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.

Dataset

The dataset I chose is the affairs dataset that comes with Statsmodels. It was derived from a survey of women in 1974 by Redbook magazine, in which married women were asked about their participation in extramarital affairs. More information about the study is available in a 1978 paper from the Journal of Political Economy.

Description of Variables

The dataset contains 6366 observations of 9 variables:

rate\_marriage: woman's rating of her marriage (1 = very poor, 5 = very good)

age: woman's age

yrs\_married: number of years married

children: number of children

religious: woman's rating of how religious she is (1 = not religious, 4 = strongly religious)

educ: level of education (9 = grade school, 12 = high school, 14 = some college, 16 =college graduate, 17 = some graduate school, 20 = advanced degree)

occupation: woman's occupation (1 = student, 2 = farming/semi-skilled/unskilled, 3 = "white collar", 4 = teacher/nurse/writer/technician/skilled, 5 = managerial/business, 6 = professional with advanced degree)

occupation\_husb: husband's occupation (same coding as above)

affairs: time spent in extra-marital affairs

Code to loading data and modules

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

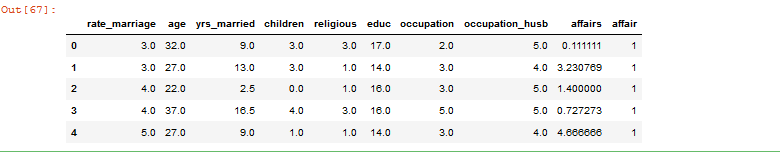
import metrics from sklearn.cross\_validation

import cross\_val\_score dta = sm.datasets.fair.load\_pandas().data

**# add "affair" column: 1 represents having affairs, 0 represents not**

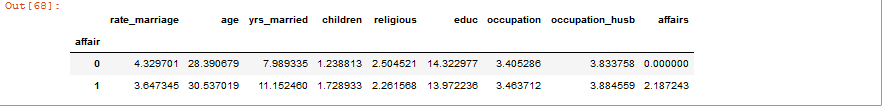
dta['affair'] = (dta.affairs > 0).astype(int)

dta.head()



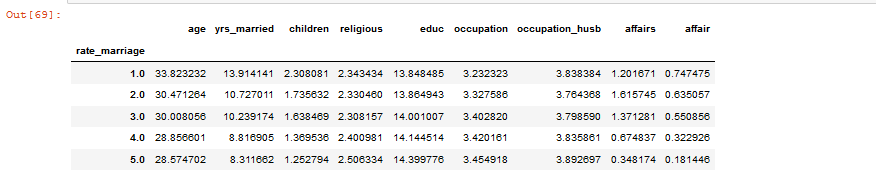
**#data exploration**

dta.groupby('affair').mean()



**#groupby rate\_marriage**

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



**#show plots in notebook**

%matplotlib inline

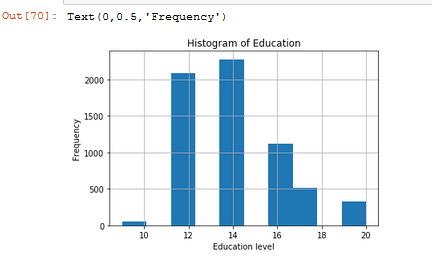
#histogram of education

dta.educ.hist()

plt.title('Histogram of Education')

plt.xlabel('Education level')

plt.ylabel('Frequency')



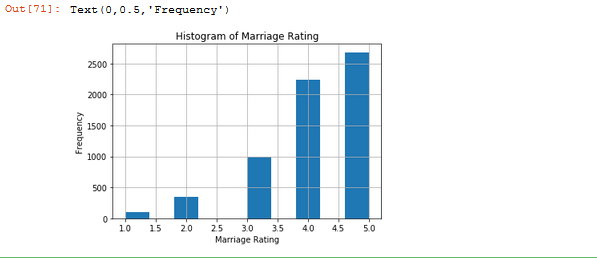
**#histogram of marriage rating**

dta.rate\_marriage.hist()

plt.title('Histogram of Marriage Rating')

plt.xlabel('Marriage Rating')

plt.ylabel('Frequency')



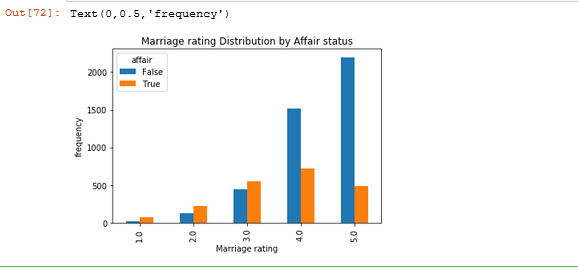
**#barplot - marriage rating grouped by affair(true or false)**

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

plt.title('Marriage rating Distribution by Affair status')

plt.xlabel('Marriage rating')

plt.ylabel('frequency')



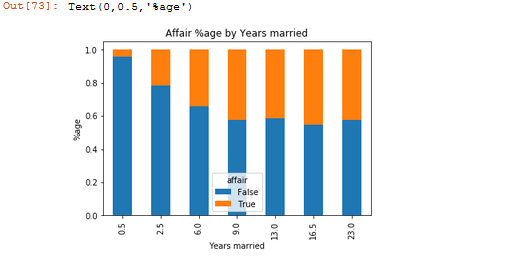
affair\_yrs\_married=pd.crosstab(dta.yrs\_married,dta.affair.astype(bool))

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

plt.title('Affair %age by Years married')

plt.xlabel('Years married')

plt.ylabel('%age')



**# create dataframes with an intercept column and dummy variables for**

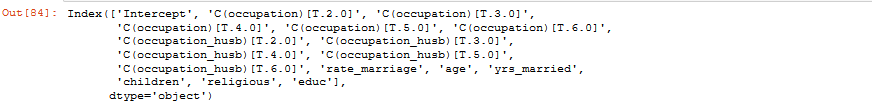
**# occupation and occupation\_husb**

y, X = dmatrices('affair ~ rate\_marriage + age + yrs\_married + children + \

religious + educ + C(occupation) + C(occupation\_husb)',

dta, return\_type="dataframe")

X.columns



**#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',

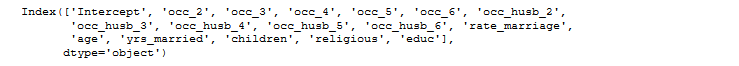
'C(occupation\_husb)[T.4.0]':'occ\_husb\_4',

'C(occupation\_husb)[T.5.0]':'occ\_husb\_5',

'C(occupation\_husb)[T.6.0]':'occ\_husb\_6'})

y = np.ravel(y)#flatten y into a 1-D array

print (X.columns)



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

model = LogisticRegression()

model = model.fit(X, y)

**# check the accuracy on the training set**

model.score(X, y)

OP10.PNG

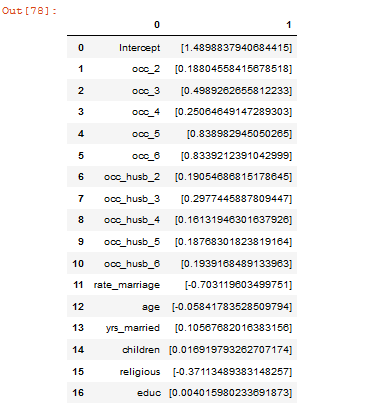
**# what percentage had affairs?**

y.mean()

op11.PNG

**# examine the coefficients**

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

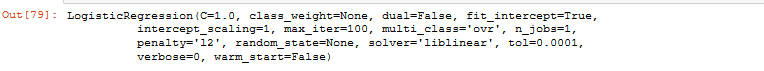


**# evaluate the model by splitting into train and test sets**

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)

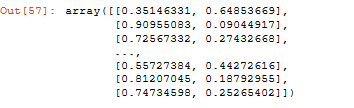
predicted

op14.PNG

# generate class probabilities

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

probs



**# generate evaluation metrics**

print(metrics.accuracy\_score(y\_test, predicted))

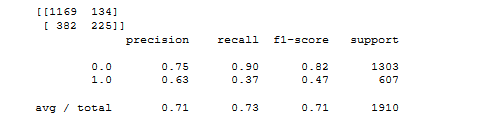
print(metrics.roc\_auc\_score(y\_test, probs[:, 1]))

op16.PNG

**#confusion matrix and a classification report with other metrics.**

print(metrics.confusion\_matrix(y\_test, predicted))

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



**# evaluate the model using 10-fold cross-validation**

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

scores, scores.mean()

op18.PNG

**#the probability of an affair for a random woman not present in the dataset. She's a 25-year-old teacher who graduated college, has been married for 3 years, has 1 child, rates herself as strongly religious, rates her marriage as fair, and her husband is a farmer.**

model.predict\_proba(np.array([[1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 3, 25, 3, 1, 4,16]]))

OP19.PNG