

**0**

51

No

Travel\_Rarely

Sales

6

2

Life Sciences

1

1

Female

**2**

32

Research &

Development

17

4

Other

1

3

Male

**4**

32

No

Travel\_Rarely

Research &

Development

10

1

Medical

1

5

Male

**6**

28

Yes

Travel\_Rarely

Research &

Development

11

2

Medical

1

7

Male

**8**

31

No

Travel\_Rarely

Research &

Development

3

Life Sciences

1

9

Male

In [1]:

**import pandas as pd**

**import numpy as np**

In [2]:

s=pd.read\_csv('general\_data.csv')

In [3]:

s.head(10 )

Out[3]:

**Age Attrition BusinessTravel Department DistanceFromHome Education EducationField EmployeeCount EmployeeID Gender ...**

...

**1** 31 Yes Travel\_Frequently

No Travel\_Frequently

**3** 38 No Non-Travel

**5** 46 No Travel\_Rarely

**7** 29 No Travel\_Rarely

**9** 25 No Non-Travel

Research &

Development

Research &

Development

Research &

Development

Research &

Development

Research &

Development

10 1 Life Sciences 1 2 Female ...

...

2 5 Life Sciences 1 4 Male ...

...

8 3 Life Sciences 1 6 Female ...

...

18 3 Life Sciences 1 8 Male ... 1

...

7 4 Medical 1 10 Female ...

10 rows × 24 columns

**checking for null values**

In [4]:

s.isnull() .sum()

Out[4]:

Age 0

1. ttrition 0
2. usinessTravel 0

Department 0

1. istanceFromHome 0
2. ducation 0

EducationField 0

EmployeeCount 0

EmployeeID 0

Gender 0

JobLevel 0

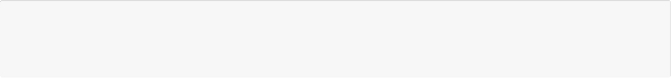
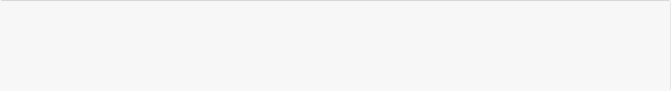
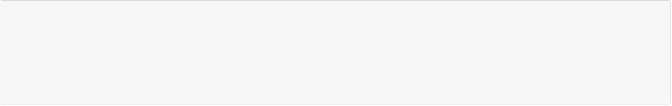
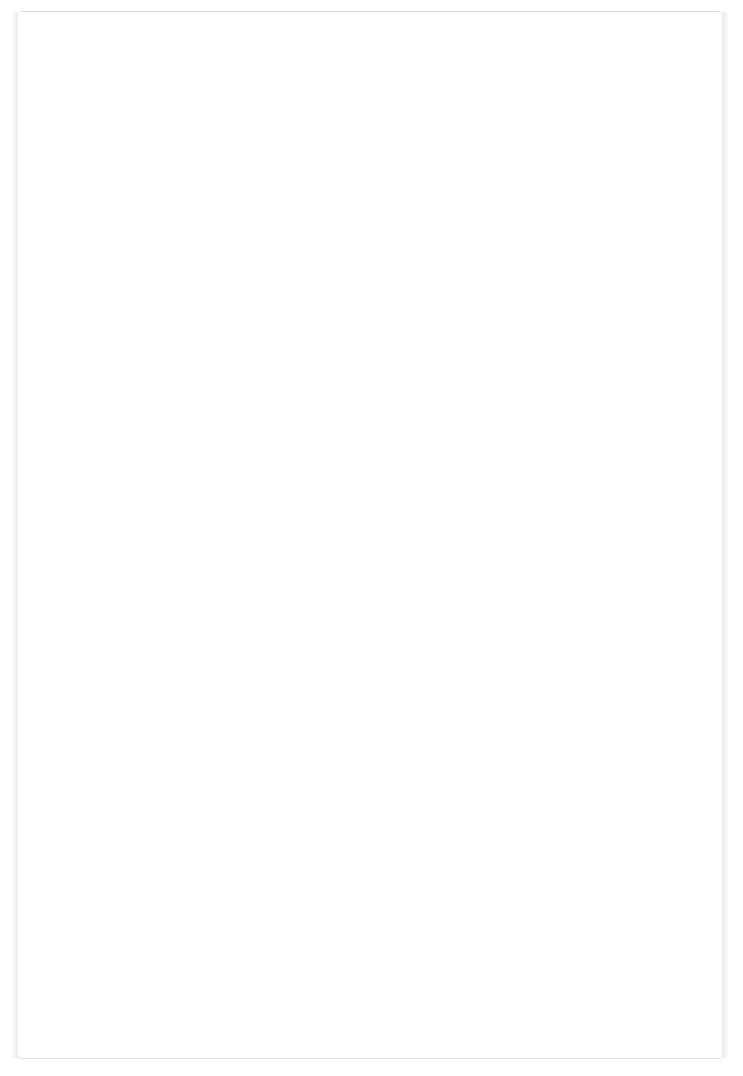
JobRole 0

MaritalStatus 0

1. onthlyIncome 0
2. umCompaniesWorked 19
3. ver18 0
4. ercentSalaryHike 0

StandardHours 0

StockOptionLevel 0



**0**

51

No

Travel\_Rarely

Sales

6

2

Life Sciences

1

1

Female

**2**

32

Research &

Development

17

4

Other

1

3

Male

**4**

32

No

Travel\_Rarely

Research &

Development

10

1

Medical

1

5

Male

1. tockOptionLevel 0
2. otalWorkingYears 9

TrainingTimesLastYear 0

YearsAtCompany 0

YearsSinceLastPromotion 0

YearsWithCurrManager 0

dtype: int64

**filling null values**

In [5]:

df=s['NumCompaniesWorked']

d=df.fillna( 2)

s['NumCompaniesWorked']=d

df1=s['TotalWorkingYears' ]

d1=df1.fillna( 11)

s['TotalWorkingYears' ]=d1

**converting float to int**

In [6]:

df=pd.DataFrame(s)

df1=pd.DataFrame(s)

df.NumCompaniesWorked = df .NumCompaniesWorked.astype(int)

df1.TotalWorkingYears=df.TotalWorkingYears.astype( int)

s.head(5)

Out[6]:

**Age Attrition BusinessTravel Department DistanceFromHome Education EducationField EmployeeCount EmployeeID Gender ...**

...

**1** 31 Yes Travel\_Frequently

No Travel\_Frequently

**3** 38 No Non-Travel

Research &

Development

Research &

Development

10 1 Life Sciences 1 2 Female ...

...

2 5 Life Sciences 1 4 Male ...

...

5 rows × 24 columns

**converting categorical string values to integer values**

In [7]:

**from sklearn import** preprocessing

l\_encoder =preprocessing.LabelEncoder() encoded\_g =l\_encoder .fit\_transform(s['Attrition']) s['Attrition' ]=encoded\_g

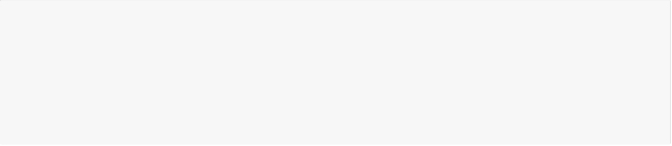
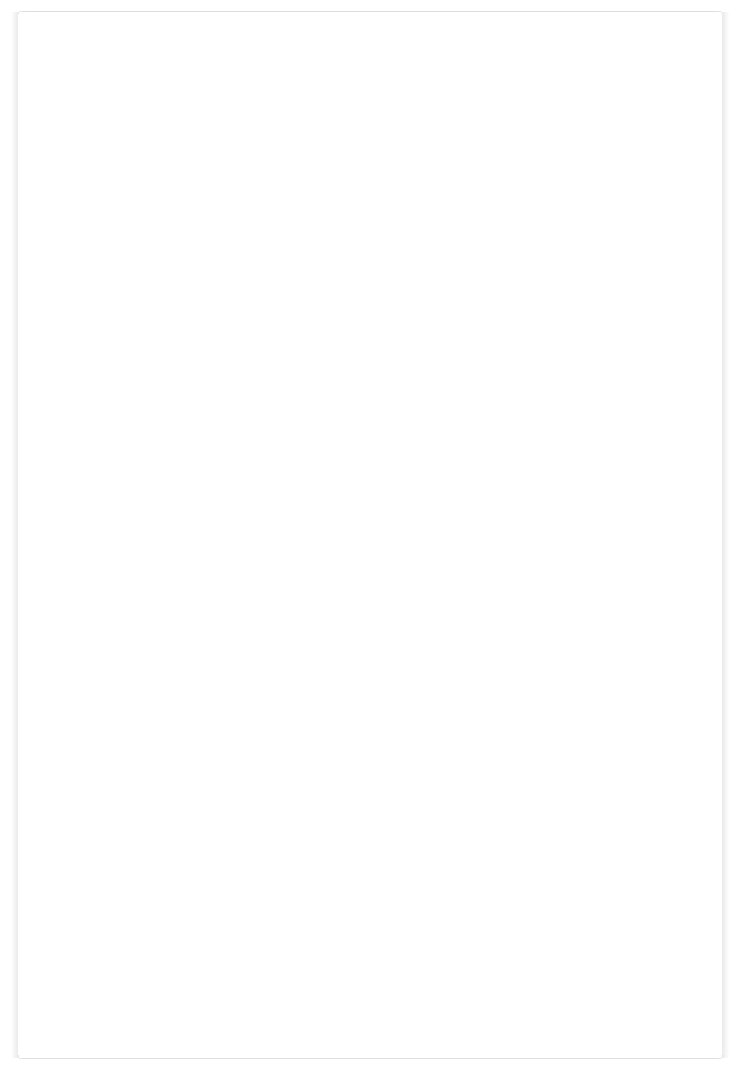
In [8]:

encoded\_g =l\_encoder .fit\_transform(s['JobLevel' ]) s['JobLevel']=encoded\_g

In [9]:

encoded\_g =l\_encoder .fit\_transform(s['JobRole' ]) s['JobRole' ]=encoded\_g

In [10]:



In [10]:

encoded\_g =l\_encoder .fit\_transform(s['MaritalStatus'])

s['MaritalStatus']=encoded\_g

In [11]:

encoded\_g =l\_encoder .fit\_transform(s['Over18'])

s['Over18']=encoded\_g

In [12]:

encoded\_g =l\_encoder .fit\_transform(s['Gender'])

s['Gender']=encoded\_g

In [13]:

encoded\_g =l\_encoder .fit\_transform(s['EducationField' ])

s['EducationField']=encoded\_g

In [14]:

encoded\_g =l\_encoder .fit\_transform(s['Department' ])

s['Department']=encoded\_g

In [15]:

encoded\_g =l\_encoder .fit\_transform(s['BusinessTravel' ])

s['BusinessTravel']=encoded\_g

In [16]:

s.columns

Out[16]:

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')

**Implementing Random Forest**

In [17]:

**from sklearn.ensemble import** RandomForestClassifier

model=RandomForestClassifier(n\_estimators=100, max\_features=2,oob\_score =**True**) features=['Age', 'BusinessTravel', 'Department', 'DistanceFromHome',

'Education' , 'EducationField' , 'EmployeeCount', 'EmployeeID', 'Gender' , 'JobLevel', 'JobRole', 'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked', 'Over18', 'PercentSalaryHike' , 'StandardHours', 'StockOptionLevel', 'TotalWorkingYears' , 'TrainingTimesLastYear', 'YearsAtCompany', 'YearsSinceLastPromotion', 'YearsWithCurrManager']

model.fit(X=s[features],y=s['Attrition' ])

Out[17]:

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,

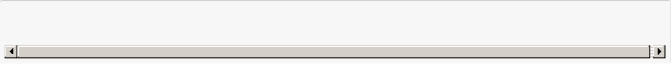
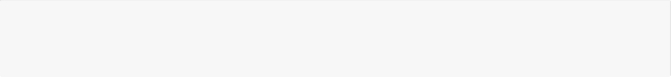
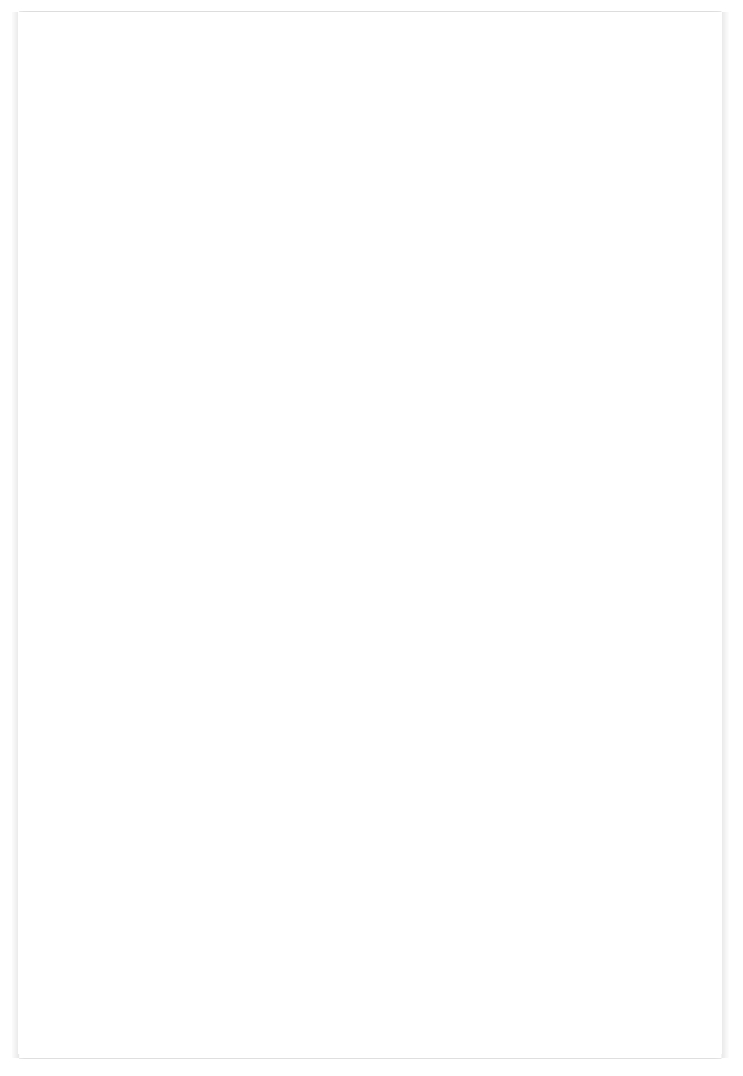
min\_impurity\_decrease=0.0, min\_impurity\_split=None,

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

min\_weight\_fraction\_leaf=0.0, n\_estimators=100,

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

verbose=0, warm\_start=False)



In [18]:

**for** feature, imp **in** zip(features,model.feature\_importances\_):

print(feature,imp)

1. ge 0.09284594778714068
2. usinessTravel 0.027811357387407667

Department 0.025239762939747

1. istanceFromHome 0.06703518158137467
2. ducation 0.03933503092101838

EducationField 0.04028713857731915

EmployeeCount 0.0

EmployeeID 0.038435829425269796

Gender 0.01840932946196571

JobLevel 0.03568245170354088

JobRole 0.053851522400355935

MaritalStatus 0.03951135245995175

1. onthlyIncome 0.08841004491030224
2. umCompaniesWorked 0.054844600675763064
3. ver18 0.0
4. ercentSalaryHike 0.06321516251614118

StandardHours 0.0

1. tockOptionLevel 0.03217938926633844
2. otalWorkingYears 0.07970993498567504

TrainingTimesLastYear 0.043274540980987077

YearsAtCompany 0.0650065237039809

YearsSinceLastPromotion 0.04113916098208512

YearsWithCurrManager 0.05377573733363543

**The attributes TotalWorkingYears,MonthlyIncome and Age have higher accuracy than other attributes. Hence these three attributes will be selected for generating Decision Tree**

**Implementing Decision Tree**

In [19]:

**from sklearn import** tree

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

predictors=pd.DataFrame([s['MonthlyIncome' ],s["TotalWorkingYears" ],s['Age']]).T

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

Out[19]:

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

max\_depth=6, 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')

In [20]:

**with** open ("DT\_Attrition" ,'w') **as** f:

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

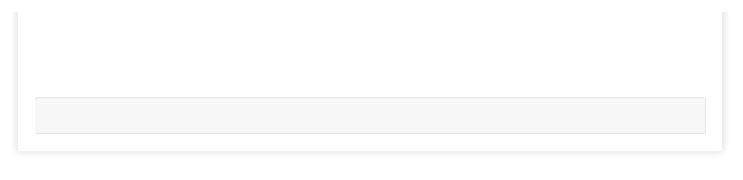
**Model Accuracy**

In [21]:

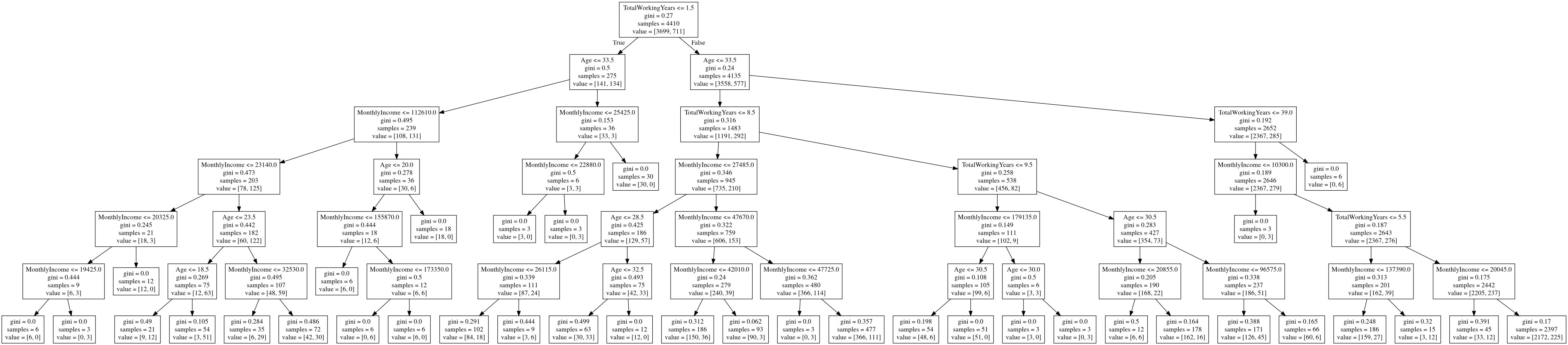
tree\_model.score(X=predictors,y=s['Attrition' ])

Out[21]:

0.8650793650793651



**The model accuracy is 86.5%** In [ ]:



Inferences :

* If TotalWorkingYears <=1.5 and age<=33.5 and MonthlyIncome<=19425 then survived and if MonthlyIncome>19425.0 then no attrition.
* If TotalWorkingYears <=1.5 and age<=33.5 and MonthlyIncome>20325 then attrition
* If TotalWorkingYears <=1.5 and age<=23.5 and MonthlyIncome<=32530 then attrition
* If TotalWorkingYears <=1.5 and age<=18.5 then attrition
* If TotalWorkingYears <=1.5 and MonthlyIncome<=112610 and age>20 then attrition
* If TotalWorkingYears <=1.5 and MonthlyIncome<=155870 then attrition.
* If TotalWorkingYears <=1.5 and age<=20 MonthlyIncome<=173350 then no attrition.
* If TotalWorkingYears <=1.5 and age<=20 and MonthlyIncome>1773350 then attrition is there.
* If TotalWorkingYears >=1.5 and age<=33.5 and TotalWorkingYears<=39 then no attrition
* If TotalWorkingYears >=1.5 and age<=33.5 and TotalWorkingYears<=39 and MonthlyIncome<=10300 then no attrition
* If TotalWorkingYears >=1.5 and age<=33.5 and TotalWorkingYears>5.5 then attrition
* If TotalWorkingYears >=1.5 and age<=33.5 and TotalWorkingYears<=5.5 and MonthlyIncome<=137390 then attrition
* If age<=33.5 and TotalWorkingYears<=39 and MonthlyIncome<=10300 then no attrition