Project 2: Build Decision Tree for Attrition Rate Analysis

DV - "Attrition"

IDV - Output of RF Algorithm

Step 1: Load the dataset

import pandas as pd

import numpy as np

attrition\_dataset = pd.read\_csv("D:/AI\_ML\_Course/Day24/general\_data.csv")

attrition\_dataset.columns

Out[63]:

Index(['Age', 'Attrition', 'BusinessTravel', 'Department', 'DistanceFromHome',

'Education', 'EducationField', 'EmployeeCount', 'EmployeeID', 'Gender',

'JobLevel', 'JobRole', 'MaritalStatus', 'MonthlyIncome',

'NumCompaniesWorked', 'Over18', 'PercentSalaryHike', 'StandardHours',

'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',

'YearsAtCompany', 'YearsSinceLastPromotion', 'YearsWithCurrManager'],

dtype='object')

attrition\_dataset1 = attrition\_dataset.drop("EmployeeCount", axis=1)

attrition\_dataset2 = attrition\_dataset1.drop("EmployeeID", axis=1)

Take the mean of individual Variables & fill the empty/NaN values

attrition\_dataset2["Age"].mean() # 37

new\_age\_var = np.where(attrition\_dataset2["Age"].isnull(),37,attrition\_dataset2["Age"]) #replacing null values .

attrition\_dataset2['Age']=new\_age\_var # re-assigning

attrition\_dataset2["DistanceFromHome"].mean() # 9

new\_dfh\_var = np.where(attrition\_dataset2["DistanceFromHome"].isnull(),9,attrition\_dataset2["DistanceFromHome"]) #replacing null values .

attrition\_dataset2['DistanceFromHome']=new\_dfh\_var # re-assigning

attrition\_dataset2["Education"].mean() # 2.91

new\_ed\_var = np.where(attrition\_dataset2["Education"].isnull(),2.91,attrition\_dataset2["Education"]) #replacing null values .

attrition\_dataset2['Education']=new\_ed\_var # re-assigning

attrition\_dataset2["MonthlyIncome"].mean() # 65029

new\_mi\_var = np.where(attrition\_dataset2["MonthlyIncome"].isnull(),65029,attrition\_dataset2["MonthlyIncome"]) #replacing null values .

attrition\_dataset2['MonthlyIncome']=new\_mi\_var # re-assigning

attrition\_dataset2["PercentSalaryHike"].mean() # 15.2

new\_psh\_var = np.where(attrition\_dataset2["PercentSalaryHike"].isnull(),15.2,attrition\_dataset2["PercentSalaryHike"]) #replacing null values .

attrition\_dataset2['PercentSalaryHike']=new\_psh\_var # re-assigning

attrition\_dataset2["TotalWorkingYears"].mean() # 11.2

new\_twy\_var = np.where(attrition\_dataset2["TotalWorkingYears"].isnull(),11.2,attrition\_dataset2["TotalWorkingYears"]) #replacing null values .

attrition\_dataset2['TotalWorkingYears']=new\_twy\_var # re-assigning

attrition\_dataset2["YearsAtCompany"].mean() # 7

new\_yac\_var = np.where(attrition\_dataset2["YearsAtCompany"].isnull(),7,attrition\_dataset2["YearsAtCompany"]) #replacing null values .

attrition\_dataset2['YearsAtCompany']=new\_yac\_var # re-assigning

attrition\_dataset2["YearsSinceLastPromotion"].mean() # 2.18

new\_yslp\_var = np.where(attrition\_dataset2["YearsSinceLastPromotion"].isnull(),2.18,attrition\_dataset2["YearsSinceLastPromotion"]) #replacing null values .

attrition\_dataset2['YearsSinceLastPromotion']=new\_yslp\_var # re-assigning

attrition\_dataset2["YearsWithCurrManager"].mean() # 4.12

new\_ywc\_var = np.where(attrition\_dataset2["YearsWithCurrManager"].isnull(),4.12,attrition\_dataset2["YearsWithCurrManager"]) #replacing null values .

attrition\_dataset2['YearsWithCurrManager']=new\_ywc\_var # re-assigning

Step 2: Use the Random forest Algorithm to find the important features.

from sklearn import tree

from sklearn import preprocessing

from sklearn.ensemble import RandomForestClassifier

label\_encoder = preprocessing.LabelEncoder()

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

features = ['Age', 'DistanceFromHome', 'Education', 'MonthlyIncome', 'PercentSalaryHike', 'TotalWorkingYears', 'YearsAtCompany', 'YearsSinceLastPromotion', 'YearsWithCurrManager']

rf\_model.fit(X=attrition\_dataset2[features],y=attrition\_dataset2["Attrition"])

print("OOB Accuracy:")

print(rf\_model.oob\_score\_);

0.999546485260771

for feature, imp in zip(features, rf\_model.feature\_importances\_):

print(feature, imp)

# Age 0.15547281992753195

# DistanceFromHome 0.12922824165212077

# Education 0.06896446533919699

# MonthlyIncome 0.1926500759906495

# PercentSalaryHike 0.11059345227632589

# TotalWorkingYears 0.12818055754427296

# YearsAtCompany 0.08903552149690543

# YearsSinceLastPromotion 0.0573176228458971

# YearsWithCurrManager 0.06855724292709944

We observe that for the above parameters the P Value is more than 0.5, so this can be used for Decision Tree.

Step 3: Build the Decision Tree with important features (Independent variable) as Age, DistanceFromHome, MonthlyIncome, Education, PercentSalaryHike, TotalWorkingYears, YearsAtCompany, YearsSinceLastPromotion, YearsWithCurrManager and Dependent variable as Attrition

Model 1:

test\_features = pd.DataFrame([attrition\_dataset2["Age"],attrition\_dataset2["DistanceFromHome"],attrition\_dataset2["Education"],attrition\_dataset2["MonthlyIncome"],attrition\_dataset2["PercentSalaryHike"],attrition\_dataset2["TotalWorkingYears"],attrition\_dataset2["YearsAtCompany"],attrition\_dataset2["YearsSinceLastPromotion"],attrition\_dataset2["YearsWithCurrManager"]]).T

tree\_model = tree.DecisionTreeClassifier(max\_depth=18)

tree\_model.fit(X=test\_features, y=attrition\_dataset2['Attrition'])

with open("D:/AI\_ML\_Course/Day24/Dtree3.dot",'w') as f:

f = tree.export\_graphviz(tree\_model, feature\_names=['Age', 'DistanceFromHome', 'Education', 'MonthlyIncome', 'PercentSalaryHike', 'TotalWorkingYears', 'YearsAtCompany', 'YearsSinceLastPromotion', 'YearsWithCurrManager'],out\_file=f);



Step 4: To calculate the accuracy of the model

tree\_model.score(X=test\_features, y=attrition\_dataset2['Attrition'])

#Out[173]: 0.991156462585034

Inference: The model is 99.11% accurate

Step 5: Based on the training Dataset predicted the test dataset output

test\_preds = tree\_model.predict(X=test\_features)# .model\_selection.predict(test\_features)

predicted\_output =pd.DataFrame({"EmployeeID":attrition\_dataset['EmployeeID'],'Attrition':test\_preds})

predicted\_output.to\_csv("D:/Growth/AI\_ML\_Course/Day24/OutputAR.csv",index=False)



Model 2:

Reduce the Independent Variable count to 5 as Age, DistanceFromHome, MonthlyIncome, PercentSalaryHike, TotalWorkingYears and Dependent variable as Attrition

test\_features = pd.DataFrame([attrition\_dataset2["Age"],attrition\_dataset2["DistanceFromHome"],attrition\_dataset2["MonthlyIncome"],attrition\_dataset2["PercentSalaryHike"],attrition\_dataset2["TotalWorkingYears"]]).T

tree\_model = tree.DecisionTreeClassifier(max\_depth=10)

tree\_model.fit(X=test\_features,y=attrition\_dataset2['Attrition'])

with open("D:/AI\_ML\_Course/Day24/Dtree4.dot",'w') as f:

f = tree.export\_graphviz(tree\_model,feature\_names=['Age', 'DistanceFromHome', 'MonthlyIncome', 'PercentSalaryHike', 'TotalWorkingYears'],out\_file=f);



tree\_model.score(X=test\_features, y=attrition\_dataset2['Attrition'])

#Out[182]: 0.9045351473922902

Inference: The model accuracy is 90.4%

Model 3:

Reduce the Independent Variable count to 3 as Age, DistanceFromHome, MonthlyIncome and Dependent variable as Attrition

test\_features = pd.DataFrame([attrition\_dataset2["Age"],attrition\_dataset2["DistanceFromHome"],attrition\_dataset2["MonthlyIncome"]]).T

tree\_model = tree.DecisionTreeClassifier(max\_depth=6)

tree\_model.fit(X=test\_features, y=attrition\_dataset2['Attrition'])

with open("D:/Growth/AI\_ML\_Course/Day24/Dtree5.dot",'w') as f:

f = tree.export\_graphviz(tree\_model,feature\_names=['Age', 'DistanceFromHome', 'MonthlyIncome',],out\_file=f);



tree\_model.score(X=test\_features,y=attrition\_dataset2['Attrition'])

#Out[187]: 0.854421768707483

Inference: The model accuracy is 85.44