hw6_answer

February 20, 2019

1 Assignment 6

1.0.1 MACS 30150, Dr. Evans

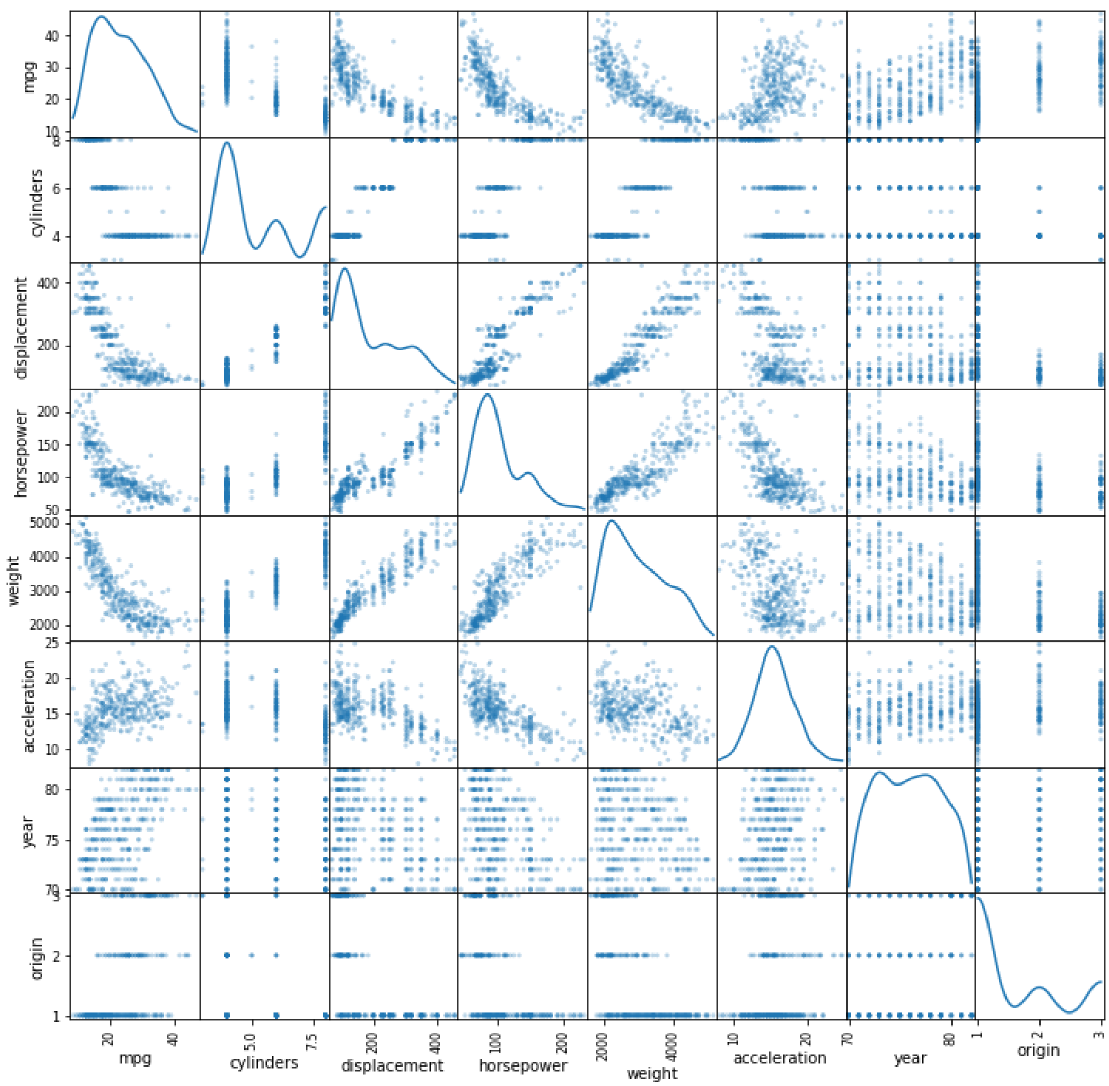
1.0.2 Dongcheng Yang

problem1 (a)

```
In [1]: import pandas as pd

        df1 = pd.read_csv('data\Auto.csv',na_values="?")
        df1.dropna(inplace=True)
        df1.head()

Out[1]:     mpg  cylinders  displacement  horsepower  weight  acceleration  year  \
        0  18.0          8         307.0       130.0    3504          12.0    70
        1  15.0          8         350.0       165.0    3693          11.5    70
        2  18.0          8         318.0       150.0    3436          11.0    70
        3  16.0          8         304.0       150.0    3433          12.0    70
        4  17.0          8         302.0       140.0    3449          10.5    70

           origin                       name
        0       1  chevrolet chevelle malibu
        1       1          buick skylark 320
        2       1         plymouth satellite
        3       1              amc rebel sst
        4       1                ford torino
```

(b)

```
In [3]: from pandas.plotting import scatter_matrix
        import matplotlib.pyplot as plt
        pd.plotting.scatter_matrix(df1, alpha=0.3, figsize=(12, 12),diagonal='kde')
        plt.show()
```



(c)

```
In [4]: df1.corr()
```

```
Out[4]:                    mpg  cylinders  displacement  horsepower    weight  \
        mpg           1.000000  -0.777618     -0.805127   -0.778427 -0.832244
        cylinders    -0.777618   1.000000      0.950823    0.842983  0.897527
        displacement -0.805127   0.950823      1.000000    0.897257  0.932994
        horsepower   -0.778427   0.842983      0.897257    1.000000  0.864538
        weight       -0.832244   0.897527      0.932994    0.864538  1.000000
        acceleration  0.423329  -0.504683     -0.543800   -0.689196 -0.416839
        year          0.580541  -0.345647     -0.369855   -0.416361 -0.309120
        origin        0.565209  -0.568932     -0.614535   -0.455171 -0.585005
```

```
              acceleration      year    origin
mpg               0.423329  0.580541  0.565209
cylinders        -0.504683 -0.345647 -0.568932
displacement     -0.543800 -0.369855 -0.614535
horsepower       -0.689196 -0.416361 -0.455171
weight           -0.416839 -0.309120 -0.585005
acceleration      1.000000  0.290316  0.212746
year              0.290316  1.000000  0.181528
origin            0.212746  0.181528  1.000000
```

(d)

```
In [5]: import statsmodels.api as sm
        df1['const'] = 1
        reg1 = sm.OLS(endog=df1['mpg'], exog=df1[['const', 'cylinders', \
                        'displacement', 'horsepower', 'weight', 'acceleration' \
                        ,'year', 'origin']], missing='drop')
        results = reg1.fit()
        print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.821
Model:                            OLS   Adj. R-squared:                  0.818
Method:                 Least Squares   F-statistic:                     252.4
Date:                Tue, 19 Feb 2019   Prob (F-statistic):          2.04e-139
Time:                        22:40:33   Log-Likelihood:                -1023.5
No. Observations:                 392   AIC:                             2063.
Df Residuals:                     384   BIC:                             2095.
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const          -17.2184      4.644     -3.707      0.000     -26.350      -8.087
cylinders       -0.4934      0.323     -1.526      0.128      -1.129       0.142
displacement     0.0199      0.008      2.647      0.008       0.005       0.035
horsepower      -0.0170      0.014     -1.230      0.220      -0.044       0.010
weight          -0.0065      0.001     -9.929      0.000      -0.008      -0.005
acceleration     0.0806      0.099      0.815      0.415      -0.114       0.275
year             0.7508      0.051     14.729      0.000       0.651       0.851
origin           1.4261      0.278      5.127      0.000       0.879       1.973
==============================================================================
Omnibus:                       31.906   Durbin-Watson:                   1.309
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.100
Skew:                           0.529   Prob(JB):                     2.95e-12
Kurtosis:                       4.460   Cond. No.                     8.59e+04
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.59e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

(1) The coefficients of the variables displacement, weight, year and origin are significant at 1% level.

(2) The coefficients of the variables cylinders, horsepower and acceleration are not statistically significant at the 10% level.

(3) Holding the other varialbles constant, one year later of production of the vehicle is expected to increase 0.7508 miles per gallon on average.

(e)

(1) The variables displacement, horsepower and weight are most likely to have a nonlinear relationship with mpg.

In [7]:
```
df1['displacement_2'] = df1['displacement']**2
df1['horsepower_2'] = df1['horsepower']**2
df1['weight_2'] = df1['weight']**2
df1['acceleration_2'] = df1['acceleration']**2

reg2 = sm.OLS(endog=df1['mpg'], exog=df1[['const', 'cylinders', \
                'displacement', 'displacement_2', 'horsepower', \
                'horsepower_2', 'weight','weight_2', 'acceleration', \
                'acceleration_2','year', 'origin']], missing='drop')
results2 = reg2.fit()
print(results2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.870
Model:                            OLS   Adj. R-squared:                  0.866
Method:                 Least Squares   F-statistic:                     230.2
Date:                Tue, 19 Feb 2019   Prob (F-statistic):          1.75e-160
Time:                        22:47:20   Log-Likelihood:                -962.02
No. Observations:                 392   AIC:                             1948.
Df Residuals:                     380   BIC:                             1996.
Df Model:                          11
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const             20.1084      6.696      3.003      0.003       6.943      33.274
cylinders          0.2519      0.326      0.773      0.440      -0.389       0.893
```

```
displacement      -0.0169      0.020     -0.828      0.408      -0.057       0.023
displacement_2  2.257e-05   3.61e-05      0.626      0.532   -4.83e-05    9.35e-05
horsepower        -0.1635      0.041     -3.971      0.000      -0.244      -0.083
horsepower_2       0.0004      0.000      2.943      0.003       0.000       0.001
weight            -0.0136      0.003     -5.069      0.000      -0.019      -0.008
weight_2        1.514e-06   3.69e-07      4.105      0.000    7.89e-07    2.24e-06
acceleration      -2.0884      0.557     -3.752      0.000      -3.183      -0.994
acceleration_2     0.0576      0.016      3.496      0.001       0.025       0.090
year               0.7810      0.045     17.512      0.000       0.693       0.869
origin             0.6104      0.263      2.320      0.021       0.093       1.128
==============================================================================
Omnibus:                       33.614   Durbin-Watson:                   1.576
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               77.985
Skew:                           0.438   Prob(JB):                     1.16e-17
Kurtosis:                       5.002   Cond. No.                     5.13e+08
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.13e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
```

(2) The adjusted R-squared statistic is 0.866, which is better than 0.818 from part (d).

(3) The coefficient of the displacement variable is no longer statistically significant at 1% level. Its coefficient and squared term coefficient are both not significant at even 10% level.

(4) The p value of the coefficient of cylinders variable increases from 0.128 to 0.408. It is still not significant at the 10% level.

(f)

```
In [13]: results2.predict(exog=[1, 6, 200, 200**2, 100, 100**2, 3100, 3100**2, 15.1, 15.1**2, 99
```

```
Out[13]: 38.73211109801073
```

The predicted miles per gallon is 38.73211109801073.
problem2 (a)

```
In [15]: df2 = pd.DataFrame([[0, 3, 0, 'Red'], [2, 0, 0, 'Red'], [0, 1, 3, 'Red'], \
                              [0, 1, 2, 'Green'], [-1, 0, 1, 'Green'], [1, 1, 1, 'Red']])
         df2.columns = ['X1', 'X2', 'X3', 'Y']
         df2.index+=1
         df2
```

```
Out[15]:    X1  X2  X3      Y
         1   0   3   0    Red
         2   2   0   0    Red
```

```
3   0  1  3    Red
4   0  1  2  Green
5  -1  0  1  Green
6   1  1  1    Red
```

```
In [16]: df2['dist'] = (df2['X1'] ** 2 + df2['X2'] ** 2 + df2['X3'] ** 2) ** 0.5
         df2
```

```
Out[16]:    X1  X2  X3      Y      dist
         1   0   3   0    Red  3.000000
         2   2   0   0    Red  2.000000
         3   0   1   3    Red  3.162278
         4   0   1   2  Green  2.236068
         5  -1   0   1  Green  1.414214
         6   1   1   1    Red  1.732051
```

The euclidean distance for observations 1-6 is 3, 2, $\sqrt{10}$, $\sqrt{5}$, $\sqrt{2}$, $\sqrt{3}$ respectively.

(b)

When K=1, since the closest observation is number 5, the KNN prediction would be Green.

(c)

When K=3, since the three closest observations are number 5, 6 and 2 and both Y of 2 and 6 are red, the KNN prediction would be Red.

(d)

We would expect the best value for K to be large. This is because when K is large, the features of surrounding points in all directions could be captured better. Since the decision boundary boundary is highly non-linear, small K might neglect some important information and be problematic.

(e)

```
In [29]: from sklearn.neighbors import KNeighborsClassifier
         # 0 for red, 1 for green
         X = [[0,3,0],[2,0,0],[0,1,3],[0,1,2],[-1,0,1],[1,1,1]]
         y = ['red','red','red','green','green','red']
         neigh = KNeighborsClassifier(n_neighbors=2, weights='distance')
         cls = neigh.fit(X, y)
         print('The KNN classifier of the test point is',cls.predict([[1,1,1]])[0])
```

```
The KNN classifier of the test point is red
```

problem3 (a)

```
In [17]: import numpy as np
         tf = df1['mpg']>=df1['mpg'].median()
         df1['mpg_high'] = np.where(tf,1,0)

         reg3 = sm.Logit(endog=df1['mpg_high'], exog=df1[['const', 'cylinders', \
                         'displacement', 'horsepower', 'weight', 'acceleration' \
                         ,'year', 'origin']], missing='drop')
         results3 = reg3.fit()
         print(results3.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.200944
         Iterations 9
                           Logit Regression Results
==============================================================================
Dep. Variable:               mpg_high   No. Observations:                  392
Model:                          Logit   Df Residuals:                      384
Method:                           MLE   Df Model:                            7
Date:                Tue, 19 Feb 2019   Pseudo R-squ.:                  0.7101
Time:                        23:00:30   Log-Likelihood:                -78.770
converged:                       True   LL-Null:                       -271.71
                                        LLR p-value:                 2.531e-79
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const          -17.1549      5.764     -2.976      0.003     -28.452      -5.858
cylinders       -0.1626      0.423     -0.384      0.701      -0.992       0.667
displacement     0.0021      0.012      0.174      0.862      -0.021       0.026
horsepower      -0.0410      0.024     -1.718      0.086      -0.088       0.006
weight          -0.0043      0.001     -3.784      0.000      -0.007      -0.002
acceleration     0.0161      0.141      0.114      0.910      -0.261       0.293
year             0.4295      0.075      5.709      0.000       0.282       0.577
origin           0.4773      0.362      1.319      0.187      -0.232       1.187
================================================================================

Possibly complete quasi-separation: A fraction 0.14 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

The coefficients of the variables weight and year are significant at 5% level.

(b)

```
In [20]: from sklearn.model_selection import train_test_split
         Y = df1['mpg_high']
         X = df1[['cylinders', 'displacement','horsepower', 'weight', \
                 'acceleration', 'year', 'origin']]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, \
                    test_size = 0.5, random_state=10)
```

(c)

```
In [21]: from sklearn.linear_model import LogisticRegression
         clf = LogisticRegression(random_state=10, solver='lbfgs', \
             multi_class='multinomial', max_iter=1000).fit(X_train, y_train)
         print(clf.intercept_, clf.coef_[0])

[-12.68077021] [-0.69849056  0.01052742  0.00758743 -0.00366149  0.06907034  0.29951915
  0.08759656]
```

(d)

```
In [22]: y_pred = clf.predict(X_test)
         score = clf.score(X_test, y_test)
         score

Out[22]: 0.8877551020408163

In [24]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_pred)
         print("Confusion matrix:")
         print(cm)

Confusion matrix:
[[86 13]
 [ 9 88]]

In [26]: from sklearn.metrics import classification_report
         print("Classification report:")
         print(classification_report(y_pred, y_test,target_names=['Low mpg', 'High mpg']))

Classification report:
             precision    recall  f1-score   support

    Low mpg       0.87      0.91      0.89        95
   High mpg       0.91      0.87      0.89       101

avg / total       0.89      0.89      0.89       196
```

The above result shows almost the same F1 score, precision and recall. Thus, this model predicts high mpg and low mpg almost equally well.