**STAT-656 Assignment -5**

Submited by

A. Pawan Dixit

UIN: 224004020

**Part 2: Python**

1. **A listing of your python code.**

**from** Class\_tree **import** DecisionTree

**from** Class\_replace\_impute\_encode **import** ReplaceImputeEncode

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.tree **import** export\_graphviz

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.model\_selection **import** cross\_val\_score

**from** sklearn.metrics **import** accuracy\_score, precision\_score, recall\_score

**from** sklearn.metrics **import** f1\_score, confusion\_matrix, classification\_report

**from** pydotplus **import** graph\_from\_dot\_data

**import** graphviz

**import** pandas as pd

**import** numpy as np

file\_path = '/Users/Pawan/Desktop/MS Folder 😇😇✈️/4th Sem/STAT 656/Week 5/'

df = pd.read\_excel(file\_path+"CreditHistory\_Clean.xlsx")

**print**("Authentication Data with %i observations & %i attributes.\n" %df.shape, df.head())

# Replace str values of atrribute 'purpose' with float values

list\_1 = [0,1,2,3,4,5,6,7,8,9,10]

df['purpose'].replace(['0','1','2','3','4','5','6','7','8','9','X'],list\_1, inplace = True)

attribute\_map = {

    'age':[0,(1, 120),[0,0]],

    'amount':[0,(0, 100000),[0,0]],

    'duration':[0,(1,1000),[0,0]],

    'checking':[2,(1, 2, 3, 4),[0,0]],

    'coapp':[2,(1,2,3),[0,0]],

    'depends':[1,(1,2),[0,0]],

    'employed':[2,(1,2,3,4,5),[0,0]],

    'existcr':[2,(1,2,3,4),[0,0]],

    'foreign':[1,(1,2),[0,0]],

    'good\_bad':[1,('bad', 'good'),[0,0]],

    'history':[2,(0,1,2,3,4),[0,0]],

    'housing':[2,(1,2,3,),[0,0]],

    'installp':[2,(1,2,3,4),[0,0]],

    'job':[2,(1,2,3,4),[0,0]],

    'marital':[2,(1,2,3,4),[0,0]],

    'other':[2,(1,2,3),[0,0]],

    'property':[2,(1,2,3,4),[0,0]],

    'purpose':[1,(0,1,2,3,4,5,6,7,8,9,10),[0,0]],

    'resident':[2,(1,2,3,4),[0,0]],

    'savings':[2,(1,2,3,4,5),[0,0]],

    'telephon':[1,(1,2),[0,0]] }

encoding = None # Categorical encoding:  Use 'SAS', 'one-hot' or None

scale    = None  # Interval scaling:  Use 'std', 'robust' or None

scaling  = 'No'  # Text description for interval scaling

rie = ReplaceImputeEncode(data\_map=attribute\_map, nominal\_encoding=encoding,

                          interval\_scale = scale, display=False)

#features\_map = rie.draft\_features\_map(df)

encoded\_df = rie.fit\_transform(df)

# First Integer Designates Data Type

# 0=Interval, 1=Binary, 2=Nominal, 3=Other (No Changes, do not include)

#good\_bad is the name of the binary target

varlist = ['good\_bad']

X = encoded\_df.drop(varlist, axis=1)

y = encoded\_df[varlist]

X\_train, X\_validate, y\_train, y\_validate = train\_test\_split(X,y,test\_size = 0.3, random\_state=7)

# 10 fold CV for trees with different 'max\_depth' varying from 1 to 10

**for** m **in** range(0,10):

    dtc = DecisionTreeClassifier(criterion='gini', max\_depth = m+1,min\_samples\_split=5, min\_samples\_leaf=5)

    dtc = dtc.fit(X\_train,y\_train)

    features = X\_train.columns.values.tolist()

    classes = ['Good', 'Bad']

    FI = list();

**for** i **in** range(0,20):

        FI.append(dtc.feature\_importances\_[i])

    len(FI)

    FI\_df = pd.DataFrame({'Feature':features,'Importance':FI})

    predictions = dtc.predict(X\_validate)

    dtc\_10\_f1 = cross\_val\_score(dtc, X\_train, y\_train, cv=10,scoring='f1')

    dtc\_10\_accuracy = cross\_val\_score(dtc, X\_train, y\_train, cv=10,scoring='accuracy')

    dtc\_10\_precision = cross\_val\_score(dtc, X\_train, y\_train, cv=10,scoring='precision')

    dtc\_10\_recall = cross\_val\_score(dtc, X\_train, y\_train, cv=10,scoring='recall')

    dot\_data = export\_graphviz(dtc, filled=True, rounded=True,class\_names=classes, feature\_names = features, out\_file=None)

    #write tree to png file 'credit\_history\_tree'

    graph\_png = graph\_from\_dot\_data(dot\_data)

    graph\_path = '/Users/Pawan/Desktop/MS Folder 😇😇✈️/4th Sem/STAT 656/Week 5/'

    graph\_png.write\_png(graph\_path+'credit\_history\_tree\_'+str(m+1)+'.png')

    #Display tree in pdf file 'credit\_history\_tree.pdf'

    #graph\_pdf = graphviz.Source(dot\_data)

    #graph\_pdf.view('credit\_history\_tree') #Displays tree

    Matrix = np.empty([10,4])

**for** n **in** range(0,10):

        Matrix[n,0] = dtc\_10\_precision[n]

        Matrix[n,1] = dtc\_10\_recall[n]

        Matrix[n,2] = dtc\_10\_f1[n]

        Matrix[n,3] = dtc\_10\_accuracy[n]

    Index = ['1st Fold','2nd Fold','3rd Fold','4th Fold','5th Fold','6th Fold','7th Fold','8th Fold','9th Fold','10th Fold'];

    Columns = ['Precision', 'Recall/TPR', 'F1\_Score', 'Accuracy']

    Matrix = pd.DataFrame(Matrix, index = Index, columns = Columns)

**print**('\n\nTable of metrics for 10 Folds CV of model with MAX\_TREE\_DEPTH = '+str(m+1)+'\n')

**print**(Matrix)

**print**('\nAvg\_Precision = '+ str(Matrix['Precision'].mean()))

**print**('Avg\_Recall = '+ str(Matrix['Recall/TPR'].mean()))

**print**('Avg\_F1 = '+ str(Matrix['F1\_Score'].mean()))

**print**('Avg\_Accuracy = '+ str(Matrix['Accuracy'].mean()))

# Best model based on 10-fold CV results

dtc = DecisionTreeClassifier(criterion='gini', max\_depth = 8,min\_samples\_split=5, min\_samples\_leaf=5)

dtc = dtc.fit(X\_train,y\_train)

features = X\_train.columns.values.tolist()

classes = ['Good', 'Bad']

FI = list();

**for** i **in** range(0,20):

    FI.append(dtc.feature\_importances\_[i])

len(FI)

FI\_df = pd.DataFrame({'Feature':features,'Importance':FI})

**print**('\nDisplaying Feature Importance for the best Model:\n')

**print**(FI\_df)

predictions = dtc.predict(X\_validate)

acc = accuracy\_score(y\_validate, predictions)

pre = precision\_score(y\_validate, predictions)

tpr = recall\_score(y\_validate, predictions)

f1 =  f1\_score(y\_validate,predictions)

Matrix = np.empty([1,4])

Matrix[0,0] = pre

Matrix[0,1] = tpr

Matrix[0,2] = f1

Matrix[0,3] = acc

Index = ['Max\_depth\_8'];

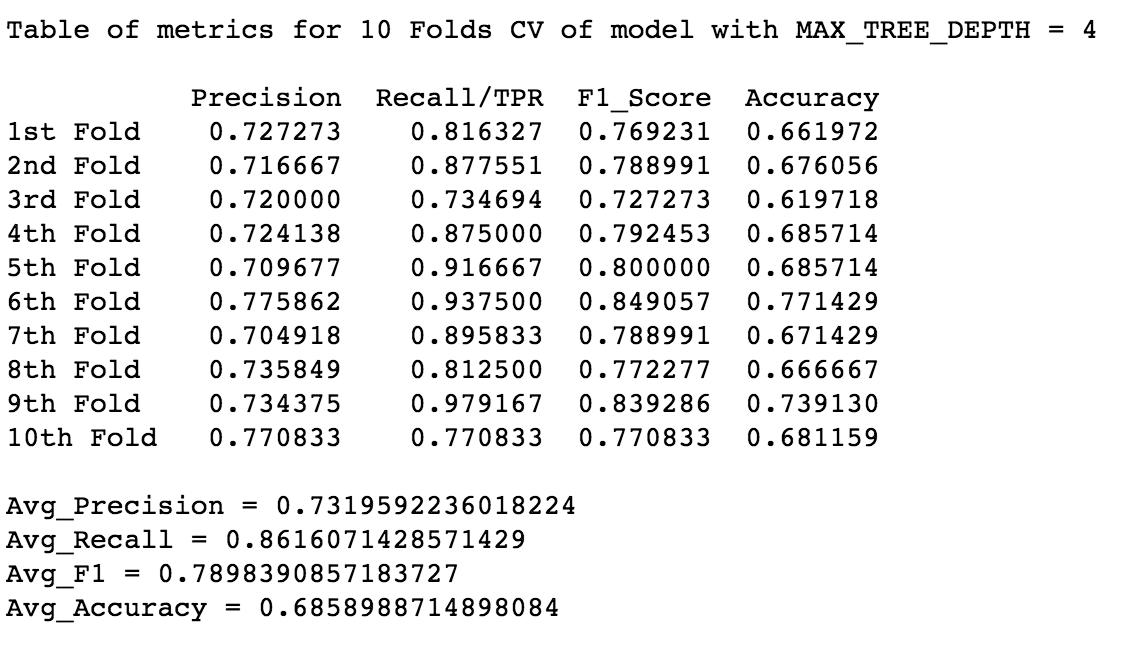
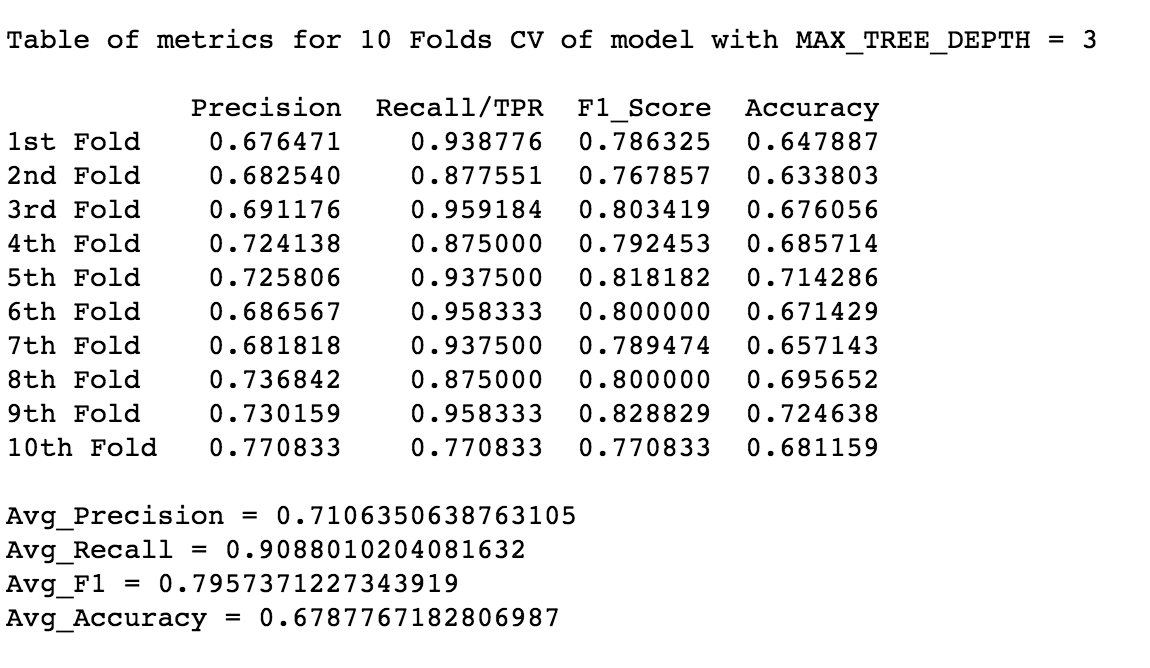
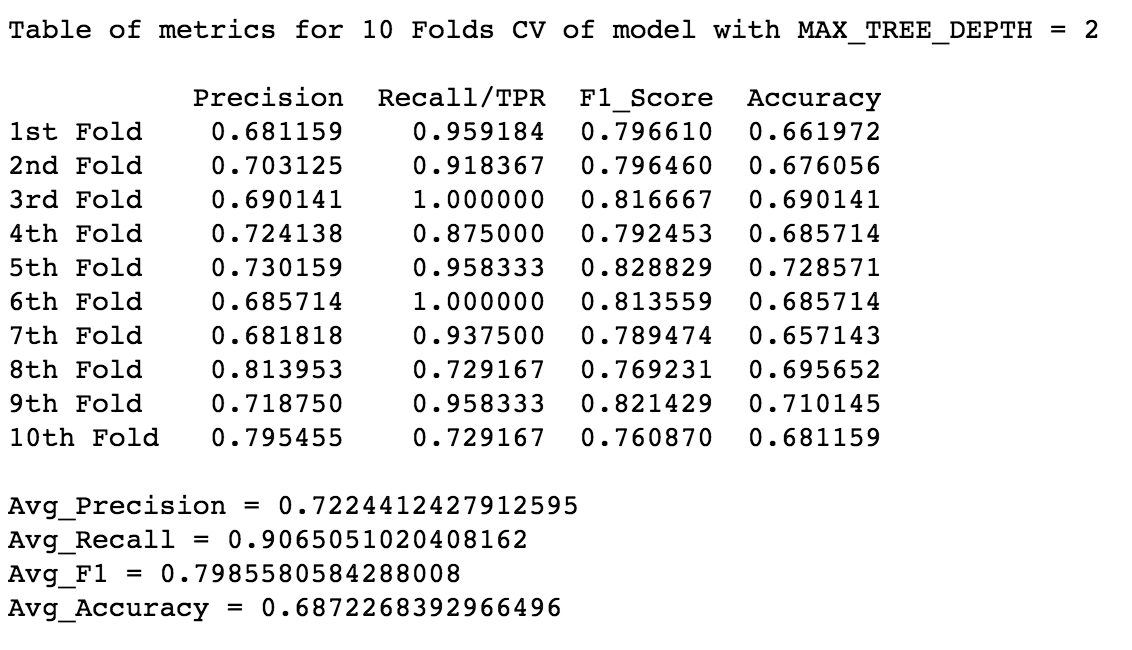
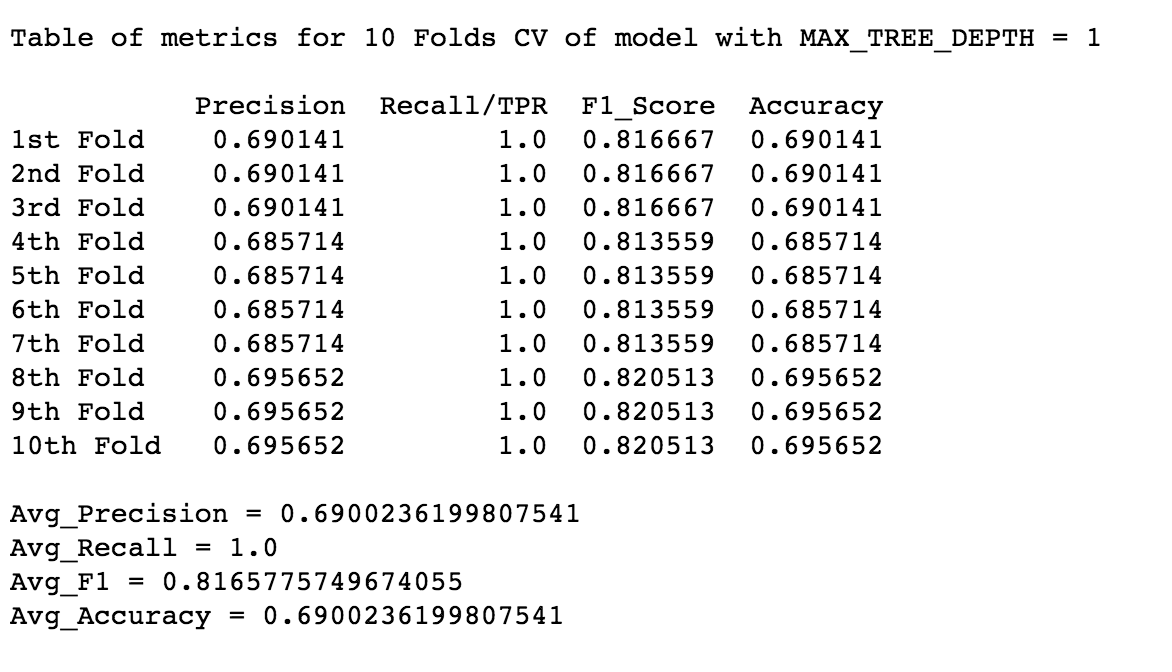
Columns = ['Precision', 'Recall/TPR', 'F1\_Score', 'Accuracy']

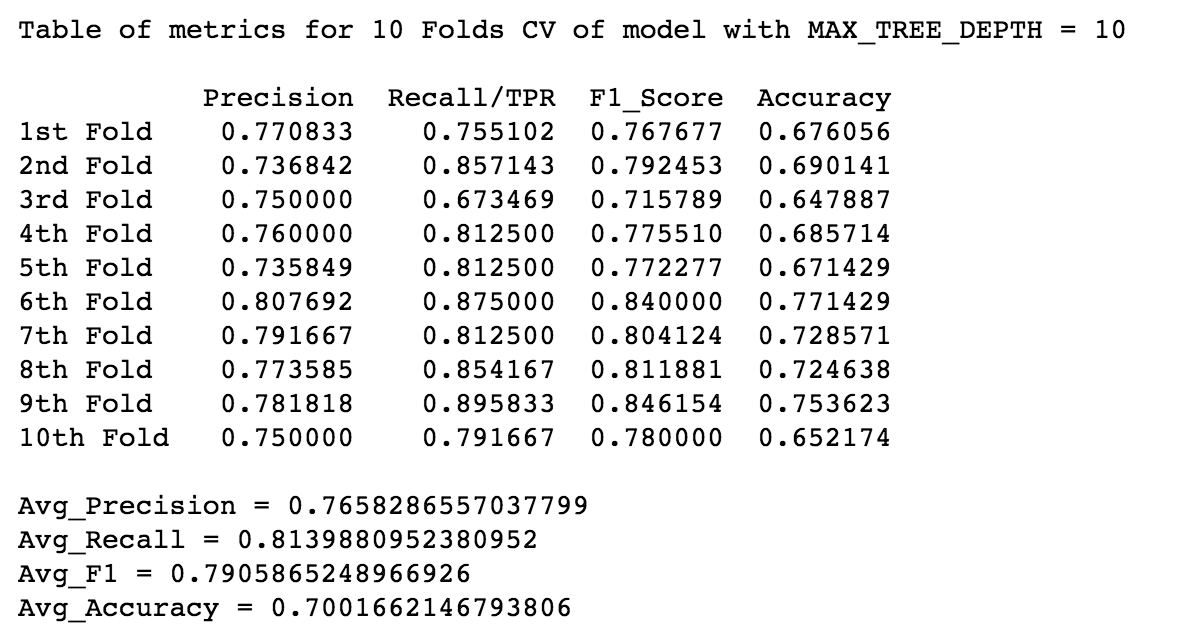
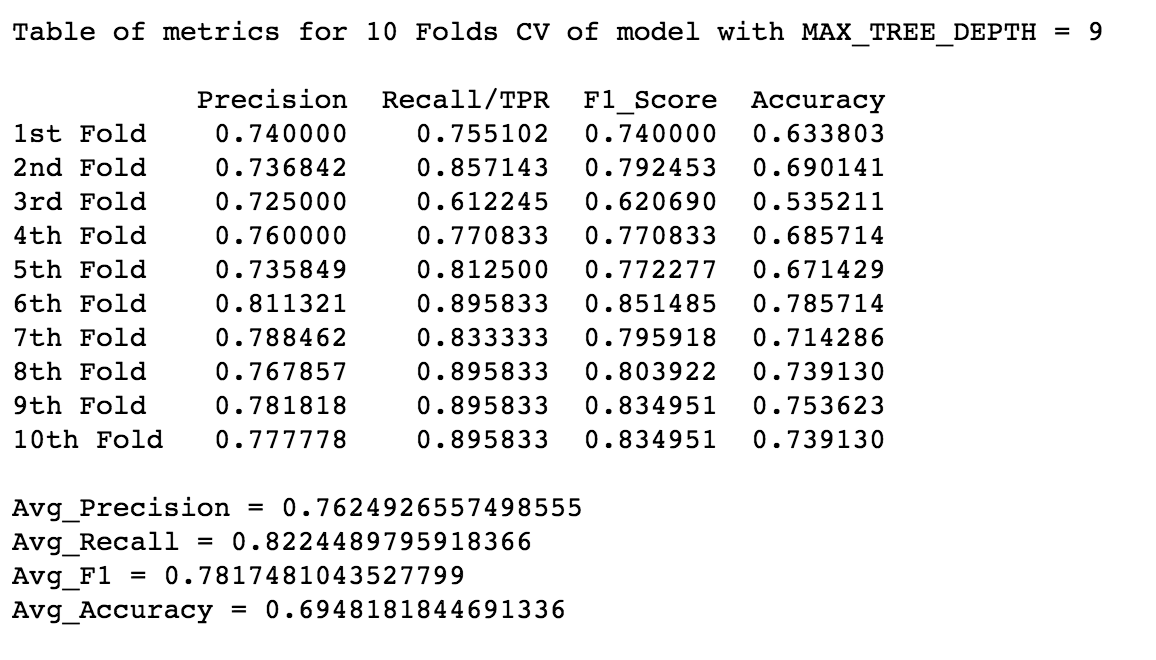
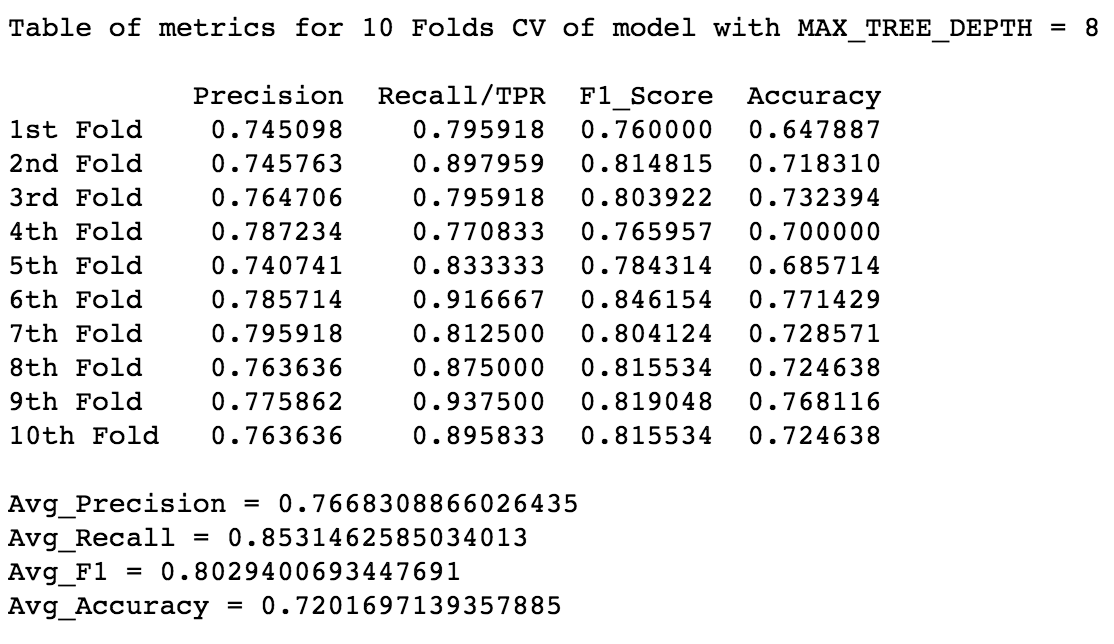
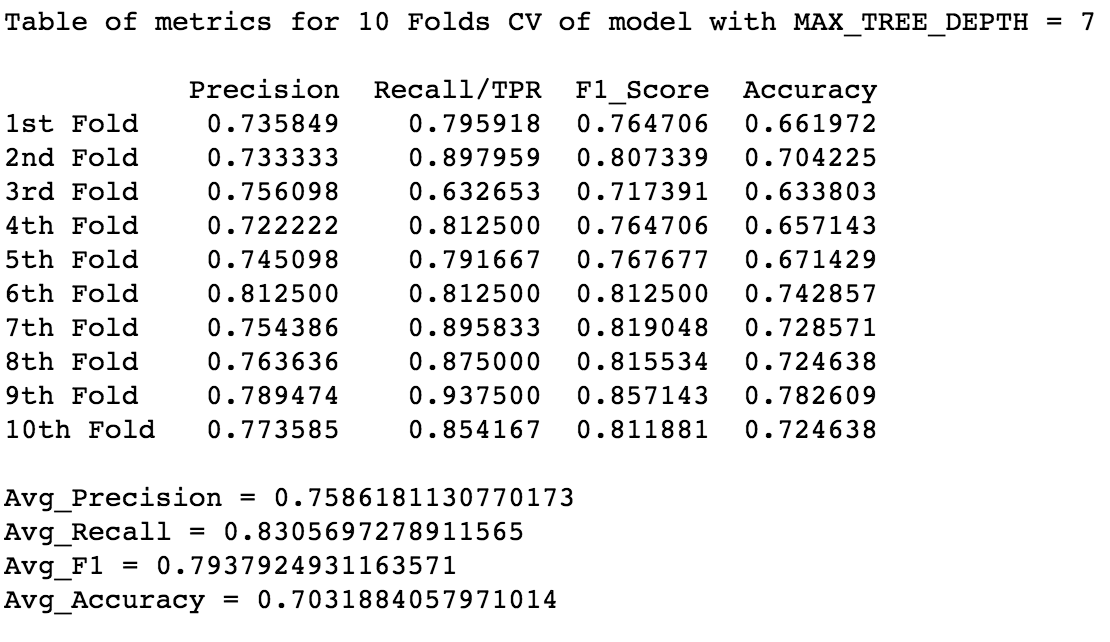
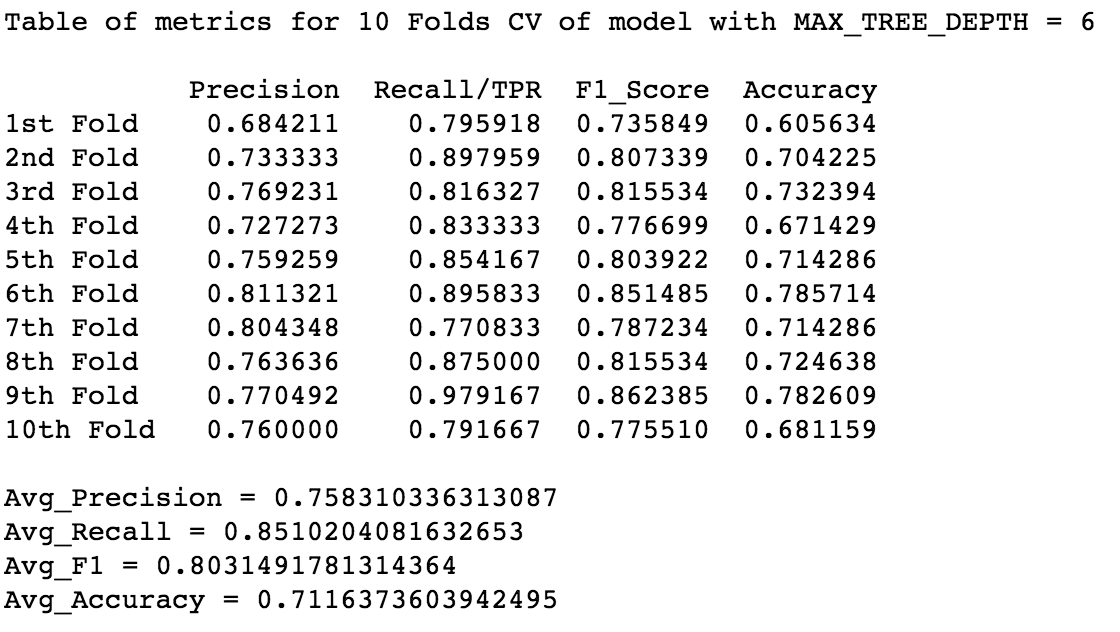
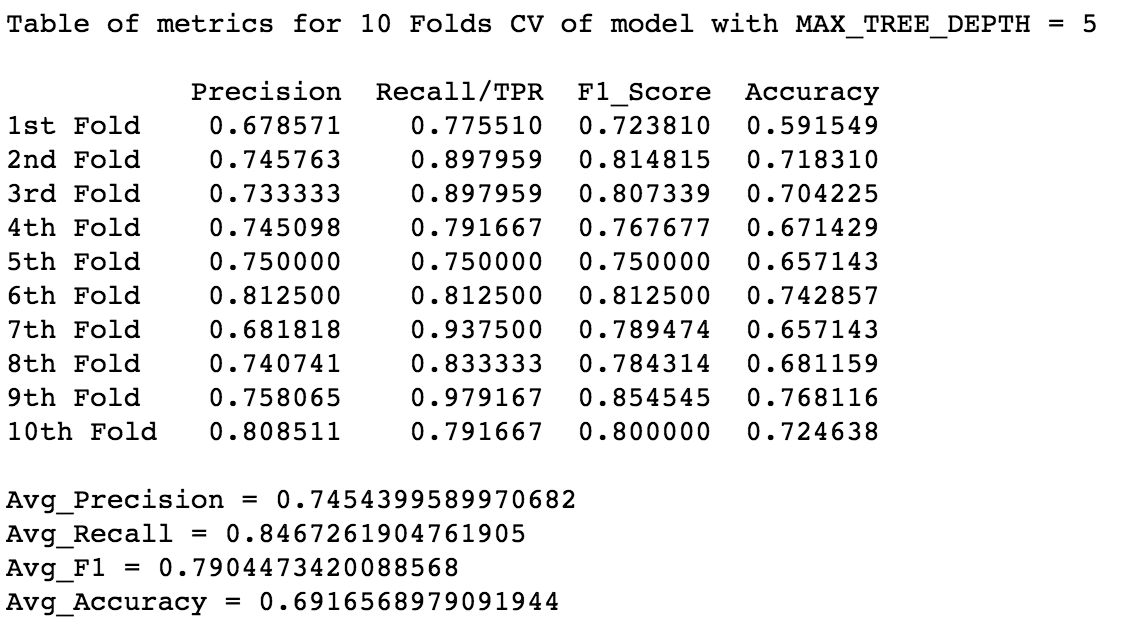
Matrix = pd.DataFrame(Matrix, index = Index, columns = Columns)

**print**("\n\nA table of metrics calculated for best model\n")

**print**(Matrix)

1. **A table of the metrics (recall, accuracy, precision and F1) calculated for each of your 10 cross-validation folds.**

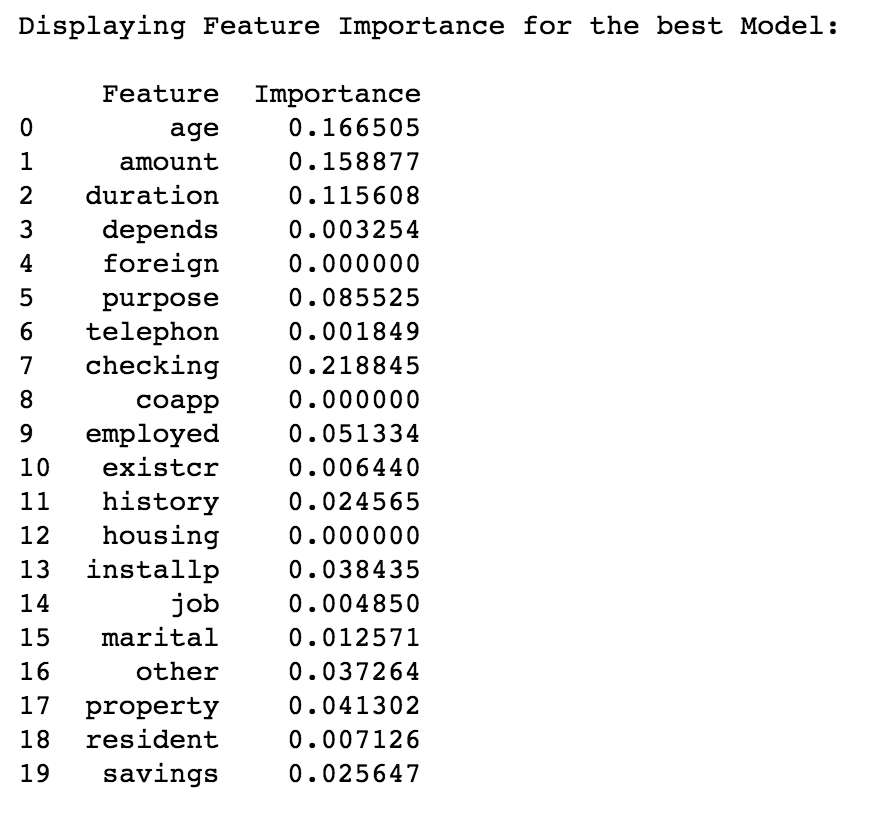
****

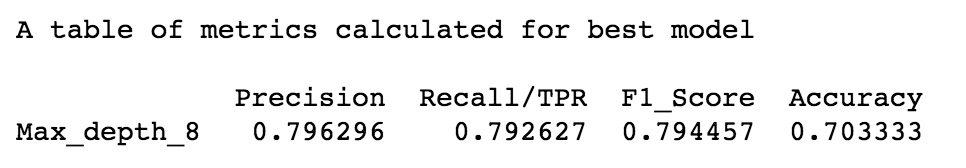
****

1. **Describe which model you selected from Cross-Validation, and why.**

Model with maximum tree depth 8 has been selected as the best model as it has yielded an accuracy of 72% over 10-folds cross validation. Other metrics also support this decision.

1. **A table of the metrics (recall, accuracy, precision and F1) for the 70/30 split using your selected model.**

****

****

1. **A screen shot of your tree.**