Keertana V. Chidambaram
PS 6 Solutions

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns

import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn import neighbors
from sklearn.linear_model import LogisticRegression
```

Problem 1

In [2]:
```
# Solution 1.a.
auto_df = pd.read_csv('data/auto.csv', na_values=['?'])
auto_df.dropna(inplace=True)
print(auto_df.shape)
auto_df.head()
```

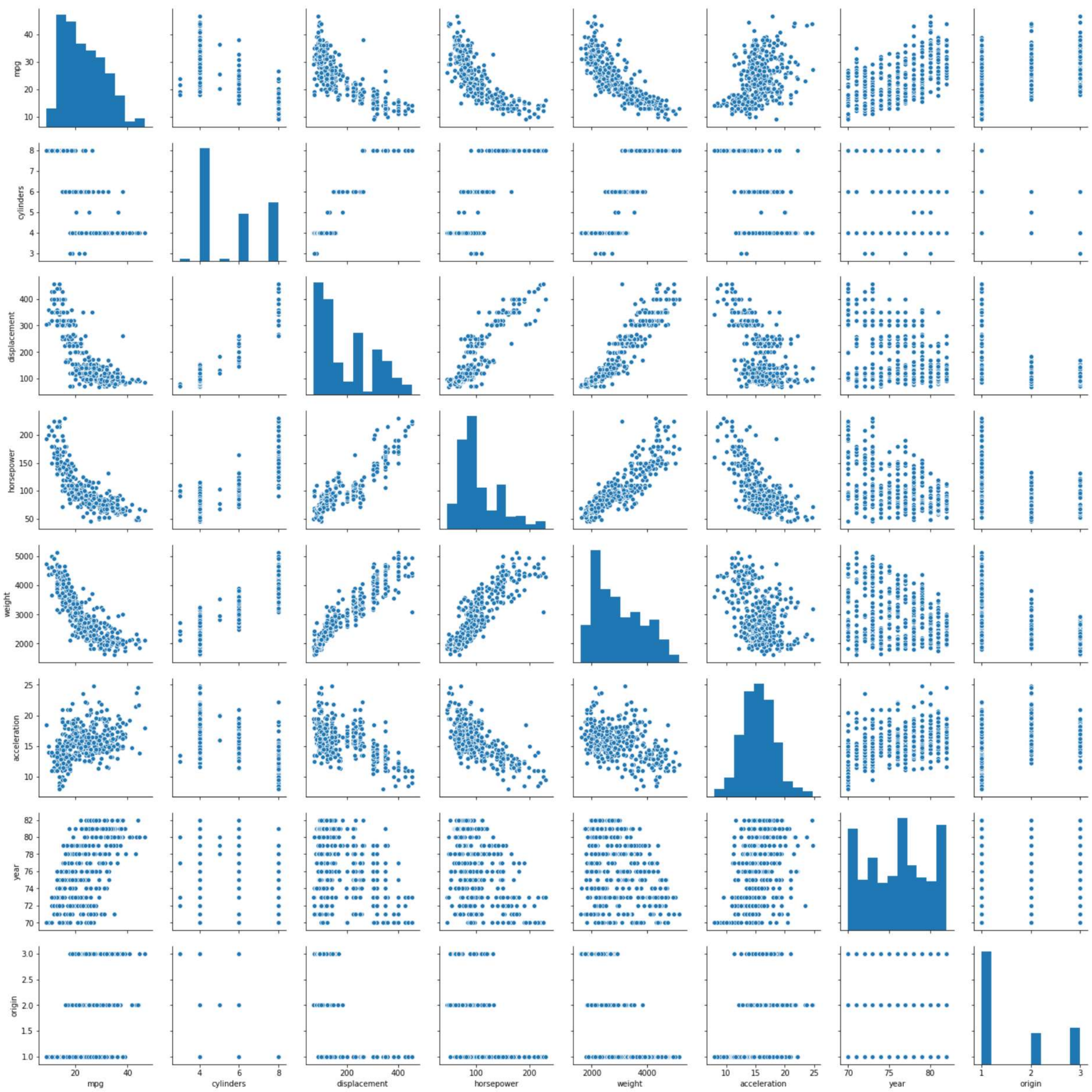
```
(392, 9)
```

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

In [3]:
```
# Solution 1.b.
sns.pairplot(auto_df, dropna=True)
```

Out[3]: <seaborn.axisgrid.PairGrid at 0x211561fdcc0>



In [4]:
```
# Solution 1.c.
auto_df.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

In [5]:
```
# Solution 1.d.
auto_df['const'] = 1
endo = auto_df['mpg']
exo = auto_df[['const', 'cylinders', 'displacement', 'horsepower', 'weight',
'acceleration', 'year', 'origin']]
reg1 = sm.OLS(endog=endo, exog=exo, missing='drop')
results1 = reg1.fit()
print(results1.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:                Mon, 18 Feb 2019   Prob (F-statistic):          2.04e-13
9
Time:                        21:23:23   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.
```

The coefficients that are statistically significant at the 1% level (i.e. p < 0.01):

1. β_2 (displacement)
2. β_4 (weight)
3. β_6 (year)
4. β_7 (origin)

The coefficients that are not statistically significant at the 10% level (i.e. p > 0.10):

1. β_1 (cylinders)
2. β_3 (horsepower)
3. β_5 (acceleration)

$\beta_6 = 0.7508$ is statistically significant at the 1% level with p value = 0.000. It means that if the variable $year$ increases by one unit (and other variables are held constant), the predicted value of mpg is expected to increase by 0.7508 units according to the model.

In [6]:
```
# Solution 1.e.
```

From plot (b), displacement, horsepower, and weight are the three variables that look most likely to have a nonlinear relationship with mpg.

In [7]:
```
auto_df['const'] = 1

auto_df['displacement_sq'] = auto_df['displacement'] ** 2
auto_df['horsepower_sq'] = auto_df['horsepower'] ** 2
auto_df['weight_sq'] = auto_df['weight'] ** 2
auto_df['acceleration_sq'] = auto_df['acceleration'] ** 2

endo2 = auto_df['mpg']
exo2 = auto_df[['const', 'cylinders', 'displacement_sq', 'horsepower_sq', 'wei
ght_sq', 'acceleration_sq', 'year', 'origin']]
reg2 = sm.OLS(endog=endo2, exog=exo2, missing='drop')
results2 = reg2.fit()
print(results2.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                    mpg   R-squared:                       0.80
2
Model:                            OLS   Adj. R-squared:                  0.79
9
Method:                 Least Squares   F-statistic:                     222.
6
Date:                Mon, 18 Feb 2019   Prob (F-statistic):          6.37e-13
1
Time:                        21:23:23   Log-Likelihood:                -1043.
5
No. Observations:                 392   AIC:                             210
3.
Df Residuals:                     384   BIC:                             213
5.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
======
                      coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------
const             -25.4628      4.442     -5.732      0.000     -34.197     -
16.729
cylinders          -1.2260      0.284     -4.321      0.000      -1.784
-0.668
displacement_sq  6.412e-05   1.35e-05      4.736      0.000    3.75e-05    9.
07e-05
horsepower_sq   -5.615e-05   4.97e-05     -1.130      0.259      -0.000    4.
16e-05
weight_sq       -9.095e-07    8.9e-08    -10.215      0.000   -1.08e-06   -7.
34e-07
acceleration_sq     0.0060      0.003      2.245      0.025       0.001
0.011
year                0.7606      0.053     14.304      0.000       0.656
0.865
origin              1.6707      0.276      6.062      0.000       1.129
2.213
==============================================================================
=
Omnibus:                       20.589   Durbin-Watson:                   1.32
9
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               29.35
4
Skew:                           0.409   Prob(JB):                     4.23e-0
7
Kurtosis:                       4.062   Cond. No.                     2.77e+0
8
==============================================================================
=

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

```
[2] The condition number is large, 2.77e+08. This might indicate that there a
re
strong multicollinearity or other numerical problems.
```

Adjusted R-squared (part e.) = 0.799
Adjusted R-squared (part d.) = 0.818
Hence we are getting a lower R-squared value in the new model.

The p-value for displacement coefficient in the first model was 0.008.
The p-value for squared displacement's coefficient in the second model is 0.000.
Hence the statistical significance of displacement coefficient has increased after squaring. But it is significant in both the models.

The p-value for cylinders coefficient in the first model was 0.128.
The p-value for cylinders coefficient in the second model is 0.000.
Hence the statistical significance of cylinders coefficient has also increased in the new model. It is significant in the new model but not in the old (at 10 significance).

In [8]:
```
# Solution 1.f.
a = results2.predict(exog=[1, 6, 200, 100, 3100, 15.1, 99, 1])
print("Predicted value of mpg = ", a[0])
```

```
Predicted value of mpg =  44.24571361822012
```

Problem 2

In [9]:
```
# Solution 2.a.
train_data = 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']])
train_data.columns = ['X1', 'X2', 'X3', 'Y']
```

In [10]:
```
train_data
```

Out[10]:

	X1	X2	X3	Y
0	0	3	0	Red
1	2	0	0	Red
2	0	1	3	Red
3	0	1	2	Green
4	-1	0	1	Green
5	1	1	1	Red

In [11]:
```
train_data['dist'] = (train_data['X1'] ** 2 + train_data['X2'] ** 2 + train_da
ta['X3'] ** 2) ** 0.5
train_data
```

Out[11]:

	X1	X2	X3	Y	dist
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

In [12]:
```
# Solutions 2.b. and 2.c.
```

When $K = 1$, prediction = $Green$. This is because we use the marker of the closest point which is $Green$.

When $K = 3$, prediction = Red. This is because the closest 3 points are marked $Green$, Red and Red. Hence $P(Green) = 1/3$ and $P(Red) = 2/3$. Red has a higher probability, hence we predict Red.

In [13]:
```
# Solutions 2.d.
```

If the optimal boundary is highly non-lienar, it implies high variance, which correponds to small K. If we pick a large K, then it could potentially smooth out important non-linear trends in the data.

In [14]:
```
# Solutions 2.e.
model = neighbors.KNeighborsClassifier(n_neighbors = 2)
x_vars = ['X1', 'X2', 'X3']
train_data[x_vars]
res = model.fit(train_data[x_vars], train_data['Y'])
print('Predicted value for (1,1,1) is:')
print(res.predict([(0,0,0)]))
#print("The KNN classifier of the test point X1 = X2 = X3 = 1 with K = 2:",
#      neigh.predict([(0,0,0)])[0])
```

```
Predicted value for (1,1,1) is:
['Green']
```

Problem 3

In [15]:
```
print(auto_df['mpg'].median())
auto_df['mpg_high'] = (auto_df['mpg'] >= auto_df['mpg'].median()).astype(int)
```

```
22.75
```

In [16]:
```
# Solutions 3.a.
xvars = ['cylinders','displacement','horsepower','weight','acceleration','yea
r','origin']
X = auto_df[xvars]
X['const'] = 1
y = auto_df['mpg_high']

LogitModel = sm.Logit(y, X, missing='drop')
LogitReg_sm = LogitModel.fit()
print(LogitReg_sm.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:               Mon, 18 Feb 2019   Pseudo R-squ.:                  0.710
1
Time:                       21:23:25   Log-Likelihood:                -78.77
0
converged:                      True   LL-Null:                       -271.7
1
                                       LLR p-value:                 2.531e-7
9
=============================================================================
===
                  coef    std err          z      P>|z|      [0.025      0.9
75]
-----------------------------------------------------------------------------
---
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
const         -17.1549      5.764     -2.976      0.003     -28.452      -5.
858
=============================================================================
===

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.
```

```
C:\Users\admin\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  after removing the cwd from sys.path.
```

The variables that are significant at the 5% level ($p \leq 0.05$) are: $weight$ and $year$.

In [17]:
```
# Solutions 3.b.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, rando
m_state=10)
```

In [18]:
```
# Solutions 3.c.
clf = LogisticRegression().fit(X_train, y_train)
coeff = pd.DataFrame(clf.coef_.tolist()[0], columns=['Coefficient Value'])
var_name = pd.DataFrame(xvars+['const'], columns=['Variable Name'])
pd.concat([var_name, coeff], axis=1)
```

```
C:\Users\admin\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
  FutureWarning)
```

Out[18]:

	Variable Name	Coefficient Value
0	cylinders	-0.734368
1	displacement	0.007140
2	horsepower	-0.035617
3	weight	-0.005104
4	acceleration	-0.124616
5	year	0.298453
6	origin	-0.163247
7	const	-0.076834

In [19]:
```
# Solutions 3.d.
y_pred = clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print("Confusion matrix:")
print(cm)

print("Classification report:")
print(classification_report(y_test, y_pred))
```

```
Confusion matrix:
[[86 13]
 [12 85]]
Classification report:
              precision    recall  f1-score   support

           0       0.88      0.87      0.87        99
           1       0.87      0.88      0.87        97

   micro avg       0.87      0.87      0.87       196
   macro avg       0.87      0.87      0.87       196
weighted avg       0.87      0.87      0.87       196
```

Both have the same precision, recall and f1-score, hence the classifier predicts high and low equally well.