Data Science Masters :Assignment 21

Problem:

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.

Solution:

In [1]:

```
# Import 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
```

```
C:\Users\mkarthikeyan\AppData\Local\Continuum\anaconda3\lib\site-packages
\sklearn\cross_validation.py:41: DeprecationWarning: This module was depre
cated in version 0.18 in favor of the model_selection module into which al
l the refactored classes and functions are moved. Also note that the inter
face of the new CV iterators are different from that of this module. This
module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [3]:

```
# Data Pre processing
# load dataset
dta = sm.datasets.fair.load_pandas().data
dta.head(2)
```

Out[3]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupati
0	3.0	32.0	9.0	3.0	3.0	17.0	2.0	5.0
1	3.0	27.0	13.0	3.0	1.0	14.0	3.0	4.0

In [6]:

```
# add "affair" column: 1 represents having affairs, 0 represents not
dta['affair'] = (dta.affairs > 0).astype(int)
dta.head(2)
```

Out[6]:

	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupati
0	3.0	32.0	9.0	3.0	3.0	17.0	2.0	5.0
1	3.0	27.0	13.0	3.0	1.0	14.0	3.0	4.0

In [7]:

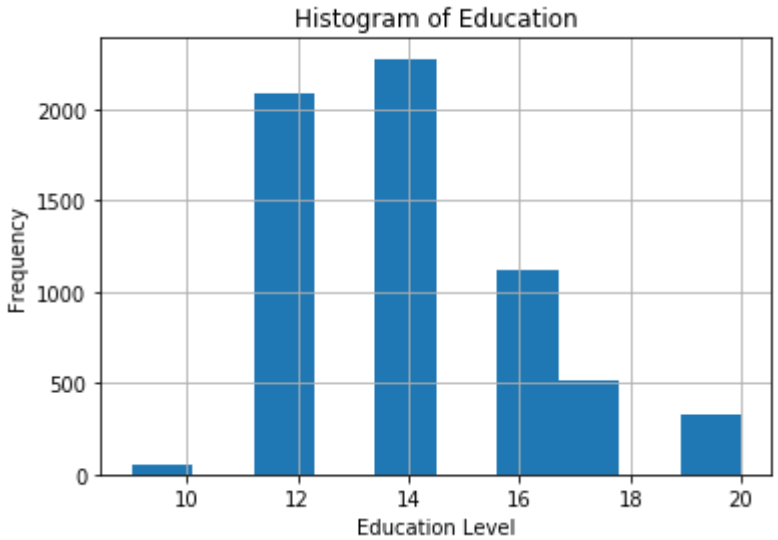
```
# Data Exploration
dta.groupby('affair').mean()
```

Out[7]:

	rate_marriage	age	yrs_married	children	religious	educ	occu
affair							
0	4.329701	28.390679	7.989335	1.238813	2.504521	14.322977	3.405
1	3.647345	30.537019	11.152460	1.728933	2.261568	13.972236	3.463

In [11]:

```
# Data Visualization
# show plots in the notebook
%matplotlib inline
# histogram of education
dta.educ.hist()
plt.title('Histogram of Education')
plt.xlabel('Education Level')
plt.ylabel('Frequency')
plt.show()
```

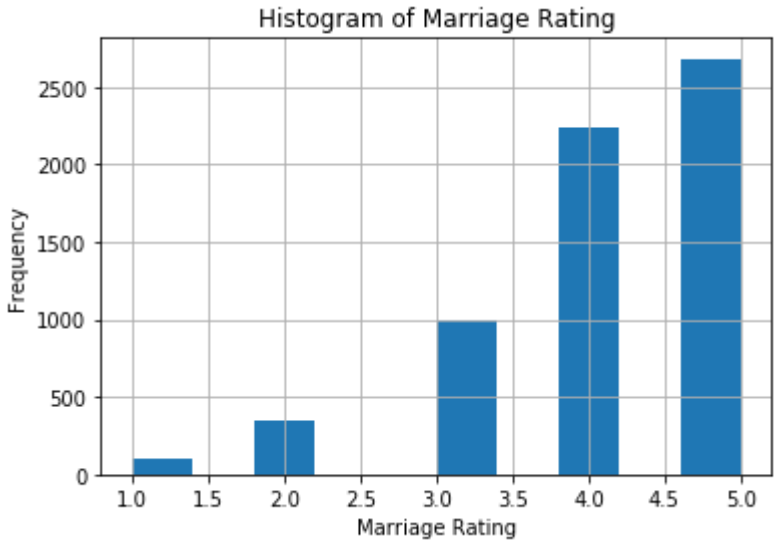


In [12]:

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

Out[12]:

Text(0,0.5,'Frequency')

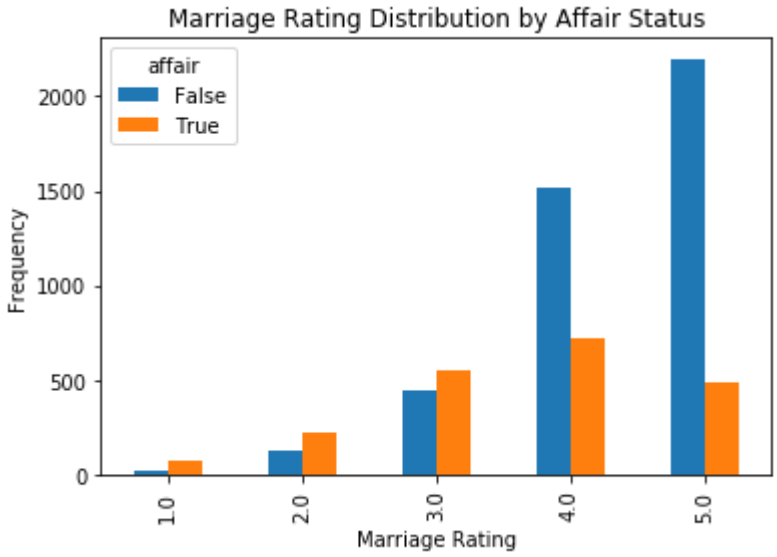


In [13]:

```
# 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[13]:

Text(0,0.5,'Frequency')



In [15]:

```
# Prepare Data for Logistic Regression
# 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
```

Out[15]:

```
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 [18]:

```
# 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 [19]:

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

Out[19]:

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

In [21]:

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

Out[21]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
          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 [22]:

```
# predict class labels for the test set
predicted = model2.predict(X_test)
predicted
```

Out[22]:

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

In [23]:

```
# generate class probabilities
probs = model2.predict_proba(X_test)
probs
```

Out[23]:

```
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]])
```

In [24]:

```
predicted_train = model2.predict(X_train)
print(metrics.accuracy_score(y_train, predicted_train))
```

```
0.723967684021544
```

In [25]:

```
# generate evaluation metrics
print(metrics.accuracy_score(y_test, predicted))
```

```
0.7298429319371728
```

In [26]:

```
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
```

Model Evaluation Using Cross-Validation

In [27]:

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

Out[27]:

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

Looks good. It's performing at 73% accuracy.