1.Download the Employee Attrition Dataset https://www.kaggle.com/datasets/patelprashant/employee-attrition

2.Perfrom Data Preprocessing

3.Model Building using Logistic Regression and Decision Tree and Random Forest

4.Calculate Performance metrics

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df=pd.read_csv('/content/WA_Fn-UseC_-HR-Employee-Attrition.csv')
df
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educatic
0	41	Yes	Travel_Rarely	1102	Sales	1	
1	49	No	Travel_Frequently	279	Research & Development	8	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	
4	27	No	Travel_Rarely	591	Research & Development	2	
...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	
1466	39	No	Travel_Rarely	613	Research & Development	6	
1467	27	No	Travel_Rarely	155	Research & Development	4	
1468	49	No	Travel_Frequently	1023	Sales	2	
1469	34	No	Travel_Rarely	628	Research & Development	8	

1470 rows × 35 columns

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	..
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	.
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	.
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	.
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	.
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	.

5 rows × 35 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
```

```
 1   Attrition                1470 non-null   object
 2   BusinessTravel           1470 non-null   object
 3   DailyRate                1470 non-null   int64
 4   Department               1470 non-null   object
 5   DistanceFromHome         1470 non-null   int64
 6   Education                1470 non-null   int64
 7   EducationField           1470 non-null   object
 8   EmployeeCount            1470 non-null   int64
 9   EmployeeNumber           1470 non-null   int64
 10  EnvironmentSatisfaction  1470 non-null   int64
 11  Gender                   1470 non-null   object
 12  HourlyRate               1470 non-null   int64
 13  JobInvolvement           1470 non-null   int64
 14  JobLevel                 1470 non-null   int64
 15  JobRole                  1470 non-null   object
 16  JobSatisfaction          1470 non-null   int64
 17  MaritalStatus            1470 non-null   object
 18  MonthlyIncome            1470 non-null   int64
 19  MonthlyRate              1470 non-null   int64
 20  NumCompaniesWorked       1470 non-null   int64
 21  Over18                   1470 non-null   object
 22  OverTime                 1470 non-null   object
 23  PercentSalaryHike        1470 non-null   int64
 24  PerformanceRating        1470 non-null   int64
 25  RelationshipSatisfaction 1470 non-null   int64
 26  StandardHours            1470 non-null   int64
 27  StockOptionLevel         1470 non-null   int64
 28  TotalWorkingYears        1470 non-null   int64
 29  TrainingTimesLastYear    1470 non-null   int64
 30  WorkLifeBalance          1470 non-null   int64
 31  YearsAtCompany           1470 non-null   int64
 32  YearsInCurrentRole       1470 non-null   int64
 33  YearsSinceLastPromotion  1470 non-null   int64
 34  YearsWithCurrManager     1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

```
df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInv
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	14
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	

8 rows × 26 columns

```
df.shape
```

```
(1470, 35)
```

```
#Checking for Null Values
df.isnull().any()
```

```
Age                        False
Attrition                  False
BusinessTravel             False
DailyRate                  False
Department                 False
DistanceFromHome           False
Education                  False
EducationField             False
EmployeeCount              False
EmployeeNumber             False
EnvironmentSatisfaction    False
Gender                     False
HourlyRate                 False
JobInvolvement             False
JobLevel                   False
```

```
JobRole                   False
JobSatisfaction           False
MaritalStatus             False
MonthlyIncome             False
MonthlyRate               False
NumCompaniesWorked        False
Over18                    False
OverTime                  False
PercentSalaryHike         False
PerformanceRating         False
RelationshipSatisfaction  False
StandardHours             False
StockOptionLevel          False
TotalWorkingYears         False
TrainingTimesLastYear     False
WorkLifeBalance           False
YearsAtCompany            False
YearsInCurrentRole        False
YearsSinceLastPromotion   False
YearsWithCurrManager      False
dtype: bool
```

```
df.isnull().sum()
```

```
Age                       0
Attrition                 0
BusinessTravel            0
DailyRate                 0
Department                0
DistanceFromHome          0
Education                 0
EducationField            0
EmployeeCount             0
EmployeeNumber            0
EnvironmentSatisfaction   0
Gender                    0
HourlyRate                0
JobInvolvement            0
JobLevel                  0
JobRole                   0
JobSatisfaction           0
MaritalStatus             0
MonthlyIncome             0
MonthlyRate               0
NumCompaniesWorked        0
Over18                    0
OverTime                  0
PercentSalaryHike         0
PerformanceRating         0
RelationshipSatisfaction  0
StandardHours             0
StockOptionLevel          0
TotalWorkingYears         0
TrainingTimesLastYear     0
WorkLifeBalance           0
YearsAtCompany            0
YearsInCurrentRole        0
YearsSinceLastPromotion   0
YearsWithCurrManager      0
dtype: int64
```

```
sns.distplot(df['Age'])
```

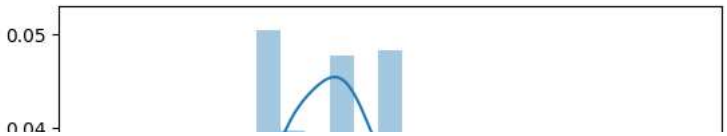
```
<ipython-input-494-0fafe04ea3f6>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Age'])
<Axes: xlabel='Age', ylabel='Density'>
```




```
df.corr()
```

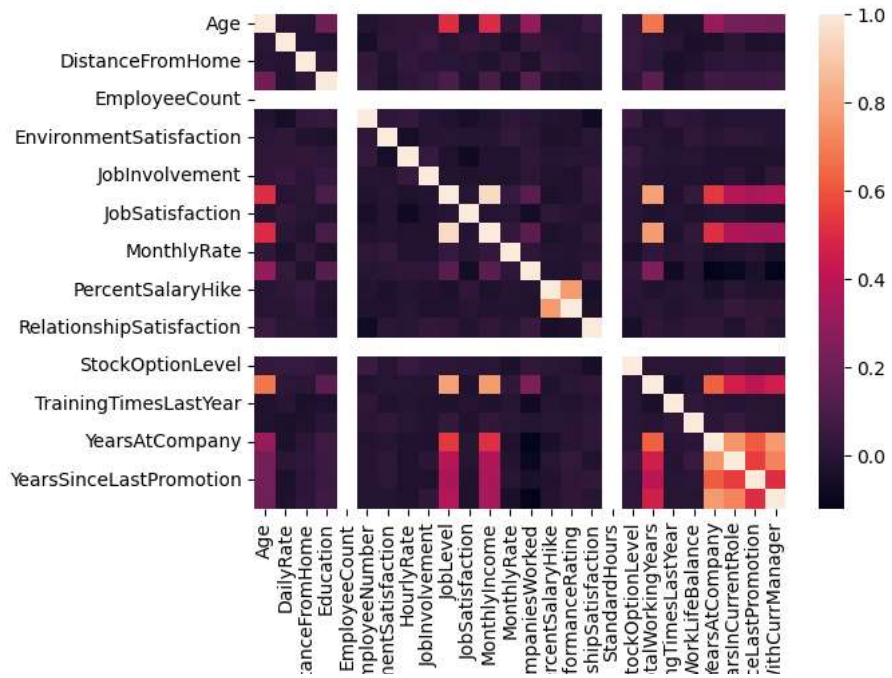
```
<ipython-input-495-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versio
  df.corr()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Hourly
Age	1.000000	0.010661	-0.001686	0.208034	NaN	-0.010145	0.010146	0.02
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.018355	0.02
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN	0.032916	-0.016075	0.03
Education	0.208034	-0.016806	0.021042	1.000000	NaN	0.042070	-0.027128	0.01
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN	1.000000	0.017621	0.03
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN	0.017621	1.000000	-0.04
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN	0.035179	-0.049857	1.00
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN	-0.006888	-0.008278	0.04
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN	-0.018519	0.001212	-0.02
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.006784	-0.07
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.006259	-0.01
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN	0.012648	0.037600	-0.01
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN	-0.001251	0.012594	0.02
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.031701	-0.00
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	NaN	-0.020359	-0.029548	-0.00
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	NaN	-0.069861	0.007665	0.00
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	NaN	0.062227	0.003432	0.05
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	NaN	-0.014365	-0.002693	-0.00
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	NaN	0.023603	-0.019359	-0.00
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	NaN	0.010309	0.027627	-0.00
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	NaN	-0.011240	0.001458	-0.01
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	NaN	-0.008416	0.018007	-0.02
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	NaN	-0.009019	0.016194	-0.02
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	NaN	-0.009197	-0.004999	-0.02

26 rows × 26 columns

```
sns.heatmap(df.corr())
```

```
<ipython-input-496-aa4f4450a243>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versio
  sns.heatmap(df.corr())
<Axes: >
```



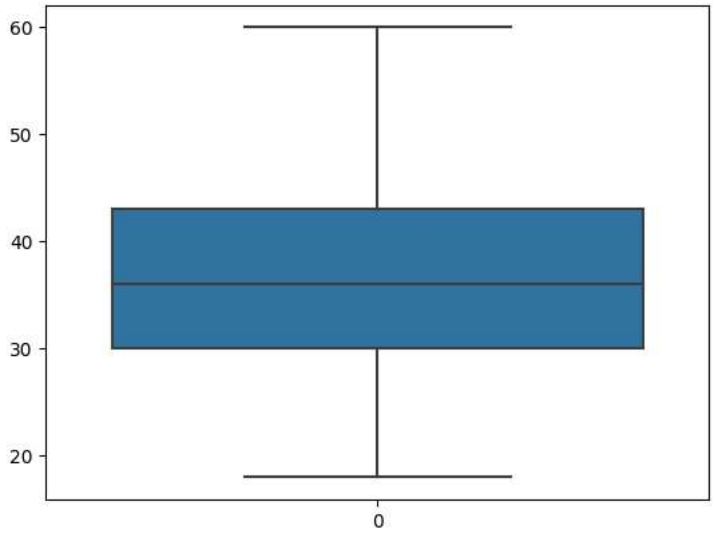

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	..
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	.
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	.
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	.
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	.
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	.

5 rows × 35 columns

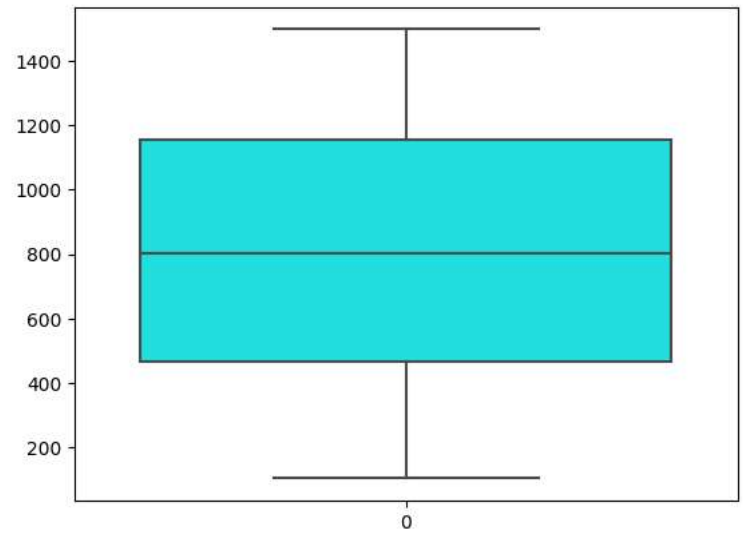
```
sns.boxplot(df['Age'])
```

```
<Axes: >
```



```
sns.boxplot(df['DailyRate'],color='cyan')
```

```
<Axes: >
```




```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	..
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	.
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	.
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	.
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	.
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	.

5 rows × 35 columns

```
#Splitting of dependent and independent variables
x=df.iloc[:,[i for i in range(df.shape[1]) if i!=1]]
x.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSa
0	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	
1	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	
2	37	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	
4	27	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	

5 rows × 34 columns

```
y=df.iloc[:,1:2]
y.head()
```

	Attrition
0	Yes
1	No

```
print(x.shape)
print(y.shape)
```

```
(1470, 34)
(1470, 1)
```

```
#LabelEncoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
a=['Age','Department','EducationField','BusinessTravel','Gender','JobRole','MaritalStatus','Over18','OverTime']
for i in a:
  x[i]=le.fit_transform(x[i])
x[a]
```

```
<ipython-input-504-8cae39b6351b>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  x[i]=le.fit_transform(x[i])
<ipython-input-504-8cae39b6351b>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  x[i]=le.fit_transform(x[i])
<ipython-input-504-8cae39b6351b>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
#x=x.drop('Over18',axis=1)
#x=x.drop('OverTime',axis=1)
```

```
<ipython-input-504-8cae39b6351b>:6: SettingWithCopyWarning:
```

```
y['Attrition']=le.fit_transform(y['Attrition'])
```

```
y.head()
```

	Attrition
0	1
1	0
2	1
3	0
4	0

```
<ipython-input-504-8cae39b6351b>:6: SettingWithCopyWarning:
```

```
#Feature Scaling
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
x_scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
```

```
<ipython-input-504-8cae39b6351b>:6: SettingWithCopyWarning:
```

```
#Splitting data into train and test data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.3)
```

```
  x[i]=le.fit_transform(x[i])
```

```
#Model Building- LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
pred=model.predict(x_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d
  y = column_or_1d(y, warn=True)
```

```
pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0])
```

```
#Evaluation of Model- LOGISTIC REGRESSION
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
accuracy_score(y_test,pred)
```

```
0.8458049886621315
```

```
confusion_matrix(y_test,pred)
```

```
array([[355,  13],
       [ 55,  18]])
```

```
print(y_test.shape)
print(pred.shape)
```

```
(441, 1)
(441,)
```

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.96      0.91       368
           1       0.58      0.25      0.35        73

    accuracy                           0.85       441
   macro avg       0.72      0.61      0.63       441
weighted avg       0.82      0.85      0.82       441
```

```
#Model Building- Decision Tree Clasifier
from sklearn.tree import DecisionTreeClassifier
model1 = DecisionTreeClassifier(max_depth=4,splitter='best',criterion='entropy')
model1.fit(x_train,y_train)
d_y_predict = model1.predict(x_test)
```

```
d_y_predict
```

```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0])
```

```
#Evaluating Metrics- Decision Tree
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
print('Testing Accuracy = ', accuracy_score(y_test,d_y_predict))
```

```
Testing Accuracy =  0.8344671201814059
```

```
confusion_matrix(y_test,d_y_predict)
```

```
array([[343,  25],
       [ 48,  25]])
```

```
print(classification_report(y_test,d_y_predict))
```

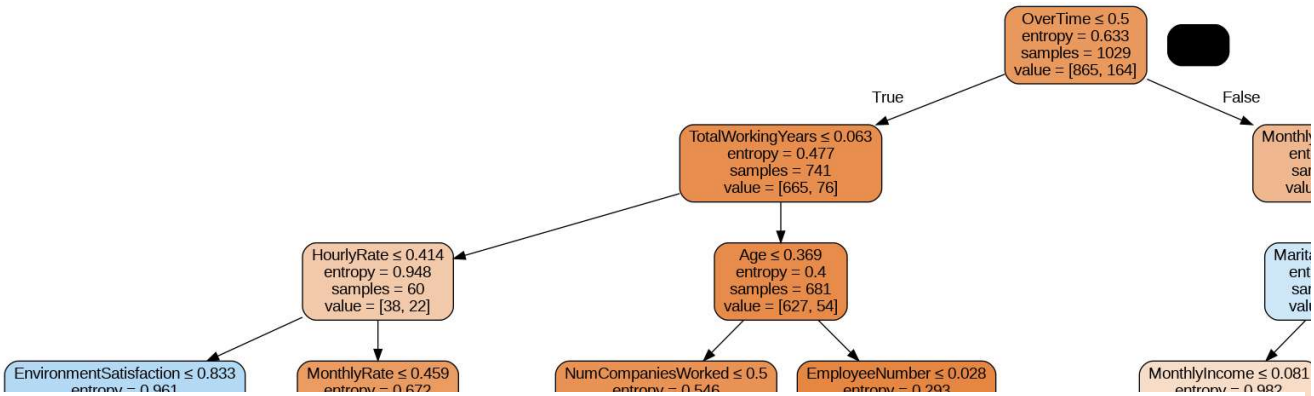
```
                   precision    recall  f1-score   support

                0       0.88      0.93      0.90       368
                1       0.50      0.34      0.41        73

         accuracy                           0.83       441
        macro avg       0.69      0.64      0.66       441
     weighted avg       0.81      0.83      0.82       441
```

```
from six import StringIO
from IPython.display import Image
import pydotplus
from sklearn.tree import export_graphviz
```

```
dot_data =StringIO()
export_graphviz(model1,out_file=dot_data,feature_names= x.columns,
                filled=True,rounded= True,special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```




```
#Model Building- Random Forest Classifier(Ensemble Model)
from sklearn.ensemble import RandomForestClassifier
model2 =RandomForestClassifier(criterion='entropy')
model2.fit(x_train,y_train)
```

```
<ipython-input-522-4b132850df9e>:4: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the
  model2.fit(x_train,y_train)
```

```
RandomForestClassifier(criterion='entropy')
```

```
r_y_predict = model2.predict(x_test)
r_y_predict_train = model2.predict(x_train)
```

```
#Evaluation Metrics- Random Forest
print('Testing Accuracy = ', accuracy_score(y_test,r_y_predict))
print('Training Accuracy = ', accuracy_score(y_train,r_y_predict_train))
```

```
Testing Accuracy =  0.8458049886621315
Training Accuracy =  1.0
```

```
print(classification_report(y_test,r_y_predict))
```

```
                   precision    recall  f1-score   support

                0       0.85      0.99      0.91       368
                1       0.73      0.11      0.19        73

         accuracy                           0.85       441
        macro avg       0.79      0.55      0.55       441
     weighted avg       0.83      0.85      0.79       441
```

```
confusion_matrix(y_test,r_y_predict)
```

```
array([[365,   3],
       [ 65,   8]])
```