Importing required libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn  as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics  import accuracy_score
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

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?

Reading datasets

In [2]:
```
#Loding Dataset
df=pd.read_csv("HR-Employee-Attrition.csv")
df
```

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Lif
1	49	No	Travel_Frequently	279	Research & Development	8	1	Lif
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Lif
4	27	No	Travel_Rarely	591	Research & Development	2	1	
...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Lif
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

1470 rows × 35 columns

Display all the rows and columns

In [3]:
```
pd.set_option('display.max_row',1500)
pd.set_option('display.max_columns',1500)
df.head()
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educatio
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sc
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sc
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sc
4	27	No	Travel_Rarely	591	Research & Development	2	1	M

Checking total no.rows and no.of columns

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

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

Data Preprocessing and EDA

Checking datatyoes of all the columns

In [5]:
```
df.info()
```

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

In [6]:
```
df.columns
```

Out[6]:
```
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')
```

Checking if there are any null values

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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
```

Hence. we can see there are no null values in the dataset as value is 0

In [8]:
```
df.nunique()
```

Out[8]:
```
Age                          43
Attrition                     2
BusinessTravel                3
DailyRate                   886
Department                    3
DistanceFromHome             29
Education                     5
EducationField                6
EmployeeCount                 1
EmployeeNumber             1470
EnvironmentSatisfaction       4
Gender                        2
HourlyRate                   71
JobInvolvement                4
JobLevel                      5
JobRole                       9
JobSatisfaction               4
MaritalStatus                 3
MonthlyIncome              1349
MonthlyRate                1427
NumCompaniesWorked           10
Over18                        1
OverTime                      2
PercentSalaryHike            15
PerformanceRating             2
RelationshipSatisfaction      4
StandardHours                 1
StockOptionLevel              4
TotalWorkingYears            40
TrainingTimesLastYear         7
WorkLifeBalance               4
YearsAtCompany               37
YearsInCurrentRole           19
YearsSinceLastPromotion      16
YearsWithCurrManager         18
dtype: int64
```

Summary of the whole Datasets

In [9]:
```
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	m
Age	1470.0	36.923810	9.135373	18.0	30.00	36.0	43.00	6(
DailyRate	1470.0	802.485714	403.509100	102.0	465.00	802.0	1157.00	149
DistanceFromHome	1470.0	9.192517	8.106864	1.0	2.00	7.0	14.00	2
Education	1470.0	2.912925	1.024165	1.0	2.00	3.0	4.00	5
EmployeeCount	1470.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1
EmployeeNumber	1470.0	1024.865306	602.024335	1.0	491.25	1020.5	1555.75	206
EnvironmentSatisfaction	1470.0	2.721769	1.093082	1.0	2.00	3.0	4.00	4
HourlyRate	1470.0	65.891156	20.329428	30.0	48.00	66.0	83.75	10(
JobInvolvement	1470.0	2.729932	0.711561	1.0	2.00	3.0	3.00	4
JobLevel	1470.0	2.063946	1.106940	1.0	1.00	2.0	3.00	5
JobSatisfaction	1470.0	2.728571	1.102846	1.0	2.00	3.0	4.00	4
MonthlyIncome	1470.0	6502.931293	4707.956783	1009.0	2911.00	4919.0	8379.00	1999
MonthlyRate	1470.0	14313.103401	7117.786044	2094.0	8047.00	14235.5	20461.50	2699
NumCompaniesWorked	1470.0	2.693197	2.498009	0.0	1.00	2.0	4.00	9
PercentSalaryHike	1470.0	15.209524	3.659938	11.0	12.00	14.0	18.00	25
PerformanceRating	1470.0	3.153741	0.360824	3.0	3.00	3.0	3.00	4
RelationshipSatisfaction	1470.0	2.712245	1.081209	1.0	2.00	3.0	4.00	4
StandardHours	1470.0	80.000000	0.000000	80.0	80.00	80.0	80.00	8(
StockOptionLevel	1470.0	0.793878	0.852077	0.0	0.00	1.0	1.00	3
TotalWorkingYears	1470.0	11.279592	7.780782	0.0	6.00	10.0	15.00	4(
TrainingTimesLastYear	1470.0	2.799320	1.289271	0.0	2.00	3.0	3.00	6
WorkLifeBalance	1470.0	2.761224	0.706476	1.0	2.00	3.0	3.00	4
YearsAtCompany	1470.0	7.008163	6.126525	0.0	3.00	5.0	9.00	4(
YearsInCurrentRole	1470.0	4.229252	3.623137	0.0	2.00	3.0	7.00	18
YearsSinceLastPromotion	1470.0	2.187755	3.222430	0.0	0.00	1.0	3.00	15
YearsWithCurrManager	1470.0	4.123129	3.568136	0.0	2.00	3.0	7.00	17

Dividing the columns into 2 categories(Continuous and categorical)

In [10]:

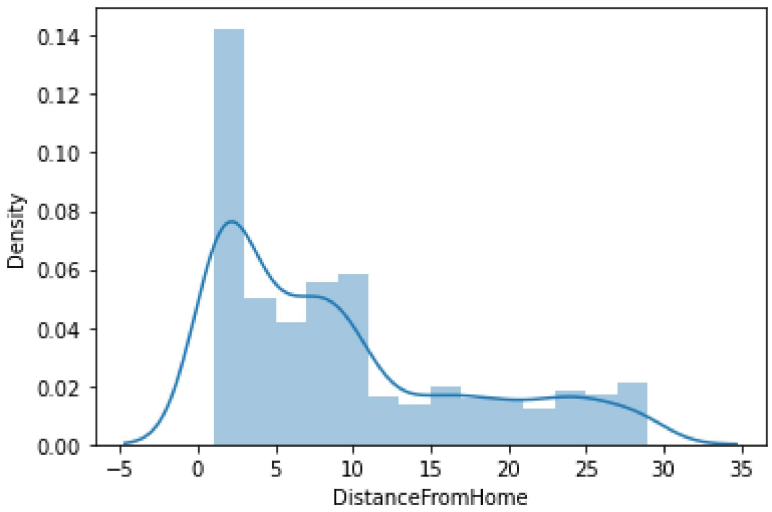
```
cat=[]
con=[]
for i in df.columns:
    if df[i].dtypes=='object':
        cat.append(i)
```

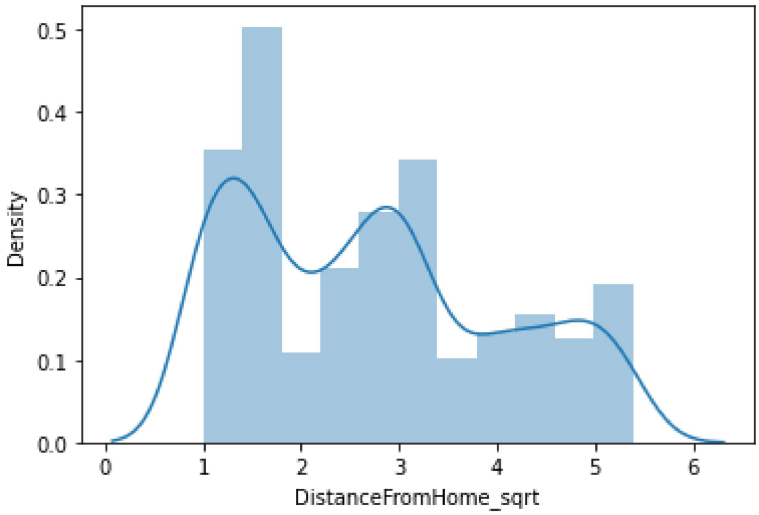
```
    else:
        con.append(i)
```

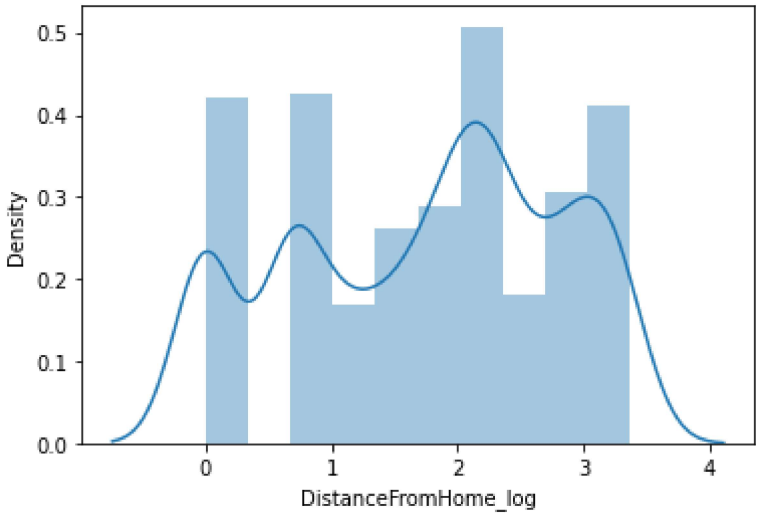
In [11]:
```
cat_df=df[['Attrition','BusinessTravel','Department','EducationField','Gender','JobRol
           'MaritalStatus', 'Over18','OverTime']]
```

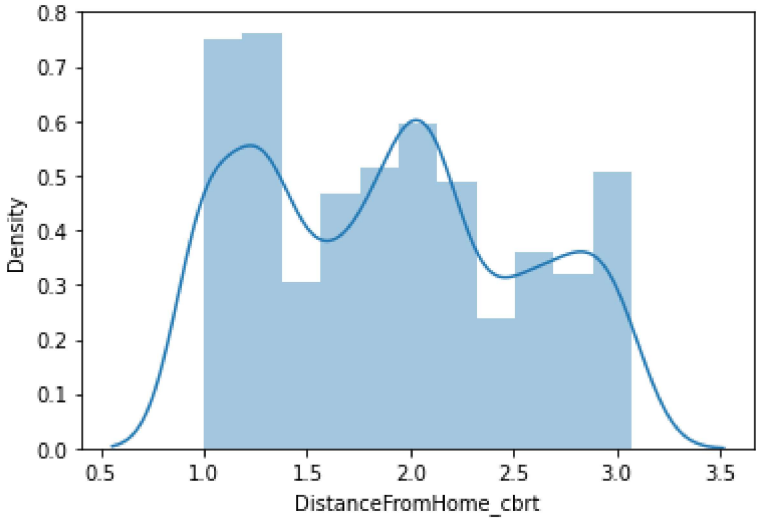
In [12]:
```
con_df=df[['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager']]

# 3 column are removed as the have constant values and of no use in the dataset furthe
#'EmployeeNumber','StandardHours'
```

In [13]:
```
#distanceFromHome

sb.distplot(df.DistanceFromHome)
plt.show()

df['DistanceFromHome_sqrt']=np.sqrt(df.DistanceFromHome)
sb.distplot(df.DistanceFromHome_sqrt)
plt.show()

df['DistanceFromHome_log']=np.log(df.DistanceFromHome)
sb.distplot(df.DistanceFromHome_log)
plt.show()

df['DistanceFromHome_cbrt']=np.cbrt(df.DistanceFromHome)
sb.distplot(df.DistanceFromHome_cbrt)
plt.show()
```



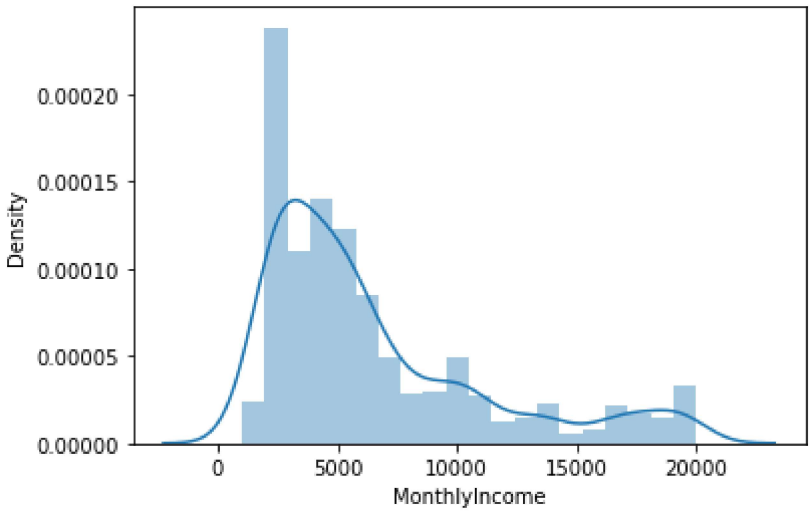


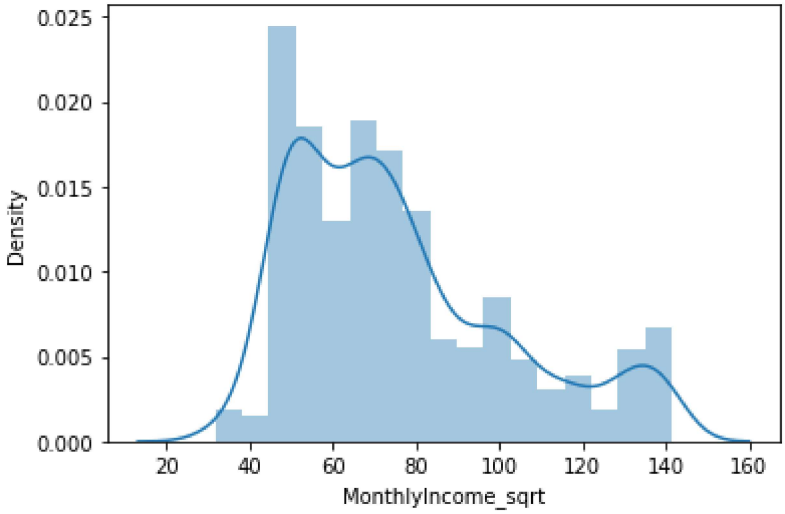


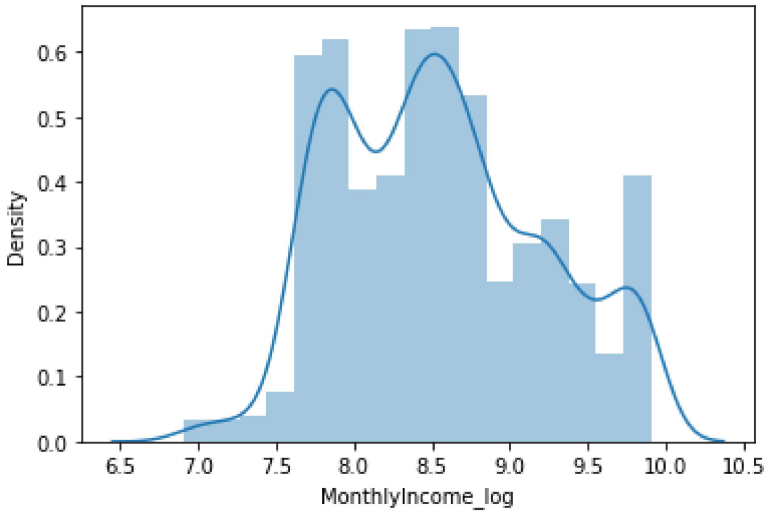


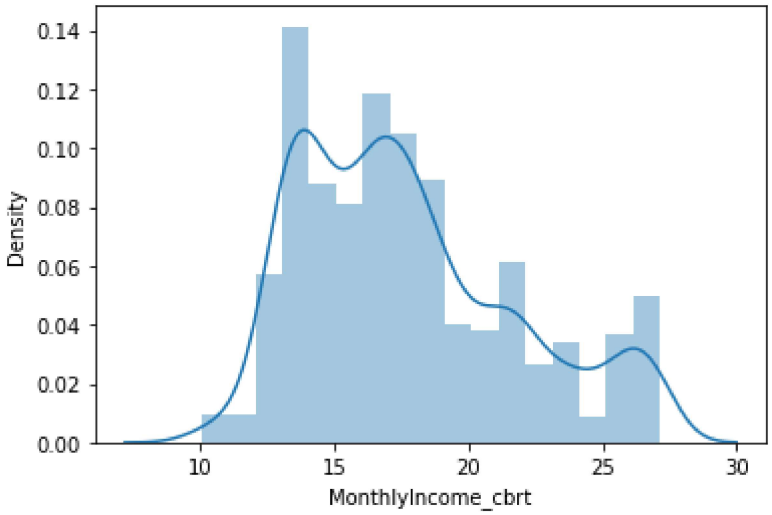
In [14]:
```
#MonthlyIncome

sb.distplot(df.MonthlyIncome)
plt.show()

df['MonthlyIncome_sqrt']=np.sqrt(df.MonthlyIncome)
sb.distplot(df.MonthlyIncome_sqrt)
plt.show()
```

```
df['MonthlyIncome_log']=np.log(df.MonthlyIncome)
sb.distplot(df.MonthlyIncome_log)
plt.show()

df['MonthlyIncome_cbrt']=np.cbrt(df.MonthlyIncome)
sb.distplot(df.MonthlyIncome_cbrt)
plt.show()
```

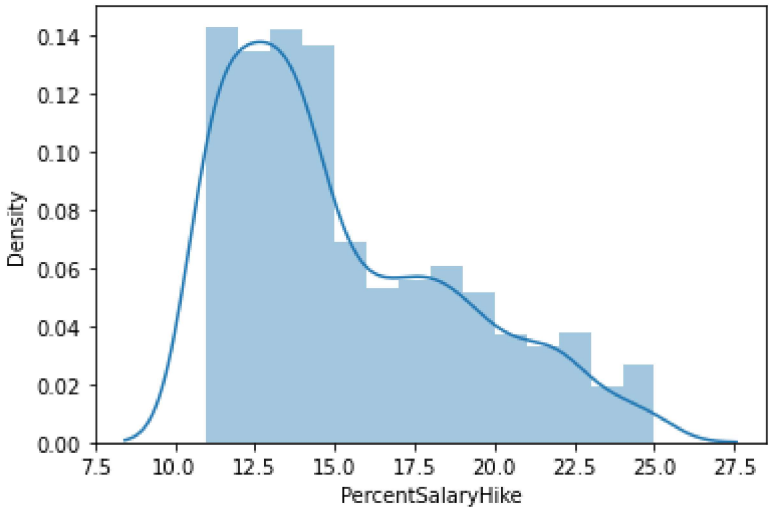


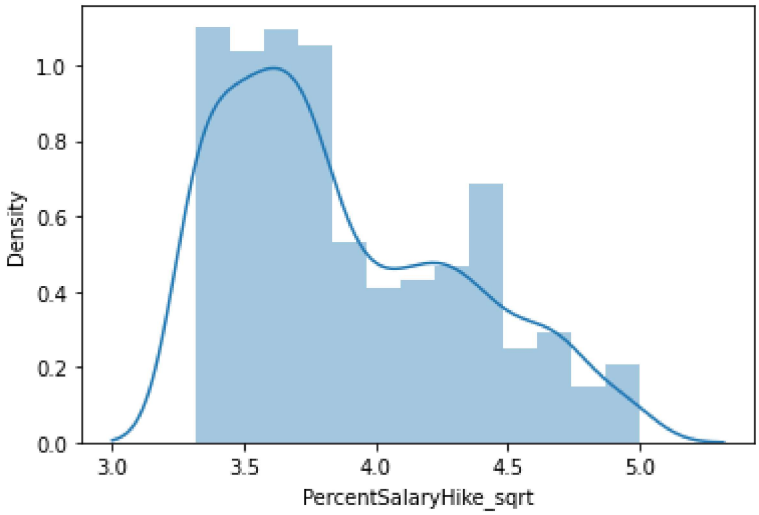


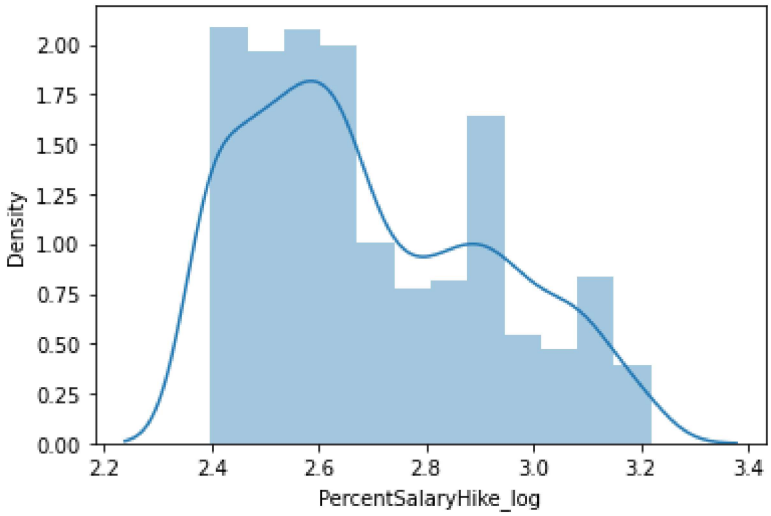


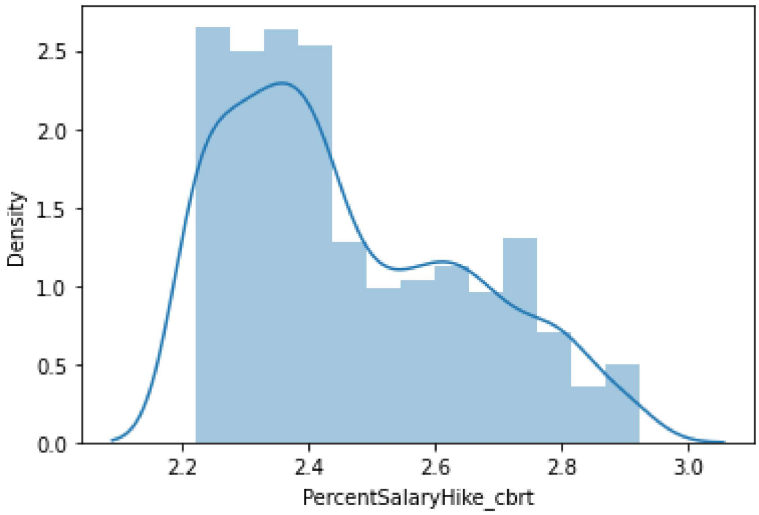


In [15]:
```
#PercentSalaryHike

sb.distplot(df.PercentSalaryHike)
plt.show()

df['PercentSalaryHike_sqrt']=np.sqrt(df.PercentSalaryHike)
sb.distplot(df.PercentSalaryHike_sqrt)
plt.show()

df['PercentSalaryHike_log']=np.log(df.PercentSalaryHike)
sb.distplot(df.PercentSalaryHike_log)
plt.show()

df['PercentSalaryHike_cbrt']=np.cbrt(df.PercentSalaryHike)
sb.distplot(df.PercentSalaryHike_cbrt)
plt.show()
```



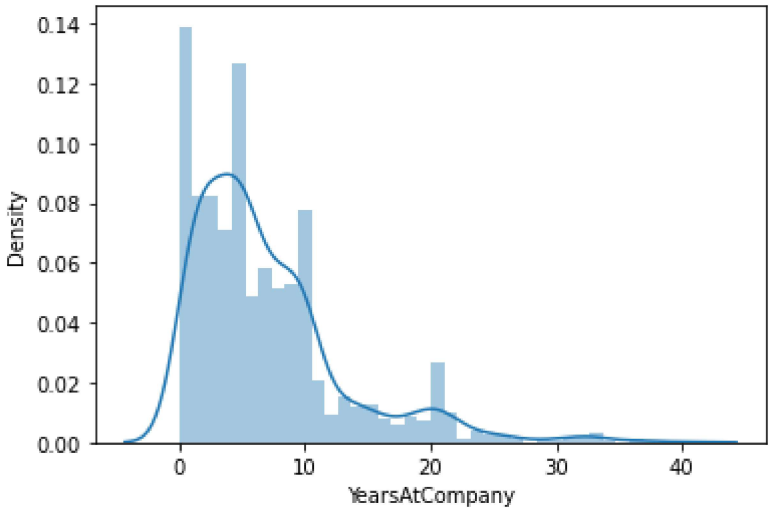


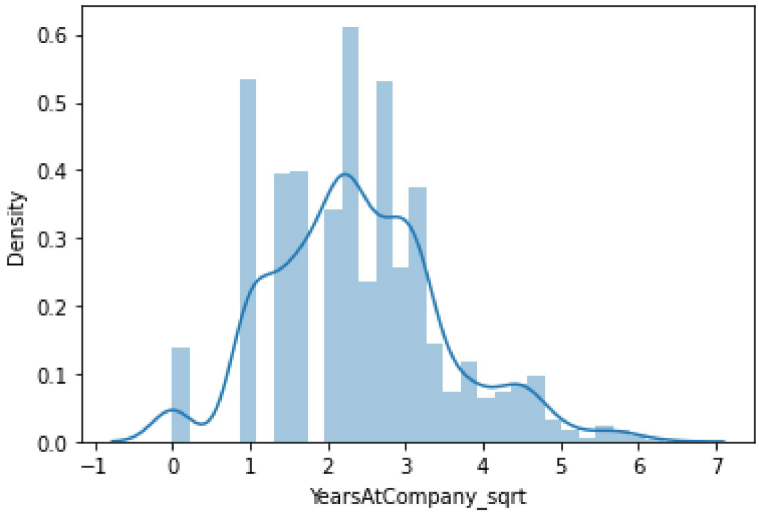


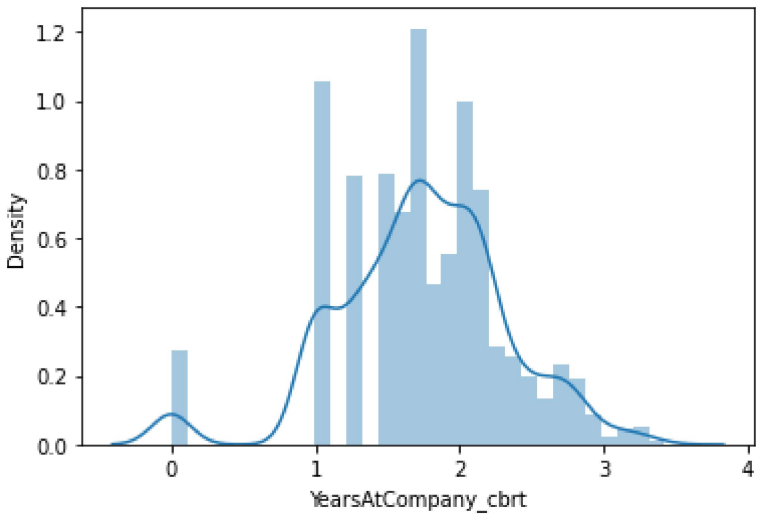


In [16]:
```
#YearsAtCompany

sb.distplot(df.YearsAtCompany)
plt.show()

df['YearsAtCompany_sqrt']=np.sqrt(df.YearsAtCompany)
sb.distplot(df.YearsAtCompany_sqrt)
plt.show()
```

```
df['YearsAtCompany_cbrt']=np.cbrt(df.YearsAtCompany)
sb.distplot(df.YearsAtCompany_cbrt)
plt.show()
```

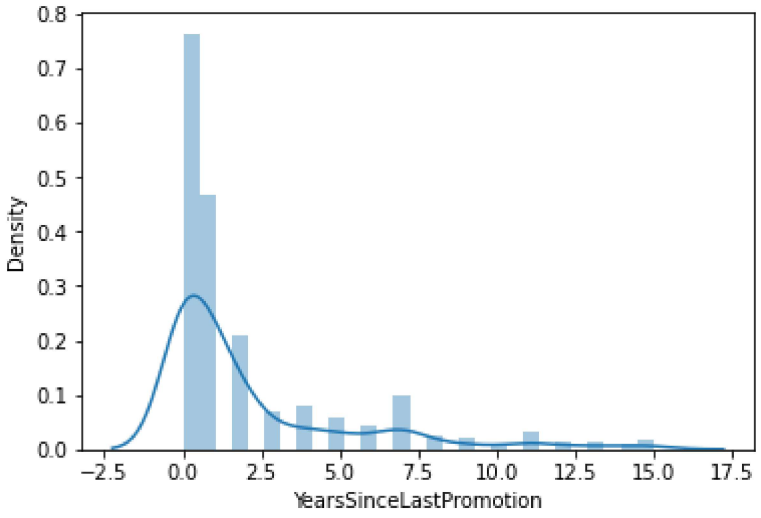


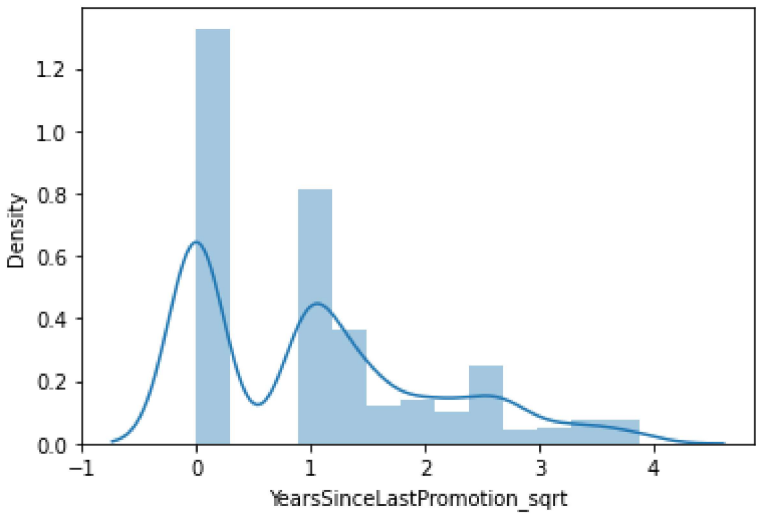




In [17]:
```
#YearsSinceLastPromotion

sb.distplot(df.YearsSinceLastPromotion)
plt.show()

df['YearsSinceLastPromotion_sqrt']=np.sqrt(df.YearsSinceLastPromotion)
```

```
sb.distplot(df.YearsSinceLastPromotion_sqrt)
plt.show()
```

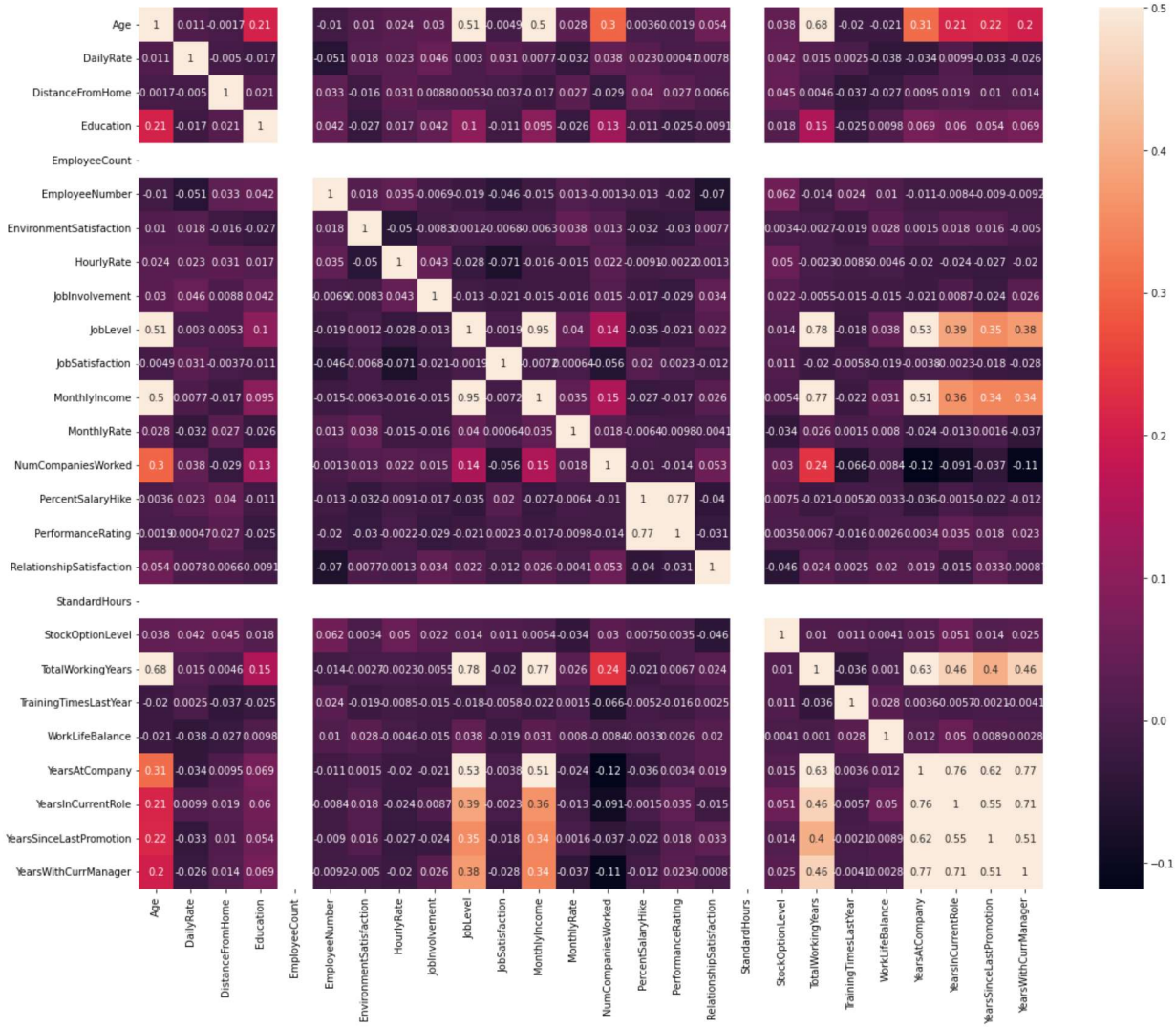




Checking correlation through Heatmap

In [18]:
```
corr_heatmap=con_df.corr()
ax=plt.subplots(figsize=(20,16))
sb.heatmap(corr_heatmap,vmax=0.5,annot=True)
```

Out[18]:
```
<AxesSubplot:>
```

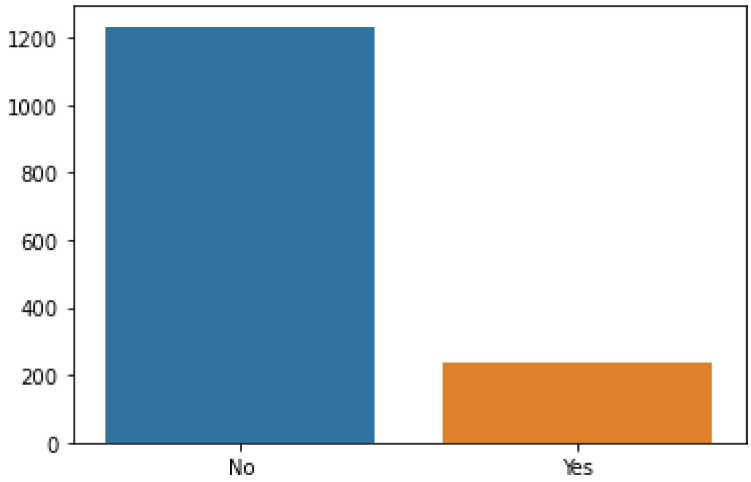


In [19]:
```
#show how much %employee left the organization is as follow:
df.Attrition.value_counts(normalize=True)

#16% of the employees left the companay 84% did not leave the company(class imbalace p
```

Out[19]:
```
No     0.838776
Yes    0.161224
Name: Attrition, dtype: float64
```

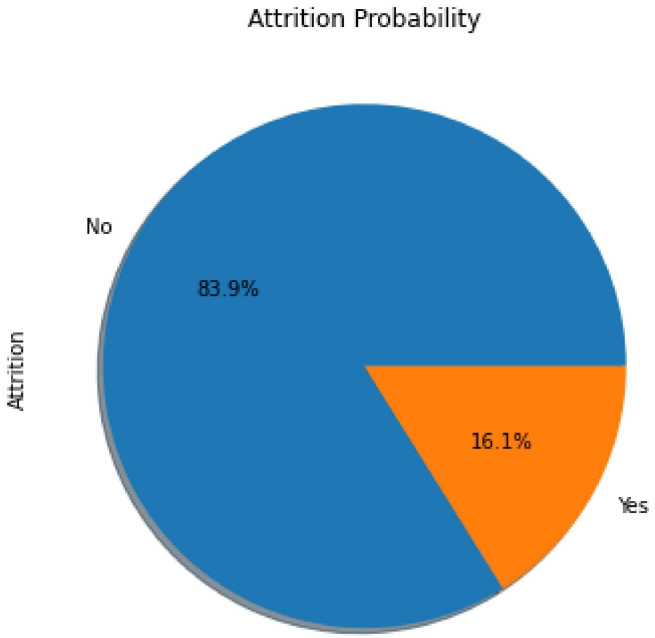
In [20]:
```
Attrition=df.Attrition.value_counts()
sb.barplot(x=Attrition.index,y=Attrition.values)
```

Out[20]:
```
<AxesSubplot:>
```



In [21]:
```
ax=plt.subplots(figsize=(6,6))
ax=df['Attrition'].value_counts().plot.pie(explode=[0,0],autopct='%1.1f%%',shadow=True
ax.set_title('Attrition Probability')
```

Out[21]:
```
Text(0.5, 1.0, 'Attrition Probability')
```



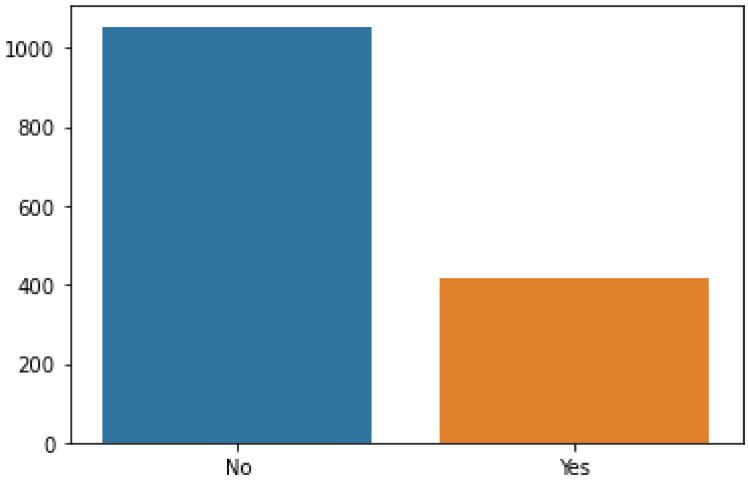
This cleary shown that no.of "NO" records is 84% and no. of "16%" this means data is imbalanced. this is class imbalanced problem

In [22]:
```
df.OverTime.value_counts(normalize=True)
```

Out[22]:
```
No     0.717007
Yes    0.282993
Name: OverTime, dtype: float64
```

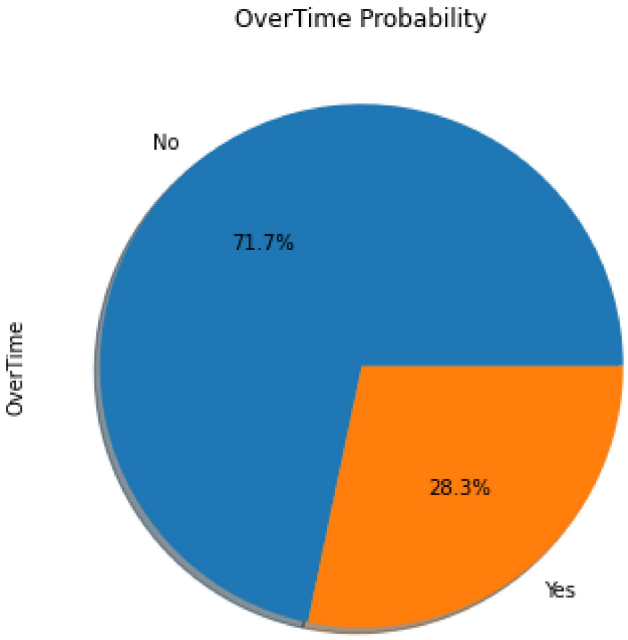
In [23]:
```
OverTime=df.OverTime.value_counts()
sb.barplot(x=OverTime.index,y=OverTime.values)
```

Out[23]:
```
<AxesSubplot:>
```



In [24]:
```
ax=plt.subplots(figsize=(6,6))
ax=df['OverTime'].value_counts().plot.pie(explode=[0,0],autopct='%1.1f%%',shadow=True)
ax.set_title('OverTime Probability')
```

Out[24]:
```
Text(0.5, 1.0, 'OverTime Probability')
```

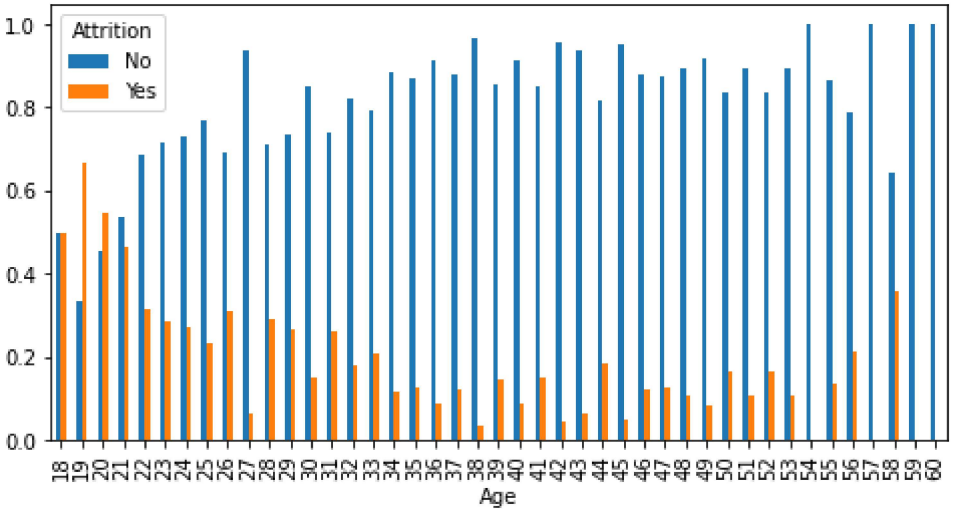


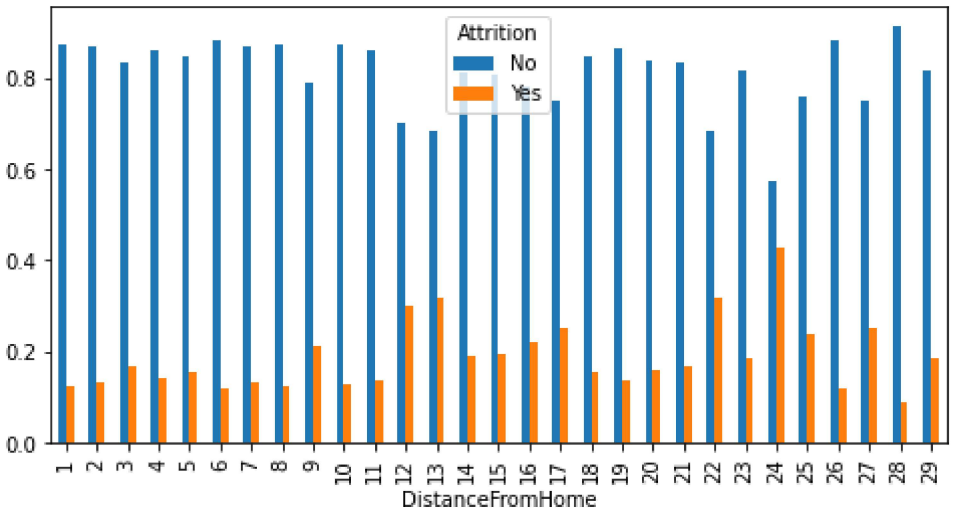
Bar Plots

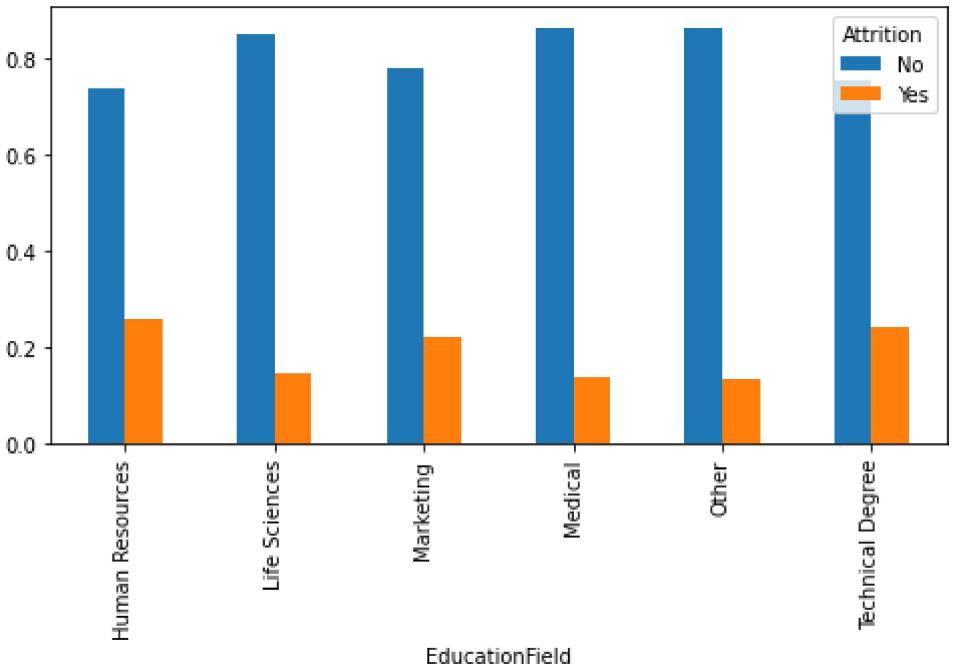
In [25]:
```
Barplot_columns=['Age','DistanceFromHome','EducationField','JobInvolvement','JobLevel'
                 'OverTime','TotalWorkingYears','TrainingTimesLastYear','WorkLifeBalanc
                  'YearsInCurrentRole']
```

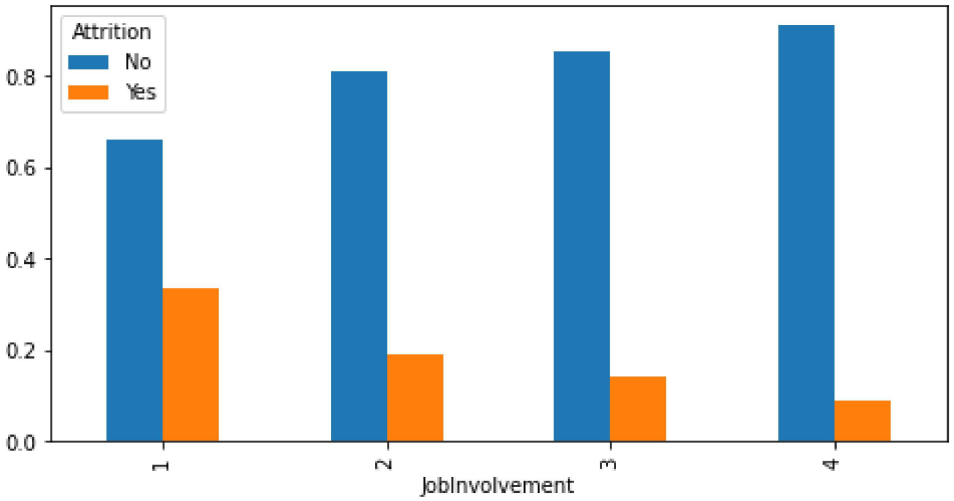
In [26]:
```
#Mehod the perform Bar plot

def Bar_plots(var):
    col=pd.crosstab(df[var],df.Attrition)
    col.div(col.sum(1).astype(float),axis=0).plot(kind='bar',stacked=False,figsize=(8,
    plt.xticks(rotation=90)
```

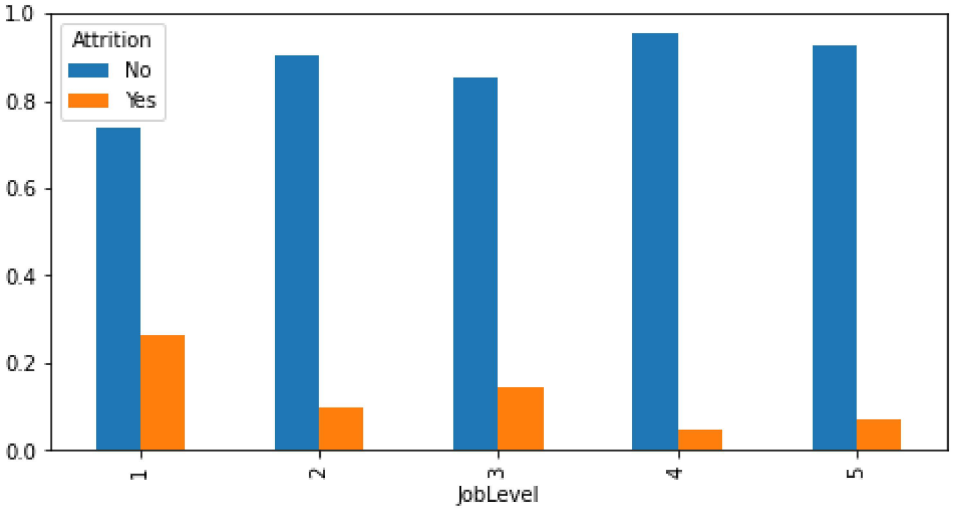
In [27]:
```
for col in Barplot_columns:
    Bar_plots(col)
```

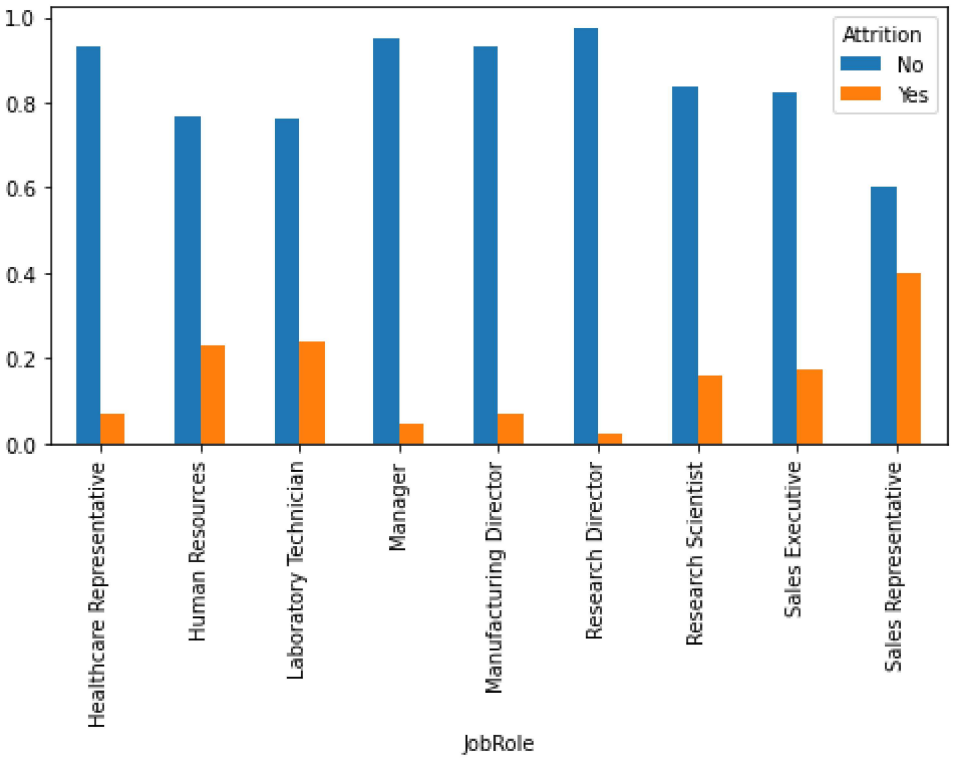


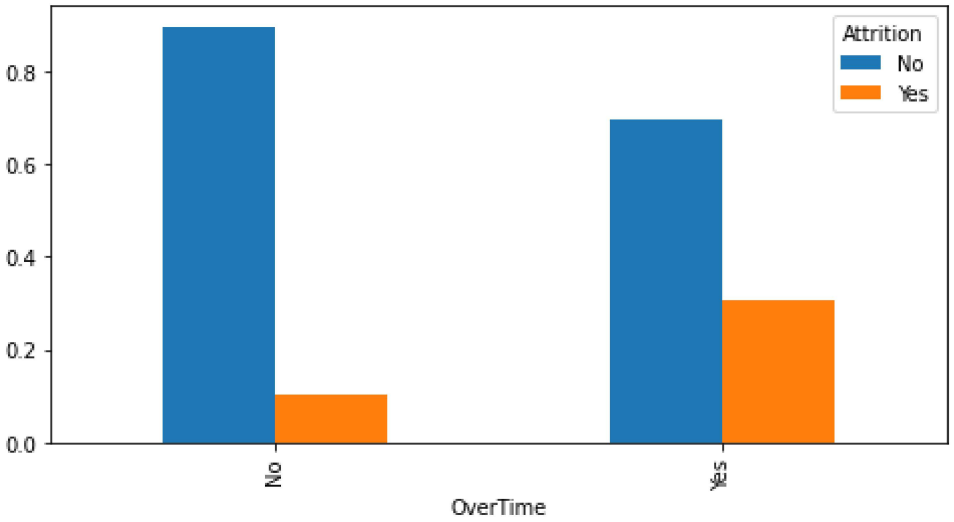


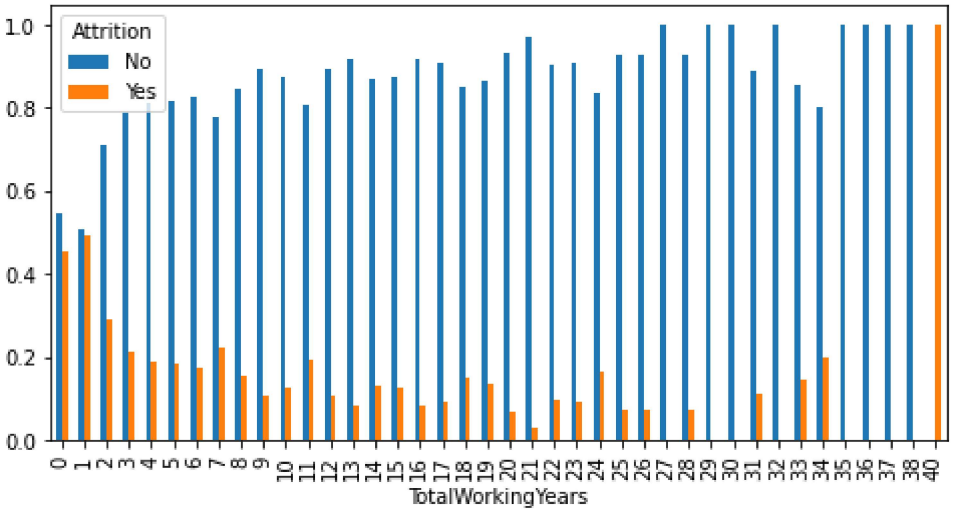


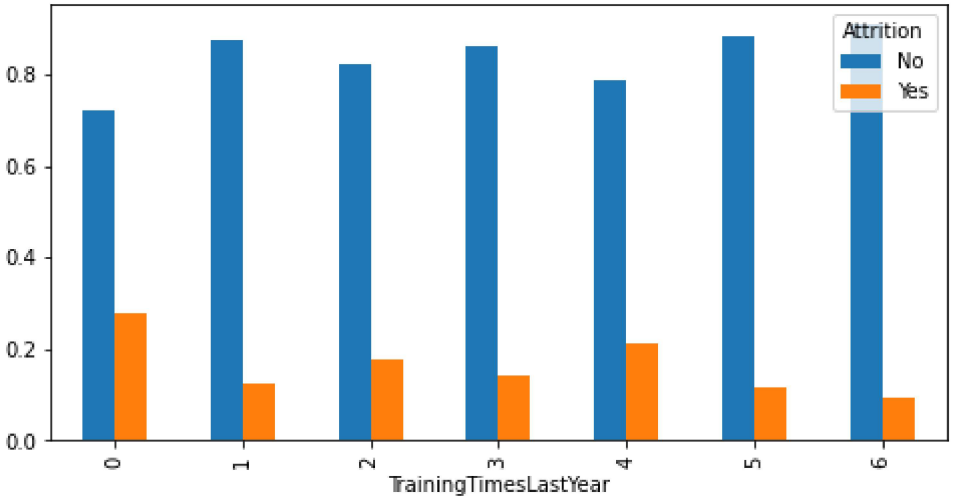

Attrition
No
Yes
0.0
0.2
0.4
0.6
0.8
1
2
3
4
JobInvolvement

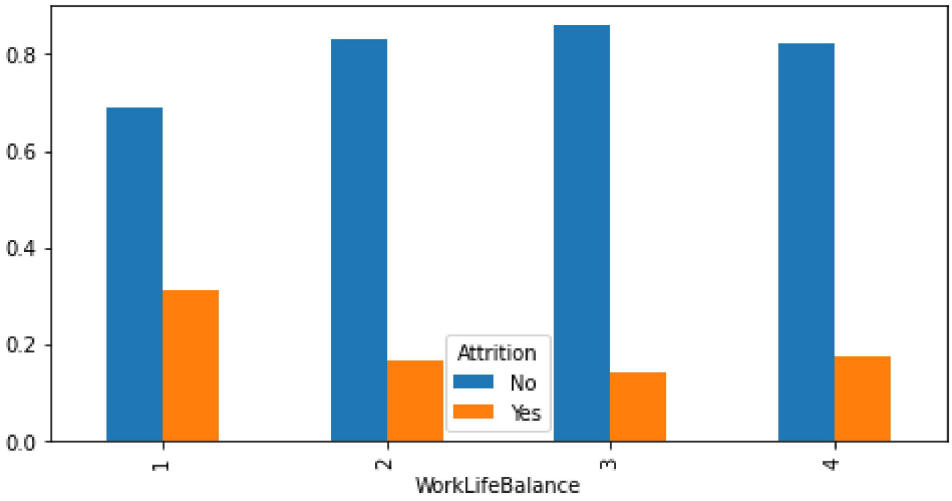

Attrition
No
Yes
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5
JobLevel

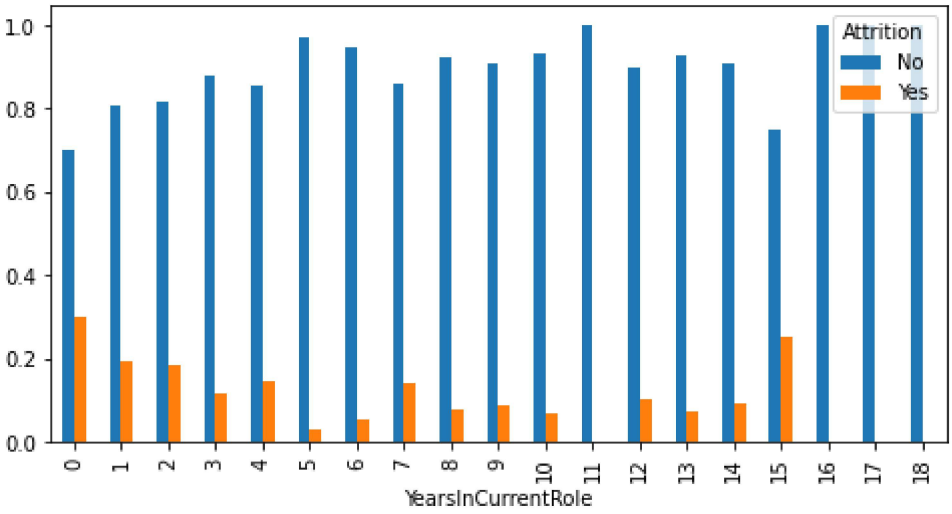

Attrition
No
Yes
0.0
0.2
0.4
0.6
0.8
1.0
Healthcare Representative
Human Resources
Laboratory Technician
Manager
Manufacturing Director
Research Director
Research Scientist
Sales Executive
Sales Representative
JobRole


Attrition
No
Yes
0.0
0.2
0.4
0.6
0.8
No
Yes
OverTime


Attrition
No
Yes
0.0
0.2
0.4
0.6
0.8
1.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 40
TotalWorkingYears


Attrition
No
Yes
0.0
0.2
0.4
0.6
0.8
0 1 2 3 4 5 6
TrainingTimesLastYear





Insights:-

1) Attrition is very high with employees having ahe in between 18 to 22
This might be due to students who are doing intership or small contract jobs or on bench

2) Attrition is more when the distance of office is more from home.

3) Attrition is high with employees education in the fiel of Human Resurces,
Marketing and Technical Degree to other fields

4) Attrition is high with employees whose jon role is in the field of Human Resources,
Sales Representative,Sales Excecutive,Laboratory Technical

5) Employees who work over time have high attrition than employees who did not do over

6) Employees who are working less than 2 years have more attrition

In [53]:
```
#show data int64 only columns
```

```
df.select_dtypes(include=['int64']).columns
```

Out[53]:
```
Index(['Age', 'DailyRate', 'DistanceFromHome', 'Education', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobSatisfaction', 'MonthlyIncome',
       'MonthlyRate', 'NumCompaniesWorked', 'PercentSalaryHike',
       'PerformanceRating', 'RelationshipSatisfaction', 'StandardHours',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager'],
      dtype='object')
```

In [59]:
```
#show data object only columns
df.select_dtypes(include=['object']).columns
```

Out[59]:
```
Index(['Attrition', 'BusinessTravel', 'Department', 'EducationField', 'Gender',
       'JobRole', 'MaritalStatus', 'Over18', 'OverTime'],
      dtype='object')
```

In [67]:
```
#show data float64 only columns
df.select_dtypes(include=['float64']).columns
```

Out[67]:
```
Index(['DistanceFromHome_sqrt', 'DistanceFromHome_log',
       'DistanceFromHome_cbrt', 'MonthlyIncome_sqrt', 'MonthlyIncome_log',
       'MonthlyIncome_cbrt', 'PercentSalaryHike_sqrt', 'PercentSalaryHike_log',
       'PercentSalaryHike_cbrt', 'YearsAtCompany_sqrt', 'YearsAtCompany_cbrt',
       'YearsSinceLastPromotion_sqrt'],
      dtype='object')
```

Outlier Detection and Treatment

In [54]:
```
import numpy as np
from scipy import stats
```

In [57]:
```
z=np.abs(stats.zscore(df[['Age', 'DailyRate', 'DistanceFromHome', 'Education',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobSatisfaction', 'MonthlyIncome',
       'MonthlyRate', 'NumCompaniesWorked', 'PercentSalaryHike',
       'PerformanceRating', 'RelationshipSatisfaction', 'StandardHours',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager']]))


#Looking the code and the output above,it is difficult to say which data point is an o
#Let's try and define a threshold to identify an outlier.

threshold=3
print(np.where(z>3))
```

```
(array([  28,   45,   62,   62,   63,   64,   85,   98,   98,  110,  123,
        123,  123,  126,  126,  126,  153,  178,  187,  187,  190,  190,
        218,  231,  231,  237,  237,  270,  270,  281,  326,  386,  386,
        401,  411,  425,  425,  427,  445,  466,  473,  477,  535,  561,
        561,  584,  592,  595,  595,  595,  616,  624,  635,  653,  653,
        677,  686,  701,  716,  746,  749,  752,  799,  838,  861,  861,
        875,  875,  894,  914,  914,  918,  922,  926,  926,  937,  956,
        962,  976,  976, 1008, 1024, 1043, 1078, 1078, 1086, 1086, 1093,
       1111, 1116, 1116, 1135, 1138, 1138, 1156, 1184, 1221, 1223, 1242,
       1295, 1301, 1301, 1303, 1327, 1331, 1348, 1351, 1401, 1414, 1430],
      dtype=int64), array([24, 23, 21, 23, 22, 23, 18, 18, 21, 23, 22, 23, 24, 18, 2
1, 23, 24,
       23, 18, 24, 21, 22, 23, 22, 24, 21, 23, 18, 21, 22, 23, 23, 24, 18,
       21, 21, 23, 23, 18, 22, 21, 21, 23, 21, 24, 23, 21, 18, 21, 23, 24,
       18, 24, 21, 23, 21, 24, 23, 22, 22, 21, 23, 23, 23, 21, 23, 23, 24,
       18, 21, 23, 21, 23, 23, 24, 23, 18, 21, 22, 23, 23, 22, 18, 23, 24,
       21, 23, 23, 21, 18, 21, 21, 21, 23, 23, 18, 23, 23, 23, 23, 18, 23,
       23, 22, 23, 24, 22, 18, 23, 22], dtype=int64))
```

Removal of Outliers

In [65]:
```
df_out=df[(z<3).all(axis=1)]
```

In [68]:
```
df_out1=df_out.drop(['DistanceFromHome_sqrt', 'DistanceFromHome_log',
       'DistanceFromHome_cbrt', 'MonthlyIncome_sqrt', 'MonthlyIncome_log',
       'MonthlyIncome_cbrt', 'PercentSalaryHike_sqrt', 'PercentSalaryHike_log',
       'PercentSalaryHike_cbrt', 'YearsAtCompany_sqrt', 'YearsAtCompany_cbrt',
       'YearsSinceLastPromotion_sqrt'],axis=1)
```

In [69]:
```
df_out1
```

Out[69]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationFie

Here no outliers found

In [73]:
```
#Checking rows and columns
df.shape
```

Out[73]:
```
(1470, 47)
```

Dividing final dataset into categorical and continuous variables

In [103…
```
numerical_df=df.select_dtypes(include=np.number)
categorical_df=df.select_dtypes(exclude=np.number)
numeric_cols=list(numerical_df.columns)
categorical_cols=list(categorical_df.columns)
```

Converting categorical variable to binary

In [104…
```
categorical_df_dummies=pd.get_dummies(df[categorical_cols],drop_first=True)
final_df=pd.concat([categorical_df_dummies,numerical_df],axis=1)
final_df.head()
```

Out[104]:

	Attrition_Yes	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	Department_Research & Development
0	1	0	1	0
1	0	1	0	1
2	1	0	1	1
3	0	1	0	1
4	0	0	1	1

In [105…
```
final_df.shape #After converting data
```

Out[105]:
```
(1470, 60)
```

In [106…
```
df.shape #Original dataset
```

Out[106]:
```
(1470, 47)
```

Split the data

In [110…
```
x=final_df.drop('Attrition_Yes',axis=1)
y=final_df['Attrition_Yes']

print("shape of x =",x.shape)
print("shape of y =",y.shape)
```

```
shape of x = (1470, 59)
shape of y = (1470,)
```

In [111…
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=51)

print('shape of x_train =', x_train.shape)
print('shape of y_train =', y_train.shape)
print('shape of x_test =', x_test.shape)
print('shape of y_test =', y_test.shape)
```

```
shape of x_train = (1176, 59)
shape of y_train = (1176,)
shape of x_test = (294, 59)
shape of y_test = (294,)
```

Train Random Forest Classification Model

In [121…
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(n_estimators=100,criterion='gini')
rfc.fit(x_train,y_train)
rfc.score(x_test,y_test)
```

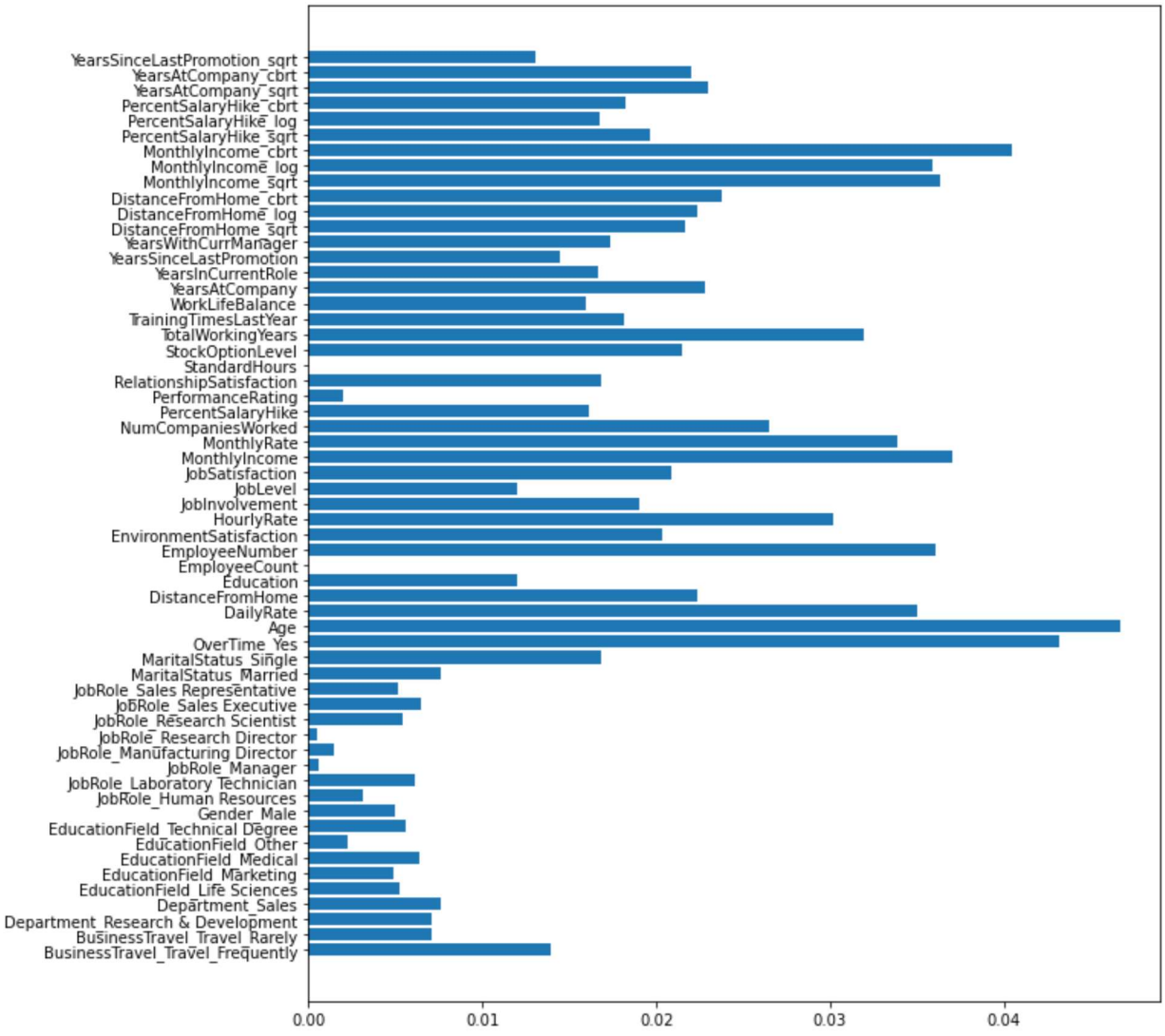
Out[121]:
```
0.8775510204081632
```

In [122…
```
rfc.feature_importances_
```

Out[122]:
```
array([0.0139377 , 0.00712648, 0.00712625, 0.00758676, 0.00527661,
       0.00486879, 0.00642305, 0.00228469, 0.00556031, 0.00497098,
       0.00309704, 0.00610068, 0.00061142, 0.00149082, 0.00047025,
       0.00539684, 0.00644728, 0.00511033, 0.00763569, 0.01683197,
       0.04313971, 0.04665605, 0.03497547, 0.02237621, 0.01198525,
       0.        , 0.03604359, 0.02033375, 0.03018873, 0.01904045,
       0.01204145, 0.02086368, 0.036979  , 0.0338203 , 0.02648966,
       0.01608937, 0.00200203, 0.0168563 , 0.        , 0.02151851,
       0.03193125, 0.0181248 , 0.01594546, 0.02275758, 0.01664508,
       0.01448618, 0.01735743, 0.02165617, 0.02233043, 0.02375463,
       0.03628631, 0.03590501, 0.04040348, 0.01961044, 0.01672837,
       0.01827089, 0.02300157, 0.02196516, 0.01308632])
```

In [123…
```
ax=plt.subplots(figsize=(10,12))
plt.barh(x.columns,rfc.feature_importances_)
```

Out[123]:
```
<BarContainer object of 59 artists>
```



Logistic Regression

In [124…
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
lr.score(x_test,y_test)
```

Out[124]:
```
0.8469387755102041
```

In [128…
```
pred_lr=lr.predict (x_test)
y_test['predict']=pred_lr
```

In [144…
```
y_test.head()
```

Out[144]:
```
527    0
183    0
80     0
713    0
493    0
Name: Attrition_Yes, dtype: object
```

Model Saving

In [146…
```
import pickle
filename='HR-Employee-Attrition'
pickle.dump(lr, open(filename,'wb'))
```

In [ ]:
```

```