Yuming Liu PS6

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
```

Problem 1(a)

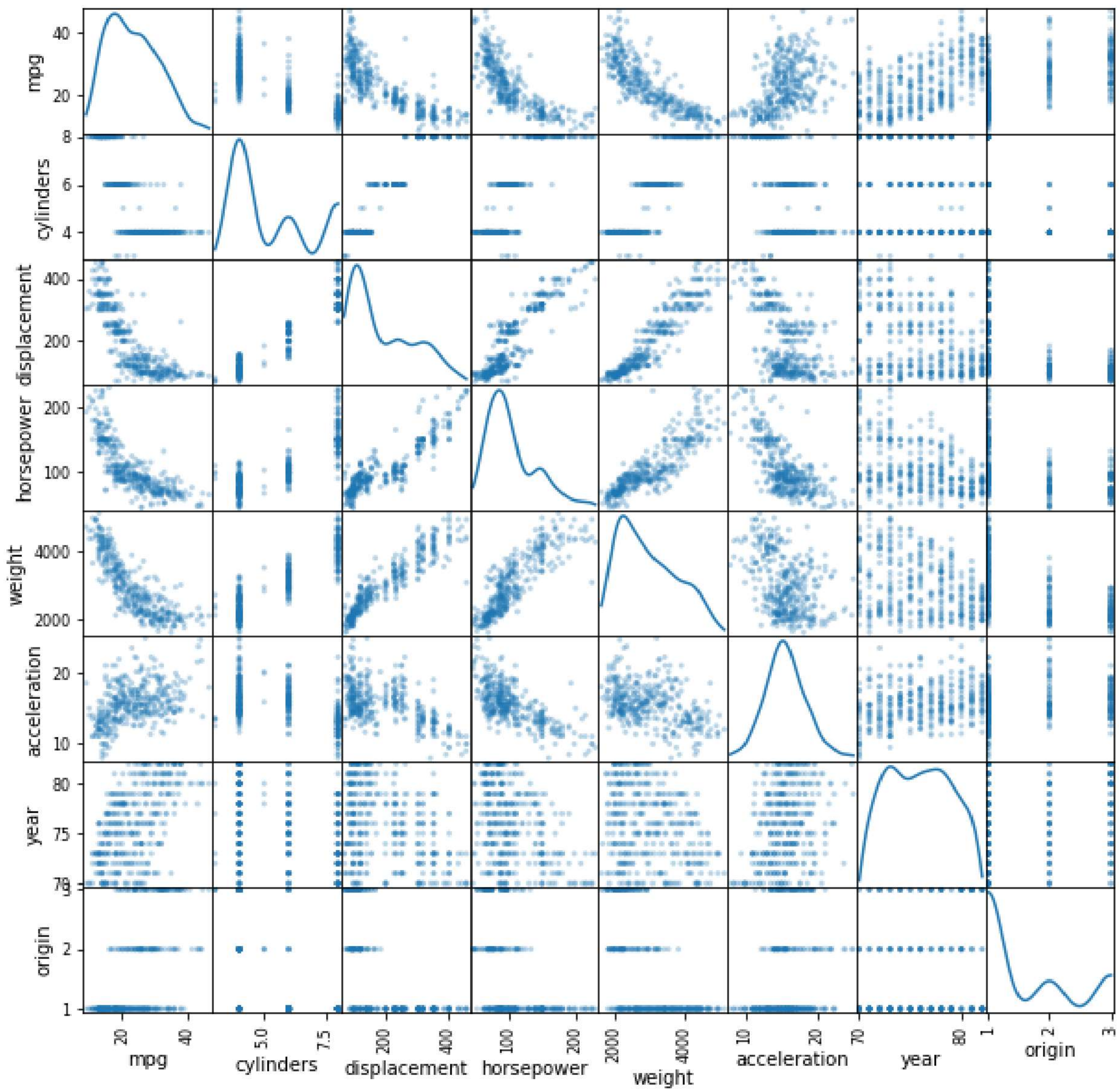
In [2]:
```
df = pd.read_csv("data/Auto.csv", na_values='?')
df.dropna(inplace = True)
df.head()
```

Out[2]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	1	ford torino

Problem 1(b)

In [3]:
```
df_quant = df[['mpg','cylinders', 'displacement', 'horsepower', 'weight', 'acc
eleration', 'year', 'origin']]
scatter_matrix(df_quant, alpha=0.3, ax=None, figsize=(10, 10), diagonal='kde')
plt.show()
```



Problem 1(c)

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

Out[4]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year
mpg	1.000000	-0.777618	-0.805127	-0.778427	-0.832244	0.423329	0.580541
cylinders	-0.777618	1.000000	0.950823	0.842983	0.897527	-0.504683	-0.345647
displacement	-0.805127	0.950823	1.000000	0.897257	0.932994	-0.543800	-0.369855
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361
weight	-0.832244	0.897527	0.932994	0.864538	1.000000	-0.416839	-0.309120
acceleration	0.423329	-0.504683	-0.543800	-0.689196	-0.416839	1.000000	0.290316
year	0.580541	-0.345647	-0.369855	-0.416361	-0.309120	0.290316	1.000000
origin	0.565209	-0.568932	-0.614535	-0.455171	-0.585005	0.212746	0.181528

Problem 1(d)

In [5]:
```
df_quant['const'] = 1
```

In [6]:
```
reg1 = sm.OLS(endog=df_quant['mpg'], exog=df_quant[['const','cylinders', 'disp
lacement', 'horsepower', 'weight', 'acceleration', 'year', 'origin']], missing
='drop')
results = reg1.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                    mpg   R-squared:                       0.82
1
Model:                            OLS   Adj. R-squared:                  0.81
8
Method:                 Least Squares   F-statistic:                     252.
4
Date:                Sun, 16 Feb 2020   Prob (F-statistic):          2.04e-13
9
Time:                        17:18:40   Log-Likelihood:                -1023.
5
No. Observations:                 392   AIC:                             206
3.
Df Residuals:                     384   BIC:                             209
5.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
===
                   coef    std err          t      P>|t|      [0.025      0.9
75]
------------------------------------------------------------------------------
---
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:                   1.30
9
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.10
0
Skew:                           0.529   Prob(JB):                     2.95e-1
2
Kurtosis:                       4.460   Cond. No.                     8.59e+0
4
==============================================================================
=

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

```
[2] The condition number is large, 8.59e+04. This might indicate that there a
re
strong multicollinearity or other numerical problems.
```

i. We have 'displacement', 'year', 'weight', and 'origin' are significant at 1% level.

ii. We have 'horsepower', 'cylinders', and 'acceleration' are not significant at 10% level.

iii. Suppose other variables will not change. We have 1 unit of year increase would bring mpg about 0.7508 unit increase.

Problem 1(e)

In [7]:

```
df_quant['displacement2'] = np.square(df_quant['displacement'])
df_quant['horsepower2'] = np.square(df_quant['horsepower'])
df_quant['acceleration2'] = np.square(df_quant['acceleration'])
df_quant['weight2'] = np.square(df_quant['weight'])
```

In [8]:
```
reg2 = sm.OLS(endog=df_quant['mpg'], exog=df_quant[['const','cylinders', 'disp
lacement', 'horsepower', 'weight', 'acceleration', 'year', 'origin', 'displace
ment2', 'horsepower2', 'weight2', 'acceleration2']], missing='drop')
results2 = reg2.fit()
print(results2.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                    mpg   R-squared:                       0.87
0
Model:                            OLS   Adj. R-squared:                  0.86
6
Method:                 Least Squares   F-statistic:                     230.
2
Date:                Sun, 16 Feb 2020   Prob (F-statistic):          1.75e-16
0
Time:                        17:18:40   Log-Likelihood:                -962.0
2
No. Observations:                 392   AIC:                             194
8.
Df Residuals:                     380   BIC:                             199
6.
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       3
3.274
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
displacement2  2.257e-05   3.61e-05      0.626      0.532   -4.83e-05    9.35
e-05
horsepower2       0.0004      0.000      2.943      0.003       0.000
0.001
weight2        1.514e-06   3.69e-07      4.105      0.000    7.89e-07    2.24
e-06
acceleration2     0.0576      0.016      3.496      0.001       0.025
0.090
==============================================================================
=
Omnibus:                       33.614   Durbin-Watson:                   1.57
6
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               77.98
5
Skew:                           0.438   Prob(JB):                     1.16e-1
7
```

```
Kurtosis:                       5.002   Cond. No.                    5.13e+0
8
==============================================================================
=

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

ii. The adjusted R-squared stats is better than which of part(d).

iii. The terms are both non-significant at 10% level.

iv. It is not significant at 10% level, and its p-value is greater that which of the previous model.

Problem 1(f)

In [9]:
```
X = [1, 6, 200, 100, 3100, 15.1, 99, 1, 200**2, 100**2, 3100**2, 15.1**2]
prediction = results2.predict(X)
print('The prediction mpg of cylinders displacement of 200, horsepower of 100,
a weight of 3,100, acceleration of 15.1, model year of 1999, and origin of 1 i
s', prediction[0])
```

```
The prediction mpg of cylinders displacement of 200, horsepower of 100, a wei
ght of 3,100, acceleration of 15.1, model year of 1999, and origin of 1 is 3
8.73211109753366
```

Problem 2(a)

In [10]:
```
df2 = 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']*3+['Green']*2+['Red']})
```

In [11]:
```
df2['Eucl. Dist from X1=X2=X3=0'] = np.sqrt(df2['X1']**2+df2['X2']**2+df2['X3'
]**2)
```

In [12]:
```
df2
```

Out[12]:

	X1	X2	X3	Y	Eucl. Dist from X1=X2=X3=0
0	0	3	0	Red	3.000000
1	2	0	0	Red	2.000000
2	0	1	3	Red	3.162278
3	0	1	2	Green	2.236068
4	-1	0	1	Green	1.414214
5	1	1	1	Red	1.732051

Problem 2(b)

In [13]:
```
min(df2['Eucl. Dist from X1=X2=X3=0'])
```

Out[13]:
```
1.4142135623730951
```

Since the fifth row has distance closest to $X1 = X2 = X3 = 0$, we have the prediction is green.

Problem 2(c)

Since the second, fifth, and sixth rows have distances closest to $X1 = X2 = X3 = 0$, we have the prediction is more likely to be red.

Problem 2(d)

If the Bayes (optimal) decision boundary in this problem is highly nonlinear, we would expect the best value for K to be large. Larger K can cover more points near the target, so the prediction could be more accurate.

Problem 2(e)

In [14]:
```
neigh = KNeighborsClassifier(n_neighbors=2)
X = df2[['X1','X2','X3']]
Y = df2['Y']
neigh.fit(X, Y)
print('The KNN prediction for X1=X2=X3=1 and K=2 is', neigh.predict([(1,1,1)])
[0])
```

```
The KNN prediction for X1=X2=X3=1 and K=2 is Green
```

Problem 3(a)

In [15]:
```
df_quant['mpg_high'] = np.where(df_quant['mpg'] >= np.median(df['mpg']), 1, 0)
```

In [16]:
```
reg3 = sm.Logit(endog=df_quant['mpg_high'], exog=df_quant[['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:                  39
2
Model:                          Logit   Df Residuals:                      38
4
Method:                           MLE   Df Model:
7
Date:                Sun, 16 Feb 2020   Pseudo R-squ.:                  0.710
1
Time:                        17:18:40   Log-Likelihood:                -78.77
0
converged:                       True   LL-Null:                       -271.7
1
Covariance Type:            nonrobust   LLR p-value:                 2.531e-7
9
==============================================================================
===
                   coef    std err          z      P>|z|      [0.025      0.9
75]
------------------------------------------------------------------------------
---
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.
```

We have weight and year are significant at 5% level.

Problem 3(b)

In [17]:
```
X = df_quant[['const','cylinders', 'displacement', 'horsepower', 'weight', 'ac
celeration', 'year', 'origin']]
Y = df_quant['mpg_high']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.5, rando
m_state=10)
```

Problem 3(c)

In [18]:
```
clf = LogisticRegression(max_iter = 10000).fit(X_train, y_train)

for i in range(7):
    print('The coeffecient for',X.columns[i],'is',clf.coef_[0][i])
```

```
The coeffecient for const is -0.0014062712185044482
The coeffecient for cylinders is -1.1505902795202694
The coeffecient for displacement is 0.01692195670266103
The coeffecient for horsepower is 0.014535241600150541
The coeffecient for weight is -0.007220539658275798
The coeffecient for acceleration is 0.1521838862892103
The coeffecient for year is 0.5780143927460908
```

Problem 3(d)

In [19]:
```
predict_y = clf.predict(X_test)
compare_y = confusion_matrix(y_test, predict_y)
```

In [20]:
```
compare_y
```

Out[20]:
```
array([[85, 14],
       [ 9, 88]], dtype=int64)
```

In [21]:
```
print(classification_report(y_test, predict_y))
```

```
              precision    recall  f1-score   support

           0       0.90      0.86      0.88        99
           1       0.86      0.91      0.88        97

    accuracy                           0.88       196
   macro avg       0.88      0.88      0.88       196
weighted avg       0.88      0.88      0.88       196
```

The model predicts both equally well.