LAB 9 : Employee Hopping Prediction Using Random Forests

Dinesh Kumar K

225229108

Step 1 : Understand Data

In [42]:
```
import pandas as pd
```

In [43]:
```
emp = pd.read_csv("Employee_hopping.csv")
```

In [44]:
```
emp
```

Out[44]:

	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	32	No	Travel_Frequently	1005	Research & Development	2	2	Life Sciences	1	8
6	59	No	Travel_Rarely	1324	Research & Development	3	3	Medical	1	10

In [45]:
```
emp.shape
```

Out[45]:
```
(1470, 35)
```

In [46]:
```
emp.head
```

```
5      32        No  Travel_Frequently       1005  Research & Development
6      59        No      Travel_Rarely       1324  Research & Development
7      30        No      Travel_Rarely       1358  Research & Development
8      38        No  Travel_Frequently        216  Research & Development
9      36        No      Travel_Rarely       1299  Research & Development
10     35        No      Travel_Rarely        809  Research & Development
11     29        No      Travel_Rarely        153  Research & Development
12     31        No      Travel_Rarely        670  Research & Development
13     34        No      Travel_Rarely       1346  Research & Development
14     28       Yes      Travel_Rarely        103  Research & Development
15     29        No      Travel_Rarely       1389  Research & Development
16     32        No      Travel_Rarely        334  Research & Development
17     22        No         Non-Travel       1123  Research & Development
18     53        No      Travel_Rarely       1219                   Sales
19     38        No      Travel_Rarely        371  Research & Development
20     24        No         Non-Travel        673  Research & Development
21     36       Yes      Travel_Rarely       1218                   Sales
22     34        No      Travel_Rarely        419  Research & Development
23     21        No      Travel_Rarely        391  Research & Development
```

In [47]:
```
emp.columns
```

Out[47]:
```
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 [48]:
```
emp.count
```

```
8      38        No  Travel_Frequently        216  Research & Development
9      36        No      Travel_Rarely       1299  Research & Development
10     35        No      Travel_Rarely        809  Research & Development
11     29        No      Travel_Rarely        153  Research & Development
12     31        No      Travel_Rarely        670  Research & Development
13     34        No      Travel_Rarely       1346  Research & Development
14     28       Yes      Travel_Rarely        103  Research & Development
15     29        No      Travel_Rarely       1389  Research & Development
16     32        No      Travel_Rarely        334  Research & Development
17     22        No         Non-Travel       1123  Research & Development
18     53        No      Travel_Rarely       1219                   Sales
19     38        No      Travel_Rarely        371  Research & Development
20     24        No         Non-Travel        673  Research & Development
21     36       Yes      Travel_Rarely       1218                   Sales
22     34        No      Travel_Rarely        419  Research & Development
23     21        No      Travel_Rarely        391  Research & Development
24     34       Yes      Travel_Rarely        699  Research & Development
25     53        No      Travel_Rarely       1282  Research & Development
26     32       Yes  Travel_Frequently       1125  Research & Development
```

In [49]:
```
emp.info
```

```
25     53        No      Travel_Rarely       1282  Research & Development
26     32       Yes  Travel_Frequently       1125  Research & Development
27     42        No      Travel_Rarely        691                   Sales
28     44        No      Travel_Rarely        477  Research & Development
29     46        No      Travel_Rarely        705                   Sales
...   ...       ...                ...        ...                     ...
1440   36        No  Travel_Frequently        688  Research & Development
1441   56        No         Non-Travel        667  Research & Development
1442   29       Yes      Travel_Rarely       1092  Research & Development
1443   42        No      Travel_Rarely        300  Research & Development
1444   56       Yes      Travel_Rarely        310  Research & Development
1445   41        No      Travel_Rarely        582  Research & Development
1446   34        No      Travel_Rarely        704                   Sales
1447   36        No         Non-Travel        301                   Sales
1448   41        No      Travel_Rarely        930                   Sales
1449   32        No      Travel_Rarely        529  Research & Development
1450   35        No      Travel_Rarely       1146        Human Resources
1451   38        No      Travel_Rarely        345                   Sales
1452   50       Yes  Travel_Frequently        878                   Sales
1453   36        No      Travel_Rarely       1120                   Sales
```

In [50]:
```
emp['Department'].value_counts
```

Out[50]:
```
<bound method IndexOpsMixin.value_counts of 0                        Sales
1       Research & Development
2       Research & Development
3       Research & Development
4       Research & Development
5       Research & Development
6       Research & Development
7       Research & Development
8       Research & Development
9       Research & Development
10      Research & Development
11      Research & Development
12      Research & Development
13      Research & Development
14      Research & Development
15      Research & Development
16      Research & Development
17      Research & Development
18                       Sales
19      Research & Development
20      Research & Development
21                       Sales
22      Research & Development
23      Research & Development
24      Research & Development
25      Research & Development
26      Research & Development
27                       Sales
28      Research & Development
29                       Sales
                 ...
1440    Research & Development
1441    Research & Development
1442    Research & Development
1443    Research & Development
1444    Research & Development
1445    Research & Development
1446                     Sales
1447                     Sales
1448                     Sales
1449    Research & Development
1450           Human Resources
1451                     Sales
1452                     Sales
1453                     Sales
1454                     Sales
1455    Research & Development
1456    Research & Development
1457    Research & Development
1458    Research & Development
1459    Research & Development
1460    Research & Development
1461                     Sales
1462                     Sales
1463    Research & Development
1464                     Sales
1465    Research & Development
1466    Research & Development
1467    Research & Development
1468                     Sales
1469    Research & Development
Name: Department, Length: 1470, dtype: object>
```

Step 2 : Extract X and Y

In [61]:
```
X=emp.drop('Age', axis=1)
```

In [52]:
```
Y=emp['Education'].values
```

In [53]:
```
X.head()
```

Out[53]:

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

5 rows × 34 columns

In [54]:
```
Y
```

Out[54]:
```
array([2, 1, 2, ..., 3, 3, 3], dtype=int64)
```

Step 3 : Feature Engineering

In [55]:
```
encoding = pd.get_dummies(emp, columns = ['BusinessTravel','Department','EducationField','EmployeeCount','Gender','JobRo
encoding
```

Out[55]:

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

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...
1459	29	No	1378	13	2	2053	4	46	2	2	...
1460	29	No	468	28	4	2054	4	73	2	1	...
1461	50	Yes	410	28	3	2055	4	39	2	3	...
1462	39	No	722	24	1	2056	2	60	2	4	...
1463	31	No	325	5	3	2057	2	74	3	2	...
1464	26	No	1167	5	3	2060	4	30	2	1	...
1465	36	No	884	23	2	2061	3	41	4	2	...
1466	39	No	613	6	1	2062	4	42	2	3	...
1467	27	No	155	4	3	2064	2	87	4	2	...
1468	49	No	1023	2	3	2065	4	63	2	2	...
1469	34	No	628	8	3	2068	2	82	4	2	...

1470 rows × 56 columns

Step 4 : Now, Check shape of X and Y

In [72]:
```
X=encoding.drop(['Attrition'], axis=1)
```

In [73]:
```
X.shape
```

Out[73]:
```
(1470, 55)
```

In [76]:
```
Y=encoding['Age'].values
```

In [77]:
```
Y.shape
```

Out[77]:
```
(1470,)
```

Step 5 : Model Development

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

In [98]:
```
from sklearn.model_selection import train_test_split
```

In [99]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size =0.2, random_state =42)
```

In [100]:
```
X_train.shape
```

Out[100]:
```
(1176, 55)
```

In [101]:
```
Y_train.shape
```

Out[101]:
```
(1176,)
```

In [102]:
```
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier(n_estimators=100, max_features=0.3)
```

In [103]:
```
RFC.fit(X_train,Y_train)
```

Out[103]:
```
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 [104]:
```
RFC_y_pred = RFC.predict(X_test)
RFC_y_pred
```

Out[104]:
```
array([28, 38, 24, 45, 36, 34, 35, 40, 45, 35, 38, 44, 43, 40, 22, 30, 29,
       36, 45, 34, 31, 46, 32, 38, 33, 55, 27, 33, 37, 34, 30, 26, 39, 37,
       36, 35, 40, 41, 28, 28, 38, 39, 32, 38, 33, 38, 40, 33, 49, 19, 46,
       31, 50, 34, 29, 32, 40, 33, 34, 39, 45, 20, 31, 50, 37, 35, 27, 37,
       28, 40, 36, 46, 37, 35, 29, 55, 34, 34, 36, 30, 53, 39, 58, 26, 29,
       44, 41, 43, 32, 30, 37, 18, 39, 30, 37, 31, 29, 37, 45, 46, 55, 28,
       45, 27, 48, 38, 51, 32, 37, 50, 35, 45, 31, 30, 38, 45, 51, 29, 48,
       33, 40, 34, 42, 53, 38, 19, 25, 39, 39, 31, 29, 18, 40, 33, 26, 34,
       21, 31, 38, 43, 28, 42, 36, 27, 49, 32, 54, 21, 32, 25, 32, 46, 58,
       32, 39, 25, 31, 22, 46, 32, 38, 34, 45, 42, 34, 30, 30, 39, 50, 35,
       37, 40, 35, 24, 32, 34, 34, 42, 42, 24, 56, 28, 42, 39, 35, 18, 40,
       49, 46, 29, 40, 46, 29, 42, 28, 45, 44, 42, 35, 38, 43, 35, 34, 41,
       53, 33, 35, 26, 42, 58, 44, 34, 25, 38, 31, 30, 36, 40, 31, 49, 45,
       24, 37, 29, 38, 27, 30, 32, 35, 33, 40, 31, 43, 34, 42, 38, 45, 25,
       58, 46, 33, 34, 26, 43, 37, 54, 34, 36, 40, 47, 39, 41, 50, 31, 40,
       30, 29, 41, 40, 43, 37, 30, 30, 34, 33, 47, 50, 30, 35, 50, 35, 50,
       42, 34, 30, 31, 45, 43, 38, 41, 35, 30, 44, 29, 49, 35, 28, 32, 31,
       34, 34, 50, 37, 45], dtype=int64)
```

Step : 6 TESTING

In [105]:
```
from sklearn.metrics import accuracy_score,classification_report
```

In [106]:
```
RFC_acc = accuracy_score(Y_test,RFC_y_pred)
RFC_acc
```

Out[106]:
```
0.7993197278911565
```

In [107]:
```
print(classification_report(Y_test, RFC_y_pred))
```

```
             precision    recall  f1-score   support

         18       0.33      1.00      0.50         1
         19       1.00      1.00      1.00         2
         20       0.00      0.00      0.00         0
         21       1.00      0.50      0.67         4
         22       1.00      1.00      1.00         2
         24       1.00      1.00      1.00         4
         25       1.00      0.83      0.91         6
         26       0.80      1.00      0.89         4
         27       1.00      0.83      0.91         6
         28       1.00      0.90      0.95        10
         29       0.92      1.00      0.96        11
         30       1.00      1.00      1.00        16
         31       1.00      1.00      1.00        14
         32       1.00      1.00      1.00        13
         33       1.00      1.00      1.00        11
         34       1.00      1.00      1.00        22
         35       0.94      1.00      0.97        16
         36       1.00      1.00      1.00         8
         37       0.93      1.00      0.96        13
         38       0.69      1.00      0.81        11
         39       0.82      0.75      0.78        12
         40       0.81      0.87      0.84        15
         41       0.83      0.83      0.83         6
         42       0.55      0.75      0.63         8
         43       0.25      0.50      0.33         4
         44       0.20      0.11      0.14         9
         45       0.43      0.75      0.55         8
         46       0.67      0.75      0.71         8
         47       0.00      0.00      0.00         6
         48       0.50      0.33      0.40         3
         49       0.60      0.33      0.43         9
         50       0.44      0.57      0.50         7
         51       0.00      0.00      0.00         2
         52       0.00      0.00      0.00         2
         53       1.00      0.75      0.86         4
         54       1.00      1.00      1.00         2
         55       1.00      1.00      1.00         3
         56       1.00      0.25      0.40         4
         57       0.00      0.00      0.00         1
         58       0.25      0.50      0.33         2
         59       0.00      0.00      0.00         3
         60       0.00      0.00      0.00         2

avg / total       0.79      0.80      0.78       294
```

Step 7 : Feature importance value

In [108]:
```
print(RFC.feature_importances_)
```

```
[0.38056055 0.03663568 0.0280913  0.0170511  0.03449571 0.01427878
 0.03235762 0.01157546 0.00852012 0.0146504  0.03962492 0.03568141
 0.02001884 0.02426619 0.00370627 0.01487806 0.         0.01163563
 0.04620171 0.0165339  0.01132283 0.0267818  0.01829193 0.01786921
 0.02019311 0.00284774 0.00499468 0.00603348 0.00128175 0.0050114
 0.00473539 0.00074589 0.00656279 0.00316088 0.0065129  0.00230257
 0.00314961 0.         0.00585066 0.00583096 0.00320794 0.0012906
 0.00467852 0.00144063 0.00332829 0.00151814 0.0044543  0.00354256
 0.00229274 0.0053517  0.00633099 0.0074596  0.         0.00507279
 0.00578798]
```

In [109]:
```
feature_name = pd.DataFrame(RFC.feature_importances_, index=X_train.columns, columns=['Importance_Feature'])
feature_name
```

Out[109]:

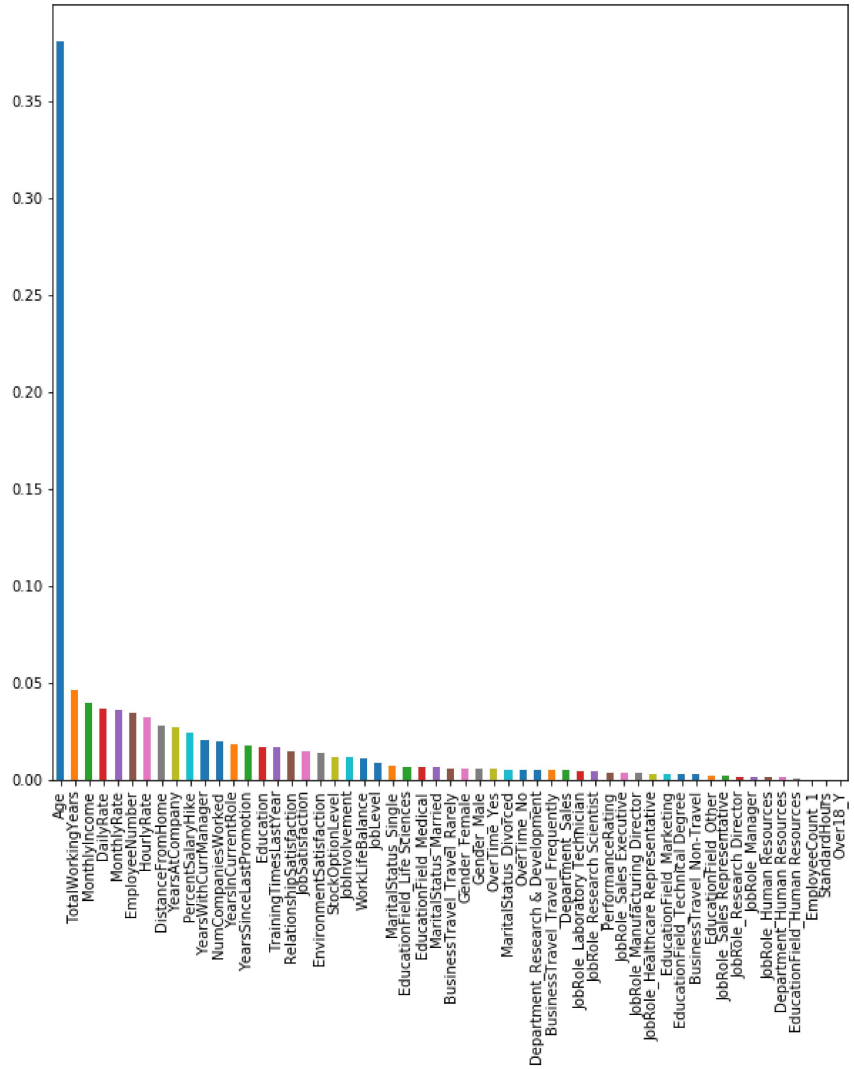
	Importance_Feature
Age	0.380561
DailyRate	0.036636
DistanceFromHome	0.028091
Education	0.017051
EmployeeNumber	0.034496
EnvironmentSatisfaction	0.014279
HourlyRate	0.032358
JobInvolvement	0.011575
JobLevel	0.008520
JobSatisfaction	0.014650
MonthlyIncome	0.039625
MonthlyRate	0.035681
NumCompaniesWorked	0.020019
PercentSalaryHike	0.024266
PerformanceRating	0.003706
RelationshipSatisfaction	0.014878
StandardHours	0.000000
StockOptionLevel	0.011636
TotalWorkingYears	0.046202
TrainingTimesLastYear	0.016534
WorkLifeBalance	0.011323
YearsAtCompany	0.026782
YearsInCurrentRole	0.018292
YearsSinceLastPromotion	0.017869
YearsWithCurrManager	0.020193
BusinessTravel_Non-Travel	0.002848
BusinessTravel_Travel_Frequently	0.004995
BusinessTravel_Travel_Rarely	0.006033
Department_Human Resources	0.001282
Department_Research & Development	0.005011
Department_Sales	0.004735
EducationField_Human Resources	0.000746
EducationField_Life Sciences	0.006563
EducationField_Marketing	0.003161
EducationField_Medical	0.006513
EducationField_Other	0.002303
EducationField_Technical Degree	0.003150
EmployeeCount_1	0.000000
Gender_Female	0.005851
Gender_Male	0.005831
JobRole_Healthcare Representative	0.003208
JobRole_Human Resources	0.001291
JobRole_Laboratory Technician	0.004679
JobRole_Manager	0.001441
JobRole_Manufacturing Director	0.003328
JobRole_Research Director	0.001518
JobRole_Research Scientist	0.004454
JobRole_Sales Executive	0.003543
JobRole_Sales Representative	0.002293
MaritalStatus_Divorced	0.005352
MaritalStatus_Married	0.006331

	Importance_Feature
MaritalStatus_Single	0.007460
Over18_Y	0.000000
OverTime_No	0.005073
OverTime_Yes	0.005788

In [110]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [117]:
```
pd.Series(RFC.feature_importances_, index=X_train.columns).sort_values(ascending=False).plot(kind='bar', figsize=(10,10)
```

Out[117]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26a72931e10>
```



Step 8 : Visualize your RF Decision tree using graphviz

In [132]:
```
estimator = RFC.estimators_[5]
```

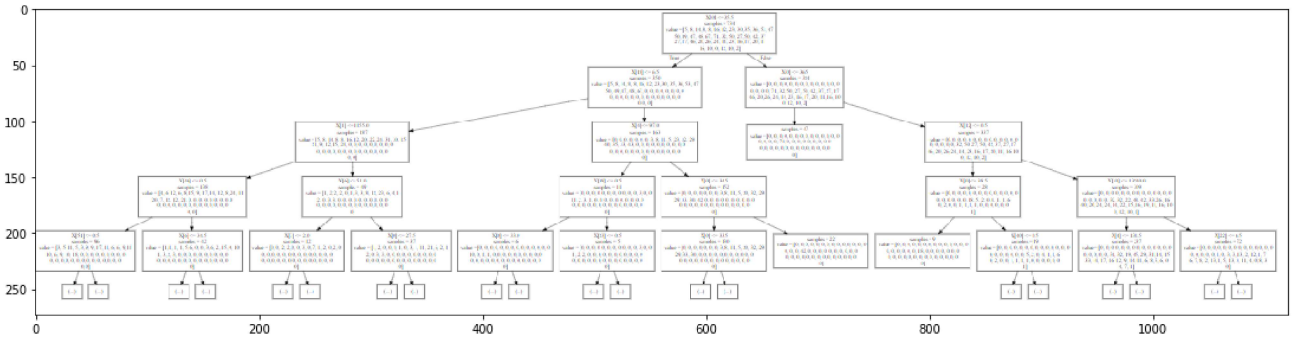
In [134]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
with open("RFDT.dot", 'w') as f:
    f = tree.export_graphviz(estimator, out_file=f, max_depth=4, impurity=False)
```

In [135]:
```
!dot - Tpng RFDT.dot -o RFDT.png
```

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

In [137]:
```
import matplotlib.pyplot as plt
image = plt.imread('RFDT.png')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

Out[137]: <matplotlib.image.AxesImage at 0x26a73021cf8>



Step 9 : RF with a range of trees

In [118]:
```
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
```

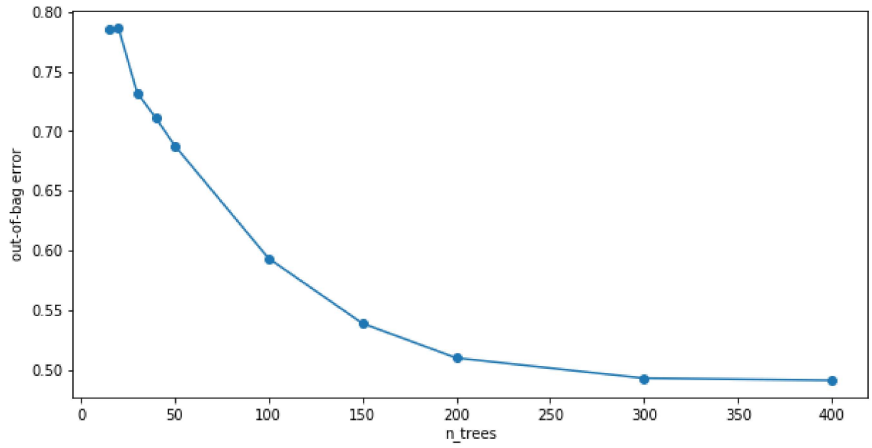
Out[118]:

n_trees	oob
15.0	0.784864
20.0	0.786565
30.0	0.732143
40.0	0.710884
50.0	0.687925
100.0	0.593537
150.0	0.539116
200.0	0.510204
300.0	0.493197
400.0	0.491497

Step 10 : plot oob-error for each tree

In [119]:
```
ax = rf_oob_df.plot(legend=False, marker='o', figsize=(10,5))
ax.set(ylabel='out-of-bag error')
```

Out[119]:
```
[Text(0,0.5,'out-of-bag error')]
```



step 11 : Compare with DecisionTreeClassifier

In [121]:
```
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[121]:
```
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 [122]:
```
y_pred1 = clf.predict(X_test)
y_pred1
```

Out[122]:
```
array([30, 44, 30, 44, 36, 34, 35, 40, 44, 35, 44, 44, 44, 40, 30, 30, 30,
       36, 44, 34, 31, 44, 32, 40, 33, 44, 30, 33, 44, 34, 30, 30, 40, 37,
       36, 35, 40, 44, 30, 30, 40, 40, 32, 40, 33, 40, 40, 33, 44, 30, 44,
       31, 44, 34, 30, 32, 40, 33, 34, 40, 44, 30, 31, 44, 37, 35, 30, 37,
       30, 40, 36, 44, 37, 35, 30, 44, 34, 34, 36, 30, 44, 40, 44, 30, 30,
       44, 44, 44, 32, 30, 37, 30, 40, 30, 37, 31, 30, 37, 44, 44, 44, 30,
       44, 30, 44, 40, 44, 32, 37, 44, 35, 44, 31, 30, 40, 44, 44, 30, 44,
       33, 44, 34, 44, 44, 40, 30, 30, 40, 40, 31, 30, 30, 44, 33, 30, 34,
       30, 31, 44, 44, 30, 44, 36, 30, 44, 32, 44, 30, 32, 30, 32, 44, 44,
       32, 40, 30, 31, 30, 44, 32, 40, 34, 44, 44, 34, 30, 30, 40, 44, 35,
       37, 40, 35, 30, 32, 34, 34, 44, 44, 30, 44, 30, 44, 40, 35, 30, 40,
       44, 44, 30, 40, 44, 30, 44, 30, 44, 44, 44, 35, 40, 44, 35, 34, 44,
       44, 33, 35, 30, 44, 44, 44, 34, 30, 40, 31, 30, 36, 40, 31, 44, 44,
       30, 37, 30, 40, 30, 30, 32, 35, 33, 40, 31, 44, 34, 44, 40, 44, 30,
       44, 44, 33, 34, 30, 44, 37, 44, 34, 36, 40, 44, 40, 44, 44, 31, 40,
       30, 30, 44, 40, 44, 37, 30, 30, 34, 33, 44, 44, 30, 35, 44, 35, 44,
       44, 34, 30, 31, 44, 44, 40, 44, 44, 30, 44, 30, 44, 35, 30, 32, 31,
       34, 34, 44, 37, 44], dtype=int64)
```

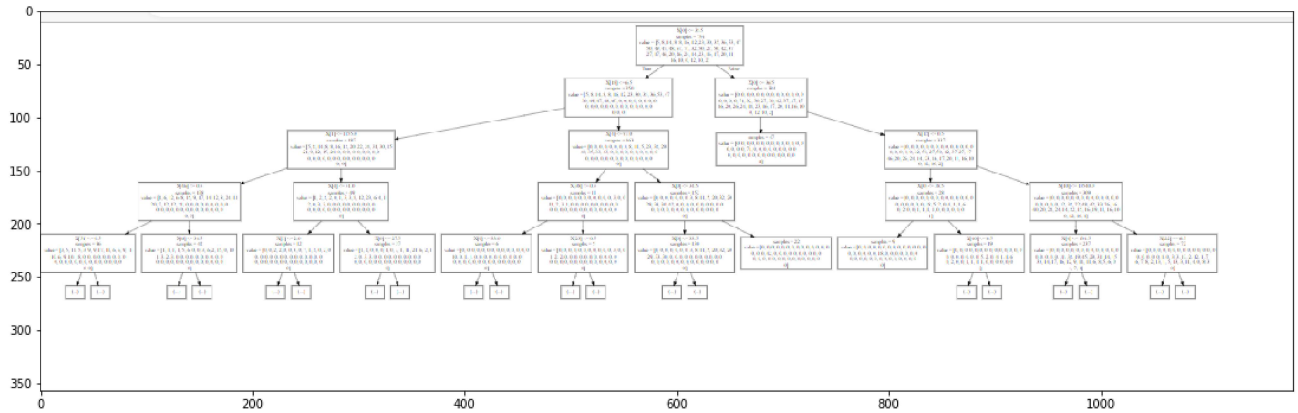
In [139]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
with open("DTC2.dot", 'w') as f:
    f = tree.export_graphviz(clf,out_file=f,max_depth = 4,impurity = False)
```

In [140]:
```
!dot -Tpng DTC2.dot -o DTC2.png
```

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

In [142]:
```
image = plt.imread('RFDT2.png')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

Out[142]: <matplotlib.image.AxesImage at 0x26a730778d0>



In [144]:
```
print("Accuracy of test :",clf.score(X_test,Y_test))
```

```
Accuracy of test : 0.46598639455782315
```

In [146]:
```
print(classification_report(Y_test,RFC_y_pred))
```

```
             precision    recall  f1-score   support

         18       0.33      1.00      0.50         1
         19       1.00      1.00      1.00         2
         20       0.00      0.00      0.00         0
         21       1.00      0.50      0.67         4
         22       1.00      1.00      1.00         2
         24       1.00      1.00      1.00         4
         25       1.00      0.83      0.91         6
         26       0.80      1.00      0.89         4
         27       1.00      0.83      0.91         6
         28       1.00      0.90      0.95        10
         29       0.92      1.00      0.96        11
         30       1.00      1.00      1.00        16
         31       1.00      1.00      1.00        14
         32       1.00      1.00      1.00        13
         33       1.00      1.00      1.00        11
         34       1.00      1.00      1.00        22
         35       0.94      1.00      0.97        16
         36       1.00      1.00      1.00         8
         37       0.93      1.00      0.96        13
         38       0.69      1.00      0.81        11
         39       0.82      0.75      0.78        12
         40       0.81      0.87      0.84        15
         41       0.83      0.83      0.83         6
         42       0.55      0.75      0.63         8
         43       0.25      0.50      0.33         4
         44       0.20      0.11      0.14         9
         45       0.43      0.75      0.55         8
         46       0.67      0.75      0.71         8
         47       0.00      0.00      0.00         6
         48       0.50      0.33      0.40         3
         49       0.60      0.33      0.43         9
         50       0.44      0.57      0.50         7
         51       0.00      0.00      0.00         2
         52       0.00      0.00      0.00         2
         53       1.00      0.75      0.86         4
         54       1.00      1.00      1.00         2
         55       1.00      1.00      1.00         3
         56       1.00      0.25      0.40         4
         57       0.00      0.00      0.00         1
         58       0.25      0.50      0.33         2
         59       0.00      0.00      0.00         3
         60       0.00      0.00      0.00         2

avg / total       0.79      0.80      0.78       294
```

In [ ]: