다음의 리포트는 고려대학교 멀티스케일데이터 마이닝 과목에서 아래의 문제에 대하여 실제 학생이 수행한 숙제이다. 사용한 데이터는 강의에서 제공한 data05_boston 이다. 이 데이터셋은 medv 변수를 다른 변수를 이용하여 설명하고자 하는 데이터이다.

Homework 1 (by 11:59pm 10/14 Sun)

- Analysis of Boston data set
  - sklearn.linear_model.LinearRegression
  - fit, predict, score
  - StatsModels
- (1) Find the best model with <= 3 predictors in terms of RMSE to predict medv in the whole Boston data set
- (2) Find the best mode with <= 3 predictors including log, square, cubic transformation in the whole data set

(1) 전체 보스턴 데이터 셋에서 3 개 이하의 변수를 선택하여 medv 에 대한 선형 회귀 모델을 만들 때, 가장 RMSE 가 낮은 조합을 찾으시오.

(2) 각 변수에 log, 제곱, 세제곱 하는 것을 허용하고, 3 개 이하의 변수를 선택하여 medv 에 대한 선형 회귀 모델을 만들 때, 가장 RMSE 가 낮은 조합을 찾으시오.

[ECE663] 멀티스케일데이터마이닝

Homework 1

1. Find the best model with <= 3 predictors in terms of RMSE to predict medv in the whole Boston data set

1.1. 수행방법

Predictor의 선택은 총 3단계에 걸쳐 이루어졌다. 첫번째 predictor의 선택은 한번에 하나의 feature만 사용하여 회귀를 진행하였을 때 가장 작은 RMSE값을 가지는 feature를 첫번째 predictor로 선택한다. 두번째 predictor의 선택은 모든 feature에 첫번째 predictor의 열을 join하여 회귀를 진행하였을 때 가장 작은 RMSE값을 가지는 feature를 두번째 predictor로 선택하였다. 세번째 predictor는 두번째 predictor를 선택할 때 처럼 첫번째 predictor와 두번째 predictor의 열을 join하여 최소의 RMSE값을 가지는 feature를 세번째 predictor로 선택하였다.

1.2. 결과

lstat, rm, ptratio 이렇게 세개의 feature를 선택했을 때 가장 작은 RMSE값인 5.2의 값을 가졌고 0.68의 socore를 가졌다.

2. Find the best mode with <= 3 predictors including log, square, cubic transformation in the whole data set

2.1. 수행방법

앞서 1.1에서 수행한 방식과 동일한 방식을 사용하되 feature에 log를 씌운 column과 feature를 제곱, 세제곱한 column을 추가하여 작업을 수행하였다.

2.2. 결과

lstat, cubic_rm, rm 이렇게 세개를 선택하였을 때 가장 작은 RMSE 값인 4.8의 값을 가졌고 0.73의 스코어를 가졌다.

3. 코드와 실행결과

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [2]:
```
data=pd.read_csv('data05_boston.csv')
```

In [3]:
```
feauture = data.iloc[:,0:-1]
y = data.iloc[:,[-1]];
```

In [4]:
```
best_index=[0,0,0]
for i in range(3):
    X = feauture.iloc[:,[0] + best_index[0:i]];
    reg = LinearRegression().fit(X,y);
    y_pred = reg.predict(X)
    best_root_mean_squared_error = mean_squared_error(y, y_pred)**0.5
    for ii in range(1,feauture.shape[1]):
        X = feauture.iloc[:,[ii] + best_index[0:i]];
        reg = LinearRegression().fit(X,y);
        y_pred = reg.predict(X)
        root_mean_squared_error = mean_squared_error(y, y_pred)**0.5
        if best_root_mean_squared_error > root_mean_squared_error:
            best_root_mean_squared_error = root_mean_squared_error
            best_index[i] = ii
    print('predictor : '+', '.join(list(key for key in feauture.keys()[best_index[0:i+1]])))
    print('score : %f'%(LinearRegression().fit(feauture.iloc[:,best_index[0:i+1]],y).score(feauture.iloc[:,best_index[0:i+1]],y)))
    print('RMSE : %f'%(best_root_mean_squared_error))
```

```
predictor : lstat
score : 0.544146
RMSE : 6.203464
predictor : lstat, rm
score : 0.638562
RMSE : 5.523809
predictor : lstat, rm, ptratio
score : 0.678624
RMSE : 5.208686
```

In [5]:
```
feauture = data.iloc[:,0:-1]₩
.join(data.loc[:,data.min()!=0].iloc[:,0:-1].add_prefix('log_'))₩
.join((data**2).iloc[:,0:-1].add_prefix('square_'))₩
.join((data**3).iloc[:,0:-1].add_prefix('cubic_'))
```

In [6]:
```
best_index=[0,0,0]
for i in range(3):
    X = feauture.iloc[:,[0] + best_index[0:i]];
    reg = LinearRegression().fit(X,y);
    y_pred = reg.predict(X)
    best_root_mean_squared_error = mean_squared_error(y, y_pred)**0.5
    for ii in range(1,feauture.shape[1]):
        X = feauture.iloc[:,[ii] + best_index[0:i]];
        reg = LinearRegression().fit(X,y);
        y_pred = reg.predict(X)
        root_mean_squared_error = mean_squared_error(y, y_pred)**0.5
        if best_root_mean_squared_error > root_mean_squared_error:
            best_root_mean_squared_error = root_mean_squared_error
            best_index[i] = ii
    print('predictor : '+', '.join(list(key for key in feauture.keys()[best_index[0:i+1]])))
    print('score : %f'%(LinearRegression().fit(feauture.iloc[:,best_index[0:i+1]],y).score(feauture.iloc[:,best_index[0:i+1]],y)))
    print('RMSE : %f'%(best_root_mean_squared_error))
```

```
predictor : lstat
score : 0.544146
RMSE : 6.203464
predictor : lstat, cubic_rm
score : 0.682057
RMSE : 5.180792
predictor : lstat, cubic_rm, rm
score : 0.730109
RMSE : 4.773266
```