Machine Learning Assiagnment 3

Classify whether a woman in the given sample of data had an affair.

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

  (1 = student, 2 = farming/semi-skilled/unskilled, 3 ="white collar", 4 = teacher/nurse/writer/technician/skilled, 5 = managerial/business, 6 = professional with advanced degree)
- affairs: time spent in extra-marital affairs

In [1]:

```
#import 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.model_selection import cross_val_score
dta = sm.datasets.fair.load_pandas().data
df_affair = dta.copy()
```

```
C:\Users\santhu\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41:
DeprecationWarning: This module was deprecated in version 0.18 in favor of t
he model_selection module into which all the refactored classes and function
s are moved. Also note that the interface of the new CV iterators are differ
ent from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [2]:

```
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)
```

In [3]:

```
dta.head()
```

Out[3]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupation_husb	
0	3.0	32.0	9.0	3.0	3.0	17.0	2.0	5.0	0
1	3.0	27.0	13.0	3.0	1.0	14.0	3.0	4.0	3.
2	4.0	22.0	2.5	0.0	1.0	16.0	3.0	5.0	1.
3	4.0	37.0	16.5	4.0	3.0	16.0	5.0	5.0	0.
4	5.0	27.0	9.0	1.0	1.0	14.0	3.0	4.0	4.

In [4]:

```
dta.shape
```

Out[4]:

```
(6366, 10)
```

In [5]:

```
X.head()
```

Out[5]:

	Intercept	occ_2	occ_3	occ_4	occ_5	occ_6	occ_husb_2	occ_husb_3	occ_husb_4	occ_
0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	
2	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
4	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	

In [6]:

```
y
```

Out[6]:

```
array([1., 1., 1., ..., 0., 0., 0.])
```

Summary Statistics & Looking at the data

In [7]:

```
dta.describe()
```

Out[7]:

	rate_marriage	age	yrs_married	children	religious	educ	occup
count	6366.000000	6366.000000	6366.000000	6366.000000	6366.000000	6366.000000	6366.0(
mean	4.109645	29.082862	9.009425	1.396874	2.426170	14.209865	3.4
std	0.961430	6.847882	7.280120	1.433471	0.878369	2.178003	0.9
min	1.000000	17.500000	0.500000	0.000000	1.000000	9.000000	1.0(
25%	4.000000	22.000000	2.500000	0.000000	2.000000	12.000000	3.0(
50%	4.000000	27.000000	6.000000	1.000000	2.000000	14.000000	3.0(
75%	5.000000	32.000000	16.500000	2.000000	3.000000	16.000000	4.0(
max	5.000000	42.000000	23.000000	5.500000	4.000000	20.000000	6.0(

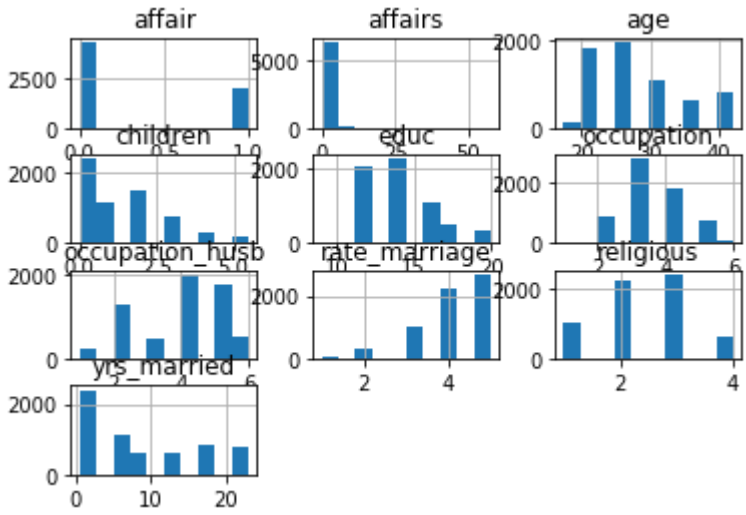
In [8]:

```
# plot all of the columns
%matplotlib inline
plt.figure(figsize=(20,18))
dta.hist()
```

Out[8]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000000000B6F3DA0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000B7BF0F0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000B7B4400
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000000000B829710
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000B6B7A20
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000B6B7A58
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000000000B7AA080
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000B58B390
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000B5B46A0
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000000000B5DD9B0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000B845CC0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000000B86EFD0
>]],
      dtype=object)
```

```
<Figure size 1440x1296 with 0 Axes>
```



Split the data into training and test set

In [9]:

```
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)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(4456, 17)
(4456,)
(1910, 17)
(1910,)
```

Logistic Regression with statsmodel

In [10]:

```
#Since we're doing a logistic regression, we're going to use the statsmodels Logit function
logit = sm.Logit(y_train, X_train)

# fit the model
result = logit.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.544479
         Iterations 6
```

Predictions by the statsmodel

In [11]:

```
predictions =  result.predict(X_test)
```

In [12]:

```
predictions
```

Out[12]:

```
2764    0.653211
4481    0.087718
5360    0.273074
5802    0.249471
1220    0.249630
5812    0.166215
3719    0.160619
3848    0.202858
1865    0.760648
2535    0.310242
2505    0.104535
6273    0.186280
3710    0.075798
4229    0.300914
1262    0.736723
5321    0.593884
3790    0.296514
994     0.732196
5644    0.296810
2252    0.156072
1804    0.203707
861     0.448163
1601    0.089842
1718    0.471631
2976    0.168971
3603    0.161252
4130    0.396058
5824    0.361373
5901    0.252252
4408    0.087566
          ...
5615    0.242151
1737    0.515613
2701    0.185493
4024    0.419446
1012    0.129569
3888    0.143331
4746    0.207049
5607    0.132057
1946    0.874461
3119    0.133824
156     0.604838
1752    0.653266
624     0.612704
4622    0.556290
1788    0.693977
500     0.264134
726     0.220550
4162    0.087718
48      0.158934
1691    0.572606
5882    0.152058
2244    0.395267
1985    0.841614
2853    0.223091
```

```
18      0.803795
3053    0.144139
1875    0.207506
5851    0.437646
4962    0.190124
1995    0.249630
Length: 1910, dtype: float64
```

scipy version 1.0.0 does not support chisqprob, this has been missed.

Hence we might get an error AttributeError: module 'scipy.stats' has no attribute 'chisqprob'. Below code is the workaround.

In [13]:

```
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
```

Interpreting the results

In [14]:

```
result.summary()
```

Out[14]:

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	4456
Model:	Logit	**Df Residuals:**	4439
Method:	MLE	**Df Model:**	16
Date:	Mon, 22 Oct 2018	**Pseudo R-squ.:**	0.1360
Time:	21:05:32	**Log-Likelihood:**	-2426.2
converged:	True	**LL-Null:**	-2808.3
		LLR p-value:	2.844e-152

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	2.4842	0.777	3.198	0.001	0.961	4.007
occ_2	0.9414	0.658	1.432	0.152	-0.347	2.230
occ_3	1.2324	0.652	1.890	0.059	-0.046	2.511
occ_4	0.9731	0.653	1.490	0.136	-0.307	2.254
occ_5	1.6017	0.657	2.436	0.015	0.313	2.890
occ_6	1.8242	0.707	2.581	0.010	0.439	3.209
occ_husb_2	0.0649	0.215	0.302	0.762	-0.356	0.486
occ_husb_3	0.1976	0.235	0.841	0.400	-0.263	0.658
occ_husb_4	0.0304	0.208	0.146	0.884	-0.377	0.438
occ_husb_5	-0.0052	0.210	-0.025	0.980	-0.417	0.406
occ_husb_6	-0.0183	0.236	-0.078	0.938	-0.481	0.445
rate_marriage	-0.7145	0.038	-18.929	0.000	-0.788	-0.640
age	-0.0577	0.012	-4.686	0.000	-0.082	-0.034
yrs_married	0.1081	0.013	8.243	0.000	0.082	0.134
children	-0.0126	0.038	-0.329	0.742	-0.088	0.062
religious	-0.3889	0.042	-9.342	0.000	-0.470	-0.307
educ	0.0046	0.021	0.224	0.823	-0.036	0.045

In [15]:

```
#sm.show_versions()
```

Logistic Regression with scikit-learn

In [16]:

```
dta.head()
```

Out[16]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupation_husb	
0	3.0	32.0	9.0	3.0	3.0	17.0	2.0	5.0	0
1	3.0	27.0	13.0	3.0	1.0	14.0	3.0	4.0	3.
2	4.0	22.0	2.5	0.0	1.0	16.0	3.0	5.0	1.
3	4.0	37.0	16.5	4.0	3.0	16.0	5.0	5.0	0.
4	5.0	27.0	9.0	1.0	1.0	14.0	3.0	4.0	4.

Data Exploration

In [17]:

```
# people having affair is represented with 1 and not having affair is represented with 0
dta.affair.value_counts()
```

Out[17]:

```
0    4313
1    2053
Name: affair, dtype: int64
```

In [18]:

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

Out[18]:

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

- We can see that on average, women who have affairs rate their marriages lower.

In [19]:

```
#### checking other parameter like rate_marriage
```

In [20]:

```
dta.groupby('rate_marriage').mean()
```

Out[20]:

rate_marriage	age	yrs_married	children	religious	educ	occupation	occupation_h
1.0	33.823232	13.914141	2.308081	2.343434	13.848485	3.232323	3.838
2.0	30.471264	10.727011	1.735632	2.330460	13.864943	3.327586	3.764
3.0	30.008056	10.239174	1.638469	2.308157	14.001007	3.402820	3.798
4.0	28.856601	8.816905	1.369536	2.400981	14.144514	3.420161	3.835
5.0	28.574702	8.311662	1.252794	2.506334	14.399776	3.454918	3.892

- An increase in age, yrs_married, and children appears to correlate with increase in affairs.

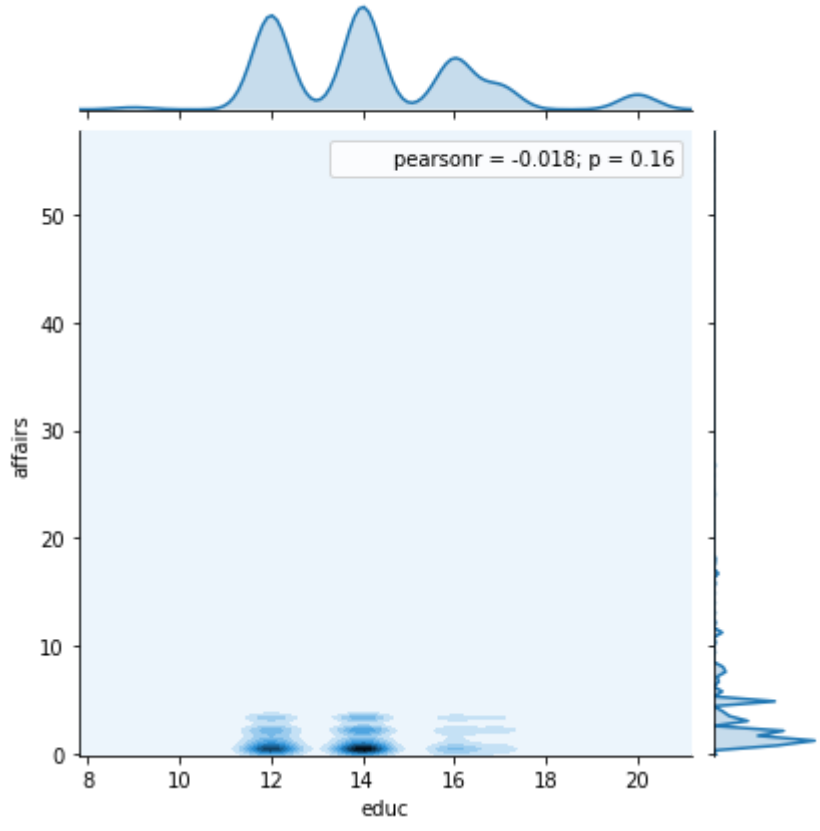
Data Visualization

In [21]:

```
import seaborn as sns
sns.jointplot(x='educ',y='affairs',data=dta,kind='kde')
```

Out[21]:

```
<seaborn.axisgrid.JointGrid at 0xc47bc18>
```

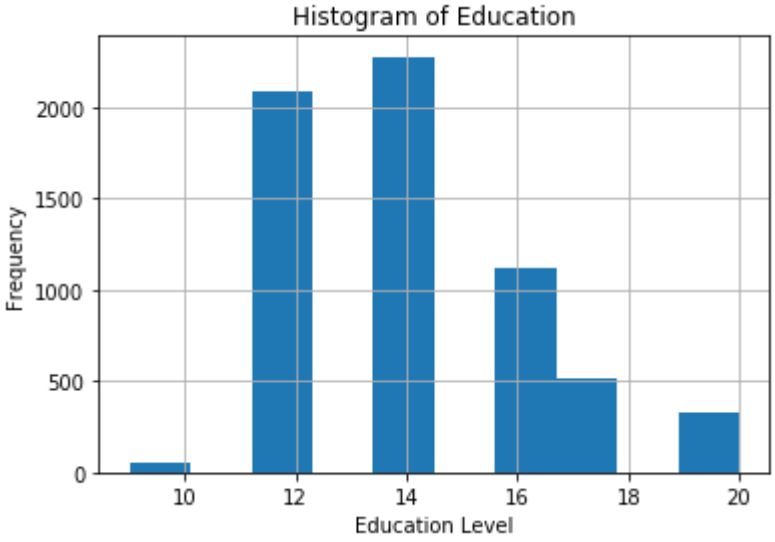


In [22]:

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

Out[22]:

```
Text(0,0.5,'Frequency')
```

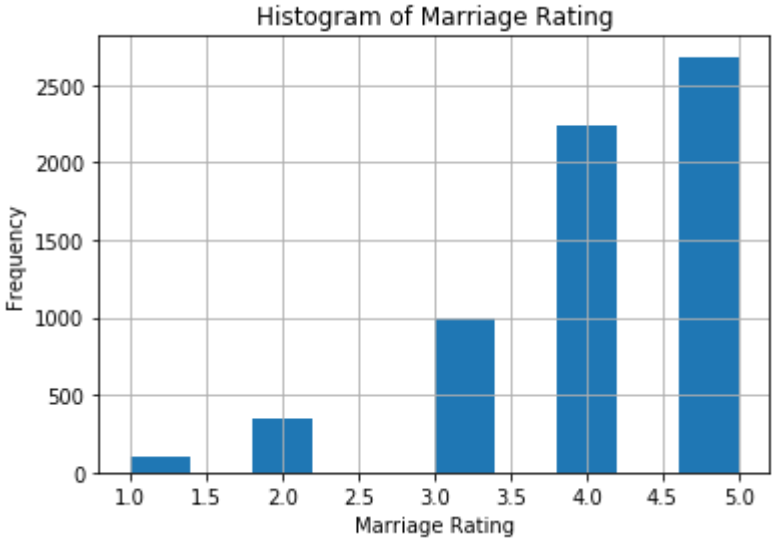


In [23]:

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

Out[23]:

```
Text(0,0.5,'Frequency')
```

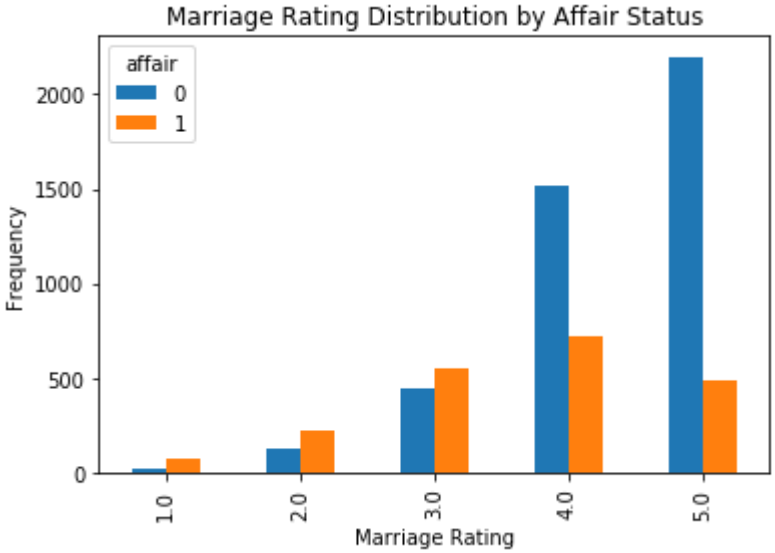


In [24]:

```
# barplot of marriage rating grouped by affair (True or False)
pd.crosstab(dta.rate_marriage, dta.affair).plot(kind='bar')
plt.title('Marriage Rating Distribution by Affair Status')
plt.xlabel('Marriage Rating')
plt.ylabel('Frequency')
```

Out[24]:

```
Text(0,0.5,'Frequency')
```

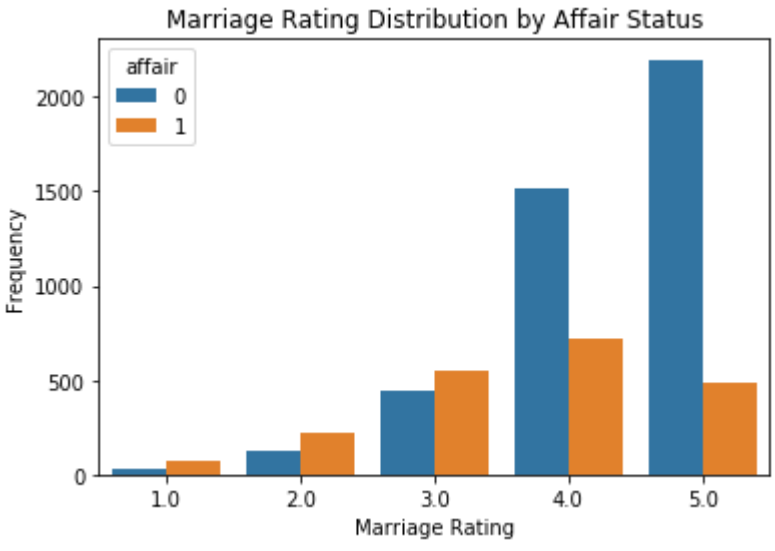


In [25]:

```
sns.countplot(x='rate_marriage',data=dta,hue='affair')
plt.title('Marriage Rating Distribution by Affair Status')
plt.xlabel('Marriage Rating')
plt.ylabel('Frequency')
```

Out[25]:

```
Text(0,0.5,'Frequency')
```

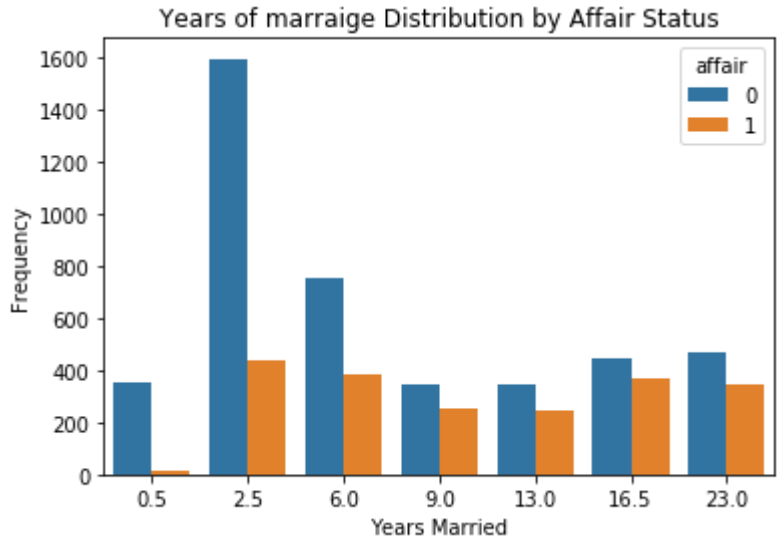


In [26]:

```
sns.countplot(x='yrs_married',data=dta,hue='affair')
plt.title('Years of marraige Distribution by Affair Status')
plt.xlabel('Years Married')
plt.ylabel('Frequency')
```

Out[26]:

```
Text(0,0.5,'Frequency')
```

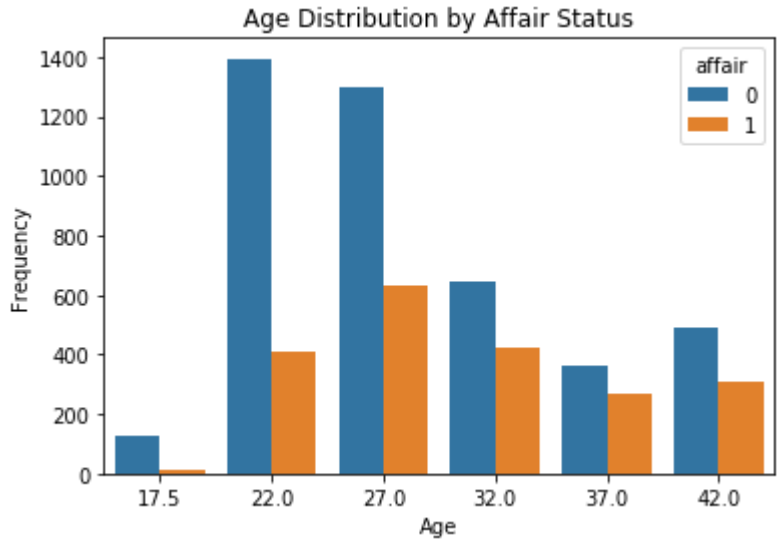


In [27]:

```
import seaborn as sns
sns.countplot(x='age',data=dta,hue='affair')
plt.title('Age Distribution by Affair Status')
plt.xlabel('Age')
plt.ylabel('Frequency')
```

Out[27]:

```
Text(0,0.5,'Frequency')
```



Model Evaluation Using a Validation Set

In [28]:

```
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)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(4456, 17)
(4456,)
(1910, 17)
(1910,)
```

Build the model and fit the training data

In [29]:

```
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[29]:

```
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 [30]:

```
model.score(X_train,y_train)
```

Out[30]:

```
0.723967684021544
```

- The training data set has an accuracy of 73 %

Use the test data set to predict the class / labels

In [31]:

```
# predict class labels for the test set
predicted = model.predict(X_test)
predicted
```

Out[31]:

```
array([1., 0., 0., ..., 0., 0., 0.])
```

In [32]:

```
# generate class probabilities
probs = model.predict_proba(X_test)
probs
```

Out[32]:

```
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]])
```

- As you can see, the classifier is predicting a 1 (having an affair) any time the probability in the second column is greater than 0.5.

Evaluate the model

In [33]:

```
# generate evaluation metrics
print(metrics.accuracy_score(y_test,predicted))
print(metrics.roc_auc_score(y_test, probs[:, 1]))
```

```
0.7298429319371728
0.745950606950631
```

- The accuracy of the model is 73% whic is the same as that encountered for training data.

In [34]:

```
# evaluate the model using 10-fold cross-validation
scores = cross_val_score(LogisticRegression(), X, y, scoring='accuracy', cv=10)
scores, scores.mean()
```

Out[34]:

```
(array([0.72100313, 0.70219436, 0.73824451, 0.70597484, 0.70597484,
        0.72955975, 0.7327044 , 0.70440252, 0.75157233, 0.75      ]),
 0.7241630685514876)
```

Predicting the Probability of an Affair

- let's predict 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

In [35]:

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

Out[35]:

```
array([[0.77301478, 0.22698522]])
```

In [ ]:

```

```