Assignment 6: Decision Tree

A. Import the usual libraries for pandas and plotting.

In [1]:

```
import pandas as pd
import seaborn as sb
import numpy as np
import sklearn
from matplotlib import pyplot as plt
```

B. Use pandas to read loan_data.csv as a dataframe called loans.

In [2]:

```
loans = pd.read_csv("loan_data.csv")
```

C. Check out the info(), head(), and describe() methods on loans.

In [3]:

```
loans.info()
```

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

In [4]:

```
loans.head()
```

Out[4]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.li
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.9583
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.0000
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.0000
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.9583
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.0000

In [5]:

```
loans.describe()
```

Out[5]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.w
count	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	957
mean	0.804970	0.122640	319.089413	10.932117	12.606679	710.846314	456
std	0.396245	0.026847	207.071301	0.614813	6.883970	37.970537	249
min	0.000000	0.060000	15.670000	7.547502	0.000000	612.000000	17
25%	1.000000	0.103900	163.770000	10.558414	7.212500	682.000000	282
50%	1.000000	0.122100	268.950000	10.928884	12.665000	707.000000	413
75%	1.000000	0.140700	432.762500	11.291293	17.950000	737.000000	573
max	1.000000	0.216400	940.140000	14.528354	29.960000	827.000000	1763

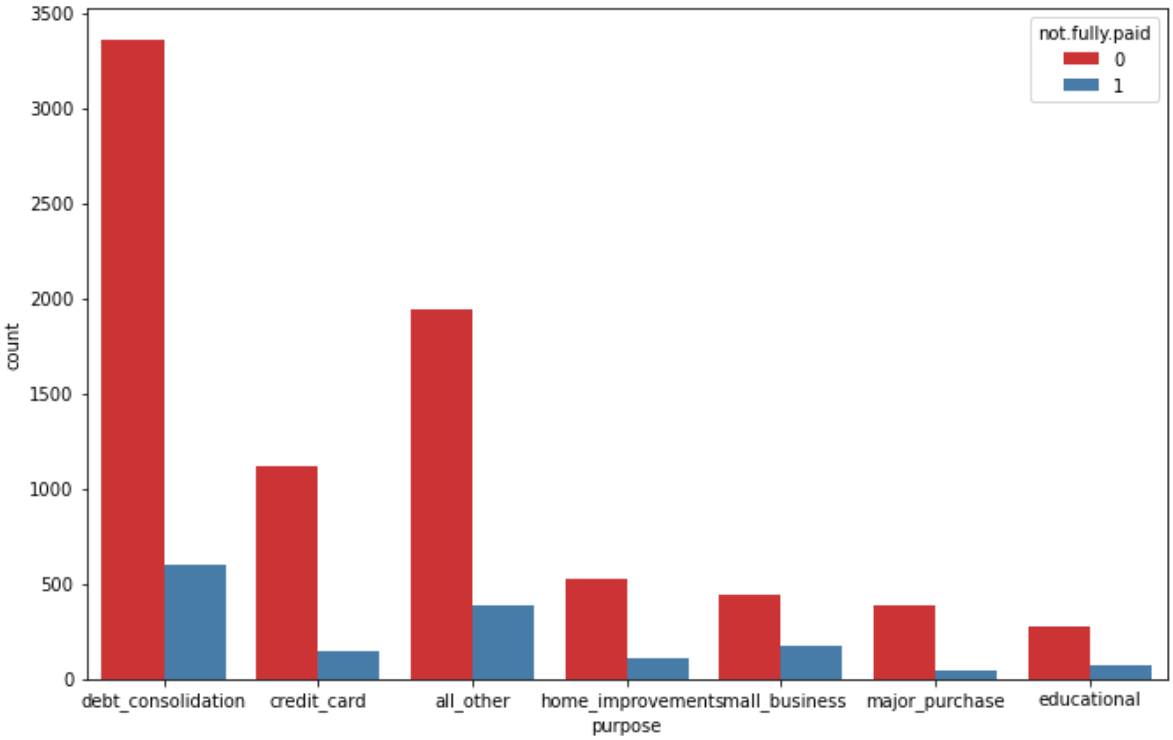
D. Create a countplot using seaborn showing the counts of loans by purpose, with the color hue defined by not.fully.paid.

In [6]:

```
plt.figure(figsize=(11,7))
sb.countplot(x='purpose',hue='not.fully.paid',data=loans,palette='Set1')
```

Out[6]:

```
<AxesSubplot:xlabel='purpose', ylabel='count'>
```



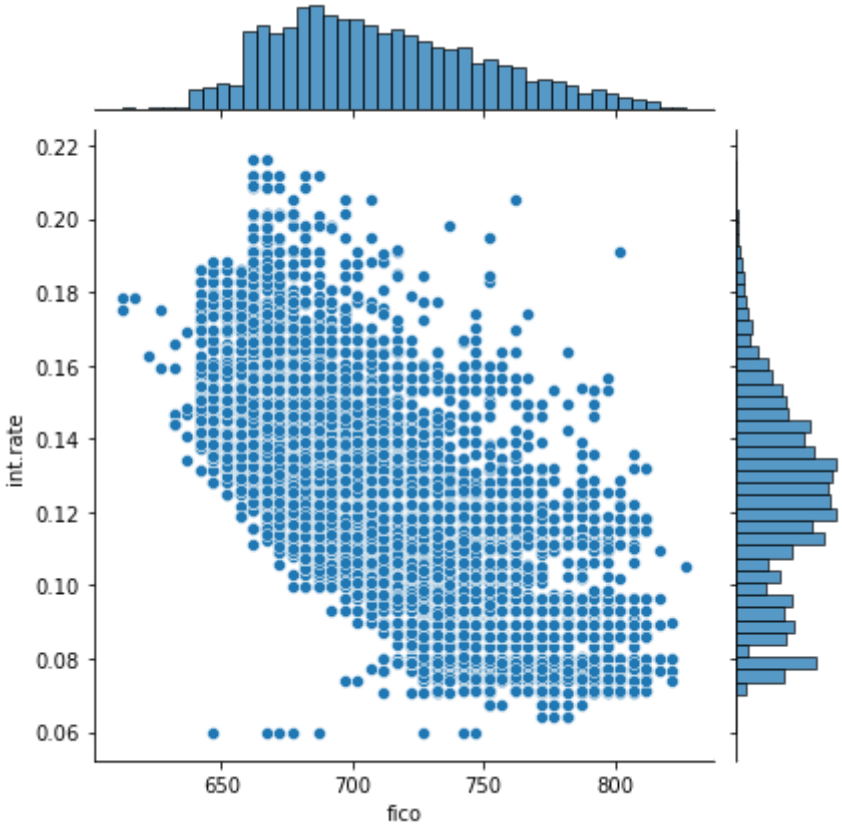
E. Let's see the trend between FICO score and interest rate.

In [7]:

```
sb.jointplot(x="fico",y="int.rate",data=loans)
```

Out[7]:

```
<seaborn.axisgrid.JointGrid at 0x26f7a567d00>
```



F. Create a list of 1 element containing the string 'purpose'. Call this list cat_feats.

In [8]:

```
cat_feats=["purpose"]
```

G. Now use pd.get_dummies(loans,columns=cat_feats,drop_first=True) to create a fixed larger dataframe that has new feature columns with dummy variables. Set this dataframe as final_data.

In [9]:

```
final_data=pd.get_dummies(loans,columns=cat_feats,drop_first=True)
final_data.head()
```

Out[9]:

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

H. Now display the information of final data.

In [10]:

```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 19 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   credit.policy               9578 non-null   int64  
 1   int.rate                    9578 non-null   float64
 2   installment                 9578 non-null   float64
 3   log.annual.inc              9578 non-null   float64
 4   dti                         9578 non-null   float64
 5   fico                        9578 non-null   int64  
 6   days.with.cr.line           9578 non-null   float64
 7   revol.bal                   9578 non-null   int64  
 8   revol.util                  9578 non-null   float64
 9   inq.last.6mths              9578 non-null   int64  
 10  delinq.2yrs                 9578 non-null   int64  
 11  pub.rec                     9578 non-null   int64  
 12  not.fully.paid              9578 non-null   int64  
 13  purpose_credit_card         9578 non-null   uint8  
 14  purpose_debt_consolidation  9578 non-null   uint8  
 15  purpose_educational         9578 non-null   uint8  
 16  purpose_home_improvement    9578 non-null   uint8  
 17  purpose_major_purchase      9578 non-null   uint8  
 18  purpose_small_business      9578 non-null   uint8  
dtypes: float64(6), int64(7), uint8(6)
memory usage: 1.0 MB
```

I. Use sklearn to split your data into a training set and a testing set as we've done in the past.

In [11]:

```
from sklearn.model_selection import train_test_split
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_state=100)
```

J. Import DecisionTreeClassifier.

In [12]:

```
from sklearn.tree import DecisionTreeClassifier
```

K. Create an instance of DecisionTreeClassifier() called dtree and fit it to the training data.

In [13]:

```
dtree=DecisionTreeClassifier()
```

In [14]:

```
dtree.fit(x_train,y_train)
```

Out[14]:

```
DecisionTreeClassifier()
```

L. Create predictions from the test set and create a classification report and a confusion matrix.

In [15]:

```
y_predict=dtree.predict(x_test)
from sklearn.metrics import confusion_matrix,classification_report
```

In [16]:

```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.85      0.84      0.84      2392
           1       0.24      0.24      0.24       482

    accuracy                           0.74      2874
   macro avg       0.54      0.54      0.54      2874
weighted avg       0.74      0.74      0.74      2874
```

In [17]:

```
print(confusion_matrix(y_test,y_predict))
```

```
[[2014  378]
 [ 365  117]]
```