ProblemSet6_Answer

February 20, 2019

0.0.1 Problem Set 6

MACS 30150, Dr. Evans

Due Wednesday, Feb. 20 at 11:30am

Tianxin Zheng

1. Multiple linear regression

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from pandas.plotting import scatter_matrix
        import statsmodels.api as sm
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import classification_report
```

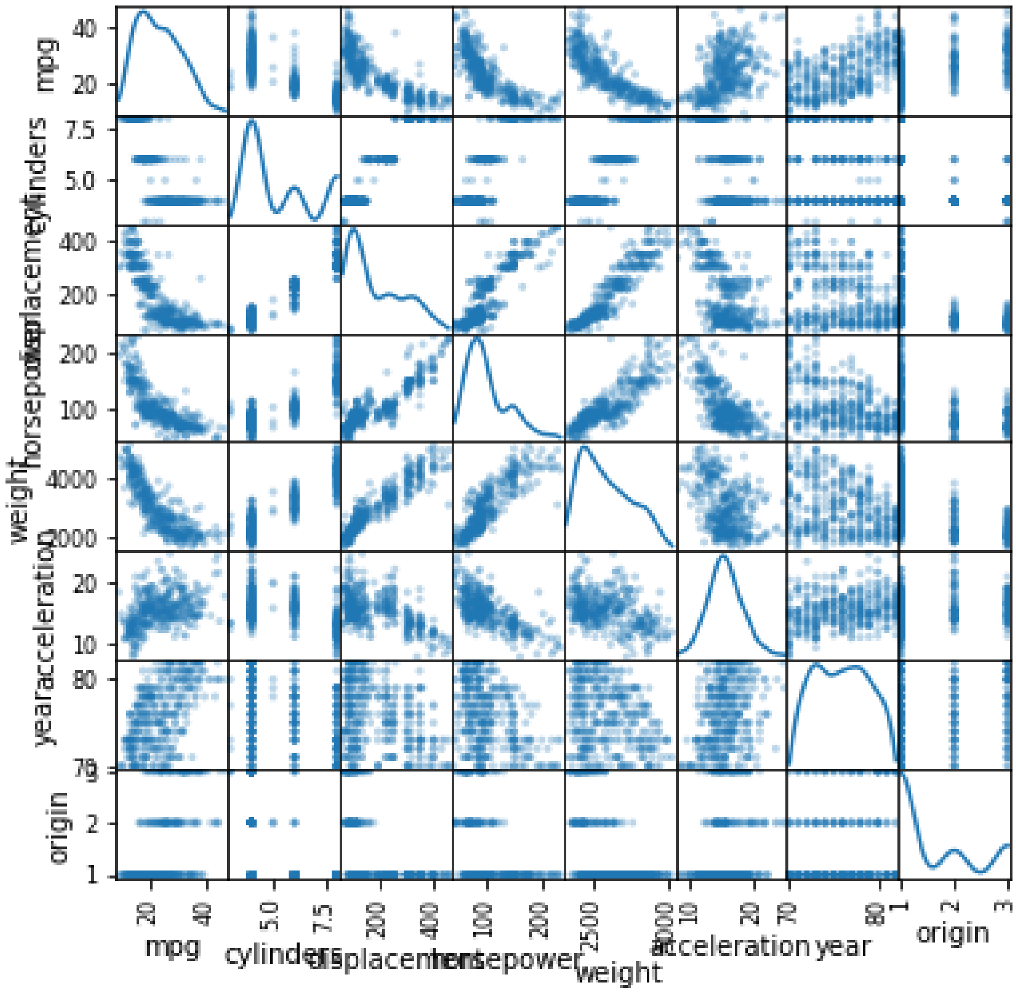
(a)

```
In [2]: df=pd.read_csv("data/Auto.csv", na_values='?')
        df.dropna(inplace=True)
```

(b)

```
In [3]: df_quant = df[['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleratio

In [4]: scatter_matrix(df_quant, alpha=0.3, figsize=(6, 6), diagonal='kde')
        plt.show()
```



(c)

```
In [5]: df_quant.corr()
```

```
Out[5]:                   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 [6]: df['const'] = 1

In [7]: reg1 = sm.OLS(endog=df['mpg'], exog=df[['const', 'cylinders', 'displacement', 'horsepo
        results1 = reg1.fit()
        print(results1.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:                Wed, 20 Feb 2019   Prob (F-statistic):          2.04e-139
Time:                        09:12:02   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.
```

(d.i) The coefficients of 'displacement', 'weight', 'year' and 'origin' are statistically significant at the 1% level.

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

(d.iii) Other variables held constant, with 'vehicle year' increasing one unit, 'miles per gallon' will increase around 0.7508 unit.

(e) From the scatterplot matrix from part (b), three variables that look most likely to have a nonlinear relationship with 'mpg' are 'displacement', 'horsepower' and 'weight'.

(e.i)

```
In [8]: df['displacement^2']=np.square(df['displacement'])
        df['horsepower^2']=np.square(df['horsepower'])
        df['weight^2']=np.square(df['weight'])
        df['acceleration^2']=np.square(df['acceleration'])
```

```
In [9]: reg2 = sm.OLS(endog=df['mpg'], exog=df[['const', 'cylinders', 'displacement', 'horsepo
                                               'displacement^2', 'horsepower^2', 'weight^2',
        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:                Wed, 20 Feb 2019   Prob (F-statistic):         1.75e-160
Time:                        09:12:03   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
horsepower        -0.1635      0.041     -3.971      0.000      -0.244      -0.083
weight            -0.0136      0.003     -5.069      0.000      -0.019      -0.008
acceleration      -2.0884      0.557     -3.752      0.000      -3.183      -0.994
```

```
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
displacement^2    2.257e-05   3.61e-05      0.626      0.532   -4.83e-05    9.35e-05
horsepower^2         0.0004      0.000      2.943      0.003       0.000       0.001
weight^2          1.514e-06   3.69e-07      4.105      0.000    7.89e-07    2.24e-06
acceleration^2       0.0576      0.016      3.496      0.001       0.025       0.090
==============================================================================
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.
```

(e.ii) The adjusted R-squared is 0.866, which is better than 0.818 in the first model.

(e.iii) The coefficients on displacement and its squared term changes from statistically significant at 1% level to nonsignificant at 10% level.

(e.iv) The coefficients on cylinders are not statistically significant at the 10% level.

(f)

```
In [10]: results2.predict(exog=[1, 6, 200, 100, 3100, 15.1, 99, 1, 200**2, 100**2, 3100**2, 15

Out[10]: array([38.7321111])
```

The predicted miles per gallon mpg of a car with 6 cylinders, displacement of 200, horsepower of 100, a weight of 3,100, acceleration of 15.1, model year of 1999, and origin of 1, would be 38.73.

2. Classification problem: KNN by hand and in Python

```
In [11]: table=pd.DataFrame({"X1":[0,2,0,0,-1,1], "X2":[3,0,1,1,0,1],
                             "X3":[0,0,3,2,1,1],"Y":["Red","Red","Red","Green","Green","Red"]})
         table["Eucl.Dist.from X1=X2=X3=0 "]=round(np.sqrt((table["X1"]-0)**2+(table["X2"]-0)**
         table.index+=1
         table
```

Out[11]:

	X1	X2	X3	Y	Eucl.Dist.from X1=X2=X3=0
1	0	3	0	Red	3.000
2	2	0	0	Red	2.000
3	0	1	3	Red	3.162
4	0	1	2	Green	2.236
5	-1	0	1	Green	1.414
6	1	1	1	Red	1.732

(a) The Euclidean distance between each observation and the test point X1 =X2 =X3 =0 shows below:

d1 = 3
d2 = 2
d3 = $\sqrt{10}$
d4 = $\sqrt{5}$
d5 = $\sqrt{2}$
d6 = $\sqrt{3}$

(b) The KNN prediction with K = 1 is Green, because the closest observation to X1=X2=X3=0 is the 5th observation, which is green.

(c) The KNN prediction with K = 3 is Red, because the nearest 3 observations to X1=X2=X3=0 are respectively observation 2,5,6. Observation 2 and 6 are red and observation 5 is green, so the prediction is red.

(d) If the Bayes (optimal) decision boundary in this problem is highly nonlinear, then we would expect the best value for K to be large. Since the boundary in this problem is highly nonlinear, large K can capture the feature of surrounding points in all directions better. We could better approximate the optimal decision boundary by increasing K.

(e)

```
In [12]: Points = np.array([[0,3,0], [2,0,0], [0,1,3], [0,1,2], [-1,0,1], [1,1,1]])
         ys = np.array(['red','red','red','green','green','red']).reshape(-1, 1)
         test_point = np.array([1,1,1]).reshape(1,-1)
         knn = KNeighborsClassifier(n_neighbors=2,weights='distance').fit(Points, ys)
         print("The KNN classifier of the test point X1 = X2 = X3 = 1 with K = 2 is {}.".forma
```

```
The KNN classifier of the test point X1 = X2 = X3 = 1 with K = 2 is red.


/Users/tianxinzheng/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:4: DataConvers
  after removing the cwd from sys.path.
```

3. Multivariable logistic (logit) regression

```
In [13]: df['mpg_high']=np.where(df['mpg']>np.median(df['mpg']), 1, 0)
```

(a)

```
In [14]: reg3 = sm.Logit(endog=df['mpg_high'], exog=df[['const', 'cylinders', 'displacement',
         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:                Wed, 20 Feb 2019   Pseudo R-squ.:                  0.7101
Time:                        09:12:03   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.
```

Coefficients of weight and year are statistically significant at the 5% level.

(b)

```
In [15]: Y = df['mpg_high']
         X = df[['cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'year',
         X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.5, random_sta
```

(c)

```
In [16]: reg4 = LogisticRegression(random_state=10, solver='lbfgs', multi_class='multinomial',r
         print('The estimated intercept is: ', reg4.intercept_)
         print('The estimated coefficient is: ', reg4.coef_[0])
```

```
The estimated intercept is:  [-0.10026869]
The estimated coefficient is:  [-0.65773519  0.00857663 -0.01766136 -0.00257161 -0.10958242  0
 -0.04697382]
```

```
In [17]: coef = pd.DataFrame({"coefficient":['constant','cylinders','displacement','horsepower
                              "estimate":list(reg4.intercept_) + list(reg4.coef_[0])})
         print(coef)
```

```
    coefficient  estimate
0      constant -0.100269
1     cylinders -0.657735
2  displacement  0.008577
3    horsepower -0.017661
4        weight -0.002572
5  acceleration -0.109582
6          year  0.167356
7        origin -0.046974
```

(d)

```
In [18]: y_pred = reg4.predict(X_test)
         cm = confusion_matrix(y_test, y_pred)
         print("Confusion matrix:")
         print(cm)
```

```
Confusion matrix:
[[86 13]
 [12 85]]
```

```
In [19]: print("Classification report:")
         print(classification_report(y_test, y_pred))
```

```
Classification report:
             precision    recall  f1-score   support

          0       0.88      0.87      0.87        99
          1       0.87      0.88      0.87        97

avg / total       0.87      0.87      0.87       196
```

The F1-scores are same. This model predicts equally well on low mpg and high mpg.