In [1]:
```
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.cross_validation import train_test_split ## cross_validation is
 deprecated
from sklearn.model_selection import train_test_split

from sklearn import metrics
#from sklearn.cross_validation import cross_val_score
# cross validation above was replaced my model_selection
from sklearn.model_selection import cross_val_score
dta = sm.datasets.fair.load_pandas().data

# add "affair" column: 1 represents having affairs, 0 represents not
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)
```

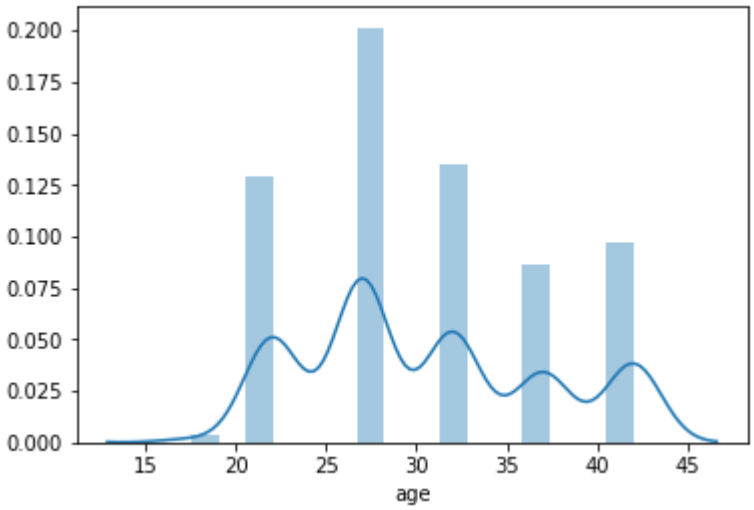
```
C:\Users\Kunal\Anaconda3\lib\site-packages\statsmodels\compat\pandas.py:56: F
utureWarning: The pandas.core.datetools module is deprecated and will be remo
ved in a future version. Please use the pandas.tseries module instead.
  from pandas.core import datetools
```

In [2]:
```
cheaters = sm.datasets.fair.load_pandas().data
cheaters.describe()
```

Out[2]:

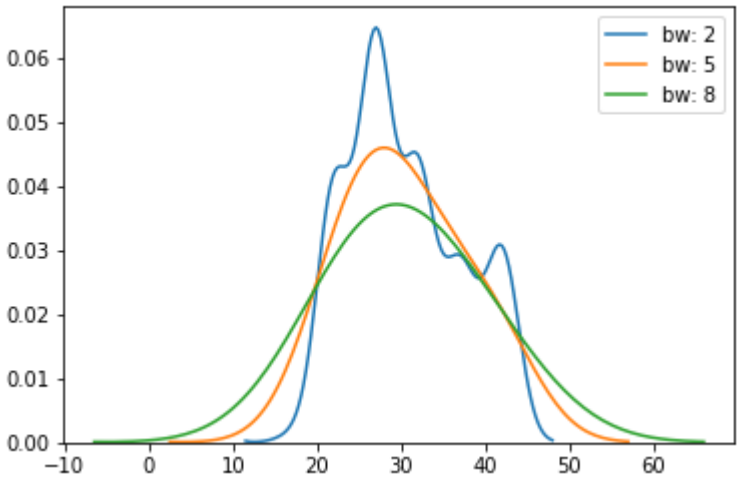
	rate_marriage	age	yrs_married	children	religious	educ
count	6366.000000	6366.000000	6366.000000	6366.000000	6366.000000	6366.000000
mean	4.109645	29.082862	9.009425	1.396874	2.426170	14.209865
std	0.961430	6.847882	7.280120	1.433471	0.878369	2.178003
min	1.000000	17.500000	0.500000	0.000000	1.000000	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 [3]:
```
cheaters_age=cheaters[(cheaters['affairs'] > 0) & cheaters['age']]
sns.distplot(cheaters_age.age);
```



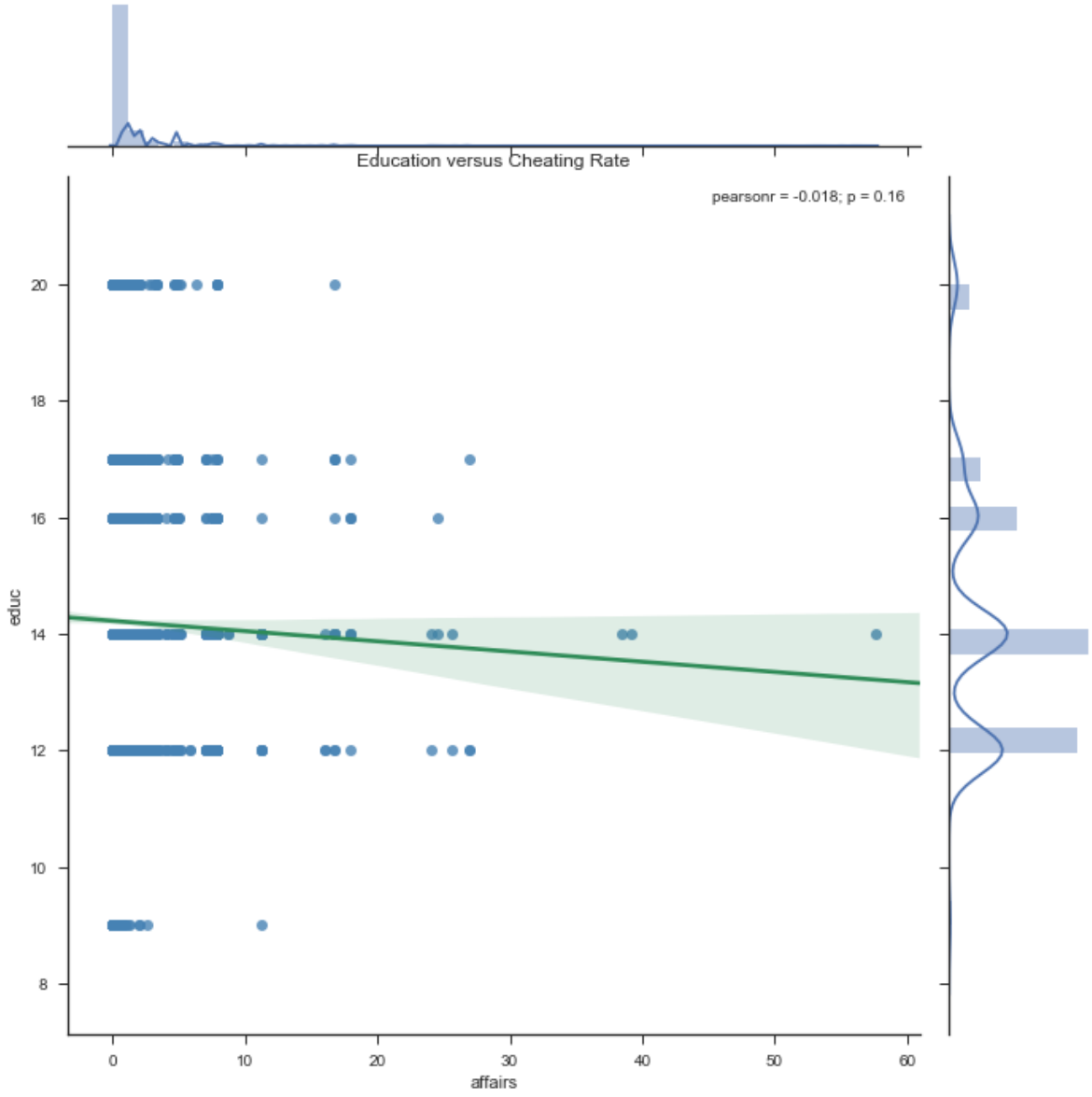
In [4]:
```
sns.kdeplot(cheaters_age.age, bw=2, label="bw: 2")
sns.kdeplot(cheaters_age.age, bw=5, label="bw: 5")
sns.kdeplot(cheaters_age.age, bw=8, label="bw: 8")
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0xb95b358>



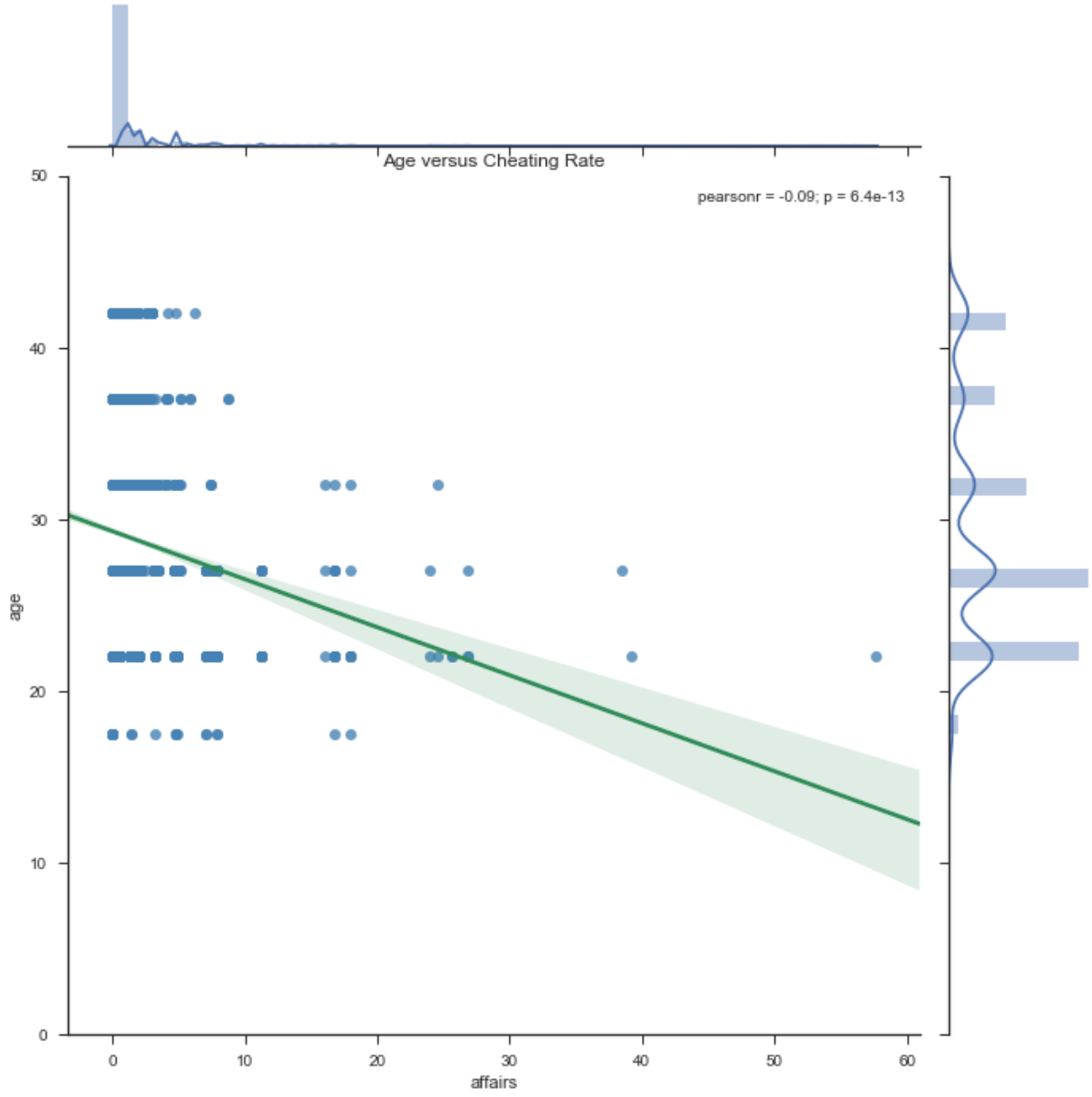
In [5]:
```
sns.set(style='ticks')
#fig, ax = plt.subplots()
# the size of A4 paper
#fig.set_size_inches(11.7, 8.27)
sns.jointplot(y="educ", x="affairs", data=cheaters, size=10, kind='reg', joint
_kws={'color':'steelblue'}, line_kws={'color':'seagreen'})
#plt.ylim(0,5) # set Y axis range to minimum of zero
plt.title("Education versus Cheating Rate")
```

Out[5]: Text(0.5,1,'Education versus Cheating Rate')

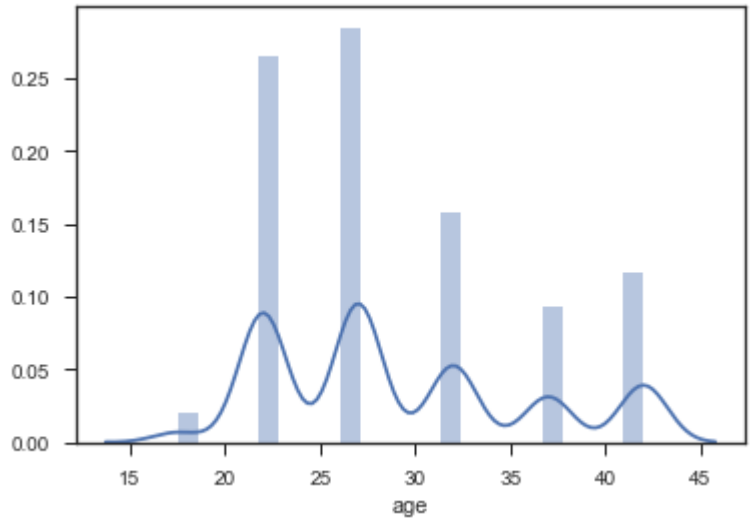


In [6]:
```
sns.set(style='ticks')
#fig, ax = plt.subplots()
# the size of A4 paper
#fig.set_size_inches(11.7, 8.27)
sns.jointplot(y="age", x="affairs", data=cheaters, size=10, kind='reg', joint_
kws={'color':'steelblue'}, line_kws={'color':'seagreen'})
plt.ylim(0,50) # set Y axis range to minimum of zero
plt.title("Age versus Cheating Rate")
```

Out[6]: Text(0.5,1,'Age versus Cheating Rate')

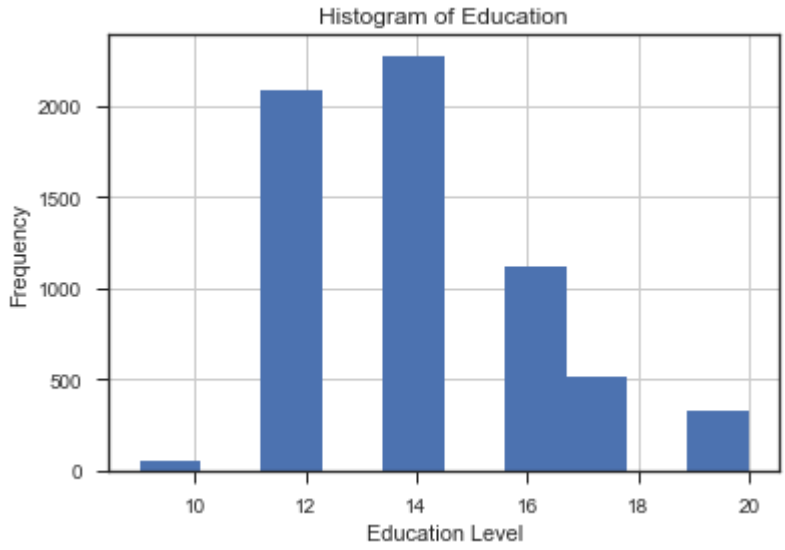


In [7]:
```
%matplotlib inline
sns.distplot(cheaters.age);
```



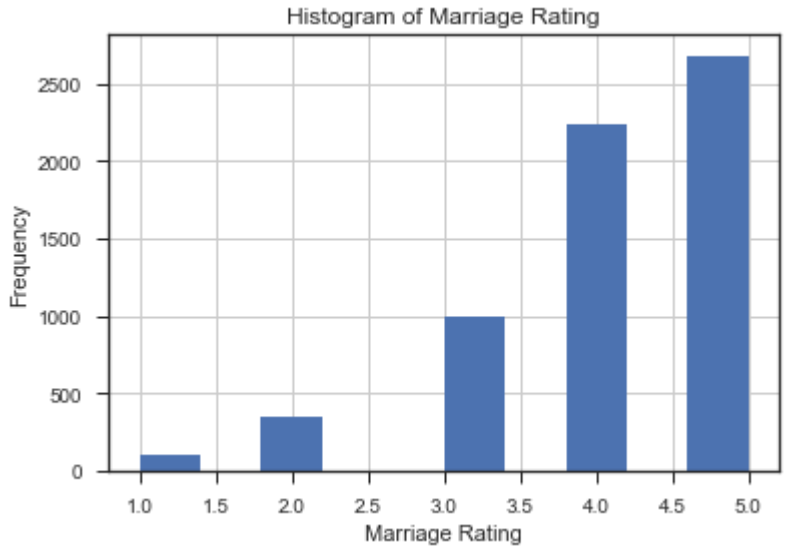
In [8]:
```
# histogram of education
dta.educ.hist()
plt.title('Histogram of Education')
plt.xlabel('Education Level')
plt.ylabel('Frequency')
```

Out[8]: Text(0,0.5,'Frequency')



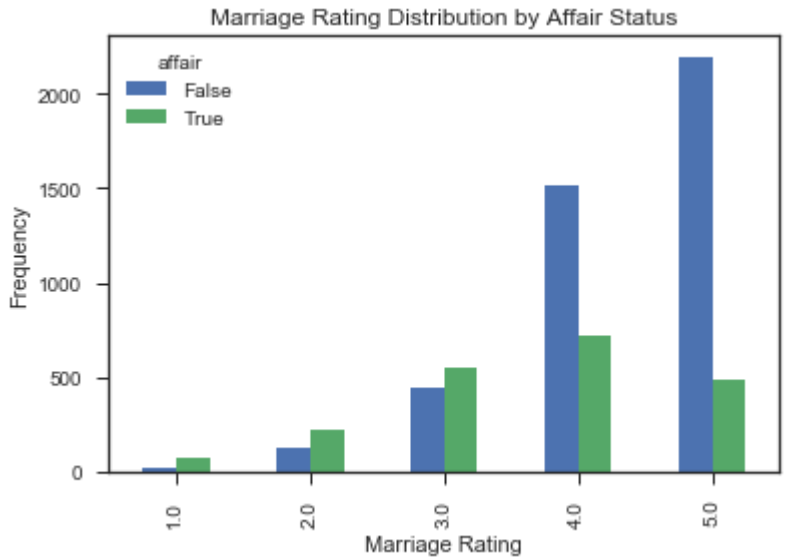
In [9]:
```
# histogram of marriage rating
dta.rate_marriage.hist()
plt.title('Histogram of Marriage Rating')
plt.xlabel('Marriage Rating')
plt.ylabel('Frequency')
```

Out[9]: Text(0,0.5,'Frequency')



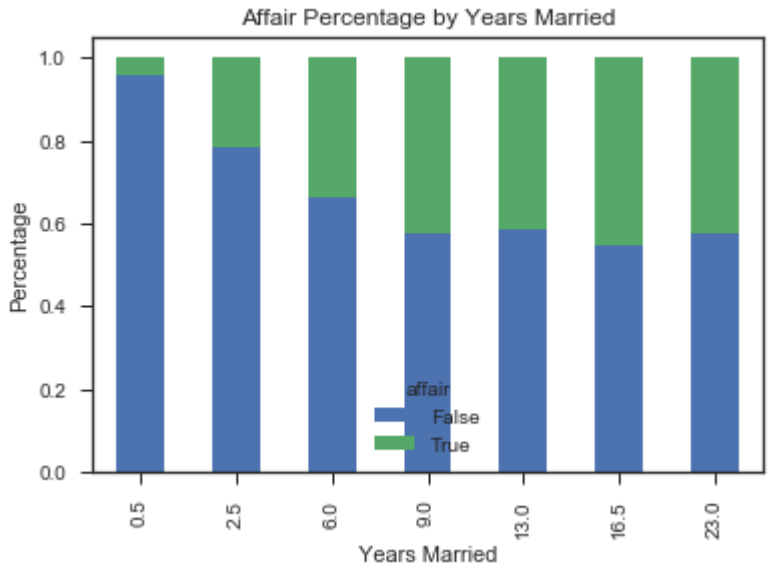
In [10]:
```
# barplot of marriage rating grouped by affair (True or False)
pd.crosstab(dta.rate_marriage, dta.affair.astype(bool)).plot(kind='bar')
plt.title('Marriage Rating Distribution by Affair Status')
plt.xlabel('Marriage Rating')
plt.ylabel('Frequency')
```

Out[10]: Text(0,0.5,'Frequency')



In [11]:
```
#stacked barplot to look at the percentage of women having affairs by number o
f years of marriage
affair_yrs_married = pd.crosstab(dta.yrs_married, dta.affair.astype(bool))
affair_yrs_married.div(affair_yrs_married.sum(1).astype(float), axis=0).plot(k
ind='bar', stacked=True)
plt.title('Affair Percentage by Years Married')
plt.xlabel('Years Married')
plt.ylabel('Percentage')
```

Out[11]: Text(0,0.5,'Percentage')



In [12]:
```
# run logistic regression on entire dataset to determine base accuracy
# 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[12]: 0.7258875274897895

In [13]:
```
# what percentage had affairs?
y.mean()
```

Out[13]: 0.3224945020420987

In [14]:
```
# examine the coefficients
X.columns, np.transpose(model.coef_)
```

Out[14]:
```
(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_marriag
e',
        '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 ]]))
```

In [15]:
```
# 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, rando
m_state=0)
model2 = LogisticRegression()
model2.fit(X_train, y_train)
```

Out[15]:
```
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 [16]:
```
# predict class labels for the test set
predicted = model2.predict(X_test)
predicted
```

Out[16]:
```
array([1., 0., 0., ..., 0., 0., 0.])
```

In [17]:
```
# generate class probabilities
probs = model2.predict_proba(X_test)
probs
```

Out[17]:
```
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 [18]:
```
# generate evaluation metrics
print(metrics.accuracy_score(y_test, predicted))
print(metrics.roc_auc_score(y_test, probs[:, 1]))
```

```
0.7298429319371728
0.745950606950631
```

In [19]:
```
#see the confusion matrix and a classification report with other metrics.
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 [20]:
```
# evaluate the model using 10-fold cross-validation
scores = cross_val_score(LogisticRegression(), X, y, scoring='accuracy', cv=10
)
scores, scores.mean()
```

Out[20]:
```
(array([0.72100313, 0.70219436, 0.73824451, 0.70597484, 0.70597484,
        0.72955975, 0.7327044 , 0.70440252, 0.75157233, 0.75      ]),
 0.7241630685514876)
```