Lab9.Employees Hopping prediction using Random Forests

NAME : BERCHMANS KEVIN S

ROLL NO :215229107

STEP -1 UNDERSTAND DATA

In [44]:
```
import pandas as pd
```

In [45]:
```
df = pd.read_csv('Employee_hopping.csv')
```

Properties

In [46]:
```
df.head()
```

Out[46]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educati
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life S
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life S
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life S
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

In [47]:
```
df.shape
```

Out[47]: (1470, 35)

In [48]:
```
df.columns
```

Out[48]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [49]:
```
df.dtypes
```

Out[49]:
```
Age                          int64
Attrition                   object
BusinessTravel              object
DailyRate                    int64
Department                  object
DistanceFromHome             int64
Education                    int64
EducationField              object
EmployeeCount                int64
EmployeeNumber               int64
EnvironmentSatisfaction      int64
Gender                      object
HourlyRate                   int64
JobInvolvement               int64
JobLevel                     int64
JobRole                     object
JobSatisfaction              int64
MaritalStatus               object
MonthlyIncome                int64
MonthlyRate                  int64
NumCompaniesWorked           int64
Over18                      object
OverTime                    object
PercentSalaryHike            int64
PerformanceRating            int64
RelationshipSatisfaction     int64
StandardHours                int64
StockOptionLevel             int64
TotalWorkingYears            int64
TrainingTimesLastYear        int64
WorkLifeBalance              int64
YearsAtCompany               int64
YearsInCurrentRole           int64
YearsSinceLastPromotion      int64
YearsWithCurrManager         int64
dtype: object
```

In [50]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
Age                         1470 non-null int64
Attrition                   1470 non-null object
BusinessTravel              1470 non-null object
DailyRate                   1470 non-null int64
Department                  1470 non-null object
DistanceFromHome            1470 non-null int64
Education                   1470 non-null int64
EducationField              1470 non-null object
EmployeeCount               1470 non-null int64
EmployeeNumber              1470 non-null int64
EnvironmentSatisfaction     1470 non-null int64
Gender                      1470 non-null object
HourlyRate                  1470 non-null int64
JobInvolvement              1470 non-null int64
JobLevel                    1470 non-null int64
JobRole                     1470 non-null object
JobSatisfaction             1470 non-null int64
MaritalStatus               1470 non-null object
MonthlyIncome               1470 non-null int64
MonthlyRate                 1470 non-null int64
NumCompaniesWorked          1470 non-null int64
Over18                      1470 non-null object
OverTime                    1470 non-null object
PercentSalaryHike           1470 non-null int64
PerformanceRating           1470 non-null int64
RelationshipSatisfaction    1470 non-null int64
StandardHours               1470 non-null int64
StockOptionLevel            1470 non-null int64
TotalWorkingYears           1470 non-null int64
TrainingTimesLastYear       1470 non-null int64
WorkLifeBalance             1470 non-null int64
YearsAtCompany              1470 non-null int64
YearsInCurrentRole          1470 non-null int64
YearsSinceLastPromotion     1470 non-null int64
YearsWithCurrManager        1470 non-null int64
dtypes: int64(26), object(9)
memory usage: 402.0+ KB
```

In [51]:
```
df['JobRole'].value_counts()
```

Out[51]:
```
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64
```

STEP -2 EXTRACT X AND Y

In [52]:
```
X = df.drop(['Attrition'],axis=1)
y = df['Attrition']
```

In [53]:
```
y = y.apply(lambda x:1 if x == 'Yes' else 0)
```

In [54]:
```
X
```

5	32	Travel_Frequently	1005	Research & Development	2	2	Life Scienc
6	59	Travel_Rarely	1324	Research & Development	3	3	Medi
7	30	Travel_Rarely	1358	Research & Development	24	1	Life Scienc
8	38	Travel_Frequently	216	Research & Development	23	3	Life Scienc
9	36	Travel_Rarely	1299	Research & Development	27	3	Medi
10	35	Travel_Rarely	809	Research & Development	16	3	Medi
11	29	Travel_Rarely	153	Research & Development	15	2	Life Scienc
12	31	Travel_Rarely	670	Research & Development	26	1	Life Scienc

In [55]:
```
y
```

Out[55]:
```
0       1
1       0
2       1
3       0
4       0
5       0
6       0
7       0
8       0
9       0
10      0
11      0
12      0
13      0
14      1
15      0
16      0
17      0
18      0
19      0
20      0
21      1
22      0
23      0
24      1
25      0
26      1
27      0
28      0
29      0
       ..
1440    0
1441    0
1442    1
1443    0
1444    1
1445    0
1446    0
1447    0
1448    0
1449    0
1450    0
1451    0
1452    1
1453    0
1454    0
1455    0
1456    0
1457    0
1458    0
1459    0
1460    0
1461    1
1462    0
1463    0
```

```
1464    0
1465    0
1466    0
1467    0
1468    0
1469    0
Name: Attrition, Length: 1470, dtype: int64
```

STEP - 3 FEATURE ENGINEERING

In [56]:
```
df = pd.get_dummies(df,columns=['BusinessTravel','Department','EducationField','G
```

In [57]:
```
df
```

Out[57]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber
0	41	Yes	1102	1	2	1	1
1	49	No	279	8	1	1	2
2	37	Yes	1373	2	2	1	4
3	33	No	1392	3	4	1	5
4	27	No	591	2	1	1	7
5	32	No	1005	2	2	1	8
6	59	No	1324	3	3	1	10
7	30	No	1358	24	1	1	11
8	38	No	216	23	3	1	12
9	36	No	1299	27	3	1	13
10	35	No	809	16	3	1	14
11	29	No	153	15	2	1	15
12	31	No	670	26	1	1	16
13	34	No	1346	19	2	1	18
14	28	Yes	103	24	3	1	19
15	29	No	1389	21	4	1	20
16	32	No	334	5	2	1	21
17	22	No	1123	16	2	1	22
18	53	No	1219	2	4	1	23
19	38	No	371	2	3	1	24
20	24	No	673	11	2	1	26
21	36	Yes	1218	9	4	1	27
22	34	No	419	7	4	1	28
23	21	No	391	15	2	1	30
24	34	Yes	699	6	1	1	31
25	53	No	1282	5	3	1	32
26	32	Yes	1125	16	1	1	33
27	42	No	691	8	4	1	35
28	44	No	477	7	4	1	36
29	46	No	705	2	4	1	38
...	...	...	...	...	...	...	...
1440	36	No	688	4	2	1	2025
1441	56	No	667	1	4	1	2026
1442	29	Yes	1092	1	4	1	2027

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber
1443	42	No	300	2	3	1	2031
1444	56	Yes	310	7	2	1	2032
1445	41	No	582	28	4	1	2034
1446	34	No	704	28	3	1	2035
1447	36	No	301	15	4	1	2036
1448	41	No	930	3	3	1	2037
1449	32	No	529	2	3	1	2038
1450	35	No	1146	26	4	1	2040
1451	38	No	345	10	2	1	2041
1452	50	Yes	878	1	4	1	2044
1453	36	No	1120	11	4	1	2045
1454	45	No	374	20	3	1	2046
1455	40	No	1322	2	4	1	2048
1456	35	No	1199	18	4	1	2049
1457	40	No	1194	2	4	1	2051
1458	35	No	287	1	4	1	2052
1459	29	No	1378	13	2	1	2053
1460	29	No	468	28	4	1	2054
1461	50	Yes	410	28	3	1	2055
1462	39	No	722	24	1	1	2056
1463	31	No	325	5	3	1	2057
1464	26	No	1167	5	3	1	2060
1465	36	No	884	23	2	1	2061
1466	39	No	613	6	1	1	2062
1467	27	No	155	4	3	1	2064
1468	49	No	1023	2	3	1	2065
1469	34	No	628	8	3	1	2068

1470 rows × 56 columns

STEP - 4 CHECK SHAPE OF X AND Y

In [58]:
```
X = df.drop(['Attrition'],axis=1)
print('X Shape : ',X.shape)
print('y Shape : ',y.shape)
```

```
X Shape :  (1470, 55)
y Shape :  (1470,)
```

STEP- 5:MODEL DEVELOPMENT

In [59]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [60]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size =0.2, random_s
```

In [61]:
```
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier(n_estimators=100, max_features=0.3)
```

In [62]:
```
RFC.fit(X_train,y_train)
```

Out[62]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features=0.3,
                       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=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [63]:
```
RFC_y_pred = RFC.predict(X_test)
RFC_y_pred
```

Out[63]:
```
array([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, 1, 0, 0, 1, 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, 0, 1, 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, 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, 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], dtype=int64)
```

STEP- 6 TESTING

In [64]:
```
from sklearn.metrics import accuracy_score,classification_report
```

In [65]:
```
RFC_acc = accuracy_score(y_test,RFC_y_pred)
RFC_acc
```

Out[65]:
```
0.8741496598639455
```

In [66]:
```
print(classification_report(y_test, RFC_y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       255
           1       0.62      0.13      0.21        39

    accuracy                           0.87       294
   macro avg       0.75      0.56      0.57       294
weighted avg       0.85      0.87      0.84       294
```

STEP- 7 FEATURE IMPORTANT VALUE

In [67]:
```
print(RFC.feature_importances_)
```

```
[0.052496   0.04945766 0.04068529 0.014859   0.         0.04587253
 0.02412126 0.03948358 0.0191574  0.02457012 0.02323603 0.08830025
 0.04121782 0.03442585 0.02505677 0.00331897 0.01726355 0.
 0.02956323 0.04319382 0.02189407 0.02172904 0.04013969 0.02237757
 0.02554315 0.02439125 0.00262102 0.01177421 0.00508338 0.00220734
 0.00505255 0.00913669 0.00257177 0.00494035 0.00516521 0.00583274
 0.00305116 0.00760378 0.00481866 0.00521577 0.00177857 0.00213611
 0.00770178 0.0006249  0.00223041 0.00133642 0.00567526 0.00600314
 0.00938339 0.00459042 0.00492129 0.02293532 0.         0.04226756
 0.04098691]
```

In [68]:
```
feature_name = pd.DataFrame(RFC.feature_importances_, index=X_train.columns, colu
feature_name
```

Out[68]:

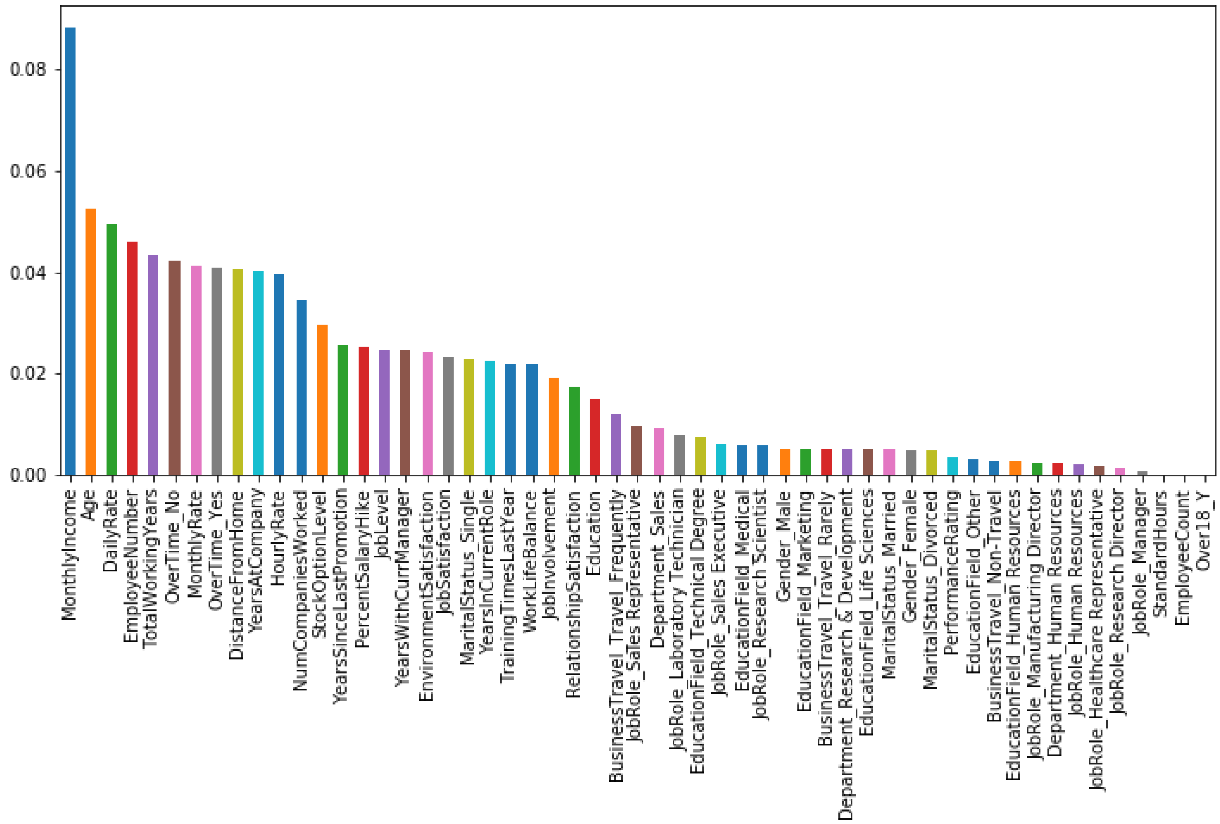
	Important_Feature
Age	0.052496
DailyRate	0.049458
DistanceFromHome	0.040685
Education	0.014859
EmployeeCount	0.000000
EmployeeNumber	0.045873
EnvironmentSatisfaction	0.024121
HourlyRate	0.039484
JobInvolvement	0.019157
JobLevel	0.024570
JobSatisfaction	0.023236
MonthlyIncome	0.088300
MonthlyRate	0.041218
NumCompaniesWorked	0.034426
PercentSalaryHike	0.025057
PerformanceRating	0.003319
RelationshipSatisfaction	0.017264
StandardHours	0.000000
StockOptionLevel	0.029563
TotalWorkingYears	0.043194
TrainingTimesLastYear	0.021894
WorkLifeBalance	0.021729
YearsAtCompany	0.040140
YearsInCurrentRole	0.022378
YearsSinceLastPromotion	0.025543
YearsWithCurrManager	0.024391
BusinessTravel_Non-Travel	0.002621
BusinessTravel_Travel_Frequently	0.011774
BusinessTravel_Travel_Rarely	0.005083
Department_Human Resources	0.002207
Department_Research & Development	0.005053
Department_Sales	0.009137
EducationField_Human Resources	0.002572

	Important_Feature
EducationField_Life Sciences	0.004940
EducationField_Marketing	0.005165
EducationField_Medical	0.005833
EducationField_Other	0.003051
EducationField_Technical Degree	0.007604
Gender_Female	0.004819
Gender_Male	0.005216
JobRole_Healthcare Representative	0.001779
JobRole_Human Resources	0.002136
JobRole_Laboratory Technician	0.007702
JobRole_Manager	0.000625
JobRole_Manufacturing Director	0.002230
JobRole_Research Director	0.001336
JobRole_Research Scientist	0.005675
JobRole_Sales Executive	0.006003
JobRole_Sales Representative	0.009383
MaritalStatus_Divorced	0.004590
MaritalStatus_Married	0.004921
MaritalStatus_Single	0.022935
Over18_Y	0.000000
OverTime_No	0.042268
OverTime_Yes	0.040987

In [69]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [70]:
```
pd.Series(RFC.feature_importances_, index=X_train.columns).sort_values(ascending=
```

Out[70]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x174175b3e80>
```



STEP- 8 Visualize your RF Decision Tree using graphviz

In [71]:
```
estimator = RFC.estimators_[5]
```

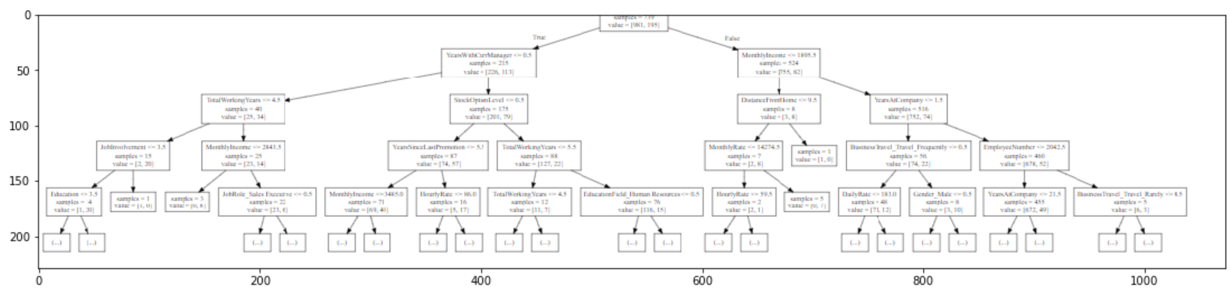
In [72]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
with open("RFDT.dot", 'w') as f:
    f = tree.export_graphviz(estimator, out_file=f, max_depth=4, impurity=False, 
```

In [73]:
```
!dot - Tpng RFDT.dot -o RFDT.png
```

```
'dot' is not recognized as an internal or external command,
operable program or batch file.
```

In [74]:
```
import matplotlib.pyplot as plt
image = plt.imread('RFDT.png')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

Out[74]: <matplotlib.image.AxesImage at 0x174178d06d8>



STEP- 9:RF WITH A RANGE OF TREES

In [75]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [76]:
```
rf2 = RandomForestClassifier(oob_score=True, random_state=42, warm_start=True, n_
oob_list = list()
for n_trees in [15, 20, 30, 40, 50, 100, 150, 200, 300, 400]:
    rf2.set_params(n_estimators=n_trees)
    rf2.fit(X_train, y_train)
    oob_error = 1 - rf2.oob_score_
    oob_list.append(pd.Series({'n_trees': n_trees, 'oob': oob_error}))

rf_oob_df = pd.concat(oob_list, axis=1).T.set_index('n_trees')
rf_oob_df
```

Out[76]:

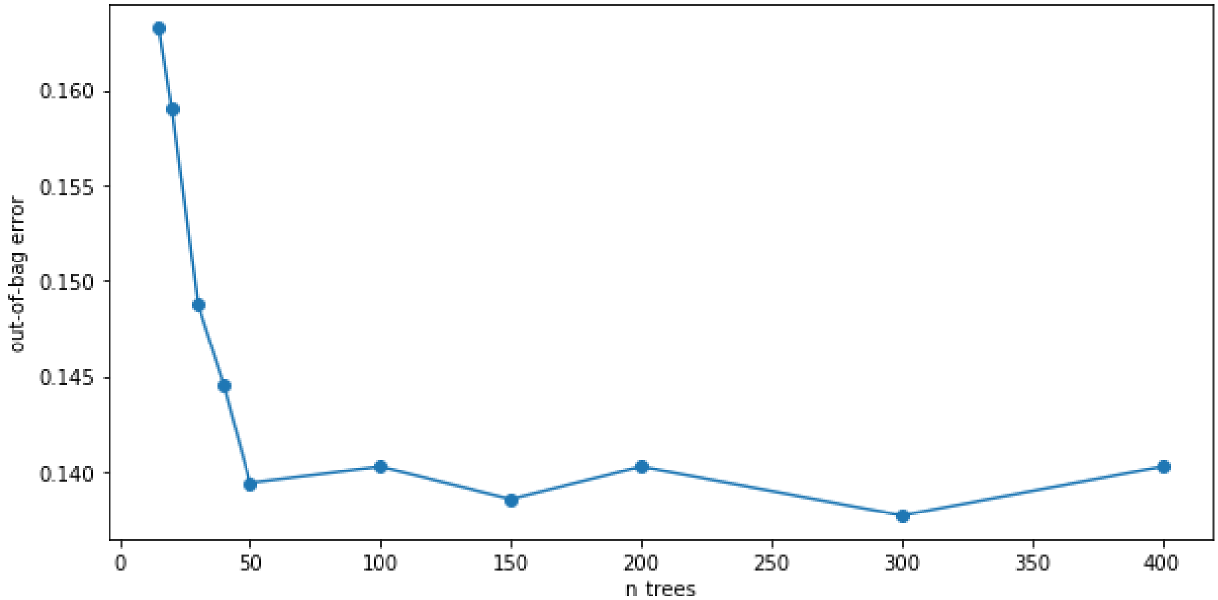
n_trees	oob
15.0	0.163265
20.0	0.159014
30.0	0.148810
40.0	0.144558
50.0	0.139456
100.0	0.140306
150.0	0.138605
200.0	0.140306
300.0	0.137755
400.0	0.140306

STEP- 10 PLOT OOB -ERROR FOR EACH TREE

The following lines will help you

In [77]:
```
ax = rf_oob_df.plot(legend=False, marker='o', figsize=(10,5))
ax.set(ylabel='out-of-bag error')
```

Out[77]:
```
[Text(0, 0.5, 'out-of-bag error')]
```



STEP- 11 COMPARE WITH DECISION TREE CLASSIFIER

Create DecisionTreeClassifier, fit and predict on test set

Visualize the tree using graphviz

Print accuracy score

Print classification report

What is the result of the comparision between RF and DT models? Which gives best accuracy?.

What is your comment on precision, recall, f1 score values?

In [78]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report
clf = DecisionTreeClassifier(max_depth=4, random_state=42)
clf.fit(X_test,y_test)
```

Out[78]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=4, 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=42, splitter='best')
```

In [79]:
```
y_pred1 = clf.predict(X_test)
y_pred1
```

Out[79]:
```
array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 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, 1, 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, 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, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 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, 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, 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, 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], dtype=int64)
```

In [80]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
with open("DTC2.dot", 'w') as f:
    f = tree.export_graphviz(clf,out_file=f,max_depth = 4,impurity = False,featur
```

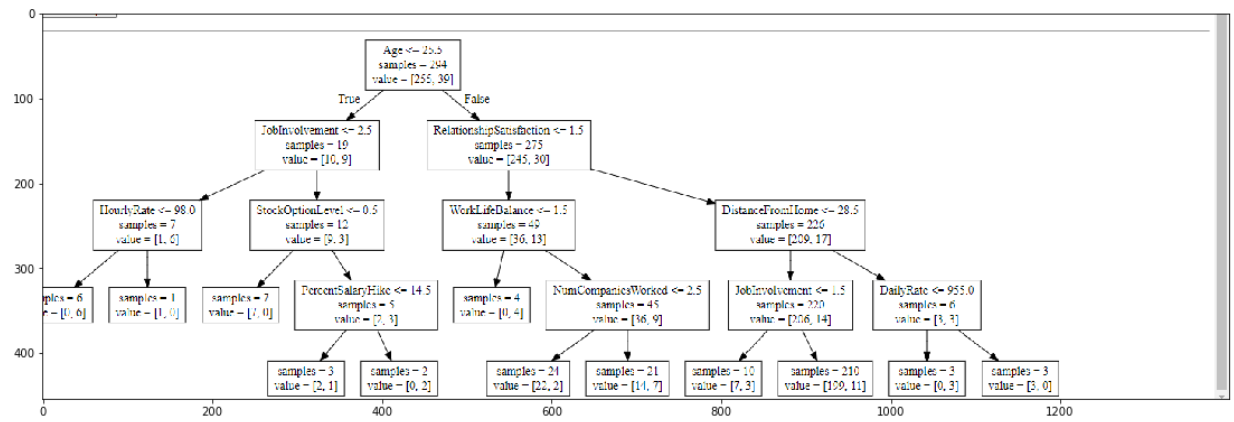
Now open treedtc.dot file which will be created in your working directory then Copy and paste the code to [http://webgraphviz.com/ (http://webgraphviz.com/)](http://webgraphviz.com/)

In [81]:
```
!dot -Tpng DTC2.dot -o DTC2.png
```

```
'dot' is not recognized as an internal or external command,
operable program or batch file.
```

In [82]:
```python
image = plt.imread('DTC2.png')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

Out[82]: <matplotlib.image.AxesImage at 0x174191b8320>



In [83]:
```python
print("Accuracy of test :",clf.score(X_test,y_test))
```

```
Accuracy of test : 0.9183673469387755
```

In [84]:
```python
print(classification_report(y_test,RFC_y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       255
           1       0.62      0.13      0.21        39

    accuracy                           0.87       294
   macro avg       0.75      0.56      0.57       294
weighted avg       0.85      0.87      0.84       294
```

In [85]:
```python
from sklearn.metrics import precision_score, recall_score, accuracy_score, roc_au
```

In [86]:
```
print("RF model :",accuracy_score(y_test,RFC_y_pred))
print("RF Precision:",precision_score(y_test,RFC_y_pred))
print("RF Recall :",recall_score(y_test,RFC_y_pred))
print("RF F1 score :",f1_score(y_test,RFC_y_pred))
print("\n")
print("DT model :",accuracy_score(y_test,y_pred1))
print("DT Precision:",precision_score(y_test,y_pred1))
print("DT Recall :",recall_score(y_test,y_pred1))
print("DT F1 score :",f1_score(y_test,y_pred1))
```

```
RF model : 0.8741496598639455
RF Precision: 0.625
RF Recall : 0.1282051282051282
RF F1 score : 0.21276595744680848


DT model : 0.9183673469387755
DT Precision: 1.0
DT Recall : 0.38461538461538464
DT F1 score : 0.5555555555555556
```