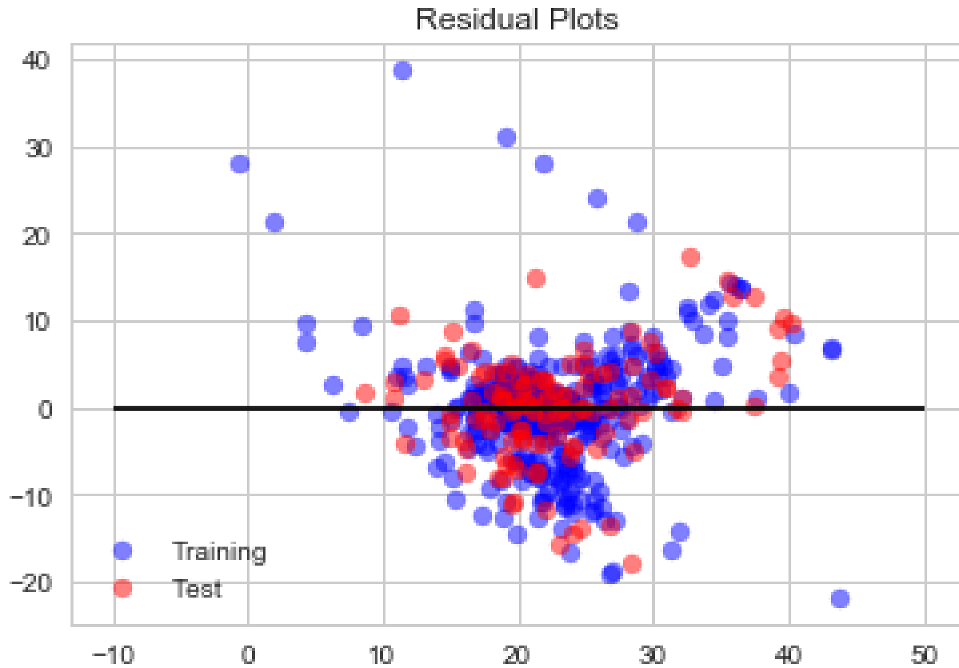
In [139]:
```
pred_train = lreg.predict(X_train)
pred_test = lreg.predict(X_test)
```

In [140]:
```
print "Fit a model X_trian, and calculate the MSE with the Y_train: %.2f"

print "Fit a model X_trian, and calculate the MSE with the X_test and Y_t
```

```
Fit a model X_trian, and calculate the MSE with the Y_train: 46.32
Fit a model X_trian, and calculate the MSE with the X_test and Y_test: 36.23
```

In [141]:
```
# 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(pred_test,(Y_test-pred_test),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')
```

Out[141]: <matplotlib.text.Text at 0x11e2e4b90>




```
In [142]: sns.residplot('RM', 'Price', data = boston_df)
```

Out[142]: <matplotlib.axes._subplots.AxesSubplot at 0x11e0f4410>