Attrition Analytics - Exploratory Analysis & Predictive Modeling

Import the libraries

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

%matplotlib inline

#using the seaborn style for graphs
plt.style.use("seaborn")

## Read the dataset
employee_data = pd.read_excel("Attrition.xlsx")

employee_data.head()
```

```
   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  \
0          ...                                  1            80
1          ...                                  4            80
2          ...                                  2            80
```

```
3         ...                                    3           80
4         ...                                    4           80

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

   YearsAtCompany YearsInCurrentRole  YearsSinceLastPromotion  \
0               6                  4                        0
1              10                  7                        1
2               0                  0                        0
3               8                  7                        3
4               2                  2                        2

   YearsWithCurrManager
0                     5
1                     7
2                     0
3                     0
4                     2

[5 rows x 35 columns]
```

```python
##looking for any missing values

employee_data.isnull().sum()
```

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
```

```
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

```
employee_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
Age                         1470 non-null int64
Attrition                   1470 non-null object
BusinessTravel              1470 non-null object
DailyRate                   1470 non-null int64
Department                  1470 non-null object
DistanceFromHome            1470 non-null int64
Education                   1470 non-null int64
EducationField              1470 non-null object
EmployeeCount               1470 non-null int64
EmployeeNumber              1470 non-null int64
EnvironmentSatisfaction     1470 non-null int64
Gender                      1470 non-null object
HourlyRate                  1470 non-null int64
JobInvolvement              1470 non-null int64
JobLevel                    1470 non-null int64
JobRole                     1470 non-null object
JobSatisfaction             1470 non-null int64
MaritalStatus               1470 non-null object
MonthlyIncome               1470 non-null int64
MonthlyRate                 1470 non-null int64
NumCompaniesWorked          1470 non-null int64
```

```
Over18                          1470 non-null object
OverTime                        1470 non-null object
PercentSalaryHike               1470 non-null int64
PerformanceRating               1470 non-null int64
RelationshipSatisfaction        1470 non-null int64
StandardHours                   1470 non-null int64
StockOptionLevel                1470 non-null int64
TotalWorkingYears               1470 non-null int64
TrainingTimesLastYear           1470 non-null int64
WorkLifeBalance                 1470 non-null int64
YearsAtCompany                  1470 non-null int64
YearsInCurrentRole              1470 non-null int64
YearsSinceLastPromotion         1470 non-null int64
YearsWithCurrManager            1470 non-null int64
dtypes: int64(26), object(9)
memory usage: 402.0+ KB
```

Exploratory Data Analysis

```python
## basic descriptive statistics
employee_data.describe()
```

```
               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  \
count  1470.000000          ...                        1470.000000
mean      2.063946          ...                           2.712245
std       1.106940          ...                           1.081209
min       1.000000          ...                           1.000000
25%       1.000000          ...                           2.000000
50%       2.000000          ...                           3.000000
75%       3.000000          ...                           4.000000
max       5.000000          ...                           4.000000

       StandardHours  StockOptionLevel  TotalWorkingYears  \
count         1470.0       1470.000000        1470.000000
mean            80.0          0.793878          11.279592
std              0.0          0.852077           7.780782
min             80.0          0.000000           0.000000
25%             80.0          0.000000           6.000000
50%             80.0          1.000000          10.000000
75%             80.0          1.000000          15.000000
max             80.0          3.000000          40.000000

       TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany  \
count            1470.000000      1470.000000     1470.000000
mean                2.799320         2.761224        7.008163
std                 1.289271         0.706476        6.126525
min                 0.000000         1.000000        0.000000
25%                 2.000000         2.000000        3.000000
50%                 3.000000         3.000000        5.000000
75%                 3.000000         3.000000        9.000000
max                 6.000000         4.000000       40.000000

       YearsInCurrentRole  YearsSinceLastPromotion
YearsWithCurrManager
count         1470.000000              1470.000000
1470.000000
mean             4.229252                 2.187755
4.123129
std              3.623137                 3.222430
3.568136
```

```
min                0.000000                  0.000000
0.000000
25%                2.000000                  0.000000
2.000000
50%                3.000000                  1.000000
3.000000
75%                7.000000                  3.000000
7.000000
max               18.000000                 15.000000
17.000000

[8 rows x 26 columns]
```

```
#Mapping the attrition 1 - yes and 0 - no in the new column

employee_data["left"] = np.where(employee_data["Attrition"] ==
"Yes",1,0)

employee_data.head()
```

```
   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

   ...   StandardHours StockOptionLevel  TotalWorkingYears  \
0  ...              80                0                  8
1  ...              80                1                 10
2  ...              80                0                  7
3  ...              80                0                  8
```

```
4  ...                 80                  1                   6

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

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

[5 rows x 36 columns]

#supressing all the warnings
import warnings
warnings.filterwarnings('ignore')
```

- Remove not usefull features

```
def NumericalVariables_targetPlots(df,segment_by,target_var =
"Attrition"):
    """A function for plotting the distribution of numerical variables
and its effect on attrition"""

    fig, ax = plt.subplots(ncols= 2, figsize = (14,6))

    #boxplot for comparison
    sns.boxplot(x = target_var, y = segment_by, data=df, ax=ax[0])
    ax[0].set_title("Comparision of " + segment_by + " vs " +
target_var)

    #distribution plot
    ax[1].set_title("Distribution of "+segment_by)
    ax[1].set_ylabel("Frequency")
    sns.distplot(a = df[segment_by], ax=ax[1], kde=False)

    plt.show()
```

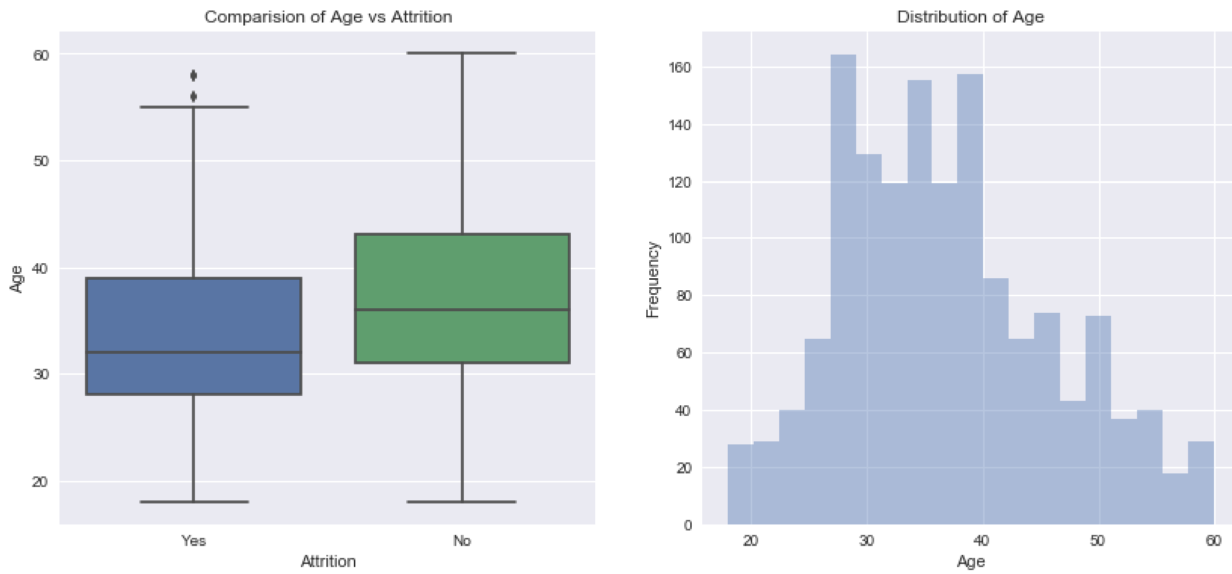
```
def CategoricalVariables_targetPlots(df, segment_by,invert_axis = False, target_var = "left"):

    """A function for Plotting the effect of variables(categorical data) on attrition """

    fig, ax = plt.subplots(ncols= 2, figsize = (14,6))

    #countplot for distribution along with target variable
    #invert axis variable helps to inter change the axis so that names of categories doesn't overlap
    if invert_axis == False:
        sns.countplot(x = segment_by, data=df,hue="Attrition",ax=ax[0])
    else:
        sns.countplot(y = segment_by, data=df,hue="Attrition",ax=ax[0])

    ax[0].set_title("Comparision of " + segment_by + " vs " + "Attrition")

    #plot the effect of variable on attrition
    if invert_axis == False:
        sns.barplot(x = segment_by, y = target_var ,data=df,ci=None)
    else:
        sns.barplot(y = segment_by, x = target_var ,data=df,ci=None)

    ax[1].set_title("Attrition rate by {}".format(segment_by))
    ax[1].set_ylabel("Average(Attrition)")
    plt.tight_layout()

    plt.show()
```

Analyizing the variables

- Numerical Variables

Age

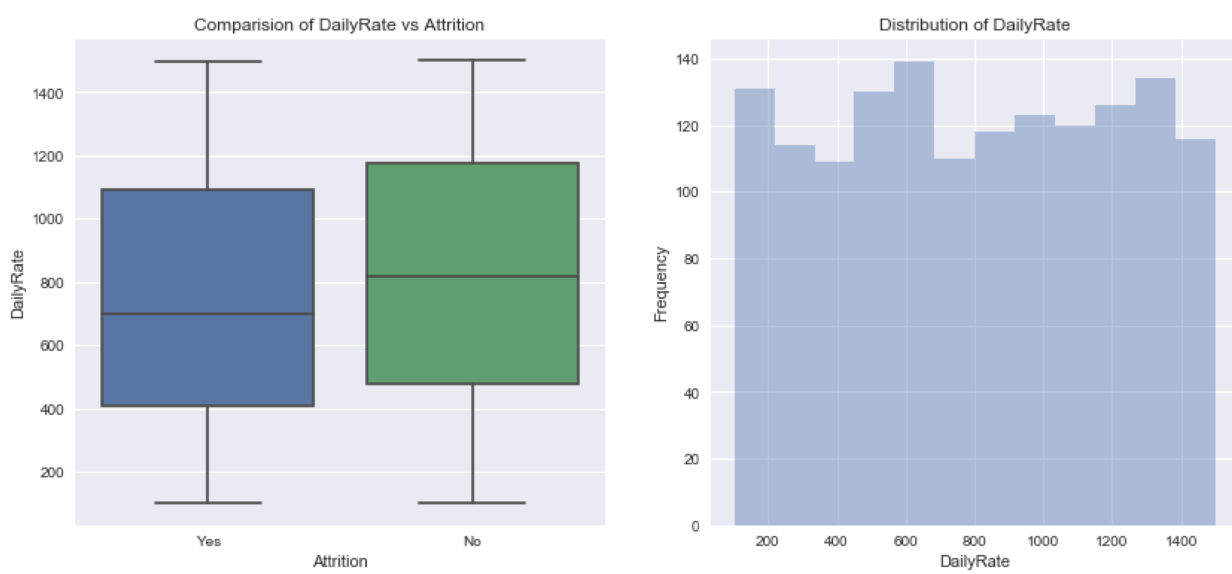
```
# we are checking the distribution of employee age and its related to attrition or not

NumericalVariables_targetPlots(employee_data,segment_by="Age")
```



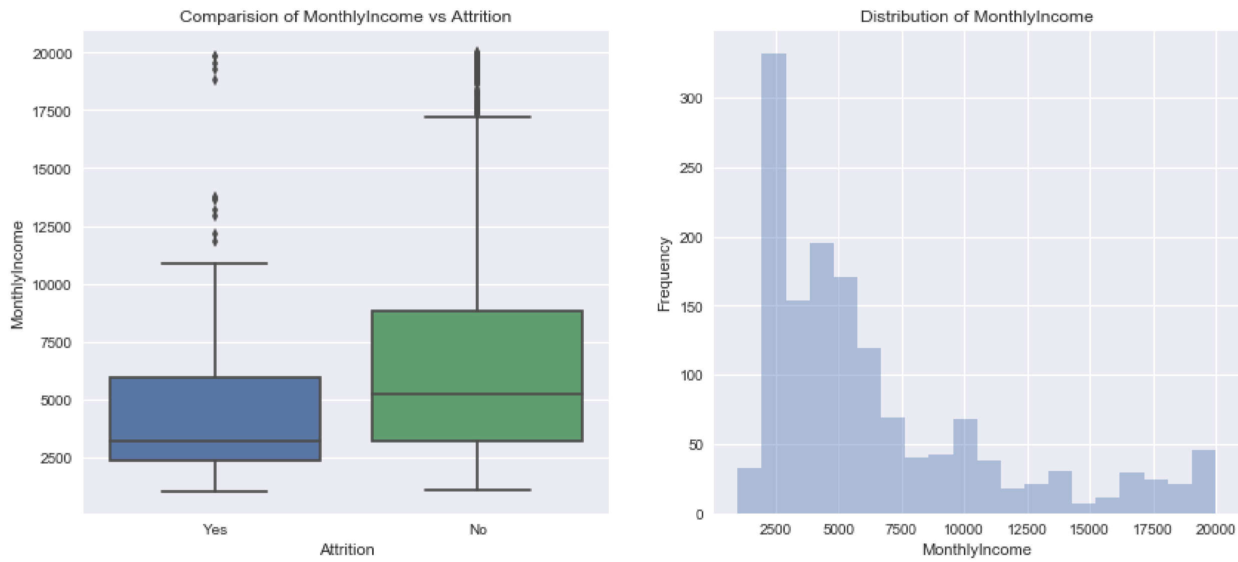
- We found that median age of employee's in the company is 30 - 40 Yrs. Minimum age is 18 Yrs and Maximum age is 60 Yrs.
- From the Age Comparision boxplot, majority of people who left the company are below 40 Yrs and among the people who didn't left the company are of age 32 to 40 years

Daily Rate & Montly Income & HourlyRate

```
#Analyzing the daily wage rate vs employee left the company or not

NumericalVariables_targetPlots(employee_data,"DailyRate")
```



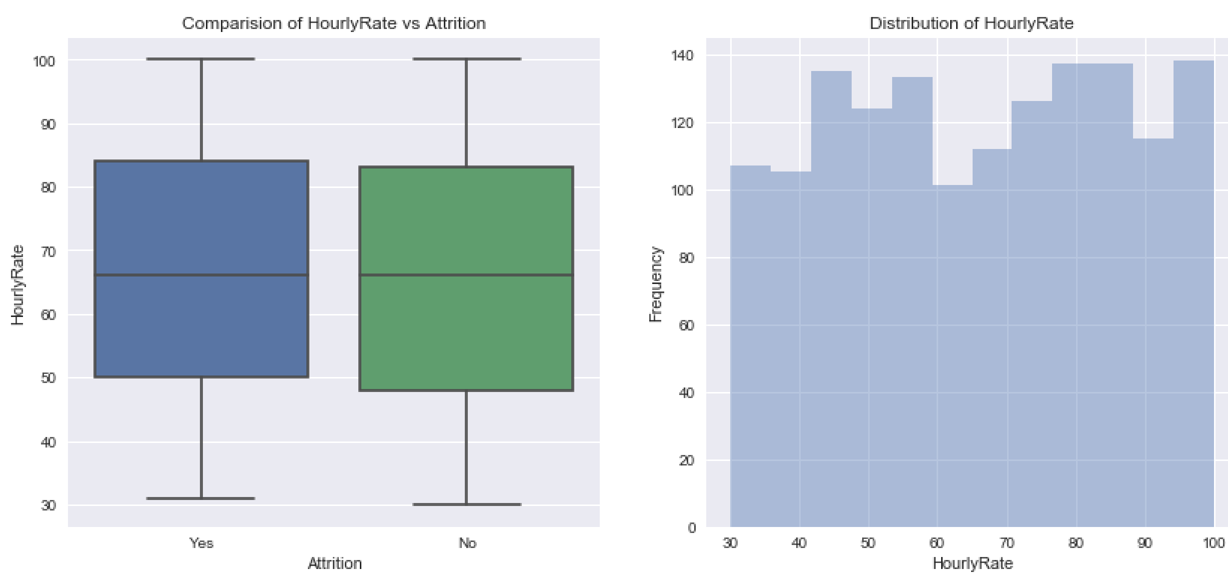

```
NumericalVariables_targetPlots(employee_data,"MonthlyIncome")
```



- Employee's working with lower daily rates are more prone to leave the company than compared to the employee's working with higher rates. The same trend is resonated with monthly income too.

Hourly Rate

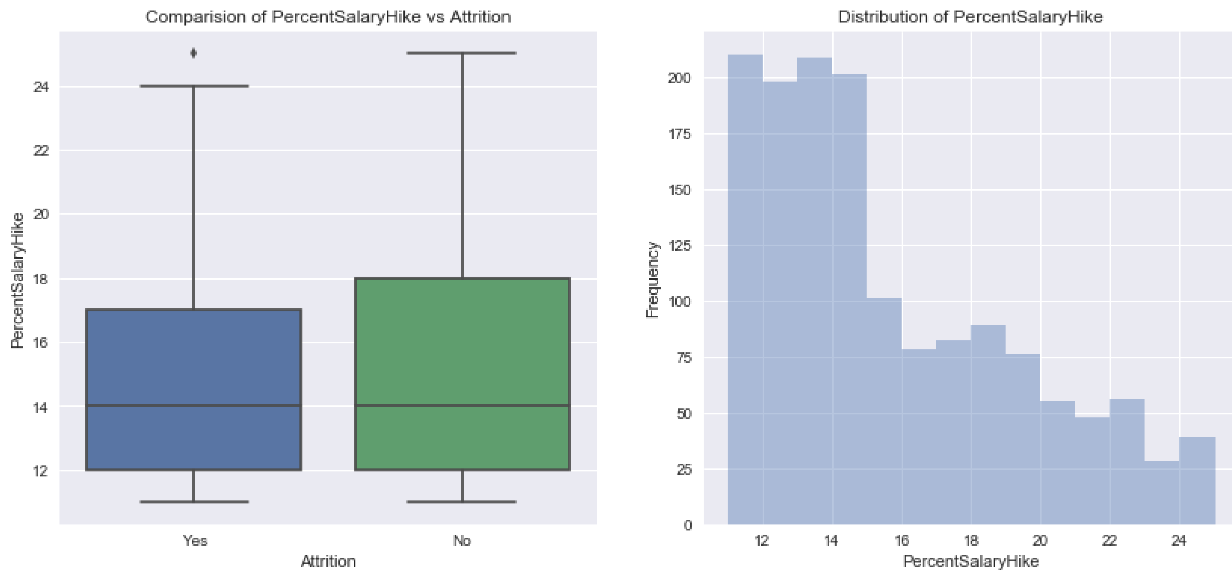
```
NumericalVariables_targetPlots(employee_data,"HourlyRate")
```



- From plot we have seen that there is no significant difference in the hourly rate and attrition. Therefore hourly rate is considered as not signifcant to attrition

PercentSalaryHike

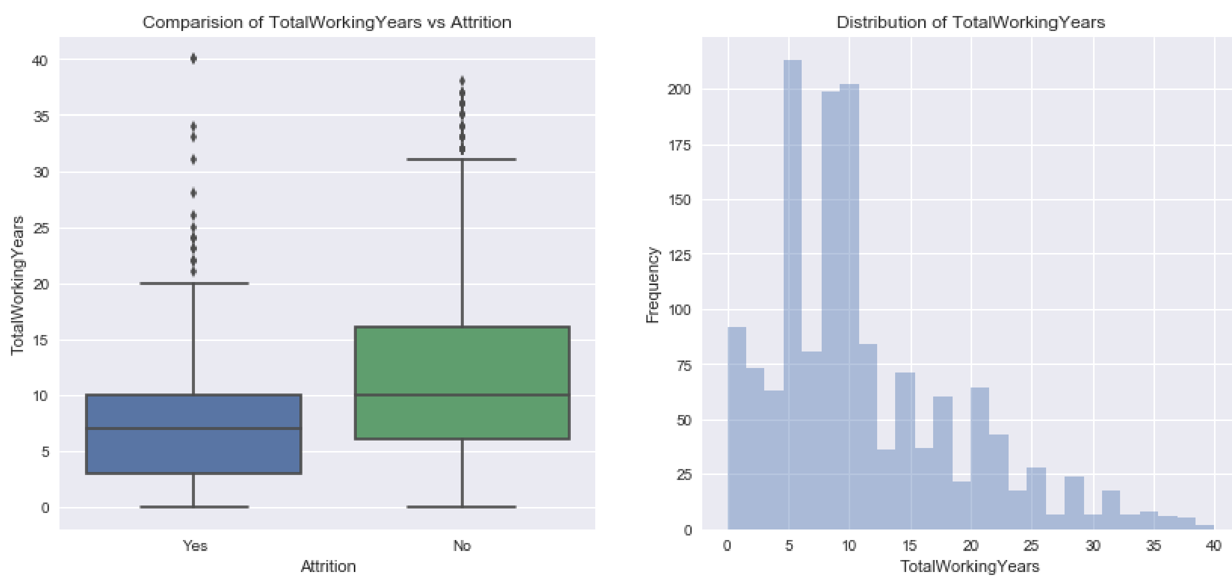
```
NumericalVariables_targetPlots(employee_data,"PercentSalaryHike")
```



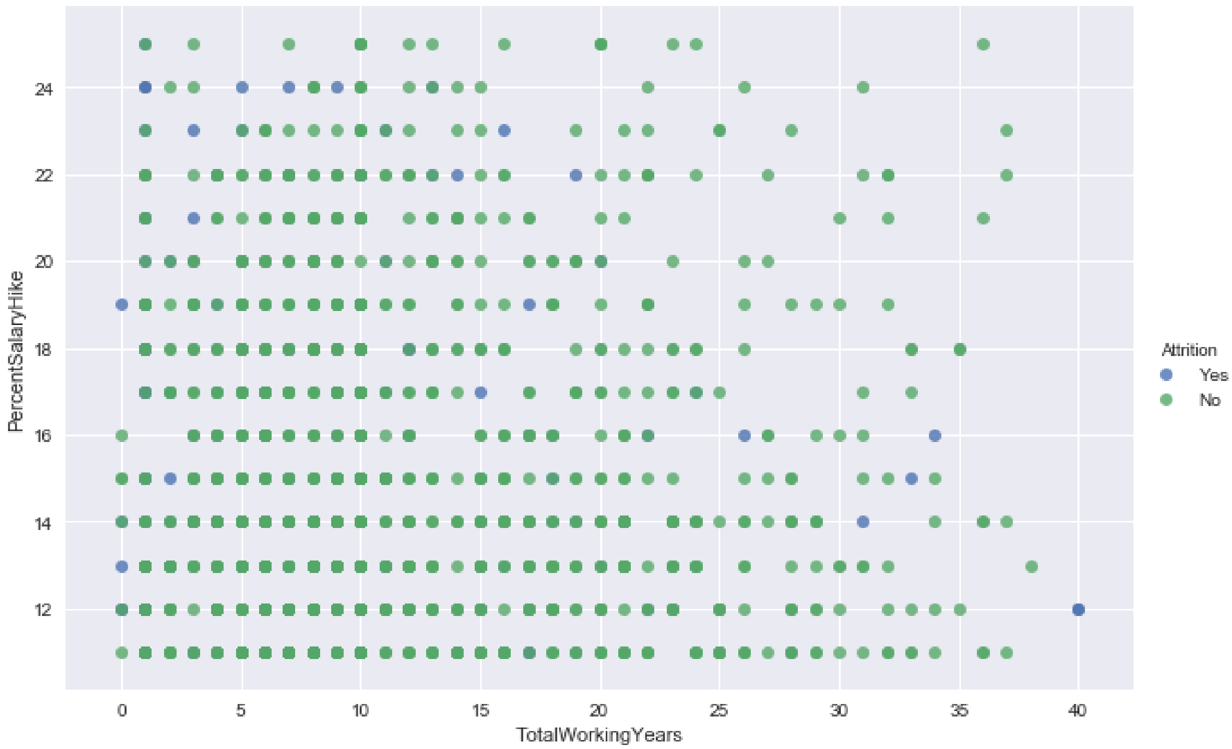
- Majority (60% of total strength) of employee's receive 16% salary hike in the company, employee's who received less salary hike have left the company.

Total Working years

```
NumericalVariables_targetPlots(employee_data,"TotalWorkingYears")
```



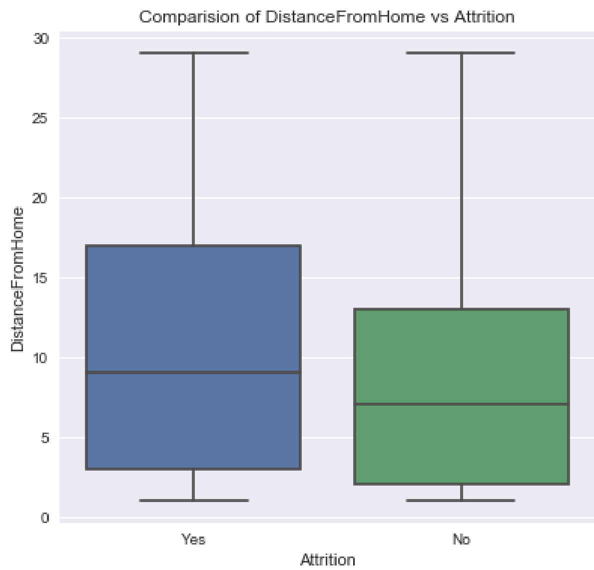

```
sns.lmplot(x = "TotalWorkingYears", y = "PercentSalaryHike",
data=employee_data,fit_reg=False,hue="Attrition",size=6,
           aspect=1.5)

plt.show()
```

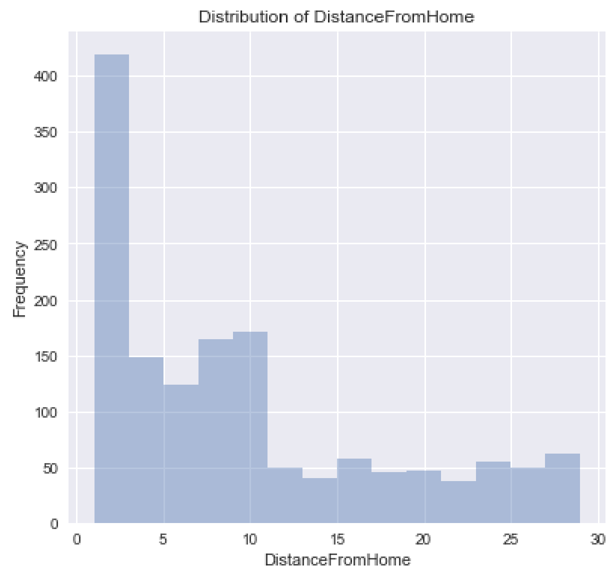


- Employee's with less working years have received 25% Salary hike when they switch to another company, but there is no linear relationship between working years and salary hike.
- Attrition is not seen amomg the employee's having more than 20 years of experience if their salary hike is more than 20%, even if the salary hike is below 20% attrition rate among the employee's is very low.
- Employee's with lesser years of experience are prone to leave the company in search of better pay, irrespective of salary hike

Distance From Home

```
NumericalVariables_targetPlots(employee_data,"DistanceFromHome")
```





- There is a higher number of people who reside near to offices and hence the attrition levels are lower for distance less than 10. With increase in distance from home, attrition rate also increases

Analyizing the variables

- Categorical Variables

Job Involvement

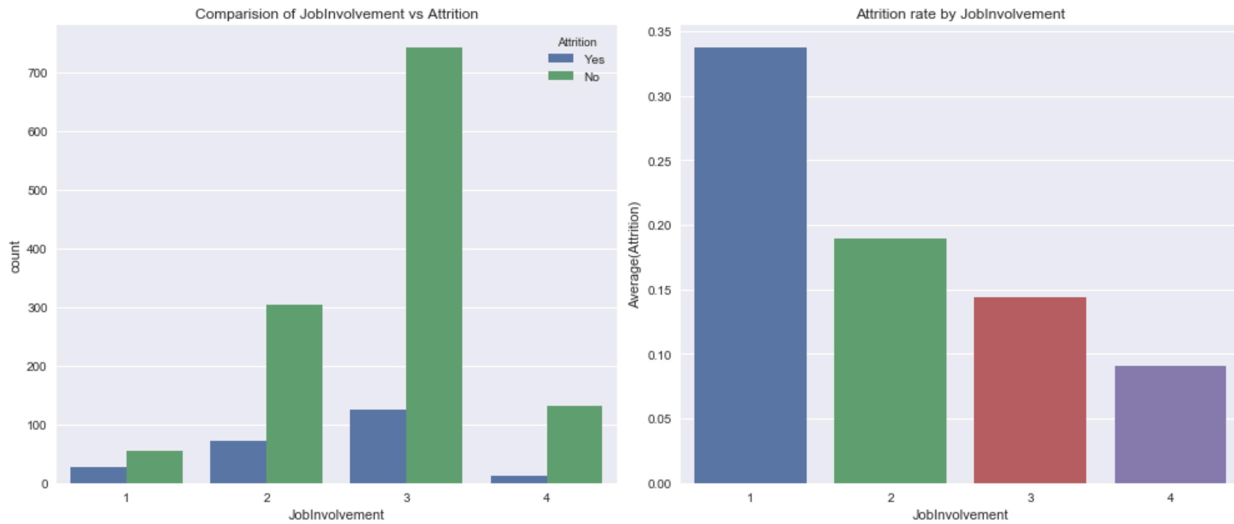
```
#cross tabulation between attrition and JobInvolvement
pd.crosstab(employee_data.JobInvolvement,employee_data.Attrition)
```

Attrition	No	Yes
JobInvolvement		
1	55	28
2	304	71
3	743	125
4	131	13

```
#calculating the percentage of people having different job involvement rate
round(employee_data.JobInvolvement.value_counts()/employee_data.shape[0] * 100,2)
```

```
3    59.05
2    25.51
4     9.80
1     5.65
Name: JobInvolvement, dtype: float64
```

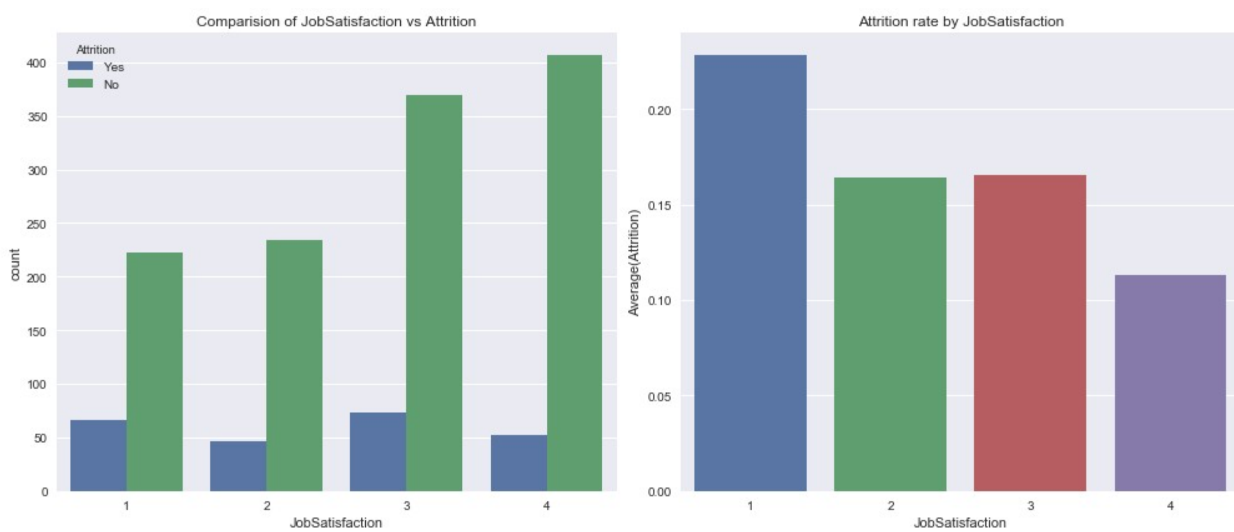
```
CategoricalVariables_targetPlots(employee_data,"JobInvolvement")
```



1. In the total data set, 59% have high job involvement whereas 25% have medium involvement rate
2. From above plot we can observe that round 50% of people in low job involvement (level 1 & 2) have left the company.
3. Even the people who have high job involmenent have higher attrition rate around 15% in that category have left company

JobSatisfaction

```
CategoricalVariables_targetPlots(employee_data,"JobSatisfaction")
```



As expected, people with low satisfaction have left the company around 23% in that category. what surprising is out of the people who rated medium and high job satisfaction around 32% has left the company. There should be some other factor which triggers their exit from the company

Performance Rating

```
#checking the number of categories under performance rating
employee_data.PerformanceRating.value_counts()
```

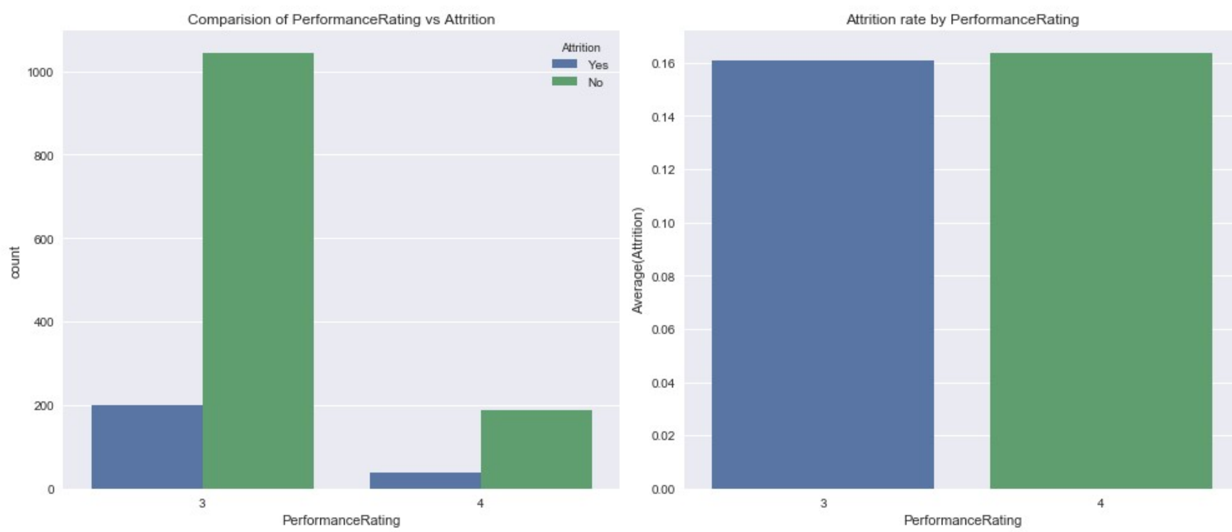
```
3    1244
4     226
Name: PerformanceRating, dtype: int64
```

```
#calculate the percentage of performance rating per category in the whole dataset
round(employee_data.PerformanceRating.value_counts()/employee_data.shape[0] * 100,2)
```

```
3    84.63
4    15.37
Name: PerformanceRating, dtype: float64
```

Around 85% of people in the company rated as Excellent and remaining 15% rated as Outstanding

```
CategoricalVariables_targetPlots(employee_data,"PerformanceRating")
```



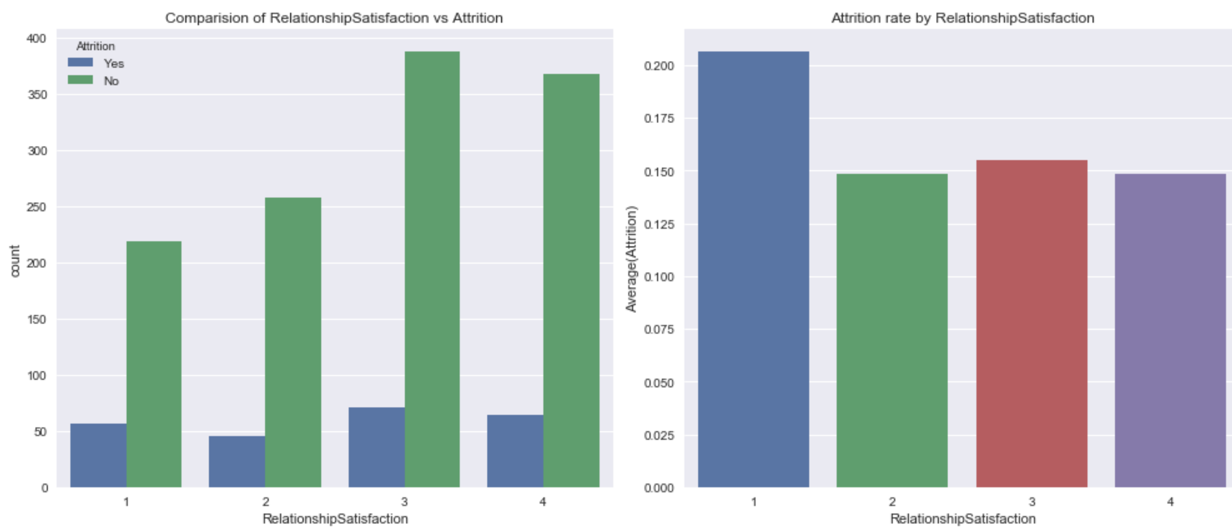
Contrary to normal belief that employee's having higher rating will not leave the company. It may be seen that there is no significant difference between the performance rating and Attrition Rate.

RelationshipSatisfaction

```
#percentage of each relationship satisfaction category across the data
round(employee_data.RelationshipSatisfaction.value_counts()/employee_data.shape[0],2)
```

```
3    0.31
4    0.29
2    0.21
1    0.19
Name: RelationshipSatisfaction, dtype: float64
```

```
CategoricalVariables_targetPlots(employee_data,"RelationshipSatisfaction")
```



In this too, we found that almost 30% of employees with high and very high RelationshipSatisfaction have left the company. Here also there is no visible trend among the relationshipsatisfaction and attrition rate

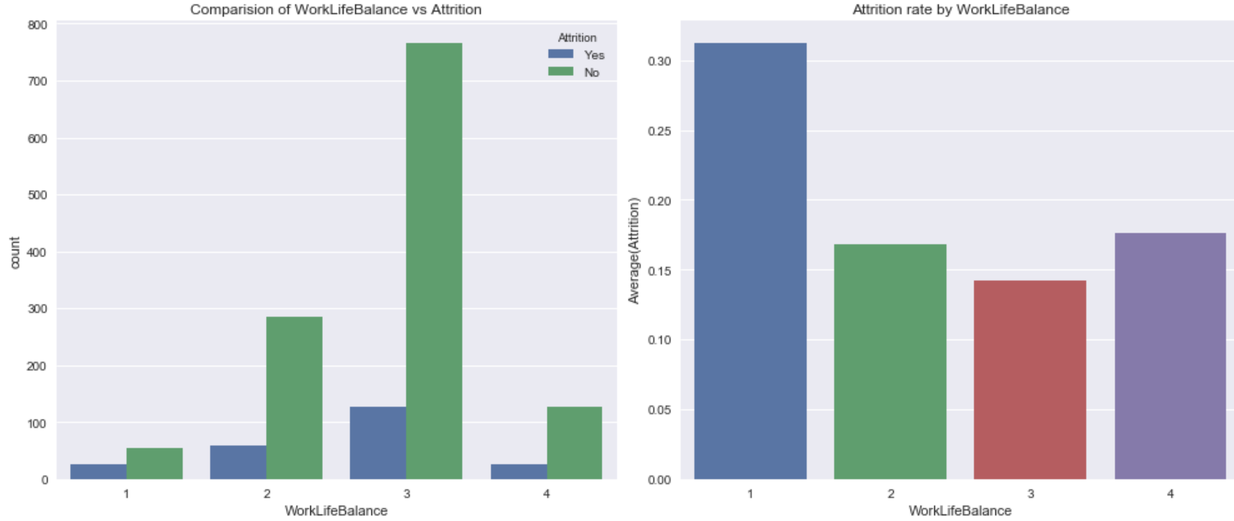
WorkLifeBalance

```
#percentage of worklife balance rating across the company data
round(employee_data.WorkLifeBalance.value_counts()/employee_data.shape[0],2)
```

```
3    0.61
2    0.23
4    0.10
1    0.05
Name: WorkLifeBalance, dtype: float64
```

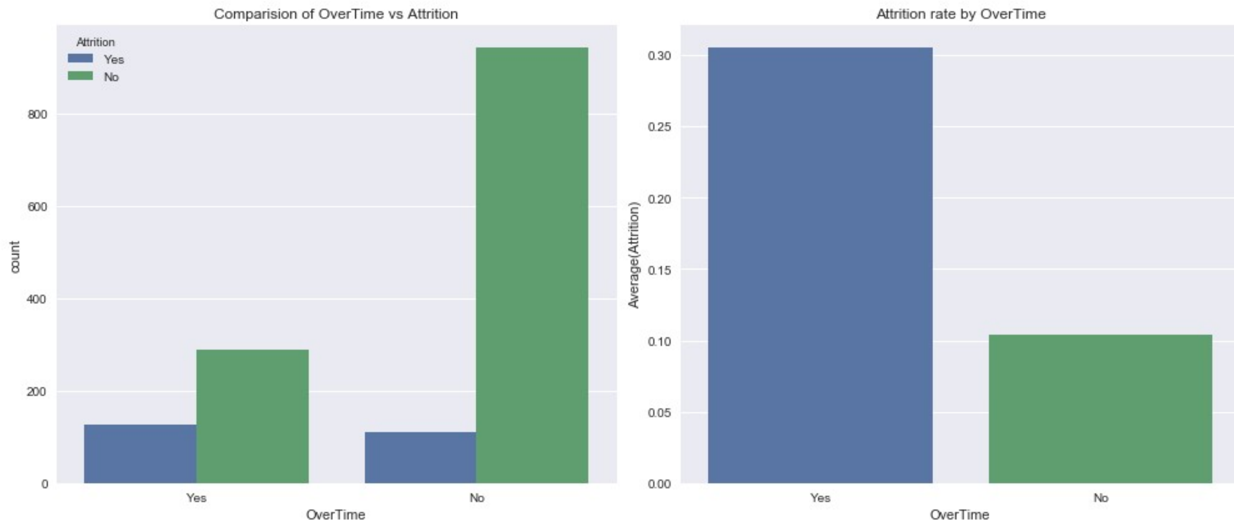
More than 60% of the employee's rated that they have Better worklife balance and 10% rated for Best worklife balance

```
CategoricalVariables_targetPlots(employee_data,"WorkLifeBalance")
```



- As expected more than 30% of the people who rated as Bad WorkLifeBalance have left the company and around 15% of the people who rated for Best WorkLifeBalance also left the company

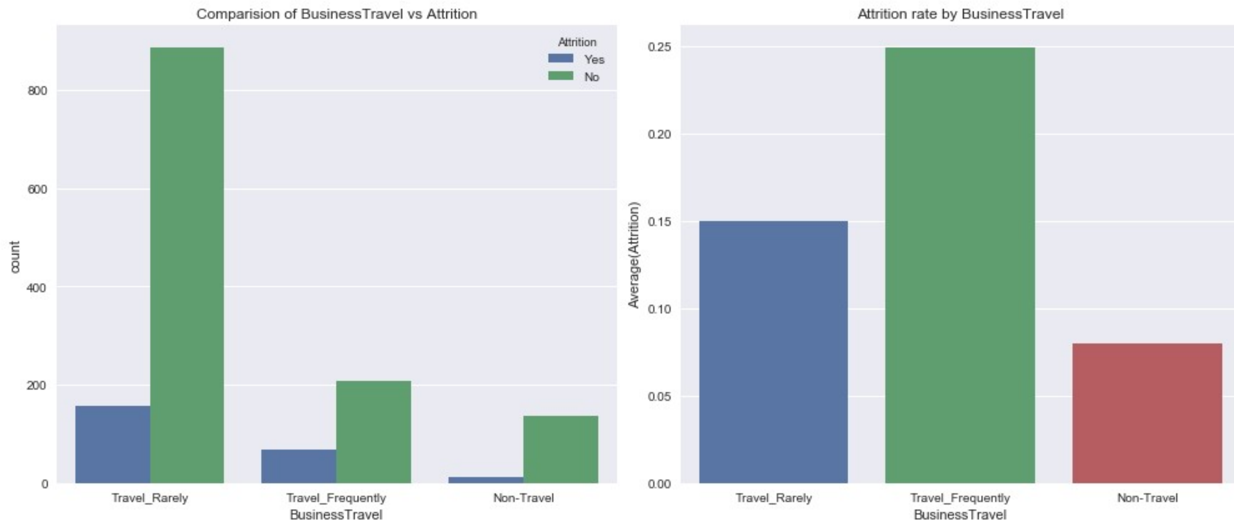
```
CategoricalVariables_targetPlots(employee_data,"OverTime")
```



More than 30% of employee's who worked overtime has left the company, where as 90% of employee's who have not experienced overtime has not left the company. Therefore overtime is a strong indicator of attrition

BusinessTravel

```
CategoricalVariables_targetPlots(employee_data,segment_by="BusinessTra
vel")
```



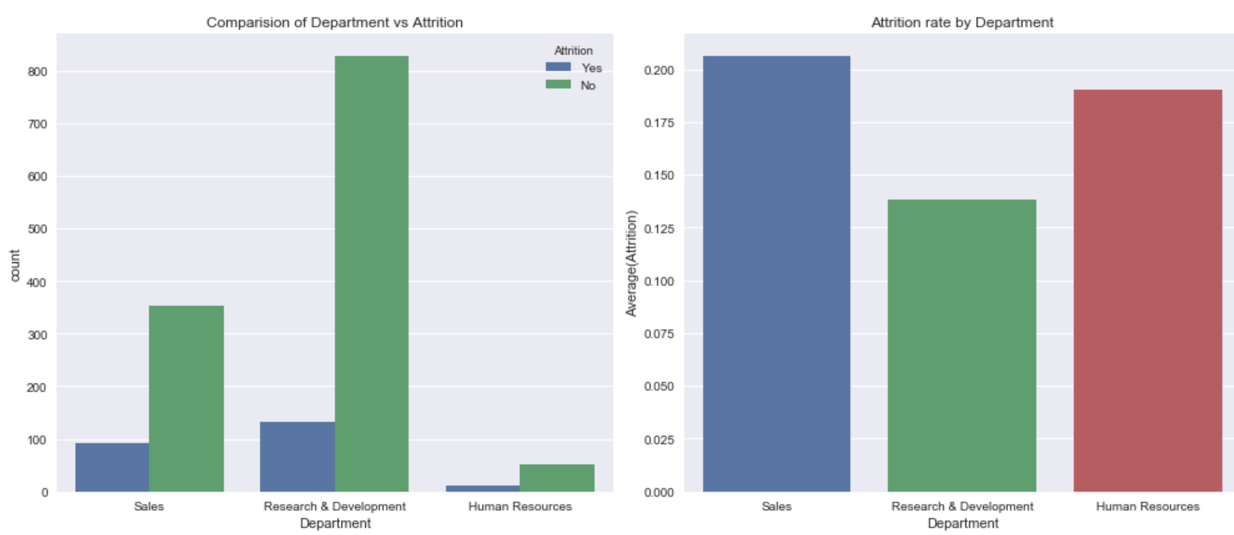
- There are more people who travel rarely compared to people who travel frequently. In case of people who travel Frequently around 25% of people have left the company and in other cases attrition rate doesn't vary significantly on travel

Department

```
employee_data.Department.value_counts()
```

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

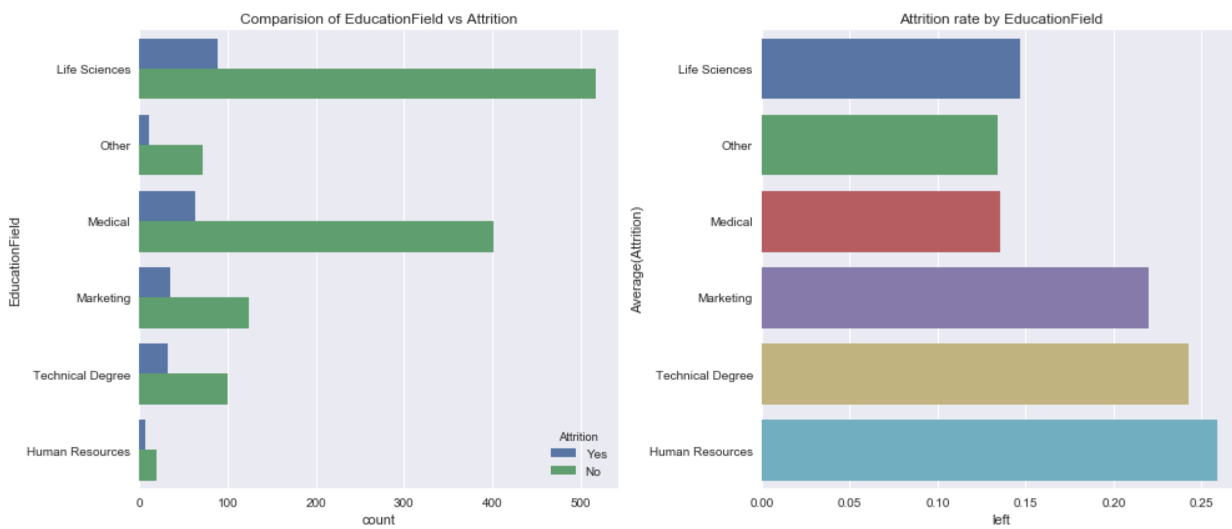
```
CategoricalVariables_targetPlots(employee_data,segment_by="Department"
)
```



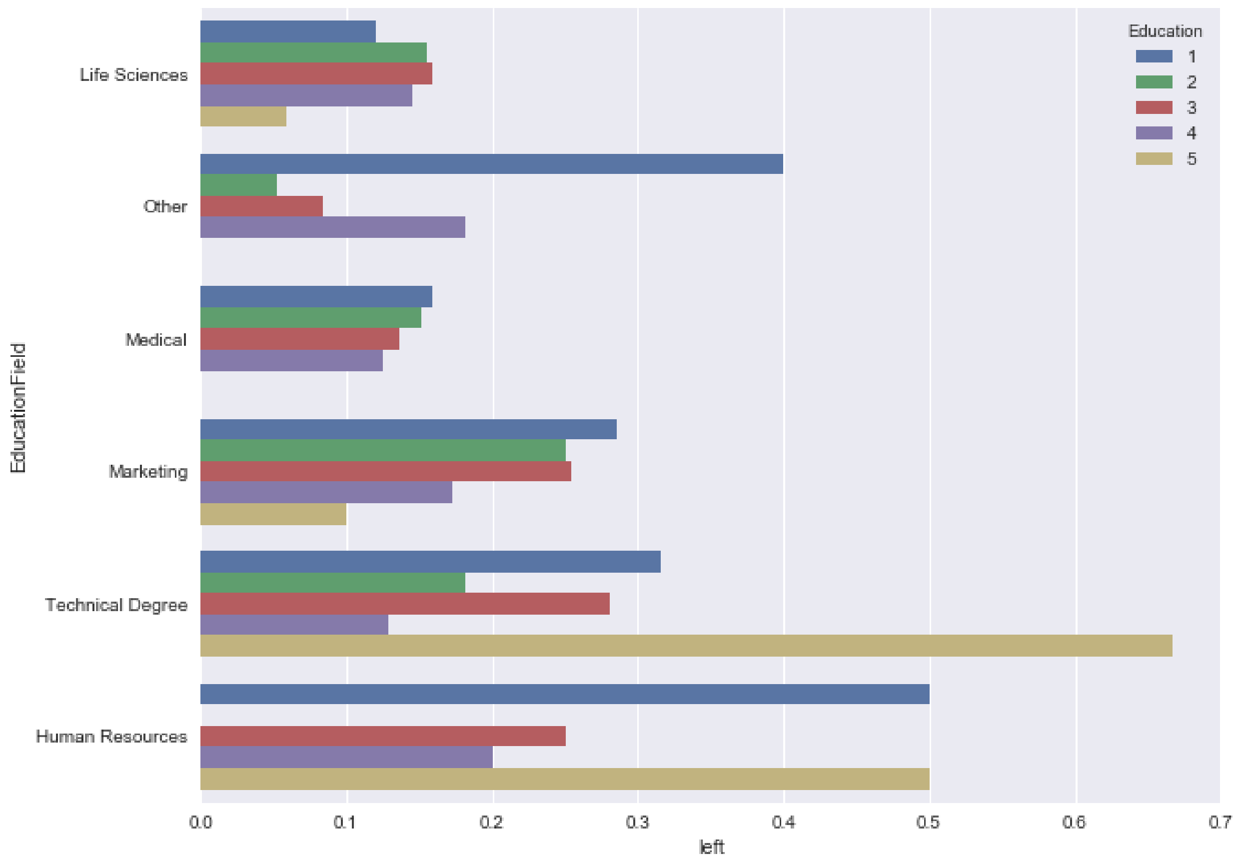
- On comparing departmentwise,we can conclude that HR has seen only a marginal high in turnover rates whereas the numbers are significant in sales department with turnover rates of 39 %. The attrition levels are not appreciable in R & D where 67 % have recorded no attrition.
- Sales has seen higher attrition levels about 20.6% followed by HR around 18%

EducationField

```
employee_data.EducationField.value_counts()

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

CategoricalVariables_targetPlots(employee_data,"EducationField",invert
_axis=True)
```



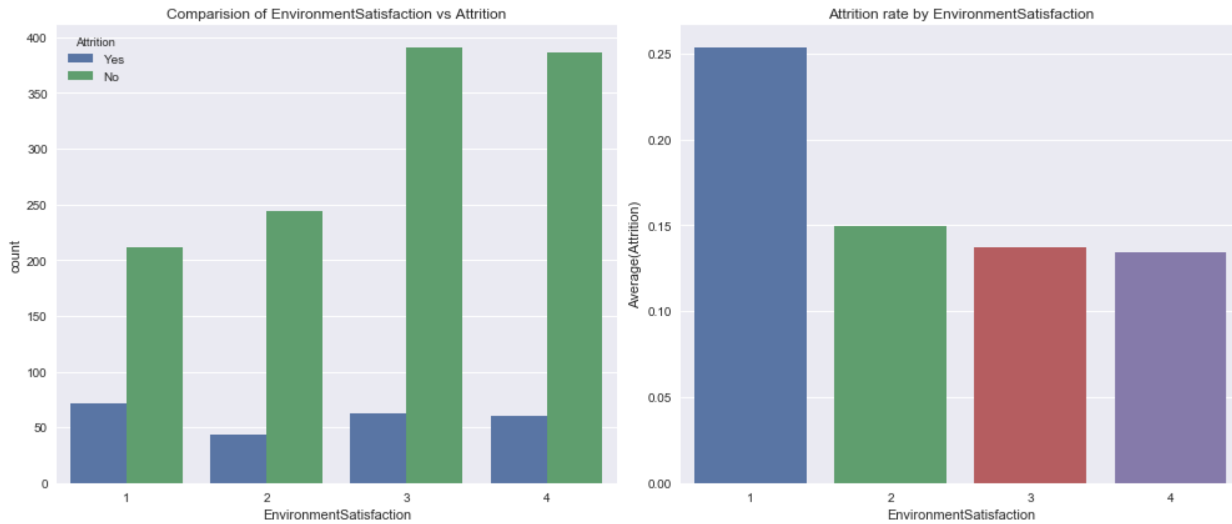

```
plt.figure(figsize=(10,8))
sns.barplot(y = "EducationField", x = "left", hue="Education",
data=employee_data,ci=None)
plt.show()
```



- There are more people with a Life sciences followed by medical and marketing
- Employee's in the EducationField of Human Resources and Technical Degree have highest attrition levels around 26% and 23% respectively
- When compared with Education level, we have observed that employees in the highest level of education in there field of study have left the company. We can conclude that EducationField is a strong indicator of attrition

EnvironmentSatisfaction

```
CategoricalVariables_targetPlots(employee_data,"EnvironmentSatisfactio
n")
```

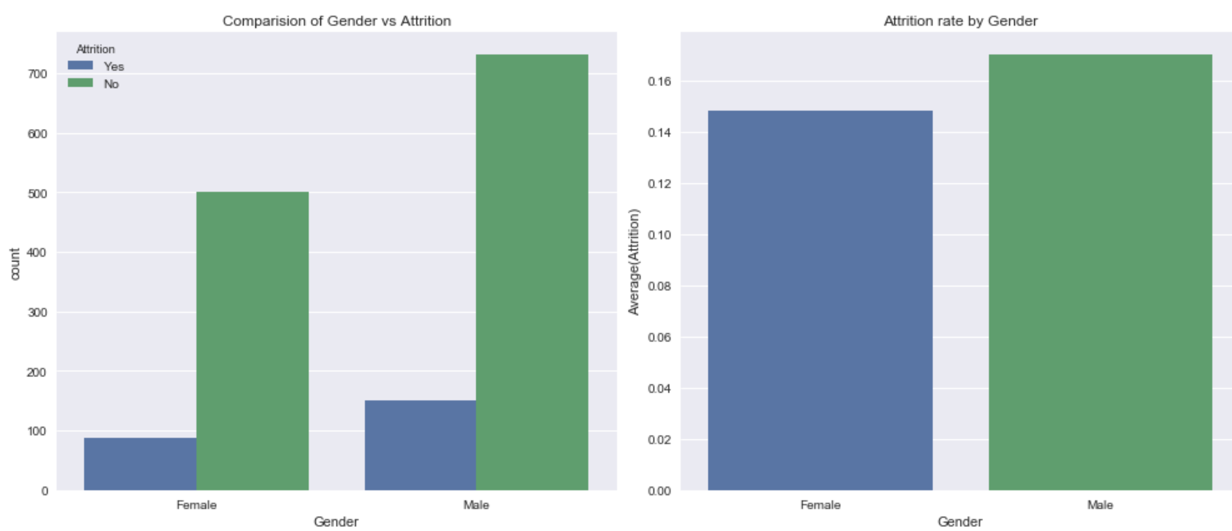


we can see that people having low environment satisfaction 25% leave the company

Gender Vs Attrition

```
sns.boxplot(employee_data['Gender'], employee_data['MonthlyIncome'])
plt.title('MonthlyIncome vs Gender Box Plot', fontsize=20)
plt.xlabel('MonthlyIncome', fontsize=16)
plt.ylabel('Gender', fontsize=16)
plt.show()
```

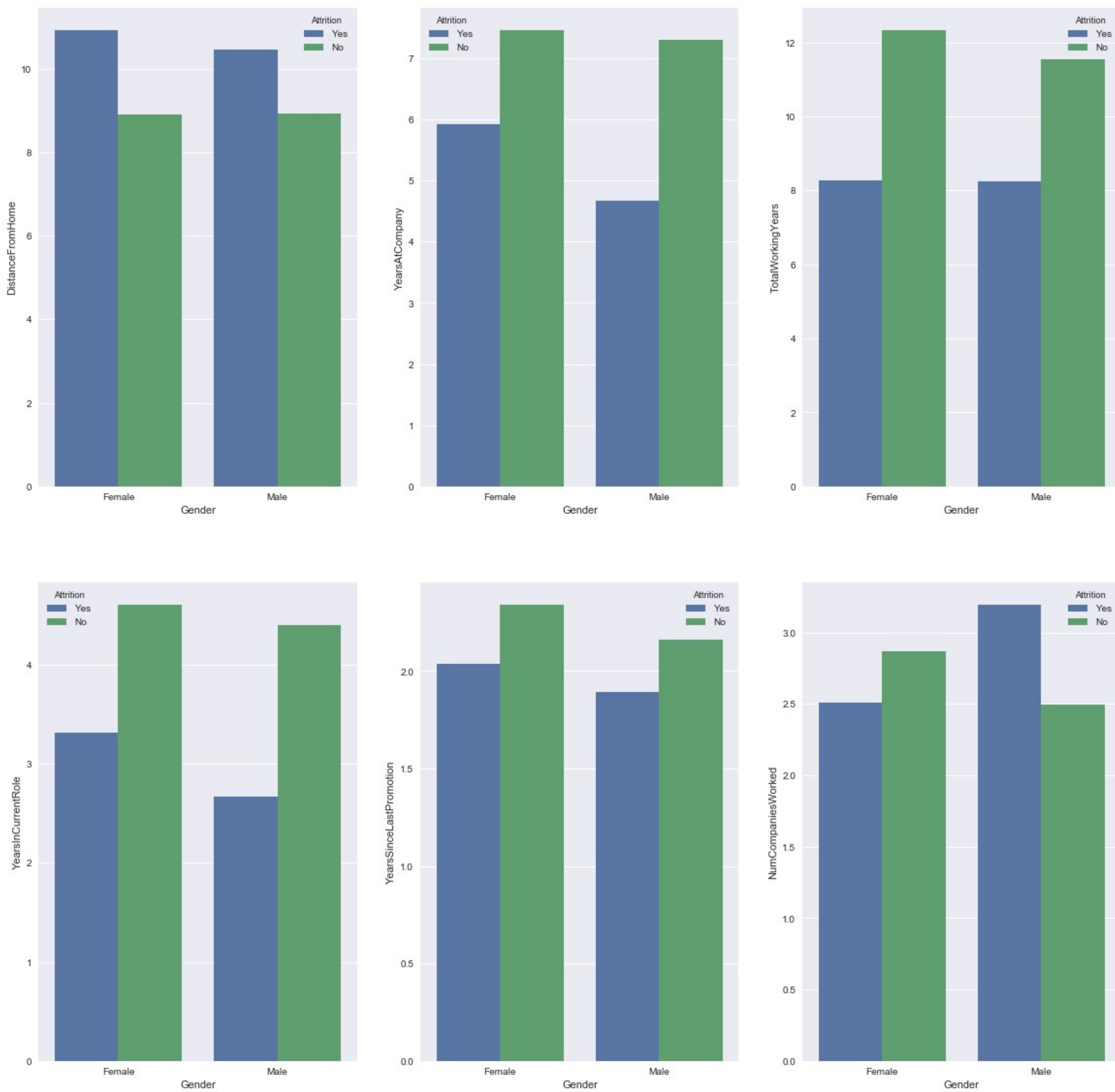

```
CategoricalVariables_targetPlots(employee_data,"Gender")
```



- Monthly Income distribution for Male and Female is almost similar, so the attrition rate of Male and Female is almost the same around 15%. Gender is not a strong indicator of attrition

```
fig,ax = plt.subplots(2,3, figsize=(20,20))               # 'ax' has
references to all the four axes
plt.suptitle("Comparision of various factors vs Gender", fontsize=20)
sns.barplot(employee_data['Gender'],employee_data['DistanceFromHome'],
hue = employee_data['Attrition'], ax = ax[0,0],ci=None);
sns.barplot(employee_data['Gender'],employee_data['YearsAtCompany'],hu
e = employee_data['Attrition'], ax = ax[0,1],ci=None);
sns.barplot(employee_data['Gender'],employee_data['TotalWorkingYears']
,hue = employee_data['Attrition'], ax = ax[0,2],ci=None);
sns.barplot(employee_data['Gender'],employee_data['YearsInCurrentRole'
],hue = employee_data['Attrition'], ax = ax[1,0],ci=None);
sns.barplot(employee_data['Gender'],employee_data['YearsSinceLastPromo
tion'],hue = employee_data['Attrition'], ax = ax[1,1],ci=None);
sns.barplot(employee_data['Gender'],employee_data['NumCompaniesWorked'
],hue = employee_data['Attrition'], ax = ax[1,2],ci=None);
plt.show()
```

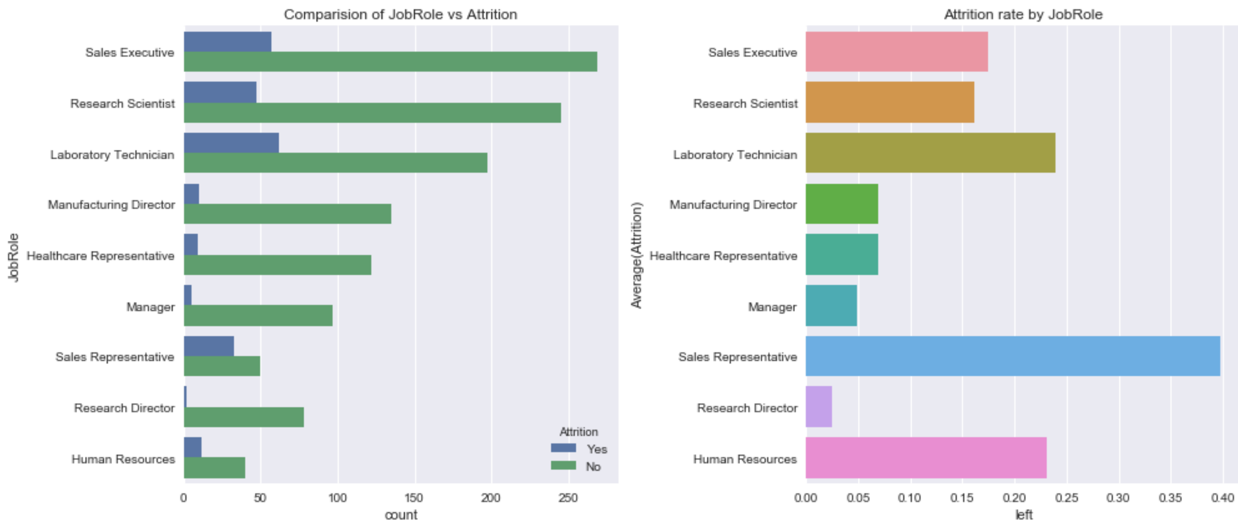
Comparision of various factors vs Gender



1. Distance from home matters to women employees more than men.
2. Female employes are spending more years in one company compare to their counterpart.
3. Female employes spending more years in current company are more inclined to switch.

Job Role

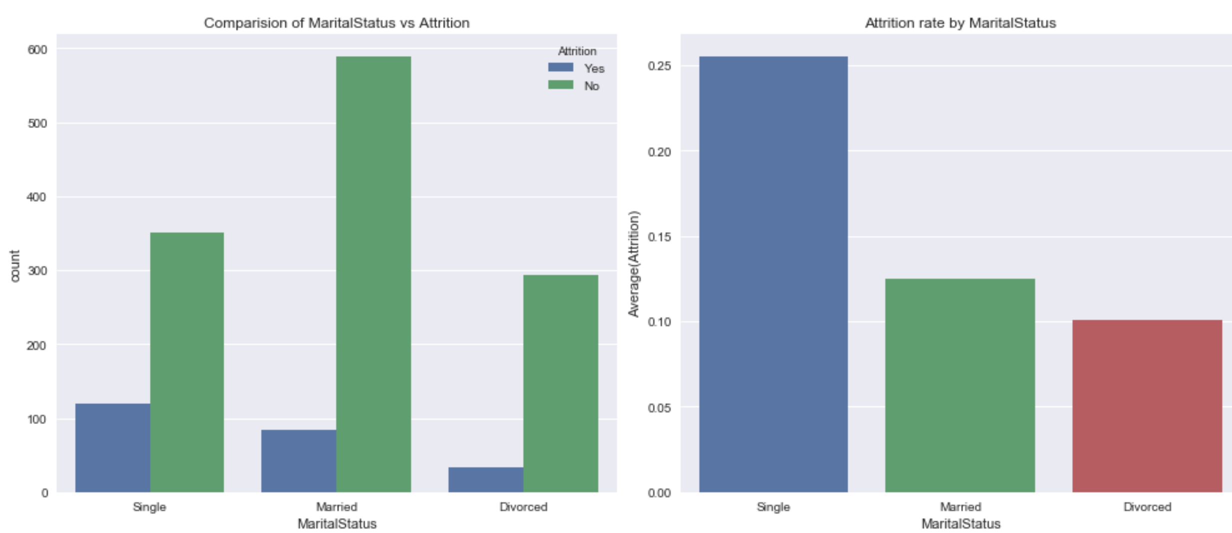
```
CategoricalVariables_targetPlots(employee_data,"JobRole",invert_axis=True)
```



1. Jobs held by the employee is maximum in Sales Executive, then R&D , then Laboratory Technician
2. People working in Sales department is most likely quit the company followed by Laboratory Technician and Human Resources there attrition rates are 40%, 24% and 22% respectively

Marital Status

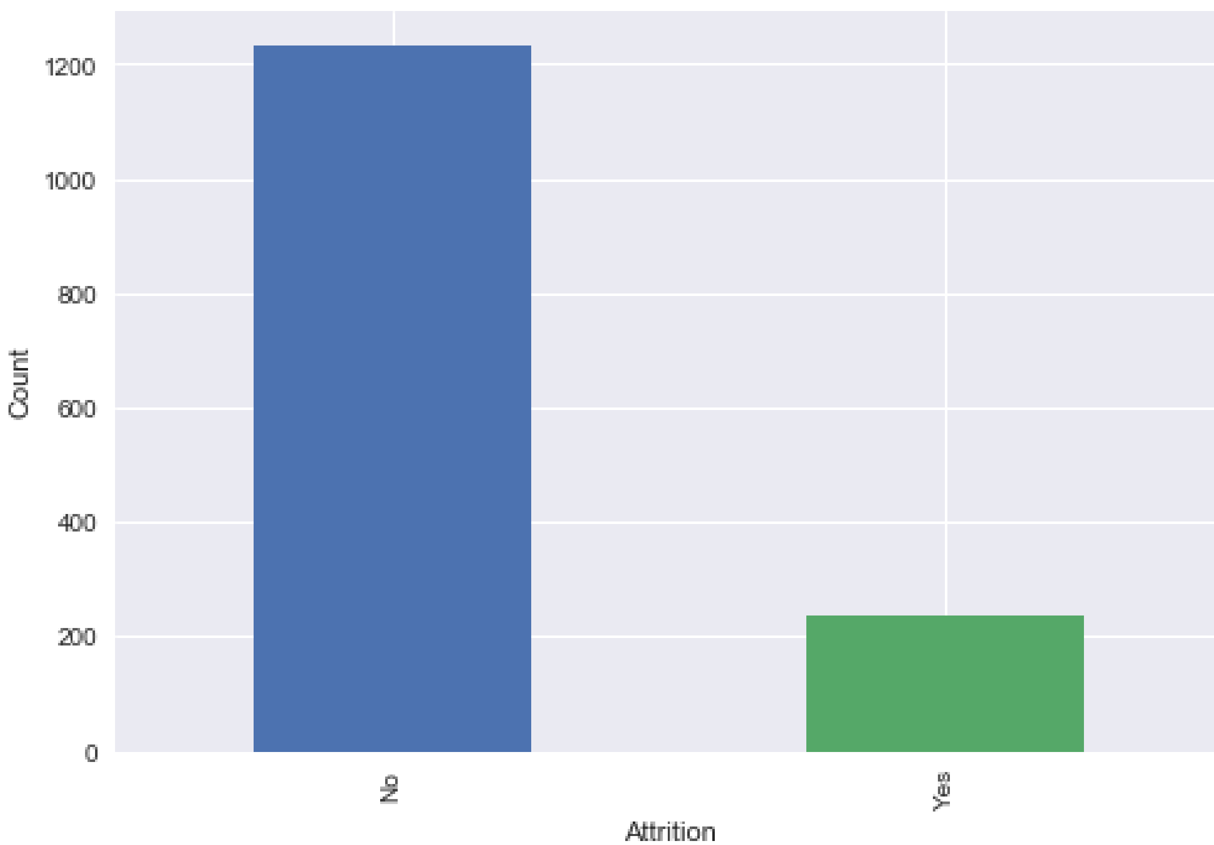
```
CategoricalVariables_targetPlots(employee_data,"MaritalStatus")
```



From the plot,it is understood that irrespective of the marital status,there are large people who stay with the company and do not leave.Therefore,marital status is a weak predictor of attrition

Building Decision Tree

```
from sklearn.model_selection import train_test_split

#for fitting classification tree
from sklearn.tree import DecisionTreeClassifier

#to create a confusion matrix
from sklearn.metrics import confusion_matrix

#import whole class of metrics
from sklearn import metrics

employee_data.Attrition.value_counts().plot(kind = "bar")
plt.xlabel("Attrition")
plt.ylabel("Count")
plt.show()
```




```
employee_data["Attrition"].value_counts()
```

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

From the Exploratory data analysis, variable that are not significant to attrition are:

- EmployeeCount, EmployeeNumber, Gender, HourlyRate, JobLevel, MaritalStatus, Over18, StandardHours

```
#copying the main employee data to another dataframe
employee_data_new = employee_data.copy()

#dropping the not significant variables
employee_data_new.drop(["EmployeeCount","EmployeeNumber","Gender","Hou
rlyRate","Over18","StandardHours","left"], axis=1,inplace=True)
```

Handling Categorical Variables

- Segregate the numerical and Categorical variables
- Convert Categorical variables to dummy variables

```
#data types of variables
dict(employee_data_new.dtypes)

{'Age': dtype('int64'),
 'Attrition': dtype('O'),
 'BusinessTravel': dtype('O'),
 'DailyRate': dtype('int64'),
 'Department': dtype('O'),
 'DistanceFromHome': dtype('int64'),
 'Education': dtype('int64'),
 'EducationField': dtype('O'),
 'EnvironmentSatisfaction': dtype('int64'),
 'JobInvolvement': dtype('int64'),
 'JobLevel': dtype('int64'),
 'JobRole': dtype('O'),
 'JobSatisfaction': dtype('int64'),
 'MaritalStatus': dtype('O'),
 'MonthlyIncome': dtype('int64'),
 'MonthlyRate': dtype('int64'),
 'NumCompaniesWorked': dtype('int64'),
 'OverTime': dtype('O'),
 'PercentSalaryHike': dtype('int64'),
 'PerformanceRating': dtype('int64'),
 'RelationshipSatisfaction': dtype('int64'),
 'StockOptionLevel': dtype('int64'),
 'TotalWorkingYears': dtype('int64'),
```

```
 'TrainingTimesLastYear': dtype('int64'),
 'WorkLifeBalance': dtype('int64'),
 'YearsAtCompany': dtype('int64'),
 'YearsInCurrentRole': dtype('int64'),
 'YearsSinceLastPromotion': dtype('int64'),
 'YearsWithCurrManager': dtype('int64')}

#segregating the variables based on datatypes

numeric_variable_names  = [key for key in
dict(employee_data_new.dtypes) if dict(employee_data_new.dtypes)[key]
in ['float64', 'int64', 'float32', 'int32']]

categorical_variable_names = [key for key in
dict(employee_data_new.dtypes) if dict(employee_data_new.dtypes)[key]
in ["object"]]

categorical_variable_names

['Attrition',
 'BusinessTravel',
 'Department',
 'EducationField',
 'JobRole',
 'MaritalStatus',
 'OverTime']

#store the numerical variables data in seperate dataset

employee_data_num = employee_data_new[numeric_variable_names]

#store the categorical variables data in seperate dataset

employee_data_cat = employee_data_new[categorical_variable_names]
#dropping the attrition
employee_data_cat.drop(["Attrition"],axis=1,inplace=True)

#converting into dummy variables

employee_data_cat = pd.get_dummies(employee_data_cat)

#Merging the both numerical and categorical data

employee_data_final = pd.concat([employee_data_num,
employee_data_cat,employee_data_new[["Attrition"]]],axis=1)

employee_data_final.head()

   Age  DailyRate  DistanceFromHome  Education
EnvironmentSatisfaction  \
0   41       1102                 1          2
2
```

```
1   49        279                 8          1
3
2   37       1373                 2          2
4
3   33       1392                 3          4
4
4   27        591                 2          1
1

   JobInvolvement  JobLevel  JobSatisfaction  MonthlyIncome
MonthlyRate  \
0                3         2                4           5993
19479
1                2         2                2           5130
24907
2                2         1                3           2090
2396
3                3         1                3           2909
23159
4                3         1                2           3468
16632

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

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

   MaritalStatus_Divorced  MaritalStatus_Married  MaritalStatus_Single
\
0                       0                      0                     1

1                       0                      1                     0

2                       0                      0                     1

3                       0                      1                     0

4                       0                      1                     0


   OverTime_No  OverTime_Yes  Attrition
```

```
0             0             1        Yes
1             1             0         No
2             0             1        Yes
3             0             1         No
4             1             0         No

[5 rows x 49 columns]

#final features
features =
list(employee_data_final.columns.difference(["Attrition"]))

features

['Age',
 'BusinessTravel_Non-Travel',
 'BusinessTravel_Travel_Frequently',
 'BusinessTravel_Travel_Rarely',
 'DailyRate',
 'Department_Human Resources',
 'Department_Research & Development',
 'Department_Sales',
 'DistanceFromHome',
 'Education',
 'EducationField_Human Resources',
 'EducationField_Life Sciences',
 'EducationField_Marketing',
 'EducationField_Medical',
 'EducationField_Other',
 'EducationField_Technical Degree',
 'EnvironmentSatisfaction',
 'JobInvolvement',
 'JobLevel',
 'JobRole_Healthcare Representative',
 'JobRole_Human Resources',
 'JobRole_Laboratory Technician',
 'JobRole_Manager',
 'JobRole_Manufacturing Director',
 'JobRole_Research Director',
 'JobRole_Research Scientist',
 'JobRole_Sales Executive',
 'JobRole_Sales Representative',
 'JobSatisfaction',
 'MaritalStatus_Divorced',
 'MaritalStatus_Married',
 'MaritalStatus_Single',
 'MonthlyIncome',
 'MonthlyRate',
 'NumCompaniesWorked',
 'OverTime_No',
```

```
 'OverTime_Yes',
 'PercentSalaryHike',
 'PerformanceRating',
 'RelationshipSatisfaction',
 'StockOptionLevel',
 'TotalWorkingYears',
 'TrainingTimesLastYear',
 'WorkLifeBalance',
 'YearsAtCompany',
 'YearsInCurrentRole',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager']
```

Separating the Target and the Predictors

```
#seperating the target and predictors

X = employee_data_final[features]
y = employee_data_final[["Attrition"]]

X.shape
```

```
(1470, 48)
```

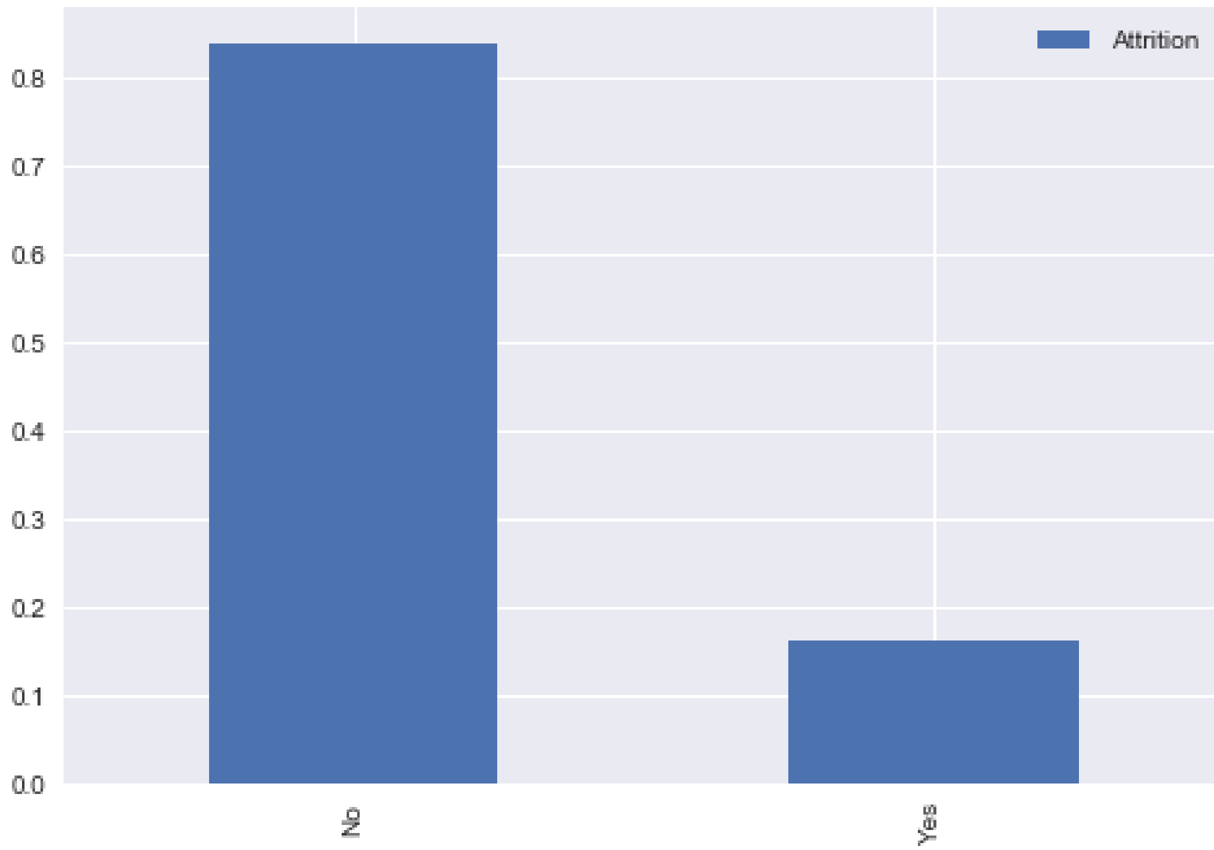
Train-Test Split(Stratified Sampling of Y)

```
# Function for creating model pipelines
from sklearn.pipeline import make_pipeline

#function for crossvalidate score
from sklearn.model_selection import cross_validate

#to find the best
from sklearn.model_selection import GridSearchCV

X_train, X_test, y_train,y_test = train_test_split(X,y,test_size =
0.3,stratify = y,random_state = 100)

#Checks
#Proportion in training data
y_train.Attrition.value_counts()/len(y_train)
```

```
No     0.838678
Yes    0.161322
Name: Attrition, dtype: float64
```

```
#Checks
#Proportion in training data
pd.DataFrame(y_train.Attrition.value_counts()/len(y_train)).plot(kind
```

```
= "bar")
plt.show()
```




```
#Proportion of test data
y_test.Attrition.value_counts()/len(y_test)

No     0.839002
Yes    0.160998
Name: Attrition, dtype: float64

#make a pipeline for decision tree model

pipelines = {
    "clf":
make_pipeline(DecisionTreeClassifier(max_depth=3,random_state=100))
}
```

Cross Validate

- To check the accuracy of the pipeline

```
scores = cross_validate(pipelines['clf'], X_train,
y_train,return_train_score=True)
```

```
scores['test_score'].mean()
```

```
0.8338352836423178
```

Average accuracy of pipeline with Decision Tree Classifier is 83.48%

Cross-Validation and Hyper Parameters Tuning

Cross Validation is the process of finding the best combination of parameters for the model by traning and evaluating the model for each combination of the parameters

- Declare a hyper-parameters to fine tune the Decision Tree Classifier

Decision Tree is a greedy alogritum it searches the entire space of possible decision trees. so we need to find a optimum parameter(s) or criteria for stopping the decision tree at some point. We use the hyperparameters to prune the decision tree

```
decisiontree_hyperparameters = {
    "decisiontreeclassifier__max_depth": np.arange(3,12),
    "decisiontreeclassifier__max_features": np.arange(3,10),
    "decisiontreeclassifier__min_samples_split":
[2,3,4,5,6,7,8,9,10,11,12,13,14,15],
    "decisiontreeclassifier__min_samples_leaf" : np.arange(1,3)
}

pipelines['clf']
```

```
Pipeline(memory=None,
     steps=[('decisiontreeclassifier',
DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=3,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False,
random_state=100,
            splitter='best'))])
```

Decision Tree classifier with gini index

Fit and tune models with cross-validation

Now that we have our pipelines and hyperparameters dictionaries declared, we're ready to tune our models with cross-validation.

- We are doing 5 fold cross validation

```
#Create a cross validation object from decision tree classifier and
it's hyperparameters
```

```
clf_model = GridSearchCV(pipelines['clf'],
decisiontree_hyperparameters, cv=5, n_jobs=-1)
```

```
#fit the model with train data
clf_model.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, error_score='raise',
       estimator=Pipeline(memory=None,
     steps=[('decisiontreeclassifier',
DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=3,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False,
random_state=100,
            splitter='best'))]),
       fit_params=None, iid=True, n_jobs=-1,
       param_grid={'decisiontreeclassifier__max_depth': array([ 3,  4,
5,  6,  7,  8,  9, 10, 11]), 'decisiontreeclassifier__max_features':
array([3, 4, 5, 6, 7, 8, 9]),
'decisiontreeclassifier__min_samples_split': [2, 3, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 14, 15], 'decisiontreeclassifier__min_samples_leaf':
array([1, 2])},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

```
#Display the best parameters for Decision Tree Model
clf_model.best_params_
```

```
{'decisiontreeclassifier__max_depth': 3,
 'decisiontreeclassifier__max_features': 7,
 'decisiontreeclassifier__min_samples_leaf': 1,
 'decisiontreeclassifier__min_samples_split': 10}
```

```
#Display the best score for the fitted model
clf_model.best_score_
```

```
0.8561710398445093
```

```
#In Pipeline we can use the string names to get the
decisiontreeclassifer

clf_model.best_estimator_.named_steps['decisiontreeclassifier']
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=3,
            max_features=7, max_leaf_nodes=None,
min_impurity_decrease=0.0,
```

```
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=10, min_weight_fraction_leaf=0.0,
            presort=False, random_state=100, splitter='best')

#saving into a variable to get graph

clf_best_model =
clf_model.best_estimator_.named_steps['decisiontreeclassifier']
```

Model Performance Evaluation

- On Test Data

```
#Making a dataframe of actual and predicted data from test set

tree_test_pred = pd.concat([y_test.reset_index(drop =
True),pd.DataFrame(clf_model.predict(X_test))],axis=1)
tree_test_pred.columns = ["actual","predicted"]

#setting the index to original index
tree_test_pred.index = y_test.index

tree_test_pred.head()

     actual predicted
34      Yes       Yes
1432     No        No
334      No        No
1068    Yes        No
736      No        No

#keeping only positive condition (yes for attrition)

pred_probability = pd.DataFrame(p[1] for p in
clf_model.predict_proba(X_test))
pred_probability.columns = ["predicted_prob"]
pred_probability.index = y_test.index

#merging the predicted data and its probability value

tree_test_pred = pd.concat([tree_test_pred,pred_probability],axis=1)

tree_test_pred.head()

     actual predicted  predicted_prob
34      Yes       Yes        0.632184
1432     No        No        0.220859
334      No        No        0.072165
1068    Yes        No        0.145985
736      No        No        0.072165
```

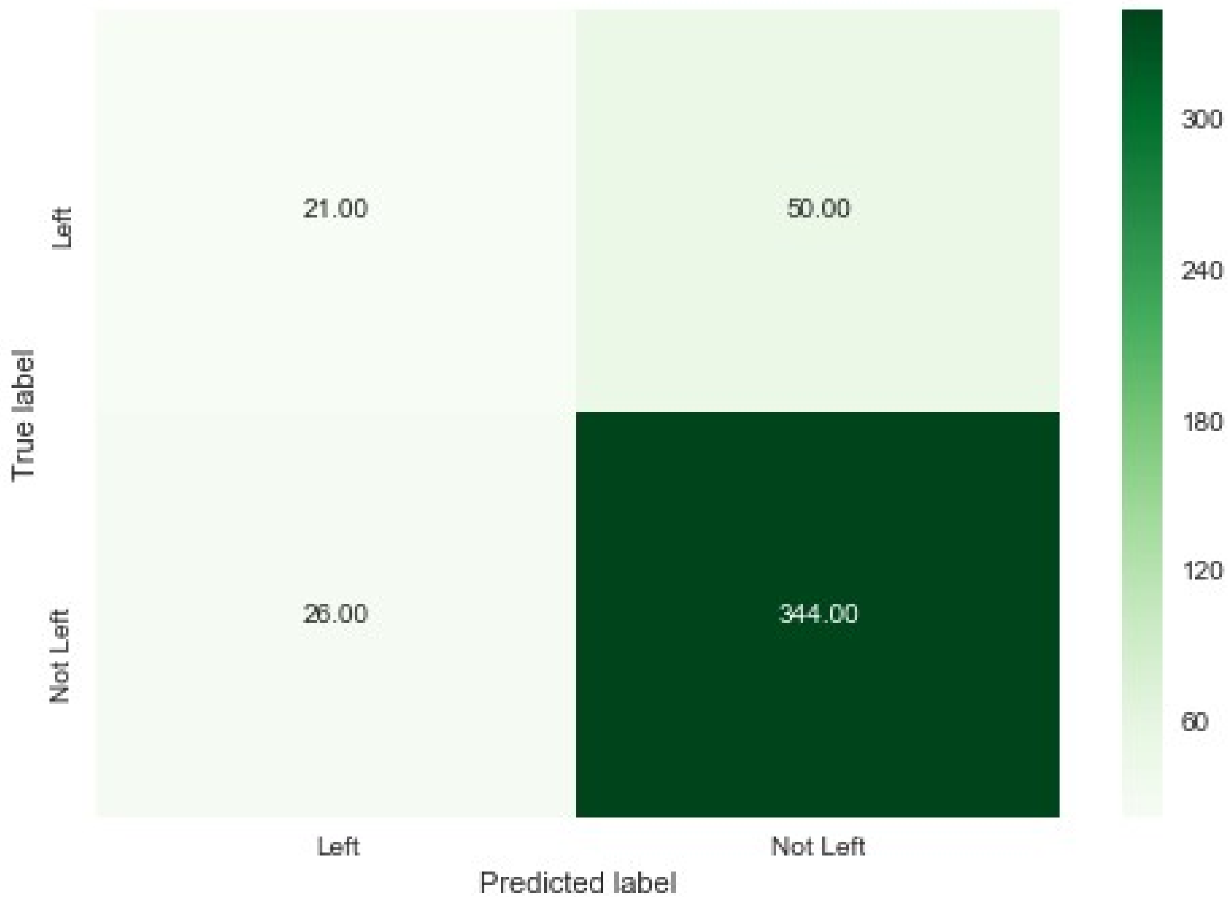
```
#converting the labels Yes --> 1 and No --> 0 for further operations
below

tree_test_pred["actual_left"] = np.where(tree_test_pred["actual"] ==
"Yes",1,0)
tree_test_pred["predicted_left"] =
np.where(tree_test_pred["predicted"] == "Yes",1,0)

tree_test_pred.head()
```

	actual	predicted	predicted_prob	actual_left	predicted_left
34	Yes	Yes	0.632184	1	1
1432	No	No	0.220859	0	0
334	No	No	0.072165	0	0
1068	Yes	No	0.145985	1	0
736	No	No	0.072165	0	0

Confusion Matrix

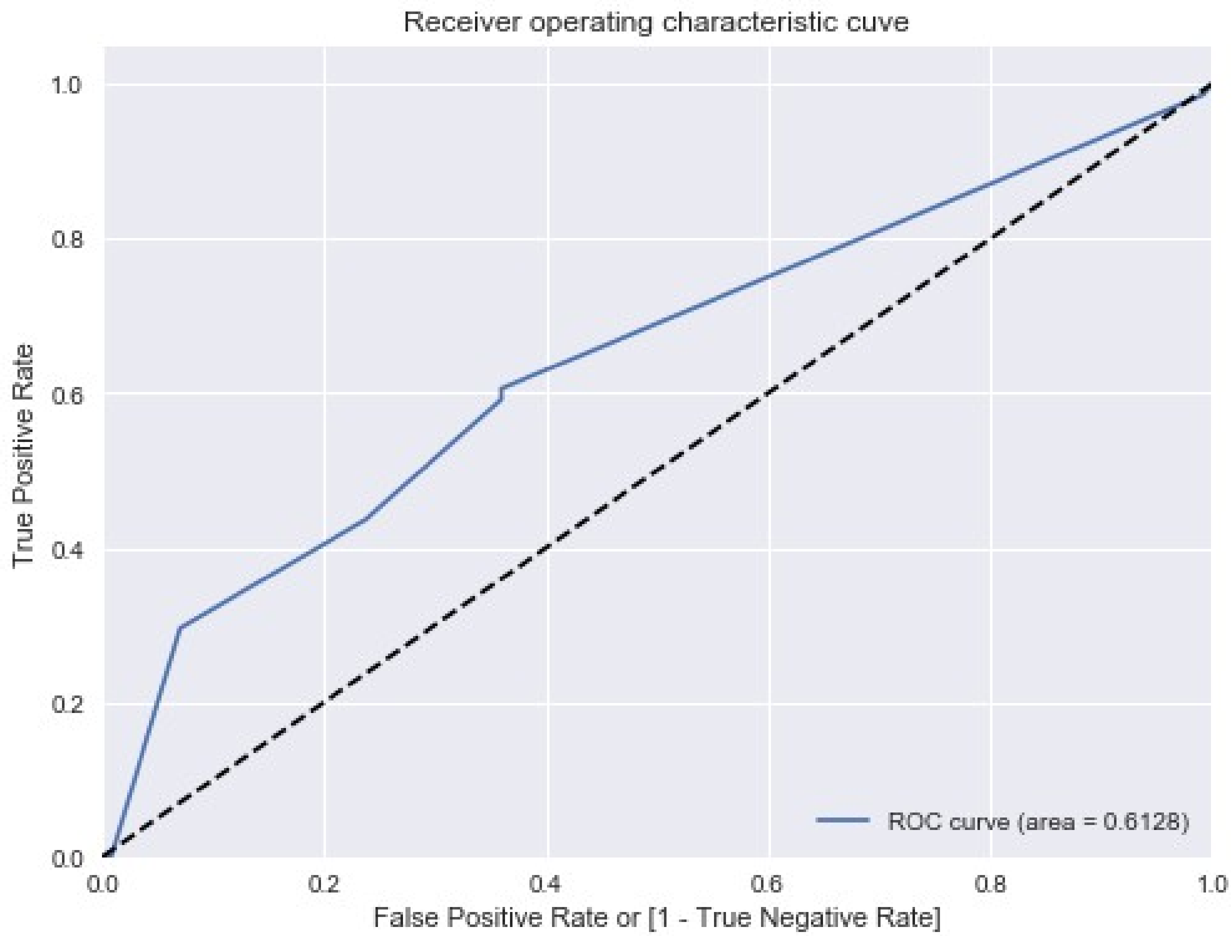
The confusion matrix is a way of tabulating the number of misclassifications, i.e., the number of predicted classes which ended up in a wrong classification bin based on the true classes.

```
#confusion matrix
metrics.confusion_matrix(tree_test_pred.actual,tree_test_pred.predicte
d,labels=["Yes","No"])

array([[ 21,  50],
       [ 26, 344]], dtype=int64)

#confusion matrix visualization using seaborn heatmap

sns.heatmap(metrics.confusion_matrix(tree_test_pred.actual,tree_test_p
red.predicted,

labels=["Yes","No"]),cmap="Greens",annot=True,fmt=".2f",
           xticklabels = ["Left", "Not Left"] , yticklabels = ["Left",
"Not Left"])

plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```




```
#Area Under ROC Curve

auc_score_test =
metrics.roc_auc_score(tree_test_pred.actual_left,tree_test_pred.predic
ted_left)
print("AUROC Score:",round(auc_score_test,4))

AUROC Score: 0.6128

##Plotting the ROC Curve

fpr, tpr, thresholds = metrics.roc_curve(tree_test_pred.actual_left,
tree_test_pred.predicted_prob,drop_intermediate=False)


plt.figure(figsize=(8, 6))
plt.plot( fpr, tpr, label='ROC curve (area = %0.4f)' % auc_score_test)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver operating characteristic cuve')
plt.legend(loc="lower right")
plt.show()
```



From the ROC Curve, we have a choice to make depending on the value we place on true positive and tolerance for false positive rate

- If we wish to find the more people who are leaving, we could increase the true positive rate by adjusting the probability cutoff for classification. However by doing so would also increase the false positive rate. we need to find the optimum value of cutoff for classification

Metrics

- Recall: Ratio of the total number of correctly classified positive examples divide to the total number of positive examples. High Recall indicates the class is correctly recognized
- Precision: To get the value of precision we divide the total number of correctly classified positive examples by the total number of predicted positive examples. High Precision indicates an example labeled as positive is indeed positive

```
#calculating the recall score
```

```
print("Recall
Score:",round(metrics.recall_score(tree_test_pred.actual_left,tree_tes
t_pred.predicted_left) * 100,3))
```

```
Recall Score: 29.577
```

```
#calculating the precision score

print("Precision
Score:",round(metrics.precision_score(tree_test_pred.actual_left,tree_
test_pred.predicted_left) * 100,3))
```

```
Precision Score: 44.681
```

```
print(metrics.classification_report(tree_test_pred.actual_left,tree_te
st_pred.predicted_left))
```

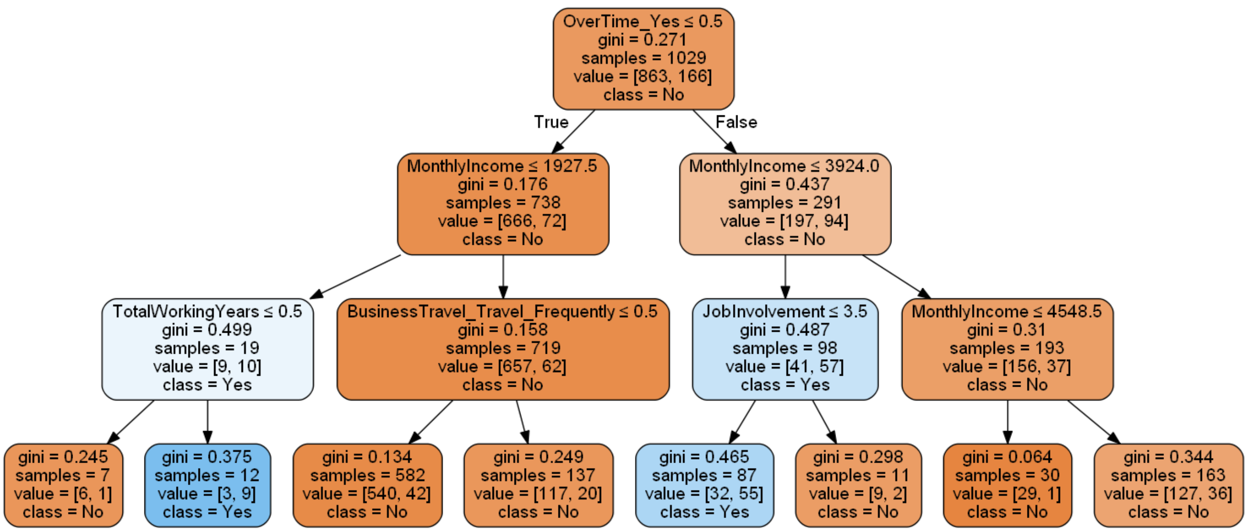
```
             precision    recall  f1-score   support

          0       0.87      0.93      0.90       370
          1       0.45      0.30      0.36        71

avg / total       0.80      0.83      0.81       441
```

Visualization of Decision Tree

- Dependencies
  - Need to install graphviz (conda install pydot graphviz)
  - Set the environment path variable to graphviz folder

```
# conda install pydot graphviz
#! pip install pydotplus

from sklearn.tree import export_graphviz

import pydotplus as pdot

from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus as pdot

#import os
#os.environ["PATH"] += os.pathsep +
'C:/Users/NiranjanKumar/Anaconda3/Library/bin/graphviz'

#write the dot data
dot_data = StringIO()

#export the decision tree along with the feature names into a dot file
format
```

```
export_graphviz(clf_best_model,out_file=dot_data,filled=True,
                rounded=True,special_characters=True,feature_names =
X_train.columns.values,class_names = ["No","Yes"])

#make a graph from dot file
graph = pdot.graph_from_dot_data(dot_data.getvalue())

Image(graph.create_png())
```




```
#export the tree diagram
graph.write_png("employee_attirtion.png")

True
```