In [1]:
```python
from IPython.display import Image
Image(filename='Prblm_statement_img.png')
```

Out[1]:

<u>Problem Statement:</u> Within the context of human resources (HR), attrition is a reduction in the workforce caused by retirement or resignation. This is a serious problem faced by several organizations around the world as attrition is economically damaging to the organizations as the replacement employees have to be hired at a cost and trained again at a cost. High Rates of Attrition also damages the brand value of the company.

Now the Dataset belongs to a very fast-growing company. This company has witnessed several employees leaving the company in the last 3 years. The company's HR team has always been reactive to attrition but now the team wants to be proactive and wished to predict attrition of employees using the data they have in hand.

The goal here is to predict whether an employee will leave the company based upon the various variables given in the dataset.

Working with Data

Data has been split into two groups and provided in the module:

- training set
- test set

The training set is used to build your machine learning model. For the training set, we provide the attrition details of an employee.

The test set should be used to see how well your model performs on unseen data. For the test set, it is your job to predict the attrition value of an employee.

In [2]:
```python
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
import warnings
warnings.filterwarnings("ignore")
pd.set_option('display.max_rows',None)
pd.set_option('display.max_columns',None)
pd.set_option('display.width',None)
```

In [3]:
```python
dataset_train = pd.read_csv("Train_Dataset.csv")
dataset_test = pd.read_csv("Test_Dataset.csv")
```

In [4]:
```python
dataset_train.head(5)
```

Out[4]:

	EmployeeID	Attrition	Age	TravelProfile	Department	HomeToWork	EducationField	Gender	HourlnWeek	Ir
0	5110001.0	0.0	35.0	Rarely	Analytics	5.0	CA	Male	69.0	
1	5110002.0	1.0	32.0	Yes	Sales	5.0	Statistics	Female	62.0	
2	5110003.0	0.0	31.0	Rarely	Analytics	5.0	Statistics	F	45.0	
3	5110004.0	0.0	34.0	Yes	Sales	10.0	Statistics	Female	32.0	
4	5110005.0	0.0	37.0	No	Analytics	27.0	Statistics	Female	49.0	

In [5]:
```python
dataset_test.head(5)
```

Out[5]:

	EmployeeID	Age	TravelProfile	Department	HomeToWork	EducationField	Gender	HourlnWeek	Involvement
0	6110001	18.0	No	NaN	9.0	CA	Male	80.0	3
1	6110002	20.0	Rarely	Analytics	28.0	Statistics	Female	59.0	1
2	6110003	50.0	Rarely	Analytics	19.0	CA	Female	76.0	3
3	6110004	32.0	Rarely	Sales	23.0	Statistics	Female	73.0	5
4	6110005	39.0	Rarely	Analytics	7.0	CA	Male	42.0	4

In [6]:
```python
dataset_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7810 entries, 0 to 7809
Data columns (total 22 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   EmployeeID          5180 non-null   float64
 1   Attrition           5180 non-null   float64
 2   Age                 4864 non-null   float64
 3   TravelProfile       5180 non-null   object 
 4   Department          5056 non-null   object 
 5   HomeToWork          4925 non-null   float64
 6   EducationField      5180 non-null   object 
 7   Gender              5134 non-null   object 
 8   HourlnWeek          4893 non-null   float64
 9   Involvement         5180 non-null   float64
 10  WorkLifeBalance     5180 non-null   float64
 11  Designation         5142 non-null   object 
 12  JobSatisfaction     5180 non-null   float64
 13  ESOPs               5180 non-null   float64
 14  NumCompaniesWorked  5180 non-null   float64
 15  OverTime            5180 non-null   float64
 16  SalaryHikelastYear  5011 non-null   float64
 17  WorkExperience      4993 non-null   float64
 18  LastPromotion       5110 non-null   float64
 19  CurrentProfile      4869 non-null   float64
 20  MaritalStatus       5180 non-null   object 
 21  MonthlyIncome       5087 non-null   float64
dtypes: float64(16), object(6)
memory usage: 1.3+ MB
```

In [7]:
```python
dataset_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2630 entries, 0 to 2629
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   EmployeeID          2630 non-null   int64  
 1   Age                 2488 non-null   float64
 2   TravelProfile       2630 non-null   object 
 3   Department          2572 non-null   object 
 4   HomeToWork          2504 non-null   float64
 5   EducationField      2630 non-null   object 
 6   Gender              2600 non-null   object 
 7   HourlnWeek          2494 non-null   float64
 8   Involvement         2630 non-null   int64  
 9   WorkLifeBalance     2630 non-null   int64  
 10  Designation         2600 non-null   object 
 11  JobSatisfaction     2630 non-null   int64  
 12  ESOPs               2630 non-null   int64  
 13  NumCompaniesWorked  2630 non-null   int64  
 14  OverTime            2630 non-null   int64  
 15  SalaryHikelastYear  2536 non-null   float64
 16  WorkExperience      2508 non-null   float64
 17  LastPromotion       2573 non-null   float64
 18  CurrentProfile      2496 non-null   float64
 19  MaritalStatus       2630 non-null   object 
 20  MonthlyIncome       2597 non-null   float64
dtypes: float64(8), int64(7), object(6)
memory usage: 431.6+ KB
```

In [ ]:

In [8]:
```
dataset_train.tail(5)
```

Out[8]:

	EmployeeID	Attrition	Age	TravelProfile	Department	HomeToWork	EducationField	Gender	HourlnWeek
7805	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7806	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7807	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7808	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7809	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [9]:
```
dataset_train.shape
```

Out[9]:
```
(7810, 22)
```

In [10]:
```
dataset_test.shape
```

Out[10]:
```
(2630, 21)
```

In [11]:
```
dataset_train.isnull().sum()
```

Out[11]:
```
EmployeeID            2630
Attrition             2630
Age                   2946
TravelProfile         2630
Department            2754
HomeToWork            2885
EducationField        2630
Gender                2676
HourlnWeek            2917
Involvement           2630
WorkLifeBalance       2630
Designation           2668
JobSatisfaction       2630
ESOPs                 2630
NumCompaniesWorked    2630
OverTime              2630
SalaryHikelastYear    2799
WorkExperience        2817
LastPromotion         2700
CurrentProfile        2941
MaritalStatus         2630
MonthlyIncome         2723
dtype: int64
```

In [12]:
```
#droping null row
dataset_train.dropna(axis=0, subset=['EmployeeID'],inplace=True)
```

In [13]:
```
dataset_train.shape
```

Out[13]:
```
(5180, 22)
```

In [14]:
```
dataset_train.isnull().sum()/len(dataset_train)
```

Out[14]:
```
EmployeeID            0.000000
Attrition             0.000000
Age                   0.061004
TravelProfile         0.000000
```

```
Department           0.023938
HomeToWork           0.049228
EducationField       0.000000
Gender               0.008880
HourlnWeek           0.055405
Involvement          0.000000
WorkLifeBalance      0.000000
Designation          0.007336
JobSatisfaction      0.000000
ESOPs                0.000000
NumCompaniesWorked   0.000000
OverTime             0.000000
SalaryHikelastYear   0.032625
WorkExperience       0.036100
LastPromotion        0.013514
CurrentProfile       0.060039
MaritalStatus        0.000000
MonthlyIncome        0.017954
dtype: float64
```

In [15]:

```
#lets merge train and test data with adding flag to each dataset to indicate as train an
dataset_train['flag']='train'
dataset_test['flag']='test'
```

In [16]:

```
dataset_test.head()
```

Out[16]:

	EmployeeID	Age	TravelProfile	Department	HomeToWork	EducationField	Gender	HourlnWeek	Involvement
0	6110001	18.0	No	NaN	9.0	CA	Male	80.0	3
1	6110002	20.0	Rarely	Analytics	28.0	Statistics	Female	59.0	1
2	6110003	50.0	Rarely	Analytics	19.0	CA	Female	76.0	3
3	6110004	32.0	Rarely	Sales	23.0	Statistics	Female	73.0	5
4	6110005	39.0	Rarely	Analytics	7.0	CA	Male	42.0	4

In [17]:

```
dataset=pd.concat([dataset_train, dataset_test], axis=0)
```

In [18]:

```
dataset.shape
```

Out[18]:

```
(7810, 23)
```

In [19]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7810 entries, 0 to 2629
Data columns (total 23 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   EmployeeID          7810 non-null   float64
 1   Attrition           5180 non-null   float64
 2   Age                 7352 non-null   float64
 3   TravelProfile       7810 non-null   object
 4   Department          7628 non-null   object
 5   HomeToWork          7429 non-null   float64
 6   EducationField      7810 non-null   object
 7   Gender              7734 non-null   object
 8   HourlnWeek          7387 non-null   float64
 9   Involvement         7810 non-null   float64
 10  WorkLifeBalance     7810 non-null   float64
 11  Designation         7742 non-null   object
```

```
 12   JobSatisfaction     7810 non-null   float64
 13   ESOPs               7810 non-null   float64
 14   NumCompaniesWorked  7810 non-null   float64
 15   OverTime            7810 non-null   float64
 16   SalaryHikelastYear  7547 non-null   float64
 17   WorkExperience      7501 non-null   float64
 18   LastPromotion       7683 non-null   float64
 19   CurrentProfile      7365 non-null   float64
 20   MaritalStatus       7810 non-null   object
 21   MonthlyIncome       7684 non-null   float64
 22   flag                7810 non-null   object
dtypes: float64(16), object(7)
memory usage: 1.4+ MB
```

In [20]:
```
dataset.isnull().sum()/len(dataset)
```

Out[20]:
```
EmployeeID            0.000000
Attrition             0.336748
Age                   0.058643
TravelProfile         0.000000
Department            0.023303
HomeToWork            0.048784
EducationField        0.000000
Gender                0.009731
HourlnWeek            0.054161
Involvement           0.000000
WorkLifeBalance       0.000000
Designation           0.008707
JobSatisfaction       0.000000
ESOPs                 0.000000
NumCompaniesWorked    0.000000
OverTime              0.000000
SalaryHikelastYear    0.033675
WorkExperience        0.039565
LastPromotion         0.016261
CurrentProfile        0.056978
MaritalStatus         0.000000
MonthlyIncome         0.016133
flag                  0.000000
dtype: float64
```

HANDLING MISSING VALUES

In [21]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7810 entries, 0 to 2629
Data columns (total 23 columns):
 #    Column              Non-Null Count  Dtype
---   ------              --------------  -----
 0    EmployeeID          7810 non-null   float64
 1    Attrition           5180 non-null   float64
 2    Age                 7352 non-null   float64
 3    TravelProfile       7810 non-null   object
 4    Department          7628 non-null   object
 5    HomeToWork          7429 non-null   float64
 6    EducationField      7810 non-null   object
 7    Gender              7734 non-null   object
 8    HourlnWeek          7387 non-null   float64
 9    Involvement         7810 non-null   float64
 10   WorkLifeBalance     7810 non-null   float64
 11   Designation         7742 non-null   object
 12   JobSatisfaction     7810 non-null   float64
 13   ESOPs               7810 non-null   float64
```

```
 14  NumCompaniesWorked  7810 non-null   float64
 15  OverTime            7810 non-null   float64
 16  SalaryHikelastYear  7547 non-null   float64
 17  WorkExperience      7501 non-null   float64
 18  LastPromotion       7683 non-null   float64
 19  CurrentProfile      7365 non-null   float64
 20  MaritalStatus       7810 non-null   object
 21  MonthlyIncome       7684 non-null   float64
 22  flag                7810 non-null   object
dtypes: float64(16), object(7)
memory usage: 1.4+ MB
```

In [22]:
```
# Gender - object
# MaritalStatus - object
# TravelProfile  - object
# Department - object
# EducationField -   object
# Designation  -object
# Age - float64
# HomeToWork - float64
# Involvement - float64
# WorkLifeBalance - float64
# JobSatisfaction - float64
# ESOPs-  float64
# NumCompaniesWorked -float64
# OverTime -  float64
# SalaryHikelastYear-  float64
# WorkExperience-  float64
# LastPromotion   -  float64
# CurrentProfile -  float64
# MonthlyIncome  - float64
```

In [23]:
```
dataset['Gender'].value_counts()
```

Out[23]:
```
Gender
Male      4668
Female    2020
F         1046
Name: count, dtype: int64
```

In [24]:
```
# CONVERTING F TO FEMALE
dataset['Gender']=np.where(dataset['Gender']=='F','Female',dataset['Gender'])
```

In [25]:
```
dataset['Gender'].value_counts()
```

Out[25]:
```
Gender
Male      4668
Female    3066
Name: count, dtype: int64
```

In [26]:
```
# CONVERTING M TO Married
dataset['MaritalStatus']=np.where(dataset['MaritalStatus']=='M','Married',dataset['Marit
```

In [27]:
```
#REPLACING NULL WITH MALE BCZ MOST FREQUENT USED IN OUR DATASAET.
dataset['Gender'] =dataset['Gender'].fillna('Male')
```

In [28]:
```
dataset['Department'].value_counts()
```

Out[28]:
```
Department
Analytics    4894
Sales        2407
Marketing     327
Name: count, dtype: int64
```

In [29]:
```
#REPLACING NULL WITH Analytics BCZ MOST FREQUENT USED IN OUR DATASAET.
```

```python
dataset['Department'] =dataset['Department'].fillna('Analytics')
```

In [30]:
```python
dataset['Designation'].value_counts()
```

Out[30]:
```
Designation
Executive         3065
Manager           2676
Senior Manager    1154
AVP                507
VP                 340
Name: count, dtype: int64
```

In [31]:
```python
#REPLACING NULL WITH Executive BCZ MOST FREQUENT USED IN OUR DATASAET.
dataset['Designation'] =dataset['Designation'].fillna('Executive')
```

In [32]:
```python
dataset.describe(exclude=["object"])
```

Out[32]:

	EmployeeID	Attrition	Age	HomeToWork	HourlnWeek	Involvement	WorkLifeBalance	JobSa
count	7.810000e+03	5180.000000	7352.000000	7429.000000	7387.000000	7810.000000	7810.000000	78
mean	5.448909e+06	0.278958	37.215860	11.215507	57.940436	3.230986	3.031754	
std	4.720273e+05	0.448530	9.286258	8.590705	13.076675	0.876355	1.412770	
min	5.110001e+06	0.000000	18.000000	1.000000	10.000000	1.000000	1.000000	
25%	5.111953e+06	0.000000	30.000000	5.000000	49.000000	3.000000	2.000000	
50%	5.113906e+06	0.000000	36.000000	9.000000	59.000000	3.000000	3.000000	
75%	6.110678e+06	1.000000	43.000000	16.000000	67.000000	4.000000	4.000000	
max	6.112630e+06	1.000000	61.000000	123.000000	110.000000	5.000000	5.000000	

In [33]:
```python
#MOST OF THE NUMERICAL FEATURE CONTAINS OUTLIER SO FOR SAFER SIDE FILLING NULL VALUE WIT
dataset['Age'] = dataset['Age'].fillna(dataset['Age'].median())
dataset['HomeToWork'] = dataset['HomeToWork'].fillna(dataset['HomeToWork'].median())
dataset['HourlnWeek'] = dataset['HourlnWeek'].fillna(dataset['HourlnWeek'].median())
dataset['SalaryHikelastYear'] = dataset['SalaryHikelastYear'].fillna(dataset['SalaryHike
dataset['WorkExperience'] = dataset['WorkExperience'].fillna(dataset['WorkExperience'].m
dataset['LastPromotion'] = dataset['LastPromotion'].fillna(dataset['LastPromotion'].medi
dataset['CurrentProfile'] = dataset['CurrentProfile'].fillna(dataset['CurrentProfile'].m
dataset['MonthlyIncome'] = dataset['MonthlyIncome'].fillna(dataset['MonthlyIncome'].medi
```

In [34]:
```python
dataset.isnull().sum()/len(dataset)
```

Out[34]:
```
EmployeeID            0.000000
Attrition             0.336748
Age                   0.000000
TravelProfile         0.000000
Department            0.000000
HomeToWork            0.000000
EducationField        0.000000
Gender                0.000000
HourlnWeek            0.000000
Involvement           0.000000
WorkLifeBalance       0.000000
Designation           0.000000
JobSatisfaction       0.000000
ESOPs                 0.000000
NumCompaniesWorked    0.000000
OverTime              0.000000
SalaryHikelastYear    0.000000
WorkExperience        0.000000
LastPromotion         0.000000
```

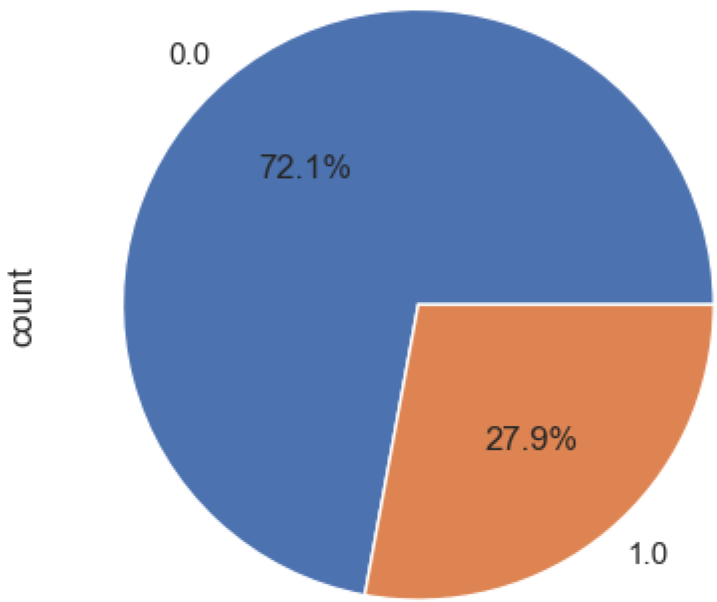
```
CurrentProfile        0.000000
MaritalStatus         0.000000
MonthlyIncome         0.000000
flag                  0.000000
dtype: float64
```

In [35]:
```python
#Droping unwanted feature
dataset.drop(columns=['EmployeeID'],inplace=True)
```

EDA

In [36]:
```python
dataset['Attrition'].value_counts().plot(kind='pie', autopct='%1.1f%%')
```

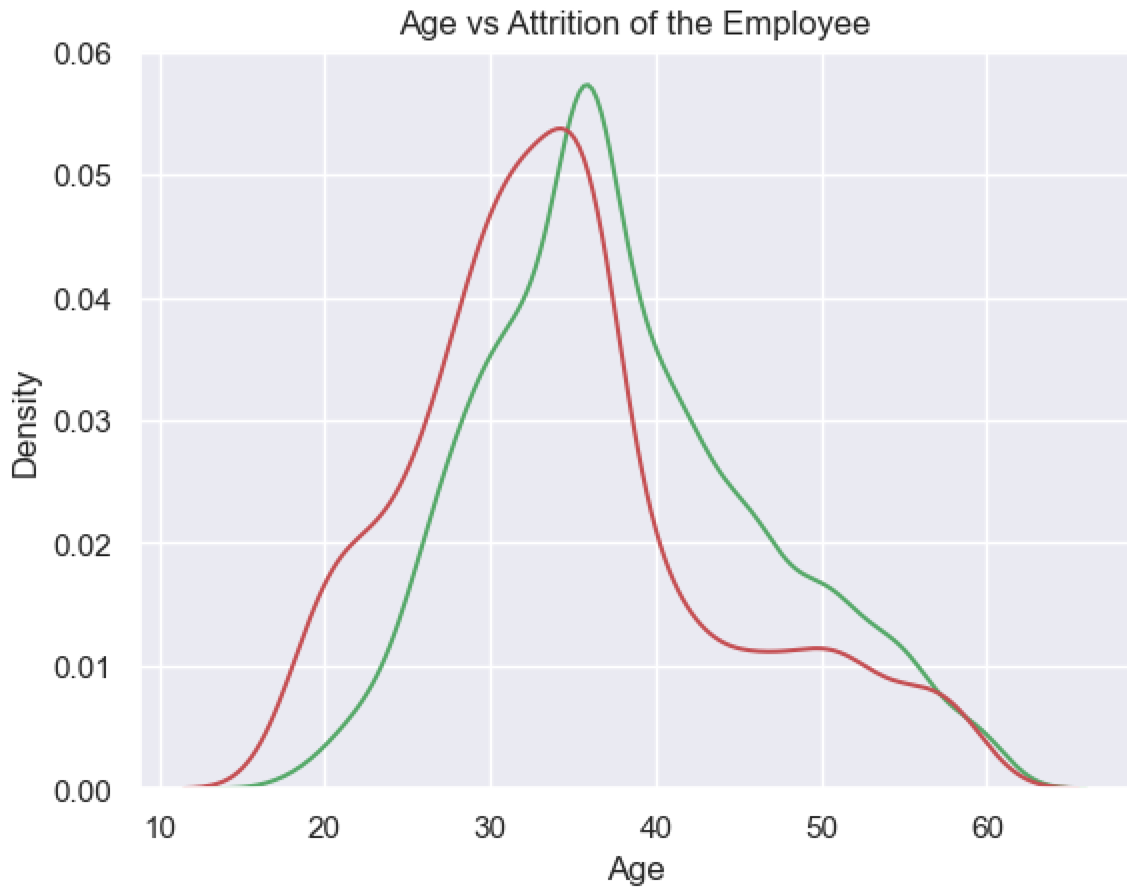
Out[36]: <Axes: ylabel='count'>



Above pie chart shows that 28% approx attrition rate in a company

In [37]:
```python
# To analyse - Age vs Attrition
sns.distplot(dataset[dataset['Attrition']==0]['Age'],hist=False, label='No',color='g')
sns.distplot(dataset[dataset['Attrition']==1]['Age'],hist=False,color='r'
             ,label='Yes')
plt.title("Age vs Attrition of the Employee")
```

Out[37]: Text(0.5, 1.0, 'Age vs Attrition of the Employee')



Above chart shows that avg age is between 25-40 who leave company

In [38]:
```
department = pd.crosstab(dataset['Department'], dataset['Attrition'])
department['Total'] = department[0] + department[1]
department['Pecentage'] = department[1]/department['Total']*100
department
```

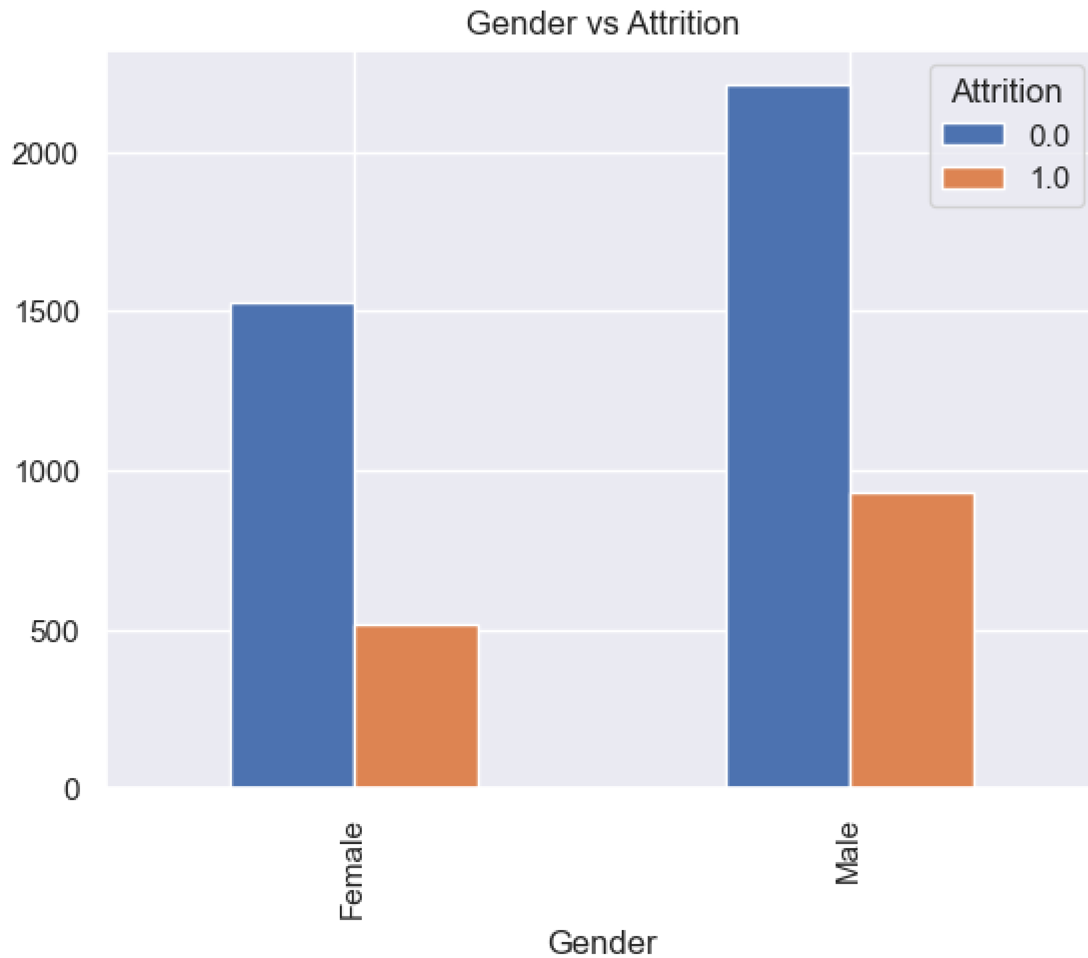
Out[38]:

Attrition Department	0.0	1.0	Total	Pecentage
Analytics	2500	843	3343	25.216871
Marketing	155	67	222	30.180180
Sales	1080	535	1615	33.126935

Sales department has highest Attrition rate

In [39]:
```
# Relation between Gender vs Attrition
gender_wise = pd.crosstab(dataset['Gender'], dataset['Attrition'])
gender_wise.plot(kind='bar')
plt.title("Gender vs Attrition")
```

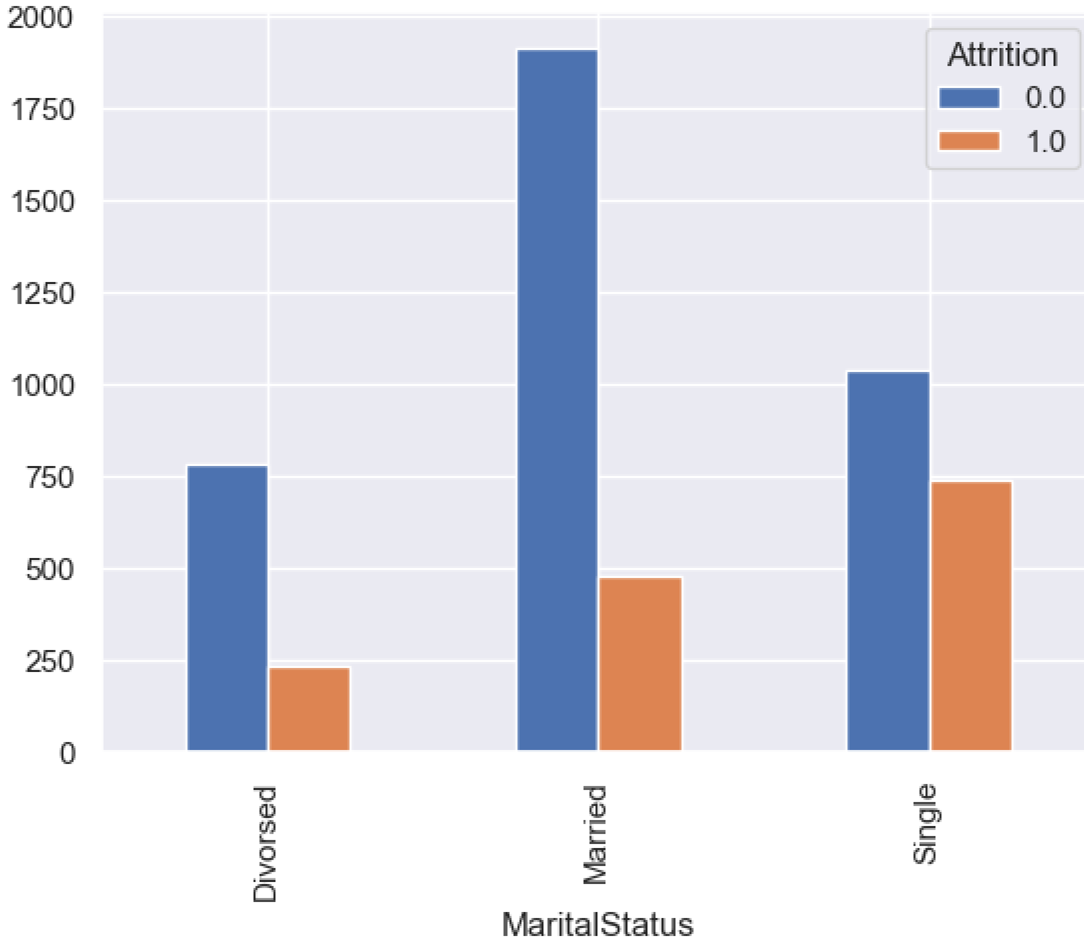
Out[39]:
```
Text(0.5, 1.0, 'Gender vs Attrition')
```



Male has highest Attrition rate

In [40]:
```
pd.crosstab(dataset.MaritalStatus, dataset.Attrition).plot(kind='bar')
```

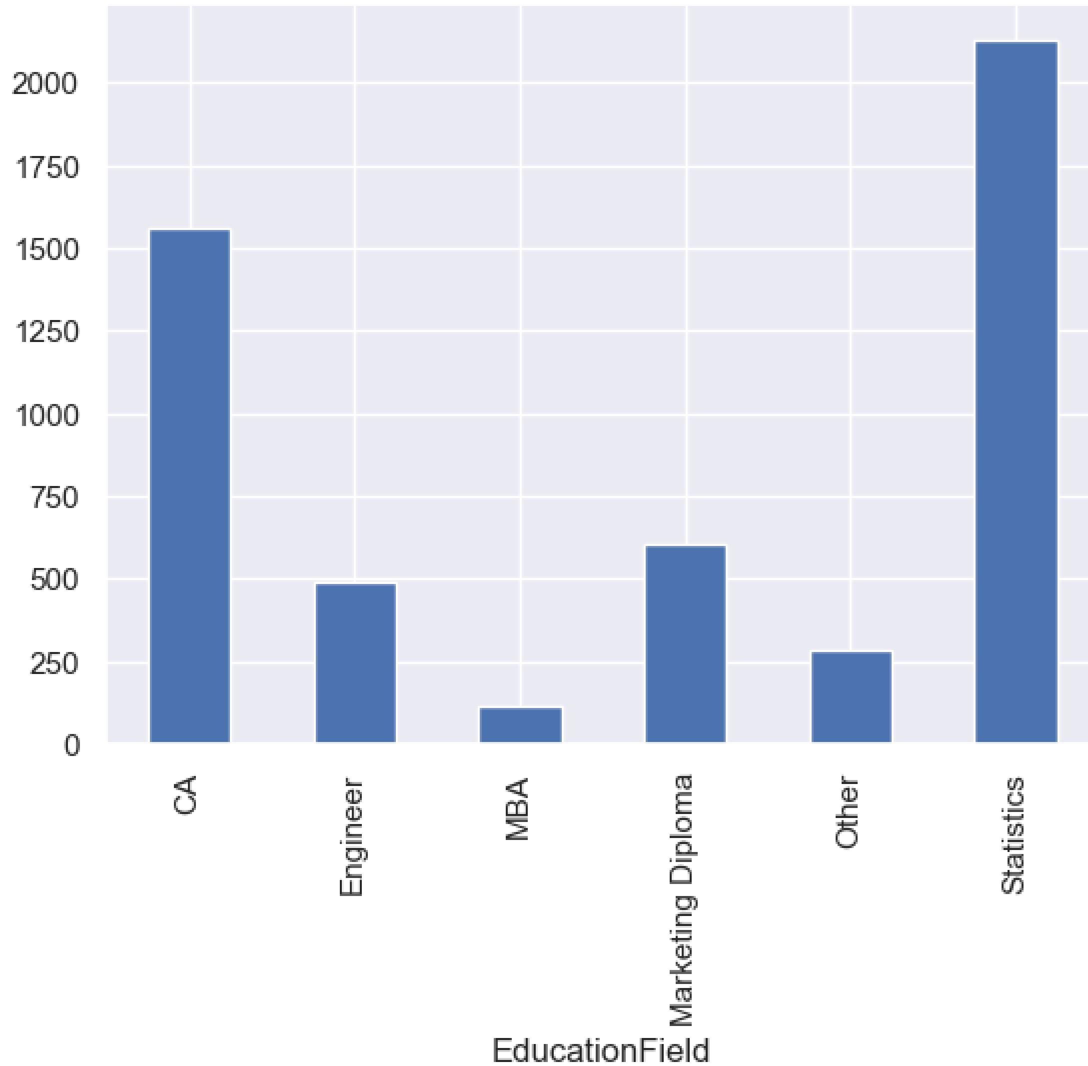
Out[40]:
```
<Axes: xlabel='MaritalStatus'>
```



Single has highest Attrition rate

In [41]:
```
dataset.groupby(['EducationField'])['Attrition'].count().plot(kind='bar')
```

Out[41]:
```
<Axes: xlabel='EducationField'>
```



handling char/object feature

In [42]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7810 entries, 0 to 2629
Data columns (total 22 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Attrition            5180 non-null   float64
 1   Age                  7810 non-null   float64
 2   TravelProfile        7810 non-null   object 
 3   Department           7810 non-null   object 
 4   HomeToWork           7810 non-null   float64
 5   EducationField       7810 non-null   object 
 6   Gender               7810 non-null   object 
 7   HourlnWeek           7810 non-null   float64
 8   Involvement          7810 non-null   float64
 9   WorkLifeBalance      7810 non-null   float64
 10  Designation          7810 non-null   object 
 11  JobSatisfaction      7810 non-null   float64
 12  ESOPs                7810 non-null   float64
 13  NumCompaniesWorked   7810 non-null   float64
 14  OverTime             7810 non-null   float64
 15  SalaryHikelastYear   7810 non-null   float64
 16  WorkExperience       7810 non-null   float64
 17  LastPromotion        7810 non-null   float64
 18  CurrentProfile       7810 non-null   float64
```

```
 19  MaritalStatus       7810 non-null   object
 20  MonthlyIncome       7810 non-null   float64
 21  flag                7810 non-null   object
dtypes: float64(15), object(7)
memory usage: 1.4+ MB
```

In [43]:
```
dataset['TravelProfile'].value_counts()
```

Out[43]:
```
TravelProfile
Rarely    5489
Yes       1580
No         741
Name: count, dtype: int64
```

In [44]:
```
dataset['Department'].value_counts()
```

Out[44]:
```
Department
Analytics    5076
Sales        2407
Marketing     327
Name: count, dtype: int64
```

In [45]:
```
dataset['EducationField'].value_counts()
```

Out[45]:
```
EducationField
Statistics            3169
CA                    2417
Marketing Diploma      894
Engineer               750
Other                  429
MBA                    151
Name: count, dtype: int64
```

In [46]:
```
dataset['Designation'].value_counts()
```

Out[46]:
```
Designation
Executive         3133
Manager           2676
Senior Manager    1154
AVP                507
VP                 340
Name: count, dtype: int64
```

In [47]:
```
dataset['MaritalStatus'].value_counts()
```

Out[47]:
```
MaritalStatus
Married     3608
Single      2709
Divorsed    1493
Name: count, dtype: int64
```

Above feature has more than 2 values so OHE is best to use

In [48]:
```
dataset['Gender'].value_counts()
```

Out[48]:
```
Gender
Male      4744
Female    3066
Name: count, dtype: int64
```

In [49]:
```
#One hot encoder
dataset = pd.get_dummies(dataset, columns=['TravelProfile'],drop_first=True,dtype='int64
dataset = pd.get_dummies(dataset, columns=['Department'],drop_first=True,dtype='int64')
dataset = pd.get_dummies(dataset, columns=['EducationField'],drop_first=True,dtype='int6
dataset = pd.get_dummies(dataset, columns=['Designation'],drop_first=True,dtype='int64')
dataset = pd.get_dummies(dataset, columns=['MaritalStatus'],drop_first=True,dtype='int64
```

In [50]:
```python
dataset['Gender'] = dataset['Gender'].astype('category')
dataset['Gender'] = dataset['Gender'].cat.codes
```

In [51]:
```python
dataset.sample(2)
```

Out[51]:

	Attrition	Age	HomeToWork	Gender	HourlnWeek	Involvement	WorkLifeBalance	JobSatisfaction	ESOPs
2502	NaN	27.0	9.0	0	56.0	1.0	1.0	3.0	0.0
4856	1.0	53.0	5.0	0	59.0	5.0	5.0	2.0	0.0

In [52]:
```python
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7810 entries, 0 to 2629
Data columns (total 32 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Attrition                         5180 non-null   float64
 1   Age                               7810 non-null   float64
 2   HomeToWork                        7810 non-null   float64
 3   Gender                            7810 non-null   int8
 4   HourlnWeek                        7810 non-null   float64
 5   Involvement                       7810 non-null   float64
 6   WorkLifeBalance                   7810 non-null   float64
 7   JobSatisfaction                   7810 non-null   float64
 8   ESOPs                             7810 non-null   float64
 9   NumCompaniesWorked                7810 non-null   float64
 10  OverTime                          7810 non-null   float64
 11  SalaryHikelastYear                7810 non-null   float64
 12  WorkExperience                    7810 non-null   float64
 13  LastPromotion                     7810 non-null   float64
 14  CurrentProfile                    7810 non-null   float64
 15  MonthlyIncome                     7810 non-null   float64
 16  flag                              7810 non-null   object
 17  TravelProfile_Rarely              7810 non-null   int64
 18  TravelProfile_Yes                 7810 non-null   int64
 19  Department_Marketing              7810 non-null   int64
 20  Department_Sales                  7810 non-null   int64
 21  EducationField_Engineer           7810 non-null   int64
 22  EducationField_MBA                7810 non-null   int64
 23  EducationField_Marketing Diploma  7810 non-null   int64
 24  EducationField_Other              7810 non-null   int64
 25  EducationField_Statistics         7810 non-null   int64
 26  Designation_Executive             7810 non-null   int64
 27  Designation_Manager               7810 non-null   int64
 28  Designation_Senior Manager        7810 non-null   int64
 29  Designation_VP                    7810 non-null   int64
 30  MaritalStatus_Married             7810 non-null   int64
 31  MaritalStatus_Single              7810 non-null   int64
dtypes: float64(15), int64(15), int8(1), object(1)
memory usage: 1.9+ MB
```

In [53]:
```python
dataset.describe()
```

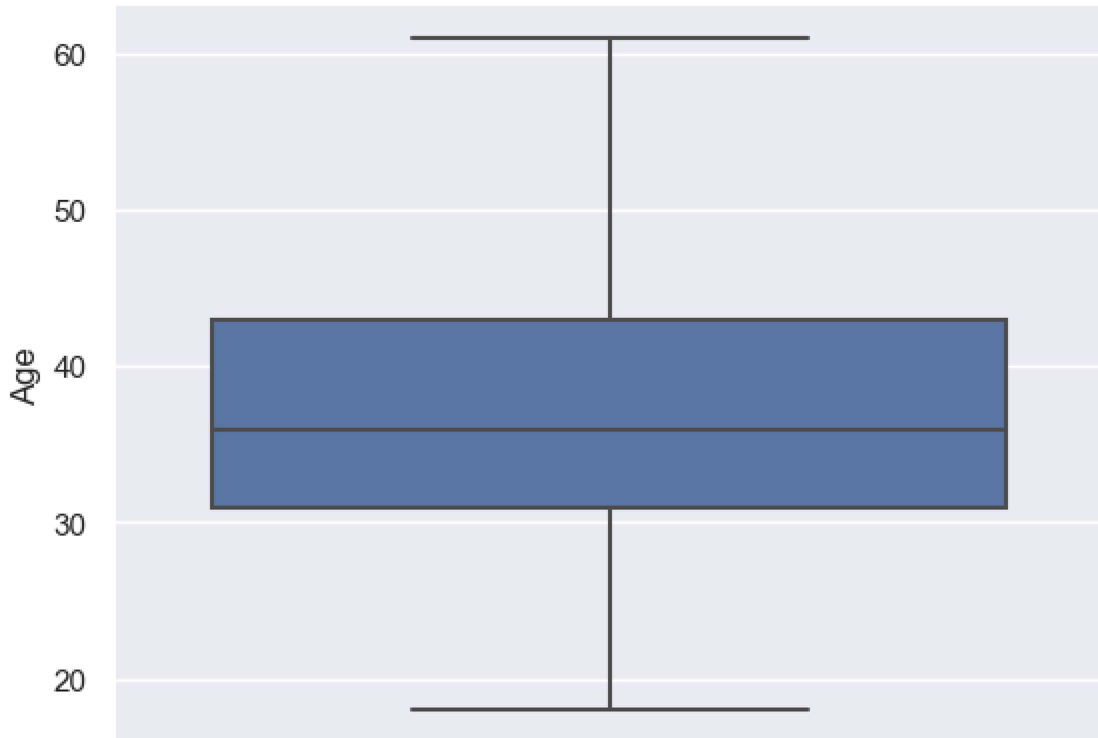
Out[53]:

	Attrition	Age	HomeToWork	Gender	HourlnWeek	Involvement	WorkLifeBalance	JobSati
count	5180.000000	7810.000000	7810.000000	7810.000000	7810.000000	7810.000000	7810.000000	781
mean	0.278958	37.144558	11.107426	0.607426	57.997823	3.230986	3.031754	
std	0.448530	9.014351	8.392098	0.488354	12.719835	0.876355	1.412770	

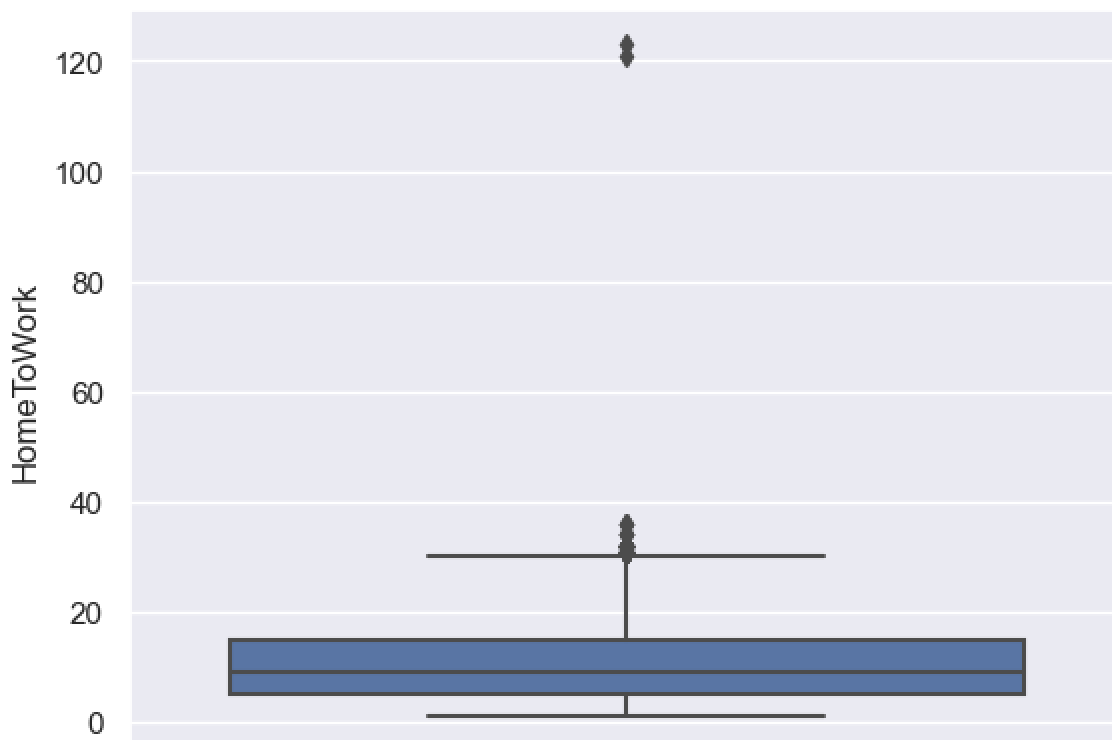
min	0.000000	18.000000	1.000000	0.000000	10.000000	1.000000	1.000000
25%	0.000000	31.000000	5.000000	0.000000	50.000000	3.000000	2.000000
50%	0.000000	36.000000	9.000000	1.000000	59.000000	3.000000	3.000000
75%	1.000000	43.000000	15.000000	1.000000	67.000000	4.000000	4.000000
max	1.000000	61.000000	123.000000	1.000000	110.000000	5.000000	5.000000

Detect Outlier using Boxplot method

In [54]:
```
def boxplots(col):
    sns.boxplot(y=col, data=dataset)
    plt.show()

for i in list(dataset.select_dtypes(exclude=["object"]).columns)[0:]:
    boxplots(i)
```

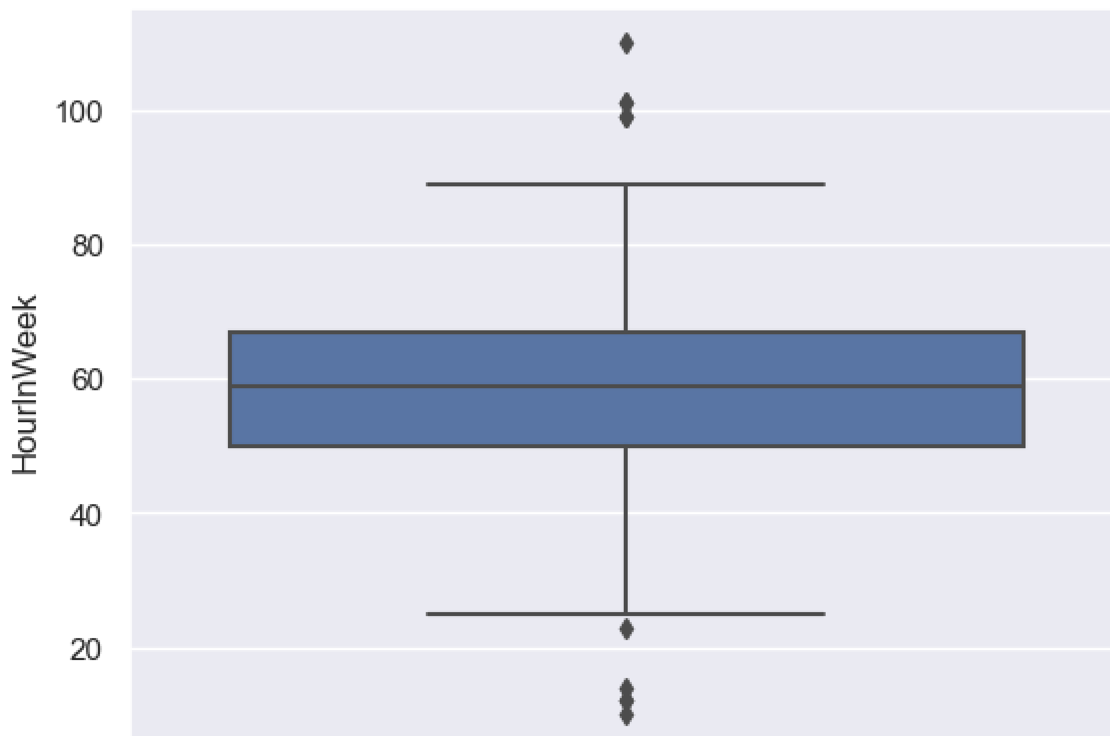


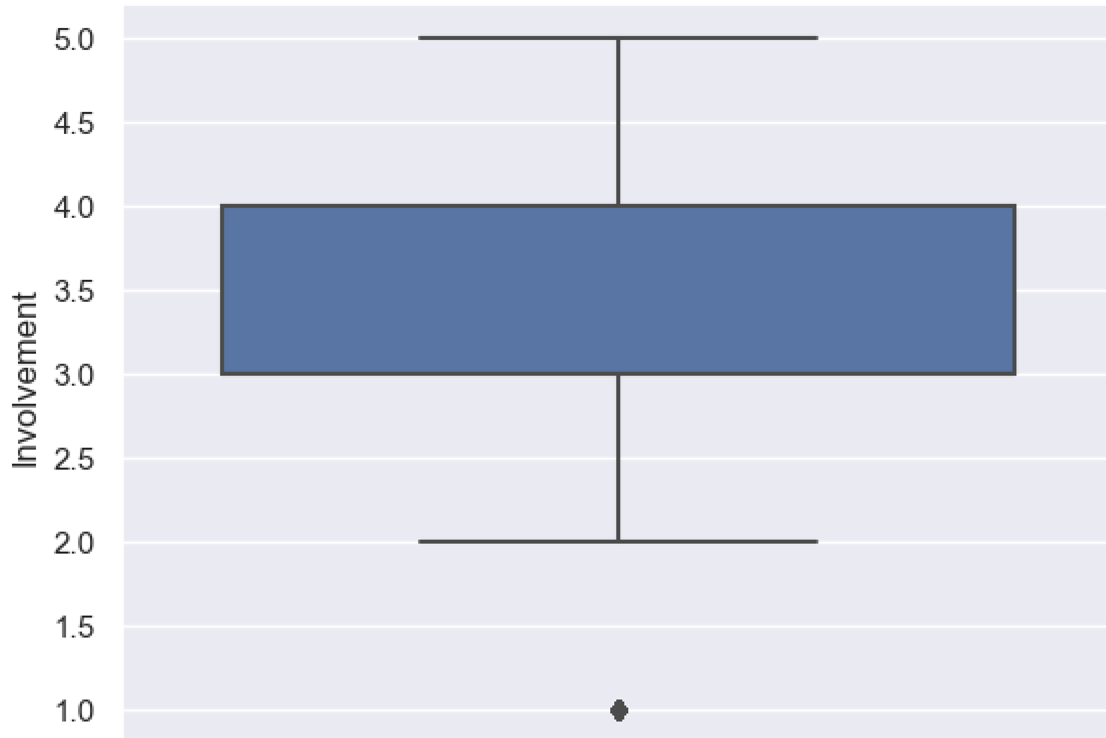
60
50
40
30
20
Age



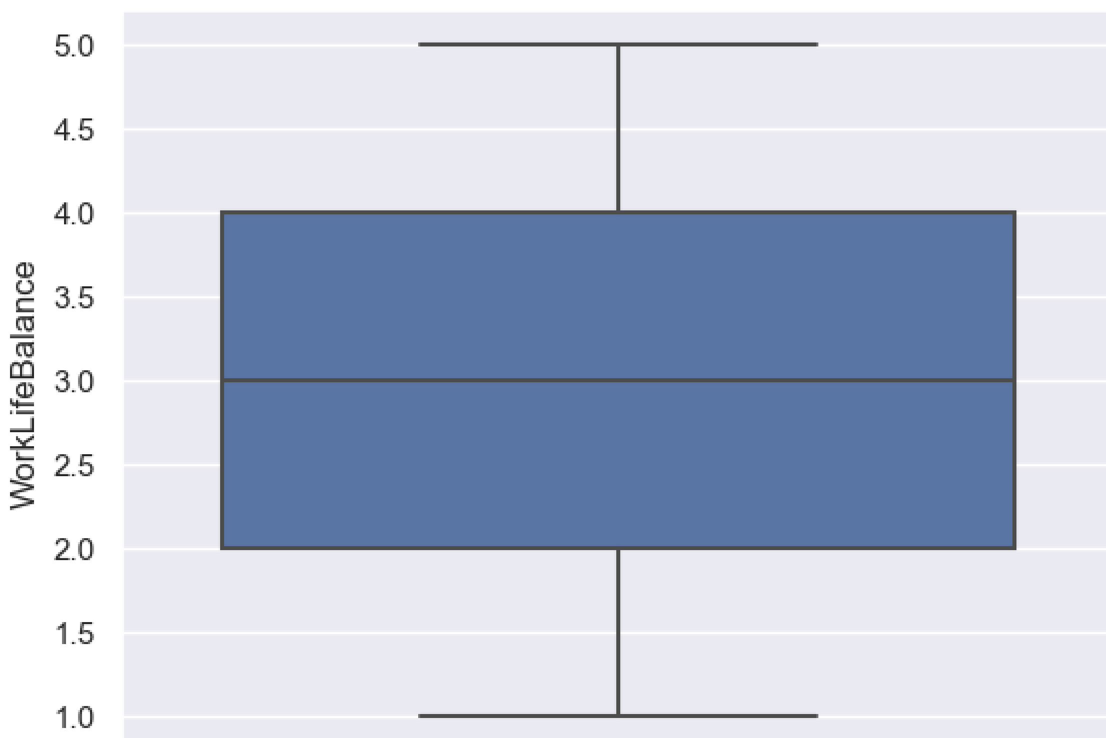
120
100
80
60
40
20
0
HomeToWork


1.0
0.8
0.6
Gender
0.4
0.2
0.0

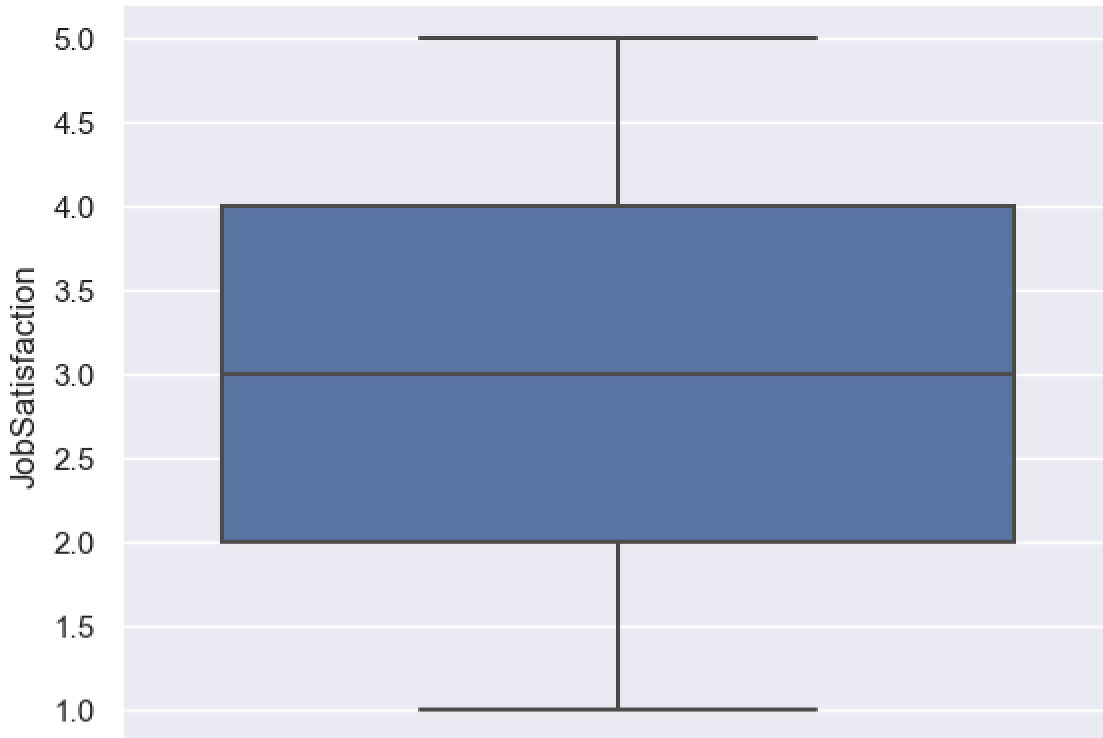

100
80
HourInWeek
60
40
20



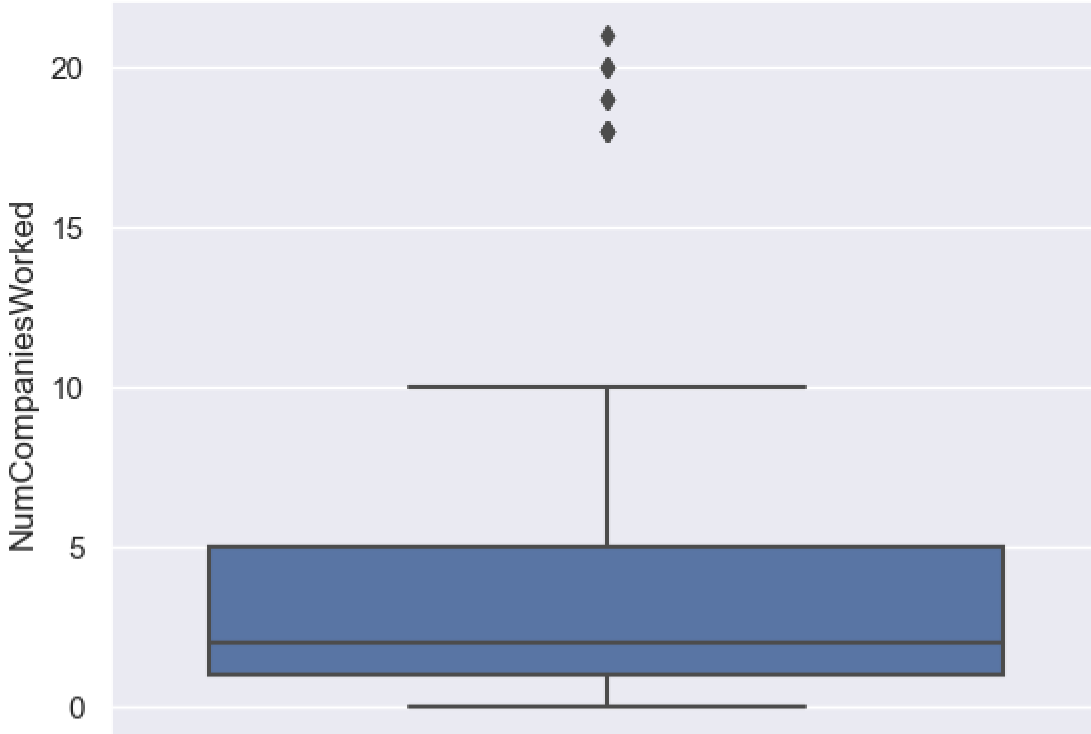
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Involvement



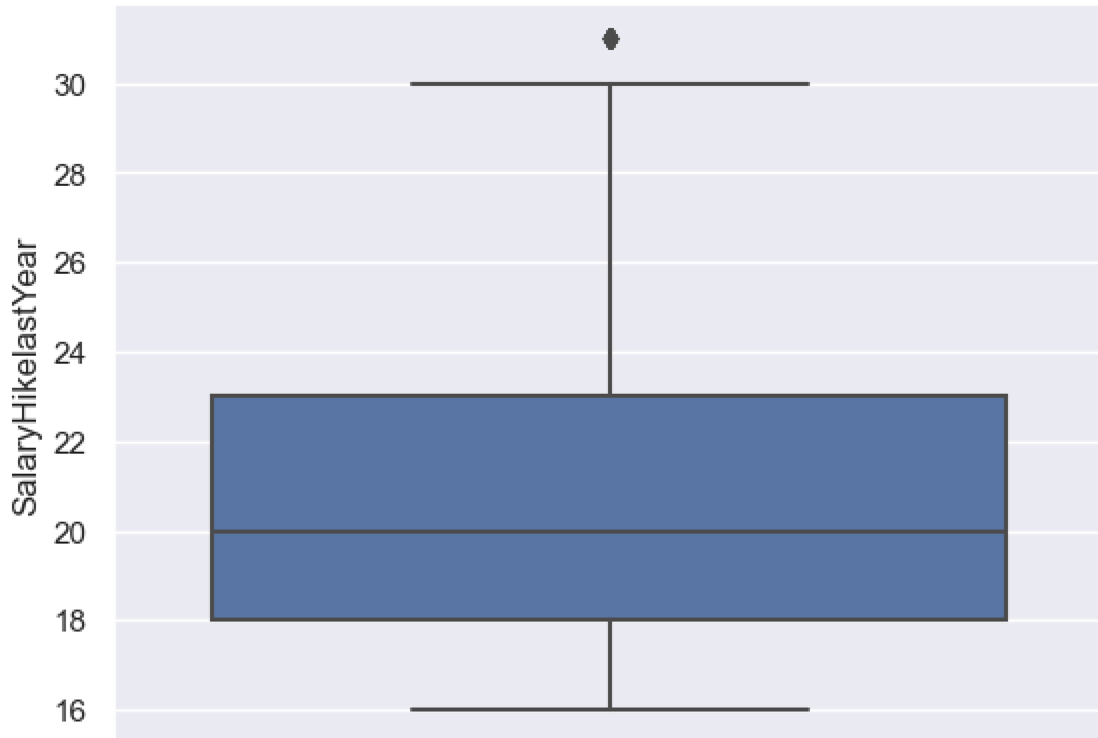
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
WorkLifeBalance


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
JobSatisfaction

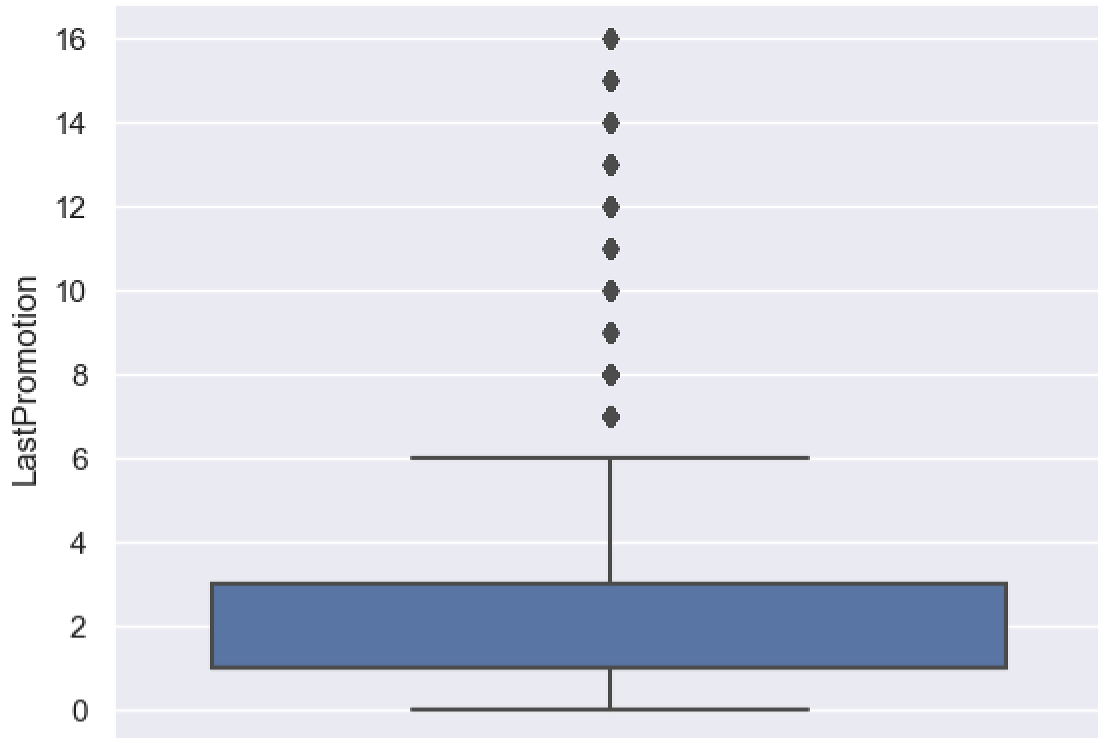

1.0
0.8
0.6
0.4
0.2
0.0
ESOPs


NumCompaniesWorked
0
5
10
15
20

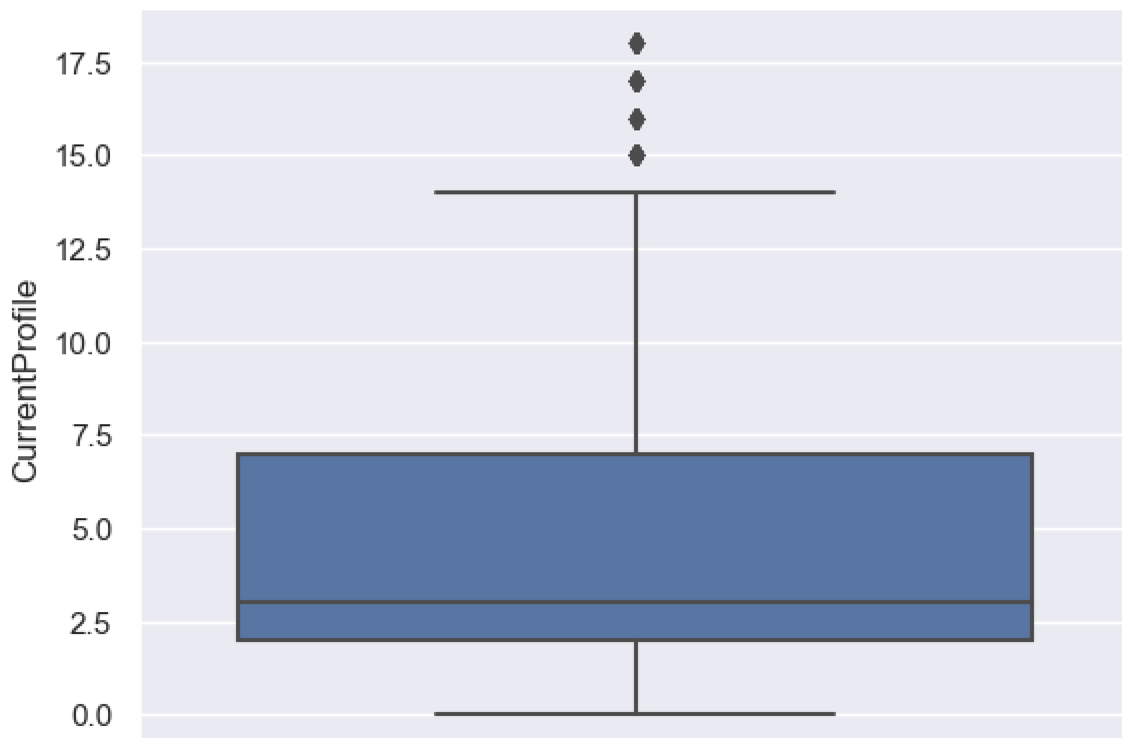

OverTime
0.0
0.2
0.4
0.6
0.8
1.0


SalaryHikelastYear
30
28
26
24
22
20
18
16

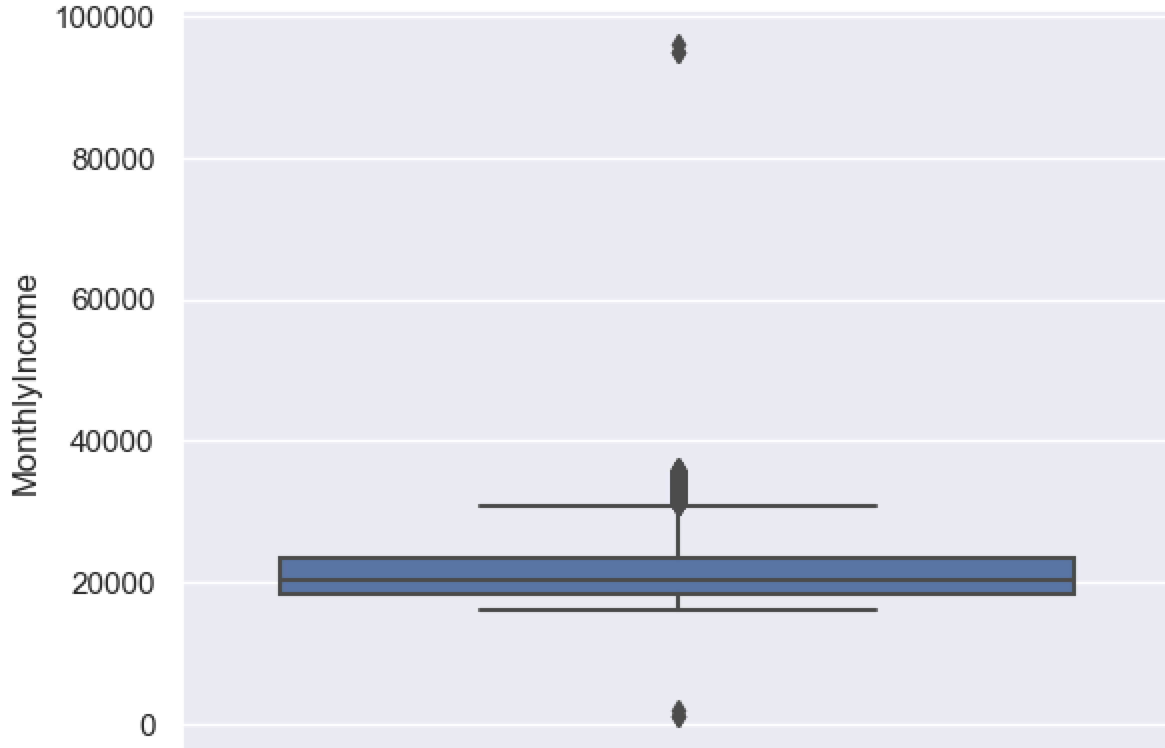

WorkExperience
40
30
20
10
0



LastPromotion
0
2
4
6
8
10
12
14
16



CurrentProfile
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


100000
80000
60000
40000
20000
0
MonthlyIncome


1.0
0.8
0.6
0.4
0.2
0.0
TravelProfile_Rarely

TravelProfile_Yes
1.0
0.8
0.6
0.4
0.2
0.0

Department_Marketing
1.0
0.8
0.6
0.4
0.2
0.0

1.0
0.8
0.6
0.4
0.2
0.0
Department_Sales

1.0
0.8
0.6
0.4
0.2
0.0
EducationField_Engineer


1.0
0.8
0.6
0.4
0.2
0.0
EducationField_MBA


1.0
0.8
0.6
0.4
0.2
0.0
EducationField_Marketing Diploma

1.0
0.8
0.6
0.4
0.2
0.0
EducationField_Other

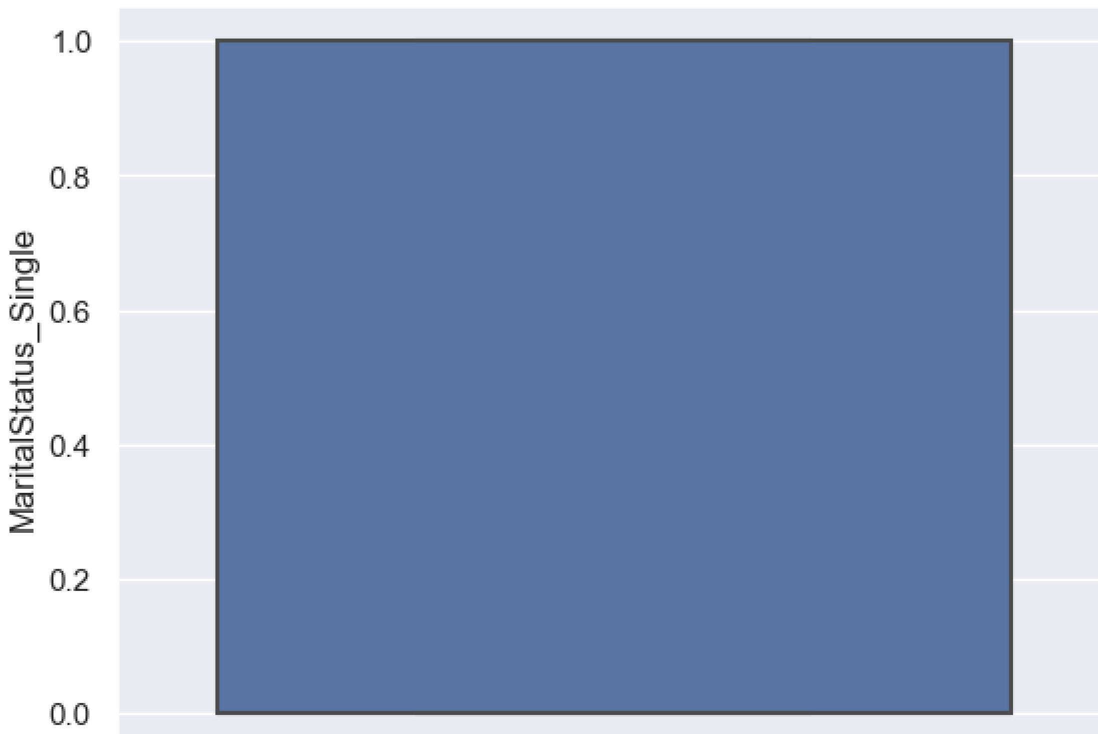
1.0
0.8
0.6
0.4
0.2
0.0
EducationField_Statistics


1.0
0.8
0.6
0.4
0.2
0.0
Designation_Executive


1.0
0.8
0.6
0.4
0.2
0.0
Designation_Manager


Designation_Senior Manager
1.0
0.8
0.6
0.4
0.2
0.0


Designation_VP
1.0
0.8
0.6
0.4
0.2
0.0



Below features contain outlier ,Will treat by using IQR Technique

In [55]:
```
def outlierCapping(col):
    # Finding the IQR
    Q1 = dataset[col].quantile(0.25)
    Q3 = dataset[col].quantile(0.75)
    IQR = Q3 - Q1
    print("Column--",col)
    print("Percentile25 :", Q1)
    print("Percentile75 :", Q3)
    print("InterQuartileRange :", IQR)
```

```python
    upper_limit = Q3 + 1.5 * IQR
    lower_limit = Q1 - 1.5 * IQR

    print("Upper Limit :", upper_limit)
    print("Lower Limit :", lower_limit)
    print("#############################")
    dataset[col] = np.where(dataset[col] > upper_limit,
                                             upper_limit,
                                              np.where(dataset[col] < lower_limit,
                                                       lower_limit,
                                                      dataset[col]))
```

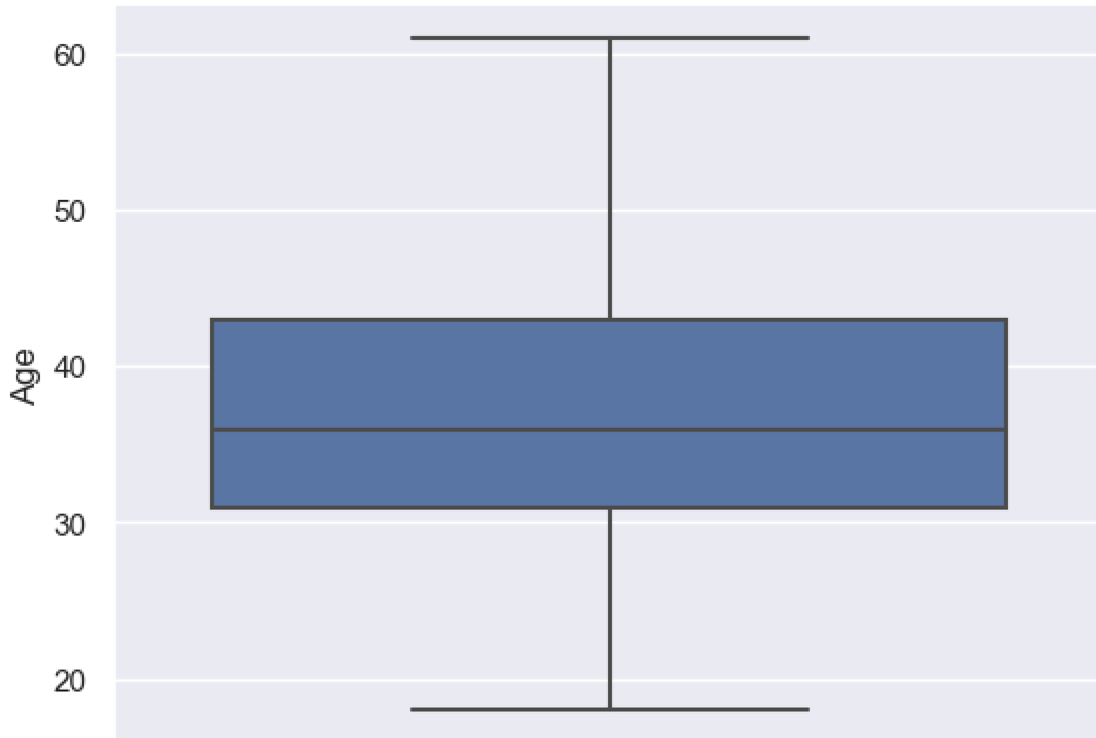
In [56]:
```python
col=['HomeToWork','HourlnWeek','Involvement','NumCompaniesWorked','SalaryHikelastYear','
for i in col:
    outlierCapping(i)
```

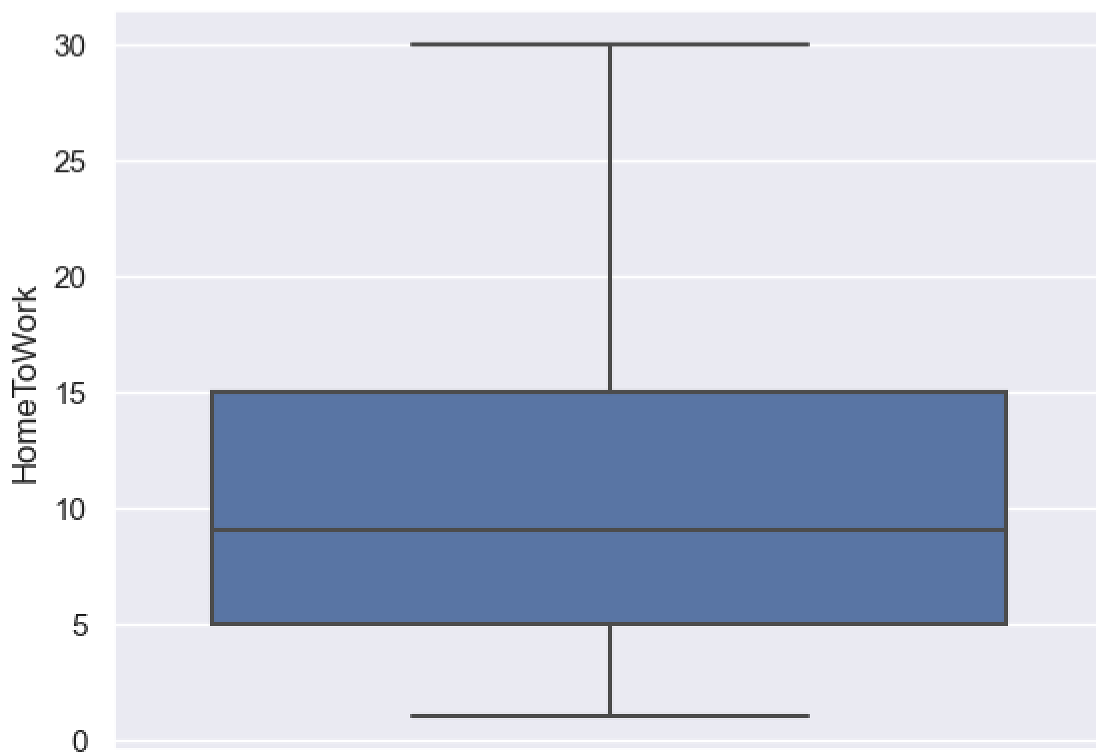
```
Column-- HomeToWork
Percentile25 : 5.0
Percentile75 : 15.0
InterQuartileRange : 10.0
Upper Limit : 30.0
Lower Limit : -10.0
#############################
Column-- HourlnWeek
Percentile25 : 50.0
Percentile75 : 67.0
InterQuartileRange : 17.0
Upper Limit : 92.5
Lower Limit : 24.5
#############################
Column-- Involvement
Percentile25 : 3.0
Percentile75 : 4.0
InterQuartileRange : 1.0
Upper Limit : 5.5
Lower Limit : 1.5
#############################
Column-- NumCompaniesWorked
Percentile25 : 1.0
Percentile75 : 5.0
InterQuartileRange : 4.0
Upper Limit : 11.0
Lower Limit : -5.0
#############################
Column-- SalaryHikelastYear
Percentile25 : 18.0
Percentile75 : 23.0
InterQuartileRange : 5.0
Upper Limit : 30.5
Lower Limit : 10.5
#############################
Column-- WorkExperience
Percentile25 : 6.0
Percentile75 : 15.0
InterQuartileRange : 9.0
Upper Limit : 28.5
Lower Limit : -7.5
#############################
Column-- CurrentProfile
Percentile25 : 2.0
Percentile75 : 7.0
InterQuartileRange : 5.0
Upper Limit : 14.5
Lower Limit : -5.5
#############################
```

```
Column-- LastPromotion
Percentile25 : 1.0
Percentile75 : 3.0
InterQuartileRange : 2.0
Upper Limit : 6.0
Lower Limit : -2.0
#############################
Column-- MonthlyIncome
Percentile25 : 18398.0
Percentile75 : 23380.75
InterQuartileRange : 4982.75
Upper Limit : 30854.875
Lower Limit : 10923.875
#############################
```

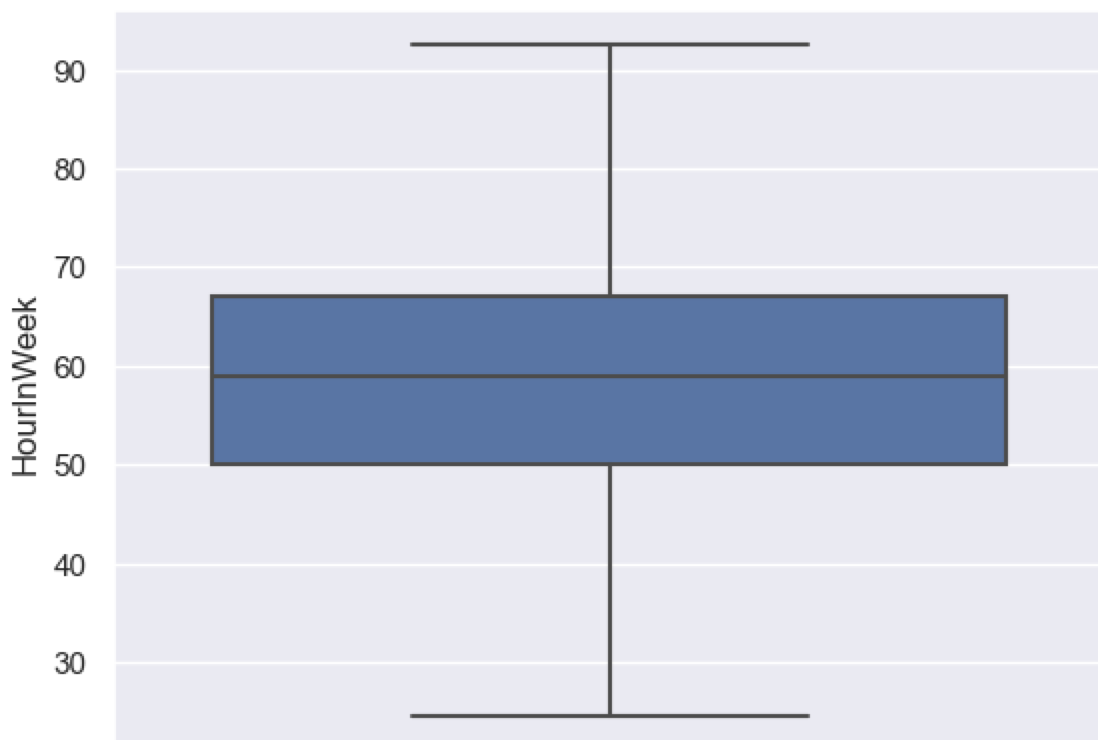
In [57]:

```python
def boxplots(col):
    sns.boxplot(y=col, data=dataset)
    plt.show()

for i in list(dataset.select_dtypes(exclude=["object"]).columns)[0:]:
    boxplots(i)
```

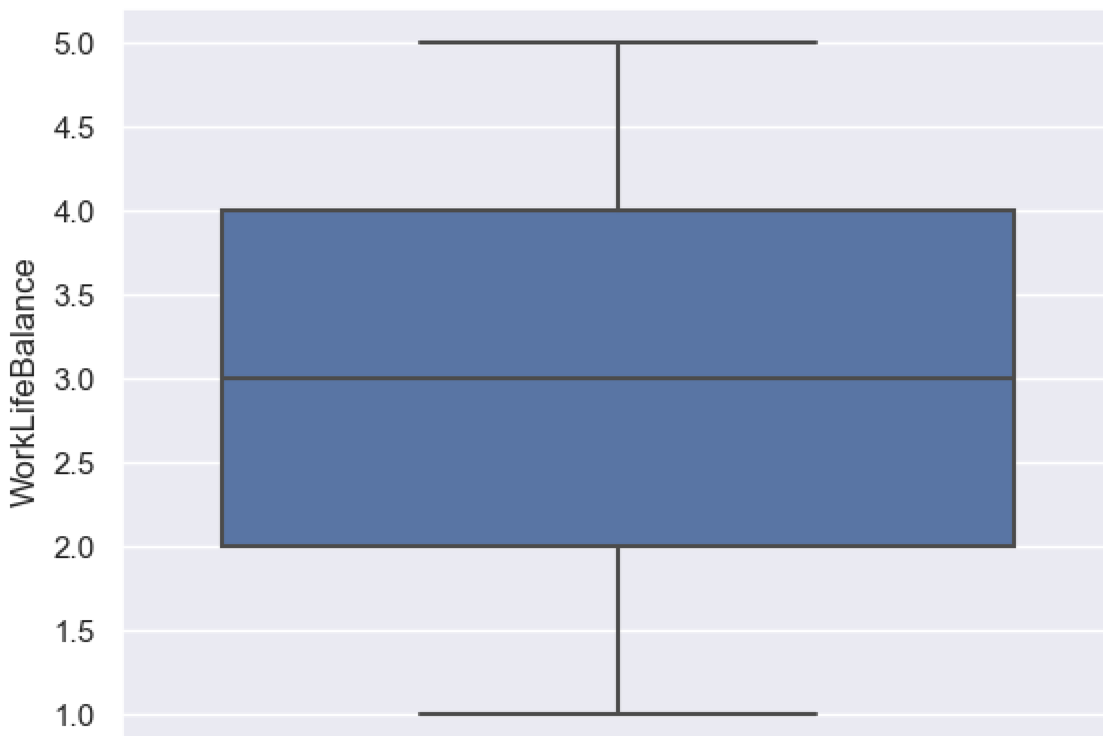

60
50
40
30
20
Age


30
25
20
15
10
5
0
HomeToWork


1.0
0.8
0.6
0.4
0.2
0.0
Gender

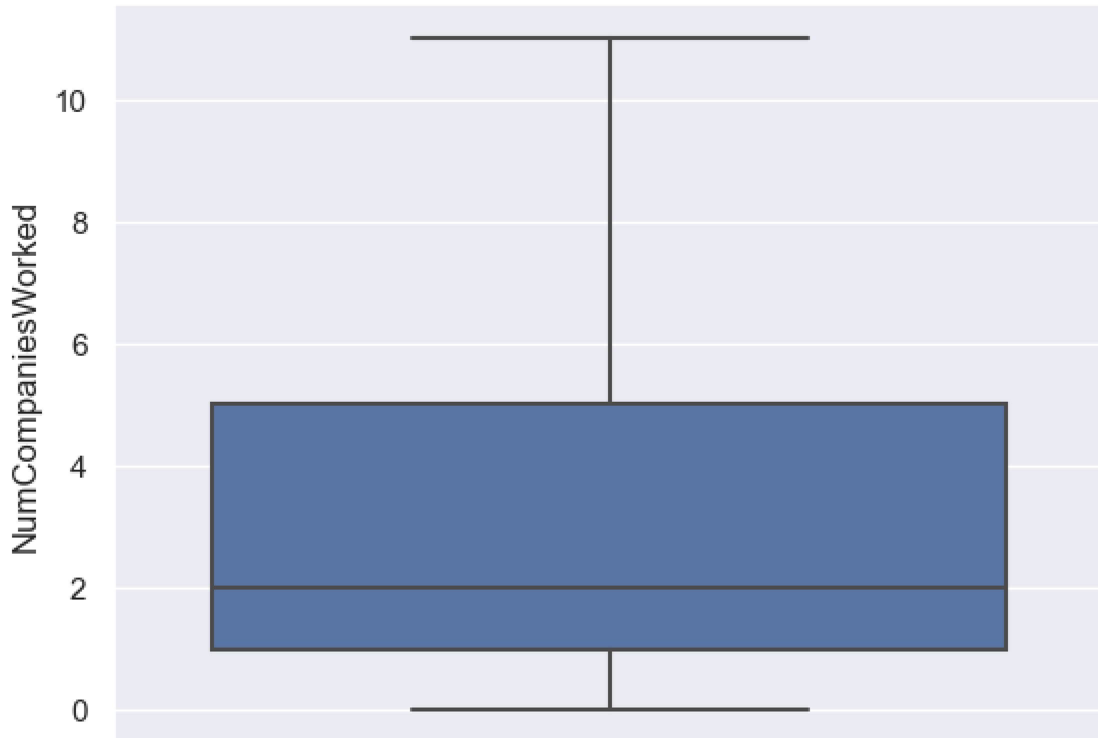

90
80
70
60
50
40
30
HourInWeek


Involvement
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5


WorkLifeBalance
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0

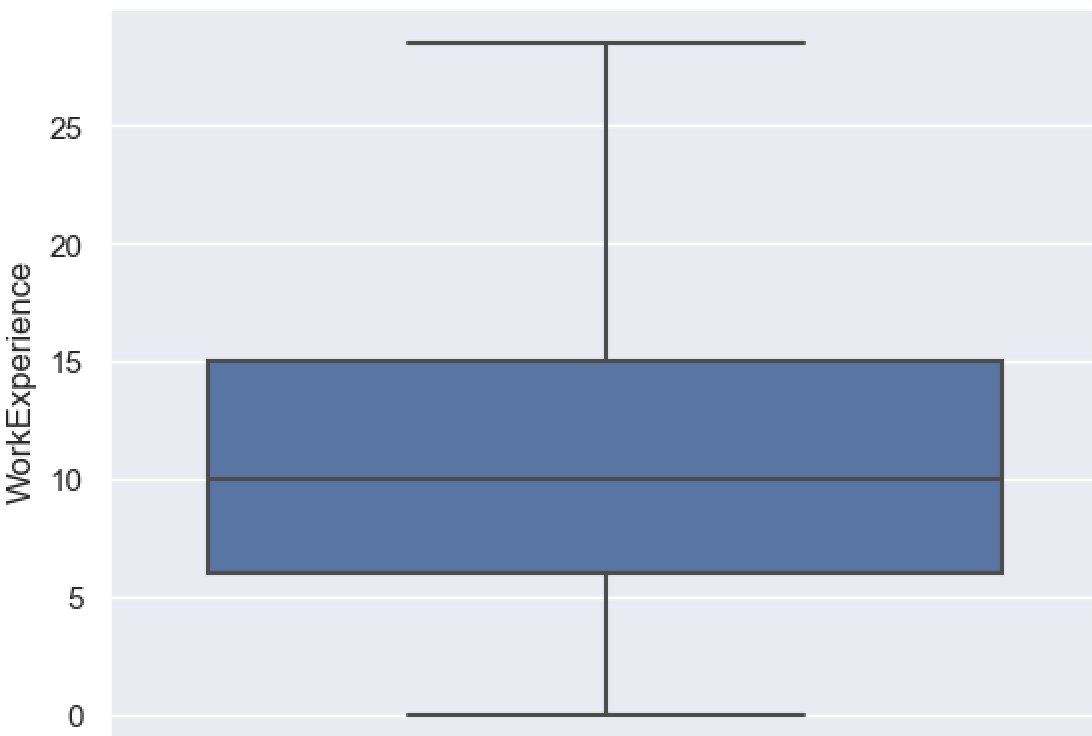
JobSatisfaction
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0

ESOPs
1.0
0.8
0.6
0.4
0.2
0.0

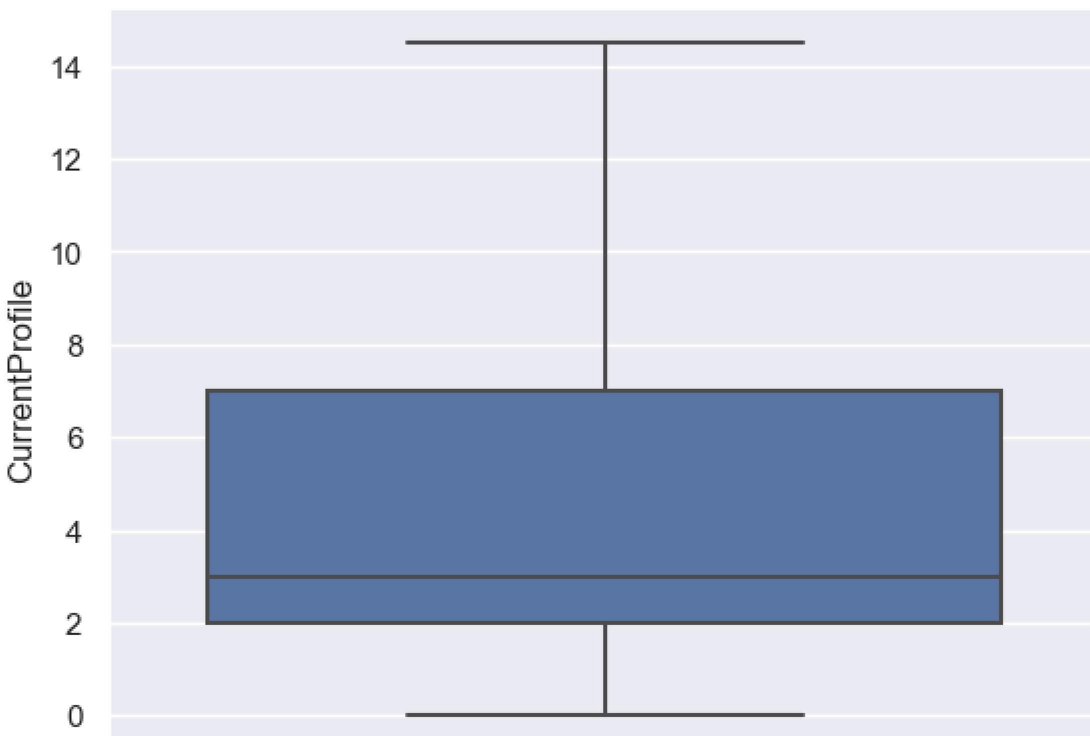

NumCompaniesWorked
10
8
6
4
2
0


OverTime
1.0
0.8
0.6
0.4
0.2
0.0

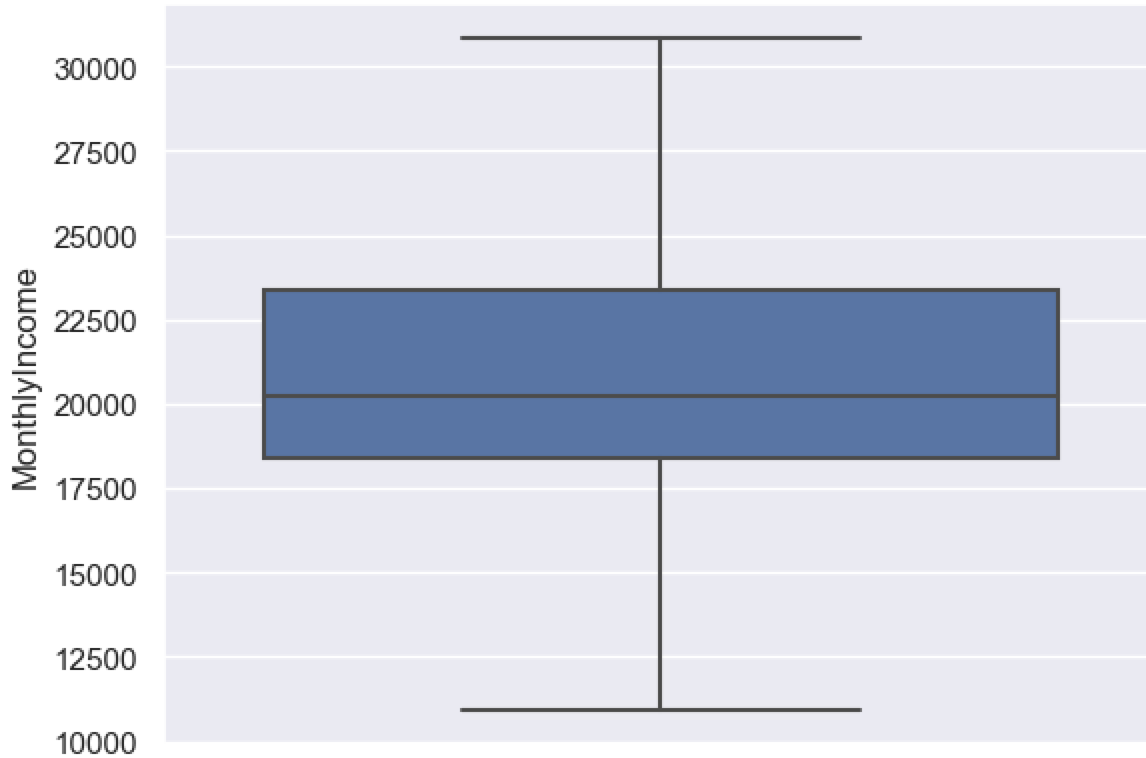

SalaryHikelastYear
30
28
26
24
22
20
18
16


WorkExperience
25
20
15
10
5
0

6
5
4
3
2
1
0
LastPromotion



14
12
10
8
6
4
2
0
CurrentProfile



30000
27500
25000
22500
20000
17500
15000
12500
10000
MonthlyIncome

1.0
0.8
0.6
0.4
0.2
0.0
TravelProfile_Rarely

1.0
0.8
0.6
0.4
0.2
0.0
TravelProfile_Yes

1.0
0.8
0.6
0.4
0.2
0.0
Department_Marketing


Department_Sales
1.0
0.8
0.6
0.4
0.2
0.0


EducationField_Engineer
1.0
0.8
0.6
0.4
0.2
0.0

1.0
0.8
0.6
0.4
0.2
0.0
EducationField_MBA

1.0
0.8
0.6
0.4
0.2
0.0
EducationField_Marketing Diploma


1.0
0.8
0.6
0.4
0.2
0.0
EducationField_Other


1.0
0.8
0.6
0.4
0.2
0.0
EducationField_Statistics

Designation_Executive
1.0
0.8
0.6
0.4
0.2
0.0

Designation_Manager
1.0
0.8
0.6
0.4
0.2
0.0

1.0
0.8
0.6
0.4
0.2
0.0
Designation_Senior Manager

1.0
0.8
0.6
0.4
0.2
0.0
Designation_VP

Split train data and test data

In [58]:
```
train_dataset=dataset[dataset["flag"]=='train']
test_dataset=dataset[dataset["flag"]=='test']
```

In [59]:
```
train_dataset.shape
```

Out[59]:
```
(5180, 32)
```

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

Out[60]:
```
(2630, 32)
```

In [61]:
```
test_dataset.head(2)
```

Out[61]:

	Attrition	Age	HomeToWork	Gender	HourInWeek	Involvement	WorkLifeBalance	JobSatisfaction	ESOPs	N
0	NaN	18.0	9.0	1	80.0	3.0	2.0	3.0	1.0	
1	NaN	20.0	28.0	0	59.0	1.5	3.0	1.0	1.0	

In [62]:
```
train_dataset.drop(columns=['flag'],inplace=True)
```

In [63]:
```
test_dataset.drop(columns=['Attrition','flag'],inplace=True)
```

In [64]:
```
# splitting the data into independent and dependent variable
x = train_dataset.drop(['Attrition'], axis=1)
y = train_dataset['Attrition']
```

In [65]:
```
x.head(3)
```

Out[65]:

	Age	HomeToWork	Gender	HourInWeek	Involvement	WorkLifeBalance	JobSatisfaction	ESOPs	NumCompar
0	35.0	5.0	1	69.0	1.5	1.0	1.0	1.0	
1	32.0	5.0	0	62.0	4.0	3.0	2.0	0.0	
2	31.0	5.0	0	45.0	5.0	3.0	2.0	1.0	

In [66]:
```
y.head(3)
```

Out[66]:
```
0    0.0
1    1.0
2    0.0
Name: Attrition, dtype: float64
```

In [67]:
```
y.value_counts()
```

Out[67]:
```
Attrition
0.0    3735
1.0    1445
Name: count, dtype: int64
```

In [68]:
```
# imbalance treatment required however we use stratify to balance training and test data
#!pip uninstall scikit-learn
#!pip install scikit-learn==1.2.2
import imblearn
from imblearn.over_sampling import SMOTE
smote= SMOTE()
x_smote, y_smote = smote.fit_resample(x,y)
print("Original-",y.value_counts())
print("After SMOTE-",y_smote.value_counts())
```

```
Original- Attrition
0.0    3735
1.0    1445
Name: count, dtype: int64
After SMOTE- Attrition
0.0    3735
1.0    3735
Name: count, dtype: int64
```

Split training and testing data

In [69]:
```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_smote, y_smote, test_size=0.2, ran
```

In [70]:
```python
x_train.head()
```

Out[70]:

	Age	HomeToWork	Gender	HourlnWeek	Involvement	WorkLifeBalance	JobSatisfaction	ESOPs	NumCon
3990	53.0	8.0	0	78.0	4.0	5.0	1.0	1.0	
2397	34.0	3.0	1	65.0	3.0	2.0	3.0	1.0	
286	56.0	5.0	1	39.0	3.0	4.0	1.0	0.0	
2590	37.0	1.0	0	80.0	3.0	5.0	1.0	1.0	
1145	25.0	29.0	1	41.0	4.0	4.0	3.0	0.0	

In [71]:
```python
# data leakage problem -

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [72]:
```python
x_train=pd.DataFrame(x_train,columns=x_smote.columns)
x_train.head()
```

Out[72]:

	Age	HomeToWork	Gender	HourlnWeek	Involvement	WorkLifeBalance	JobSatisfaction	ESOPs	N
0	1.875342	-0.406306	-1.180775	1.628067	0.859431	1.479603	-1.714340	1.056724	
1	-0.239966	-1.035123	0.846901	0.556708	-0.420851	-0.769582	-0.179739	1.056724	
2	2.209339	-0.783596	0.846901	-1.586009	-0.420851	0.729874	-1.714340	-1.047318	
3	0.094030	-1.286649	-1.180775	1.792891	-0.420851	1.479603	-1.714340	1.056724	
4	-1.241955	2.234723	0.846901	-1.421185	0.859431	0.729874	-0.179739	-1.047318	

In [73]:
```python
x_test=pd.DataFrame(x_test,columns=x_smote.columns)
x_test.head()
```

Out[73]:

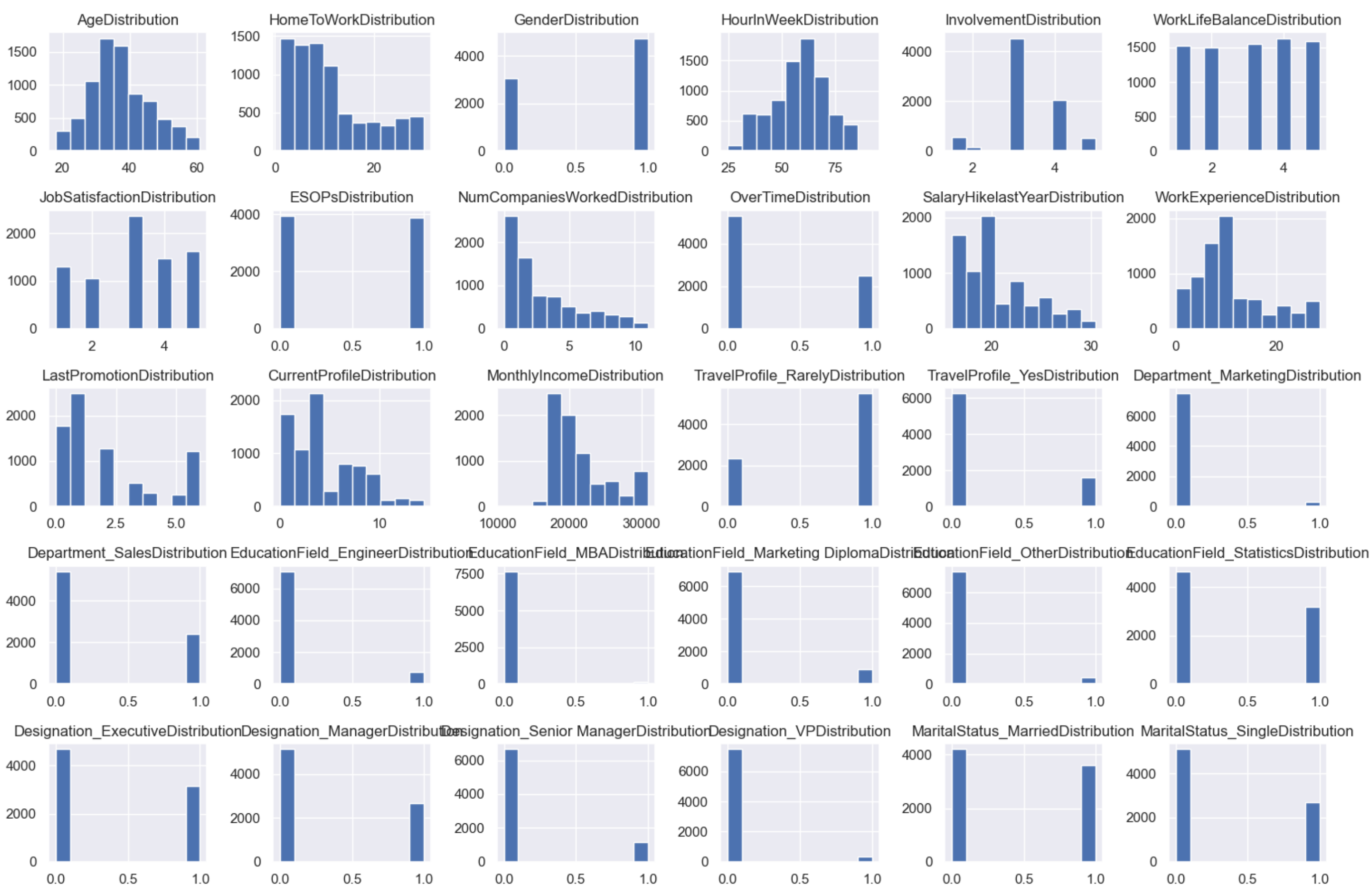
	Age	HomeToWork	Gender	HourlnWeek	Involvement	WorkLifeBalance	JobSatisfaction	ESOPs	N
0	-0.940106	-0.795701	0.846901	-0.070861	-0.420851	-1.374986	1.354862	0.854208	
1	-0.278159	1.640421	-1.180775	-1.762142	-0.420851	-1.519310	-0.947039	-0.557397	
2	-0.128634	0.977090	-1.180775	0.721532	-0.420851	-0.019854	-0.179739	1.056724	
3	0.205362	-0.029016	0.846901	1.051181	-0.420851	-1.519310	-0.179739	1.056724	
4	1.845127	1.830915	0.846901	0.634303	-0.265828	0.639093	1.169044	1.056724	

In [74]:
```python
x_test.shape
```

Out[74]:
```
(1494, 30)
```

In [75]:
```python
x_train.shape
```

Out[75]:
```
(5976, 30)
```

In [76]:
```
y_train.value_counts()
```
Out[76]:
```
Attrition
1.0    3005
0.0    2971
Name: count, dtype: int64
```
In [77]:
```
y_test.value_counts()
```
Out[77]:
```
Attrition
0.0    764
1.0    730
Name: count, dtype: int64
```
In [78]:
```
def draw_histogram(dataset, variables, n_rows, n_cols):
    fig = plt.figure(figsize=(15,10))
    for i, var_name in enumerate(variables):
        ax = fig.add_subplot(n_rows, n_cols, i+1)
        dataset[var_name].hist(bins=10, ax=ax)
        ax.set_title(var_name + "Distribution")
    fig.tight_layout()
    plt.show()

draw_histogram(dataset, x_train,5,6)
```


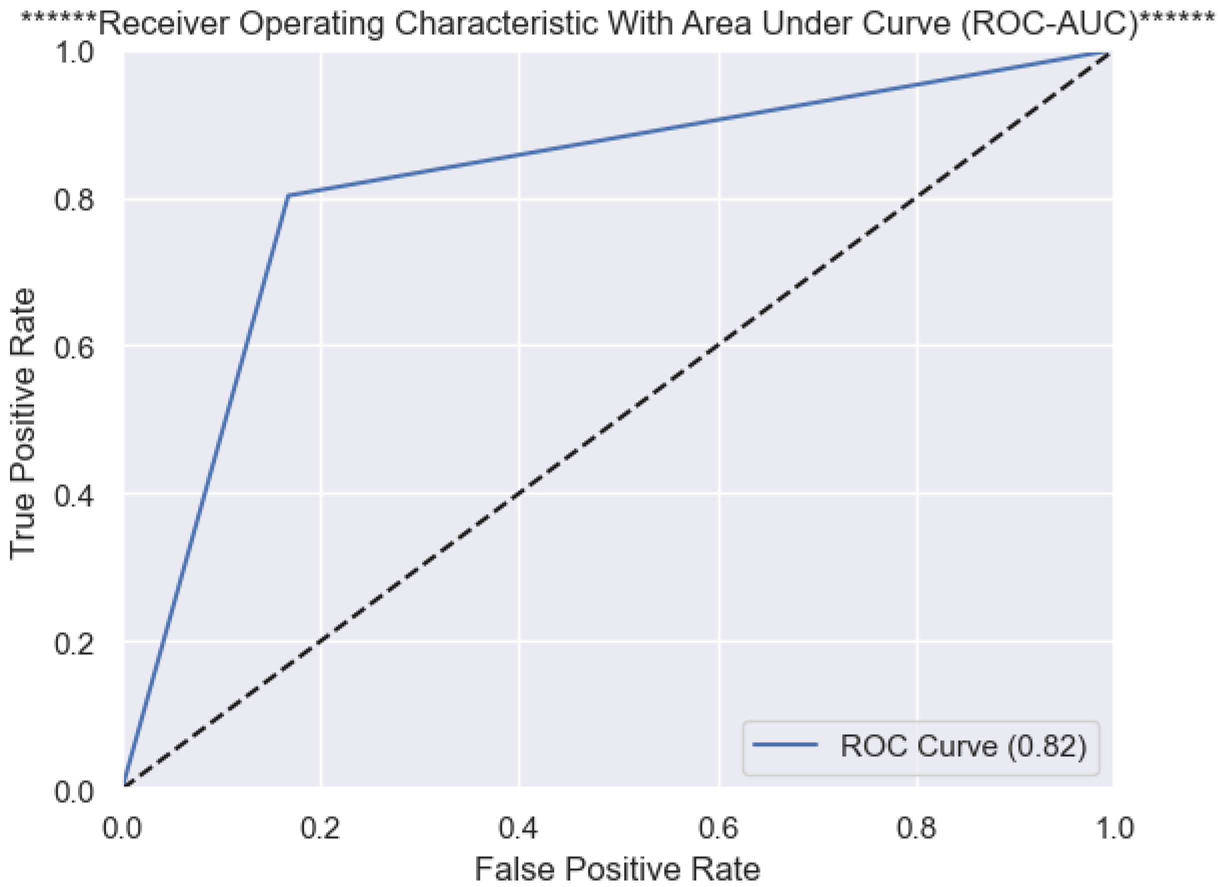
Model Building

Model 1 : AdaBoost

In [79]:
```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.metrics import roc_auc_score
```

```python
from sklearn.metrics import roc_curve
ada = AdaBoostClassifier()
ada.fit(x_train, y_train)
y_pred_train_ada = ada.predict(x_train)
y_pred_test_ada = ada.predict(x_test)
print("**************************")
print(confusion_matrix(y_train, y_pred_train_ada))
print()
print(confusion_matrix(y_test, y_pred_test_ada))
print("**************************")
print(classification_report(y_train, y_pred_train_ada))
print()
print(classification_report(y_test, y_pred_test_ada))
print("**************************")
print("Accuracy Train Score-",accuracy_score(y_train, y_pred_train_ada))
print()
print("Accuracy Test Score-",accuracy_score(y_test, y_pred_test_ada))

ada_roc_auc = roc_auc_score(y_test, y_pred_test_ada)
ada_roc_auc


fpr, tpr, thresholds = roc_curve(y_test, y_pred_test_ada)

plt.figure()
plt.plot(fpr, tpr, label="ROC Curve (%0.2f)" % ada_roc_auc)
plt.plot([0,1],[0,1],'k--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("******Receiver Operating Characteristic With Area Under Curve (ROC-AUC)******
plt.legend(loc='lower right')
plt.show()
```

```
**************************
[[2519  452]
 [ 499 2506]]

[[636 128]
 [144 586]]
**************************
              precision    recall  f1-score   support

         0.0       0.83      0.85      0.84      2971
         1.0       0.85      0.83      0.84      3005

    accuracy                           0.84      5976
   macro avg       0.84      0.84      0.84      5976
weighted avg       0.84      0.84      0.84      5976


              precision    recall  f1-score   support

         0.0       0.82      0.83      0.82       764
         1.0       0.82      0.80      0.81       730

    accuracy                           0.82      1494
   macro avg       0.82      0.82      0.82      1494
weighted avg       0.82      0.82      0.82      1494

**************************
Accuracy Train Score- 0.8408634538152611

Accuracy Test Score- 0.8179384203480589
```



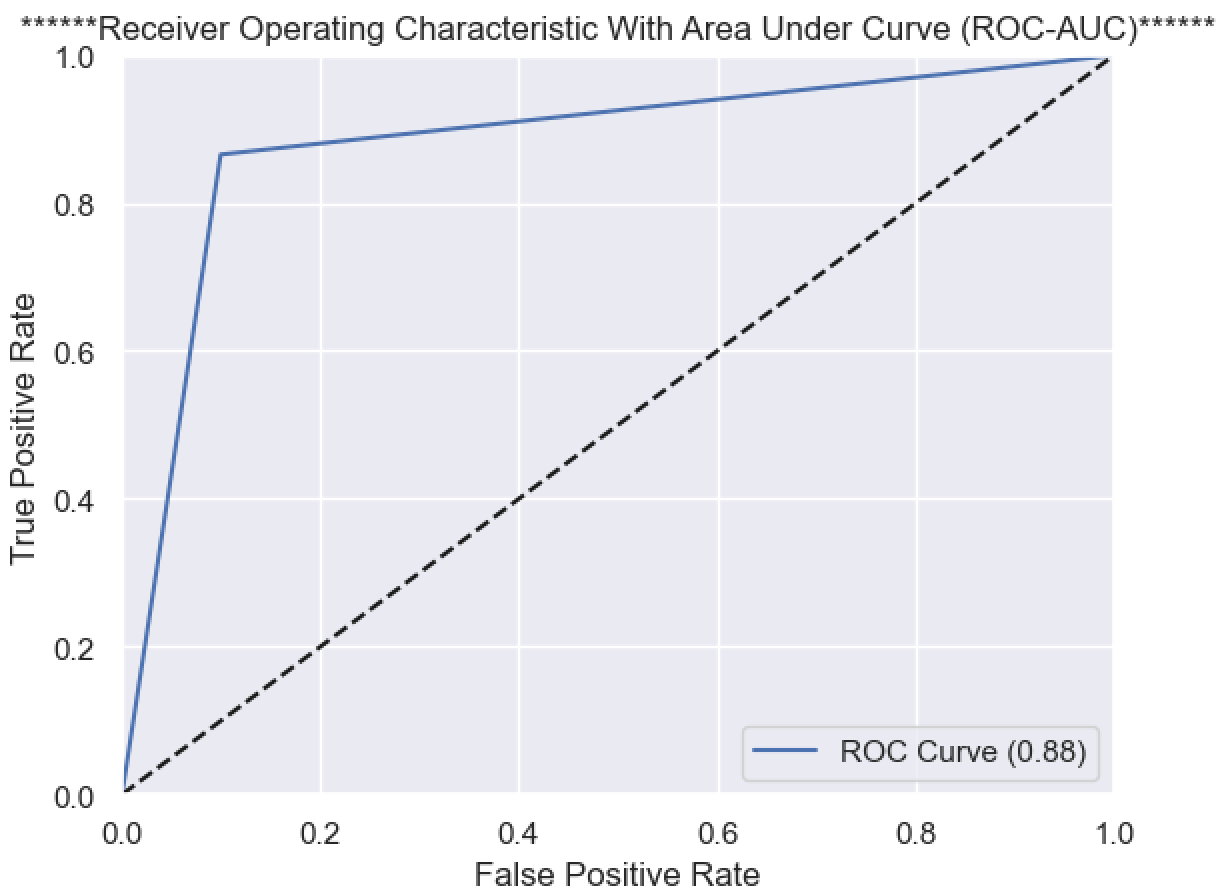
Model 2 - Gradient Boosting Algorithm

In [80]:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.ensemble import GradientBoostingClassifier
# from sklearn.ensemble import GradientBoostingRegressor - regression problem
gdm = GradientBoostingClassifier()
gdm.fit(x_train, y_train)
y_pred_train_gdm = gdm.predict(x_train)
y_pred_test_gdm = gdm.predict(x_test)
print("***************************")
print(confusion_matrix(y_train, y_pred_train_gdm))
print()
print(confusion_matrix(y_test, y_pred_test_gdm))
print("***************************")
print(classification_report(y_train, y_pred_train_gdm))
print()
print(classification_report(y_test, y_pred_test_gdm))
print("***************************")
print("Accuracy Train Score-",accuracy_score(y_train, y_pred_train_gdm))
print()
print("Accuracy Test Score-",accuracy_score(y_test, y_pred_test_gdm))

gdm_roc_auc = roc_auc_score(y_test, y_pred_test_gdm)
gdm_roc_auc


fpr, tpr, thresholds = roc_curve(y_test, y_pred_test_gdm)

plt.figure()
plt.plot(fpr, tpr, label="ROC Curve (%0.2f)" % gdm_roc_auc)
plt.plot([0,1],[0,1],'k--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
```

```
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("******Receiver Operating Characteristic With Area Under Curve (ROC-AUC)******
plt.legend(loc='lower right')
plt.show()
```

```
**************************
[[2747  224]
 [ 360 2645]]

[[688  76]
 [ 98 632]]
**************************
              precision    recall  f1-score   support

         0.0       0.88      0.92      0.90      2971
         1.0       0.92      0.88      0.90      3005

    accuracy                           0.90      5976
   macro avg       0.90      0.90      0.90      5976
weighted avg       0.90      0.90      0.90      5976


              precision    recall  f1-score   support

         0.0       0.88      0.90      0.89       764
         1.0       0.89      0.87      0.88       730

    accuracy                           0.88      1494
   macro avg       0.88      0.88      0.88      1494
weighted avg       0.88      0.88      0.88      1494

**************************
Accuracy Train Score- 0.9022757697456493

Accuracy Test Score- 0.8835341365461847
```



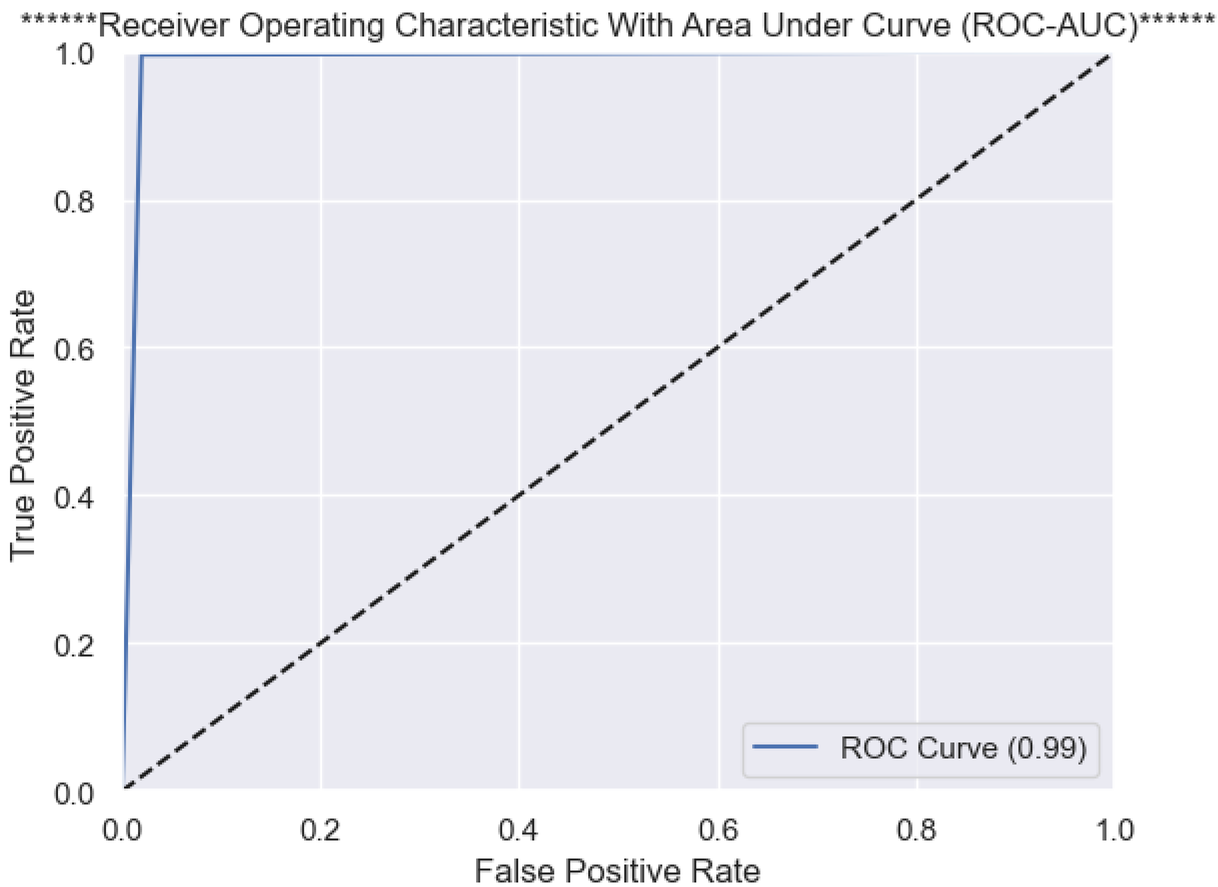
Model 3 - XGBoost Classification

In [81]:

```
#!pip install xgboost
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(x_train, y_train)
y_pred_train_xgb = xgb.predict(x_train)
y_pred_test_xgb = xgb.predict(x_test)
print("***************************")
print(confusion_matrix(y_train, y_pred_train_xgb))
print()
print(confusion_matrix(y_test, y_pred_test_xgb))
print("***************************")
print(classification_report(y_train, y_pred_train_xgb))
print()
print(classification_report(y_test, y_pred_test_xgb))
print("***************************")
print("Accuracy Train Score-",accuracy_score(y_train, y_pred_train_xgb))
print()
print("Accuracy Test Score-",accuracy_score(y_test, y_pred_test_xgb))

xgb_roc_auc = roc_auc_score(y_test, y_pred_test_xgb)
xgb_roc_auc


fpr, tpr, thresholds = roc_curve(y_test, y_pred_test_xgb)

plt.figure()
plt.plot(fpr, tpr, label="ROC Curve (%0.2f)" % xgb_roc_auc)
plt.plot([0,1],[0,1],'k--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("******Receiver Operating Characteristic With Area Under Curve (ROC-AUC)******
plt.legend(loc='lower right')
plt.show()
```

```
***************************
[[2971    0]
 [   0 3005]]

[[749  15]
 [  3 727]]
***************************
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2971
         1.0       1.00      1.00      1.00      3005

    accuracy                           1.00      5976
   macro avg       1.00      1.00      1.00      5976
weighted avg       1.00      1.00      1.00      5976


              precision    recall  f1-score   support

         0.0       1.00      0.98      0.99       764
         1.0       0.98      1.00      0.99       730

    accuracy                           0.99      1494
   macro avg       0.99      0.99      0.99      1494
weighted avg       0.99      0.99      0.99      1494
```

```
**************************
Accuracy Train Score- 1.0

Accuracy Test Score- 0.9879518072289156
```

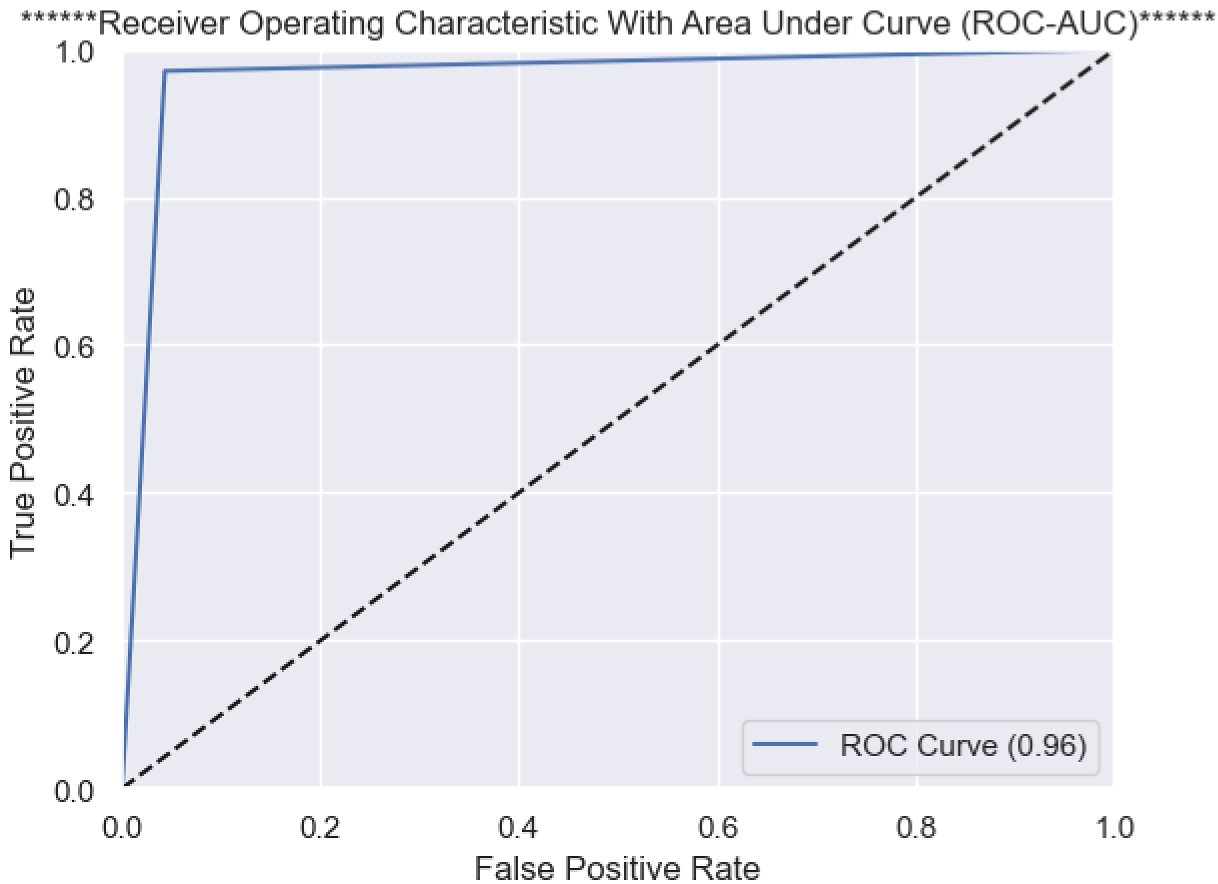


Model 4 - Bagging Classifier

In [82]:
```
from sklearn.ensemble import BaggingClassifier
bagging = BaggingClassifier()
bagging.fit(x_train, y_train)
y_pred_train_bagging = bagging.predict(x_train)
y_pred_test_bagging = bagging.predict(x_test)
print("**************************")
print(confusion_matrix(y_train, y_pred_train_bagging))
print()
print(confusion_matrix(y_test, y_pred_test_bagging))
print("**************************")
print(classification_report(y_train, y_pred_train_bagging))
print()
print(classification_report(y_test, y_pred_test_bagging))
print("**************************")
print("Accuracy Train Score-",accuracy_score(y_train, y_pred_train_bagging))
print()
print("Accuracy Test Score-",accuracy_score(y_test, y_pred_test_bagging))

bagging_roc_auc = roc_auc_score(y_test, y_pred_test_bagging)
bagging_roc_auc


fpr, tpr, thresholds = roc_curve(y_test, y_pred_test_bagging)

plt.figure()
plt.plot(fpr, tpr, label="ROC Curve (%0.2f)" % bagging_roc_auc)
```

```
plt.plot([0,1],[0,1],'k--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("******Receiver Operating Characteristic With Area Under Curve (ROC-AUC)******
plt.legend(loc='lower right')
plt.show()
```

```
***************************
[[2966    5]
 [   8 2997]]

[[731  33]
 [ 21 709]]
***************************
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2971
         1.0       1.00      1.00      1.00      3005

    accuracy                           1.00      5976
   macro avg       1.00      1.00      1.00      5976
weighted avg       1.00      1.00      1.00      5976


              precision    recall  f1-score   support

         0.0       0.97      0.96      0.96       764
         1.0       0.96      0.97      0.96       730

    accuracy                           0.96      1494
   macro avg       0.96      0.96      0.96      1494
weighted avg       0.96      0.96      0.96      1494

***************************
Accuracy Train Score- 0.9978246318607764

Accuracy Test Score- 0.963855421686747
```

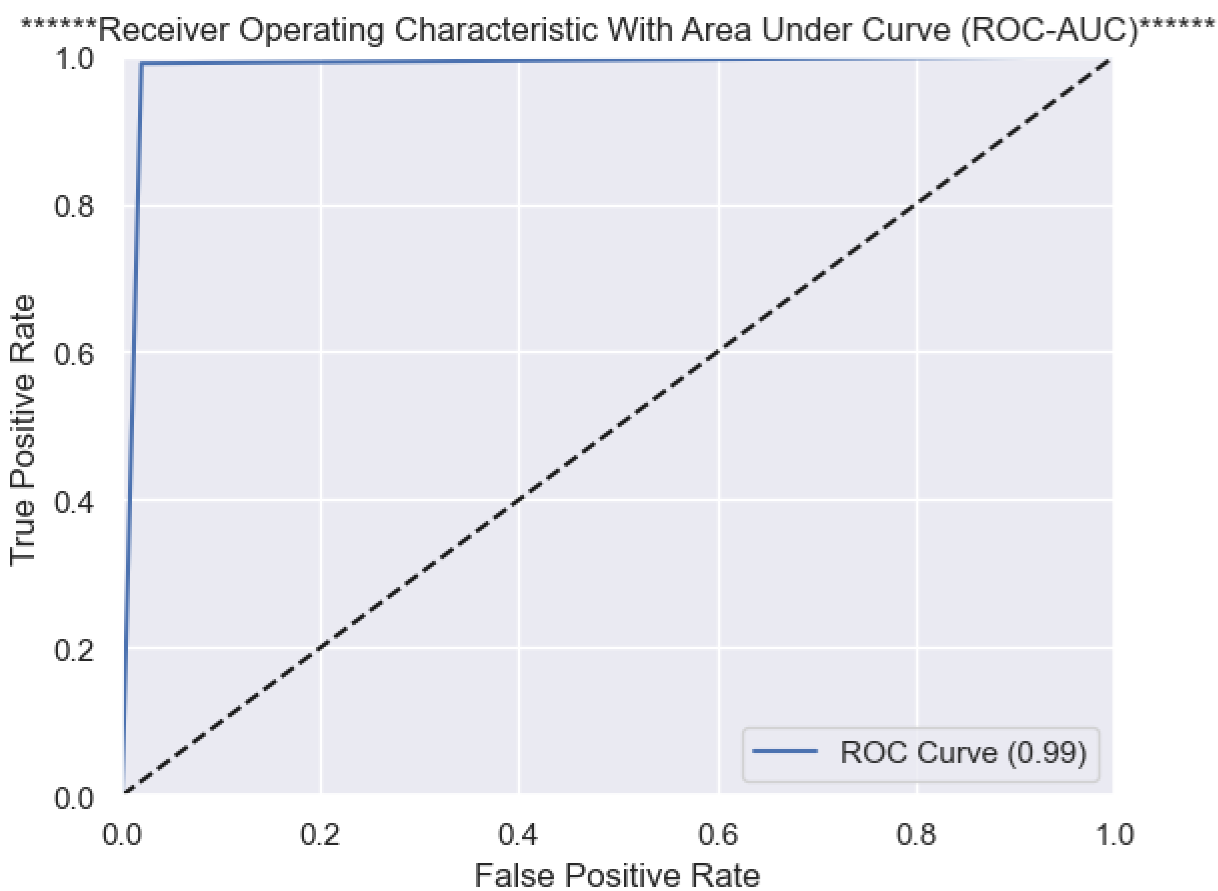


Model 5 - RandomForest Classification

In [83]:
```
from sklearn.ensemble import RandomForestClassifier
rfm = RandomForestClassifier()
rfm.fit(x_train, y_train)
y_pred_train_rfm = rfm.predict(x_train)
y_pred_test_rfm = rfm.predict(x_test)
print("***************************")
print(confusion_matrix(y_train, y_pred_train_rfm))
print()
print(confusion_matrix(y_test, y_pred_test_rfm))
print("***************************")
print(classification_report(y_train, y_pred_train_rfm))
print()
print(classification_report(y_test, y_pred_test_rfm))
print("***************************")
print("Accuracy Train Score-",accuracy_score(y_train, y_pred_train_rfm))
print()
print("Accuracy Test Score-",accuracy_score(y_test, y_pred_test_rfm))

rfm_roc_auc = roc_auc_score(y_test, y_pred_test_rfm)
rfm_roc_auc


fpr, tpr, thresholds = roc_curve(y_test, y_pred_test_rfm)

plt.figure()
plt.plot(fpr, tpr, label="ROC Curve (%0.2f)" % rfm_roc_auc)
plt.plot([0,1],[0,1],'k--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
plt.title("******Receiver Operating Characteristic With Area Under Curve (ROC-AUC)******
plt.legend(loc='lower right')
plt.show()
```

```
**************************
[[2971    0]
 [   0 3005]]

[[749  15]
 [  7 723]]
**************************
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      2971
         1.0       1.00      1.00      1.00      3005

    accuracy                           1.00      5976
   macro avg       1.00      1.00      1.00      5976
weighted avg       1.00      1.00      1.00      5976


              precision    recall  f1-score   support

         0.0       0.99      0.98      0.99       764
         1.0       0.98      0.99      0.99       730

    accuracy                           0.99      1494
   macro avg       0.99      0.99      0.99      1494
weighted avg       0.99      0.99      0.99      1494

**************************
Accuracy Train Score- 1.0

Accuracy Test Score- 0.9852744310575636
```



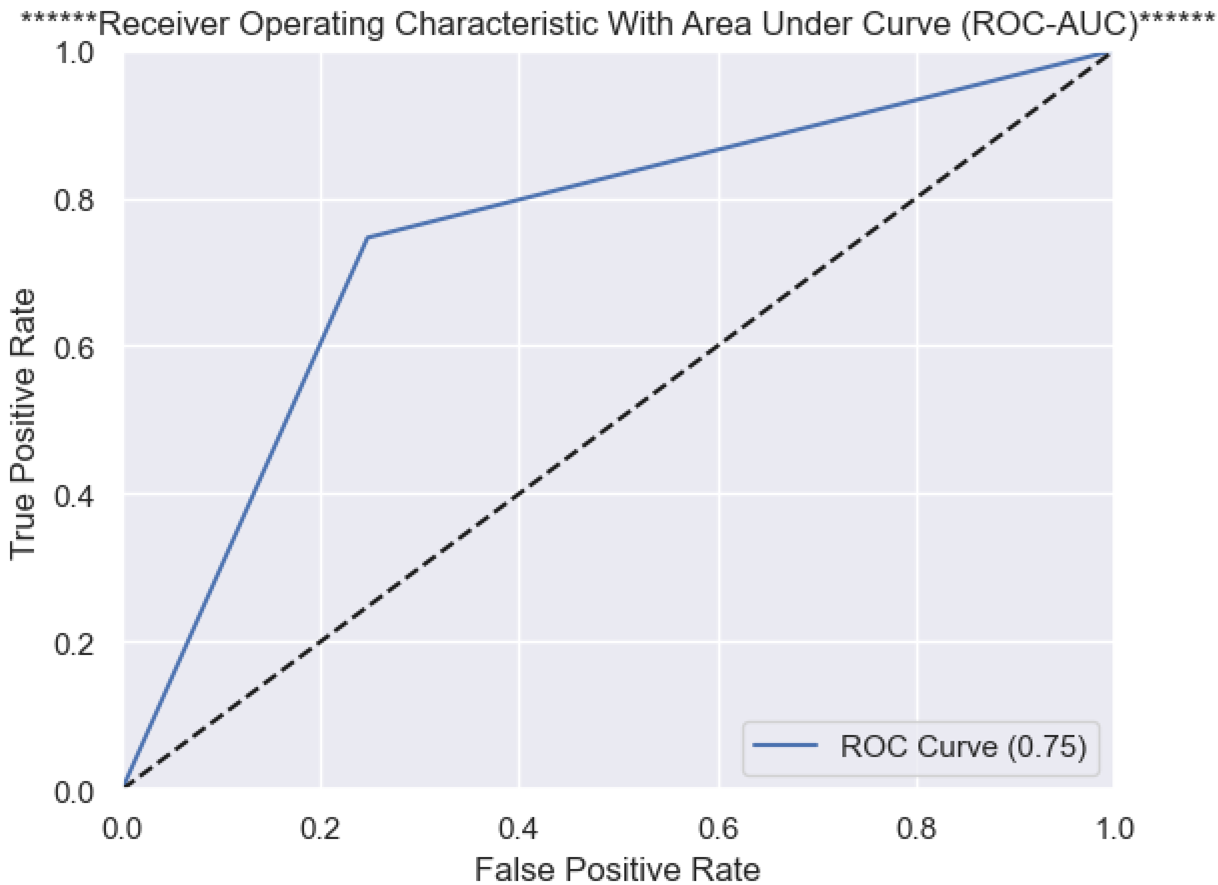
Model 6 - LogisticRegression

In [84]:

```
from sklearn.linear_model import LogisticRegression
lrm = LogisticRegression()
lrm.fit(x_train, y_train)
y_pred_train_lrm = lrm.predict(x_train)
y_pred_test_lrm = lrm.predict(x_test)
print("***************************")
print(confusion_matrix(y_train, y_pred_train_lrm))
print()
print(confusion_matrix(y_test, y_pred_test_lrm))
print("***************************")
print(classification_report(y_train, y_pred_train_lrm))
print()
print(classification_report(y_test, y_pred_test_lrm))
print("***************************")
print("Accuracy Train Score-",accuracy_score(y_train, y_pred_train_lrm))
print()
print("Accuracy Test Score-",accuracy_score(y_test, y_pred_test_lrm))

lrm_roc_auc = roc_auc_score(y_test, y_pred_test_lrm)
lrm_roc_auc


fpr, tpr, thresholds = roc_curve(y_test, y_pred_test_lrm)

plt.figure()
plt.plot(fpr, tpr, label="ROC Curve (%0.2f)" % lrm_roc_auc)
plt.plot([0,1],[0,1],'k--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("******Receiver Operating Characteristic With Area Under Curve (ROC-AUC)******
plt.legend(loc='lower right')
plt.show()
```

```
***************************
[[2294  677]
 [ 665 2340]]

[[575 189]
 [185 545]]
***************************
              precision    recall  f1-score   support

         0.0       0.78      0.77      0.77      2971
         1.0       0.78      0.78      0.78      3005

    accuracy                           0.78      5976
   macro avg       0.78      0.78      0.78      5976
weighted avg       0.78      0.78      0.78      5976


              precision    recall  f1-score   support

         0.0       0.76      0.75      0.75       764
         1.0       0.74      0.75      0.74       730

    accuracy                           0.75      1494
   macro avg       0.75      0.75      0.75      1494
weighted avg       0.75      0.75      0.75      1494

***************************
```

```
Accuracy Train Score- 0.7754350736278447

Accuracy Test Score- 0.749665327978581
```



From above 6 model, XGBoost gives best result so lets test our hidden data with this model

In [85]:
```
#As train data we did scaling so lets transform test data
test_dataset_og=test_dataset
test_dataset = sc.transform(test_dataset)
```

In [86]:
```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from xgboost import XGBClassifier
y_pred_test_xgb = xgb.predict(test_dataset)
```

In [87]:
```
y_pred_test_xgb=pd.DataFrame(y_pred_test_xgb)
```

In [88]:
```
y_pred_test_xgb.shape
```

Out[88]:
```
(2630, 1)
```

In [89]:
```
test_dataset_og=pd.DataFrame(test_dataset_og,columns=x_smote.columns)
```

In [90]:
```
test_dataset_og.head(4)
```

Out[90]:

	Age	HomeToWork	Gender	HourInWeek	Involvement	WorkLifeBalance	JobSatisfaction	ESOPs	NumCompar
0	18.0	9.0	1	80.0	3.0	2.0	3.0	1.0	
1	20.0	28.0	0	59.0	1.5	3.0	1.0	1.0	
2	50.0	19.0	0	76.0	3.0	3.0	5.0	0.0	

3	32.0	23.0	0	73.0	5.0	2.0	3.0	0.0

Appending result with Orignal datasets

In [91]:
```
test_dataset_og['Attrition']=y_pred_test_xgb
```

In [92]:
```
test_dataset_og.sample(4)
```

Out[92]:

	Age	HomeToWork	Gender	HourInWeek	Involvement	WorkLifeBalance	JobSatisfaction	ESOPs	NumCom
2320	53.0	30.0	1	42.0	3.0	2.0	1.0	0.0	
1491	33.0	10.0	0	77.0	3.0	2.0	5.0	1.0	
2000	27.0	3.0	0	68.0	4.0	3.0	3.0	0.0	
1716	55.0	22.0	0	67.0	3.0	1.0	5.0	0.0	

The end