PML_Lab-9_205229118_Mahalakshmi.S

May 27, 2021

0.0.1 Lab9. Employee Hopping Prediction using Random Forests

[16]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.metrics import precision_score,␣
↪recall_score,accuracy_score,roc_auc_score,classification_report,f1_score
from sklearn.tree import export_graphviz
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

0.0.2 Step1. [Understand Data].

Using Pandas, import "Employee_Hopping.csv" file and print properties such as head, shape, columns, dtype, info and value_counts.

[17]:
```
emp = pd.read_csv("Employee_Hopping.csv")
```

[18]:
```
emp.head()
```

[18]:
```
   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales
1   49        No  Travel_Frequently        279  Research & Development
2   37       Yes      Travel_Rarely       1373  Research & Development
3   33        No  Travel_Frequently       1392  Research & Development
4   27        No      Travel_Rarely        591  Research & Development

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

```
   …  RelationshipSatisfaction StandardHours  StockOptionLevel  \
0  …                         1            80                 0
1  …                         4            80                 1
2  …                         2            80                 0
3  …                         3            80                 0
4  …                         4            80                 1

   TotalWorkingYears  TrainingTimesLastYear WorkLifeBalance  YearsAtCompany  \
0                  8                      0               1               6
1                 10                      3               3              10
2                  7                      3               3               0
3                  8                      3               3               8
4                  6                      3               3               2

  YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
0                  4                        0                     5
1                  7                        1                     7
2                  0                        0                     0
3                  7                        3                     0
4                  2                        2                     2

[5 rows x 35 columns]
```

[19]:
```
emp.shape
```

[19]:
```
(1470, 35)
```

[20]:
```
emp.columns
```

[20]:
```
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')
```

[21]:
```
emp.dtypes
```

[21]:
```
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
```

[22]:
```
emp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
```

```
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

[23]:
```
emp.value_counts
```

[23]:
```
<bound method DataFrame.value_counts of       Age Attrition     BusinessTravel
DailyRate              Department  \
0      41       Yes      Travel_Rarely       1102                   Sales
1      49        No  Travel_Frequently        279  Research & Development
2      37       Yes      Travel_Rarely       1373  Research & Development
3      33        No  Travel_Frequently       1392  Research & Development
4      27        No      Travel_Rarely        591  Research & Development
…     …        …                 …        …                       …
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
…                 …         …            …             …
1465                23          2        Medical              1
1466                 6          1        Medical              1
1467                 4          3  Life Sciences              1
1468                 2          3        Medical              1
1469                 8          3        Medical              1

      EmployeeNumber  …  RelationshipSatisfaction StandardHours  \
0                  1  …                         1            80
1                  2  …                         4            80
2                  4  …                         2            80
3                  5  …                         3            80
4                  7  …                         4            80
…               …  …                         …            …
1465            2061  …                         3            80
1466            2062  …                         1            80
1467            2064  …                         2            80
1468            2065  …                         4            80
1469            2068  …                         1            80

      StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
0                    0                  8                      0
1                    1                 10                      3
2                    0                  7                      3
3                    0                  8                      3
4                    1                  6                      3
…                 …                …                     …
1465                 1                 17                      3
1466                 1                  9                      5
1467                 1                  6                      0
1468                 0                 17                      3
1469                 0                  6                      3

     WorkLifeBalance  YearsAtCompany YearsInCurrentRole  \
0                  1               6                  4
1                  3              10                  7
2                  3               0                  0
3                  3               8                  7
4                  3               2                  2
…               …             …                 …
1465               3               5                  2
1466               3               7                  7
```

```
1467               3              6                 2
1468               2              9                 6
1469               4              4                 3

      YearsSinceLastPromotion  YearsWithCurrManager
0                           0                     5
1                           1                     7
2                           0                     0
3                           3                     0
4                           2                     2
…                         …                     …
1465                        0                     3
1466                        1                     7
1467                        0                     3
1468                        0                     8
1469                        1                     2

[1470 rows x 35 columns]>
```

[24]:
```
emp.isnull().sum()
```

[24]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
```

```
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

0.0.3 Step2. [Extract X and y].

Create X and y columns from the dataframe

[25]:
```
X = emp.drop(['Attrition'],axis=1)
y = emp.Attrition
```

[26]:
```
y = y.apply(lambda x:1 if x == 'Yes' else 0)
```

[27]:
```
emp.select_dtypes(include=['object']).dtypes
```

[27]:
```
Attrition         object
BusinessTravel    object
Department        object
EducationField    object
Gender            object
JobRole           object
MaritalStatus     object
Over18            object
OverTime          object
dtype: object
```

0.0.4 Step3. [Feature Engineering]

There are 8 categorical columns (where dtype="object"). Perform one hot encoding and create new columns

[28]:
```
emp=pd.
↪get_dummies(emp,columns=["BusinessTravel","Department",'EducationField',"Gender","JobRole",
emp.head()
```

[28]:
```
   Age Attrition  DailyRate  DistanceFromHome  Education  EmployeeCount  \
0   41       Yes       1102                 1          2              1
1   49        No        279                 8          1              1
2   37       Yes       1373                 2          2              1
3   33        No       1392                 3          4              1
4   27        No        591                 2          1              1

   EmployeeNumber  EnvironmentSatisfaction  HourlyRate  JobInvolvement  …  \
```

```
0                1                        2           94                3  ...
1                2                        3           61                2  ...
2                4                        4           92                2  ...
3                5                        4           56                3  ...
4                7                        1           40                3  ...

   JobRole_Research Director  JobRole_Research Scientist  \
0                          0                           0
1                          0                           1
2                          0                           0
3                          0                           1
4                          0                           0

   JobRole_Sales Executive  JobRole_Sales Representative  \
0                        1                             0
1                        0                             0
2                        0                             0
3                        0                             0
4                        0                             0

   MaritalStatus_Divorced  MaritalStatus_Married  MaritalStatus_Single  \
0                       0                      0                     1
1                       0                      1                     0
2                       0                      0                     1
3                       0                      1                     0
4                       0                      1                     0

   Over18_Y  OverTime_No  OverTime_Yes
0         1            0             1
1         1            1             0
2         1            0             1
3         1            0             1
4         1            1             0

[5 rows x 56 columns]
```

0.0.5 Step4. Now, check shape of X and y.

[29]:
```
X = emp.drop(['Attrition'],axis=1)
X.shape
```

[29]:
```
(1470, 55)
```

[30]:
```
y.shape
```

[30]:
```
(1470,)
```

0.0.6 Step5. [Model Development]

Split X and y for training and testing

[31]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪3,random_state=0)
```

Create RandomForestClassifier model, fit (no need to scale) and predict

[32]:
```
seed = 0
rfc = RandomForestClassifier(n_estimators=1000, max_features=0.3, max_depth=4,␣
↪min_samples_leaf=2, n_jobs=-1, random_state=seed ,warm_start=True, verbose=0)
```

[33]:
```
rfc.fit(X_train,y_train)
```

[33]:
```
RandomForestClassifier(max_depth=4, max_features=0.3, min_samples_leaf=2,
                       n_estimators=1000, n_jobs=-1, random_state=0,
                       warm_start=True)
```

[34]:
```
y_pred=rfc.predict(X_test)
y_pred
```

[34]:
```
array([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, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 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, 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, 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, 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, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 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, 1, 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], dtype=int64)
```

0.0.7 Step6. [Testing]

Print accuracy score between y_test and y_pred

[35]:
```
accuracy_score(y_test,y_pred)
```

[35]:
```
0.8639455782312925
```

Print classification report between y_test and y_pred and observe the results

[36]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       371
           1       0.86      0.17      0.29        70

    accuracy                           0.86       441
   macro avg       0.86      0.58      0.61       441
weighted avg       0.86      0.86      0.82       441
```

0.0.8 Step7. [Feature importance value]

You can look at feature importance values using the property, rf.feature_importances_

[37]:
```
print(rfc.feature_importances_)
```

```
[6.98321450e-02 3.65227658e-02 2.00159132e-02 5.21457627e-03
 0.00000000e+00 1.93136701e-02 3.09477993e-02 2.51635390e-02
 1.66715720e-02 5.04884918e-02 1.37024559e-02 1.15212766e-01
 1.89715997e-02 1.83688986e-02 1.16183893e-02 6.52783762e-04
 9.05995065e-03 0.00000000e+00 2.83057741e-02 6.73115246e-02
 5.96985891e-03 1.88628396e-02 5.02462199e-02 1.64563675e-02
 8.78154018e-03 4.70126629e-02 3.99898213e-03 1.62801031e-02
 2.47672036e-03 5.60666617e-04 3.92298011e-03 4.67197125e-03
 2.85590037e-03 1.92092323e-03 3.48874178e-03 2.75324633e-03
 3.67258786e-04 4.33772973e-03 1.65455825e-03 1.66961888e-03
 3.66392947e-04 7.05642476e-04 4.35437163e-03 3.37746500e-04
 1.17707045e-03 6.70465871e-05 4.98737913e-03 6.52221544e-03
 1.25986442e-02 2.39045147e-03 3.31770313e-03 2.24531725e-02
 0.00000000e+00 9.21168370e-02 9.29418217e-02]
```

Print feature name and its rf.feature_importances_ values and understand important features

[38]:
```
fea_imp = pd.DataFrame(rfc.feature_importances_,index=X_train.
 ↪columns,columns=['Important score']).sort_values(by='Important␣
 ↪score',ascending=False)
```

[39]:
```
from operator import itemgetter

x = fea_imp.index
y = fea_imp['Important score']

lst = []
print("Feature Name - Feature Importance Score")
print("__________________________________________")
```

```python
for i in range(55):
    lst.append((x[i], y[i]))
sorted(lst,key=itemgetter(1))
FIS = lst[:]
FIS
```

```
Feature Name - Feature Importance Score
-----------------------------------------
```

[39]:

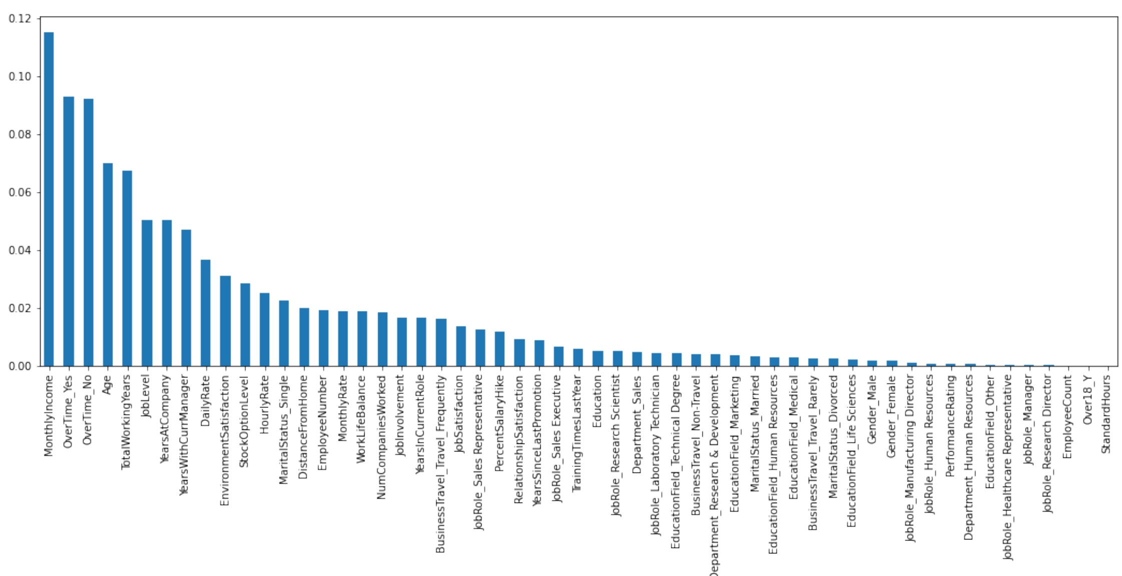
```
[('MonthlyIncome', 0.11521276580234263),
 ('OverTime_Yes', 0.09294182174617778),
 ('OverTime_No', 0.09211683703047299),
 ('Age', 0.06983214503413517),
 ('TotalWorkingYears', 0.06731152457797149),
 ('JobLevel', 0.050488491822907974),
 ('YearsAtCompany', 0.05024621988832041),
 ('YearsWithCurrManager', 0.04701266289517826),
 ('DailyRate', 0.036522765839450674),
 ('EnvironmentSatisfaction', 0.030947799266825053),
 ('StockOptionLevel', 0.028305774120963006),
 ('HourlyRate', 0.025163538961806283),
 ('MaritalStatus_Single', 0.022453172508509013),
 ('DistanceFromHome', 0.02001591318605209),
 ('EmployeeNumber', 0.019313670070972264),
 ('MonthlyRate', 0.01897159972501488),
 ('WorkLifeBalance', 0.018862839595341577),
 ('NumCompaniesWorked', 0.018368898619670174),
 ('JobInvolvement', 0.016671571984204086),
 ('YearsInCurrentRole', 0.016456367461815606),
 ('BusinessTravel_Travel_Frequently', 0.016280103058574972),
 ('JobSatisfaction', 0.013702455921311134),
 ('JobRole_Sales Representative', 0.012598644206356679),
 ('PercentSalaryHike', 0.011618389303011392),
 ('RelationshipSatisfaction', 0.009059950651190586),
 ('YearsSinceLastPromotion', 0.008781540180973284),
 ('JobRole_Sales Executive', 0.006522215444916321),
 ('TrainingTimesLastYear', 0.005969858907921l235),
 ('Education', 0.005214576269184216),
 ('JobRole_Research Scientist', 0.004987379130407648),
 ('Department_Sales', 0.004671971253536818),
 ('JobRole_Laboratory Technician', 0.004354371633245582),
 ('EducationField_Technical Degree', 0.00433772972987835),
 ('BusinessTravel_Non-Travel', 0.003998982125104187),
 ('Department_Research & Development', 0.003922980113577673),
 ('EducationField_Marketing', 0.003488741780169301),
 ('MaritalStatus_Married', 0.003317703126868009),
 ('EducationField_Human Resources', 0.002855900366990764),
```

```
('EducationField_Medical', 0.0027532463334835272),
('BusinessTravel_Travel_Rarely', 0.002476720364451602),
('MaritalStatus_Divorced', 0.0023904514748599582),
('EducationField_Life Sciences', 0.0019209232322679954),
('Gender_Male', 0.0016696188807234103),
('Gender_Female', 0.0016545582499910018),
('JobRole_Manufacturing Director', 0.0011770704472686203),
('JobRole_Human Resources', 0.0007056424757231417),
('PerformanceRating', 0.0006527837623514303),
('Department_Human Resources', 0.0005606666173878862),
('EducationField_Other', 0.0003672587863643348 4),
('JobRole_Healthcare Representative', 0.0003663929467656666),
('JobRole_Manager', 0.0003377464999189519),
('JobRole_Research Director', 6.704658709318701e-05),
('Over18_Y', 0.0),
('StandardHours', 0.0),
('EmployeeCount', 0.0)]
```

Show a Bar plot between feature column names and feature_importances_ score.

[40]:
```
pd.Series(rfc.feature_importances_, index=X_train.columns).
↪sort_values(ascending=False).plot(kind='bar', figsize=(18,6))
```

[40]: <AxesSubplot:>



0.0.9 Step8. [Visualize your RF Decision Tree using graphviz]

http://www.webgraphviz.com/.

[41]:
```
estim = rfc.estimators_[5]
```

[42]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
with open("RFDT.dot", 'w') as f:
    f = tree.export_graphviz(estim, out_file=f, max_depth=4, impurity=False,␣
↪feature_names=X_train.columns.values, class_names=['yes','no'],␣
↪rounded=True, proportion=False, precision=2, filled= True)
```

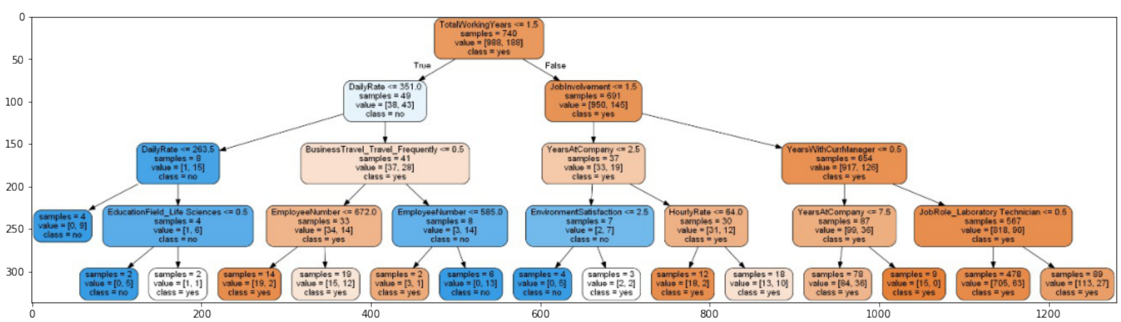
[43]:
```
!dot -Tjpg RFDT.dot -o RF.jpg
```

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

[44]:
```
import matplotlib.pyplot as plt

image = plt.imread('RF.jpg')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

[44]: <matplotlib.image.AxesImage at 0x22fd5fe2f70>



0.0.10 Step9. [RF with a range of trees]

Fit random forest models with a range of tree numbers [15, 20, 30, 40, 50, 100, 150, 200, 300, 400] and print Out-Of-Bag error for each of these model. Use model.oob_score_ to get score and subtract this score from 1 to get the oob-error. That is, oob-error = 1 - model.oob_score_. Hint: since the only thing changing is the number of trees, the `warm_start` flag can be used so that the model just adds more trees to the existing model each time. Use the `set_params` method to update the number of trees. The following code many help to understand this part.

[45]:
```
rf3 =␣
↪RandomForestClassifier(oob_score=True,random_state=42,warm_start=True,n_jobs=-1)
oob_list = list()
# Iterate through all of the possibilities for number of trees
```

```
for n_trees in [15, 20, 30, 40, 50, 100, 150, 200, 300, 400]:
    rf3.set_params(n_estimators=n_trees)
    rf3.fit(X_train, y_train)

    # Get the oob error

    oob_error = 1 - rf3.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
```

[45]:

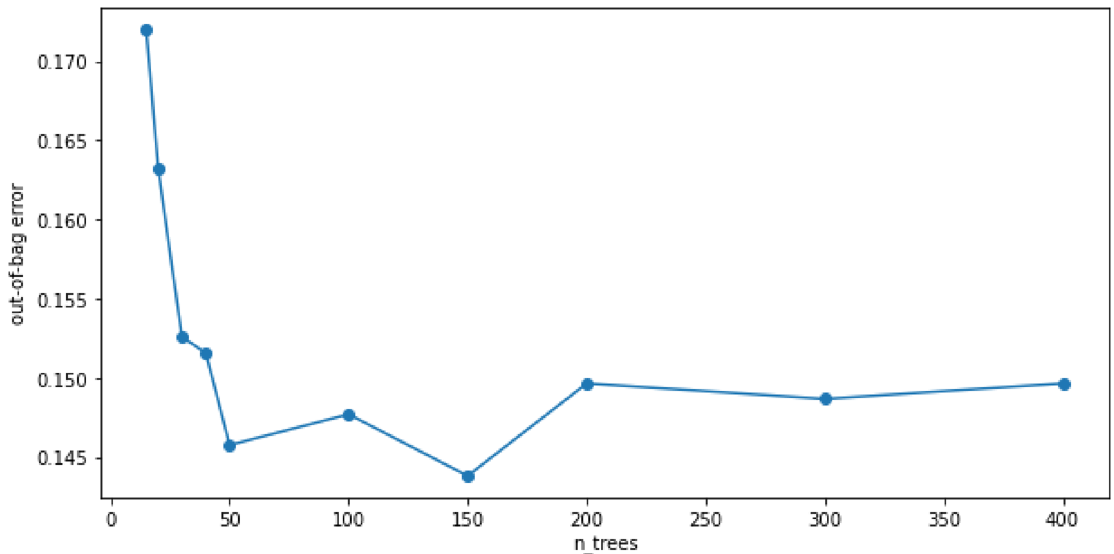
n_trees	oob
15.0	0.172012
20.0	0.163265
30.0	0.152575
40.0	0.151603
50.0	0.145773
100.0	0.147716
150.0	0.143829
200.0	0.149660
300.0	0.148688
400.0	0.149660

0.0.11 Step10. [Plot oob-error for each tree]

The following lines will help you

[46]:
```
ax = rf_oob_df.plot(legend=False, marker='o', figsize=(10,5))
ax.set(ylabel='out-of-bag error')
```

[46]:
```
[Text(0, 0.5, 'out-of-bag error')]
```



0.0.12 Step11. [Compare with DecisionTreeClassifier]

Create DecisionTreeClassifier, fit and predict on test set

[47]:
```
clf2 = DecisionTreeClassifier(criterion='gini',max_depth=4, random_state=42)
```

[48]:
```
clf2.fit(X_train,y_train)
```

[48]:
```
DecisionTreeClassifier(max_depth=4, random_state=42)
```

[49]:
```
y_predict=clf2.predict(X_test)
y_predict
```

[49]:
```
array([0, 0, 0, 0, 1, 1, 1, 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, 1, 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, 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, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 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, 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, 1, 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, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       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, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 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, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 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, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0], dtype=int64)
```

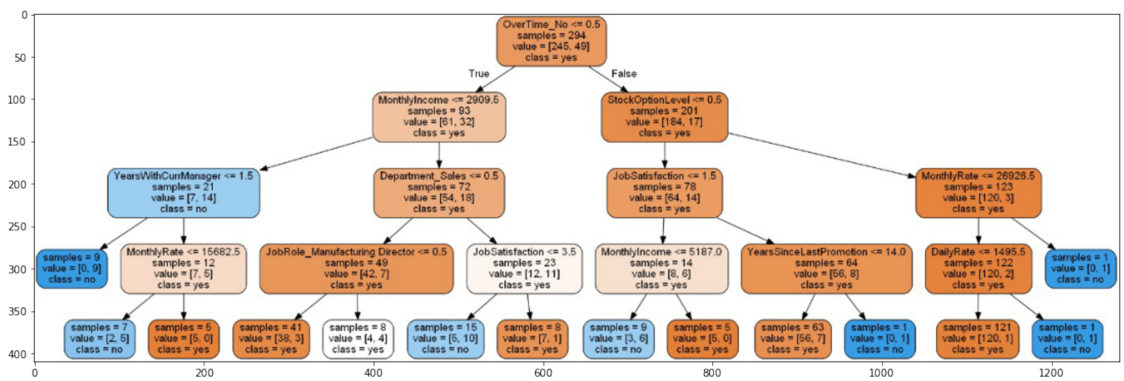
Visualize the tree using graphviz

[50]:
```
!dot -Tjpg DTC2.dot -o DT.jpg
```

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

[51]:
```
image = plt.imread('DT.jpg')
plt.figure(figsize=(19,15))
plt.imshow(image)
```

[51]:
```
<matplotlib.image.AxesImage at 0x22fd5d310d0>
```



http://www.webgraphviz.com/.

Print accuracy score

[52]:
```
accuracy_score(y_test,y_predict)
```

[52]:
```
0.8480725623582767
```

Print classification report

[53]:
```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.89      0.94      0.91       371
           1       0.53      0.39      0.45        70

    accuracy                           0.85       441
```

```
   macro avg       0.71      0.66      0.68       441
weighted avg       0.83      0.85      0.84       441
```

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?

[54]:
```
print("RF model:      ",accuracy_score(y_test,y_pred))
print("RF Precision:  ",precision_score(y_test,y_pred))
print("RF Recall:     ",recall_score(y_test,y_pred))
print("RF F1 score:   ",f1_score(y_test,y_pred))
print("\n")
print("DT model:      ",accuracy_score(y_test,y_predict))
print("DT Precision:  ",precision_score(y_test,y_predict))
print("DT Recall:     ",recall_score(y_test,y_predict))
print("DT F1 score:   ",f1_score(y_test,y_predict))
```

```
RF model:       0.8639455782312925
RF Precision:   0.8571428571428571
RF Recall:      0.17142857142857143
RF F1 score:    0.2857142857142857


DT model:       0.8480725623582767
DT Precision:   0.5294117647058824
DT Recall:      0.38571428571428573
DT F1 score:    0.4462809917355372
```