In [11]:
```
#Lending club Project 2007 - 2010 data
#Evaluation of borrowers (prediction of not fully paid)
#Create a Decision Tree and a Random Forest Model
#Import Libraries
#Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [12]:
```
loans = pd.read_csv('loan_data.csv')
```

In [9]:
```
loans.head()
```

Out[9]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	r
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	

In [13]:
```
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 [6]:
```
sns.pairplot(loans, hue='credit.policy')
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x1a21bd06a0>

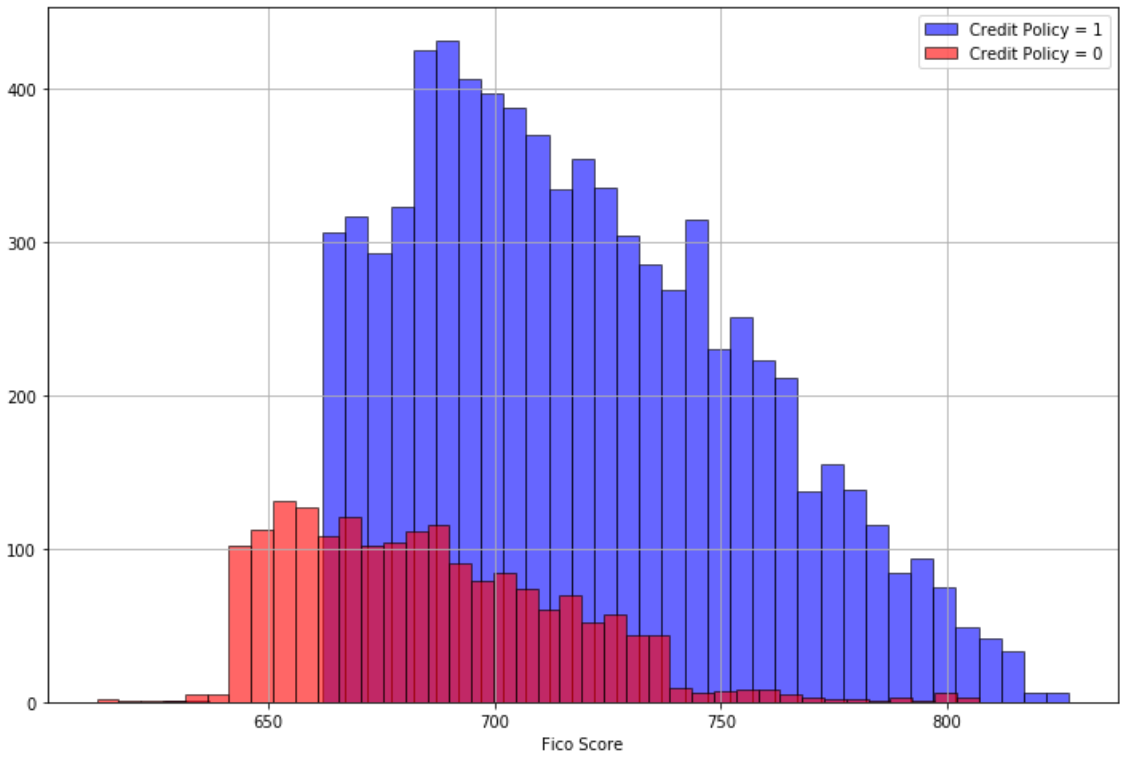
In [14]:
```
loans.describe()
```

Out[14]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	
count	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9.578
mean	0.804970	0.122640	319.089413	10.932117	12.606679	710.846314	4560.767197	1.691
std	0.396245	0.026847	207.071301	0.614813	6.883970	37.970537	2496.930377	3.375
min	0.000000	0.060000	15.670000	7.547502	0.000000	612.000000	178.958333	0.000
25%	1.000000	0.103900	163.770000	10.558414	7.212500	682.000000	2820.000000	3.187
50%	1.000000	0.122100	268.950000	10.928884	12.665000	707.000000	4139.958333	8.596
75%	1.000000	0.140700	432.762500	11.291293	17.950000	737.000000	5730.000000	1.824
max	1.000000	0.216400	940.140000	14.528354	29.960000	827.000000	17639.958330	1.207

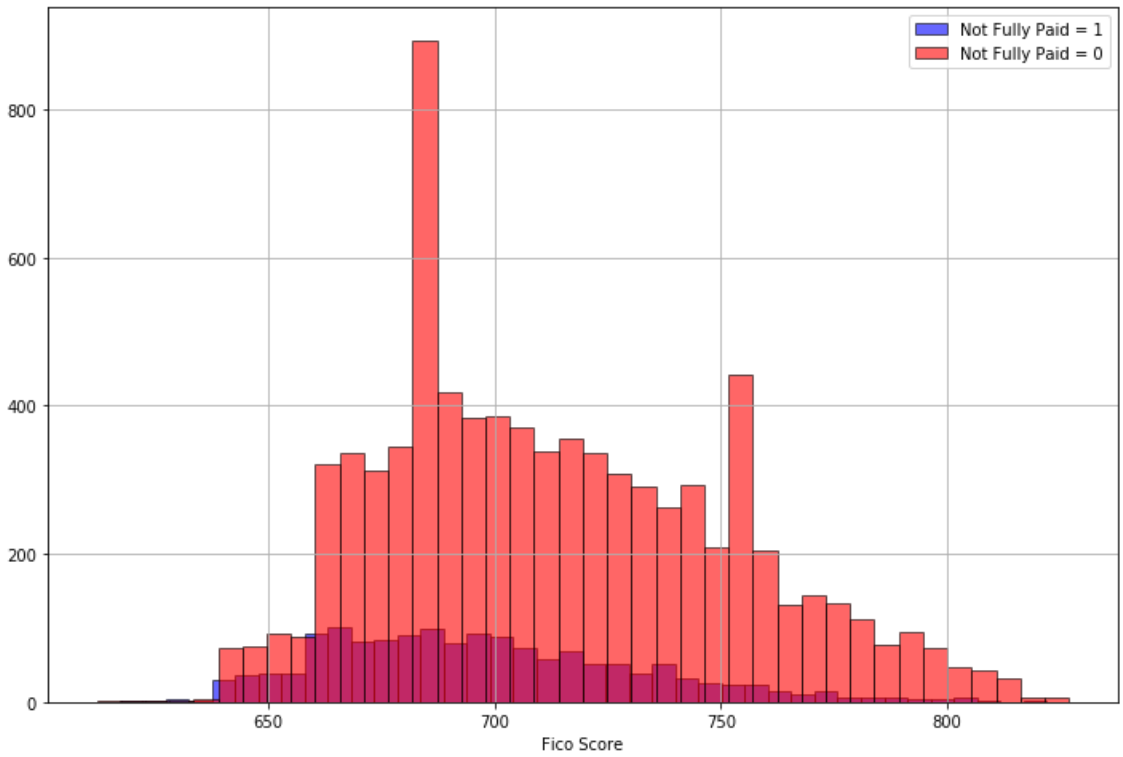
In [24]:
```
plt.figure(figsize=(12,8))
loans[loans['credit.policy']==1]['fico'].hist(bins=40,
                                              color='blue', label='Credit Policy
= 1',
                                             alpha=0.6, edgecolor='black')
loans[loans['credit.policy']==0]['fico'].hist(bins=40,
                                              color='red', label='Credit Policy
= 0',
                                             alpha=0.6, edgecolor='black')
plt.legend()
plt.xlabel('Fico Score')
```

Out[24]:
```
Text(0.5, 0, 'Fico Score')
```



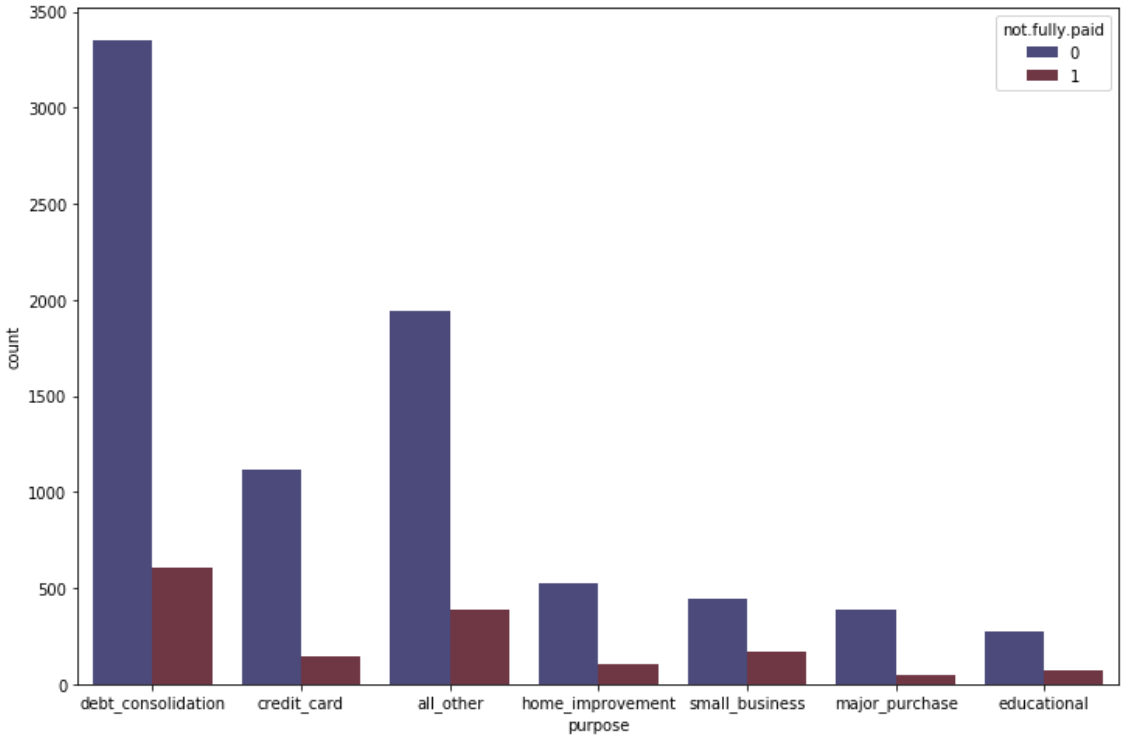
In [26]:
```
plt.figure(figsize=(12,8))
loans[loans['not.fully.paid']==1]['fico'].hist(bins=40,
                                               color='blue', label='Not Fully Pai
d = 1',
                                              alpha=0.6, edgecolor='black')
loans[loans['not.fully.paid']==0]['fico'].hist(bins=40,
                                               color='red', label='Not Fully Paid
= 0',
                                              alpha=0.6, edgecolor='black')
plt.legend()
plt.xlabel('Fico Score')
```

Out[26]:
```
Text(0.5, 0, 'Fico Score')
```



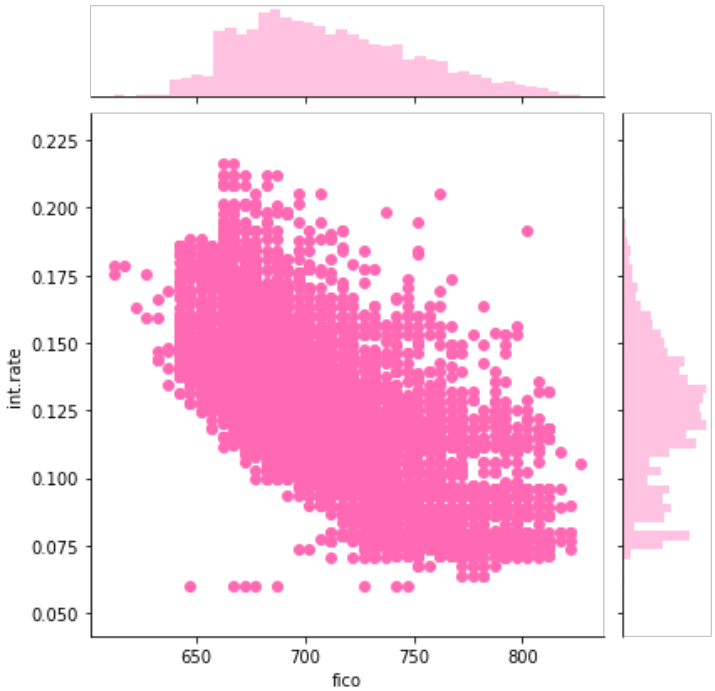
In [37]:
```
plt.figure(figsize=(12,8))
sns.countplot(x='purpose', hue='not.fully.paid',data=loans,palette='icefire')
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2d7d42b0>



In [44]:
```
sns.jointplot(x='fico',y='int.rate',data=loans,color='hotpink')
```

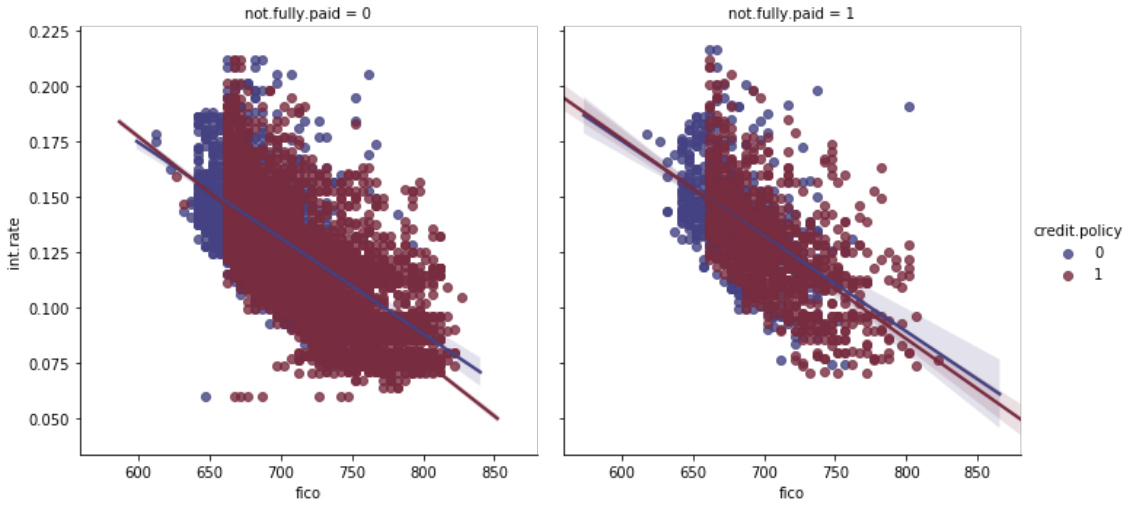
Out[44]: <seaborn.axisgrid.JointGrid at 0x1a2f8bed68>



In [47]:
```
plt.figure(figsize=(12,8))
sns.lmplot(x='fico',y='int.rate',data=loans,hue='credit.policy',
           col='not.fully.paid', palette='icefire')
```

Out[47]:
```
<seaborn.axisgrid.FacetGrid at 0x1a30496470>

<Figure size 864x576 with 0 Axes>
```



In [48]:
```
#Create a purpose column
cat_purpose = ['purpose']
```

In [49]:
```
final_data = pd.get_dummies(loans,columns=cat_purpose,drop_first=True)
```

In [50]:
```
#purpose colum created with 1 or 0 as values
final_data.head()
```

Out[50]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6m
0	1	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	
1	1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	
2	1	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	
3	1	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	
4	1	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	

In [52]:
```
#train and test
from sklearn.model_selection import train_test_split
```

In [55]:
```
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, random_s
tate=101)
```

In [56]:
```
#Create Decision Tree
from sklearn.tree import DecisionTreeClassifier
```

In [57]:
```
dtree = DecisionTreeClassifier()
```

In [58]:
```
dtree.fit(X_train,y_train)
```

Out[58]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       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=None, splitter='best')
```

In [59]:
```
pred1 = dtree.predict(X_test)
```

In [60]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [61]:
```
print(classification_report(y_test,pred1))
```

```
              precision    recall  f1-score   support

           0       0.85      0.83      0.84      2431
           1       0.18      0.21      0.20       443

    accuracy                           0.73      2874
   macro avg       0.52      0.52      0.52      2874
weighted avg       0.75      0.73      0.74      2874
```

In [62]:
```
print(confusion_matrix(y_test,pred1))
```

```
[[2011  420]
 [ 349   94]]
```

In [71]:
```
#Training the random Forrest
from sklearn.ensemble import RandomForestClassifier
```

In [72]:
```
 rfc = RandomForestClassifier(n_estimators=350)
```

In [73]:
```
rfc.fit(X_train,y_train)
```

Out[73]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       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=350,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [74]:
```
# Predictions
pred1 = rfc.predict(X_test)
```

In [75]:
```
print(classification_report(y_test,pred1))
```

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92      2431
           1       0.47      0.02      0.04       443

    accuracy                           0.85      2874
   macro avg       0.66      0.51      0.48      2874
weighted avg       0.79      0.85      0.78      2874
```

In [76]:
```
print(confusion_matrix(y_test,pred1))
```

```
[[2421   10]
 [ 434    9]]
```

In [ ]:
```
```