In [26]:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
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
dta = sm.datasets.fair.load_pandas().data

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 [27]:

```
cheaters = sm.datasets.fair.load_pandas().data
cheaters.describe()
```

Out[27]:

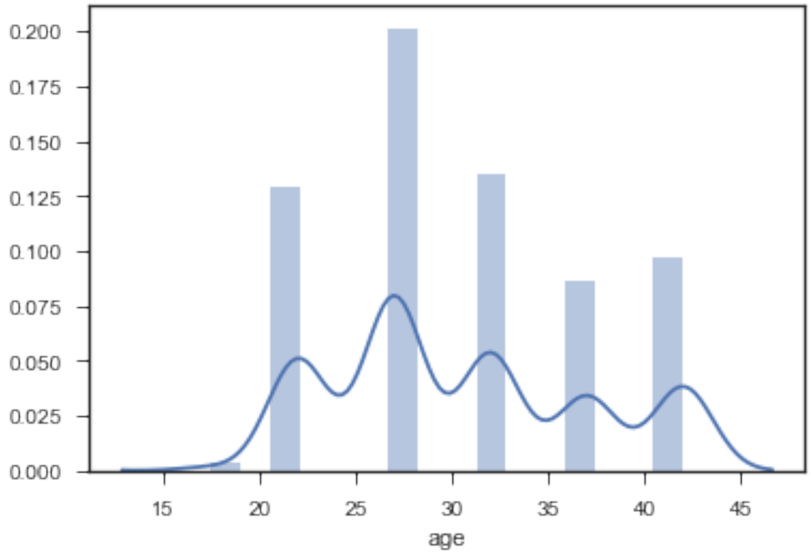
	rate_marriage	age	yrs_married	children	religious	edu
count	6366.000000	6366.000000	6366.000000	6366.000000	6366.000000	6366.00000
mean	4.109645	29.082862	9.009425	1.396874	2.426170	14.209865
std	0.961430	6.847882	7.280120	1.433471	0.878369	2.178003
min	1.000000	17.500000	0.500000	0.000000	1.000000	9.000000
25%	4.000000	22.000000	2.500000	0.000000	2.000000	12.000000
50%	4.000000	27.000000	6.000000	1.000000	2.000000	14.000000
75%	5.000000	32.000000	16.500000	2.000000	3.000000	16.000000
max	5.000000	42.000000	23.000000	5.500000	4.000000	20.000000

In [28]:

```
cheaters_age=cheaters[(cheaters['affairs'] > 0) & cheaters['age']]
```

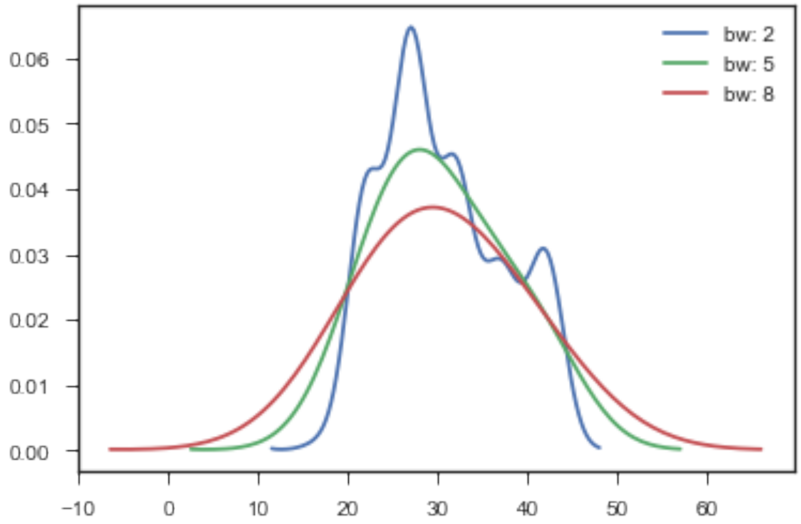
In [29]:

```
sns.distplot(cheaters_age.age);
```



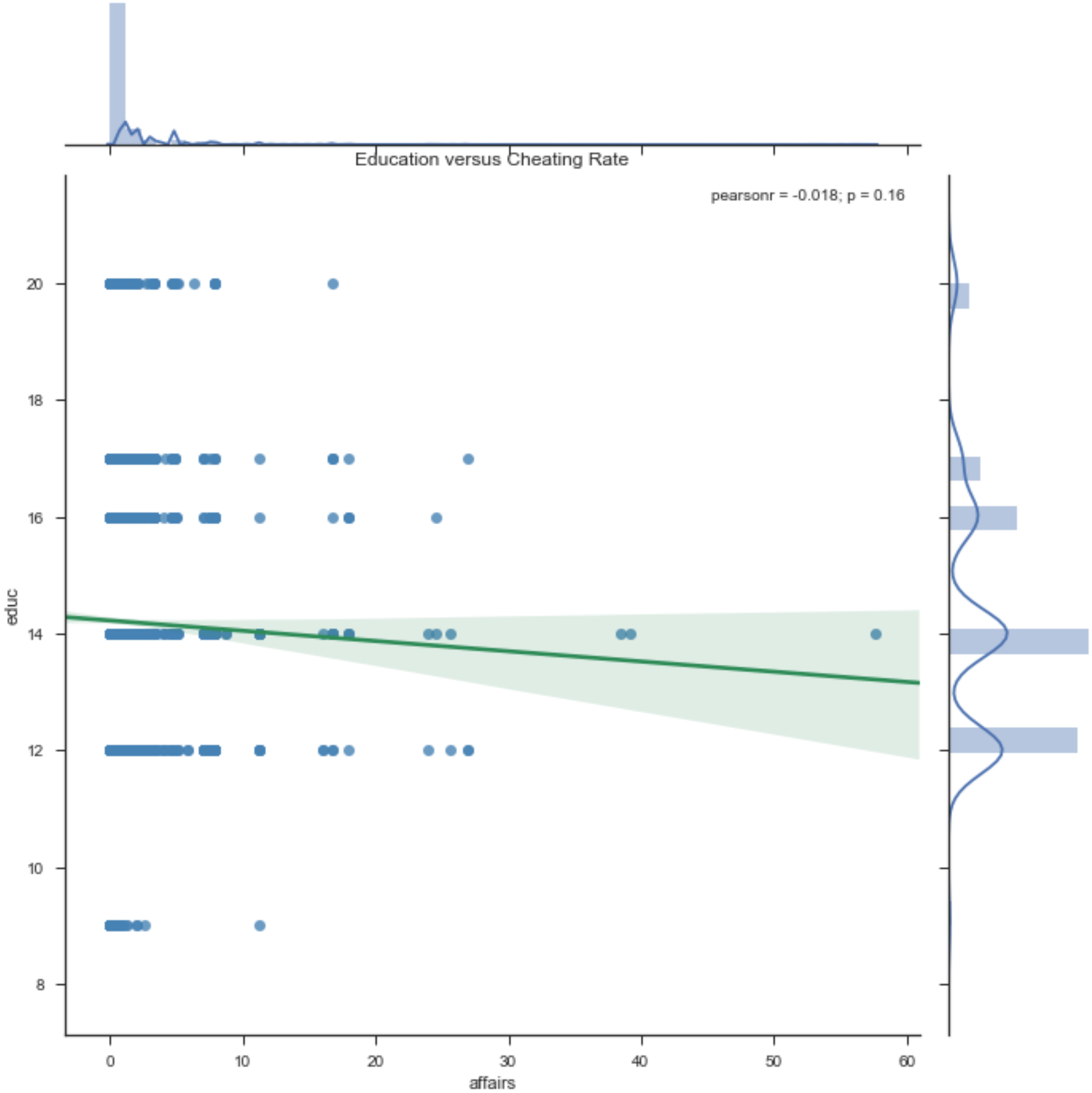
```
In [30]: sns.kdeplot(cheaters_age.age, bw=2, label="bw: 2")
         sns.kdeplot(cheaters_age.age, bw=5, label="bw: 5")
         sns.kdeplot(cheaters_age.age, bw=8, label="bw: 8")
```

```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x11a32fd68>
```



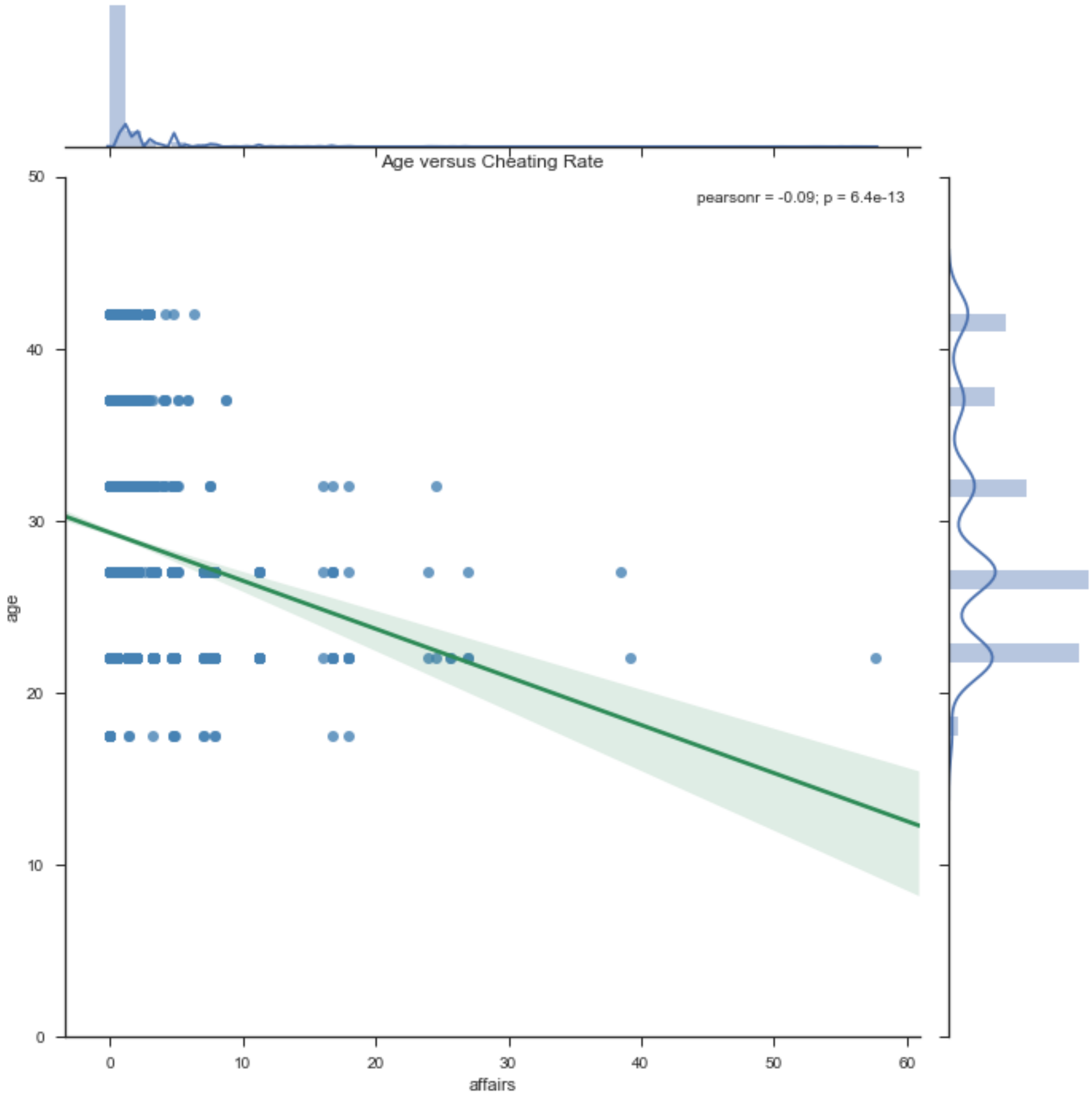
```
In [31]: sns.set(style='ticks')
         sns.jointplot(y="educ", x="affairs", data=cheaters, size=10, kind='reg'
         , joint_kws={'color':'steelblue'}, line_kws={'color':'seagreen'})
         plt.title("Education versus Cheating Rate")
```

```
Out[31]: <matplotlib.text.Text at 0x11a486080>
```

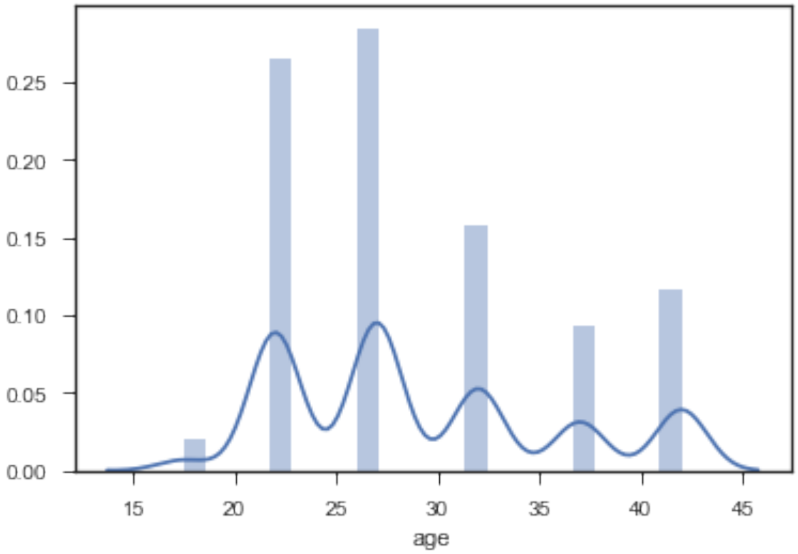


```
In [32]: sns.set(style='ticks')
         sns.jointplot(y="age", x="affairs", data=cheaters, size=10, kind='reg',
         joint_kws={'color':'steelblue'}, line_kws={'color':'seagreen'})
         plt.ylim(0,50) # set Y axis range to minimum of zero
         plt.title("Age versus Cheating Rate")
```

```
Out[32]: <matplotlib.text.Text at 0x11a34b898>
```

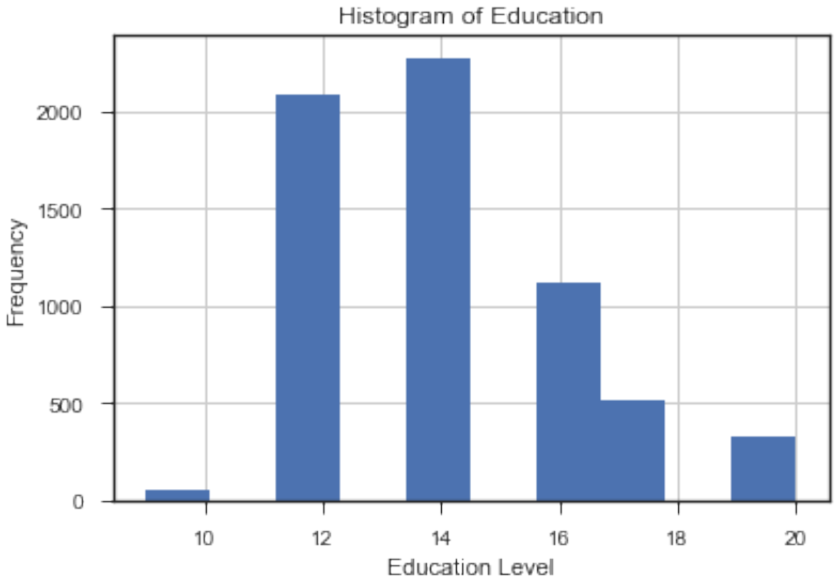


```
In [33]: %matplotlib inline
In [34]: sns.distplot(cheaters.age);
```



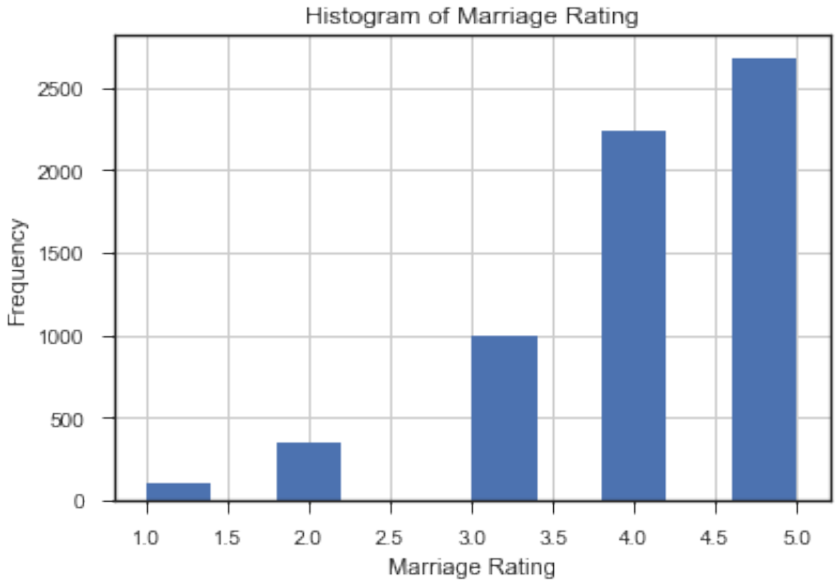
```
In [35]: # histogram of education
         dta.educ.hist()
         plt.title('Histogram of Education')
         plt.xlabel('Education Level')
         plt.ylabel('Frequency')
```

```
Out[35]: <matplotlib.text.Text at 0x11a5b9e80>
```



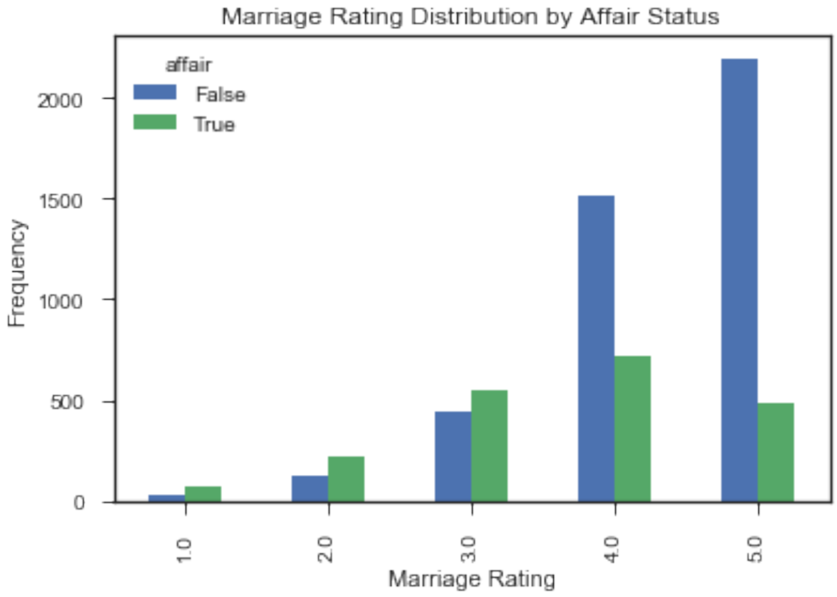
```
In [36]: dta.rate_marriage.hist()
         plt.title('Histogram of Marriage Rating')
         plt.xlabel('Marriage Rating')
         plt.ylabel('Frequency')

Out[36]: <matplotlib.text.Text at 0x11aed3828>
```



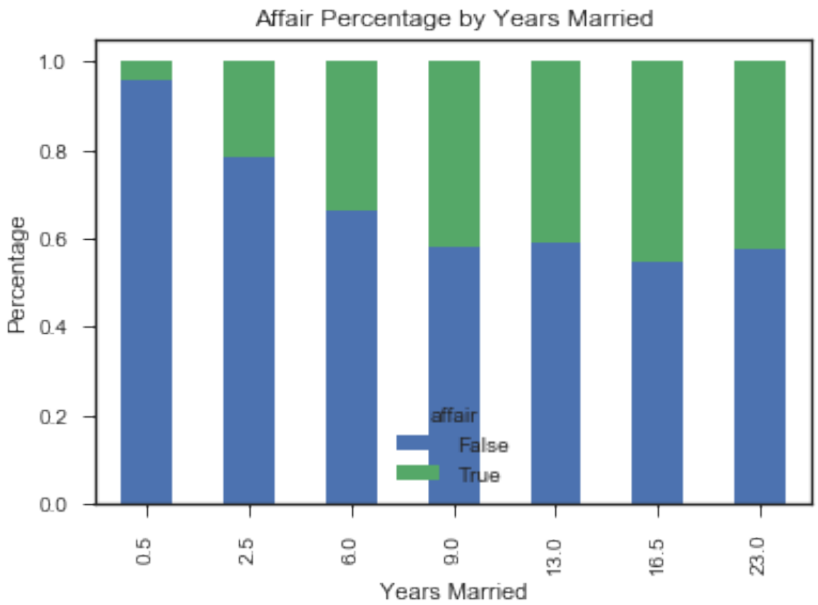
```
In [37]: pd.crosstab(dta.rate_marriage, dta.affair.astype(bool)).plot(kind='bar'
         )
         plt.title('Marriage Rating Distribution by Affair Status')
         plt.xlabel('Marriage Rating')
         plt.ylabel('Frequency')

Out[37]: <matplotlib.text.Text at 0x11a5f2cf8>
```



In [38]:
```
affair_yrs_married = pd.crosstab(dta.yrs_married, dta.affair.astype(boo
l))
affair_yrs_married.div(affair_yrs_married.sum(1).astype(float), axis=0)
.plot(kind='bar', stacked=True)
plt.title('Affair Percentage by Years Married')
plt.xlabel('Years Married')
plt.ylabel('Percentage')
```

Out[38]:
```
<matplotlib.text.Text at 0x11b28dd30>
```



In [39]:
```
model = LogisticRegression()
model = model.fit(X, y)
model.score(X, y)
```

Out[39]:
```
0.72588752748978946
```

In [40]:
```
y.mean()
```

Out[40]:
```
0.32249450204209867
```

In [41]:

```
X.columns, np.transpose(model.coef_)
```

Out[41]:

```
(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([[ 1.48986218],
        [ 0.18804163],
        [ 0.49891989],
        [ 0.25064098],
        [ 0.83897702],
        [ 0.83400806],
        [ 0.19057993],
        [ 0.29777985],
        [ 0.16135353],
        [ 0.18771785],
        [ 0.19394845],
        [-0.70311486],
        [-0.05841779],
        [ 0.10567662],
        [ 0.01692042],
        [-0.37113345],
        [ 0.00401539]]))
```

In [42]:

```
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[42]:

```
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=0.0
001,
          verbose=0, warm_start=False)
```

In [43]:

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

Out[43]:

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

```
In [44]: probs = model2.predict_proba(X_test)
         probs
```

```
Out[44]: array([[ 0.35142683,  0.64857317],
                [ 0.90952466,  0.09047534],
                [ 0.72576735,  0.27423265],
                ...,
                [ 0.55737244,  0.44262756],
                [ 0.81213767,  0.18786233],
                [ 0.74729529,  0.25270471]])
```

```
In [45]: print(metrics.accuracy_score(y_test, predicted))
         print(metrics.roc_auc_score(y_test, probs[:, 1]))
```

```
0.729842931937
0.74596198609
```

```
In [46]: 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 [47]: scores = cross_val_score(LogisticRegression(), X, y, scoring='accuracy'
         , cv=10)
         scores, scores.mean()
```

```
Out[47]: (array([ 0.72100313,  0.70219436,  0.73824451,  0.70597484,  0.7059748
         4,
                  0.72955975,  0.7327044 ,  0.70440252,  0.75157233,  0.75
         ]),
          0.7241630685514876)
```

```
In [48]: model.predict_proba(np.array([[1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 3, 25,
         3, 1, 4,
                                       16]]))
```

```
Out[48]: array([[ 0.77472417,  0.22527583]])
```