IBM HR Analytics Employee Attrition & Performance

Uncover the factors that lead to employee attrition and explore important questions such as ‘show me a breakdown of distance from home by job role and attrition’ or ‘compare average monthly income by education and attrition’. This is a fictional data set created by IBM data scientists.

Employee attrition is the gradual reduction in employee numbers. Employee attrition happens when the size of your workforce diminishes over time. This means that employees are leaving faster than they are hired.

Employee attrition happens when employees retire, resign, or simply aren’t replaced.

Although employee attrition can be company-wide, it may also be confined to specific parts of a business. This is often the case when employees are replaced by automation or the adoption of new technologies.

Employee attrition can happen for several reasons. These include unhappiness about employee benefits or the pay structure, a lack of employee development opportunities, and even poor conditions in the workplace.

In a world where the skill sets required are constantly changing, some positions also become obsolete over time. As employees leave and a new future of work emerges, not every role is filled in the same cookie-cutter way.

With this, a new world of work means a new look at leadership.

In some cases, this is driven by a desire to modernize. In others, it’s due to a lack of skilled younger talent in certain industries and geographies.

Employee attrition also refers to the downsizing of an organization’s workforce. This means that attrition can be voluntary or involuntary.

Employee attrition can be problematic as it often reduces talent within the company and the workforce in general.

But employee attrition isn’t all bad.

It can be positive because it allows the company to identify and address problematic issues for its employees. For example, a high attrition rate could be from employees leaving due to a poor workplace culture. Only by investigating the reasons for this employee attrition can management make changes to improve the organization’s work culture for other employees.

While companies will usually try to avoid employee attrition, it can sometimes help cut down costs associated with labor. It can also attract new employees with fresh talent to organizations.

Employee attrition, also known as employee turnover, refers to the rate at which employees leave an organization and are replaced by new hires. High rates of employee attrition can be costly for companies, as they may have to spend resources on training and onboarding new employees, as well as potentially experiencing a decrease in productivity due to the loss of experienced workers. To address employee attrition, organizations may focus on retaining their current employees through strategies such as offering competitive benefits, providing professional development opportunities, and creating a positive work culture. They may also try to reduce the likelihood of employees leaving by improving the hiring process to ensure that new hires are a good fit for the company. Ultimately, managing employee attrition is an important aspect of maintaining a healthy and successful organization.

DATA ANALYSIS

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:

```
employee_data = pd.read_csv('Employee-Attrition.csv')
```

Exploratory Data Analysis

In [ ]:

```
employee_data.head()
```

Out[265]:

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

5 rows × 35 columns

In [ ]:

```
employee_data.shape
```

Out[266]:

```
(1470, 35)
```

Let's import the dataset and make of a copy of the source file for this analysis.

The dataset contains 1,470 rows and 35 columns.

In [ ]:

```
employee_data.columns
```

Out[267]:

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

The dataset contains several numerical and categorical columns providing various information on employee's personal and employment details.

In [ ]:

```
employee_data.isnull().sum()
```

Out[268]:

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [ ]:

```
employee_data.info()
```

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

In [ ]:

```
employee_data.describe()
```

Out[270]:

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

8 rows × 26 columns

count #How many records there are

average #Average value

std #Standard deviation value

min #minimum value in this series

25% #Average value of 25% of total records

50% #Average value of 50% of total records

75% #Average value of 75% of total records max #Maximum value in this series

In [ ]:

```
employee_data.select_dtypes(include=['object']).dtypes
```

Out[271]:

```
Attrition         object
BusinessTravel    object
Department        object
EducationField    object
Gender            object
JobRole           object
MaritalStatus     object
Over18            object
OverTime          object
dtype: object
```

In [ ]:

```
employee_data['Attrition'].value_counts()
```

Out[272]:

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

In [ ]:

```
employee_data['Attrition'] = employee_data['Attrition'].factorize(['No','Yes'])[0]
employee_data.head()
```

Out[273]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	..
0	41	1	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	..
1	49	0	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	..
2	37	1	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	..
3	33	0	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	..
4	27	0	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	..

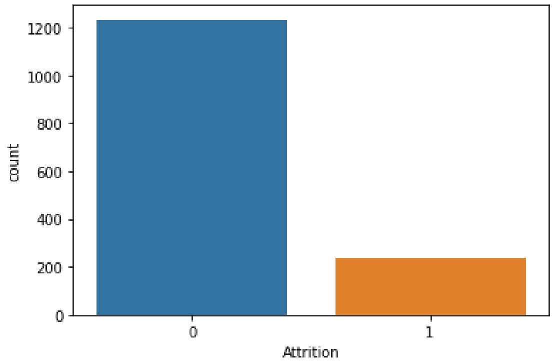
5 rows × 35 columns

In [ ]:

```
sns.countplot(x='Attrition',data=employee_data)
```

Out[274]:

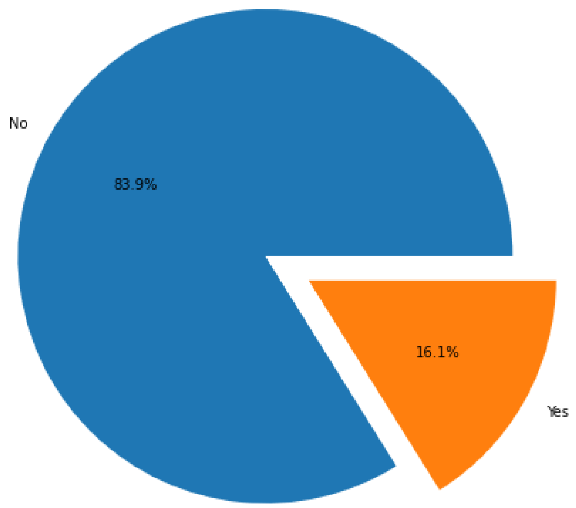
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3de050be0>
```



In our dataset, we showed the quitters and non-employeds in a column chart.

In [ ]:

```
plt.figure(figsize=(8,8))
pie = employee_data.groupby('Attrition')['Attrition'].count()
plt.pie(pie, explode=[0.1, 0.1], labels=['No', 'Yes'], autopct='%1.1f%%');
```



As shown on the chart above, we see this is an imbalanced class problem. Indeed, the percentage of Current Employees in our dataset is 83.9% and the percentage of Ex-employees is: 16.1%

In [ ]:

```
employee_data.select_dtypes(include=['int64']).dtypes
```

Out[276]:

```
Age                         int64
Attrition                   int64
DailyRate                   int64
DistanceFromHome            int64
Education                   int64
EmployeeCount               int64
EmployeeNumber              int64
EnvironmentSatisfaction     int64
HourlyRate                  int64
JobInvolvement              int64
JobLevel                    int64
JobSatisfaction             int64
MonthlyIncome               int64
MonthlyRate                 int64
NumCompaniesWorked          int64
PercentSalaryHike           int64
PerformanceRating           int64
RelationshipSatisfaction    int64
StandardHours               int64
StockOptionLevel            int64
TotalWorkingYears           int64
TrainingTimesLastYear       int64
WorkLifeBalance             int64
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

We changed our data types.

In [ ]:

```
#Data Corr
```

Let's take a look at some of most significant correlations. It is worth remembering that correlation coefficients only measure linear correlations.

What is correlation?

Correlation is a statistical term describing the degree to which two variables move in coordination with one another. If the two variables move in the same direction, then those variables are said to have a positive correlation. If they move in opposite directions, then they have a negative correlation.

In [ ]:

```
employee_data.corr()
```

Out[278]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfa
Age	1.000000	-0.159205	0.010661	-0.001686	0.208034	NaN	-0.010145	0.01
Attrition	-0.159205	1.000000	-0.056652	0.077924	-0.031373	NaN	-0.010577	-0.10
DailyRate	0.010661	-0.056652	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.01
DistanceFromHome	-0.001686	0.077924	-0.004985	1.000000	0.021042	NaN	0.032916	-0.01
Education	0.208034	-0.031373	-0.016806	0.021042	1.000000	NaN	0.042070	-0.02
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.010577	-0.050990	0.032916	0.042070	NaN	1.000000	0.01
EnvironmentSatisfaction	0.010146	-0.103369	0.018355	-0.016075	-0.027128	NaN	0.017621	1.00
HourlyRate	0.024287	-0.006846	0.023381	0.031131	0.016775	NaN	0.035179	-0.04
JobInvolvement	0.029820	-0.130016	0.046135	0.008783	0.042438	NaN	-0.006888	-0.00
JobLevel	0.509604	-0.169105	0.002966	0.005303	0.101589	NaN	-0.018519	0.00
JobSatisfaction	-0.004892	-0.103481	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.00
MonthlyIncome	0.497855	-0.159840	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.00
MonthlyRate	0.028051	0.015170	-0.032182	0.027473	-0.026084	NaN	0.012648	0.03
NumCompaniesWorked	0.299635	0.043494	0.038153	-0.029251	0.126317	NaN	-0.001251	0.01
PercentSalaryHike	0.003634	-0.013478	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.03
PerformanceRating	0.001904	0.002889	0.000473	0.027110	-0.024539	NaN	-0.020359	-0.02
RelationshipSatisfaction	0.053535	-0.045872	0.007846	0.006557	-0.009118	NaN	-0.069861	0.00
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
StockOptionLevel	0.037510	-0.137145	0.042143	0.044872	0.018422	NaN	0.062227	0.00
TotalWorkingYears	0.680381	-0.171063	0.014515	0.004628	0.148280	NaN	-0.014365	-0.00
TrainingTimesLastYear	-0.019621	-0.059478	0.002453	-0.036942	-0.025100	NaN	0.023603	-0.01
WorkLifeBalance	-0.021490	-0.063939	-0.037848	-0.026556	0.009819	NaN	0.010309	0.02
YearsAtCompany	0.311309	-0.134392	-0.034055	0.009508	0.069114	NaN	-0.011240	0.00
YearsInCurrentRole	0.212901	-0.160545	0.009932	0.018845	0.060236	NaN	-0.008416	0.01
YearsSinceLastPromotion	0.216513	-0.033019	-0.033229	0.010029	0.054254	NaN	-0.009019	0.01
YearsWithCurrManager	0.202089	-0.156199	-0.026363	0.014406	0.069065	NaN	-0.009197	-0.00

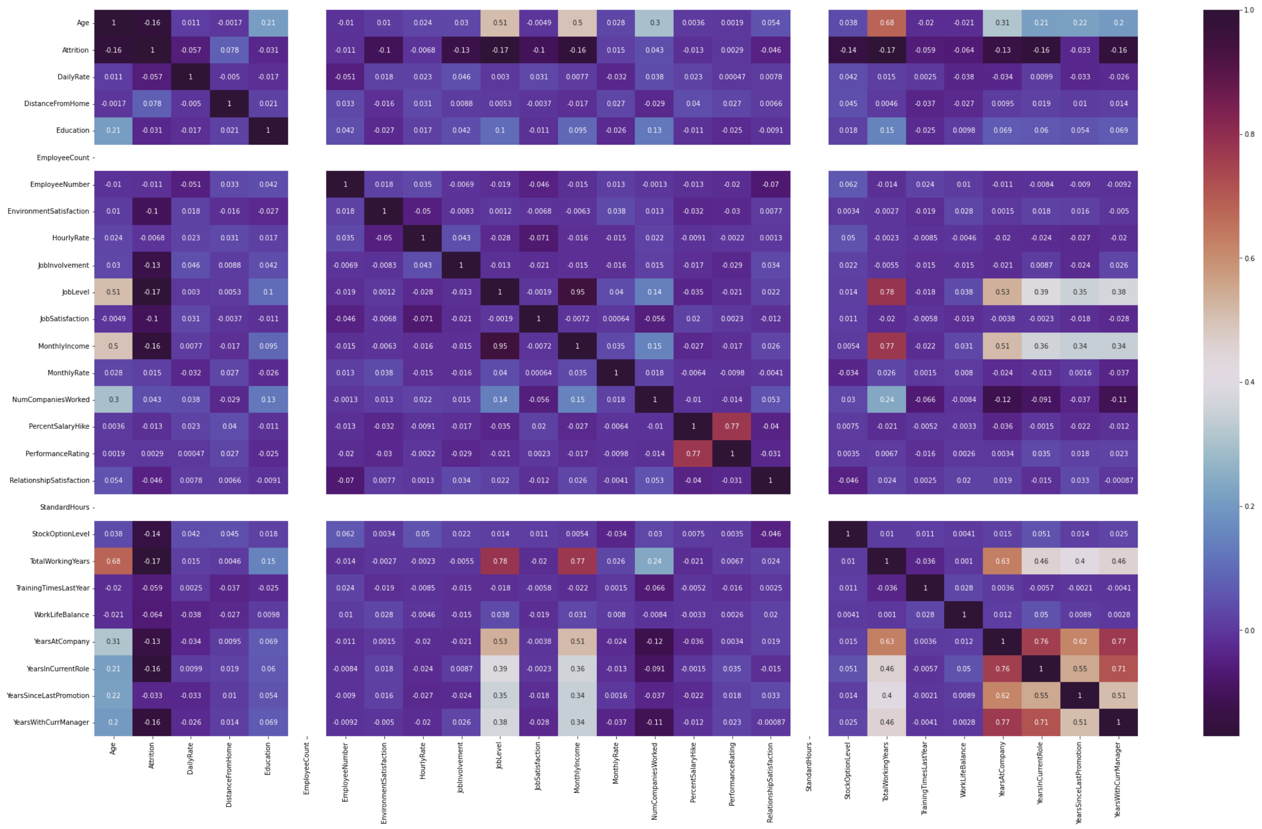
27 rows × 27 columns

In [ ]:

```
plt.figure(figsize=(35,20))
sns.heatmap(employee_data.corr(),annot=True,cmap='twilight_shifted')
```

Out[279]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3dddc92b0>
```



Correlation is a statistical measure that reflects the strength and direction of a relationship between two variables. It is used to determine whether there is a relationship between two variables and, if so, how strong that relationship is. Correlation can be positive, negative, or zero. A positive correlation means that as one variable increases, the other variable also increases. A negative correlation means that as one variable increases, the other variable decreases.

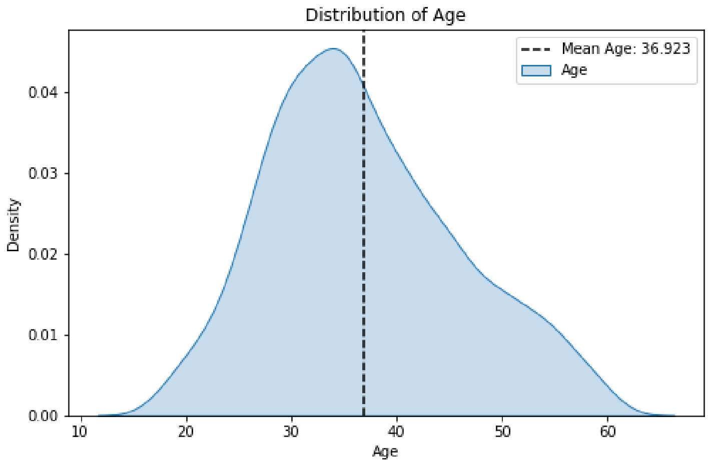
There are many reasons why we might want to make a correlation. For example, we might want to understand the relationship between two variables in order to make predictions about future outcomes, to identify patterns or trends, to identify the causes of certain phenomena, or to understand the relationships between different variables in a system. Additionally, correlations can be used to identify the strength and direction of relationships between variables, which can be useful for making decisions or for designing interventions.

As we can see, the target column does not have a very strong correlation with any numeric column. However;

More senior employees have higher total years of work (very obvious) Higher performance grades lead to higher pay rise The more years an employee has, the higher their monthly income Over the years, many employees remain in their current role and under the same manager, which means they are not promoted, and this can be a major contributing factor to attrition. From this we can conclude that lack of promotion can be a very important factor for attrition.

In [ ]:

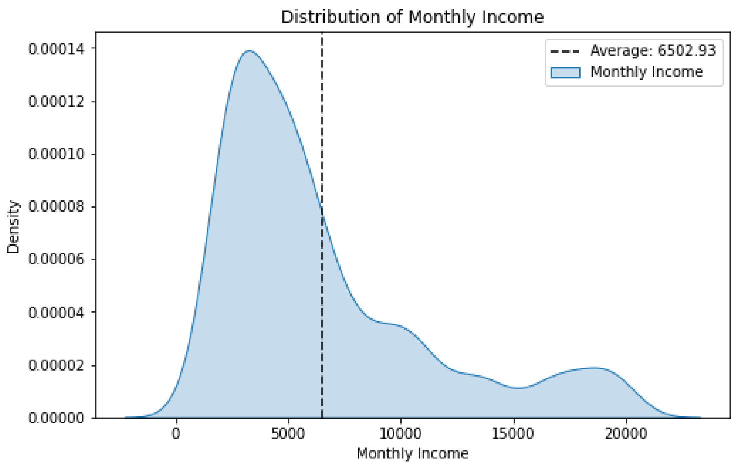
```
plt.figure(figsize=(8,5))
sns.kdeplot(x=employee_data['Age'],shade=True,label='Age')
plt.axvline(x=employee_data['Age'].mean(),color='k',linestyle ="--",label='Mean Age: 36.923')
plt.legend()
plt.title('Distribution of Age')
plt.show()
```



The mean and distribution of age in the data set is shown in the graph.

In [ ]:

```
plt.figure(figsize=(8,5))
sns.kdeplot(x=employee_data['MonthlyIncome'],shade=True,label='Monthly Income')
plt.axvline(x=employee_data['MonthlyIncome'].mean(),color='k',linestyle ="--",label='Average: 6502.93')
plt.xlabel('Monthly Income')
plt.legend()
plt.title('Distribution of Monthly Income')
plt.show()
```



The average and distribution of monthly income in the data set is shown in the graph.

In [ ]:

```
employee_data[['Age']].value_counts().sort_values(ascending=False).head(10)
```

Out[282]:

```
Age
35     78
34     77
36     69
31     69
29     68
32     61
30     60
38     58
33     58
40     57
dtype: int64
```

In [ ]:

```
employee_data[['Age']].value_counts().sort_values(ascending=False).tail()
```

Out[283]:

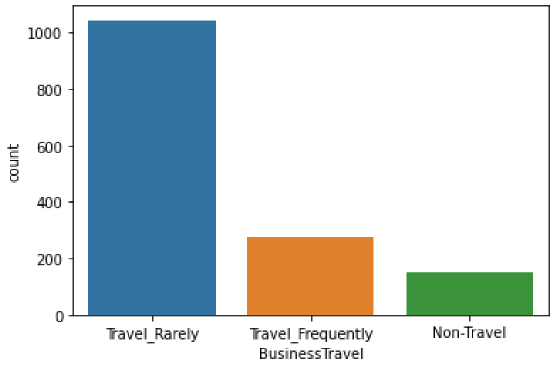
```
Age
59     10
19      9
18      8
60      5
57      4
dtype: int64
```

In [ ]:

```
sns.countplot(x='BusinessTravel',data=employee_data)
```

Out[284]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3df5a4280>
```

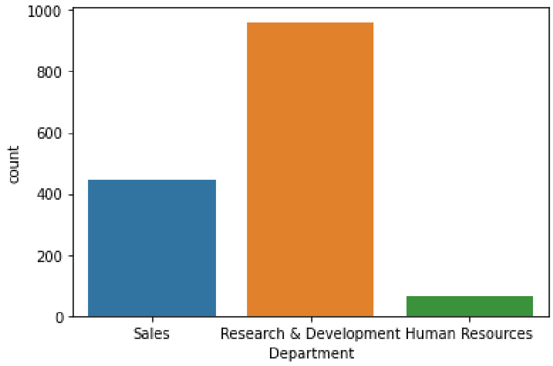


In [ ]:

```
sns.countplot(x='Department',data=employee_data)
```

Out[285]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3df59f280>
```

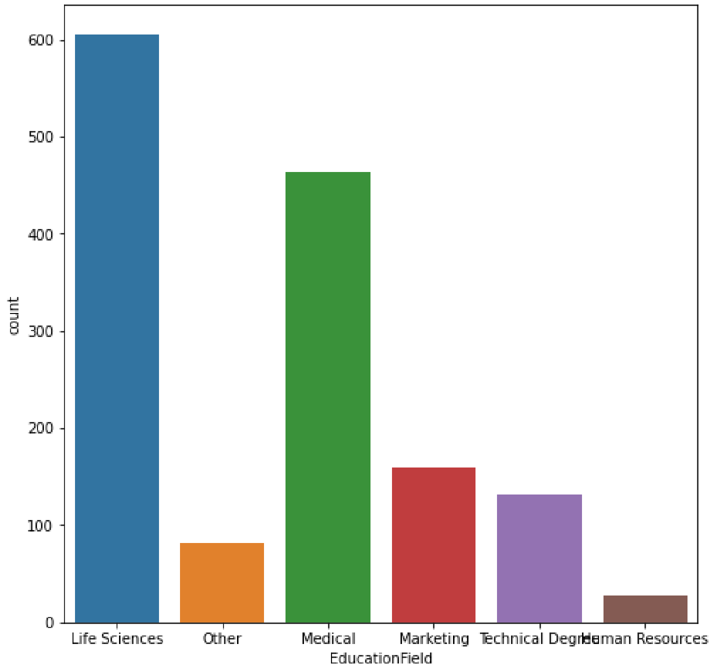


In [ ]:

```
plt.figure(figsize=(8,8))
sns.countplot(x='EducationField',data=employee_data)
```

Out[286]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3df8e4400>
```

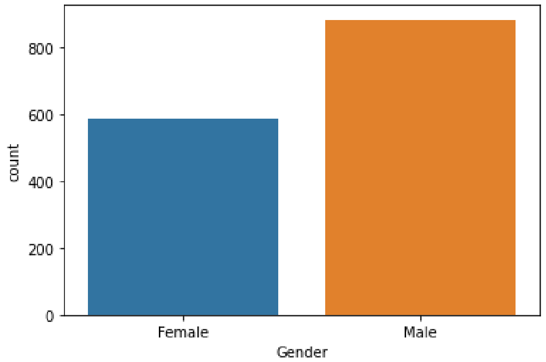


In [ ]:

```
sns.countplot(x='Gender',data=employee_data)
```

Out[287]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3de2e37f0>
```



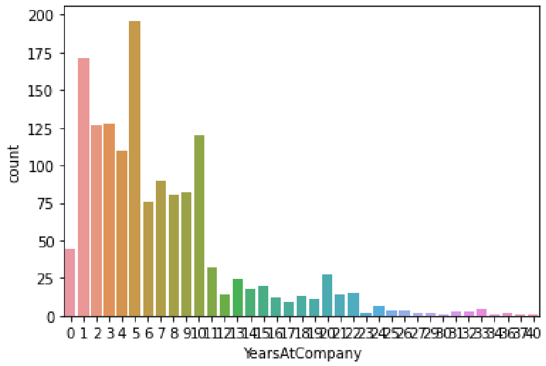
In [ ]:

```
sns.countplot(employee_data["YearsAtCompany"])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a k
eyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[288]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3deb2b670>
```



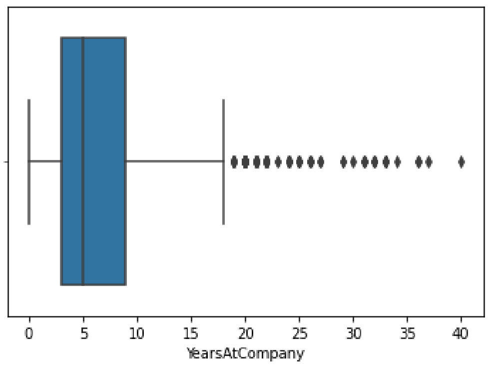
In [ ]:

```
sns.boxplot(employee_data["YearsAtCompany"])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a k
eyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[289]:

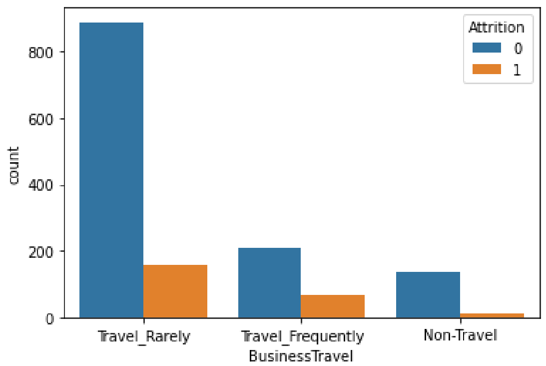
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3dec07af0>
```



Most employees stay with the company for 3-9 years, with a median of 5 years.

In [ ]:

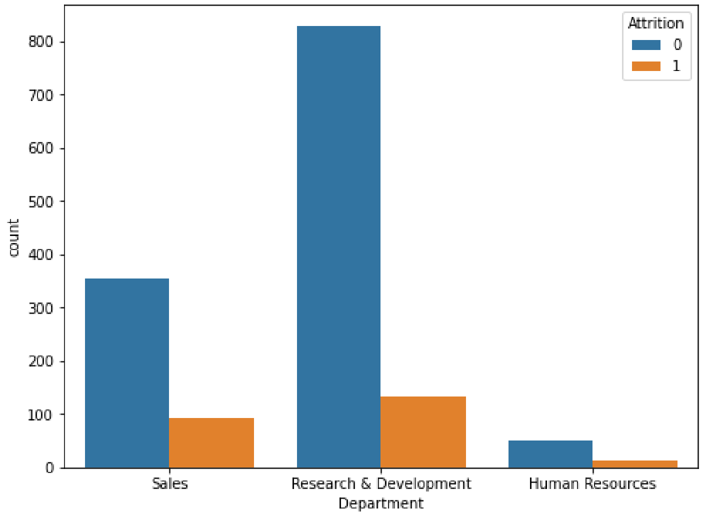
```
sns.countplot(x='BusinessTravel', hue='Attrition', data=employee_data);
```



Most employees who travel rarely leave the company. As far as we can tell, sending employees on business trips doesn't make much of a difference and doesn't have a significant impact on attrition.

In [ ]:

```
plt.figure(figsize=(8,6))
sns.countplot(x='Department', hue='Attrition', data=employee_data);
```



In [ ]:

```
employee_data['Department'].value_counts()
```

Out[292]:

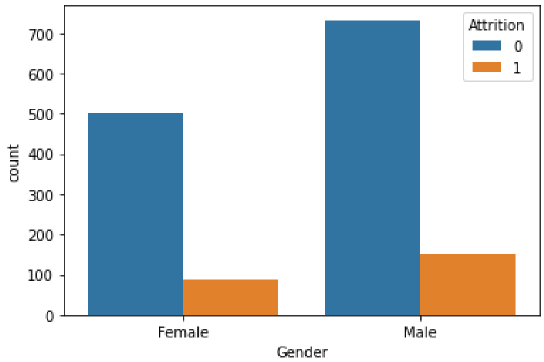
```
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
```

Most attrition comes from the research and development department, with the sales department in the second place by a small margin. HUMAN resources have the least attrition. However, we should not forget that R&D has many more employees than sales and HR.

If we take into account the percentage of attrition per department, we see that the HR department has the most attrition.

In [ ]:

```
sns.countplot(x='Gender', hue='Attrition', data=employee_data);
```



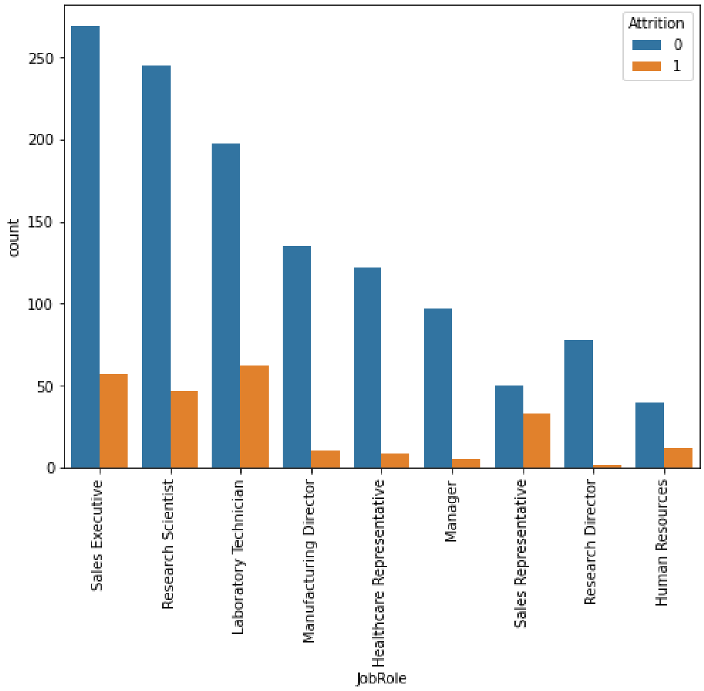
Clearly there are more men than women in the organization so the attrition is higher but only slightly. IW does not consider gender a major factor behind attrition.

In [ ]:

```
plt.figure(figsize=(8,6))
sns.countplot(x='JobRole', hue='Attrition', data=employee_data);
plt.xticks(rotation=90)
```

Out[294]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 <a list of 9 Text major ticklabel objects>)
```



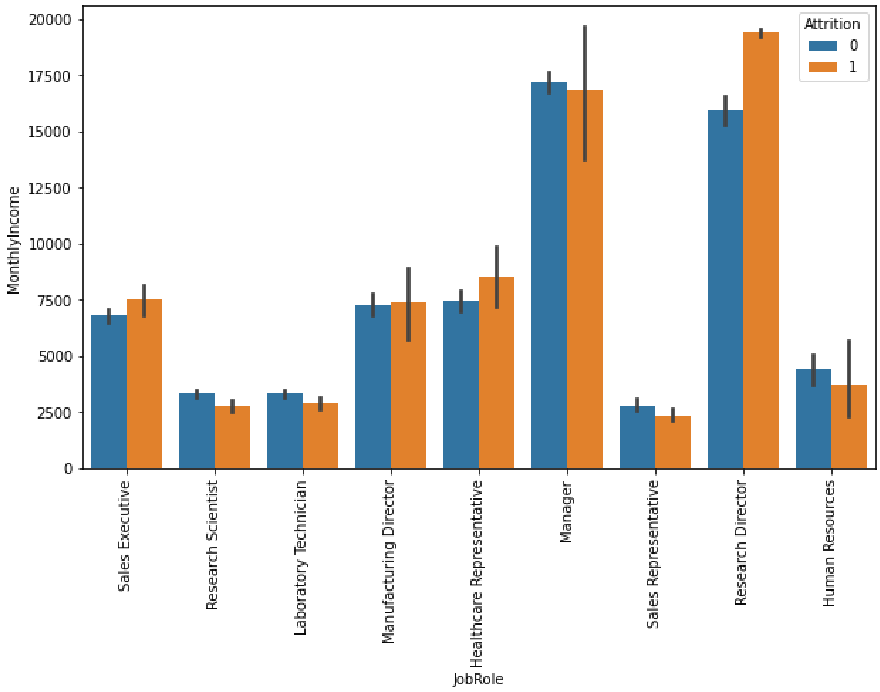
Among job roles, most lab technicians left their jobs, with only research scientists, sales managers, and sales reps (% wise) left behind. We can look at the salaries of each job role and see if that is the reason.

In [ ]:

```
plt.figure(figsize=(10,6))
sns.barplot(x='JobRole', y='MonthlyIncome', hue='Attrition', data=employee_data)
plt.xticks(rotation=90)
```

Out[295]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 <a list of 9 Text major ticklabel objects>)
```



As suspected, the salaries of laboratory technicians, research scientists and sales representatives and managers are very low, and this may be a major factor behind attrition.

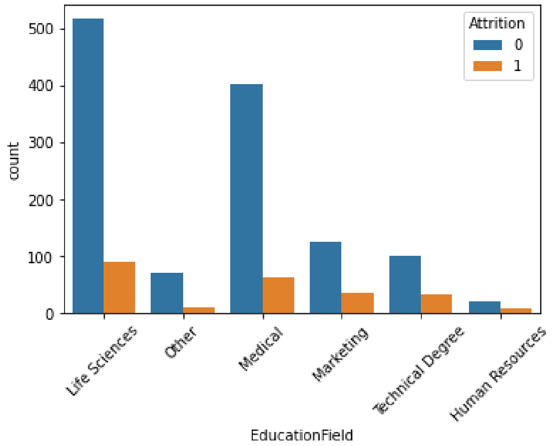
Also, as we've seen before, the HR department had the most attrition and we can see that their salaries are also very low, so it's something to think about once again.

In [ ]:

```
sns.countplot(x='EducationField', hue='Attrition', data=employee_data);
plt.xticks(rotation=45)
```

Out[296]:

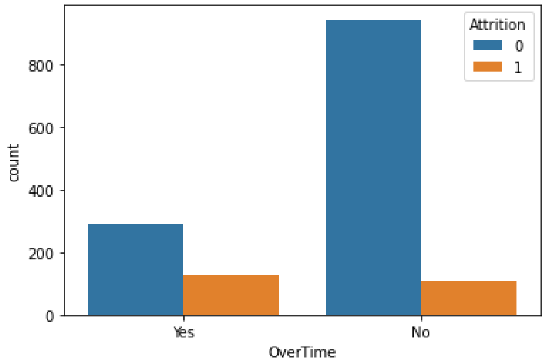
```
(array([0, 1, 2, 3, 4, 5]), <a list of 6 Text major ticklabel objects>)
```



Employee ratings really matter here, as most of the attrition numbers are similar.

In [ ]:

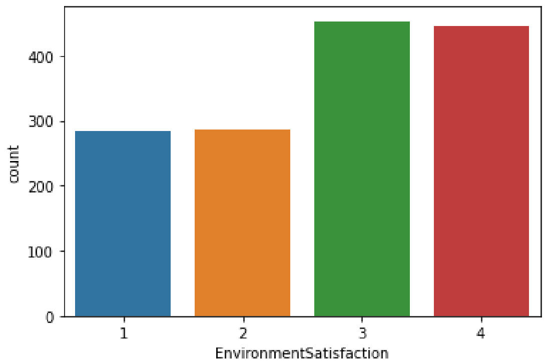
```
sns.countplot(x='OverTime', hue='Attrition', data=employee_data);
```



Overtime hours are also not a very important factor.

In [ ]:

```
sns.countplot(x='EnvironmentSatisfaction', data=employee_data);
```

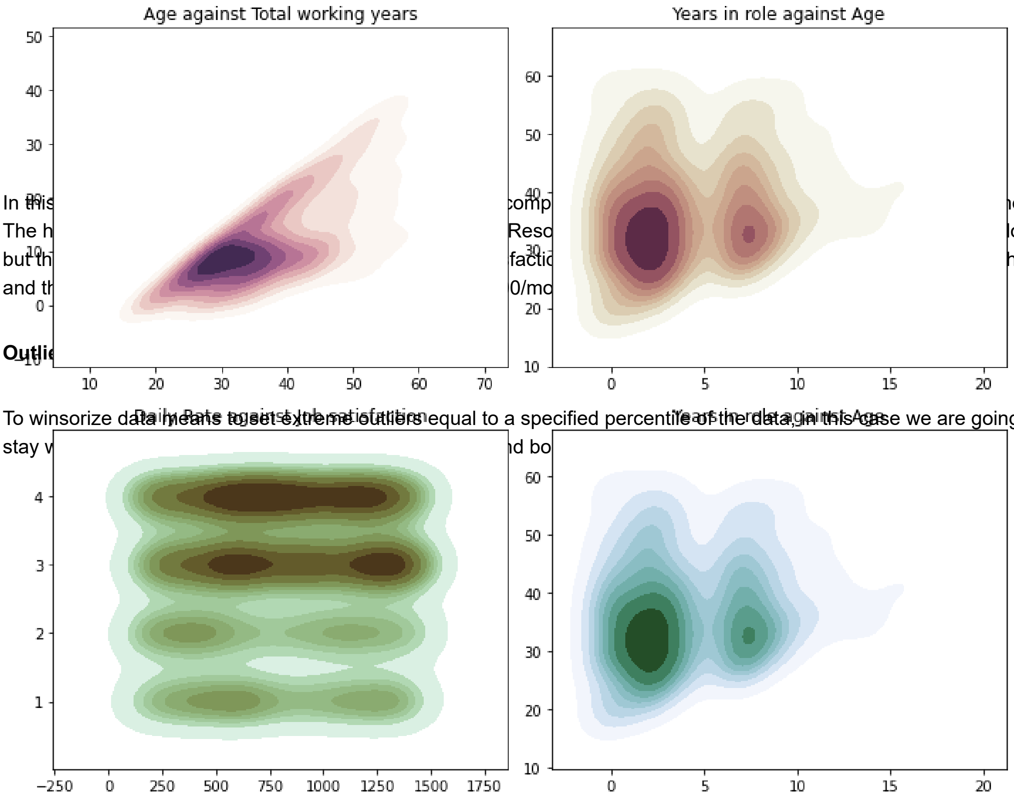


Most employees seem satisfied with the work environment.

In [ ]:

```
f, axes = plt.subplots(2, 2, figsize=(10, 8),
                       sharex=False, sharey=False)

# Defining our colormap scheme
s = np.linspace(0, 3, 10)
cmap = sns.cubehelix_palette(start=0.0, light=1, as_cmap=True)

# Generate and plot
x = employee_data['Age'].values
y = employee_data['TotalWorkingYears'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, cut=5, ax=axes[0,0])
axes[0,0].set( title = 'Age against Total working years')

cmap = sns.cubehelix_palette(start=0.333333333333, light=1, as_cmap=True)

# Generate and plot
x = employee_data['YearsInCurrentRole'].values
y = employee_data['Age'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[0,1])
axes[0,1].set( title = 'Years in role against Age')

cmap = sns.cubehelix_palette(start=1.0, light=1, as_cmap=True)

# Generate and plot
x = employee_data['DailyRate'].values
y = employee_data['JobSatisfaction'].values
sns.kdeplot(x, y, cmap=cmap, shade=True,  ax=axes[1,0])
axes[1,0].set( title = 'Daily Rate against Job satisfaction')

cmap = sns.cubehelix_palette(start=1.666666666667, light=1, as_cmap=True)
# Generate and plot
x = employee_data['YearsInCurrentRole'].values
y = employee_data['Age'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[1,1])
axes[1,1].set( title = 'Years in role against Age')

cmap = sns.cubehelix_palette(start=1.0, light=1, as_cmap=True)

f.tight_layout()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a k
eyword arg: y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a k
eyword arg: y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a k
eyword arg: y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a k
eyword arg: y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In this … comp… he Research and Development department. The h… Reso… lowest work-life balance left the company, but th… factio… he salary was found to be significantly lower and th… 0/mo…

Outlie…

To winsorize data means to set extreme outliers equal to a specified percentile of the data, in this case we are going to be conservative and we are going to stay w… nd bo…

In [ ]:

```
def Winsorization_outliers(df):
    q1 = np.percentile(df , 1)
    q3 = np.percentile(df , 99)
    out=[]
    for i in df:
        if (i > q3 or i < q1) and i>0:
            out.append(i)
    print("Outliers:",out)
    return out;
def remove_outliers(df):
    print("Registers in the initial dataset:",df.shape[0])
    for col in df.columns[1:]:
        if df[col].dtype != 'object':
            print(col)
            data_filter = Winsorization_outliers(df[col])
            df = df[~df[col].isin(data_filter)]
            print("Registers without outliers in "+col+" :"+ str(df.shape[0]))
    return df;


employee_data_cleaned = remove_outliers(employee_data)
```

```
Registers in the initial dataset: 1470
Attrition
Outliers: []
Registers without outliers in Attrition :1470
DailyRate
Outliers: [103, 1488, 111, 1496, 111, 106, 1490, 1490, 1499, 1495, 102, 109, 1492, 111, 116, 107, 1498, 1495, 1490,
1496, 115, 104, 1495, 1490, 116, 105]
Registers without outliers in DailyRate :1444
DistanceFromHome
Outliers: []
Registers without outliers in DistanceFromHome :1444
Education
Outliers: []
Registers without outliers in Education :1444
EmployeeCount
Outliers: []
Registers without outliers in EmployeeCount :1444
EmployeeNumber
Outliers: [1, 2, 4, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16, 18, 20, 2048, 2049, 2051, 2052, 2053, 2054, 2055, 2056, 20
57, 2060, 2061, 2062, 2064, 2065, 2068]
Registers without outliers in EmployeeNumber :1414
EnvironmentSatisfaction
Outliers: []
Registers without outliers in EnvironmentSatisfaction :1414
HourlyRate
Outliers: []
Registers without outliers in HourlyRate :1414
JobInvolvement
Outliers: []
Registers without outliers in JobInvolvement :1414
JobLevel
Outliers: []
Registers without outliers in JobLevel :1414
JobSatisfaction
Outliers: []
Registers without outliers in JobSatisfaction :1414
MonthlyIncome
Outliers: [1232, 19926, 1102, 19999, 1200, 1009, 1281, 19859, 1051, 19973, 19845, 1052, 19627, 19943, 19740, 1223,
1118, 19847, 19717, 19701, 1359, 1261, 1274, 19658, 19833, 19665, 1081, 1091, 19636, 1129]
Registers without outliers in MonthlyIncome :1384
MonthlyRate
Outliers: [2094, 26959, 26897, 26820, 2302, 2137, 26767, 26707, 26914, 2227, 2288, 2112, 2125, 26894, 2104, 2243, 2
6968, 2253, 26933, 2261, 2097, 26997, 26841, 2125, 2122, 26862, 26849, 26956]
Registers without outliers in MonthlyRate :1356
NumCompaniesWorked
Outliers: []
Registers without outliers in NumCompaniesWorked :1356
PercentSalaryHike
Outliers: []
Registers without outliers in PercentSalaryHike :1356
PerformanceRating
Outliers: []
Registers without outliers in PerformanceRating :1356
RelationshipSatisfaction
Outliers: []
Registers without outliers in RelationshipSatisfaction :1356
StandardHours
Outliers: []
Registers without outliers in StandardHours :1356
StockOptionLevel
Outliers: []
Registers without outliers in StockOptionLevel :1356
TotalWorkingYears
Outliers: [37, 38, 40, 36, 37, 36, 37, 40, 35, 36, 35, 36, 36, 37]
Registers without outliers in TotalWorkingYears :1342
TrainingTimesLastYear
Outliers: []
Registers without outliers in TrainingTimesLastYear :1342
WorkLifeBalance
Outliers: []
Registers without outliers in WorkLifeBalance :1342
YearsAtCompany
Outliers: [27, 33, 29, 27, 32, 34, 31, 33, 33, 32, 33, 30]
Registers without outliers in YearsAtCompany :1330
YearsInCurrentRole
Outliers: [16, 18, 17, 16, 16, 16, 16, 17, 17, 17, 16]
Registers without outliers in YearsInCurrentRole :1319
YearsSinceLastPromotion
Outliers: [15, 14, 15, 15, 15, 14, 15, 15, 14, 14, 14, 14, 14]
Registers without outliers in YearsSinceLastPromotion :1306
YearsWithCurrManager
Outliers: [15, 17, 15, 17, 14, 17, 14, 16, 14]
Registers without outliers in YearsWithCurrManager :1297
```

Drop columns

In [ ]:

```
employee_data.drop('EmployeeCount',axis=1,inplace=True)
employee_data.drop('StandardHours',axis=1,inplace=True)
```

Show input X and output y

In [ ]:

```
X=employee_data.drop('Attrition',axis=1)
y=employee_data.iloc[:,1]
key=X.keys()
```

Show X

In [ ]:

```
X
```

Out[303]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfaction
0	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	2
1	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	2	3
2	37	Travel_Rarely	1373	Research & Development	2	2	Other	4	4
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	5	4
4	27	Travel_Rarely	591	Research & Development	2	1	Medical	7	1
...	...	...	...	...	...	...	...	...	...
1465	36	Travel_Frequently	884	Research & Development	23	2	Medical	2061	3
1466	39	Travel_Rarely	613	Research & Development	6	1	Medical	2062	4
1467	27	Travel_Rarely	155	Research & Development	4	3	Life Sciences	2064	2
1468	49	Travel_Frequently	1023	Sales	2	3	Medical	2065	4
1469	34	Travel_Rarely	628	Research & Development	8	3	Medical	2068	2

1470 rows × 32 columns

Show y

In [ ]:

```
y
```

Out[304]:

```
0       1
1       0
2       1
3       0
4       0
       ..
1465    0
1466    0
1467    0
1468    0
1469    0
Name: Attrition, Length: 1470, dtype: int64
```

Transform y

In [ ]:

```
label=LabelEncoder()
y=label.fit_transform(y)
pd.DataFrame(y,columns=['Attrition'])
```

Out[305]:

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

1470 rows × 1 columns

Transform X

In [ ]:

```
object=['BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','Over18','OverTime']
for col in object:
    X[col]=label.fit_transform(X[col])
pd.DataFrame(X,columns=key)
```

Out[306]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfaction	G
0	41	2	1102	2	1	2	1	1	2	
1	49	1	279	1	8	1	1	2	3	
2	37	2	1373	1	2	2	4	4	4	
3	33	1	1392	1	3	4	1	5	4	
4	27	2	591	1	2	1	3	7	1	
...	...	...	...	...	...	...	...	...	...	
1465	36	1	884	1	23	2	3	2061	3	
1466	39	2	613	1	6	1	3	2062	4	
1467	27	2	155	1	4	3	1	2064	2	
1468	49	1	1023	2	2	3	3	2065	4	
1469	34	2	628	1	8	3	3	2068	2	

1470 rows × 32 columns

Feature Engineering

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
```

In [ ]:

```
label=LabelEncoder()
y=label.fit_transform(y)
pd.DataFrame(y,columns=['Attrition'])
```

Out[308]:

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

1470 rows × 1 columns

In [ ]:

```
object=['BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','Over18','OverTime']
for col in object:
    X[col]=label.fit_transform(X[col])
pd.DataFrame(X,columns=key)
```

Out[309]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfaction	G
0	41	2	1102	2	1	2	1	1	2	
1	49	1	279	1	8	1	1	2	3	
2	37	2	1373	1	2	2	4	4	4	
3	33	1	1392	1	3	4	1	5	4	
4	27	2	591	1	2	1	3	7	1	
...	...	...	...	...	...	...	...	...	...	
1465	36	1	884	1	23	2	3	2061	3	
1466	39	2	613	1	6	1	3	2062	4	
1467	27	2	155	1	4	3	1	2064	2	
1468	49	1	1023	2	2	3	3	2065	4	
1469	34	2	628	1	8	3	3	2068	2	

1470 rows × 32 columns

What is Label Encoding?

Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

What is a Dummy Variable?

A dummy variable is a numeric variable that encodes categorical information.

Dummy variables have two possible values: 0 or 1.

MinMaxScaler for Data

In [ ]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [ ]:

```
scaler = MinMaxScaler(copy=True, feature_range=(0, 1))
X = scaler.fit_transform(X)
pd.DataFrame(X,columns=key)
```

Out[311]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	EnvironmentSatisfactio
0	0.547619	1.0	0.715820	1.0	0.000000	0.25	0.2	0.000000	0.33333
1	0.738095	0.5	0.126700	0.5	0.250000	0.00	0.2	0.000484	0.66666
2	0.452381	1.0	0.909807	0.5	0.035714	0.25	0.8	0.001451	1.00000
3	0.357143	0.5	0.923407	0.5	0.071429	0.75	0.2	0.001935	1.00000
4	0.214286	1.0	0.350036	0.5	0.035714	0.00	0.6	0.002903	0.00000
...	...	...	...	...	...	...	...	...	
1465	0.428571	0.5	0.559771	0.5	0.785714	0.25	0.6	0.996613	0.66666
1466	0.500000	1.0	0.365784	0.5	0.178571	0.00	0.6	0.997097	1.00000
1467	0.214286	1.0	0.037938	0.5	0.107143	0.50	0.2	0.998065	0.33333
1468	0.738095	0.5	0.659270	1.0	0.035714	0.50	0.6	0.998549	1.00000
1469	0.380952	1.0	0.376521	0.5	0.250000	0.50	0.6	1.000000	0.33333

1470 rows × 32 columns

In [ ]:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix,accuracy_score,f1_score,precision_score,classification_report
from sklearn.metrics import recall_score
from sklearn.neighbors import KNeighborsClassifier
```

Split Data

In [ ]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.25,shuffle=True,random_state=33)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(1102, 32)
(1102,)
(368, 32)
(368,)
```

Applying RandomForestClassifier Model

In [ ]:

```
RandomForestClassifierModel = RandomForestClassifier(criterion = 'gini',n_estimators=100,max_depth=20,random_state=33) #criterion
RandomForestClassifierModel.fit(X[:1350], y[:1350])
```

Out[314]:

```
RandomForestClassifier(max_depth=20, random_state=33)
```

Calculating Details

In [ ]:

```
print('RandomForestClassifierModel Train Score is : ' , RandomForestClassifierModel.score(X_train, y_train))
print('RandomForestClassifierModel Test Score is : ' , RandomForestClassifierModel.score(X_test, y_test))
print('RandomForestClassifierModel features importances are : ' , RandomForestClassifierModel.feature_importances_)
```

```
RandomForestClassifierModel Train Score is :  0.9927404718693285
RandomForestClassifierModel Test Score is :  0.9945652173913043
RandomForestClassifierModel features importances are :  [0.06193159 0.00980547 0.05453782 0.01319056 0.04297764 0.0
1823934
 0.02274604 0.04949241 0.03033041 0.00863908 0.04433097 0.02482741
 0.02240029 0.03016047 0.02836658 0.01983262 0.07709716 0.04634821
 0.03036206 0.         0.05693691 0.03501782 0.00397256 0.02209253
 0.02987303 0.0487174  0.02736877 0.02547426 0.04067343 0.02360385
 0.02361724 0.02703607]
```

Applying KNeighborsClassifier Model

In [ ]:

```
KNNClassifierModel = KNeighborsClassifier(n_neighbors= 5,weights ='uniform', # it can be distance
                                          algorithm='auto') # it can be ball_tree, kd_tree,brute
KNNClassifierModel.fit(X[:1350], y[:1350])
```

Out[316]:

```
KNeighborsClassifier()
```

Calculating Details

In [ ]:

```
print('KNNClassifierModel Train Score is : ' , KNNClassifierModel.score(X_train, y_train))
print('KNNClassifierModel Test Score is : ' , KNNClassifierModel.score(X_test, y_test))
```

```
KNNClassifierModel Train Score is :  0.8774954627949183
KNNClassifierModel Test Score is :  0.8695652173913043
```

Calculating Prediction

In [ ]:

```
y_pred = RandomForestClassifierModel.predict(X_test)
y_pred_prob = RandomForestClassifierModel.predict_proba(X_test)
print('Predicted Value for RandomForestClassifierModel is : ' , y_pred[:10])
print('Prediction Probabilities Value for RandomForestClassifierModel is : ' , y_pred_prob[:10])
```

```
Predicted Value for RandomForestClassifierModel is :  [0 1 0 0 0 0 0 0 0 0]
Prediction Probabilities Value for RandomForestClassifierModel is :  [[0.99 0.01]
 [0.16 0.84]
 [0.94 0.06]
 [0.95 0.05]
 [0.97 0.03]
 [0.8  0.2 ]
 [0.99 0.01]
 [0.9  0.1 ]
 [0.92 0.08]
 [0.96 0.04]]
```

Calculating Confusion Matrix

In [ ]:

```
CM = confusion_matrix(y_test, y_pred)
CM
```

Out[319]:

```
array([[305,   1],
       [  1,  61]])
```

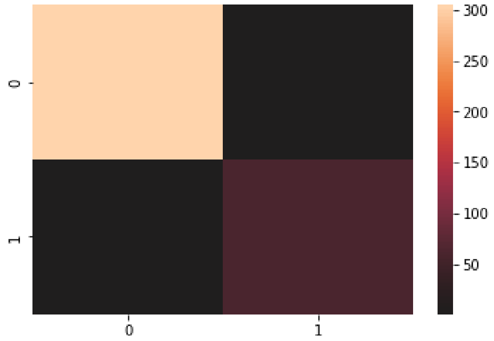
Drawing Confusion Matrix

In [ ]:

```
sns.heatmap(CM, center = True)
```

Out[320]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3dd3978b0>
```



Calculating Confusion Matrix

In [ ]:

```
CM = confusion_matrix(y_train,RandomForestClassifierModel.predict(X_train))
CM
```

Out[321]:

```
array([[927,   0],
       [  8, 167]])
```

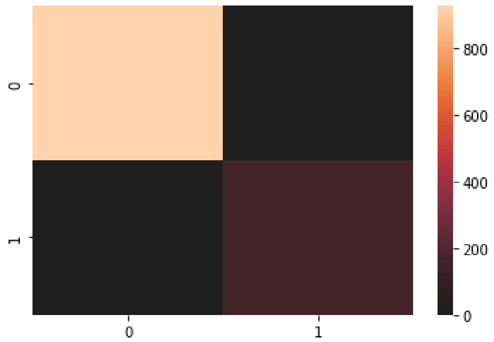
Drawing Confusion Matrix

In [ ]:

```
sns.heatmap(CM, center = True)
```

Out[322]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3dd3328b0>
```



Calculating Accuracy Score : ((TP + TN) / float(TP + TN + FP + FN))

In [ ]:

```
AccScore = accuracy_score(y_test, y_pred, normalize=False)
print('Accuracy Score is : ', AccScore)
```

```
Accuracy Score is :  366
```

Calculating F1 Score : 2 (precision recall) / (precision + recall)

In [ ]:

```
F1Score = f1_score(y_test, y_pred, average='micro') #it can be : binary,macro,weighted,samples
print('F1 Score is : ', F1Score)
```

```
F1 Score is :  0.9945652173913043
```

Calculating Recall Score : (Sensitivity) (TP / float(TP + FN)) 1 / 1+2

In [ ]:

```
RecallScore = recall_score(y_test, y_pred, average='micro') #it can be : binary,macro,weighted,samples
print('Recall Score is : ', RecallScore)
```

```
Recall Score is :  0.9945652173913043
```

Calculating Precision Score : (Specificity) #(TP / float(TP + FP))

In [ ]:

```
PrecisionScore = precision_score(y_test, y_pred, average='micro') #it can be : binary,macro,weighted,samples
print('Precision Score is : ', PrecisionScore)
```

```
Precision Score is :  0.9945652173913043
```

Calculating classification Report

In [ ]:

```
ClassificationReport = classification_report(y_test,y_pred)
print('Classification Report is : ', ClassificationReport )
```

```
Classification Report is :                precision    recall  f1-score   support

           0       1.00      1.00      1.00       306
           1       0.98      0.98      0.98        62

    accuracy                           0.99       368
   macro avg       0.99      0.99      0.99       368
weighted avg       0.99      0.99      0.99       368
```

In [ ]:

```
sub=[]
for i in y_pred:
    if i==0:
        sub.append('No')
    else:
        sub.append('yes')
submission=pd.DataFrame(sub,columns=['Attrition'])

submission
```

Out[328]:

	Attrition
0	No
1	yes
2	No
3	No
4	No
...	...
363	No
364	No
365	No
366	No
367	No

368 rows × 1 columns

REFERENCES

https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset (https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset)

https://www.betterup.com/blog/employee-attrition#:~:text=Employee%20attrition%20is%20the%20gradual,or%20simply%20aren't%20replaced (https://www.betterup.com/blog/employee-attrition#:~:text=Employee%20attrition%20is%20the%20gradual,or%20simply%20aren't%20replaced).

https://www.investopedia.com/terms/c/correlation.asp#:~:text=Correlation%20is%20a%20statistical%20term,they%20have%20a%20negative%20correlation (https://www.investopedia.com/terms/c/correlation.asp#:~:text=Correlation%20is%20a%20statistical%20term,they%20have%20a%20negative%20correlation)

https://www.kaggle.com/code/robertobonilla/explained-100-advanced-classification-beginners (https://www.kaggle.com/code/robertobonilla/explained-100-advanced-classification-beginners) https://chat.openai.com/chat (https://chat.openai.com/chat)

https://scikit-learn.org/stable/index.html (https://scikit-learn.org/stable/index.html)