Name : Arul Kumar ARK

Roll No. : 225229103

Lab : 9

```
                              Employee Hopping Prediction using Random
Forests
```

Step : 1

In [1]:
```
import pandas as pd
```

In [2]:
```
data = pd.read_csv("Employee_hopping.csv")
```

In [3]:
```
data.head(10)
```

Out[3]:

	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	32	No	Travel_Frequently	1005	Research & Development	2	2	Life S
6	59	No	Travel_Rarely	1324	Research & Development	3	3	
7	30	No	Travel_Rarely	1358	Research & Development	24	1	Life S
8	38	No	Travel_Frequently	216	Research & Development	23	3	Life S
9	36	No	Travel_Rarely	1299	Research & Development	27	3	

10 rows × 35 columns

In [4]:
```
data.shape
```

Out[4]:
```
(1470, 35)
```

In [5]:
```
data.columns
```

Out[5]:
```
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 [6]:
```
data.dtypes
```

Out[6]:
```
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 [7]:
```
data.info
```

```
1460   29     No      Travel_Rarely       468  Research & Development
1461   50    Yes      Travel_Rarely       410                   Sales
1462   39     No      Travel_Rarely       722                   Sales
1463   31     No         Non-Travel       325  Research & Development
1464   26     No      Travel_Rarely      1167                   Sales
1465   36     No  Travel_Frequently       884  Research & Development
1466   39     No      Travel_Rarely       613  Research & Development
1467   27     No      Travel_Rarely       155  Research & Development
1468   49     No  Travel_Frequently      1023                   Sales
1469   34     No      Travel_Rarely       628  Research & Development

      DistanceFromHome  Education    EducationField  EmployeeCount  \
0                    1          2     Life Sciences              1
1                    8          1     Life Sciences              1
2                    2          2             Other              1
3                    3          4     Life Sciences              1
4                    2          1           Medical              1
5                    2          2     Life Sciences              1
6                    3          3           Medical              1
7                   24          1     Life Sciences              1
```

In [8]:
```
data['WorkLifeBalance'].value_counts
```

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

```
1464    3
1465    3
1466    3
1467    3
1468    2
1469    4
Name: WorkLifeBalance, Length: 1470, dtype: int64>
```

Step : 2

In [9]:
```
X = data.drop(['Attrition'],axis=1)
```

In [10]:
```
y = data['Attrition']
```

In [11]:
```
y = y.apply(lambda x:1 if x == 'Yes' else 0)
```

In [12]:
```
X.shape
```

Out[12]:
```
(1470, 34)
```

In [13]:
```
y.shape
```

Out[13]:
```
(1470,)
```

Step : 3

In [14]:
```
t_dummies(data, columns = ['BusinessTravel','Department','Gender','EducationField'
)
```

Out[14]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber
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

10 rows × 56 columns

Step : 4

In [15]:
```
X = data.drop(['Attrition'],axis=1)
```

In [16]:
```
X.shape
```

Out[16]:
```
(1470, 55)
```

In [17]:
```
y=data['Attrition']
```

In [18]:
```
y = y.apply(lambda x:1 if x == 'Yes' else 0)
```

In [19]:
```
y.shape
```

Out[19]:
```
(1470,)
```

Step : 5

In [20]:
```
from sklearn.model_selection import train_test_split
```

In [21]:
```
X_train, X_test,y_train, y_test = train_test_split(X,y,random_state=42,test_size=
```

In [22]:
```
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier(n_estimators=100, max_features=0.3)
```

In [23]:
```
RFC.fit(X_train,y_train)
```

Out[23]:
```
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 [24]:
```
RFC_y_pred = RFC.predict(X_test)
RFC_y_pred
```

Out[24]:
```
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, 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, 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, 0, 1, 0, 0, 0, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 0, 0], dtype=int64)
```

Step : 6

In [25]:
```
from sklearn.metrics import accuracy_score,classification_report
```

In [26]:
```
RFC_acc = accuracy_score(y_test,RFC_y_pred)
RFC_acc
```

Out[26]:
```
0.8668478260869565
```

In [27]:
```
print(classification_report(y_test, RFC_y_pred))
```

```
             precision    recall  f1-score   support

          0       0.88      0.98      0.93       320
          1       0.44      0.08      0.14        48

avg / total       0.82      0.87      0.83       368
```

Step : 7

In [28]:
```
print(RFC.feature_importances_)
```

```
[0.0622317  0.04799897 0.0392698  0.0148707  0.         0.0460996
 0.02389722 0.03670234 0.01809801 0.02075047 0.02381954 0.08537203
 0.04164458 0.03690209 0.02941133 0.0029484  0.01773311 0.
 0.02536605 0.05112048 0.0240187  0.01776033 0.03956104 0.02509407
 0.02245903 0.02704615 0.00281675 0.01158259 0.00418142 0.0012341
 0.00657366 0.00894114 0.00649062 0.00602609 0.00263417 0.00521458
 0.00669461 0.00493512 0.00308905 0.00753628 0.0037791  0.00796889
 0.01847273 0.00172468 0.00206743 0.00667129 0.00097804 0.002338
 0.00072113 0.00539647 0.00687337 0.00716096 0.         0.04148436
 0.0362376 ]
```

In [36]:
```
feature_name = pd.DataFrame(RFC.feature_importances_, index=X_train.columns,colum
feature_name
```

Out[36]:

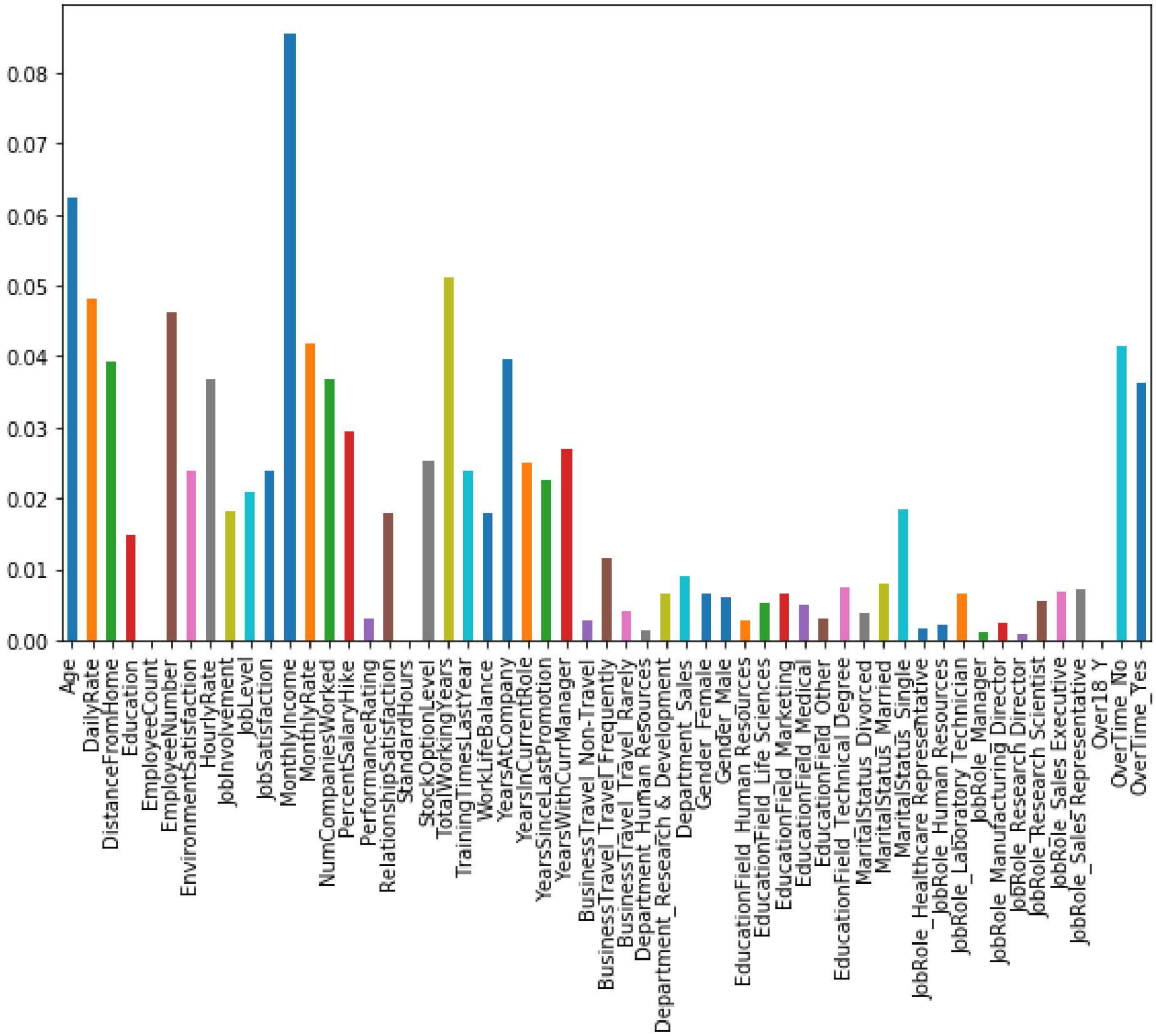
	Important Feature
Age	0.062232
DailyRate	0.047999
DistanceFromHome	0.039270
Education	0.014871
EmployeeCount	0.000000
EmployeeNumber	0.046100
EnvironmentSatisfaction	0.023897
HourlyRate	0.036702
JobInvolvement	0.018098
JobLevel	0.020750
JobSatisfaction	0.023820
MonthlyIncome	0.085372
MonthlyRate	0.041645
NumCompaniesWorked	0.036902
PercentSalaryHike	0.029411
PerformanceRating	0.002948
RelationshipSatisfaction	0.017733
StandardHours	0.000000
StockOptionLevel	0.025366
TotalWorkingYears	0.051120
TrainingTimesLastYear	0.024019
WorkLifeBalance	0.017760
YearsAtCompany	0.039561
YearsInCurrentRole	0.025094
YearsSinceLastPromotion	0.022459
YearsWithCurrManager	0.027046
BusinessTravel_Non-Travel	0.002817
BusinessTravel_Travel_Frequently	0.011583
BusinessTravel_Travel_Rarely	0.004181
Department_Human Resources	0.001234
Department_Research & Development	0.006574
Department_Sales	0.008941
Gender_Female	0.006491

	Important Feature
Gender_Male	0.006026
EducationField_Human Resources	0.002634
EducationField_Life Sciences	0.005215
EducationField_Marketing	0.006695
EducationField_Medical	0.004935
EducationField_Other	0.003089
EducationField_Technical Degree	0.007536
MaritalStatus_Divorced	0.003779
MaritalStatus_Married	0.007969
MaritalStatus_Single	0.018473
JobRole_Healthcare Representative	0.001725
JobRole_Human Resources	0.002067
JobRole_Laboratory Technician	0.006671
JobRole_Manager	0.000978
JobRole_Manufacturing Director	0.002338
JobRole_Research Director	0.000721
JobRole_Research Scientist	0.005396
JobRole_Sales Executive	0.006873
JobRole_Sales Representative	0.007161
Over18_Y	0.000000
OverTime_No	0.041484
OverTime_Yes	0.036238

In [37]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [45]:
```
fig = plt.figure(figsize = (10, 6))
pd.Series(RFC.feature_importances_, index=X_train.columns).plot.bar()
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x1fc9107f208>



Step : 8

In [54]:
```
estimator = RFC.estimators_[5]
```

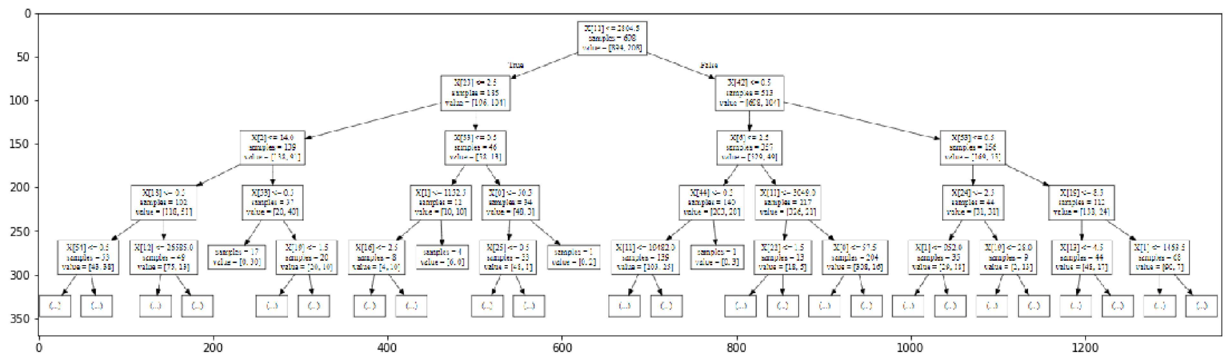
In [55]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
with open("RFDT.dot", 'w') as f:
    f = tree.export_graphviz(estimator, out_file=f, max_depth=4, impurity=False)
```

In [56]:
```
!dot - Tpng RFDT.dot -o RFDT.png
```

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

In [59]:
```
import matplotlib.pyplot as plt
image = plt.imread('Screenshot 2023-03-07 104946.png')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

Out[59]: <matplotlib.image.AxesImage at 0x1fc91423c88>



Step : 9

In [62]:
```
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
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\ensemble\forest.py:453: UserWarning: Some inputs do not have OOB
scores. This probably means too few trees were used to compute any reliable oob
estimates.
  warn("Some inputs do not have OOB scores. "
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\ensemble\forest.py:458: RuntimeWarning: invalid value encountered
in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\ensemble\forest.py:453: UserWarning: Some inputs do not have OOB
scores. This probably means too few trees were used to compute any reliable oob
estimates.
  warn("Some inputs do not have OOB scores. "
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\ensemble\forest.py:458: RuntimeWarning: invalid value encountered
in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
```

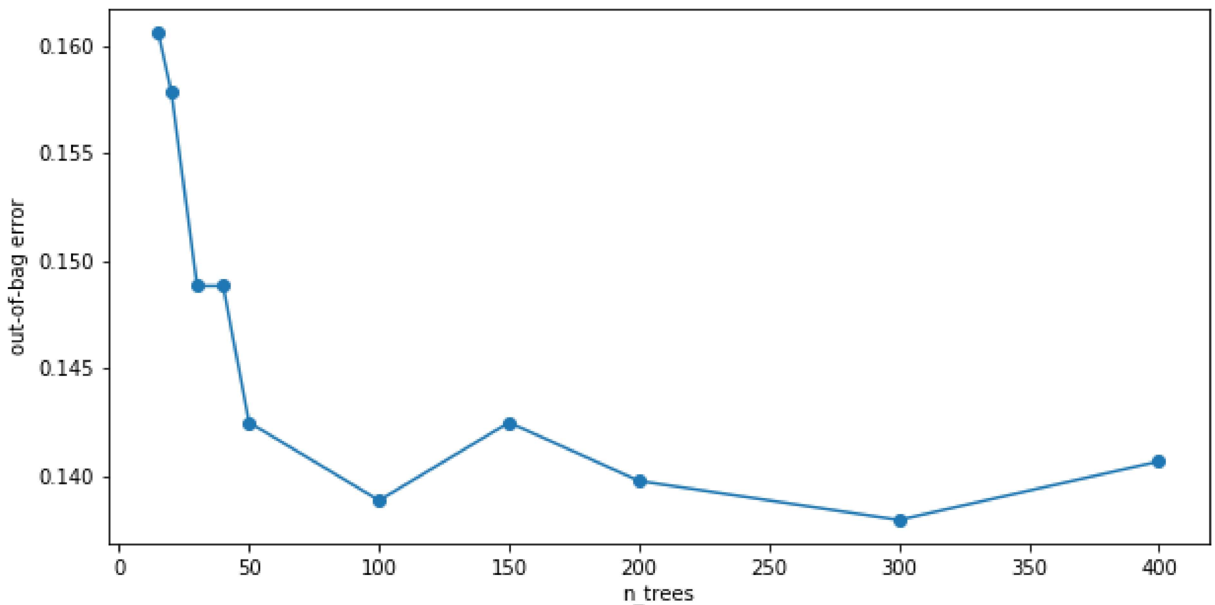
Out[62]:

	oob
n_trees	
15.0	0.160617
20.0	0.157895
30.0	0.148820
40.0	0.148820
50.0	0.142468
100.0	0.138838
150.0	0.142468
200.0	0.139746
300.0	0.137931
400.0	0.140653

Step : 10

In [63]:
```
ax = rf_oob_df.plot(legend=False, marker='o', figsize=(10,5))
ax.set(ylabel='out-of-bag error')
```

Out[63]:
```
[Text(0,0.5,'out-of-bag error')]
```



In [64]:
```
Step : 11
```

In [65]:
```
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[65]:
```
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 [66]:
```
y_pred1 = clf.predict(X_test)
y_pred1
```

Out[66]:
```
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, 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, 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, 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, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 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, 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, 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, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

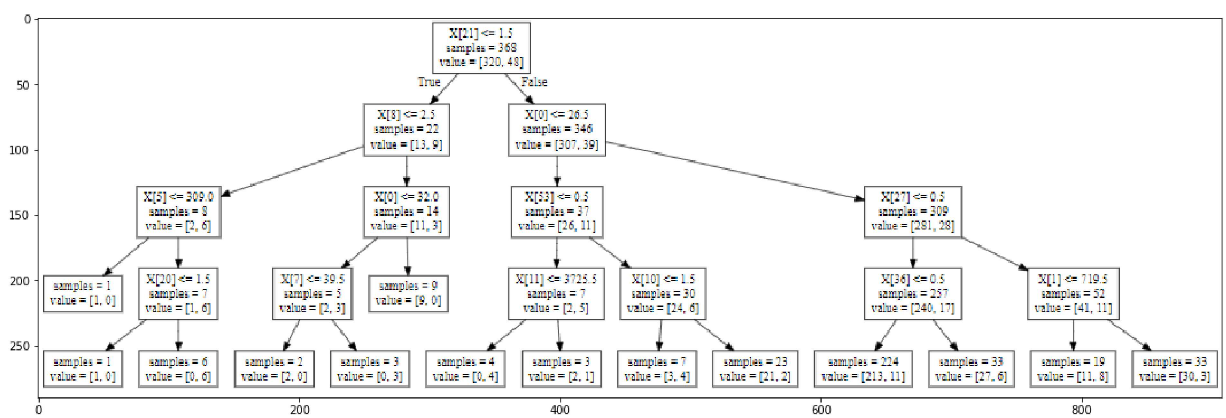
In [68]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
with open("DTC2.dot", 'w') as f:
    f = tree.export_graphviz(clf,out_file=f,max_depth = 4,impurity = False)
```

In [69]:
```
!dot -Tpng DTC2.dot -o DTC2.png
```

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

In [70]:
```
image = plt.imread('Screenshot 2023-03-07 111040.png')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

Out[70]: <matplotlib.image.AxesImage at 0x1fc916ce908>



In [71]:
```
print("Accuracy of test :",clf.score(X_test,y_test))
```

```
Accuracy of test : 0.907608695652174
```

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

```
             precision    recall  f1-score   support

          0       0.88      0.98      0.93       320
          1       0.44      0.08      0.14        48

avg / total       0.82      0.87      0.83       368
```

In [74]:
```
from sklearn.metrics import precision_score, recall_score, accuracy_score, roc_au
```

In [75]:
```
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 : 0.8668478260869565
RF Precision: 0.4444444444444444
RF Recall : 0.08333333333333333
RF F1 score : 0.14035087719298245


DT model : 0.907608695652174
DT Precision: 0.85
DT Recall : 0.3541666666666667
DT F1 score : 0.5
```

In [ ]: