2. Problem Statement 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 marr 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) 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) NOTE: The solution shared through Github should contain the source code used and the screenshot of the output.

Answer:

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',

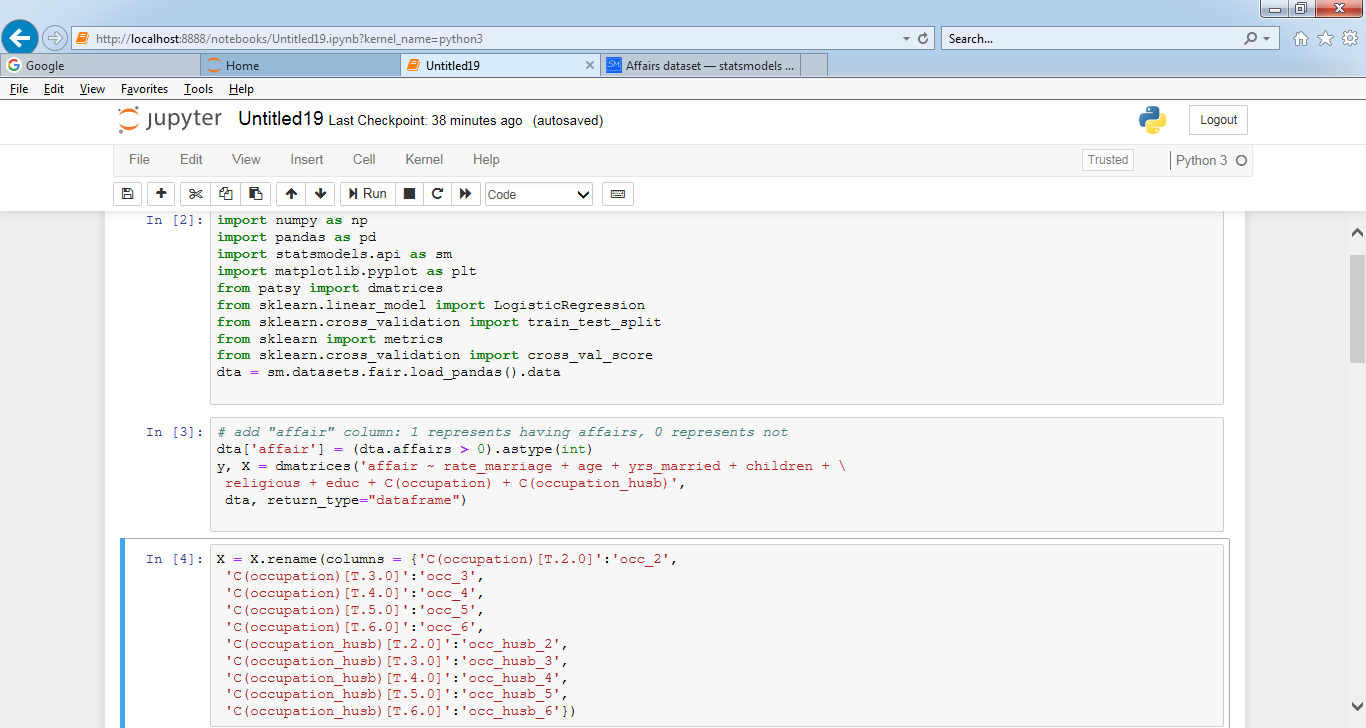
'C(occupation\_husb)[T.3.0]':'occ\_husb\_3',

'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)



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

model = LogisticRegression()

model = model.fit(X, y)

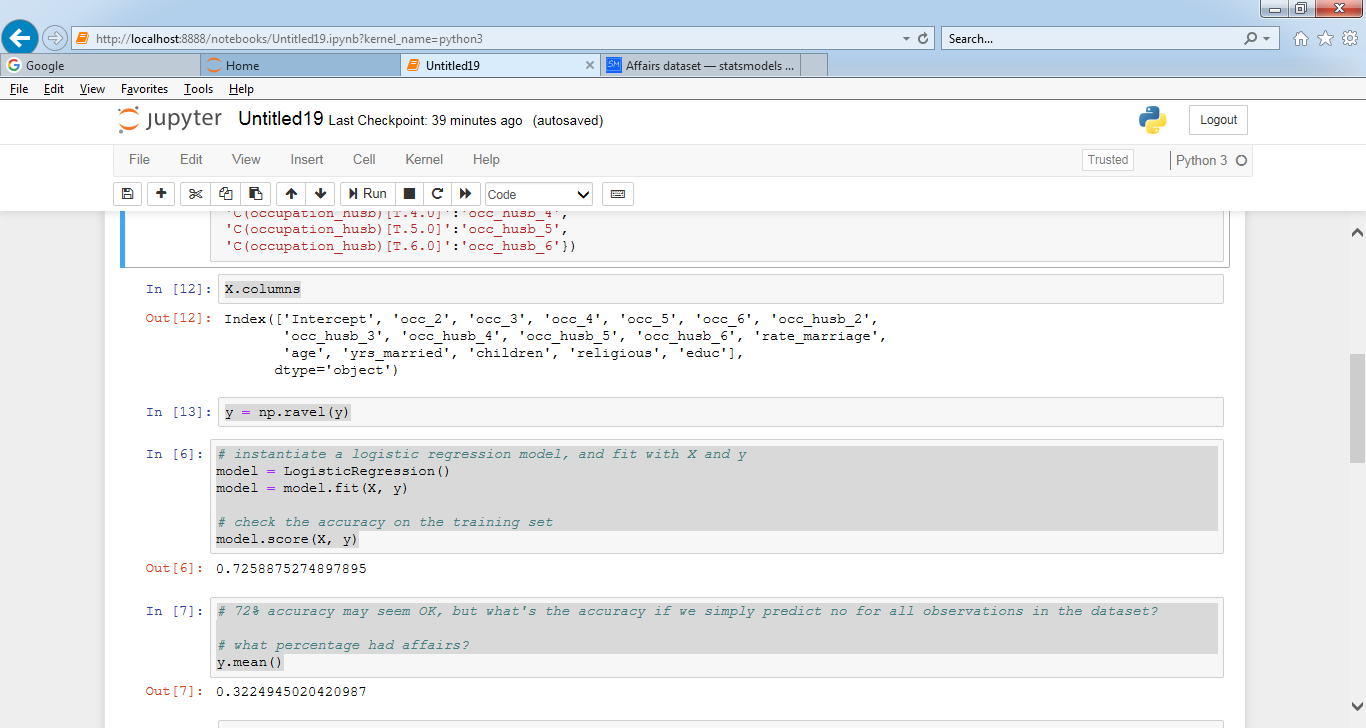
# check the accuracy on the training set

model.score(X, y)

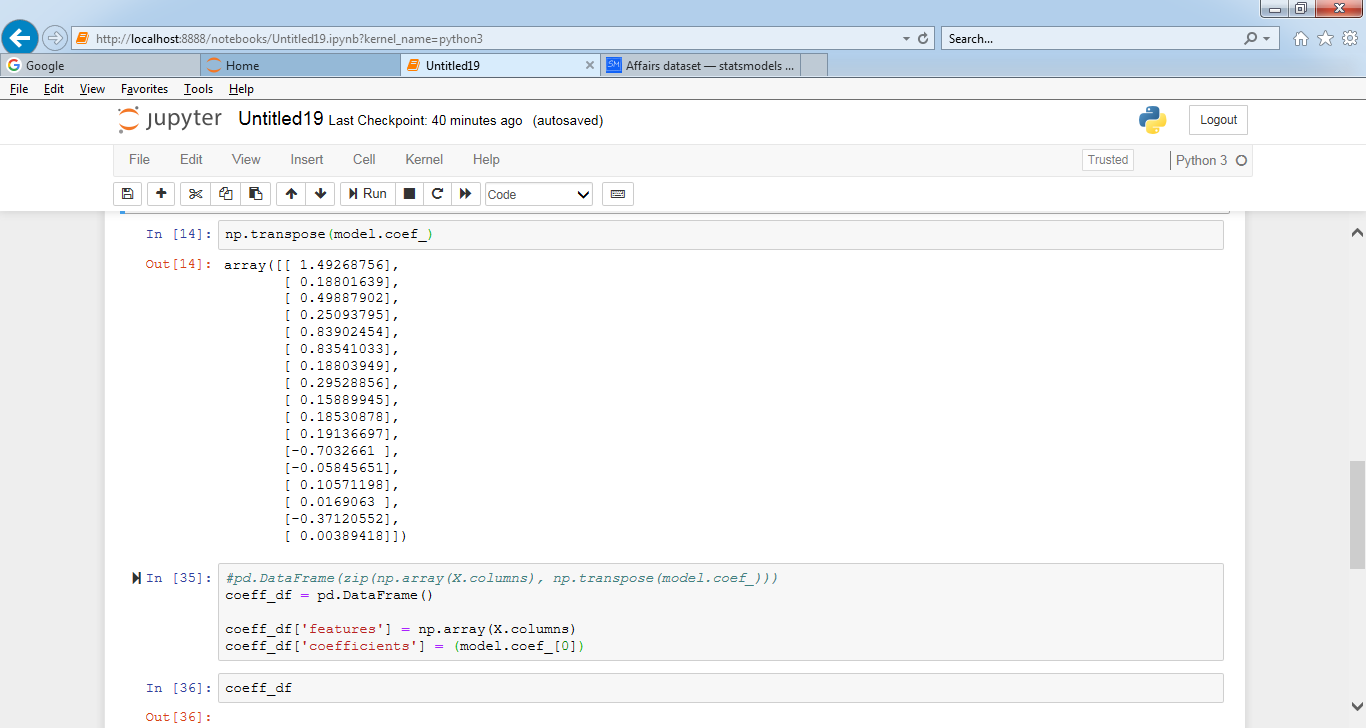
# 72% accuracy may seem OK, but what's the accuracy if we simply predict no for all observations in the dataset?

# what percentage had affairs?

y.mean()



np.transpose(model.coef\_)

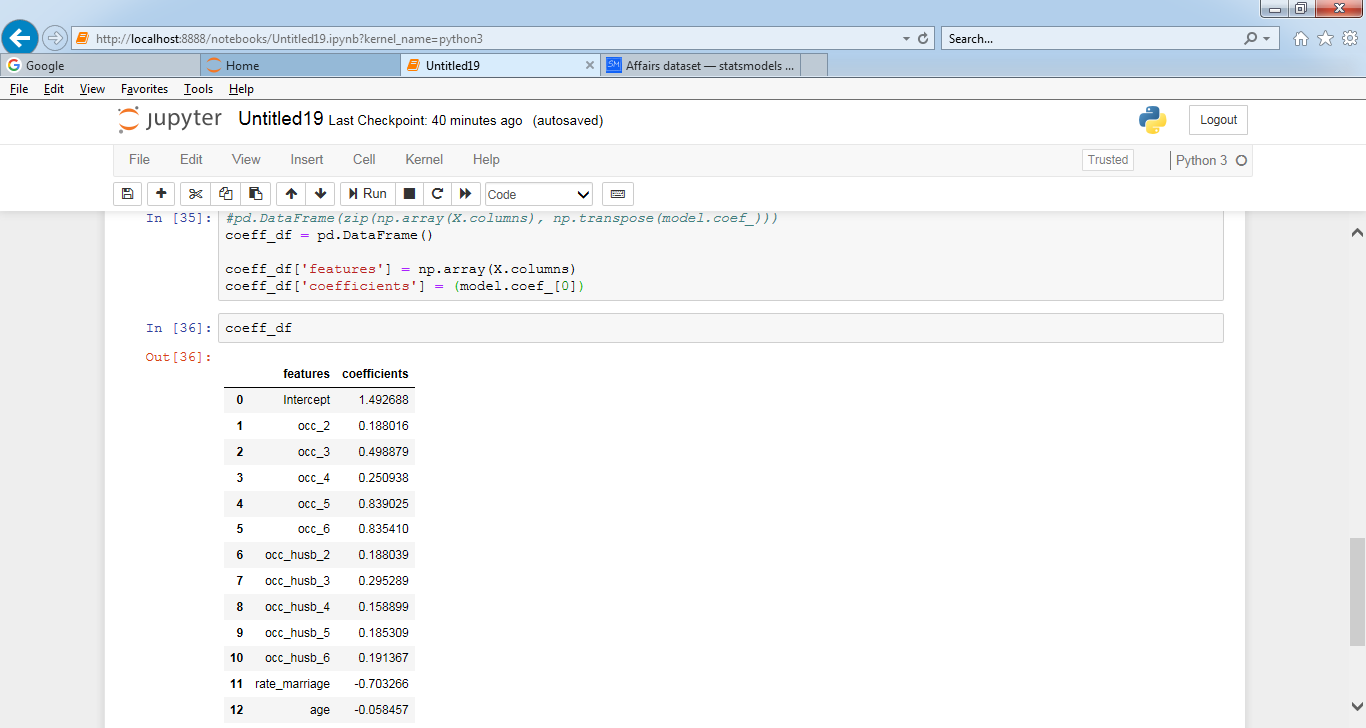


coeff\_df = pd.DataFrame()

coeff\_df['features'] = np.array(X.columns)

coeff\_df['coefficients'] = (model.coef\_[0])

coeff\_df



**Model Evaluation Using a Validation Set**

from sklearn.model\_selection import train\_test\_split

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

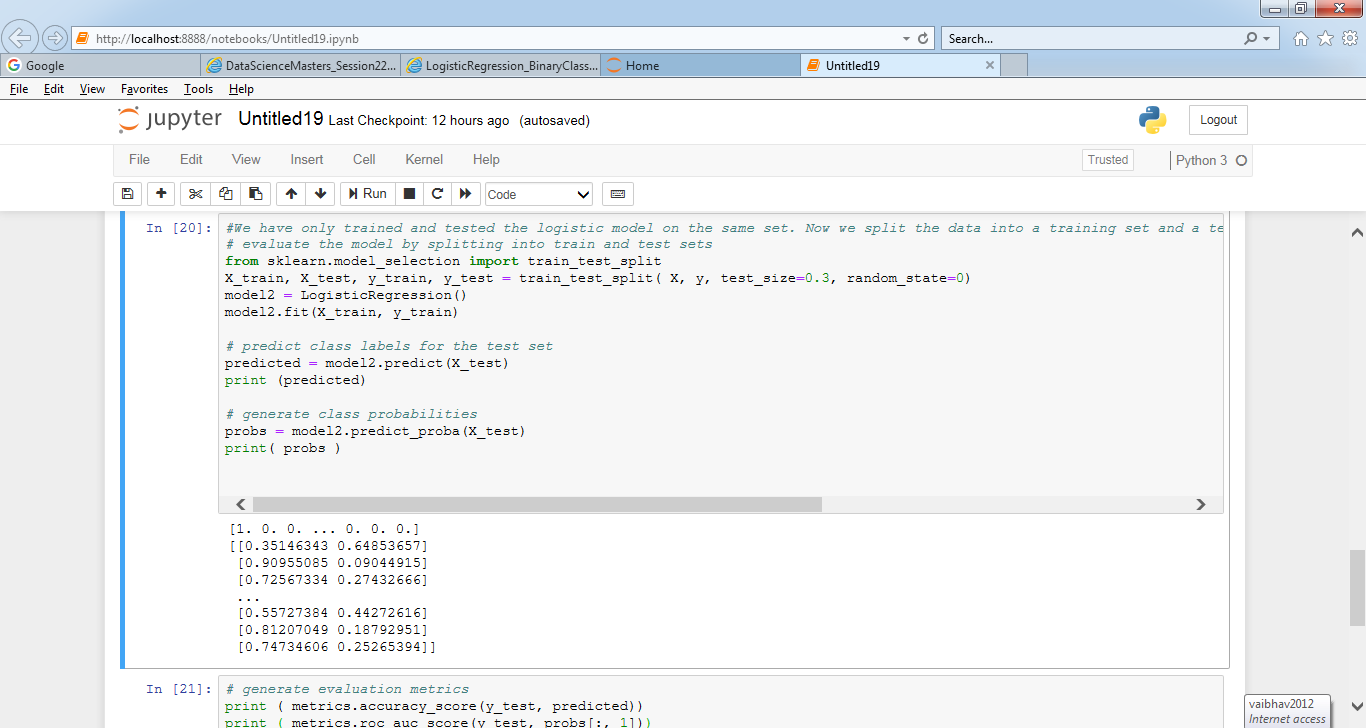
predicted = model2.predict(X\_test)

print (predicted)

# generate class probabilities

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

print( probs )



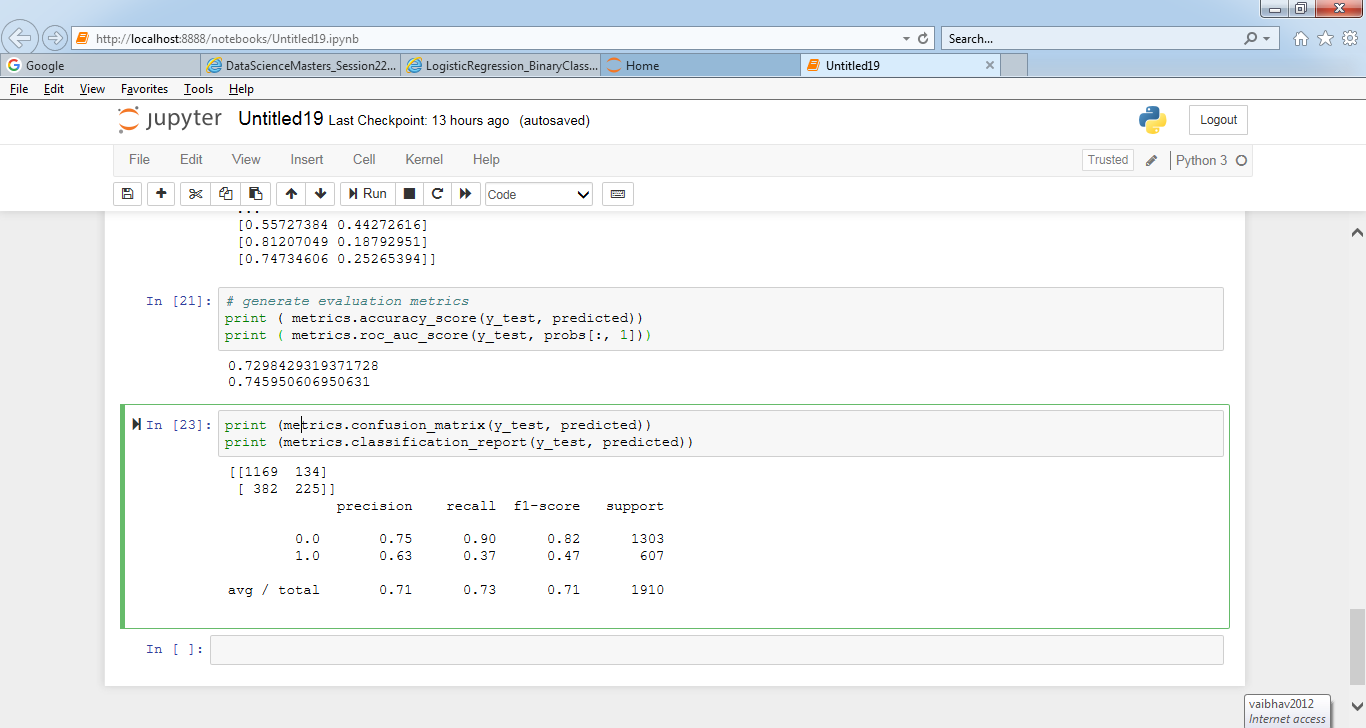
# generate evaluation metrics

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

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

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

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



**Using Cross-Validation**

