```
<h1>SKLearn Tutorial: Linear Regression on Boston Data</h1>
This is following  the <a
href="https://github.com/tensorflow/tensorflow/blob/master/tensorflow/exam
ples/skflow/boston.py">https://github.com/tensorflow/tensorflow/blob/maste
r/tensorflow/examples/skflow/boston.py</a> and
<a href="http://bigdataexaminer.com/uncategorized/how-to-run-linear-
regression-in-python-scikit-
learn/">http://bigdataexaminer.com/uncategorized/how-to-run-linear-
regression-in-python-scikit-learn/</a> tutorials.
<br />
D. Thiebaut
<br />August 2016
```

Get the Boston Data

This part is basically taken directly from the bigdataexaminer (http://bigdataexaminer.com/uncategorized/how-to-run-linear-regression-in-python-scikit-learn/) tutorial. All the Imports first

In [94]:
```
"""Example of DNNRegressor for Housing dataset."""

from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
from sklearn import cross_validation
from sklearn import metrics
from sklearn import preprocessing
import tensorflow as tf
from tensorflow.contrib import learn
import pandas as pd
```

Get the data...

In [95]:
```
from sklearn.datasets import load_boston
boston = load_boston()
print( "type of boston = ", type(boston))
```

```
type of boston =  <class 'sklearn.datasets.base.Bunch'>
```

In [96]:
```
boston.keys()
```

Out[96]:
```
['data', 'feature_names', 'DESCR', 'target']
```

In [97]:
```
boston.data.shape
```

Out[97]:
```
(506, 13)
```

In [98]:
```
print( boston.feature_names )
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

In [99]:
```
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 rive
r; 0 otherwise)
        - 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 (http://archive.ics.uci.ed
u/ml/datasets/Housing)

This dataset was taken from the StatLib library which is maintained at Ca
rnegie Mellon University.

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 diagnos
tics
...', Wiley, 1980.   N.B. Various transformations are used in the table o
n
pages 244-261 of the latter.

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

References

```
   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influent
ial Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learnin
g. In Proceedings on the Tenth International Conference of Machine Learni
ng, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
   - many more! (see http://archive.ics.uci.edu/ml/datasets/Housing) (htt
p://archive.ics.uci.edu/ml/datasets/Housing))
```

In [100]:
```
print( "target = ",
          ",".join( str(k) for k in boston.target[0:5]  ),
          "...",
          ", ".join( str(k) for k in boston.target[-5:] ) )
```

```
target =  24.0,21.6,34.7,33.4,36.2 ... 22.4, 20.6, 23.9, 22.0, 11.9
```

Convert the boston data into a panda data-frame

In [101]:
```
bostonDF = pd.DataFrame( boston.data )
bostonDF.head()
```

Out[101]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

Add column names

In [102]:
```
bostonDF.columns = boston.feature_names
bostonDF.head()
```

Out[102]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
0	0.00632	18	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

Adding the target to the data frame...

In [103]:
```
bostonDF['PRICE' ]= boston.target
bostonDF.head()
```

Out[103]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
0	0.00632	18	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

So now we have a pandas data frame holding the data.

Predicting Housing Prices with Linear Regression

In [104]:
```
from sklearn.linear_model import LinearRegression
```

In [109]:
```
X = bostonDF.drop( 'PRICE', axis = 1 )
y = bostonDF[ 'PRICE']
lm = LinearRegression()
lm
```

Out[109]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Fal
se)
```

The LinearRegression objects supports several methods:

- fit(): fits a linear model
- predict(): predicts Y using the linear model's estimated coeffs
- score(): returns the coef of determination R^2
- get_params():
- mro():
- register():
- set_params():

Fitting the Model

We are going to use all 13 parameters to fit a linear regression model

In [110]:
```
lm.fit( X, y)
```

Out[110]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Fal
se)
```

In [114]:
```
print( "Estimated intercept coeff: ", lm.intercept_ )
print( "Number of coeffs: ", len( lm.coef_ ))
print( "Coeffs = ", lm.coef_ )
```

```
Estimated intercept coeff:  36.4911032804
Number of coeffs:  13
Coeffs =  [ -1.07170557e-01   4.63952195e-02   2.08602395e-02   2.6885614
0e+00
  -1.77957587e+01   3.80475246e+00   7.51061703e-04  -1.47575880e+00
   3.05655038e-01  -1.23293463e-02  -9.53463555e-01   9.39251272e-03
  -5.25466633e-01]
```

Create a dataframe with the coeffs

In [115]:
```
pd.DataFrame( zip(X.columns, lm.coef_),
              columns=['features', 'estimatedCoeffs'])
```

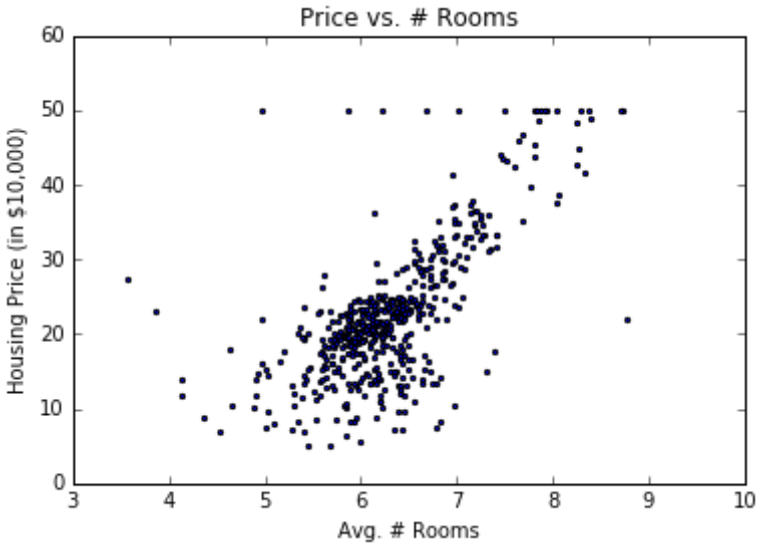
Out[115]:

	features	estimatedCoeffs
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

Generate a plot of Price versus RM (Avg # of Rooms per dwelling)

In [138]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter( bostonDF.RM, bostonDF.PRICE, s=5 )
plt.xlabel( "Avg. # Rooms" )
plt.ylabel( "Housing Price (in $10,000)" )
plt.title( "Price vs. # Rooms")
```

Out[138]:
```
<matplotlib.text.Text at 0x1170f1f10>
```



Predicting Prices

In [131]:
```
lm.predict( X)[0:10]
```

Out[131]:
```
array([ 30.00821269,  25.0298606 ,  30.5702317 ,  28.60814055,
        27.94288232,  25.25940048,  23.00433994,  19.5347558 ,
        11.51696539,  18.91981483])
```

Plot prediction against real values

In [137]:
```
plt.scatter( bostonDF.PRICE, lm.predict(X), s=5 )
plt.xlabel( "Prices")
plt.ylabel( "Predicted Prices")
plt.title( "Real vs Predicted Housing Prices")
```

Out[137]: <matplotlib.text.Text at 0x116cc7dd0>

Let's compute the mean squared error:

In [139]:
```
mse = np.mean( (bostonDF.PRICE-lm.predict(X))**2)
print( "Mean squared error = ", mse )
```

```
Mean squared error =  21.8977792177
```

Training and Validating

In [142]:
```
X_train, X_test, y_train, y_test = \
     cross_validation.train_test_split( X,
                                        bostonDF.PRICE,
                                        test_size=0.33,
                                        random_state=5 )
print( X_train.shape, X_test.shape, y_train.shape, y_test.shape )
```

```
(339, 13) (167, 13) (339,) (167,)
```

Building a linear regression model using only the train data:

In [170]:
```
lm = LinearRegression()
lm.fit( X_train, y_train )
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-170-a70c6f3ff2f8> in <module>()
      1 lm = LinearRegression()
----> 2 lm.fit( X_train, y_train, logdir='/tmp/SKLearnLinReg/' )

TypeError: fit() got an unexpected keyword argument 'logdir'
```

In [146]:
```
pred_train = lm.predict( X_train )
pred_test  = lm.predict( X_test )
print( "mse_train = ", np.mean( (y_train-lm.predict(X_train))**2) )
print( "mse_test  = ", np.mean( (y_test-lm.predict(X_test))**2) )
```

```
mse_train =  19.5467584735
mse_test  =  28.5413672756
```

Plotting the Residuals

In [164]:
```
plt.scatter( lm.predict(X_train), lm.predict(X_train) - y_train,
            c ='b', s=30, alpha=0.4 )
plt.scatter( lm.predict(X_test), lm.predict(X_test) - y_test,
            c ='g', s=30 )
plt.hlines( y=0, xmin=-5, xmax=55)
plt.title( "Residuals" )
plt.ylabel( "Residuals" )
```

Out[164]: <matplotlib.text.Text at 0x118987b90>