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 l in list1:
        dtc10 = cross_val_score(dtc, x, y, scoring=l, cv=10)
        mean = dtc10.mean()
        std = dtc10.std()
        mean_score.append(mean)
        std_score.append(std)
        print("{:.<13s}{:>7.4f}{:>10.4f}".format(l, 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]
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())
```

2)

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= 8

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= 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= 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= 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= 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

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

3)

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