Lab 9 : Employee Hopping Prediction using Random Forests

Name : Murali kumar R

Roll no : 225229120

Step 1 : [Understand Data]

In [47]:
```
import pandas as pd
```

In [48]:
```
emp=pd.read_csv('Employee_hopping.csv')
emp.head()
```

Out[48]:

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

5 rows × 35 columns

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

Out[49]:
```
(1470, 35)
```

In [4]:
```
emp.columns
```

Out[4]:
```
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 [5]:
```
type(emp)
```

Out[5]:
```
pandas.core.frame.DataFrame
```

In [6]:
```
emp.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 [7]:
```
emp['YearsWithCurrManager'].value_counts()
```

Out[7]:
```
2     344
0     263
7     216
3     142
8     107
4      98
1      76
9      64
5      31
6      29
10     27
11     22
12     18
13     14
17      7
14      5
15      5
16      2
Name: YearsWithCurrManager, dtype: int64
```

Step 2 : [Extract X and y]

In [50]:
```
x=emp.drop('Attrition',axis=1)
y=emp.Attrition
y=y.apply(lambda x:1 if x=='Yes' else 0)
```

In [51]:
```
X.head()
```

Out[51]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction	HourlyR
0	41	1102	1	2	1	2	
1	49	279	8	1	2	3	
2	37	1373	2	2	4	4	
3	33	1392	3	4	5	4	
4	27	591	2	1	7	1	

5 rows × 55 columns

In [52]:
```
y
```

Out[52]:
```
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 [53]:
```
encoding = pd.get_dummies(emp, columns = ['BusinessTravel','Department','EducationF
encoding
```

Out[53]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfa
0	41	Yes	1102	1	2	1	
1	49	No	279	8	1	2	
2	37	Yes	1373	2	2	4	
3	33	No	1392	3	4	5	
4	27	No	591	2	1	7	
5	32	No	1005	2	2	8	
6	59	No	1324	3	3	10	
7	30	No	1358	24	1	11	
8	38	No	216	23	3	12	
9	36	No	1299	27	3	13	

Step 4 : [Check Shape]

In [54]:
```
X=encoding.drop(['Attrition'],axis=1)
```

In [55]:
```
X.shape
```

Out[55]: (1470, 55)

In [57]:
```
y.shape
```

Out[57]: (1470,)

Step 5 : [Model test]

In [58]:
```
from sklearn.model_selection import train_test_split
```

In [62]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y, test_size=0.2,random_state=42)
```

In [63]:
```
X_train.shape
```
Out[63]:
```
(1176, 55)
```

In [64]:
```
y_train.shape
```
Out[64]:
```
(1176,)
```

In [65]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [66]:
```
model=RandomForestClassifier(n_estimators=100, max_features=0.3)
```

In [67]:
```
model.fit(X_train,y_train)
```
Out[67]:
```
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 [68]:
```
y_pred=model.predict(X_test)
```

In [69]:
```
y_pred
```
Out[69]:
```
array([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, 0, 0, 0,
       1, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 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 [70]:
```
from sklearn.metrics import accuracy_score,classification_report
```

In [71]:
```
acc = accuracy_score(y_test,y_pred)
acc
```
Out[71]:
```
0.8741496598639455
```

In [72]:
```
print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

          0       0.88      0.99      0.93       255
          1       0.62      0.13      0.21        39

avg / total       0.85      0.87      0.84       294
```

Step 7 : [Feature importance value]

In [73]:
```
print(model.feature_importances_)
```

```
[0.05476655 0.04923782 0.03995739 0.01649987 0.04772205 0.02484385
 0.03614221 0.01891676 0.0283701  0.02249348 0.08108421 0.04587755
 0.03350505 0.02607883 0.00312619 0.01857444 0.         0.02662302
 0.05036514 0.02102396 0.02052009 0.04150218 0.02069596 0.02358036
 0.02478822 0.00450541 0.01395065 0.00296644 0.00214223 0.00631532
 0.00784448 0.00189612 0.00485546 0.00489594 0.00593342 0.00320703
 0.00751661 0.         0.00409036 0.00422273 0.00172996 0.00229542
 0.00718595 0.00103069 0.00242365 0.00087401 0.00468521 0.00843061
 0.00613837 0.00509151 0.00562347 0.02296624 0.         0.04564577
 0.03524168]
```

In [74]:
```
feature_name = pd.DataFrame(model.feature_importances_, index=X_train.columns, colu
feature_name
```

Out[74]:

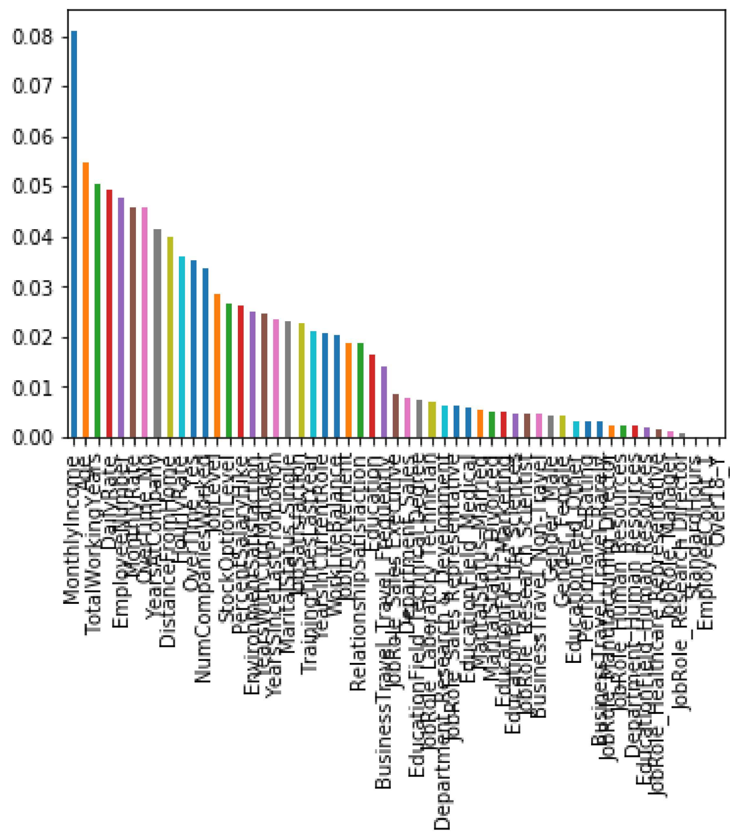
	Importance_features
Age	0.054767
DailyRate	0.049238
DistanceFromHome	0.039957
Education	0.016500
EmployeeNumber	0.047722
EnvironmentSatisfaction	0.024844
HourlyRate	0.036142
JobInvolvement	0.018917
JobLevel	0.028370
JobSatisfaction	0.022493
MonthlyIncome	0.081084
MonthlyRate	0.045878
NumCompaniesWorked	0.033505
PercentSalaryHike	0.026079
PerformanceRating	0.003126
RelationshipSatisfaction	0.018574
StandardHours	0.000000
StockOptionLevel	0.026623
TotalWorkingYears	0.050365
TrainingTimesLastYear	0.021024
WorkLifeBalance	0.020520
YearsAtCompany	0.041502
YearsInCurrentRole	0.020696
YearsSinceLastPromotion	0.023580
YearsWithCurrManager	0.024788
BusinessTravel_Non-Travel	0.004505
BusinessTravel_Travel_Frequently	0.013951
BusinessTravel_Travel_Rarely	0.002966
Department_Human Resources	0.002142
Department_Research & Development	0.006315
Department_Sales	0.007844
EducationField_Human Resources	0.001896
EducationField_Life Sciences	0.004855
EducationField_Marketing	0.004896
EducationField_Medical	0.005933
EducationField_Other	0.003207

	Importance_features
EducationField_Technical Degree	0.007517
EmployeeCount_1	0.000000
Gender_Female	0.004090
Gender_Male	0.004223
JobRole_Healthcare Representative	0.001730
JobRole_Human Resources	0.002295
JobRole_Laboratory Technician	0.007186
JobRole_Manager	0.001031
JobRole_Manufacturing Director	0.002424
JobRole_Research Director	0.000874
JobRole_Research Scientist	0.004685
JobRole_Sales Executive	0.008431
JobRole_Sales Representative	0.006138
MaritalStatus_Divorced	0.005092
MaritalStatus_Married	0.005623
MaritalStatus_Single	0.022966
Over18_Y	0.000000
OverTime_No	0.045646
OverTime_Yes	0.035242

In [77]:
```
import matplotlib.pyplot as plt
```

In [78]:
```
import seaborn as sns
```

In [84]:
```
pd.Series(model.feature_importances_,index=X_train.columns).sort_values(ascending=F
plt.show()
```



STEP- 8 Visualize your RF Decision Tree using graphviz

In [87]:
```
estimator = model.estimators_[5]
```

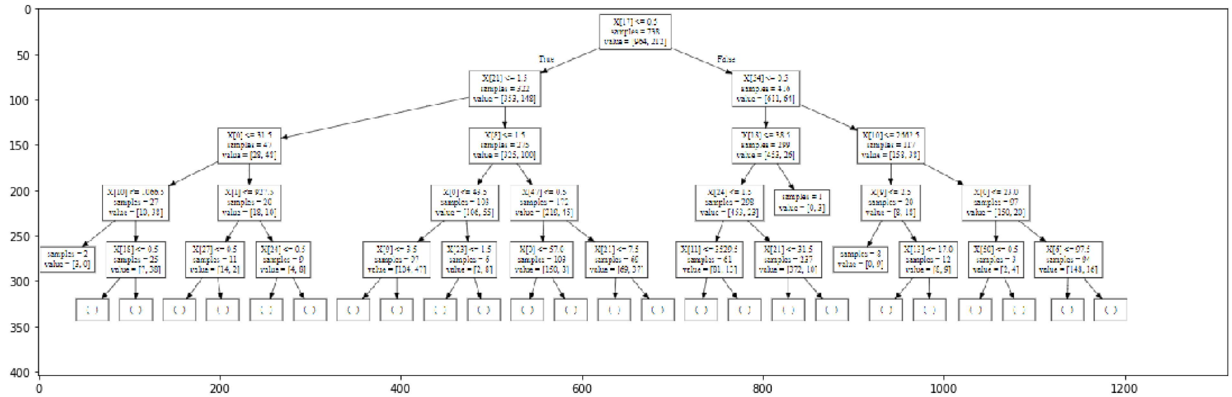
In [91]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
with open("RFDT.dot", 'w') as f:
        f = tree.export_graphviz(estimator, out_file=f, max_depth=4, impurity=False
```

In [92]:
```
!dot - Tpng RFDT.dot -o RFDT.png
```

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

In [97]:
```
import matplotlib.pyplot as plt
image = plt.imread('RFDT.png')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

Out[97]: <matplotlib.image.AxesImage at 0x29ff9854e80>



STEP- 9:RF WITH A RANGE OF TREES

In [98]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [102]:
```
rf2 = RandomForestClassifier(oob_score=True, random_state=42, warm_start=True, n_jol
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
```

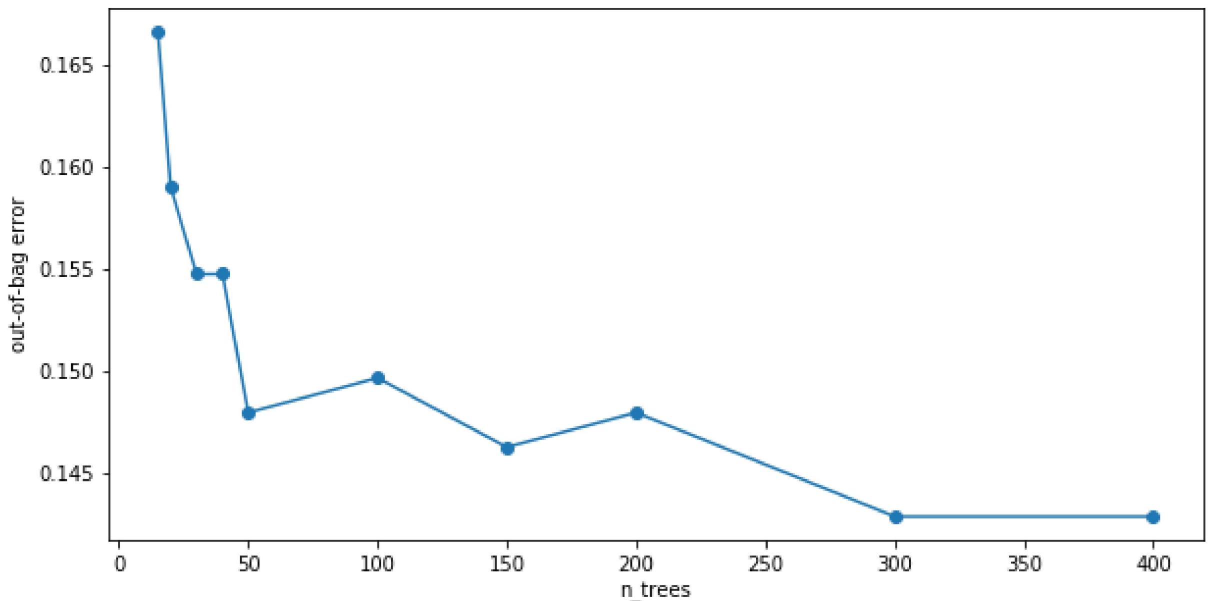
Out[102]:

	oob
n_trees	
15.0	0.166667
20.0	0.159014
30.0	0.154762
40.0	0.154762
50.0	0.147959
100.0	0.149660
150.0	0.146259
200.0	0.147959
300.0	0.142857
400.0	0.142857

Step 10:

In [103]:
```
ax = rf_oob_df.plot(legend=False, marker='o', figsize=(10,5))
ax.set(ylabel='out-of-bag error')
```

Out[103]:
```
[Text(0,0.5,'out-of-bag error')]
```



Step 11:

In [105]:
```
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[105]:
```
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 [106]:
```
y_pred1 = clf.predict(X_test)
y_pred1
```

Out[106]:
```
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 [107]:
```
estimator = model.estimators_[5]
```

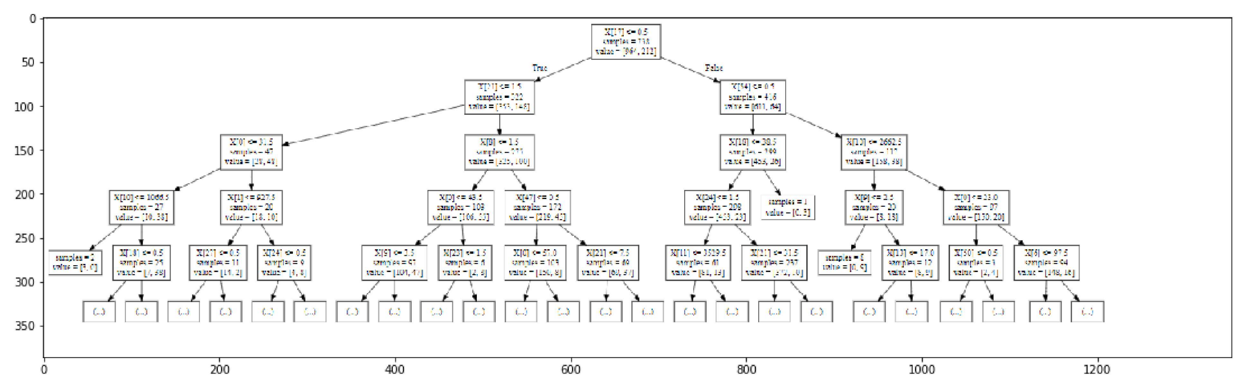
In [110]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
with open("DTC2.dot", 'w') as f:
        f = tree.export_graphviz(estimator, out_file=f, max_depth=4, impurity=False
```

In [112]:
```
!dot - Tpng DTC2.dot -o DTC2.png
```

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

In [113]:
```
import matplotlib.pyplot as plt
image = plt.imread('DTC2.png')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

Out[113]: <matplotlib.image.AxesImage at 0x29ffbc27dd8>



In [ ]: