Unified Mentor Data Analytics Internship

Name: Suman Das

Project: Employee Attrition Analysis Data

Problem Statement

XYZ company which was established a few years back is facing around a 15% attrition rate for a couple of years. And it's majorly affecting the company in many aspects. In order to understand why employees are leaving the company and reduce the attrition rate XYZ company has approached an HR analytics consultancy for analyzing the data they have.

In [1]:
```
# import libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# ignore any error type warnings
import warnings
warnings.filterwarnings('ignore')

# suppressing any scientific notations
np.set_printoptions(suppress= True)

# controlling over maximum rows and columns to be displayed
pd.options.display.max_rows= 100
pd.options.display.max_columns= 50
```

Load Dataset

In [2]:
```
# read the Employee Attrition (EA) csv file
EA = pd.read_csv(filepath_or_buffer= 'Attrition data.csv', encoding='latin-1', sep=',')

# total no of rows and columns
print('no of rows and columns in the dataset: ', EA.shape)

# check duplicate values in the dataset
EA = EA.drop_duplicates()

print('no of rows and columns in the dataset after removing the duplicate rows: ', EA.shape)
# there is no duplicate values here

print('*'*50)

# print first 50 rows of the dataset to see a quick glance
EA.head(10)
```

```
no of rows and columns in the dataset:  (4410, 29)
no of rows and columns in the dataset after removing the duplicate rows:
(4410, 29)
**************************************************
```

Out[2]:

	EmployeeID	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education
0	1	51	No	Travel_Rarely	Sales	6	2
1	2	31	Yes	Travel_Frequently	Research & Development	10	1
2	3	32	No	Travel_Frequently	Research & Development	17	4
3	4	38	No	Non-Travel	Research & Development	2	5
4	5	32	No	Travel_Rarely	Research & Development	10	1
5	6	46	No	Travel_Rarely	Research & Development	8	3
6	7	28	Yes	Travel_Rarely	Research & Development	11	2
7	8	29	No	Travel_Rarely	Research & Development	18	3
8	9	31	No	Travel_Rarely	Research & Development	1	3
9	10	25	No	Non-Travel	Research & Development	7	4

Basic Exploratory Data Analysis

In [3]:
```
# find datatype, no of columns, no of non-null vales
EA.info()  # information about the dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4410 entries, 0 to 4409
Data columns (total 29 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   EmployeeID                4410 non-null   int64  
 1   Age                       4410 non-null   int64  
 2   Attrition                 4410 non-null   object 
 3   BusinessTravel            4410 non-null   object 
 4   Department                4410 non-null   object 
 5   DistanceFromHome          4410 non-null   int64  
 6   Education                 4410 non-null   int64  
 7   EducationField            4410 non-null   object 
 8   EmployeeCount             4410 non-null   int64  
 9   Gender                    4410 non-null   object 
 10  JobLevel                  4410 non-null   int64  
 11  JobRole                   4410 non-null   object 
 12  MaritalStatus             4410 non-null   object 
 13  MonthlyIncome             4410 non-null   int64  
 14  NumCompaniesWorked        4391 non-null   float64
 15  Over18                    4410 non-null   object 
 16  PercentSalaryHike         4410 non-null   int64  
 17  StandardHours             4410 non-null   int64  
 18  StockOptionLevel          4410 non-null   int64  
 19  TotalWorkingYears         4401 non-null   float64
 20  TrainingTimesLastYear     4410 non-null   int64  
 21  YearsAtCompany            4410 non-null   int64  
 22  YearsSinceLastPromotion   4410 non-null   int64  
 23  YearsWithCurrManager      4410 non-null   int64  
 24  EnvironmentSatisfaction   4385 non-null   float64
 25  JobSatisfaction           4390 non-null   float64
 26  WorkLifeBalance           4372 non-null   float64
 27  JobInvolvement            4410 non-null   int64  
 28  PerformanceRating         4410 non-null   int64  
dtypes: float64(5), int64(16), object(8)
memory usage: 1.0+ MB
```

In [4]:
```
# check total no of unique values under each variable
EA.nunique()
```

Out[4]:
```
EmployeeID                4410
Age                         43
Attrition                    2
BusinessTravel               3
Department                   3
DistanceFromHome            29
Education                    5
EducationField               6
EmployeeCount                1
Gender                       2
JobLevel                     5
JobRole                      9
MaritalStatus                3
MonthlyIncome             1349
NumCompaniesWorked          10
Over18                       1
PercentSalaryHike           15
StandardHours                1
StockOptionLevel             4
TotalWorkingYears           40
TrainingTimesLastYear        7
YearsAtCompany              37
YearsSinceLastPromotion     16
YearsWithCurrManager        18
EnvironmentSatisfaction      4
JobSatisfaction              4
WorkLifeBalance              4
JobInvolvement               4
PerformanceRating            2
dtype: int64
```

In [5]:
```
# find statistical values of each columns
EA.describe(include = 'all')
```

Out[5]:

	EmployeeID	Age	Attrition	BusinessTravel	Department	DistanceFromHome
count	4410.000000	4410.000000	4410	4410	4410	4410.000000
unique	NaN	NaN	2	3	3	NaN
top	NaN	NaN	No	Travel_Rarely	Research & Development	NaN
freq	NaN	NaN	3699	3129	2883	NaN
mean	2205.500000	36.923810	NaN	NaN	NaN	9.192517
std	1273.201673	9.133301	NaN	NaN	NaN	8.105026
min	1.000000	18.000000	NaN	NaN	NaN	1.000000
25%	1103.250000	30.000000	NaN	NaN	NaN	2.000000
50%	2205.500000	36.000000	NaN	NaN	NaN	7.000000
75%	3307.750000	43.000000	NaN	NaN	NaN	14.000000
max	4410.000000	60.000000	NaN	NaN	NaN	29.000000

In [6]:
```
# detect null values in the dataset
EA.isnull().sum()  # null values present in some columns
```

Out[6]:
```
EmployeeID                 0
Age                        0
Attrition                  0
BusinessTravel             0
Department                 0
DistanceFromHome           0
Education                  0
EducationField             0
EmployeeCount              0
Gender                     0
JobLevel                   0
JobRole                    0
MaritalStatus              0
MonthlyIncome              0
NumCompaniesWorked        19
Over18                     0
PercentSalaryHike          0
StandardHours              0
StockOptionLevel           0
TotalWorkingYears          9
TrainingTimesLastYear      0
YearsAtCompany             0
YearsSinceLastPromotion    0
YearsWithCurrManager       0
EnvironmentSatisfaction   25
JobSatisfaction           20
WorkLifeBalance           38
JobInvolvement             0
PerformanceRating          0
dtype: int64
```

In [7]:
```
# print name of all columns present in dataset
EA.columns
```

Out[7]:
```
Index(['EmployeeID', 'Age', 'Attrition', 'BusinessTravel', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'Gender', 'JobLevel', 'JobRole', 'MaritalStatus', 'MonthlyIncome',
       'NumCompaniesWorked', 'Over18', 'PercentSalaryHike', 'StandardHour
s',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'YearsAtCompany', 'YearsSinceLastPromotion', 'YearsWithCurrManage
r',
       'EnvironmentSatisfaction', 'JobSatisfaction', 'WorkLifeBalance',
       'JobInvolvement', 'PerformanceRating'],
      dtype='object')
```

EmployeeID column has all unique values, so it is not analyzed properly. Same as EmployeeCount, Over18, StandardHours columns. All have only one value, it is also not analyzed properly. These columns are useless columns, need to be removed.

Remove Useless Columns

In [8]:
```
Useless= ['EmployeeID', 'EmployeeCount', 'Over18', 'StandardHours']

EA = EA.drop(Useless, axis= 1)  # axis= 1 is used for column dropping
```

In [9]:
```
# print name of all useful columns present in dataset
EA.columns
```

Out[9]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'Department', 'DistanceFromHo
me',
       'Education', 'EducationField', 'Gender', 'JobLevel', 'JobRole',
       'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked',
       'PercentSalaryHike', 'StockOptionLevel', 'TotalWorkingYears',
       'TrainingTimesLastYear', 'YearsAtCompany', 'YearsSinceLastPromotio
n',
       'YearsWithCurrManager', 'EnvironmentSatisfaction', 'JobSatisfactio
n',
       'WorkLifeBalance', 'JobInvolvement', 'PerformanceRating'],
      dtype='object')
```

Missing Value Treatment

Columns which have continuous variable, missing values can be replaced with median of that columns.
Columns which have categorical variable, missing values can be replaced with mode of that columns.

In [10]:
```
EA['NumCompaniesWorked'].fillna(value = EA['NumCompaniesWorked'].mode()[0],
inplace= True)
```

In [11]:
```
EA['EnvironmentSatisfaction'].fillna(value = EA['EnvironmentSatisfaction'].
mode()[0], inplace= True)
```

In [12]:
```
EA['JobSatisfaction'].fillna(value = EA['JobSatisfaction'].mode()[0], inpla
ce= True)
```

In [13]:
```
EA['WorkLifeBalance'].fillna(value = EA['WorkLifeBalance'].mode()[0], inpla
ce= True)
```

In [14]:
```
EA['TotalWorkingYears'].fillna(value = EA['TotalWorkingYears'].median(), in
place= True)
```

In [ ]:
```

```

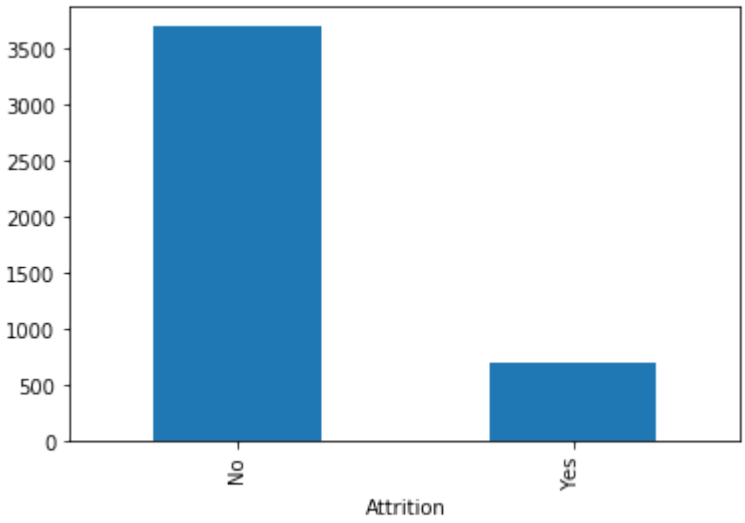
Visual Exploratory Data Analysis

here, start the visualization part

Distribution of Target Variable

In [15]:
```
# As target variable i.e. Attrition column is a categorical column, so here bar chart is applicable

EA.groupby('Attrition').size().plot(kind= 'bar')
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23c2a72fac8>
```



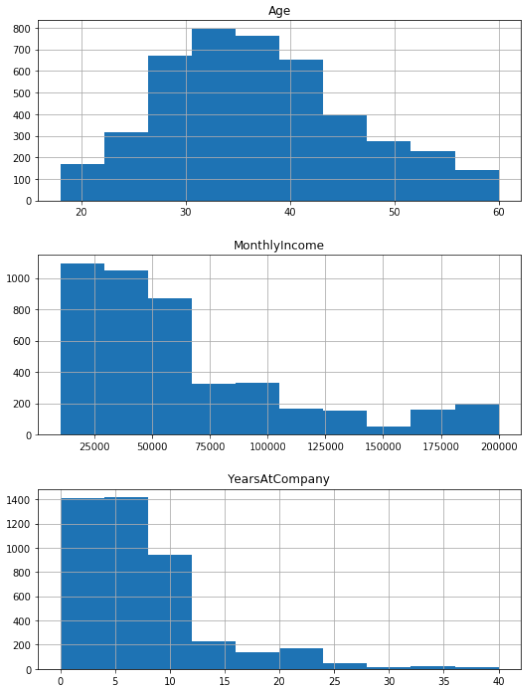
In the problem statement, it is mentioned that only 15% attrition rate is present, so it is quite obvious that 'No' subcategory is much higher than 'Yes' subcategory. But, both bars have enough data to explain further analysis.

Distribution of Continuous Variable

Those variables which are continuous in nature like Age, Monthly Income require 'Histogram' type of chart to explain the distribution and check outliers

In [16]:
```
Continuous= ['Age', 'DistanceFromHome', 'MonthlyIncome', 'TotalWorkingYear
s', 'YearsAtCompany']
EA.hist(Continuous, figsize=(20,12))
```

Out[16]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023C2A6A1C8
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023C2AE6BD0
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023C2AEA474
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023C2AEDE6C
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000023C2AF167C
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023C2AF4D8C
8>]],
      dtype=object)
```

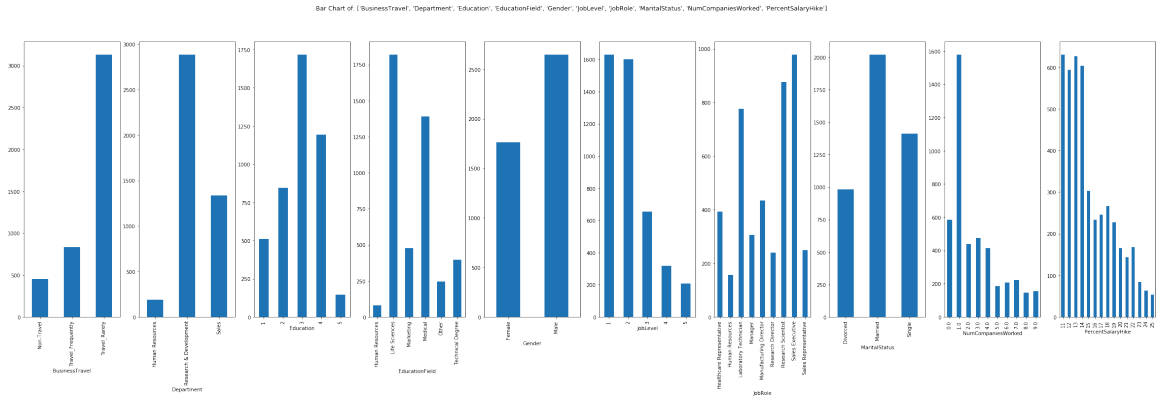


There is no outliers in any columns, so no need to treat outliers

Distribution of Categorical Variable

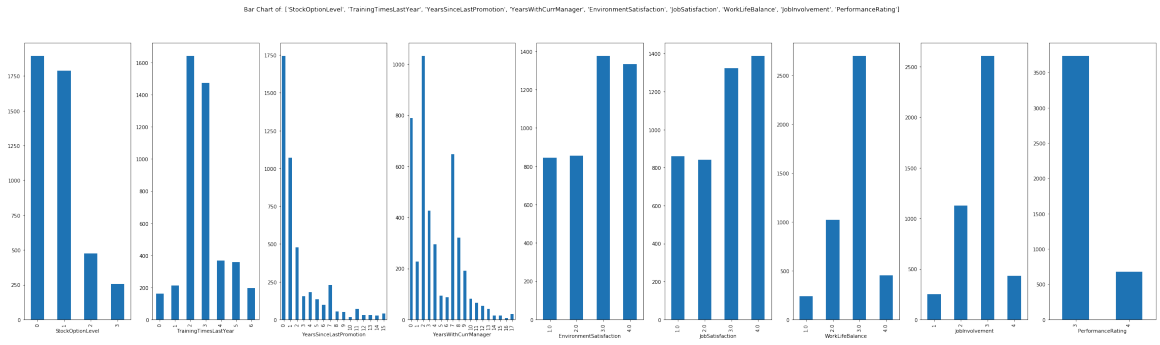
Those variables which are categorical in nature require 'Bar Chart' type of chart to explain the distribution

In [17]:
```
Categorical= ['BusinessTravel', 'Department', 'Education', 'EducationFiel
d', 'Gender', 'JobLevel', 'JobRole',
              'MaritalStatus', 'NumCompaniesWorked', 'PercentSalaryHike']

fig, canvas= plt.subplots(nrows= 1, ncols= len(Categorical), figsize=(40,1
0))
fig.suptitle('Bar Chart of: '+ str(Categorical))

for predictor, i in zip(Categorical, range(len(Categorical))):
    EA.groupby(predictor).size().plot(kind= 'bar', ax= canvas[i])
```



here all bars have sufficient information to explain, no such outliers are present

In [18]:
```
Categorical= ['StockOptionLevel', 'TrainingTimesLastYear', 'YearsSinceLastP
romotion', 'YearsWithCurrManager',
      'EnvironmentSatisfaction', 'JobSatisfaction', 'WorkLifeBalance', 'Job
Involvement', 'PerformanceRating']

fig, canvas= plt.subplots(nrows= 1, ncols= len(Categorical), figsize=(40,1
0))
fig.suptitle('Bar Chart of: '+ str(Categorical))

for predictor, i in zip(Categorical, range(len(Categorical))):
    EA.groupby(predictor).size().plot(kind= 'bar', ax= canvas[i])
```



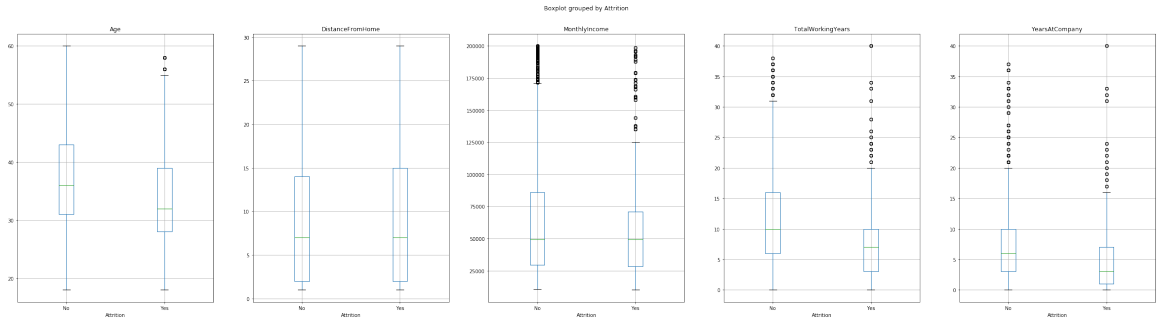
here all bars have sufficient information to explain, no such outliers are present

Feature Selection --- Bi-Variate Analysis

distribution of continuous variables according to target variable

Here, Target Variable is Categorical one. So, for Continuous variable vs Categorical Variable situation, box plot can be good option to analysis. If the boxes are mis-aligned, then that particular column has a correlation with target column, so they are good for Machine Learning analysis, otherwise not. One can detect this more accurately by using statistical Anova Test. If Anova Test Result, p-value is less than 0.05, then one can consider this variable for Machine learning analysis.

In [19]:
```
Continuous= ['Age', 'DistanceFromHome', 'MonthlyIncome', 'TotalWorkingYear
s', 'YearsAtCompany']

fig, canvas = plt.subplots(nrows= 1, ncols= len(Continuous), figsize= (40,1
0))

for predictor, i in zip(Continuous, range(len(Continuous))):
    EA.boxplot(column= predictor, by= 'Attrition', vert= True, ax= canvas
[i])
```



In [20]:
```
# ANOVA Test

from scipy.stats import f_oneway

def FunctionAnova(inpData, TargetVariable, PredictorList):

    FinalPredictor=[]

    for predictor in PredictorList:
        AnovaTest= inpData.groupby(TargetVariable)[predictor].apply(list)
        AnovaResult= f_oneway(*AnovaTest)

        if(AnovaResult[1] < 0.05):
            print(predictor, 'is correlated with ', TargetVariable, '|| P-V
alue: ', AnovaResult[1])
            FinalPredictor.append(predictor)

        else:
            print(predictor, 'is NOT correlated with ', TargetVariable, '||
P-Value: ', AnovaResult[1])

    return(FinalPredictor)
```

In [21]:
```
# call the function

Continuous= ['Age', 'DistanceFromHome', 'MonthlyIncome', 'TotalWorkingYear
s', 'YearsAtCompany']
FunctionAnova(inpData = EA, TargetVariable = 'Attrition', PredictorList = C
ontinuous)
```

```
Age is correlated with  Attrition || P-Value:  1.9968016158845057e-26
DistanceFromHome is NOT correlated with  Attrition || P-Value:  0.51828604
28065224
MonthlyIncome is correlated with  Attrition || P-Value:  0.038427484905971
684
TotalWorkingYears is correlated with  Attrition || P-Value:  5.26004001021
0012e-30
YearsAtCompany is correlated with  Attrition || P-Value:  3.16388312248436
e-19
```

Out[21]:
```
['Age', 'MonthlyIncome', 'TotalWorkingYears', 'YearsAtCompany']
```

Featured Engineering

Featured Engineering is a process to include a new column based on some existing columns and some conditions. Here some new columns are added based on some existing columns like Age, MonthlyIncome, TotalWorkingYears, YearsAtCompany to analysis further.

In [22]:
```
# formation of some categorical groups based on Age column

def FunctionAge(inpAge):
    if(inpAge <= 30):
        return 'junior'
    elif(inpAge <= 45):
        return 'senior'
    else:
        return 'highly experienced'

# call the function
EA['AgeGroup'] = EA['Age'].apply(FunctionAge)
```

In [23]:
```
# formation of some categorical groups based on MonthlyIncome column

def FunctionIncome(inpIncome):
    if(inpIncome <= 30000):
        return 'level1'
    elif(inpIncome <= 60000):
        return 'level2'
    elif(inpIncome <= 90000):
        return 'level3'
    elif(inpIncome <= 120000):
        return 'level4'
    elif(inpIncome <= 150000):
        return 'level5'
    else:
        return 'level6'

# call the function
EA['MonthlyIncomeGroup'] = EA['MonthlyIncome'].apply(FunctionIncome)
```

In [24]:
```
# formation of some categorical groups based on TotalWorkingYears column

def FunctionWorkingYears(inpYears):
    if(inpYears < 5):
        return 'less than 5 years'
    elif(inpYears < 10):
        return 'less than 10 years'
    elif(inpYears < 15):
        return 'less than 15 years'
    elif(inpYears < 20):
        return 'less than 20 years'
    elif(inpYears < 30):
        return 'less than 30 years'
    else:
        return 'greater than 30 years'

# caal the function
EA['WorkingYearsGroup'] = EA['TotalWorkingYears'].apply(FunctionWorkingYear
s)
```

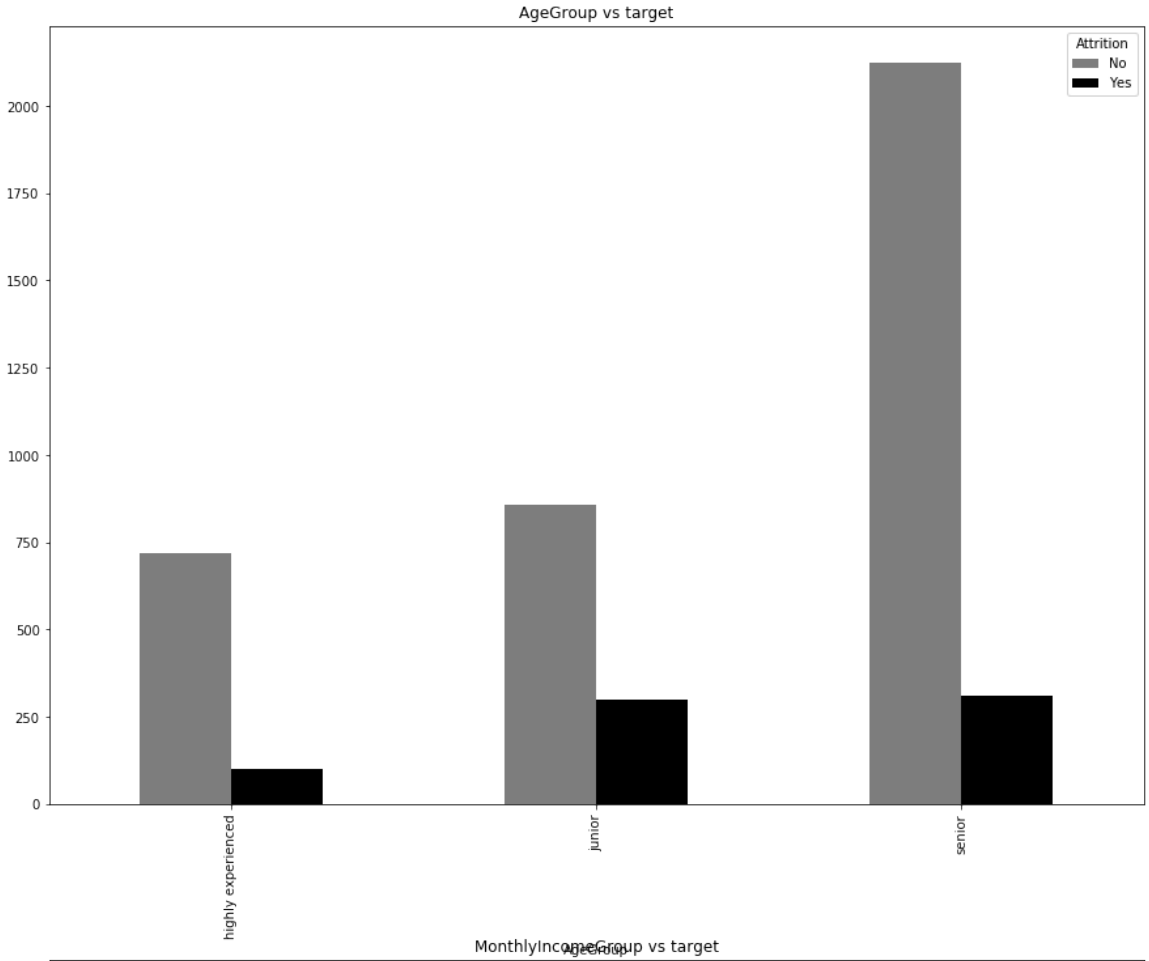
In [25]:
```
# formation of some categorical groups based on YearsAtCompany column

def FunctionYearsCompany(inpYear):
    if(inpYear <= 5):
        return '5 years at company'
    elif(inpYear <= 10):
        return '10 years at company'
    elif(inpYear <= 15):
        return '15 years at company'
    elif(inpYear <= 20):
        return '20 years at company'
    else:
        return 'more than 20 years at company'

# call the function
EA['YearsatCompany'] = EA['YearsAtCompany'].apply(FunctionYearsCompany)
```

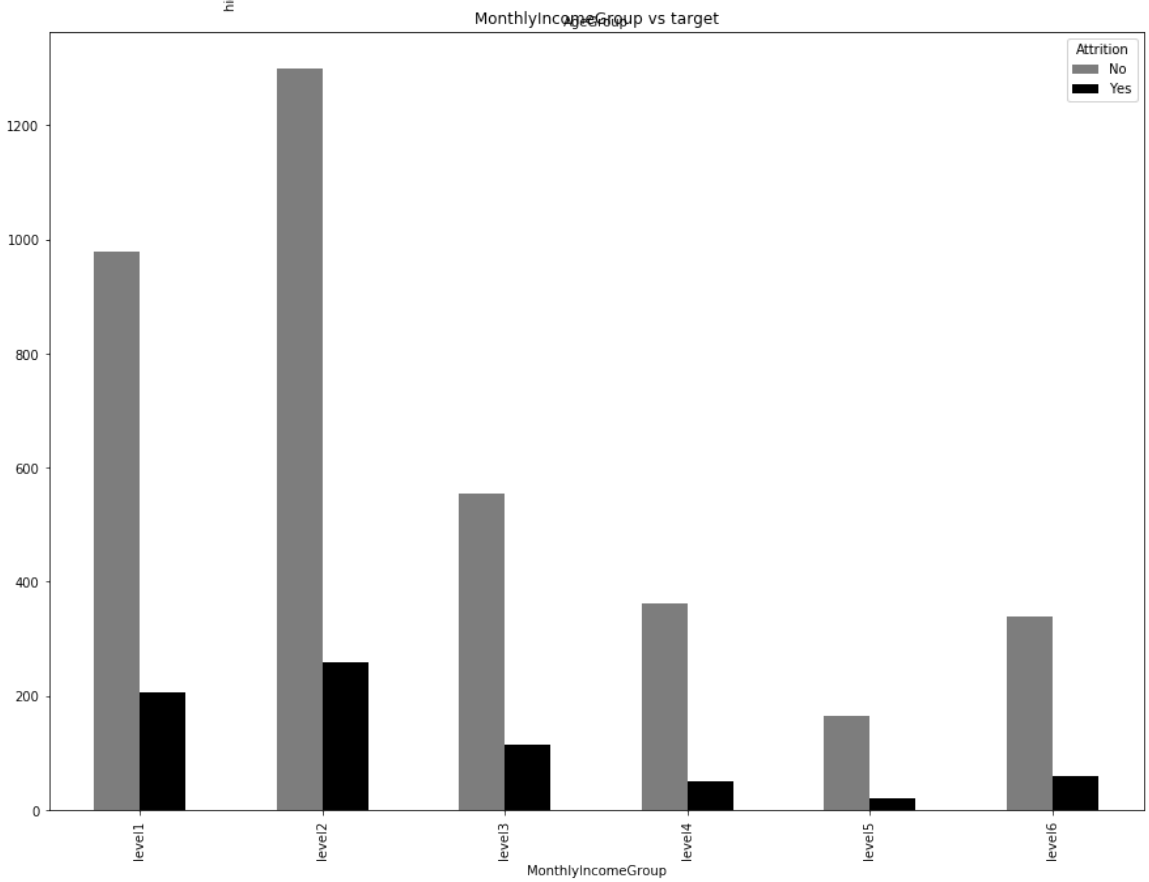
distribution of categorical variables according to target variable

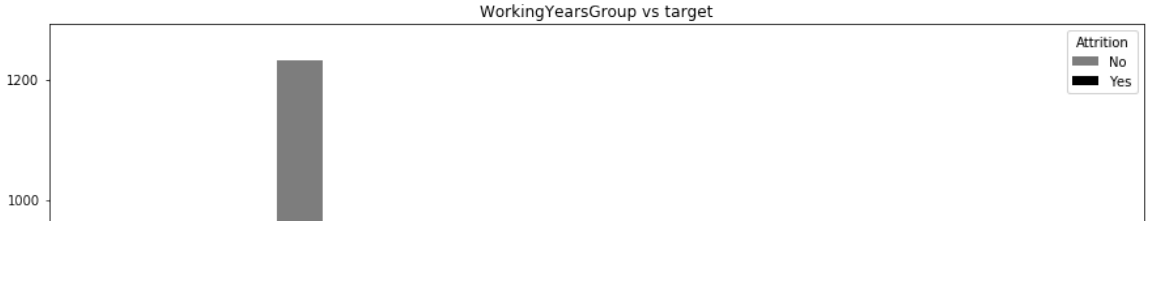
Here, Target Variable is Categorical one. So, for Categorical variable vs Categorical Variable situation, grouped bar chart can be a good option to analysis. If the ratios of cross-tab are different, then that particular column has correlation with target column, so they are good for Machine Learning analysis, otherwise not. One can detect this more accurately by using statistical Chi-Square Test. If Chi-Square Test Result, p-value is less than 0.05, then one can consider this variable for Machine learning analysis.

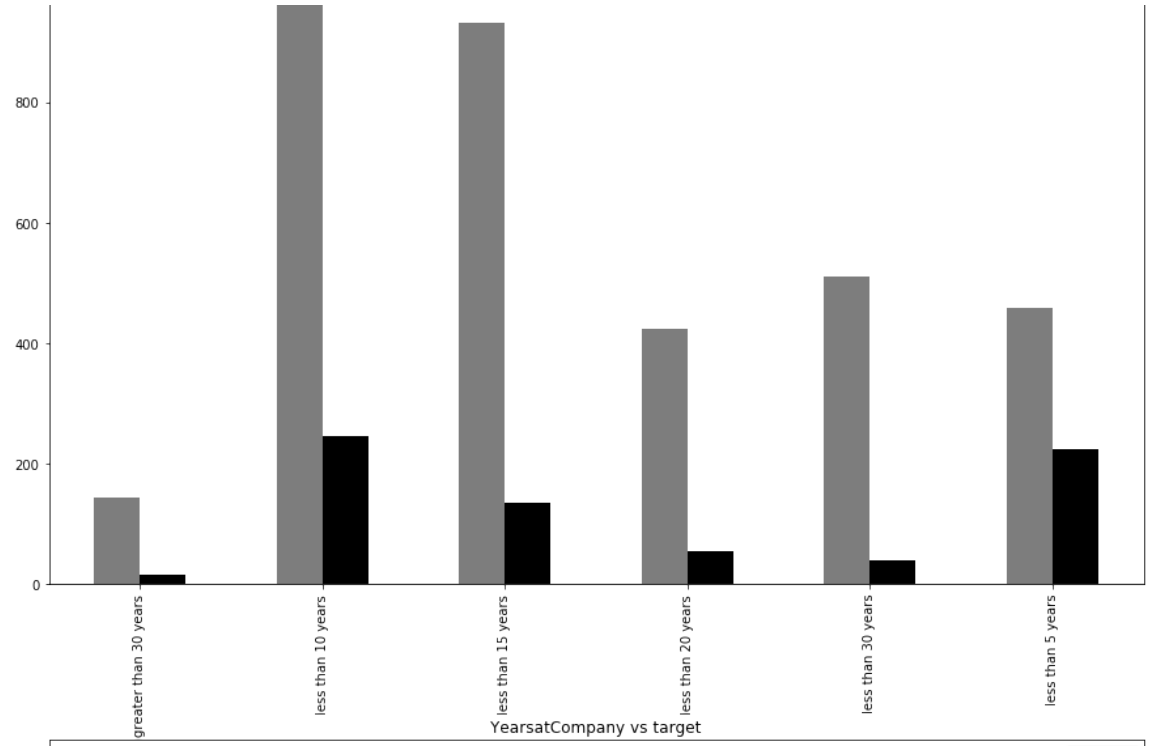
In [26]:

```
Categorical = ['AgeGroup', 'MonthlyIncomeGroup', 'WorkingYearsGroup', 'Yearsat
Company', 'BusinessTravel', 'Department',
               'Education', 'EducationField', 'Gender', 'JobLevel', 'JobRole',
              'MaritalStatus', 'NumCompaniesWorked', 'PercentSalaryHike',
'StockOptionLevel', 'TrainingTimesLastYear',
              'YearsSinceLastPromotion', 'YearsWithCurrManager', 'EnvironmentS
atisfaction', 'JobSatisfaction',
               'WorkLifeBalance', 'JobInvolvement', 'PerformanceRating']

fig, canvas = plt.subplots(ncols=1, nrows= len(Categorical), figsize=(15,300))

for predictor, i in zip(Categorical, range(len(Categorical))):
    CrossTabResult= pd.crosstab(index= EA[predictor], columns= EA['Attrition'])
    CrossTabResult.plot(kind='bar', color= ['grey', 'black'], ax= canvas
[i], title= predictor + ' vs '+ 'target')
```

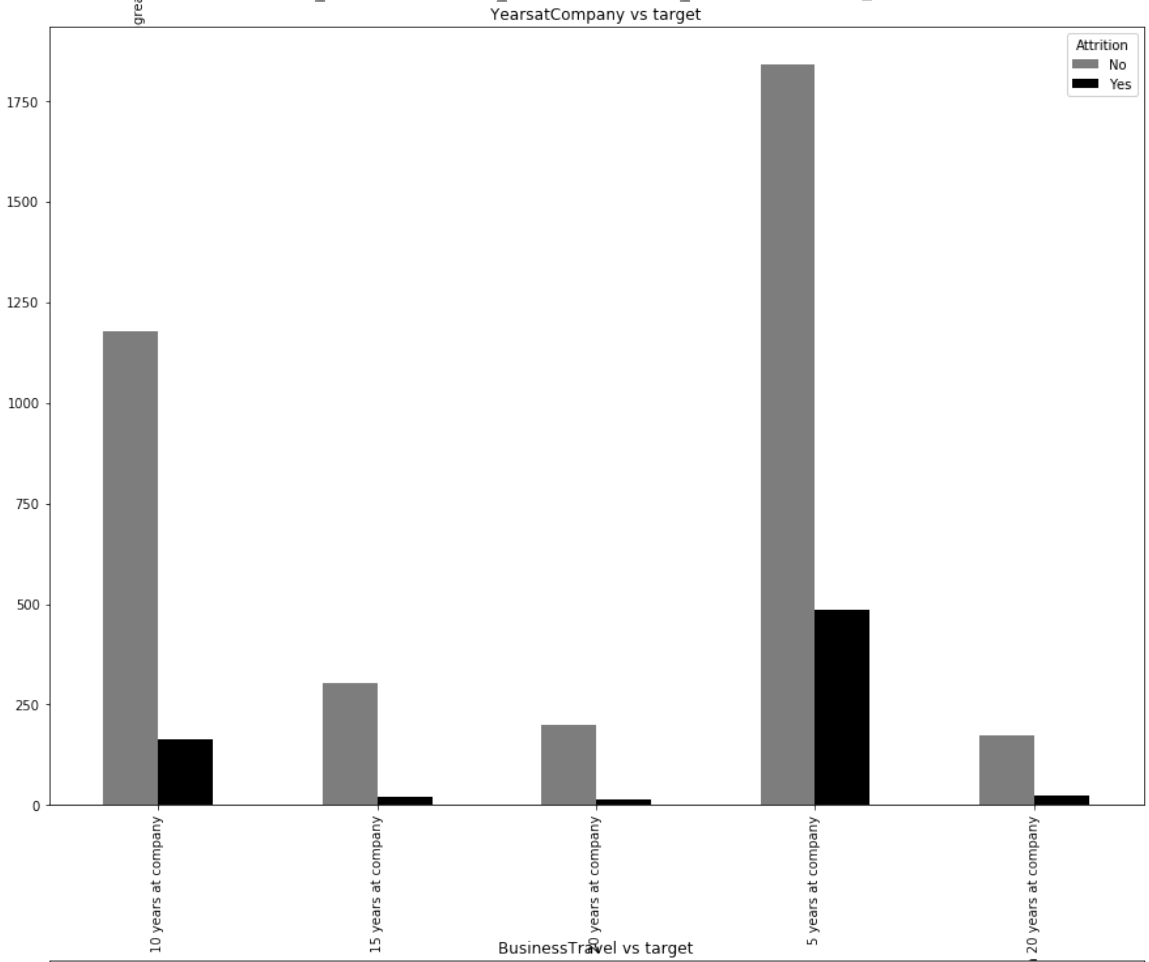

AgeGroup vs target
Attrition
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
highly experienced
junior
senior


MonthlyIncomeGroup vs target
Attrition
No
Yes
1200
1000
800
600
400
200
0
level1
level2
level3
level4
level5
level6
MonthlyIncomeGroup

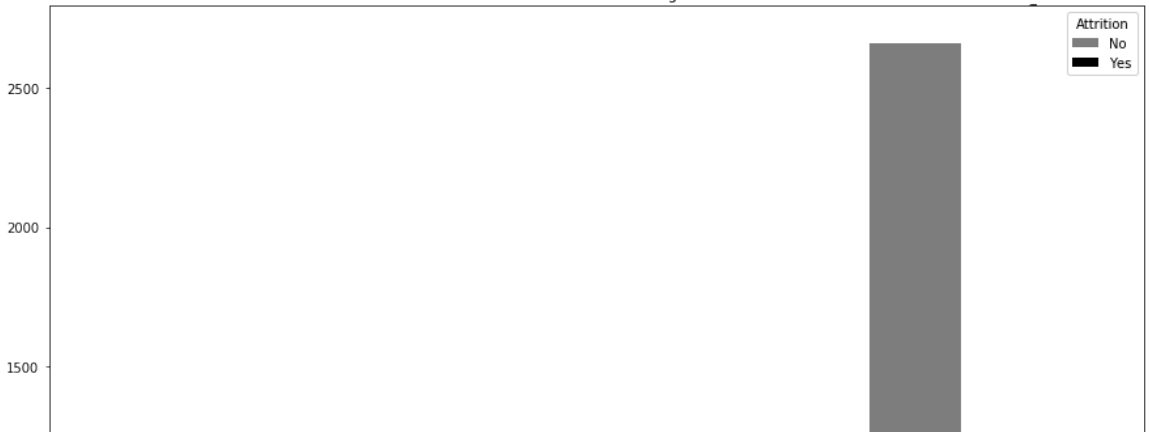

WorkingYearsGroup vs target
Attrition
No
Yes
1200
1000

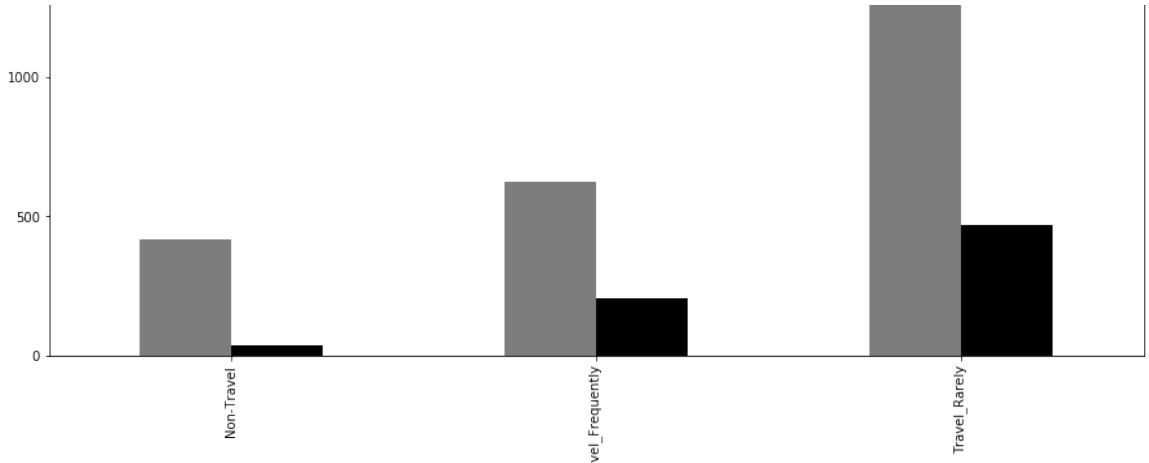


YearsatCompany vs target



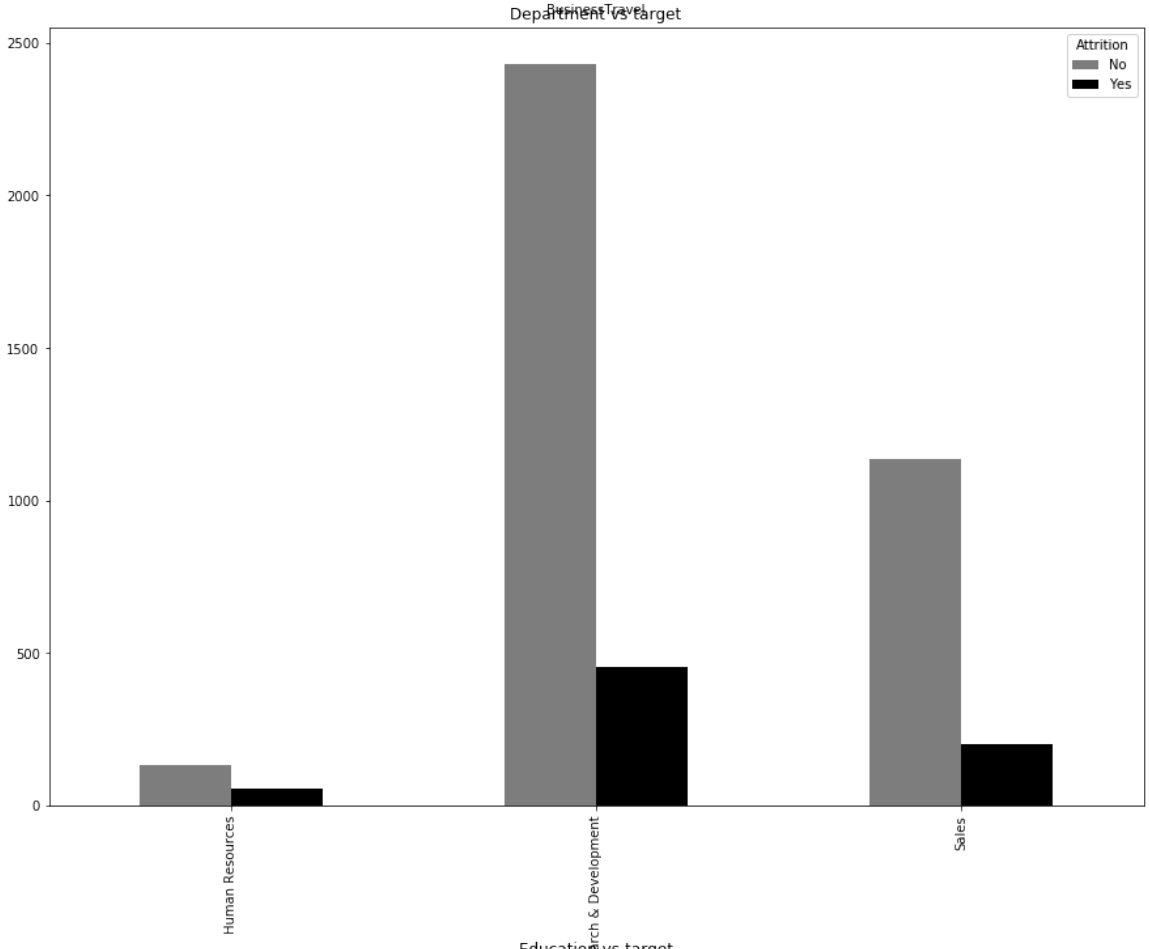
BusinessTravel vs target



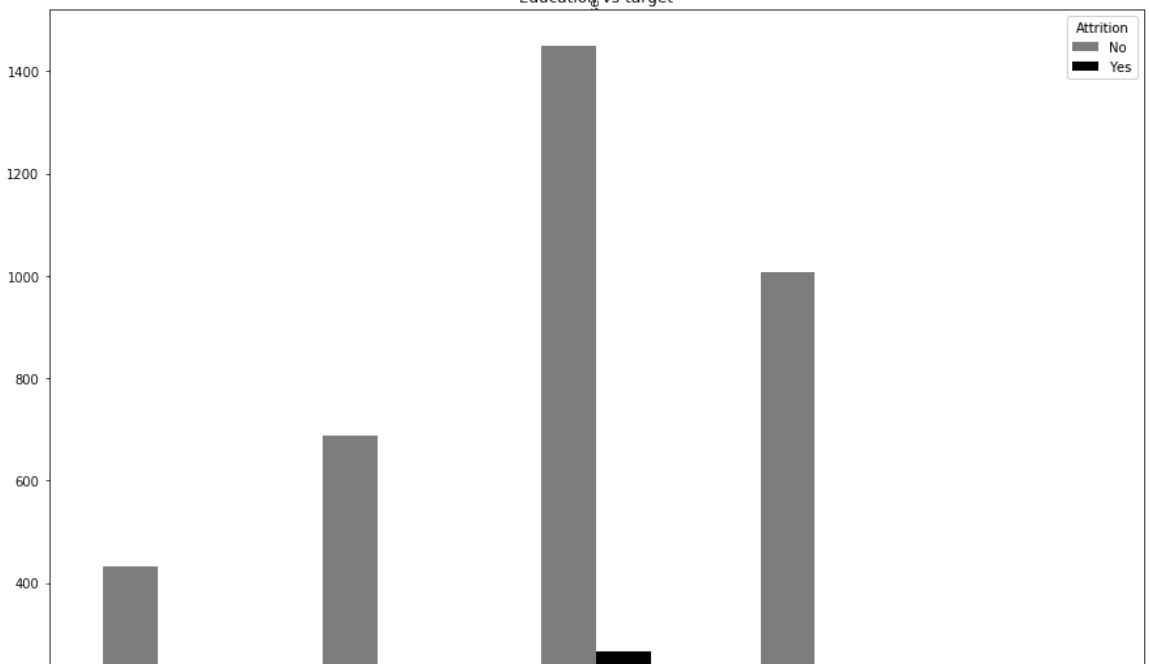


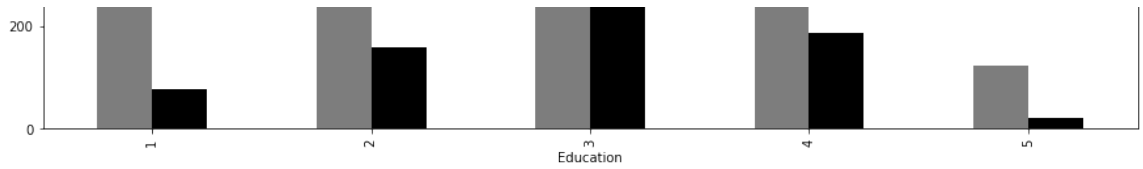
BusinessTravel

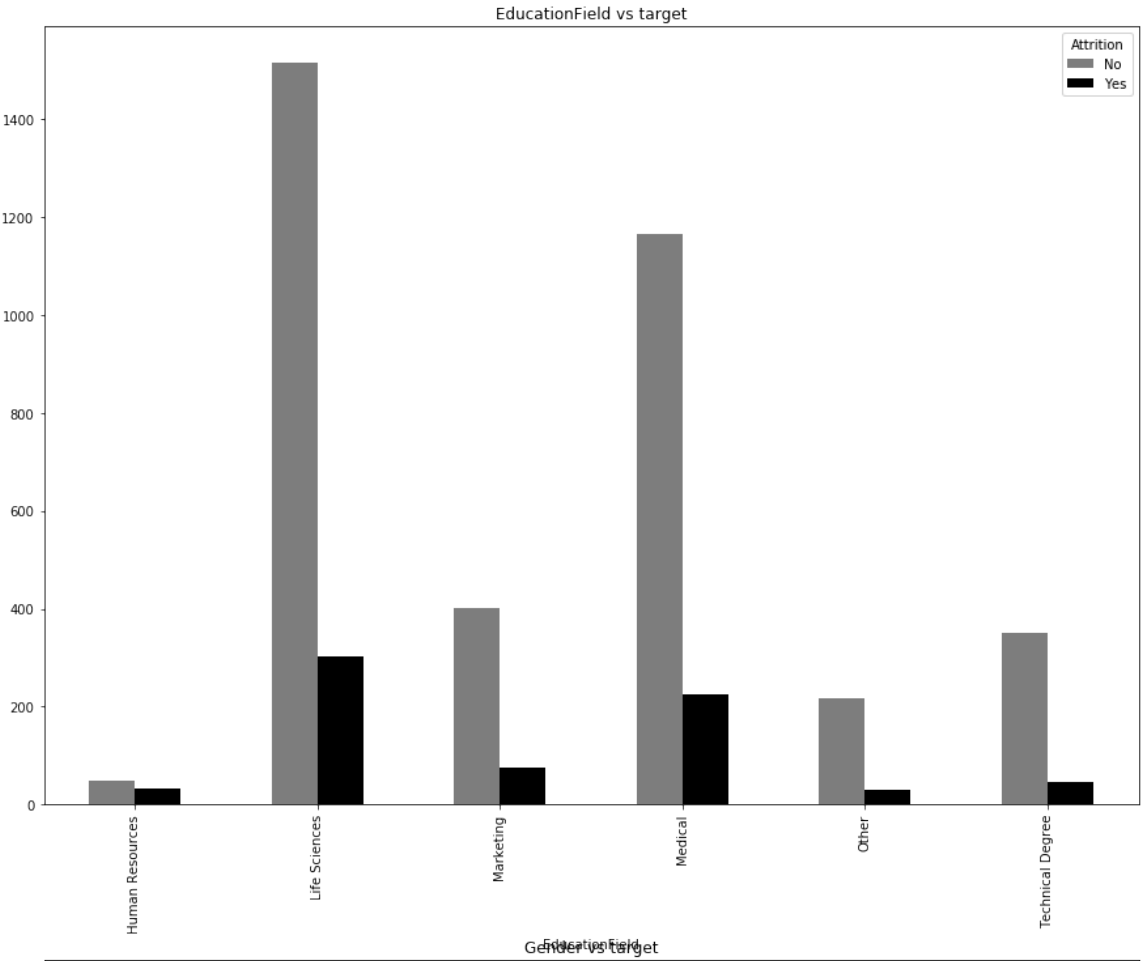
Department vs target

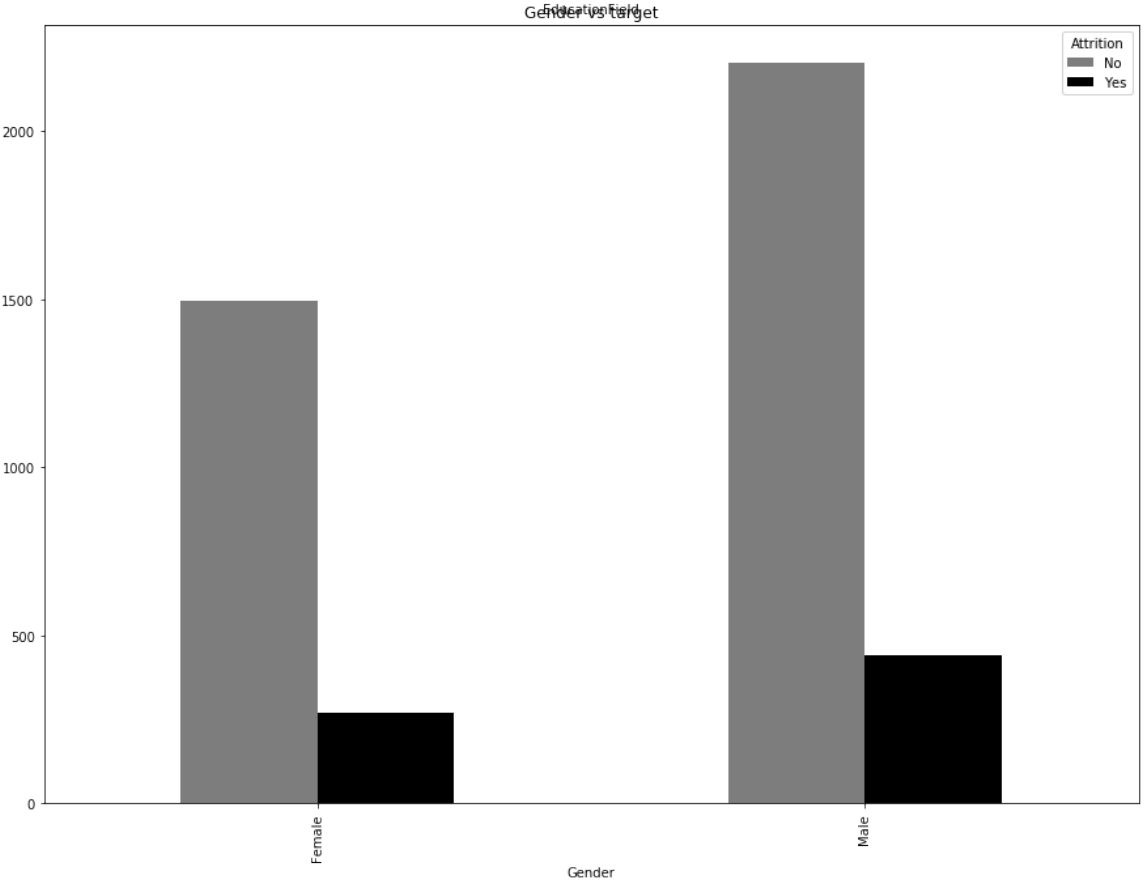


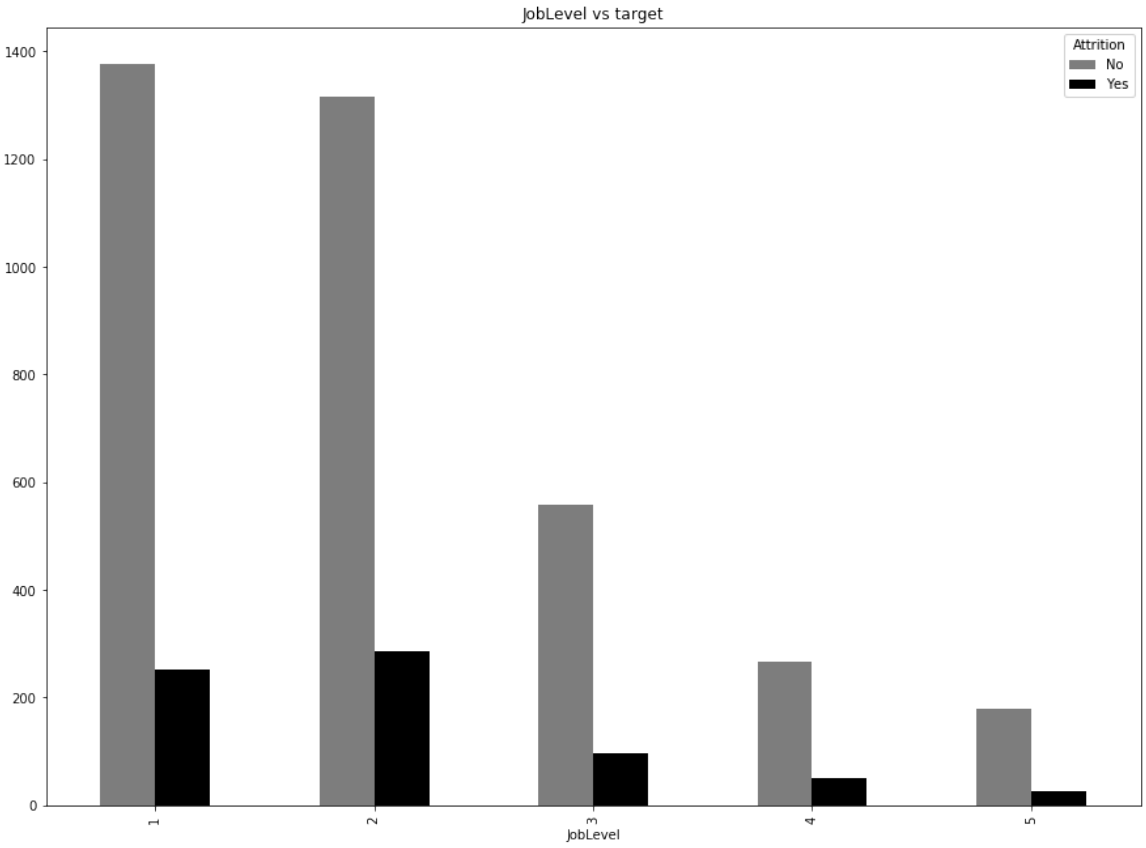
Education vs target

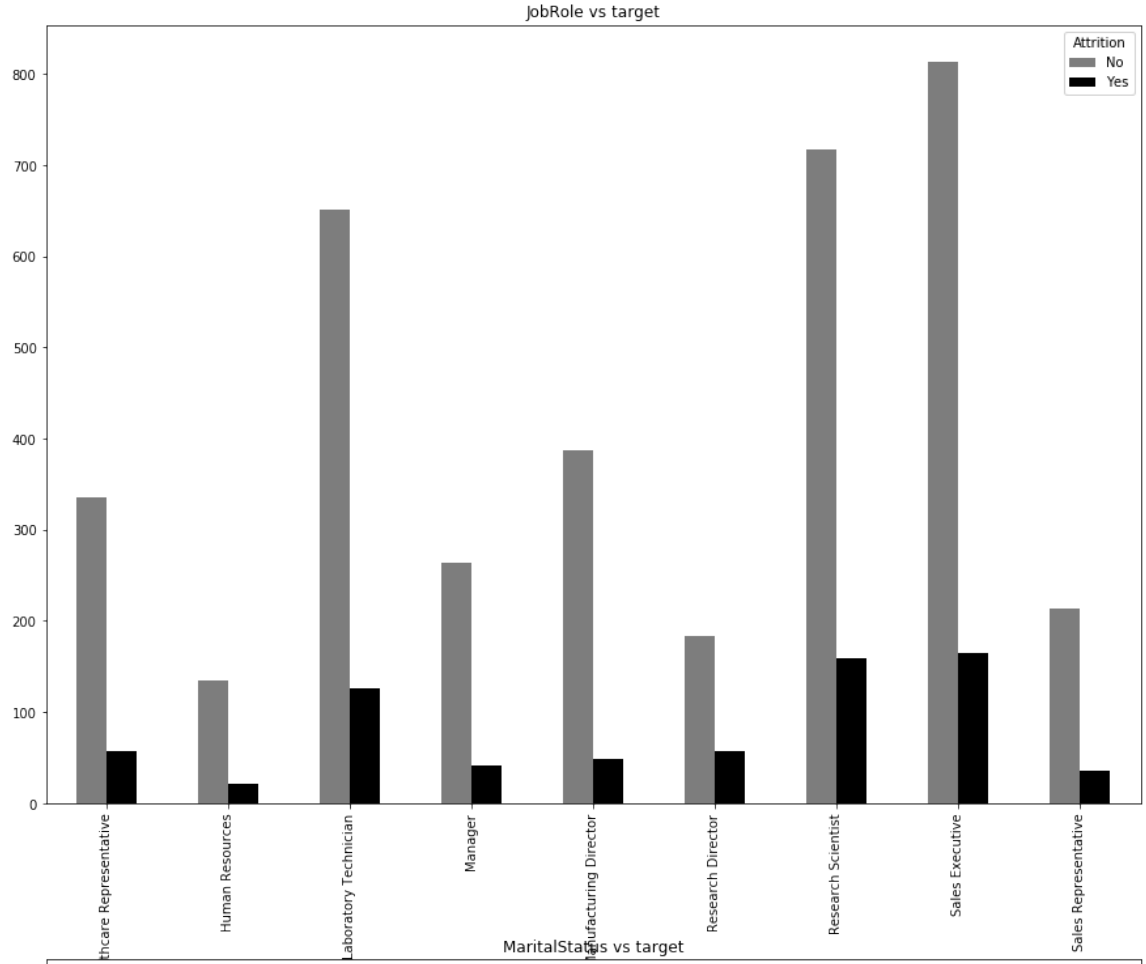


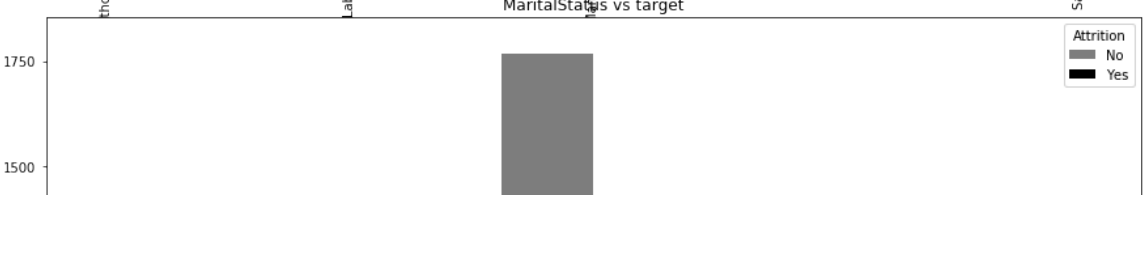

200
0
1
2
3
4
5
Education

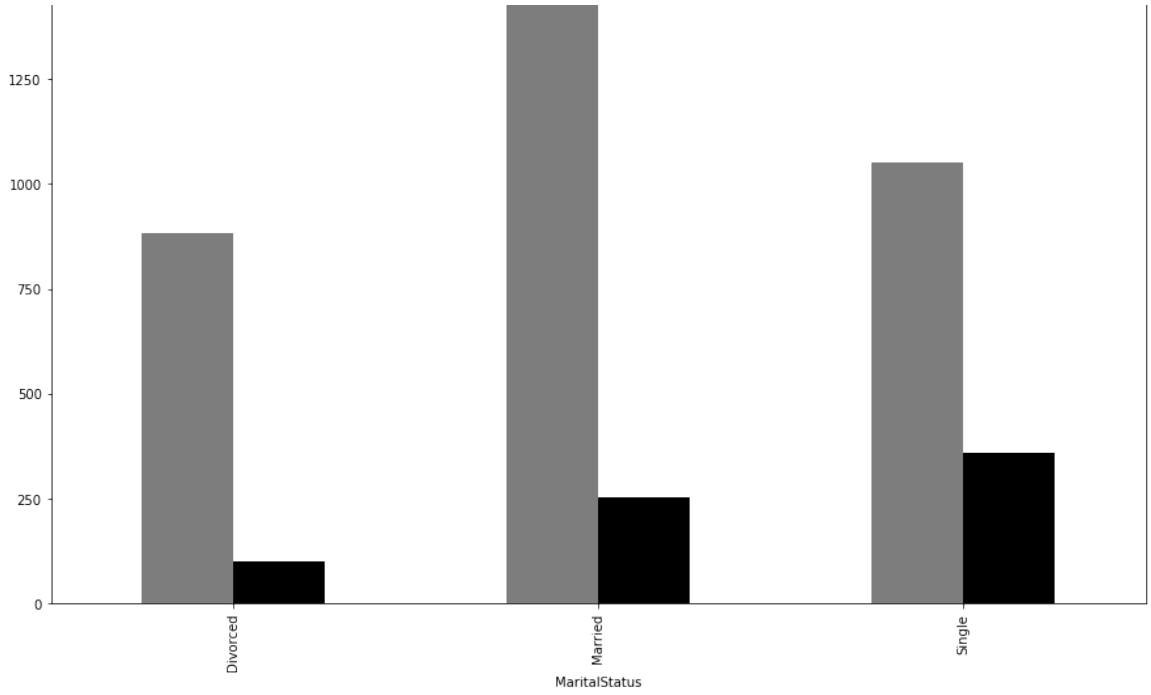

EducationField vs target
Attrition
No
Yes
1400
1200
1000
800
600
400
200
0
Human Resources
Life Sciences
Marketing
Medical
Other
Technical Degree
EducationField

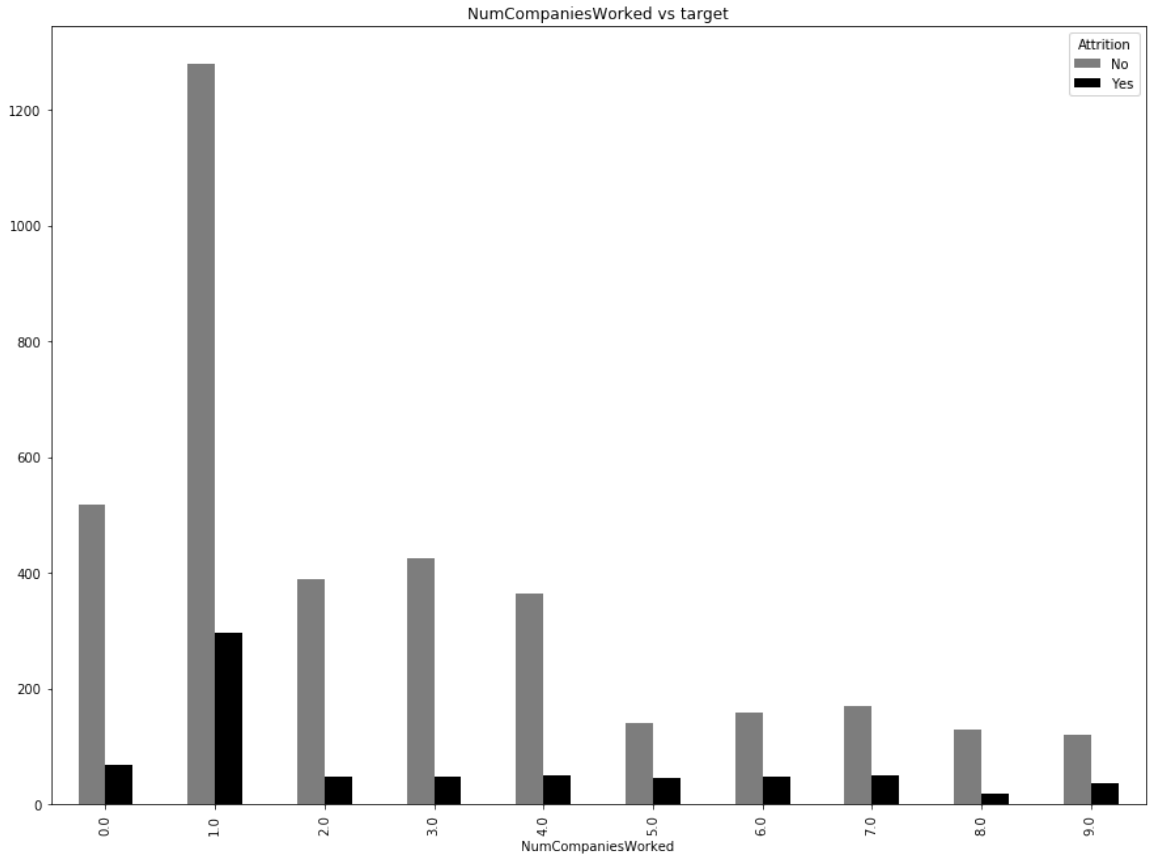

Gender vs target
Attrition
No
Yes
2000
1500
1000
500
0
Female
Male
Gender

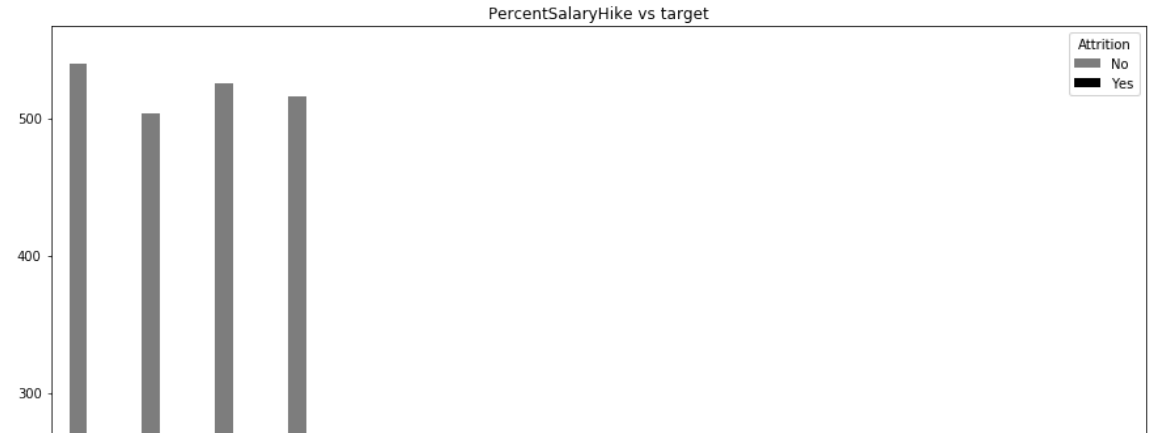

JobLevel vs target
Attrition
No
Yes
1400
1200
1000
800
600
400
200
0
1
2
3
4
5
JobLevel

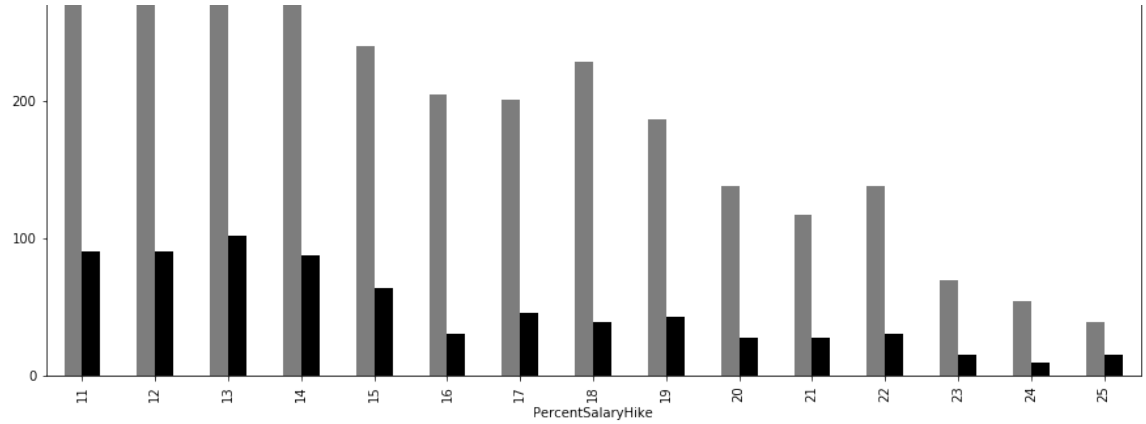

JobRole vs target
Attrition
No
Yes
800
700
600
500
400
300
200
100
0
thcare Representative
Human Resources
Laboratory Technician
Manager
ufacturing Director
Research Director
Research Scientist
Sales Executive
Sales Representative

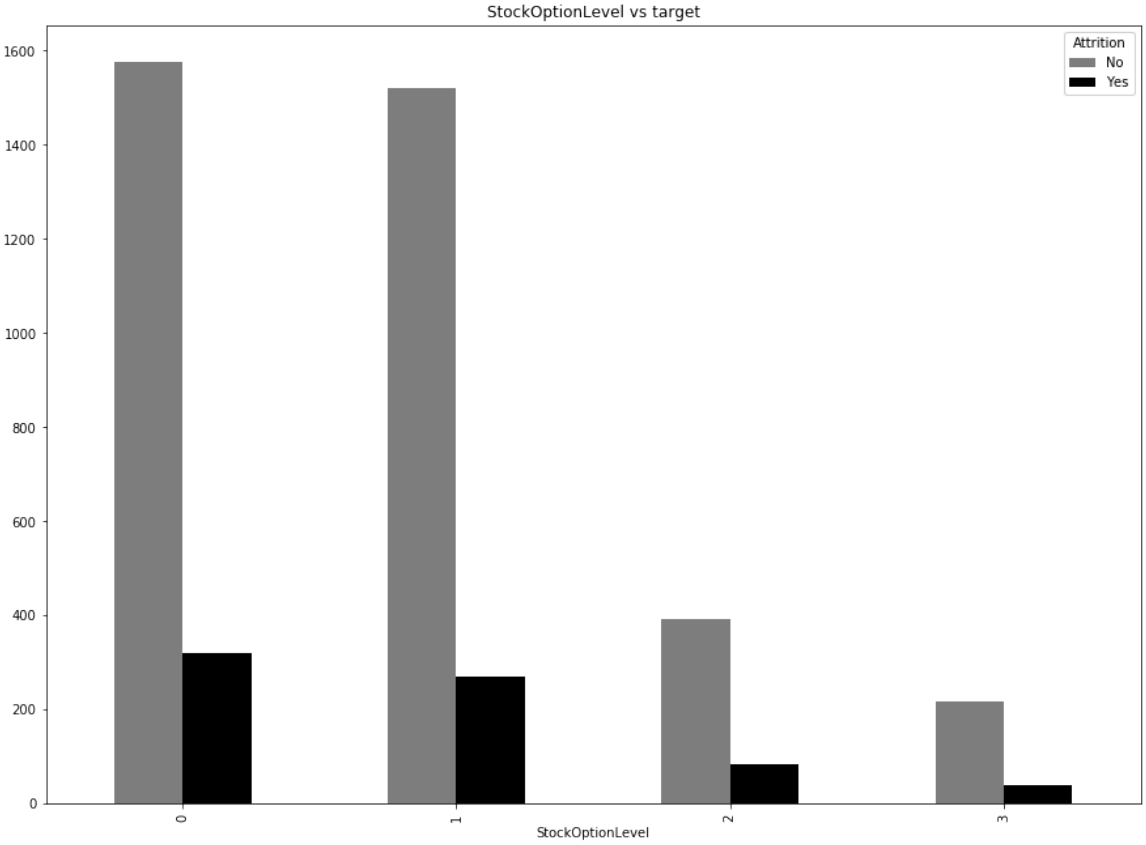

MaritalStatus vs target
Attrition
No
Yes
1750
1500

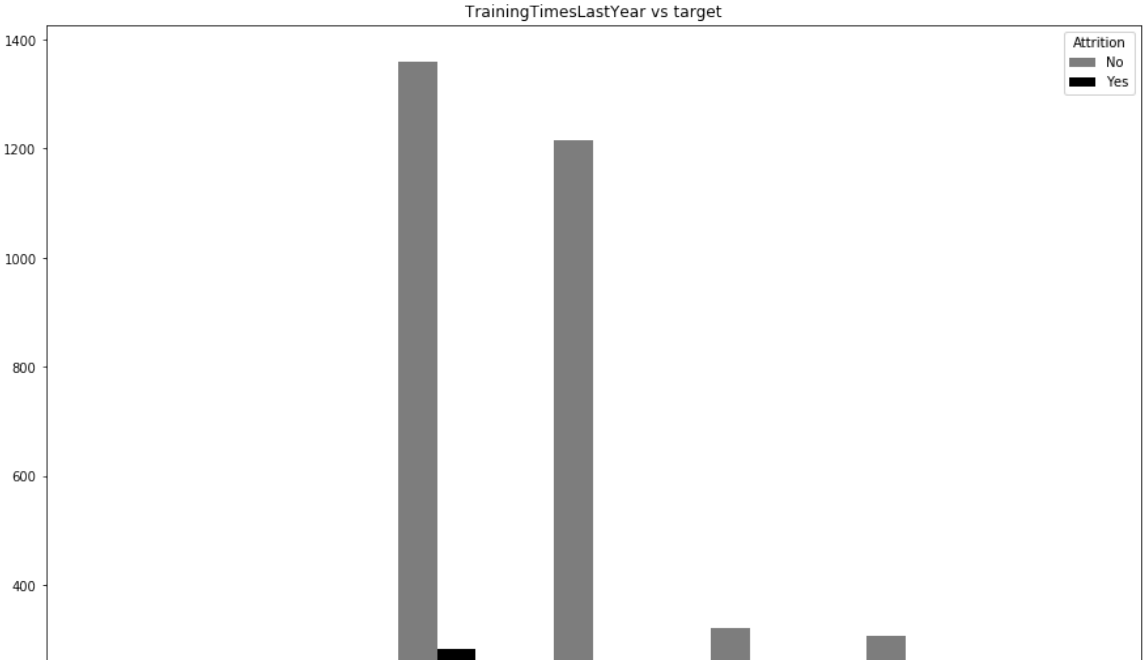

1250
1000
750
500
250
0
Divorced
Married
Single
MaritalStatus

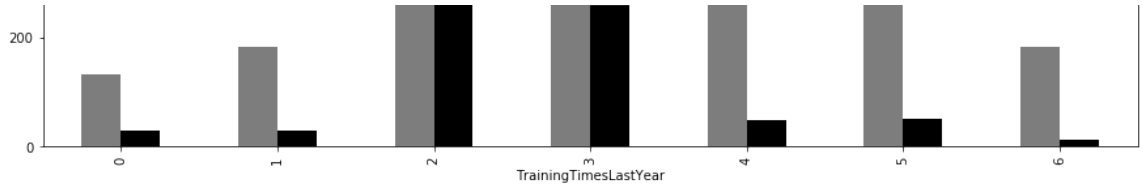

NumCompaniesWorked vs target
Attrition
No
Yes
1200
1000
800
600
400
200
0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
NumCompaniesWorked

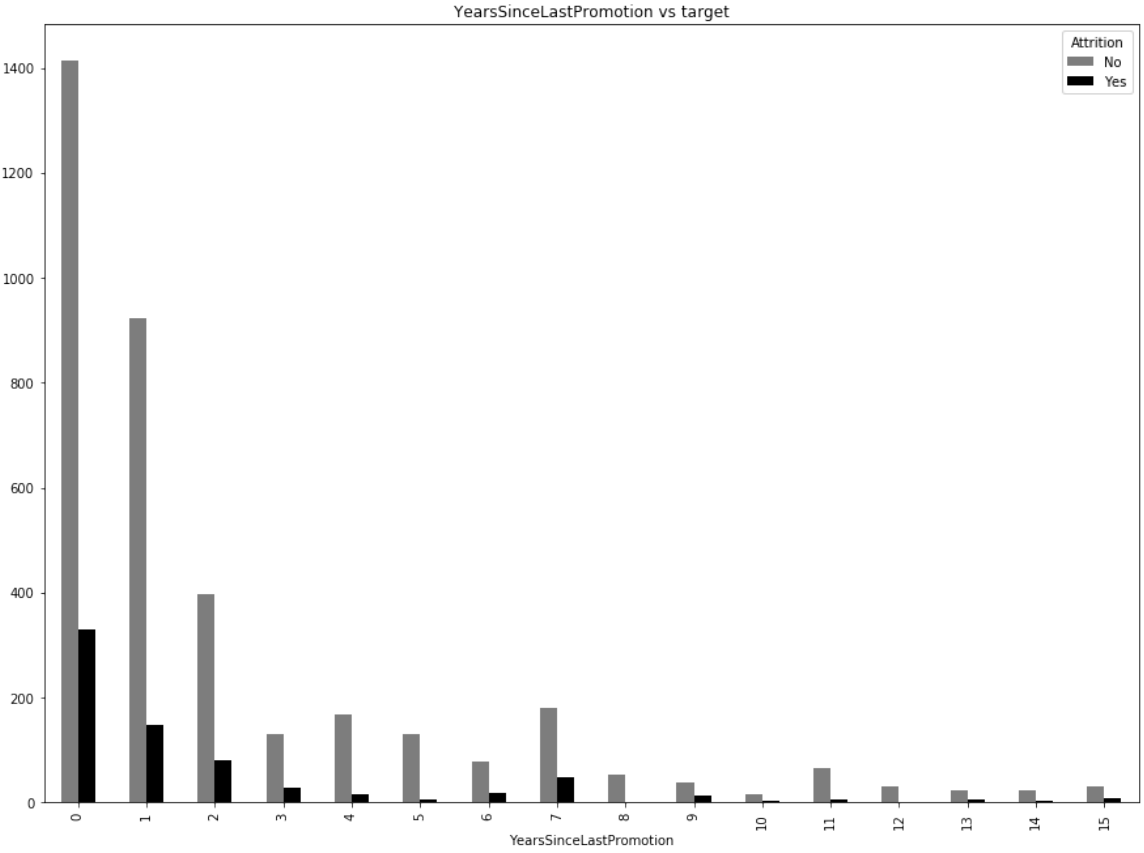

PercentSalaryHike vs target
Attrition
No
Yes
500
400
300

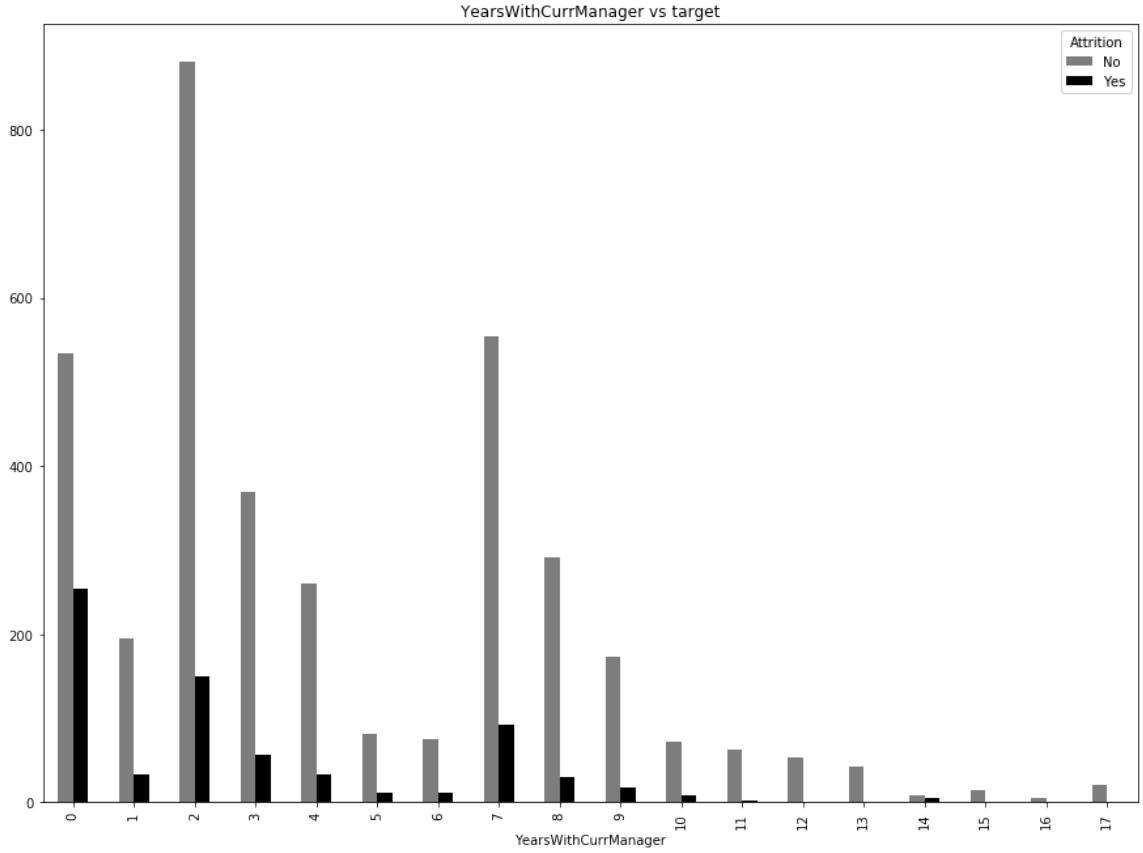

200
100
0
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
PercentSalaryHike

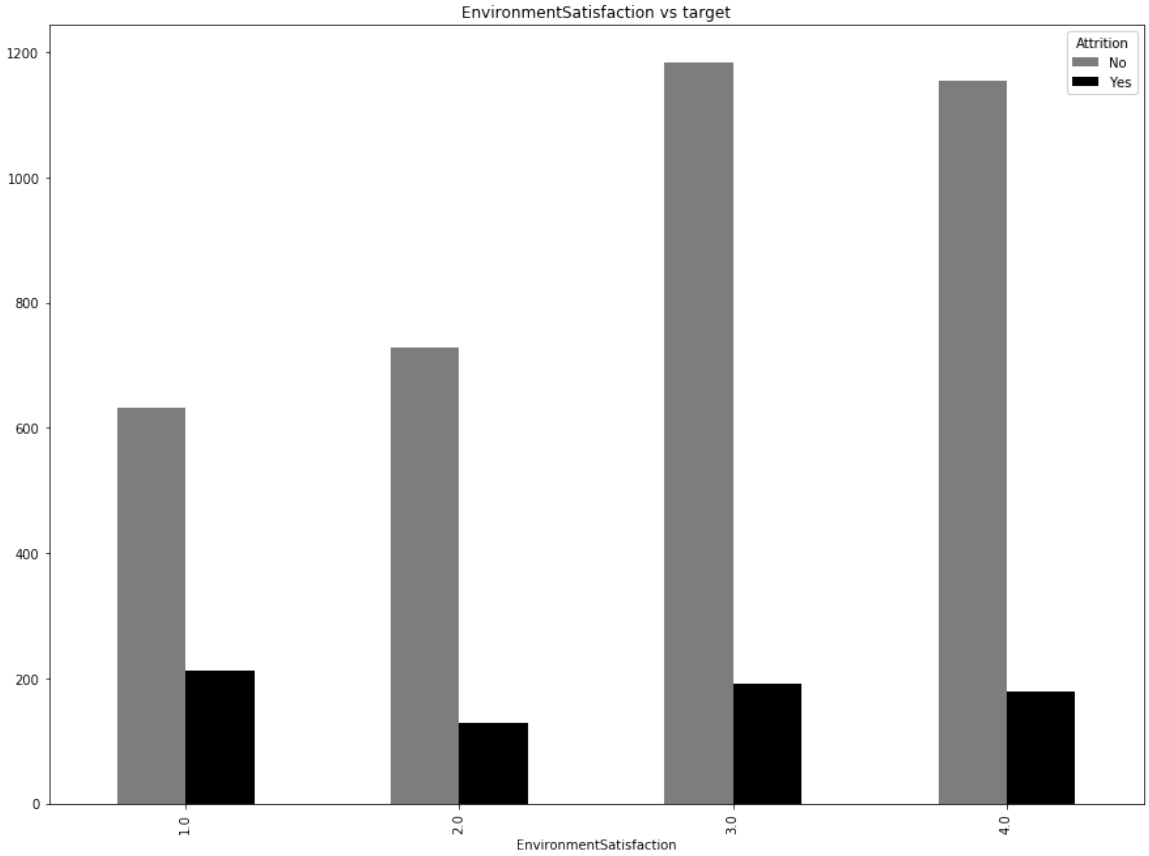

StockOptionLevel vs target
Attrition
No
Yes
1600
1400
1200
1000
800
600
400
200
0
0
1
2
3
StockOptionLevel

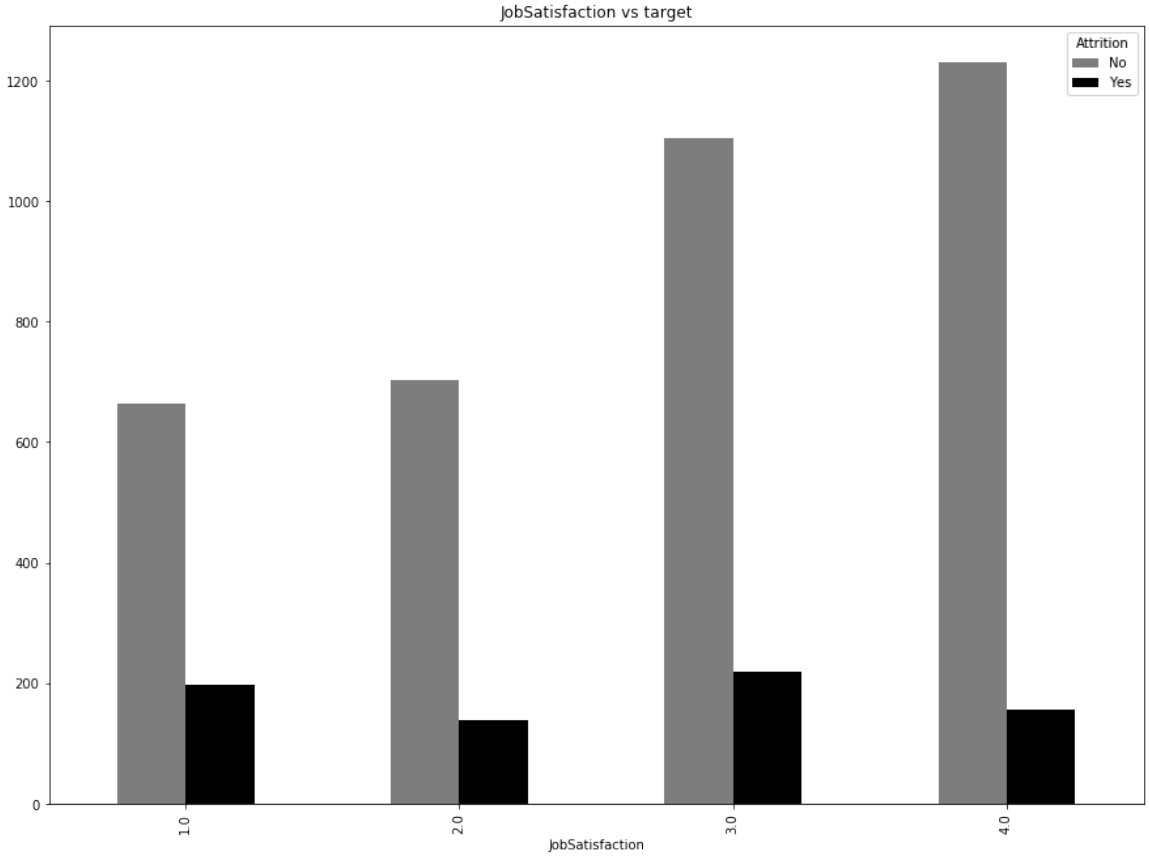

TrainingTimesLastYear vs target
Attrition
No
Yes
1400
1200
1000
800
600
400


200
0
0
1
2
3
4
5
6
TrainingTimesLastYear

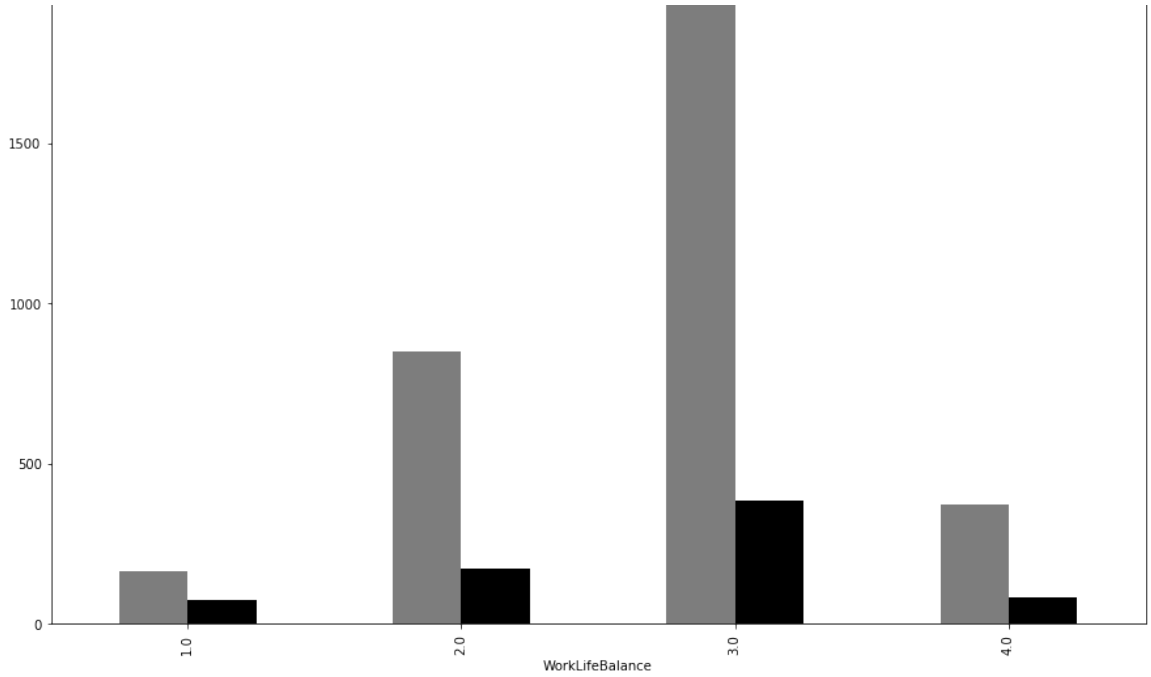

YearsSinceLastPromotion vs target
Attrition
No
Yes
1400
1200
1000
800
600
400
200
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
YearsSinceLastPromotion

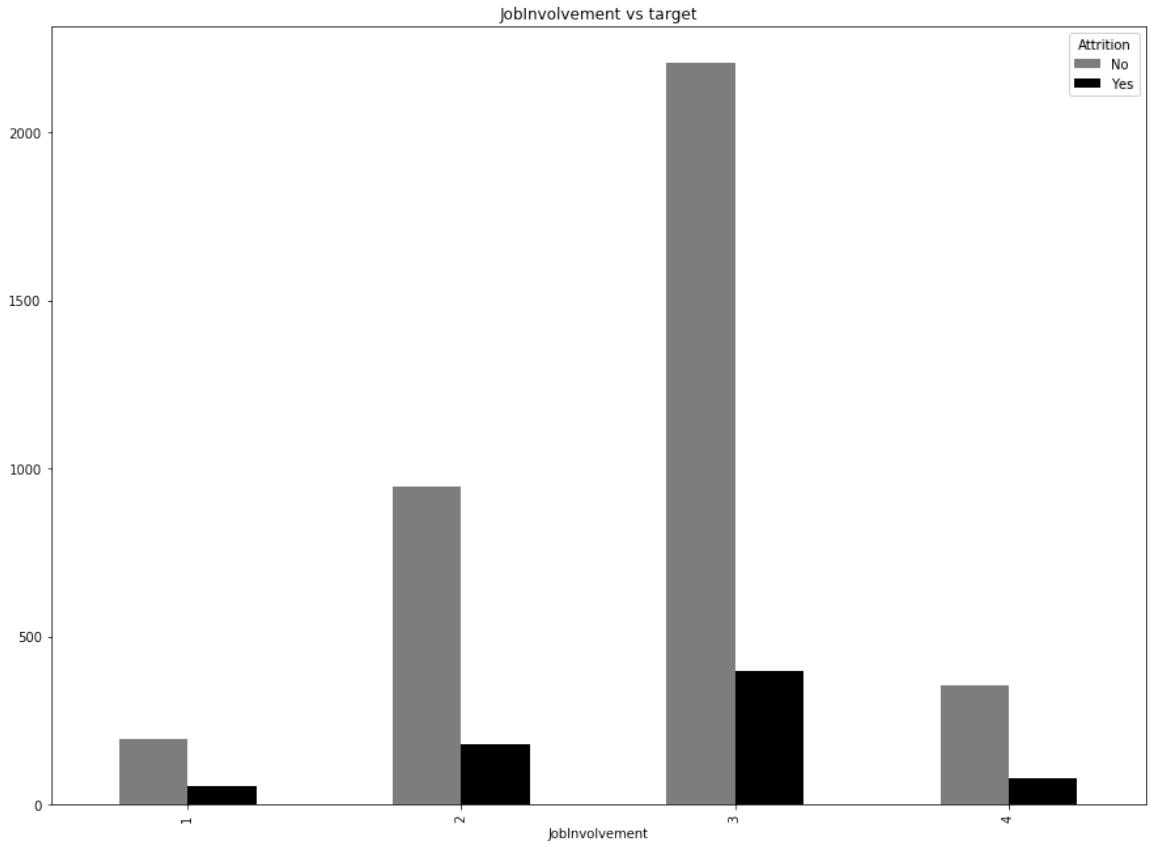

YearsWithCurrManager vs target
Attrition
No
Yes
800
600
400
200
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
YearsWithCurrManager


EnvironmentSatisfaction vs target
Attrition
No
Yes
1200
1000
800
600
400
200
0
1.0
2.0
3.0
4.0
EnvironmentSatisfaction

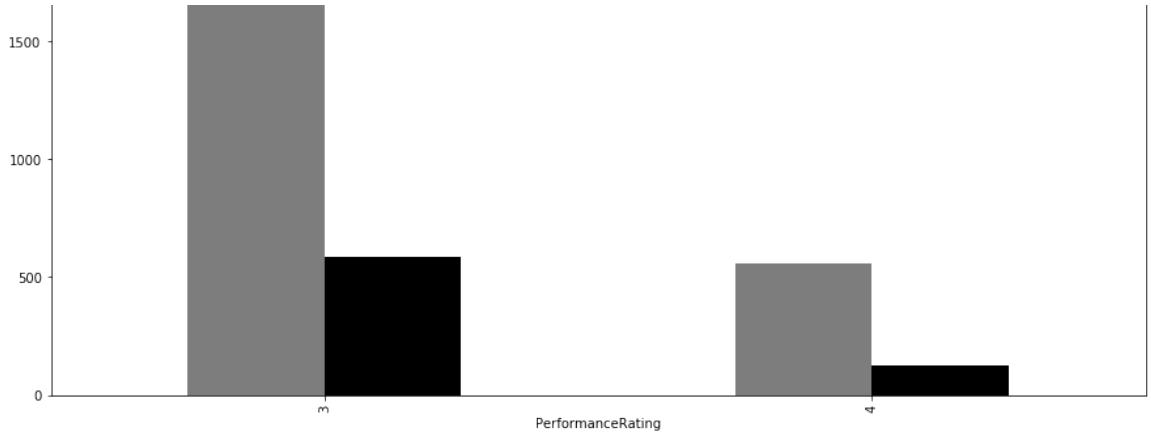

JobSatisfaction vs target
Attrition
No
Yes
1200
1000
800
600
400
200
0
1.0
2.0
3.0
4.0
JobSatisfaction


WorkLifeBalance vs target
Attrition
No
Yes
2000


1500
1000
500
0
1.0
2.0
3.0
4.0
WorkLifeBalance


JobInvolvement vs target
Attrition
No
Yes
2000
1500
1000
500
0
1
2
3
4
JobInvolvement


PerformanceRating vs target
Attrition
No
Yes
3000
2500
2000



In [27]:

```
# CHI-SQ Test

from scipy.stats import chi2_contingency

def FunctionChiSq(inpData, TargetVariable, PredictorList):

    FinalPredictor=[]

    for predictor in PredictorList:
        CrossTabResult= pd.crosstab(index= EA[predictor], columns= EA['Attr
ition'])
        ChiSqResult = chi2_contingency(CrossTabResult)

        if(ChiSqResult[1] < 0.05):
            print(predictor, 'is correlated with ', TargetVariable, '|| P-V
alue: ', ChiSqResult[1])
            FinalPredictor.append(predictor)

        else:
            print(predictor, 'is NOT correlated with ', TargetVariable, '||
P-Value: ', ChiSqResult[1])

    return(FinalPredictor)
```

In [28]:
```
# call the function

FunctionChiSq(inpData = EA, TargetVariable = 'Attrition', PredictorList = C
ategorical)
```

```
AgeGroup is correlated with  Attrition || P-Value:  7.149932240410399e-25
MonthlyIncomeGroup is NOT correlated with  Attrition || P-Value:  0.069157
62089842155
WorkingYearsGroup is correlated with  Attrition || P-Value:  1.18220216893
6935e-40
YearsatCompany is correlated with  Attrition || P-Value:  5.81518685865511
e-19
BusinessTravel is correlated with  Attrition || P-Value:  1.76427697298318
9e-16
Department is correlated with  Attrition || P-Value:  4.820888218170407e-0
7
Education is NOT correlated with  Attrition || P-Value:  0.227598263017823
93
EducationField is correlated with  Attrition || P-Value:  8.28891746957417
9e-09
Gender is NOT correlated with  Attrition || P-Value:  0.24529482862926827
JobLevel is NOT correlated with  Attrition || P-Value:  0.1799276801337186
JobRole is correlated with  Attrition || P-Value:  0.0014855447448152669
MaritalStatus is correlated with  Attrition || P-Value:  8.45385940605786e
-31
NumCompaniesWorked is correlated with  Attrition || P-Value:  1.3273960872
57985e-12
PercentSalaryHike is NOT correlated with  Attrition || P-Value:  0.1598806
6739973145
StockOptionLevel is NOT correlated with  Attrition || P-Value:  0.38454683
657380506
TrainingTimesLastYear is correlated with  Attrition || P-Value:  0.0009070
057637258118
YearsSinceLastPromotion is correlated with  Attrition || P-Value:  2.75300
23754864433e-08
YearsWithCurrManager is correlated with  Attrition || P-Value:  4.37385677
4654096e-38
EnvironmentSatisfaction is correlated with  Attrition || P-Value:  5.78051
169604359e-14
JobSatisfaction is correlated with  Attrition || P-Value:  1.6836928333881
086e-11
WorkLifeBalance is correlated with  Attrition || P-Value:  9.7688827729507
01e-11
JobInvolvement is correlated with  Attrition || P-Value:  0.04322252837372
166
PerformanceRating is NOT correlated with  Attrition || P-Value:  0.1342932
423449371
```

Out[28]:

```
['AgeGroup',
 'WorkingYearsGroup',
 'YearsatCompany',
 'BusinessTravel',
 'Department',
 'EducationField',
 'JobRole',
 'MaritalStatus',
 'NumCompaniesWorked',
 'TrainingTimesLastYear',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager',
 'EnvironmentSatisfaction',
 'JobSatisfaction',
 'WorkLifeBalance',
 'JobInvolvement']
```

Final Predictor for Machine Learning Purpose

Move towards machine learning process. Select only those columns which are fitted for ML analysis. Here in this step, we only include predictors, not target variable.

In [29]:
```
SelectedColumns= ['Age', 'MonthlyIncome', 'TotalWorkingYears', 'YearsAtCompany', 'BusinessTravel', 'Department',
 'EducationField', 'JobRole', 'MaritalStatus', 'NumCompaniesWorked', 'TrainingTimesLastYear', 'YearsSinceLastPromotion',
 'YearsWithCurrManager', 'EnvironmentSatisfaction', 'JobSatisfaction', 'WorkLifeBalance', 'JobInvolvement']

DataforML = EA[SelectedColumns]

DataforML.head()
```

Out[29]:

	Age	MonthlyIncome	TotalWorkingYears	YearsAtCompany	BusinessTravel	Department
0	51	131160	1.0	1	Travel_Rarely	Sales
1	31	41890	6.0	5	Travel_Frequently	Research & Development
2	32	193280	5.0	5	Travel_Frequently	Research & Development
3	38	83210	13.0	8	Non-Travel	Research & Development
4	32	23420	9.0	6	Travel_Rarely	Research & Development

Data Pre-Processing for Machine Learning

Machine learning process doesnot recognise any string value, so all columns convert to numeric ones.

In [30]:
```
# converting all nominal categorical columns to numeric using pd.get_dummies()
DataforML_numeric = pd.get_dummies(DataforML)

# binary nominal category column into numeric using 1/0
EA['Attrition'].replace({'Yes':1, 'No':0}, inplace = True)

# add Target Varible along with other predictors
DataforML_numeric['Attrition'] = EA['Attrition']

DataforML_numeric.head()
```

Out[30]:

	Age	MonthlyIncome	TotalWorkingYears	YearsAtCompany	NumCompaniesWorked	Training
0	51	131160	1.0	1	1.0	
1	31	41890	6.0	5	0.0	
2	32	193280	5.0	5	1.0	
3	38	83210	13.0	8	3.0	
4	32	23420	9.0	6	4.0	

In [31]:
```
# no of rows and columns under new dataset
DataforML_numeric.shape
```

Out[31]: (4410, 37)

Splitting the data into Training and Testing

Split the whole dataset into 2 parts with ratio 70:30. The purpose is to use machine learning on the training part and make a predictive model and and pass the testing data into this and ask model to generate prediction, and compare with target values of test data.

In [32]:
```
DataforML_numeric.columns
```

Out[32]:
```
Index(['Age', 'MonthlyIncome', 'TotalWorkingYears', 'YearsAtCompany',
       'NumCompaniesWorked', 'TrainingTimesLastYear',
       'YearsSinceLastPromotion', 'YearsWithCurrManager',
       'EnvironmentSatisfaction', 'JobSatisfaction', 'WorkLifeBalance',
       'JobInvolvement', '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', 'JobRole_Healthcare Representati
ve',
       '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', 'Attrition'],
      dtype='object')
```

In [33]:
```
TargetVariable= 'Attrition'
Predictors= ['Age', 'MonthlyIncome', 'TotalWorkingYears', 'YearsAtCompany',
       'NumCompaniesWorked', 'TrainingTimesLastYear',
       'YearsSinceLastPromotion', 'YearsWithCurrManager',
       'EnvironmentSatisfaction', 'JobSatisfaction', 'WorkLifeBalance',
       'JobInvolvement', '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', 'JobRole_Healthcare Representativ
e',
       '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']

# extract only values of predictor and target variable
X= DataforML_numeric[Predictors].values
Y= DataforML_numeric[TargetVariable].values

# Split the data
from sklearn.model_selection import train_test_split
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y, train_size= 0.7,
random_state= 1234)

print(X_Train.shape)
print(X_Test.shape)
print(Y_Train.shape)
print(Y_Test.shape)
```

```
(3087, 36)
(1323, 36)
(3087,)
(1323,)
```

Apply StandardScaler for Standardization

In the dataset, Age and MonthlyIncome column values are relatively higher compared to other columns, which may lead to some undue advantages on those two columns in ML process. So, to minimise this advantages, all columns are need to make a same scale.

In [34]:
```
from sklearn.preprocessing import StandardScaler

# create an object of standardscaler
PredictorScaler=StandardScaler()

# fitting the data into x i.e. predictor variable
PredictorScalerFit= PredictorScaler.fit(X)

# transform x i.e. calculation of standardization
X= PredictorScalerFit.transform(X)

# splitting the dataset into testing and training again after standardizati
on
from sklearn.model_selection import train_test_split

X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y, train_size= 0.7,
random_state= 1234)
```

In [35]:
```
X_Train[0:5]
```

Out[35]:
```
array([[ 0.88435781, -0.466801  ,  1.89393073,  2.77443066, -0.27548694,
         0.15570708, -0.36871529,  1.92795644, -1.58311992, -0.66610662,
         0.33620609,  0.37967213, -0.33709993, -0.48185865,  0.63984058,
         4.72581563, -1.37405084, -0.65995975,  7.31057073, -0.83748964,
        -0.34825488, -0.67914108, -0.24305927, -0.31409347, -0.3127846 ,
        -0.1914977 ,  2.16233107, -0.27305934, -0.33080804, -0.23990406,
        -0.49787324, -0.53382116, -0.24462499, -0.53487311, -0.91892141,
         1.45864991],
       [-0.64866811, -0.83523898, -0.16431549,  0.48850773, -0.6761787 ,
         0.15570708,  0.5625759 ,  0.80654148,  0.25220049, -0.66610662,
         0.33620609,  0.37967213, -0.33709993,  2.07529738, -1.5628893 ,
        -0.21160368, -1.37405084,  1.51524392, -0.13678823, -0.83748964,
        -0.34825488,  1.47244811, -0.24305927, -0.31409347, -0.3127846 ,
        -0.1914977 , -0.46246387, -0.27305934, -0.33080804, -0.23990406,
        -0.49787324,  1.87328654, -0.24462499, -0.53487311, -0.91892141,
         1.45864991],
       [-0.2106607 , -0.07669019, -0.16431549, -0.81773395, -0.27548694,
        -0.62018922, -0.05828489, -0.59522723,  0.25220049,  1.14886415,
         0.33620609,  0.37967213, -0.33709993,  2.07529738, -1.5628893 ,
        -0.21160368,  0.72777511, -0.65995975, -0.13678823, -0.83748964,
        -0.34825488,  1.47244811, -0.24305927, -0.31409347, -0.3127846 ,
        -0.1914977 , -0.46246387, -0.27305934, -0.33080804,  4.168333  ,
        -0.49787324, -0.53382116, -0.24462499, -0.53487311, -0.91892141,
         1.45864991],
       [ 0.77485596, -0.27089568, -0.67887704, -0.32789332, -0.6761787 ,
        -0.62018922, -0.36871529, -0.31487349,  1.1698607 , -0.66610662,
        -1.08533644,  1.78551099, -0.33709993, -0.48185865,  0.63984058,
        -0.21160368,  0.72777511, -0.65995975, -0.13678823, -0.83748964,
        -0.34825488, -0.67914108,  4.1142228 , -0.31409347, -0.3127846 ,
        -0.1914977 , -0.46246387, -0.27305934, -0.33080804, -0.23990406,
         2.0085434 , -0.53382116, -0.24462499, -0.53487311, -0.91892141,
         1.45864991],
       [ 1.76037262, -0.63444666,  0.22160568, -0.0013329 , -0.27548694,
        -1.39608553,  0.5625759 ,  0.24583399, -0.66545972,  0.24137877,
        -2.50687897, -1.02616674, -0.33709993, -0.48185865,  0.63984058,
        -0.21160368, -1.37405084,  1.51524392, -0.13678823,  1.19404463,
        -0.34825488, -0.67914108, -0.24305927, -0.31409347, -0.3127846 ,
        -0.1914977 , -0.46246387, -0.27305934,  3.0229011 , -0.23990406,
        -0.49787324, -0.53382116, -0.24462499, -0.53487311,  1.08823234,
        -0.68556546]])
```

In [36]:
```
X_Test[0:5]
```

Out[36]:
```
array([[ 1.97937632, -0.46403878,  0.09296529,  0.32522752,  0.52589657,
         0.15570708,  1.49386709, -0.31487349,  0.25220049,  1.14886415,
        -1.08533644,  1.78551099, -0.33709993, -0.48185865,  0.63984058,
         4.72581563, -1.37405084, -0.65995975, -0.13678823, -0.83748964,
        -0.34825488, -0.67914108, -0.24305927,  3.18376564, -0.3127846 ,
        -0.1914977 , -0.46246387, -0.27305934,  3.0229011 , -0.23990406,
        -0.49787324, -0.53382116, -0.24462499, -0.53487311, -0.91892141,
         1.45864991],
       [-1.41518107, -0.76852068, -1.32207899, -0.98101416, -0.6761787 ,
         1.70749969, -0.67914568, -1.15593471,  1.1698607 , -0.66610662,
        -2.50687897,  0.37967213, -0.33709993, -0.48185865,  0.63984058,
         4.72581563, -1.37405084, -0.65995975,  7.31057073, -0.83748964,
        -0.34825488, -0.67914108, -0.24305927, -0.31409347, -0.3127846 ,
        -0.1914977 , -0.46246387, -0.27305934,  3.0229011 , -0.23990406,
        -0.49787324, -0.53382116, -0.24462499, -0.53487311,  1.08823234,
        -0.68556546],
       [ 1.76037262, -0.70987658,  1.89393073,  1.14162857,  0.52589657,
         0.93160339,  0.5625759 ,  1.08689522,  0.25220049,  1.14886415,
        -1.08533644,  0.37967213, -0.33709993, -0.48185865,  0.63984058,
        -0.21160368, -1.37405084,  1.51524392, -0.13678823,  1.19404463,
        -0.34825488, -0.67914108, -0.24305927, -0.31409347, -0.3127846 ,
         5.2219949 , -0.46246387, -0.27305934, -0.33080804, -0.23990406,
        -0.49787324, -0.53382116, -0.24462499,  1.86960232, -0.91892141,
        -0.68556546],
       [-0.2106607 , -0.71455111,  0.73616723,  1.63146919, -0.6761787 ,
        -0.62018922,  0.8730063 ,  3.0493714 ,  1.1698607 ,  0.24137877,
         0.33620609, -1.02616674, -0.33709993, -0.48185865,  0.63984058,
        -0.21160368,  0.72777511, -0.65995975, -0.13678823,  1.19404463,
        -0.34825488, -0.67914108, -0.24305927, -0.31409347,  3.19708837,
        -0.1914977 , -0.46246387, -0.27305934, -0.33080804, -0.23990406,
        -0.49787324, -0.53382116, -0.24462499, -0.53487311,  1.08823234,
        -0.68556546],
       [-0.10115885, -0.37947227,  0.09296529,  0.65178794, -0.6761787 ,
         0.15570708, -0.67914568,  0.80654148,  0.25220049, -0.66610662,
        -1.08533644,  0.37967213, -0.33709993,  2.07529738, -1.5628893 ,
        -0.21160368, -1.37405084,  1.51524392, -0.13678823, -0.83748964,
         2.87146008, -0.67914108, -0.24305927, -0.31409347, -0.3127846 ,
        -0.1914977 ,  2.16233107, -0.27305934, -0.33080804, -0.23990406,
        -0.49787324, -0.53382116, -0.24462499, -0.53487311,  1.08823234,
        -0.68556546]])
```

In [37]:
```
Y_Train[0:5]
```

Out[37]:
```
array([1, 1, 0, 0, 0], dtype=int64)
```

In [38]:
```
Y_Test[0:5]
```

Out[38]:
```
array([1, 1, 0, 0, 0], dtype=int64)
```

Logistic Regression

In [39]:

```
from sklearn.linear_model import LogisticRegression

# choose penalty between 'l1' or 'l2'
# choose value of C (no of rows): 1% of total rows
# choose solver: 'newton-cg', 'lbfgs', 'liblinear', 'saga', 'sag'

LogModel=LogisticRegression(C=1, penalty='l2', solver='newton-cg')

print(LogModel)

# creating a model to fit training data
LOG= LogModel.fit(X_Train, Y_Train)

# generating predictions on testing data
prediction= LOG.predict(X_Test)

# creating a dataframe to see prediction in test data
TestDataResult=pd.DataFrame(Y_Test, columns=['Attrition'])
TestDataResult['PredictedAttrition']= prediction
print(TestDataResult.head())

print('************************************************')

# measuring accuracy on testing data
from sklearn import metrics
print(metrics.classification_report(Y_Test, prediction))
print(metrics.confusion_matrix(Y_Test, prediction))

# printing over all accuracy of test data
F1_score= metrics.f1_score(Y_Test, prediction, average='weighted')

print('Accuracy of the model on TestingSample: ', round(F1_score,2))

# import cross validation , k-fold validation
from sklearn.model_selection import cross_val_score
AccuracyScore= cross_val_score(LogModel, X, Y, cv=10, scoring='f1_weighte
d')

print('Accuracy value for 10 fold cross validation: ', AccuracyScore)
print('Final Average accuracy of the model: ', round(AccuracyScore.mean(),
2))
```

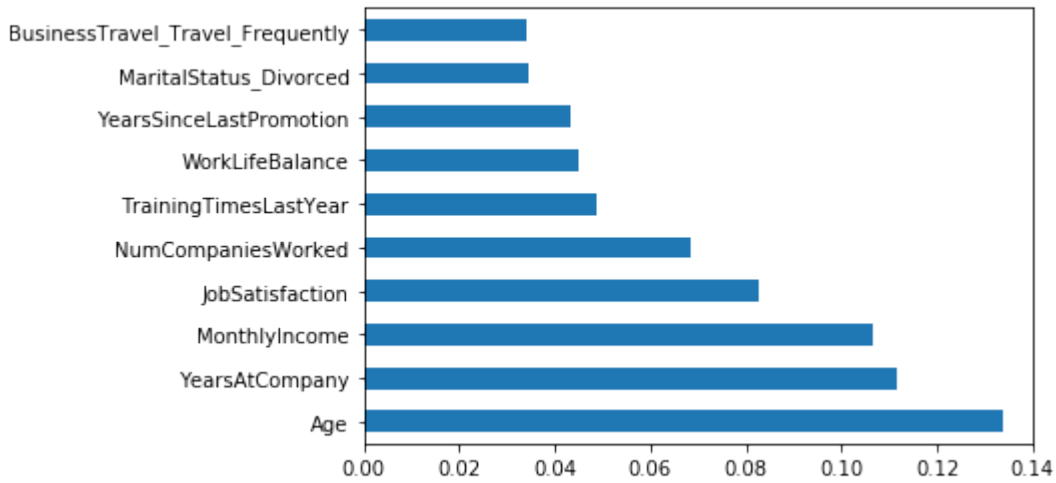
```
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='newton-cg', tol=0.0001, verb
ose=0,
                   warm_start=False)
   Attrition  PredictedAttrition
0          1                   0
1          1                   0
2          0                   0
3          0                   0
4          0                   0
**************************************************
              precision    recall  f1-score   support

           0       0.85      0.98      0.91      1097
           1       0.62      0.14      0.22       226

    accuracy                           0.84      1323
   macro avg       0.73      0.56      0.57      1323
weighted avg       0.81      0.84      0.79      1323

[[1078   19]
 [ 195   31]]
Accuracy of the model on TestingSample:  0.79
Accuracy value for 10 fold cross validation:  [0.8291074  0.80217881 0.813
64671 0.81938066 0.79667139 0.82341098
 0.80557741 0.8251146  0.82171684 0.79180816]
Final Average accuracy of the model:  0.81
```

Decision Tree

In [40]:

```
from sklearn.tree import DecisionTreeClassifier

# choose between 'entropy' and 'gini'
LogModel= DecisionTreeClassifier(max_depth=10, criterion='gini')

print(LogModel)

# creating a model on training data
DT= LogModel.fit(X_Train, Y_Train)

prediction= DT.predict(X_Test)

# printing some rows to see the prediction in test data
TestDataResult=pd.DataFrame(Y_Test, columns=['Attrition'])
TestDataResult['PredictedAttrition']=prediction
print(TestDataResult.head())

# measuring accuracy on testing data
from sklearn import metrics

print(metrics.classification_report(Y_Test, prediction))
print(metrics.confusion_matrix(Y_Test, prediction))

# printing overall accuracy of model
F1_Score= metrics.f1_score(Y_Test, prediction, average='weighted')
print('Accuracy on model test data: ', round(F1_Score,2))

# plotting feature importances for top 10 most imp columns
%matplotlib inline
feature_importances= pd.Series(DT.feature_importances_, index=Predictors)
feature_importances.nlargest(10).plot(kind='barh')

# import cross validation, running k fold validation
from sklearn.model_selection import cross_val_score

# passing a full x and y , as k-fold split data and automatically choose tr
ain and test
AccuracyScore= cross_val_score(LogModel, X, Y, cv=10, scoring='f1_weighte
d')

print('Accuracy for 10 fold cross validation: ', AccuracyScore)
print('Final Average Accuracy of the model: ', round(AccuracyScore.mean(),
2))
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=10, max_features=None, max_leaf_nodes=Non
e,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
   Attrition  PredictedAttrition
0          1                   1
1          1                   1
2          0                   0
3          0                   0
4          0                   0
              precision    recall  f1-score   support

           0       0.94      0.98      0.96      1097
           1       0.87      0.67      0.76       226

    accuracy                           0.93      1323
   macro avg       0.90      0.83      0.86      1323
weighted avg       0.92      0.93      0.92      1323

[[1075   22]
 [  74  152]]
Accuracy on model test data:  0.92
Accuracy for 10 fold cross validation:  [0.9369863  0.93323483 0.94350067
0.93104837 0.93917351 0.92828926
 0.9326659  0.91913302 0.92612326 0.94145903]
Final Average Accuracy of the model:  0.93
```



Thank You

In [ ]: