BLOG SUBMISSION

HR Analytics Project - Understanding the Attrition in HR

Submitted By:

M Ashok Kumar

ACKNOWLEDGMENT

In this blog, I will be analysing the HR Analytics Project- Understanding the Attrition in HR using Machine Learning dataset, using essential exploratory data analysis techniques and also, I will be performing some data visualizations to better understand our data.

In the dataset, there are many columns like Age, Attrition, Daily Rate, Department, Education and so on. By doing data pre-processing, data analysis, feature selection, and many other techniques we built our cool and fancy machine learning model. And at the end, we applied many ML algorithms to get the very good accuracy of our model.

Many thanks to Data Trained for providing me required knowledge with this project to understand the Realtime Field work present in the Data Science Industry.

I am very Thankful to the entire team and also my family members for encouraging and supporting me in submitting this blog.

ABSTRACT

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?

TAKEAWAYS FROM THE BLOG

In this article, we do prediction using machine learning which leads to the below takeaways:

1. EDA: Learn the complete process of EDA
2. Data analysis: Learn to withdraw some insights from the dataset both mathematically and visualize it.
3. Data visualization: Visualizing the data to get better insight from it.
4. Feature engineering: We will also see what kind of stuff we can do in the feature engineering part.

PROBLEM STATEMENT:

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

HR Analytics:

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

Attrition in HR:

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

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

Attrition affecting Companies

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

Size of the Dataset:

The Dataset Contains 1470 rows × 35 columns.

ABOUT THE DATASET

You can find the dataset in the link below:

- https://github.com/dsrscientist/IBM_HR_Attrition_Rate_Analytics

Importing Important Libraries:

We need some libraries to be imported to work upon the dataset, we would import the dataset by using pandas read_csv method.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix, accuracy_score
```

Loading Data Set into a Desired variable:

Here I am loading the Dataset into the variable hr_df.

```
hr_df = pd.read_csv("Hr Analysis.csv")
hr_df
```

Here we have imported the dataset into our callable variable and the dataset contains 1470 columns and 35 rows.

Exploratory Data Analysis:

Before you start a machine learning project, it's important to ensure that the data is ready for modelling work.

Exploratory Data Analysis (EDA) ensures the readiness of the data for Machine Learning.

In fact, EDA is primarily used to see what data can reveal beyond the formal modelling or hypothesis testing task and provides a better understanding of data set variables and the relationships between them.

Let's do the EDA using the statistical tequines.

Getting detailed info about the Dataset:

```
hr_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
```

There are 26 integer columns and 9 object type columns are present in the dataset

and there are no null values present in the dataset.

Let's Check the statistical Data for the Dataset:

Here, we use describe() to get the statistical values of the Dataset.

```
hr_df.describe()
```

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

8 rows × 26 columns

By observing the data:

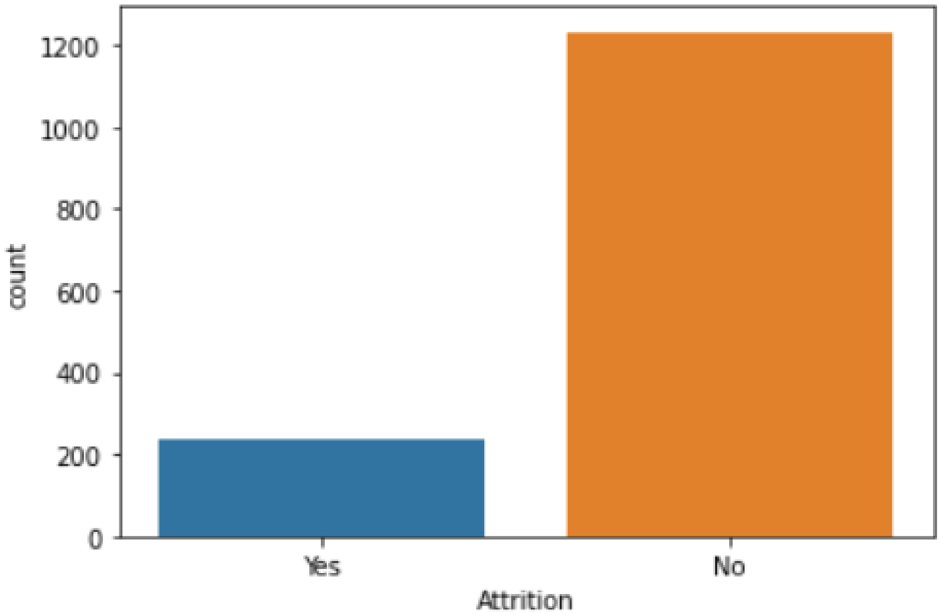
There is a huge difference for 75th percentile and maximum values for EmployeeNumber. so there may be outliers present in the dataset.

Let's Analyse the single variables separately by using the Univariate Analysis

Checking the categorical columns and Plotting graphs for better insight of Data Distribution

```
#Checking the target column
sns.countplot(x=hr_df['Attrition'])
print(hr_df['Attrition'].value_counts())
```

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



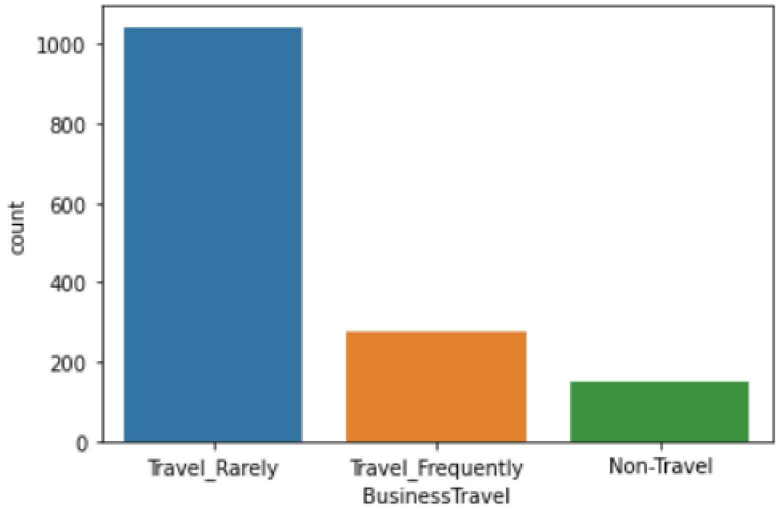
Here we can see that out of 1470 employees, 237 employees which is around 16% of total employees left their job due to some reasons.

whereas other 1233 employees, which is 84% of the employees preferred to continue in their job at the company.

Also, here we can observe that there is huge difference between two types of Attrition. So, the data is imbalanced. So, we will apply SMOTE analysis before ML of final model.

```
sns.countplot(x=hr_df['BusinessTravel'])
print(hr_df[ 'BusinessTravel'].value_counts())
```

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

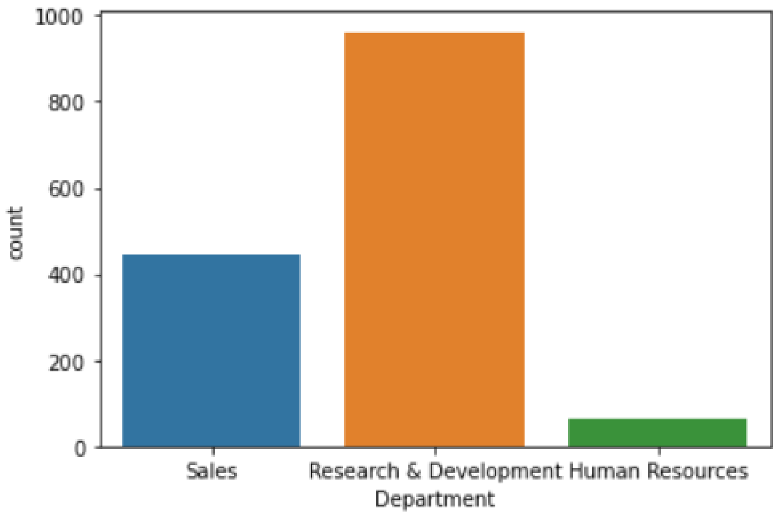


Here, business travel is divided in 3 categories with least number of non-travellers.

It has maximum number of employees who travels rarely.

```
sns.countplot(x=hr_df['Department'])
print(hr_df['Department'].value_counts())
```

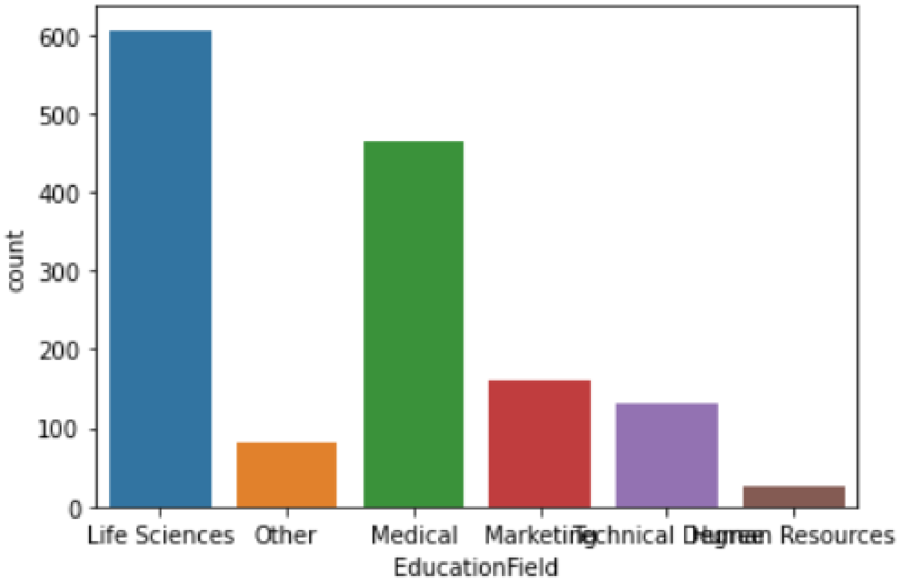
```
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
```



Out of 3 different departments in the dataset Research & Development has maximum number of employees.

```
sns.countplot(x=hr_df['EducationField'])
print(hr_df['EducationField'].value_counts())
```

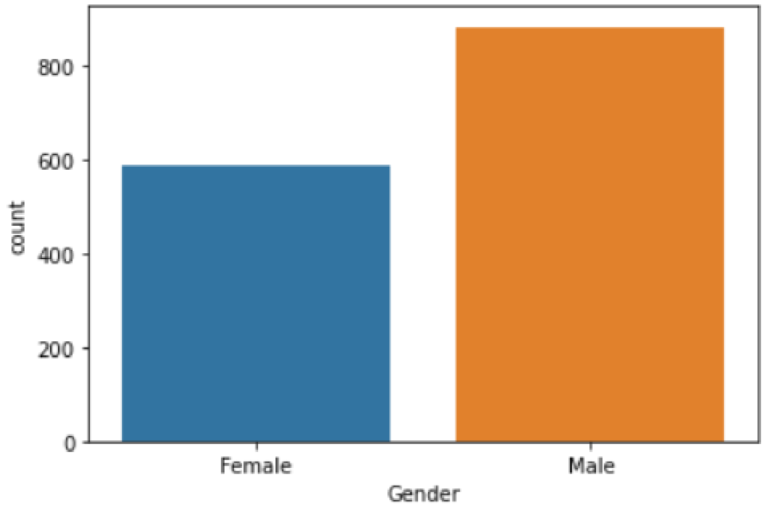
```
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
```



Most of the employees are from life sciences background and least are from human resource.

```
sns.countplot(x=hr_df['Gender'])
print(hr_df['Gender'].value_counts())
```

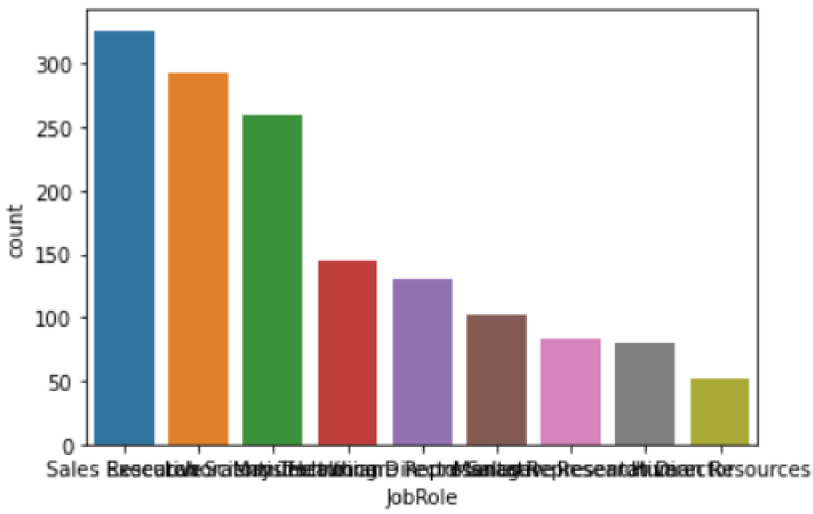
```
Male      882
Female    588
Name: Gender, dtype: int64
```



We can Observe that male employee has more attrition than female employees.

```
sns.countplot(x=hr_df['JobRole'])
print(hr_df['JobRole'].value_counts())
```

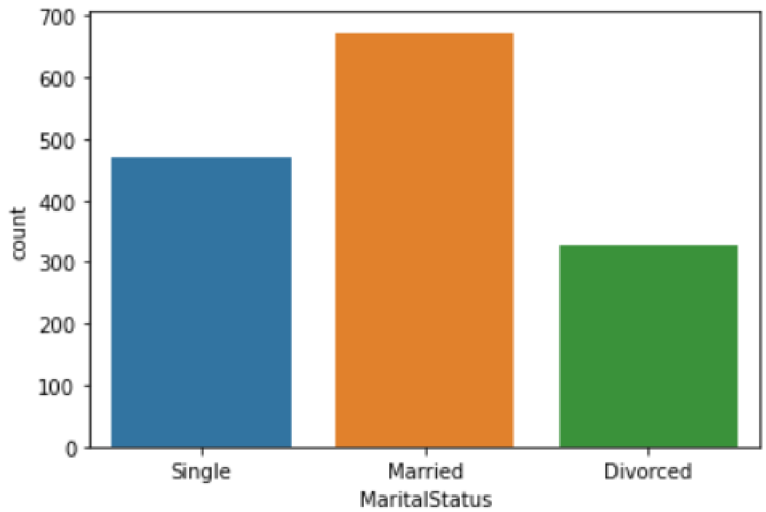
```
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52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



Most of the employee are from Sales department. And least are from HR.

```
sns.countplot(x=hr_df['MaritalStatus'])
print(hr_df['MaritalStatus'].value_counts())
```

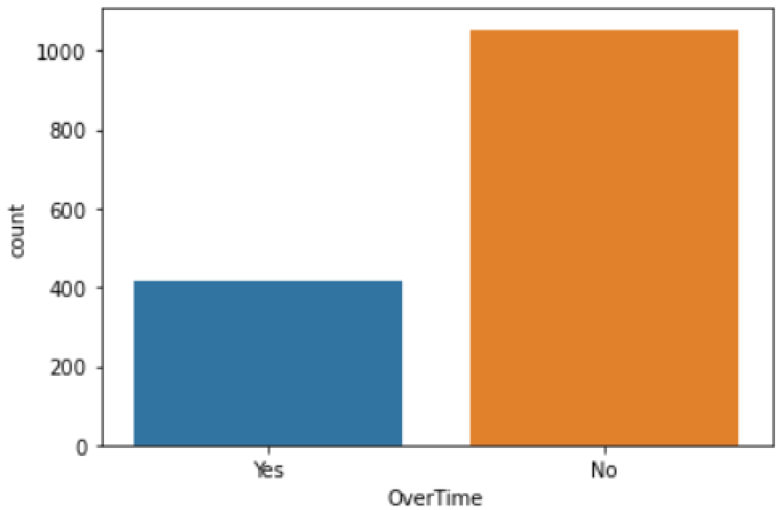
```
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
```



Most of the employees are married.

```
sns.countplot(x=hr_df['OverTime'])
print(hr_df['OverTime'].value_counts())
```

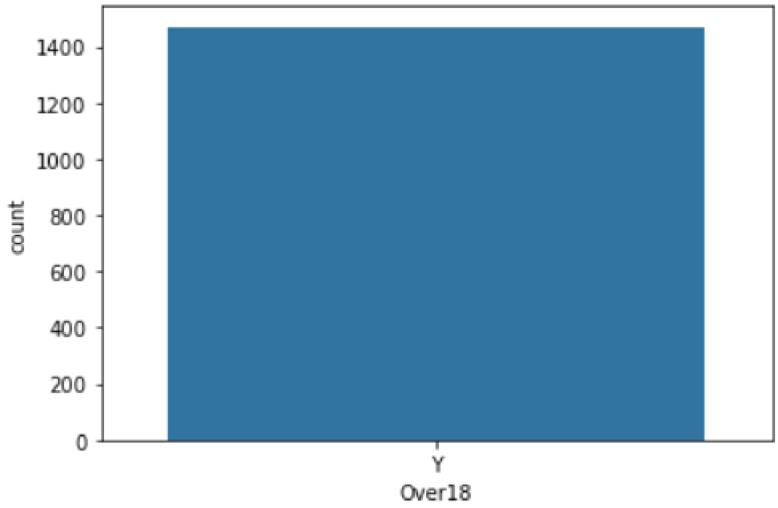
```
No     1054
Yes     416
Name: OverTime, dtype: int64
```



Only 416 employees work overtime.

```
sns.countplot(x=hr_df['Over18'])
```

```
<AxesSubplot:xlabel='Over18', ylabel='count'>
```



All the employees are above 18.

so, we can drop this column as there is no use of this.

Analysis of the Rating Features JobSatisfaction

EnvironmentSatisfaction

RelationshipSatisfaction

JobInvolvement

WorkLifeBalance

PerformanceRating

```python
fig = plt.figure()

ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(223)
ax4 = fig.add_subplot(224)

labels = 'Low','Medium','High','Very High'

hr_df['JobSatisfaction'].astype(str).value_counts().plot(kind='pie',
                            figsize=(15, 6),
                            autopct='%1.1f%%',
                            startangle=90,
                            shadow=True,
                            labels=None,ax=ax1) # add to subplot 2
ax1.set_title ('Rating of Job Satisfaction by Employees')
fig.legend(labels=labels,loc='center')

hr_df['EnvironmentSatisfaction'].astype(str).value_counts().plot(kind='pie',
                            figsize=(15, 6),
                            autopct='%1.1f%%',
                            startangle=90,
                            shadow=True,
                            labels=None,ax=ax2)
ax2.set_title('Rating of Environmental Satisfaction by Employees')

hr_df['RelationshipSatisfaction'].astype(str).value_counts().plot(kind='pie',
                            figsize=(15, 6),
                            autopct='%1.1f%%',
                            startangle=90,
                            shadow=True,
                            labels=None,ax=ax3)
ax3.set_title('Rating of Relationship Satisfaction by Employees')

hr_df['JobInvolvement'].astype(str).value_counts().plot(kind='pie',
                            figsize=(15, 6),
                            autopct='%1.1f%%',
                            startangle=90,
                            shadow=True,
                            labels=None,ax=ax4)
ax4.set_title('Rating of Job Involvement by Employees')

plt.show()
```

Here we can see the rating features are low for each cases which also effects the performance of the employees.

Employees are

Not Satisfied in their Job

Not Satisfied with their Work Environment

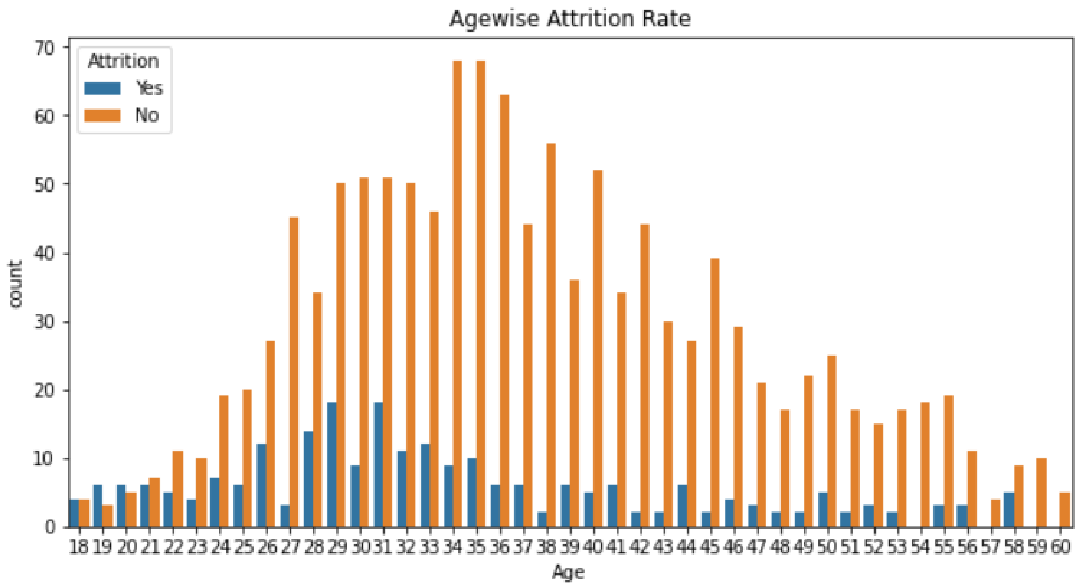
Not Satisfied in their Relationship

Not Getting involved in their job

Let's do the Bivariate Analysis:

```
plt.figure(figsize=(10,5))
sns.countplot(x='Age',hue='Attrition',data=hr_df)
plt.title('Agewise Attrition Rate')
print(hr_df.groupby('Age')['Attrition'].value_counts())
```

```
Age  Attrition
18   No            4
     Yes           4
19   Yes           6
     No            3
20   Yes           6
                  ..
57   No            4
58   No            9
     Yes           5
59   No           10
60   No            5
Name: Attrition, Length: 82, dtype: int64
```

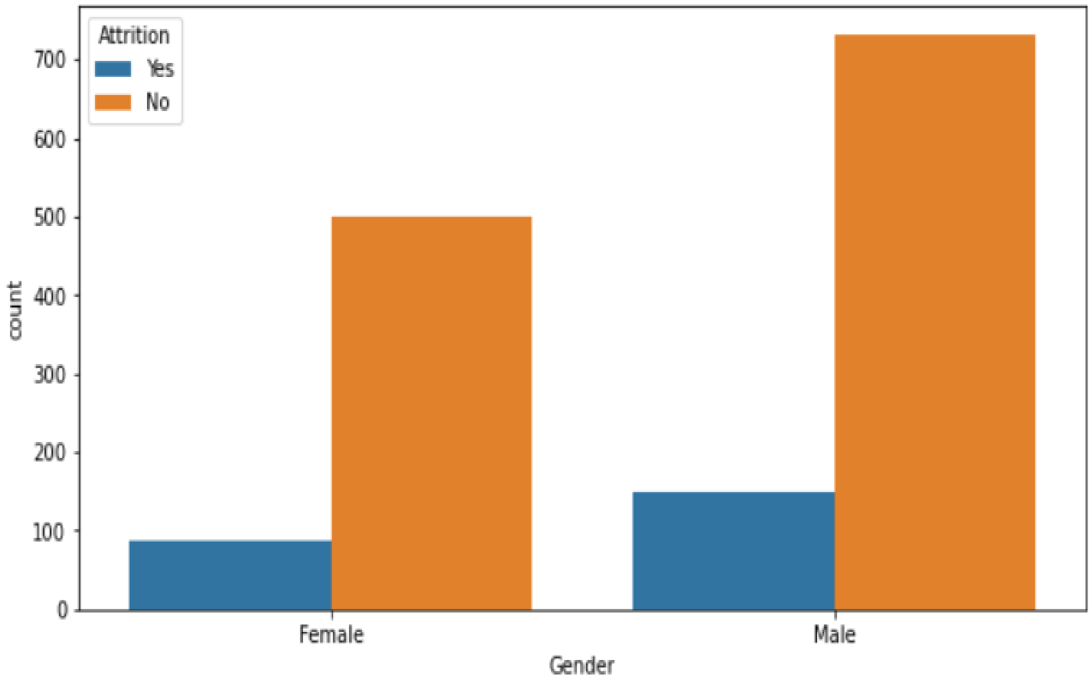


Here we can see the employees between 25yrs to 35 years of age have highest attrition rate.

Let’s Check which gender has more attrition.

```
plt.figure(figsize=(10,5))
sns.countplot(x='Gender',hue='Attrition',data=hr_df)
print(hr_df.groupby('Gender')['Attrition'].value_counts())
```

```
Gender  Attrition
Female  No           501
        Yes           87
Male    No           732
        Yes          150
Name: Attrition, dtype: int64
```

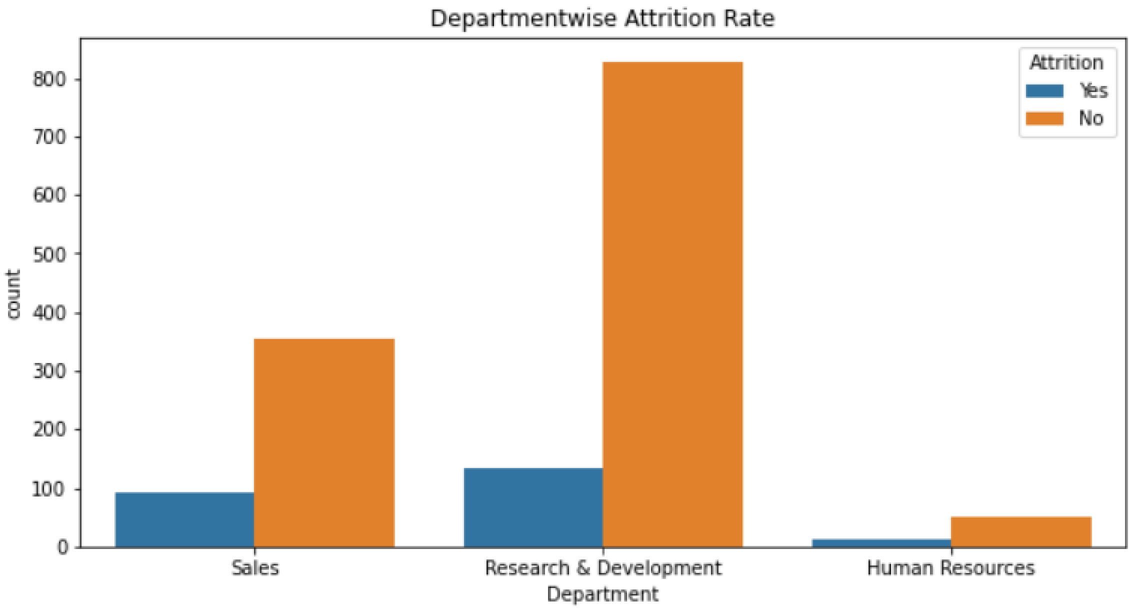


Here, we can see attrition does not much depend on sex of the employees.

Let’s check which department has highest attrition.

```
plt.figure(figsize=(10,5))
sns.countplot(x='Department',hue='Attrition',data=hr_df)
plt.title('Departmentwise Attrition Rate')
print(hr_df.groupby('Department')['Attrition'].value_counts())
```

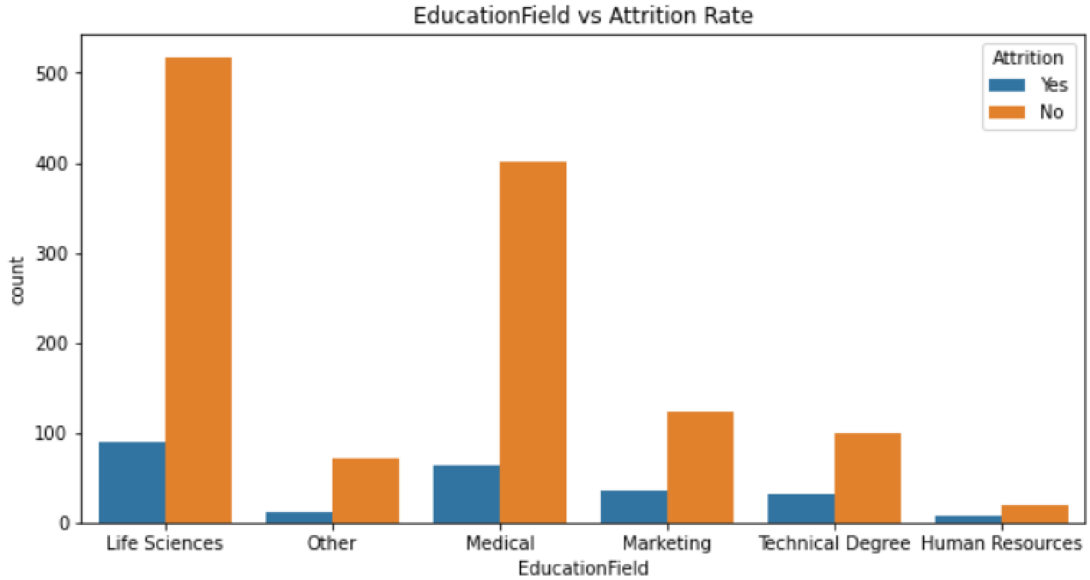
```
Department              Attrition
Human Resources         No            51
                        Yes           12
Research & Development  No           828
                        Yes          133
Sales                   No           354
                        Yes           92
Name: Attrition, dtype: int64
```



From the above graph we can see that Highest attrition is in Research & Development department.

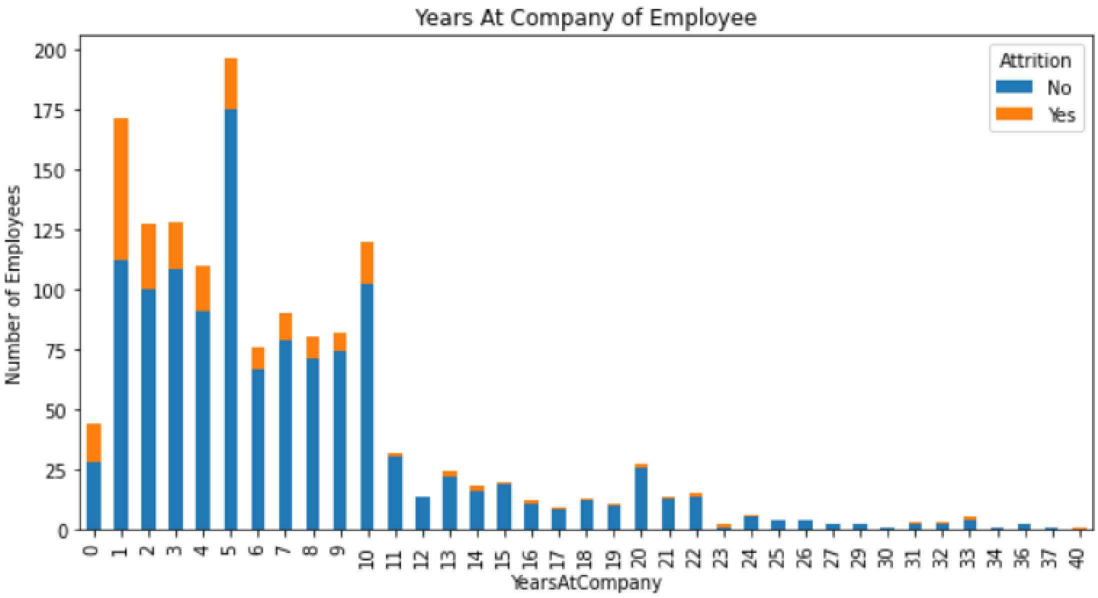
```
plt.figure(figsize=(10,5))
sns.countplot(x='EducationField',hue='Attrition',data=hr_df)
plt.title('EducationField vs Attrition Rate')
print(hr_df.groupby('EducationField')['Attrition'].value_counts())
```

```
EducationField    Attrition
Human Resources   No            20
                  Yes            7
Life Sciences     No           517
                  Yes           89
Marketing         No           124
                  Yes           35
Medical           No           401
                  Yes           63
Other             No            71
                  Yes           11
Technical Degree  No           100
                  Yes           32
Name: Attrition, dtype: int64
```



We can see that the people who are in life sciences have more attrition rate.

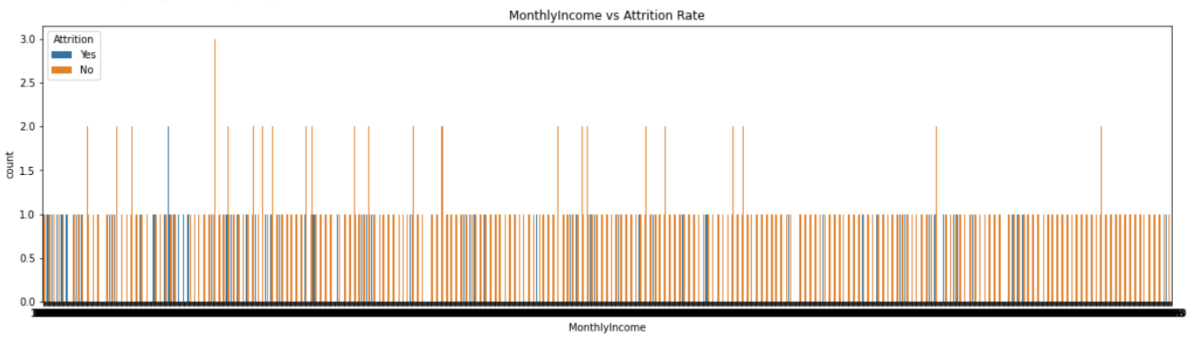
```
yac = hr_df.groupby("YearsAtCompany")['Attrition'].value_counts(normalize=False).unstack()

yac.plot(kind='bar', stacked='False',figsize=(10,5))

plt.title('Years At Company of Employee')
plt.ylabel('Number of Employees')
plt.show()
```



From the above graph It is observed that the newly arriving employees quit their jobs most.

```
plt.figure(figsize=(20,5))
sns.countplot(x='MonthlyIncome',hue='Attrition',data=hr_df)
plt.title('MonthlyIncome vs Attrition Rate')
print(hr_df.groupby('MonthlyIncome')['Attrition'].value_counts())
```

```
MonthlyIncome  Attrition
1009           Yes          1
1051           No           1
1052           No           1
1081           Yes          1
1091           Yes          1
                           ..
19859          Yes          1
19926          No           1
19943          No           1
19973          No           1
19999          No           1
Name: Attrition, Length: 1388, dtype: int64
```



Employees who left their jobs tend to have low monthly income than those who continued their job in the company.

Let's check the data with Multivariate analysis:

```
# distribution of the data
hr_df.iloc[:,1:-1].hist(figsize=(20,20));
```

Converting categorical values to numerical values using label encoder:

```
# use label encoder to change data type in type and region columns
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
list=['Attrition','BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','OverTime']
for val in list:
    hr_df[val]=le.fit_transform(hr_df[val].astype(str))
```

For Attrition column, "Yes":1,"No":0

For BusinessTravel column, "non-Travel":0,"Travel_Frequently":1,"Travel_Rarely":2

For Department column, “Human Resources":0,"Research & Development":1,"Sales":2.

For Gender column,"Female":0,"Male":1

For MaritalStatus column, "Divorced":0,"Married":1,"Single":2

For OverTime column,"Yes":0,"No":1

For JobRole column,'Sales Executive': 7,'Research Scientist': 5,laboratory Technician': 2,'Manufacturing Director': 3, 'Healthcare Representative': 1,'Manager': 4,'Sales Representative': 8,'Research Director': 6,'Human Resources': 0

For EducationField column,Life Sciences': 2,'Other': 1,'Medical':4 ,'Marketing': 3,'Technical Degree': 5, 'HumanResources': 0

Over18 and EmployeeCount have only one value thus it will not provide any information about the data. Considering EmployeeNumber is emp ID thus deleting it.

```
hr_df.drop(columns =["Over18","EmployeeCount","EmployeeNumber","StandardHours"],axis =1, inplace = True)
```

```
hr_df.head(2)
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction	Gender	...	PerformanceRating
0	41	1	2	1102	2	1	2	1	2	0	...	3
1	49	0	1	279	1	8	1	1	3	1	...	4

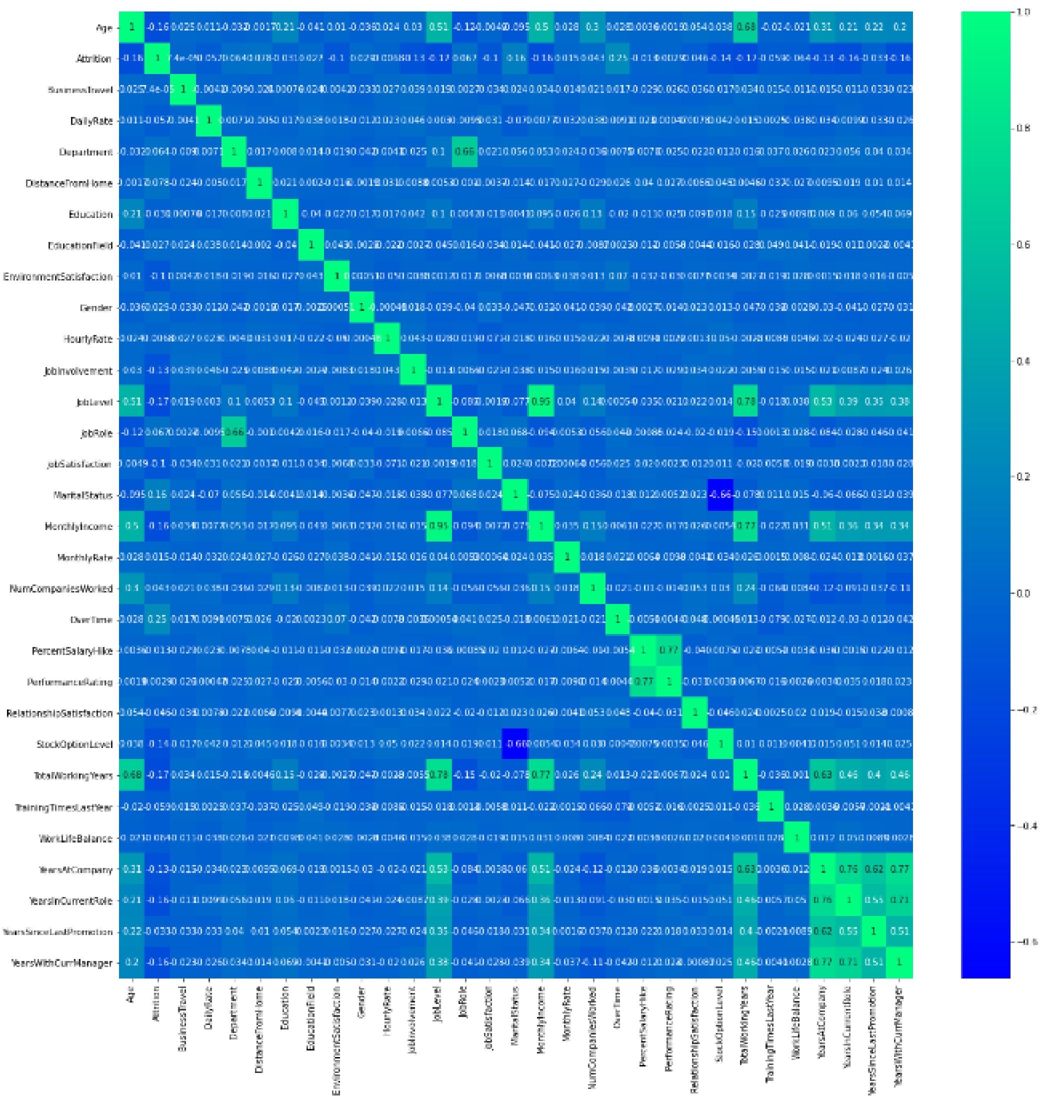
2 rows × 31 columns

Now there are 31 columns left.

Now, Let’s check the correlation:

```
#Checking with heatmap
plt.figure(figsize=(20,20))
sns.heatmap(hr_df.corr(),annot=True,cmap='winter')
```

```
<AxesSubplot:>
```



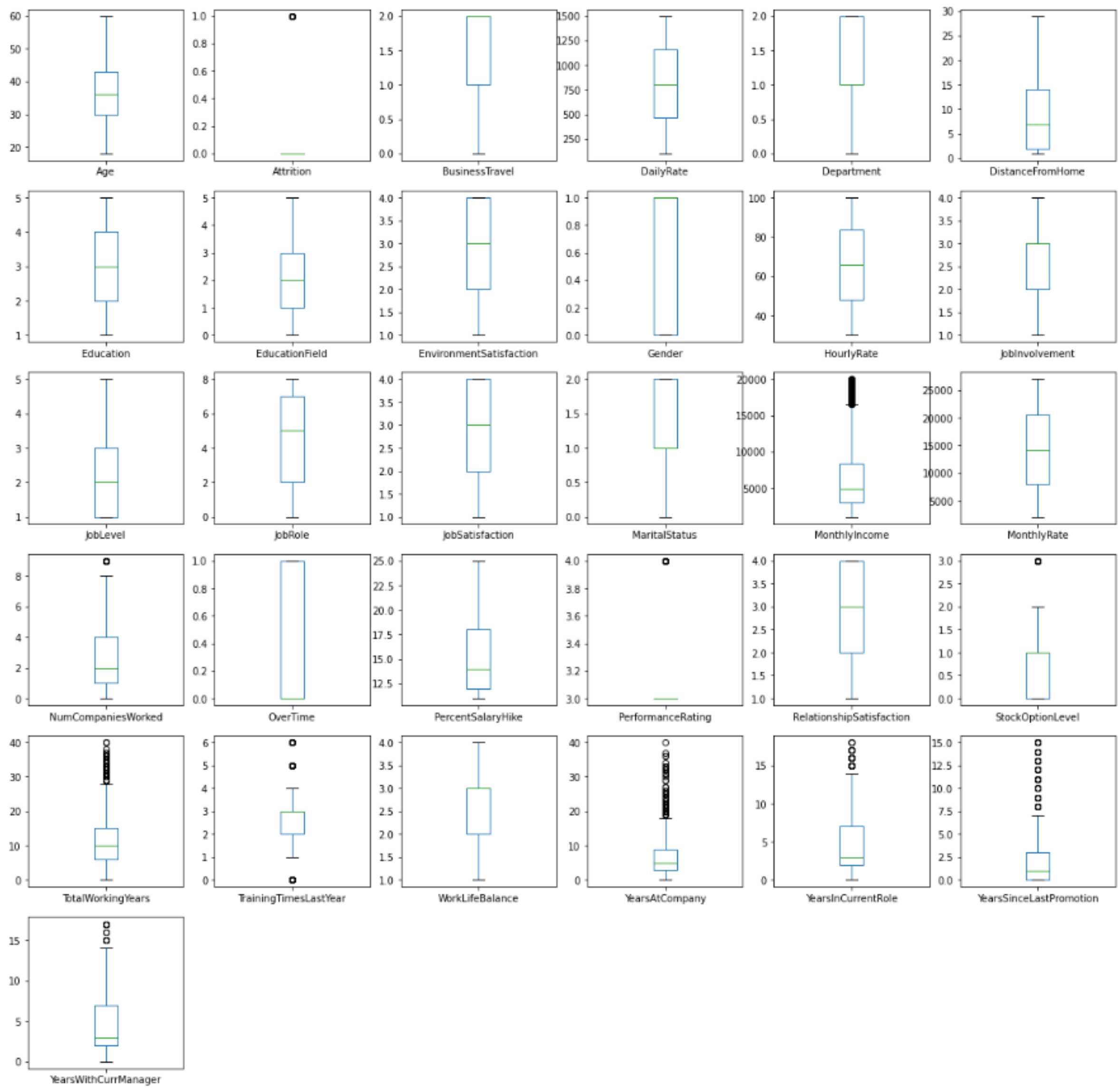
None of the column is highly correlated with Attrition. So many columns are negatively correlated with Attrition.

Job role is highly correlated with department, job level is highly positively correlated with monthlyincome and totalworkingyear, total working year is correlated with Age, job level, monthly income.

year of company correlated with year at current role and year with current manager.

Let's check whether there are outliers present in the data or not.

```
hr_df.plot(kind='box',subplots=True,layout=(6,6),figsize=(20,20))
```



There are outliers in MonthlyIncome, Stock option Level,Total Working years,Training Times last year, Years at company,years in current role and Years with current manager.

Let’s Remove the outliers

```
from scipy.stats import zscore

z=np.abs(zscore(hr_df))
z
```

Let’s check the shape of the data.

```
z.shape
```

```
(1470, 31)
```

```
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([30, 29, 27, 29, 28, 29, 24, 24, 27, 29, 28, 29, 30, 24, 27, 29, 30,
       29, 24, 30, 27, 28, 29, 28, 30, 27, 29, 24, 27, 28, 29, 29, 30, 24,
       27, 27, 29, 29, 24, 28, 27, 27, 29, 27, 30, 29, 27, 24, 27, 29, 30,
       24, 30, 27, 29, 27, 30, 29, 28, 28, 27, 29, 29, 29, 27, 29, 29, 30,
       24, 27, 29, 27, 29, 29, 30, 29, 24, 27, 28, 29, 29, 28, 24, 29, 30,
       27, 29, 29, 27, 24, 27, 27, 27, 29, 29, 24, 29, 29, 29, 29, 24, 29,
       29, 28, 29, 30, 28, 24, 29, 28], dtype=int64))
```

```
print(hr_df.shape)
print(hr_new.shape)
```

```
(1470, 31)
(1387, 31)
```

From the above we can understand that total dataset contains 1470 rows and 31 columns.

But after removing the outliers, the data contains 1387 rows and 31 columns.

So, there is a loss of 5.71% of the data.

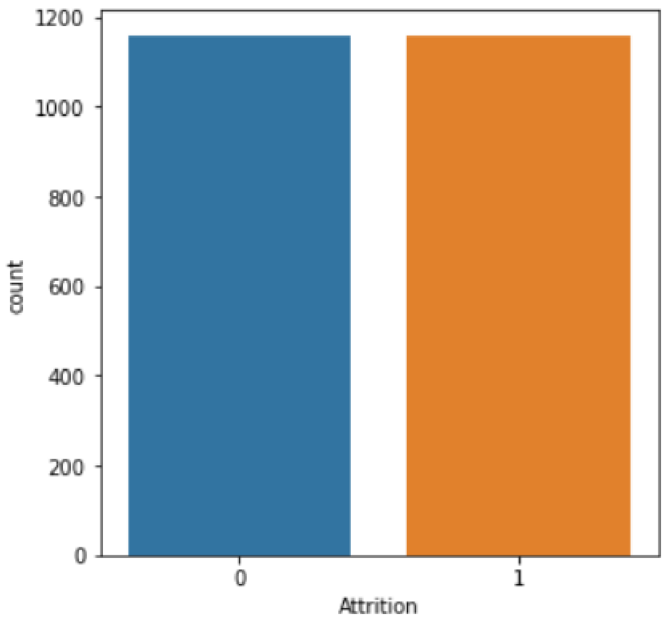
Let’s Handle the imbalance data by using SMOTE Analysis.

```
# Seprating data into X and y
x_new = hr_new.drop("Attrition",axis =1)
y_new= hr_new["Attrition"]
```

```
# implementing oversampling for handling imbalance data
from imblearn.over_sampling import SMOTE
smt=SMOTE()
sm_x,sm_y=smt.fit_resample(x_new,y_new)
#y.value_counts()
sm_y.value_counts()
```

```
0    1158
1    1158
Name: Attrition, dtype: int64
```

```
plt.figure(figsize=(5,5))
sns.countplot(sm_y)
plt.show()
```



Now the data is Balanced.

Let’s do the Modelling:

```
maxAcc = 0
maxRS = 0

for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(sm_x,sm_y,random_state=i,test_size=0.20)
    LR = LogisticRegression()
    LR.fit(x_train,y_train)
    predrf= LR.predict(x_test)
    acc=accuracy_score(y_test,predrf)
    if acc>maxAcc:
        maxAcc=acc
        maxRS=i
print('Best score is',maxAcc,'on Random State',maxRS)
```

```
Best score is 0.8685344827586207 on Random State 75
```

Let’s Make input and output variables into train and test data.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(sm_x,sm_y,random_state=75,test_size=0.20)
```

Let’s check the size of the test and train:

```
print(y_train.shape,'\t',y_test.shape)
```

```
(1852,)          (464,)
```

Let’s check with different algorithms:

```
model=[LogisticRegression(),GaussianNB(),SVC(),DecisionTreeClassifier(),KNeighborsClassifier(),]

for m in model:
    m.fit(x_train,y_train)
    m.score(x_train,y_train)
    predm=m.predict(x_test)
    print('Accuracy score of',m, 'is:')
    print(accuracy_score(y_test,predm))
    print(confusion_matrix(y_test,predm))
    print(classification_report(y_test,predm))
    print('\n')
```

```
Accuracy score of LogisticRegression() is:
0.8685344827586207
[[212  31]
 [ 30 191]]
              precision    recall  f1-score   support

           0       0.88      0.87      0.87       243
           1       0.86      0.86      0.86       221

    accuracy                           0.87       464
   macro avg       0.87      0.87      0.87       464
weighted avg       0.87      0.87      0.87       464



Accuracy score of GaussianNB() is:
0.75
[[169  74]
 [ 42 179]]
              precision    recall  f1-score   support

           0       0.80      0.70      0.74       243
           1       0.71      0.81      0.76       221

    accuracy                           0.75       464
   macro avg       0.75      0.75      0.75       464
weighted avg       0.76      0.75      0.75       464



Accuracy score of SVC() is:
0.9267241379310345
[[224  19]
 [ 15 206]]
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       243
           1       0.92      0.93      0.92       221

    accuracy                           0.93       464
   macro avg       0.93      0.93      0.93       464
weighted avg       0.93      0.93      0.93       464



Accuracy score of DecisionTreeClassifier() is:
0.8405172413793104
[[196  47]
 [ 27 194]]
              precision    recall  f1-score   support

           0       0.88      0.81      0.84       243
           1       0.80      0.88      0.84       221

    accuracy                           0.84       464
   macro avg       0.84      0.84      0.84       464
weighted avg       0.84      0.84      0.84       464



Accuracy score of KNeighborsClassifier() is:
0.8577586206896551
[[178  65]
 [  1 220]]
              precision    recall  f1-score   support

           0       0.99      0.73      0.84       243
           1       0.77      1.00      0.87       221

    accuracy                           0.86       464
   macro avg       0.88      0.86      0.86       464
weighted avg       0.89      0.86      0.86       464
```

We got the highest accuracy with SVC.

```
# Finding out best paramter using GridsearchCV
from sklearn.model_selection import GridSearchCV
parameters={'kernel':['linear','rbf','poly','sigmoid'],'C':[0,1,2,3,4,5,6,7,8,9,10,11,12]}
svc=SVC()
gs=GridSearchCV(svc,parameters)
gs.fit(sm_x,sm_y)
```

```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],
                         'kernel': ['linear', 'rbf', 'poly', 'sigmoid']})
```

```
print(gs.best_params_)
```

```
{'C': 10, 'kernel': 'rbf'}
```

```
#Using SVC model with best results
sv=SVC(kernel='rbf',C=10)
sv.fit(x_train,y_train)
sv.score(x_train,y_train)
predsv=sv.predict(x_test)
print('Accuracy score of',sv, 'is:')
print(accuracy_score(y_test,predsv))
print(confusion_matrix(y_test,predsv))
print(classification_report(y_test,predsv))
print('\n')
```

```
Accuracy score of SVC(C=10) is:
0.9482758620689655
[[225  18]
 [  6 215]]
              precision    recall  f1-score   support

           0       0.97      0.93      0.95       243
           1       0.92      0.97      0.95       221

    accuracy                           0.95       464
   macro avg       0.95      0.95      0.95       464
weighted avg       0.95      0.95      0.95       464
```

Let's use Ensemble Technique to boost up score:

```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n_estimators=100,random_state=75,criterion='gini')
#RandomForestClassifier(100)---Default
rf.fit(x_train,y_train)
predrf=rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
0.9461206896551724
[[230  13]
 [ 12 209]]
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       243
           1       0.94      0.95      0.94       221

    accuracy                           0.95       464
   macro avg       0.95      0.95      0.95       464
weighted avg       0.95      0.95      0.95       464
```

```
from sklearn.ensemble import AdaBoostClassifier
ad=AdaBoostClassifier(n_estimators=100,random_state=75,base_estimator=sv,algorithm='SAMME',learning_rate=0.01)
ad.fit(x_train,y_train)
ad_pred=ad.predict(x_test)
print(accuracy_score(y_test,ad_pred))
print(confusion_matrix(y_test,ad_pred))
print(classification_report(y_test,ad_pred))
```

```
0.47629310344827586
[[  0 243]
 [  0 221]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       243
           1       0.48      1.00      0.65       221

    accuracy                           0.48       464
   macro avg       0.24      0.50      0.32       464
weighted avg       0.23      0.48      0.31       464
```

```python
# Finding out best paramter using GridsearchCV
from sklearn.model_selection import GridSearchCV
parameters={'random_state':range(35,100)}
gc=GridSearchCV(sv,parameters)
gc.fit(sm_x,sm_y)
gc.best_params_
```

```
{'random_state': 35}
```

Hyperparameter Tuning

```python
RF = RandomForestClassifier() # Randomforest instance
from sklearn.model_selection import RandomizedSearchCV,cross_val_score

# we are checking best parameter which suited most for the model

para = {
        "criterion":["gini","entropy"],
        "max_features" : ["auto", "sqrt", "log2"],
        "min_samples_split" :[2,4,6,8],
        "max_depth" :[2,3,4,5],
        "min_samples_leaf":[3,5,8,10],
        "max_samples" :[0.2,0.3,0.4,0.5]
}

grid = RandomizedSearchCV(RF, para, scoring= "accuracy" )

grid.fit(x_train,y_train)
print("score",grid.best_score_)
print("best parameter",grid.best_params_)
print("best estimator",grid.best_estimator_)
grid_pred=grid.best_estimator_.predict(x_test) # predicting with best parameters
```

```
score 0.8326174692212428
best parameter {'min_samples_split': 4, 'min_samples_leaf': 3, 'max_samples': 0.4, 'max_features': 'log2', 'max_depth': 5, 'criterion': 'gini'}
best estimator RandomForestClassifier(max_depth=5, max_features='log2', max_samples=0.4,
                       min_samples_leaf=3, min_samples_split=4)
```

Cross Validation

```python
best_parameter_RF = RandomForestClassifier(min_samples_split= 4, min_samples_leaf = 3, max_samples =0.4, max_features = 'log2', max_depth = 5, crite

for i in range(2,7):
    cv = cross_val_score(best_parameter_RF,sm_x,sm_y,cv=i)
    print(f'at CV {i} The mean is  {cv.mean()} and the SD is {cv.std()}')
```

```
at CV 2 The mean is  0.8385146804835923 and the SD is 0.03108808290155446
at CV 3 The mean is  0.8337651122625216 and the SD is 0.04518644370276731
at CV 4 The mean is  0.8354922279792746 and the SD is 0.05899275910295336
at CV 5 The mean is  0.8247598123184628 and the SD is 0.07208950384366851
at CV 6 The mean is  0.8380829015544041 and the SD is 0.06157506645991527
```

Observation

since Randomforest worked well out of all other model, so we have done the hyperparameter tuning to set the best parameter for final model. Now i have checked the best CV as well that at level of CV is generated the best score and we have found CV 6 is at best

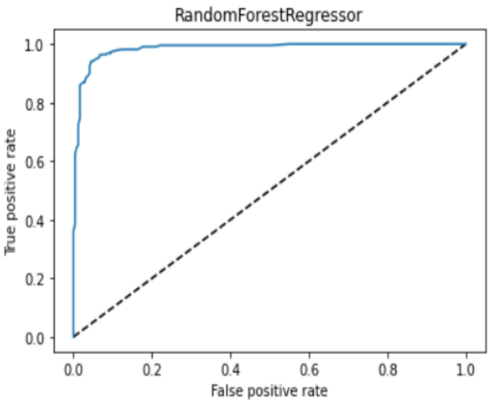
```python
# Cross validate of RandomForestClassifier using cv=6
from sklearn.model_selection import cross_val_score
score=cross_val_score(rf,sm_x,sm_y,cv=6,scoring='accuracy')
print('Score:', score)
print('Mean Score:', score.mean())
print('Standard Deviation:', score.std())
```

```
Score: [0.73834197 0.92746114 0.94041451 0.9507772  0.94041451 0.93005181]
Mean Score: 0.9045768566493955
Standard Deviation: 0.07473268822844518
```

```
auc_score=roc_auc_score(y_test,rf.predict(x_test))
print(auc_score)
```

```
0.9461017075396161
```

```
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr,label='RandomForestRegressor')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('RandomForestRegressor')
plt.show()
```



I found out best result with RandomForestRegressor by using varius technics. So i will save RandomForestRegressor as my final model.

Conclusion:

So, as we saw that we have done a complete EDA process, getting data insights, feature engineering, and data visualization as well so after all these steps one can go for the prediction using machine learning model-making steps.

We have training and test file separately available with us. All the independent variables are categorical in nature and the dependent variable of train data i.e., the attriton is the classical data type. So, I applied the regression method for prediction.

I found out best result with RandomForestRegressor by using varius technics. So i will save RandomForestRegressor as my final model.

I hope this article helped you to understand Data Analysis, Data Preparation, and Model building approaches in a much simpler way.

Thank you for reading this blog