Written by Juhi Mishra

Predicting the Attrition in HR

This dataset is about how the HR plays an important role for company by analyzing, starting from filling vacancy, preparing compensation and benefit such that employees will be much benefited and will try to be there for longer period. If not attrition comes into picture which is directly or indirectly loss for a company as company invest on each employee starting from hiring ,training, appointing then making them understand the pattern of work and environment of the company. Ultimately if an old employee leaves it will be a great loss for a company as again the same process to be started with new one. So HR main critical role comes here to minimise attrition rate for any company specially for those employees who are good performer. In this dataset will go through HR analysis on attrition rate and controlling them for more productivity and make the company more profitable.

Importing the Important Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

import warnings
warnings.filterwarnings('ignore')

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

from sklearn.model_selection import KFold,
cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import SelectKBest,
f_classic
from sklearn.metrics import
confusion_matrix,classification_report
from sklearn.model_selection import GridSearchCV
```

Get the Dataset(CSV form)

```
df = pd.read_csv('/Users/juhimishra/Downloads/
IBM_HR_Attrition_Rate_Analytics-master/WA_Fn-UseC_-
HR-Employee-Attrition.csv')
df.head()
```

This dataset is a classification based problem as our target consist of binary information which is either in yes/no so we need to build our model accordingly. Also need to check whether our data is imbalanced dataset or not as our target is also a categorical column and mostly in categorical column we should check whether there dataset is a balanced or imbalanced dataset.

For all columns visualisation will use pd.set_option as

```
pd. set_option("display.max_columns", None)
```

EDA(Explanatory Data Analysis)

```
print('No of rows:',df.shape[0])
print('no of columns:',df.shape[1])
```

```
Total rows:  1470
Total columns:  35
```

No of column here we are having 35 and rows 1470 .

Let's check what are the columns out dataset do have

```
df.columns
```

```
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')
```

With the help of info method will able to analyse objective and integer columns. Null or missing values will also be able to identify

```
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)
```

No missing values as of now but having some object based columns. Let's segregate object and integer columns and will check total no for the same.

```
# Check the name of coloumns which contain string
df.select_dtypes(include='object').columns
```

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

```
# Check the no. of coloumns which contain string
len(df.select_dtypes(include='object').columns)
```

```
9
```

we are getting total 9 object/string datatype which we need to convert

to numeric form for our machine algorithm to understand

```
# Check the name of coloumns which contain numerical
value
df.select_dtypes(include=['int64',
'float64']).columns
```

```
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')
```

```
# Check the no. of coloumns which contain float or
integer
len(df.select_dtypes(include=['int64','float64']).col
umns)
```

```
26
```

Total integer or float data type are 26.

Below Listed the detail of column with short descriptions :
Feature Variable :

1. Age = Age group of employees in a company
2. Busineess Travel = How frequently or rarely company sends employee on travelling . From this will able to know which age group or which field employee travels a

lot or less

3. Daily Rate = per day wages that employee get
4. Department = In which depratment the particular employee works
5. DistanceFromHome = how long or shorter distance which employee stays
6. Education = under one education field how many employee fall of same age group means how many employees are from life science background or medical etc.

7. EducationField = from which stream the particular employee is
8. Employee count = no of employees falls under each age group
9. Employee Number = How many employees are from same age group
10. Environment Satisfaction = How many employees are satisfied from working environment
11. Gender = specifying category of male and female in a company
12. Hourly Rate = Per hour wages of an employee
13. Job involvement = How much an employee is involved in job assigned to them rating given as per that.
14. Job Label = on which position which particular employee is
15. Job Role = Particular Role assigned to each employee as per their education background and ability
16. Job Satisfaction = How much an employee is satisfied with the job they have been assigned
17. Marital Status = Married or not
18. Monthly Income = How much basic an employee get in a month
19. Monthly Rate = Total salary getting in a month
20. NoCompanieSWorked = how many company does an employee worked.
21. Over18 = Whether an employee age is above 18 or not
22. Overtime = overtime done or not by an employee
23. percentsalaryhike = what percentage hike does an employee get

ever year

24. performance rating = Rating provided by HR or manager as per the performance of an employee
25. Relationship Satisfaction = how much an employee is satisfied with peer or subordinate or manager.
26. Standard Hours = working hours fixed by the company.
27. Stock option label = No of employees who got the right to purchase no of shares in lieu of salary in the company at a discounted price
28. Total Working Years = Total no of years an employee worked from when he/she started their career.
29. TrainingTimesLastYear = No of times an employee sent for traing by the company to improve their skill.
30. WorkLifeBalance = the achievement by employees of equality between time spent working and personal life
31. YearsAtCompany = How long the employee is working in present company
32. YearsInCurrentRole = Since how long in a job role assigned to them.
33. YearsSinceLastPromotion = after getting last promotion how long an employee is continuing.
34. YearsWithCurrManager = how long an employee is working with his/her current manager.

Target Variable

Attrition here is our target on which will going to build our model for prediction. Clearly we can see that this problem is classification based problem and binary problem as column consist of only yes and no . Further will check whether our data is imbalanced data or not as for categorical column it is mandatory to check otherwise it will affect model performance.

To check the static data of customer churn used describe method. With the help of describe method we

can see all continuous data static details along with missing values, mean, std and Quantile detail.

```
# Checking static data using describe method
df.describe()
```

	Age	Attrition	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
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000
mean	36.923810	0.161224	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	2.728571	6502.931293	14313.103401	2.693197	15.209524	3.153741	2.712245	80.0	0.793878	11.279592	2.799320	2.761224	6.989116	4.229252	2.187755	4.123129	4.153741
std	9.135373	0.367863	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	1.102846	4707.956783	7117.786044	2.498009	3.659938	0.360824	1.081209	0.0	0.852077	7.780782	1.289271	0.706476	6.126525	3.623137	3.222430	3.568136	3.623137
min	18.000000	0.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	1.000000	1009.000000	2094.000000	0.000000	11.000000	3.000000	1.000000	80.0	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	30.000000	0.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	2.000000	2911.000000	8047.000000	1.000000	12.000000	3.000000	2.000000	80.0	0.000000	6.000000	2.000000	2.000000	3.000000	2.000000	0.000000	2.000000	2.000000
50%	36.000000	0.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	3.000000	4919.000000	14235.500000	2.000000	14.000000	3.000000	3.000000	80.0	1.000000	10.000000	3.000000	3.000000	5.000000	3.000000	1.000000	3.000000	3.000000
75%	43.000000	0.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	4.000000	8379.000000	20462.000000	4.000000	18.000000	3.000000	4.000000	80.0	1.000000	15.000000	3.000000	3.000000	9.000000	7.000000	3.000000	7.000000	7.000000
max	60.000000	1.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	4.000000	19999.000000	26999.000000	9.000000	25.000000	4.000000	4.000000	80.0	3.000000	40.000000	6.000000	4.000000	40.000000	18.000000	15.000000	15.000000	17.000000

	mean	std	min	25%
36.923810	36.923810	9.135373	18.000000	30.000000
0.161224	0.161224	0.367863	0.000000	0.000000
802.485714	802.485714	403.509100	102.000000	465.000000
9.192517	9.192517	8.106864	1.000000	2.000000
2.912925	2.912925	1.024165	1.000000	2.000000
1.0	1.0	0.0	1.0	1.0
1024.865306	1024.865306	602.024335	1.000000	491.250000
2.721769	2.721769	1.093082	1.000000	2.000000
65.891156	65.891156	20.329428	30.000000	48.000000
2.729932	2.729932	0.711561	1.000000	2.000000
2.063946	2.063946	1.106940	1.000000	1.000000
2.728571	2.728571	1.102846	1.000000	2.000000
650.293129	650.293129	470.795678	100.900000	291.100000
1431.310401	1431.310401	711.786044	209.400000	804.700000
2.693197	2.693197	2.498009	0.000000	1.000000
15.209524	15.209524	3.659938	11.000000	12.000000
3.153741	3.153741	0.360824	3.000000	3.000000
2.712245	2.712245	1.081209	1.000000	2.000000
80.0	80.0	0.0	80.0	80.0
0.793878	0.793878	0.852077	0.000000	0.000000
11.279592	11.279592	7.780782	0.000000	6.000000
2.799320	2.799320	1.289271	0.000000	2.000000
2.761224	2.761224	0.706476	1.000000	2.000000
7.008163	7.008163	6.126525	0.000000	3.000000
4.222952	4.222952	3.623137	0.000000	2.000000
2.187755	2.187755	3.222430	0.000000	0.000000
4.123129	4.123129	3.568136	0.000000	2.000000

50%	36.000000	0.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	3.000000	4919.000000	14235.500000	2.000000	14.000000	3.000000	3.000000	80.0	1.000000	10.000000	3.000000	3.000000	5.000000	3.000000	1.000000	3.000000
75%	43.000000	0.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	4.000000	8379.000000	20461.500000	4.000000	18.000000	3.000000	4.000000	80.0	1.000000	15.000000	3.000000	3.000000	9.000000	7.000000	3.000000	7.000000
max	60.000000	1.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	4.000000	19999.000000	26999.000000	9.000000	25.000000	4.000000	4.000000	80.0	3.000000	40.000000	6.000000	4.000000	40.000000	18.000000	15.000000	17.000000

Below are the observations drawn from above static data:

1. There are no missing values in any of the column
2. Min age as per the company law is 18 and max is 60 years
3. daily rate min wage is 102 and max is 1499
4. min distance from home starts min from 1 km and max is 29km
5. Education here is categorical column
6. Categorical columns as per above data are - Education,Employee Count,EnvironmentSatisfaction,JobInvolvement,JobLavel, `JobSatisfaction,PerformanceRating,RelationshipSat`

```
isfaction,StockOptionLevel,WorkLifebalance
```

As above with the help of describe method we can only find the detail of continuous variables not categorical variable so will use isna method . Using isna method can find the missing values even if any in categorical columns.

```
df.isna().sum().to_frame(name = 'Missing Values')
```

Missing Values	
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

No missing values even in categorical columns.

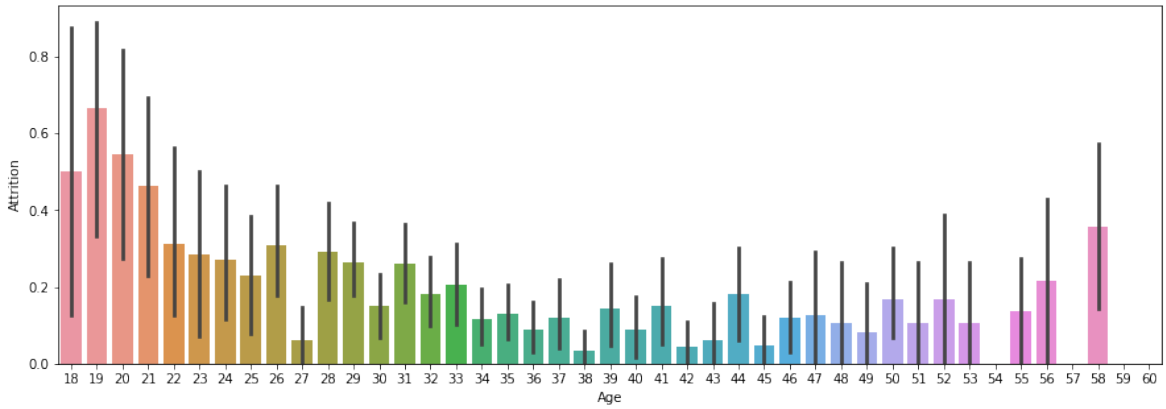
Analysing attrition on the basis of few seletive features using Visualisation Technique

Here for visualisation technique will use seaborn and matplotlib. Become easy and more clear for any one to understand what as an analyst you want to present. So here I am using some plotting technique and will have an insight what exactly our data is and what are the improvement

we can do.

```
plt.figure(figsize=(15,5))
sns.barplot(x ='Age',y='Attrition',data =df)
```

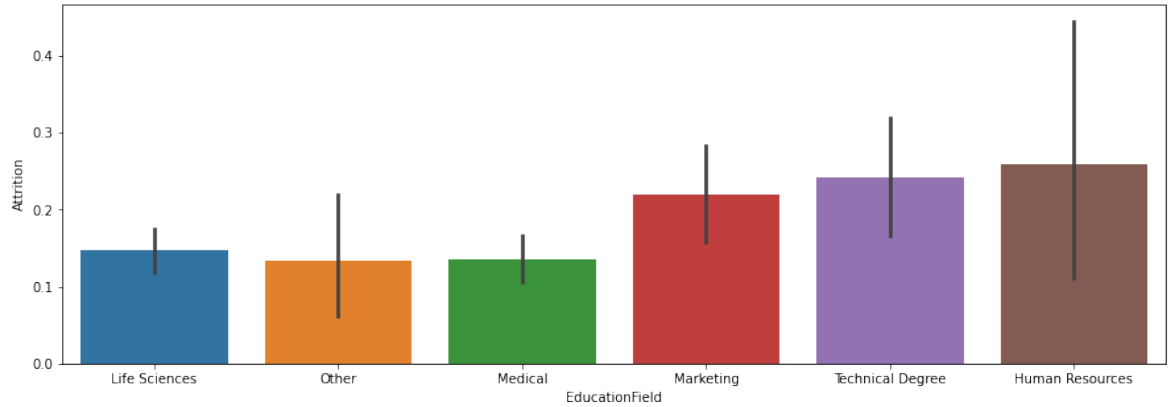
```
<AxesSubplot:xlabel='Age', ylabel='Attrition'>
```



From the above plot came to know that mostly young generations attrition are high mainly age groups of 19 years old while mid age employees try to stick to the company as we can see above age group of 38 -43 years are moving less. Youngsters may be looking for more opportunities in the market as we can consider that ,may be they are independent having less responsibilities so freely hunting for new opportunities. Company should try to search for what exactly these young talents are looking for and should try to reduce attrition rate which will not only help them to curb attrition but will also help to keep cost effective budget for company which will ultimately be profit for them.

```
plt.figure(figsize=(15,5))
sns.barplot(x ='EducationField',y='Attrition',data
=df)
```

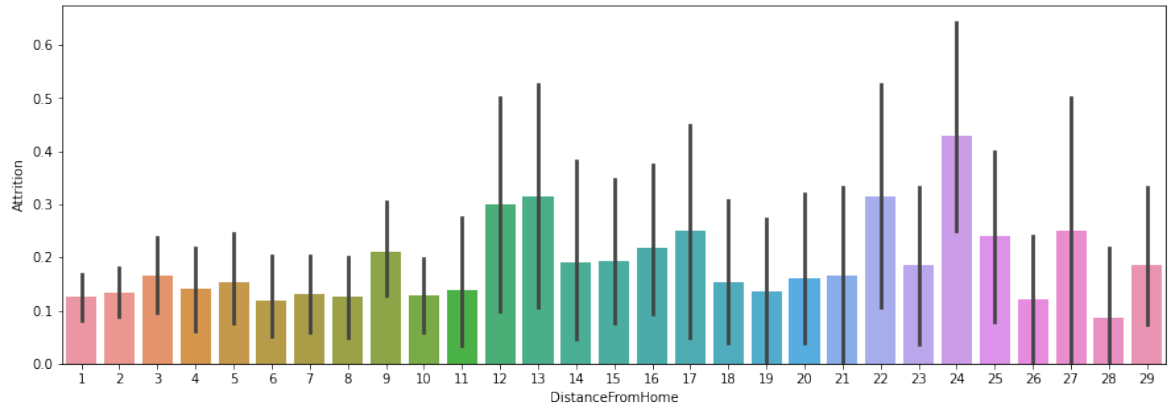
```
<AxesSubplot:xlabel='EducationField',
ylabel='Attrition'>
```



Here I used barplot to get the insight, from which education field the attrition rate is high. And we came across that mostly Human Resource background employee do have higher attrition rate which is definitely not acceptable as these people are only responsible for retaining other background employees to stay instead, these employees are showing higher attrition. We can consider either salary may be the factor or additional benefit and compensation. HR manager or whoever on an higher position should look into this and try to minimise the attrition by finding better solution.

```
plt.figure(figsize=(15,5))
sns.barplot(x ='DistanceFromHome',y='Attrition',data
=df)
```

```
<AxesSubplot:xlabel='DistanceFromHome',
ylabel='Attrition'>
```

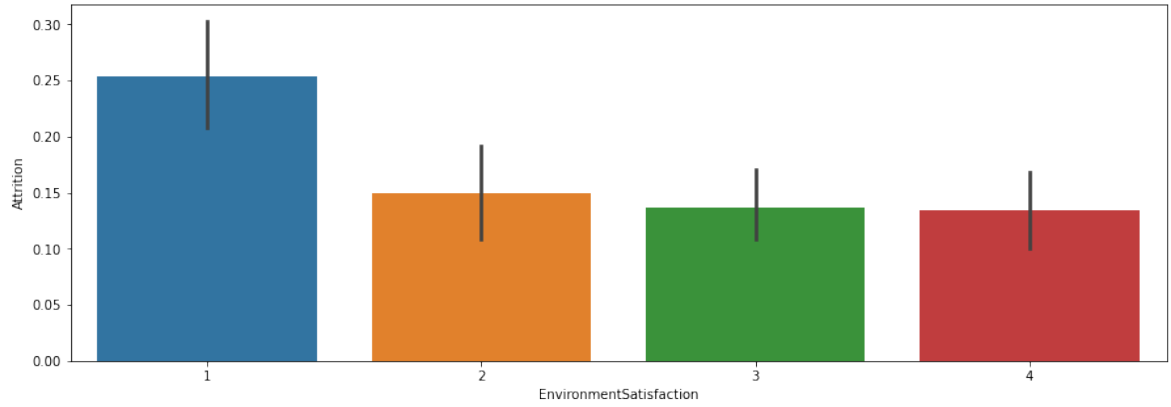


One of the factor is distance too. As from this barplot got the insight that

employee who stays far from there workplace are leaving more. As distance is more so it will definitely be time taking travelling and as we know that companies are strict for their rules and discipline maintanence, in which time is also one of the important factor. So here either company should not hire those who stays far away or should arrange something which may be helpful for them.

```
plt.figure(figsize=(15,5))
sns.barplot(x
='EnvironmentSatisfaction',y='Attrition',data =df)
```

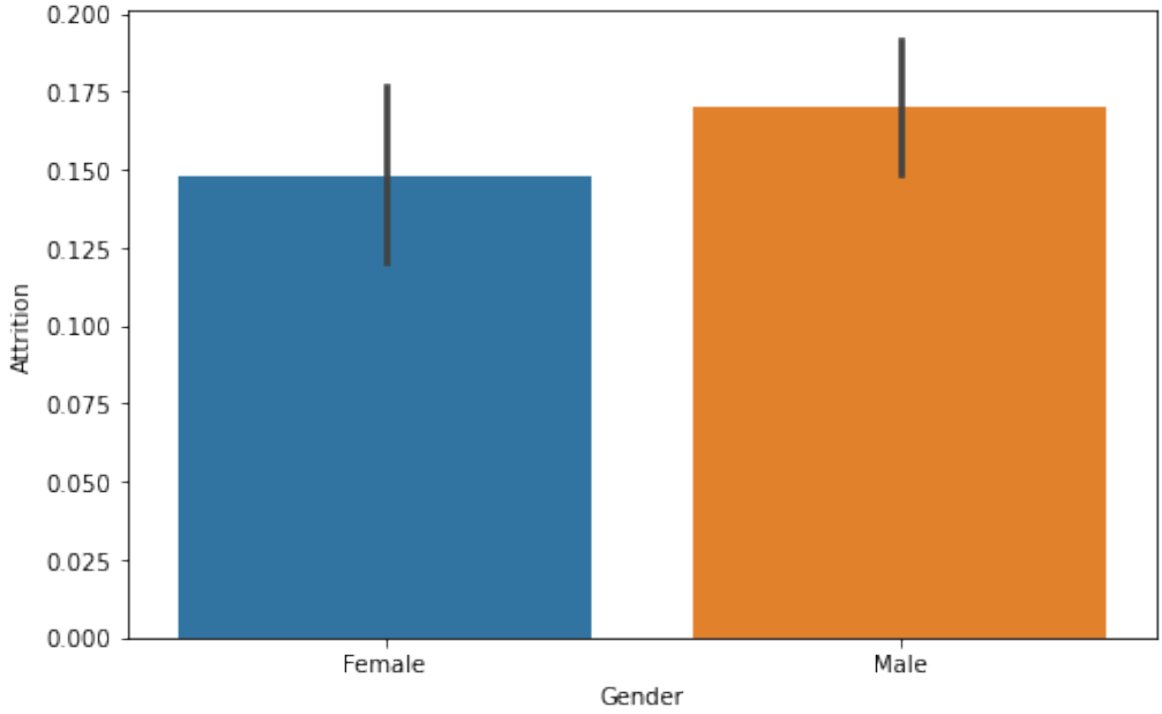
```
<AxesSubplot:xlabel='EnvironmentSatisfaction',
ylabel='Attrition'>
```



From this insight we can say that 1 is the lowest rating and 4 is the highest rating decided by company higher authority or HR dept. Most of the companies are distributing form to the employees to know there view and prospect so that company of can take better initiative on time. So here we got that environment satisfaction is also one of the serious issue for them as most of the employees rated 1 to environment satisfaction which definitely shows that most of them are leaving due to this factor too. Rest we can see that not having much difference.

```
plt.figure(figsize=(8,5))
sns.barplot(x ='Gender',y='Attrition',data =df)
```

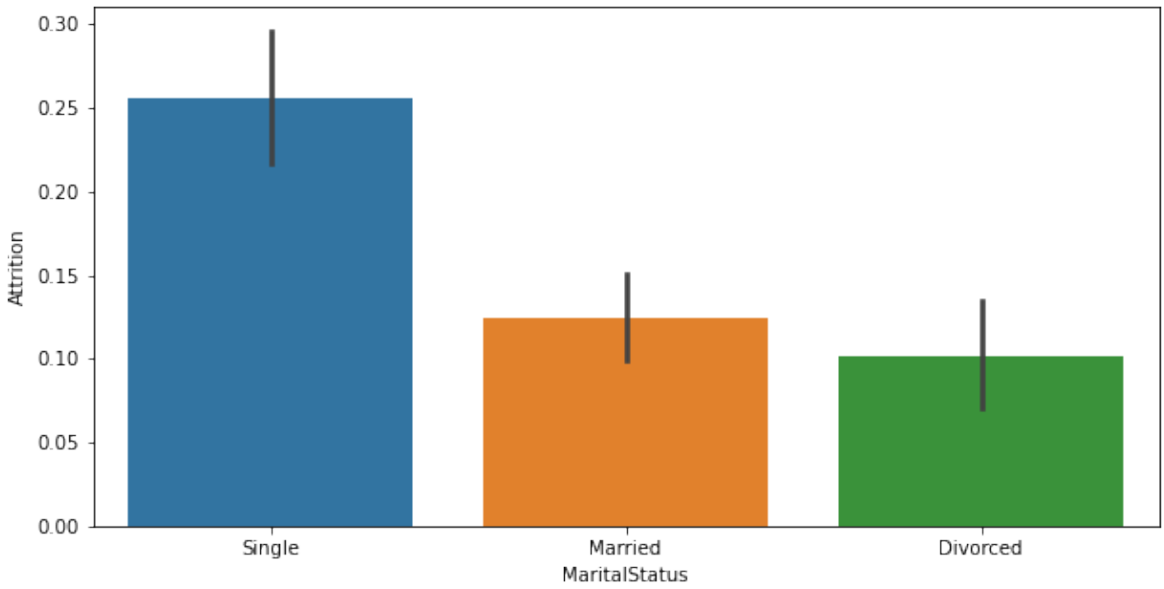
```
<AxesSubplot:xlabel='Gender', ylabel='Attrition'>
```



Male employees are leaving more compare to female employees . This may happen due to imbalance in gender as we know that in any of the company no of male are more than female so this result may be obvious.

```
plt.figure(figsize=(10,5))
sns.barplot(x ='MaritalStatus',y='Attrition',data
=df)
```

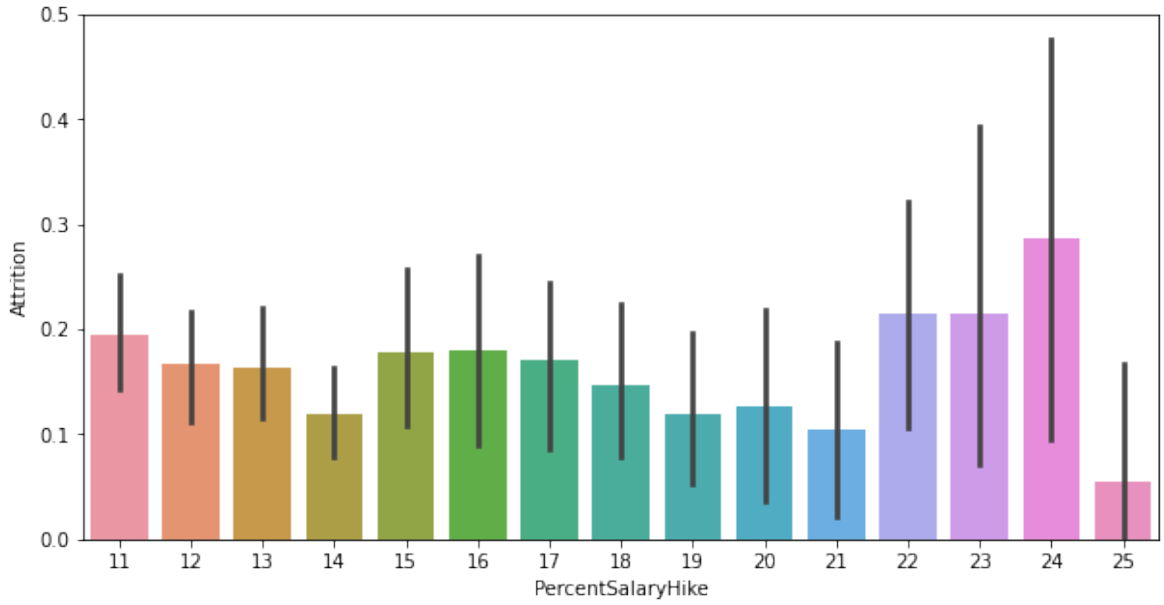
```
<AxesSubplot:xlabel='MaritalStatus',
ylabel='Attrition'>
```



Employees who are bachelor are leaving more . From this we analyse that either they are not much serious for their job or they are getting good offers from other company.

```
plt.figure(figsize=(10,5))
sns.barplot(x ='PercentSalaryHike',y='Attrition',data
=df)
```

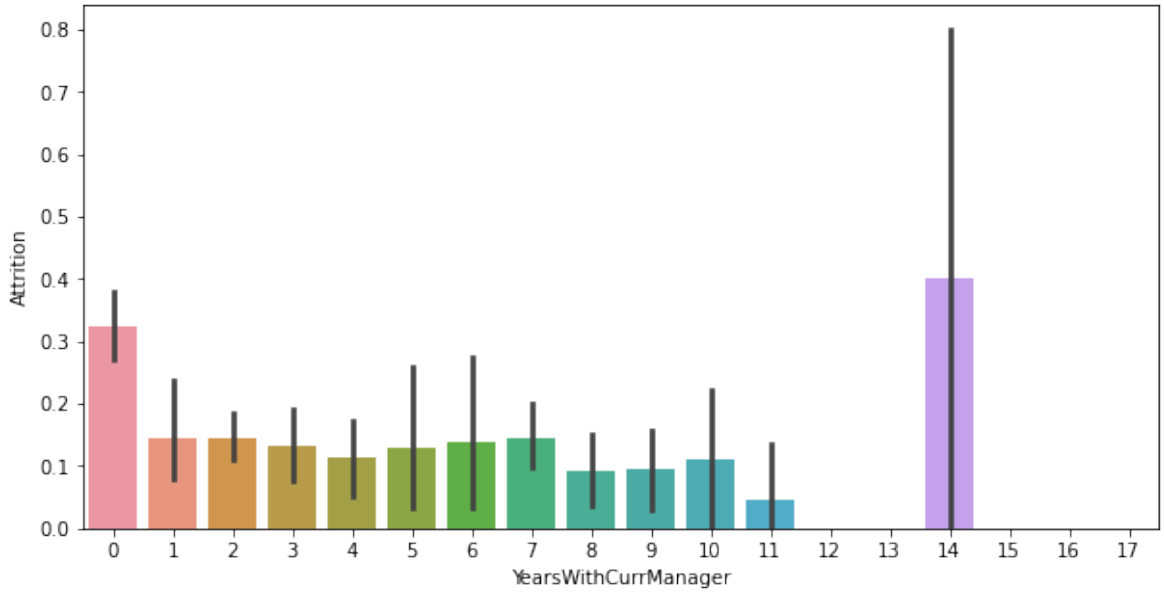
```
<AxesSubplot:xlabel='PercentSalaryHike',
ylabel='Attrition'>
```



Employees who are getting higher hike are leaving more. Here we can say that either these employees are waiting to have good hike and with this they will grab some other opportunity with more percentage of hike by showing recent hike they got from there present organisation. And mostly this happens which really become concern for most of the company.

```
plt.figure(figsize=(10,5))
sns.barplot(x="YearsWithCurrManager",y
='Attrition',data=df)
```

```
<AxesSubplot:xlabel='YearsWithCurrManager',
ylabel='Attrition'>
```



Here also we got a very interesting insight that employee who are under one manager for longer period are leaving more. That means may be they are having issue with their manager or want to to be in some other team so that they can learn more new things which may be possible not getting being in the same team.

Feature Engineering

As by seeing the data we came across that most of the columns were having two to three unique values, so here I am applying replace method to transform those columns which are having such values.

```
# Replace Label columns into binary codes
df['Gender'] =
```

```
df['Gender'].replace({'Male':1,'Female':0})
df['Over18'] = df['Over18'].replace({'Y':1})
df['OverTime']=df['OverTime'].replace({'Yes':1,'No':0
})
df['MaritalStatus']=df['MaritalStatus'].replace({'Sin
gle':1,'Married':2,'Divorced':0})
df['Department']=df['Department'].replace({'Sales':0,
'Research & Development':1,'Human Resources':2})
df['BusinessTravel']=df['BusinessTravel'].replace({'T
ravel_Rarely':1,'Travel_Frequently':2,'Non-
Travel':0})
```

Next will apply LabelEncoder as there are few columns which do have more than four unique values.

```
le = LabelEncoder()
```

```
df1 = le.fit_transform(df['EducationField'])
pd.Series(df1)
df['EducationField']=df1
df2 = le.fit_transform(df['JobRole'])
df['JobRole']=df1
```

```
df.head()
```

	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

0	41	1	1	1102	0	1	2	1	1	1	2	0	94	3	2	1	4	1	5993	19479	8	1	1	11	3	1	80	0	8	0	1	6	4	5
1	49	0	2	279	1	8	1	1	1	2	3	1	61	2	2	1	2	2	51300	24907	1	1	0	23	4	4	80	1	10	3	3	10	7	7
2	37	1	1	1373	1	2	2	4	1	4	4	1	92	2	1	4	3	1	20900	23966	6	1	1	15	3	2	80	0	7	3	3	0	0	0
3	33	0	2	1392	1	3	4	1	1	5	4	0	56	3	1	1	3	2	29009	23159	1	1	1	11	3	3	80	0	8	3	3	8	7	8
4	27	0	1	591	1	2	1	3	1	7	1	1	40	3	1	3	2	2	34688	16632	9	1	0	12	3	4	80	1	6	3	3	2	2	2

All the object has been transformed to integer form. Next we will go to check if any skewness in dataset or not.

```
df.skew()
```

```
Age                         0.413286
Attrition                   1.844366
BusinessTravel              0.082428
DailyRate                  -0.003519
Department                 -0.172231
DistanceFromHome            0.958118
Education                  -0.289681
EducationField              0.550371
EmployeeCount               0.000000
EmployeeNumber              0.016574
EnvironmentSatisfaction    -0.321654
Gender                     -0.408665
```

```
HourlyRate                -0.032311
JobInvolvement            -0.498419
JobLevel                   1.025401
JobRole                    0.550371
JobSatisfaction           -0.329672
MaritalStatus             -0.443615
MonthlyIncome              1.369817
MonthlyRate                0.018578
NumCompaniesWorked         1.026471
Over18                     0.000000
OverTime                   0.964489
PercentSalaryHike          0.821128
PerformanceRating          1.921883
RelationshipSatisfaction  -0.302828
StandardHours              0.000000
StockOptionLevel           0.968980
TotalWorkingYears          1.117172
TrainingTimesLastYear      0.553124
WorkLifeBalance           -0.552480
YearsAtCompany             1.764529
YearsInCurrentRole         0.917363
YearsSinceLastPromotion    1.984290
YearsWithCurrManager       0.833451
```

As skewness can be applied only on continous columns so here we checked whether any column is skewed or not and got the result as shown below:

```
skew_feature =['Age','MonthlyIncome']
```

Skewness found in Age and Monthly income column so will go to treat the same by using np.log method. As we know that skewness are considered for those which doesn't have normal distribution . So just to bring it to the same we apply log or power transformation which ever better shows result for respective dataset.

```
# Removing skewness using Log Transformation
df['Age'] = np.log(df['Age'])
df['MonthlyIncome'] = np.log(df['MonthlyIncome'])
```

After removing skewness rechecked the same using df.skew()

```
df.skew()
```

```
Age                   -0.154166
MonthlyIncome          0.286192
```

As we can see that almost our data are in normal distribution now as skewness range is from +5 to -5 and here both the columns are within the range.

Next will check if any null / missing values within the column by using np.where

```
df_index=np.where(df['Age'].isnull()==True)
new_df = df.loc[df_index]
new_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

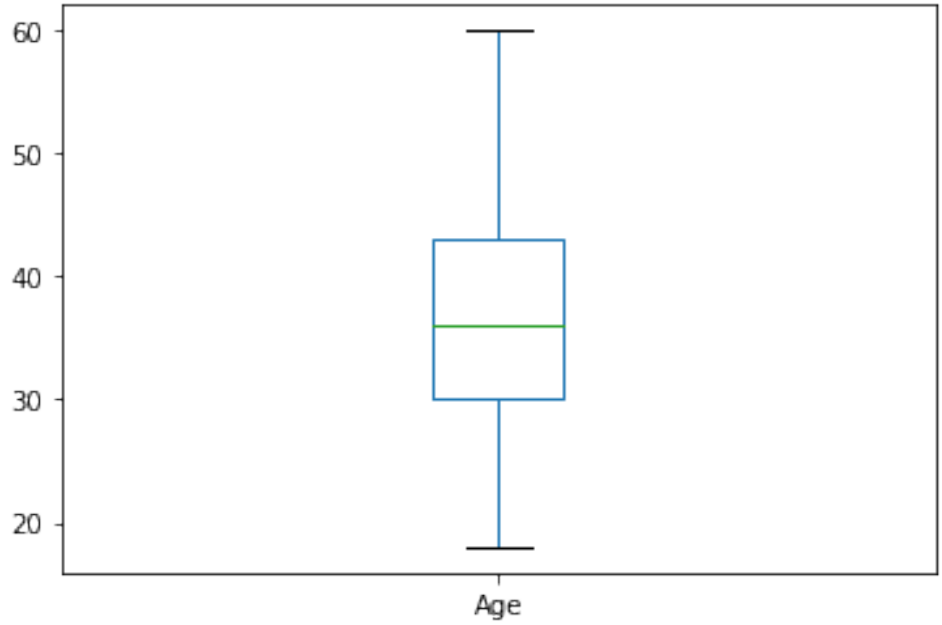
No missing / null values in any of the column.

Let's check if any outliers in any of the continuous column using box plot

Outliers are extreme values that fall a long way outside of the other observations.

```
df.boxplot(column =['Age'], grid = False)
```

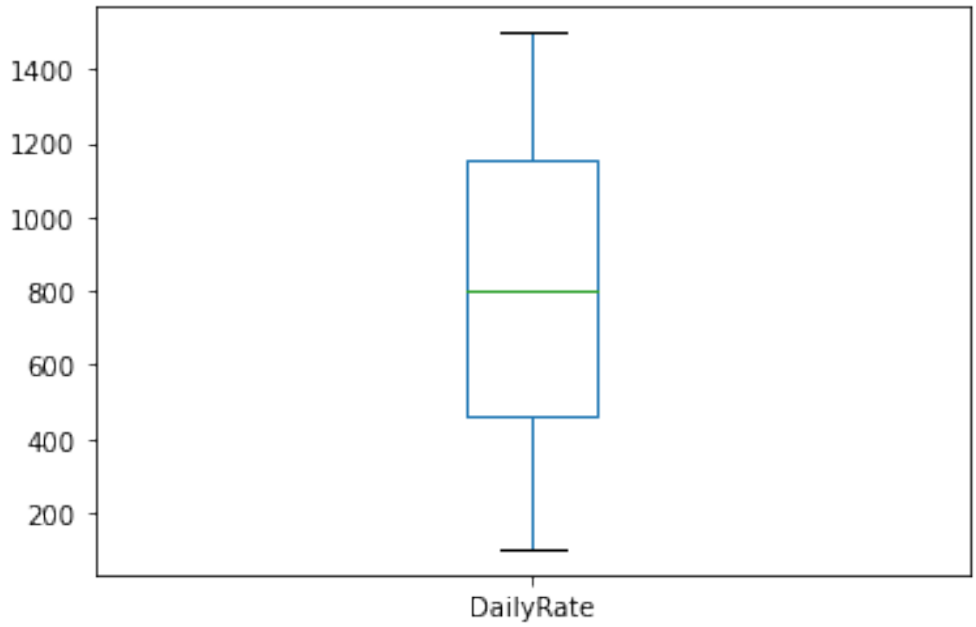
```
<AxesSubplot:>
```



No outliers found in age column

```
df.boxplot(column =['DailyRate'], grid = False)
```

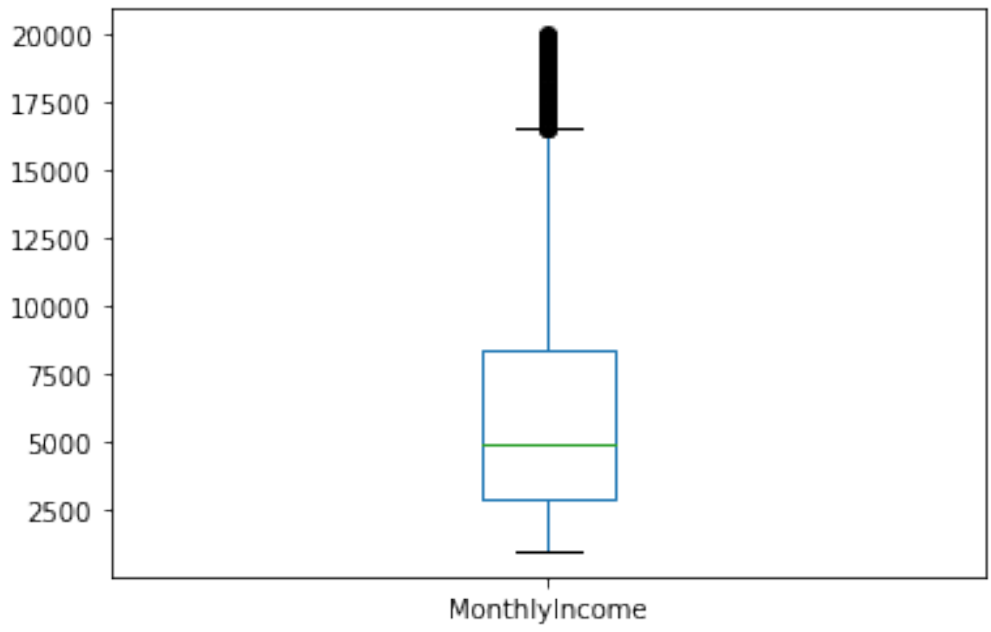
```
<AxesSubplot:>
```



No outliers found in Daily Rate

```
df.boxplot(column =['MonthlyIncome'], grid = False)
```

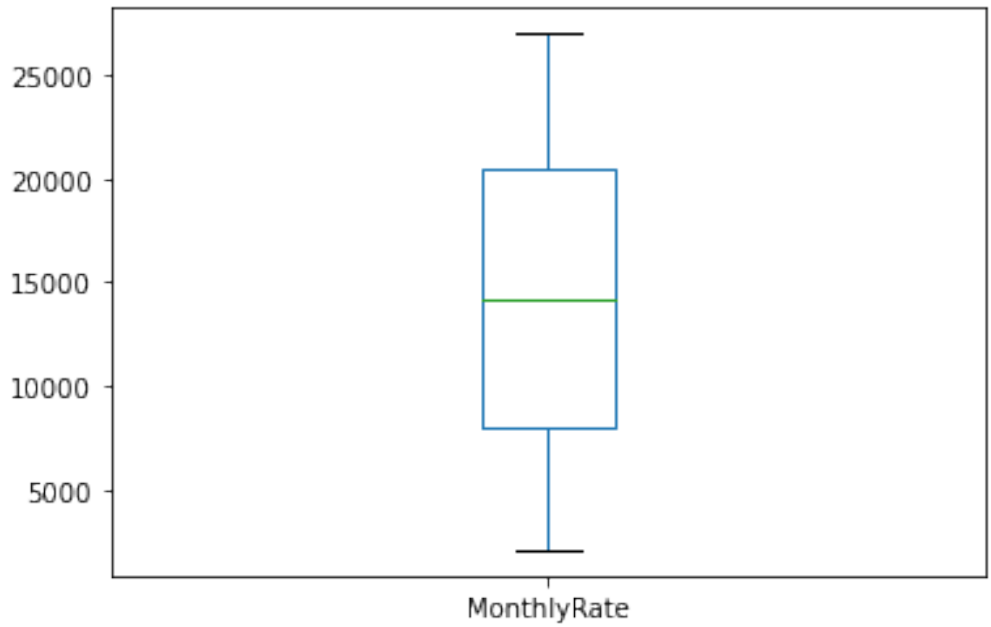
```
<AxesSubplot:>
```



Outliers present in MonthlyIncome which we need to fix

```
df.boxplot(column =['MonthlyRate'], grid = False)
```

```
<AxesSubplot:>
```



No outliers found in Monthly Rate

Now will apply quantile method to remove outliers in monthly income.

Any data point more than 1.5 interquartile ranges (IQRs) below the first quartile or above the third quartile. Low outliers are below Q1 -1.5 .IQR and high outliers are above Q3 + 1.5. IQR

```
# find the IQR (Inter Quantile Range) to identify
ouliers

# 1st Quantile
q1 = df.quantile(0.25)

# 3rd Quantile
q3 = df.quantile(0.75)

#IQR
iqr = q3 - q1
```

```
out_MonthlyIncome = (q3.MonthlyIncome +
(1.5*iqr.MonthlyIncome))
out_MonthlyIncome
```

```
10.619331667493288
```

As in monthly income outliers were on higher side so added Q3 with 1.5.IQR.

Searching for outliers if any in monthly income by using np.where

```
index =
np.where(df['MonthlyIncome']>out_MonthlyIncome)
index
```

```
(array([], dtype=int64),)
```

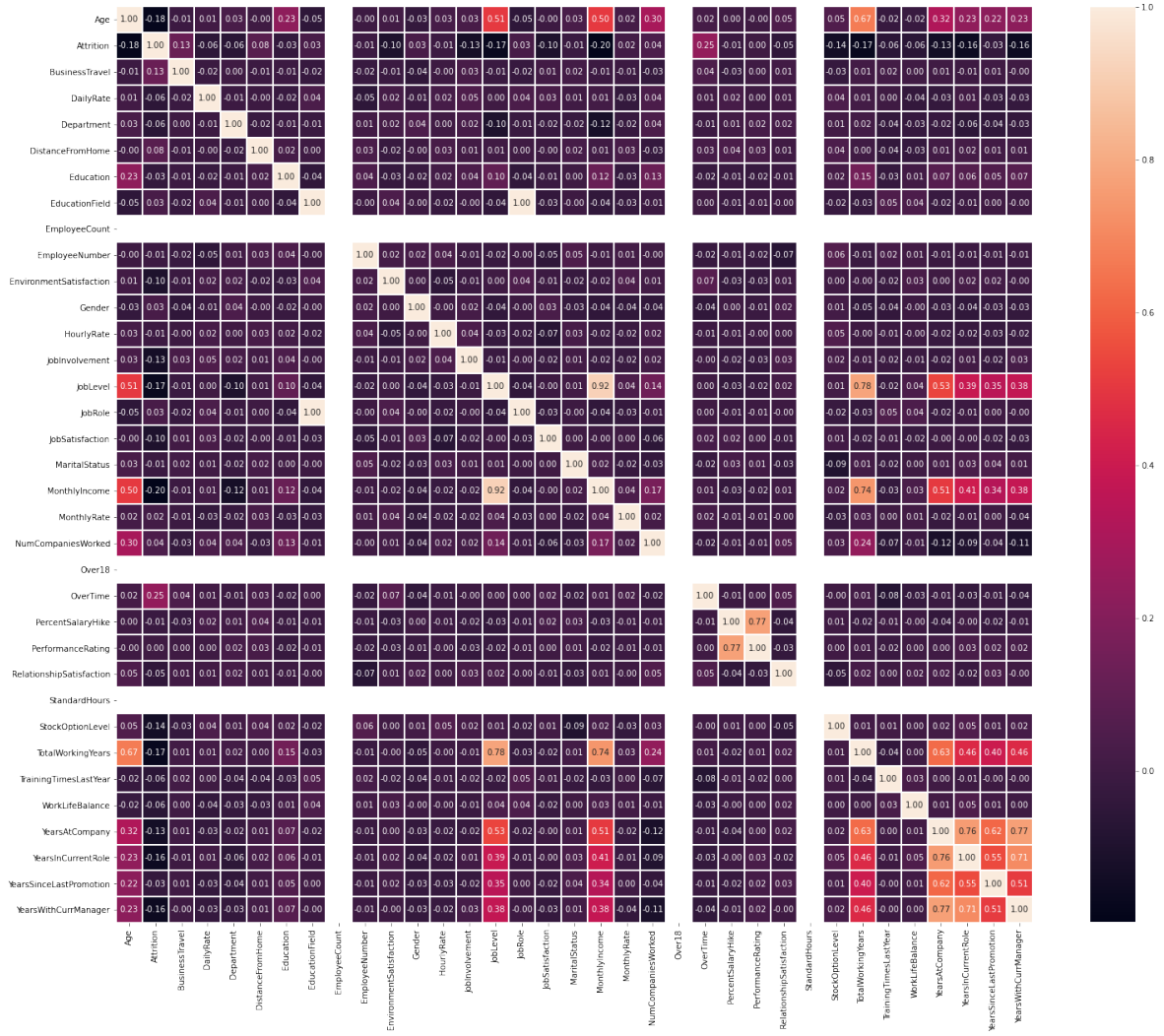
No outliers found

After treating skewness and outliers let's have a look correlation between features. Highly correlated or positive correlation means if one feature increases simultaneously the other features increases too and vice versa for negative correlation. While no correlation means no change in features if any of the other feature increases or decreases. Let's check whether any feature exist in this dataset or not.

```
 # Plotting the heatmap of cprrelation between
features

plt.figure(figsize=(25,25))

sns.heatmap(corr, cbar=True, square=True,
cbar_kws={'shrink':.82},fmt='.2f', annot=True,
annot_kws={'size':10},linewidths= True)
plt.show()
```



Here I used heat map to see the correlation between features to target and as we know that perfect correlation is when it lies between +/-1. There are few features which are having less correlation with target . Still will not going to delete any of the feature as none of the feature are showing correlation with each other so strongly.

I will use here Kselect feature as number of columns are more and want to use best feature for my model building. Even if difference between one feature to the other will not be much then will not use that feature as it may not affect my model performance much. And as we know that use of Kselect feature is to reduce the number of columns without affecting model performance. So we have to be careful while selecting features for training our model.

Preprocessing

```
# Splitting feature and target into x and
respectively
x = df.drop('Attrition',axis = 1)
y = df.Attrition
```

Splitting feature into x variable and target into y variable for model training and prediction purpose.

```
best_features = SelectKBest(score_func = f_classif, k
= 23)
fit = best_features.fit(x,y)
df_score = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(x.columns)
```

Here I selected 23 features for my model building and will check the score of each feature.

```
feature_scores = pd.concat([df_columns,
df_score],axis = 1)
feature_scores.columns = ['Feature_name','score'] #
name output column
print(feature_scores.nlargest(23,'score')) # print 23
best features
```

```
                Feature_name      score
21                  OverTime  94.656457
17             MonthlyIncome  60.032335
0                        Age  49.633511
27         TotalWorkingYears  44.252491
13                  JobLevel  43.215344
31        YearsInCurrentRole  38.838303
33      YearsWithCurrManager  36.712311
26          StockOptionLevel  28.140501
30            YearsAtCompany  27.001624
12            JobInvolvement  25.241985
1             BusinessTravel  24.068022
15           JobSatisfaction  15.890004
9    EnvironmentSatisfaction  15.855209
4            DistanceFromHome   8.968277
3                  Department   6.035877
```

```
29            WorkLifeBalance   6.026116
28     TrainingTimesLastYear    5.211646
2                  DailyRate    4.726640
24  RelationshipSatisfaction    3.095576
19        NumCompaniesWorked    2.782287
32   YearsSinceLastPromotion    1.602218
5                  Education    1.446308
10                    Gender    1.274587
```

As we can see that after DistanceFromHome there is not so much effective features which will help our model to perform much better. So for better performance selected 23 feature. Here best feature we can say is Overtime which is having highest score.

Creating data frame for selected 23 features so that we can train our model on the basis of these features.

```
df_select =
df[['OverTime','MonthlyIncome','Age','TotalWorkingYea
rs','JobLevel','YearsInCurrentRole','YearsWithCurrMan
ager',

'StockOptionLevel','YearsAtCompany','JobInvolvement',
'BusinessTravel','JobSatisfaction','EnvironmentSatisf
action',

'DistanceFromHome','Department','WorkLifeBalance','Tr
ainingTimesLastYear','DailyRate','RelationshipSatisfa
ction',

'NumCompaniesWorked','YearsSinceLastPromotion','Educa
tion','Gender']]
```

Scaling Feature

Scaling features so that all the features will be on same unit and our model can understand each feature well.

```
scaler = StandardScaler()
x_scaler = scaler.fit_transform(df_select)
```

Finding best Random State and Model Training

```
maxscore = 0
maxrs = 0

for i in range(1,1000):
    x_train,x_test,y_train,y_test =
train_test_split(x_scaler,y,test_size =
0.30,random_state = i)
    knn = KNeighborsClassifier()
    knn.fit(x_train,y_train)
    pred = knn.predict(x_test)
    rsc = accuracy_score(y_test,pred)
    if rsc>maxscore:
        maxscore=rsc
        maxrs=i
print("Best accuracy score is:",maxscore,"On Random
state: ",maxrs)
```

```
Best accuracy score is: 0.891156462585034 On Random
state:  631
```

So here we are getting accuracy score on 631 random state.
Random State are actually selecting numbers randomly and ensures that the random numbers are generated in the same order.

Training the model

```
x_train,x_test,y_train,y_test =
train_test_split(x_scaler,y,test_size = 0.25,
random_state = i)
```

Distributed 75% and 25% respectively among training and testing data. So our model will be trained on 75% data and will understand the pattern after that will predict on the basis of 25% data, which the model understood during training period.

```
knn = KNeighborsClassifier()
```

```
knn.fit(x_train,y_train)
y_pred = knn.predict(x_test)
```

Here one of the model I used is KNeighborsClassifier for training and prediction. And the matrix used is accuracy score as this dataset is a classification dataset and in classification we usually use accuracy score, Classification report which are precision, Recall, F1-score (Combination of both Precision and Recall).

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

```
array([[304,   8],
       [ 50,   6]])
```

Here True positive result are really good as we can see total no of true positive that means the prediction we assumed are almost giving 304, even false negative result we can see is very low which is only 6. So we can say that our model has understood the data really well.

Let's see the classification report for precision, recall and f1score along with accuracy score.

```
print(classification_report(y_test,y_pred,digits =
2))
```

```
              precision    recall  f1-score   support

           0       0.86      0.97      0.91       312
           1       0.43      0.11      0.17        56

    accuracy                           0.84       368
   macro avg       0.64      0.54      0.54       368
weighted avg       0.79      0.84      0.80       368
```

```
K_f = KFold(n_splits = 3, shuffle = True)
K_f
```

```
KFold(n_splits=3, random_state=None, shuffle=True)
```

```
for train, test in K_f.split([1,2,3,4,5,6,7,8,9,10]):
```

```
    print ('train:', train, 'test:', test)
```

```
cross_val_score(KNeighborsClassifier(),
x_scaler,y,cv=5).mean()
```

```
0.846938775510204
```

Here recall is having high score compare to precision and as we know that trade off is one of the concern when such score we get because F1 score consider those which are having higher score. And as a result we can see that F1 score is 91% which is the mean of Precision and Recall but we can't trust F1 Score due to it's bias nature. Our matrix performance is also pretty good which is 84%. But here difference between cross validation score and accuracy score are almost negligible . Let's see other model performance and will decide that for this dataset which model performance is best.

```
log_reg = LogisticRegression()
log_reg.fit(x_train, y_train)
y_pred = log_reg.predict(x_test)
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

```
0.8668478260869565
```

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

```
array([[297,  15],
       [ 34,  22]])
```

```
cross_val_score(log_reg, x_scaler,y,cv=5).mean()
```

```
0.880952380952381
```

Here we can see that Logistic regression score are better than KNeighborsClassifier even Cross Validation score we can see has been increased. Here I used Logistic regression because our target variable is a binary based dataset and for binary based dataset Logistic Regression

are one the best model we can say. And logistic Regression though Regression is added in the last name but usually we use this for classification based problems.
Here also True positive and false negative result are better as having low error in false negative and True positive assumption also shows pretty high which means what assumption we did was true for our dataset shows positive result.

```
rf = RandomForestClassifier()
```

```
rf.fit(x_train, y_train)
y_pred = rf.predict(x_test)
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

```
0.8641304347826086
```

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

```
array([[309,   3],
       [ 47,   9]])
```

```
cross_val_score(rf, x_scaler,y,cv=5).mean()
```

```
0.8585034013605443
```

Model performance of RandomForestClassifier is also showing good result as the difference between CV score and accuracy score are pretty less which is near about 1%

```
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
y_pred = dt.predict(x_test)
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

```
0.7934782608695652
```

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

```
conf_mat
```

```
array([[269,  43],
       [ 33,  23]])
```

```
cross_val_score(rf, x_scaler,y,cv=5).mean()
```

```
0.8578231292517007
```

Here CV score is much better than accuracy score , But the difference between two are pretty high so we can't tune on this model.

Hyperparameter Tuning

So on the basis of all the scores I will be going to do hyper parameter tuning on RandomForest as the accuracy score of both logistic and random are almost same but difference between CV score is less for Random Forest.

```
# RandomForestClassifier
param = {'n_estimators':[100,500,700,900],
        'criterion':['gini','entropy'],
        'max_depth':[10,20,30,40,50],
        'max_features':['auto','sqrt','log2'],
        'class_weight':
['balanced','balanced_subsample']}
```

Choosing parameters for our model to get trained so that we can get some improvement in our score if possible.

```
GC = GridSearchCV(rf,param,cv=5)
```

Passing the model for which we want to do Grid Search CV along with parameters we choose and CV score.

```
GC.fit(x_train,y_train)
```

```
GridSearchCV(cv=5,
estimator=RandomForestClassifier(),
             param_grid={'class_weight': ['balanced',
```

```
'balanced_subsample'],
                         'criterion': ['gini',
'entropy'],
                         'max_depth': [10, 20, 30,
40, 50],
                         'max_features': ['auto',
'sqrt', 'log2'],
                         'n_estimators': [100, 500,
700, 900]})
```

Training our model

```
GC.best_params_
```

```
{'class_weight': 'balanced_subsample',
 'criterion': 'entropy',
 'max_depth': 40,
 'max_features': 'log2',
 'n_estimators': 900}
```

Best Parameters which Gridsearch CV passed after going through the dataset are mentioned above. As we know that Grid-search is used to find the optimal hyperparameters of a model which results in the most 'accurate' predictions.

```
final_rf =
RandomForestClassifier(class_weight='balanced_subsamp
le',criterion= 'entropy',max_depth = 40, max_features
= 'log2',n_estimators = 900)
final_rf.fit(x_train,y_train)
pred = final_rf.predict(x_test)
acc = accuracy_score(y_test,pred)
print(acc*100)
88.1217829468487
```

Final score after doing GridSearchCV we got as 88% approx.2% improvement in our model performance we can see after doing hyper parameter Tuning.

Next will save the model using Joblib so that if want to load in future for any changes or improvement we

can do the same.

Save the Model

```
import joblib
joblib.dump(final_log,'FinalmodelAttrition.pk1')
```

Summary

This data is about whether the attrition rate is high or not and if high than in what percentage. So we started with importing all the important libraries required from EDA to preprocessing to model building and at last predicting on the basis of which we trained our model. Next imported Dataset which consist of 1470 rows and 35 columns. So we are having total 34 features and 1 target column which is Attrition and this column consist of either yes or no that means our dataset is a classification based problem and also is a categorical column so if a dataset is a categorical column then we have to check whether the data is a balanced data or imbalance so here this dataset is not an imbalanced dataset so no need to treat anything. Next comes EDA part with the help of which we can able to get the insight of our data. So first I checked what are the columns having object and what are in integer form with the help of info method , where we got total 9 object and 26 integer columns. Even we can say that till now the total counting shows 1470 for all the columns which was our total no of rows that means no missing values found till here. After this checked static data with the help of describe method and got no missing values even in continuous columns too , as describe method only consider continuous columns not categorical. At last to reconfirm if any missing values in categorical column or not used isna method , there also not found any missing values that means no treatment of missing values in this dataset are required. For more better clarity of data used visualisation technique using seaborn and matplotlib. Here the insight we got was really interesting and helpful for an HR to understand why the attrition happens and where they can curb this by taking necessary steps required. 1st concern was age group as we seen above that mostly young age group employees were moving to some other opportunity. Here we can have an assumption that either they are not having much responsibilities so loosing of job is not a concern for them and other one we can say is hunting for better opportunity may be due to not happy either with perk or salary they are getting which may be

a major concern or the job responsibility they are not happy with. Other concern was distance from home to workplace which was also one of the reason for more attrition as we seen above that those who are staying far leaving frequently. Even environment satisfaction was less which HR or concerned team should look after and also one of the major reason was working with same manager for a longer period. These are the concern which a company should look and try to reduce these loopholes so that they can curb the attrition rate and can retain talented employees to stick to their company. After getting the insight of dataset did feature engineering as having 9 object / string type column. Which our machine algorithm will not able to understand. So here I used replace and LabelEncoder to transform all the object to integer form. After this checked the skewness and outliers and treat the same using np.log and quantile method respectively. Once this done checked the correlation between feature and target where found may be there's a correlation between them but not deleted any of the column as I used here kbest selection feature where we can select best features to train our model.

Splitted feature and target variable respectively into x and y. Next did scaling so that all the features comes to same scale. Once scaling done then trained the model after finding best random state which was 631 here. Next used 4 models here to see which model is the best preforming model and predicting % are higher means which model understood my data well. So here I used

1. KNeighbors Classifier
2. Logistic Regression
3. RandomForestClassifier
4. DecisionTreeClassifier

And after validating all got RandomForest Classifier as a best model because difference between CV score and accuracy score for this model was less compare to others. Done hyperparameter tuning for RandomForestClassifier so that we can improve our score little more. Where I used Gridsearch CV and the score which I got after training and prediction was 87% which shown 2% improvement in model performance. Atlas saved the model using Joblib so that in future if require to change or include anything we can load and work on the same.

So from this I can say that there were more scope to have much better score than what we got by including more parameters to grid search cv or by doing more preprocessing either by checking if any feature can be removed or by doing some more EDA any other improvement could

have been done .