**Attrition Decisiontree**

**Import packages:**

import pandas as pd

import numpy as np

from sklearn import tree

from sklearn import preprocessing

# import data:

da=pd.read\_csv("general\_data.csv")

**Describe the data:**

da.describe()

da.NumCompaniesWorked.mean()

da.TotalWorkingYears.mean()

#checking for null values

da.isnull().sum()

nocom=np.where(da["NumCompaniesWorked"].isnull(),2.69,da["NumCompaniesWorked"])

twy=np.where(da["TotalWorkingYears"].isnull(),11,da["TotalWorkingYears"])

da['NumCompaniesWorked']=nocom

da['TotalWorkingYears']=twy

da.isnull().sum()

#convert catagorical value as numarical

label\_encoder=preprocessing.LabelEncoder()

da["Attrition"]=label\_encoder.fit\_transform(da["Attrition"])

da["Department"]=label\_encoder.fit\_transform(da["Department"])

da["BusinessTravel"]=label\_encoder.fit\_transform(da["BusinessTravel"])

da["EducationField"]=label\_encoder.fit\_transform(da["EducationField"])

da["Gender"]=label\_encoder.fit\_transform(da["Gender"])

da["JobRole"]=label\_encoder.fit\_transform(da["JobRole"])

da["MaritalStatus"]=label\_encoder.fit\_transform(da["MaritalStatus"])

da["EducationField"]=label\_encoder.fit\_transform(da["EducationField"])

del da['Over18']

**Build the model:**

xprr=pd.DataFrame([da['Age'],da['BusinessTravel'],da['Department'],da[ 'DistanceFromHome']

,da['Education'],da['EducationField'],da['EmployeeCount'], da['EmployeeID'],da['Gender'],da['JobLevel'],da['JobRole'], da['MaritalStatus'],da['MonthlyIncome'],da['NumCompaniesWorked'],da[ 'PercentSalaryHike'],da['StandardHours'],da['StockOptionLevel'],da[ 'TotalWorkingYears'],da['TrainingTimesLastYear'],da['YearsAtCompany'],da['YearsSinceLastPromotion']

,da['YearsWithCurrManager']]).T

tree\_model=tree.DecisionTreeClassifier(max\_depth=8)

tree\_model.fit(X=xprr,y=da['Attrition'])

**Pictorial form:**

with open("dtree.dot",'w') as f:

f=tree.export\_graphviz(tree\_model,feature\_names=list(xprr)

,out\_file=f);

**Accuracy of model:**

tree\_model.score(X=xprr,y=da['Attrition'])

Out[22]: 0.9206349206349206

#predict the values

testfea=pd.DataFrame([da['Age'],da['BusinessTravel'],da['Department'],da[ 'DistanceFromHome']

,da['Education'],da['EducationField'],da['EmployeeCount'], da['EmployeeID'],da['Gender'],da['JobLevel'],da['JobRole'], da['MaritalStatus'],da['MonthlyIncome'],da['NumCompaniesWorked'],da[ 'PercentSalaryHike'],da['StandardHours'],da['StockOptionLevel'],da[ 'TotalWorkingYears'],da['TrainingTimesLastYear'],da['YearsAtCompany'],da['YearsSinceLastPromotion']

,da['YearsWithCurrManager']]).T

testpre=tree\_model.predict(X=testfea)

preoutput=pd.DataFrame({"Age":da["Age"],"Attrition":testpre})

**Random forest:**

from sklearn.ensemble import RandomForestClassifier

rfmod=RandomForestClassifier(n\_estimators=1000,max\_features=2,oob\_score=True)

fea=pd.DataFrame([da['Age'],da['BusinessTravel'],da['Department'],da[ 'DistanceFromHome']

,da['Education'],da['Gender'],da['JobLevel'],da['JobRole'], da['MaritalStatus'],da['MonthlyIncome'],da['NumCompaniesWorked'],da[ 'PercentSalaryHike'],da['StandardHours'],da['StockOptionLevel'],da[ 'TotalWorkingYears'],da['TrainingTimesLastYear'],da['YearsAtCompany'],da['YearsSinceLastPromotion']

,da['YearsWithCurrManager']]).T

rfmod.fit(X=fea,y=da["Attrition"])

#oob accuracy score

print(rfmod.oob\_score\_);

1.0

#final model:

for feature,imp in zip(fea,rfmod.feature\_importances\_):

print((feature,imp));

**Important values:**

imp=pd.DataFrame([da['Age']

,da['MonthlyIncome'],da[ 'TotalWorkingYears']

,da[ 'DistanceFromHome'],da['YearsAtCompany'],da[ 'PercentSalaryHike'],da['JobRole']]).T

rfmod.fit(X=imp,y=da["Attrition"])

print(rfmod.oob\_score\_);

finalmod=tree.DecisionTreeClassifier(max\_depth=8)

finalmod.fit(X=imp, y=da["Attrition"])

finalmod.score(X=imp, y=da["Attrition"])

**Final tree**:

with open("ftreeA.dot",'w') as f:

f=tree.export\_graphviz(finalmod,feature\_names=list(imp)

,out\_file=f);

('Age', 0.10087168381977125)

('BusinessTravel', 0.029324939763527985)

('Department', 0.029764633435598504)

('DistanceFromHome', 0.07309712678933783)

('Education', 0.04270282102917345)

('Gender', 0.01934505882923044)

('JobLevel', 0.03888306367989993)

('JobRole', 0.05823453172736768)

('MaritalStatus', 0.04064308774536752)

('MonthlyIncome', 0.09790101635231586)

('NumCompaniesWorked', 0.05946697555385193)

('PercentSalaryHike', 0.06803374003654918)

('StandardHours', 0.0)

('StockOptionLevel', 0.03559748209467958)

('TotalWorkingYears', 0.08728264883431486)

('TrainingTimesLastYear', 0.047643954745455755)

('YearsAtCompany', 0.07158660624671899)

('YearsSinceLastPromotion', 0.04510004507436765)

('YearsWithCurrManager', 0.05452058424247169)

0.9997732426303855

**Predictor:**

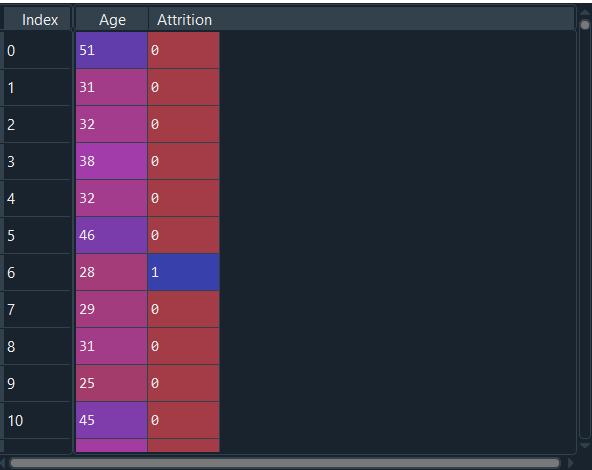
pre=pd.DataFrame([da['Age']

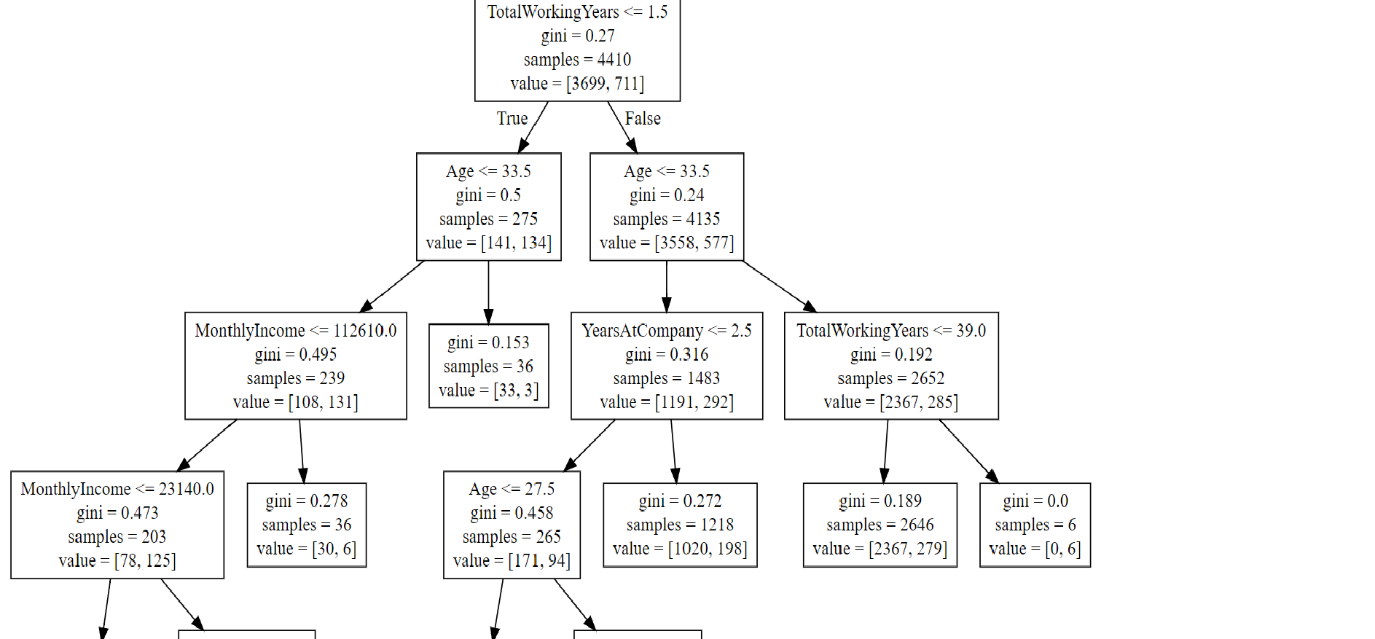
,da['MonthlyIncome'],da[ 'TotalWorkingYears']

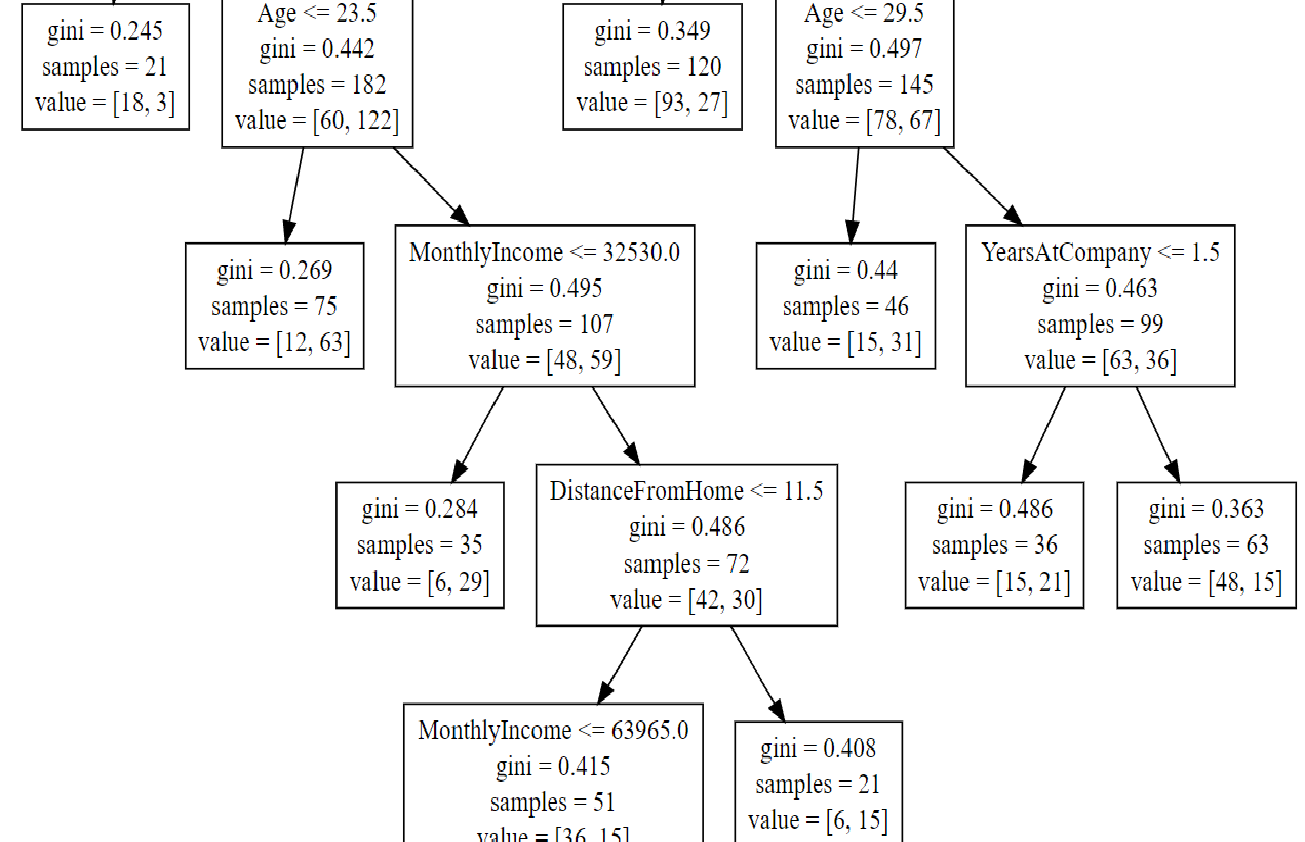
,da[ 'DistanceFromHome'],da['YearsAtCompany'],da[ 'PercentSalaryHike'],da['JobRole']]).T

testfipre=finalmod.predict(X=pre)

finoutput=pd.DataFrame({"Age":da["Age"],"Attrition":testfipre})







Rules:

Here if peoples who have TotalWorkingYears <= 1.5=False &Age <= 33.5=False&TotalWorkingYears <= 39.0=False .they would get attrition yes.

If peoples who have TotalWorkingYears <= 1.5=True& Age <= 33.5=True& MonthlyIncome <= 112610.0=True& MonthlyIncome <= 23140.0=False& MonthlyIncome <= 32530.0=False& DistanceFromHome <= 11.5=True& MonthlyIncome <= 63965.0=True.they would get attrition no.