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.

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import sklearn
import statsmodels.api as sm
from patsy import dmatrices
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: Th
e pandas.core.datetools module is deprecated and will be removed in a future version. Please
use the pandas.tseries module instead.
  from pandas.core import datetools
```

In [7]:
```
#Loading Data
df = sm.datasets.fair.load_pandas().data
```

In [8]:
```
# add "affair" column: 1 represents having affairs, 0 represents not
df['affair'] = (df.affairs > 0).astype(int)
df.head()
```

Out[8]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupation_husb	affairs	affair
0	3.0	32.0	9.0	3.0	3.0	17.0	2.0	5.0	0.111111	1
1	3.0	27.0	13.0	3.0	1.0	14.0	3.0	4.0	3.230769	1
2	4.0	22.0	2.5	0.0	1.0	16.0	3.0	5.0	1.400000	1
3	4.0	37.0	16.5	4.0	3.0	16.0	5.0	5.0	0.727273	1
4	5.0	27.0	9.0	1.0	1.0	14.0	3.0	4.0	4.666666	1

Data Exploration

In [9]:
```
# Average of all features group by affair
df.groupby('affair').mean()
```

Out[9]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupation_husb	affair
affair									
0	4.329701	28.390679	7.989335	1.238813	2.504521	14.322977	3.405286	3.833758	0.00000
1	3.647345	30.537019	11.152460	1.728933	2.261568	13.972236	3.463712	3.884559	2.18724

We can see that on average, women who have affairs rate their marriages lower, which is to be expected. Let's take another look at the rate_marriage variable.

In [10]:
```
df.groupby('rate_marriage').mean()
```

Out[10]:

	age	yrs_married	children	religious	educ	occupation	occupation_husb	affairs	a
rate_marriage									
1.0	33.823232	13.914141	2.308081	2.343434	13.848485	3.232323	3.838384	1.201671	0.74
2.0	30.471264	10.727011	1.735632	2.330460	13.864943	3.327586	3.764368	1.615745	0.63
3.0	30.008056	10.239174	1.638469	2.308157	14.001007	3.402820	3.798590	1.371281	0.55
4.0	28.856601	8.816905	1.369536	2.400981	14.144514	3.420161	3.835861	0.674837	0.32
5.0	28.574702	8.311662	1.252794	2.506334	14.399776	3.454918	3.892697	0.348174	0.18

An increase in age, yrs_married, and children appears to correlate with a declining marriage rating.

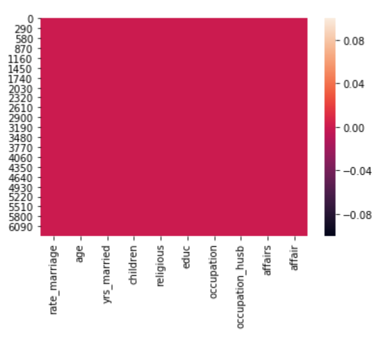
Data Visualization

In [14]:
```
df.isnull().values.any()
```

Out[14]: False

In [15]:
```
#Checking for NUllvalues if any
sns.heatmap(df.isnull())
```

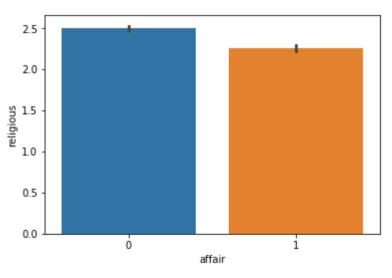
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1b5d4d2ca90>



There are no Null values in dataframe

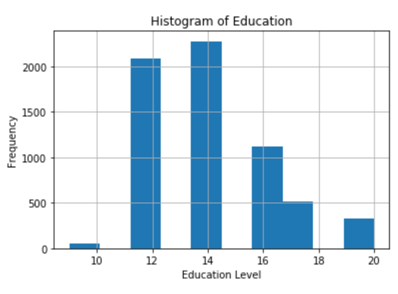
In [16]:
```
sns.barplot(x='affair',y='religious',data=df)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1b5d4bd8c50>



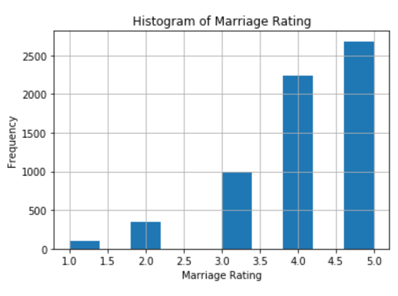
In [17]:
```
%matplotlib inline
# histogram of education
df.educ.hist()
plt.title('Histogram of Education')
plt.xlabel('Education Level')
plt.ylabel('Frequency')
```

Out[17]: Text(0,0.5,'Frequency')



In [18]:
```
# histogram of marriage rating
df.rate_marriage.hist()
plt.title('Histogram of Marriage Rating')
plt.xlabel('Marriage Rating')
plt.ylabel('Frequency')
```

Out[18]: Text(0,0.5,'Frequency')



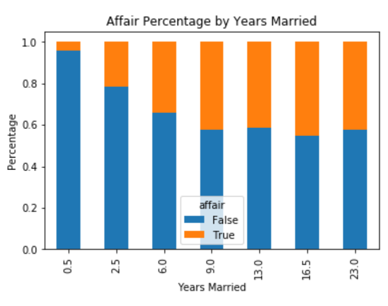
In [19]:
```
# barplot of marriage rating grouped by affair (True or False)
pd.crosstab(df.rate_marriage, df.affair.astype(bool)).plot(kind='bar')
plt.title('Marriage Rating Distribution by Affair Status')
plt.xlabel('Marriage Rating')
plt.ylabel('Frequency')
```

Out[19]: Text(0,0.5,'Frequency')

In [20]:
```
#Let's use a stacked barplot to look at the percentage of women having affairs by number of
 years of marriage.

affair_yrs_married = pd.crosstab(df.yrs_married, df.affair.astype(bool))
affair_yrs_married.div(affair_yrs_married.sum(1).astype(float), axis=0).plot(kind='bar', sta
cked=True)
plt.title('Affair Percentage by Years Married')
plt.xlabel('Years Married')
plt.ylabel('Percentage')
```

Out[20]: Text(0,0.5,'Percentage')



Prepare Data for Logistic Regression

In [21]:
```
# create dataframes with an intercept column and dummy variables for occupation and occupati
on_husb
y, X = dmatrices('affair ~ rate_marriage + age + yrs_married + children + \
                  religious + educ + C(occupation) + C(occupation_husb)',
                  df, return_type="dataframe")
X.columns
```

Out[21]:
```
Index(['Intercept', 'C(occupation)[T.2.0]', 'C(occupation)[T.3.0]',
       'C(occupation)[T.4.0]', 'C(occupation)[T.5.0]', 'C(occupation)[T.6.0]',
       'C(occupation_husb)[T.2.0]', 'C(occupation_husb)[T.3.0]',
       'C(occupation_husb)[T.4.0]', 'C(occupation_husb)[T.5.0]',
       'C(occupation_husb)[T.6.0]', 'rate_marriage', 'age', 'yrs_married',
       'children', 'religious', 'educ'],
      dtype='object')
```

In [24]:
```
# 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'})
```

In [25]:
```
# flatten y into a 1-D array
y = np.ravel(y)
```

Logistic Regression

In [27]:
```
from sklearn.linear_model import LogisticRegression

# 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)
```

Out[27]: 0.7258875274897895

In [28]:
```
# what percentage had affairs?
y.mean()
```

Out[28]: 0.3224945020420987

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.

In [29]:
```
# examine the coefficients
X.columns, np.transpose(model.coef_)
```

Out[29]:
```
(Index(['Intercept', 'occ_2', 'occ_3', 'occ_4', 'occ_5', 'occ_6', 'occ_husb_2',
        'occ_husb_3', 'occ_husb_4', 'occ_husb_5', 'occ_husb_6', 'rate_marriage',
        'age', 'yrs_married', 'children', 'religious', 'educ'],
       dtype='object'), array([[ 1.48983589],
        [ 0.18806639],
        [ 0.49894787],
        [ 0.25066856],
        [ 0.83900806],
        [ 0.83390843],
        [ 0.19063594],
        [ 0.29783271],
        [ 0.16140885],
        [ 0.18777091],
        [ 0.19401637],
        [-0.70312336],
        [-0.05841777],
        [ 0.10567654],
        [ 0.01691927],
        [-0.37113627],
        [ 0.0040165 ]]))
```

Model Evaluation Using a Validation Set

In [31]:
```
from sklearn.model_selection import train_test_split

# 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)
```

Out[31]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [32]:
```
# predict class labels for the test set

predicted = model2.predict(X_test)
predicted
```

Out[32]: array([1., 0., 0., ..., 0., 0., 0.])

In [33]:
```
# generate class probabilities

probs = model2.predict_proba(X_test)
probs
```

Out[33]:
```
array([[0.3514634 , 0.6485366 ],
       [0.90955084, 0.09044916],
       [0.72567333, 0.27432667],
       ...,
       [0.55727385, 0.44272615],
       [0.81207043, 0.18792957],
       [0.74734601, 0.25265399]])
```

The classifier is predicting a 1 (having an affair) any time the probability in the second column is greater than 0.5.

In [37]:
```
from sklearn.metrics import classification_report,accuracy_score,roc_auc_score,confusion_mat
rix

# generate evaluation metrics

print(accuracy_score(y_test, predicted))
print(roc_auc_score(y_test, probs[:, 1]))
```

```
0.7298429319371728
0.745950606950631
```

The accuracy is 73%, which is the same as we experienced when training and predicting on the same data.

In [38]:
```
print(confusion_matrix(y_test, predicted))
print(classification_report(y_test, predicted))
```

```
[[1169  134]
 [ 382  225]]
             precision    recall  f1-score   support

        0.0       0.75      0.90      0.82      1303
        1.0       0.63      0.37      0.47       607

avg / total       0.71      0.73      0.71      1910
```

Model Evaluation Using Cross-Validation

In [41]:
```
from sklearn.model_selection import cross_val_score

# evaluate the model using 10-fold cross-validation

scores = cross_val_score(LogisticRegression(), X, y, scoring='accuracy', cv=10)

print(scores)

print(scores.mean())
```

```
[0.72100313 0.70219436 0.73824451 0.70597484 0.70597484 0.72955975
 0.7327044  0.70440252 0.75157233 0.75      ]
0.7241630685514876
```

It's still performing at 73% accuracy.