DSC-530: Predictive Modeling
Week 3 Assignment: Predictive Modeling Using Decision Trees
Author: Jonathan Ibifubara Pollyn
College of Science, Engineering and Technology, Grand Canyon University

In [43]:
```
#Loading the required packages
import pandas as pd
import numpy as np
import statsmodels.tools.tools as stattools
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

1.Create a CART model using the training data set that predicts Approval using Debt to Income Ratio, FICO Score, and Request Amount. Visualize the decision tree. Describe the first few splits in the decision tree.

In [2]:
```
#Loading training Data into a variable
loan_training = pd.read_csv('C:/DataSets/Loans_Training')
```

In [4]:
```
loan_training
```

Out[4]:

	Approval	Debt-to-Income Ratio	FICO Score	Request Amount	Interest
0	F	0.00	397	1000	450.0
1	F	0.00	403	500	225.0
2	F	0.00	408	1000	450.0
3	F	0.00	408	2000	900.0
4	F	0.00	411	5000	2250.0
...	...	...	...	...	...
150297	T	0.38	709	19000	8550.0
150298	T	0.38	722	17000	7650.0
150299	T	0.38	747	11000	4950.0
150300	T	0.39	679	10000	4500.0
150301	T	0.39	769	7000	3150.0

150302 rows × 5 columns

In [28]:
```
#Set the predictor variable
predictor_x = loan_training[['Debt-to-Income Ratio','FICO Score','Request Amount','Interest']]
```

In [29]:
```
#Setting the target variable
target_y = loan_training['Approval']
```

In [43]:
```
#Running the CART Algorithm
cart_training = DecisionTreeClassifier(random_state=0,max_depth=2)
```

In [44]:
```
cart_training.fit(predictor_x, target_y)
```

Out[44]:
```
DecisionTreeClassifier(max_depth=2, random_state=0)
```

In [105]:
```
loan_training_data = export_graphviz(cart_training, out_file=None,
                      feature_names=loan_training.columns[1:],
                      class_names=loan_training["Approval"].unique(),
                      filled=True, rounded=True,
                      special_characters=True)
```

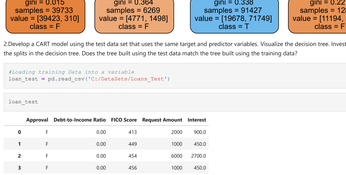
In [ ]:
```
#Installing the graphviz and pydot
conda install python-graphviz
conda install pydot
```

In [49]:
```
import graphviz
```

In [104]:
```
graph = graphviz.Source(loan_training_data)
```

In [105]:
```
graph
```

Out[105]:



2.Develop a CART model using the test data set that uses the same target and predictor variables. Visualize the decision tree. Investigate the splits in the decision tree. Does the tree built using the test data match the tree built using the training data?

In [59]:
```
#Loading training Data into a variable
loan_test = pd.read_csv('C:/DataSets/Loans_Test')
```

In [60]:
```
loan_test
```

Out[60]:

	Approval	Debt-to-Income Ratio	FICO Score	Request Amount	Interest
0	F	0.00	413	2000	900.0
1	F	0.00	449	1000	450.0
2	F	0.00	454	6000	2700.0
3	F	0.00	456	1000	450.0
4	F	0.00	457	1000	450.0
...	...	...	...	...	...
49693	T	0.38	662	14000	6300.0
49694	T	0.38	664	16000	7200.0
49695	T	0.38	676	4000	1800.0
49696	T	0.38	680	6000	2700.0
49697	T	0.39	662	500	225.0

49698 rows × 5 columns

In [61]:
```
#Set the predictor variable for the test data
predictor_test_x = loan_test[['Debt-to-Income Ratio','FICO Score','Request Amount','Interest']]
```

In [62]:
```
#Setting the target variable for the test data
target_test_y = loan_test['Approval']
```

In [66]:
```
#Running the CART Algorithm
cart_test = DecisionTreeClassifier(random_state=0,max_depth=2)
```

In [70]:
```
#Running the CART prediction
cart_test.fit(predictor_test_x, target_test_y)
```

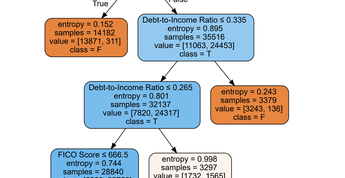
Out[70]:
```
DecisionTreeClassifier(max_depth=2, random_state=0)
```

In [82]:
```
#Obtaining the graph
loan_test_data = export_graphviz(cart_test, out_file=None,
                      feature_names=loan_training.columns[1:],
                      class_names=loan_training["Approval"].unique(),
                      filled=True, rounded=True,
                      special_characters=True)
```

In [83]:
```
graph_test = graphviz.Source(loan_test_data)
```

In [84]:
```
graph_test
```

Out[84]:



3.Build a C5.0 model using the training data set that predicts Approval using Debt to Income Ratio, FICO Score, and Request Amount. Specify a minimum of 1,000 cases per terminal node. Visualize the decision tree. Describe the first few splits in the decision tree.

In [73]:
```
#Obtain decision tree using entropy
c50_training = DecisionTreeClassifier(criterion='entropy',max_leaf_nodes=5).fit(predictor_x,target_y)
```

In [74]:
```
loan_training_c50 = export_graphviz(c50_training, out_file=None,
                      feature_names=loan_training.columns[1:],
                      class_names=loan_training["Approval"].unique(),
                      filled=True, rounded=True,
                      special_characters=True)
```

In [75]:
```
c50_graph = graphviz.Source(loan_training_c50)
```

In [76]:
```
c50_graph
```

Out[76]:

5.Create a C5.0 model using the test data set that utilizes the same target variable, predictor variables, and minimum cases criterion. Visualize the decision tree. Does the tree built using the test data match the tree built using the training data?

In [85]:
```
#Obtain decision tree using entropy
c50_test = DecisionTreeClassifier(criterion='entropy',max_leaf_nodes=5).fit(predictor_test_x,target_test_y)
```

In [87]:
```
loan_test_c50 = export_graphviz(c50_test, out_file=None,
                      feature_names=loan_training.columns[1:],
                      class_names=loan_training["Approval"].unique(),
                      filled=True, rounded=True,
                      special_characters=True)
```

In [88]:
```
c50_graph_test = graphviz.Source(loan_test_c50)
```

In [89]:
```
c50_graph_test
```

Out[89]:

1. Use random forests on the training data set to obtain the predicted value of Approval using the same predictor variables as in the CART and C5.0 models.

In [90]:
```
random_for_y = np.ravel(target_y)
```

In [91]:
```
#Create the Random Forest
random_for_train_y  = RandomForestClassifier(n_estimators=100, criterion='gini').fit(predictor_x,random_for_y)
```

In [95]:
```
random_for_train_y.predict(predictor_x)
```

Out[95]:
```
array(['F', 'F', 'F', ..., 'T', 'T', 'T'], dtype=object)
```

1. Use random forests on the test data set to obtain the predicted value of Approval in the test data set. Build a table comparing the predictions from the training and test data sets. How do they compare?

In [96]:
```
random_for_test_y = np.ravel(target_test_y)
```

In [99]:
```
#Create the Random Forest
random_for_test_y  = RandomForestClassifier(n_estimators=100, criterion='gini').fit(predictor_test_x,random_for_test_y)
```

In [100]:
```
random_for_test_y.predict(predictor_test_x)
```

Out[100]:
```
array(['F', 'F', 'F', ..., 'T', 'T', 'T'], dtype=object)
```

In [ ]:
```

```