In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [2]:
```
df=pd.read_csv('https://raw.githubusercontent.com/mrc03/IBM-HR-Analytics-Employee-Attrition-Performance/master/WA
```

In [3]:
```
df.head()
```

Out[3]:

	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 [4]:
```
df.shape
```

Out[4]:
```
(1470, 35)
```

In [5]:
```
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 [6]:
```
df.dtypes
```

Out[6]:
```
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
```

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]:
```
# Checking number of unique values in each columns
count = 1
for x in df:
    print(f'{count}. {x}: {df[x].nunique()}')
    print(f'{df[x].value_counts()}', end = '\n----------\n\n' )
    count += 1
```

```
1. Age: 43
35    78
34    77
36    69
31    69
29    68
32    61
30    60
38    58
33    58
```

```
40    57
37    50
27    48
28    48
42    46
39    42
45    41
41    40
26    39
44    33
46    33
43    32
50    30
24    26
25    26
47    24
49    24
55    22
53    19
48    19
51    19
52    18
54    18
22    16
56    14
23    14
58    14
21    13
20    11
59    10
19     9
18     8
60     5
57     4
Name: Age, dtype: int64
----------

2. Attrition: 2
No     1233
Yes     237
Name: Attrition, dtype: int64
----------

3. BusinessTravel: 3
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
----------

4. DailyRate: 886
691     6
1082    5
408     5
329     5
530     5
       ..
708     1
713     1
717     1
719     1
1499    1
Name: DailyRate, Length: 886, dtype: int64
----------

5. Department: 3
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
----------

6. DistanceFromHome: 29
2     211
1     208
10     86
9      85
3      84
7      84
8      80
5      65
4      64
6      59
```

```
16     32
11     29
24     28
23     27
29     27
18     26
15     26
20     25
25     25
26     25
28     23
19     22
14     21
17     20
12     20
22     19
13     19
21     18
27     12
Name: DistanceFromHome, dtype: int64
----------

7. Education: 5
3    572
4    398
2    282
1    170
5     48
Name: Education, dtype: int64
----------

8. EducationField: 6
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
----------

9. EmployeeCount: 1
1    1470
Name: EmployeeCount, dtype: int64
----------

10. EmployeeNumber: 1470
2048    1
1368    1
1364    1
1363    1
1362    1
       ..
648     1
647     1
645     1
644     1
2046    1
Name: EmployeeNumber, Length: 1470, dtype: int64
----------

11. EnvironmentSatisfaction: 4
3    453
4    446
2    287
1    284
Name: EnvironmentSatisfaction, dtype: int64
----------

12. Gender: 2
Male      882
Female    588
Name: Gender, dtype: int64
----------

13. HourlyRate: 71
66    29
42    28
98    28
84    28
48    28
      ..
69    15
```

```
53    14
68    14
38    13
34    12
Name: HourlyRate, Length: 71, dtype: int64
----------

14. JobInvolvement: 4
3    868
2    375
4    144
1     83
Name: JobInvolvement, dtype: int64
----------

15. JobLevel: 5
1    543
2    534
3    218
4    106
5     69
Name: JobLevel, dtype: int64
----------

16. JobRole: 9
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64
----------

17. JobSatisfaction: 4
4    459
3    442
1    289
2    280
Name: JobSatisfaction, dtype: int64
----------

18. MaritalStatus: 3
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
----------

19. MonthlyIncome: 1349
2342     4
6142     3
2610     3
2559     3
6347     3
        ..
4103     1
2705     1
6796     1
19717    1
10239    1
Name: MonthlyIncome, Length: 1349, dtype: int64
----------

20. MonthlyRate: 1427
4223     3
9150     3
9096     2
13008    2
12858    2
        ..
17071    1
23213    1
3835     1
25258    1
12287    1
Name: MonthlyRate, Length: 1427, dtype: int64
----------

21. NumCompaniesWorked: 10
```

```
1    521
0    197
3    159
2    146
4    139
7     74
6     70
5     63
9     52
8     49
Name: NumCompaniesWorked, dtype: int64
----------

22. Over18: 1
Y    1470
Name: Over18, dtype: int64
----------

23. OverTime: 2
No     1054
Yes     416
Name: OverTime, dtype: int64
----------

24. PercentSalaryHike: 15
11    210
13    209
14    201
12    198
15    101
18     89
17     82
16     78
19     76
22     56
20     55
21     48
23     28
24     21
25     18
Name: PercentSalaryHike, dtype: int64
----------

25. PerformanceRating: 2
3    1244
4     226
Name: PerformanceRating, dtype: int64
----------

26. RelationshipSatisfaction: 4
3    459
4    432
2    303
1    276
Name: RelationshipSatisfaction, dtype: int64
----------

27. StandardHours: 1
80    1470
Name: StandardHours, dtype: int64
----------

28. StockOptionLevel: 4
0    631
1    596
2    158
3     85
Name: StockOptionLevel, dtype: int64
----------

29. TotalWorkingYears: 40
10    202
6     125
8     103
9      96
5      88
7      81
1      81
4      63
12     48
3      42
15     40
16     37
```

```
13     36
11     36
21     34
17     33
14     31
2      31
20     30
18     27
19     22
23     22
22     21
24     18
28     14
25     14
26     14
0      11
29     10
31      9
32      9
27      7
30      7
33      7
36      6
34      5
37      4
35      3
40      2
38      1
Name: TotalWorkingYears, dtype: int64
----------

30. TrainingTimesLastYear: 7
2    547
3    491
4    123
5    119
1     71
6     65
0     54
Name: TrainingTimesLastYear, dtype: int64
----------

31. WorkLifeBalance: 4
3    893
2    344
4    153
1     80
Name: WorkLifeBalance, dtype: int64
----------

32. YearsAtCompany: 37
5     196
1     171
3     128
2     127
10    120
4     110
7      90
9      82
8      80
6      76
0      44
11     32
20     27
13     24
15     20
14     18
22     15
12     14
21     14
18     13
16     12
19     11
17      9
24      6
33      5
25      4
26      4
31      3
32      3
36      2
29      2
27      2
```

```
23      2
30      1
34      1
37      1
40      1
Name: YearsAtCompany, dtype: int64
----------

33. YearsInCurrentRole: 19
2     372
0     244
7     222
3     135
4     104
8      89
9      67
1      57
6      37
5      36
10     29
11     22
13     14
14     11
12     10
15      8
16      7
17      4
18      2
Name: YearsInCurrentRole, dtype: int64
----------

34. YearsSinceLastPromotion: 16
0     581
1     357
2     159
7      76
4      61
3      52
5      45
6      32
11     24
8      18
9      17
15     13
12     10
13     10
14      9
10      6
Name: YearsSinceLastPromotion, dtype: int64
----------

35. YearsWithCurrManager: 18
2     344
0     263
7     216
3     142
8     107
4      98
1      76
9      64
5      31
6      29
10     27
11     22
12     18
13     14
17      7
14      5
15      5
16      2
Name: YearsWithCurrManager, dtype: int64
----------
```

In [9]:

```
# Dropping unnecessary columns.
df.drop(['EmployeeCount', 'Over18', 'StandardHours', 'EmployeeNumber'], axis = 1, inplace = True)
```

In [10]:

```
df.describe()
```

Out[10]:

Out[10]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatis
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.
mean	36.923810	802.485714	9.192517	2.912925	2.721769	65.891156	2.729932	2.063946	2.
std	9.135373	403.509100	8.106864	1.024165	1.093082	20.329428	0.711561	1.106940	1.
min	18.000000	102.000000	1.000000	1.000000	1.000000	30.000000	1.000000	1.000000	1.
25%	30.000000	465.000000	2.000000	2.000000	2.000000	48.000000	2.000000	1.000000	2.
50%	36.000000	802.000000	7.000000	3.000000	3.000000	66.000000	3.000000	2.000000	3.
75%	43.000000	1157.000000	14.000000	4.000000	4.000000	83.750000	3.000000	3.000000	4.
max	60.000000	1499.000000	29.000000	5.000000	4.000000	100.000000	4.000000	5.000000	4.

8 rows × 23 columns

In [11]:
```
## Extracting continuous features
```

In [12]:
```
cont_data = df.select_dtypes(exclude = ['object'] )
cont_data
```

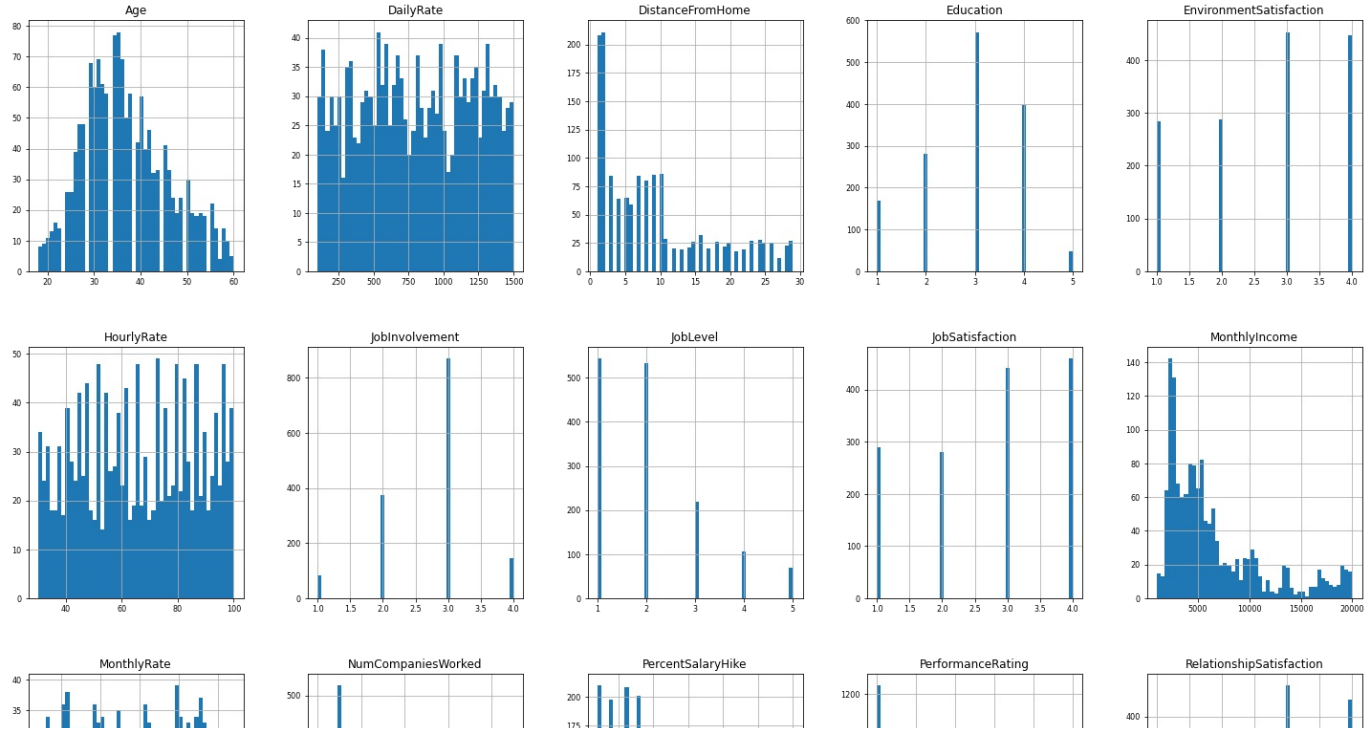
Out[12]:

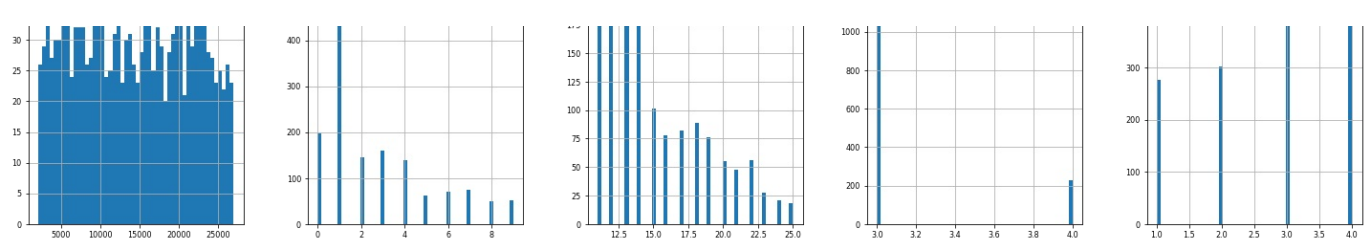
	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyI
0	41	1102	1	2	2	94	3	2	4	
1	49	279	8	1	3	61	2	2	2	
2	37	1373	2	2	4	92	2	1	3	
3	33	1392	3	4	4	56	3	1	3	
4	27	591	2	1	1	40	3	1	2	
...	...	...	...	...	...	...	...	...	...	
1465	36	884	23	2	3	41	4	2	4	
1466	39	613	6	1	4	42	2	3	1	
1467	27	155	4	3	2	87	4	2	2	
1468	49	1023	2	3	4	63	2	2	2	
1469	34	628	8	3	2	82	4	2	3	

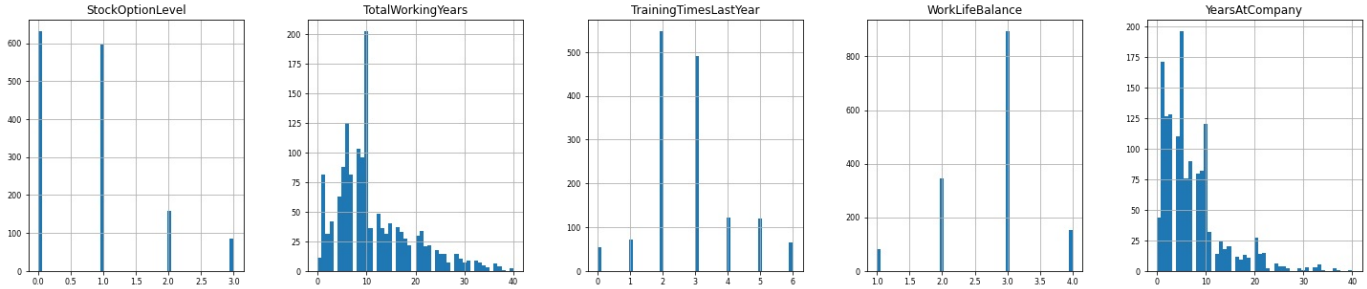
1470 rows × 23 columns

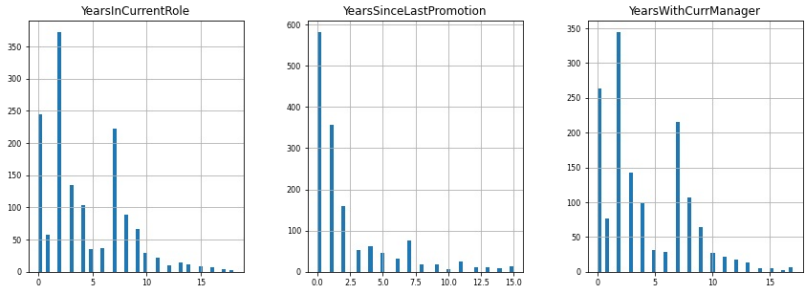
In [13]:
```
## Data Distribution
```

In [14]:
```
cont_data.hist(figsize = (25, 30), bins = 50, xlabelsize = 8, ylabelsize = 8)
plt.show()
```



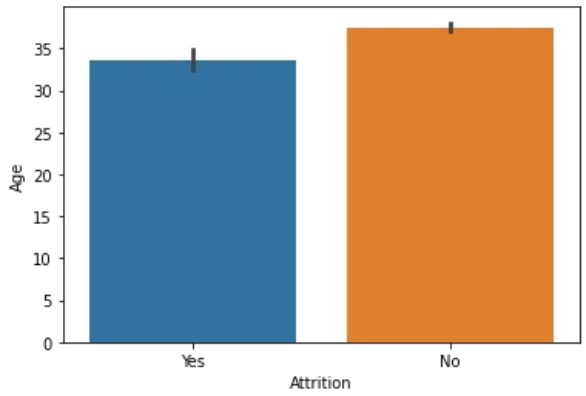


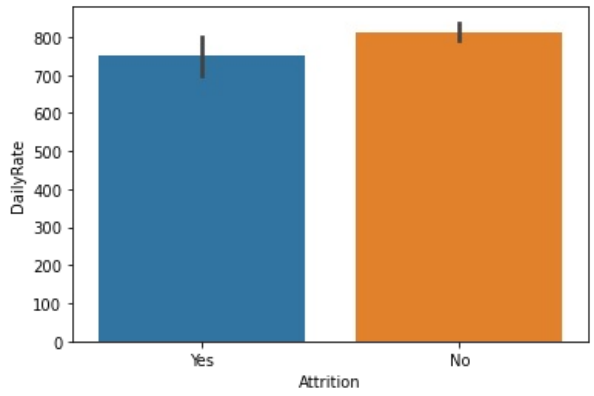


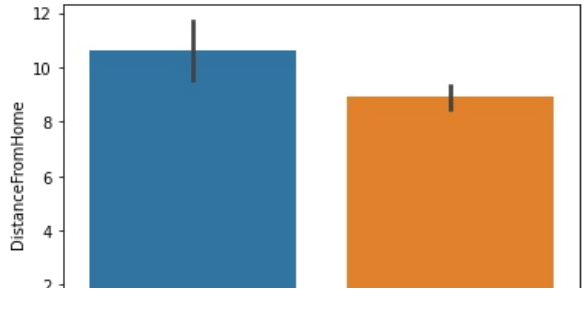


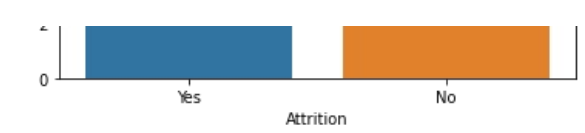
In [15]:

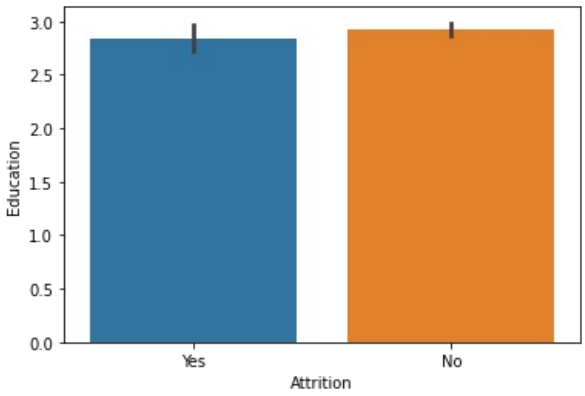
```
for i in cont_data:
    sns.barplot(y = cont_data[i], x = df['Attrition'])
    plt.show()
```

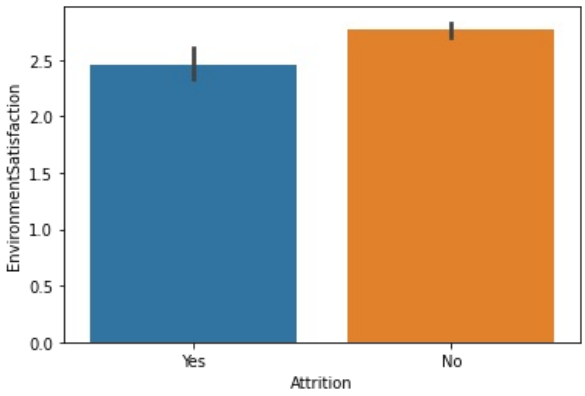


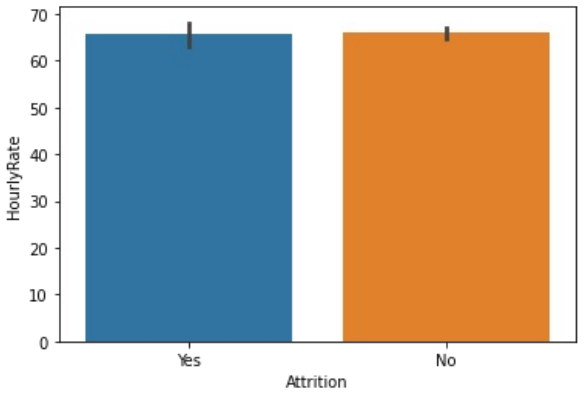


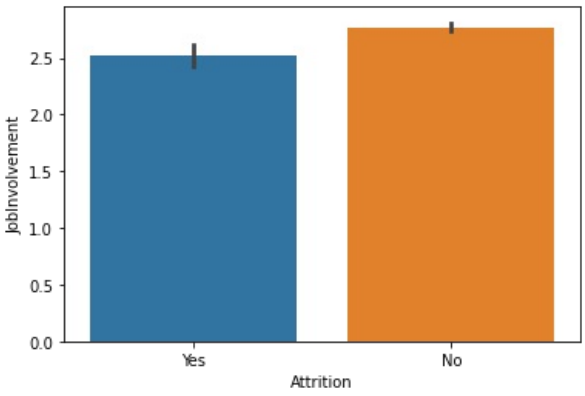


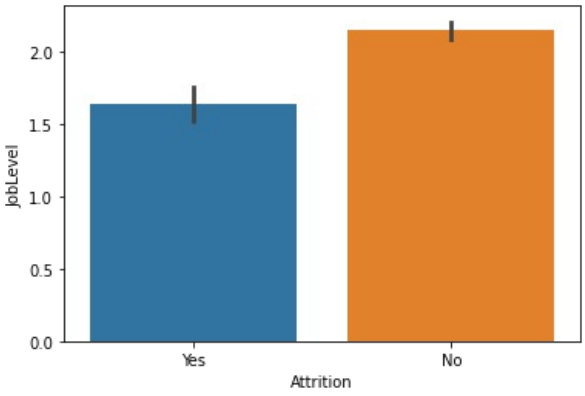

0
Yes
No
Attrition

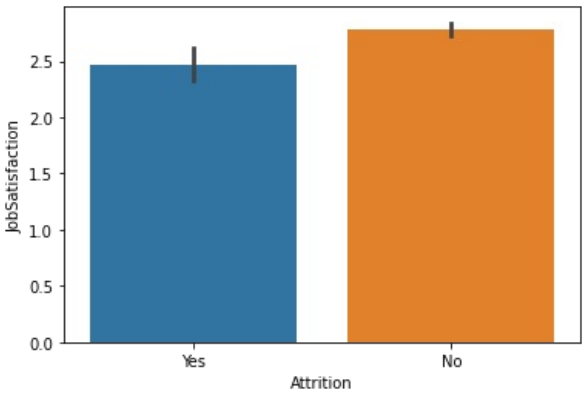

Education
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Yes
No
Attrition

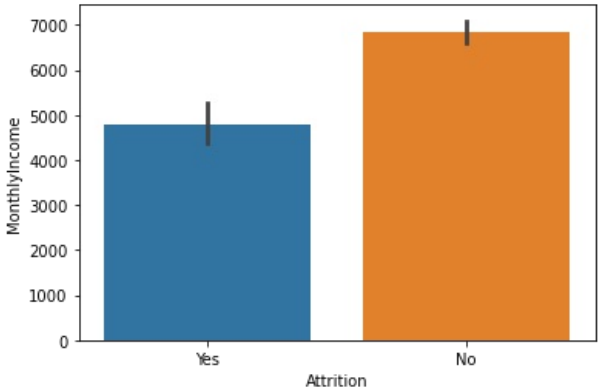

EnvironmentSatisfaction
2.5
2.0
1.5
1.0
0.5
0.0
Yes
No
Attrition

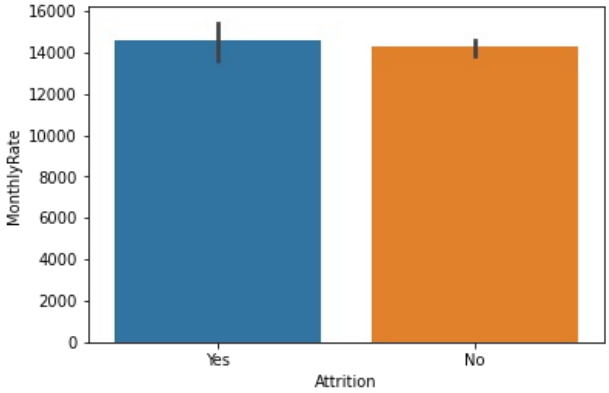

HourlyRate
70
60
50
40
30
20
10
0
Yes
No
Attrition

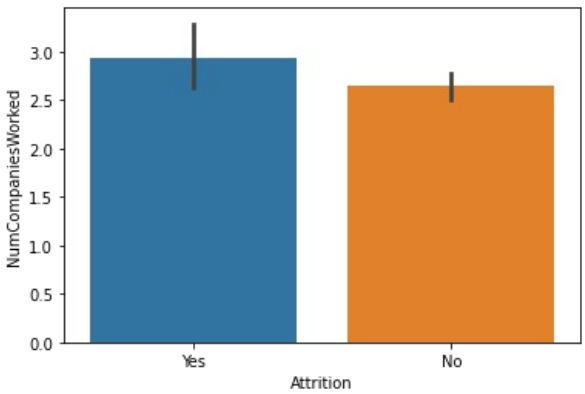

JobInvolvement
2.5
2.0
1.5
1.0
0.5
0.0
Yes
No
Attrition

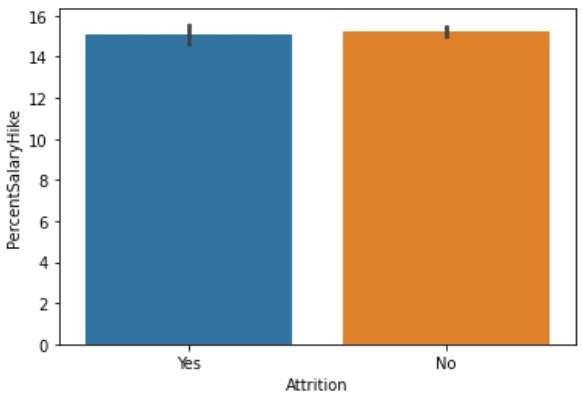

JobLevel
2.0
1.5
1.0
0.5
0.0
Yes
No
Attrition

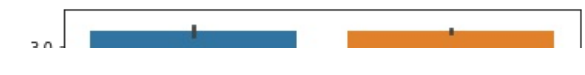

JobSatisfaction
0.0
0.5
1.0
1.5
2.0
2.5
Yes
No
Attrition

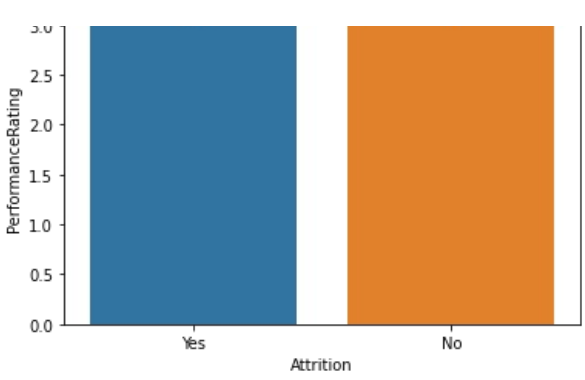

MonthlyIncome
0
1000
2000
3000
4000
5000
6000
7000
Yes
No
Attrition

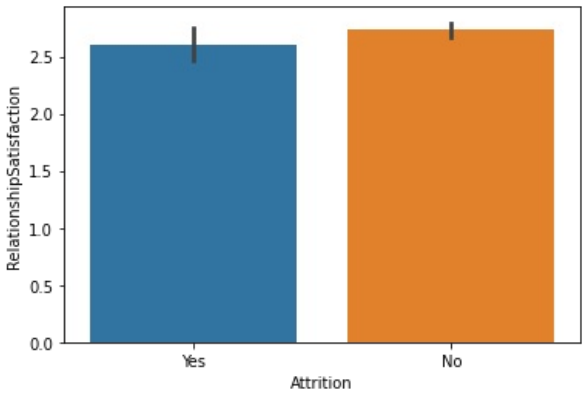

MonthlyRate
0
2000
4000
6000
8000
10000
12000
14000
16000
Yes
No
Attrition

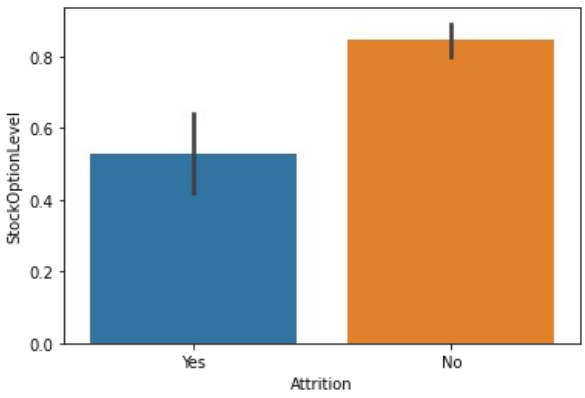

NumCompaniesWorked
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Yes
No
Attrition

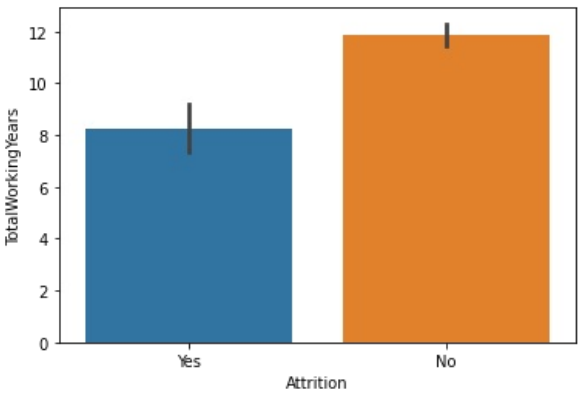

PercentSalaryHike
0
2
4
6
8
10
12
14
16
Yes
No
Attrition

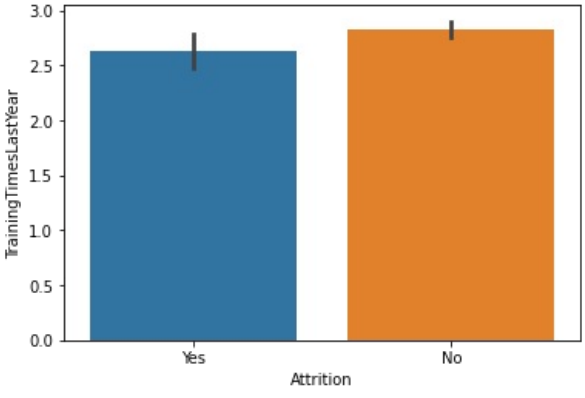



PerformanceRating
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Yes
No
Attrition

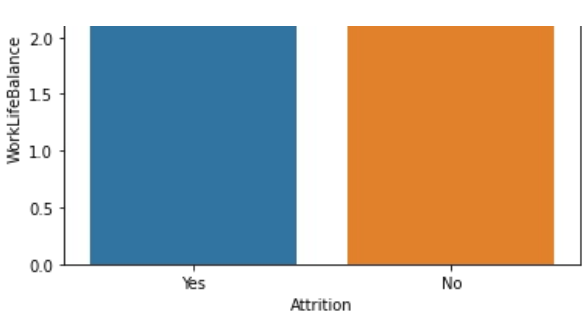

RelationshipSatisfaction
2.5
2.0
1.5
1.0
0.5
0.0
Yes
No
Attrition

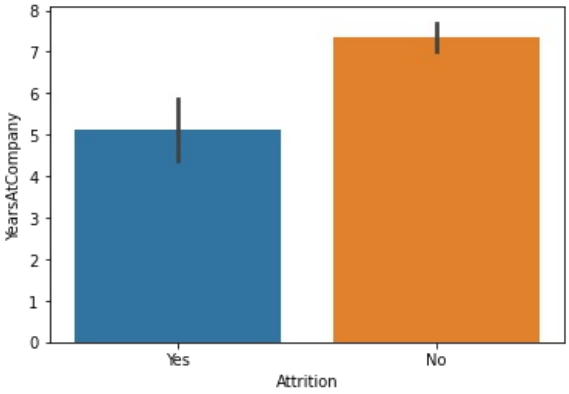

StockOptionLevel
0.8
0.6
0.4
0.2
0.0
Yes
No
Attrition

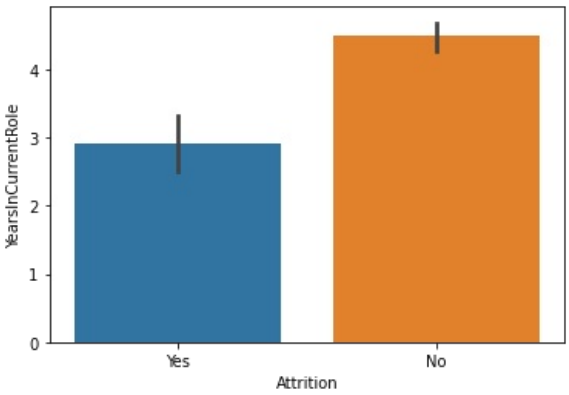

TotalWorkingYears
12
10
8
6
4
2
0
Yes
No
Attrition

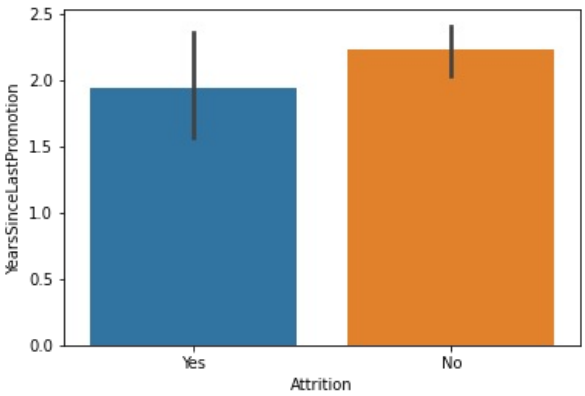

TrainingTimesLastYear
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Yes
No
Attrition

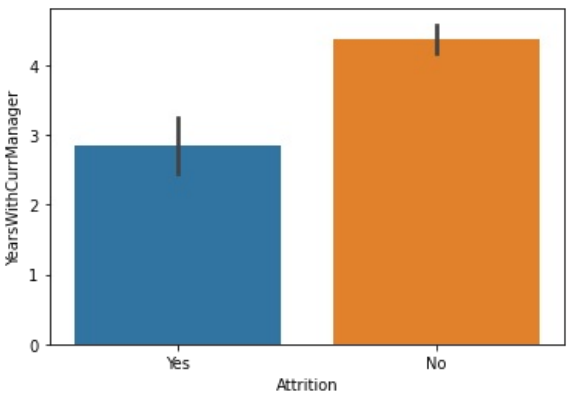

2.5


WorkLifeBalance
2.0
1.5
1.0
0.5
0.0
Yes
No
Attrition


YearsAtCompany
8
7
6
5
4
3
2
1
0
Yes
No
Attrition

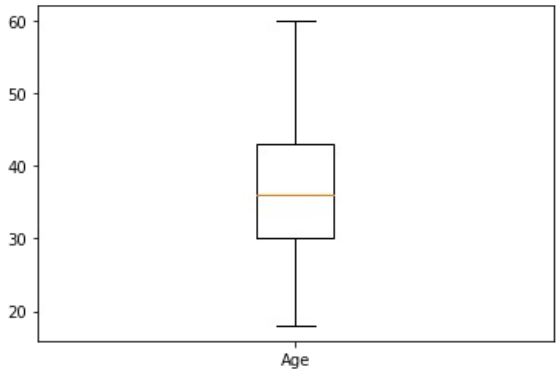

YearsInCurrentRole
4
3
2
1
0
Yes
No
Attrition

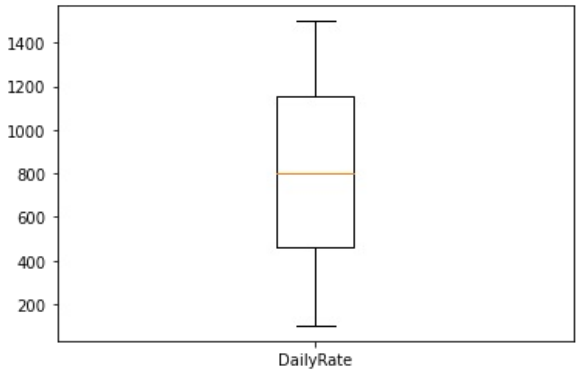

YearsSinceLastPromotion
2.5
2.0
1.5
1.0
0.5
0.0
Yes
No
Attrition

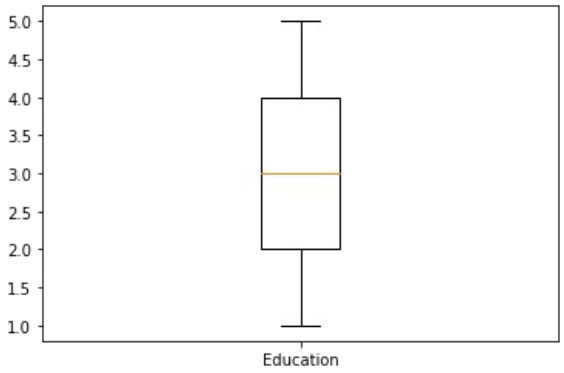

YearsWithCurrManager
4
3
2
1
0
Yes
No
Attrition

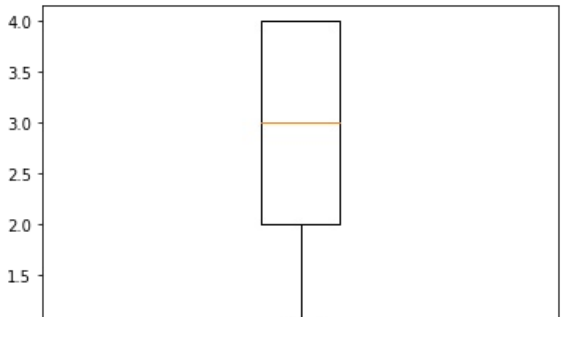
In [16]:
```
## checking for outliers
```

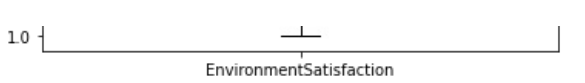
In [17]:
```
for i in cont_data:
    plt.boxplot(cont_data[i], labels = [i])
    plt.show()
```

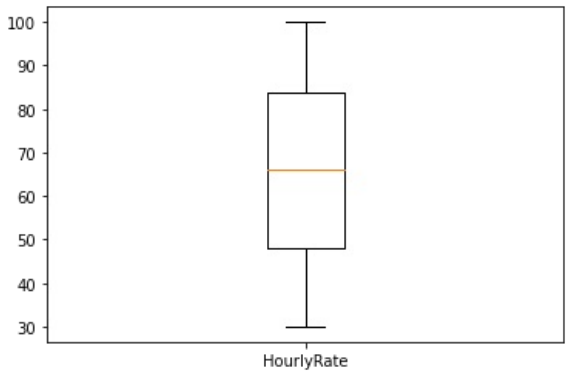


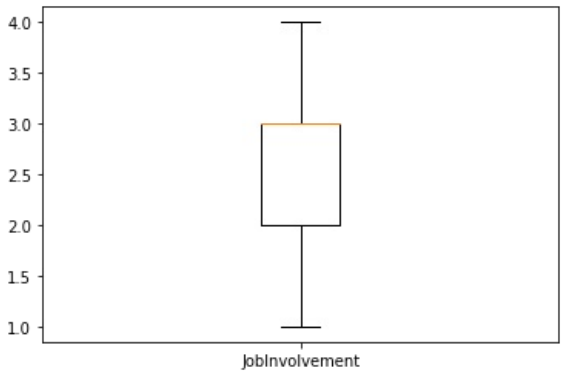


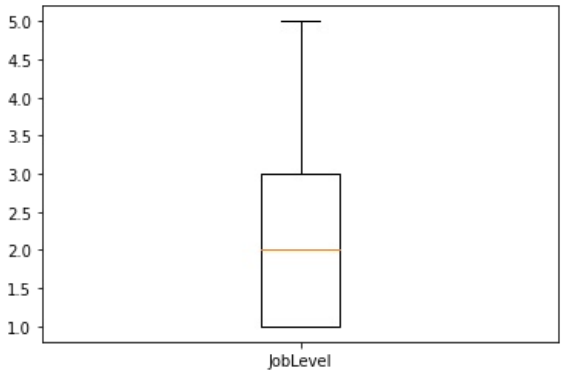




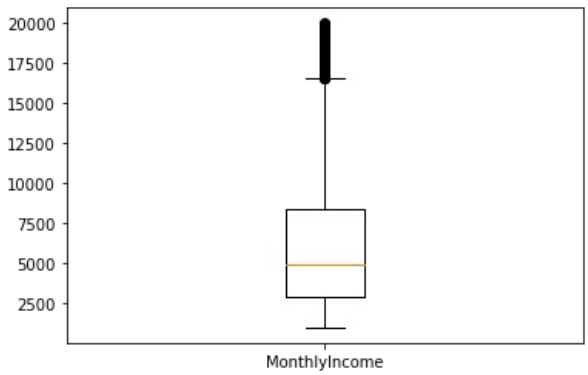

1.0
EnvironmentSatisfaction

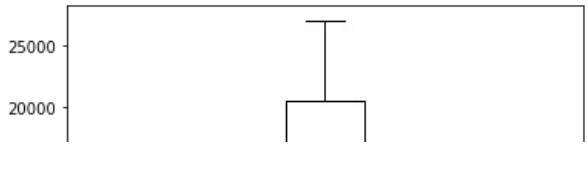

100
90
80
70
60
50
40
30
HourlyRate

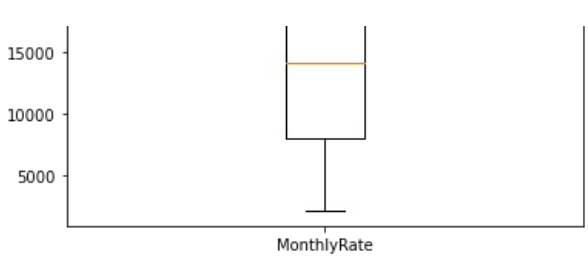

4.0
3.5
3.0
2.5
2.0
1.5
1.0
JobInvolvement

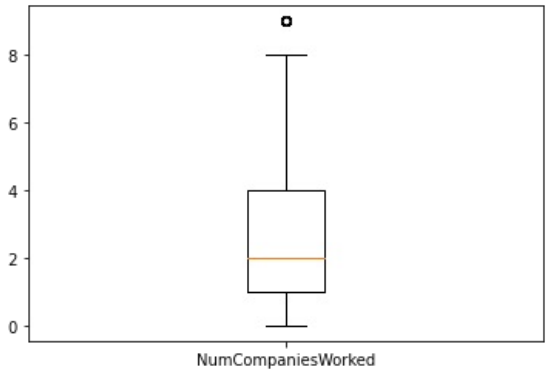

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
JobLevel

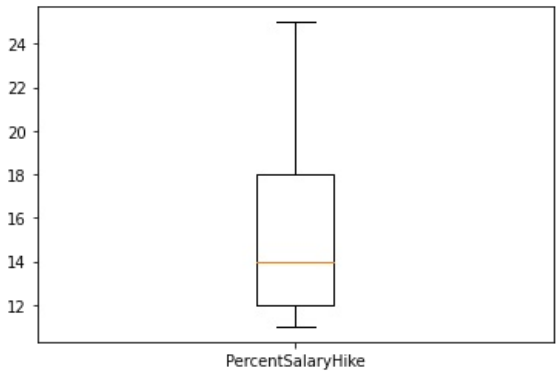

4.0
3.5
3.0
2.5
2.0
1.5
1.0
JobSatisfaction

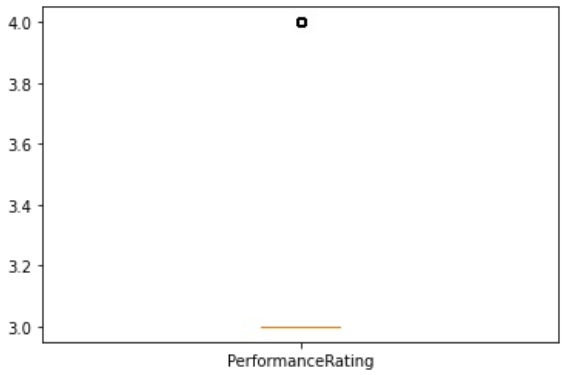

20000
17500
15000
12500
10000
7500
5000
2500
MonthlyIncome


25000
20000

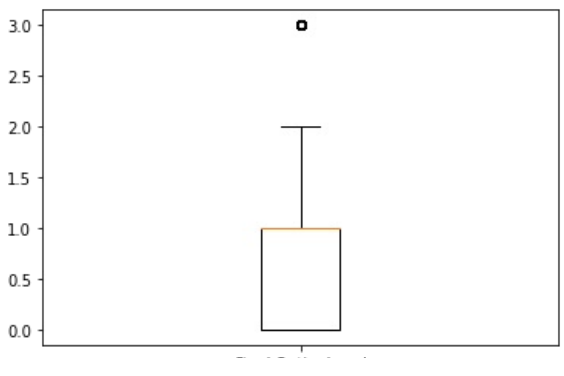

15000
10000
5000
MonthlyRate


8
6
4
2
0
NumCompaniesWorked

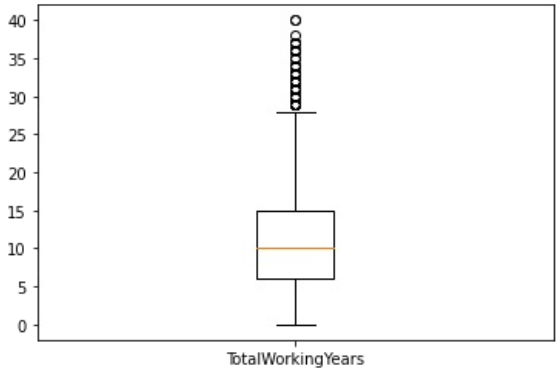

24
22
20
18
16
14
12
PercentSalaryHike


4.0
3.8
3.6
3.4
3.2
3.0
PerformanceRating

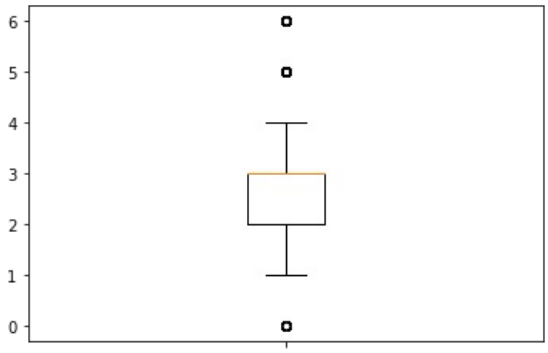

4.0
3.5
3.0
2.5
2.0
1.5
1.0
RelationshipSatisfaction


3.0
2.5
2.0
1.5
1.0
0.5
0.0

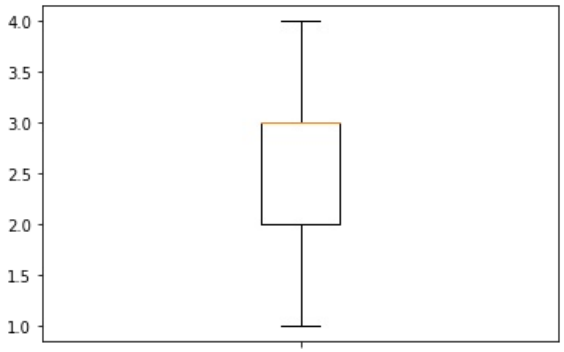
StockOptionLevel



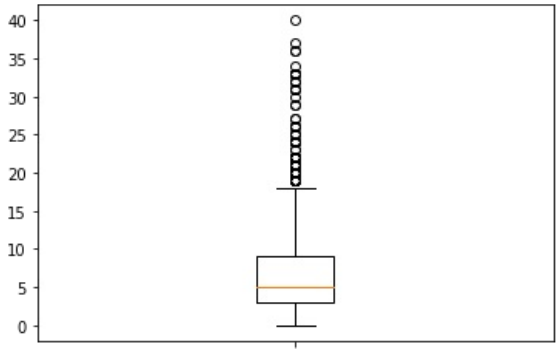
TotalWorkingYears



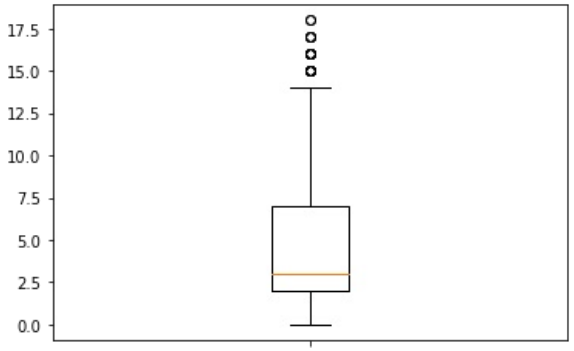
TrainingTimesLastYear



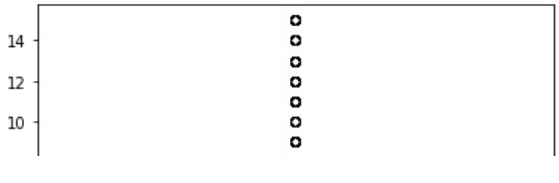
WorkLifeBalance

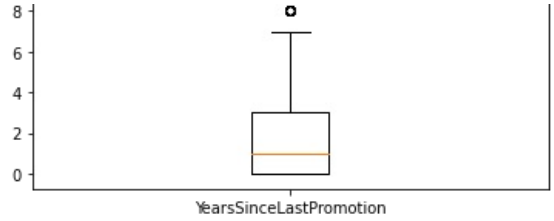


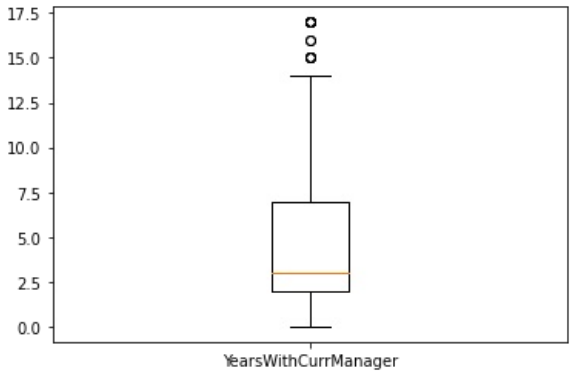
YearsAtCompany



YearsInCurrentRole



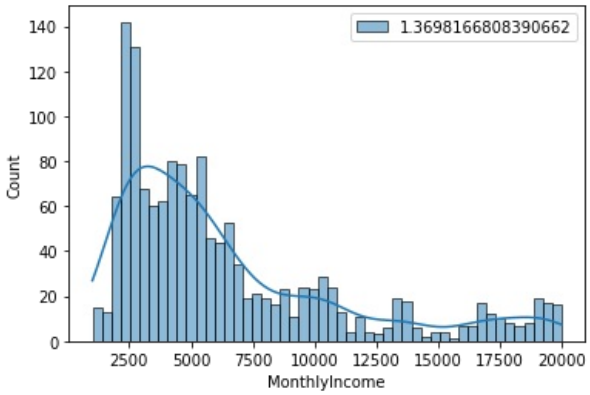


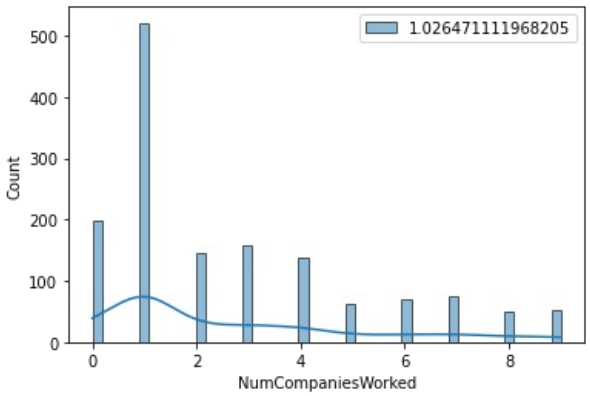


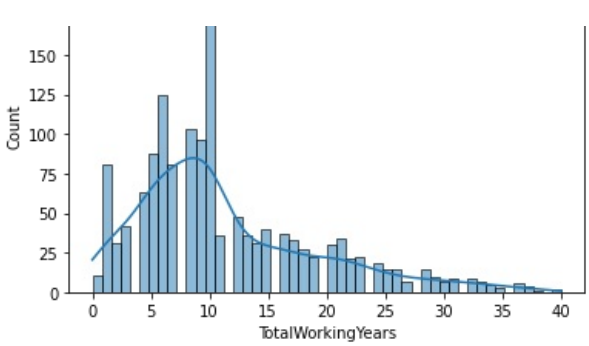
In [18]:
```
##Some variables may show outliers here in the boxplot but actually have a meaningful reason for their presence.
```

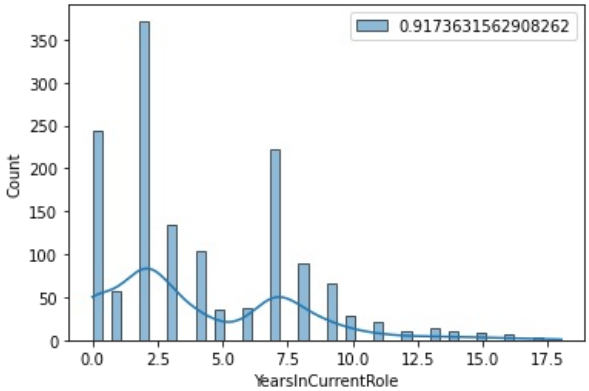
In [19]:
```
a = ['MonthlyIncome','NumCompaniesWorked', 'TotalWorkingYears', 'YearsInCurrentRole',
     'TrainingTimesLastYear', 'YearsAtCompany', 'YearsSinceLastPromotion', 'YearsWithCurrManager']
```

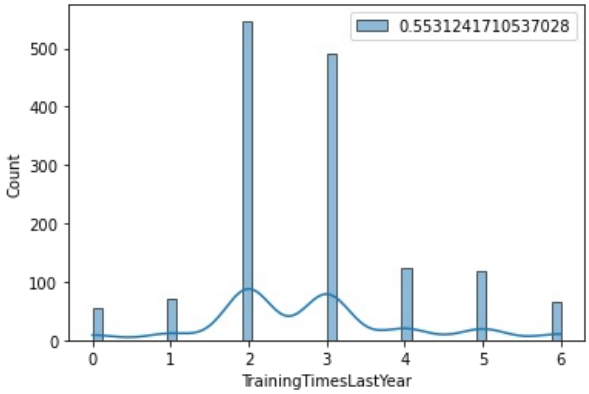
In [20]:
```
for i in a:
    sns.histplot(cont_data[i], kde = True, bins = 50, label = cont_data[i].skew())
    plt.legend(loc = 'upper right')
    plt.show()
```

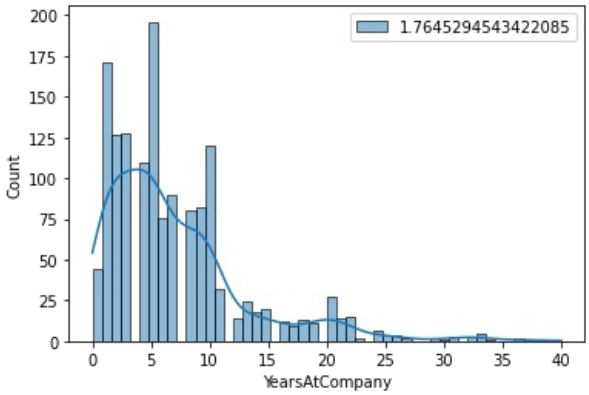


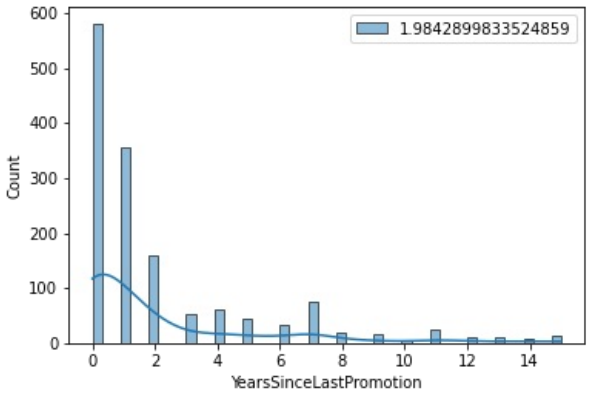


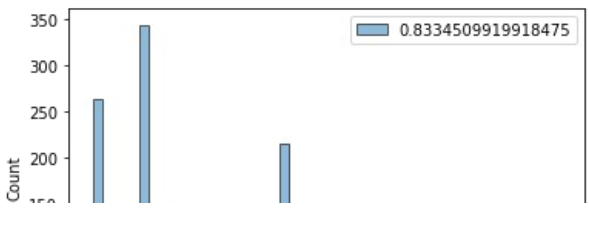

Count
TotalWorkingYears

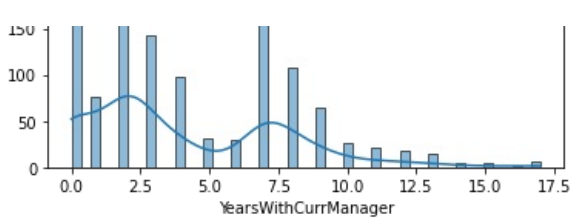

0.9173631562908262
Count
YearsInCurrentRole


0.5531241710537028
Count
TrainingTimesLastYear


1.7645294543422085
Count
YearsAtCompany


1.9842899833524859
Count
YearsSinceLastPromotion


0.8334509919918475
Count



In [21]:
```
out_vars = ['MonthlyIncome', 'TotalWorkingYears', 'TrainingTimesLastYear',
            'YearsAtCompany', 'YearsSinceLastPromotion', 'YearsInCurrentRole', 'YearsWithCurrManager']
```

In [22]:
```
def outlierTreat(x):
    upper = x.quantile(.75) + 1.5 * (x.quantile(.75) - x.quantile(.25))
    lower = x.quantile(.25) - 1.5 * (x.quantile(.75) - x.quantile(.25))
    return x.clip(lower, upper)
```

In [23]:
```
cont_data.loc[:, out_vars] = cont_data.loc[:, out_vars].apply(outlierTreat)
cont_data.loc[:, out_vars]
```

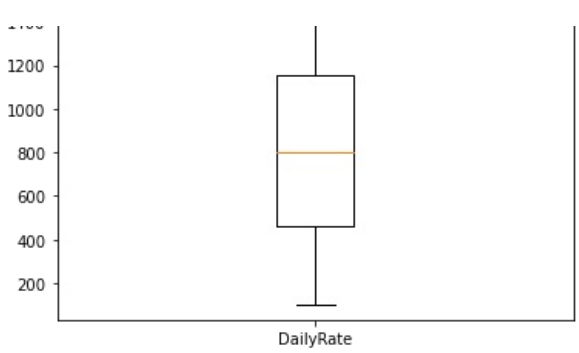
```
C:\Users\Rakesh Lodem\anaconda3\lib\site-packages\pandas\core\indexing.py:1787: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retur
ning-a-view-versus-a-copy
  self._setitem_single_column(loc, val, pi)
```

Out[23]:

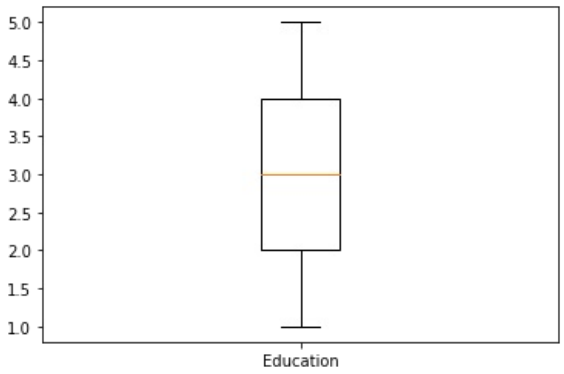
	MonthlyIncome	TotalWorkingYears	TrainingTimesLastYear	YearsAtCompany	YearsSinceLastPromotion	YearsInCurrentRole	YearsWithCurr
0	5993	8.0	0.5	6	0.0	4.0	
1	5130	10.0	3.0	10	1.0	7.0	
2	2090	7.0	3.0	0	0.0	0.0	
3	2909	8.0	3.0	8	3.0	7.0	
4	3468	6.0	3.0	2	2.0	2.0	
...	...	...	...	...	...	...	
1465	2571	17.0	3.0	5	0.0	2.0	
1466	9991	9.0	4.5	7	1.0	7.0	
1467	6142	6.0	0.5	6	0.0	2.0	
1468	5390	17.0	3.0	9	0.0	6.0	
1469	4404	6.0	3.0	4	1.0	3.0	

1470 rows × 7 columns

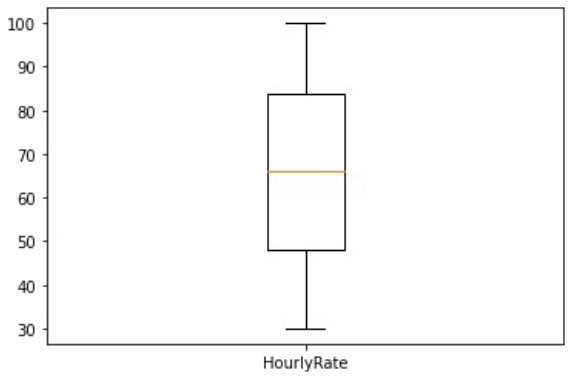
In [24]:
```
# Using box plot for checking the presence of outliers.
for i in cont_data:
    plt.boxplot(cont_data[i], labels = [i])
    plt.show()
```

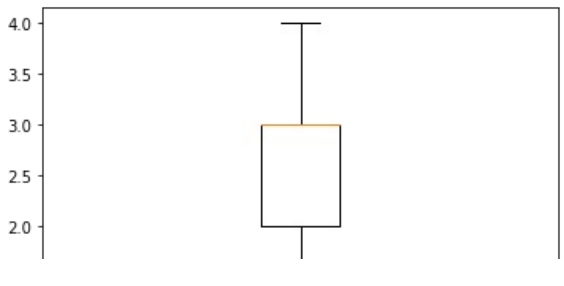

1200
1000
800
600
400
200
DailyRate

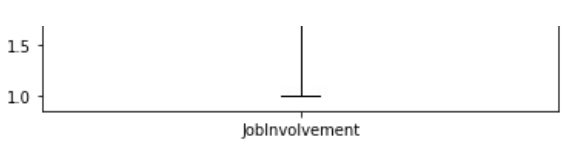

30
25
20
15
10
5
0
DistanceFromHome

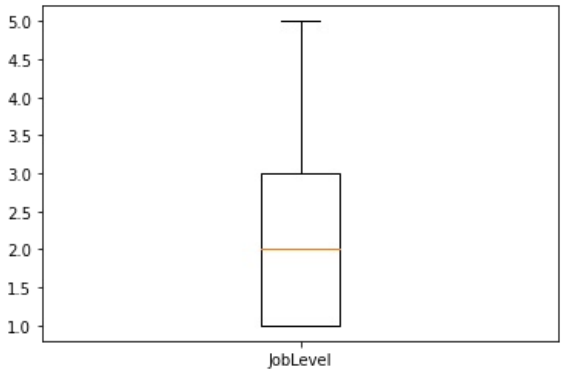

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Education

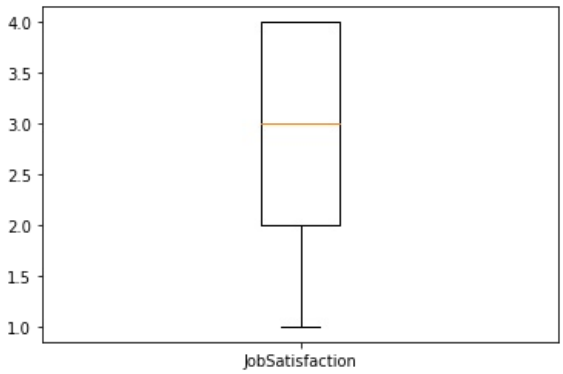

4.0
3.5
3.0
2.5
2.0
1.5
1.0
EnvironmentSatisfaction

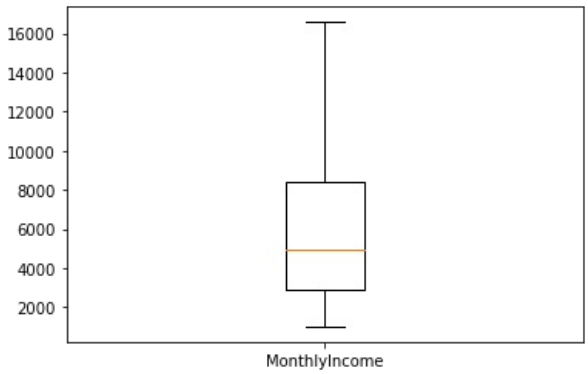

100
90
80
70
60
50
40
30
HourlyRate

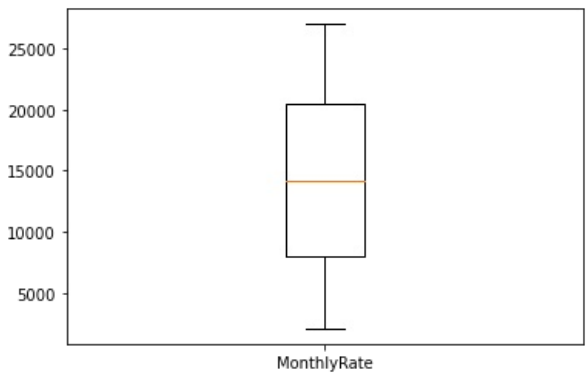

4.0
3.5
3.0
2.5
2.0

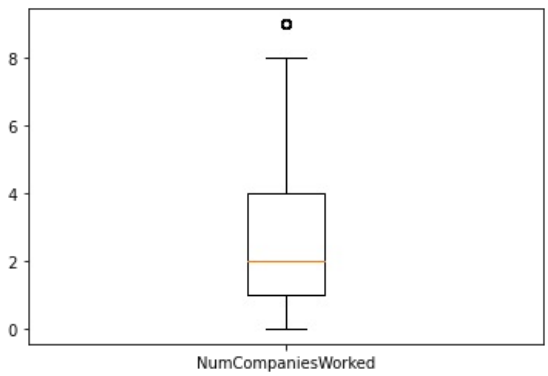

1.5
1.0
JobInvolvement

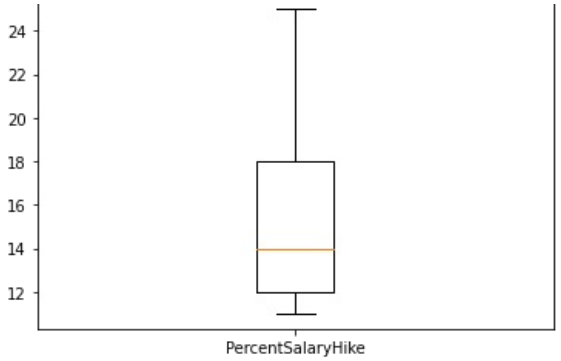

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
JobLevel

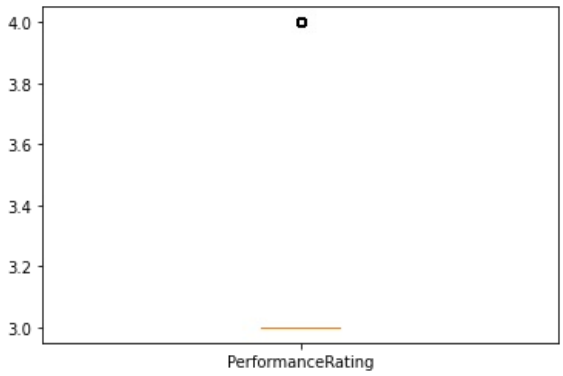

4.0
3.5
3.0
2.5
2.0
1.5
1.0
JobSatisfaction

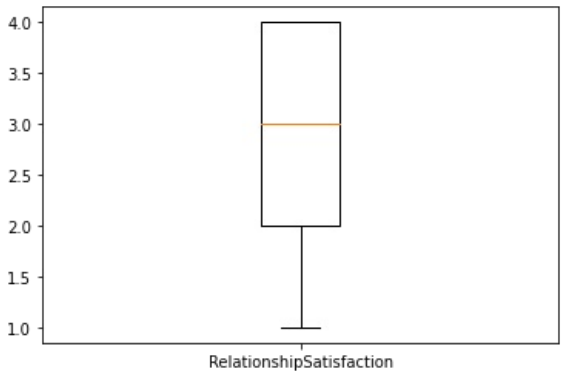

16000
14000
12000
10000
8000
6000
4000
2000
MonthlyIncome

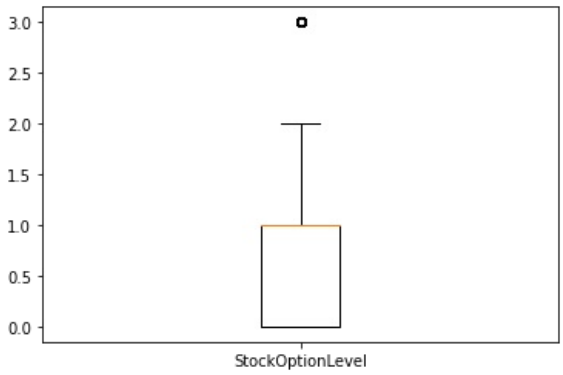

25000
20000
15000
10000
5000
MonthlyRate

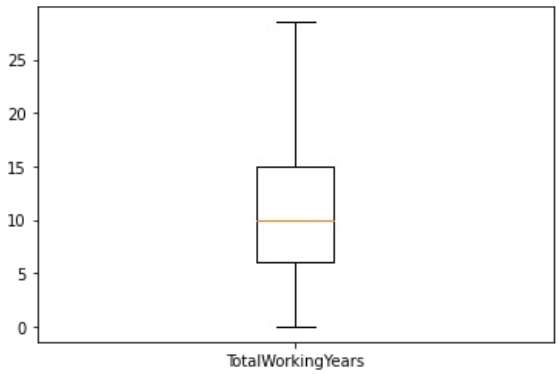

8
6
4
2
0
NumCcompaniesWorked

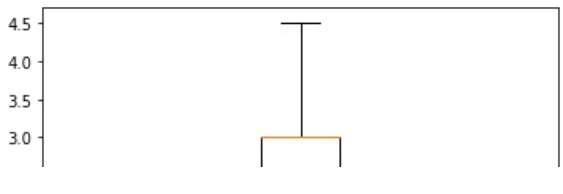

24
22
20
18
16
14
12
PercentSalaryHike

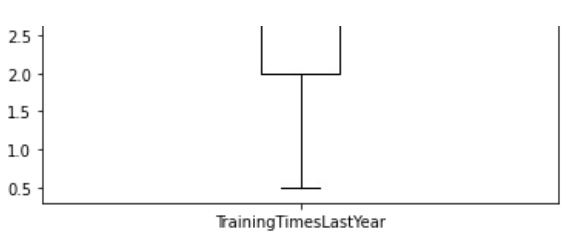

4.0
3.8
3.6
3.4
3.2
3.0
PerformanceRating

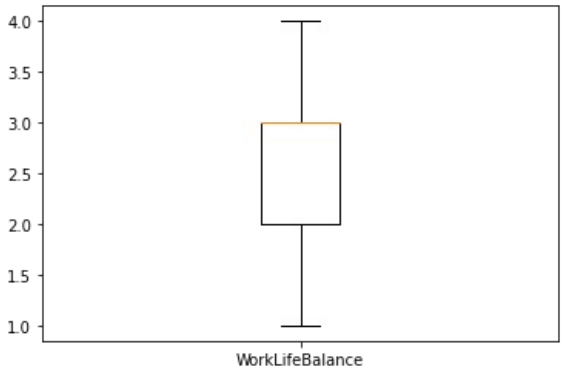

4.0
3.5
3.0
2.5
2.0
1.5
1.0
RelationshipSatisfaction

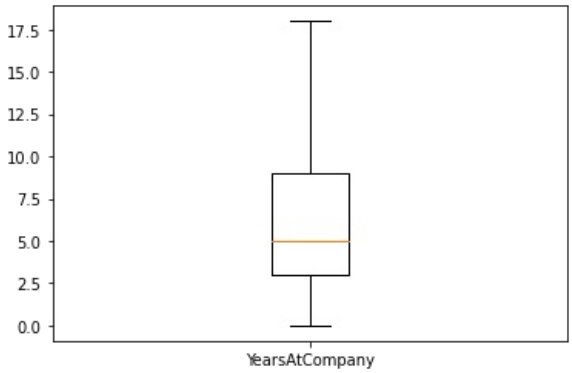

3.0
2.5
2.0
1.5
1.0
0.5
0.0
StockOptionLevel

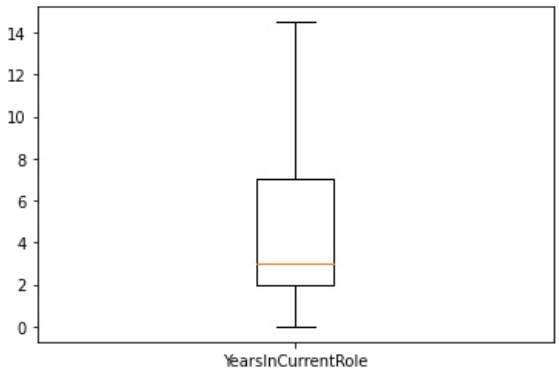

25
20
15
10
5
0
TotalWorkingYears

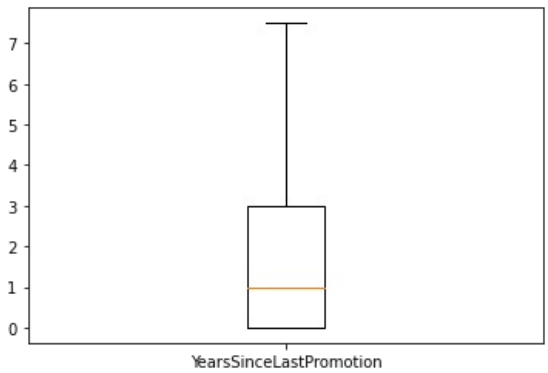

4.5
4.0
3.5
3.0

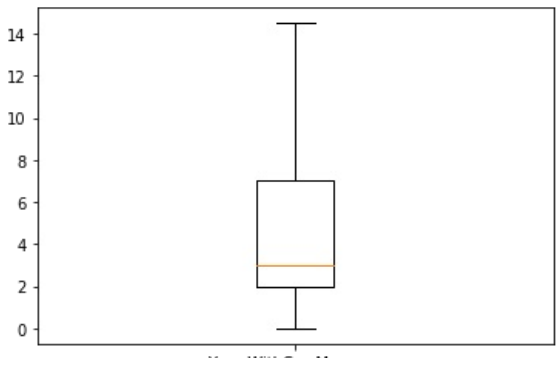

2.5
2.0
1.5
1.0
0.5
TrainingTimesLastYear


4.0
3.5
3.0
2.5
2.0
1.5
1.0
WorkLifeBalance


17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
YearsAtCompany


14
12
10
8
6
4
2
0
YearsInCurrentRole


7
6
5
4
3
2
1
0
YearsSinceLastPromotion


14
12
10
8
6
4
2
0

YearsWithCurrManager

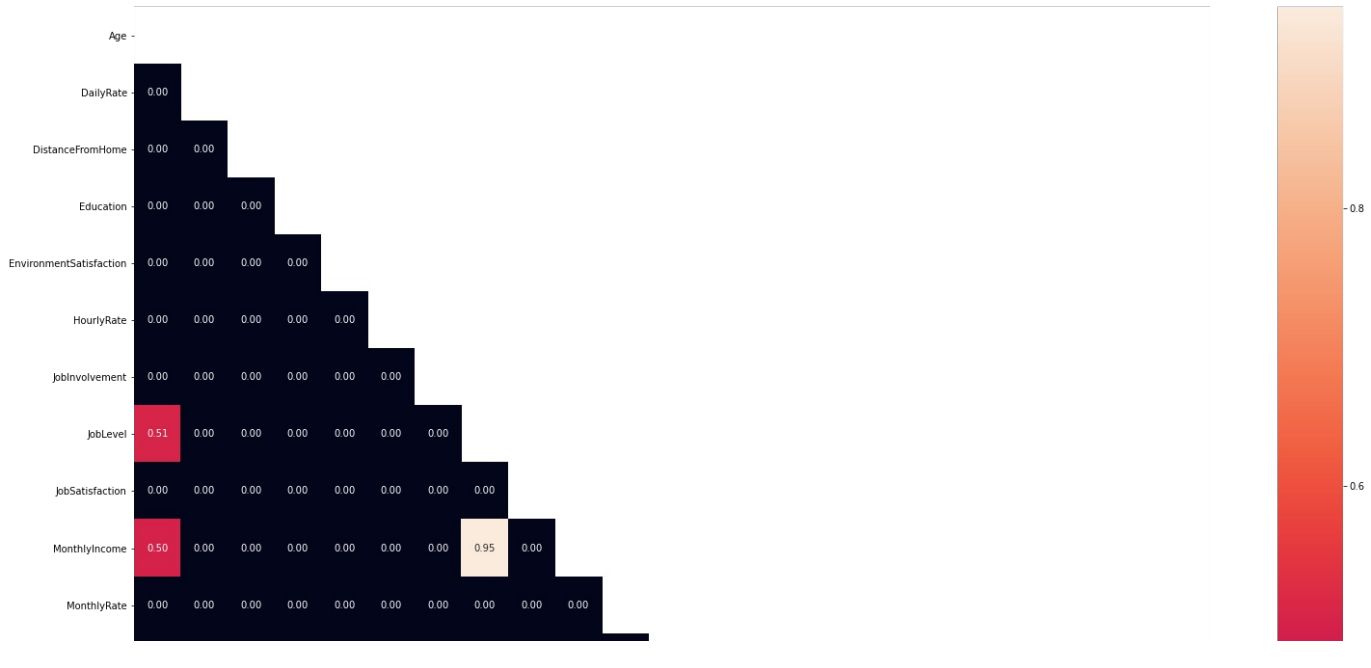
In [25]:
```
## checking the presence of missing values
```

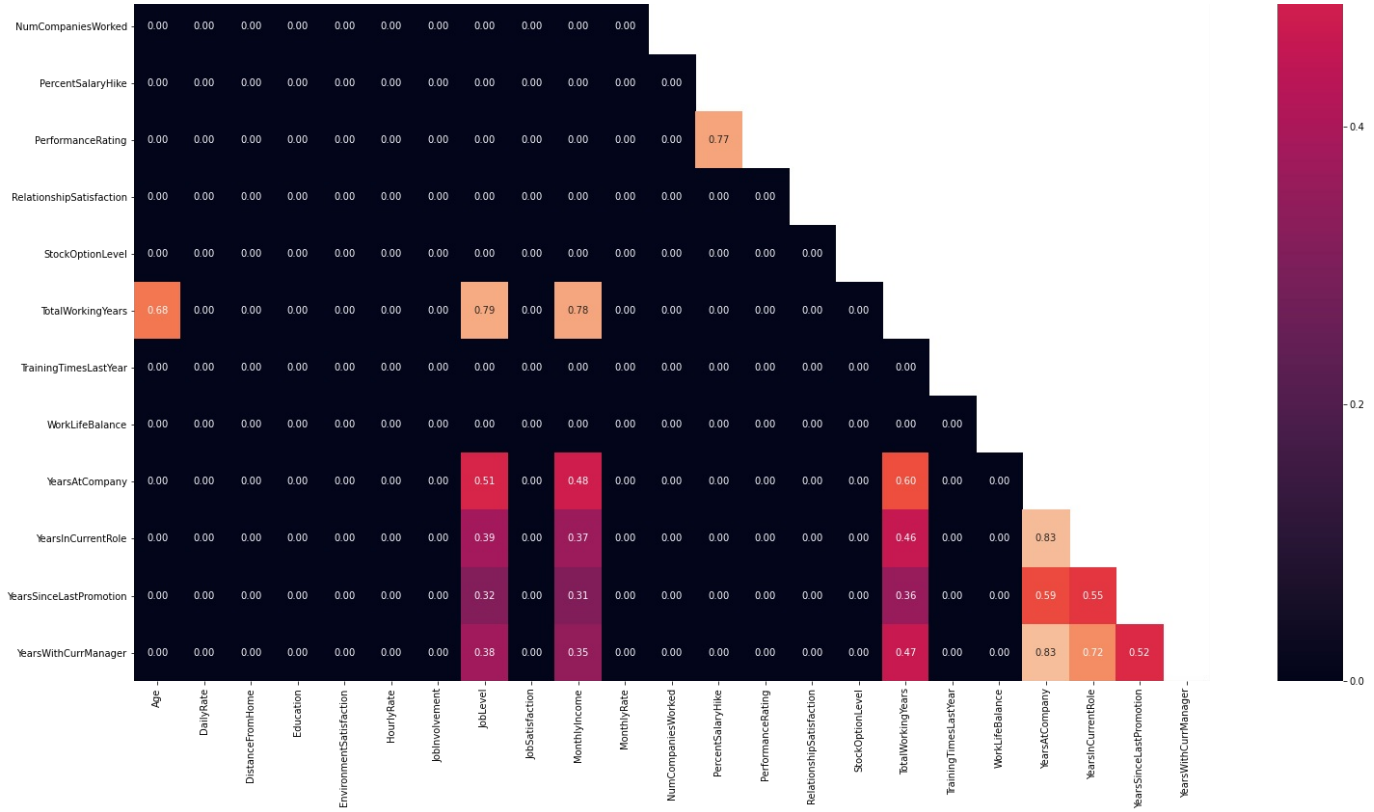
In [26]:
```
for i in cont_data:
    print(f'{i}: {cont_data.shape[0] - cont_data[i].count()}')
```

```
Age: 0
DailyRate: 0
DistanceFromHome: 0
Education: 0
EnvironmentSatisfaction: 0
HourlyRate: 0
JobInvolvement: 0
JobLevel: 0
JobSatisfaction: 0
MonthlyIncome: 0
MonthlyRate: 0
NumCompaniesWorked: 0
PercentSalaryHike: 0
PerformanceRating: 0
RelationshipSatisfaction: 0
StockOptionLevel: 0
TotalWorkingYears: 0
TrainingTimesLastYear: 0
WorkLifeBalance: 0
YearsAtCompany: 0
YearsInCurrentRole: 0
YearsSinceLastPromotion: 0
YearsWithCurrManager: 0
```

In [27]:
```
# Finding the correlation.
corr = cont_data.corr()

# Setting the size of figure.
plt.rcParams['figure.figsize'] = (25, 25)

# Argument Trimming out the values above the main diagonal.
mask = np.triu(corr)

# Setting low correlation value to 0.
corr[(corr.values < 0.3) & (corr.values > -0.3)] = 0

# Plotting the heatmap.
sns.heatmap(corr, annot = True, fmt = '.2f', mask = mask)
```

Out[27]: <AxesSubplot:>





In [28]:
```
## exploring the categorical variables
```

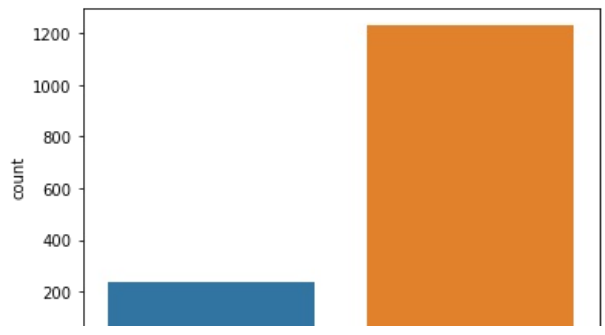
In [29]:
```
cat_vars = df.select_dtypes(include = ['object'])
cat_vars
```

Out[29]:

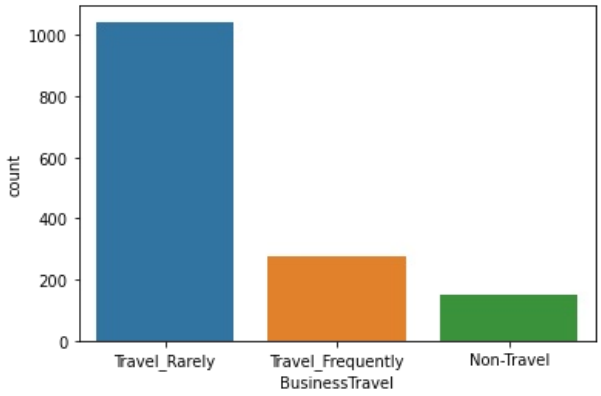
	Attrition	BusinessTravel	Department	EducationField	Gender	JobRole	MaritalStatus	OverTime
0	Yes	Travel_Rarely	Sales	Life Sciences	Female	Sales Executive	Single	Yes
1	No	Travel_Frequently	Research & Development	Life Sciences	Male	Research Scientist	Married	No
2	Yes	Travel_Rarely	Research & Development	Other	Male	Laboratory Technician	Single	Yes
3	No	Travel_Frequently	Research & Development	Life Sciences	Female	Research Scientist	Married	Yes
4	No	Travel_Rarely	Research & Development	Medical	Male	Laboratory Technician	Married	No
...	...	...	...	...	...	...	...	...
1465	No	Travel_Frequently	Research & Development	Medical	Male	Laboratory Technician	Married	No
1466	No	Travel_Rarely	Research & Development	Medical	Male	Healthcare Representative	Married	No
1467	No	Travel_Rarely	Research & Development	Life Sciences	Male	Manufacturing Director	Married	Yes
1468	No	Travel_Frequently	Sales	Medical	Male	Sales Executive	Married	No
1469	No	Travel_Rarely	Research & Development	Medical	Male	Laboratory Technician	Married	No

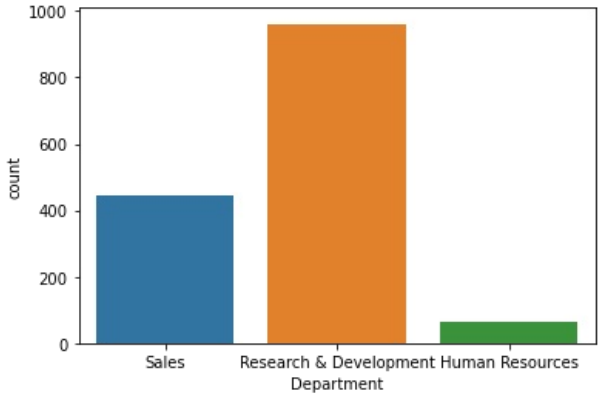
1470 rows × 8 columns

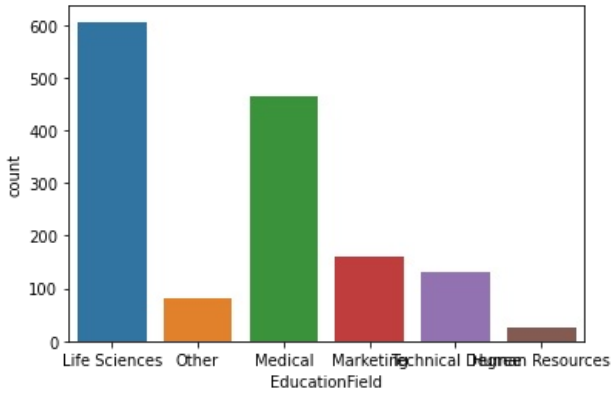
In [30]:
```
# Looking at the data distribution for different values.
plt.rcParams['figure.figsize'] = (6, 4)
for i in cat_vars:
    sns.countplot(x = cat_vars[i])
    plt.show()
```

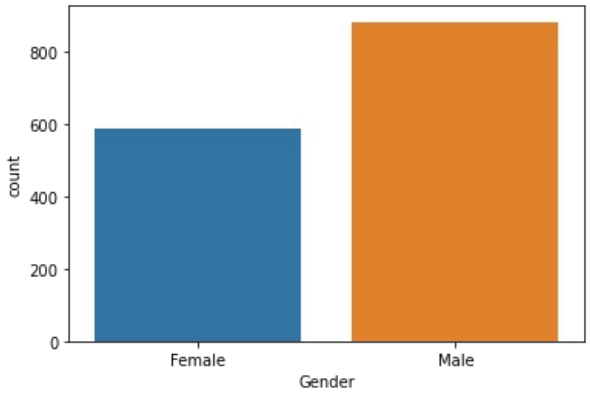


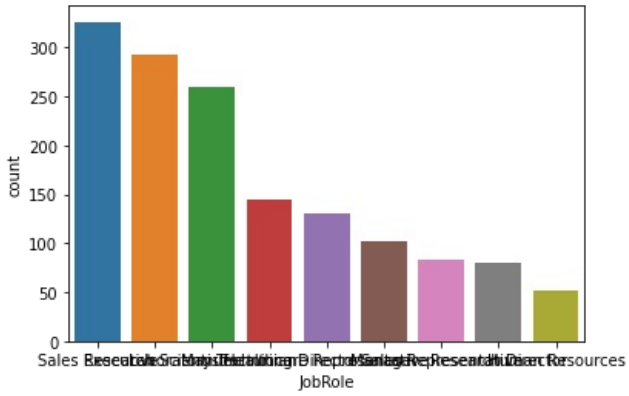

0
Yes
No
Attrition


1000
800
600
400
200
0
count
Travel_Rarely
Travel_Frequently
Non-Travel
BusinessTravel

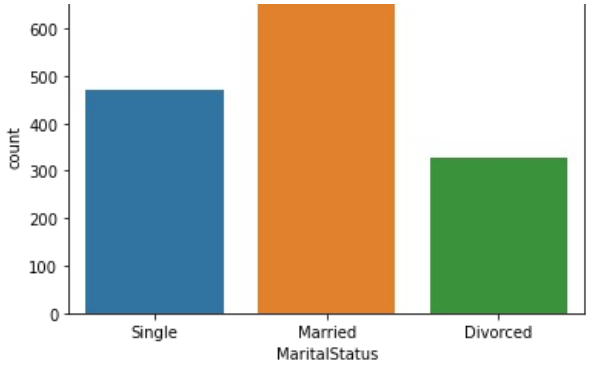

1000
800
600
400
200
0
count
Sales
Research & Development
Human Resources
Department

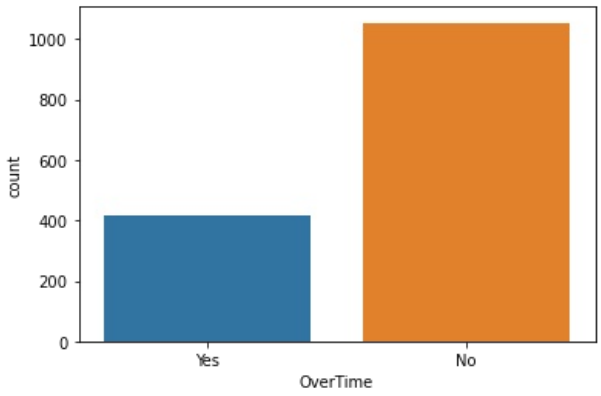

600
500
400
300
200
100
0
count
Life Sciences
Other
Medical
EducationField


800
600
400
200
0
count
Female
Male
Gender


300
250
200
150
100
50
0
count
JobRole


700





In [31]:

```
# Count values of different values for each variables.
for i in cat_vars:
    print(cat_vars[i].value_counts(), end = '\n---------\n\n')
```

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

Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
---------

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

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

Male      882
Female    588
Name: Gender, dtype: int64
---------

Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64
---------
```

```
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
---------

No     1054
Yes     416
Name: OverTime, dtype: int64
---------
```

In [32]:

```
cat_vars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Attrition       1470 non-null   object
 1   BusinessTravel  1470 non-null   object
 2   Department      1470 non-null   object
 3   EducationField  1470 non-null   object
 4   Gender          1470 non-null   object
 5   JobRole         1470 non-null   object
 6   MaritalStatus   1470 non-null   object
 7   OverTime        1470 non-null   object
dtypes: object(8)
memory usage: 92.0+ KB
```

In [51]:

```
cat_data = cat_vars.copy()
cat_data = pd.get_dummies(cat_data, drop_first = True) ## numerical features to continuos features
cat_data
```

Out[51]:

	Attrition_Yes	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	Department_Research & Development	Department_Sales	EducationField_Lif Science
0	1	0	1	0	1	
1	0	1	0	1	0	
2	1	0	1	1	0	
3	0	1	0	1	0	
4	0	0	1	1	0	
...	...	...	...	...	...	
1465	0	1	0	1	0	
1466	0	0	1	1	0	
1467	0	0	1	1	0	
1468	0	1	0	0	1	
1469	0	0	1	1	0	

1470 rows × 22 columns

In [57]:

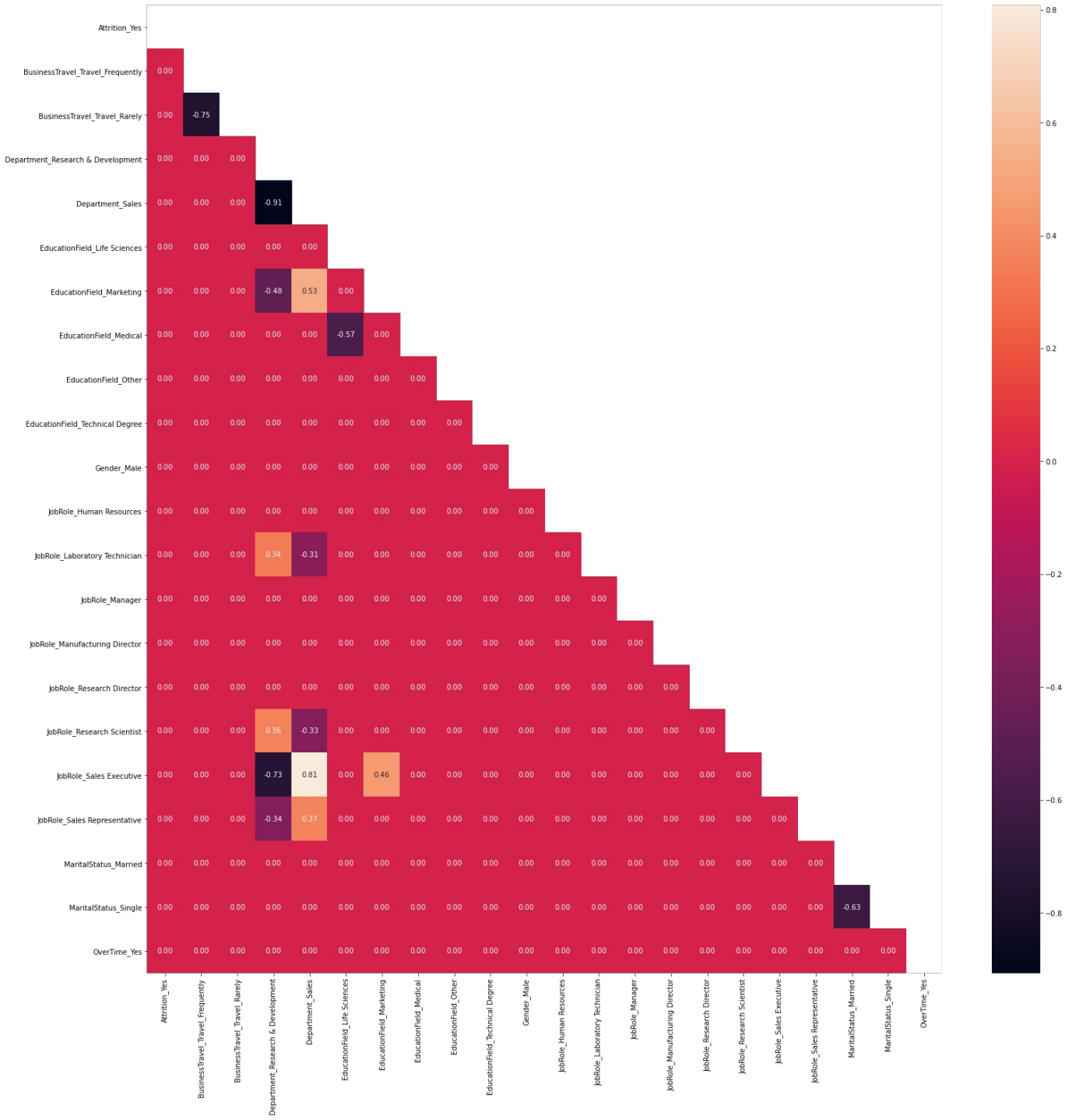
```
cat_data['Attrition_Yes']
```

Out[57]:

```
0       1
1       0
2       1
3       0
4       0
       ..
1465    0
1466    0
1467    0
1468    0
1469    0
Name: Attrition_Yes, Length: 1470, dtype: uint8
```

In [34]:
```
# Finding the correlation.
corr = cat_data.corr()

# Setting the size of figure.
plt.rcParams['figure.figsize'] = (25, 25)

# Argument Trimming out the values above the main diagonal.
mask = np.triu(corr)

# Setting low correlation value to 0.
corr[(corr.values < 0.3) & (corr.values > -0.3)] = 0

# Plotting the heatmap.
sns.heatmap(corr, annot = True, fmt = '.2f', mask = mask)
```

Out[34]: <AxesSubplot:>



In [52]:
```
# Combining Numerical and Categorical data.
final_data = pd.concat([cont_data, cat_data], axis = 1)
final_data
```

Out[52]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIr
0	41	1102		1	2		2	94	3	2	4	

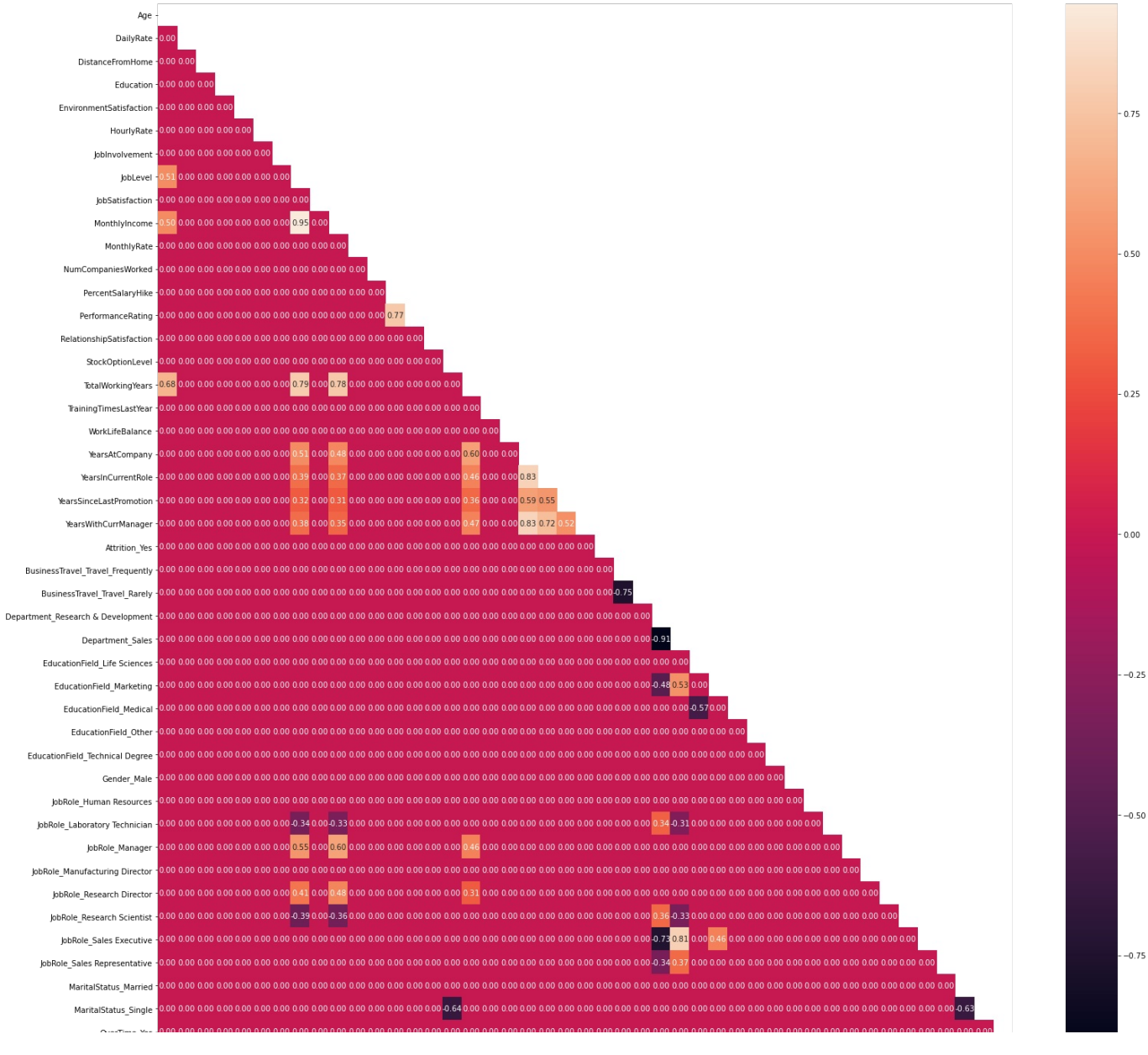
1	49	279	8	1	3	61	2	2	2
2	37	1373	2	2	4	92	2	1	3
3	33	1392	3	4	4	56	3	1	3
4	27	591	2	1	1	40	3	1	2
...	...	...	...	...	...	...	...	...	...
1465	36	884	23	2	3	41	4	2	4
1466	39	613	6	1	4	42	2	3	1
1467	27	155	4	3	2	87	4	2	2
1468	49	1023	2	3	4	63	2	2	2
1469	34	628	8	3	2	82	4	2	3

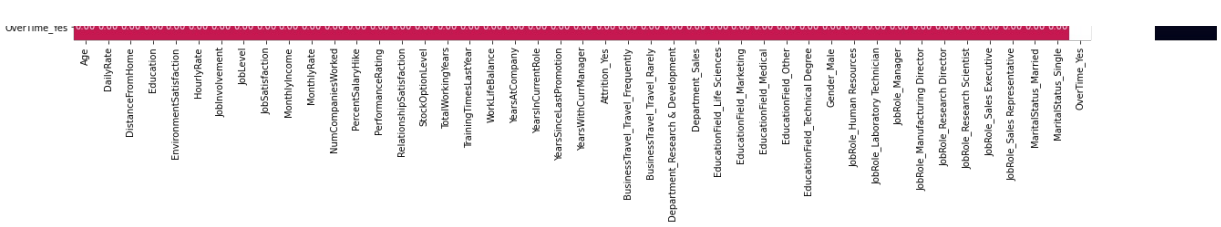
1470 rows × 45 columns

In [36]:

```
# Finding the correlation.
corr = final_data.corr()

# Setting the size of figure.
plt.rcParams['figure.figsize'] = (25, 25)

# Argument Trimming out the values above the main diagonal.
mask = np.triu(corr)

# Setting low correlation value to 0.
corr[(corr.values < 0.3) & (corr.values > -0.3)] = 0

# Plotting the heatmap.
sns.heatmap(corr, annot = True, fmt = '.2f', mask = mask)
```

Out[36]: <AxesSubplot:>





In [37]:
```
final_data.head(10)
```

Out[37]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyInco
0	41	1102	1	2	2	94	3	2	4	59
1	49	279	8	1	3	61	2	2	2	51
2	37	1373	2	2	4	92	2	1	3	20
3	33	1392	3	4	4	56	3	1	3	29
4	27	591	2	1	1	40	3	1	2	34
5	32	1005	2	2	4	79	3	1	4	30
6	59	1324	3	3	3	81	4	1	1	26
7	30	1358	24	1	4	67	3	1	3	26
8	38	216	23	3	4	44	2	3	3	95
9	36	1299	27	3	3	94	3	2	3	52

10 rows × 45 columns

In [ ]:
```
## we will drop the variables which shows the correlation value greater than 0.7
```

In [45]:
```
def correlation(dataset,threshold):
    col_corr=set()
    corr_matrix=dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i,j])>threshold:
                colname=corr_matrix.columns[i]
                col_corr.add(colname)
                return col_corr
```

In [46]:
```
corr_features=correlation(final_data,0.6)
len(set(corr_features))
```

Out[46]: 1

In [47]:
```
corr_features
```

Out[47]: {'MonthlyIncome'}

In [48]:
```
final_data.drop(corr_features,axis=1)
```

Out[48]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyR
0	41	1102	1	2	2	94	3	2	4	19
1	49	279	8	1	3	61	2	2	2	24
2	37	1373	2	2	4	92	2	1	3	2
3	33	1392	3	4	4	56	3	1	3	23
4	27	591	2	1	1	40	3	1	2	16
...	...	...	...	...	...	...	...	...	...	
1465	36	884	23	2	3	41	4	2	4	12
1466	39	613	6	1	4	42	2	3	1	21

1467	27	155	4	3	2	87	4	2	2	5
1468	49	1023	2	3	4	63	2	2	2	13
1469	34	628	8	3	2	82	4	2	3	10

1470 rows × 44 columns

In [54]:
```
x=final_data
x
```

Out[54]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyI
0	41	1102	1	2	2	94	3	2	4	
1	49	279	8	1	3	61	2	2	2	
2	37	1373	2	2	4	92	2	1	3	
3	33	1392	3	4	4	56	3	1	3	
4	27	591	2	1	1	40	3	1	2	
...	...	...	...	...	...	...	...	...	...	
1465	36	884	23	2	3	41	4	2	4	
1466	39	613	6	1	4	42	2	3	1	
1467	27	155	4	3	2	87	4	2	2	
1468	49	1023	2	3	4	63	2	2	2	
1469	34	628	8	3	2	82	4	2	3	

1470 rows × 45 columns

In [60]:
```
x.shape
```

Out[60]: (1470, 45)

In [59]:
```
y=cat_data['Attrition_Yes']
y
```

Out[59]:
```
0       1
1       0
2       1
3       0
4       0
       ..
1465    0
1466    0
1467    0
1468    0
1469    0
Name: Attrition_Yes, Length: 1470, dtype: uint8
```

In [61]:
```
y.shape
```

Out[61]: (1470,)

In [75]:
```

```

In [76]:
```
from sklearn.feature_selection import SelectPercentile
```

In [77]:
```
from sklearn.feature_selection import chi2
```

In [78]:
```
x
```

Out[78]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyI
0	41	1102	1	2	2	94	3	2	4	
1	49	279	8	1	3	61	2	2	2	
2	37	1373	2	2	4	92	2	1	3	
3	33	1392	3	4	4	56	3	1	3	
4	27	591	2	1	1	40	3	1	2	
...	...	...	...	...	...	...	...	...	...	
1465	36	884	23	2	3	41	4	2	4	
1466	39	613	6	1	4	42	2	3	1	
1467	27	155	4	3	2	87	4	2	2	
1468	49	1023	2	3	4	63	2	2	2	
1469	34	628	8	3	2	82	4	2	3	

1470 rows × 45 columns

In [79]:
```
y
```

Out[79]:
```
0       1
1       0
2       1
3       0
4       0
       ..
1465    0
1466    0
1467    0
1468    0
1469    0
Name: Attrition_Yes, Length: 1470, dtype: uint8
```

In [80]:
```
sp=SelectPercentile(score_func=chi2,percentile=80)
```

In [81]:
```
sp_=sp.fit(x,y)
```

In [82]:
```
x.columns
```

Out[82]:
```
Index(['Age', 'DailyRate', 'DistanceFromHome', 'Education',
       'EnvironmentSatisfaction', 'HourlyRate', 'JobInvolvement', 'JobLevel',
       'JobSatisfaction', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager', 'Attrition_Yes',
       'BusinessTravel_Travel_Frequently', 'BusinessTravel_Travel_Rarely',
       'Department_Research & Development', 'Department_Sales',
       'EducationField_Life Sciences', 'EducationField_Marketing',
       'EducationField_Medical', 'EducationField_Other',
       'EducationField_Technical Degree', 'Gender_Male',
       'JobRole_Human Resources', 'JobRole_Laboratory Technician',
       'JobRole_Manager', 'JobRole_Manufacturing Director',
       'JobRole_Research Director', 'JobRole_Research Scientist',
       'JobRole_Sales Executive', 'JobRole_Sales Representative',
       'MaritalStatus_Married', 'MaritalStatus_Single', 'OverTime_Yes'],
      dtype='object')
```

In [84]:
```
cols=sp_.get_support(indices=True)
```

In [85]:
```
features=x.columns[cols]
```

In [86]:
```
features
```

Out[86]:
```
Index(['Age', 'DailyRate', 'DistanceFromHome', 'EnvironmentSatisfaction',
```

```
       'JobInvolvement', 'JobLevel', 'JobSatisfaction', 'MonthlyIncome',
       'MonthlyRate', 'NumCompaniesWorked', 'RelationshipSatisfaction',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager', 'Attrition_Yes',
       'BusinessTravel_Travel_Frequently', 'BusinessTravel_Travel_Rarely',
       'Department_Research & Development', 'Department_Sales',
       'EducationField_Marketing', 'EducationField_Medical',
       'EducationField_Technical Degree', 'JobRole_Human Resources',
       'JobRole_Laboratory Technician', 'JobRole_Manager',
       'JobRole_Manufacturing Director', 'JobRole_Research Director',
       'JobRole_Sales Representative', 'MaritalStatus_Married',
       'MaritalStatus_Single', 'OverTime_Yes'],
      dtype='object')
```

In [102…

```
df_scores=pd.DataFrame({'features':x.columns,'chi2score':sp_.scores_,'pvalue':sp_.pvalues_})
df_scores
```

Out[102…

	features	chi2score	pvalue
0	Age	84.155277	4.574015e-20
1	DailyRate	956.580494	4.923924e-210
2	DistanceFromHome	63.772142	1.396748e-15
3	Education	0.520642	4.705679e-01
4	EnvironmentSatisfaction	6.890594	8.665045e-03
5	HourlyRate	0.431779	5.111173e-01
6	JobInvolvement	4.605616	3.186740e-02
7	JobLevel	24.939242	5.916575e-07
8	JobSatisfaction	7.011947	8.096760e-03
9	MonthlyIncome	118817.349480	0.000000e+00
10	MonthlyRate	1196.633553	3.287933e-262
11	NumCompaniesWorked	6.438654	1.116632e-02
12	PercentSalaryHike	0.235027	6.278213e-01
13	PerformanceRating	0.000506	9.820524e-01
14	RelationshipSatisfaction	1.332333	2.483906e-01
15	StockOptionLevel	25.268826	4.987041e-07
16	TotalWorkingYears	230.277490	5.185931e-52
17	TrainingTimesLastYear	1.558402	2.118994e-01
18	WorkLifeBalance	1.085543	2.974609e-01
19	YearsAtCompany	145.425656	1.733416e-33
20	YearsInCurrentRole	115.075652	7.575177e-27
21	YearsSinceLastPromotion	6.593247	1.023663e-02
22	YearsWithCurrManager	108.374377	2.225158e-25
23	Attrition_Yes	1233.000000	4.107324e-270
24	BusinessTravel_Travel_Frequently	15.816623	6.978671e-05
25	BusinessTravel_Travel_Rarely	1.047857	3.060012e-01
26	Department_Research & Development	3.702916	5.431747e-02
27	Department_Sales	6.694465	9.671267e-03
28	EducationField_Life Sciences	0.924044	3.364153e-01
29	EducationField_Marketing	4.079154	4.341540e-02
30	EducationField_Medical	2.222133	1.360450e-01
31	EducationField_Other	0.444606	5.049078e-01
32	EducationField_Technical Degree	6.435860	1.118390e-02
33	Gender_Male	0.510087	4.751014e-01
34	JobRole_Human Resources	1.859753	1.726534e-01
35	JobRole_Laboratory Technician	11.699495	6.251707e-04
36	JobRole_Manager	9.496136	2.059051e-03
37	JobRole_Manufacturing Director	9.126589	2.519210e-03
38	JobRole_Research Director	10.978010	9.219939e-04
39	JobRole_Research Scientist	0.000152	9.901534e-01

40	JobRole_Sales Executive	0.447333	5.036040e-01
41	JobRole_Sales Representative	34.290268	4.747499e-09
42	MaritalStatus_Married	6.597586	1.021171e-02
43	MaritalStatus_Single	30.771669	2.902446e-08
44	OverTime_Yes	63.845067	1.345990e-15

In [100…
```
import numpy as np
```

In [103…
```
print(cols)
```

```
[ 0  1  2  4  6  7  8  9 10 11 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 29 30 32 34 35 36 37 38 41 42 43 44]
```

In [104…
```
print(features)
```

```
Index(['Age', 'DailyRate', 'DistanceFromHome', 'EnvironmentSatisfaction',
       'JobInvolvement', 'JobLevel', 'JobSatisfaction', 'MonthlyIncome',
       'MonthlyRate', 'NumCompaniesWorked', 'RelationshipSatisfaction',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager', 'Attrition_Yes',
       'BusinessTravel_Travel_Frequently', 'BusinessTravel_Travel_Rarely',
       'Department_Research & Development', 'Department_Sales',
       'EducationField_Marketing', 'EducationField_Medical',
       'EducationField_Technical Degree', 'JobRole_Human Resources',
       'JobRole_Laboratory Technician', 'JobRole_Manager',
       'JobRole_Manufacturing Director', 'JobRole_Research Director',
       'JobRole_Sales Representative', 'MaritalStatus_Married',
       'MaritalStatus_Single', 'OverTime_Yes'],
      dtype='object')
```

In [112…
```
x_new=x[features]
x_new
```

Out[112…

	Age	DailyRate	DistanceFromHome	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	MonthlyRate	Nu
0	41	1102	1	2	3	2	4	5993	19479	
1	49	279	8	3	2	2	2	5130	24907	
2	37	1373	2	4	2	1	3	2090	2396	
3	33	1392	3	4	3	1	3	2909	23159	
4	27	591	2	1	3	1	2	3468	16632	
...	...	...	...	...	...	...	...	...	...	...
1465	36	884	23	3	4	2	4	2571	12290	
1466	39	613	6	4	2	3	1	9991	21457	
1467	27	155	4	2	4	2	2	6142	5174	
1468	49	1023	2	4	2	2	2	5390	13243	
1469	34	628	8	2	4	2	3	4404	10228	

1470 rows × 36 columns

In [113…
```
y
```

Out[113…
```
0       1
1       0
2       1
3       0
4       0
       ..
1465    0
1466    0
1467    0
1468    0
```

```
1469    0
Name: Attrition_Yes, Length: 1470, dtype: uint8
```

In [227…
```
from sklearn.model_selection import train_test_split
```

In [228…
```
from sklearn.preprocessing import StandardScaler
```

In [229…
```
st=StandardScaler()
```

In [230…
```
st.fit_transform(x_new)
```

Out[230…
```
array([[ 0.4463504 ,  0.74252653, -1.01090934, ..., -0.91892141,
         1.45864991,  1.59174553],
       [ 1.32236521, -1.2977746 , -0.14714972, ...,  1.08823234,
        -0.68556546, -0.62824112],
       [ 0.008343  ,  1.41436324, -0.88751511, ..., -0.91892141,
         1.45864991,  1.59174553],
       ...,
       [-1.08667552, -1.60518328, -0.64072665, ...,  1.08823234,
        -0.68556546,  1.59174553],
       [ 1.32236521,  0.54667746, -0.88751511, ...,  1.08823234,
        -0.68556546, -0.62824112],
       [-0.32016256, -0.43256792, -0.14714972, ...,  1.08823234,
        -0.68556546, -0.62824112]])
```

In [341…
```
from sklearn.model_selection import train_test_split,cross_val_score
#importing models
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
```

In [342…
```
x_train,x_test,y_train,y_test=train_test_split(x_new,y,test_size=0.30,random_state=41)
```

In [343…
```
kn=KNeighborsClassifier()
```

In [344…
```
kn.fit(x_train,y_train)
```

Out[344…
```
KNeighborsClassifier()
```

In [345…
```
y_pred=kn.predict(x_test)
```

In [346…
```
y_pred
```

Out[346…
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0], dtype=uint8)
```

In [347…
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [348…
```
accuracy_score(y_pred,y_test)
```

Out[348…
```
0.8390022675736961
```

In [349…
```
confusion_matrix(y_test,y_pred)
```

Out[349…
```
array([[367,  10],
       [ 61,   3]], dtype=int64)
```

In [350…
```
## support vector machine
```

In [351…
```
classification_report(y_test,y_pred)
```

Out[351…
```
'              precision    recall  f1-score   support\n\n           0       0.86      0.97      0.91       377\n
1       0.23      0.05      0.08        64\n\n    accuracy                           0.84       441\n   macro avg
0.54      0.51      0.49       441\nweighted avg       0.77      0.84      0.79       441\n'
```

In [352…
```
sv=SVC()
```

In [353…
```
sv.fit(x_train,y_train)
```

Out[353…
```
SVC()
```

In [354…
```
y_pred=sv.predict(x_test)
y_pred
```

Out[354…
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0], dtype=uint8)
```

In [355…
```
accuracy_score(y_test,y_pred)
```

Out[355…
```
0.854875283446712
```

In [356…
```
confusion_matrix(y_test,y_pred)
```

Out[356…
```
array([[377,   0],
       [ 64,   0]], dtype=int64)
```

In [357…
```
import warnings
warnings.filterwarnings('ignore')
```

In [358…
```
classification_report(y_test,y_pred)
```

Out[358…
```
'              precision    recall  f1-score   support\n\n           0       0.85      1.00      0.92       377\n
1       0.00      0.00      0.00        64\n\n    accuracy                           0.85       441\n   macro avg
0.43      0.50      0.46       441\nweighted avg       0.73      0.85      0.79       441\n'
```

In [359…
```
##DecisionTreeclassifier
```

In [360…
```
dt=DecisionTreeClassifier()
```

In [361…
```
dt.fit(x_train,y_train)
```

Out[361…
```
DecisionTreeClassifier()
```

In [362…
```
y_pred=dt.predict(x_test)
```

In [363…
```
accuracy_score(y_test,y_pred)
```

Out[363…
```
1.0
```

In [364…
```
y_test.shape
```

Out[364…
```
(441,)
```

In [365…
```
confusion_matrix(y_test,y_pred)
```

Out[365…
```
array([[377,   0],
       [  0,  64]], dtype=int64)
```

In [366…
```
classification_report(y_test,y_pred)
```

Out[366…
```
'              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00       377\n
1       1.00      1.00      1.00        64\n\n    accuracy                           1.00       441\n   macro avg
1.00      1.00      1.00       441\nweighted avg       1.00      1.00      1.00       441\n'
```

In [367…
```
## decision tree classifier works well
```

In [368…
```
## RandomForestClassifier
```

In [369…
```
rf=RandomForestClassifier()
```

In [370…
```
rf.fit(x_train,y_train)
```

Out[370…
```
RandomForestClassifier()
```

In [371…
```
y_pred=rf.predict(x_test)
```

In [372…
```
accuracy_score(y_test,y_pred)
```

Out[372…
```
1.0
```

In [373…
```
confusion_matrix(y_test,y_pred)
```

Out[373…
```
array([[377,   0],
       [  0,  64]], dtype=int64)
```

In [374…
```
classification_report(y_test,y_pred)
```

Out[374…
```
'              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00       377\n
1       1.00      1.00      1.00        64\n\n    accuracy                           1.00       441\n   macro avg
1.00      1.00      1.00       441\nweighted avg       1.00      1.00      1.00       441\n'
```

In [375…
```
## GradientBoostingClassifier
```

In [376…
```
gb=GradientBoostingClassifier()
```

In [377…
```
gb.fit(x_train,y_train)
```

Out[377…
```
GradientBoostingClassifier()
```

In [378…
```
y_pred=gb.predict(x_test)
```

In [379…
```
accuracy_score(y_pred,y_test)
```

Out[379…
```
1.0
```

In [380…
```
confusion_matrix(y_test,y_pred)
```

Out[380…
```
array([[377,   0],
       [  0,  64]], dtype=int64)
```

In [381…
```
classification_report(y_test,y_pred)
```

Out[381…
```
'              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00       377\n
1       1.00      1.00      1.00        64\n\n    accuracy                           1.00       441\n   macro avg
1.00      1.00      1.00       441\nweighted avg       1.00      1.00      1.00       441\n'
```

In [382…
```
## since three models are overfitting we will consider the best one i.e support vector classifier() for hyperpara
```

In [383…
```
## support vector classifier
```

In [384…
```
## HYPER PARAMETER TUNING for svc
```

In [385…
```
from sklearn.model_selection import GridSearchCV
```

In [386…
```
params={'C':[0.01,2,3,4,5,6,7,10],'gamma':[0.1,0.2,0.3,0.4,0.5,0.6]}
```

In [387…
```
g=GridSearchCV(SVC(),params,cv=10)
```

In [388…
```
g.fit(x_train,y_train)
```

Out[388…
```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': [0.01, 2, 3, 4, 5, 6, 7, 10],
                         'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]})
```

In [389…
```
print(g.best_params_)
```

```
{'C': 0.01, 'gamma': 0.1}
```

In [390…
```
s=SVC(C=0.01,gamma=0.1)
```

In [391…
```
s.fit(x_train,y_train)
```

Out[391…
```
SVC(C=0.01, gamma=0.1)
```

In [392…
```
y_pred=s.predict(x_test)
```

In [393…
```
accuracy_score(y_test,y_pred)
```

Out[393…
```
0.854875283446712
```

In [394…
```
##
```

In [395…
```
##
```

In [396…
```
s=SVC(C=0.01,gamma=0.1)
```

In [397…
```
s.fit(x_train,y_train)
```

Out[397…
```
SVC(C=0.01, gamma=0.1)
```

In [398…
```
y_pred=s.predict(x_test)
```

In [399…
```
accuracy_score(y_test,y_pred)
```

Out[399…
```
0.854875283446712
```

In [400…
```
confusion_matrix(y_pred,y_test)
```

Out[400…
```
array([[377,  64],
       [  0,   0]], dtype=int64)
```

In [401…
```
classification_report(y_test,y_pred)
```

Out[401…
```
'              precision    recall  f1-score   support\n\n           0       0.85      1.00      0.92       377\n
1       0.00      0.00      0.00        64\n\n    accuracy                           0.85       441\n   macro avg
0.43      0.50      0.46       441\nweighted avg       0.73      0.85      0.79       441\n'
```

In [402…
```
## finalizin the RandomForestClassifier
```

In [403…
```
import numpy as np
```

In [404…
```
a=np.array(y_test)
```

In [405…
```
a
```

Out[405…
```
array([1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0], dtype=uint8)
```

In [406…
```
pred=np.array(rf.predict(x_test))
```

In [407…
```
pred
```

Out[407…
```
array([1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0], dtype=uint8)
```

In [408…
```
df_con=pd.DataFrame({'true':a,'pred':pred},index=range(len(a)))
```

In [409…
```
df_con
```

Out[409…

	true	pred
0	1	1
1	0	0
2	0	0
3	1	1
4	0	0
...	...	...

436	0	0
437	0	0
438	0	0
439	0	0
440	0	0

441 rows × 2 columns

In [410…]
```
## saving the model
```

In [411…]
```
import pickle
```

In [412…]
```
filename='IBMHRANALYTICS.pkl'
```

In [413…]
```
pickle.dump(rf,open(filename,'wb'))
```

In [ ]:
```

```