DSC-530: Predictive Modeling

Week 7 Assignment: Presenting and Interpreting Results of Predictive Models

Author: Jonathan Ibifubara Pollyn

College of Science, Engineering and Technology, Grand Canyon University

In [86]:
```
#loading the required packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
import statsmodels.tools.tools as stattools
import sklearn
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [39]:
```
#Importing the required data sets
loans_training = pd.read_csv('C:/School/DSC-530/DataSets/Loans_Training')
loans_test = pd.read_csv('C:/School/DSC-530/DataSets/Loans_Test')
```

In [40]:
```
#Convert T and F to 1 and 0
loans_training["Approval"]=loans_training["Approval"].apply(lambda x:  1 if x in("T") else 0)

features = loans_training.columns
print(features)
```

```
Index(['Approval', 'Debt-to-Income Ratio', 'FICO Score', 'Request Amount',
       'Interest'],
      dtype='object')
```

In [43]:
```
# Extractng the Predictors
X = loans_training[['Debt-to-Income Ratio','FICO Score','Request Amount']]
print(X.columns)

# Extractng the vector
y = loans_training[['Approval']]
print(y.columns)
```

```
Index(['Debt-to-Income Ratio', 'FICO Score', 'Request Amount'], dtype='object')
Index(['Approval'], dtype='object')
```

Using the training data set, create a C5.0 model (Model 1) to predict a loan applicant's Approval using Debt-to-Income Ratio, FICO Score, and Request Amount. Obtain the predicted responses.

In [32]:
```
#Creating the C5.0 Model (Model1)
c50_model1 = DecisionTreeClassifier(criterion='entropy',max_leaf_nodes=5).fit(X,y)
```

In [33]:
```
#Obtaining the classification
c50_model1.predict(X)
```

Out[33]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

Evaluate Model 1 using the test data set. Construct a contingency table to compare the actual and predicted values of Approval.

In [45]:
```
#Convert T and F to 1 and 0 on the test data
loans_test["Approval"]=loans_test["Approval"].apply(lambda x:  1 if x in("T") else 0)

features = loans_test.columns
print(features)
```

```
Index(['Approval', 'Debt-to-Income Ratio', 'FICO Score', 'Request Amount',
       'Interest'],
      dtype='object')
```

In [47]:
```
# Extractng the Predictors for the test data
X_test = loans_test[['Debt-to-Income Ratio','FICO Score','Request Amount']]
print(X.columns)

# Extractng the vector
y_test = loans_training[['Approval']]
print(y.columns)
```

```
Index(['Debt-to-Income Ratio', 'FICO Score', 'Request Amount'], dtype='object')
Index(['Approval'], dtype='object')
```

In [54]:
```
#Evaluating model 1
y_predicted = c50_model1.predict(X_test)
```

In [72]:
```
#Creating a contingency table to compare the actual and predicted values of Approval
ypred = pd.crosstab(loans_test['Approval'], y_predicted, rownames= ['Actual'], colnames= ['Predicted'])
```

In [73]:
```
ypred
```

Out[73]:

Predicted	0	1
Actual		
0	17665	7269
1	751	24013

In [69]:
```
ypred['Total'] = ypred.sum(axis=1); ypred.loc['Total'] = ypred.sum(); ypred
```

Out[69]:

Predicted	0	1	Total
Actual			
0	17665	7269	24934
1	751	24013	24764
Total	18416	31282	49698

In [78]:
```
#Calculation for the below following can be found on the attached excel spreadsheet.
# 1 . Accuracy
# 2 . Error rate
# 3 . Sensitivity
# 4 . Specificity
# 5 . Precision
# 6 . F1
# 7 . F2
# 8 . F0.5
```

Calculate the mean Request Amount per loan applicant from the training data set. Set this value to be the cost of a false positive.

In [83]:
```
loans_training['Request Amount'].mean()
```

Out[83]:
```
13427.080145307447
```

1. Compute the mean of the Interest per loan applicant from the training data set. Set the negative of that value to be the cost of a true positive.

In [82]:
```
loans_training['Interest'].mean()
```

Out[82]:
```
6042.186065388351
```

Obtain the simplified data-driven cost matrix.

In [84]:
```
y_pred = classifier.predict(xtest)

cm = confusion_matrix(ytest, y_pred)

print ("Confusion Matrix : \n", cm)
```

```
Confusion Matrix :
 [[14436  4199]
 [ 3159 15782]]
```

In [ ]:
```

```