**Attrition Rate Analysis Decision Tree**

import pandas as pd

import numpy as np

from sklearn import preprocessing

from sklearn import tree

**Loading Data and Data Treatment:**

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

Attrition\_dataset.head(2)

Out[28]:

Age Attrition ... YearsSinceLastPromotion YearsWithCurrManager

0 51 No ... 0 0

1 31 Yes ... 1 4

[2 rows x 24 columns]

Attrition\_dataset.isnull().sum()

Out[29]:

Age 0

Attrition 0

BusinessTravel 0

Department 0

DistanceFromHome 0

Education 0

EducationField 0

EmployeeCount 0

EmployeeID 0

Gender 0

JobLevel 0

JobRole 0

MaritalStatus 0

MonthlyIncome 0

NumCompaniesWorked 19

Over18 0

PercentSalaryHike 0

StandardHours 0

StockOptionLevel 0

TotalWorkingYears 9

TrainingTimesLastYear 0

YearsAtCompany 0

YearsSinceLastPromotion 0

YearsWithCurrManager 0

dtype: int64

Attrition\_dataset.dtypes

Out[30]:

Age int64

Attrition object

BusinessTravel object

Department object

DistanceFromHome int64

Education int64

EducationField object

EmployeeCount int64

EmployeeID int64

Gender object

JobLevel int64

JobRole object

MaritalStatus object

MonthlyIncome int64

NumCompaniesWorked float64

Over18 object

PercentSalaryHike int64

StandardHours int64

StockOptionLevel int64

TotalWorkingYears float64

TrainingTimesLastYear int64

YearsAtCompany int64

YearsSinceLastPromotion int64

YearsWithCurrManager int64

dtype: object

Attrition\_dataset['NumCompaniesWorked'].mean()

Out[31]: 2.6948303347756775

Attrition\_dataset['TotalWorkingYears'].mean()

Out[32]: 11.279936378095888

Attrition\_dataset = Attrition\_dataset.fillna(Attrition\_dataset.mean().round())

**Encoding Categorical Variables**

label\_encoder = preprocessing.LabelEncoder()

Attrition\_dataset['Attrition']= label\_encoder.fit\_transform(Attrition\_dataset['Attrition'])

Attrition\_dataset['BusinessTravel']= label\_encoder.fit\_transform(Attrition\_dataset['BusinessTravel'])

Attrition\_dataset['Department']= label\_encoder.fit\_transform(Attrition\_dataset['Department'])

Attrition\_dataset['EducationField']= label\_encoder.fit\_transform(Attrition\_dataset['EducationField'])

Attrition\_dataset['Gender']= label\_encoder.fit\_transform(Attrition\_dataset['Gender'])

Attrition\_dataset['JobRole']= label\_encoder.fit\_transform(Attrition\_dataset['JobRole'])

Attrition\_dataset['MaritalStatus']= label\_encoder.fit\_transform(Attrition\_dataset['MaritalStatus'])

**Random Forest Algorithm to find imp Variables**

Attrition\_dataset.columns

Out[53]:

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')

features = ['Age', 'BusinessTravel', 'Department', 'DistanceFromHome',

'Education', 'EducationField', 'Gender',

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

'NumCompaniesWorked', 'PercentSalaryHike',

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

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

from sklearn.ensemble import RandomForestClassifier

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

rf\_model.fit(X= Attrition\_dataset[features], y= Attrition\_dataset['Attrition'])

Out[57]:

RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0, class\_weight=None,

criterion='gini', max\_depth=None, max\_features=2,

max\_leaf\_nodes=None, max\_samples=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=1000,

n\_jobs=None, oob\_score=True, random\_state=None,

verbose=0, warm\_start=False)

print("RF Model Accuracy:", rf\_model.oob\_score\_)

***RF Model Accuracy: 1.0***

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

print(feature,imp)

Age 0.09613519722199168

BusinessTravel 0.02801366900918839

Department 0.026226853193719875

DistanceFromHome 0.06999570939318761

Education 0.0410716440494469

EducationField 0.04181610388060817

Gender 0.01828282020883205

JobLevel 0.03745736849475484

JobRole 0.054763041247650156

MaritalStatus 0.039995957293941474

MonthlyIncome 0.09432953065023403

NumCompaniesWorked 0.05550785784873224

PercentSalaryHike 0.06515856485702474

StockOptionLevel 0.03427463376640986

TotalWorkingYears 0.08669342399172603

TrainingTimesLastYear 0.04469918689728264

YearsAtCompany 0.06771291312655314

YearsSinceLastPromotion 0.0437594427746118

YearsWithCurrManager 0.05410608209410441

**Generating Decision Tree Model**

predictors1 = ['Age','DistanceFromHome','JobRole','MonthlyIncome','NumCompaniesWorked','PercentSalaryHike','TotalWorkingYears','YearsAtCompany','YearsWithCurrManager']

predictors = ['Age','DistanceFromHome','MonthlyIncome','PercentSalaryHike','TotalWorkingYears','YearsAtCompany']

**Model 1: [ 6 features (imp value > 0.06) and max-depth = 10] - Accuracy- 92%**

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

tree\_model.fit(X= Attrition\_dataset[predictors], y= Attrition\_dataset['Attrition'])

Out[63]:

DecisionTreeClassifier(ccp\_alpha=0.0, class\_weight=None, criterion='gini',

max\_depth=10, max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort='deprecated',

random\_state=None, splitter='best')

with open("Attrition\_DTree1.dot","w") as f:

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

print("DTree Model Accuracy:", tree\_model.score(X= Attrition\_dataset[predictors], y= Attrition\_dataset['Attrition']))

***DTree Model Accuracy: 0.9238095238095239***

**Model 2: [ 6 features (imp value > 0.06) and max-depth = 12] - Accuracy- 94.8%**

tree\_model = tree.DecisionTreeClassifier(max\_depth= 12)

tree\_model.fit(X= Attrition\_dataset[predictors], y= Attrition\_dataset['Attrition'])

Out[67]:

DecisionTreeClassifier(ccp\_alpha=0.0, class\_weight=None, criterion='gini',

max\_depth=12, max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort='deprecated',

random\_state=None, splitter='best')

print("DTree Model Accuracy:", tree\_model.score(X= Attrition\_dataset[predictors], y= Attrition\_dataset['Attrition']))

***DTree Model Accuracy: 0.9480725623582766***

with open("Attrition\_DTree2.dot","w") as f:

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

**Model 3: [ 9 features (imp value > 0.05) and max-depth = 12] - Accuracy- 95.8%**

tree\_model.fit(X= Attrition\_dataset[predictors1], y= Attrition\_dataset['Attrition'])

Out[70]:

DecisionTreeClassifier(ccp\_alpha=0.0, class\_weight=None, criterion='gini',

max\_depth=12, max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort='deprecated',

random\_state=None, splitter='best')

print("DTree Model Accuracy:", tree\_model.score(X= Attrition\_dataset[predictors1], y= Attrition\_dataset['Attrition']))

***DTree Model Accuracy: 0.9582766439909297***

with open("Attrition\_DTree3.dot","w") as f:

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

**Model 4: [ 9 features (imp value > 0.05) and max-depth = 18] - Accuracy- 99.3%**

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

tree\_model.fit(X= Attrition\_dataset[predictors1], y= Attrition\_dataset['Attrition'])

Out[76]:

DecisionTreeClassifier(ccp\_alpha=0.0, class\_weight=None, criterion='gini',

max\_depth=18, max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort='deprecated',

random\_state=None, splitter='best')

print("DTree Model Accuracy:", tree\_model.score(X= Attrition\_dataset[predictors1], y= Attrition\_dataset['Attrition']))

***DTree Model Accuracy: 0.9931972789115646***

with open("Attrition\_DTree4.dot","w") as f:

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

**Inference:**

1. Based on the importance value generated with Random forest algorithm, it is seen that the features **‘Age', 'DistanceFromHome', 'JobRole', 'MonthlyIncome', 'NumCompaniesWorked', 'PercentSalaryHike', 'TotalWorkingYears', 'YearsAtCompany' and 'YearsWithCurrManager'** are more significant for decision tree generation.
2. Four Models are generated on basis of tree’s max-depth and the sets of features used for Decision tree creation. Summary is as follows

Set 1: imp value > 0.06 – ['Age','DistanceFromHome','MonthlyIncome','PercentSalaryHike','TotalWorkingYears','YearsAtCompany']

Set 2: imp value > 0.05 - ['Age','DistanceFromHome','JobRole','MonthlyIncome','NumCompaniesWorked','PercentSalaryHike','TotalWorkingYears','YearsAtCompany','YearsWithCurrManager']

|  |  |  |
| --- | --- | --- |
| Feature Set | Max-depth | Accuracy |
| Set 1 | 10 | 92% |
| Set 1 | 12 | 94.8% |
| Set 2 | 12 | 95.8% |
| Set 2 | 18 | 99.3% |

1. Increasing the no. of significant features and max-depth, increases the accuracy of the model. But the Decision tree becomes complex.