In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv("HR-Employee-Attrition.csv")
df.head(5)
```

Out[1]:

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

5 rows × 35 columns

In [2]:

```
df.shape
```

Out[2]:

```
(1470, 35)
```

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [4]:
```
df.describe()
```

Out[4]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeI
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024
std	9.135373	403.509100	8.106864	1.024165	0.0	602
min	18.000000	102.000000	1.000000	1.000000	1.0	1
25%	30.000000	465.000000	2.000000	2.000000	1.0	491
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068

8 rows × 26 columns

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
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
```

In [6]:
```
# check no of duplicated row
df.duplicated().sum()
```

Out[6]: 0

In [7]:
```
# check misssinh value in percentile form
df.isnull().sum()/len(df)*100
```

Out[7]:
```
Age                         0.0
Attrition                   0.0
BusinessTravel              0.0
DailyRate                   0.0
Department                  0.0
DistanceFromHome            0.0
Education                   0.0
EducationField              0.0
EmployeeCount               0.0
EmployeeNumber              0.0
EnvironmentSatisfaction     0.0
Gender                      0.0
HourlyRate                  0.0
JobInvolvement              0.0
JobLevel                    0.0
JobRole                     0.0
JobSatisfaction             0.0
MaritalStatus               0.0
MonthlyIncome               0.0
MonthlyRate                 0.0
NumCompaniesWorked          0.0
Over18                      0.0
OverTime                    0.0
PercentSalaryHike           0.0
PerformanceRating           0.0
RelationshipSatisfaction    0.0
StandardHours               0.0
StockOptionLevel            0.0
TotalWorkingYears           0.0
TrainingTimesLastYear       0.0
WorkLifeBalance             0.0
YearsAtCompany              0.0
YearsInCurrentRole          0.0
YearsSinceLastPromotion     0.0
YearsWithCurrManager        0.0
dtype: float64
```

In [8]:
```
# check data types
df.dtypes
```

Out[8]:
```
Age                          int64
Attrition                   object
BusinessTravel              object
DailyRate                    int64
Department                  object
DistanceFromHome             int64
Education                    int64
EducationField              object
EmployeeCount                int64
EmployeeNumber               int64
EnvironmentSatisfaction      int64
Gender                      object
HourlyRate                   int64
JobInvolvement               int64
JobLevel                     int64
JobRole                     object
JobSatisfaction              int64
MaritalStatus               object
MonthlyIncome                int64
MonthlyRate                  int64
NumCompaniesWorked           int64
Over18                      object
OverTime                    object
PercentSalaryHike            int64
PerformanceRating            int64
RelationshipSatisfaction     int64
StandardHours                int64
StockOptionLevel             int64
TotalWorkingYears            int64
TrainingTimesLastYear        int64
WorkLifeBalance              int64
YearsAtCompany               int64
YearsInCurrentRole           int64
YearsSinceLastPromotion      int64
YearsWithCurrManager         int64
dtype: object
```

EDA(Exploratory data analysis)

In [9]:
```
#Attrition Rate
#Attrition rate: The attrition rate measures the percentage of employees who lea
#company in a given period of time.
df.columns
```

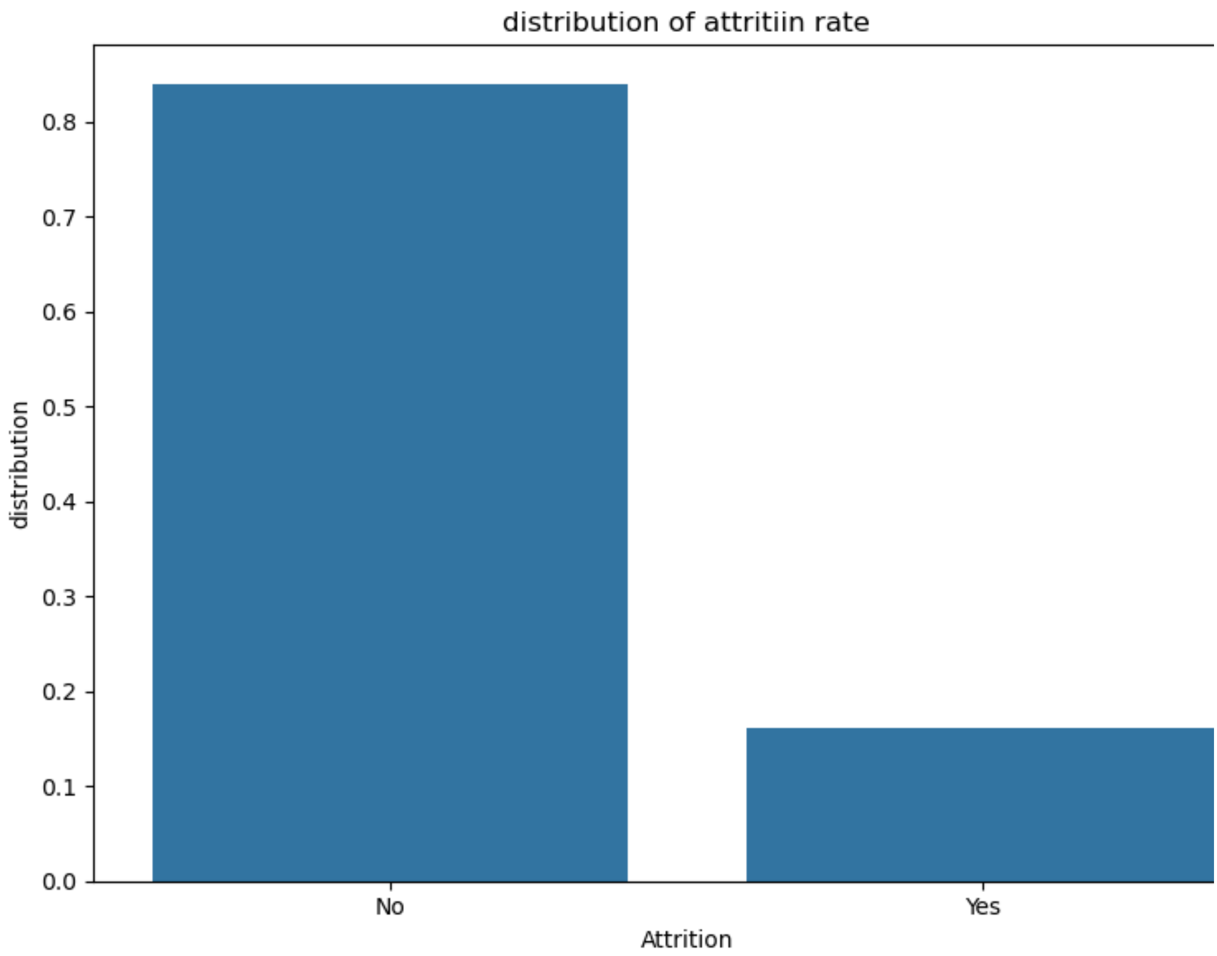
Out[9]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [10]:
```
df['Attrition'].value_counts(normalize=True)
```

Out[10]:
```
Attrition
No     0.838776
Yes    0.161224
Name: proportion, dtype: float64
```

In [11]:
```
attrition=df['Attrition'].value_counts(normalize=True)
```

In [14]:
```
plt.figure(figsize=(8,6))
sns.barplot(x=attrition.index,y=attrition)
plt.title("distribution of attritiin rate")
plt.xlabel("Attrition")
plt.ylabel("distribution")
plt.tight_layout()
plt.show()
```



In [15]:
```
#Average of Tenure
#Average tenure: The average tenure measures the average number of years an
#employee stays with the company before leaving.
df.columns
```

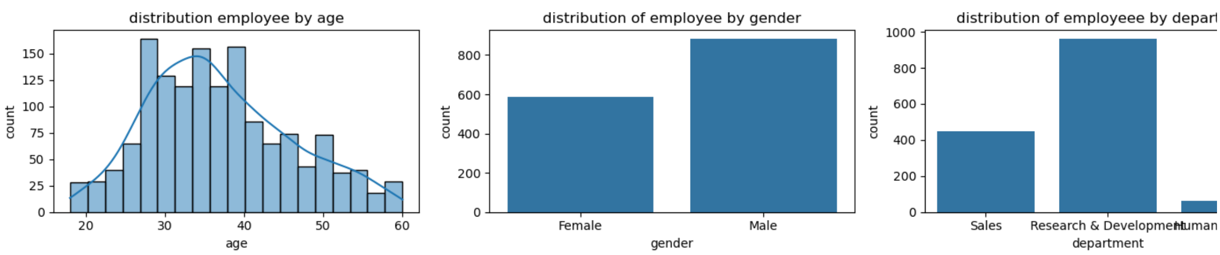
```
Out[15]: Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
               'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
               'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
               'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
               'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
               'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
               'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
               'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
               'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
               'YearsWithCurrManager'],
              dtype='object')
```

In [16]:
```
avg_tenure=df['YearsAtCompany'].mean()
```

In [17]:
```
print(f"the average tenure of the comapny is{avg_tenure}")
```

```
the average tenure of the comapny is7.0081632653061225
```

In [22]:
```
# Employee's Demographics
fig,axes=plt.subplots(nrows=1,ncols=3,figsize=(15,3))
df.columns
sns.histplot(data=df,x='Age',kde=True,ax=axes[0])
axes[0].set_title("distribution employee by age")
axes[0].set_xlabel("age")
axes[0].set_ylabel("count")

sns.countplot(data=df,x="Gender",ax=axes[1])
axes[1].set_title("distribution of employee by gender")
axes[1].set_xlabel("gender")
axes[1].set_ylabel("count")

sns.countplot(data=df,x="Department",ax=axes[2])
axes[2].set_title("distribution of employeee by department")
axes[2].set_xlabel("department")
axes[2].set_ylabel("count")


plt.tight_layout()
plt.show()
```



In [ ]:
```
# from the above visualization we can conclude that
# age:Most of the company's employees are in the 30-35 age group.
#gender:The majority of employees at this company are male.
# department:Most of the company's employees are concentrated in theresearch and
```

In [23]:
```
df_attrition=df[df['Attrition'] == 'Yes']
df_attrition.head(5)
```

Out[23]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Edu
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Li
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
14	28	Yes	Travel_Rarely	103	Research & Development	24	3	Li
21	36	Yes	Travel_Rarely	1218	Sales	9	4	Li
24	34	Yes	Travel_Rarely	699	Research & Development	6	1	

5 rows × 35 columns

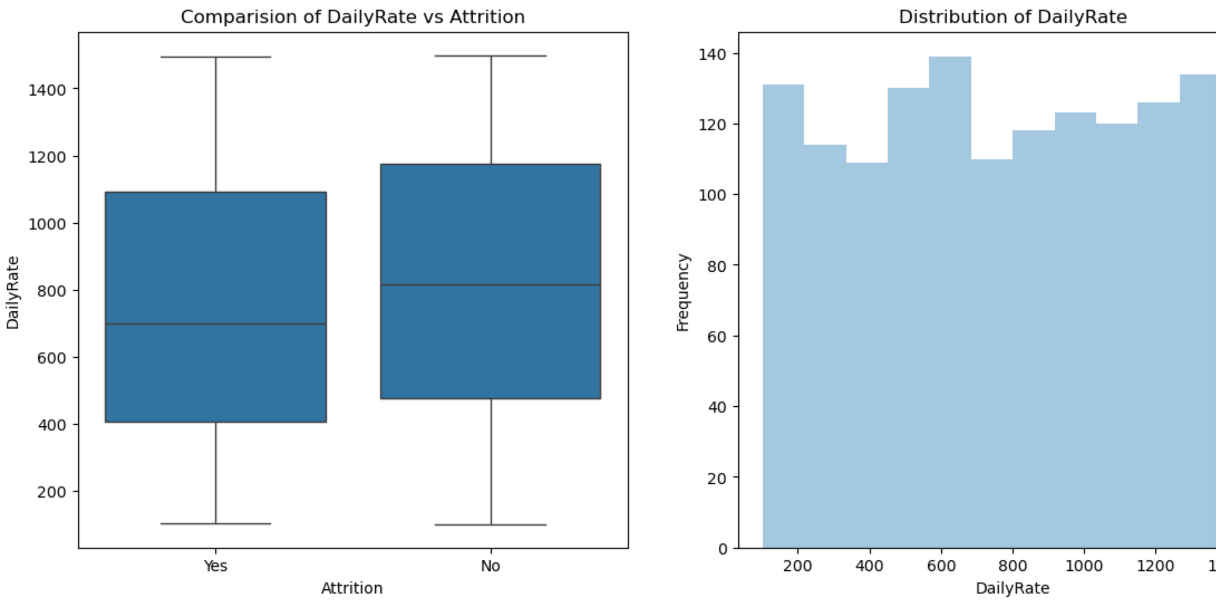
In [24]:

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

    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()
```

In [26]:

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

NumericalVariables_targetPlots(df,"DailyRate")
```

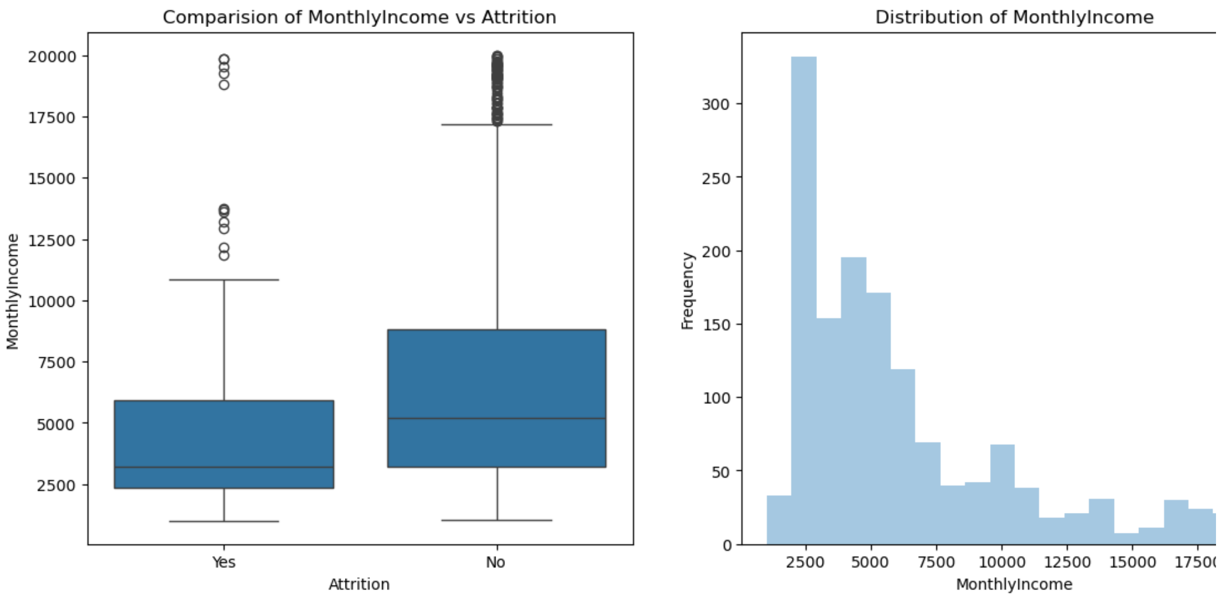
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_11348\1104822344.py:13: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(a = df[segment_by], ax=ax[1], kde=False)
```



In [ ]:

In [27]:
```
NumericalVariables_targetPlots(df,"MonthlyIncome")
```

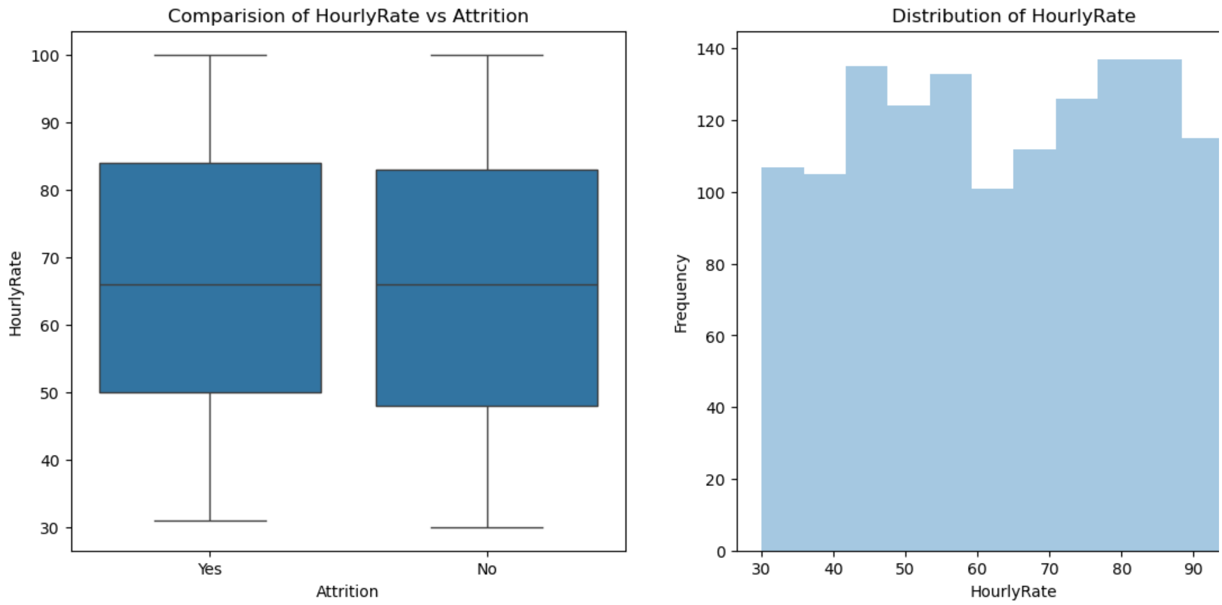
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_11348\1104822344.py:13: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(a = df[segment_by], ax=ax[1], kde=False)
```



In [ ]:
```
#Employee's working with lower daily rates are more prone to leave the company t
#working with higher rates.
```

In [28]:
```
NumericalVariables_targetPlots(df,"HourlyRate")
```

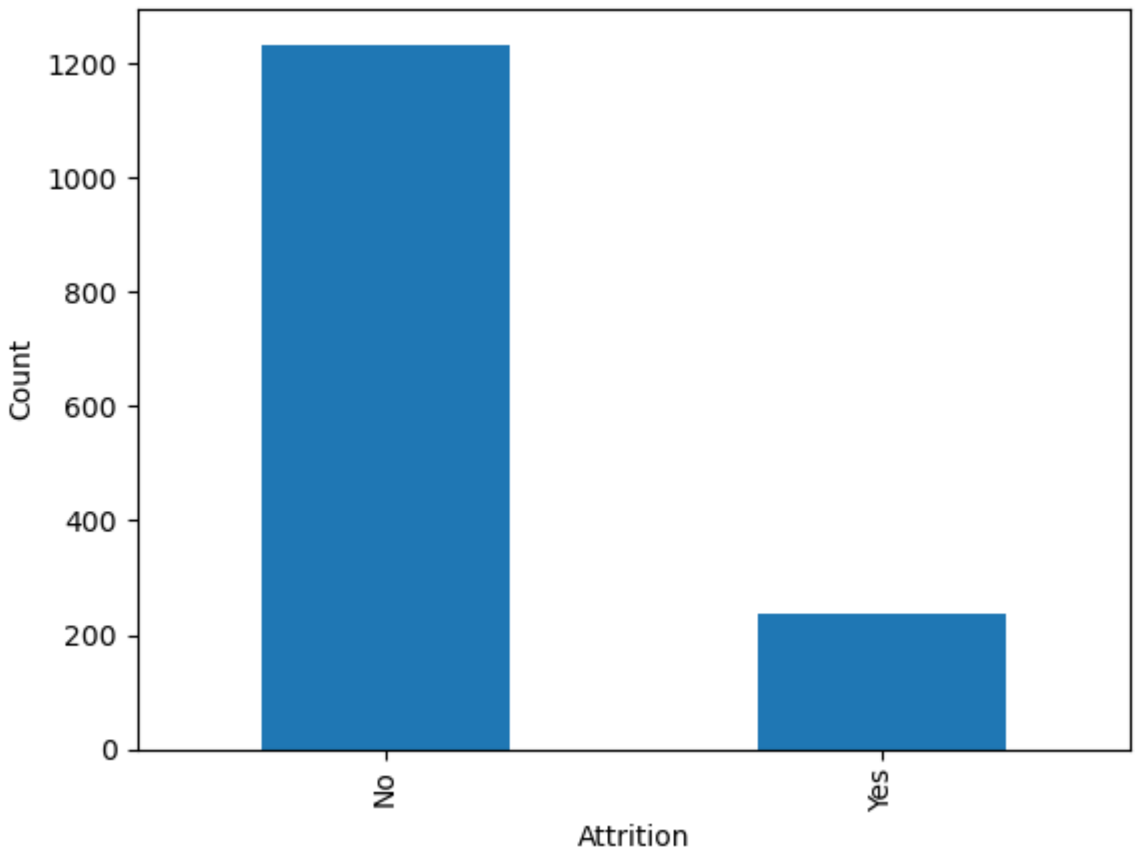
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_11348\1104822344.py:13: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(a = df[segment_by], ax=ax[1], kde=False)
```



In [ ]:
```
# from the above visualization we have seen that there is no significant differe
```

building decision treee

In [34]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
#to create a confusion matrix
from sklearn.metrics import confusion_matrix
from sklearn import metrics
df.Attrition.value_counts().plot(kind="bar")
plt.xlabel("Attrition")
plt.ylabel("Count")
plt.show()
```



In [35]:
```
df["Attrition"].value_counts()
```

Out[35]:
```
Attrition
No      1233
Yes      237
Name: count, dtype: int64
```

In [36]:
```
#From the Exploratory data analysis, variable that are not significant to attrit

#EmployeeCount, EmployeeNumber, Gender, HourlyRate, JobLevel, MaritalStatus, Ove
df_new=df.copy()
df_new.head(5)
```

Out[36]:

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

5 rows × 35 columns

In [15]:
```
df_new.drop(["EmployeeCount", "EmployeeNumber", "Gender", "HourlyRate", "JobLeve
```

In [16]:
```
df_new.shape
```

Out[16]: (1470, 27)

In [17]:
```
# handling categorical columns
#Segregate the numerical and Categorical variables
#Convert Categorical variables to dummy variables
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv("HR-Employee-Attrition.csv")

df_new=df.copy()
dict(df_new.dtypes)
```

Out[3]:
```
{'Age': dtype('int64'),
 'Attrition': dtype('O'),
 'BusinessTravel': dtype('O'),
 'DailyRate': dtype('int64'),
 'Department': dtype('O'),
 'DistanceFromHome': dtype('int64'),
 'Education': dtype('int64'),
 'EducationField': dtype('O'),
 'EmployeeCount': dtype('int64'),
 'EmployeeNumber': dtype('int64'),
 'EnvironmentSatisfaction': dtype('int64'),
 'Gender': dtype('O'),
 'HourlyRate': 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'),
 'Over18': dtype('O'),
 'OverTime': dtype('O'),
 'PercentSalaryHike': dtype('int64'),
 'PerformanceRating': dtype('int64'),
 'RelationshipSatisfaction': dtype('int64'),
 'StandardHours': 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')}
```

In [4]:
```
#segregating the variables based on datatypes
numeric_variable_names=[key for key in dict(df_new.dtypes)if dict(df_new.dtypes)
categorical_varibale_name=[key for key in dict(df_new.dtypes)if dict(df_new.dtyp
```

In [5]:
```
numeric_variable_names
```

Out[5]:
```
['Age',
 'DailyRate',
 'DistanceFromHome',
 'Education',
 'EmployeeCount',
 'EmployeeNumber',
 'EnvironmentSatisfaction',
 'HourlyRate',
 'JobInvolvement',
 'JobLevel',
 'JobSatisfaction',
 'MonthlyIncome',
 'MonthlyRate',
 'NumCompaniesWorked',
 'PercentSalaryHike',
 'PerformanceRating',
 'RelationshipSatisfaction',
 'StandardHours',
 'StockOptionLevel',
 'TotalWorkingYears',
 'TrainingTimesLastYear',
 'WorkLifeBalance',
 'YearsAtCompany',
 'YearsInCurrentRole',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager']
```

In [6]:
```
categorical_varibale_name
```

Out[6]:
```
['Attrition',
 'BusinessTravel',
 'Department',
 'EducationField',
 'Gender',
 'JobRole',
 'MaritalStatus',
 'Over18',
 'OverTime']
```

In [7]:
```
#store the numerical variables data in seperate dataset
df_numeric=df_new[numeric_variable_names]
```

In [8]:
```
#store the categorical variables data in seperate dataset
df_categorical=df_new[categorical_varibale_name]
```

In [ ]:
```
#dropping the attrition columns
```

In [9]:
```
df_categorical.drop(['Attrition'],axis=1,inplace=True)
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_18784\4285812731.py:1: SettingWithCopyWar
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
  df_categorical.drop(['Attrition'],axis=1,inplace=True)
```

In [11]:
```
df_categorical.shape
```

Out[11]: (1470, 8)

In [12]:
```
#converting into dummy variables

df_categorical=pd.get_dummies(df_categorical)
```

In [13]:
```
#Merging the both numerical and categorical data
df_final_data=pd.concat([df_numeric,df_categorical,df_new[["Attrition"]]],axis=1
```

In [18]:
```
df_final_data.head(5)
```

Out[18]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	Environm
0	41	1102	1	2	1	1	
1	49	279	8	1	1	2	
2	37	1373	2	2	1	4	
3	33	1392	3	4	1	5	
4	27	591	2	1	1	7	

5 rows × 56 columns

In [19]:
```
#final features
features =  list(df_final_data.columns.difference(["Attrition"]))
```

In [20]:
```
features
```

Out[20]:
```
['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',
 'EmployeeCount',
 'EmployeeNumber',
 'EnvironmentSatisfaction',
 'Gender_Female',
 'Gender_Male',
 'HourlyRate',
 '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',
 'Over18_Y',
 'OverTime_No',
 'OverTime_Yes',
 'PercentSalaryHike',
 'PerformanceRating',
 'RelationshipSatisfaction',
 'StandardHours',
 'StockOptionLevel',
 'TotalWorkingYears',
 'TrainingTimesLastYear',
 'WorkLifeBalance',
 'YearsAtCompany',
 'YearsInCurrentRole',
 'YearsSinceLastPromotion',
 'YearsWithCurrManager']
```

Separating the Target and the Predictors

In [24]:

```
X=df_final_data[features]
y=df_final_data[["Attrition"]]
```

In [25]:
```
X.shape
```

Out[25]:
```
(1470, 55)
```

In [23]:
```
y.shape
```

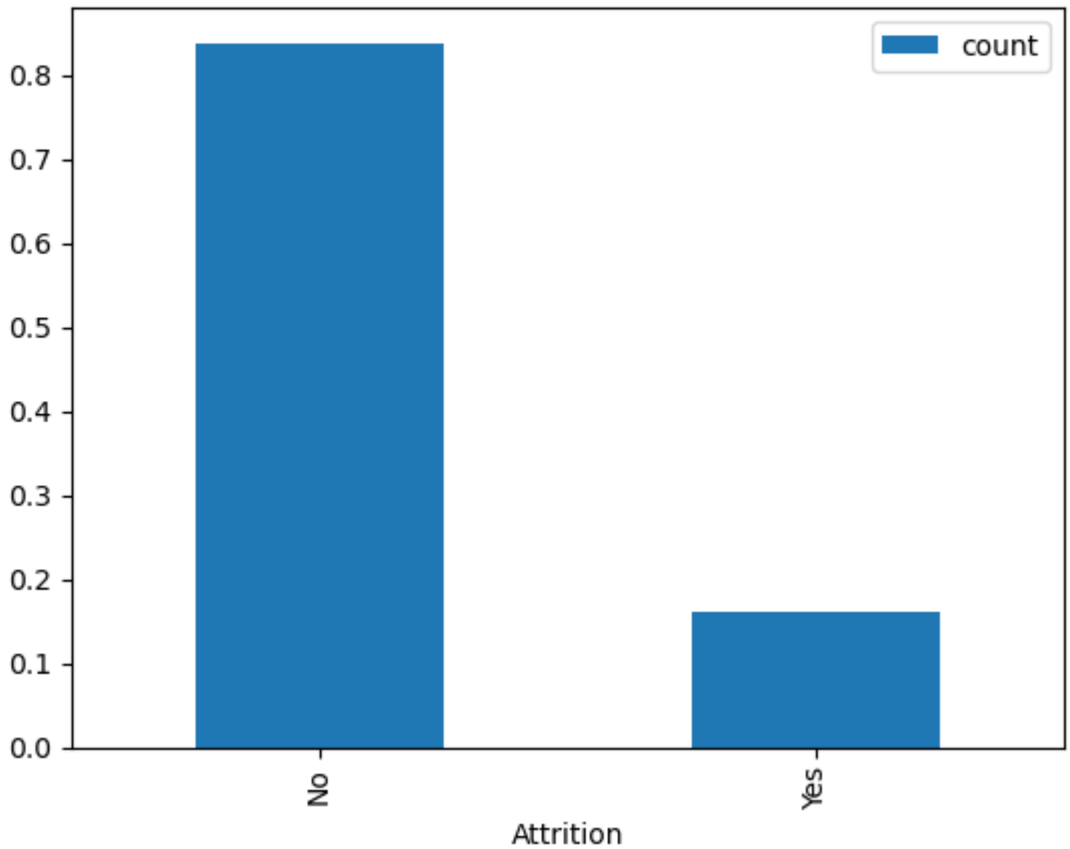
Out[23]:
```
(1470, 1)
```

In [26]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test = train_test_split(X,y,test_size = 0.3,stratify
```

In [27]:
```
#Checks
#Proportion in training data
y_train.Attrition.value_counts()/len(y_train)
```

Out[27]:
```
Attrition
No     0.838678
Yes    0.161322
Name: count, dtype: float64
```

In [28]:
```
#Checks
#Proportion in training data
pd.DataFrame(y_train.Attrition.value_counts()/len(y_train)).plot(kind = "bar")
plt.show()
```



In [29]:
```
#Proportion of test data
y_test.Attrition.value_counts()/len(y_test)
```

Out[29]:
```
Attrition
No     0.839002
Yes    0.160998
Name: count, dtype: float64
```

In [31]:
```
# 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
#make a pipeline for decision tree model
from sklearn.tree import DecisionTreeClassifier
pipelines = {
    "clf": make_pipeline(DecisionTreeClassifier(max_depth=3,random_state=100))
}
```

In [32]:
```
#Cross Validate
#To check the accuracy of the pipeline
scores = cross_validate(pipelines['clf'], X_train, y_train,return_train_score=Tr
```

In [33]:
```
scores['test_score'].mean()
```

Out[33]:
```
0.8396590101823348
```

In [ ]:
```
#Average accuracy of pipeline with Decision Tree Classifier is 83.48%
```