SUBMITTED BY: MONIKA SINGH

Blogs /Articles Details

Problem Statement:

Every year a lot of companies hire a number of employees. The companies invest time and money in training those employees, not just this but there are training programs within the companies for their existing employees as well. The aim of these programs is to increase the effectiveness of their employees. But where HR Analytics fit in this? and is it just about improving the performance of employees?

HR Analytics:

Human resource analytics (HR analytics) is an area in the field of analytics that refers to applying analytic processes to the human resource department of an organization in the hope of improving employee performance and therefore getting a better return on investment. HR analytics does not just deal with gathering data on employee efficiency. Instead, **it aims to provide insight into each process by gathering data and then using it to make relevant decisions about how to improve these processes.**

Attrition in HR:

Attrition in human resources refers to the gradual loss of employees overtime. In general, relatively high attrition is problematic for companies. HR professionals often assume a leadership role in designing company compensation programs, work culture, and motivation systems that help the organization retain top employees.

How does Attrition affect companies? and how does HR Analytics help in analyzing attrition? We will discuss the first question here and for the second question, we will write the code and try to understand the process step by step.

Attrition affecting Companies:

A major problem in high employee attrition is its cost to an organization. Job postings, hiring processes, paperwork, and new hire training are some of the common expenses of losing employees and replacing them. Additionally, regular employee turnover prohibits your organization from increasing its collective knowledge base and experience over time. This is especially concerning if your business is customer-facing, as customers often prefer to interact with familiar people. Errors and issues are more likely if you constantly have new workers.

Importing Libraries

In [10]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [11]:

```
df=pd.read_csv(r'HR-Employee-Attrition.csv')
df
```

Out[11]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfacti
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	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061	...	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062	...	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064	...	
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065	...	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	2068	...	

1470 rows × 35 columns

In [12]:

```
#checking the shape of the data
df.shape
```

Out[12]: (1470, 35)

In [13]:

```
#checking the info of the data set
df.info()
```

```
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
```

```
RelationshipSatisfaction     int64
StandardHours                int64
StockOptionLevel             int64
TotalWorkingYears            int64
TrainingTimesLastYear        int64
WorkLifeBalance              int64
YearsAtCompany               int64
YearsInCurrentRole           int64
YearsSinceLastPromotion      int64
YearsWithCurrManager         int64
dtype: object
```

In this dataset we have 9 attributes of object data types and 27 attributes of integer.

In [15]:

```
#checking the null values
df.isnull().sum()
```

Out[15]:

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
```

In [14]:

```
#checking the data types
df.dtypes
```

Out[14]:

```
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
```

In [15]:

```
#checking the null values
df.isnull().sum()
```

Out[15]:

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
```

Heatmap

In [16]:

```
#plotting the heatmap for checking the null value
plt.figure(figsize=(15,10))
sns.heatmap(df.isnull());
```

In [17]:

```
#stats
df.describe()
```

Out[17]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...	Rela
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	...	
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	...	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	...	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	...	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	...	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	...	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	...	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	...	

8 rows × 26 columns

In [18]:

```
#checking the unique values in column
print(df['EmployeeCount'].unique())
print(df['StandardHours'].unique())
```

```
[1]
[80]
```

In [20]:

```
# dropping duplicate cols
df_1 = df.drop(['EmployeeCount','StandardHours','EmployeeNumber'],axis=1)
```

Categorcial Attributes

In [21]:

```
# segregating the object datatype.
ob=df_1.select_dtypes(include='object')
```

In [22]:

```
ob
```

Out[22]:

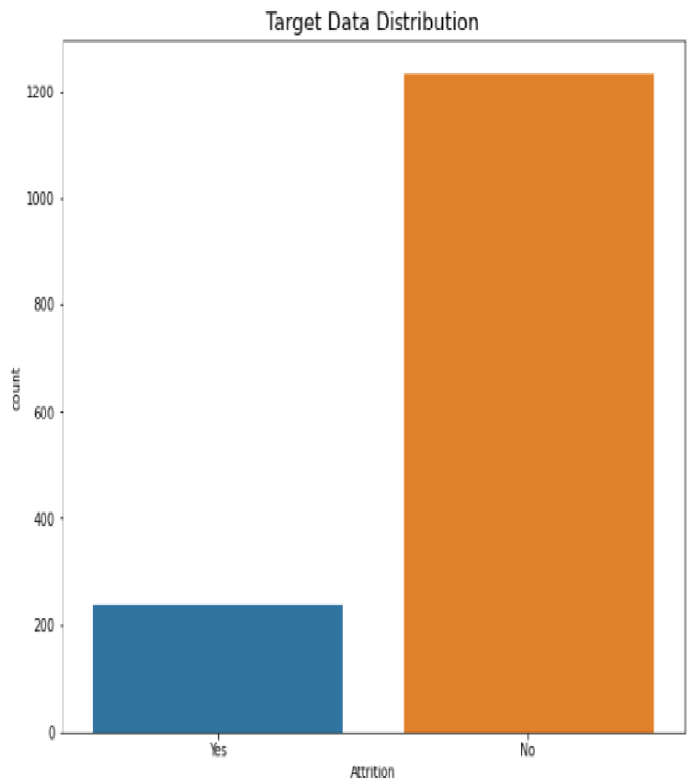
	Attrition	BusinessTravel	Department	EducationField	Gender	JobRole	MaritalStatus	Over18	OverTime
0	Yes	Travel_Rarely	Sales	Life Sciences	Female	Sales Executive	Single	Y	Yes
1	No	Travel_Frequently	Research & Development	Life Sciences	Male	Research Scientist	Married	Y	No
2	Yes	Travel_Rarely	Research & Development	Other	Male	Laboratory Technician	Single	Y	Yes
3	No	Travel_Frequently	Research & Development	Life Sciences	Female	Research Scientist	Married	Y	Yes
4	No	Travel_Rarely	Research & Development	Medical	Male	Laboratory Technician	Married	Y	No
...	...	...	...	...	...	...	...	...	...
1465	No	Travel_Frequently	Research & Development	Medical	Male	Laboratory Technician	Married	Y	No
1466	No	Travel_Rarely	Research & Development	Medical	Male	Healthcare Representative	Married	Y	No
1467	No	Travel_Rarely	Research & Development	Life Sciences	Male	Manufacturing Director	Married	Y	Yes
1468	No	Travel_Frequently	Sales	Medical	Male	Sales Executive	Married	Y	No
1469	No	Travel_Rarely	Research & Development	Medical	Male	Laboratory Technician	Married	Y	No

1470 rows × 9 columns

Data Visualization

In [24]:

```
plt.figure(figsize=(10,8))
plt.title('Target Data Distribution',fontsize=16)
sns.countplot(df_1['Attrition'],data=df_1);
```



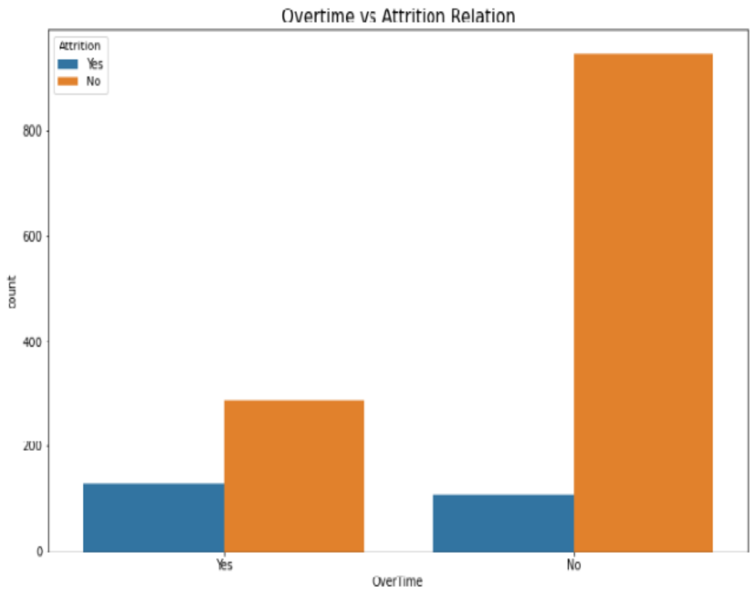
Data is imbalanced in nature.

In [25]:

Data is imbalanced in nature.

In [25]:

```
plt.figure(figsize=(12,8))
plt.title('Overtime vs Attrition Relation',fontsize=15)
sns.countplot(df_1['OverTime'],hue='Attrition',data=df_1);
```



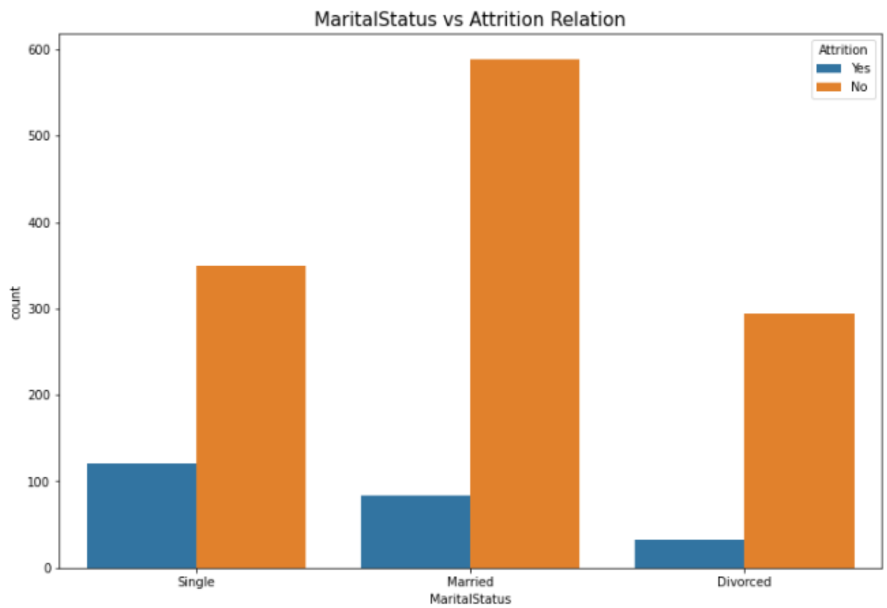
People who do over time have higher chances to left the company as compair to person who don't do overtime.

In [26]:

```
plt.figure(figsize=(12,8))
```

People who do over time have higher chances to left the company as compair to person who don't do overtime.

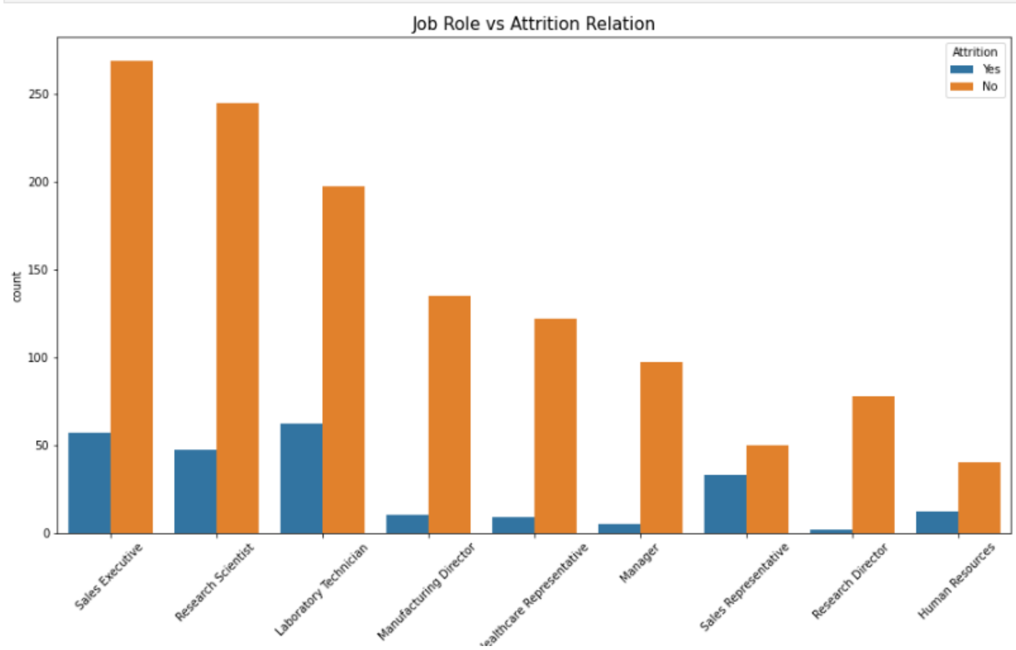
In [26]:
```
plt.figure(figsize=(12,8))
plt.title('MaritalStatus vs Attrition Relation',fontsize=15)
sns.countplot(df_1['MaritalStatus'],hue='Attrition',data=df_1);
```



unmarried people have higher tendency to leave the company as comapired to Married

Divorced/seperated has lowest chanaces to left the company.

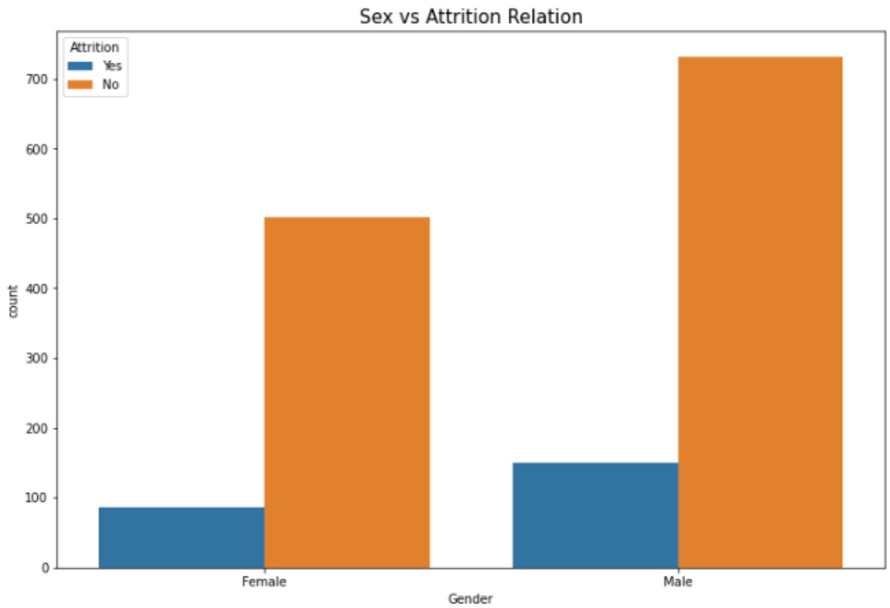
In [27]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation=45)
plt.title('Job Role vs Attrition Relation',fontsize=15)
sns.countplot(df_1['JobRole'],hue='Attrition',data=df_1);
```



Laboratory Technician ,Sales Executive, Sales representatives research scientis have higher tendency to leave job

In [28]:

```
plt.figure(figsize=(12,8))
plt.title('Sex vs Attrition Relation',fontsize=15)
sns.countplot(df_1['Gender'],hue='Attrition',data=df_1);
```

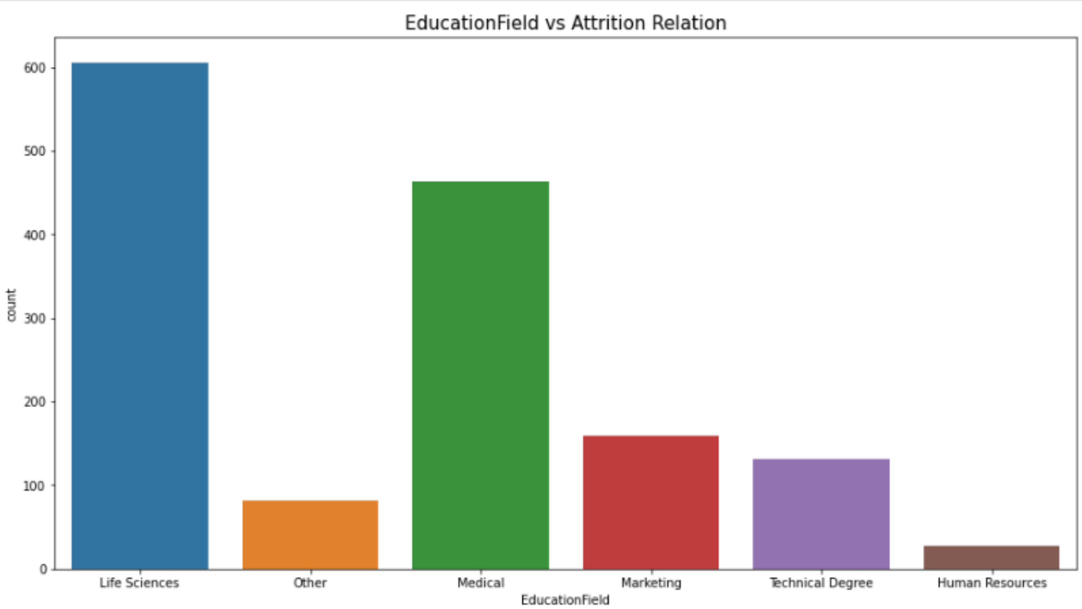


In [29]:

```
plt.figure(figsize=(10,8))
plt.title('Sex vs Attrition',fontsize=15)
sns.countplot(df_1['Gender'],data=df_1[df_1['Attrition']=='Yes']);
```

In [30]:

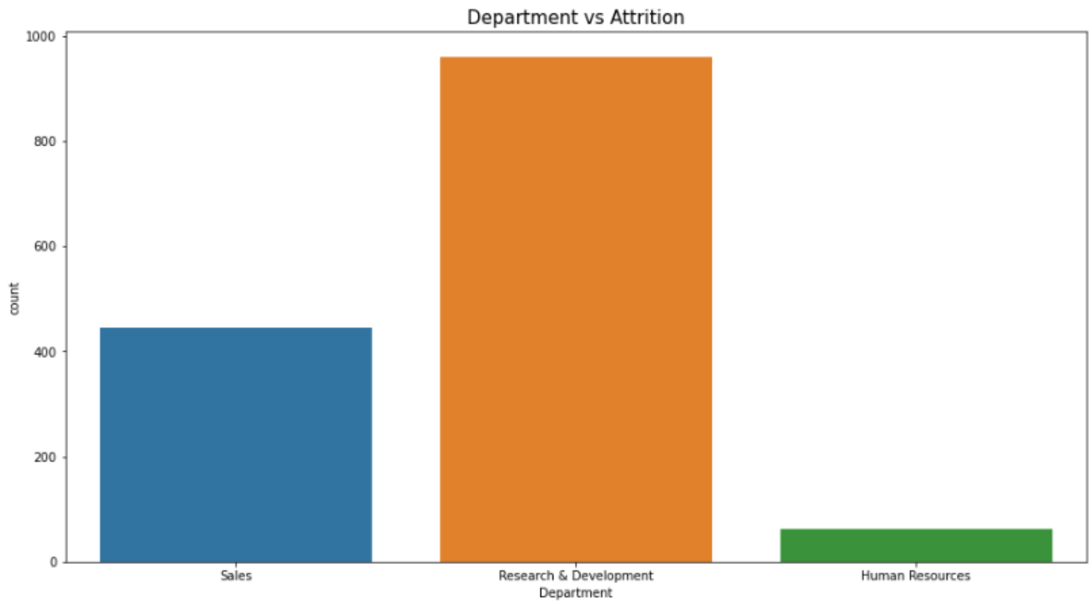
```
plt.figure(figsize=(15,8))
plt.title('EducationField vs Attrition Relation',fontsize=15)
sns.countplot(df_1['EducationField'],data=df_1[df_1['Attrition']=='Yes']);
```



life science and Medical two major Education fields which has higher demand in market and people are switch their job more frequent.

In [31]:

```
plt.figure(figsize=(15,8))
plt.title('Department vs Attrition',fontsize=15)
sns.countplot(df_1['Department'],data=df_1[df_1['Attrition']=='Yes']);
```



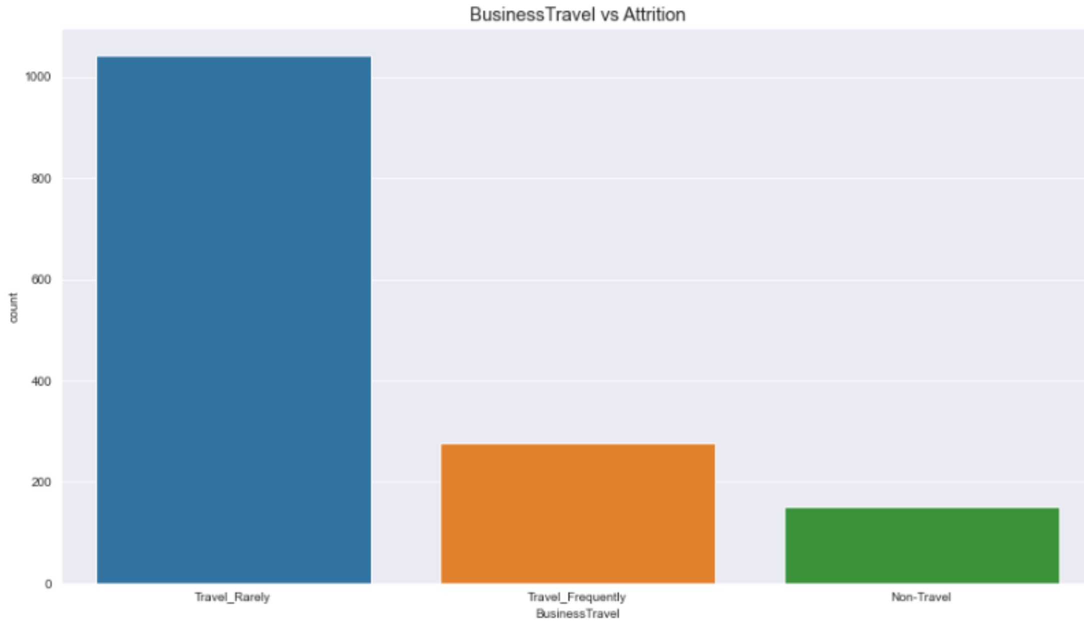
people who works in Research & development have higher chances to leave the company.

In [32]:

```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('BusinessTravel vs Attrition',fontsize=15)
sns.countplot(df_1['BusinessTravel'],hue='Attrition',data=df_1);
```

In [33]:

```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('BusinessTravel vs Attrition',fontsize=15)
sns.countplot(df_1['BusinessTravel'],data=df_1[df_1['Attrition']=='Yes']);
```

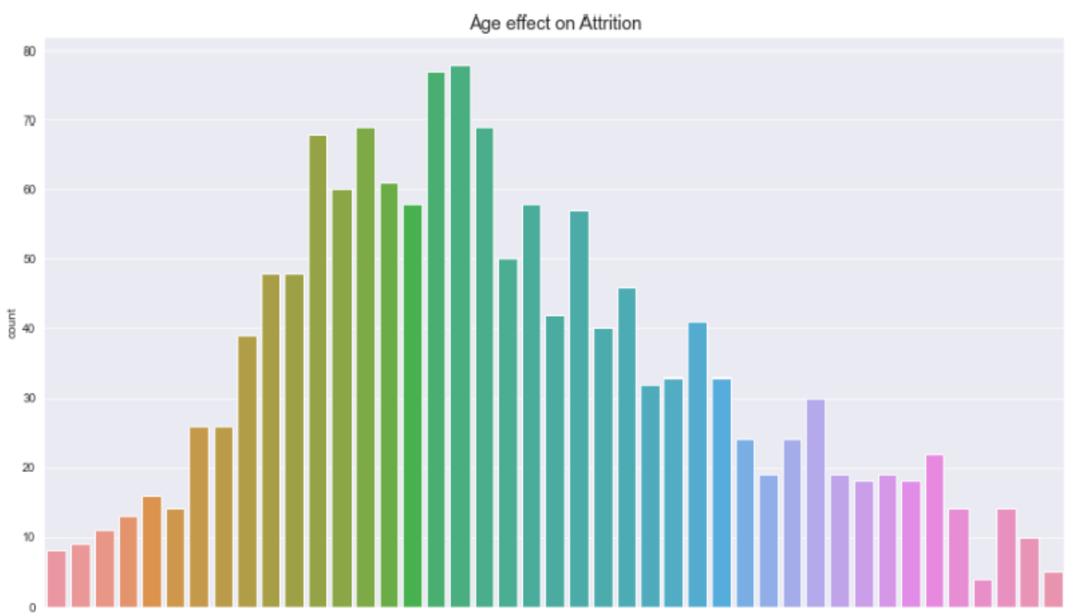


Both countplot it is clear that an employee who travels rarely to other places have higher chances to left job.

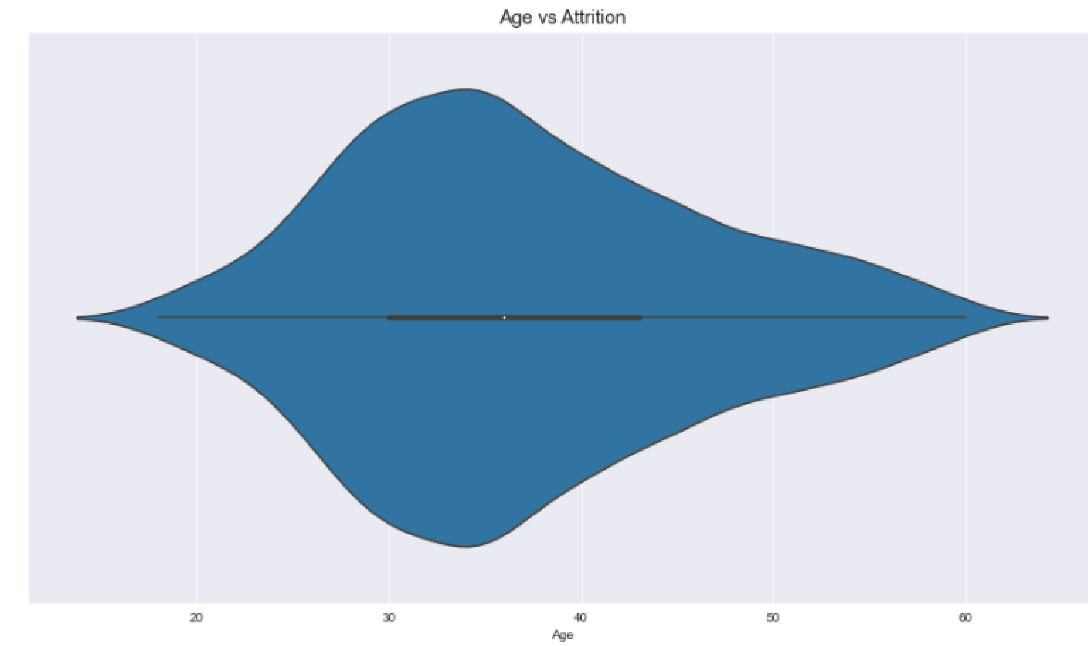
Both countplot it is clear that an employee who travels rarely to other places have higher chances to left job.

In [34]:

```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('Age effect on Attrition',fontsize=15)
sns.countplot(df_1['Age'],data=df_1[df_1["Attrition"]=='Yes']);
```

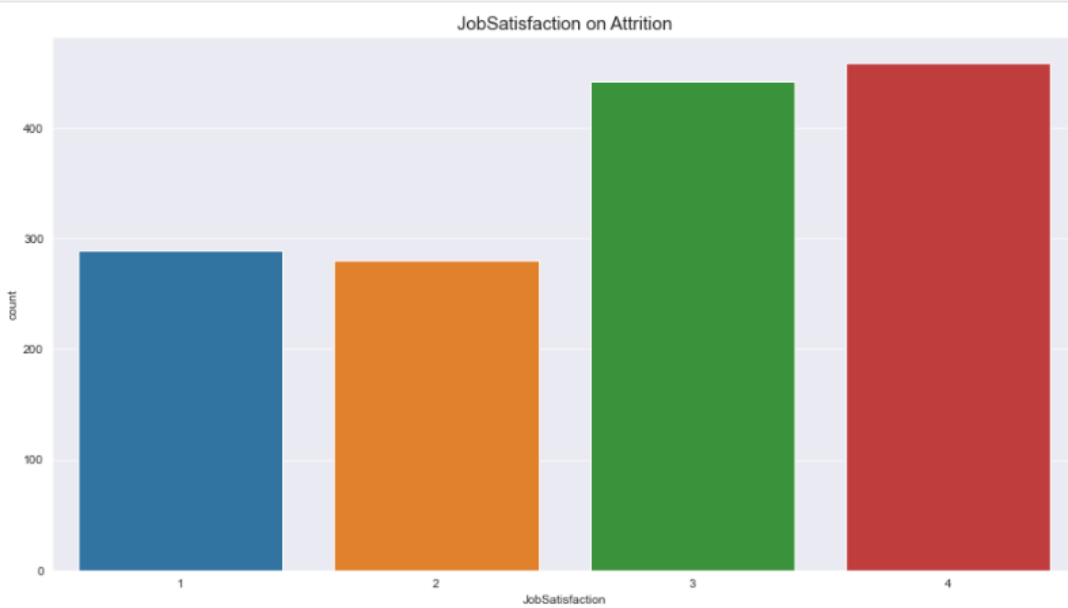


In [35]:
```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('Age vs Attrition',fontsize=15)
sns.violinplot(df_1['Age'],data=df_1[df_1['Attrition']=='Yes']);
```



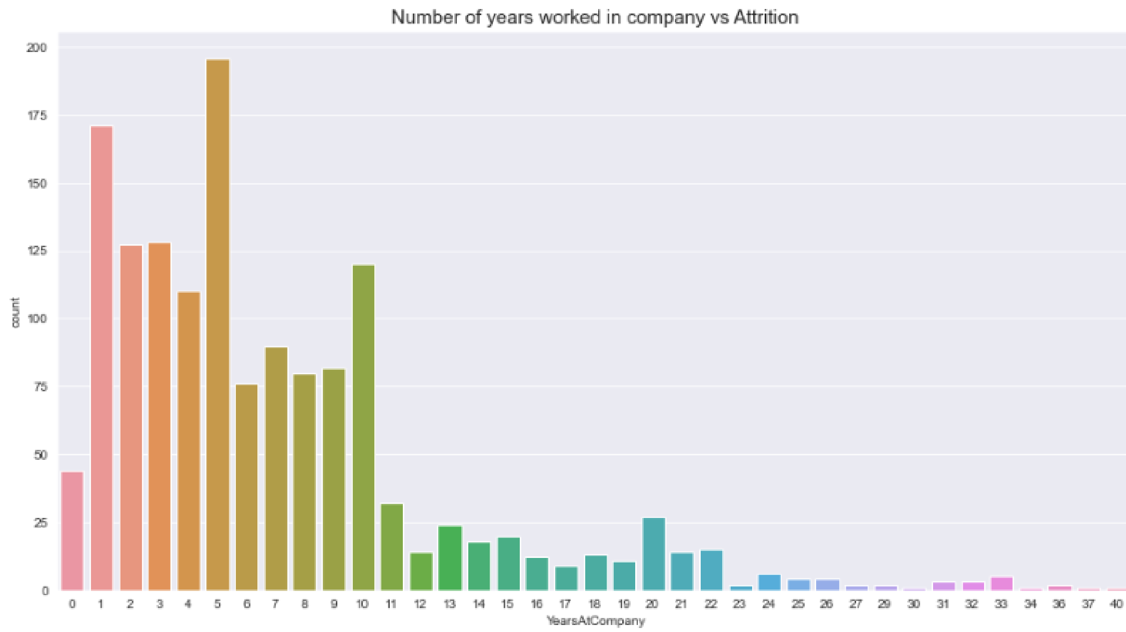
Two plots we can visualize that a employees whose age group is between 28-39 has highest tendency to job change

In [36]:
```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('JobSatisfaction on Attrition',fontsize=15)
sns.countplot(df_1['JobSatisfaction'],data=df_1[df_1['Attrition']=='Yes']);
```



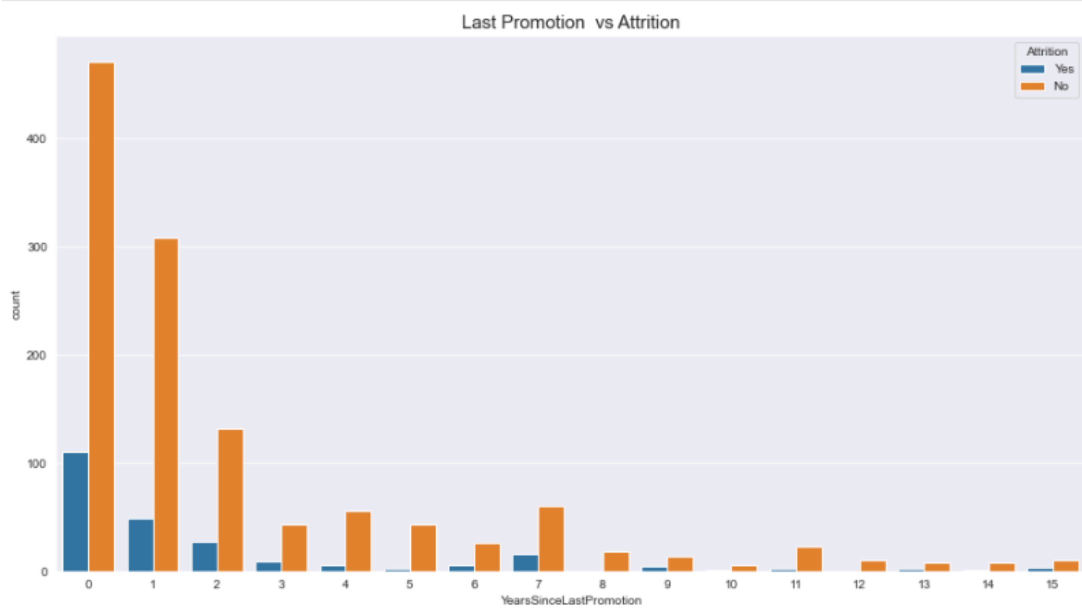
In [37]:

```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('Number of years worked in company vs Attrition',fontsize=15)
sns.countplot(df_1['YearsAtCompany'],data=df_1[df_1['Attrition']=='Yes']);
```



In [38]:

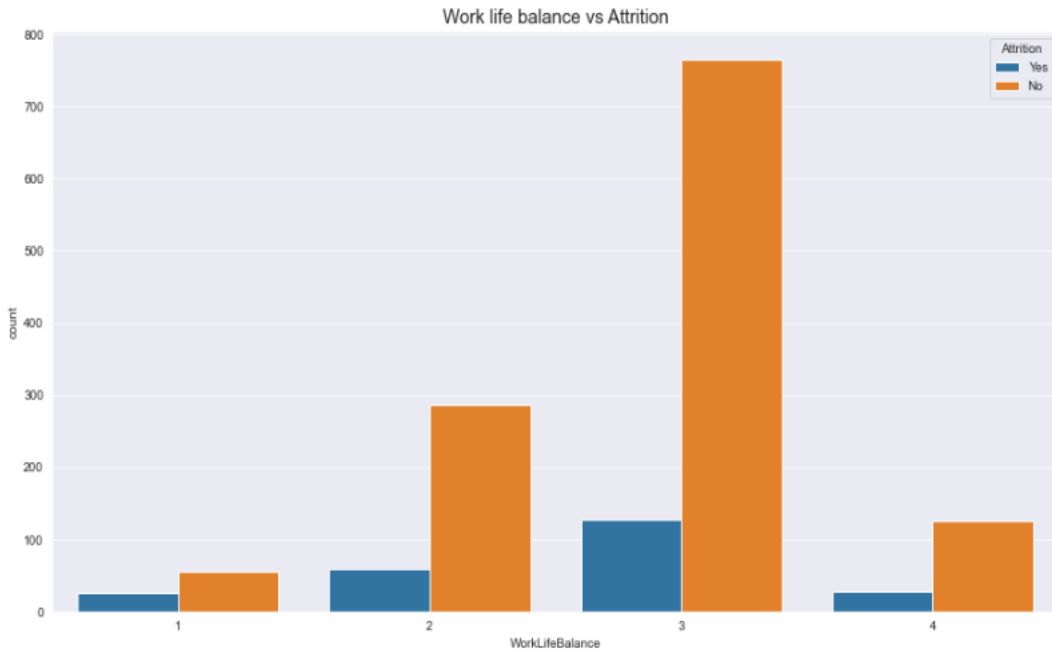
```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('Last Promotion  vs Attrition',fontsize=15)
sns.countplot(df_1['YearsSinceLastPromotion'],hue='Attrition',data=df_1);
```



who either got promoted recently or Joined recently has the highest chances of job change. Employee who did not get promtion for 1-2 years also hav
for job change.

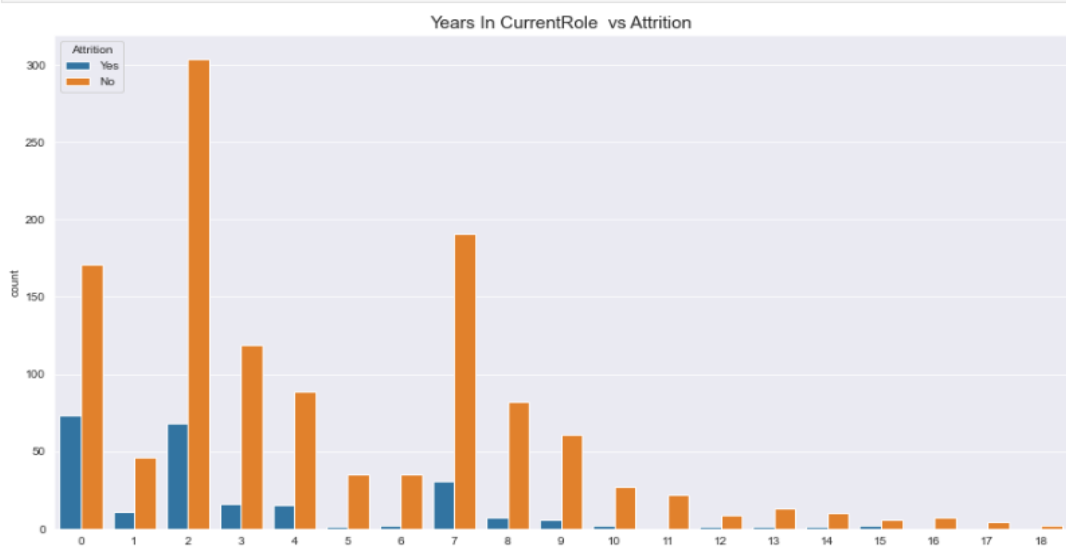
In [40]:

```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('Work life balance vs Attrition',fontsize=15)
sns.countplot(df_1['WorkLifeBalance'],hue='Attrition',data=df_1);
```



In [39]:

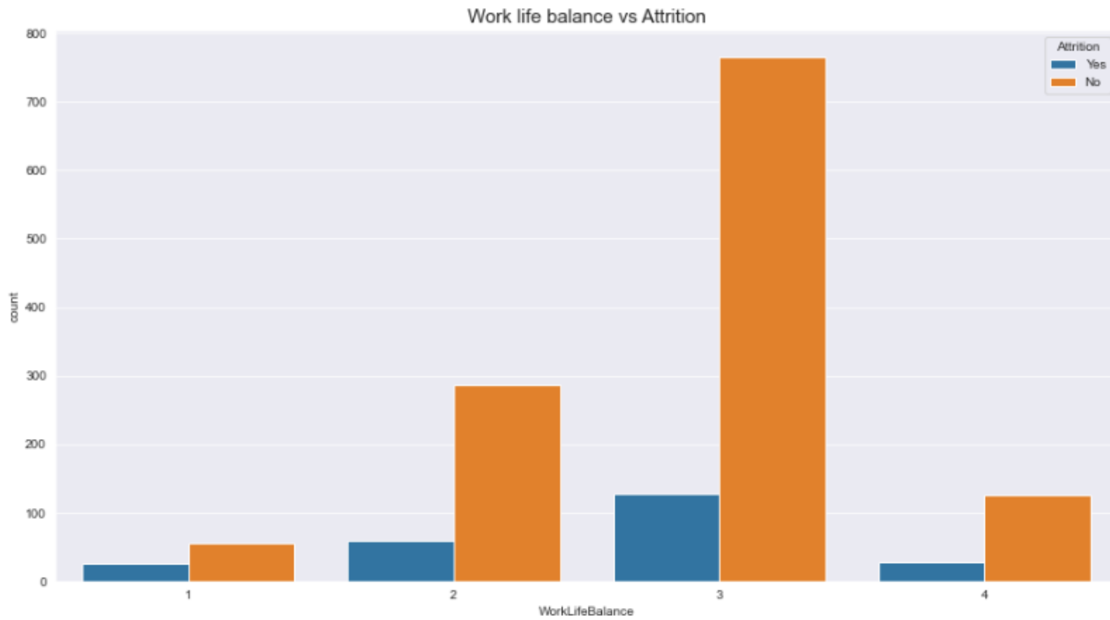
```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('Years In CurrentRole  vs Attrition',fontsize=15)
sns.countplot(df_1['YearsInCurrentRole'],hue='Attrition',data=df_1);
```



peoson who has experience of 2 years or who joined recently and woking on same profile has fair chances to left job

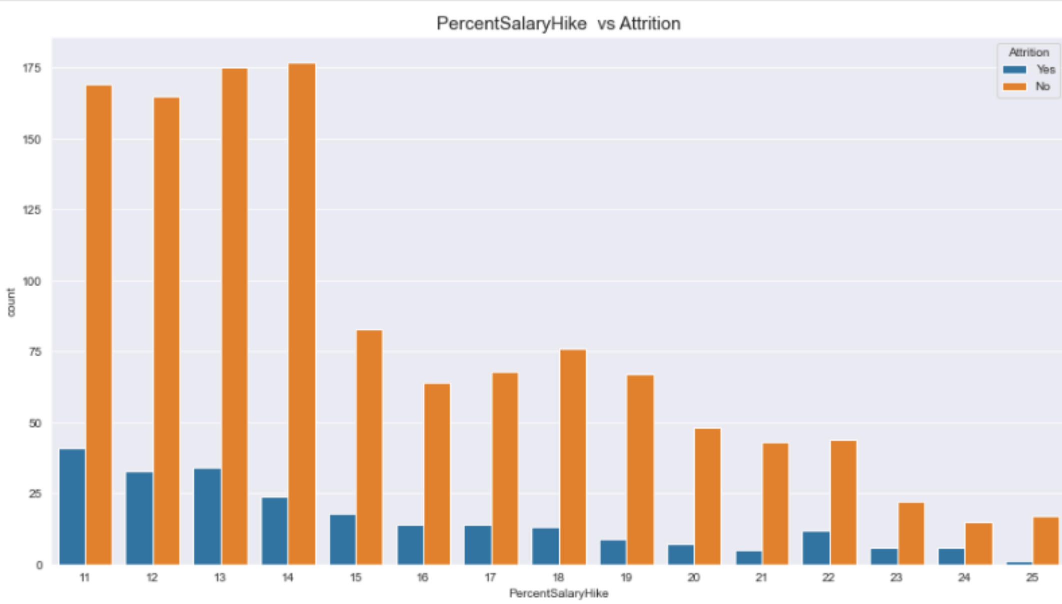
In [40]:

```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('Work life balance vs Attrition',fontsize=15)
sns.countplot(df_1['WorkLifeBalance'],hue='Attrition',data=df_1);
```



In [41]:

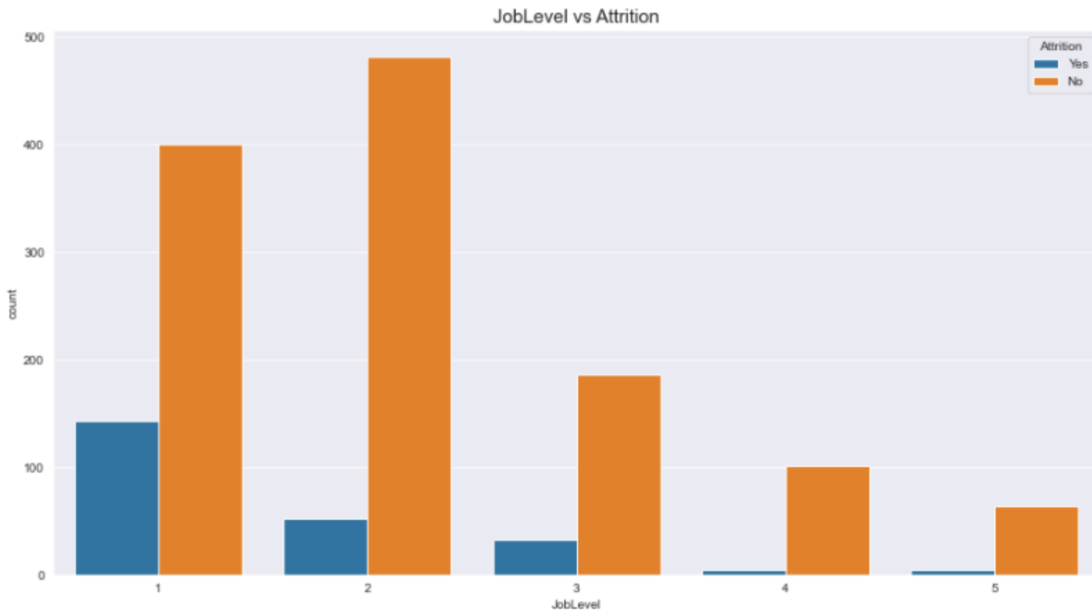
```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('PercentSalaryHike  vs Attrition',fontsize=15)
sns.countplot(df_1['PercentSalaryHike'],hue='Attrition',data=df_1);
```



It is clearly visible that an employee who got least salary hike in year have higher tendency for job change.

In [42]:

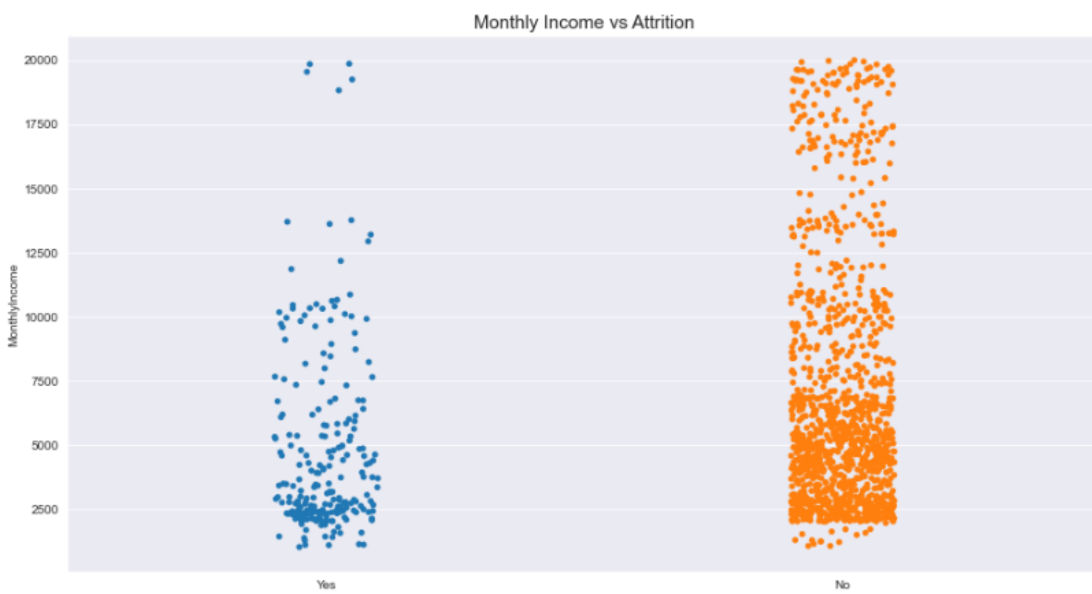
```
plt.figure(figsize=(15,8))
sns.set_style('darkgrid')
plt.title('JobLevel vs Attrition',fontsize=15)
sns.countplot(df_1['JobLevel'],hue='Attrition',data=df_1);
```



Employee who has job band/level 1 has tendency to change job more frequent then job level 4 or 5

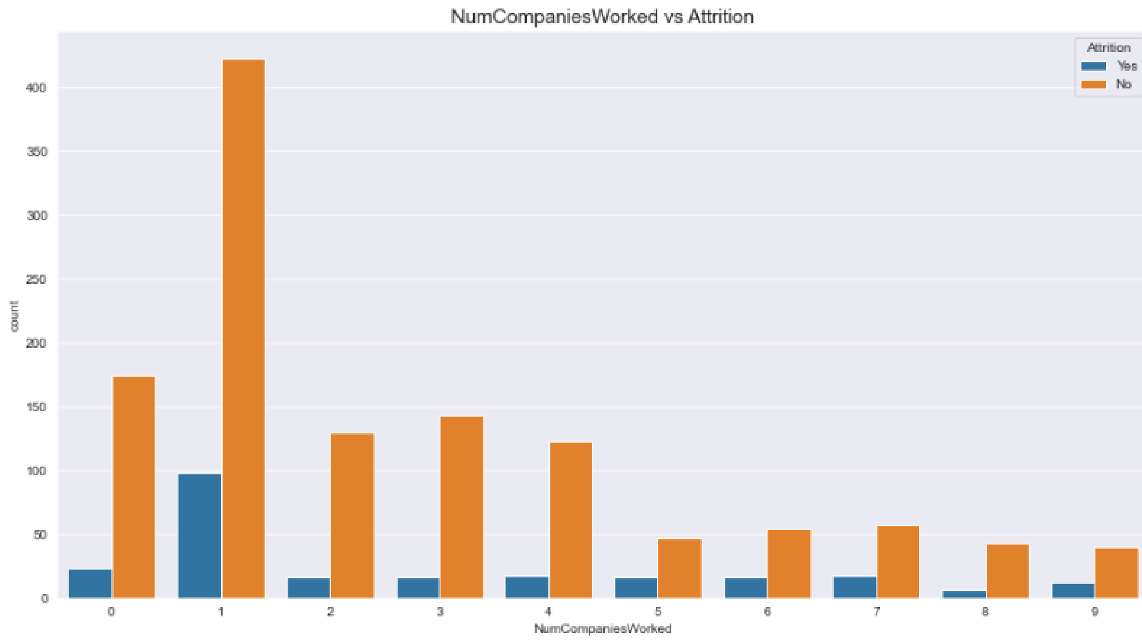
In [43]:

```
plt.figure(figsize=(15,8))
plt.title('Monthly Income vs Attrition',fontsize=15)
sns.stripplot(df_1['Attrition'],df_1['MonthlyIncome'],data=df_1);
```



In [44]:

```
plt.figure(figsize=(15,8))
plt.title('NumCompaniesWorked vs Attrition',fontsize=15)
sns.countplot(df_1['NumCompaniesWorked'],hue='Attrition',data=df_1);
```

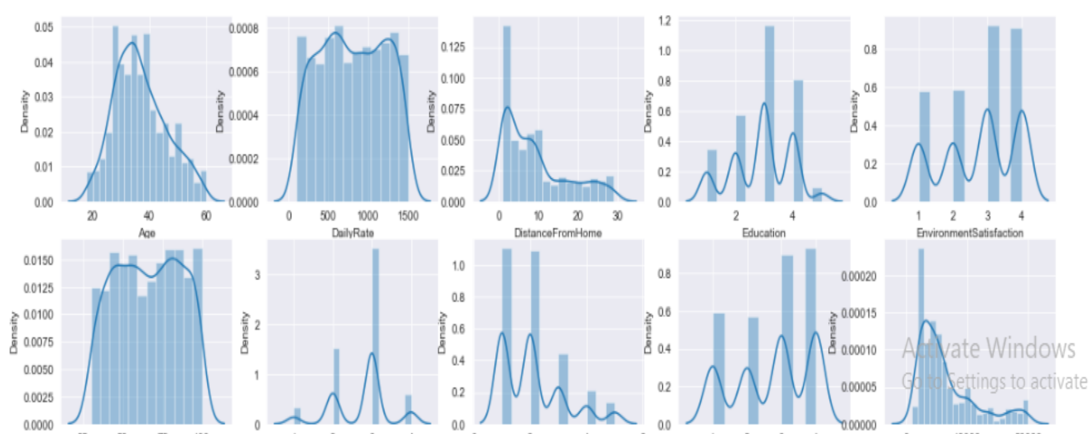


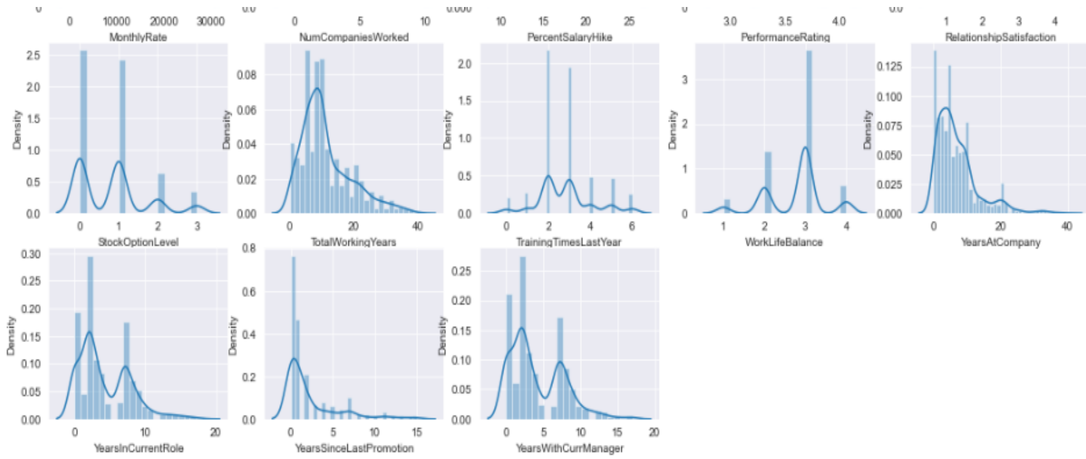
employee who has worked with 1 company previously are more looking for job change

Data Distribution

In [45]:

```
dist=df_1.select_dtypes(exclude='object')
plt.figure(figsize=(18,15))
plot=1
for col in dist:
    if plot<=25:
        plt.subplot(5,5,plot)
        sns.distplot(df_1[col])
        plt.xlabel(col)
        plot=plot+1
plt.show();
```





1- Most of our data is Normally distributed.

2- Attributes like Total working years,Yearsatcompany, Years since last promotion etc. are right skewed.

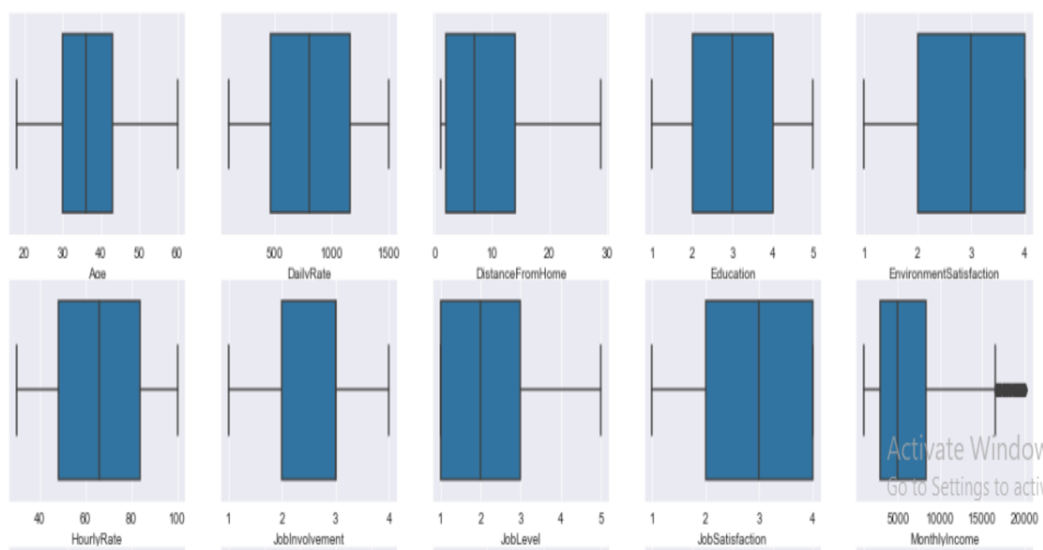
3-This shows some outliers must be present in our dataset.

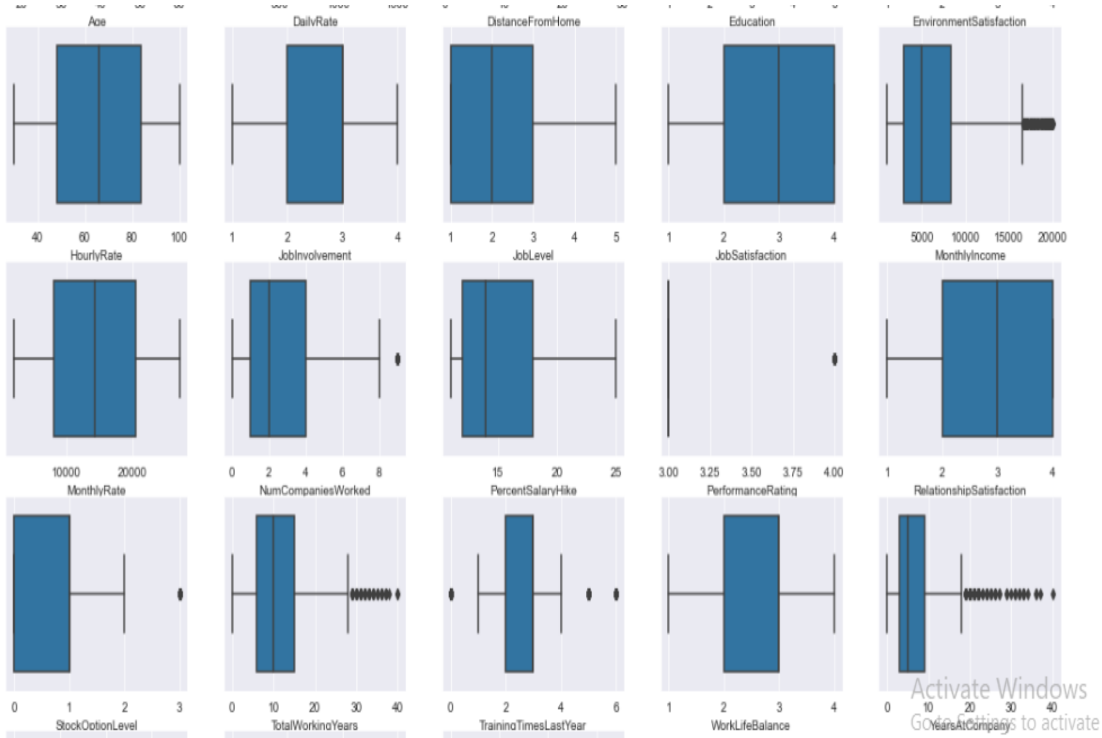
4- we will remove skewness by some trasformation methods.

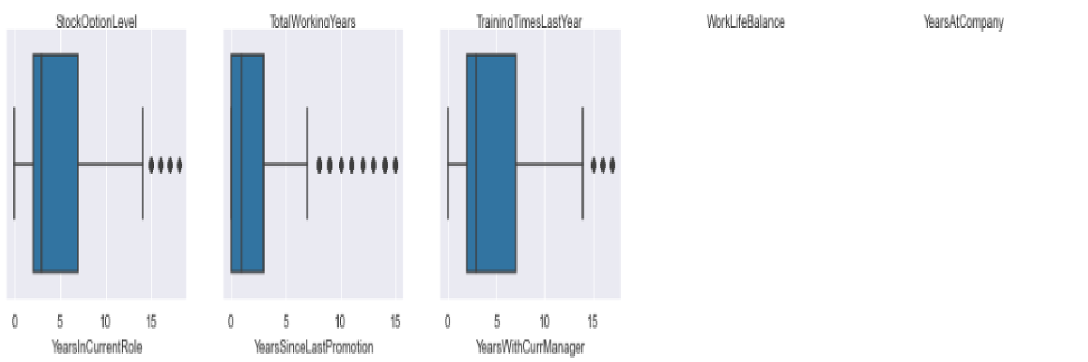
Finding outliers

In [46]:

```
plt.figure(figsize=(18,15))
plot=1
for col in dist:
    if plot<=25:
        plt.subplot(5,5,plot)
        sns.boxplot(df_1[col])
        plt.xlabel(col)
        plot=plot+1
plt.show();
```







In [47]:

```
## Removing Outliers
from scipy.stats import zscore
z =np.abs(zscore(dist))
print(z.shape)
df_1 = df_1.loc[(z<3).all(axis=1)]
print(df_1.shape)
```

```
(1470, 23)
(1387, 32)
```

Skewness

In [48]:
```
df_1.skew()
```

Out[48]:
```
Age                         0.472280
DailyRate                  -0.017078
DistanceFromHome            0.954752
Education                  -0.289024
EnvironmentSatisfaction    -0.325285
HourlyRate                 -0.030481
JobInvolvement             -0.501401
JobLevel                    1.126075
JobSatisfaction            -0.345612
MonthlyIncome               1.544770
MonthlyRate                 0.030596
NumCompaniesWorked          1.037715
PercentSalaryHike           0.800592
PerformanceRating           1.931566
RelationshipSatisfaction   -0.295686
StockOptionLevel            0.962332
TotalWorkingYears           1.034487
TrainingTimesLastYear       0.577614
WorkLifeBalance            -0.557100
YearsAtCompany              1.248623
YearsInCurrentRole          0.726675
YearsSinceLastPromotion     1.756335
YearsWithCurrManager        0.694506
dtype: float64
```

```
In [49]: for i in dist:
             if df_1[i].skew()>.55:
                 df_1[i]=np.log1p(df_1[i])
```

```
In [50]: df_1.skew()
```

```
Out[50]: Age                         0.472280
         DailyRate                  -0.017078
         DistanceFromHome           -0.031570
         Education                  -0.289024
         EnvironmentSatisfaction    -0.325285
         HourlyRate                 -0.030481
         JobInvolvement             -0.501401
         JobLevel                    0.497167
         JobSatisfaction            -0.345612
         MonthlyIncome               0.318873
         MonthlyRate                 0.030596
         NumCompaniesWorked          0.101288
         PercentSalaryHike           0.496106
         PerformanceRating           1.931566
         RelationshipSatisfaction   -0.295686
         StockOptionLevel            0.275912
         TotalWorkingYears          -0.728348
         TrainingTimesLastYear      -1.044321
         WorkLifeBalance            -0.557100
         YearsAtCompany             -0.379527
         YearsInCurrentRole         -0.390406
         YearsSinceLastPromotion     0.695348
         YearsWithCurrManager       -0.347018
         dtype: float64
```


Encoding Categorical Value

```
In [51]: from sklearn.preprocessing import LabelEncoder
         le=LabelEncoder()
         for col in ob:
             df_1[col]=le.fit_transform(df_1[col])
```

```
In [52]: df_1.head()
```

Out[52]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...	Performance
0	41	1	2	1102	2	0.693147	2	1	2	0	...	1.3
1	49	0	1	279	1	2.197225	1	1	3	1	...	1.6
2	37	1	2	1373	1	1.098612	2	4	4	1	...	1.3
3	33	0	1	1392	1	1.386294	4	1	4	0	...	1.3
4	27	0	2	591	1	1.098612	1	3	1	1	...	1.3

5 rows × 32 columns

Splitting the data

In [53]:
```
x=df_1.drop(['Attrition','Over18'],axis=1)
y=df_1[['Attrition']]
```

In [54]:
```
x
```

Out[54]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate	...	Perfor
0	41	2	1102	2	0.693147	2	1	2	0	94	...	
1	49	1	279	1	2.197225	1	1	3	1	61	...	
2	37	2	1373	1	1.098612	2	4	4	1	92	...	
3	33	1	1392	1	1.386294	4	1	4	0	56	...	
4	27	2	591	1	1.098612	1	3	1	1	40	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1465	36	1	884	1	3.178054	2	3	3	1	41	...	
1466	39	2	613	1	1.945910	1	3	4	1	42	...	
1467	27	2	155	1	1.609438	3	1	2	1	87	...	
1468	49	1	1023	2	1.098612	3	3	4	1	63	...	
1469	34	2	628	1	2.197225	3	3	2	1	82	...	

1387 rows × 30 columns

In [55]:
```
y
```

Out[55]:

	Attrition
0	1
1	0
2	1
3	0
4	0
...	...
1465	0
1466	0
1467	0
1468	0
1469	0

1387 rows × 1 columns

Feature Scaling

In [56]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x_scaled=ss.fit_transform(x)
x=pd.DataFrame(x_scaled,columns=x.columns)
x
```

Out[56]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	HourlyRate	...
0	0.536681	0.593126	0.734325	1.405373	-1.502086	-0.876177	-0.940815	-0.665328	-1.229911	1.388670	...
1	1.442111	-0.905354	-1.307769	-0.496337	0.253886	-1.853858	-0.940815	0.251978	0.813067	-0.239091	...
2	0.083966	0.593126	1.406752	-0.496337	-1.028716	-0.876177	1.305159	1.169285	0.813067	1.290017	...
3	-0.368749	-0.905354	1.453896	-0.496337	-0.692855	1.079185	-0.940815	1.169285	-1.229911	-0.485721	...
4	-1.047821	0.593126	-0.533609	-0.496337	-1.028716	-1.853858	0.556501	-1.582635	0.813067	-1.274939	...
...	...	...	...	...	...	...	...	...	...	...	...
1382	-0.029213	-0.905354	0.193406	-0.496337	1.398978	-0.876177	0.556501	0.251978	0.813067	-1.225613	...
1383	0.310324	0.593126	-0.479021	-0.496337	-0.039518	-1.853858	0.556501	1.169285	0.813067	-1.176286	...
1384	-1.047821	0.593126	-1.615447	-0.496337	-0.432340	0.101504	-0.940815	-0.665328	0.813067	1.043387	...
1385	1.442111	-0.905354	0.538304	1.405373	-1.028716	0.101504	0.556501	1.169285	0.813067	-0.140439	...
1386	-0.255570	0.593126	-0.441802	-0.496337	0.253886	0.101504	0.556501	-0.665328	0.813067	0.796757	...

1387 rows × 30 columns

feature importance

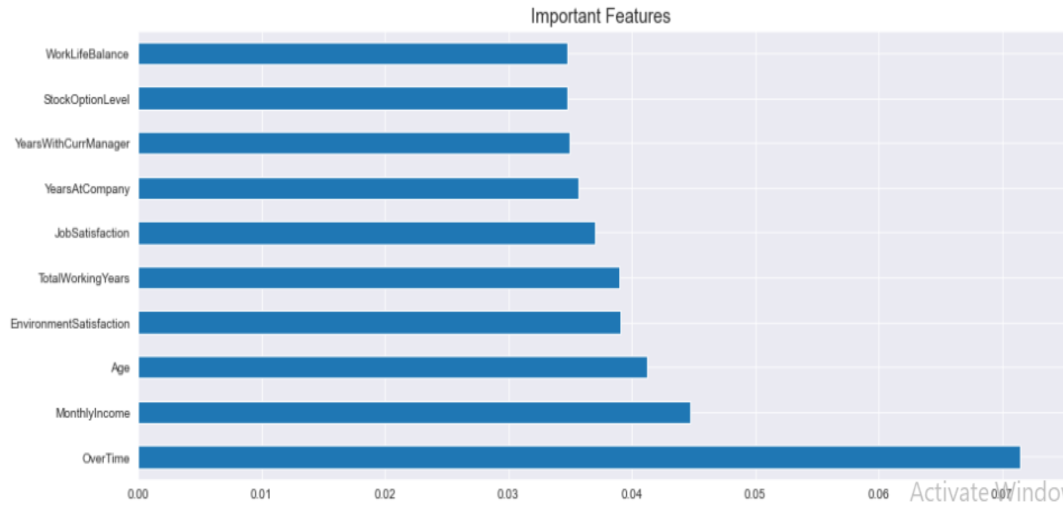
In [57]:
```
from sklearn.ensemble import ExtraTreesClassifier
extra=ExtraTreesClassifier()
extra.fit(x,y)
```

Out[57]: ExtraTreesClassifier()

In [58]:
```
print(extra.feature_importances_)
```

```
[0.04126763 0.02577878 0.03179439 0.02407701 0.03415031 0.02660997
 0.02816611 0.03906499 0.02080244 0.03137638 0.03272339 0.03367664
 0.03427738 0.03701433 0.03214459 0.0447477  0.032548   0.03174293
 0.07149409 0.0313795  0.01416903 0.03185999 0.03476039 0.03896788
 0.02932368 0.03475591 0.0357093  0.03246993 0.02813328 0.03501406]
```

In [59]:
```
plt.figure(figsize=(15,6))
plt.title('Important Features',fontsize=15)
feat_importance=pd.Series(extra.feature_importances_,index=x.columns)
feat_importance.nlargest(10).plot(kind='barh')
plt.show();
```



PCA

In [60]:
```
from sklearn import decomposition
from sklearn.decomposition import PCA
covar_matrix=PCA(n_components=30)
```

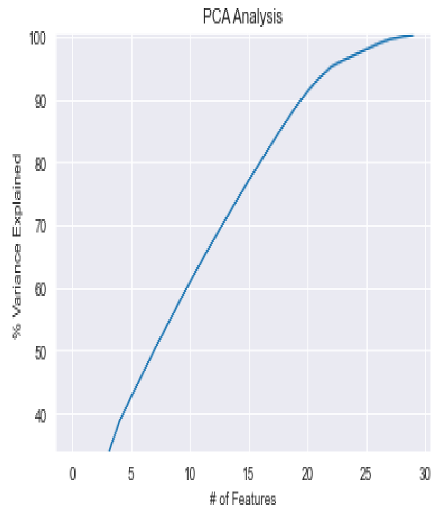
In [61]:
```
#Calculate Eigenvalues
covar_matrix.fit(x)  ## x should be scaled
variance = covar_matrix.explained_variance_ratio_ #calculate variance ratios

var=np.cumsum(np.round(covar_matrix.explained_variance_ratio_, decimals=3)*100)
var #cumulative sum of variance explained with [n] features
## draw the graph
plt.ylabel('% Variance Explained')
plt.xlabel('# of Features')
plt.title('PCA Analysis')
plt.ylim(34,100.5)
plt.style.context('seaborn-whitegrid')


plt.plot(var)
```

Out[61]: [<matplotlib.lines.Line2D at 0x1ec02bc39d0>]

Out[61]: [<matplotlib.lines.Line2D at 0x1ec02bc39d0>]



Build the models

In [82]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,roc_auc_score,f1_score,roc_curve,auc
```

In [83]:
```
def max_accuracy_score(clf,x,y):
    max_accuracy=0
    for i in range(42,100):
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i,stratify=y)
        clf.fit(x_train,y_train)
        pred=clf.predict(x_test)
        accuracy_check=accuracy_score(y_test,pred)
        if accuracy_check>max_accuracy:
            max_accuracy=accuracy_check
            final_r=i
    print('max accuracy score corresponding to',final_r,'is',max_accuracy*100)
    print('\n')
    print('cross validation score',cross_val_score(clf,x,y,scoring='accuracy').mean()*100)
    print('\n')
    print('Standard Deviation',cross_val_score(clf,x,y,scoring='accuracy').std()*100)
    print('\n')
    print('F1 score',f1_score(y_test,pred)*100)
    print('\n')
    print('Training accuracy',clf.score(x_train,y_train)*100)
    print('\n')
    print('Test Accuracy',clf.score(x_test,y_test)*100)
    print('\n')
    print('Confusion Matrix',confusion_matrix(y_test,pred))
    print('\n')
    print('Classification Report',classification_report(y_test,pred))
    print('\n')
    print('Roc_auc Score',roc_auc_score(y_test,pred)*100)
    return final_r
```



```
Classification Report               precision    recall  f1-score   support

           0       0.90      0.97      0.93       232
           1       0.71      0.43      0.54        46

    accuracy                           0.88       278
   macro avg       0.81      0.70      0.74       278
weighted avg       0.87      0.88      0.87       278



Roc_auc Score 70.01499250374813
```

Out[84]: 44

Decision Tree

In [85]:
```
dt = DecisionTreeClassifier()
max_accuracy_score(dt,x,y)
```

```
max accuracy score corresponding to 46 is 83.09352517985612


cross validation score 77.35942653819183


Standard Deviation 1.7987936464167595


F1 score 40.816326530612244


Training accuracy 100.0


Test Accuracy 79.13669064748201


Confusion Matrix [[200  32]
 [ 26  20]]



Test Accuracy 79.13669064748201


Confusion Matrix [[200  32]
 [ 26  20]]


Classification Report               precision    recall  f1-score   support

           0       0.88      0.86      0.87       232
           1       0.38      0.43      0.41        46

    accuracy                           0.79       278
   macro avg       0.63      0.65      0.64       278
weighted avg       0.80      0.79      0.80       278



Roc_auc Score 64.84257871064467
```

Out[85]: 46

KNN

In [86]:
```
knn = KNeighborsClassifier()
max_accuracy_score(knn,x,y)
```

```
max accuracy score corresponding to 44 is 85.97122302158273


cross validation score 84.93208321429499


Standard Deviation 0.7225415674798252


F1 score 22.222222222222218


Training accuracy 87.2858431018936


Test Accuracy 84.89208633093526


Confusion Matrix [[230   2]
 [ 40   6]]
```


```
Confusion Matrix [[230   2]
 [ 40   6]]
```


```
Classification Report               precision    recall  f1-score   support

           0       0.85      0.99      0.92       232
           1       0.75      0.13      0.22        46

    accuracy                           0.85       278
   macro avg       0.80      0.56      0.57       278
weighted avg       0.83      0.85      0.80       278



Roc_auc Score 56.09070464767616
```

Out[86]: 44

Random Forest

In [87]:
```
rf =RandomForestClassifier()
max_accuracy_score(rf,x,y)
```

```
max accuracy score corresponding to 66 is 87.76978417266187


cross validation score 85.36399761057581


Standard Deviation 0.642139252890573


F1 score 28.07017543859649


Training accuracy 100.0


Test Accuracy 85.25179856115108


Confusion Matrix [[229   3]
 [ 38   8]]
```

```
Confusion Matrix [[229   3]
 [ 38   8]]

Classification Report               precision    recall  f1-score   support

           0       0.86      0.99      0.92       232
           1       0.73      0.17      0.28        46

    accuracy                           0.85       278
   macro avg       0.79      0.58      0.60       278
weighted avg       0.84      0.85      0.81       278


Roc_auc Score 58.0491004497751
```

Out[87]: 66

AdaBoost

In [88]:
```
adb= AdaBoostClassifier()
max_accuracy_score(adb,x,y)
```

```
max accuracy score corresponding to 44 is 91.36690647482014


cross validation score 86.73402072565774


Standard Deviation 1.9947756669159884


F1 score 61.53846153846153


Training accuracy 89.54012623985572


Test Accuracy 89.20863309352518


Confusion Matrix [[224   8]
 [ 22  24]]


-        --


Classification Report               precision    recall  f1-score   support

           0       0.91      0.97      0.94       232
           1       0.75      0.52      0.62        46

    accuracy                           0.89       278
   macro avg       0.83      0.74      0.78       278
weighted avg       0.88      0.89      0.88       278



Roc_auc Score 74.36281859070466
```

Out[88]: 44

Gradient Boost

In [89]:
```
gb = GradientBoostingClassifier()
max_accuracy_score(gb,x,y)
```

```
max accuracy score corresponding to 60 is 89.568345323741


cross validation score 86.08524011116016


Standard Deviation 1.395928216614923


F1 score 46.37681159420289


Training accuracy 96.12263300270514


Test Accuracy 86.6906474820144


Confusion Matrix [[225   7]
 [ 30  16]]


Classification Report               precision    recall  f1-score   support

           0       0.88      0.97      0.92       232
           1       0.70      0.35      0.46        46

    accuracy                           0.87       278
   macro avg       0.79      0.66      0.69       278
weighted avg       0.85      0.87      0.85       278



Roc_auc Score 65.88268365817093
```

Out[89]: 60

As we can see that Logistic Regression gives the best accuracy as compaired to others nodel and Roc _auc Score is also good among all the models. so, Logistic Regression is thebestmodel for this data set

Hyperparameter Tuning

In [91]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=60,test_size=.20,stratify=y)
## Upsampling
lg=LogisticRegression()
param={'penalty':['l2','l1'],'C':[.0001,.001,.01,1,10],'solver':['liblinear','saga']}

grid=GridSearchCV(estimator=lg,param_grid=param,scoring='accuracy',n_jobs=-1)

grid.fit(x_train,y_train)

grid.best_params_
```

Out[91]:
```
{'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}
```

In [93]:
```
lg_final=LogisticRegression(C=10,penalty='l2',solver='liblinear')
lg_final.fit(x_train,y_train)
pred=lg_final.predict(x_test)
print('Final Accuracy_score :',accuracy_score(pred,y_test)*100)
print('\n')
print('Final f_1 score :',f1_score(pred,y_test)*100)
print('\n')
print('Final roc_auc score :',roc_auc_score(pred,y_test))
print('\n')
print('Final classification Report :',classification_report(pred,y_test))
print('\n')
print('Final confusion Matrix :',confusion_matrix(pred,y_test))
```



```
Final Accuracy_score : 89.92805755395683


Final f_1 score : 60.0


Final roc_auc score : 0.8882874015748031


Final classification Report :               precision    recall  f1-score   support

           0       0.99      0.90      0.94       254
           1       0.46      0.88      0.60        24

    accuracy                           0.90       278
   macro avg       0.72      0.89      0.77       278
weighted avg       0.94      0.90      0.91       278



Final confusion Matrix : [[229  25]
 [  3  21]]
```

saving model

In [94]:
```
import joblib
joblib.dump(lg_final,'Attrition_lg.pkl')
```

Out[94]:
```
['Attrition_lg.pkl']
```

loading and testing

In [96]:
```
loaded_model=joblib.load('Attrition_lg.pkl')
prediction=loaded_model.predict(x_test)
```

In [97]:
```
prediction
```

Out[97]:
```
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, 1, 1, 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, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 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, 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, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 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, 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, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```