## STAT 656 Homework 4

Name-Mayank Jaggi UIN-526005299

```
PART 2
1) PYTHON PROGRAM
# -*- coding: utf-8 -*-
Created on Wed Feb 13 11:16:05 2019
@author: mayank
from AdvancedAnalytics import DecisionTree
from AdvancedAnalytics import ReplaceImputeEncode
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
import pandas as pd
df2 = pd.read excel("CreditHistory Clean.xlsx")
                                                      #data file name
df2.rename(columns={'telephon':'telephone'},inplace = True)
attribute map = {
        age':['I', (19, 120)],
        'amount':['I', (0, 20000)],
        'checking':['N',(1,2,3,4)],
        'coapp':['N',(1, 2, 3)],
        'depends':['B',(1, 2)],
        'duration':['I',(1,72)],
        'employed':['N',(1,2,3,4,5)],
        'existcr':['N',(1,2,3,4)],
        'foreign':['B', (1,2)],
        'good_bad':['B', ('bad','good')],
        'history':['N', (0,1,2,3,4)],
        'housing':['N', (1,2,3)],
        'installp':['N', (1,2,3,4)],
        'job':['N', (1,2,3,4)],
        'marital':['N', (1,2,3,4)],
        'other':['N', (1,2,3)],
        'property':['N', (1,2,3,4)],
'purpose':['N',('0','1','2','3','4','5','6','8','9','X')],
        'resident':['N', (1,2,3,4)],
        'savings':['N', (1,2,3,4,5)],
        'telephone':['B', (1,2)] }
```

```
# Data Preprocessing, Replace outlier, impute missing values and encode
rie = ReplaceImputeEncode(data map=attribute map,
nominal_encoding='one-hot',interval_scale=None, drop=False, display=True)
# Now request replace-impute-encode for your dataframe
encoded df = rie.fit transform(df2)
print("\nData after replacing outliers, impute missing and encoding:")
print(encoded df.head())
# Defining target and input variables
y = encoded_df['good_bad']
                             #target
x = encoded_df.drop('good_bad',axis=1) #input
#10 fold Cross validation
list1 = ['accuracy', 'recall', 'precision', 'f1']
search_depths = [5,6,7,8,10,12,15,20,25]
for d in search depths:
    dtc = DecisionTreeClassifier(criterion='gini', max_depth=d, min_samples_split=5,
min_samples_leaf=5)
    mean_score = []
    std score = []
    print("max_depth=", d)
    print("{:.<13s}{:>6s}{:>13s}".format("Metric", "Mean", "Std. Dev."))
    for 1 in list1:
        dtc10 = cross_val_score(dtc, x, y, scoring=1, cv=10)
       mean = dtc10.mean()
        std = dtc10.std()
       mean_score.append(mean)
        std score.append(std)
        print("{:.<13s}{:>7.4f}{:>10.4f}".format(1, mean, std))
# Results suggest that the depth=5 is optimium decision tree
#Optimum Decision Tree
dtc = DecisionTreeClassifier(criterion='gini', max depth=5,min samples split=5,
min samples leaf=5)
x_train, x_validate, y_train, y_validate = train_test_split(x, y, test_size=0.3,
random_state=1) # Data Partition
dtc = dtc.fit(x_train,y_train)
classes = [ 'good', 'bad']
col = rie.col
col.remove('good bad')
DecisionTree.display_importance(dtc, col)
DecisionTree.display_binary_split_metrics(dtc, x_train, y_train, x_validate,
y validate)
```

# # Tree Image from IPython.display import Image from sklearn.externals.six import StringIO from pydotplus import graph\_from\_dot\_data dotdata = StringIO() feature\_Names=encoded\_df[0:68]

feature\_Names=encoded\_df[0:68]
export\_graphviz(dtc,out\_file=dotdata, class\_names= ['1:Good','0:Bad'], filled=True,
rounded=True, special\_characters=True)
tree = graph\_from\_dot\_data(dotdata.getvalue())
Image(tree.create\_png())

#### **TABLE OF METRICS**

#### max\_depth= 5

Metric	Mean	Std. Dev.
accuracy	0.719	0.0291
recall	0.8686	0.0355
precision	0.7653	0.0398
f1	0.8124	0.0131

#### max\_depth= 6

Metric	Mean	Std. Dev.
accuracy	0.711	0.0212
recall	0.8443	0.0497
precision	0.7711	0.0436
f1	0.8019	0.0073

### max\_depth= 7

Metric	Mean	Std. Dev.
accuracy	0.706	0.035
recall	0.8329	0.0491
precision	0.7675	0.0397
f1	0.7978	0.021

#### max\_depth= 15

Metric	Mean	Std. Dev.
accuracy	0.704	0.0284
recall	0.78	0.0524
precision	0.7944	0.0312
f1	0.7858	0.0278

#### max\_depth= 20

Metric	Mean	Std. Dev.
accuracy	0.698	0.0279
recall	0.78	0.0483
precision	0.793	0.0332
f1	0.787	0.0257

#### max\_depth= 25

Metric	Mean	Std. Dev.
accuracy	0.701	0.0274
recall	0.7743	0.0556
precision	0.797	0.0317
f1	0.7849	0.0201

#### max\_depth= 8

<u>_</u>	· •	
Metric	Mean	Std. Dev.
accuracy	0.7	0.0272
recall	0.8071	0.0462
precision	0.784	0.0366
f1	0.7906	0.0169

### max\_depth= 10

Metric	Mean	Std. Dev.
accuracy	0.705	0.0415
recall	0.7957	0.0515
precision	0.7881	0.0448
f1	0.7878	0.0248

#### max\_depth= 12

Metric	Mean	Std. Dev.
accuracy	0.697	0.0422
recall	0.7743	0.0538
precision	0.7888	0.0337
f1	0.7818	0.0251

Based on the aforementioned metrics, f1 is maximum for max\_depth=5. The ideal value for f1 metric is 1. So, our optimum model would be max\_depth=5

4)

#### **TABLE OF METRICS**

Model Metrics	Training	Validation
Observations	700	300
Features	68	68
Maximum Tree Depth	5	5
Minimum Leaf Size	5	5
Minimum split Size	5	5
Mean Absolute Error	0.2822	0.3405
Avg Squared Error	0.1411	0.201
Accuracy	0.7743	0.7
Precision	0.8026	0.7541
Recall (Sensitivity)	0.8951	0.8598
F1-score	0.8463	0.8035
MISC (Misclassification)	22.60%	30.00%
class 0	50.00%	69.80%
class 1	10.50%	14.00%

#### Training

Confusion Matrix	Class 0	Class 1
Class 0	107	107
Class 1	51	435

#### **Validation**

Confusion Matrix	Class 1	Class 1
Class 0	26	60
Class 1	30	184

#### 5) DECISION TREE X<sub>9</sub> ≤ 0.5 gini = 0.425 samples = 700 value = [214, 486] class = 0:Bad False. X<sub>44</sub> ≤ 0.5 gini = 0.221 X<sub>2</sub> ≤ 22.5 gini = 0.488 samples = 424 samples = 276 value = [35, 241 value = [179, 245] class = 0:Bad class = 0:Bad X<sub>26</sub> ≤ 0.5 gini = 0.44 X<sub>63</sub> ≤ 0.5 gini = 0.496 X<sub>57</sub> ≤ 0.5 gini = 0.418 X<sub>26</sub> ≤ 0.5 gini = 0.167 samples = 229 value = [21, 208] class = 0:Bad samples = 239 value = [78, 161] samples = 185 samples = 47 value = [101, 84] class = 1:Good value = [14, 33] class = 0:Bad class = 0:Bad X<sub>10</sub> ≤ 0.5 X<sub>10</sub> ≤ 0.5 gini = 0.222 X<sub>24</sub> ≤ 0.5 gini = 0.476 gini = 0.444 gini = 0.332 gini = 0.467 gini = 0.251 gini = 0.021 samples = 93 value = [1, 92 class = 0:Bad gini = 0.474 samples = 9 samples = 184 value = [71, 113] class = 0:Bad samples = 55 value = [7, 48] class = 0:Bad samples = 64 value = [25, 39] class = 0:Bad samples = 136 value = [20, 116] class = 0:Bad samples = 121 value = [76, 45] samples = 38 value = [8, 30] value = [6, 3] class = 1:Good class = 1:Good X<sub>18</sub> ≤ 0.5 gini = 0.367 $X_1 \le 5865.5$ X<sub>1</sub> ≤ 2249.0 X<sub>1</sub> ≤ 1373.0 X<sub>1</sub> ≤ 1504.0 $X_1 \le 4367.5$ gini = 0.48 gini = 0.32 gini = 0.49 gini = 0.491 gini = 0.495 gini = 0.091 gini = 0.48 gini = 0.226 gini = 0.493 gini = 0.159 samples = 5 value = [1, 4] class = 0:Bad samples = 7 samples = 5 amples = 8 samples = 33 value = [8, 25] class = 0:Bad samples = 23 value = [2, 21] class = 0:Bad samples = 131 value = [17, 114] class = 0:Bad samples = 22 value = [1, 21 samples = 162 samples = 31 samples = 100 samples = 21 value = [20, 1 samples = 15 value = [4, 3] value = [3, 2] value = [17, 14] class = 1:Good value = [6, 9] class = 0:Bad value = [70, 92] value = [56, 44] class = 1:Good class = 1:Good class = 0:Ba class = 0:Ba class = 0:Bad class = 0:Bad class = 1:Good class = 1:Good gini = 0.444 gini = 0.0 gini = 0.447 gini = 0.0 gini = 0.0 gini = 0.493 gini = 0.444 gini = 0.0 gini = 0.484 gini = 0.117 gini = 0.483 gini = 0.198 gini = 0.346 gini = 0.499 gini = 0.32 gini = 0.375 gini = 0.0 gini = 0.48 gini = 0.154

samples = 73

value = [35, 38]

class = 0:Bad

samples = 5

value = [4, 1]

class = 1:Good

samples = 16

value = [16, 0

class = 1:Goo

samples = 5 value = [0, 5

:lass = 0:Ba

mples = 1

samples = 8

value = [2, 6] class = 0:Bad samples = 10

value = [6, 4]

class = 1:Good

samples = 10

value = [9, 98]

class = 0:Bad

samples = 24

value = [8, 16]

class = 0:Bad

samples = 70

value = [39, 31]

class = 1:Good

samples = 5

value = [1, 4

class = 0:Bac

amples = 1

lass = 0:Ba

samples = 92

value = [31, 61]

class = 0:Bad

samples = 9

value = [3, 6] class = 0:Bad

class = 0:Ba

samples = 17

value = [7, 10] class = 0:Bad samples = 22

value = [9, 13]

class = 0:Bad

samples = 9

value = [8, 1]

class = 1:Good

samples = 27

value = [21, 6]

class = 1:Good

samples = 16

class = 0:Bac