SURUTHI S

225229141

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

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

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

Out[3]:

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

5 rows × 35 columns

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

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

In [5]:
```
1 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
```

Value counts for Categotical Variables

In [6]:

```
for cols in df.select_dtypes(include='object').columns:
    print()
    print(cols)
    print(df[cols].value_counts())
```

```
Attrition
No     1233
Yes     237
Name: Attrition, dtype: int64

BusinessTravel
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64

Department
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64

EducationField
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64

Gender
Male      882
Female    588
Name: Gender, dtype: int64

JobRole
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64

MaritalStatus
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64

Over18
Y    1470
Name: Over18, dtype: int64

OverTime
No     1054
Yes     416
Name: OverTime, dtype: int64
```

In [7]:
```
X = df.drop('Attrition',axis=1)
```

In [8]:
```
X
```

Out[8]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Emp
0	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	
1	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	
2	37	Travel_Rarely	1373	Research & Development	2	2	Other	
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	
4	27	Travel_Rarely	591	Research & Development	2	1	Medical	
...	...	...	...	...	...	...	...	
1465	36	Travel_Frequently	884	Research & Development	23	2	Medical	
1466	39	Travel_Rarely	613	Research & Development	6	1	Medical	
1467	27	Travel_Rarely	155	Research & Development	4	3	Life Sciences	
1468	49	Travel_Frequently	1023	Sales	2	3	Medical	
1469	34	Travel_Rarely	628	Research & Development	8	3	Medical	

1470 rows × 34 columns

In [9]:
```
y = df['Attrition'].map({"Yes":1,"No":0})
```

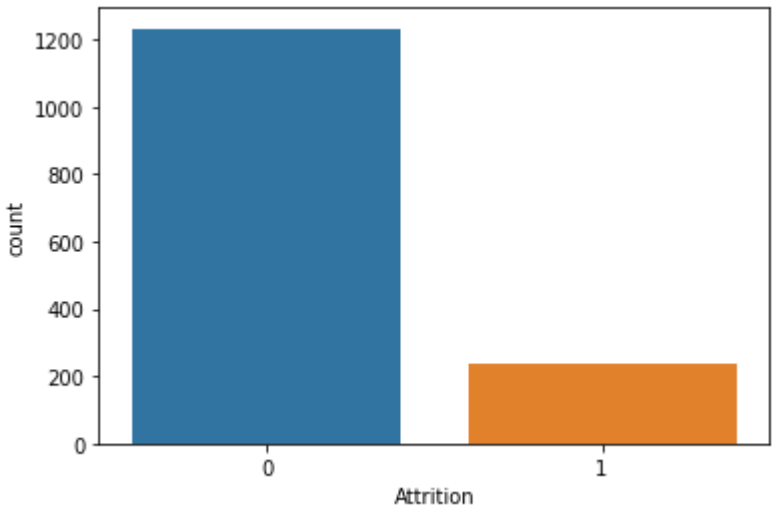
In [10]:
```
y
```

Out[10]:
```
0       1
1       0
2       1
3       0
4       0
       ..
1465    0
1466    0
1467    0
1468    0
1469    0
Name: Attrition, Length: 1470, dtype: int64
```

In [11]:
```
import seaborn as sns
```

In [12]:
```
1 sns.countplot(y)
```

Out[12]: <AxesSubplot:xlabel='Attrition', ylabel='count'>



In [13]:
```
1 from sklearn.preprocessing import OneHotEncoder
```

https://pythonsimplified.com/difference-between-onehotencoder-and-get_dummies/ (https://pythonsimplified.com/difference-between-onehotencoder-and-get_dummies/)

For machine learning, you almost definitely want to use sklearn.OneHotEncoder. For other tasks like simple analyses, you might be able to use pd.get_dummies, which is a bit more convenient.

https://stats.stackexchange.com/questions/224051/one-hot-vs-dummy-encoding-in-scikit-learn#:~:text=One-hot%20encoding%20converts%20it%20into%20n%20variables%2C%20while,while%20dummy%20encoding%20converts%20it%20into%20n%20variables%2C%20while,while%20dummy%20encoding%20converts%20it%20into%20n%20variables) (https://stats.stackexchange.com/questions/224051/one-hot-vs-dummy-encoding-in-scikit-learn#:~:text=One-hot%20encoding%20converts%20it%20into%20n%20variables%2C%20while,while%20dummy%20encoding%20converts%20it%20into%20n%20variables).

https://stackoverflow.com/questions/36631163/what-are-the-pros-and-cons-between-get-dummies-pandas-and-onehotencoder-sciki (https://stackoverflow.com/questions/36631163/what-are-the-pros-and-cons-between-get-dummies-pandas-and-onehotencoder-sciki)

In [14]:
```
1 cols = X.select_dtypes(include='object').columns
2 col_list = list(cols)
3 col_list
```

Out[14]:
```
['BusinessTravel',
 'Department',
 'EducationField',
 'Gender',
 'JobRole',
 'MaritalStatus',
 'Over18',
 'OverTime']
```

In [15]:
```
1 len(X.select_dtypes(include='object').columns)
```

Out[15]: 8

In [16]:
```
X = pd.get_dummies(X,columns = col_list)
```

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

Out[19]: (1470, 55)

In [24]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

In [46]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.40,random_state
```

In [47]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [48]:
```
RFC = RandomForestClassifier()
RFC.fit(X_train,y_train)
ypred_test = RFC.predict(X_test)
y_pred_train = RFC.predict(X_train)

print(classification_report(y_test,ypred_test))


```

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       507
           1       0.87      0.16      0.27        81

    accuracy                           0.88       588
   macro avg       0.87      0.58      0.60       588
weighted avg       0.88      0.88      0.84       588
```

In [49]:
```
RFC.feature_importances_
```

Out[49]:
```
array([0.05472128, 0.04709325, 0.0439671 , 0.01783273, 0.        ,
       0.04795256, 0.02468413, 0.04150197, 0.02052876, 0.01870739,
       0.0194101 , 0.06245323, 0.0381813 , 0.03637125, 0.02771901,
       0.00378677, 0.02158229, 0.        , 0.02544158, 0.04333003,
       0.02543695, 0.0187112 , 0.04057669, 0.03113869, 0.0192276 ,
       0.03191836, 0.00283148, 0.01215663, 0.00636687, 0.00262162,
       0.00929094, 0.00811654, 0.00282502, 0.00831918, 0.00540129,
       0.0079989 , 0.0034544 , 0.00859079, 0.00783926, 0.0082484 ,
       0.00170952, 0.00267911, 0.00783848, 0.00144455, 0.00253028,
       0.00045742, 0.00521418, 0.00796846, 0.00695907, 0.00457047,
       0.00860646, 0.02640622, 0.        , 0.03480691, 0.03247331])
```

In [113]:
```
1 X_train.columns
```

Out[113]:
```
Index(['Age', 'DailyRate', 'DistanceFromHome', 'Education', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobSatisfaction', 'MonthlyIncome',
       'MonthlyRate', 'NumCompaniesWorked', 'PercentSalaryHike',
       'PerformanceRating', 'RelationshipSatisfaction', 'StandardHours',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager',
       'BusinessTravel_Non-Travel', 'BusinessTravel_Travel_Frequently',
       'BusinessTravel_Travel_Rarely', 'Department_Human Resources',
       'Department_Research & Development', 'Department_Sales',
       'EducationField_Human Resources', 'EducationField_Life Sciences',
       'EducationField_Marketing', 'EducationField_Medical',
       'EducationField_Other', 'EducationField_Technical Degree',
       'Gender_Female', 'Gender_Male', 'JobRole_Healthcare Representative',
       'JobRole_Human Resources', 'JobRole_Laboratory Technician',
       'JobRole_Manager', 'JobRole_Manufacturing Director',
       'JobRole_Research Director', 'JobRole_Research Scientist',
       'JobRole_Sales Executive', 'JobRole_Sales Representative',
       'MaritalStatus_Divorced', 'MaritalStatus_Married',
       'MaritalStatus_Single', 'Over18_Y', 'OverTime_No', 'OverTime_Yes'],
      dtype='object')
```

In [55]:
```
1 RFC.estimators_
```

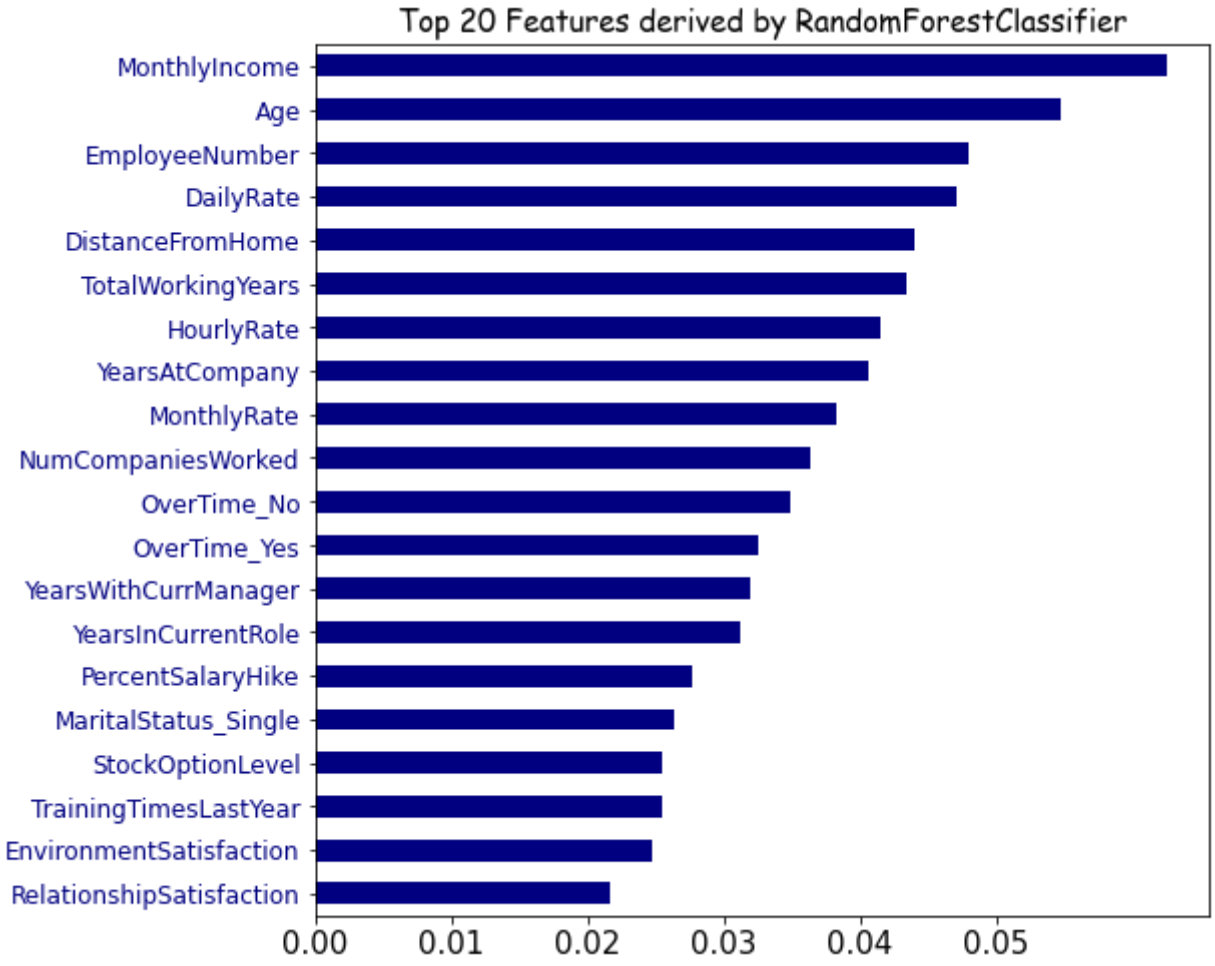
...

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

In [108]:
```
1  def plot(model,n):
2
3      (pd.Series(model.feature_importances_, index=X_train.columns)
4      .nlargest(n)
5      .plot(kind='barh', figsize=[8, n/2.5],color='navy')
6      .invert_yaxis())    # most important feature is on top, ie, descending order
7
8
9      ticks_x = np.linspace(0, 0.05, 6)   # (start, end, number of ticks)
10     plt.xticks(ticks_x, fontsize=15, color='black')
11     plt.yticks(size=12, color='navy' )
12     plt.title(f'Top {n} Features derived by RandomForestClassifier', family='fan
13     print(list((pd.Series(model.feature_importances_, index=X.columns).nlargest(
14
15
```

In [109]:
```
plot(RFC,20)
```

```
['MonthlyIncome', 'Age', 'EmployeeNumber', 'DailyRate', 'DistanceFromHome', 'TotalW
orkingYears', 'HourlyRate', 'YearsAtCompany', 'MonthlyRate', 'NumCompaniesWorked',
'OverTime_No', 'OverTime_Yes', 'YearsWithCurrManager', 'YearsInCurrentRole', 'Perce
ntSalaryHike', 'MaritalStatus_Single', 'StockOptionLevel', 'TrainingTimesLastYear',
'EnvironmentSatisfaction', 'RelationshipSatisfaction']
```



In [111]:
```
! pip install graphviz
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: graphviz in c:\users\1mscdsa41\appdata\roaming\pytho
n\python36\site-packages (0.19.1)
```

In [118]:
```
from sklearn import tree
from sklearn.tree import export_graphviz
```

In [117]:
```
feature_names = list(X_train.columns)
feature_names
```

...

In [121]:
```
estimator = RFC.estimators_[5]
```

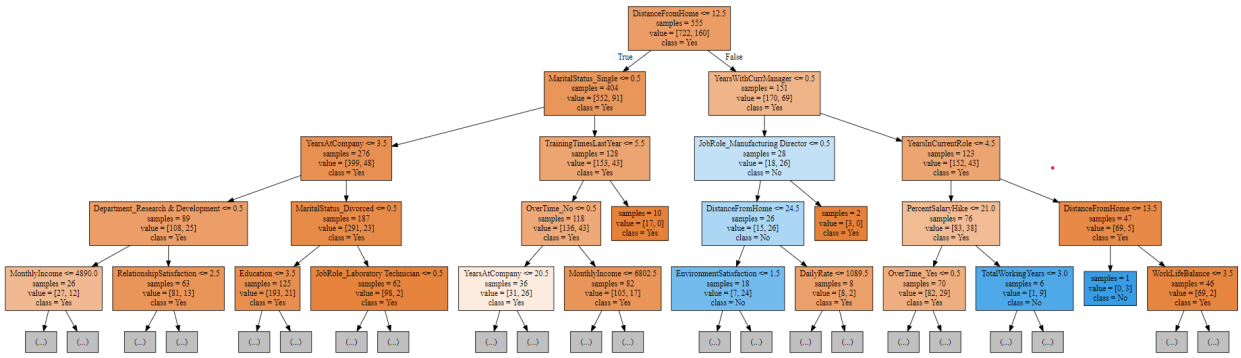
In [127]:
```
with open("RFDT.dot", 'w') as f:
    f = tree.export_graphviz(estimator, out_file=f, max_depth=4, impurity=False,
                             class_names = ['Yes','No'],filled=True)
fig.savefig("tree.png")
```

In [130]:
```
1 !type RFDT.dot
```

...

In [128]:
```
1 !dot - Tpng RFDT.dot -o RFDT.png
```

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



dot_data = tree.export_graphviz(RFC, feature_names=feature_names,
class_names=class_names,
filled=True, rounded=True,
special_characters=True, out_file=None,) graph = graphviz.Source(dot_data) graph

https://stackoverflow.com/questions/67710425/how-to-plot-feature-importance-for-random-forest-in-python (https://stackoverflow.com/questions/67710425/how-to-plot-feature-importance-for-random-forest-in-python)

In [132]:
```
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

```

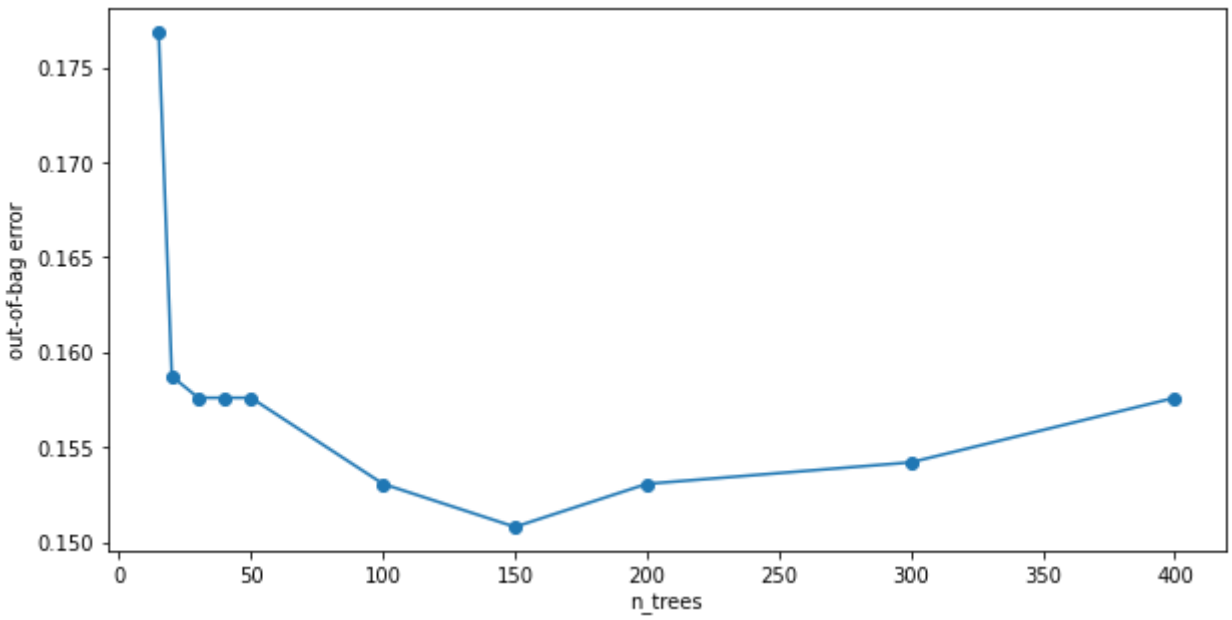
```
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\ensemble\_
forest.py:541: UserWarning: Some inputs do not have OOB scores. This probably means
too few trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\Users\1mscdsa41\AppData\Roaming\Python\Python36\site-packages\sklearn\ensemble\_
forest.py:546: RuntimeWarning: invalid value encountered in true_divide
  predictions[k].sum(axis=1)[:, np.newaxis])
```

Out[132]:

n_trees	oob
15.0	0.176871
20.0	0.158730
30.0	0.157596
40.0	0.157596
50.0	0.157596
100.0	0.153061
150.0	0.150794
200.0	0.153061
300.0	0.154195
400.0	0.157596

In [133]:
```
ax = rf_oob_df.plot(legend=False, marker='o', figsize=(10,5))
ax.set(ylabel='out-of-bag error')
```

Out[133]:
```
[Text(0, 0.5, 'out-of-bag error')]
```



In [135]:

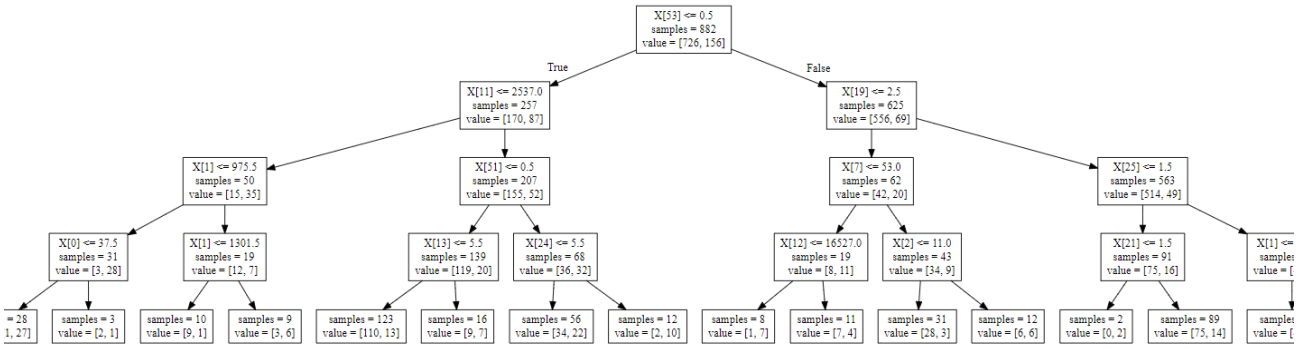
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report
clf = DecisionTreeClassifier(max_depth=4, random_state=42)
clf.fit(X_train,y_train)


y_pred1 = clf.predict(X_test)
y_pred1



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)

print("Accuracy of test :",clf.score(X_test,y_test))

print(classification_report(y_test,y_pred1))
```

```
Accuracy of test : 0.858843537414966
              precision    recall  f1-score   support

           0       0.88      0.97      0.92       507
           1       0.46      0.16      0.24        81

    accuracy                           0.86       588
   macro avg       0.67      0.57      0.58       588
weighted avg       0.82      0.86      0.83       588
```

In [136]:

```
!type DTC2.dot
```

```
digraph Tree {
node [shape=box] ;
0 [label="X[53] <= 0.5\nsamples = 882\nvalue = [726, 156]"] ;
1 [label="X[11] <= 2537.0\nsamples = 257\nvalue = [170, 87]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="X[1] <= 975.5\nsamples = 50\nvalue = [15, 35]"] ;
1 -> 2 ;
3 [label="X[0] <= 37.5\nsamples = 31\nvalue = [3, 28]"] ;
2 -> 3 ;
4 [label="samples = 28\nvalue = [1, 27]"] ;
3 -> 4 ;
5 [label="samples = 3\nvalue = [2, 1]"] ;
3 -> 5 ;
6 [label="X[1] <= 1301.5\nsamples = 19\nvalue = [12, 7]"] ;
2 -> 6 ;
7 [label="samples = 10\nvalue = [9, 1]"] ;
6 -> 7 ;
8 [label="samples = 9\nvalue = [3, 6]"] ;
6 -> 8 ;
```


X[53] <= 0.5
samples = 882
value = [726, 156]
True
False
X[11] <= 2537.0
samples = 257
value = [170, 87]
X[19] <= 2.5
samples = 625
value = [556, 69]
X[1] <= 975.5
samples = 50
value = [15, 35]
X[51] <= 0.5
samples = 207
value = [155, 52]
X[7] <= 53.0
samples = 62
value = [42, 20]
X[25] <= 1.5
samples = 563
value = [514, 49]
X[0] <= 37.5
samples = 31
value = [3, 28]
X[1] <= 1301.5
samples = 19
value = [12, 7]
X[13] <= 5.5
samples = 139
value = [119, 20]
X[24] <= 5.5
samples = 68
value = [36, 32]
X[12] <= 16527.0
samples = 19
value = [8, 11]
X[2] <= 11.0
samples = 43
value = [34, 9]
X[21] <= 1.5
samples = 91
value = [75, 16]
= 28
1, 27]
samples = 3
value = [2, 1]
samples = 10
value = [9, 1]
samples = 9
value = [3, 6]
samples = 123
value = [110, 13]
samples = 16
value = [9, 7]
samples = 56
value = [34, 22]
samples = 12
value = [2, 10]
samples = 8
value = [1, 7]
samples = 11
value = [7, 4]
samples = 31
value = [28, 3]
samples = 12
value = [6, 6]
samples = 2
value = [0, 2]
samples = 89
value = [75, 14]

In [ ]:

```
1
```

In [ ]:

```
1
```