6.5.1 Best Subset Selection

May 18, 2018

Here we apply the best subset selection approach to the Hitters data. We wish to predict a baseball player's Salary on the basis of various statistics associated with performance in the previous year.

Now we will download the dataset

```
In [1]: # conventional way to import pandas
        import pandas as pd
        # conventional way to import seaborn
        import seaborn as sns
        # conventional way to import numpy
        import numpy as np

        from sklearn import metrics
        import matplotlib.pyplot as plt

        data = pd.read_csv("https://vincentarelbundock.github.io/Rdatasets/csv/ISLR/Hitters.csv

        data.head()
```

```
Out[1]:                    AtBat  Hits  HmRun  Runs  RBI  Walks  Years  CAtBat  CHits  \
        -Andy Allanson       293    66      1    30   29     14      1     293     66
        -Alan Ashby          315    81      7    24   38     39     14    3449    835
        -Alvin Davis         479   130     18    66   72     76      3    1624    457
        -Andre Dawson        496   141     20    65   78     37     11    5628   1575
        -Andres Galarraga    321    87     10    39   42     30      2     396    101

                           CHmRun  CRuns  CRBI  CWalks League Division  PutOuts  \
        -Andy Allanson          1     30    29      14      A        E      446
        -Alan Ashby            69    321   414     375      N        W      632
        -Alvin Davis           63    224   266     263      A        W      880
        -Andre Dawson         225    828   838     354      N        E      200
        -Andres Galarraga      12     48    46      33      N        E      805

                           Assists  Errors  Salary NewLeague
        -Andy Allanson          33      20     NaN         A
        -Alan Ashby             43      10   475.0         N
        -Alvin Davis            82      14   480.0         A
        -Andre Dawson           11       3   500.0         N
        -Andres Galarraga       40       4    91.5         N
```

After listing the data we can see that some have missing data for their Salary. Next drop all the rows that contain NaN data.

```
In [2]: data = data.dropna()
        data.index.name = 'Player'
        data.head()
```

```
Out[2]:                    AtBat  Hits  HmRun  Runs  RBI  Walks  Years  CAtBat  CHits  \
        Player
        -Alan Ashby          315    81      7    24   38     39     14    3449    835
        -Alvin Davis         479   130     18    66   72     76      3    1624    457
        -Andre Dawson        496   141     20    65   78     37     11    5628   1575
        -Andres Galarraga    321    87     10    39   42     30      2     396    101
        -Alfredo Griffin     594   169      4    74   51     35     11    4408   1133

                           CHmRun  CRuns  CRBI  CWalks League Division  PutOuts  \
        Player
        -Alan Ashby            69    321   414     375      N        W      632
        -Alvin Davis           63    224   266     263      A        W      880
        -Andre Dawson         225    828   838     354      N        E      200
        -Andres Galarraga      12     48    46      33      N        E      805
        -Alfredo Griffin       19    501   336     194      A        W      282

                           Assists  Errors  Salary NewLeague
        Player
        -Alan Ashby             43      10   475.0         N
        -Alvin Davis            82      14   480.0         A
        -Andre Dawson           11       3   500.0         N
        -Andres Galarraga       40       4    91.5         N
        -Alfredo Griffin       421      25   750.0         A
```

Now we performs best Best Subset Selection by identifying the best model that contains a given number of predictors, where best is quantified using RSS. Change the string values into dummy values

```
In [3]: dummieVariables = pd.get_dummies(data[['League', 'Division', 'NewLeague']])
        dummieVariables.info()
        print(dummieVariables.head())

        y = data.Salary
        X_ = data.drop(['Salary', 'League', 'Division', 'NewLeague'], axis=1).astype('float64'

        X = pd.concat([X_, dummieVariables[['League_N', 'Division_W', 'NewLeague_N']]], axis=1
        X.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 263 entries, -Alan Ashby to -Willie Wilson
Data columns (total 6 columns):
League_A       263 non-null uint8
```

```
League_N       263 non-null uint8
Division_E     263 non-null uint8
Division_W     263 non-null uint8
NewLeague_A    263 non-null uint8
NewLeague_N    263 non-null uint8
dtypes: uint8(6)
memory usage: 3.6+ KB
                   League_A  League_N  Division_E  Division_W  NewLeague_A  \
Player
-Alan Ashby               0         1           0           1            0
-Alvin Davis              1         0           0           1            1
-Andre Dawson             0         1           1           0            0
-Andres Galarraga         0         1           1           0            0
-Alfredo Griffin          1         0           0           1            1

                   NewLeague_N
Player
-Alan Ashby                  1
-Alvin Davis                 0
-Andre Dawson                1
-Andres Galarraga            1
-Alfredo Griffin             0
```

```
Out[3]:                    AtBat   Hits  HmRun  Runs   RBI  Walks  Years  CAtBat  \
        Player
        -Alan Ashby        315.0   81.0    7.0  24.0  38.0   39.0   14.0  3449.0
        -Alvin Davis       479.0  130.0   18.0  66.0  72.0   76.0    3.0  1624.0
        -Andre Dawson      496.0  141.0   20.0  65.0  78.0   37.0   11.0  5628.0
        -Andres Galarraga  321.0   87.0   10.0  39.0  42.0   30.0    2.0   396.0
        -Alfredo Griffin   594.0  169.0    4.0  74.0  51.0   35.0   11.0  4408.0

                            CHits  CHmRun  CRuns   CRBI  CWalks  PutOuts  Assists  \
        Player
        -Alan Ashby         835.0    69.0  321.0  414.0   375.0    632.0     43.0
        -Alvin Davis        457.0    63.0  224.0  266.0   263.0    880.0     82.0
        -Andre Dawson      1575.0   225.0  828.0  838.0   354.0    200.0     11.0
        -Andres Galarraga   101.0    12.0   48.0   46.0    33.0    805.0     40.0
        -Alfredo Griffin   1133.0    19.0  501.0  336.0   194.0    282.0    421.0

                           Errors  League_N  Division_W  NewLeague_N
        Player
        -Alan Ashby          10.0         1           1            1
        -Alvin Davis         14.0         0           1            0
        -Andre Dawson         3.0         1           0            1
        -Andres Galarraga     4.0         1           0            1
        -Alfredo Griffin     25.0         0           1            0
```

Next we need to defined a few funktions because there is no libary that can help us do it so we will have to do the funktions. We make use of itertools to get a better speed, as said before this can take a very long time.

```
In [4]: import itertools
        import statsmodels.api as sm

        # Functions found at "https://github.com/qx0731/ISL_python/blob/master/Chapter_6_sec_6
        def CalulateRSS(y, X, predictors_list):
            model = sm.OLS(y, X[list(predictors_list)]).fit()
            RSS   = ((model.predict(X[list(predictors_list)]) - y) ** 2)
            RSS   = RSS.sum()
            return {'Model':model, "RSS":RSS}

        def pickBestPredictorsForModel(y, X, K):
            results = []
            for c in itertools.combinations(X.columns, K):
                results.append(CalulateRSS(y, X, c))
            model_all =  pd.DataFrame(results)

            best_model = model_all.loc[model_all["RSS"].argmin()]
            return best_model
```

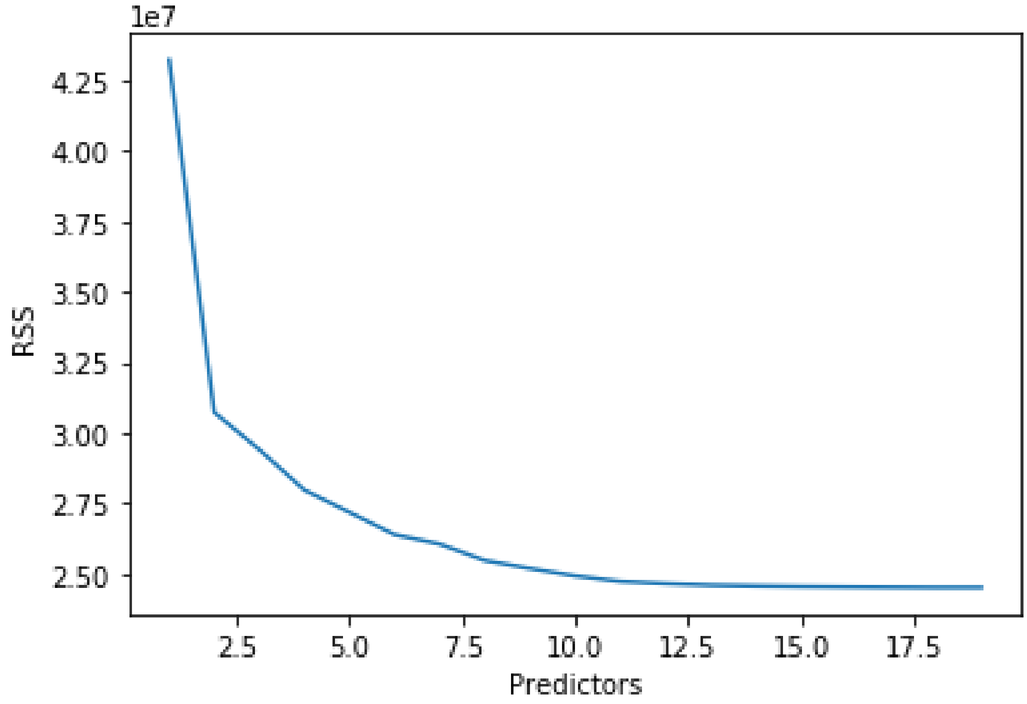
```
C:\Users\au479931\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmodels\compat\pandas
  from pandas.core import datetools
```

As we know the best Best subset selection can take a very long time because. lang tid,fordi... Therefor we only want 3 predictors

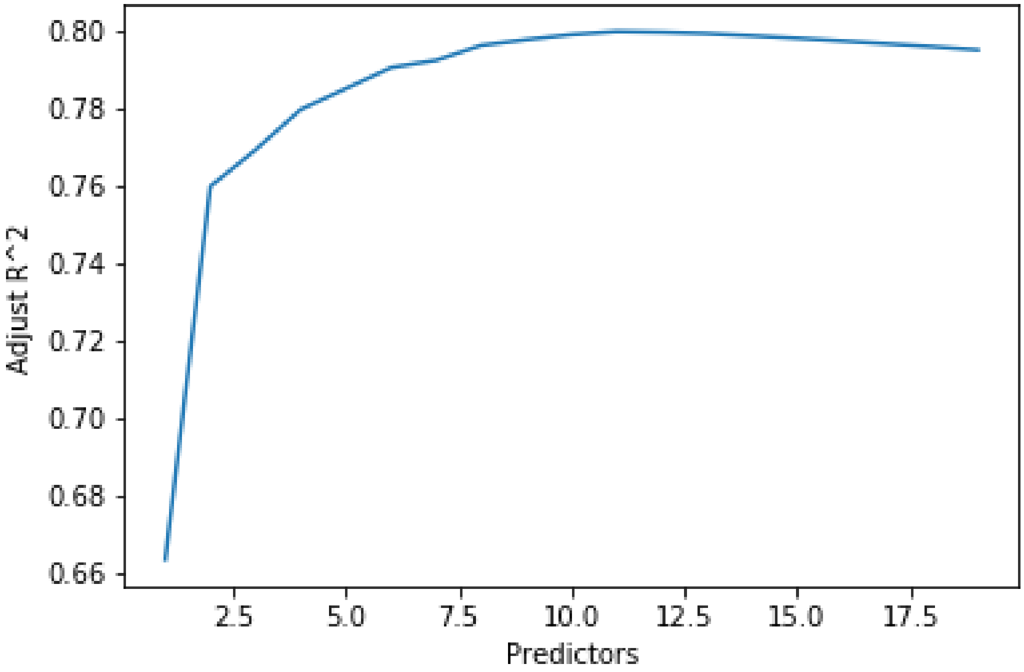
```
In [5]: max_predictors = 19
        models = pd.DataFrame(columns=["RSS", "Model"])
        for i in range(1,(max_predictors+1)):  # for illustration purpuse, I just run for 1 - 
            models.loc[i] = pickBestPredictorsForModel(y, X, i)

        print(models.loc[2, 'Model'].summary() )
        # this summay confirms that the best two variable model contains the variables Hits an
```

```
C:\Users\au479931\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:17


                            OLS Regression Results
==============================================================================
Dep. Variable:                 Salary   R-squared:                       0.761
Model:                            OLS   Adj. R-squared:                  0.760
Method:                 Least Squares   F-statistic:                     416.7
Date:                Fri, 04 May 2018   Prob (F-statistic):           5.80e-82
Time:                        17:35:01   Log-Likelihood:                -1907.6
No. Observations:                 263   AIC:                             3819.
Df Residuals:                     261   BIC:                             3826.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Hits           2.9538      0.261     11.335      0.000       2.441       3.467
CRBI           0.6788      0.066     10.295      0.000       0.549       0.809
==============================================================================
Omnibus:                      117.551   Durbin-Watson:                   1.933
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              654.612
Skew:                           1.729   Prob(JB):                    7.12e-143
Kurtosis:                       9.912   Cond. No.                         5.88
==============================================================================

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

```
In [6]: plt.figure()
        plt.plot(models["RSS"])
        plt.xlabel('Predictors')
        plt.ylabel('RSS')
        plt.show()
```



```
In [7]: rsquared_adj = models.apply(lambda row: row[1].rsquared_adj, axis=1)
In [8]: plt.figure()
        plt.plot(rsquared_adj)
        plt.xlabel('Predictors')
        plt.ylabel('Adjust R^2')
        plt.show()
```


Adjust R^2
Predictors