**Assignment 21**

**Problem Statement 1:**

I decided to treat this as a classification problem by creating a new binary variable affair (did the woman have at least one affair?) and trying to predict the classification for each woman.

Answer :

## Loading Libraries

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

from \_\_future\_\_ import unicode\_literals

# Load dataset

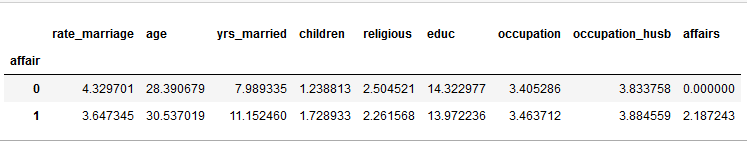
train = sm.datasets.fair.load\_pandas().data

# Add 'affair' column: 1 rpresents having affair, 0 doesn't

train['affair'] = (train.affairs > 0).astype(int)

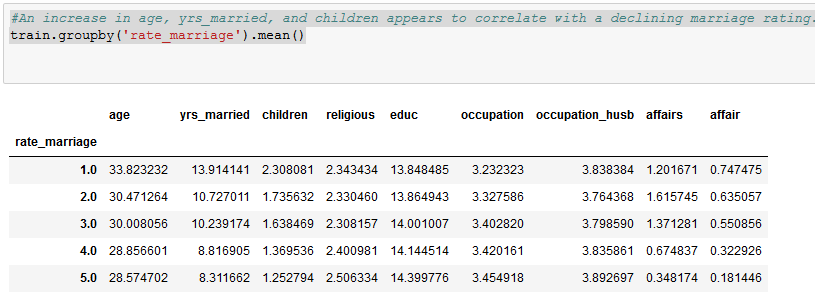
# data exploration women whose mariage rate is lower has an affair

train.groupby('affair').mean()



#An increase in age, yrs\_married, and children appears to correlate with a declining marriage rating.

train.groupby('rate\_marriage').mean()



#Data Visualization

%matplotlib notebook

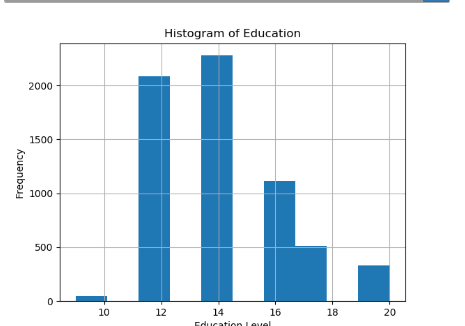
# histogram of education

train.educ.hist()

plt.title('Histogram of Education')

plt.xlabel('Education Level')

plt.ylabel('Frequency')



#see below distribution of marriage ratings for those having affairs versus those not having affairs.

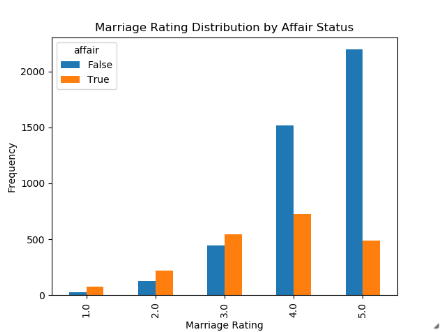
# barplot of marriage rating grouped by affair (True or False)

pd.crosstab(train.rate\_marriage, train.affair.astype(bool)).plot(kind='bar')

plt.title('Marriage Rating Distribution by Affair Status')

plt.xlabel('Marriage Rating')

plt.ylabel('Frequency')



#using stacked barplot to look at the percentage of women having affairs by number of years of marriage.

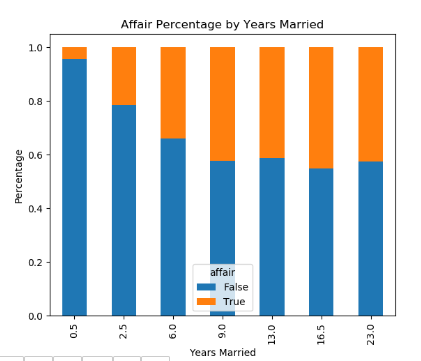
affair\_yrs\_married = pd.crosstab(train.yrs\_married, train.affair.astype(bool))

affair\_yrs\_married.div(affair\_yrs\_married.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)

plt.title('Affair Percentage by Years Married')

plt.xlabel('Years Married')

plt.ylabel('Percentage')



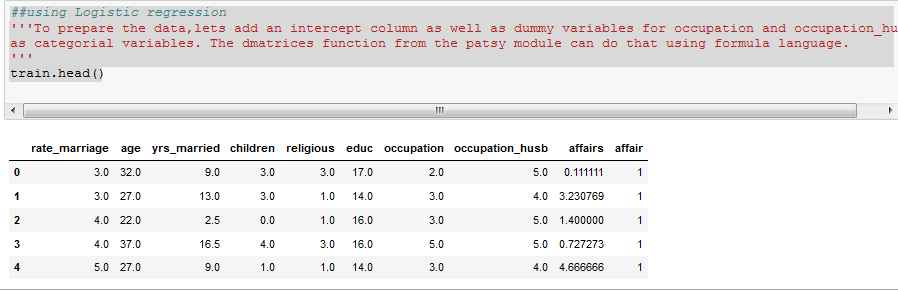
##using Logistic regression

'''To prepare the data,lets add an intercept column as well as dummy variables for occupation and occupation\_husb,

as categorial variables. The dmatrices function from the patsy module can do that using formula language.

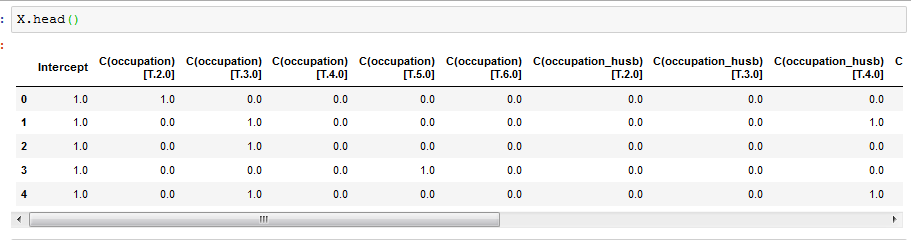
'''

train.head()



y, X = dmatrices('affair ~ rate\_marriage + age + yrs\_married + children + religious + educ + C(occupation) + C(occupation\_husb)', train, return\_type="dataframe")

print(X.columns)



Rename dummy variables

# fix column names of X

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'})

# flatten y into a 1-D array

y = np.ravel(y)

##Using Logistic regression

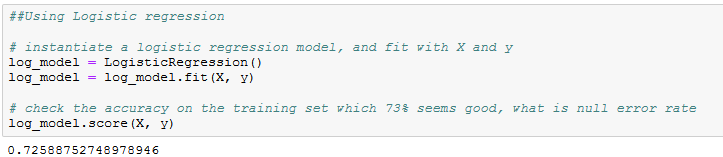
# instantiate a logistic regression model, and fit with X and y

log\_model = LogisticRegression()

log\_model = log\_model.fit(X, y)

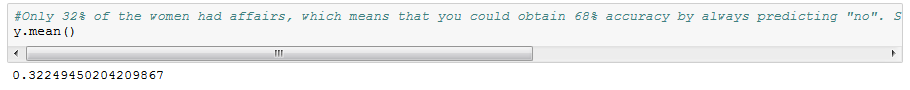
# check the accuracy on the training set which 73% seems good, what is null error rate

log\_model.score(X, y)



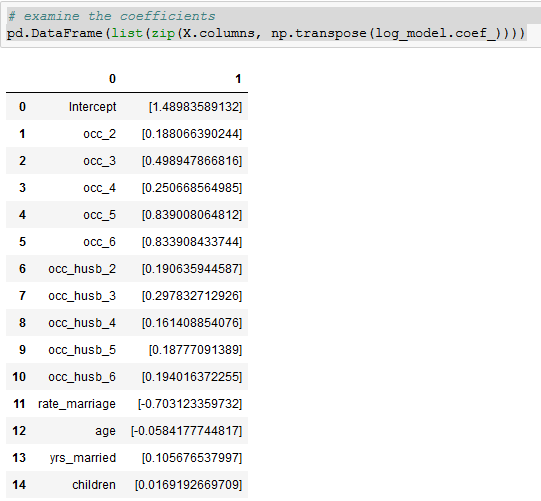
#Only 32% of the women had affairs, which means that you could obtain 68% accuracy by always predicting "no". So we're doing better than the null error rate, but not by much.

y.mean()



# examine the coefficients

pd.DataFrame(list(zip(X.columns, np.transpose(log\_model.coef\_))))

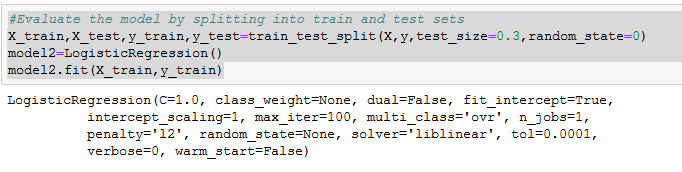


#Evaluate the model by splitting into train and test sets

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3,random\_state=0)

model2=LogisticRegression()

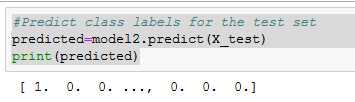
model2.fit(X\_train,y\_train)



#Predict class labels for the test set

predicted=model2.predict(X\_test)

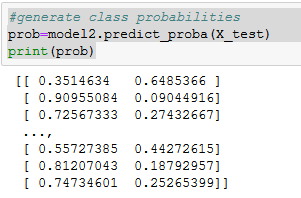
print(predicted)



#generate class probabilities

prob=model2.predict\_proba(X\_test)

print(prob)



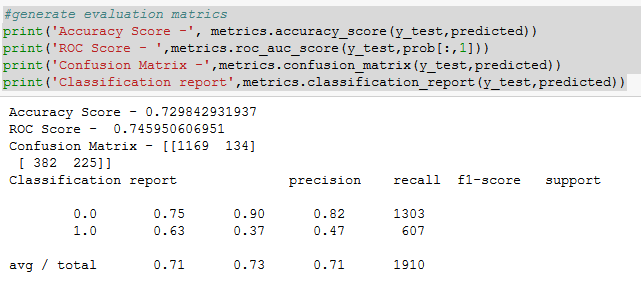
#generate evaluation matrics

print('Accuracy Score -', metrics.accuracy\_score(y\_test,predicted))

print('ROC Score - ',metrics.roc\_auc\_score(y\_test,prob[:,1]))

print('Confusion Matrix -',metrics.confusion\_matrix(y\_test,predicted))

print('Classification report',metrics.classification\_report(y\_test,predicted))



#Evaluating the model based on 10 fold cross validation

scores=cross\_val\_score(LogisticRegression(),X,y,scoring='accuracy',cv=10)

print(scores)

print(scores.mean())

