In [24]:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from patsy import dmatrices
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn import metrics
from sklearn.cross_validation import cross_val_score
dta = sm.datasets.fair.load_pandas().data

# add "affair" column: 1 represents having affairs, 0 represents not
dta['affair'] = (dta.affairs > 0).astype(int)


y, X = dmatrices('affair ~ rate_marriage + age + yrs_married + children + \
                  religious + educ + C(occupation) + C(occupation_husb)',
                  dta, return_type="dataframe")

X = X.rename(columns = {'C(occupation)[T.2.0]':'occ_2',
                        'C(occupation)[T.3.0]':'occ_3',
                        'C(occupation)[T.4.0]':'occ_4',
                        'C(occupation)[T.5.0]':'occ_5',
                        'C(occupation)[T.6.0]':'occ_6',
                        'C(occupation_husb)[T.2.0]':'occ_husb_2',
                        'C(occupation_husb)[T.3.0]':'occ_husb_3',
                        'C(occupation_husb)[T.4.0]':'occ_husb_4',
                        'C(occupation_husb)[T.5.0]':'occ_husb_5',
                        'C(occupation_husb)[T.6.0]':'occ_husb_6'})

y = np.ravel(y)

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15, random_state = 30)

# Feature Scaling
'''
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
'''
```

```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 30)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix\n",cm)
print("\nAccuracy %.3f\n" %(metrics.accuracy_score(y_test, y_pred)*100))
df2 = pd.DataFrame(np.random.randint(low=0, high=10, size=(955, 2)), columns=['Test', 'Predicted'])
df2['Test'] = y_test
df2['Predicted'] = y_pred
print(df2)
```

```
Confusion Matrix
 [[618  61]
 [167 109]]

Accuracy 76.126

     Test  Predicted
0     0.0        0.0
1     0.0        1.0
2     1.0        1.0
3     0.0        0.0
4     0.0        1.0
5     1.0        0.0
6     0.0        0.0
7     0.0        0.0
8     1.0        0.0
9     0.0        0.0
10    1.0        0.0
11    0.0        0.0
12    0.0        0.0
13    1.0        0.0
14    1.0        1.0
15    0.0        0.0
16    1.0        1.0
17    0.0        0.0
18    0.0        0.0
19    0.0        0.0
```

```
20    0.0        0.0
21    1.0        1.0
22    0.0        0.0
23    0.0        0.0
24    0.0        0.0
25    0.0        0.0
26    0.0        1.0
27    0.0        0.0
28    1.0        0.0
29    0.0        0.0
..    ...        ...
925   1.0        1.0
926   0.0        0.0
927   0.0        0.0
928   0.0        0.0
929   0.0        0.0
930   0.0        0.0
931   0.0        0.0
932   1.0        1.0
933   0.0        0.0
934   0.0        0.0
935   1.0        0.0
936   1.0        0.0
937   0.0        0.0
938   1.0        1.0
939   0.0        1.0
940   0.0        0.0
941   0.0        0.0
942   0.0        0.0
943   0.0        0.0
944   0.0        0.0
945   0.0        0.0
946   0.0        0.0
947   1.0        0.0
948   1.0        0.0
949   1.0        0.0
950   0.0        0.0
951   1.0        1.0
952   0.0        1.0
953   1.0        0.0
954   0.0        0.0

[955 rows x 2 columns]
```