lab 9

In [1]:
```
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_
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')
```

Step1. [Understand Data].

In [2]:
```
df = pd.read_csv("Employee_Hopping.csv")
```

In [3]:
```
df.head()
```

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 rows × 35 columns

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

Out[4]: (1470, 35)

In [5]:
```
df.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]:
```
df.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]:
```
df.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
```

In [8]:
```
df.value_counts
```

Out[8]:
```
<bound method DataFrame.value_counts of       Age Attrition     BusinessTrave
l  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
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
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
```

Step2. [Extract X and y].

In [10]:
```
X = df.drop(['Attrition'],axis=1)
y = df.Attrition
```

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

In [12]:
```
df.select_dtypes(include=['object']).dtypes
```

Out[12]:
```
Attrition         object
BusinessTravel    object
Department        object
EducationField    object
Gender            object
JobRole           object
MaritalStatus     object
Over18            object
OverTime          object
dtype: object
```

Step3. [Feature Engineering]

In [13]:
```
df=pd.get_dummies(df,columns=["BusinessTravel","Department",'EducationField',"Ger
df.head()
```

Out[13]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	En
0	41	Yes	1102	1	2	1	1	
1	49	No	279	8	1	1	2	
2	37	Yes	1373	2	2	1	4	
3	33	No	1392	3	4	1	5	
4	27	No	591	2	1	1	7	

5 rows × 56 columns

Step4. Now, check shape of X and y.

In [14]:
```
X = df.drop(['Attrition'],axis=1)
X.shape
```

Out[14]:
```
(1470, 55)
```

In [15]:
```
y.shape
```

Out[15]:
```
(1470,)
```

Step5. [Model Development]

In [16]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=(
```

In [17]:
```
seed = 0
rfc = RandomForestClassifier(n_estimators=1000, max_features=0.3, max_depth=4, mi
```

In [18]:
```
rfc.fit(X_train,y_train)
```

Out[18]:
```
RandomForestClassifier(max_depth=4, max_features=0.3, min_samples_leaf=2,
                       n_estimators=1000, n_jobs=-1, random_state=0,
                       warm_start=True)
```

In [19]:
```
y_pred=rfc.predict(X_test)
y_pred
```

Out[19]:
```
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)
```

Step6. [Testing]

In [20]:
```
accuracy_score(y_test,y_pred)
```

Out[20]:
```
0.8639455782312925
```

In [21]:
```
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
```

Step7. [Feature importance value]

In [22]:
```
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]
```

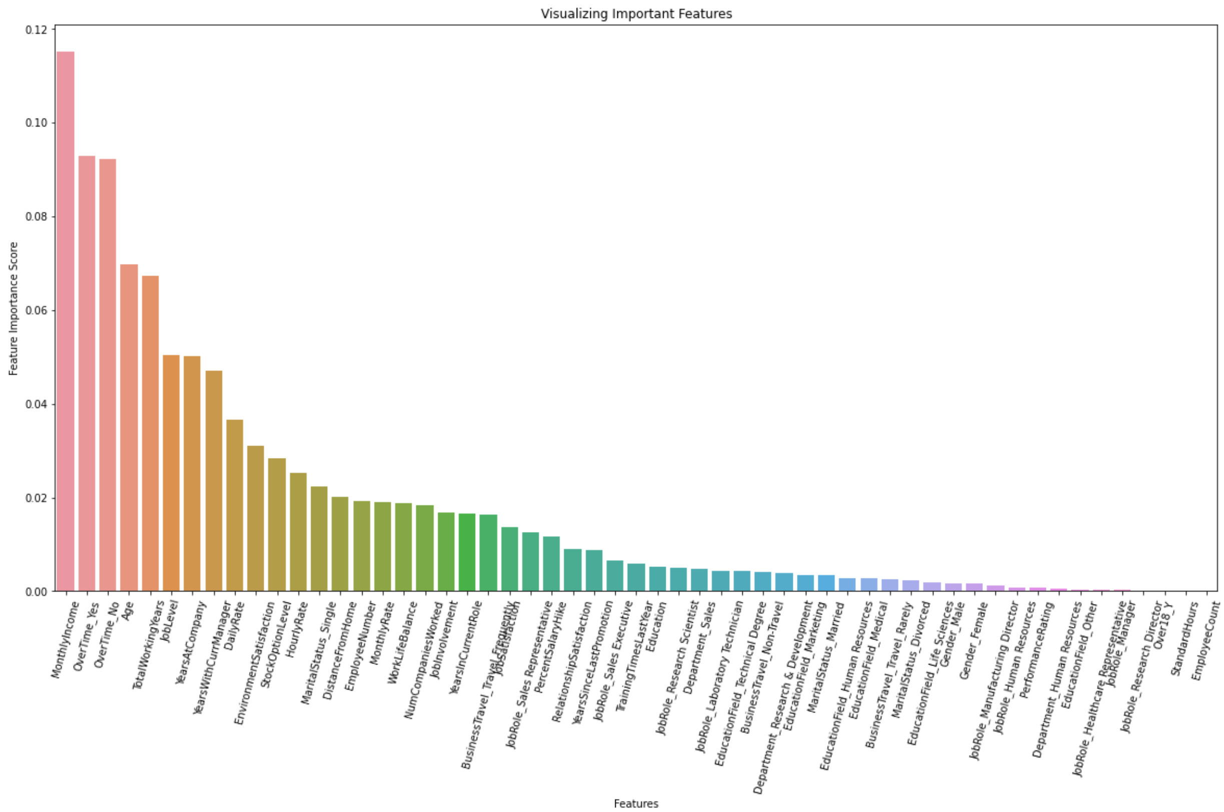
In [23]:
```
feature_imp = pd.DataFrame(rfc.feature_importances_,index=X_train.columns,columns
feature_imp
```

Out[23]:

	Important score
MonthlyIncome	0.115213
OverTime_Yes	0.092942
OverTime_No	0.092117
Age	0.069832
TotalWorkingYears	0.067312
JobLevel	0.050488
YearsAtCompany	0.050246
YearsWithCurrManager	0.047013
DailyRate	0.036523
EnvironmentSatisfaction	0.030948
StockOptionLevel	0.028306
HourlyRate	0.025164
MaritalStatus_Single	0.022453
DistanceFromHome	0.020016
EmployeeNumber	0.019314
MonthlyRate	0.018972
WorkLifeBalance	0.018863
NumCompaniesWorked	0.018369
JobInvolvement	0.016672
YearsInCurrentRole	0.016456
BusinessTravel_Travel_Frequently	0.016280
JobSatisfaction	0.013702
JobRole_Sales Representative	0.012599
PercentSalaryHike	0.011618
RelationshipSatisfaction	0.009060
YearsSinceLastPromotion	0.008782
JobRole_Sales Executive	0.006522
TrainingTimesLastYear	0.005970
Education	0.005215
JobRole_Research Scientist	0.004987
Department_Sales	0.004672
JobRole_Laboratory Technician	0.004354
EducationField_Technical Degree	0.004338

	Important score
BusinessTravel_Non-Travel	0.003999
Department_Research & Development	0.003923
EducationField_Marketing	0.003489
MaritalStatus_Married	0.003318
EducationField_Human Resources	0.002856
EducationField_Medical	0.002753
BusinessTravel_Travel_Rarely	0.002477
MaritalStatus_Divorced	0.002390
EducationField_Life Sciences	0.001921
Gender_Male	0.001670
Gender_Female	0.001655
JobRole_Manufacturing Director	0.001177
JobRole_Human Resources	0.000706
PerformanceRating	0.000653
Department_Human Resources	0.000561
EducationField_Other	0.000367
JobRole_Healthcare Representative	0.000366
JobRole_Manager	0.000338
JobRole_Research Director	0.000067
Over18_Y	0.000000
StandardHours	0.000000
EmployeeCount	0.000000

In [24]:
```
plt.figure(figsize=(20,10))
sns.barplot(x=feature_imp.index, y=feature_imp['Important score'])
# Add labels to your graph

plt.ylabel('Feature Importance Score')
plt.xlabel('Features')
plt.title("Visualizing Important Features")
plt.xticks(rotation=75)
plt.show()
```



Step8. [Visualize your RF Decision Tree using graphviz]

http://www.webgraphviz.com/ (http://www.webgraphviz.com/).

In [25]:
```
estimator = rfc.estimators_[5]
```

In [36]:
```
export_graphviz(estimator, out_file='rtree1.txt',feature_names = X_train.columns.
```

Step9. [RF with a range of trees]

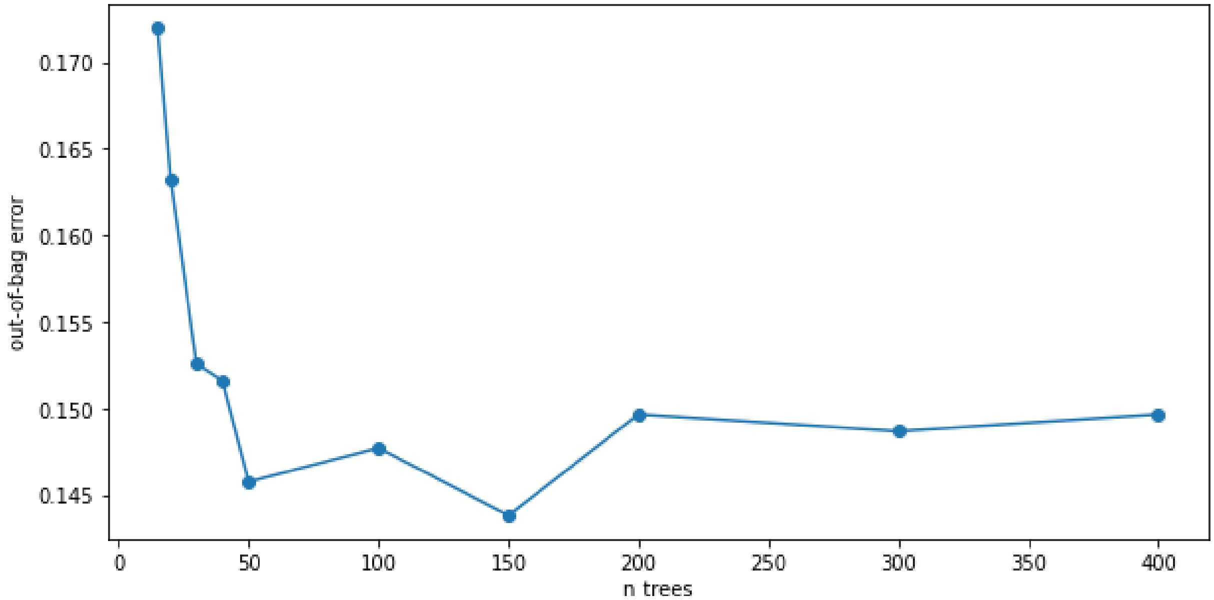
In [27]:
```
rf2 = RandomForestClassifier(oob_score=True,random_state=42,warm_start=True,n_job
oob_list = list()
# Iterate through all of the possibilities for number of trees

for n_trees in [15, 20, 30, 40, 50, 100, 150, 200, 300, 400]:
    rf2.set_params(n_estimators=n_trees)
    rf2.fit(X_train, y_train)

    # Get the oob error

    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[27]:

	oob
n_trees	
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

In [28]:
```
ax = rf_oob_df.plot(legend=False, marker='o', figsize=(10,5))
ax.set(ylabel='out-of-bag error')
```

Out[28]:
```
[Text(0, 0.5, 'out-of-bag error')]
```



Step11. [Compare with DecisionTreeClassifier]

In [29]:
```
clf_1 = DecisionTreeClassifier(criterion='gini',max_depth=4, random_state=42)
```

In [30]:
```
clf_1.fit(X_train,y_train)
```

Out[30]:
```
DecisionTreeClassifier(max_depth=4, random_state=42)
```

In [31]:
```
y_pred1=clf_1.predict(X_test)
y_pred1
```

Out[31]:
```
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)
```

In [32]:
```
with open("rtree2.txt", 'w') as f:
    f = tree.export_graphviz(clf_1,out_file=f,max_depth = 4,impurity = False,feat
```

[http://www.webgraphviz.com/ (http://www.webgraphviz.com/)](http://www.webgraphviz.com/).

In [33]:
```
accuracy_score(y_test,y_pred1)
```

Out[33]:
```
0.8480725623582767
```

In [34]:
```
print(classification_report(y_test,y_pred1))
```

```
              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
```

In [35]:
```
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_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:       0.8639455782312925
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
```