In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
loans = pd.read_csv('loan_data.csv')
```

In [3]:
```
loans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
credit.policy        9578 non-null int64
purpose              9578 non-null object
int.rate             9578 non-null float64
installment          9578 non-null float64
log.annual.inc       9578 non-null float64
dti                  9578 non-null float64
fico                 9578 non-null int64
days.with.cr.line    9578 non-null float64
revol.bal            9578 non-null int64
revol.util           9578 non-null float64
inq.last.6mths       9578 non-null int64
delinq.2yrs          9578 non-null int64
pub.rec              9578 non-null int64
not.fully.paid       9578 non-null int64
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

In [4]:
```
loans.describe()
```

Out[4]:

| | credit.policy | int.rate | installment | log.annual.inc | dti | fico | days.witl |
|---|---|---|---|---|---|---|---|

|  | credit.policy | int.rate | installment | log.annual.inc | dti | fico | days.witl |
|---|---|---|---|---|---|---|---|
| **count** | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | 9578.000000 | 9578 |
| **mean** | 0.804970 | 0.122640 | 319.089413 | 10.932117 | 12.606679 | 710.846314 | 4560 |
| **std** | 0.396245 | 0.026847 | 207.071301 | 0.614813 | 6.883970 | 37.970537 | 2496 |
| **min** | 0.000000 | 0.060000 | 15.670000 | 7.547502 | 0.000000 | 612.000000 | 178 |
| **25%** | 1.000000 | 0.103900 | 163.770000 | 10.558414 | 7.212500 | 682.000000 | 2820 |
| **50%** | 1.000000 | 0.122100 | 268.950000 | 10.928884 | 12.665000 | 707.000000 | 4139 |
| **75%** | 1.000000 | 0.140700 | 432.762500 | 11.291293 | 17.950000 | 737.000000 | 5730 |
| **max** | 1.000000 | 0.216400 | 940.140000 | 14.528354 | 29.960000 | 827.000000 | 17639 |

In [5]:

```
loans.head()
```

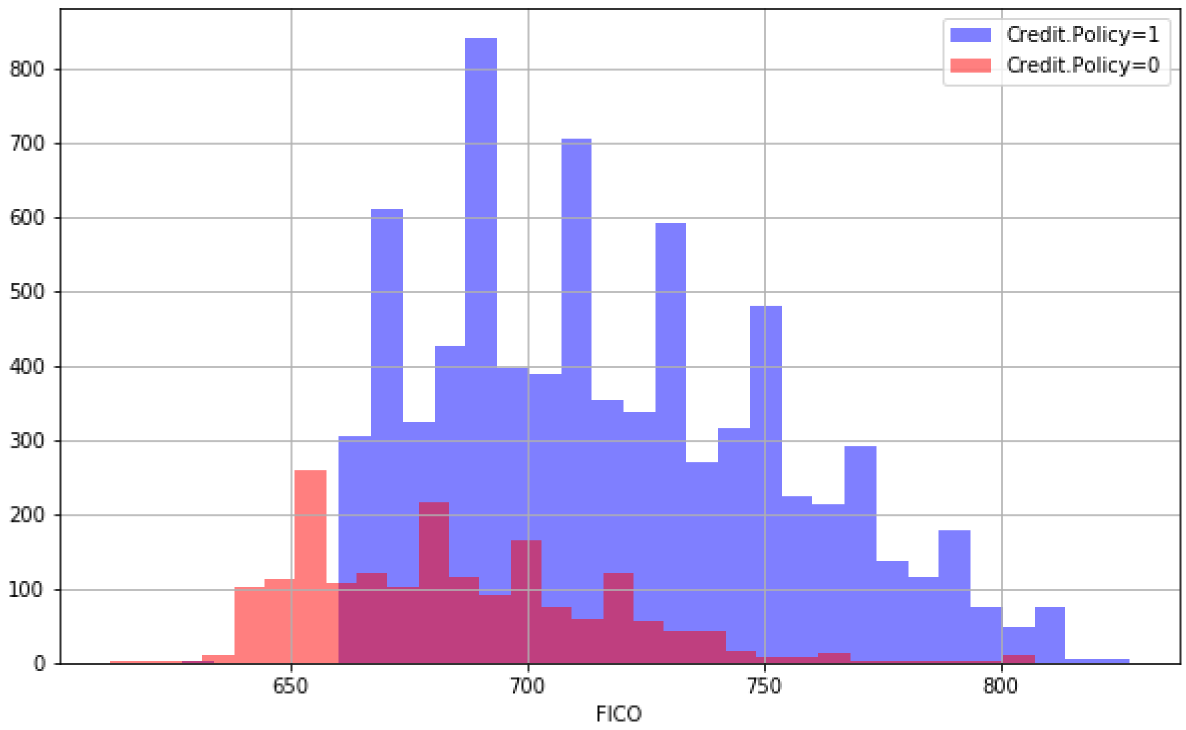
Out[5]:

|  | credit.policy | purpose | int.rate | installment | log.annual.inc | dti | fico | days.with.cr.line |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | debt_consolidation | 0.1189 | 829.10 | 11.350407 | 19.48 | 737 | 5639.95833 |
| **1** | 1 | credit_card | 0.1071 | 228.22 | 11.082143 | 14.29 | 707 | 2760.00000 |
| **2** | 1 | debt_consolidation | 0.1357 | 366.86 | 10.373491 | 11.63 | 682 | 4710.00000 |
| **3** | 1 | debt_consolidation | 0.1008 | 162.34 | 11.350407 | 8.10 | 712 | 2699.95833 |
| **4** | 1 | credit_card | 0.1426 | 102.92 | 11.299732 | 14.97 | 667 | 4066.00000 |

In [6]:

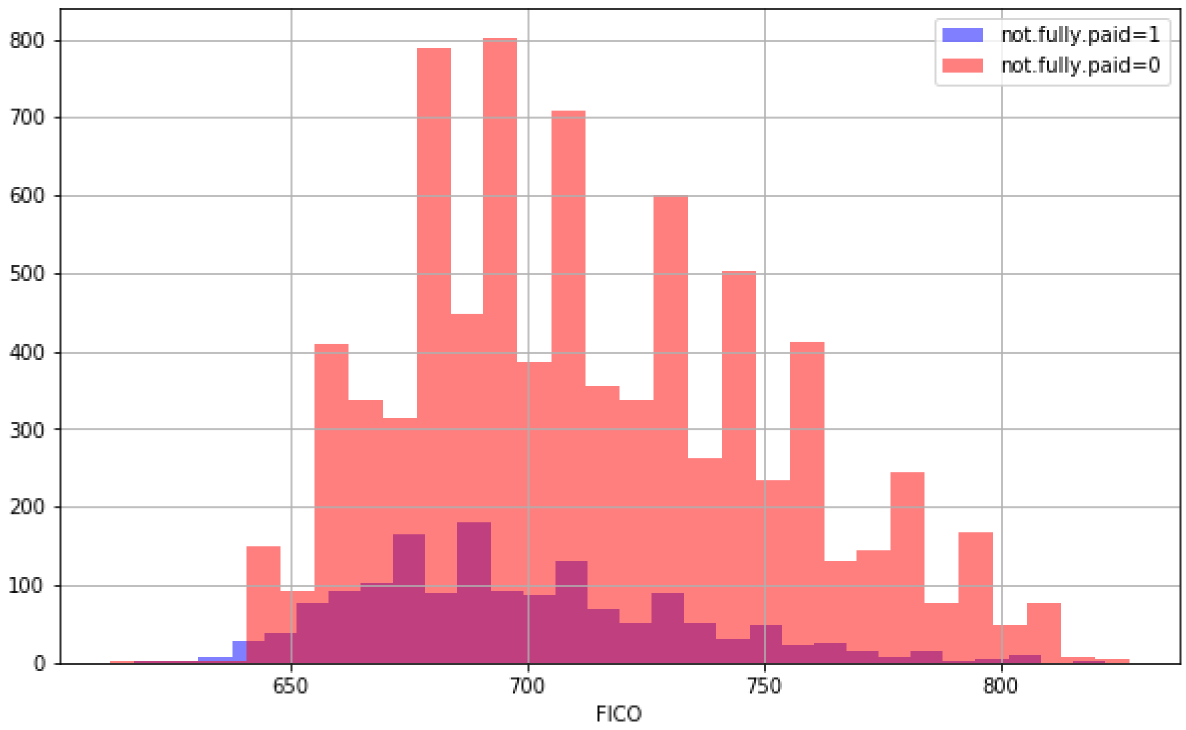
```
plt.figure(figsize=(10,6))
loans[loans['credit.policy']==1]['fico'].hist(alpha=0.5,color='blue',
                                              bins=30,label='Credit.Policy=1')
loans[loans['credit.policy']==0]['fico'].hist(alpha=0.5,color='red',
                                              bins=30,label='Credit.Policy=0')
plt.legend()
plt.xlabel('FICO')
```

```
Out[6]: Text(0.5,0,'FICO')
```



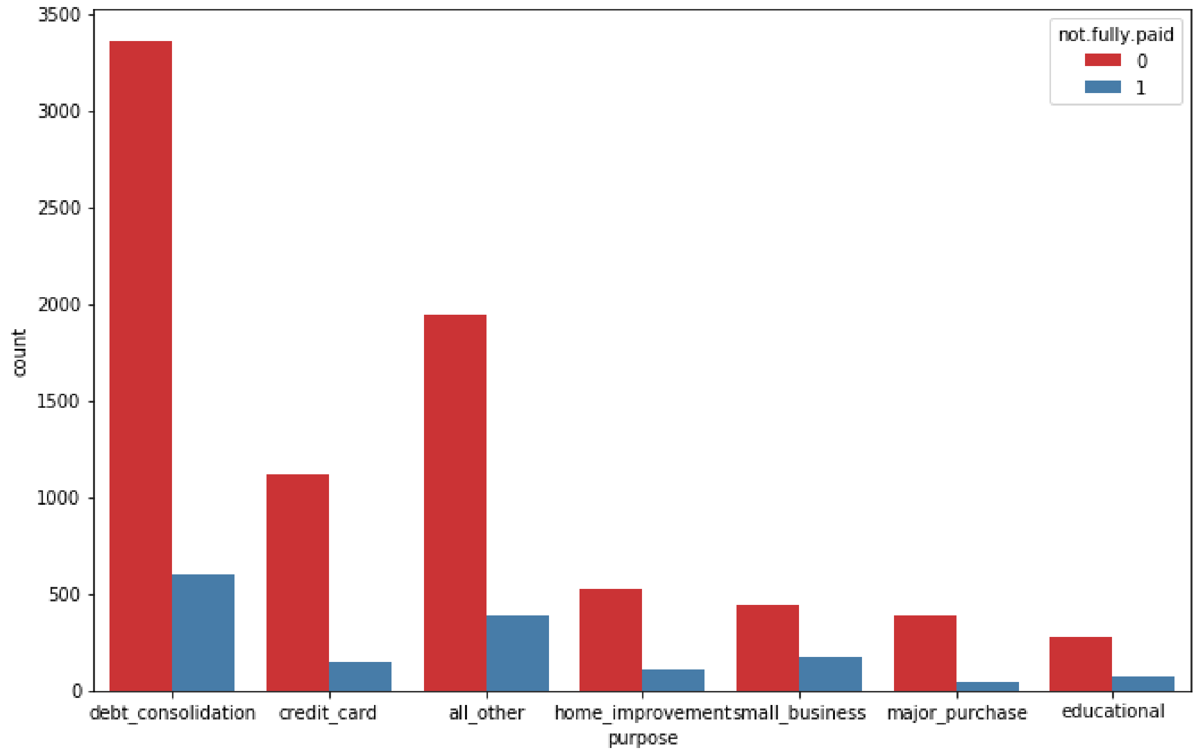
```
In [7]: plt.figure(figsize=(10,6))
        loans[loans['not.fully.paid']==1]['fico'].hist(alpha=0.5,color='blue',
                                                     bins=30,label='not.fully.
        paid=1')
        loans[loans['not.fully.paid']==0]['fico'].hist(alpha=0.5,color='red',
                                                     bins=30,label='not.fully.
        paid=0')
        plt.legend()
        plt.xlabel('FICO')
```

```
Out[7]: Text(0.5,0,'FICO')
```



```
In [8]: plt.figure(figsize=(11,7))
        sns.countplot(x='purpose',hue='not.fully.paid',data=loans,palette='Set1')
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x106921b38>
```



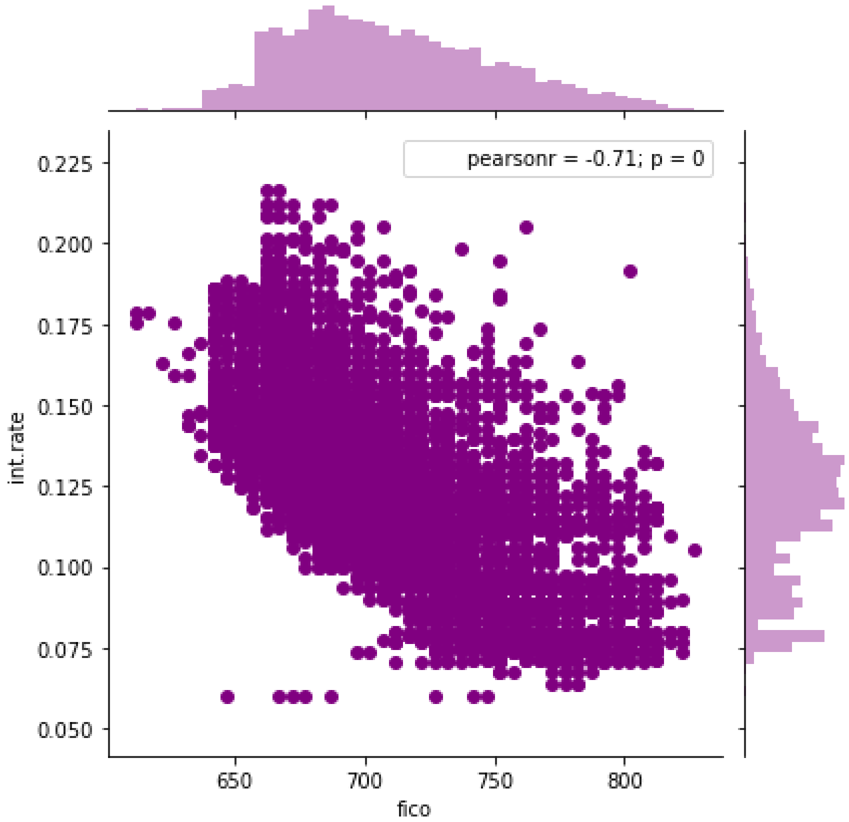
In [9]:

```
sns.jointplot(x='fico',y='int.rate',data=loans,color='purple')
```

```
/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462: U
serWarning: The 'normed' kwarg is deprecated, and has been replaced by
the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462: U
serWarning: The 'normed' kwarg is deprecated, and has been replaced by
the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[9]:

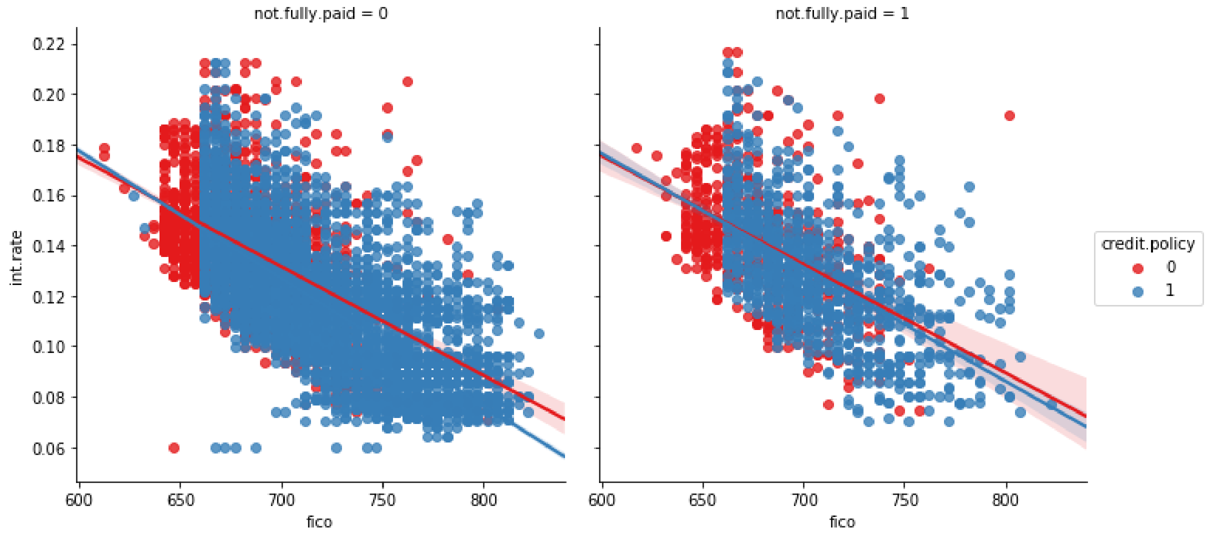
```
<seaborn.axisgrid.JointGrid at 0x1a0cc92550>
```



In [10]:
```
plt.figure(figsize=(11,7))
sns.lmplot(y='int.rate',x='fico',data=loans,hue='credit.policy',
           col='not.fully.paid',palette='Set1')
```

Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x1a18d922e8>
```

```
<Figure size 792x504 with 0 Axes>
```



In [11]:

```
loans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
credit.policy        9578 non-null int64
purpose              9578 non-null object
int.rate             9578 non-null float64
installment          9578 non-null float64
log.annual.inc       9578 non-null float64
dti                  9578 non-null float64
fico                 9578 non-null int64
days.with.cr.line    9578 non-null float64
revol.bal            9578 non-null int64
revol.util           9578 non-null float64
inq.last.6mths       9578 non-null int64
delinq.2yrs          9578 non-null int64
pub.rec              9578 non-null int64
not.fully.paid       9578 non-null int64
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

In [12]:
```
cat_feats = ['purpose']
```

In [13]:
```
final_data = pd.get_dummies(loans,columns=cat_feats,drop_first=True)
```

In [14]:
```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 19 columns):
credit.policy                 9578 non-null int64
int.rate                      9578 non-null float64
installment                   9578 non-null float64
log.annual.inc                9578 non-null float64
dti                           9578 non-null float64
fico                          9578 non-null int64
days.with.cr.line             9578 non-null float64
revol.bal                     9578 non-null int64
revol.util                    9578 non-null float64
inq.last.6mths                9578 non-null int64
delinq.2yrs                   9578 non-null int64
pub.rec                       9578 non-null int64
not.fully.paid                9578 non-null int64
purpose_credit_card           9578 non-null uint8
purpose_debt_consolidation    9578 non-null uint8
purpose_educational           9578 non-null uint8
purpose_home_improvement      9578 non-null uint8
purpose_major_purchase        9578 non-null uint8
purpose_small_business        9578 non-null uint8
dtypes: float64(6), int64(7), uint8(6)
memory usage: 1.0 MB
```

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
X = final_data.drop('not.fully.paid',axis=1)
y = final_data['not.fully.paid']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
0, random_state=101)
```

```
In [17]: from sklearn.tree import DecisionTreeClassifier
```

```
In [18]: dtree = DecisionTreeClassifier()
```

```
In [19]: dtree.fit(X_train,y_train)
```

```
Out[19]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=N
         one,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=N
         one,
                     splitter='best')
```

```
In [20]: predictions = dtree.predict(X_test)
```

```
In [21]: from sklearn.metrics import classification_report,confusion_matrix
```

```
In [22]: print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

          0       0.86      0.82      0.84      2431
          1       0.20      0.24      0.22       443

avg / total       0.75      0.73      0.74      2874
```

```
In [23]: print(confusion_matrix(y_test,predictions))
```

```
[[1996  435]
 [ 336  107]]
```

```
In [24]: from sklearn.ensemble import RandomForestClassifier
In [25]: rfc = RandomForestClassifier(n_estimators=600)
```

```
In [26]: rfc.fit(X_train,y_train)
```

```
Out[26]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gi
         ni',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=600, n_jobs=1,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False)
```

```
In [27]: predictions = rfc.predict(X_test)
```

```
In [28]: from sklearn.metrics import classification_report,confusion_matrix
```

```
In [29]: print(classification_report(y_test,predictions))
```

```
                      precision    recall  f1-score   support

                   0       0.85      1.00      0.92      2431
                   1       0.59      0.02      0.04       443

         avg / total       0.81      0.85      0.78      2874
```

```
In [30]: print(confusion_matrix(y_test,predictions))
```

```
         [[2424    7]
          [ 433   10]]
```

```
In [ ]:
```