TrachtPS6

February 19, 2019

1 Problem Set 6

1.1 Daniel Tracht

1.2 Problem 1

1.2.1 Part a

We wish to import the data from the Auto.csv file, and replace the values that seem to be out of place.

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

        autos = pd.read_csv('data/Auto.csv')

        autos.dtypes
```

```
Out[1]: mpg             float64
        cylinders         int64
        displacement    float64
        horsepower       object
        weight            int64
        acceleration    float64
        year              int64
        origin            int64
        name             object
        dtype: object
```

We expect horesepower to be a numeric.

```
In [2]: autos.sort_values(by="horsepower").tail()
```

```
Out[2]:       mpg  cylinders  displacement horsepower  weight  acceleration  year  \
        336  23.6          4         140.0          ?    2905          14.3    80
        126  21.0          6         200.0          ?    2875          17.0    74
        354  34.5          4         100.0          ?    2320          15.8    81
        32   25.0          4          98.0          ?    2046          19.0    71
        330  40.9          4          85.0          ?    1835          17.3    80
```

```
     origin                  name
336       1   ford mustang cobra
126       1         ford maverick
354       2           renault 18i
32        1            ford pinto
330       2  renault lecar deluxe
```

We have a question mark for our not applicable values. Let's reimport with the right option.

```
In [3]: autos = pd.read_csv('data/Auto.csv', na_values="?")
```

1.2.2 Part b

We wish to produce a scatterplot matrix which includes all of the quantatative variables:

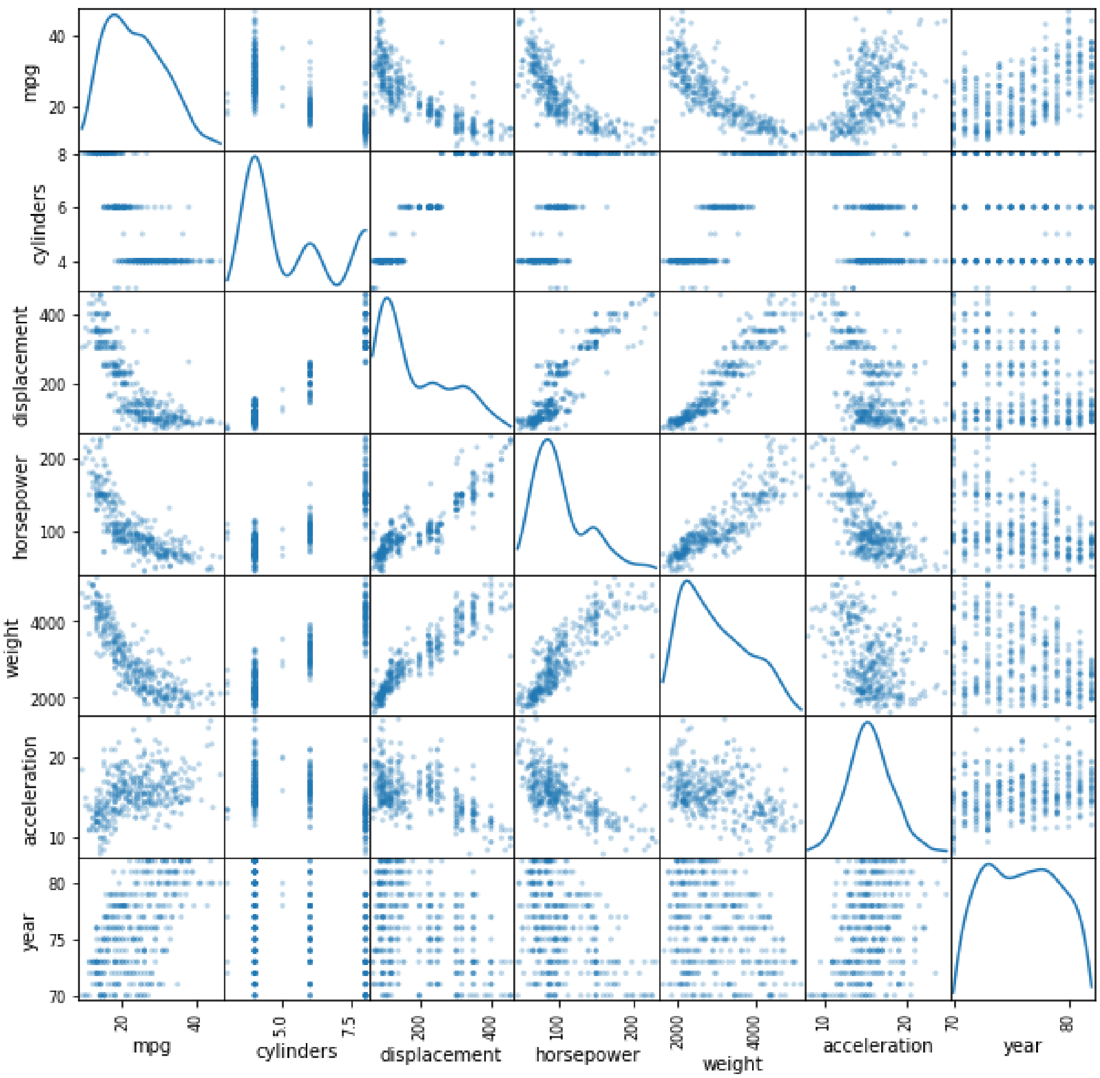
```
In [5]: from pandas.plotting import scatter_matrix

        # making a data frame of the quantative variables
        # while the origin variable is stored as a numeric in our data, it is a categor
        # autos is a pretty well known data set
        # 1 is USA, 2 is Europe, 3 is Japan
        #df_quant = autos[["mpg", "cylinders", "displacement", "horsepower", "weight",
        #                  "acceleration", "year", "origin"]]
        # ^ would be the line you asked for
        df_quant = autos[["mpg", "cylinders", "displacement", "horsepower",
                          "weight", "acceleration", "year"]]

        #scatter_matrix(df_quant, alpha=0.3, figsize=(6, 6), diagonal="kde")
        # ^ would be the line you asked for, but  6, 6 was just too small to see anything
        scatter_matrix(df_quant, alpha=0.3, figsize=(10, 10), diagonal="kde")
```

```
Out[5]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBAA86908>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFCDD710>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFD03C88>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFD360B8>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFD5D630>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFD84BA8>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFDB6160>],
               [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFE1C710>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFE1C748>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFE78208>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFE9D780>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFEC6CF8>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFF782B0>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFF9D828>],
               [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFFC6DA0>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BBFFF7358>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC00218D0>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0048E48>,
```

```
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC04D8400>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0502978>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0529EF0>],
 [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC055B4A8>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0581A20>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC05ABF98>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC05DB550>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0602AC8>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0636080>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC065C5F8>],
 [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0686B70>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC06B6128>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC06DF6A0>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0706C18>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC07371D0>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC075F748>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0787CC0>],
 [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC07B7278>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC07DE7F0>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0806D68>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0837320>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0860898>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC088AE10>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC08B93C8>],
 [<matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC08E1940>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC090BEB8>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC0939470>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC09649E8>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC098DF60>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC09BC518>,
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000020BC09E2A90>]],
dtype=object)
```



1.2.3 Part c

We wish to compute the correlation matrix for the quantative variables:

```
In [6]: df_quant.corr()
```

```
Out[6]:                    mpg  cylinders  displacement  horsepower    weight  \
        mpg           1.000000  -0.776260     -0.804443   -0.778427 -0.831739
        cylinders    -0.776260   1.000000      0.950920    0.842983  0.897017
        displacement -0.804443   0.950920      1.000000    0.897257  0.933104
        horsepower   -0.778427   0.842983      0.897257    1.000000  0.864538
        weight       -0.831739   0.897017      0.933104    0.864538  1.000000
        acceleration  0.422297  -0.504061     -0.544162   -0.689196 -0.419502
        year          0.581469  -0.346717     -0.369804   -0.416361 -0.307900
```

```
                acceleration      year
mpg                 0.422297  0.581469
cylinders          -0.504061 -0.346717
displacement       -0.544162 -0.369804
horsepower         -0.689196 -0.416361
weight             -0.419502 -0.307900
acceleration        1.000000  0.282901
year                0.282901  1.000000
```

1.2.4 Part d

We wish to estimate a multiple linear regression model:

```
In [7]: import statsmodels.api as sm

        # definning a column of 1s as the constant
        autos["const"] = 1

        # making a dataframe of the exogenous variables

        exog_origin = autos[["const", "cylinders", "displacement", "horsepower", "weight",
                             "acceleration", "year", "origin"]]
        # ^ the line with origin as a quantative, not categorical variable

        # We have to make some dummies first
        origins = pd.get_dummies(autos["origin"], drop_first=True)
        autos = pd.concat([autos, origins], axis=1)
        autos.rename(columns={2: "Europe", 3: "Japan"}, inplace=True)

        exog = autos[["const", "cylinders", "displacement", "horsepower", "weight",
                      "acceleration", "year", "Europe", "Japan"]]

        # running the regression
        reg1_origin = sm.OLS(endog=autos['mpg'], exog=exog_origin, missing='drop')
        results1_origin = reg1_origin.fit()
        print(results1_origin.summary())

        reg1 = sm.OLS(endog=autos['mpg'], exog=exog, missing='drop')
        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:                Tue, 19 Feb 2019   Prob (F-statistic):          2.04e-139
```

```
Time:                        15:26:27   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.
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.824
Model:                            OLS   Adj. R-squared:                  0.821
Method:                 Least Squares   F-statistic:                     224.5
Date:                Tue, 19 Feb 2019   Prob (F-statistic):          1.79e-139
Time:                        15:26:27   Log-Likelihood:                -1020.5
No. Observations:                 392   AIC:                             2059.
Df Residuals:                     383   BIC:                             2095.
Df Model:                           8
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const          -17.9546      4.677     -3.839      0.000     -27.150      -8.759
cylinders       -0.4897      0.321     -1.524      0.128      -1.121       0.142
displacement     0.0240      0.008      3.133      0.002       0.009       0.039
horsepower      -0.0182      0.014     -1.326      0.185      -0.045       0.009
weight          -0.0067      0.001    -10.243      0.000      -0.008      -0.005
acceleration     0.0791      0.098      0.805      0.421      -0.114       0.272
year             0.7770      0.052     15.005      0.000       0.675       0.879
```

```
Europe             2.6300      0.566      4.643      0.000       1.516       3.744
Japan              2.8532      0.553      5.162      0.000       1.766       3.940
==============================================================================
Omnibus:                       23.395   Durbin-Watson:                   1.291
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               34.452
Skew:                           0.444   Prob(JB):                     3.30e-08
Kurtosis:                       4.150   Cond. No.                     8.70e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.7e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

If origin is included in the regression as a categorical variable, we find that β_0, β_2, β_4, β_6, and β_7 are statistically significant at the 1% level. We find that β_1, β_3, and β_5 are not statistically significant at the 10% level. In words, an automobile model that is 1 year newer would have 0.7508 more miles per gallon, ceteris paribus.

If origin is included as a categorical variable, we find that β_0, β_2, β_4, β_6, as well as the two coefficients for the origin dummies (being from Europe or Japan relative to the United States) are statistically significant at the 1% level. We find that β_1, β_3, and β_5 are not statistically significant at the 10% level. In words, an automobile model that is 1 year newer would have 0.7770 more miles per gallon, ceteris paribus.

1.2.5 Part e

From the scatterplot, it seems that displacement, horsepower, and weight are most likely to have a non-linear relationship with mpg_i. We wish to estimate a linear regression with a squared term to these three as well as $acceleration_i$:

```
In [8]: # Generating the square terms
        autos["disp_sq"] = autos["displacement"]**2
        autos["horses_sq"] = autos["horsepower"]**2
        autos["weight_sq"] = autos["weight"]**2
        autos["accel_sq"] = autos["acceleration"]**2

        # taking the square terms into a data frame and joining them with the others
        exog_sq = autos[["disp_sq", "horses_sq", "weight_sq", "accel_sq"]]
        exog2_origin = pd.concat([exog_origin, exog_sq], axis=1)
        exog2 = pd.concat([exog, exog_sq], axis=1)

        # running the regression with origin as quantatative
        reg2_origin = sm.OLS(endog=autos['mpg'], exog=exog2_origin, missing='drop')
        results2_origin = reg2_origin.fit()
        print(results2_origin.summary())

        # running the regression with origin as categorical
```

```
reg2 = sm.OLS(endog=autos['mpg'], exog=exog2, 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:                        15:26:27   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
disp_sq       2.257e-05   3.61e-05      0.626      0.532   -4.83e-05    9.35e-05
horses_sq        0.0004      0.000      2.943      0.003       0.000       0.001
weight_sq     1.514e-06   3.69e-07      4.105      0.000    7.89e-07    2.24e-06
accel_sq         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.
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.870
Model:                            OLS   Adj. R-squared:                  0.866
Method:                 Least Squares   F-statistic:                     210.7
Date:                Tue, 19 Feb 2019   Prob (F-statistic):         2.25e-159
Time:                        15:26:27   Log-Likelihood:                -961.83
```

```
No. Observations:                 392   AIC:                             1950.
Df Residuals:                     379   BIC:                             2001.
Df Model:                          12
Covariance Type:            nonrobust
=================================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const          19.8341      6.848      2.896      0.004       6.369      33.299
cylinders       0.2219      0.330      0.673      0.501      -0.427       0.871
displacement   -0.0130      0.021     -0.605      0.546      -0.055       0.029
horsepower     -0.1611      0.041     -3.892      0.000      -0.242      -0.080
weight         -0.0140      0.003     -5.070      0.000      -0.019      -0.009
acceleration   -2.0281      0.566     -3.585      0.000      -3.140      -0.916
year            0.7877      0.046     17.134      0.000       0.697       0.878
Europe          0.9074      0.550      1.650      0.100      -0.174       1.989
Japan           1.2505      0.529      2.365      0.019       0.211       2.290
disp_sq      1.796e-05   3.69e-05      0.487      0.626   -5.45e-05    9.04e-05
horses_sq       0.0004      0.000      2.899      0.004       0.000       0.001
weight_sq    1.554e-06   3.75e-07      4.147      0.000    8.17e-07    2.29e-06
accel_sq        0.0559      0.017      3.349      0.001       0.023       0.089
==============================================================================
Omnibus:                       32.033   Durbin-Watson:                   1.571
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               73.327
Skew:                           0.420   Prob(JB):                     1.19e-16
Kurtosis:                       4.945   Cond. No.                     5.24e+08
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.24e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
```

With origin included as a quantatative variable, the adjusted R^2 statistic is now 0.866, which is better than the 0.818 from our previous regression. The statistical significance of the coefficient on *displacement* has reduced dramatically between the two regressions, and its square's significance is not good either. The statistical signficiance of the coefficient for *cylinders* has also fallen dramatically.

With origin included as a categorical variable, the R^2 statistic is now 0.866, which is better than the 0.821 from our previous regression. As with origin as a quantatative variable, the statistical significance of the coefficient on *displacement* has reduced dramatically between the two regressions, and its square's significance is not good either. The statistical signficiance of the coefficient for *cylinders* has also fallen dramatically.

1.2.6 Part f

We wish to generated the predicted miles per gallon for a car with 6 cylinders, a displacement of 200, horsepower of 100, a weight of 3100, accleration of 15.1, a model year of 1999, and an origin of 1 using our regression including the square terms:

```
In [9]: # defining the parameters
        const = 1
        cylinders = 6
        displacement = 200
        horsepower = 100
        weight = 3100
        acceleration = 15.1
        # note the 2 digit year, not 4 digit
        year = 99
        origin = 1
        disp_sq = 200**2
        horses_sq = 100**2
        weight_sq = 3100**2
        accel_sq = 15.1**2
        europe = 0
        japan = 0

        predictors_origin = [const, cylinders, displacement, horsepower, weight, acceleration,
                             year, origin, disp_sq, horses_sq, weight_sq, accel_sq]
        predictors = [const, cylinders, displacement, horsepower, weight, acceleration, year,
                      europe, japan, disp_sq, horses_sq, weight_sq, accel_sq]

        print(results2_origin.predict(exog=predictors_origin))
        print(results2.predict(exog=predictors))
```

```
[38.7321111]
[38.83998021]
```

With origin as a quantative variable, our model predicts that such a car would get about 38.73 miles per gallon. With origin as a categorical variable, out model predicts that such a car would get about 38.84 miles per gallon.

1.3 Problem 2

1.3.1 Part a

For this, we wish to compute the Euclidean distance between each observation and the origin. For observation 1, this is 3. For observation 2, this is 2. sqrt(10), about 3.16 sqrt(5), about 2.23 sqrt(2), about 1.41 sqrt(3), about 1.73

1.3.2 Part b

For this, we wish to learn what the KNN prediction for the origin is when $K = 1$. This is Green. As we computed above, the closest observation to the origin is observation 5. It's value is Green. So when $K = 1$, we would classify the origin as Green as well.

1.3.3 Part c

For this, we wish to learn what the KNN prediction for the origin is when $K = 3$. This is Red. As we computed above, the closest three observations are observations 5, 6, and 2. While observation 5 is Green, both Observations 6 and 2 are Red. Thus, when $K = 3$, we would classify the origin as Red.

1.3.4 Part d

If the Bayes optimal decision boundary in the problem is highly non-linear, then we would expect that the best value of K would be ???

1.3.5 Part e

For this, we wish to use Python to estimate the KNN classifer of the test point $X_1 = X_2 = X_3 = 1$ with $K = 2$

```
In [10]: from sklearn import neighbors

         # Creating our data
         # Observation 7 added as our target, with Green Y randomly
         df = pd.DataFrame({"X_1": [0, 2, 0, 0, -1, 1, 1],
                            "X_2": [3, 0, 1, 1, 0, 1, 1],
                            "X_3": [0, 0, 3, 2, 1, 1, 1],
                            "Y": ["R", "R", "R", "G", "G", "R", "G"]},
                           index=[1,2,3,4,5,6,7])
         df_train = df[0:6]
         X_train = df_train[["X_1", "X_2", "X_3"]]
         y_train = df_train["Y"]

         df_test = df[6:7]
         X_test = df_test[["X_1", "X_2", "X_3"]]
         y_test = df_test["Y"]

         knn = neighbors.KNeighborsClassifier(n_neighbors=3)
         knn.fit(X_train, y_train).score(X_test, y_test)

Out[10]: 0.0
```

For $K = 3$, our dummy test of Green was wrong, so it must be that it is Red. I'm sure that there is a better way to do this, but I was following the code presented it class.

1.4 Problem 3

For this problem, we want to analyize the same auto data as in Problem 1 using a multivariable logisitic regression. First, we need to create a binary variable to study.

```
In [11]: # find median of column
         median_mpg = autos["mpg"].median()
```

```
# begin new variable at 0
autos["mpg_high"] = 0
# replace values for which mpg is greater than the median
autos.loc[(autos["mpg"] > median_mpg), "mpg_high"] = 1
```

1.4.1 Part a

For this problem, we wish to estimate a logistic regression of our new binary variable on the renamed regressors from Problem 1.

```
In [12]: import numpy as np
         import statsmodels.api as sm

         # Renaming columns to desired names
         autos.rename(columns={"cylinders": "cyl", "displacement": "dspl",
                               "horsepower" : "hpwr", "weight" : "wgt",
                               "acceleration" : "accl", "year" : "yr", "origin" : "orgn"}, inpl

         # Dropping na values for logit analysis
         autos.dropna(inplace=True)

         # Create matrices of X and y values
         # Useful when splitting in the next part
         X_origin = autos[["cyl", "dspl", "hpwr", "wgt", "accl", "yr", "orgn"]].values
         X = autos[["cyl", "dspl", "hpwr", "wgt", "accl", "yr", "Europe", "Japan"]].values
         y = autos["mpg_high"].values

         # Adding a constant to our X matrix
         num_obs = X.shape[0]
         const_vec = np.ones(num_obs).reshape((num_obs, 1))
         Xconst_origin = np.hstack((const_vec, X_origin))
         Xconst = np.hstack((const_vec, X))

         # Running the model using statsmodel.api
         LogitModel_origin = sm.Logit(y, Xconst_origin)
         LogitReg_origin = LogitModel_origin.fit()
         print(LogitReg_origin.summary())

         LogitModel = sm.Logit(y, Xconst)
         LogitReg = LogitModel.fit()
         print(LogitReg.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.189320
         Iterations 9
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  392
```

```
Model:                          Logit   Df Residuals:                      384
Method:                           MLE   Df Model:                            7
Date:                Tue, 19 Feb 2019   Pseudo R-squ.:                  0.7265
Time:                        15:26:29   Log-Likelihood:                -74.213
converged:                       True   LL-Null:                       -271.30
                                        LLR p-value:                 4.235e-81
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -22.7150      6.140     -3.700      0.000     -34.749     -10.681
x1            -0.0633      0.437     -0.145      0.885      -0.919       0.792
x2            -0.0002      0.013     -0.017      0.987      -0.026       0.025
x3            -0.0399      0.025     -1.618      0.106      -0.088       0.008
x4            -0.0048      0.001     -3.935      0.000      -0.007      -0.002
x5            -0.0178      0.141     -0.126      0.899      -0.294       0.258
x6             0.5196      0.084      6.169      0.000       0.355       0.685
x7             0.4990      0.360      1.385      0.166      -0.207       1.205
==============================================================================

Possibly complete quasi-separation: A fraction 0.18 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
Optimization terminated successfully.
         Current function value: 0.183983
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  392
Model:                          Logit   Df Residuals:                      383
Method:                           MLE   Df Model:                            8
Date:                Tue, 19 Feb 2019   Pseudo R-squ.:                  0.7342
Time:                        15:26:29   Log-Likelihood:                -72.121
converged:                       True   LL-Null:                       -271.30
                                        LLR p-value:                 4.205e-81
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -24.6273      6.285     -3.919      0.000     -36.945     -12.310
x1            -0.1998      0.456     -0.438      0.661      -1.094       0.695
x2             0.0126      0.015      0.850      0.396      -0.016       0.042
x3            -0.0394      0.025     -1.565      0.118      -0.089       0.010
x4            -0.0061      0.001     -4.221      0.000      -0.009      -0.003
x5            -0.0211      0.142     -0.149      0.882      -0.300       0.257
x6             0.5770      0.094      6.161      0.000       0.393       0.761
x7             1.7950      0.761      2.357      0.018       0.303       3.287
x8             1.1271      0.719      1.568      0.117      -0.282       2.536
==============================================================================
```

```
Possibly complete quasi-separation: A fraction 0.19 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

When including origin as a quantatative variable, we find that β_0, β_4, and β_6, which are the coefficients for the constant, the weight, and the year, are statistically significant at the 5 percent level.

When incldSuing origin as a categorical variable, we find that β_0, β_4, β_6, and β_7, which are the coefficients for the constant, the weight, the year, and being from Japan, are statistically significant at the 5 percent level.

1.4.2 Part b

We wish to randomly and equally divide the data in a training set and a test set.

```
In [13]: # It seems that train_test_split is is model_selection, not cross_validation
         from sklearn.model_selection import train_test_split
         # following from the problem set
         X_train, X_test, y_train, y_test = \
             train_test_split(X, y, test_size = 0.5, random_state = 10)
         X_train_origin, X_test_origin, y_train_origin, y_test_origin = \
             train_test_split(X_origin, y, test_size = 0.5, random_state = 10)
```

1.4.3 Part c

We wish to estimate a logistic regression on the training set using the given method

```
In [14]: from sklearn.linear_model import LogisticRegression

         # Define the logistic regression
         LogReg_origin = LogisticRegression(random_state=0, solver="lbfgs", max_iter=1000)
         train_origin = LogReg_origin.fit(X_train_origin, y_train_origin)
         print("With origin as a quantatative variable:")
         print("Intercept:", train_origin.intercept_)
         print("Betas 1 through 6", train_origin.coef_[0,0:6])
         print("Beta 7", train_origin.coef_[0,6:7])

         LogReg = LogisticRegression(random_state=0, solver="lbfgs", max_iter=1000)
         train = LogReg.fit(X_train, y_train)
         print("With origin as a categorical variable:")
         print("Intercept:", train.intercept_)
         print("Betas 1 through 6", train.coef_[0,0:6])
         print("Betas 7 and 8", train.coef_[0,6:8])
```

```
With origin as a quantatative variable:
Intercept: [-30.29184382]
Betas 1 through 6 [-0.99645974  0.02130091  0.01681134 -0.00809184  0.14283266  0.65319314]
Beta 7 [0.41513925]
```

```
With origin as a categorical variable:
Intercept: [-29.77528723]
Betas 1 through 6 [-0.95411633  0.02123185  0.01677392 -0.00829327  0.1353172   0.65853586]
Betas 7 and 8 [0.66594604 0.33380007]
```

1.4.4 Part d

We wish to create predicted values for our test data using our training set and calculate a confusion matrix and classification report.

```
In [15]: from sklearn.metrics import confusion_matrix
         from sklearn.metrics import classification_report

         # Predict new values from our logistic regressions
         y_pred_origin = LogReg_origin.predict(X_test_origin)
         y_pred = LogReg.predict(X_test)

         confusion_matrix_origin = confusion_matrix(y_test_origin, y_pred_origin)
         print(confusion_matrix_origin)
         print(classification_report(y_test_origin, y_pred_origin))


         confusion_matrix = confusion_matrix(y_test, y_pred)
         print(confusion_matrix)
         print(classification_report(y_test, y_pred))
```

```
[[91 14]
 [ 7 84]]
              precision    recall  f1-score   support

           0       0.93      0.87      0.90       105
           1       0.86      0.92      0.89        91

   micro avg       0.89      0.89      0.89       196
   macro avg       0.89      0.89      0.89       196
weighted avg       0.90      0.89      0.89       196

[[90 15]
 [ 7 84]]
              precision    recall  f1-score   support

           0       0.93      0.86      0.89       105
           1       0.85      0.92      0.88        91

   micro avg       0.89      0.89      0.89       196
   macro avg       0.89      0.89      0.89       196
weighted avg       0.89      0.89      0.89       196
```

When including origin as a quantatative variable, we are able to correctly classify 91 out of 105 low-mpg cars, and 84 out of 91 high-mpg cars. Respectively these shares are 87 and 92 percent. Of the 98 cars that we classify as low-mpg cars, only 91 actually are. Of the 108 cars we classify as high-mpg cars, only 84 actually are. Respectively, these shares are 93 and 86 percent. Using the average of these, as reflected in the f1-score column of the classification report, we might conclude that our model is better at classifying low-mpg cars than high-mpg cars.

When including origin as a categorical variable, we are able to correctly classify 90 out of 105 low-mpg cars, and 84 out of 91 high-mpg cars. Respectively these shares are 86 and 92 percent. Of the 97 cars that we classify as low-mpg cars, only 90 actually are. Of the 109 cars we classify as high-mpg cars, only 84 actually are. Respectively, these shares are 93 and 85 percent. Using the average of these, as reflected in the f1-score column of the classification report, we might conclude that our model is better at classifying low-mpg cars than high-mpg cars.