HR_Attrition_Analysis

December 9, 2024

1 Import Libraries and Data

[1]:
```
!pip install plotly --upgrade --quiet
```

[2]:
```
# Python libraries
import pandas as pd
import numpy as np
from datetime import datetime
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV,␣
↪cross_val_score, learning_curve, train_test_split
from sklearn.metrics import precision_score, roc_auc_score, recall_score,␣
↪confusion_matrix, roc_curve, precision_recall_curve, accuracy_score
import xgboost as xgb
import warnings
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.figure_factory as ff

import plotly.io as pio ## Important for colab
pio.renderers.default = 'colab'

warnings.filterwarnings('ignore')
```

[4]:
```
file_id = "1tv2dfgghMKzN_mKmdn0ymCUrkwVvBk3w"
url = f"https://drive.google.com/uc?id={file_id}"
df = pd.read_csv(url)
df.head()
```

[4]:
```
   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]
```

[5]:
```python
data = df.copy()
```

2 Data Summarization

[6]:
```python
df.shape
```

[6]:
```
(1470, 35)
```

[7]:
```python
df.Attrition.value_counts(normalize='True')
```

[7]:
```
Attrition
No     0.838776
Yes    0.161224
Name: proportion, dtype: float64
```

[8]:
```python
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
```

[9]:
```python
df.describe(include='object')
```

[9]:
```
       Attrition BusinessTravel              Department EducationField Gender  \
count       1470           1470                    1470           1470   1470
unique         2              3                       3              6      2
top           No  Travel_Rarely  Research & Development  Life Sciences   Male
freq        1233           1043                     961            606    882

                 JobRole MaritalStatus Over18 OverTime
count               1470          1470   1470     1470
unique                 9             3      1        2
top      Sales Executive       Married      Y       No
freq                 326           673   1470     1054
```

[10]:
```python
df.describe(include='int64')
```

[10]:
```
               Age    DailyRate  DistanceFromHome    Education  EmployeeCount  \
count  1470.000000  1470.000000       1470.000000  1470.000000         1470.0
mean     36.923810   802.485714          9.192517     2.912925            1.0
std       9.135373   403.509100          8.106864     1.024165            0.0
min      18.000000   102.000000          1.000000     1.000000            1.0
25%      30.000000   465.000000          2.000000     2.000000            1.0
50%      36.000000   802.000000          7.000000     3.000000            1.0
75%      43.000000  1157.000000         14.000000     4.000000            1.0
max      60.000000  1499.000000         29.000000     5.000000            1.0

       EmployeeNumber  EnvironmentSatisfaction   HourlyRate  JobInvolvement  \
count     1470.000000              1470.000000  1470.000000     1470.000000
mean      1024.865306                 2.721769    65.891156        2.729932
std        602.024335                 1.093082    20.329428        0.711561
min          1.000000                 1.000000    30.000000        1.000000
25%        491.250000                 2.000000    48.000000        2.000000
50%       1020.500000                 3.000000    66.000000        3.000000
75%       1555.750000                 4.000000    83.750000        3.000000
max       2068.000000                 4.000000   100.000000        4.000000

          JobLevel  …  RelationshipSatisfaction  StandardHours  \
count  1470.000000  …               1470.000000         1470.0
mean      2.063946  …                  2.712245           80.0
std       1.106940  …                  1.081209            0.0
min       1.000000  …                  1.000000           80.0
25%       1.000000  …                  2.000000           80.0
50%       2.000000  …                  3.000000           80.0
75%       3.000000  …                  4.000000           80.0
max       5.000000  …                  4.000000           80.0

       StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
count       1470.000000        1470.000000            1470.000000
mean           0.793878          11.279592               2.799320
```

```
std           0.852077         7.780782              1.289271
min           0.000000         0.000000              0.000000
25%           0.000000         6.000000              2.000000
50%           1.000000        10.000000              3.000000
75%           1.000000        15.000000              3.000000
max           3.000000        40.000000              6.000000

       WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
count      1470.000000     1470.000000         1470.000000
mean          2.761224        7.008163            4.229252
std           0.706476        6.126525            3.623137
min           1.000000        0.000000            0.000000
25%           2.000000        3.000000            2.000000
50%           3.000000        5.000000            3.000000
75%           3.000000        9.000000            7.000000
max           4.000000       40.000000           18.000000

       YearsSinceLastPromotion  YearsWithCurrManager
count              1470.000000           1470.000000
mean                  2.187755              4.123129
std                   3.222430              3.568136
min                   0.000000              0.000000
25%                   0.000000              2.000000
50%                   1.000000              3.000000
75%                   3.000000              7.000000
max                  15.000000             17.000000

[8 rows x 26 columns]
```

[11]:
```python
null_feat = pd.DataFrame(len(data['Attrition']) - data.isnull().sum(),
 columns=['Count'])
null_feat
```

[11]:
```
                          Count
Age                        1470
Attrition                  1470
BusinessTravel             1470
DailyRate                  1470
Department                 1470
DistanceFromHome           1470
Education                  1470
EducationField             1470
EmployeeCount              1470
EmployeeNumber             1470
EnvironmentSatisfaction    1470
Gender                     1470
HourlyRate                 1470
```

```
JobInvolvement              1470
JobLevel                    1470
JobRole                     1470
JobSatisfaction             1470
MaritalStatus               1470
MonthlyIncome               1470
MonthlyRate                 1470
NumCompaniesWorked          1470
Over18                      1470
OverTime                    1470
PercentSalaryHike           1470
PerformanceRating           1470
RelationshipSatisfaction    1470
StandardHours               1470
StockOptionLevel            1470
TotalWorkingYears           1470
TrainingTimesLastYear       1470
WorkLifeBalance             1470
YearsAtCompany              1470
YearsInCurrentRole          1470
YearsSinceLastPromotion     1470
YearsWithCurrManager        1470
```

[12]:

```
# Calculate missing values count

# Create a bar trace for the missing values
trace = go.Bar(
            x=null_feat.index,
            y=null_feat['Count'],
            opacity=0.8,
            marker=dict(
                color='lightgrey',
                line=dict(color='#000000', width=1.5)
    )
)

# Define the layout
layout = go.Layout(
    title="Missing Values",
    xaxis_title="Features",
    yaxis_title="Count of Non-Missing Values",
    template="plotly_white"
)

# Combine trace and layout into a figure
fig = go.Figure(data=[trace], layout=layout)
```

```
# Show the figure
fig.show()
```

[13]:
```
data.columns.to_list()
```

[13]:
```
['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']
```

2.1 Reassign the target variable and drop the unneccesary columns

[14]:
```
# Reassign target
data.Attrition.replace(to_replace = dict(Yes = 1, No = 0), inplace = True)
# Drop useless feat
# data = data.drop(columns=['StandardHours',
```

```
#                           'EmployeeCount',
#                           'Over18',
#                         ])
```

3 EDA

[15]:
```
attrition = data[(data['Attrition'] != 0)]
no_attrition = data[(data['Attrition'] == 0)]

#------------COUNT-----------------------
trace = go.Bar(x = (len(attrition), len(no_attrition)), y = ['Yes_attrition',␣
↪'No_attrition'], orientation = 'h', opacity = 0.8, marker=dict(
        color=['gold', 'lightskyblue'],
        line=dict(color='#000000',width=1.5)))

layout = dict(title =  'Count of attrition variable')

fig = dict(data = [trace], layout=layout)
py.iplot(fig)


#------------PERCENTAGE-------------------
trace = go.Pie(labels = ['No_attrition', 'Yes_attrition'], values =␣
↪data['Attrition'].value_counts(),
               textfont=dict(size=15), opacity = 0.8,
               marker=dict(colors=['lightskyblue','gold'],
                           line=dict(color='#000000', width=1.5)))


layout = dict(title =  'Distribution of attrition variable')

fig = dict(data = [trace], layout=layout)
py.iplot(fig)
```

[15]:
```
```

[16]:
```
# # ------------ COUNT -----------------------
# # Horizontal bar chart for the count of attrition
# trace1 = go.Bar(
#     x=[len(attrition), len(no_attrition)],
#     y=['Yes_attrition', 'No_attrition'],
#     orientation='h',
#     opacity=0.8,
#     marker=dict(
#         color=['gold', 'lightskyblue'],
#         line=dict(color='#000000', width=1.5)
```

```
#     )
# )

# layout1 = go.Layout(
#     title='Count of Attrition Variable',
#     xaxis_title="Count",
#     yaxis_title="Attrition Status",
#     template='plotly_white'
# )

# fig1 = go.Figure(data=[trace1], layout=layout1)
# fig1.show()

# # ------------ PERCENTAGE -------------------
# # Pie chart for the distribution of attrition
# trace2 = go.Pie(
#     labels=['No_attrition', 'Yes_attrition'],
#     values=data['Attrition'].value_counts(),
#     textfont=dict(size=15),
#     opacity=0.8,
#     marker=dict(
#         colors=['lightskyblue', 'gold'],
#         line=dict(color='#000000', width=1.5)
#     )
# )

# layout2 = go.Layout(
#     title='Distribution of Attrition Variable',
#     template='plotly_white'
# )

# fig2 = go.Figure(data=[trace2], layout=layout2)
# fig2.show()
```

3.1 Feature Distribution and barplot (hue = Atrrition)

[17]:

```
def plot_distribution(var_select, bin_size) :
# Calculate the correlation coefficient between the new variable and the target
    corr = data['Attrition'].corr(data[var_select])
    corr = np.round(corr,3)
    tmp1 = attrition[var_select]
    tmp2 = no_attrition[var_select]
    hist_data = [tmp1, tmp2]

    group_labels = ['Yes_attrition', 'No_attrition']
    colors = ['#FFD700', '#7EC0EE']
```

```
    fig = ff.create_distplot(hist_data, group_labels, colors = colors,␣
↪show_hist = True, curve_type='kde', bin_size = bin_size)

    fig['layout'].update(title = var_select+' '+'(corr target ='+ str(corr)+')')

    py.iplot(fig, filename = 'Density plot')
```

[18]:
```
tpi = pd.DataFrame(pd.crosstab(data['OverTime'],data['Attrition']), )
tpi
```

[18]:

Attrition	0	1
OverTime		
No	944	110
Yes	289	127

[19]:
```
def barplot(var_select, x_no_numeric) :
    tmp1 = data[(data['Attrition'] != 0)]
    tmp2 = data[(data['Attrition'] == 0)]
    tmp3 = pd.DataFrame(pd.crosstab(data[var_select],data['Attrition']), )
    tmp3['Attr%'] = tmp3[1] / (tmp3[1] + tmp3[0]) * 100
    if x_no_numeric == True  :
        tmp3 = tmp3.sort_values(1, ascending = False)

    color=['lightskyblue','gold' ]
    trace1 = go.Bar(
        x=tmp1[var_select].value_counts().keys().tolist(),
        y=tmp1[var_select].value_counts().values.tolist(),
        name='Yes_Attrition',opacity = 0.8, marker=dict(
        color='gold',
        line=dict(color='#000000',width=1)))


    trace2 = go.Bar(
        x=tmp2[var_select].value_counts().keys().tolist(),
        y=tmp2[var_select].value_counts().values.tolist(),
        name='No_Attrition', opacity = 0.8, marker=dict(
        color='lightskyblue',
        line=dict(color='#000000',width=1)))

    trace3 =  go.Scatter(
        x=tmp3.index,
        y=tmp3['Attr%'],
        yaxis = 'y2',
        name='% Attrition', opacity = 0.6, marker=dict(
        color='black',
        line=dict(color='#000000',width=0.5
        )))
```

```
layout = dict(title =  str(var_select),
          xaxis=dict(),
          yaxis=dict(title= 'Count'),
          yaxis2=dict(range= [-0, 75],
                      overlaying= 'y',
                      anchor= 'x',
                      side= 'right',
                      zeroline=False,
                      showgrid= False,
                      title= '% Attrition'
                     ))

fig = go.Figure(data=[trace1, trace2, trace3], layout=layout)
py.iplot(fig)
```

3.1.1 plot_distribution and bar_plot

[20]:
```
plot_distribution('Age', False)
```

[21]:
```
barplot('Age', False)
```

[22]:
```
plot_distribution('DailyRate', 100)
```

[23]:
```
plot_distribution('DistanceFromHome', False)
```

[24]:
```
barplot('DistanceFromHome', False)
```

[25]:
```
plot_distribution('HourlyRate', False)
```

[26]:
```
plot_distribution('MonthlyIncome', 100)
```

[27]:
```
plot_distribution('MonthlyRate', 100)
```

[28]:
```
plot_distribution('NumCompaniesWorked', False)
```

[29]:
```
barplot('NumCompaniesWorked',False)
```

[30]:
```
plot_distribution('PercentSalaryHike', False)
```

[31]:
```
barplot('PercentSalaryHike', False)
```

[32]:
```
plot_distribution('TotalWorkingYears', False)
```

[33]:
```
barplot('TotalWorkingYears', False)
```

[34]:
```
plot_distribution('TrainingTimesLastYear', False)
```

[35]:
```
barplot('TrainingTimesLastYear',False)
```

[36]:
```
plot_distribution('YearsAtCompany', False)
```

[37]:
```
barplot('YearsAtCompany', False)
```

[38]:
```
plot_distribution('YearsInCurrentRole', False)
```

[39]:
```
plot_distribution('YearsInCurrentRole', False)
```

[40]:
```
barplot('YearsInCurrentRole', False)
```

[41]:
```
plot_distribution('YearsSinceLastPromotion', False)
```

[42]:
```
barplot('YearsSinceLastPromotion', False)
```

[43]:
```
plot_distribution('YearsWithCurrManager', False)
```

[44]:
```
barplot('YearsWithCurrManager', False)
```

3.1.2 Pie Plot and Bar Plot

[45]:
```
def plot_pie(var_select) :

    colors = ['gold', 'lightgreen', 'lightcoral', 'lightskyblue', 'lightgrey',␣
 ↪'orange', 'white', 'lightpink']
    trace1 = go.Pie(values  = attrition[var_select].value_counts().values.
 ↪tolist(),
                    labels  = attrition[var_select].value_counts().keys().
 ↪tolist(),
                    textfont=dict(size=15), opacity = 0.8,
                    hoverinfo = "label+percent+name",
                    domain  = dict(x = [0,.48]),
                    name    = "attrition employes",
                    marker  = dict(colors = colors, line = dict(width = 1.5)))
    trace2 = go.Pie(values  = no_attrition[var_select].value_counts().values.
 ↪tolist(),
                    labels  = no_attrition[var_select].value_counts().keys().
 ↪tolist(),
                    textfont=dict(size=15), opacity = 0.8,
                    hoverinfo = "label+percent+name",
                    marker  = dict(colors = colors, line = dict(width = 1.5)),
                    domain  = dict(x = [.52,1]),
                    name    = "Non attrition employes" )
```

```
    layout = go.Layout(dict(title = var_select + " distribution in employes␣
↪attrition ",
                            annotations = [dict(text = "Yes_attrition",
                                                font = dict(size = 13),
                                                showarrow = False,
                                                x = .22, y = -0.1),
                                            dict(text = "No_attrition",
                                                font = dict(size = 13),
                                                showarrow = False,
                                                x = .8,y = -.1)]))


    fig  = go.Figure(data = [trace1,trace2],layout = layout)
    py.iplot(fig)
```

[46]:
```
plot_pie("Gender")
```

[47]:
```
barplot('Gender',True)
```

[48]:
```
plot_pie('OverTime')
```

[49]:
```
barplot('OverTime',True)
```

[50]:
```
plot_pie('BusinessTravel')
```

[51]:
```
barplot('BusinessTravel',True)
```

[52]:
```
plot_pie('JobRole')
```

[53]:
```
barplot('JobRole',True)
```

[54]:
```
plot_pie('Department')
```

[55]:
```
barplot('Department',True)
```

[56]:
```
plot_pie('MaritalStatus')
```

[57]:
```
barplot('MaritalStatus',True)
```

[58]:
```
plot_pie('EducationField')
```

[59]:
```
barplot('EducationField',True)
```

[60]:
```
plot_pie('Education')
```

[61]:
```
barplot('Education',False)
```

[62]:
```
plot_pie('EnvironmentSatisfaction')
```

[63]:
```
barplot('EnvironmentSatisfaction',False)
```

[64]:
```
plot_pie('JobInvolvement')
```

[65]:
```
barplot('JobInvolvement', False)
```

[66]:
```
plot_pie('JobLevel')
```

[67]:
```
barplot('JobLevel',False)
```

[68]:
```
plot_pie('JobSatisfaction')
```

[69]:
```
barplot('JobSatisfaction',False)
```

[70]:
```
plot_pie('PerformanceRating')
```

[71]:
```
barplot('PerformanceRating',False)
```

[72]:
```
plot_pie('RelationshipSatisfaction')
```

[73]:
```
barplot('RelationshipSatisfaction', False)
```

[74]:
```
plot_pie('StockOptionLevel')
```

[75]:
```
barplot('StockOptionLevel', False)
```

[76]:
```
plot_pie('WorkLifeBalance')
```

[77]:
```
barplot('WorkLifeBalance', False)
```

4 Feature Engineering and Selection

[78]:
```
df1 = data.copy()
```

[79]:
```
def SalesDpt(data) :
    if data['Department'] == 'Sales':
        return 1
    else:
        return 0
data['SalesDpt'] = data.apply(lambda data:SalesDpt(data) ,axis = 1)

def JobInvCut(data) :
    if data['JobInvolvement'] < 2.5 :
        return 1
```

```
    else:
        return 0
data['JobInvCut'] = data.apply(lambda data:JobInvCut(data) ,axis = 1)

def MiddleTraining(data) :
    if data['TrainingTimesLastYear'] >= 3 and data['TrainingTimesLastYear'] <=␣
 ↪6:
        return 1
    else:
        return 0
data['MiddleTraining'] = data.apply(lambda data:MiddleTraining(data) ,axis = 1)

def MoovingPeople(data) :
    if data['NumCompaniesWorked'] > 4:
        return 1
    else:
        return 0
data['MoovingPeople'] = data.apply(lambda data:MoovingPeople(data), axis = 1)

data['TotalSatisfaction_mean'] = (data['RelationshipSatisfaction']  +␣
 ↪data['EnvironmentSatisfaction'] + data['JobSatisfaction'] +␣
 ↪data['JobInvolvement'] + data['WorkLifeBalance'])/5

def NotSatif(data) :
    if  data['TotalSatisfaction_mean'] < 2.35 :
        return 1
    else :
        return 0
data['NotSatif'] = data.apply(lambda data:NotSatif(data) ,axis = 1)

def LongDisWL1(data) :
    if  data['DistanceFromHome'] > 11 and data['WorkLifeBalance'] == 1 :
        return 1
    else :
        return 0
data['LongDisWL1'] = data.apply(lambda data:LongDisWL1(data) ,axis = 1)

def LongDis(data) :
    if  data['DistanceFromHome'] > 11:
        return 1
    else :
        return 0
data['LongDis'] = data.apply(lambda data:LongDis(data) ,axis = 1)

def LongDisJobS1(data) :
    if  data['DistanceFromHome'] > 11 and data['JobSatisfaction'] == 1 :
        return 1
```

```
    else :
        return 0
data['LongDisJobS1'] = data.apply(lambda data:LongDisJobS1(data) ,axis = 1)

def LongDisJL1(data) :
    if  data['DistanceFromHome'] > 11 and data['JobLevel'] == 1 :
        return 1
    else :
        return 0
data['LongDisJL1'] = data.apply(lambda data:LongDisJL1(data) ,axis = 1)

def ShortDisNotSingle(data) :
    if  data['MaritalStatus'] != 'Single' and data['DistanceFromHome'] < 5:
        return 1
    else :
        return 0
data['ShortDisNotSingle'] = data.apply(lambda data:ShortDisNotSingle(data)␣
 ↪,axis = 1)

def LongDisSingle(data) :
    if  data['MaritalStatus'] == 'Single' and data['DistanceFromHome'] > 11:
        return 1
    else :
        return 0
data['LongDisSingle'] = data.apply(lambda data:LongDisSingle(data) ,axis = 1)

def Engaged(data) :
    if data['Age'] > 35 and data['MaritalStatus'] != 'Single':
        return 1
    else :
        return 0
data['Engaged'] = data.apply(lambda data:Engaged(data) ,axis = 1)

def YoungAndBadPaid(data) :
    if data['Age'] < 35 and data['Age'] > 23 and (data['MonthlyIncome'] < 3500):
        return 1
    else :
        return 0
data['YoungAndBadPaid'] = data.apply(lambda data:YoungAndBadPaid(data) ,axis =␣
 ↪1)

def YoungNeverEngaged(data) :
    if data['Age'] < 24 and data['MaritalStatus'] == 'Single' :
        return 1
    else :
        return 0
```

```
data['YoungNeverEngaged'] = data.apply(lambda data:YoungNeverEngaged(data)␣
 ↪,axis = 1)

data['Time_in_each_comp'] = (data['Age'] - 20) / ((data)['NumCompaniesWorked']␣
 ↪+ 1)
data['RelSatisf_mean'] = (data['RelationshipSatisfaction']  +␣
 ↪data['EnvironmentSatisfaction']) / 2
data['JobSatisf_mean'] = (data['JobSatisfaction'] + data['JobInvolvement']) / 2
data['Income_Distance'] = data['MonthlyIncome'] / data['DistanceFromHome']
data['Hrate_Mrate'] = data['HourlyRate'] / data['MonthlyRate']
data['Stability'] = data['YearsInCurrentRole'] / data['YearsAtCompany']
data['Stability'].fillna((data['Stability'].mean()), inplace=True)
data['Income_YearsComp'] = data['MonthlyIncome'] / data['YearsAtCompany']
data['Income_YearsComp'] = data['Income_YearsComp'].replace(np.Inf, 0)
data['Fidelity'] = (data['NumCompaniesWorked']) / data['TotalWorkingYears']
data['Fidelity'] = data['Fidelity'].replace(np.Inf, 0)
```

[80]:
```
data.columns.tolist()
```

[80]:
```
['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',
 'SalesDpt',
 'JobInvCut',
 'MiddleTraining',
 'MoovingPeople',
 'TotalSatisfaction_mean',
 'NotSatif',
 'LongDisWL1',
 'LongDis',
 'LongDisJobS1',
 'LongDisJL1',
 'ShortDisNotSingle',
 'LongDisSingle',
 'Engaged',
 'YoungAndBadPaid',
 'YoungNeverEngaged',
 'Time_in_each_comp',
 'RelSatisf_mean',
 'JobSatisf_mean',
 'Income_Distance',
 'Hrate_Mrate',
 'Stability',
 'Income_YearsComp',
 'Fidelity']
```

[81]:
```
barplot('Engaged', False)
barplot('YoungAndBadPaid', False)
barplot('YoungNeverEngaged', False)
barplot('LongDisSingle', False)
barplot('LongDisJL1', False)
barplot('ShortDisNotSingle', False)
```

4.1 Drop some Features

[82]:
```
data = data.drop(columns=[
                        'Age',
                        'MonthlyIncome',
                        'YearsAtCompany',
                        'DistanceFromHome',
                        'PerformanceRating',
                        'NumCompaniesWorked'
```

```
                    ])

print ("\nMissing values :  ", data.isnull().sum().values.sum())
```

```
Missing values :   0
```

[83]:
```
df2 = data.copy()
```

[84]:
```
#customer id col
Id_col     = ['EmployeeNumber']
#Target columns
target_col = ["Attrition"]

#categorical columns
cat_cols   = data.nunique()[data.nunique() < 10].keys().tolist()
cat_cols   = [x for x in cat_cols if x not in target_col]


#numerical columns
num_cols   = [x for x in data.columns if x not in cat_cols + target_col +␣
 ↪Id_col]

#Binary columns with 2 values
bin_cols   = data.nunique()[data.nunique() == 2].keys().tolist()

#Columns more than 2 values
multi_cols = [i for i in cat_cols if i not in bin_cols]
```

[85]:
```
data.nunique()[data.nunique() == 2].keys().tolist()
```

[85]:
```
['Attrition',
 'Gender',
 'OverTime',
 'SalesDpt',
 'JobInvCut',
 'MiddleTraining',
 'MoovingPeople',
 'NotSatif',
 'LongDisWL1',
 'LongDis',
 'LongDisJobS1',
 'LongDisJL1',
 'ShortDisNotSingle',
 'LongDisSingle',
 'Engaged',
 'YoungAndBadPaid',
```

```
 'YoungNeverEngaged']
```

[86]:
```
num_cols
```

[86]:
```
['DailyRate',
 'HourlyRate',
 'MonthlyRate',
 'PercentSalaryHike',
 'TotalWorkingYears',
 'YearsInCurrentRole',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager',
 'TotalSatisfaction_mean',
 'Time_in_each_comp',
 'Income_Distance',
 'Hrate_Mrate',
 'Stability',
 'Income_YearsComp',
 'Fidelity']
```

[87]:
```
from sklearn.preprocessing import LabelEncoder, StandardScaler


#Label encoding Binary columns
le = LabelEncoder()
for i in bin_cols :
    data[i] = le.fit_transform(data[i])

#Duplicating columns for multi value columns
data = pd.get_dummies(data = data,columns = multi_cols )

#Scaling Numerical columns
std = StandardScaler()
scaled = std.fit_transform(data[num_cols])
scaled = pd.DataFrame(scaled,columns=num_cols)
```

[87]:

[88]:
```
#dropping original values merging scaled values for numerical columns
df_data_og = data.copy()

data = data.drop(columns = num_cols,axis = 1)
data = data.merge(scaled, left_index=True, right_index=True, how = "left")
data = data.drop(['EmployeeNumber'],axis = 1)
```

[89]:
```
data.head()
```

[89]:
```
   Attrition  Gender  OverTime  SalesDpt  JobInvCut  MiddleTraining  \
0          1       0         1         1          0               0
1          0       1         0         0          1               1
2          1       1         1         0          1               1
3          0       0         1         0          0               1
4          0       1         0         0          0               1

   MoovingPeople  NotSatif  LongDisWL1  LongDis  …  YearsInCurrentRole  \
0              1         1           0        0  …           -0.063296
1              0         0           0        0  …            0.764998
2              1         0           0        0  …           -1.167687
3              0         0           0        0  …            0.764998
4              1         0           0        0  …           -0.615492

   YearsSinceLastPromotion  YearsWithCurrManager  TotalSatisfaction_mean  \
0                -0.679146              0.245834               -1.238894
1                -0.368715              0.806541                0.161650
2                -0.679146             -1.155935                0.161650
3                 0.252146             -1.155935                1.095346
4                -0.058285             -0.595227               -0.305198

   Time_in_each_comp  Income_Distance  Hrate_Mrate     Stability  \
0          -0.774273         1.328107    -0.330471  2.234862e-01
1           1.578035        -0.451684    -0.715919  3.289956e-01
2          -0.755860        -0.317412     5.114320 -3.514170e-16
3           0.031312        -0.342465    -0.720954  8.829202e-01
4          -1.090062        -0.088277    -0.723072  1.278581e+00

   Income_YearsComp  Fidelity
0         -0.241733  1.775509
1         -0.473382 -0.620450
2         -0.717985  1.395198
3         -0.544605 -0.553896
4          0.108800  3.106597

[5 rows x 114 columns]
```

[90]:
```python
#correlation
correlation = data.corr()

#tick labels
matrix_cols = correlation.columns.tolist()

#convert to array
corr_array  = np.array(correlation)

#Plotting
```

```
trace = go.Heatmap(z = corr_array,
                   x = matrix_cols,
                   y = matrix_cols,
                   colorscale='Viridis',
                   colorbar   = dict() ,
                  )
layout = go.Layout(dict(title = 'Correlation Matrix for variables',
                        autosize = False,
                        #height  = 1400,
                        #width   = 1600,
                        margin  = dict(r = 0 ,l = 210,
                                       t = 25,b = 210,
                                     ),
                        yaxis   = dict(tickfont = dict(size = 9)),
                        xaxis   = dict(tickfont = dict(size = 9)),
                       )
                  )
fig = go.Figure(data = [trace],layout = layout)
py.iplot(fig)
```

4.2 Removing Colinear Features

[91]:
```
# Threshold for removing correlated variables
threshold = 0.8

# Absolute value correlation matrix
corr_matrix = data.corr().abs()
corr_matrix.head()
```

[91]:
```
           Attrition    Gender  OverTime  SalesDpt  JobInvCut  MiddleTraining  \
Attrition   1.000000  0.029453  0.246118  0.080855   0.100493        0.050715
Gender      0.029453  1.000000  0.041924  0.032017   0.020388        0.021742
OverTime    0.246118  0.041924  1.000000  0.005864   0.001269        0.066174
SalesDpt    0.080855  0.032017  0.005864  1.000000   0.000135        0.050157
JobInvCut   0.100493  0.020388  0.001269  0.000135   1.000000        0.022493

           MoovingPeople  NotSatif  LongDisWL1   LongDis  …  \
Attrition       0.078832  0.182389    0.074893  0.090791  …
Gender          0.030026  0.048507    0.015340  0.006170  …
OverTime        0.037709  0.037499    0.038231  0.042132  …
SalesDpt        0.016171  0.042793    0.008388  0.003578  …
JobInvCut       0.010965  0.230432    0.006055  0.020556  …

           YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager  \
Attrition            0.160545                 0.033019              0.156199
Gender               0.041483                 0.026985              0.030599
OverTime             0.029758                 0.012239              0.041586
```

```
SalesDpt              0.046883                 0.034112                0.027415
JobInvCut             0.009326                 0.014139                0.015398

           TotalSatisfaction_mean  Time_in_each_comp  Income_Distance  \
Attrition                0.193395           0.142292         0.113071
Gender                   0.033969           0.020472         0.016322
OverTime                 0.062779           0.023349         0.027402
SalesDpt                 0.003156           0.016413         0.000699
JobInvCut                0.299947           0.002937         0.025481

           Hrate_Mrate  Stability  Income_YearsComp  Fidelity
Attrition     0.011526   0.105810          0.000428  0.225917
Gender        0.029134   0.002796          0.018223  0.010195
OverTime      0.015907   0.020479          0.033736  0.008214
SalesDpt      0.008906   0.045008          0.024346  0.028313
JobInvCut     0.016041   0.045266          0.039547  0.021035

[5 rows x 114 columns]
```

[92]:

```python
np.ones(corr_matrix.shape)
```

[92]:

```
array([[1., 1., 1., …, 1., 1., 1.],
       [1., 1., 1., …, 1., 1., 1.],
       [1., 1., 1., …, 1., 1., 1.],
       …,
       [1., 1., 1., …, 1., 1., 1.],
       [1., 1., 1., …, 1., 1., 1.],
       [1., 1., 1., …, 1., 1., 1.]])
```

[93]:

```python
upper_triangular = np.triu(np.ones(corr_matrix.shape), k=1)
print(upper_triangular)
```

```
[[0. 1. 1. … 1. 1. 1.]
 [0. 0. 1. … 1. 1. 1.]
 [0. 0. 0. … 1. 1. 1.]
 …
 [0. 0. 0. … 0. 1. 1.]
 [0. 0. 0. … 0. 0. 1.]
 [0. 0. 0. … 0. 0. 0.]]
```

[94]:

```python
# Upper triangle of correlations
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).
 ↪astype(bool)) # concept
upper.head()
```

[94]:

```
           Attrition    Gender  OverTime  SalesDpt  JobInvCut  MiddleTraining  \
Attrition        NaN  0.029453  0.246118  0.080855   0.100493        0.050715
```

```
Gender           NaN        NaN  0.041924  0.032017   0.020388        0.021742
OverTime         NaN        NaN       NaN  0.005864   0.001269        0.066174
SalesDpt         NaN        NaN       NaN       NaN   0.000135        0.050157
JobInvCut        NaN        NaN       NaN       NaN        NaN        0.022493

           MoovingPeople  NotSatif  LongDisWL1   LongDis  …  \
Attrition       0.078832  0.182389    0.074893  0.090791  …
Gender          0.030026  0.048507    0.015340  0.006170  …
OverTime        0.037709  0.037499    0.038231  0.042132  …
SalesDpt        0.016171  0.042793    0.008388  0.003578  …
JobInvCut       0.010965  0.230432    0.006055  0.020556  …

           YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager  \
Attrition            0.160545                 0.033019              0.156199
Gender               0.041483                 0.026985              0.030599
OverTime             0.029758                 0.012239              0.041586
SalesDpt             0.046883                 0.034112              0.027415
JobInvCut            0.009326                 0.014139              0.015398

           TotalSatisfaction_mean  Time_in_each_comp  Income_Distance  \
Attrition                0.193395           0.142292         0.113071
Gender                   0.033969           0.020472         0.016322
OverTime                 0.062779           0.023349         0.027402
SalesDpt                 0.003156           0.016413         0.000699
JobInvCut                0.299947           0.002937         0.025481

           Hrate_Mrate  Stability  Income_YearsComp  Fidelity
Attrition     0.011526   0.105810          0.000428  0.225917
Gender        0.029134   0.002796          0.018223  0.010195
OverTime      0.015907   0.020479          0.033736  0.008214
SalesDpt      0.008906   0.045008          0.024346  0.028313
JobInvCut     0.016041   0.045266          0.039547  0.021035

[5 rows x 114 columns]
```

[95]:

```python
# Select columns with correlations above threshold
to_drop = [column for column in upper.columns if any(upper[column] > threshold)]

print('There are %d columns to remove :' % (len(to_drop)))

data = data.drop(columns = to_drop)

to_drop
```

```
There are 7 columns to remove :
```

[95]: ['Department_Research & Development',
 'Department_Sales',
 'JobInvolvement_2',
 'JobInvolvement_3',
 'JobRole_Human Resources',
 'JobRole_Sales Executive',
 'TrainingTimesLastYear_2']

[96]:
```
data.columns.tolist()
```

[96]: ['Attrition',
 'Gender',
 'OverTime',
 'SalesDpt',
 'JobInvCut',
 'MiddleTraining',
 'MoovingPeople',
 'NotSatif',
 'LongDisWL1',
 'LongDis',
 'LongDisJobS1',
 'LongDisJL1',
 'ShortDisNotSingle',
 'LongDisSingle',
 'Engaged',
 'YoungAndBadPaid',
 'YoungNeverEngaged',
 'BusinessTravel_Non-Travel',
 'BusinessTravel_Travel_Frequently',
 'BusinessTravel_Travel_Rarely',
 'Department_Human Resources',
 'Education_1',
 'Education_2',
 'Education_3',
 'Education_4',
 'Education_5',
 'EducationField_Human Resources',
 'EducationField_Life Sciences',
 'EducationField_Marketing',
 'EducationField_Medical',
 'EducationField_Other',
 'EducationField_Technical Degree',
 'EmployeeCount_1',
 'EnvironmentSatisfaction_1',
 'EnvironmentSatisfaction_2',
 'EnvironmentSatisfaction_3',
 'EnvironmentSatisfaction_4',

```
'JobInvolvement_1',
'JobInvolvement_4',
'JobLevel_1',
'JobLevel_2',
'JobLevel_3',
'JobLevel_4',
'JobLevel_5',
'JobRole_Healthcare Representative',
'JobRole_Laboratory Technician',
'JobRole_Manager',
'JobRole_Manufacturing Director',
'JobRole_Research Director',
'JobRole_Research Scientist',
'JobRole_Sales Representative',
'JobSatisfaction_1',
'JobSatisfaction_2',
'JobSatisfaction_3',
'JobSatisfaction_4',
'MaritalStatus_Divorced',
'MaritalStatus_Married',
'MaritalStatus_Single',
'Over18_Y',
'RelationshipSatisfaction_1',
'RelationshipSatisfaction_2',
'RelationshipSatisfaction_3',
'RelationshipSatisfaction_4',
'StandardHours_80',
'StockOptionLevel_0',
'StockOptionLevel_1',
'StockOptionLevel_2',
'StockOptionLevel_3',
'TrainingTimesLastYear_0',
'TrainingTimesLastYear_1',
'TrainingTimesLastYear_3',
'TrainingTimesLastYear_4',
'TrainingTimesLastYear_5',
'TrainingTimesLastYear_6',
'WorkLifeBalance_1',
'WorkLifeBalance_2',
'WorkLifeBalance_3',
'WorkLifeBalance_4',
'RelSatisf_mean_1.0',
'RelSatisf_mean_1.5',
'RelSatisf_mean_2.0',
'RelSatisf_mean_2.5',
'RelSatisf_mean_3.0',
'RelSatisf_mean_3.5',
```

```
 'RelSatisf_mean_4.0',
 'JobSatisf_mean_1.0',
 'JobSatisf_mean_1.5',
 'JobSatisf_mean_2.0',
 'JobSatisf_mean_2.5',
 'JobSatisf_mean_3.0',
 'JobSatisf_mean_3.5',
 'JobSatisf_mean_4.0',
 'DailyRate',
 'HourlyRate',
 'MonthlyRate',
 'PercentSalaryHike',
 'TotalWorkingYears',
 'YearsInCurrentRole',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager',
 'TotalSatisfaction_mean',
 'Time_in_each_comp',
 'Income_Distance',
 'Hrate_Mrate',
 'Stability',
 'Income_YearsComp',
 'Fidelity']
```

[97]:
```
data.shape
```

[97]: (1470, 107)

5 Define functions

5.1 Model Evaluation Metric

5.1.1 Performance Plot

[98]:
```
def model_performance_plot(model) :
    #conf matrix
    conf_matrix = confusion_matrix(y_test, y_pred)
    trace1 = go.Heatmap(z = conf_matrix  ,x = ["0 (pred)","1 (pred)"],
                        y = ["0 (true)","1 (true)"],xgap = 2, ygap = 2,
                        colorscale = 'Viridis', showscale  = False)

    #show metrics
    tp = conf_matrix[1,1]
    fn = conf_matrix[1,0]
    fp = conf_matrix[0,1]
    tn = conf_matrix[0,0]
    Accuracy  =  ((tp+tn)/(tp+tn+fp+fn))
```

```
    Precision =  (tp/(tp+fp))
    Recall    =  (tp/(tp+fn))
    F1_score  =  (2*(((tp/(tp+fp))*(tp/(tp+fn)))/((tp/(tp+fp))+(tp/(tp+fn)))))

    show_metrics = pd.DataFrame(data=[[Accuracy , Precision, Recall, F1_score]])
    show_metrics = show_metrics.T

    colors = ['gold', 'lightgreen', 'lightcoral', 'lightskyblue']
    trace2 = go.Bar(x = (show_metrics[0].values),
                   y = ['Accuracy', 'Precision', 'Recall', 'F1_score'], text =␣
↪np.round_(show_metrics[0].values,4),
                    textposition = 'auto',
                   orientation = 'h', opacity = 0.8,marker=dict(
            color=colors,
            line=dict(color='#000000',width=1.5)))

    #plot roc curve
    model_roc_auc = round(roc_auc_score(y_test, y_score) , 3)
    fpr, tpr, t = roc_curve(y_test, y_score)
    trace3 = go.Scatter(x = fpr,y = tpr,
                        name = "Roc : ",
                        line = dict(color = ('rgb(22, 96, 167)'),width = 2),␣
↪fill='tozeroy')
    trace4 = go.Scatter(x = [0,1],y = [0,1],
                        line = dict(color = ('black'),width = 1.5,
                        dash = 'dot'))

    # Precision-recall curve
    precision, recall, thresholds = precision_recall_curve(y_test, y_score)
    trace5 = go.Scatter(x = recall, y = precision,
                        name = "Precision" + str(precision),
                        line = dict(color = ('lightcoral'),width = 2),␣
↪fill='tozeroy')

    #subplots
    fig = tls.make_subplots(rows=2, cols=2, print_grid=False,
                        subplot_titles=('Confusion Matrix',
                                        'Metrics',
                                        'ROC curve'+" "+ '('+␣
↪str(model_roc_auc)+')',
                                        'Precision - Recall curve'))

    fig.append_trace(trace1,1,1)
    fig.append_trace(trace2,1,2)
    fig.append_trace(trace3,2,1)
    fig.append_trace(trace4,2,1)
    fig.append_trace(trace5,2,2)
```

```
    fig['layout'].update(showlegend = False, title = '<b>Model performance</
↪b><br>'+str(model),
                        autosize = False, height = 900,width = 830,
                        plot_bgcolor = 'rgba(240,240,240, 0.95)',
                        paper_bgcolor = 'rgba(240,240,240, 0.95)',
                        margin = dict(b = 195))
    fig["layout"]["xaxis2"].update((dict(range=[0, 1])))
    fig["layout"]["xaxis3"].update(dict(title = "false positive rate"))
    fig["layout"]["yaxis3"].update(dict(title = "true positive rate"))
    fig["layout"]["xaxis4"].update(dict(title = "recall"), range = [0,1.05])
    fig["layout"]["yaxis4"].update(dict(title = "precision"), range = [0,1.05])
    fig.layout.titlefont.size = 14

    py.iplot(fig)
```

5.1.2 Feature Important Plot

[99]:
```
def features_imp(model, cf) :

    coefficients  = pd.DataFrame(model.feature_importances_)
    column_data     = pd.DataFrame(list(data))
    coef_sumry    = (pd.merge(coefficients,column_data,left_index= True,
                              right_index= True, how = "left"))
    coef_sumry.columns = ["coefficients","features"]
    coef_sumry    = coef_sumry.sort_values(by = "coefficients",ascending =␣
↪False)
    coef_sumry = coef_sumry[coef_sumry["coefficients"] !=0]
    trace = go.Bar(x = coef_sumry["features"],y = coef_sumry["coefficients"],
                   name = "coefficients",
                   marker = dict(color = coef_sumry["coefficients"],
                                 colorscale = "Viridis",
                                 line = dict(width = .6,color = "black")))
    layout = dict(title =  'Feature Importances xgb_cfl')

    fig = dict(data = [trace], layout=layout)
    py.iplot(fig)
```

5.1.3 Cumilative Curve plot

[100]:
```
#cumulative gain curve
def cum_gains_curve(model):
    pos = pd.get_dummies(y_test).to_numpy()
    pos = pos[:,1]
    npos = np.sum(pos)
    index = np.argsort(y_score)
```

```
    index = index[::-1]
    sort_pos = pos[index]
    #cumulative sum
    cpos = np.cumsum(sort_pos)
    #recall
    recall = cpos/npos
    #size obs test
    n = y_test.shape[0]
    size = np.arange(start=1,stop=369,step=1)
    #proportion
    size = size / n
    #plots
    model = 'xgb_cfl'
    trace1 = go.Scatter(x = size,y = recall,
                        name = "Lift curve",
                        line = dict(color = ('rgb(22, 96, 167)'),width = 2))
    trace2 = go.Scatter(x = size,y = size,
                        name = "Baseline",
                        showlegend=False,
                        line = dict(color = ('black'),width = 1.5,
                        dash = 'dot'))

    layout = dict(title = 'Cumulative gains curve'+' '+str(model),
                  yaxis = dict(title = 'Percentage positive targeted',zeroline␣
↪= False),
                  xaxis = dict(title = 'Percentage contacted', zeroline = False)
                 )

    fig  = go.Figure(data = [trace1,trace2], layout = layout)
    py.iplot(fig)
```

5.1.4 Cross Validation Metrics

[101]:
```
# Cross val metric
def cross_val_metrics(model) :
    scores = ['accuracy', 'precision', 'recall']
    for sc in scores:
        scores = cross_val_score(model, X, y, cv = 5, scoring = sc)
        print('[%s] : %0.5f (+/- %0.5f)'%(sc, scores.mean(), scores.std()))
```

5.2 Prepare Dataset

5.2.1 Spliting Data into X and Y

[102]:
```
# Define X and y
y = np.array(data['Attrition'].tolist())  # Convert the 'Attrition' column to a␣
↪NumPy array
data = data.drop('Attrition', axis=1)      # Drop the 'Attrition' column
X = data.to_numpy()                        # Convert the remaining DataFrame to␣
↪a NumPy array
```

[103]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.30,
                                                 random_state=42)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1029, 106)
(441, 106)
(1029,)
(441,)
```

5.2.2 XGboost - RandomizedSearchCV to Optimize Hyper parameters

[104]:
```
def timer(start_time=None):
    if not start_time:
        start_time = datetime.now()
        return start_time
    elif start_time:
        thour, temp_sec = divmod((datetime.now() - start_time).total_seconds(),␣
↪3600)
        tmin, tsec = divmod(temp_sec, 60)
        print('\n Time taken: %i hours %i minutes and %s seconds.' % (thour,␣
↪tmin, round(tsec, 2)))


xgb_cfl = xgb.XGBClassifier(n_jobs = -1)


# A parameter grid for XGBoost
params = {
        'n_estimators' : [100, 200, 500, 750],
        'learning_rate' : [0.01, 0.02, 0.05, 0.1, 0.25],
        'min_child_weight': [1, 5, 7, 10],
        'gamma': [0.1, 0.5, 1, 1.5, 5],
```

```
        'subsample': [0.6, 0.8, 1.0],
        'colsample_bytree': [0.6, 0.8, 1.0],
        'max_depth': [3, 4, 5, 10, 12]
        }

folds = 5
param_comb = 800

random_search = RandomizedSearchCV(xgb_cfl, param_distributions=params,␣
 ↪n_iter=param_comb, scoring='accuracy', n_jobs=-1, cv=5, verbose=3,␣
 ↪random_state=42)

# Here we go
start_time = timer(None) # timing starts from this point for "start_time"␣
 ↪variable
# random_search.fit(X, y)
timer(start_time) # timing ends here for "start_time" variable
```

```
Time taken: 0 hours 0 minutes and 0.0 seconds.
```

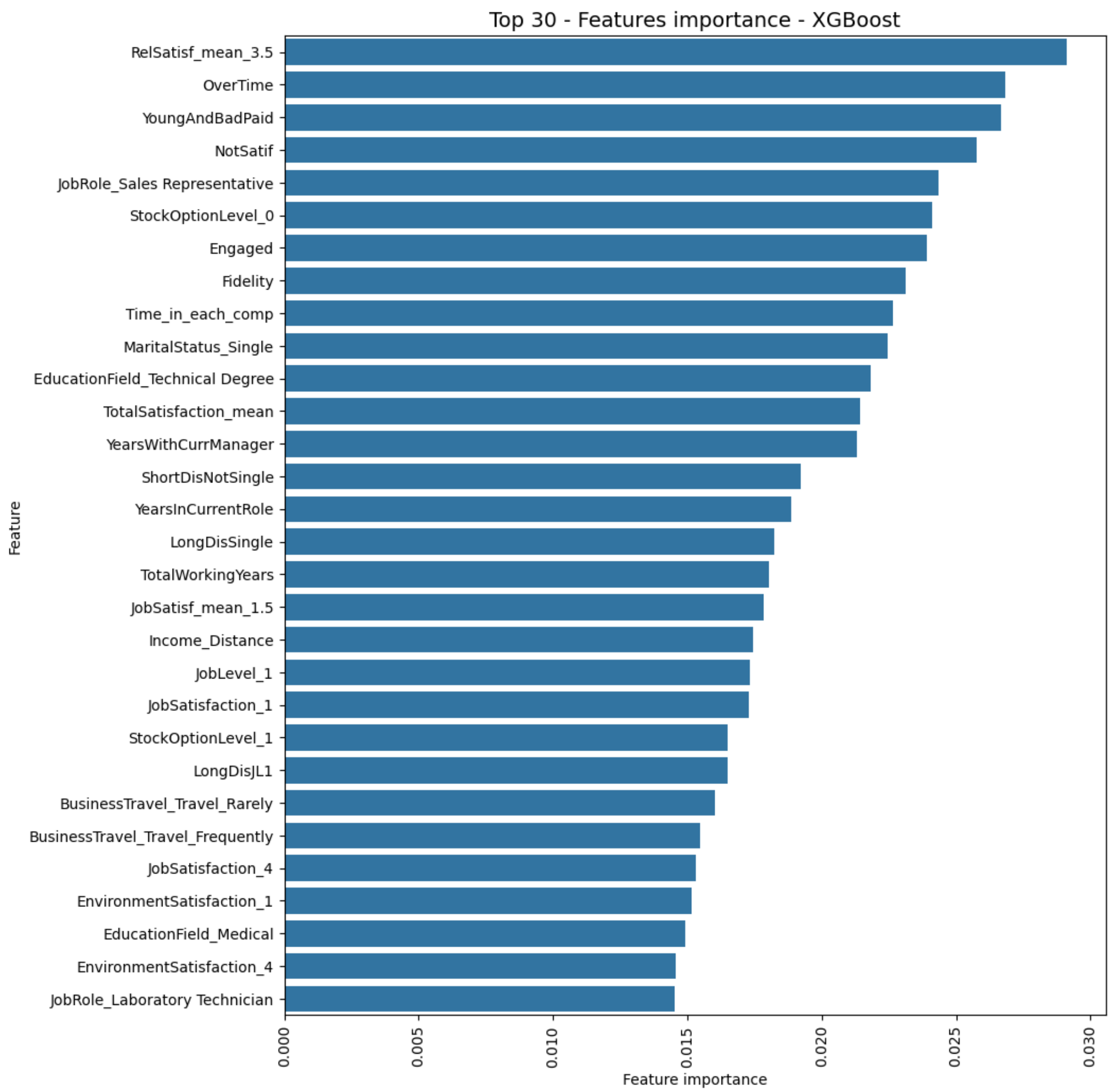
[105]:
```
import xgboost as xgb
import numpy as np

# XGBoost Classifier
xgb_clf = xgb.XGBClassifier(
    base_score=0.5,
    booster='gbtree',
    colsample_bylevel=1,
    colsample_bytree=0.8,
    gamma=1.5,
    learning_rate=0.05,
    max_delta_step=0,
    max_depth=3,
    min_child_weight=7,
    missing=np.nan,  # Fix here
    n_estimators=200,
    n_jobs=-1,
    objective='binary:logistic',
    random_state=0,
    reg_alpha=0,
    reg_lambda=1,
    scale_pos_weight=1,
    subsample=0.6
)

# Fit the model
```

```
xgb_clf.fit(x_train, y_train)

# Predictions
y_pred = xgb_clf.predict(x_test)
y_score = xgb_clf.predict_proba(x_test)[:, 1]

# Evaluate performance (assuming `model_performance_plot` is already defined)
model_performance_plot('xgb_clf')
```

[106]:
```
features_imp(xgb_clf, 'features')
```

[107]:
```
#feature importance plot TOP 40
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
def plot_feature_importance(model):
    tmp = pd.DataFrame({'Feature': list(data), 'Feature importance': model.
 ↪feature_importances_})
    tmp = tmp.sort_values(by='Feature importance',ascending=False).head(30)
    plt.figure(figsize = (10,12))
    plt.title('Top 30 - Features importance - XGBoost',fontsize=14)
    s = sns.barplot(y='Feature',x='Feature importance',data=tmp, orient='h')
    s.set_xticklabels(s.get_xticklabels(),rotation=90)
    plt.show()
```

[108]:
```
plot_feature_importance(xgb_clf)
```



[109]:
```
cum_gains_curve(xgb_clf)
```

[110]:
```
# Cross val score
cross_val_metrics(xgb_clf)
```

```
[accuracy] : 0.88639 (+/- 0.01109)
[precision] : 0.81666 (+/- 0.03359)
[recall] : 0.38457 (+/- 0.08906)
```