Name:P.Asha Belcilda

Roll no:225229104

Lab9.Employee Hopping Prediction using Random Forests

Step1.[Understand Data]

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

In [45]:
```
data=pd.read_csv("Employee_Hopping.csv")
data
```

ırdHours	StockOptionLevel	TotalWorkingYears	TrainingTimesLastYear	WorkLifeBalance	YearsAtCompany	YearsInCurrentRole	YearsSinceLastPromotion	YearsWithCurrManager
80	0	8	0	1	6	4	0	5
80	1	10	3	3	10	7	1	7
80	0	7	3	3	0	0	0	0
80	0	8	3	3	8	7	3	0
80	1	6	3	3	2	2	2	2
80	0	8	2	2	7	7	3	6
80	3	12	3	2	1	0	0	0

In [46]:
```
#head
data.head()
```

Out[46]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSa
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

In [47]:
```
#shape
data.shape
```

Out[47]:
```
(1470, 35)
```

In [48]:
```
#columns
data.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]:
```
#dtype
data.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]:
```
#info
data.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]:
```
#value_counts
data['BusinessTravel'].value_counts
```

Out[51]:
```
<bound method IndexOpsMixin.value_counts of 0           Travel_Rarely
1       Travel_Frequently
2           Travel_Rarely
3       Travel_Frequently
4           Travel_Rarely
5       Travel_Frequently
6           Travel_Rarely
7           Travel_Rarely
8       Travel_Frequently
9           Travel_Rarely
10          Travel_Rarely
11          Travel_Rarely
12          Travel_Rarely
13          Travel_Rarely
14          Travel_Rarely
15          Travel_Rarely
16          Travel_Rarely
17             Non-Travel
18          Travel_Rarely
19          Travel_Rarely
20             Non-Travel
21          Travel_Rarely
22          Travel_Rarely
23          Travel_Rarely
24          Travel_Rarely
25          Travel_Rarely
26      Travel_Frequently
27          Travel_Rarely
28          Travel_Rarely
29          Travel_Rarely
              ...
1440    Travel_Frequently
1441           Non-Travel
1442        Travel_Rarely
1443        Travel_Rarely
1444        Travel_Rarely
1445        Travel_Rarely
1446        Travel_Rarely
1447           Non-Travel
1448        Travel_Rarely
1449        Travel_Rarely
1450        Travel_Rarely
1451        Travel_Rarely
1452    Travel_Frequently
1453        Travel_Rarely
1454        Travel_Rarely
1455        Travel_Rarely
1456    Travel_Frequently
1457        Travel_Rarely
1458        Travel_Rarely
1459        Travel_Rarely
1460        Travel_Rarely
1461        Travel_Rarely
1462        Travel_Rarely
1463           Non-Travel
1464        Travel_Rarely
1465    Travel_Frequently
1466        Travel_Rarely
1467        Travel_Rarely
1468    Travel_Frequently
1469        Travel_Rarely
Name: BusinessTravel, Length: 1470, dtype: object>
```

Step2.[Extract X and y]

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

80	3	12	3	2	1	0	0	0
80	1	1	2	3	1	0	0	0
80	0	10	2	3	9	7	1	8
80	2	17	3	2	7	7	7	7
80	1	6	5	3	5	4	0	3
80	0	10	3	3	9	5	0	8
80	1	5	1	2	5	2	4	3
80	1	3	2	3	2	2	1	2

In [53]:
```
Y=data['Attrition'].values
Y
```

Out[53]: `array(['Yes', 'No', 'Yes', ..., 'No', 'No', 'No'], dtype=object)`

Step3.[Feature Engineering]

In [54]:
```
data= pd.get_dummies(data,columns=['BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','Over18','Ov
data
```

Out[54]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	...	Jot
0	41	Yes	1102	1	2	1	1	2	94	3	...	
1	49	No	279	8	1	1	2	3	61	2	...	
2	37	Yes	1373	2	2	1	4	4	92	2	...	
3	33	No	1392	3	4	1	5	4	56	3	...	
4	27	No	591	2	1	1	7	1	40	3	...	
5	32	No	1005	2	2	1	8	4	79	3	...	
6	59	No	1324	3	3	1	10	3	81	4	...	
7	30	No	1358	24	1	1	11	4	67	3	...	
8	38	No	216	23	3	1	12	4	44	2	...	
9	36	No	1299	27	3	1	13	3	94	3	...	
10	35	No	809	16	3	1	14	1	84	4	...	
11	29	No	153	15	2	1	15	4	49	2	...	
12	31	No	670	26	1	1	16	1	31	3	...	
13	34	No	1346	19	2	1	18	2	93	3	...	
14	28	Yes	103	24	3	1	19	3	50	2	...	
15	29	No	1389	21	4	1	20	2	51	4	...	
16	32	No	334	5	2	1	21	1	80	4	...	
17	22	No	1123	16	2	1	22	4	96	4	...	
18	53	No	1219	2	4	1	23	1	78	2	...	
19	38	No	371	2	3	1	24	4	45	3	...	
20	24	No	673	11	2	1	26	1	96	4	...	
21	36	Yes	1218	9	4	1	27	3	82	2	...	
22	34	No	419	7	4	1	28	1	53	3	...	
23	21	No	391	15	2	1	30	3	96	3	...	
24	34	Yes	699	6	1	1	31	2	83	3	...	
25	53	No	1282	5	3	1	32	3	58	3	...	
26	32	Yes	1125	16	1	1	33	2	72	1	...	
27	42	No	691	8	4	1	35	3	48	3	...	
28	44	No	477	7	4	1	36	1	42	2	...	
29	46	No	705	2	4	1	38	2	83	3	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1440	36	No	688	4	2	1	2025	4	97	3	...	
1441	56	No	667	1	4	1	2026	3	57	3	...	
1442	29	Yes	1092	1	4	1	2027	1	36	3	...	
1443	42	No	300	2	3	1	2031	1	56	3	...	
1444	56	Yes	310	7	2	1	2032	4	72	3	...	
1445	41	No	582	28	4	1	2034	1	60	2	...	
1446	34	No	704	28	3	1	2035	4	95	2	...	
1447	36	No	301	15	4	1	2036	4	88	1	...	
1448	41	No	930	3	3	1	2037	3	57	2	...	
1449	32	No	529	2	3	1	2038	4	78	3	...	
1450	35	No	1146	26	4	1	2040	3	31	3	...	
1451	38	No	345	10	2	1	2041	1	100	3	...	
1452	50	Yes	878	1	4	1	2044	2	94	3	...	
1453	36	No	1120	11	4	1	2045	2	100	2	...	
1454	45	No	374	20	3	1	2046	4	50	3	...	
1455	40	No	1322	2	4	1	2048	3	52	2	...	
1456	35	No	1199	18	4	1	2049	3	80	3	...	
1457	40	No	1194	2	4	1	2051	3	98	3	...	
1458	35	No	287	1	4	1	2052	3	62	1	...	
1459	29	No	1378	13	2	1	2053	4	46	2	...	
1460	29	No	468	28	4	1	2054	4	73	2	...	
1461	50	Yes	410	28	3	1	2055	4	39	2	...	
1462	39	No	722	24	1	1	2056	2	60	2	...	

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	...	Jot
1463	31	No	325	5	3	1	2057	2	74	3	...	
1464	26	No	1167	5	3	1	2060	4	30	2	...	
1465	36	No	884	23	2	1	2061	3	41	4	...	
1466	39	No	613	6	1	1	2062	4	42	2	...	
1467	27	No	155	4	3	1	2064	2	87	4	...	
1468	49	No	1023	2	3	1	2065	4	63	2	...	
1469	34	No	628	8	3	1	2068	2	82	4	...	

1470 rows × 56 columns

Step4.Shape of X and Y.

In [56]:
```
X = data.drop(['Attrition'],axis=1)
print('X Shape : ',X.shape)
print('y Shape : ',Y.shape)
```

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

Step5.[Model Development]

In [58]:
```
import warnings
warnings.filterwarnings('ignore')
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_state=42)
```

In [60]:
```
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, class_weight=None, criterion='gini',
            max_depth=None, max_features=0.3, 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, n_estimators=100, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [64]:
```
RFC_Y_pred = RFC.predict(X_test)
RFC_Y_pred
```

Out[64]:
```
array(['No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No'], dtype=object)
```

Step6.[Testing]

In [66]:
```
from sklearn.metrics import accuracy_score,classification_report
```

In [68]:
```
RFC_acc = accuracy_score(y_test,RFC_Y_pred)
RFC_acc
```

Out[68]:
```
0.8673469387755102
```

In [69]:
```
print(classification_report(y_test, RFC_Y_pred))
```

```
             precision    recall  f1-score   support

         No       0.88      0.98      0.93       255
        Yes       0.50      0.13      0.20        39

avg / total       0.83      0.87      0.83       294
```

Step7.[Feature importance value]

In [70]:
```
print(RFC.feature_importances_)
```

```
[0.05449821 0.04704951 0.04100773 0.01409436 0.         0.04652908
 0.02004845 0.04108983 0.02087918 0.03175816 0.02335801 0.08271778
 0.04297201 0.03169038 0.02937403 0.00257885 0.01654872 0.
 0.02935428 0.04238877 0.02427988 0.01788414 0.03498127 0.02269083
 0.02623195 0.02649012 0.00345672 0.01137155 0.00557633 0.00149022
 0.00542984 0.00944828 0.0021697  0.00506224 0.00621831 0.00479288
 0.00238773 0.00615479 0.00486082 0.00602701 0.00147725 0.00273962
 0.00519967 0.00104466 0.00255725 0.00069257 0.00718652 0.00817832
 0.00563226 0.00512782 0.00706891 0.02208759 0.         0.04436242
 0.04170318]
```

In [71]:
```
feature_name = pd.DataFrame(RFC.feature_importances_, index=X_train.columns, columns=['Important_Feature'])
feature_name
```

Out[71]:

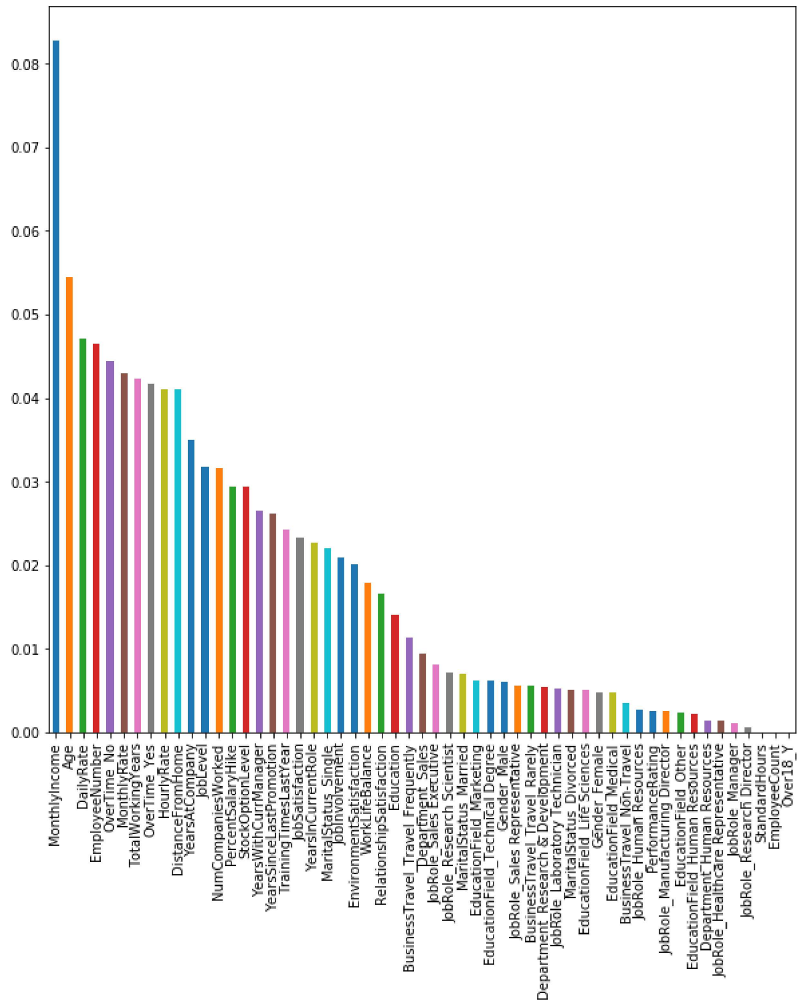
	Important_Feature
Age	0.054498
DailyRate	0.047050
DistanceFromHome	0.041008
Education	0.014094
EmployeeCount	0.000000
EmployeeNumber	0.046529
EnvironmentSatisfaction	0.020048
HourlyRate	0.041090
JobInvolvement	0.020879
JobLevel	0.031758
JobSatisfaction	0.023358
MonthlyIncome	0.082718
MonthlyRate	0.042972
NumCompaniesWorked	0.031690
PercentSalaryHike	0.029374
PerformanceRating	0.002579
RelationshipSatisfaction	0.016549
StandardHours	0.000000
StockOptionLevel	0.029354
TotalWorkingYears	0.042389
TrainingTimesLastYear	0.024280
WorkLifeBalance	0.017884
YearsAtCompany	0.034981
YearsInCurrentRole	0.022691
YearsSinceLastPromotion	0.026232
YearsWithCurrManager	0.026490
BusinessTravel_Non-Travel	0.003457
BusinessTravel_Travel_Frequently	0.011372
BusinessTravel_Travel_Rarely	0.005576
Department_Human Resources	0.001490
Department_Research & Development	0.005430
Department_Sales	0.009448
EducationField_Human Resources	0.002170
EducationField_Life Sciences	0.005062
EducationField_Marketing	0.006218
EducationField_Medical	0.004793
EducationField_Other	0.002388
EducationField_Technical Degree	0.006155
Gender_Female	0.004861
Gender_Male	0.006027
JobRole_Healthcare Representative	0.001477
JobRole_Human Resources	0.002740
JobRole_Laboratory Technician	0.005200
JobRole_Manager	0.001045
JobRole_Manufacturing Director	0.002557
JobRole_Research Director	0.000693
JobRole_Research Scientist	0.007187
JobRole_Sales Executive	0.008178
JobRole_Sales Representative	0.005632
MaritalStatus_Divorced	0.005128
MaritalStatus_Married	0.007069
MaritalStatus_Single	0.022088
Over18_Y	0.000000
OverTime_No	0.044362

	Important_Feature
OverTime_Yes	0.041703

In [72]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
```

In [77]:
```python
pd.Series(RFC.feature_importances_, index=X_train.columns).sort_values(ascending=False).plot(kind='bar' ,figsize=(10,10))
```

Out[77]: `<matplotlib.axes._subplots.AxesSubplot at 0x203c1df3630>`



Step8.[Visualize your RF Decsion Tree using graphviz]

In [78]:
```python
estimator = RFC.estimators_[5]
```

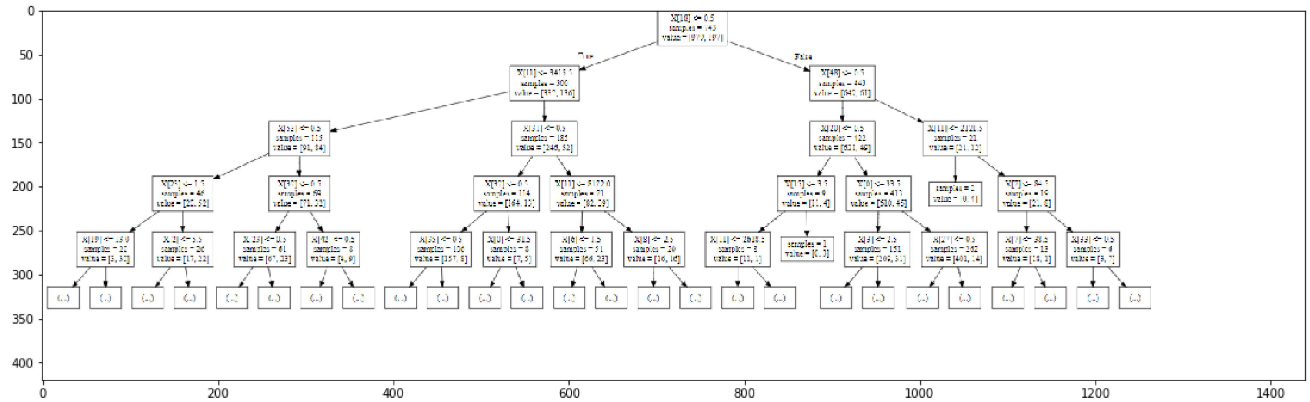
In [94]:
```python
from sklearn import tree
from sklearn.tree import export_graphviz
with open("RFDT.dot", 'w') as f:
     f = tree.export_graphviz(estimator, out_file=f, max_depth=4, impurity=False)
```

In [95]:
```python
!dot - Tpng RFDT.dot -o RFDT.png
```

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

In [96]:
```python
import matplotlib.pyplot as plt
image = plt.imread('RFDT.png')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

Out[96]: <matplotlib.image.AxesImage at 0x203c24cdda0>



Step9.[RF with a range of trees]

In [97]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [98]:
```python
rf2 = RandomForestClassifier(oob_score=True, random_state=42, warm_start=True, n_jobs=-1)
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
```

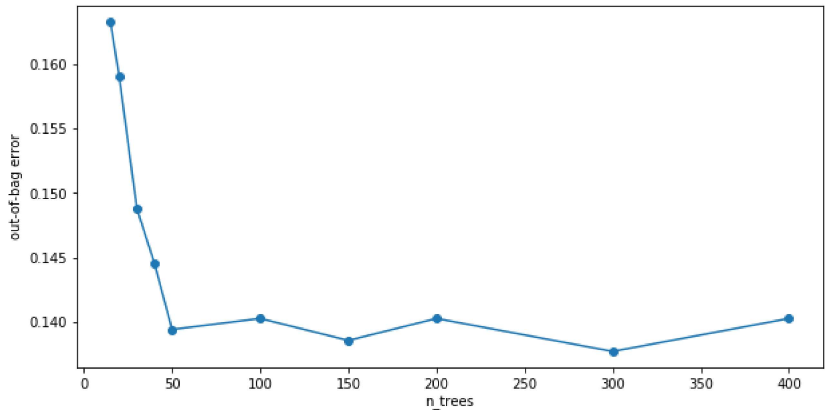
Out[98]:

	oob
n_trees	
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

Step10.[Plot oob-error for each tree]

In [99]:
```
ax = rf_oob_df.plot(legend=False, marker='o', figsize=(10,5))
ax.set(ylabel='out-of-bag error')
```

Out[99]:
```
[Text(0,0.5,'out-of-bag error')]
```



Step11.[Compare with DecisionTreeClassifier]

In [107]:
```
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[107]:
```
DecisionTreeClassifier(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=False, random_state=42,
            splitter='best')
```

In [108]:
```
y_pred1 = clf.predict(X_test)
y_pred1
```

Out[108]:
```
array(['No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No'], dtype=object)
```

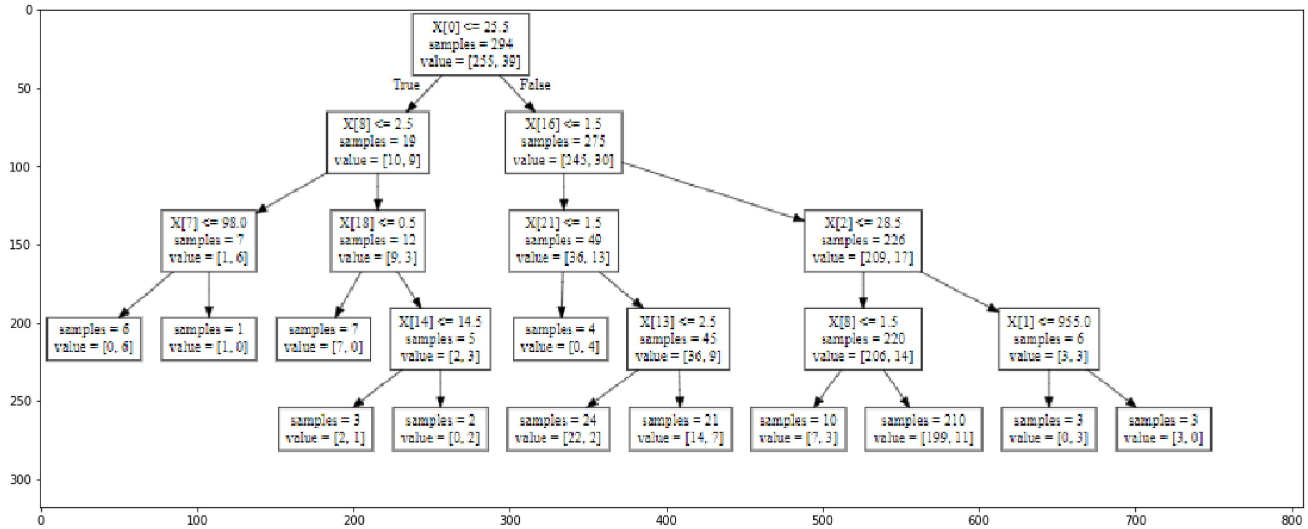
In [109]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
with open("DTC2.dot", 'w') as f:
        f = tree.export_graphviz(clf,out_file=f,max_depth = 4,impurity = False)
```

In [110]:
```
!dot -Tpng DTC2.dot -o DTC2.png
```

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

In [111]:
```
image = plt.imread('DTC2.png')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

Out[111]: <matplotlib.image.AxesImage at 0x203c25a9a58>



In [112]:
```
print("Accuracy of test :",clf.score(X_test,y_test))
```

```
Accuracy of test : 0.9183673469387755
```

In [117]:
```
print(classification_report(y_test,RFC_Y_pred))
```

```
             precision    recall  f1-score   support

         No       0.88      0.98      0.93       255
        Yes       0.50      0.13      0.20        39

avg / total       0.83      0.87      0.83       294
```

In [ ]: