In [1]:

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df=pd.read_csv("ibm-hr-analytics-employee-attrition-performance.zip")
df.head()
```

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...

5 rows × 35 columns

In [3]:

```
df.shape
```

Out[3]:

```
(1470, 35)
```

In [4]:

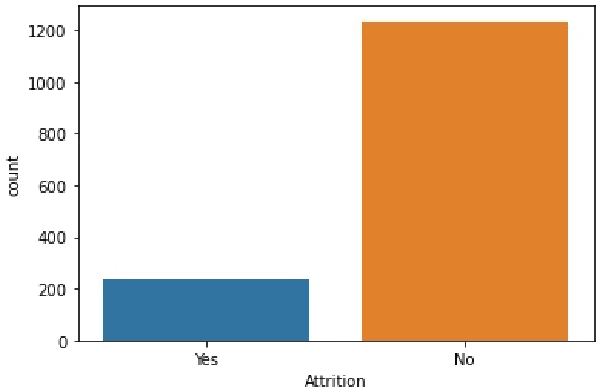
```
df.info()
```

```
<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
```

we have to encode all the object into numerical value

In [5]:

```
import seaborn as sns
import matplotlib.pyplot as plt
# check distribution for target variable
sns.countplot(x = 'Attrition', data = df);
plt.savefig('attrition.png')
```



Data is imbalance so we have to balance it later

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
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
```

there is no null values we can continue

In [7]:

```
df.describe()
```

Out[7]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	Jol
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	

mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000

8 rows × 26 columns

In [8]:

```
from sklearn.preprocessing import OrdinalEncoder
enc=OrdinalEncoder()
```

In [9]:

```
for i in df.columns:
    if df[i].dtypes=="object":
        df[i]=enc.fit_transform(df[i].values.reshape(-1,1))
```

we have converted object into numerical value

In [10]:

```
df.head()
```

Out[10]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...
0	41	1.0	2.0	1102	2.0	1	2	1.0	1	1	...
1	49	0.0	1.0	279	1.0	8	1	1.0	1	2	...
2	37	1.0	2.0	1373	1.0	2	2	4.0	1	4	...
3	33	0.0	1.0	1392	1.0	3	4	1.0	1	5	...
4	27	0.0	2.0	591	1.0	2	1	3.0	1	7	...

5 rows × 35 columns

we have to see correlation

In [11]:

```
df.corr()
```

Out[11]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Employe
Age	1.000000	-0.159205	0.024751	0.010661	-0.031882	-0.001686	0.208034	-0.040873	
Attrition	-0.159205	1.000000	0.000074	-0.056652	0.063991	0.077924	-0.031373	0.026846	
BusinessTravel	0.024751	0.000074	1.000000	-0.004086	-0.009044	-0.024469	0.000757	0.023724	
DailyRate	0.010661	-0.056652	-0.004086	1.000000	0.007109	-0.004985	-0.016806	0.037709	
Department	-0.031882	0.063991	-0.009044	0.007109	1.000000	0.017225	0.007996	0.013720	
DistanceFromHome	-0.001686	0.077924	-0.024469	-0.004985	0.017225	1.000000	0.021042	0.002013	
Education	0.208034	-0.031373	0.000757	-0.016806	0.007996	0.021042	1.000000	-0.039592	
EducationField	-0.040873	0.026846	0.023724	0.037709	0.013720	0.002013	-0.039592	1.000000	
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.010577	-0.015578	-0.050990	-0.010895	0.032916	0.042070	-0.002516	
EnvironmentSatisfaction	0.010146	-0.103369	0.004174	0.018355	-0.019395	-0.016075	-0.027128	0.043163	
Gender	-0.036311	0.029453	-0.032981	-0.011716	-0.041583	-0.001851	-0.016547	-0.002504	
HourlyRate	0.024287	-0.006846	0.026528	0.023381	-0.004144	0.031131	0.016775	-0.021941	
JobInvolvement	0.029820	-0.130016	0.039062	0.046135	-0.024586	0.008783	0.042438	-0.002655	
JobLevel	0.509604	-0.169105	0.019311	0.002966	0.101963	0.005303	0.101589	-0.044933	
JobRole	-0.122427	0.067151	0.002724	-0.009472	0.662431	-0.001015	0.004236	0.015599	
JobSatisfaction	-0.004892	-0.103481	-0.033962	0.030571	0.021001	-0.003669	-0.011296	-0.034401	
MaritalStatus	-0.095029	0.162070	0.024001	-0.069586	0.056073	-0.014437	0.004053	0.014420	
MonthlyIncome	0.497855	-0.159840	0.034319	0.007707	0.053130	-0.017014	0.094961	-0.041070	
MonthlyRate	0.028051	0.015170	-0.014107	-0.032182	0.023642	0.027473	-0.026084	-0.027182	

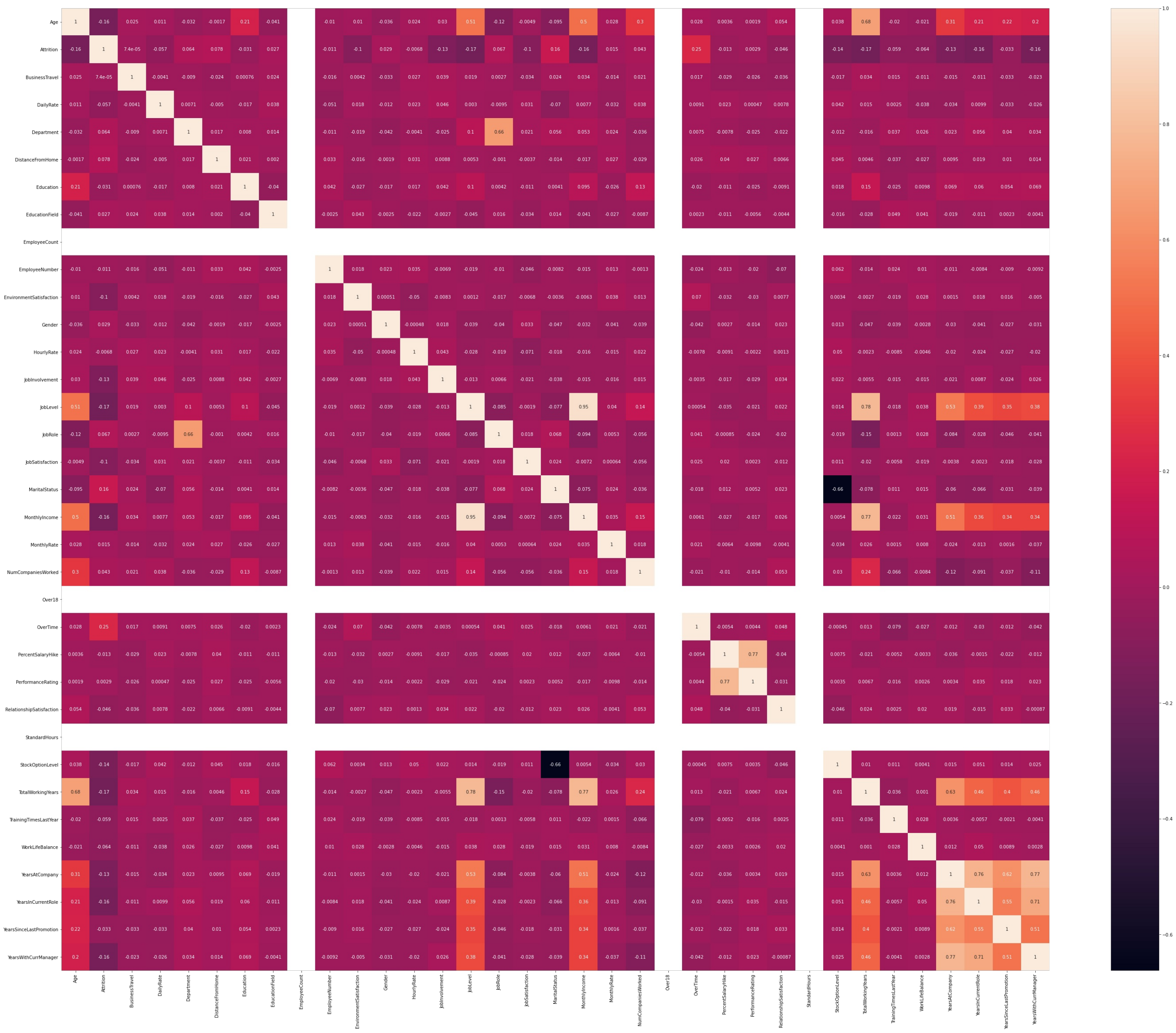
NumCompaniesWorked	0.299635	0.043494	0.020875	0.038153	-0.035882	-0.029251	0.126317	-0.008663
Over18	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
OverTime	0.028062	0.246118	0.016543	0.009135	0.007481	0.025514	-0.020322	0.002259
PercentSalaryHike	0.003634	-0.013478	-0.029377	0.022704	-0.007840	0.040235	-0.011111	-0.011214
PerformanceRating	0.001904	0.002889	-0.026341	0.000473	-0.024604	0.027110	-0.024539	-0.005614
RelationshipSatisfaction	0.053535	-0.045872	-0.035986	0.007846	-0.022414	0.006557	-0.009118	-0.004378
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
StockOptionLevel	0.037510	-0.137145	-0.016727	0.042143	-0.012193	0.044872	0.018422	-0.016185
TotalWorkingYears	0.680381	-0.171063	0.034226	0.014515	-0.015762	0.004628	0.148280	-0.027848
TrainingTimesLastYear	-0.019621	-0.059478	0.015240	0.002453	0.036875	-0.036942	-0.025100	0.049195
WorkLifeBalance	-0.021490	-0.063939	-0.011256	-0.037848	0.026383	-0.026556	0.009819	0.041191
YearsAtCompany	0.311309	-0.134392	-0.014575	-0.034055	0.022920	0.009508	0.069114	-0.018692
YearsInCurrentRole	0.212901	-0.160545	-0.011497	0.009932	0.056315	0.018845	0.060236	-0.010506
YearsSinceLastPromotion	0.216513	-0.033019	-0.032591	-0.033229	0.040061	0.010029	0.054254	0.002326
YearsWithCurrManager	0.202089	-0.156199	-0.022636	-0.026363	0.034282	0.014406	0.069065	-0.004130

35 rows × 35 columns

In [12]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize=[50,40])
sns.heatmap(data=df.corr(),annot=True)
```

Out[12]: <AxesSubplot:>



In [13]:

```
df.corr()["Attrition"].sort_values(ascending=False)
```

Out[13]:

```
Attrition                   1.000000
OverTime                    0.246118
MaritalStatus               0.162070
DistanceFromHome            0.077924
JobRole                     0.067151
Department                  0.063991
NumCompaniesWorked          0.043494
Gender                      0.029453
EducationField              0.026846
MonthlyRate                 0.015170
PerformanceRating           0.002889
BusinessTravel              0.000074
HourlyRate                 -0.006846
EmployeeNumber             -0.010577
PercentSalaryHike          -0.013478
Education                  -0.031373
YearsSinceLastPromotion    -0.033019
RelationshipSatisfaction   -0.045872
DailyRate                  -0.056652
TrainingTimesLastYear      -0.059478
WorkLifeBalance            -0.063939
EnvironmentSatisfaction    -0.103369
JobSatisfaction            -0.103481
JobInvolvement             -0.130016
YearsAtCompany             -0.134392
StockOptionLevel           -0.137145
YearsWithCurrManager       -0.156199
Age                        -0.159205
MonthlyIncome              -0.159840
YearsInCurrentRole         -0.160545
JobLevel                   -0.169105
TotalWorkingYears          -0.171063
EmployeeCount                    NaN
Over18                           NaN
StandardHours                    NaN
Name: Attrition, dtype: float64
```

We can see from overtime to buisness travel columns are positively correlated and rest are negatively correlated.we have to drop columns which not correlated.

In [14]:

```
df.drop(["StandardHours","Over18","EmployeeCount","EmployeeNumber"],axis=1,inplace=True)
```

In [15]:

```
df.head()
```

Out[15]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...	F
0	41	1.0	2.0	1102	2.0	1	2	1.0	2	0.0	...	
1	49	0.0	1.0	279	1.0	8	1	1.0	3	1.0	...	
2	37	1.0	2.0	1373	1.0	2	2	4.0	4	1.0	...	
3	33	0.0	1.0	1392	1.0	3	4	1.0	4	0.0	...	
4	27	0.0	2.0	591	1.0	2	1	3.0	1	1.0	...	

5 rows × 31 columns

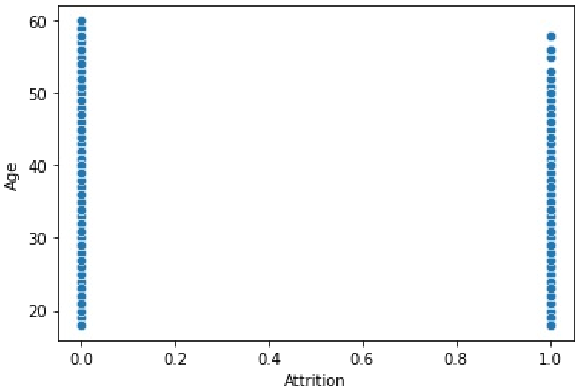
In [16]:

```
col=df.columns
print(col)
for i in col:
    print(i)
    plt.figure()
    sns.scatterplot(data=df,x='Attrition',y=i)
    plt.show()
```

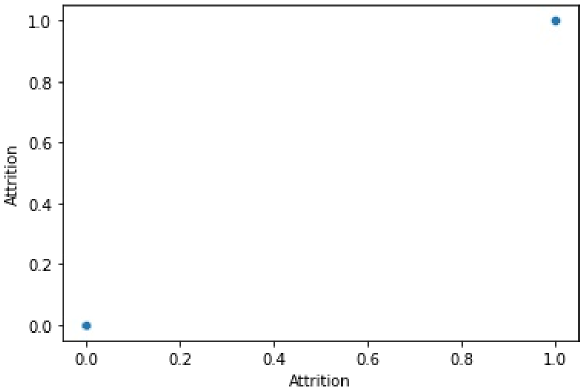
```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus',
       'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'OverTime',
       'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
```

```
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager'],
      dtype='object')
```

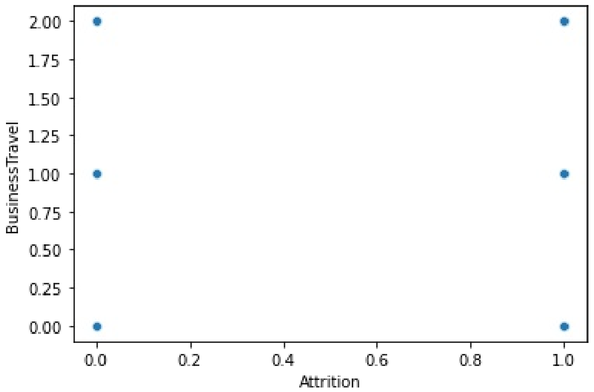
Age



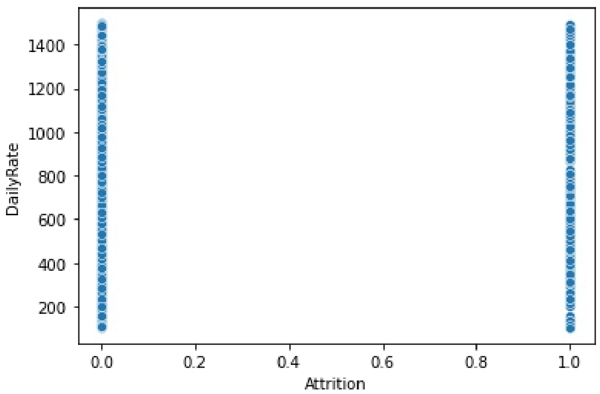
Attrition



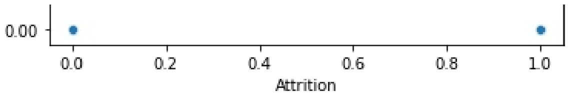
BusinessTravel



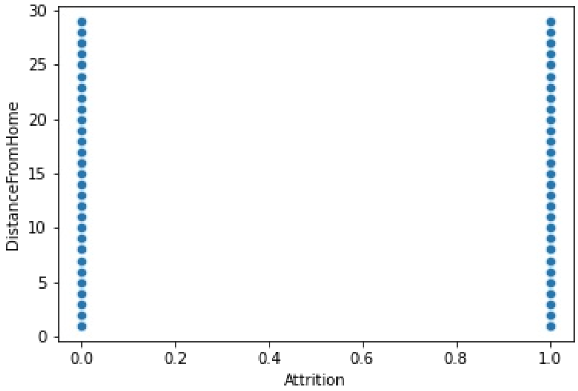
DailyRate



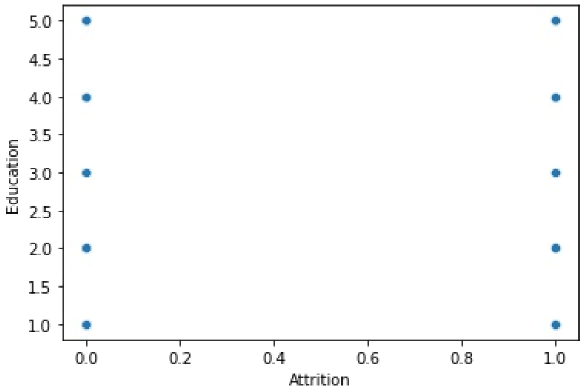
Department



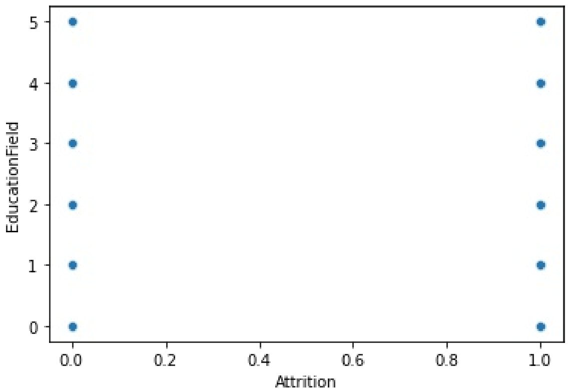
DistanceFromHome



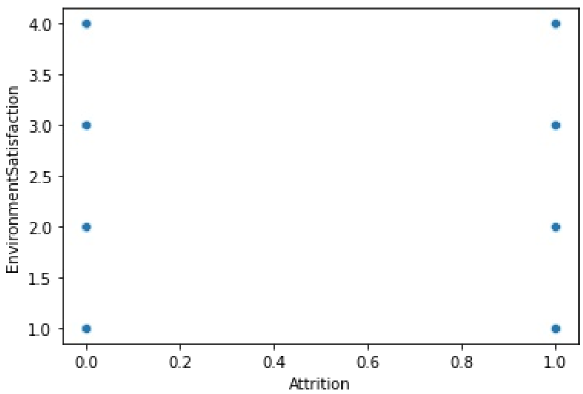
Education



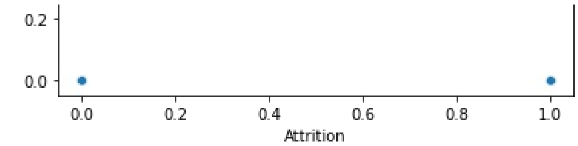
EducationField



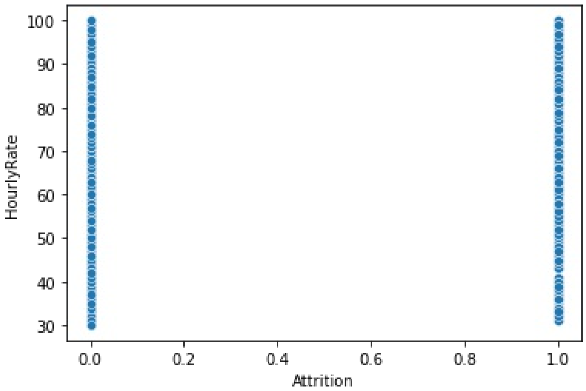
EnvironmentSatisfaction



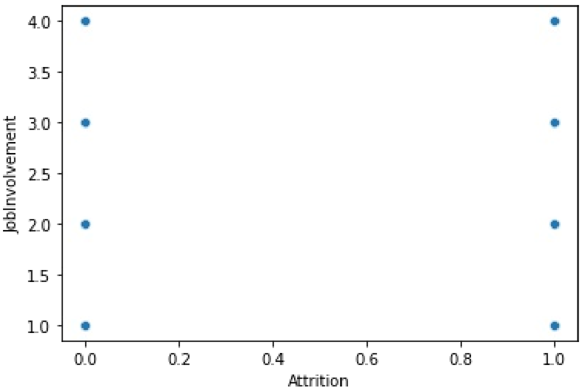
Gender



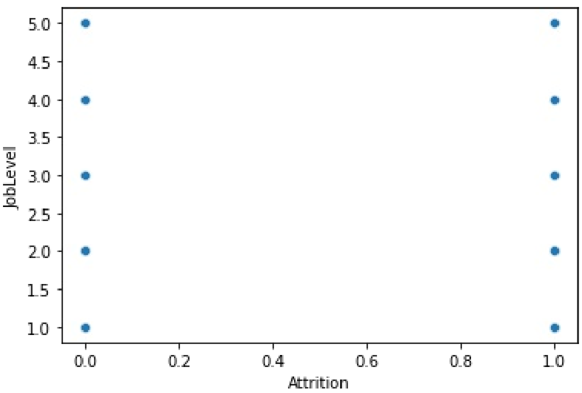
HourlyRate



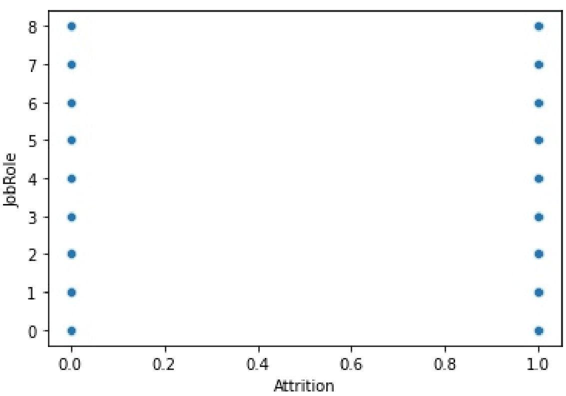
JobInvolvement



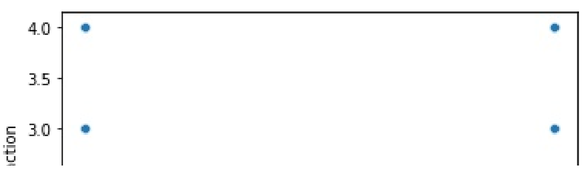
JobLevel

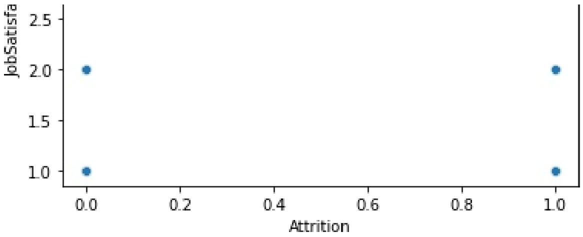


JobRole

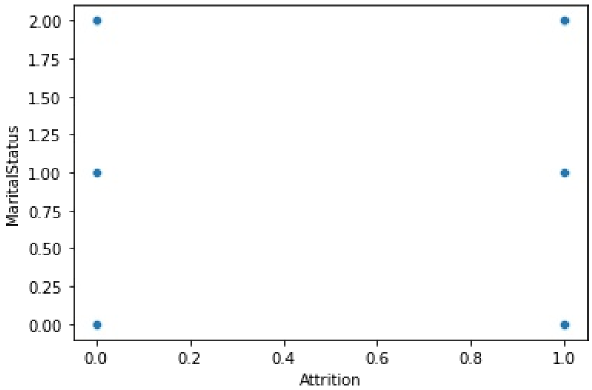


JobSatisfaction

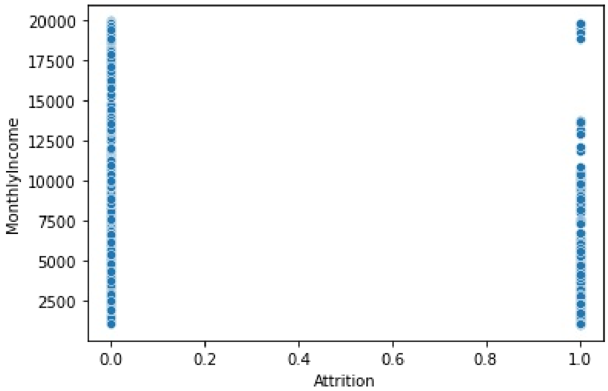




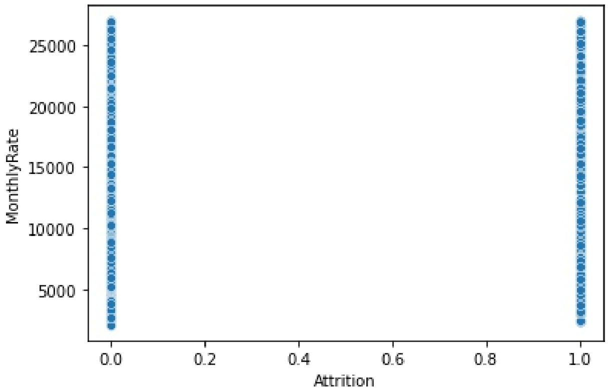
MaritalStatus



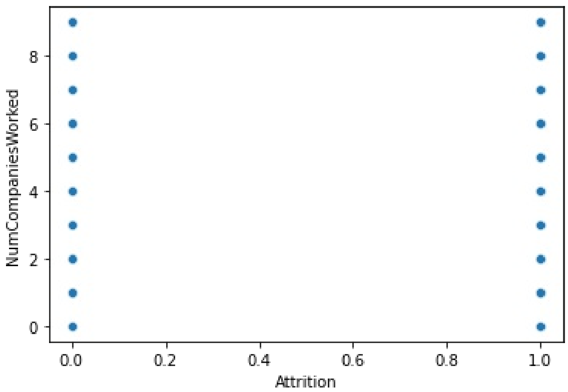
MonthlyIncome



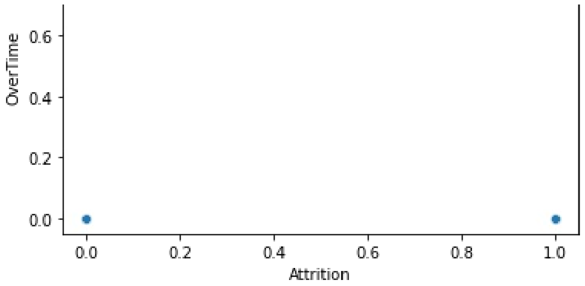
MonthlyRate



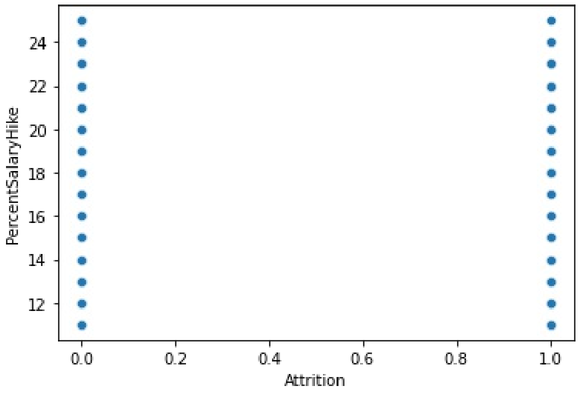
NumCompaniesWorked



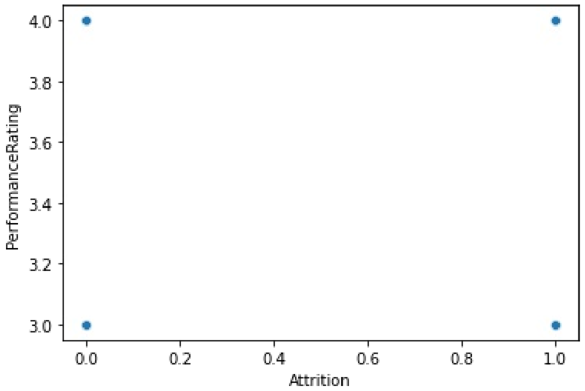
OverTime



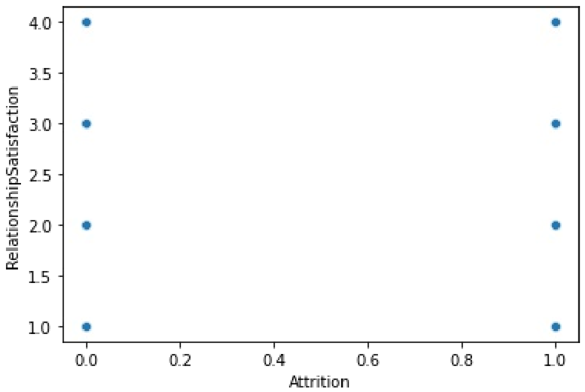
PercentSalaryHike



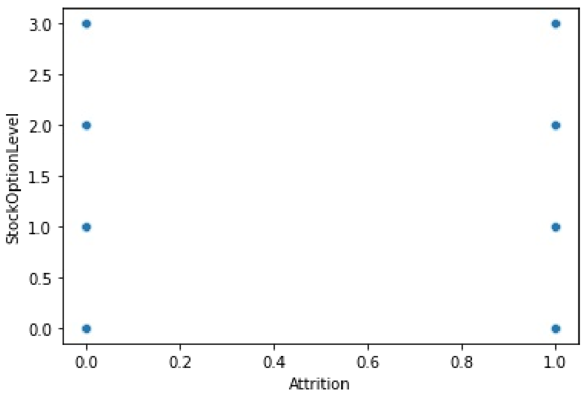
PerformanceRating



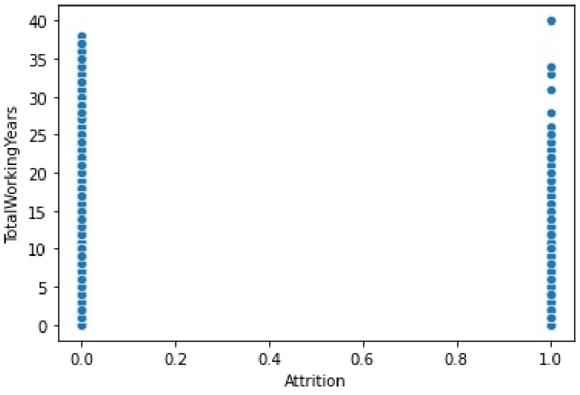
RelationshipSatisfaction



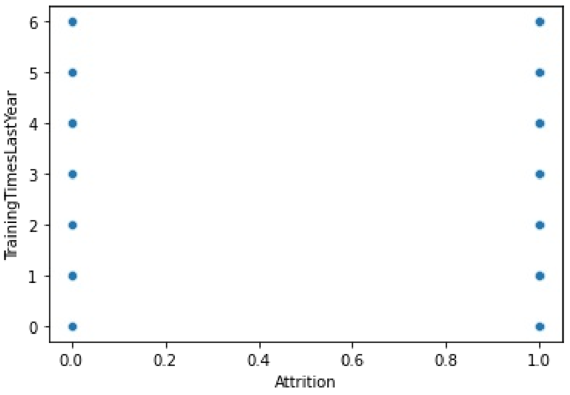
StockOptionLevel



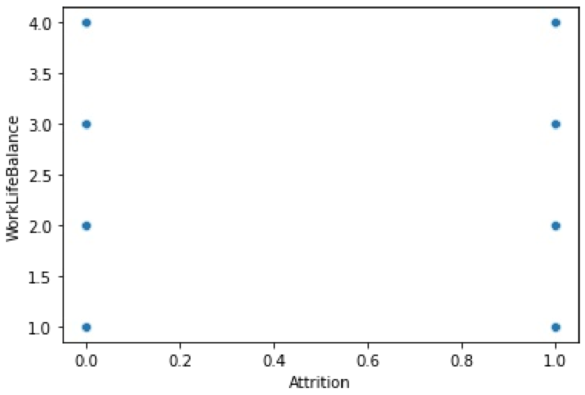
TotalWorkingYears



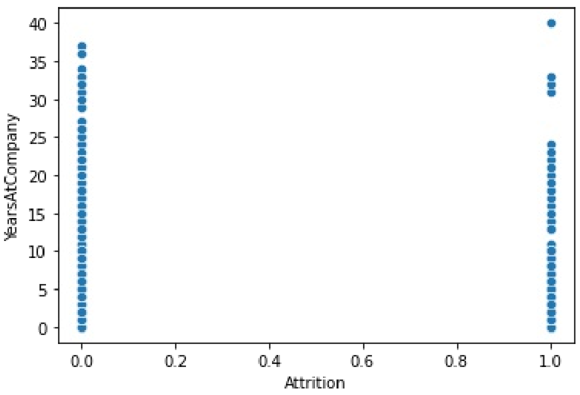
TrainingTimesLastYear



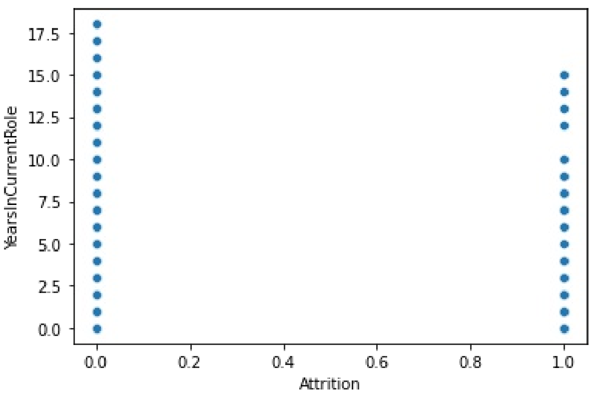
WorkLifeBalance



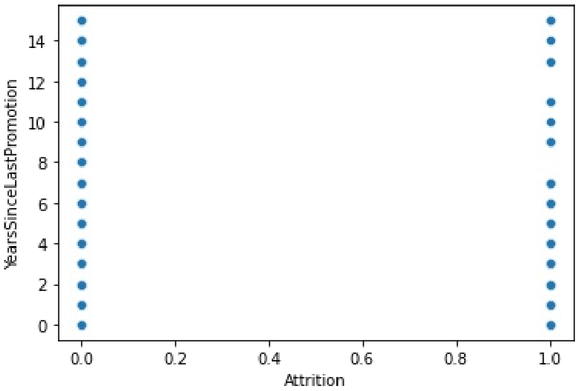
YearsAtCompany



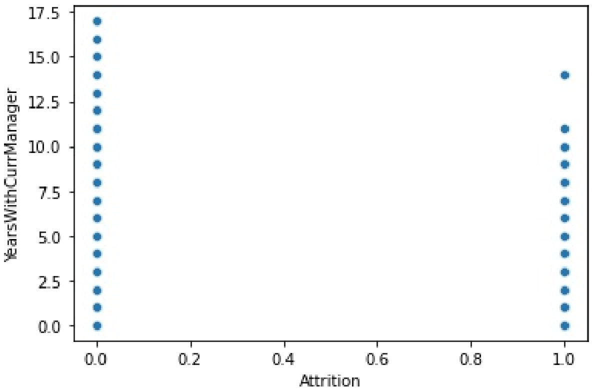
YearsInCurrentRole



YearsSinceLastPromotion



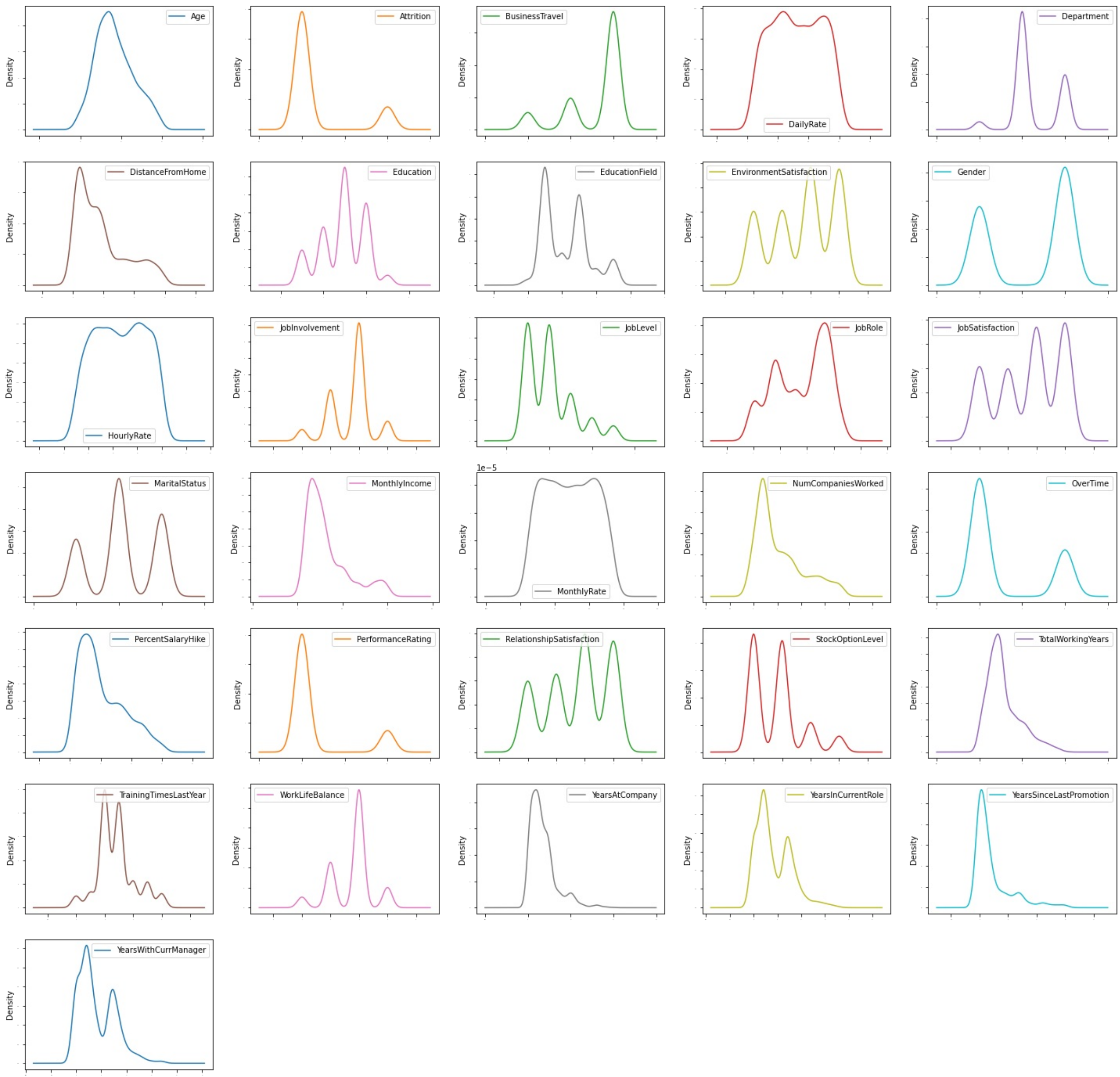
YearsWithCurrManager



we can see through graps they have some correlation

In [17]:
```
df.plot(kind="density",subplots=True,layout=(7,5),sharex=False,fontsize=1,figsize=(25,25))
plt.show
```

Out[17]: <function matplotlib.pyplot.show(close=None, block=None)>



Ther are not distributed normally.

In [18]:
```
df.shape
```

Out[18]: (1470, 31)

In [19]:
```
df.skew()
```

```
Out[19]: Age                         0.413286
Attrition                   1.844366
BusinessTravel             -1.439006
DailyRate                  -0.003519
Department                  0.172231
DistanceFromHome            0.958118
Education                  -0.289681
EducationField              0.550371
EnvironmentSatisfaction    -0.321654
Gender                     -0.408665
HourlyRate                 -0.032311
JobInvolvement             -0.498419
JobLevel                    1.025401
JobRole                    -0.357270
JobSatisfaction            -0.329672
MaritalStatus              -0.152175
MonthlyIncome               1.369817
MonthlyRate                 0.018578
NumCompaniesWorked          1.026471
OverTime                    0.964489
PercentSalaryHike           0.821128
PerformanceRating           1.921883
RelationshipSatisfaction   -0.302828
StockOptionLevel            0.968980
TotalWorkingYears           1.117172
TrainingTimesLastYear       0.553124
WorkLifeBalance            -0.552480
YearsAtCompany              1.764529
YearsInCurrentRole          0.917363
YearsSinceLastPromotion     1.984290
YearsWithCurrManager        0.833451
dtype: float64
```

we have remove skewness of this columns

```
In [20]: df.head()
```

Out[20]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...	F
0	41	1.0	2.0	1102	2.0	1	2	1.0	2	0.0	...	
1	49	0.0	1.0	279	1.0	8	1	1.0	3	1.0	...	
2	37	1.0	2.0	1373	1.0	2	2	4.0	4	1.0	...	
3	33	0.0	1.0	1392	1.0	3	4	1.0	4	0.0	...	
4	27	0.0	2.0	591	1.0	2	1	3.0	1	1.0	...	

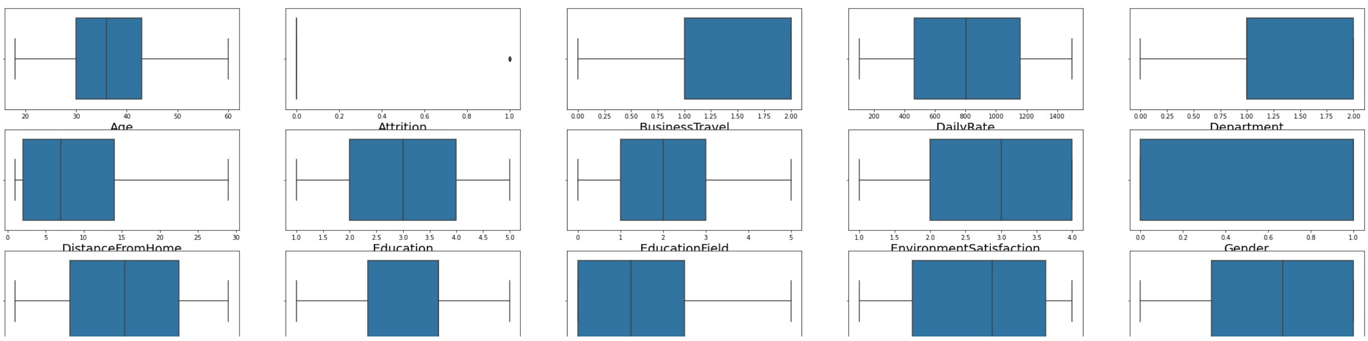
5 rows × 31 columns

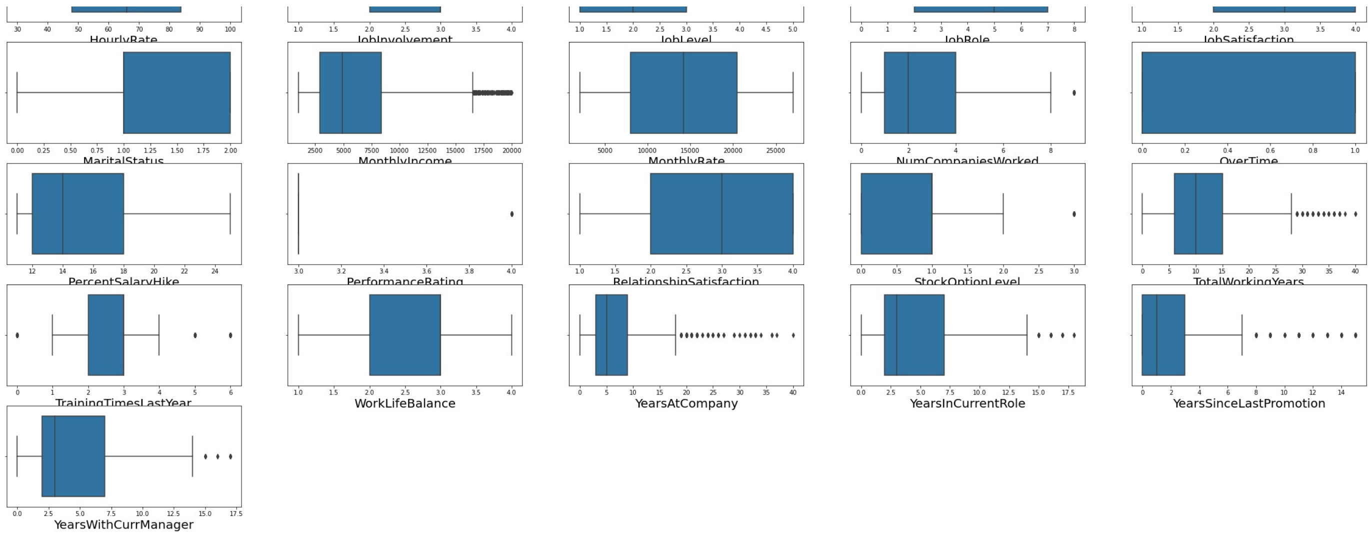
```
In [21]: #plotting box plot for my dataframe to check and remove outliers
plt.figure(figsize=(40,25),facecolor="white")
plotnumber = 1

for column in df:

    if(df[column].dtype == np.float64 or df[column].dtype == np.int64):
        if plotnumber<=34:
            ax = plt.subplot(7,5,plotnumber)
            sns.boxplot(df[column])
            plt.xlabel(column,fontsize=20)

    plotnumber +=1
plt.show()
```





now we have to remove outlier with zscore method

In [22]:

```
from scipy.stats import zscore
import numpy as np
z=np.abs(zscore(df))
threshold=3
np.where(z>3)
```

Out[22]:

```
(array([  28,   45,   62,   62,   63,   64,   85,   98,   98,  110,  123,
         123,  123,  126,  126,  126,  153,  178,  187,  187,  190,  190,
         218,  231,  231,  237,  237,  270,  270,  281,  326,  386,  386,
         401,  411,  425,  425,  427,  445,  466,  473,  477,  535,  561,
         561,  584,  592,  595,  595,  595,  616,  624,  635,  653,  653,
         677,  686,  701,  716,  746,  749,  752,  799,  838,  861,  861,
         875,  875,  894,  914,  914,  918,  922,  926,  926,  937,  956,
         962,  976,  976, 1008, 1024, 1043, 1078, 1078, 1086, 1086, 1093,
        1111, 1116, 1116, 1135, 1138, 1138, 1156, 1184, 1221, 1223, 1242,
        1295, 1301, 1301, 1303, 1327, 1331, 1348, 1351, 1401, 1414, 1430],
       dtype=int64),
 array([30, 29, 27, 29, 28, 29, 24, 24, 27, 29, 28, 29, 30, 24, 27, 29, 30,
        29, 24, 30, 27, 28, 29, 28, 30, 27, 29, 24, 27, 28, 29, 29, 30, 24,
        27, 27, 29, 29, 24, 28, 27, 27, 29, 27, 30, 29, 27, 24, 27, 29, 30,
        24, 30, 27, 29, 27, 30, 29, 28, 28, 27, 29, 29, 29, 27, 29, 29, 30,
        24, 27, 29, 27, 29, 29, 30, 29, 24, 27, 28, 29, 29, 28, 24, 29, 30,
        27, 29, 29, 27, 24, 27, 27, 27, 29, 29, 24, 29, 29, 29, 29, 24, 29,
        29, 28, 29, 30, 28, 24, 29, 28], dtype=int64))
```

In [23]:

```
df_new=df[(z<3).all(axis=1)]
df_new
```

Out[23]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...
0	41	1.0	2.0	1102	2.0	1	2	1.0	2	0.0	...
1	49	0.0	1.0	279	1.0	8	1	1.0	3	1.0	...
2	37	1.0	2.0	1373	1.0	2	2	4.0	4	1.0	...
3	33	0.0	1.0	1392	1.0	3	4	1.0	4	0.0	...
4	27	0.0	2.0	591	1.0	2	1	3.0	1	1.0	...
...	...	...	...	...	...	...	...	...	...	...	...
1465	36	0.0	1.0	884	1.0	23	2	3.0	3	1.0	...
1466	39	0.0	2.0	613	1.0	6	1	3.0	4	1.0	...
1467	27	0.0	2.0	155	1.0	4	3	1.0	2	1.0	...
1468	49	0.0	1.0	1023	2.0	2	3	3.0	4	1.0	...
1469	34	0.0	2.0	628	1.0	8	3	3.0	2	1.0	...

1387 rows × 31 columns

In [24]:

```
df_new.shape
```

Out[24]: (1387, 31)

In [25]:
```
data_loss=((1470-1387)/1470)*100
```

In [26]:
```
data_loss
```

Out[26]: 5.646258503401361

We have removed outlier and our data loss is also less se we can continue with this data.

In [27]:
```
x=df_new.drop("Attrition",axis=1)
y=df_new["Attrition"]
```

In [28]:
```
from sklearn.preprocessing import StandardScaler
sc =StandardScaler()
x_scaled=sc.fit_transform(x)
x=pd.DataFrame(x_scaled, columns=x.columns)
```

In [29]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif"]=[variance_inflation_factor(x_scaled,i)for i in range(x_scaled.shape[1])]
vif["features"]=x.columns
```

In [30]:
```
vif
```

Out[30]:

	vif	features
0	1.850568	Age
1	1.014106	BusinessTravel
2	1.029494	DailyRate
3	2.062034	Department
4	1.019457	DistanceFromHome
5	1.061558	Education
6	1.021559	EducationField
7	1.021370	EnvironmentSatisfaction
8	1.023993	Gender
9	1.022001	HourlyRate
10	1.018417	JobInvolvement
11	10.188826	JobLevel
12	1.978117	JobRole
13	1.022468	JobSatisfaction
14	1.838550	MaritalStatus
15	9.376741	MonthlyIncome
16	1.017026	MonthlyRate
17	1.259499	NumCompaniesWorked
18	1.028209	OverTime
19	2.479609	PercentSalaryHike
20	2.473394	PerformanceRating
21	1.020205	RelationshipSatisfaction
22	1.821502	StockOptionLevel
23	3.887996	TotalWorkingYears
24	1.026639	TrainingTimesLastYear
25	1.018066	WorkLifeBalance
26	4.883652	YearsAtCompany
27	2.984247	YearsInCurrentRole
28	1.436720	YearsSinceLastPromotion
29	3.323766	YearsWithCurrManager

As we see JobLevel and MonthlyIncome have high vif values so we have to drop that columns.

In [31]:
```
x.drop(["JobLevel","MonthlyIncome"],axis=1,inplace=True)
```

In [32]:
```
x.head()
```

Out[32]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyF
0	0.536681	0.593126	0.734325	1.405373	-1.011249	-0.876177	-0.940815	-0.665328	-1.229911	1.388
1	1.442111	-0.905354	-1.307769	-0.496337	-0.145521	-1.853858	-0.940815	0.251978	0.813067	-0.239
2	0.083966	0.593126	1.406752	-0.496337	-0.887573	-0.876177	1.305159	1.169285	0.813067	1.290
3	-0.368749	-0.905354	1.453896	-0.496337	-0.763898	1.079185	-0.940815	1.169285	-1.229911	-0.485
4	-1.047821	0.593126	-0.533609	-0.496337	-0.887573	-1.853858	0.556501	-1.582635	0.813067	-1.274

5 rows × 28 columns

In [33]:
```
x.shape
```

Out[33]: (1387, 28)

In [34]:
```
y.shape
```

Out[34]: (1387,)

In [35]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [36]:
```
!pip install imblearn --user
```

```
Requirement already satisfied: imblearn in c:\users\administrator\appdata\roaming\python\python38\site-packages (
0.0)
Requirement already satisfied: imbalanced-learn in c:\users\administrator\appdata\roaming\python\python38\site-pa
ckages (from imblearn) (0.9.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from imbalance
d-learn->imblearn) (2.1.0)
Requirement already satisfied: scikit-learn>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from imbalanced
-learn->imblearn) (1.0.2)
Requirement already satisfied: joblib>=0.11 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn-
>imblearn) (1.0.1)
Requirement already satisfied: scipy>=1.1.0 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn-
>imblearn) (1.6.2)
Requirement already satisfied: numpy>=1.14.6 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn
->imblearn) (1.20.1)
```

we have to balanced dataset for our label

In [37]:
```
from collections import Counter
from imblearn.over_sampling import RandomOverSampler
os=RandomOverSampler(0.75)
x_train_ns,y_train_ns=os.fit_resample(x_train,y_train)
print("The number of classes before fit {}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_ns)))
```

```
The number of classes before fit Counter({0.0: 922, 1.0: 187})
The number of classes after fit Counter({0.0: 922, 1.0: 691})
```

In [38]:
```
y_train_ns.value_counts()
```

Out[38]:
```
0.0    922
1.0    691
Name: Attrition, dtype: int64
```

Now we have to input models to get accuracy.

In [40]:
```
from sklearn.metrics import accuracy_score
```

In [41]:
```
#logisticregression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report,confusion_matrix,plot_confusion_matrix,roc_curve, roc_auc_score
lr=LogisticRegression()
lr.fit(x_train_ns,y_train_ns)
y_pred = lr.predict(x_test)
print("Accuracy",accuracy_score(y_test, y_pred)*100)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy 82.01438848920863
[[197  39]
 [ 11  31]]
              precision    recall  f1-score   support

         0.0       0.95      0.83      0.89       236
         1.0       0.44      0.74      0.55        42

    accuracy                           0.82       278
   macro avg       0.69      0.79      0.72       278
weighted avg       0.87      0.82      0.84       278
```

In [42]:
```
#RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
rfc=RandomForestClassifier()
rfc.fit(x_train_ns,y_train_ns)
y_pred = rfc.predict(x_test)
print("Accuracy ",accuracy_score(y_test, y_pred)*100)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy  86.33093525179856
[[229   7]
 [ 31  11]]
              precision    recall  f1-score   support

         0.0       0.88      0.97      0.92       236
         1.0       0.61      0.26      0.37        42

    accuracy                           0.86       278
   macro avg       0.75      0.62      0.65       278
weighted avg       0.84      0.86      0.84       278
```

In [43]:
```
#KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train_ns,y_train_ns)
y_pred = knn.predict(x_test)
print("Accuracy ",accuracy_score(y_test, y_pred)*100)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy  73.38129496402878
[[184  52]
 [ 22  20]]
              precision    recall  f1-score   support

         0.0       0.89      0.78      0.83       236
         1.0       0.28      0.48      0.35        42
```

```
    accuracy                           0.73       278
   macro avg       0.59      0.63      0.59       278
weighted avg       0.80      0.73      0.76       278
```

In [44]:

```
#DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(x_train_ns,y_train_ns)
y_pred = dt.predict(x_test)
print("Accuracy ",accuracy_score(y_test, y_pred)*100)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy  79.13669064748201
[[209  27]
 [ 31  11]]
              precision    recall  f1-score   support

         0.0       0.87      0.89      0.88       236
         1.0       0.29      0.26      0.28        42

    accuracy                           0.79       278
   macro avg       0.58      0.57      0.58       278
weighted avg       0.78      0.79      0.79       278
```

In [45]:

```
#GradientBoostingClassifier
from sklearn.ensemble import GradientBoostingClassifier
gb=GradientBoostingClassifier()
gb.fit(x_train_ns,y_train_ns)
y_pred = gb.predict(x_test)
print("Accuracy ",accuracy_score(y_test, y_pred)*100)
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy  83.45323741007195
[[209  27]
 [ 19  23]]
              precision    recall  f1-score   support

         0.0       0.92      0.89      0.90       236
         1.0       0.46      0.55      0.50        42

    accuracy                           0.83       278
   macro avg       0.69      0.72      0.70       278
weighted avg       0.85      0.83      0.84       278
```

In [46]:

```
from sklearn.model_selection import cross_val_score
scr=cross_val_score(lr, x, y, cv=5)
print("Cross validation score of Logistic Regression model :",scr.mean())

scr=cross_val_score(rfc, x, y, cv=5)
print("Cross validation score of Random Forest model :",scr.mean())

scr=cross_val_score(knn, x, y, cv=5)
print ("Cross validation score of knn model :",scr.mean())

scr=cross_val_score(gb, x, y, cv=5)
print ("Cross validation score of gb model :",scr.mean())
```

```
Cross validation score of Logistic Regression model : 0.870942523959172
Cross validation score of Random Forest model : 0.851479105524245
Cross validation score of knn model : 0.8471547671610005
Cross validation score of gb model : 0.8601303794509519
```

We conclude that random forestregressor has better acuuracy now we have to do Hypertuning to get good accuracy.

In [47]:

```
from sklearn.model_selection import KFold
kfold = KFold(n_splits=10, random_state = 6,shuffle=True)
```

In [48]:
```
lr_params = {'penalty':('l1', 'l2'),
         'C':(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10)}
```

In [49]:
```
from sklearn.model_selection import GridSearchCV
```

In [50]:
```
grid_lr = GridSearchCV(lr, lr_params, cv= kfold)
grid_lr.fit(x_train, y_train)
```

Out[50]:
```
GridSearchCV(cv=KFold(n_splits=10, random_state=6, shuffle=True),
             estimator=LogisticRegression(),
             param_grid={'C': (0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10),
                         'penalty': ('l1', 'l2')})
```

In [51]:
```
lr_res = grid_lr.cv_results_
```

In [52]:
```
best_lr_model = grid_lr.best_estimator_
best_lr_model
```

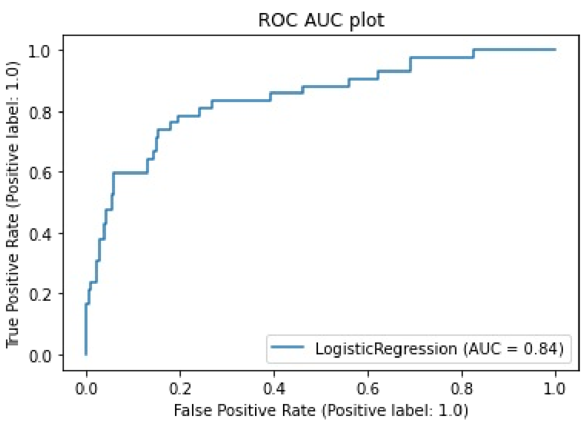
Out[52]:
```
LogisticRegression(C=5)
```

In [53]:
```
grid_lr.best_score_
```

Out[53]:
```
0.8602866502866503
```

In [54]:
```
from sklearn.metrics import plot_roc_curve
```

In [55]:
```
plot_roc_curve (grid_lr.best_estimator_,x_test,y_test)
plt.title("ROC AUC plot")
plt.show()
```



our accuracy is 83% through randomforestregressor

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js