HR Analytics Project

NAME: NITIN SINGH TATRARI

Batch: 1825

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.

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

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

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

Thus, in this study we will find out:

1) The key parameters of an employee on which attrition can be done
2) The strategies that can be applied to improve the employee retention and finding out the attrition in advance.

DATA ANALYSIC:

The data provided have 13 features and 1 target. All parameters are as follows:-

1) Age: Age of employee
2) Business travel: Upon on frequency of travel it is divided into three categories -:
    a) Travel rarely
    b) Travel frequently
    c) Non-travel
3) Daily Rate: the amount of money paid per day
4) Department: Three departments are present in the data. They are:-
    a) Research & Development
    b) Sales
    c) Human resources
5) Distance from Home: Home distance of employee from work place.
6) Education: The Education is divided into five categories (1, 2, 3, 4, 5). 'Below College' 2 'College' 3 'Bachelor' 4 'Master' 5 'Doctor'
    a) 1: 'Below College'
    b) 2: 'College'
    c) 3: 'Bachelor'
    d) 4: 'Master'
    e) 5: 'Doctorate'
7) Education Field: It is divided into sex categories:
    a) Life science
    b) Medical
    c) Marketing
    d) Technical Degree
    e) Human resources
    f) Other
8) Employee count
9) Employee Number: ID for employee
10) Environment Satisfaction: Numerical Value - SATISFACTION WITH THE JOB (1, 2, 3, 4), where 4 is highest satisfaction.
11) Gender: Male & Female
12) Hourly Rate: Numerical Value - HOURLY SALARY
13) OVER 18: (1=YES, 2=NO)
14) OVERTIME: (1=NO, 2=YES)
15) Job involvement: Numerical Value - JOB INVOLVEMENT (1, 2, 3, 4), where 4 is the highest involvement.
16) Job level: Numerical Value - LEVEL OF JOB (1, 2, 3, 4, 5), where 5 is the highest level of job.
17) Job Role: There are total 9 job role given. They are :
    a) Sales Executive
    b) Research Scientist
    c) Laboratory Technician
    d) Manufacturing Director
    e) Healthcare Representative

f) Manager
g) Sales Representative
h) Research Director
i) Human Resources

18) Job satisfaction: Numerical Value - SATISFACTION WITH THE JOB (1, 2, 3, 4) , where 4 is the highest job satisfaction level.
19) Marital status: It is divided into Three categories :
    a) Married
    b) Single
    c) Divorced
20) Monthly income: Numerical Value - MONTHLY SALARY
21) MONTHY RATE: Numerical Value - MONTHY RATE
22) NumCompaniesWorked: Numerical Value - NO. OF COMPANIES WORKED AT (0, 1, 2, 3, 4, 5, 6, 7, 8)
23) Percent Salary Hike: Numerical Value - PERCENTAGE INCREASE IN SALARY
24) Performance Rating: Numerical Value - PERFORMANCE RATING (1, 2, 3, 4)
25) Relationship Satisfaction: Numerical Value - RELATIONS SATISFACTION (1, 2, 3, 4)
26) Standard Hours: Numerical Value - STANDARD HOURS
27) Stock Option Level: Numerical Value - STOCK OPTIONS (0, 1, 2, 3)
28) TOTAL WORKING YEARS: Numerical Value - TOTAL YEARS WORKED
29) Training Times Last Year: Numerical Value - HOURS SPENT TRAINING
30) Work Life Balance: TIME SPENT BEWTWEEN WORK AND OUTSIDE. They are divided into 4 categories (1, 2, 3, 4)
31) Years at company: Numerical Value - TOTAL NUMBER OF YEARS AT THE COMPNAY
32) Years in current role: Numerical Value -YEARS IN CURRENT ROLE
33) Years Since Last Promotion: Numerical Value - LAST PROMOTION
34) Years with current manager: Numerical Value - YEARS SPENT WITH CURRENT MANAGER

Thus, we have understood all the parameter above.

1. Importing Libraries and dataset:

Firstly, we import libraries- pandas, numpy, seaborn & matplotlib.

Through pandas we import data “HRdata.csv”.

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

```
data=pd.read_csv('HRdata.csv')
data
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Relations
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	...	

2. Checking the null values in the data Set.

```
1  data.isnull().sum()
```

```
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
```

There is no null value in the dataset

3. <u>Understanding Data</u>

A) Finding unique values and their counts

Firstly, we load the data into Data Frame 'df'. Then we check all unique values and their counts in all attributes.

```
1  df=pd.DataFrame(data)
```

```
1  for i in df.columns:
2      print(i)
3      print(df[i].value_counts())
4      print('\n')
```

```
Age
35    78
34    77
31    69
36    69
29    68
32    61
30    60
33    58
38    58
40    57
37    50
27    48
28    48
42    46
39    42
45    41
41    40
26    39
46    33
44    33
43    32
50    30
24    26
25    26
47    24
49    24
55    22
48    19
51    19
53    19
52    18
54    18
22    16
56    14
58    14
23    14
21    13
20    11
59    10
19     9
18     8
60     5
57     4
Name: Age, dtype: int64
```

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

```
BusinessTravel
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
```

```
DailyRate
691     6
1082    5
329     5
1329    5
530     5
       ..
897     1
891     1
889     1
888     1
102     1
Name: DailyRate, Length: 886, dtype: int64
```

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

```
DistanceFromHome
2     211
1     208
10     86
9      85
3      84
7      84
8      80
5      65
4      64
6      59
16     32
11     29
24     28
29     27
23     27
18     26
15     26
20     25
25     25
26     25
28     23
19     22
14     21
12     20
17     20
13     19
22     19
21     18
27     12
Name: DistanceFromHome, dtype: int64
```

```
Education
3    572
4    398
2    282
1    170
5     48
Name: Education, dtype: int64
```

```
EducationField
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
```

```
JobLevel
1    543
2    534
3    218
4    106
5     69
Name: JobLevel, dtype: int64
```

```
JobRole
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64
```

```
JobSatisfaction
4    459
3    442
1    289
2    280
Name: JobSatisfaction, dtype: int64
```

```
MaritalStatus
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
```

```
MonthlyIncome
2342     4
5562     3
2741     3
2451     3
2610     3
       ..
5381     1
13577    1
12965    1
3339     1
14336    1
Name: MonthlyIncome, Length: 1349, dtype: int64
```

```
EmployeeCount
1    1470
Name: EmployeeCount, dtype: int64
```

```
EmployeeNumber
2046    1
641     1
644     1
645     1
647     1
       ..
1364    1
1367    1
1368    1
1369    1
2048    1
Name: EmployeeNumber, Length: 1470, dtype: int64
```

```
EnvironmentSatisfaction
3    453
4    446
2    287
1    284
Name: EnvironmentSatisfaction, dtype: int64
```

```
Gender
Male      882
Female    588
Name: Gender, dtype: int64
```

```
HourlyRate
66    29
42    28
98    28
48    28
84    28
      ..
31    15
68    14
53    14
38    13
34    12
Name: HourlyRate, Length: 71, dtype: int64
```

```
JobInvolvement
3    868
2    375
4    144
1     83
Name: JobInvolvement, dtype: int64
```

```
MonthlyRate
4223     3
9150     3
6670     2
7324     2
4658     2
        ..
11585    1
15682    1
3395     1
9541     1
8192     1
Name: MonthlyRate, Length: 1427, dtype: int64
```

```
NumCompaniesWorked
1    521
0    197
3    159
2    146
4    139
7     74
6     70
5     63
9     52
8     49
Name: NumCompaniesWorked, dtype: int64
```

```
Over18
Y    1470
Name: Over18, dtype: int64
```

```
OverTime
No     1054
Yes     416
Name: OverTime, dtype: int64
```

```
PercentSalaryHike
11    210
13    209
14    201
12    198
15    101
18     89
17     82
16     78
19     76
22     56
20     55
21     48
23     28
24     21
25     18
Name: PercentSalaryHike, dtype: int64
```

```
PerformanceRating
3    1244
4     226
Name: PerformanceRating, dtype: int64
```

```
RelationshipSatisfaction
3    459
4    432
2    303
1    276
Name: RelationshipSatisfaction, dtype: int64
```

```
StandardHours
80    1470
Name: StandardHours, dtype: int64
```

```
StockOptionLevel
0    631
1    596
2    158
3     85
Name: StockOptionLevel, dtype: int64
```

```
TotalWorkingYears
10    202
6     125
8     103
9      96
5      88
1      81
7      81
4      63
12     48
3      42
15     40
16     37
13     36
11     36
21     34
17     33
14     31
2      31
20     30
18     27
19     22
23     22
22     21
24     18
25     14
28     14
26     14
0      11
29     10
31      9
32      9
27      7
30      7
33      7
36      6
34      5
37      4
35      3
40      2
38      1
Name: TotalWorkingYears, dtype: int64
```

```
TrainingTimesLastYear
2    547
3    491
4    123
5    119
1     71
6     65
0     54
Name: TrainingTimesLastYear, dtype: int64
```

```
WorkLifeBalance
3    893
2    344
4    153
1     80
Name: WorkLifeBalance, dtype: int64
```

```
YearsAtCompany
5     196
1     171
3     128
2     127
10    120
4     110
7      90
9      82
8      80
6      76
0      44
11     32
20     27
13     24
15     20
14     18
22     15
12     14
21     14
18     13
16     12
19     11
17      9
24      6
33      5
25      4
26      4
31      3
32      3
36      2
27      2
29      2
23      2
30      1
34      1
37      1
40      1
Name: YearsAtCompany, dtype: int64
```

```
YearsInCurrentRole
2     372
0     244
7     222
3     135
4     104
8      89
9      67
1      57
6      37
5      36
10     29
11     22
13     14
14     11
12     10
15      8
16      7
17      4
18      2
Name: YearsInCurrentRole, dtype: int64
```

```
YearsSinceLastPromotion
0     581
1     357
2     159
7      76
4      61
3      52
5      45
6      32
11     24
8      18
9      17
15     13
13     10
12     10
14      9
10      6
Name: YearsSinceLastPromotion, dtype: int64
```

```
YearsWithCurrManager
2     344
0     263
7     216
3     142
8     107
4      98
1      76
9      64
5      31
6      29
10     27
11     22
12     18
13     14
17      7
14      5
15      5
16      2
Name: YearsWithCurrManager, dtype: int64
```

The following points we understand:-

a) Over18, Employee Count and Standard Hours have one unique value only, thus, we can drop it.
b) All employees have unique IDs, so we can also drop “EmployeeNumber’ column.

```
df.drop({'EmployeeCount','EmployeeNumber','Over18','StandardHours'},axis=1, inplace=True)
df.head()
```

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

5 rows × 31 columns

Thus, we have drop 'EmployeeCount','EmployeeNumber','Over18','StandardHours' attributes. Now, there are 30 attributes and 1 target variable.

B) Finding data types

We check the data types of all the columns.

```
df.dtypes
```

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

There are 23 string type columns and 8 integer type columns

There are 23 string type columns and 8 integer type columns

C) Transforming target variables

Now, we convert attrition values from (‘yes’ & ‘No’) to (1, 0).

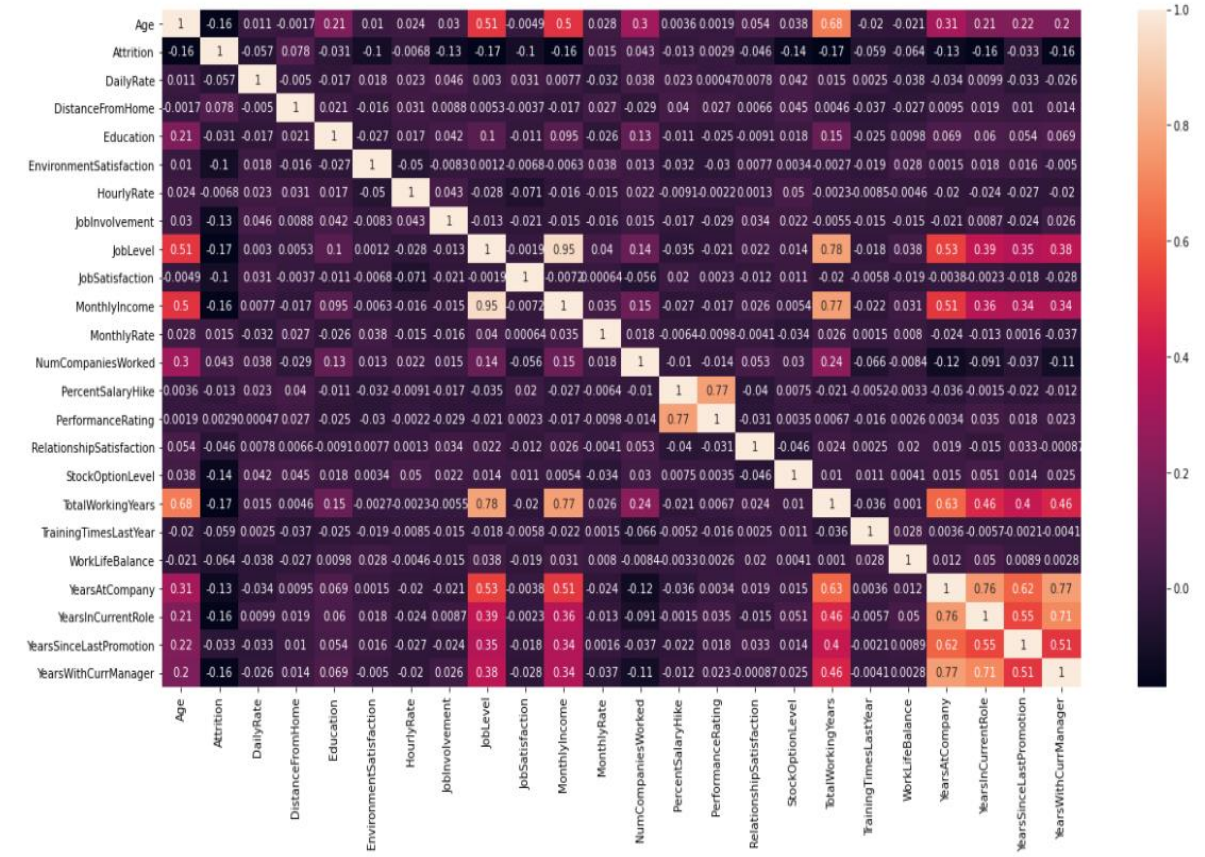
```
from sklearn import preprocessing
le=preprocessing.LabelEncoder()
df.Attrition = le.fit_transform(df.Attrition)
df
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...	Performanc
0	41	1	Travel_Rarely	1102	Sales	1	2	Life Sciences	2	Female	...	
1	49	0	Travel_Frequently	279	Research & Development	8	1	Life Sciences	3	Male	...	
2	37	1	Travel_Rarely	1373	Research & Development	2	2	Other	4	Male	...	
3	33	0	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	4	Female	...	
4	27	0	Travel_Rarely	591	Research & Development	2	1	Medical	1	Male	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1465	36	0	Travel_Frequently	884	Research & Development	23	2	Medical	3	Male	...	
1466	39	0	Travel_Rarely	613	Research & Development	6	1	Medical	4	Male	...	
1467	27	0	Travel_Rarely	155	Research & Development	4	3	Life Sciences	2	Male	...	
1468	49	0	Travel_Frequently	1023	Sales	2	3	Medical	4	Male	...	
1469	34	0	Travel_Rarely	628	Research & Development	8	3	Medical	2	Male	...	

1470 rows × 31 columns

D) Studying Correlation of target with other variables.

```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



- Attributes - 'Age', 'EnvironmentSatisfaction', 'JobInvolvement', 'JobLevel', 'JobSatisfaction', 'MonthlyIncome' ,'StockOptionLevel' ,'TotalWorkingYears', 'YearsAtCompany', 'YearsInCurrentRole' ,'YearsWithCurrManager' have considerable correlation with 'Attrition'. Thus, we will copy this column in new data frame ‘df_num’.

```
df_num=df[['Attrition','Age','EnvironmentSatisfaction','JobInvolvement','JobLevel','JobSatisfaction','MonthlyIncome','StockO
```

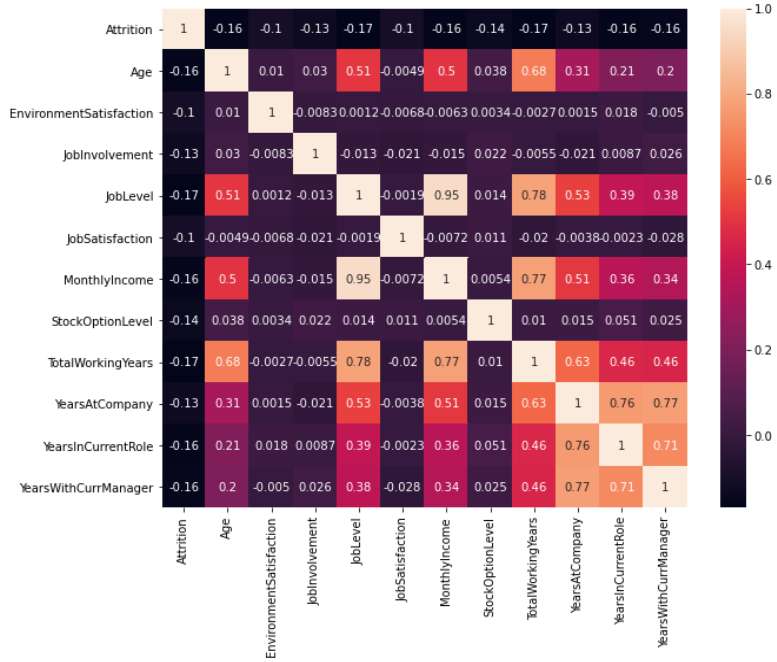
```
df_num
```

	Attrition	Age	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	StockOptionLevel	TotalWorkingYears	YearsAtCompar
0	1	41	2	3	2	4	5993	0	8	
1	0	49	3	2	2	2	5130	1	10	
2	1	37	4	2	1	3	2090	0	7	
3	0	33	4	3	1	3	2909	0	8	
4	0	27	1	3	1	2	3468	1	6	
...	...	...	...	...	...	...	...	...	...	
1465	0	36	3	4	2	4	2571	1	17	
1466	0	39	4	2	3	1	9991	1	9	
1467	0	27	2	4	2	2	6142	1	6	
1468	0	49	4	2	2	2	5390	0	17	
1469	0	34	2	4	2	3	4404	0	6	

1470 rows × 12 columns

We now check correlation of target with other variables in new data Frame ‘df-num’

```
plt.figure(figsize=(10,8))
sns.heatmap(df_num.corr(),annot=True)
plt.show()
```



'YearATCompany' have good correlation with 'YearsInCurrentRole' & 'YearsWithCurrManager' and lowest correlation with 'Attrition' as compare to both of them. So we can drop it. 'MonthlyIncome'& 'JobLevel' are highly correlated, thus we can drop any one.

- 'YearATCompany' have good correlation with 'YearsInCurrentRole' & 'YearsWithCurrManager' and lowest correlation with 'Attrition' as compare to both of them. So we can drop it. 'MonthlyIncome'& 'JobLevel' has high correlation with each other, thus we can drop any one.

```
df_num.drop({'MonthlyIncome','YearsAtCompany'},axis=1,inplace=True)
df_num.head()
```

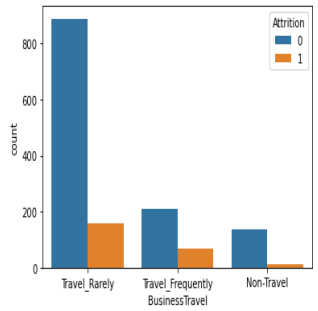
	Attrition	Age	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	StockOptionLevel	TotalWorkingYears	YearsInCurrentRole	YearsWithCurr
0	1	41	2	3	2	4	0	8	4	
1	0	49	3	2	2	2	1	10	7	
2	1	37	4	2	1	3	0	7	0	
3	0	33	4	3	1	3	0	8	7	
4	0	27	1	3	1	2	1	6	2	

Thus, we have drop 'MonthlyIncome' & 'YearsAtCompany'.

4. EDA

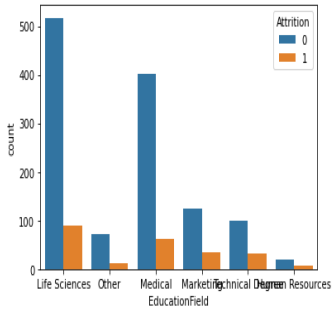
Now, we try to understand graphically relationship of Target variable with other variables.

```
sns.countplot(x='BusinessTravel',hue='Attrition',data=df)
plt.show()
```



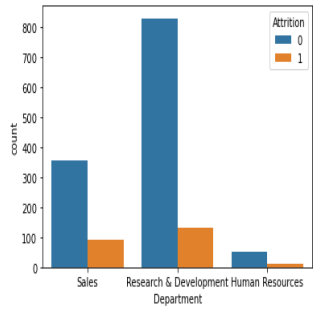
here, we observe employee who travel_frequently leaves the job comparatively than who travel rarely or not travel.

```
sns.countplot(x='EducationField',hue='Attrition',data=df)
plt.show()
```



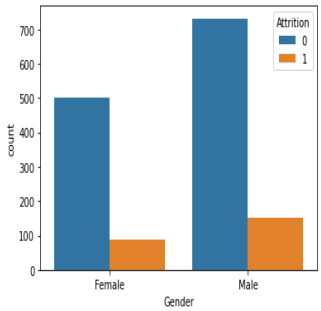
We observe employee with Marketing and Technical background have higher attrition rate.

```
sns.countplot(x='Department',hue='Attrition',data=df)
plt.show()
```



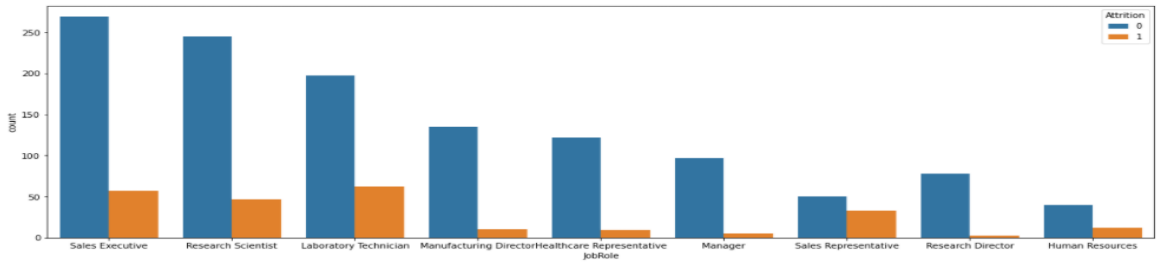
Here, we observe employee from sales Department have higher attrition ratio.

```
sns.countplot(x='Gender',hue='Attrition',data=df)
plt.show()
```



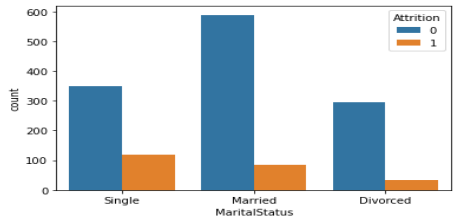
we observe Male have comparatively higher attrition rate than female. But the difference is small around 2-4%

```
plt.figure(figsize=(20,6))
sns.countplot(x='JobRole',hue='Attrition',data=df)
plt.show()
```



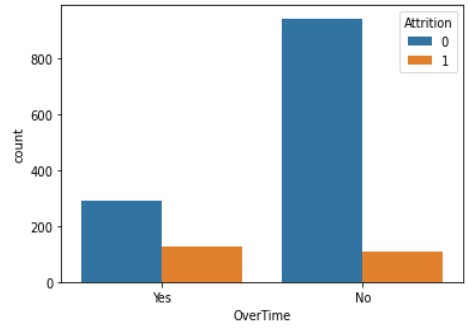
here. we observe employees with Laboratory Technician, Sale representative & Human resouse job role have higher attrition rate

```
sns.countplot(x='MaritalStatus',hue='Attrition',data=df)
plt.show()
```



We observe single employees have higher attrition rate.

```
sns.countplot(x='OverTime',hue='Attrition',data=df)
plt.show()
```



Employees working overtime have higher attrition rate

```
df_cat=df[['BusinessTravel', 'Department','EducationField','Gender', 'JobRole' , 'MaritalStatus','OverTime']].copy()
df_cat.head()
```

	BusinessTravel	Department	EducationField	Gender	JobRole	MaritalStatus	OverTime
0	Travel_Rarely	Sales	Life Sciences	Female	Sales Executive	Single	Yes
1	Travel_Frequently	Research & Development	Life Sciences	Male	Research Scientist	Married	No
2	Travel_Rarely	Research & Development	Other	Male	Laboratory Technician	Single	Yes
3	Travel_Frequently	Research & Development	Life Sciences	Female	Research Scientist	Married	Yes
4	Travel_Rarely	Research & Development	Medical	Male	Laboratory Technician	Married	No

A) The following points have been observed from the above graphs:-

1) We observe that employees, who travel frequently in job, have higher attrition rates than employees than who travel rarely or not travel.

2) We observe employees from sales Department have higher attrition ratio.

3) We observe employees with Marketing and Technical background have higher attrition rate.

4) We observe Male employees have comparatively higher attrition rate than female employees. But the difference is small around 2-4%.

5) We observe employees with Laboratory Technician, Sale representative & Human recourse job role have higher attrition rate.

6) We observe single employees have higher attrition rate.

7) Employees working overtime have higher attrition rate.

Thus we new form a new data frame ’df_cat’ with columns 'BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', 'OverTime' having non-numercial values.

B) Applying get_dummies function on df_cat Data Frame.

```
df_cat=pd.get_dummies(df_cat)
df_cat
```

	BusinessTravel_Non-Travel	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	Department_Human Resources	Department_Research & Development	Department_Sales	Educ
0	0	0	1	0	0	1	
1	0	1	0	0	1	0	
2	0	0	1	0	1	0	
3	0	1	0	0	1	0	
4	0	0	1	0	1	0	
...	...	...	...	...	...	...	
1465	0	1	0	0	1	0	
1466	0	0	1	0	1	0	
1467	0	0	1	0	1	0	
1468	0	1	0	0	0	1	
1469	0	0	1	0	1	0	

1470 rows × 28 columns

C) Joining df_num & df_cat Data fame into df_hr.

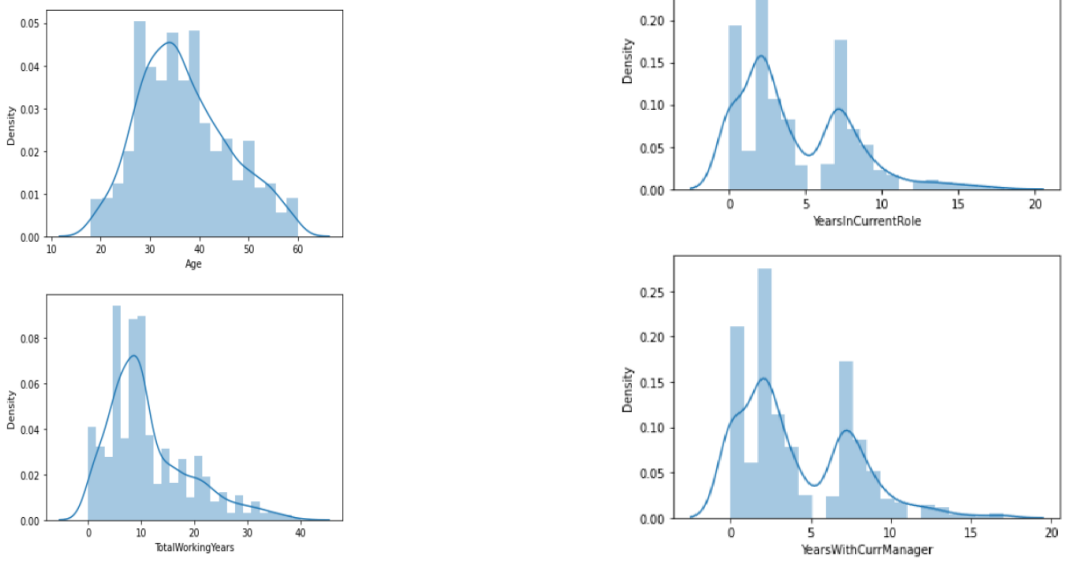
```
df_hr=pd.concat([df_num,df_cat],axis=1)
df_hr
```

	Attrition	Age	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	StockOptionLevel	TotalWorkingYears	YearsInCurrentRole	YearsWithC
0	1	41	2	3	2	4	0	8	4	
1	0	49	3	2	2	2	1	10	7	
2	1	37	4	2	1	3	0	7	0	
3	0	33	4	3	1	3	0	8	7	
4	0	27	1	3	1	2	1	6	2	
...	...	...	...	...	...	...	...	...	...	
1465	0	36	3	4	2	4	1	17	2	
1466	0	39	4	2	3	1	1	9	7	
1467	0	27	2	4	2	2	1	6	2	
1468	0	49	4	2	2	2	0	17	6	
1469	0	34	2	4	2	3	0	6	3	

1470 rows × 38 columns

D) 'Age', 'TotalWorkingYears', 'YearsInCurrentRole', 'YearsWithCurrManager' have considerable sknewness. It can be ignored, as it may affect the correlation with the target variable.

```
for i in ['Age','TotalWorkingYears','YearsInCurrentRole','YearsWithCurrManager']:
    plt.figure
    sns.distplot(df_num[i])
    plt.show()
```



Thus, we can see below, 'df_hr' data frame have all attributes of integer data type.

```
df_hr.dtypes
```

```
Attrition                                  int32
Age                                        int64
EnvironmentSatisfaction                    int64
JobInvolvement                             int64
JobLevel                                   int64
JobSatisfaction                            int64
StockOptionLevel                           int64
TotalWorkingYears                          int64
YearsInCurrentRole                         int64
YearsWithCurrManager                       int64
BusinessTravel_Non-Travel                  uint8
BusinessTravel_Travel_Frequently           uint8
BusinessTravel_Travel_Rarely               uint8
Department_Human Resources                 uint8
Department_Research & Development          uint8
Department_Sales                           uint8
EducationField_Human Resources             uint8
EducationField_Life Sciences               uint8
EducationField_Marketing                   uint8
EducationField_Medical                     uint8
EducationField_Other                       uint8
EducationField_Technical Degree            uint8
Gender_Female                              uint8
Gender_Male                                uint8
JobRole_Healthcare Representative          uint8
JobRole_Human Resources                    uint8
JobRole_Laboratory Technician              uint8
JobRole_Manager                            uint8
JobRole_Manufacturing Director             uint8
JobRole_Research Director                  uint8
JobRole_Research Scientist                 uint8
JobRole_Sales Executive                    uint8
JobRole_Sales Representative               uint8
MaritalStatus_Divorced                     uint8
MaritalStatus_Married                      uint8
MaritalStatus_Single                       uint8
OverTime_No                                uint8
OverTime_Yes                               uint8
dtype: object
```

5. Model Development and Evaluation

Now, we drop 'Attrition' target variable from df_hr and store the remaining data in x and store the 'attribute' in y.

```
x=df_hr.drop("Attrition", axis=1)
x.head()
```

	Age	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	StockOptionLevel	TotalWorkingYears	YearsInCurrentRole	YearsWithCurrManager
0	41	2	3	2	4	0	8	4	5
1	49	3	2	2	2	1	10	7	7
2	37	4	2	1	3	0	7	0	0
3	33	4	3	1	3	0	8	7	0
4	27	1	3	1	2	1	6	2	2

5 rows × 37 columns

```
x.shape
```

```
(1470, 37)
```

```
y=df_hr['Attrition']
y
```

```
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: int32
```

```
y.shape
```

```
(1470,)
```

The shape of x is (1470, 37) and y is same is (1470,1). Thus, both have same no. of columns.

The algorithms used for training and testing are LogisticRegression(), DecisionTreeClassifier(), KneighborsClassifier() & SpaceVectorClassifier()

MODEL SELECTION

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42)
```

The key metrics used are Accuracy score, Confusion matrix and classification report.

```
model=[DecisionTreeClassifier(), KNeighborsClassifier(), SVC()]
for i in model:
    print(i)
    i.fit(x_train,y_train)
    pred=i.predict(x_test)
    print('Accuracy score :', accuracy_score(y_test,pred))
    print('Confusion matrix :\n', confusion_matrix(y_test,pred))
    print('Classification report: \n ', classification_report(y_test,pred))
    print('************************************************')
    print('\n')
```

```
DecisionTreeClassifier()
Accuracy score : 0.7755102040816326
Confusion matrix :
 [[215  40]
 [ 26  13]]
Classification report:
              precision    recall  f1-score   support

           0       0.89      0.84      0.87       255
           1       0.25      0.33      0.28        39

    accuracy                           0.78       294
   macro avg       0.57      0.59      0.57       294
weighted avg       0.81      0.78      0.79       294

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

```
KNeighborsClassifier()
Accuracy score : 0.8571428571428571
Confusion matrix :
 [[247   8]
 [ 34   5]]
Classification report:
                  precision    recall  f1-score   support

           0       0.88      0.97      0.92       255
           1       0.38      0.13      0.19        39

    accuracy                           0.86       294
   macro avg       0.63      0.55      0.56       294
weighted avg       0.81      0.86      0.82       294

**************************************************


SVC()
Accuracy score : 0.8673469387755102
Confusion matrix :
 [[255   0]
 [ 39   0]]
Classification report:
                  precision    recall  f1-score   support

           0       0.87      1.00      0.93       255
           1       0.00      0.00      0.00        39

    accuracy                           0.87       294
   macro avg       0.43      0.50      0.46       294
weighted avg       0.75      0.87      0.81       294

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

Since, SVC() gives no zero fi-score to value '1', we reject it. KNeighborsClassifier() has the best accuracy score.

A) ENSEMBLE TECHNIQUE

ENSEMBLE

```
1 from sklearn.ensemble import RandomForestClassifier
2 rf=RandomForestClassifier(random_state=42)
3 rf.fit(x_train,y_train)
4 pred2=rf.predict(x_test)
5 print('Accuracy Score:',accuracy_score(y_test,pred))
6 print('Confusion matrix:',confusion_matrix(y_test,pred))
7 print('Classification report',classification_report(y_test,pred))
```

```
Accuracy Score: 0.8673469387755102
Confusion matrix: [[255   0]
 [ 39   0]]
Classification report                precision    recall  f1-score   support

           0       0.87      1.00      0.93       255
           1       0.00      0.00      0.00        39

    accuracy                           0.87       294
   macro avg       0.43      0.50      0.46       294
weighted avg       0.75      0.87      0.81       294
```

Although it give good accuracy score, but it has zero fi-score for value '1'. We reject it.

B) CROS VALIDATION

CROSS VALIDATION

```
]:  1 for i in model:
    2     cross=cross_val_score(i,x,y,cv=5)
    3     print(i)
    4     print('Score:',cross)
    5     print('Mean_score:',cross.mean())
    6     print('STD_score:',cross.std())
    7     print('**************************/n')
```

```
DecisionTreeClassifier()
Score: [0.81292517 0.82653061 0.79931973 0.74489796 0.82653061]
Mean_score: 0.8020408163265307
STD_score: 0.03030075843717018
**************************/n
KNeighborsClassifier()
Score: [0.81292517 0.84693878 0.85034014 0.83333333 0.84693878]
Mean_score: 0.8380952380952381
STD_score: 0.013874883030184445
**************************/n
SVC()
Score: [0.83673469 0.83673469 0.84013605 0.84013605 0.84013605]
Mean_score: 0.8387755102040817
STD_score: 0.0016663195529137286
**************************/n
```

```
]:  1 cross=cross_val_score(rf,x,y,cv=5)
    2 print('RandomForestClassifier()')
    3 print('Score:',cross)
    4 print('Mean_score:',cross.mean())
    5 print('STD_score:',cross.std())
    6 print('**************************/n')
```

```
RandomForestClassifier()
Score: [0.83673469 0.86394558 0.86394558 0.85714286 0.86734694]
Mean_score: 0.8578231292517007
STD_score: 0.011053113475695153
**************************/n
```

There is high difference of highest score and lowest score in DecisionTreeClassifier(). Thus, we reject it.

KNeighborsClassifier(), SVC() & RandomForestClassfier() has good cross validation.

Since KNeighborsClassifier gave the highest accuracy score. We pass it on to Hyperparameter Tuning.

HYPERPARAMETER TUNING

```
|:  1 from sklearn.model_selection import GridSearchCV
    2 parameters = {'n_neighbors':[1,10]}
    3 knc=KNeighborsClassifier()
    4 Grid=GridSearchCV(knc,parameters)
    5 Grid.fit(x_train,y_train)
```

```
|: GridSearchCV(estimator=KNeighborsClassifier(),
              param_grid={'n_neighbors': [1, 10]})
```

```
|:  1 Grid.best_params_
```

```
|: {'n_neighbors': 10}
```

```
|:  1 knc2=KNeighborsClassifier(n_neighbors=10)
    2 knc2.fit(x_train,y_train)
    3 pred=knc2.predict(x_test)
    4 print('Accuracy score :', accuracy_score(y_test,pred))
    5 print('Confusion matrix :\n', confusion_matrix(y_test,pred))
    6 print('Classification report: \n ', classification_report(y_test,pred))
```

```
Accuracy score : 0.8741496598639455
Confusion matrix :
 [[254   1]
 [ 36   3]]
Classification report:
                precision    recall  f1-score   support

           0       0.88      1.00      0.93       255
           1       0.75      0.08      0.14        39

    accuracy                           0.87       294
   macro avg       0.81      0.54      0.54       294
weighted avg       0.86      0.87      0.83       294
```

Thus , by applying hyperparameter tunning, we have find out, the algorithm will give best result with the number of neighbors to be 10.

We apply it again and found the result as shown above.

It gives an accuracy score of 87.42%