5.3.2 Leave-One-Out Cross-Validation

May 18, 2018

In [1]:
```
#Thanks to https://github.com/qx0731/ISL_python/blob/master/Chapter_5_sec_3.1_3.4.ipyn

# 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[1]:

	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
```

Now i will use the stats model lib to do a linear regssion. As we can see we get the same error as in the book.

```
In [2]: import statsmodels.formula.api as smf
        lm = smf.ols ('mpg~horsepower',data).fit()

        print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.606
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     599.7
Date:                Wed, 16 May 2018   Prob (F-statistic):           7.03e-81
Time:                        22:11:20   Log-Likelihood:                -1178.7
No. Observations:                 392   AIC:                             2361.
Df Residuals:                     390   BIC:                             2369.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     39.9359      0.717     55.660      0.000      38.525      41.347
horsepower    -0.1578      0.006    -24.489      0.000      -0.171      -0.145
==============================================================================
Omnibus:                       16.432   Durbin-Watson:                   0.920
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               17.305
Skew:                           0.492   Prob(JB):                     0.000175
Kurtosis:                       3.299   Cond. No.                         322.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

To do Leave-One-Out Cross-Validation in python we are going to use some libarys from scikitlean. Therefor we import an and use them here. Sklearn offers to do LinearRegression and we will use that here, as we can see we get the same error as the book.

```
In [3]: from sklearn.model_selection import KFold, cross_val_score
        from sklearn.preprocessing import PolynomialFeatures
        from sklearn.linear_model import LinearRegression
        from sklearn.pipeline import Pipeline
```

```
        x = pd.DataFrame(data.horsepower)
        y = data.mpg

        model = LinearRegression()
        model.fit(x, y)
        print(model.intercept_)
        print(model.coef_)
```

```
39.93586102117047
[-0.15784473]
```

```
In [4]: k_fold = KFold(n_splits=x.shape[0]) # loo use folds equal to # of observations
        test = cross_val_score(model, x, y, cv=k_fold,  scoring = 'neg_mean_squared_error', n_
        print(np.mean(-test))
```

```
24.231513517929226
```

```
In [6]: A = []
        for porder in range(1, 6):
            model = Pipeline([('poly', PolynomialFeatures(degree=porder)), ('linear', LinearReg
            k_fold = KFold(n_splits=x.shape[0]) # loo use folds equal to # of observations
            test = cross_val_score(model, x, y, cv=k_fold,  scoring = 'neg_mean_squared_error'
            A.append(np.mean(-test))

        print(A)
```

```
[24.23151351792922, 19.24821312448969, 19.334984064109666, 19.42443031091358, 19.0332089609506]
```