5.3.1 The Validation Set Approach

May 18, 2018

In [8]:
```
# conventional way to import pandas
import pandas as pd
# conventional way to import numpy
import numpy as np

from sklearn import metrics
import matplotlib.pyplot as plt

data = pd.read_csv("https://raw.github.com/vincentarelbundock/Rdatasets/master/csv/ISL
print(data.shape)
data.head()
```

```
(392, 9)
```

Out[8]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
1	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
2	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
3	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
4	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
5	17.0	8	302.0	140	3449	10.5	70	1	ford torino

ISLR Auto is a data frame with 392 observations on the following 9 variables:

```
mpg: miles per gallon
cylinders: Number of cylinders between 4 and 8
displacement: Engine displacement (cu. inches)
horsepower: Engine horsepower
weight: Vehicle weight (lbs.)
acceleration: Time to accelerate from 0 to 60 mph (sec.)
```

```
year: Model year (modulo 100)
origin: Origin of car (1. American, 2. European, 3. Japanese)
name: Vehicle name
```

We take a 196 random samples out of the data. We are using a random seed, and because of this our answers will vary from the book:

```
In [9]: np.random.seed(1)
        train = np.random.choice(data.shape[0], 196, replace=False)
        test = np.in1d(range(data.shape[0]), train)

        traindata = data[test]

        print(traindata.shape)
        traindata.head()
```

```
(196, 9)
```

```
Out[9]:     mpg  cylinders  displacement  horsepower  weight  acceleration  year  \
        1  18.0          8         307.0         130    3504          12.0    70
        5  17.0          8         302.0         140    3449          10.5    70
        6  15.0          8         429.0         198    4341          10.0    70
        7  14.0          8         454.0         220    4354           9.0    70
        9  14.0          8         455.0         225    4425          10.0    70

           origin                       name
        1       1  chevrolet chevelle malibu
        5       1               ford torino
        6       1          ford galaxie 500
        7       1          chevrolet impala
        9       1          pontiac catalina
```

We will fit a linear regression using only the observations corresponding to the training set and then,we now use the lm.predict(data) to function to estimate the response for all 392 observations and then we calulate the square_error in hand to get the MSE of the 196 observations in the validation set.

```
In [10]: import statsmodels.formula.api as smf
         lm = smf.ols ('mpg~horsepower', traindata).fit() #Train the model on the traindata.

         print(lm.summary())

         preds = lm.predict(data)
         square_error = (data['mpg'] - preds)**2
         print('--------Test Error for 1st order--------')
         print(np.mean(square_error[~test]))
```

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                    mpg   R-squared:                       0.620
Model:                            OLS   Adj. R-squared:                  0.618
Method:                 Least Squares   F-statistic:                     316.4
Date:                Fri, 18 May 2018   Prob (F-statistic):           1.28e-42
Time:                        09:05:31   Log-Likelihood:                -592.07
No. Observations:                 196   AIC:                             1188.
Df Residuals:                     194   BIC:                             1195.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     40.3338      1.023     39.416      0.000      38.316      42.352
horsepower    -0.1596      0.009    -17.788      0.000      -0.177      -0.142
==============================================================================
Omnibus:                        8.393   Durbin-Watson:                   1.061
Prob(Omnibus):                  0.015   Jarque-Bera (JB):                8.787
Skew:                           0.516   Prob(JB):                       0.0124
Kurtosis:                       2.899   Cond. No.                         328.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
--------Test Error for 1st order--------
23.36190289258724
```

Now we will try to do it for 2 and 3 order equ and calulate the MSE again. As we can see the error is smaller in the quadratic and cubic regressions. The quadratic(2st) seem to be the best because it only smaller error then the 1st order and is very close to the 3st order error.

```
In [11]: lm2 = smf.ols ('mpg~horsepower + I(horsepower ** 2.0)', traindata).fit()
         preds2 = lm2.predict(data)
         square_error2 = (data['mpg'] - preds2)**2
         print('--------Test Error for 2nd order--------')
         print(np.mean(square_error2[~test]))
```

```
--------Test Error for 2nd order--------
20.2526908583502
```

```
In [12]: lm3 = smf.ols ('mpg~horsepower + I(horsepower ** 2.0) + I(horsepower ** 3.0)', trainda
         preds3 = lm3.predict(data)
         square_error3 = (data['mpg'] - preds3)**2
         print('--------Test Error for 3rd order--------')
         print(np.mean(square_error3[~test]))
```

```
--------Test Error for 3rd order--------
20.32560936587865
```