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 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

In [1]:
```
## Import modules
```

In [2]:
```
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.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import cross_val_score
#from statsmodels.formula.api import logit, probit, poisson, ols
```

```
C:\Users\prashant_gupta1\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmodels\c
ompat\pandas.py:56: FutureWarning: The 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
```

Data Pre-Processing

First, let's load the dataset and add a binary `affair` column. `affair` column will contain 1 and 0 values only. 1 represents having affairs, 0 represents not

In [3]:
```
dta = sm.datasets.fair.load_pandas().data
# add "affair" column: 1 represents having affairs, 0 represents not
dta['affair'] = (dta.affairs > 0).astype(int)
```

In [4]:
```
dta.head()
```

Out[4]:

	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

In [5]:
```
# Additional Information
print(sm.datasets.fair.SOURCE)
```

```
Fair, Ray. 1978. "A Theory of Extramarital Affairs," `Journal of Political
Economy`, February, 45-61.

The data is available at http://fairmodel.econ.yale.edu/rayfair/pdf/2011b.htm (http://fair
model.econ.yale.edu/rayfair/pdf/2011b.htm)
```

In [6]:
```python
# Additional Information

print( sm.datasets.fair.NOTE)
```

```
::

    Number of observations: 6366
    Number of variables: 9
    Variable name definitions:

        rate_marriage   : How rate marriage, 1 = very poor, 2 = poor, 3 = fair,
                        4 = good, 5 = very good
        age             : Age
        yrs_married     : No. years married. Interval approximations. See
                        original paper for detailed explanation.
        children        : No. children
        religious       : How relgious, 1 = not, 2 = mildly, 3 = fairly,
                        4 = strongly
        educ            : Level of education, 9 = grade school, 12 = high
                        school, 14 = some college, 16 = college graduate,
                        17 = some graduate school, 20 = advanced degree
        occupation      : 1 = student, 2 = farming, agriculture; semi-skilled,
                        or unskilled worker; 3 = white-colloar; 4 = teacher
                        counselor social worker, nurse; artist, writers;
                        technician, skilled worker, 5 = managerial,
                        administrative, business, 6 = professional with
                        advanced degree
        occupation_husb : Husband's occupation. Same as occupation.
        affairs         : measure of time spent in extramarital affairs

    See the original paper for more details.
```

In [7]:
```python
dta.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6366 entries, 0 to 6365
Data columns (total 10 columns):
rate_marriage      6366 non-null float64
age                6366 non-null float64
yrs_married        6366 non-null float64
children           6366 non-null float64
religious          6366 non-null float64
educ               6366 non-null float64
occupation         6366 non-null float64
occupation_husb    6366 non-null float64
affairs            6366 non-null float64
affair             6366 non-null int32
dtypes: float64(9), int32(1)
memory usage: 472.6 KB
```

In [8]:
```
dta.describe()
```

Out[8]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupatior
count	6366.000000	6366.000000	6366.000000	6366.000000	6366.000000	6366.000000	6366.000000	6366.
mean	4.109645	29.082862	9.009425	1.396874	2.426170	14.209865	3.424128	3.
std	0.961430	6.847882	7.280120	1.433471	0.878369	2.178003	0.942399	1.
min	1.000000	17.500000	0.500000	0.000000	1.000000	9.000000	1.000000	1.
25%	4.000000	22.000000	2.500000	0.000000	2.000000	12.000000	3.000000	3.
50%	4.000000	27.000000	6.000000	1.000000	2.000000	14.000000	3.000000	4.
75%	5.000000	32.000000	16.500000	2.000000	3.000000	16.000000	4.000000	5.
max	5.000000	42.000000	23.000000	5.500000	4.000000	20.000000	6.000000	6.

Data Exploration

In [9]:
```
dta.groupby('affair').mean()
```

Out[9]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupation_husb	aff
affair									
0	4.329701	28.390679	7.989335	1.238813	2.504521	14.322977	3.405286	3.833758	0.000
1	3.647345	30.537019	11.152460	1.728933	2.261568	13.972236	3.463712	3.884559	2.187

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

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

Out[10]:

	age	yrs_married	children	religious	educ	occupation	occupation_husb	affairs	
rate_marriage									
1.0	33.823232	13.914141	2.308081	2.343434	13.848485	3.232323	3.838384	1.201671	0
2.0	30.471264	10.727011	1.735632	2.330460	13.864943	3.327586	3.764368	1.615745	0
3.0	30.008056	10.239174	1.638469	2.308157	14.001007	3.402820	3.798590	1.371281	0
4.0	28.856601	8.816905	1.369536	2.400981	14.144514	3.420161	3.835861	0.674837	0
5.0	28.574702	8.311662	1.252794	2.506334	14.399776	3.454918	3.892697	0.348174	0

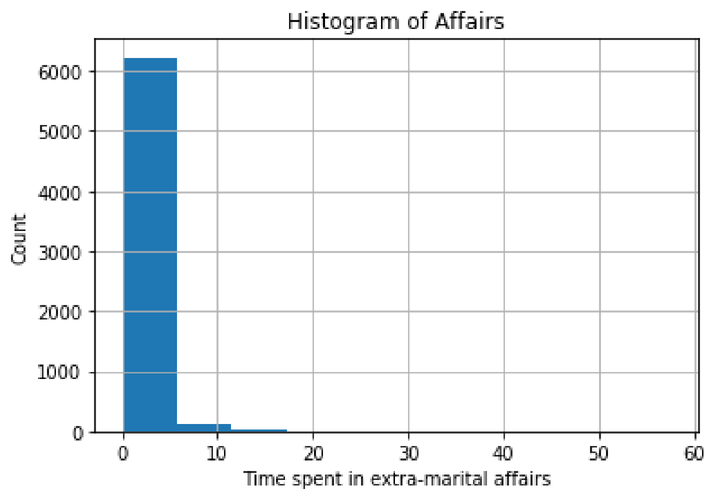
Above analysis shows that, an increase in `age`, `yrs_married`, and `children` appears to correlated with a decline in `marriage rate`.

Data Visualization

In [11]:
```
# show plots in the notebook
%matplotlib inline
```

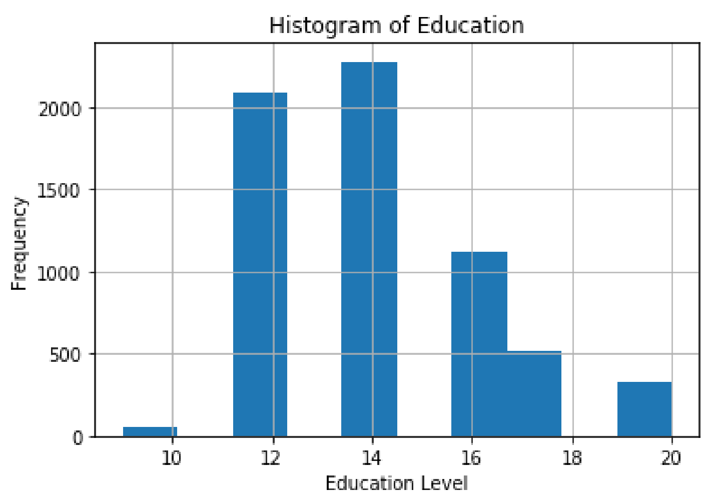
In [12]:
```
# histogram of affairs
dta.affairs.hist()
plt.title('Histogram of Affairs')
plt.xlabel('Time spent in extra-marital affairs')
plt.ylabel('Count')
```

Out[12]: `Text(0,0.5,'Count')`



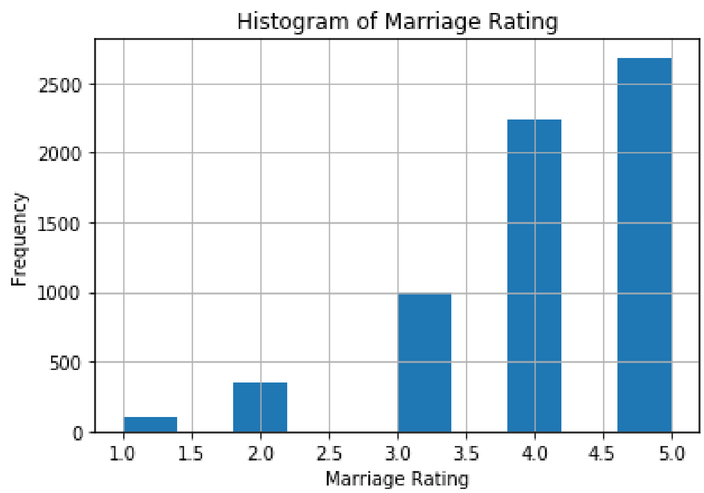
In [13]:
```
# histogram of education
dta.educ.hist()
plt.title('Histogram of Education')
plt.xlabel('Education Level')
plt.ylabel('Frequency')
```

Out[13]: `Text(0,0.5,'Frequency')`



In [14]:
```
# histogram of marriage rating
dta.rate_marriage.hist()
plt.title('Histogram of Marriage Rating')
plt.xlabel('Marriage Rating')
plt.ylabel('Frequency')
```

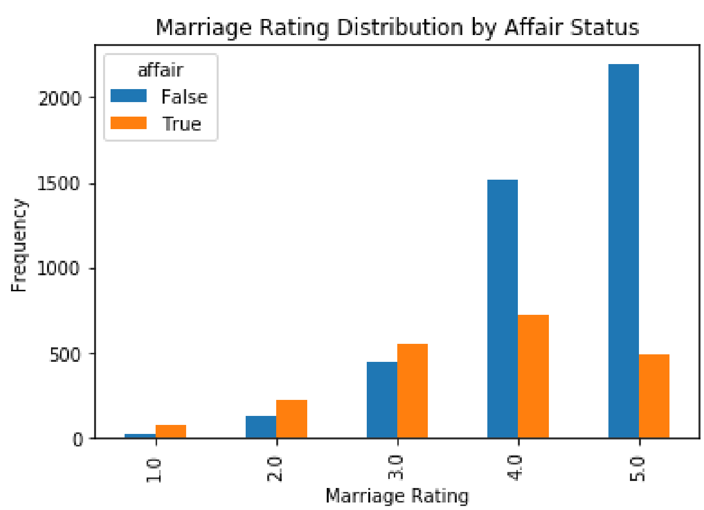
Out[14]: `Text(0,0.5,'Frequency')`



Let's take a look at the distribution of marriage ratings for those having affairs versus those not having affairs.

In [15]:
```
# barplot of 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')
```

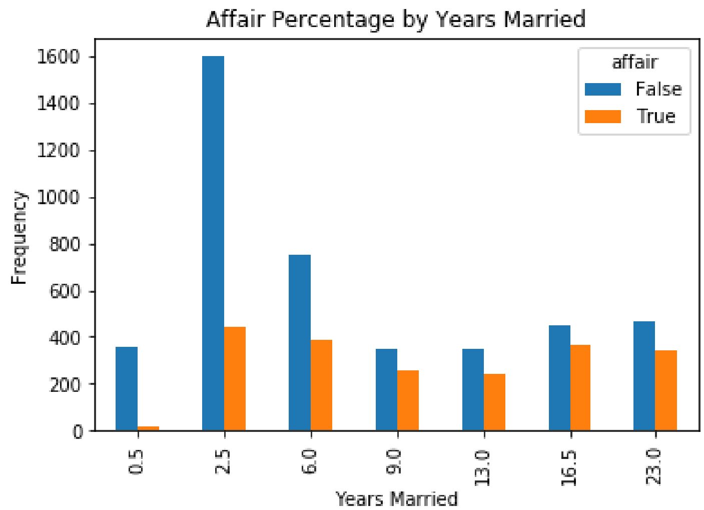
Out[15]: `Text(0,0.5,'Frequency')`



Let's use a stacked barplot to look at the percentage of women having affairs by number of years of marriage.

In [16]:
```
pd.crosstab(dta.yrs_married, dta.affair.astype(bool)).plot(kind='bar')
plt.title('Affair Percentage by Years Married')
plt.xlabel('Years Married')
plt.ylabel('Frequency')
```

Out[16]: `Text(0,0.5,'Frequency')`



In [17]:
```
dta.shape
```

Out[17]: `(6366, 10)`

Prepare Data for Logistic Regression

Looking at the information given by using following command

`print( sm.datasets.fair.NOTE)`

It looks like that `occupation` and `occupation_husb` , are the categorial variables. Here I would be using the dmatrices function from the [patsy module (http://patsy.readthedocs.org/en/latest/)](http://patsy.readthedocs.org/en/latest/) can do that using formula language.

In [18]:
```
# create dataframes with an intercept column and dummy variables for# cre
# 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")
print (X.columns)
```

```
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 [19]:
```
# 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 [20]:
```
# flatten y into a 1-D array
y = np.ravel(y)
```

In [21]:
```
# 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[21]:
```
0.7258875274897895
```

In [22]:
```
# what percentage had affairs?
y.mean()
```

Out[22]:
```
0.3224945020420987
```

In [23]:
```
# 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[23]:
```
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 [24]:
```
# predict class labels for the test set
predicted = model2.predict(X_test)
print(predicted)
```

```
[1. 0. 0. ... 0. 0. 0.]
```

In [25]:
```
probs = model2.predict_proba(X_test)
print(probs)
```

```
[[0.3514634  0.6485366 ]
 [0.90955084 0.09044916]
 [0.72567333 0.27432667]
 ...
 [0.55727385 0.44272615]
 [0.81207043 0.18792957]
 [0.74734601 0.25265399]]
```

In [26]:
```python
# generate evaluation metrics
print (metrics.accuracy_score(y_test, predicted))
print (metrics.roc_auc_score(y_test, predicted))
```

```
0.7298429319371728
0.6339179260634122
```

In [27]:
```python
print (metrics.confusion_matrix(y_test, predicted))
print (metrics.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
```

In [28]:
```python
# 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
```

In [29]:
```python
pd.DataFrame(list(zip(X.columns, np.transpose(model.coef_))))
```

Out[29]:

	0	1
0	Intercept	[1.489835891324933]
1	occ_2	[0.18806639024440983]
2	occ_3	[0.4989478668156914]
3	occ_4	[0.25066856498524825]
4	occ_5	[0.8390080648117001]
5	occ_6	[0.8339084337443315]
6	occ_husb_2	[0.1906359445867889]
7	occ_husb_3	[0.2978327129263421]
8	occ_husb_4	[0.1614088540760616]
9	occ_husb_5	[0.18777091388972483]
10	occ_husb_6	[0.19401637225511495]
11	rate_marriage	[-0.7031233597323255]
12	age	[-0.05841777448168919]
13	yrs_married	[0.10567653799735635]
14	children	[0.016919266970905608]
15	religious	[-0.3711362653137546]
16	educ	[0.00401650319563816]

In [30]:
```
model.predict_proba(np.array([[1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 3, 25, 3, 1, 4, 16]]))
```

Out[30]:
```
array([[0.77472221, 0.22527779]])
```