HR Employee Attrition

Important links:

Dataset link: https://www.kaggle.com/code/doraoliveira/eda-visualizations-hranalysis/data

About the Dataset

1. HR Analytics helps us with interpreting organizational data. It finds the people-related trends in the data and allows the HR Department to take the appropriate steps to keep the organization running smoothly and profitably. Attrition in a corporate setup is one of the complex challenges that the people managers and the HRs personnel have to deal with.

2. Interestingly, Machine Learning models can be deployed to predict potential attrition cases, helping the appropriate HR Personnel take the necessary steps to retain the employee.

Loading the data

```
''' Loading HR-Employee-Attrition.csv file into out Local '''

# using wget we have downloaded the required file into our local
# sometimes below command may not work then we need to add 'curl wget'
extension to our browser and then take the required path from there
# to download th eDataset to our local

!wget --header="Host: storage.googleapis.com" --header="User-Agent:
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36" --header="Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --
header="Accept-Language: en-GB,en-US;q=0.9,en;q=0.8" --
header="Referer: https://www.kaggle.com/"
"https://storage.googleapis.com/kagglesdsdata/datasets/2480666/4207867
/HR-Employee-Attrition.csv?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-
Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com
%2F20221103%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-
Date=20221103T181341Z&X-Goog-Expires=259200&X-Goog-
SignedHeaders=host&X-Goog-
Signature=264e877f03a05285f506857b3a20d01daa94cdd94d6b400883b7179a9969
ad8d803469a3858533ff4388c87f40bf88c1cf0761725c5e4dff483bc912a73a771ba8
0fea6058ffaf014847b014d82e05174c5a6a435361d90958d0a731e194ed2a8772e1d2
6dee8809b4804222b0c89971e422b442de3bae904f20af357c8db3899d0018e5e0add7
4afc610f33029c94df63205b9c34e4663fb001a291ff37d756e710de7ffe888d435290
735f26aa69065565cd57c040a699abaae98c409eda0a9ff543289883cfa6b4e47024f5
90618cd3bb90394a0d76b3446f231f4dd5da931babe7db5131974d5b8249f8757725f2
04248c48c944fbbaf0b1906fa5ea1cb4" -c -O 'HR-Employee-Attrition.csv'
```

```
--2022-11-03 18:17:57--
https://storage.googleapis.com/kagglesdsdata/datasets/2480666/4207867/
HR-Employee-Attrition.csv?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-
Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com
%2F20221103%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-
Date=20221103T181341Z&X-Goog-Expires=259200&X-Goog-
SignedHeaders=host&X-Goog-
Signature=264e877f03a05285f506857b3a20d01daa94cdd94d6b400883b7179a9969
ad8d803469a3858533ff4388c87f40bf88c1cf0761725c5e4dff483bc912a73a771ba8
0fea6058ffaf014847b014d82e05174c5a6a435361d90958d0a731e194ed2a8772e1d2
6dee8809b4804222b0c89971e422b442de3bae904f20af357c8db3899d0018e5e0add7
4afc610f33029c94df63205b9c34e4663fb001a291ff37d756e710de7ffe888d435290
735f26aa69065565cd57c040a699abaae98c409eda0a9ff543289883cfa6b4e47024f5
90618cd3bb90394a0d76b3446f231f4dd5da931babe7db5131974d5b8249f8757725f2
04248c48c944fbbaf0b1906fa5ea1cb4
Resolving storage.googleapis.com (storage.googleapis.com)...
64.233.189.128, 108.177.97.128, 108.177.125.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|
64.233.189.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 227977 (223K) [text/csv]
Saving to: ‘HR-Employee-Attrition.csv’

HR-Employee-Attriti 100%[===================>] 222.63K   536KB/s    in
0.4s

2022-11-03 18:17:59 (536 KB/s) - ‘HR-Employee-Attrition.csv’ saved
[227977/227977]
```

Exploratory Data Analysis (EDA)

```python
# importing required libraries / packages

%matplotlib inline

import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
import re
import pickle
import os

nltk.download('vader_lexicon')
```

```
from sklearn import preprocessing
from scipy. sparse import csr_matrix
from xgboost import XGBClassifier
from sklearn import tree
from nltk.corpus import stopwords
from sklearn.tree import DecisionTreeClassifier
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from collections import Counter
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix, roc_curve, auc,
ConfusionMatrixDisplay, RocCurveDisplay, roc_auc_score,
accuracy_score, classification_report
from sklearn.model_selection import train_test_split, cross_val_score,
KFold, StratifiedKFold, GridSearchCV, RandomizedSearchCV
from sklearn import metrics
from tqdm import tqdm
from sklearn.preprocessing import Normalizer, StandardScaler,
MinMaxScaler, OneHotEncoder
from scipy.sparse import hstack
from sklearn.naive_bayes import MultinomialNB, ComplementNB,
GaussianNB, CategoricalNB
from sklearn.svm import SVC
from scipy.stats import randint
from prettytable import PrettyTable
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

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

Shape of our Data

```
df.shape
```

```
(1470, 35)
```

Observation:

We have a total of 1470 observarions and 35 columns

```
''' Understanding each column's Data type'''
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
```

```
df.head()
```

```
   Age Attrition     BusinessTravel  DailyRate              Department
\
0   41       Yes      Travel_Rarely       1102                   Sales

1   49        No  Travel_Frequently        279  Research & Development

2   37       Yes      Travel_Rarely       1373  Research & Development

3   33        No  Travel_Frequently       1392  Research & Development

4   27        No      Travel_Rarely        591  Research & Development
```

```
   DistanceFromHome  Education EducationField  EmployeeCount  
EmployeeNumber  \
0                 1          2  Life Sciences              1  
1   
1                 8          1  Life Sciences              1  
2   
2                 2          2          Other              1  
4   
3                 3          4  Life Sciences              1  
5   
4                 2          1        Medical              1  
7   

   ...  RelationshipSatisfaction StandardHours  StockOptionLevel  \
0  ...                         1            80                 0   
1  ...                         4            80                 1   
2  ...                         2            80                 0   
3  ...                         3            80                 0   
4  ...                         4            80                 1   

   TotalWorkingYears  TrainingTimesLastYear WorkLifeBalance  
YearsAtCompany  \
0                  8                      0               1  
6   
1                 10                      3               3  
10   
2                  7                      3               3  
0   
3                  8                      3               3  
8   
4                  6                      3               3  
2   

  YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
0                  4                        0                     5
1                  7                        1                     7
2                  0                        0                     0
3                  7                        3                     0
4                  2                        2                     2

[5 rows x 35 columns]
```

```
''' Checking for null values in our entire Data'''

df.isna().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
```

Observation:

1. We dont have any Null values in our Data

```
''' Checking for duplicated values'''
df.duplicated().sum()
```

```
0
```

Observaton:

1. We dont have any duplicated values in our Data

```
''' A complete table that includes Unique, Null, Type of the columns 
and Null Percentage in each column'''
tabel = pd.DataFrame({
    'Unique':df.nunique(),
    'Null':df.isna().sum(),
    'NullPercent':df.isna().sum() / len(df),
    'Type':df.dtypes.values
```

```
})
display(tabel)
```

```
                          Unique  Null  NullPercent    Type
Age                           43     0          0.0   int64
Attrition                      2     0          0.0  object
BusinessTravel                 3     0          0.0  object
DailyRate                    886     0          0.0   int64
Department                     3     0          0.0  object
DistanceFromHome              29     0          0.0   int64
Education                      5     0          0.0   int64
EducationField                 6     0          0.0  object
EmployeeCount                  1     0          0.0   int64
EmployeeNumber              1470     0          0.0   int64
EnvironmentSatisfaction        4     0          0.0   int64
Gender                         2     0          0.0  object
HourlyRate                    71     0          0.0   int64
JobInvolvement                 4     0          0.0   int64
JobLevel                       5     0          0.0   int64
JobRole                        9     0          0.0  object
JobSatisfaction                4     0          0.0   int64
MaritalStatus                  3     0          0.0  object
MonthlyIncome               1349     0          0.0   int64
MonthlyRate                 1427     0          0.0   int64
NumCompaniesWorked            10     0          0.0   int64
Over18                         1     0          0.0  object
OverTime                       2     0          0.0  object
PercentSalaryHike             15     0          0.0   int64
PerformanceRating              2     0          0.0   int64
RelationshipSatisfaction       4     0          0.0   int64
StandardHours                  1     0          0.0   int64
StockOptionLevel               4     0          0.0   int64
TotalWorkingYears             40     0          0.0   int64
TrainingTimesLastYear          7     0          0.0   int64
WorkLifeBalance                4     0          0.0   int64
YearsAtCompany                37     0          0.0   int64
YearsInCurrentRole            19     0          0.0   int64
YearsSinceLastPromotion       16     0          0.0   int64
YearsWithCurrManager          18     0          0.0   int64
```

```
pd.set_option('display.max_columns', None) # to show all the columns
in our Data

df.head()
```

```
   Age Attrition     BusinessTravel  DailyRate              Department
\
0   41       Yes      Travel_Rarely       1102                   Sales

1   49        No  Travel_Frequently        279  Research & Development
```

```
2   37       Yes      Travel_Rarely       1373  Research & Development

3   33        No  Travel_Frequently       1392  Research & Development

4   27        No      Travel_Rarely        591  Research & Development


   DistanceFromHome  Education EducationField  EmployeeCount
EmployeeNumber  \
0                 1          2  Life Sciences              1
1
1                 8          1  Life Sciences              1
2
2                 2          2          Other              1
4
3                 3          4  Life Sciences              1
5
4                 2          1        Medical              1
7

   EnvironmentSatisfaction  Gender  HourlyRate  JobInvolvement
JobLevel  \
0                        2  Female          94               3
2
1                        3    Male          61               2
2
2                        4    Male          92               2
1
3                        4  Female          56               3
1
4                        1    Male          40               3
1

                 JobRole  JobSatisfaction MaritalStatus  MonthlyIncome
\
0        Sales Executive                4        Single           5993

1     Research Scientist                2       Married           5130

2  Laboratory Technician                3        Single           2090

3     Research Scientist                3       Married           2909

4  Laboratory Technician                2       Married           3468


   MonthlyRate  NumCompaniesWorked Over18 OverTime  PercentSalaryHike
\
0        19479                   8      Y      Yes                 11
```

```
1        24907                   1      Y       No                 23

2         2396                   6      Y      Yes                 15

3        23159                   1      Y      Yes                 11

4        16632                   9      Y       No                 12

   PerformanceRating  RelationshipSatisfaction  StandardHours  \
0                  3                         1             80
1                  4                         4             80
2                  3                         2             80
3                  3                         3             80
4                  3                         4             80

   StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
0                 0                  8                      0
1                 1                 10                      3
2                 0                  7                      3
3                 0                  8                      3
4                 1                  6                      3

   WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
0                1               6                   4
1                3              10                   7
2                3               0                   0
3                3               8                   7
4                3               2                   2

   YearsSinceLastPromotion  YearsWithCurrManager
0                        0                     5
1                        1                     7
2                        0                     0
3                        3                     0
4                        2                     2
```

Description of each column in our Data:

1. ***Age***: Age of the employee.
2. ***Attrition***: Employee attrition.
3. ***BusinessTravel***: How frequently an employee travels for business purpose.
4. ***DailyRate***: Daily wage of an employee.
5. ***Department***: Employee department.
6. ***DistanceFromHome***: Distance form home to office in KM's.
7. ***Education***: Qualification of employee (masked).
8. ***EducationField***: Stream of Education.

9. **EmployeeCount**: Employee count.
10. ***EmployeeNumber***: Employee number.
11. ***EnvironmentSatisfaction***: Show us how employess are satisfied with the work environment.
12. ***Gender***: Gender of employee.
13. ***HourlyRate***: Employee hourly rate.
14. ***JobInvolvement***: How much a eomployee is involved in their job.
15. ***JobLevel***: Level of Job.
16. ***JobRole***: Job role of an employee.
17. ***JobSatisfaction***: If employee is satisfied?
18. ***MaritalStatus***: Employee is married or not.
19. ***MonthlyIncome***: Income of an employee.
20. **MonthlyRate**: Monthly rate of an employee.
21. ***NumCompaniesWorked***: Number of companies worked for.
22. ***Over18***: Age over 18.
23. ***OverTime***: employee works over time.
24. ***PercentSalaryHike***: Salary hike.
25. ***PerformanceRating***: Performance rate.
26. ***RelationshipSatisfaction***: How much the eomployee is satisfied in their relationship
27. ***StandardHours***: per week standard work hours.
28. ***StockOptionLevel***: company stock option level.
29. ***TotalWorkingYears***: Total working years.
30. ***TrainingTimesLastYear***: Time spent on training by Employee
31. ***WorkLifeBalance***: Work life balance.
32. ***YearsAtCompany***: Total years at current company.
33. ***YearsInCurrentRole***: Total years in current role.
34. ***YearsSinceLastPromotion***: Years since last promotion.
35. ***YearsWithCurrManager***: Years worked under current manager.

```
df[["Age", "DailyRate", "DistanceFromHome", "HourlyRate",
"MonthlyIncome", "NumCompaniesWorked", "PercentSalaryHike",
"YearsAtCompany"]]\
  .describe(include="all") # knowing some statistics on certain
important columns of our Data
```

```
               Age    DailyRate  DistanceFromHome   HourlyRate
MonthlyIncome  \
count  1470.000000  1470.000000       1470.000000  1470.000000
1470.000000
mean     36.923810   802.485714          9.192517    65.891156
6502.931293
std       9.135373   403.509100          8.106864    20.329428
4707.956783
min      18.000000   102.000000          1.000000    30.000000
```

```
1009.000000
25%      30.000000   465.000000          2.000000    48.000000
2911.000000
50%      36.000000   802.000000          7.000000    66.000000
4919.000000
75%      43.000000  1157.000000         14.000000    83.750000
8379.000000
max      60.000000  1499.000000         29.000000   100.000000
19999.000000

       NumCompaniesWorked  PercentSalaryHike  YearsAtCompany
count         1470.000000        1470.000000     1470.000000
mean             2.693197          15.209524        7.008163
std              2.498009           3.659938        6.126525
min              0.000000          11.000000        0.000000
25%              1.000000          12.000000        3.000000
50%              2.000000          14.000000        5.000000
75%              4.000000          18.000000        9.000000
max              9.000000          25.000000       40.000000
```

Observations:

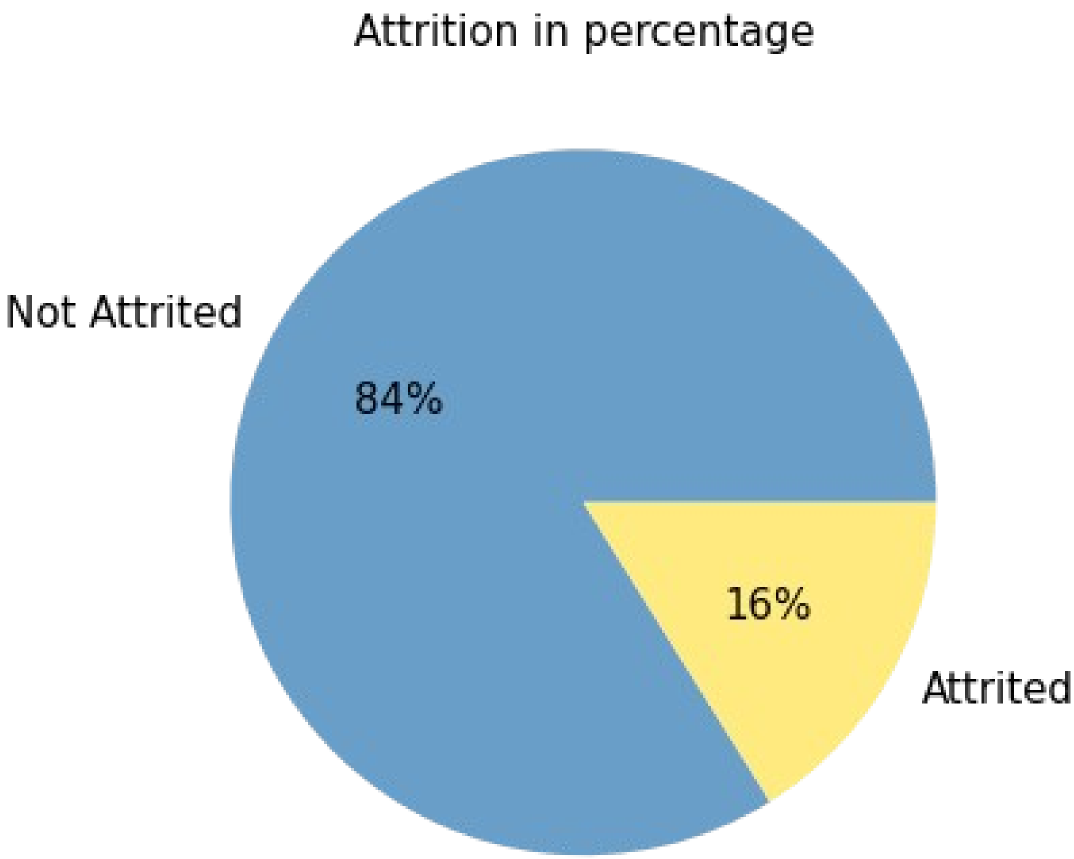
1. ***Age***: We have employees ranging from 18 to 60 years and since we have all the employess greater than 18years, we can remove the column **Over18**
2. ***DailyRate***: Daily rate of the employess is ranging from 102 to 1499.
3. ***DistanceFromHome***: We have employess who are travelling around 30km to reach the office.
4. ***HourlyRate***: Hourlt rate of employee is ranging from 20 to 100
5. ***MonthlyIncome***: Employees are earning a minimum of 1009 to maximum of 19999 per month.
6. ***NumCompaniesWorked***: We have employess who has not at all changed any company and on the other we have some of them changed 9 companies during their carrier.
7. ***PercentSalaryHike***: Employees are getting a minimum hike of 11% to a maximum hike of 25%.
8. ***YearsAtCompany***: We have employess who are working with the current company from 40 years which is very good sign that the company is very much liked by the employees.

```
df.drop(['EmployeeCount','EmployeeNumber','StandardHours','Over18'],ax
is=1,inplace=True)
# these columns doenot add much value to the Data, hence we are
removing them from our Analysis
```

```
df.Attrition.value_counts()
```

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

```
slices = df["Attrition"].value_counts()

plt.figure(figsize=(8, 6))
plt.pie(x=slices, labels=["Not Attrited", "Attrited"],
colors=["#699ec9", "#ffea80"], autopct="%1.0f%%",
textprops={'fontsize': 15})
plt.title("Attrition in percentage", fontdict = {'fontsize': 15})


plt.show()
```

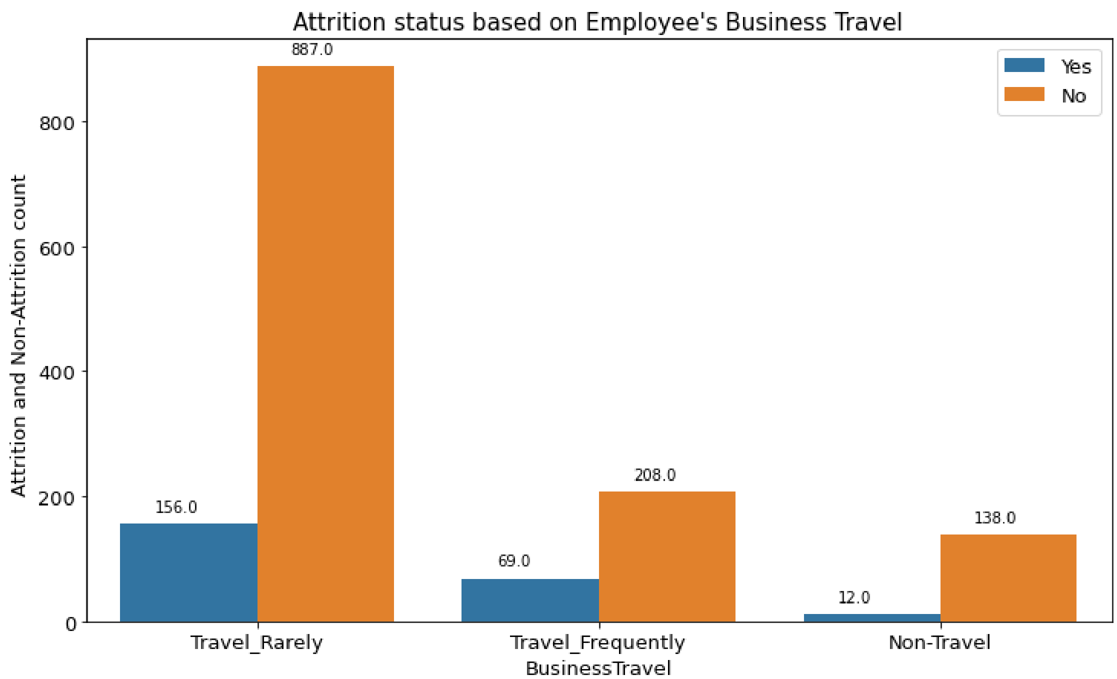


Observaton:

1. We can observer that 16% of the eomployees are effected by the Attrition which is a total of 237 employees

```
plt.figure(figsize=(12, 7))
ax = sns.countplot(data = df, x='BusinessTravel', hue='Attrition')
#ax.bar_label(ax.containers[0], label_type='edge')

plt.legend(fontsize = 13)
```

```
plt.xlabel("BusinessTravel", fontdict = {'fontsize': 13})
plt.xticks(size=13)

plt.ylabel("Attrition and Non-Attrition count ", fontdict =
{'fontsize': 13})
plt.yticks(size=13)

plt.title("Attrition status based on Employee's Business Travel",
fontdict = {'fontsize': 15})

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1,
p.get_height()+20))
plt.show()
```



Observations:

1. Employees who has got oppurtunity to go on a Business travel are having very much lower Attition rate when compared to people are going on a Business travel.
2. Based on Business travel around 156 people who Travelled Rarely has been Attrited and it is the highest Attrition among all the other types of Business travel.

```
attrition_yes = df[(df['Attrition'] == 'Yes')]

attrition_yes.head()
```

```
    Age Attrition BusinessTravel  DailyRate              Department  \
0    41       Yes  Travel_Rarely       1102                   Sales   
2    37       Yes  Travel_Rarely       1373  Research & Development   
14   28       Yes  Travel_Rarely        103  Research & Development   
```

```
21   36       Yes  Travel_Rarely       1218                  Sales
24   34       Yes  Travel_Rarely        699  Research & Development

    DistanceFromHome  Education EducationField
EnvironmentSatisfaction  \
0                  1          2  Life Sciences
2
2                  2          2          Other
4
14                24          3  Life Sciences
3
21                 9          4  Life Sciences
3
24                 6          1        Medical
2

    Gender  HourlyRate  JobInvolvement  JobLevel
JobRole  \
0   Female          94               3         2        Sales
Executive
2     Male          92               2         1  Laboratory
Technician
14    Male          50               2         1  Laboratory
Technician
21    Male          82               2         1   Sales
Representative
24    Male          83               3         1     Research
Scientist

    JobSatisfaction MaritalStatus  MonthlyIncome  MonthlyRate  \
0                 4        Single           5993        19479
2                 3        Single           2090         2396
14                3        Single           2028        12947
21                1        Single           3407         6986
24                1        Single           2960        17102

    NumCompaniesWorked OverTime  PercentSalaryHike  PerformanceRating
\
0                    8      Yes                 11                  3

2                    6      Yes                 15                  3

14                   5      Yes                 14                  3

21                   7       No                 23                  4

24                   2       No                 11                  3
```

```
    RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
0                          1                 0                  8
2                          2                 0                  7
14                         2                 0                  6
21                         2                 0                 10
24                         3                 0                  8

    TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany  \
0                       0                1               6
2                       3                3               0
14                      4                3               4
21                      4                3               5
24                      2                3               4

    YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager

0                    4                        0                     5

2                    0                        0                     0

14                   2                        0                     3

21                   3                        0                     3

24                   2                        1                     3
```

```
attrition_yes.Age.value_counts()
```

```
31    18
29    18
28    14
33    12
26    12
32    11
35    10
34     9
30     9
24     7
21     6
44     6
37     6
19     6
41     6
25     6
20     6
39     6
36     6
40     5
50     5
22     5
```

```
58    5
23    4
46    4
18    4
56    3
55    3
27    3
52    3
47    3
48    2
45    2
42    2
38    2
51    2
49    2
43    2
53    2
Name: Age, dtype: int64
```

```
plt.figure(figsize=(12, 7))
sns.countplot(data = attrition_yes, x='Age')

plt.xlabel("Age", fontdict = {'fontsize': 14})
plt.xticks(size=12)

plt.ylabel("Attition ", fontdict = {'fontsize': 14})
plt.yticks(size=12)

plt.title("Number of Employees Attrited based on their Age", fontdict
= {'fontsize': 15})
plt.show()
```

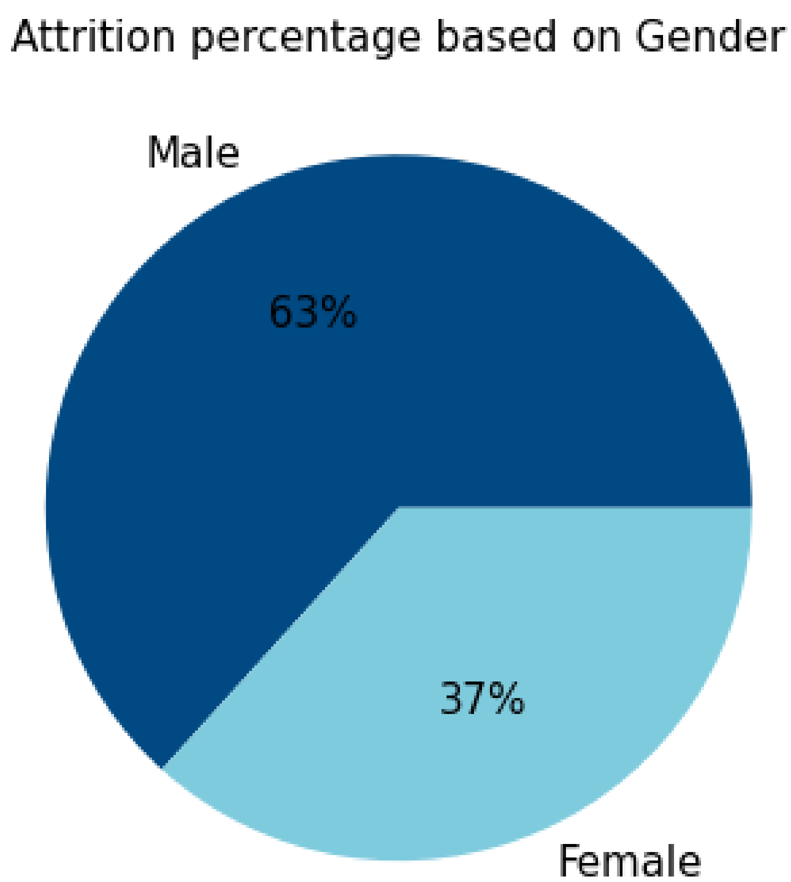
Observaitons:

1. Nearly 18 Employees have been Attrited who are having age of 31 and 29, these are the highest age groups in which Attrition was high.
2. Lowest Attrition rate was observed with the Employees who are having age in the range of 48 to 53. 2 Emplyees have been Attrited among each of these age group people.

```
attrition_yes.Gender.value_counts()
```

```
Male      150
Female     87
Name: Gender, dtype: int64
```

```
slices = attrition_yes.Gender.value_counts()

plt.figure(figsize=(8, 6))
plt.pie(x = slices, labels = ["Male", "Female"],
colors=['#004982','#7fcbde'], autopct="%1.0f%%",
textprops={'fontsize': 15})
plt.title("Attrition percentage based on Gender", fontdict =
{'fontsize': 15})


plt.show()
```

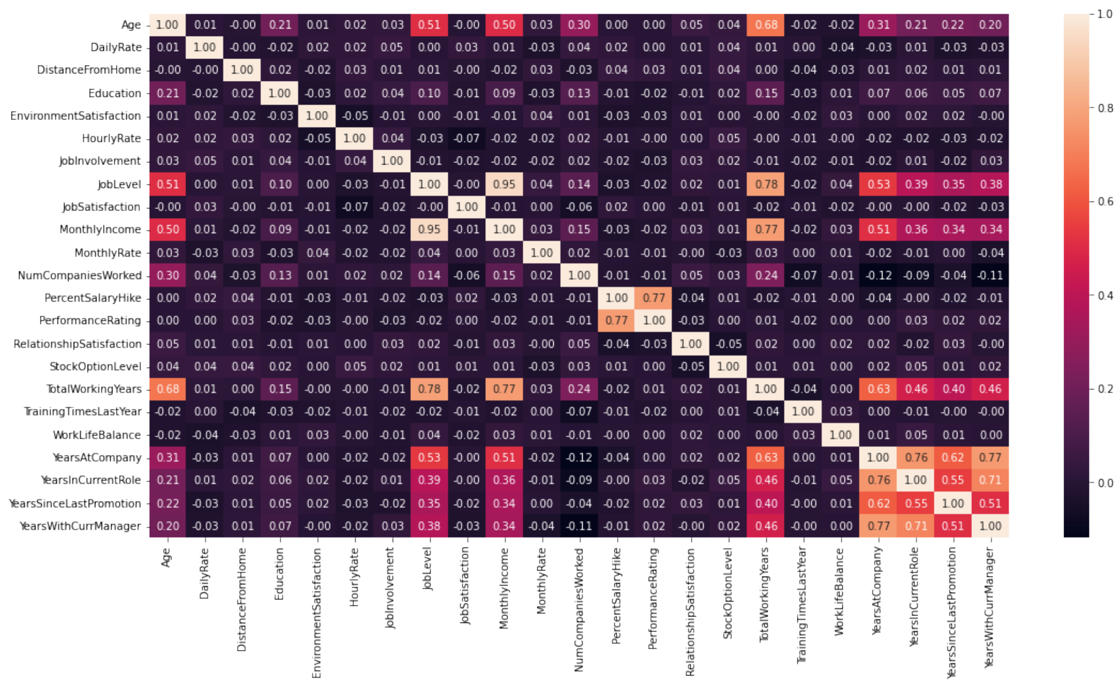



```
df.head()

   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales
1   49        No  Travel_Frequently        279  Research & Development
2   37       Yes      Travel_Rarely       1373  Research & Development
3   33        No  Travel_Frequently       1392  Research & Development
4   27        No      Travel_Rarely        591  Research & Development

   DistanceFromHome  Education EducationField  EnvironmentSatisfaction  \
0                 1          2  Life Sciences                        2
1                 8          1  Life Sciences                        3
```

```
2                 2          2         Other                       4

3                 3          4  Life Sciences                       4

4                 2          1       Medical                       1


   Gender  HourlyRate  JobInvolvement  JobLevel                JobRole
\
0  Female          94               3         2        Sales Executive

1    Male          61               2         2     Research Scientist

2    Male          92               2         1  Laboratory Technician

3  Female          56               3         1     Research Scientist

4    Male          40               3         1  Laboratory Technician


   JobSatisfaction MaritalStatus  MonthlyIncome  MonthlyRate  \
0                4        Single           5993        19479
1                2       Married           5130        24907
2                3        Single           2090         2396
3                3       Married           2909        23159
4                2       Married           3468        16632

   NumCompaniesWorked OverTime  PercentSalaryHike
PerformanceRating  \
0                   8      Yes                 11                  3

1                   1       No                 23                  4

2                   6      Yes                 15                  3

3                   1      Yes                 11                  3

4                   9       No                 12                  3


   RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
0                         1                 0                  8
1                         4                 1                 10
2                         2                 0                  7
3                         3                 0                  8
4                         4                 1                  6
```

```
   TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany
YearsInCurrentRole  \
0                      0                1               6
4
1                      3                3              10
7
2                      3                3               0
0
3                      3                3               8
7
4                      3                3               2
2

   YearsSinceLastPromotion  YearsWithCurrManager
0                        0                     5
1                        1                     7
2                        0                     0
3                        3                     0
4                        2                     2
```

```
''' Plotting a correlation map for all numeric variables '''
plt.figure(figsize=(18,9))
sns.heatmap(df.corr(), annot=True, fmt='.2f');
```



Observation:

1. Only some of the features are highly correlated with each other and whereas others are having very much less corelation.

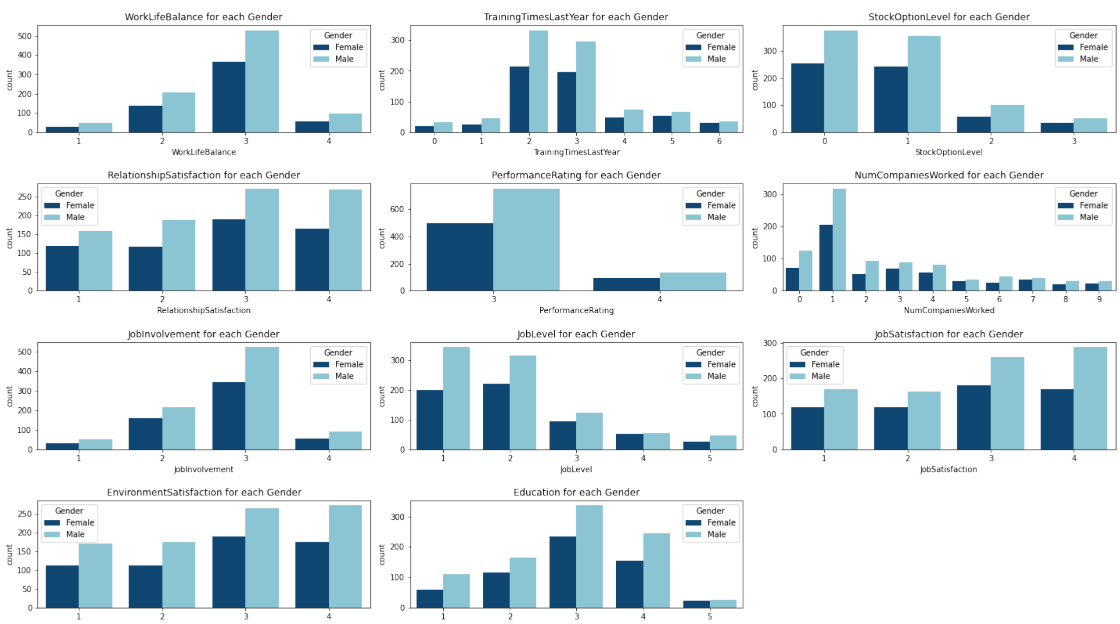
```
df.head()
```

```
   Age Attrition     BusinessTravel  DailyRate              Department
\
0   41       Yes      Travel_Rarely       1102                   Sales

1   49        No  Travel_Frequently        279  Research & Development

2   37       Yes      Travel_Rarely       1373  Research & Development

3   33        No  Travel_Frequently       1392  Research & Development

4   27        No      Travel_Rarely        591  Research & Development


   DistanceFromHome  Education EducationField  EnvironmentSatisfaction
\
0                 1          2  Life Sciences                        2

1                 8          1  Life Sciences                        3

2                 2          2          Other                        4

3                 3          4  Life Sciences                        4

4                 2          1        Medical                        1


   Gender  HourlyRate  JobInvolvement  JobLevel                JobRole
\
0  Female          94               3         2        Sales Executive

1    Male          61               2         2     Research Scientist

2    Male          92               2         1  Laboratory Technician

3  Female          56               3         1     Research Scientist

4    Male          40               3         1  Laboratory Technician


   JobSatisfaction MaritalStatus  MonthlyIncome  MonthlyRate  \
0                4        Single           5993        19479
1                2       Married           5130        24907
2                3        Single           2090         2396
3                3       Married           2909        23159
4                2       Married           3468        16632

   NumCompaniesWorked OverTime  PercentSalaryHike
PerformanceRating  \
```

```
0                   8      Yes                 11                   3

1                   1       No                 23                   4

2                   6      Yes                 15                   3

3                   1      Yes                 11                   3

4                   9       No                 12                   3


   RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
0                         1                 0                  8
1                         4                 1                 10
2                         2                 0                  7
3                         3                 0                  8
4                         4                 1                  6

   TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany
YearsInCurrentRole  \
0                      0                1               6
4
1                      3                3              10
7
2                      3                3               0
0
3                      3                3               8
7
4                      3                3               2
2

   YearsSinceLastPromotion  YearsWithCurrManager
0                        0                     5
1                        1                     7
2                        0                     0
3                        3                     0
4                        2                     2
```

```
''' Plots for various columns in our Data'''

columns =
['WorkLifeBalance','TrainingTimesLastYear','StockOptionLevel',

'RelationshipSatisfaction','PerformanceRating','NumCompaniesWorked',
   'JobInvolvement', 'JobLevel', 'JobSatisfaction',
   'EnvironmentSatisfaction','Education']

plt.figure(figsize=(20, 25))
```

```
for i,col in enumerate(columns):
    axes = plt.subplot(9, 3, i + 1)
    #sns.countplot(x=df[col], hue=df['Gender'],
palette=['#ED72A3','#8565F0'])
    sns.countplot(x = df[col], hue = df['Gender'], palette =
['#004982','#7fcbde']) # https://www.color-hex.com/color-palettes/?
keyword=blue (selecting color)
    plt.title(str(col)+" for each Gender")

# to adjust spacing between subplots
plt.subplots_adjust(left=0.1,
                    bottom=0.7,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
plt.tight_layout()
plt.show()
```



Observations:

1. ***Work Life Balance***:

Around 900 people has rated that they have work life balance as 3 which is good number as the highest rating given was 4 and the least ratig is 1.

1. ***Trainings attended***:

Aroung 1100 people has attended 2/3 trainings last year which is a very good number and by this we can understand that most of the eomplyees are interested in attending a good amount of ratings rather than attending more number of trainings.

1. ***Rating***:

Aroung 1100 people has got 3 rating where as lesser number of people got 4 rating. We can reach out to the employees who has got less ratings and try to solve any issues they find in order to improve their ratings.

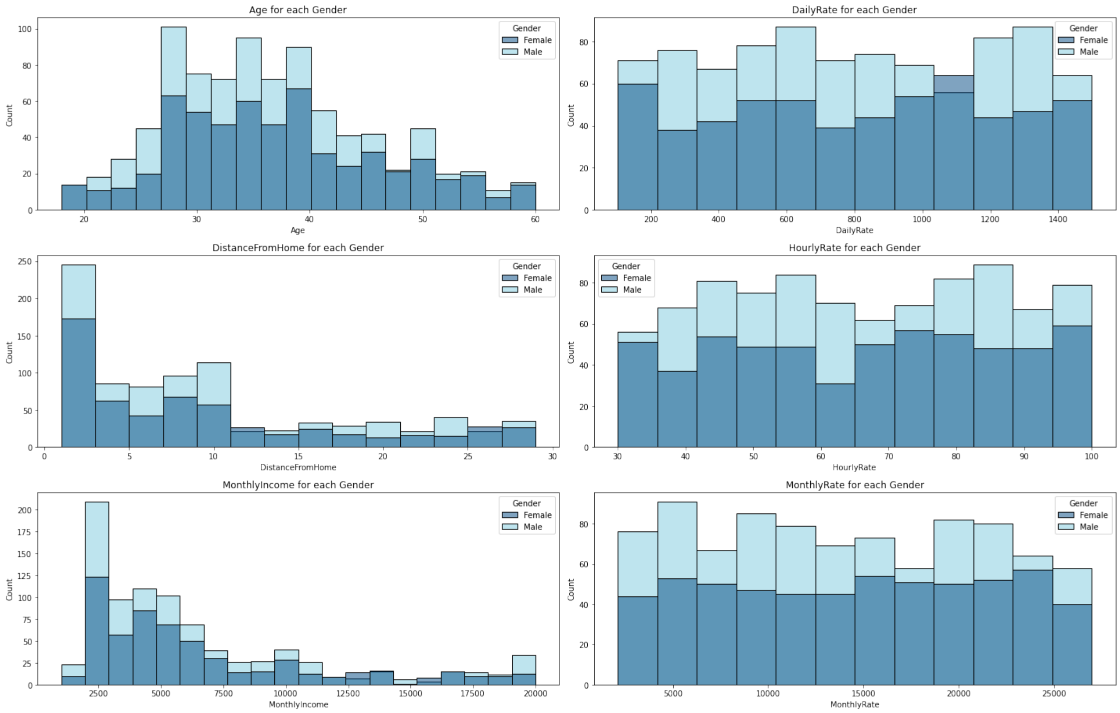
1. ***Job Satisfaction***:

    Around 950 of the employees rated their job satisfaction as 3 & 4 which a very good thing for a company and ther services that company is providing for their Employees

2. ***Enrivonment Satisfaction***:

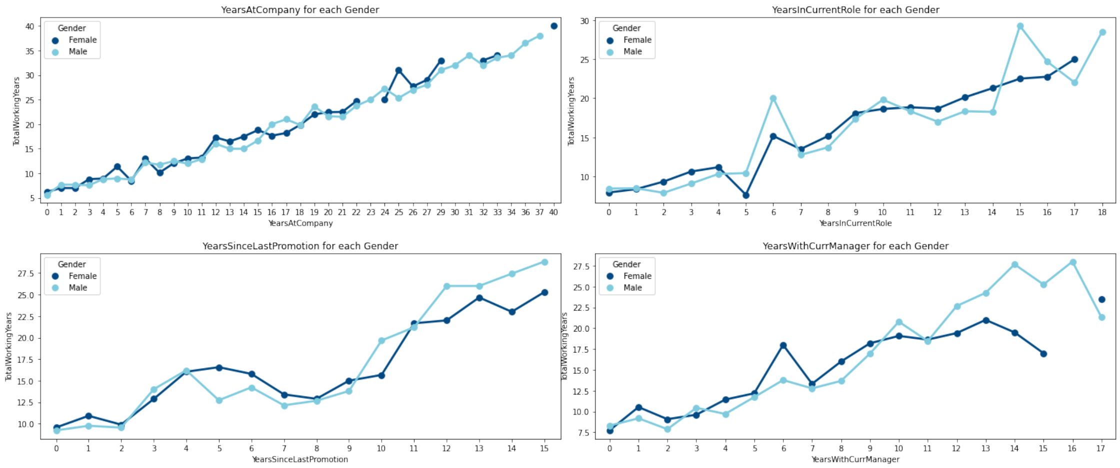
Nearly 1200 Employees satisfied with their place of work and this will indirectly motivate them to work more peacefully and happily.

```
hist = ['Age', 'DailyRate','DistanceFromHome', 'HourlyRate',
'MonthlyIncome', 'MonthlyRate']

plt.figure(figsize=(20,25))
for i,col in enumerate(hist):
    axes = plt.subplot(6,2, i + 1)
    sns.histplot(x=df[col], hue=df['Gender'],
palette=['#004982','#7fcbde'])
    plt.title(str(col)+" for each Gender")

plt.subplots_adjust(left=0.1,
                    bottom=0.7,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
plt.tight_layout()
plt.tight_layout()
plt.show()
```



Correlation between Years

```
hist = ['YearsAtCompany','YearsInCurrentRole',
'YearsSinceLastPromotion','YearsWithCurrManager']

plt.figure(figsize=(20,25))

for i,col in enumerate(hist):
    axes = plt.subplot(6,2, i + 1)
    sns.pointplot(x = df[col], y = df['TotalWorkingYears'], hue =
df['Gender'], palette = ['#004982','#7fcbde'], ci = None)
    plt.title(str(col)+" for each Gender")

plt.subplots_adjust(left=0.1,
                    bottom=0.7,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
plt.tight_layout()
plt.show()
```

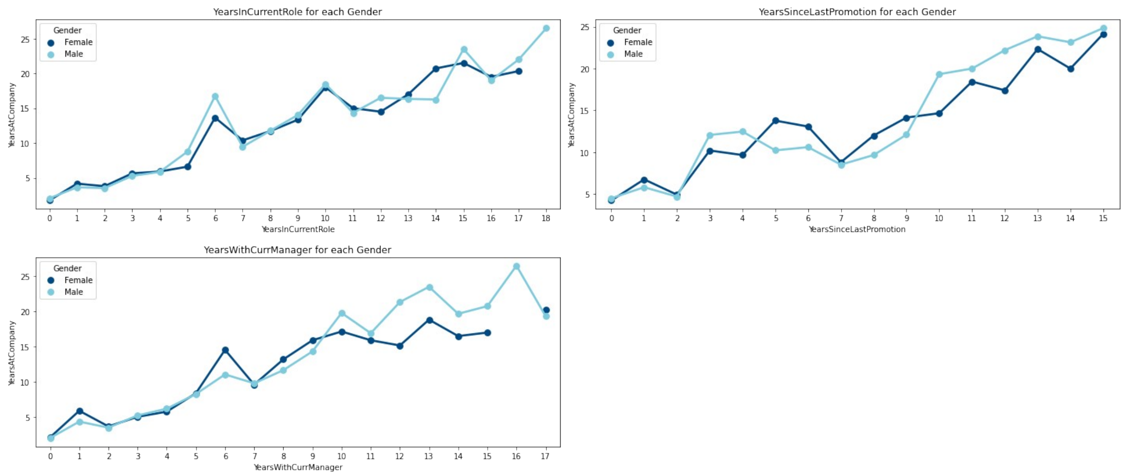


Observations:

1. There is a Liner relation shop between Number of Year at company to the Total working years. But when compared to Male employees there are some Female employees who has stayed in the same comapany for more number of years.
2. There are more Male Employees who has stayed in the same role for more years when compared to the Female Employees.
3. Male Employees who are tending to have more experience are staying with their current manager for number of years when compared to the people who are having lesser experience.

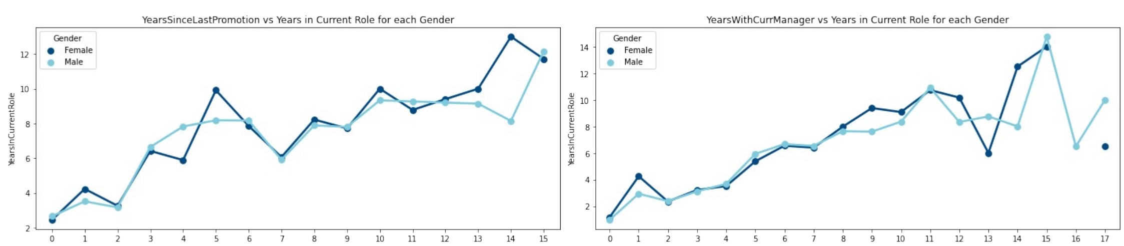
Correlation between Years in the Company

```
columns = ['YearsInCurrentRole',
'YearsSinceLastPromotion','YearsWithCurrManager']

plt.figure(figsize=(20,25))

for i,col in enumerate(columns):
    axes = plt.subplot(6,2, i + 1)
    sns.pointplot(x = df[col], y = df['YearsAtCompany'], hue =
df['Gender'], palette = ['#004982','#7fcbde'], ci = None)
    plt.title(str(col)+" for each Gender")

plt.subplots_adjust(left=0.1,
                    bottom=0.7,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
plt.tight_layout()
plt.show()
```



Correlation between Years in Current role

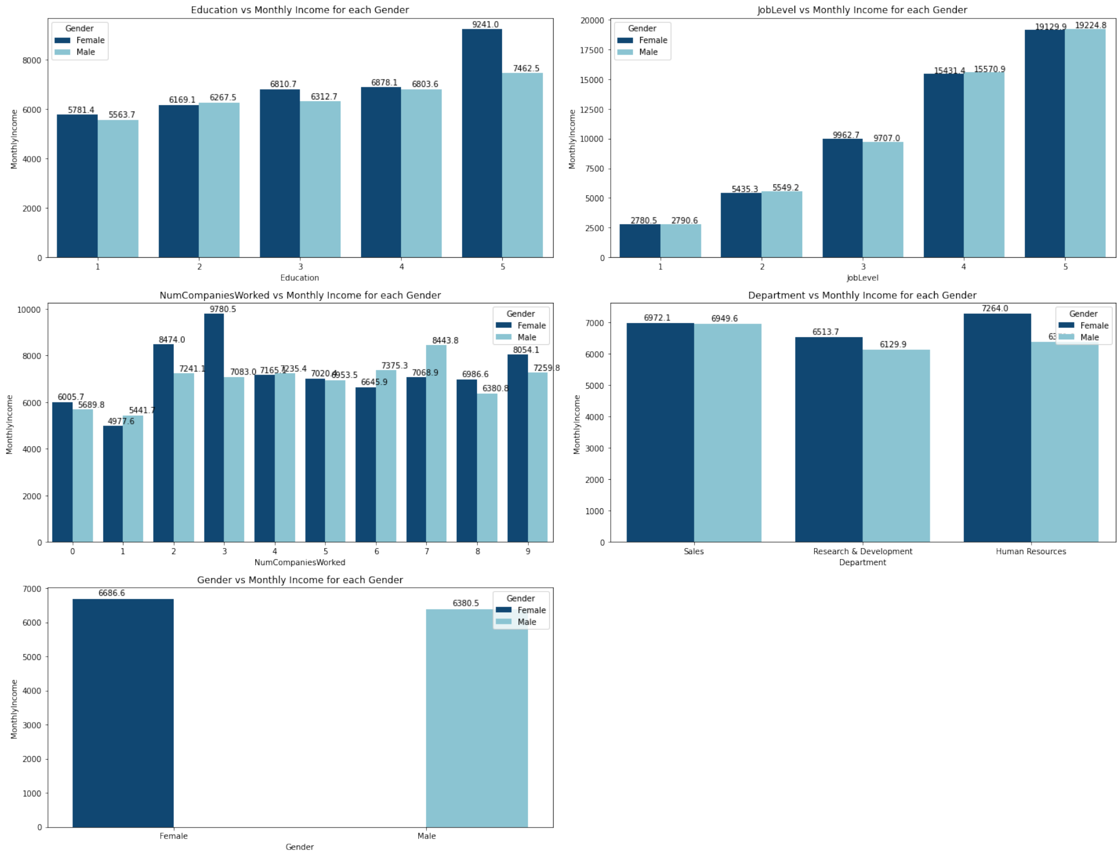
```
columns = ['YearsSinceLastPromotion','YearsWithCurrManager']

plt.figure(figsize=(20,25))

for i,col in enumerate(columns):
    axes = plt.subplot(6,2, i + 1)
    sns.pointplot(x = df[col], y = df['YearsInCurrentRole'], hue =
df['Gender'], palette = ['#004982','#7fcbde'], ci = None)
    plt.title(str(col)+" vs Years in Current Role for each Gender")

plt.subplots_adjust(left=0.1,
                    bottom=0.7,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
plt.tight_layout()
plt.show()
```



Observaitons:

1. There are more Female Employees who are working in the same role without getting a promotion when compared with the Female Employees.
2. There are some Male Employees who are working for more years with their current Manager.

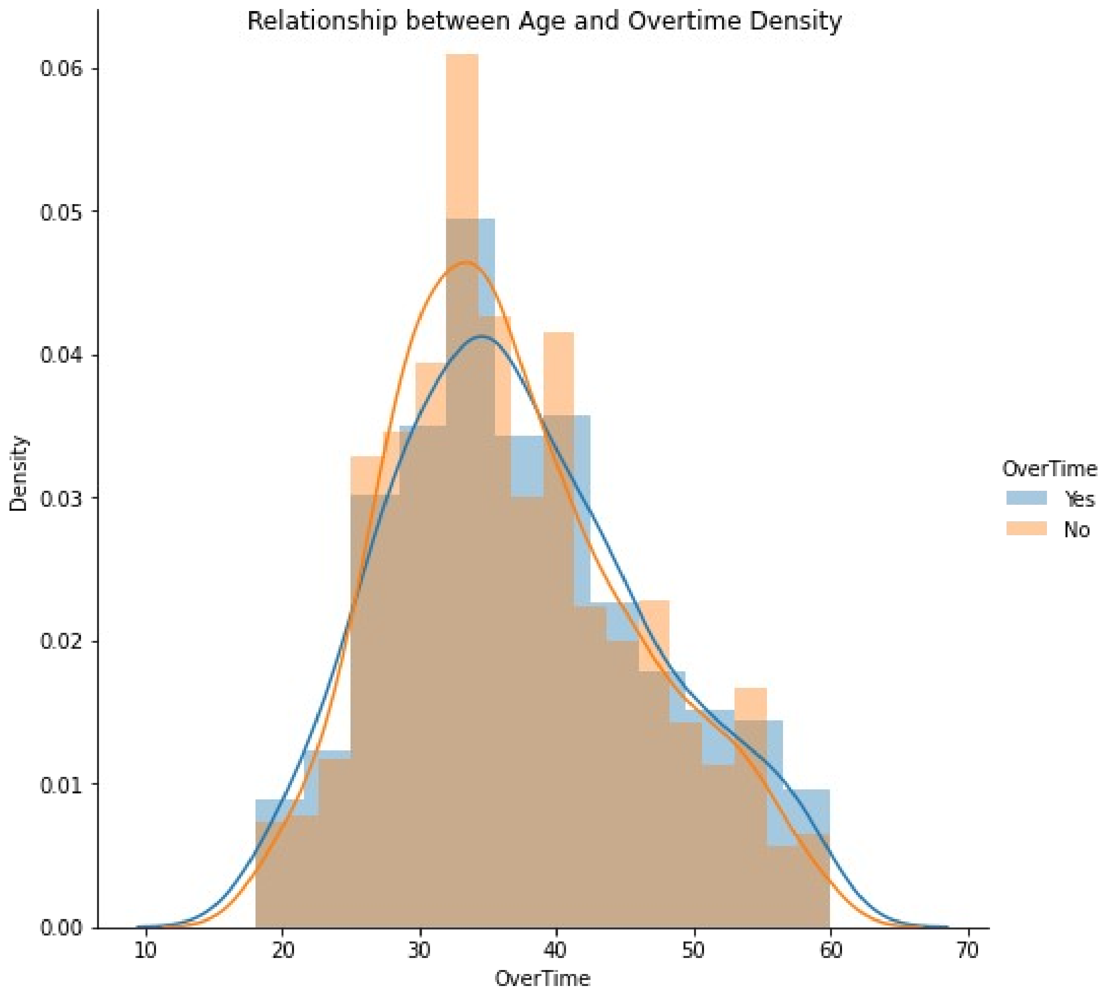
```
columns = ['Education','JobLevel','NumCompaniesWorked', 'Department',
'Gender']
#colunas =
['Education','JobLevel','NumCompaniesWorked','TotalWorkingYears','Year
sAtCompany','YearsInCurrentRole',
'YearsSinceLastPromotion','YearsWithCurrManager']

plt.figure(figsize=(20, 30))

for i,col in enumerate(columns):
    plt.subplot(6, 2, i + 1)
    axes = sns.barplot(x = df[col], y = df['MonthlyIncome'],
hue=df['Gender'], palette = ['#004982','#7fcbde'], ci=None)
    plt.title(str(col)+" vs Monthly Income for each Gender")
    plt.xticks(fontsize = 10)

    for p in axes.patches:
        axes.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1,
p.get_height()+100))

plt.subplots_adjust(left=0.1,
                    bottom=0.7,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
plt.tight_layout()
plt.show()
```



Observations:

1. ***Gender***: Monthly income for Female is greater then Male.
2. ***Education***: Employees who are have a Job and Education level 5 are having higher income when compared to other Education levels.
3. ***Department***: People of HR department are getting slighlty higher Monthly income when compared to other departments like Sales and R&D.

```
fg = sns.FacetGrid(df, hue="OverTime",size = 7)\
     .map(sns.distplot,"Age")\
     .add_legend()

fg.fig.suptitle('Relationship between Age and Overtime Density'); # 
adding title
plt.xlabel("OverTime")
plt.show();
```

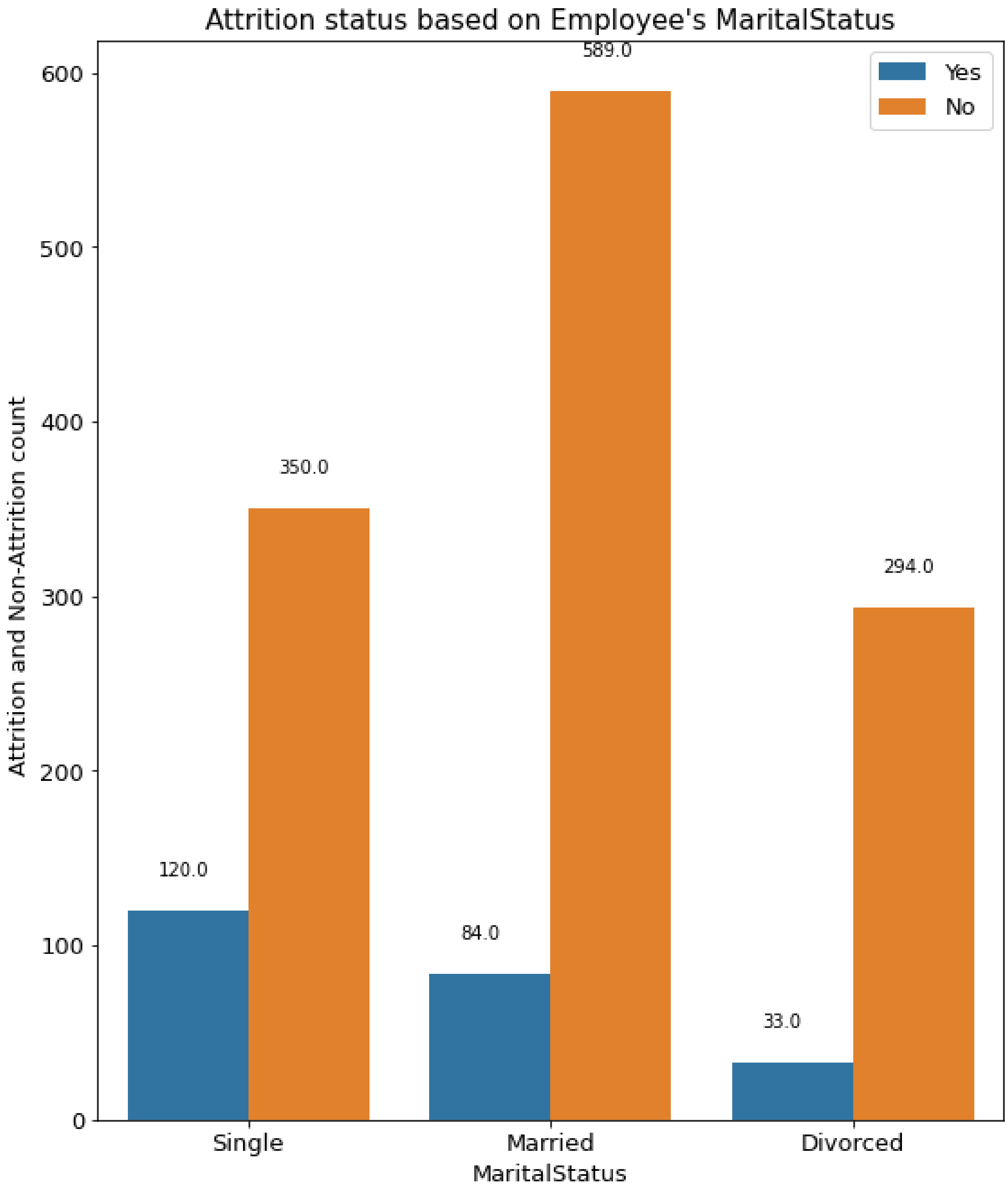


Observation:

1. We can observe that employees around the age of 30 to 40 are working overtime when compared to the other age groups.

```
df.MaritalStatus.unique()

array(['Single', 'Married', 'Divorced'], dtype=object)

plt.figure(figsize=(9, 11))
ax = sns.countplot(data = df, x = 'MaritalStatus', hue = 'Attrition')
#ax.bar_label(ax.containers[0], label_type='edge')

plt.legend(fontsize = 13)

plt.xlabel("MaritalStatus", fontdict = {'fontsize': 13})
plt.xticks(size=13)

plt.ylabel("Attrition and Non-Attrition count ", fontdict =
{'fontsize': 13})
plt.yticks(size=13)
```

```
plt.title("Attrition status based on Employee's MaritalStatus",
fontdict = {'fontsize': 15})

for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1,
p.get_height()+20))
plt.show()
```



Observations:

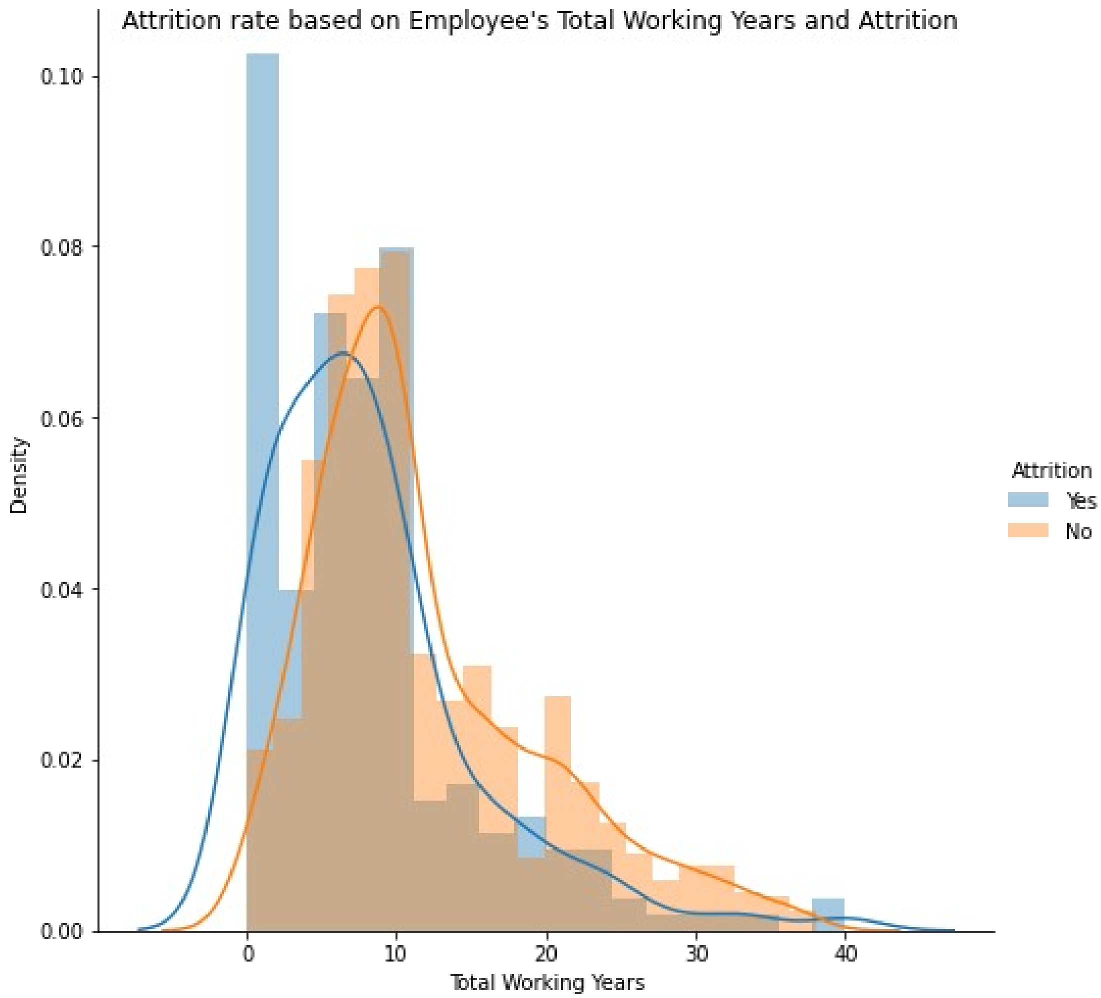
1. We can clearly understand that around 120 employees who are having Maritial Status as Single are Attrited more when compared other 2 Maritial status

```
df.head()

   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales   

1   49        No  Travel_Frequently        279  Research & Development   

2   37       Yes      Travel_Rarely       1373  Research & Development   

3   33        No  Travel_Frequently       1392  Research & Development   

4   27        No      Travel_Rarely        591  Research & Development   


   DistanceFromHome  Education EducationField  EnvironmentSatisfaction  \
0                 1          2  Life Sciences                        2   

1                 8          1  Life Sciences                        3   

2                 2          2          Other                        4   

3                 3          4  Life Sciences                        4   

4                 2          1        Medical                        1   


   Gender  HourlyRate  JobInvolvement  JobLevel                JobRole  \
0  Female          94               3         2        Sales Executive   

1    Male          61               2         2     Research Scientist   

2    Male          92               2         1  Laboratory Technician   

3  Female          56               3         1     Research Scientist   

4    Male          40               3         1  Laboratory Technician   


   JobSatisfaction MaritalStatus  MonthlyIncome  MonthlyRate  \
0                4        Single           5993        19479   
1                2       Married           5130        24907   
2                3        Single           2090         2396   
```

```
3                 3       Married           2909        23159
4                 2       Married           3468        16632

   NumCompaniesWorked OverTime  PercentSalaryHike
PerformanceRating  \
0                    8      Yes                 11                  3

1                    1       No                 23                  4

2                    6      Yes                 15                  3

3                    1      Yes                 11                  3

4                    9       No                 12                  3


   RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
0                         1                 0                  8
1                         4                 1                 10
2                         2                 0                  7
3                         3                 0                  8
4                         4                 1                  6

   TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany
YearsInCurrentRole  \
0                      0                1               6
4
1                      3                3              10
7
2                      3                3               0
0
3                      3                3               8
7
4                      3                3               2
2

   YearsSinceLastPromotion  YearsWithCurrManager
0                        0                     5
1                        1                     7
2                        0                     0
3                        3                     0
4                        2                     2
```

```
fg = sns.FacetGrid(df, hue = "Attrition",size = 7)\
     .map(sns.distplot, "TotalWorkingYears")\
     .add_legend()

fg.fig.suptitle("Attrition rate based on Employee's Total Working
Years and Attrition "); # adding title
```

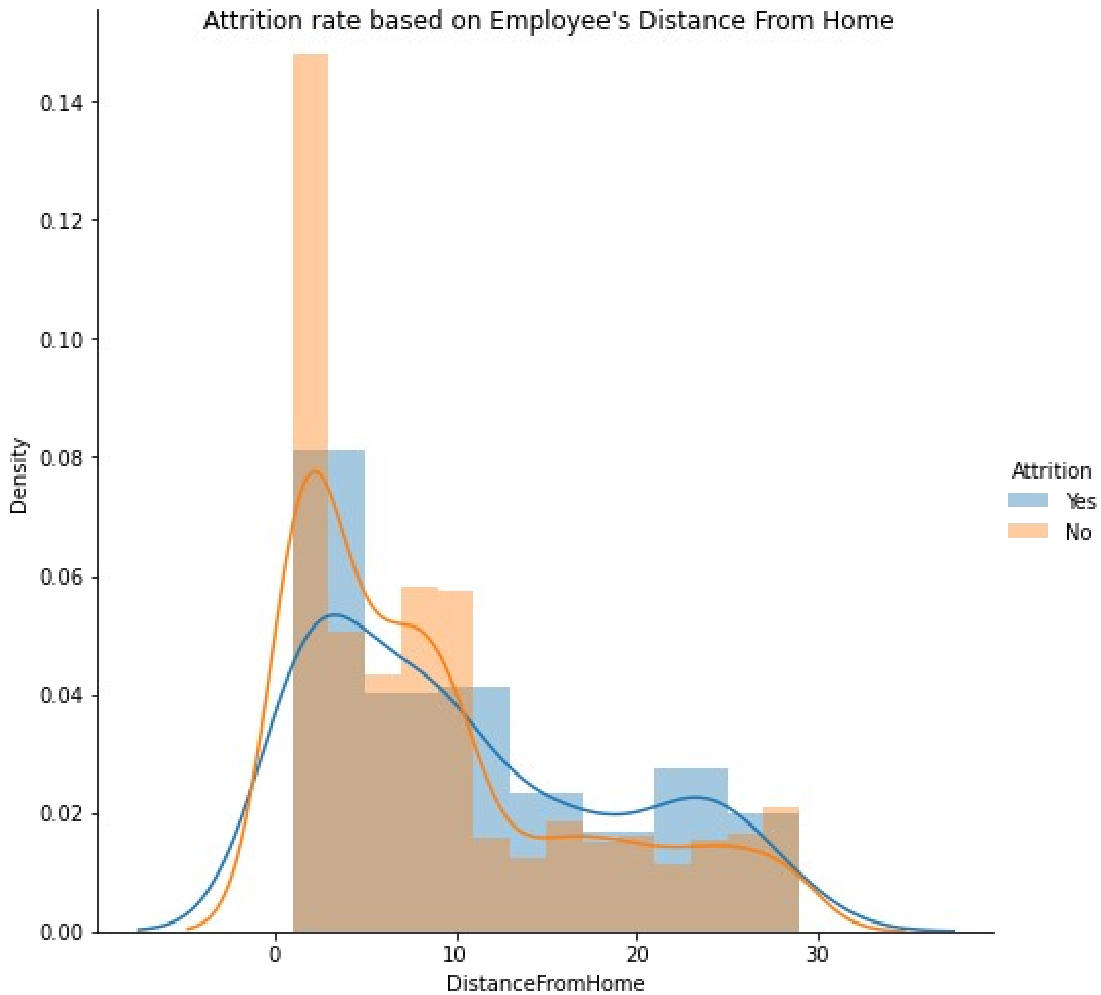
```
plt.xlabel("Total Working Years")
plt.show();
```



Observation:

1. From this we can understand that Employees who are having Total working years around 0 are the ones who has been effected by Attrition.

```
fg = sns.FacetGrid(df, hue = "Attrition",size = 7)\
     .map(sns.distplot, "DistanceFromHome")\
     .add_legend()

fg.fig.suptitle(" Attrition rate based on Employee's Distance From
Home"); # adding title
plt.xlabel("DistanceFromHome")
plt.show();
```

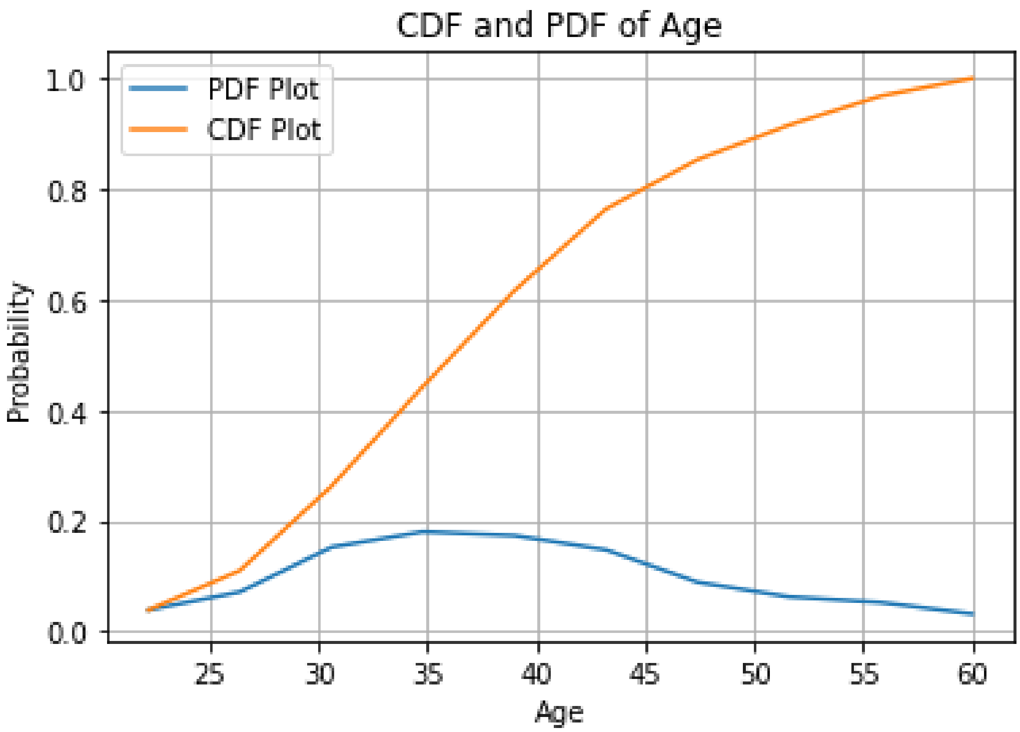


Observation:

1. Based on Employee's Distance from Home we cant clearly conclude their Attrition. However, Employees who have distance from 0 to 5 are Attrited when compared to other distances.

```
# compute pdf

counts, bin_edges = np.histogram(df["Age"],bins=10)
pdf  = counts/sum(counts)
#print(pdf)
#print(bin_edges)

#compute cdf
cdf = np.cumsum(pdf)
#print(cdf)

#plotting pdf nd cdf
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
```

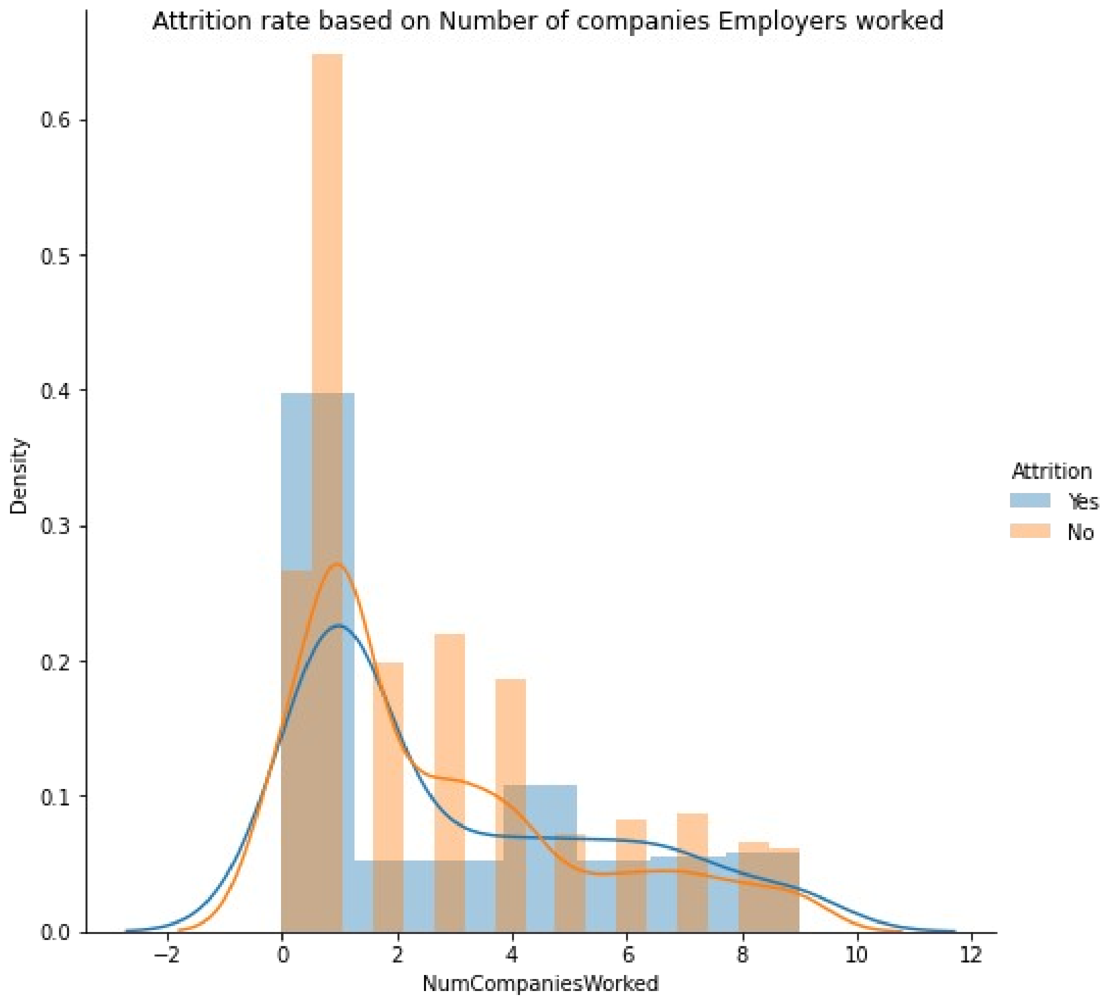
```
plt.grid()
plt.title("CDF and PDF of Age")
plt.gca().legend(('PDF Plot','CDF Plot'))
plt.xlabel("Age")
plt.ylabel("Probability")
plt.show()
```



Observation:

1. There are more Employees who are having age around 35 and nearly 80% of the them are having Age less than or equal to 45.

```
fg = sns.FacetGrid(df, hue = "Attrition",size = 7)\
     .map(sns.distplot, "NumCompaniesWorked")\
     .add_legend()

fg.fig.suptitle(" Attrition rate based on Number of companies
Employers worked"); # adding title
plt.xlabel("NumCompaniesWorked")
plt.show();
```



Observation:

1. Employees who worked in 1 company are mostly effected by the Attrition, this could be their first company and hence they have count as 1.

Overall Observations from EDA

1. Age of the Employee's is ranging from 18 to 60 years and their Daily rate is ranging from 102 to 1499, its make their Monthly rate ranging from 1009 to a maximum of 19999.
2. Around 16% which is equal to 237 employees got Attrited and out of which 37% are Females and 63% Males.
3. People who travelled rarely on a Business travel and the ones who have their maritialstatus as singlre are the ones who got mostly effected by Attriton. And Business travel can be a useful feature inorder to find the potential people who are at the risk of Attrition.
4. Even the employees who are having Age around 30 to 35 hours got effected by the Attrition which suggests us that Atttition doesnot depend on person's Age.

5. The ones who are having Education and Job level as 5 are the ones who are having higher Monthly income when compared with others.
6. Employees who has worked in 1 company or the ones who has Total working years as 1 are the ones who got effected by Attrition and this could be due to the fact that company is prefering more Senior colleagues to be retained when compared to the Juniors.
7. Daily rate and Monthly rate for Female employees is greater then Male employees.

Data Preprocessing

```
df.dtypes.unique()
```

```
array([dtype('int64'), dtype('O')], dtype=object)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 31 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   EnvironmentSatisfaction   1470 non-null   int64 
 9   Gender                    1470 non-null   object
 10  HourlyRate                1470 non-null   int64 
 11  JobInvolvement            1470 non-null   int64 
 12  JobLevel                  1470 non-null   int64 
 13  JobRole                   1470 non-null   object
 14  JobSatisfaction           1470 non-null   int64 
 15  MaritalStatus             1470 non-null   object
 16  MonthlyIncome             1470 non-null   int64 
 17  MonthlyRate               1470 non-null   int64 
 18  NumCompaniesWorked        1470 non-null   int64 
 19  OverTime                  1470 non-null   object
 20  PercentSalaryHike         1470 non-null   int64 
 21  PerformanceRating         1470 non-null   int64 
 22  RelationshipSatisfaction  1470 non-null   int64 
 23  StockOptionLevel          1470 non-null   int64 
 24  TotalWorkingYears         1470 non-null   int64 
 25  TrainingTimesLastYear     1470 non-null   int64 
 26  WorkLifeBalance           1470 non-null   int64 
 27  YearsAtCompany            1470 non-null   int64 
```

```
 28  YearsInCurrentRole        1470 non-null   int64
 29  YearsSinceLastPromotion   1470 non-null   int64
 30  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(23), object(8)
memory usage: 356.1+ KB
```

```
print("Categorical variables present in our Data are: ")
print("*"*46)
list(df.select_dtypes(['object']).columns)
```

```
Categorical variables present in our Data are: 
**********************************************

['Attrition',
 'BusinessTravel',
 'Department',
 'EducationField',
 'Gender',
 'JobRole',
 'MaritalStatus',
 'OverTime']
```

```
print("Numerical variables present in our Data are: ")
print("*"*44)
list(df.select_dtypes(['int64']).columns)
```

```
Numerical variables present in our Data are: 
********************************************

['Age',
 'DailyRate',
 'DistanceFromHome',
 'Education',
 'EnvironmentSatisfaction',
 'HourlyRate',
 'JobInvolvement',
 'JobLevel',
 'JobSatisfaction',
 'MonthlyIncome',
 'MonthlyRate',
 'NumCompaniesWorked',
 'PercentSalaryHike',
 'PerformanceRating',
 'RelationshipSatisfaction',
 'StockOptionLevel',
 'TotalWorkingYears',
 'TrainingTimesLastYear',
 'WorkLifeBalance',
 'YearsAtCompany',
 'YearsInCurrentRole',
```

```
 'YearsSinceLastPromotion',
 'YearsWithCurrManager']
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 31 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   EnvironmentSatisfaction   1470 non-null   int64 
 9   Gender                    1470 non-null   object
 10  HourlyRate                1470 non-null   int64 
 11  JobInvolvement            1470 non-null   int64 
 12  JobLevel                  1470 non-null   int64 
 13  JobRole                   1470 non-null   object
 14  JobSatisfaction           1470 non-null   int64 
 15  MaritalStatus             1470 non-null   object
 16  MonthlyIncome             1470 non-null   int64 
 17  MonthlyRate               1470 non-null   int64 
 18  NumCompaniesWorked        1470 non-null   int64 
 19  OverTime                  1470 non-null   object
 20  PercentSalaryHike         1470 non-null   int64 
 21  PerformanceRating         1470 non-null   int64 
 22  RelationshipSatisfaction  1470 non-null   int64 
 23  StockOptionLevel          1470 non-null   int64 
 24  TotalWorkingYears         1470 non-null   int64 
 25  TrainingTimesLastYear     1470 non-null   int64 
 26  WorkLifeBalance           1470 non-null   int64 
 27  YearsAtCompany            1470 non-null   int64 
 28  YearsInCurrentRole        1470 non-null   int64 
 29  YearsSinceLastPromotion   1470 non-null   int64 
 30  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(23), object(8)
memory usage: 356.1+ KB
```

```
df.head(2)
```

```
   Age Attrition     BusinessTravel  DailyRate              Department
\
0   41       Yes      Travel_Rarely       1102                   Sales

1   49        No  Travel_Frequently        279  Research & Development
```

```
   DistanceFromHome  Education EducationField  EnvironmentSatisfaction
\
0                 1          2  Life Sciences                        2

1                 8          1  Life Sciences                        3


   Gender  HourlyRate  JobInvolvement  JobLevel          JobRole  \
0  Female          94               3         2     Sales Executive
1    Male          61               2         2  Research Scientist

   JobSatisfaction MaritalStatus  MonthlyIncome  MonthlyRate  \
0                4        Single           5993        19479
1                2       Married           5130        24907

   NumCompaniesWorked OverTime  PercentSalaryHike
PerformanceRating  \
0                   8      Yes                 11                 3

1                   1       No                 23                 4


   RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
0                         1                 0                  8
1                         4                 1                 10

   TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany
YearsInCurrentRole  \
0                      0                1               6
4
1                      3                3              10
7

   YearsSinceLastPromotion  YearsWithCurrManager
0                        0                     5
1                        1                     7
```

```
''' Replacing Yes and No values in Attrition column with 1 and 0
respectively'''
# 1 - Attrited
# 0 - Not Attrited
df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})

df.head(2)
```

```
   Age  Attrition     BusinessTravel  DailyRate
Department  \
0   41          1      Travel_Rarely       1102
```

```
Sales
1   49          0  Travel_Frequently        279  Research &
Development

   DistanceFromHome  Education EducationField  EnvironmentSatisfaction
\
0                 1          2  Life Sciences                        2

1                 8          1  Life Sciences                        3


   Gender  HourlyRate  JobInvolvement  JobLevel             JobRole  \
0  Female          94               3         2     Sales Executive
1    Male          61               2         2  Research Scientist

   JobSatisfaction MaritalStatus  MonthlyIncome  MonthlyRate  \
0                4        Single           5993        19479
1                2       Married           5130        24907

   NumCompaniesWorked OverTime  PercentSalaryHike
PerformanceRating  \
0                   8      Yes                 11                  3

1                   1       No                 23                  4


   RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
0                         1                 0                  8
1                         4                 1                 10

   TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany
YearsInCurrentRole  \
0                      0                1               6
4
1                      3                3              10
7

   YearsSinceLastPromotion  YearsWithCurrManager
0                        0                     5
1                        1                     7
```

```
df.head(2)
```

```
   Age  Attrition     BusinessTravel  DailyRate
Department  \
0   41          1      Travel_Rarely       1102
Sales
1   49          0  Travel_Frequently        279  Research &
Development
```

```
   DistanceFromHome  Education EducationField  EnvironmentSatisfaction  
\
0                 1          2  Life Sciences                        2   

1                 8          1  Life Sciences                        3   


   Gender  HourlyRate  JobInvolvement  JobLevel            JobRole  \
0  Female          94               3         2    Sales Executive   
1    Male          61               2         2  Research Scientist   

   JobSatisfaction MaritalStatus  MonthlyIncome  MonthlyRate  \
0                4        Single           5993        19479   
1                2       Married           5130        24907   

   NumCompaniesWorked OverTime  PercentSalaryHike  
PerformanceRating  \
0                   8      Yes                 11                  3   

1                   1       No                 23                  4   


   RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
0                         1                 0                  8   
1                         4                 1                 10   

   TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany  
YearsInCurrentRole  \
0                      0                1               6   
4   
1                      3                3              10   
7   

   YearsSinceLastPromotion  YearsWithCurrManager  
0                        0                     5  
1                        1                     7  
```

```
df.JobRole.unique()
```

```
array(['Sales Executive', 'Research Scientist', 'Laboratory 
Technician',
       'Manufacturing Director', 'Healthcare Representative', 
'Manager',
       'Sales Representative', 'Research Director', 'Human 
Resources'],
      dtype=object)
```

Observations from Data Preprocessing:

1. We dont have to perform preprocessing because for the categorical columns we have fewer number of unique values and we can handle them using Onehot encoding or Response encoding.
2. For the Numerical columns we have to Normalize them inorder get the advantages on Normalized Data and also by doing this we can have the values scale independent, so that even if in future Data is collected in a different scale Model will not have any effect.

Model Building

```
df.head(2)

   Age  Attrition     BusinessTravel  DailyRate
Department  \
0   41          1      Travel_Rarely       1102
Sales
1   49          0  Travel_Frequently        279  Research &
Development

   DistanceFromHome  Education EducationField  EnvironmentSatisfaction
\
0                 1          2  Life Sciences                        2

1                 8          1  Life Sciences                        3


   Gender  HourlyRate  JobInvolvement  JobLevel          JobRole  \
0  Female          94               3         2  Sales Executive
1    Male          61               2         2  Research Scientist

   JobSatisfaction MaritalStatus  MonthlyIncome  MonthlyRate  \
0                4        Single           5993        19479
1                2       Married           5130        24907

   NumCompaniesWorked OverTime  PercentSalaryHike
PerformanceRating  \
0                   8      Yes                 11                 3

1                   1       No                 23                 4


   RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
0                         1                 0                  8
1                         4                 1                 10

   TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany
YearsInCurrentRole  \
```

```
0                       0                1                6
4
1                       3                3               10
7

   YearsSinceLastPromotion  YearsWithCurrManager
0                        0                     5
1                        1                     7
```

```
y = df['Attrition'].values # storing the values in the column 'Attrition' in a variable 'y'
x = df.drop(['Attrition'], axis = 1)  # droping the column 'Attrition' from our original data

y = pd.DataFrame(y)
x = pd.DataFrame(x)

print(x.shape)
print(y.shape)
```

```
(1470, 30)
(1470, 1)
```

Splitting the Data into Train and test

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, stratify = y )
# splitting the data into train, test with test data= 30% of values and  stratify on 'Y' label

print(" Number of rows and columns in Training data ",x_train.shape)
print(" Number of rows and columns in Test data ",x_test.shape)
```

```
 Number of rows and columns in Training data  (1029, 30)
 Number of rows and columns in Test data  (441, 30)
```

Normalization of Numerical Features

```
data_normalization = Normalizer()

#************************************************************************************************#

# Normalization of 'Age' column
Age_train = data_normalization.fit_transform(x_train[['Age']])
Age_test = data_normalization.transform(x_test[['Age']])

Age_train = csr_matrix(Age_train)
Age_test = csr_matrix(Age_test)


#************************************************************************************************#
```

```
# Normalization of 'DailyRate' column
DailyRate_train =
data_normalization.fit_transform(x_train[['DailyRate']])
DailyRate_test = data_normalization.transform(x_test[['DailyRate']])

DailyRate_train = csr_matrix(DailyRate_train)
DailyRate_test = csr_matrix(DailyRate_test)

#*********************************************************************
***********************************#

# Normalization of 'DistanceFromHome' column
DistanceFromHome_train =
data_normalization.fit_transform(x_train[['DistanceFromHome']])
DistanceFromHome_test =
data_normalization.transform(x_test[['DistanceFromHome']])

DistanceFromHome_train = csr_matrix(DistanceFromHome_train)
DistanceFromHome_test = csr_matrix(DistanceFromHome_test)

#*********************************************************************
***********************************#

# Normalization of 'Education' column
Education_train =
data_normalization.fit_transform(x_train[['Education']])
Education_test = data_normalization.transform(x_test[['Education']])

Education_train = csr_matrix(Education_train)
Education_test = csr_matrix(Education_test)

#*********************************************************************
***********************************#

# Normalization of 'EnvironmentSatisfaction' column
EnvironmentSatisfaction_train =
data_normalization.fit_transform(x_train[['EnvironmentSatisfaction']])
EnvironmentSatisfaction_test =
data_normalization.transform(x_test[['EnvironmentSatisfaction']])

EnvironmentSatisfaction_train =
csr_matrix(EnvironmentSatisfaction_train)
EnvironmentSatisfaction_test =
csr_matrix(EnvironmentSatisfaction_test)

#*********************************************************************
***********************************#
```

```
# Normalization of 'HourlyRate' column
HourlyRate_train =
data_normalization.fit_transform(x_train[['HourlyRate']])
HourlyRate_test = data_normalization.transform(x_test[['HourlyRate']])

HourlyRate_train = csr_matrix(HourlyRate_train)
HourlyRate_test = csr_matrix(HourlyRate_test)

#*****************************************************************************
***********************************#

# Normalization of 'JobInvolvement' column
JobInvolvement_train =
data_normalization.fit_transform(x_train[['JobInvolvement']])
JobInvolvement_test =
data_normalization.transform(x_test[['JobInvolvement']])

JobInvolvement_train = csr_matrix(JobInvolvement_train)
JobInvolvement_test = csr_matrix(JobInvolvement_test)

#*****************************************************************************
***********************************#

# Normalization of 'JobLevel' column
JobLevel_train =
data_normalization.fit_transform(x_train[['JobLevel']])
JobLevel_test = data_normalization.transform(x_test[['JobLevel']])

JobLevel_train = csr_matrix(JobLevel_train)
JobLevel_test = csr_matrix(JobLevel_test)

#*****************************************************************************
***********************************#

# Normalization of 'JobSatisfaction' column
JobSatisfaction_train =
data_normalization.fit_transform(x_train[['JobSatisfaction']])
JobSatisfaction_test =
data_normalization.transform(x_test[['JobSatisfaction']])

JobSatisfaction_train = csr_matrix(JobSatisfaction_train)
JobSatisfaction_test = csr_matrix(JobSatisfaction_test)

#*****************************************************************************
***********************************#

# Normalization of 'MonthlyIncome' column
MonthlyIncome_train =
data_normalization.fit_transform(x_train[['MonthlyIncome']])
```

```
MonthlyIncome_test =
data_normalization.transform(x_test[['MonthlyIncome']])

MonthlyIncome_train = csr_matrix(MonthlyIncome_train)
MonthlyIncome_test = csr_matrix(MonthlyIncome_test)

#****************************************************************
*********************************#
# Normalization of 'MonthlyRate' column
MonthlyRate_train =
data_normalization.fit_transform(x_train[['MonthlyRate']])
MonthlyRate_test =
data_normalization.transform(x_test[['MonthlyRate']])


MonthlyRate_train = csr_matrix(MonthlyRate_train)
MonthlyRate_test = csr_matrix(MonthlyRate_test)

#****************************************************************
*********************************#

# Normalization of 'NumCompaniesWorked' column
NumCompaniesWorked_train =
data_normalization.fit_transform(x_train[['NumCompaniesWorked']])
NumCompaniesWorked_test =
data_normalization.transform(x_test[['NumCompaniesWorked']])

NumCompaniesWorked_train = csr_matrix(NumCompaniesWorked_train)
NumCompaniesWorked_test = csr_matrix(NumCompaniesWorked_test)

#****************************************************************
*********************************#

# Normalization of 'PercentSalaryHike' column
PercentSalaryHike_train =
data_normalization.fit_transform(x_train[['PercentSalaryHike']])
PercentSalaryHike_test =
data_normalization.transform(x_test[['PercentSalaryHike']])

PercentSalaryHike_train = csr_matrix(PercentSalaryHike_train)
PercentSalaryHike_test = csr_matrix(PercentSalaryHike_test)


#****************************************************************
*********************************#

# Normalization of 'PerformanceRating' column
PerformanceRating_train =
data_normalization.fit_transform(x_train[['PerformanceRating']])
```

```
PerformanceRating_test =
data_normalization.transform(x_test[['PerformanceRating']])

PerformanceRating_train = csr_matrix(PerformanceRating_train)
PerformanceRating_test = csr_matrix(PerformanceRating_test)

#*****************************************************************************
*****************************************#

# Normalization of 'RelationshipSatisfaction' column
RelationshipSatisfaction_train =
data_normalization.fit_transform(x_train[['RelationshipSatisfaction']]
)
RelationshipSatisfaction_test =
data_normalization.transform(x_test[['RelationshipSatisfaction']])

RelationshipSatisfaction_train =
csr_matrix(RelationshipSatisfaction_train)
RelationshipSatisfaction_test =
csr_matrix(RelationshipSatisfaction_test)

#*****************************************************************************
*****************************************#

# Normalization of 'StockOptionLevel' column
StockOptionLevel_train =
data_normalization.fit_transform(x_train[['StockOptionLevel']])
StockOptionLevel_test =
data_normalization.transform(x_test[['StockOptionLevel']])

StockOptionLevel_train = csr_matrix(StockOptionLevel_train)
StockOptionLevel_test = csr_matrix(StockOptionLevel_test)


#*****************************************************************************
*****************************************#

# Normalization of 'TotalWorkingYears' column
TotalWorkingYears_train =
data_normalization.fit_transform(x_train[['TotalWorkingYears']])
TotalWorkingYears_test =
data_normalization.transform(x_test[['TotalWorkingYears']])

TotalWorkingYears_train = csr_matrix(TotalWorkingYears_train)
TotalWorkingYears_test = csr_matrix(TotalWorkingYears_test)


#*****************************************************************************
*****************************************#
```

```
# Normalization of 'TrainingTimesLastYear' column
TrainingTimesLastYear_train =
data_normalization.fit_transform(x_train[['TrainingTimesLastYear']])
TrainingTimesLastYear_test =
data_normalization.transform(x_test[['TrainingTimesLastYear']])

TrainingTimesLastYear_train = csr_matrix(TrainingTimesLastYear_train)
TrainingTimesLastYear_test = csr_matrix(TrainingTimesLastYear_test)

#*********************************************************************
***********************************#

# Normalization of 'WorkLifeBalance' column
WorkLifeBalance_train =
data_normalization.fit_transform(x_train[['WorkLifeBalance']])
WorkLifeBalance_test =
data_normalization.transform(x_test[['WorkLifeBalance']])

WorkLifeBalance_train = csr_matrix(WorkLifeBalance_train)
WorkLifeBalance_test = csr_matrix(WorkLifeBalance_test)

#*********************************************************************
***********************************#

# Normalization of 'YearsAtCompany' column
YearsAtCompany_train =
data_normalization.fit_transform(x_train[['YearsAtCompany']])
YearsAtCompany_test =
data_normalization.transform(x_test[['YearsAtCompany']])

YearsAtCompany_train = csr_matrix(YearsAtCompany_train)
YearsAtCompany_test = csr_matrix(YearsAtCompany_test)

#*********************************************************************
***********************************#

# Normalization of 'YearsInCurrentRole' column
YearsInCurrentRole_train =
data_normalization.fit_transform(x_train[['YearsInCurrentRole']])
YearsInCurrentRole_test =
data_normalization.transform(x_test[['YearsInCurrentRole']])

YearsInCurrentRole_train = csr_matrix(YearsInCurrentRole_train)
YearsInCurrentRole_test = csr_matrix(YearsInCurrentRole_test)

#*********************************************************************
***********************************#
```

```
# Normalization of 'YearsSinceLastPromotion' column
YearsSinceLastPromotion_train =
data_normalization.fit_transform(x_train[['YearsSinceLastPromotion']])
YearsSinceLastPromotion_test =
data_normalization.transform(x_test[['YearsSinceLastPromotion']])

YearsSinceLastPromotion_train =
csr_matrix(YearsSinceLastPromotion_train)
YearsSinceLastPromotion_test =
csr_matrix(YearsSinceLastPromotion_test)

#*********************************************************************
***********************************#

# Normalization of 'YearsWithCurrManager' column
YearsWithCurrManager_train =
data_normalization.fit_transform(x_train[['YearsWithCurrManager']])
YearsWithCurrManager_test =
data_normalization.transform(x_test[['YearsWithCurrManager']])

YearsSinceLastPromotion_train = csr_matrix(YearsWithCurrManager_train)
YearsWithCurrManager_test = csr_matrix(YearsWithCurrManager_test)

#*********************************************************************
***********************************#
```

OneHot encoding of Categorical variables

```
df.head(2)
```

```
   Age  Attrition     BusinessTravel  DailyRate
Department  \
0   41          1      Travel_Rarely       1102
Sales
1   49          0  Travel_Frequently        279  Research &
Development

   DistanceFromHome  Education EducationField  EnvironmentSatisfaction
\
0                 1          2  Life Sciences                        2

1                 8          1  Life Sciences                        3


   Gender  HourlyRate  JobInvolvement  JobLevel             JobRole  \
0  Female          94               3         2     Sales Executive
1    Male          61               2         2  Research Scientist

   JobSatisfaction MaritalStatus  MonthlyIncome  MonthlyRate  \
0                4        Single           5993        19479
```

```
1               2       Married          5130       24907

   NumCompaniesWorked OverTime  PercentSalaryHike
PerformanceRating  \
0                  8      Yes                11                 3

1                  1       No                23                 4


   RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
0                         1                 0                  8
1                         4                 1                 10

   TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany
YearsInCurrentRole  \
0                      0                1               6
4
1                      3                3              10
7

   YearsSinceLastPromotion  YearsWithCurrManager
0                        0                     5
1                        1                     7

ohe = OneHotEncoder(sparse=False )

# Performing OneHot encoding on BusinessTravel column

BusinessTravel_oneHot_tr =
ohe.fit_transform(x_train[['BusinessTravel']].values)
BusinessTravel_oneHot_te =
ohe.transform(x_test[['BusinessTravel']].values)

BusinessTravel_oneHot_tr = csr_matrix(BusinessTravel_oneHot_tr)
BusinessTravel_oneHot_te = csr_matrix(BusinessTravel_oneHot_te)

#**********************************************************************
**************#

# Performing OneHot encoding on Department column
Department_oneHot_tr =
ohe.fit_transform(x_train[['Department']].values)
Department_oneHot_te = ohe.transform(x_test[['Department']].values)

Department_oneHot_tr = csr_matrix(Department_oneHot_tr)
Department_oneHot_te = csr_matrix(Department_oneHot_te)

#**********************************************************************
**************#
```

```
# Performing OneHot encoding on EducationField column
EducationField_oneHot_tr =
ohe.fit_transform(x_train[['EducationField']].values)
EducationField_oneHot_te =
ohe.transform(x_test[['EducationField']].values)

EducationField_oneHot_tr = csr_matrix(EducationField_oneHot_tr)
EducationField_oneHot_te = csr_matrix(EducationField_oneHot_te)

#**********************************************************************
**************#

# Performing OneHot encoding on Gender column
Gender_oneHot_tr = ohe.fit_transform(x_train[['Gender']].values)
Gender_oneHot_te = ohe.transform(x_test[['Gender']].values)

Gender_oneHot_tr = csr_matrix(Gender_oneHot_tr)
Gender_oneHot_te = csr_matrix(Gender_oneHot_te)

#**********************************************************************
**************#

# Performing OneHot encoding on JobRole column
JobRole_oneHot_tr = ohe.fit_transform(x_train[['JobRole']].values)
JobRole_oneHot_te = ohe.transform(x_test[['JobRole']].values)

JobRole_oneHot_tr = csr_matrix(JobRole_oneHot_tr)
JobRole_oneHot_te = csr_matrix(JobRole_oneHot_te)

#**********************************************************************
**************#

# Performing OneHot encoding on MaritalStatus column
MaritalStatus_oneHot_tr =
ohe.fit_transform(x_train[['MaritalStatus']].values)
MaritalStatus_oneHot_te =
ohe.transform(x_test[['MaritalStatus']].values)

MaritalStatus_oneHot_tr = csr_matrix(MaritalStatus_oneHot_tr)
MaritalStatus_oneHot_te = csr_matrix(MaritalStatus_oneHot_te)

#**********************************************************************
**************#

# Performing OneHot encoding on OverTime column
OverTime_oneHot_tr = ohe.fit_transform(x_train[['OverTime']].values)
OverTime_oneHot_te = ohe.transform(x_test[['OverTime']].values)
```

```
OverTime_oneHot_tr = csr_matrix(OverTime_oneHot_tr)
OverTime_oneHot_te = csr_matrix(OverTime_oneHot_te)

print("Number of unique values in the column BusinessTravel are:
",df.BusinessTravel.unique().size)
print("Shape of column BusinessTravel after performing OneHot encoding
is:")
print("Train Data: ", BusinessTravel_oneHot_tr.shape, "Test Data",
BusinessTravel_oneHot_te.shape )
print("=="*35)

print("Number of unique values in the column BusinessTravel are:
",df.Department.unique().size)
print("Shape of column BusinessTravel after performing OneHot encoding
is:")
print("Train Data: ", Department_oneHot_tr.shape, "Test Data",
Department_oneHot_te.shape )
print("=="*35)

print("Number of unique values in the column BusinessTravel are:
",df.EducationField.unique().size)
print("Shape of column BusinessTravel after performing OneHot encoding
is:")
print("Train Data: ", EducationField_oneHot_tr.shape, "Test Data",
EducationField_oneHot_te.shape )
print("=="*35)

print("Number of unique values in the column BusinessTravel are:
",df.Gender.unique().size)
print("Shape of column BusinessTravel after performing OneHot encoding
is:")
print("Train Data: ", Gender_oneHot_tr.shape, "Test Data",
Gender_oneHot_te.shape )
print("=="*35)

print("Number of unique values in the column BusinessTravel are:
",df.JobRole.unique().size)
print("Shape of column BusinessTravel after performing OneHot encoding
is:")
print("Train Data: ", JobRole_oneHot_tr.shape, "Test Data",
JobRole_oneHot_tr.shape )
print("=="*35)

print("Number of unique values in the column BusinessTravel are:
",df.MaritalStatus.unique().size)
print("Shape of column BusinessTravel after performing OneHot encoding
is:")
print("Train Data: ", MaritalStatus_oneHot_tr.shape, "Test Data",
MaritalStatus_oneHot_te.shape )
print("=="*35)
```

```
print("Number of unique values in the column BusinessTravel are: ",df.OverTime.unique().size)
print("Shape of column BusinessTravel after performing OneHot encoding is:")
print("Train Data: ", OverTime_oneHot_tr.shape, "Test Data", OverTime_oneHot_te.shape )
```

```
Number of unique values in the column BusinessTravel are:  3
Shape of column BusinessTravel after performing OneHot encoding is:
Train Data:  (1029, 3) Test Data (441, 3)
======================================================================
Number of unique values in the column BusinessTravel are:  3
Shape of column BusinessTravel after performing OneHot encoding is:
Train Data:  (1029, 3) Test Data (441, 3)
======================================================================
Number of unique values in the column BusinessTravel are:  6
Shape of column BusinessTravel after performing OneHot encoding is:
Train Data:  (1029, 6) Test Data (441, 6)
======================================================================
Number of unique values in the column BusinessTravel are:  2
Shape of column BusinessTravel after performing OneHot encoding is:
Train Data:  (1029, 2) Test Data (441, 2)
======================================================================
Number of unique values in the column BusinessTravel are:  9
Shape of column BusinessTravel after performing OneHot encoding is:
Train Data:  (1029, 9) Test Data (1029, 9)
======================================================================
Number of unique values in the column BusinessTravel are:  3
Shape of column BusinessTravel after performing OneHot encoding is:
Train Data:  (1029, 3) Test Data (441, 3)
======================================================================
Number of unique values in the column BusinessTravel are:  2
Shape of column BusinessTravel after performing OneHot encoding is:
Train Data:  (1029, 2) Test Data (441, 2)
```

Combining all the above Features:

```
#print(type(OverTime_oneHot_tr), OverTime_oneHot_tr.shape )
#print(type(BusinessTravel_oneHot_tr), BusinessTravel_oneHot_tr.shape )

from scipy.sparse import hstack

# stacking up all the features into Train and Test groups

x_train_ft = hstack(( Age_train, DailyRate_train, DistanceFromHome_train, Education_train, EnvironmentSatisfaction_train, HourlyRate_train,
                     JobInvolvement_train, JobLevel_train, JobSatisfaction_train, MonthlyIncome_train, MonthlyRate_train,
```

```
NumCompaniesWorked_train,
                     PercentSalaryHike_train, PerformanceRating_train,
RelationshipSatisfaction_train, StockOptionLevel_train,
TotalWorkingYears_train,
                     TrainingTimesLastYear_train,
WorkLifeBalance_train, YearsAtCompany_train, YearsInCurrentRole_train,
YearsSinceLastPromotion_train,
                     YearsWithCurrManager_train,
YearsWithCurrManager_train, BusinessTravel_oneHot_tr,
Department_oneHot_tr, EducationField_oneHot_tr,
                     Gender_oneHot_tr, JobRole_oneHot_tr,
MaritalStatus_oneHot_tr, OverTime_oneHot_tr  )).tocsr()

x_test_ft = hstack(( Age_test, DailyRate_test, DistanceFromHome_test,
Education_test, EnvironmentSatisfaction_test, HourlyRate_test,
                     JobInvolvement_test, JobLevel_test,
JobSatisfaction_test, MonthlyIncome_test, MonthlyRate_test,
NumCompaniesWorked_test,
                     PercentSalaryHike_test, PerformanceRating_test,
RelationshipSatisfaction_test, StockOptionLevel_test,
TotalWorkingYears_test,
                     TrainingTimesLastYear_test, WorkLifeBalance_test,
YearsAtCompany_test, YearsInCurrentRole_test,
YearsSinceLastPromotion_test,
                     YearsWithCurrManager_test,
YearsWithCurrManager_test, BusinessTravel_oneHot_te,
Department_oneHot_te, EducationField_oneHot_te,
                     Gender_oneHot_te, JobRole_oneHot_te,
MaritalStatus_oneHot_te, OverTime_oneHot_te  )).tocsr()

print("Final Data Matrix is :")
print('='*22)


print("Training data shape :")
print('-'*22)
print(x_train_ft.shape , y_train.shape)

print('\n')

print("Testing data shape :")
print('-'*21)
print(x_test_ft.shape , y_test.shape)
```

```
Final Data Matrix is :
======================
Training data shape :
----------------------
(1029, 52) (1029, 1)
```

```
Testing data shape :
---------------------
(441, 52) (441, 1)
```

Function to Evaluate our Model

```
# function to evaluate our model using different metrics

# values to be passed :
# model_name = Our Model name, model = classifier used used to
predict, y_train_pred, y_test_pred, x_train, x_test
def evaluate_model(model_name, model, y_train_pred, y_test_pred,
x_train, x_test):

    # Printing Train & Test Accuracy scores
    print("Train Accuracy :", accuracy_score(y_train,
model.predict(x_train)))
    print("Test Accuracy :", accuracy_score(y_test,
model.predict(x_test)))

    print('\n')
    print("="*60)
    print('\n')


#**********************************************************************
**************************************#

    # Printing Confusion Matrix for Train & Test data
    print("Train Confusion Matrix:")
    print(confusion_matrix(y_train, model.predict(x_train)))
    print("Test Confusion Matrix:")
    print(confusion_matrix(y_test, model.predict(x_test)))

    print('\n')
    print("="*60)
    print('\n')


#**********************************************************************
***************************************#
    # Printing classification reports

    # For Train Data
    print("Classification report for our Model's Training data:")
    print("-"*52)
    print(classification_report(y_train, model.predict(x_train)))

    print('\n')
```

```
    print("="*60)
    print('\n')

    # For Train Data
    print("Classification report for our Model's Test data:")
    print("-"*52)
    print(classification_report(y_test, model.predict(x_test)))

    print('\n')
    print("="*60)
    print('\n')


#*********************************************************************
*****************************************#

    # Calculating AUC ROC scores
    auc_train_data = roc_auc_score(y_train, y_train_pred[:,1])
    auc_test_data = roc_auc_score(y_test, y_test_pred[:,1])
    print("AUC scores for \nTrain data is :", auc_train_data," & \
nTest data is :", auc_test_data)

    print('\n')
    print("="*60)
    print('\n')

    # Plotting AUC ROC scores for Train & Test data
    # ROC Curve using predict_proba method
    print("Plotting AUC ROC curves for Train and Test Data")
    tr_fpr, tr_tpr, tr_thresh = roc_curve(y_train, y_train_pred[:,1],
pos_label=1)
    te_fpr, te_tpr, te_thresh = roc_curve(y_test, y_test_pred[:,1],
pos_label=1)

    plt.style.use('seaborn')

    # plot roc curves
    plt.plot(tr_fpr, tr_tpr, linestyle='--', color='orange',
label='Train AUC ='+str(auc(tr_fpr, tr_tpr).round(3)))
    plt.plot(te_fpr, te_tpr, linestyle='--', color='green',
label='Test AUC ='+str(auc(te_fpr, te_tpr).round(3)))

    # title
    plt.title('ROC curve using '+str(model_name)+' model')
    # x label
    plt.xlabel('False Positive Rate')
    # y label
    plt.ylabel('True Positive rate')
```

```
    plt.legend(loc='best')

    plt.show();
    print('\n')


#*********************************************************************
*********************************************#

    # https://www.quantinsti.com/blog/creating-heatmap-using-python-
seaborn

    # Plotting Train & Test Confusion matrices
    print("Plotting Train and Test Confusion matrices")
    sns.set()

    con_m_train = confusion_matrix(y_train, model.predict(x_train))
    con_m_test = confusion_matrix(y_test, model.predict(x_test))

    key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
    fig, ax = plt.subplots(1,2, figsize=(12,5))

    labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value)
for key, value in zip(key.flatten(),
con_m_train.flatten())])).reshape(2,2)
    labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for
key, value in zip(key.flatten(), con_m_test.flatten())])).reshape(2,2)

    sns.heatmap(con_m_train, linewidths=.5, xticklabels=['PREDICTED :
0', 'PREDICTED : 1'], yticklabels=['ACTUAL : 0', 'ACTUAL : 1'], annot
= labels_train, fmt = '', ax=ax[0], cmap='Blues')
    sns.heatmap(con_m_test, linewidths=.5, xticklabels=['PREDICTED :
0', 'PREDICTED : 1'], yticklabels=['ACTUAL : 0', 'ACTUAL : 1'], annot
= labels_test, fmt = '', ax=ax[1], cmap='Blues')

    ax[0].set_title('Train Data')
    ax[1].set_title('Test Data')

    plt.show()


#*********************************************************************
********************#
```

Decision Tree

Performing Hyperparameter tuning using Cross Validation

```
model = DecisionTreeClassifier()

param = { 'min_samples_split' : [5, 6, 7, 10, 50, 100, 200, 500],
          'max_depth' : [1, 2, 3, 4, 5 ,10, 30, 40]
        }

clf_dt = GridSearchCV(model, param_grid = param, scoring = 'roc_auc',
cv = 10, verbose = 1,  return_train_score = True)
clf_dt.fit(x_train_ft, y_train)

print("Best value of Parameters for our Decision Tree model are  :",
clf_dt.best_estimator_)
```

```
Fitting 10 folds for each of 64 candidates, totalling 640 fits
Best value of Parameters for our Decision Tree model are  :
DecisionTreeClassifier(max_depth=30, min_samples_split=50)
```

```
clf_dt.score(x_test_ft, y_test)      # just to test the values
```

```
0.7711457936810049
```

Applying Best Paramters to the model

```
# best parameters for our Decision Tree

best_depth_DT = 30
best_min_sample_split_DT = 50

print(" Best parameters for our Decision Tree based on TFIDF are:\n
Best Depth = {0} &  \n Best Samples split per node = {1}"
       .format(best_depth_DT, best_min_sample_split_DT))

model_DT = DecisionTreeClassifier( max_depth = best_depth_DT,
min_samples_split = best_min_sample_split_DT )

# fitting our model on Train data
model_DT.fit(x_train_ft, y_train )


y_train_pred_DT = model_DT.predict_proba(x_train_ft)
y_test_pred_DT = model_DT.predict_proba(x_test_ft)
```

```
 Best parameters for our Decision Tree based on TFIDF are:
 Best Depth = 30 &
 Best Samples split per node = 50
```

Evaluting Model built using Decision Trees

```
# values to be passed to evaluate our Model:
# model_name = Our Model name, model = classifier used used to predict, y_train_pred, y_test_pred, x_train, x_test
evaluate_model ('Decision Tree', model_DT, y_train_pred_DT, y_test_pred_DT, x_train_ft, x_test_ft )
```

```
Train Accuracy : 0.8551992225461613
Test Accuracy : 0.854875283446712


============================================================


Train Confusion Matrix:
[[840  23]
 [126  40]]
Test Confusion Matrix:
[[361   9]
 [ 55  16]]


============================================================


Classification report for our Model's Training data:
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.87      0.97      0.92       863
           1       0.63      0.24      0.35       166

    accuracy                           0.86      1029
   macro avg       0.75      0.61      0.63      1029
weighted avg       0.83      0.86      0.83      1029



============================================================


Classification report for our Model's Test data:
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.87      0.98      0.92       370
           1       0.64      0.23      0.33        71

    accuracy                           0.85       441
```

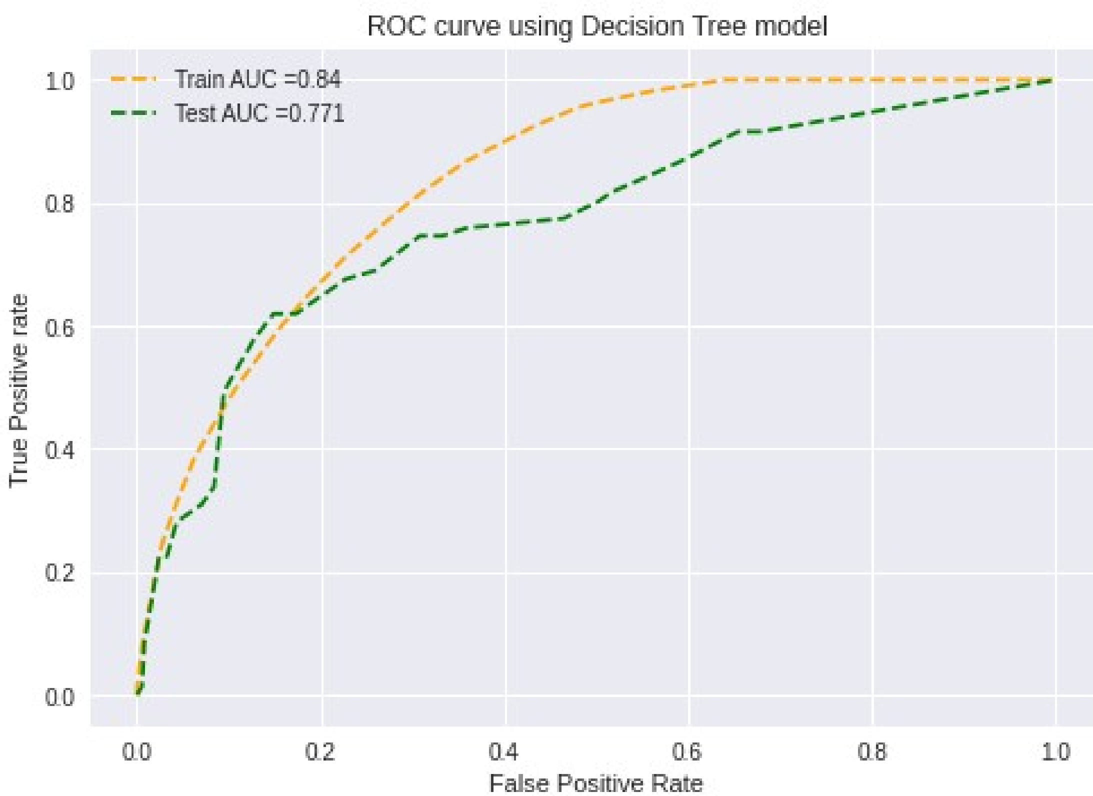
```
   macro avg       0.75      0.60      0.63       441
weighted avg       0.83      0.85      0.82       441
```

```
============================================================
```

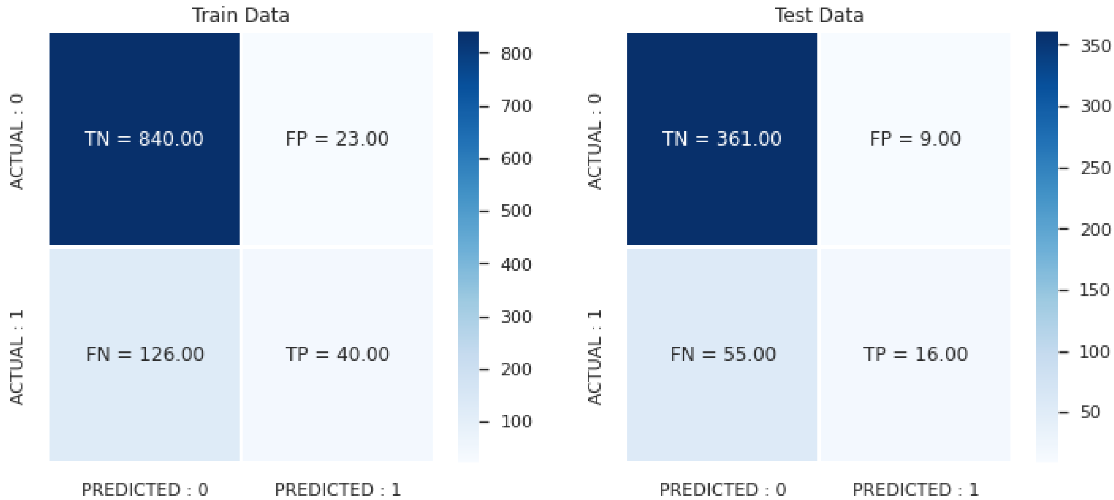
```
AUC scores for
Train data is : 0.8402357983498304  &
Test data is : 0.7711457936810049
```

```
============================================================
```

```
Plotting AUC ROC curves for Train and Test Data
```




```
Plotting Train and Test Confusion matrices
```



Naive Bayes

Performing Hyperparameter tuning using Cross Validation

```
model = ComplementNB()        # check original prediction using prior probability
#model = CategoricalNB ()
param= {'alpha': [0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.1, 0.5, 0.8, 1, 2, 4, 5, 10, 15] }

#clf=GridSearchCV(model,param,scoring='roc_auc',cv=5, return_train_score=True)
clf_NB = GridSearchCV(model, param, scoring='roc_auc', cv = 15, return_train_score = True, verbose = 1)
clf_NB.fit(x_train_ft, y_train)

#class_prior = [0.5, 0.5]

print("Best value of Alpha is obtained at :", clf_NB.best_estimator_)
```

```
Fitting 15 folds for each of 16 candidates, totalling 240 fits
Best value of Alpha is obtained at : ComplementNB(alpha=0.1)
```

Applying Best Parameters to our Model

```
# best parameters for our Naive Bayes model

best_alpha_NB = 0.1

print(" Best Parameters for our Naiye Bayes model is :\n Best Alpha = {0} "
       .format(best_alpha_NB))

model_NB = ComplementNB(alpha = best_alpha_NB) #, class_prior = [, 0.5])
```

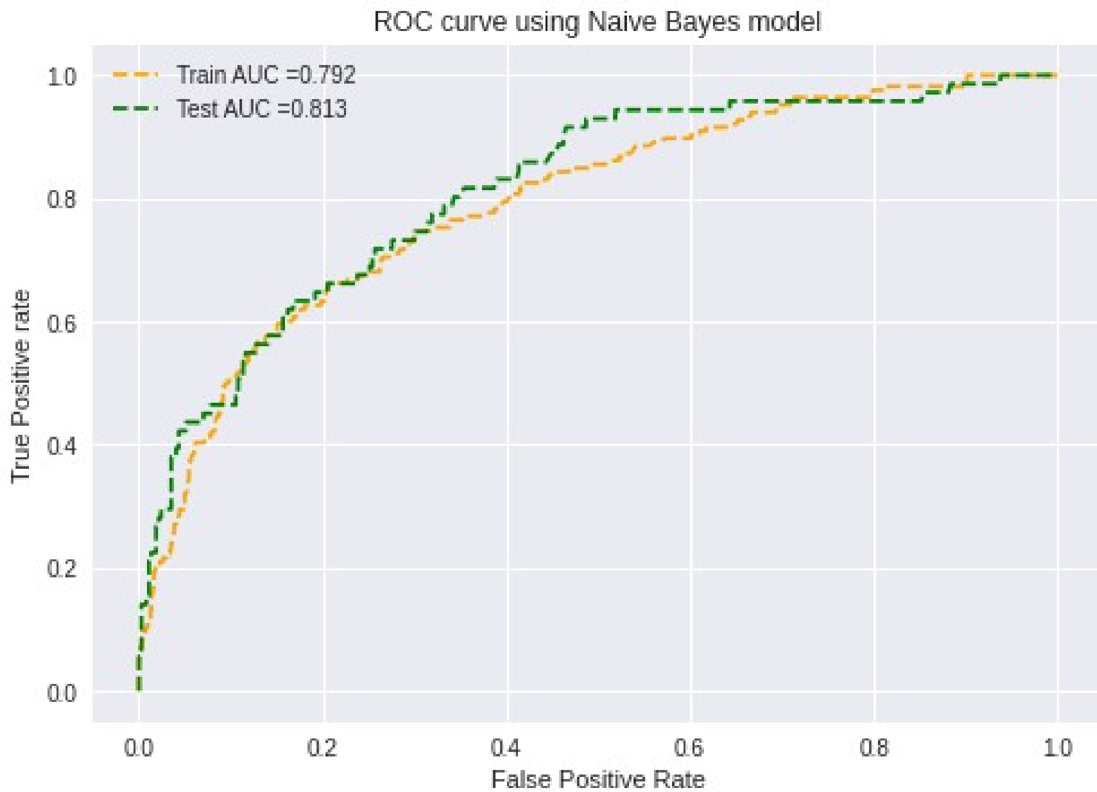
```python
# fitting our model on Train data
model_NB.fit(x_train_ft, y_train )

y_train_pred_NB = model_NB.predict_proba(x_train_ft)
y_test_pred_NB = model_NB.predict_proba(x_test_ft)
```

```
 Best Parameters for our Naiye Bayes model is :
 Best Alpha = 0.1
```

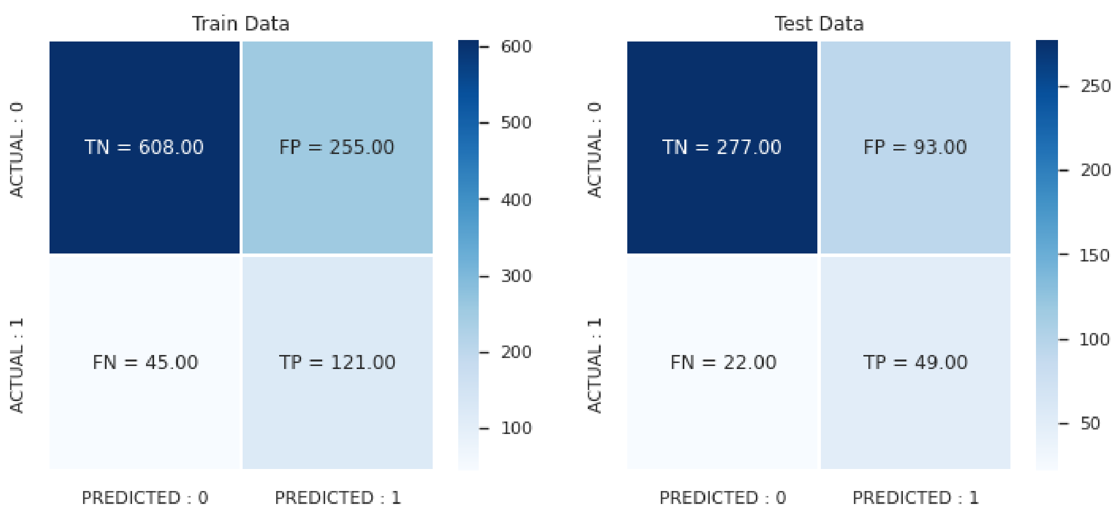
```python
# values to be passed to evaluate our Model:
# model_name = Our Model name, model = classifier used used to
predict, y_train_pred, y_test_pred, x_train, x_test
evaluate_model ('Naive Bayes', model_NB, y_train_pred_NB,
y_test_pred_NB, x_train_ft, x_test_ft )
```

```
Train Accuracy : 0.7084548104956269
Test Accuracy : 0.7392290249433107


============================================================



Train Confusion Matrix:
[[608 255]
 [ 45 121]]
Test Confusion Matrix:
[[277  93]
 [ 22  49]]


============================================================



Classification report for our Model's Training data:
-----------------------------------------------------
              precision    recall  f1-score   support

           0       0.93      0.70      0.80       863
           1       0.32      0.73      0.45       166

    accuracy                           0.71      1029
   macro avg       0.63      0.72      0.62      1029
weighted avg       0.83      0.71      0.74      1029



============================================================
```

```
Classification report for our Model's Test data:
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.93      0.75      0.83       370
           1       0.35      0.69      0.46        71

    accuracy                           0.74       441
   macro avg       0.64      0.72      0.64       441
weighted avg       0.83      0.74      0.77       441



============================================================


AUC scores for
Train data is : 0.7924339303913219  &
Test data is : 0.8128092881614009


============================================================


Plotting AUC ROC curves for Train and Test Data
```



Plotting Train and Test Confusion matrices



Support Vector Machine

Performing Hyperparameter tuning using Cross Validation

```
model = SVC()
```

```
C     = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
gamma = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
kernel = ['poly']
hyper = {'kernel': kernel, 'C':C, 'gamma': gamma}


clf_SVM = GridSearchCV(estimator = model, param_grid = hyper, n_jobs =
-1, cv = 5, verbose = 1)
clf_SVM.fit(x_train_ft, y_train)


print("Best value of our hyperparameters are obtained at :",
clf_SVM.best_estimator_)
print("Best score using best hyperparameters is obtained at :",
clf_SVM.best_score_)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Best value of our hyperparameters are obtained at : SVC(C=0.3,
gamma=0.1, kernel='poly')
Best score using best hyperparameters is obtained at :
0.8464740705659484
```

Applying our Best parameters to our Model

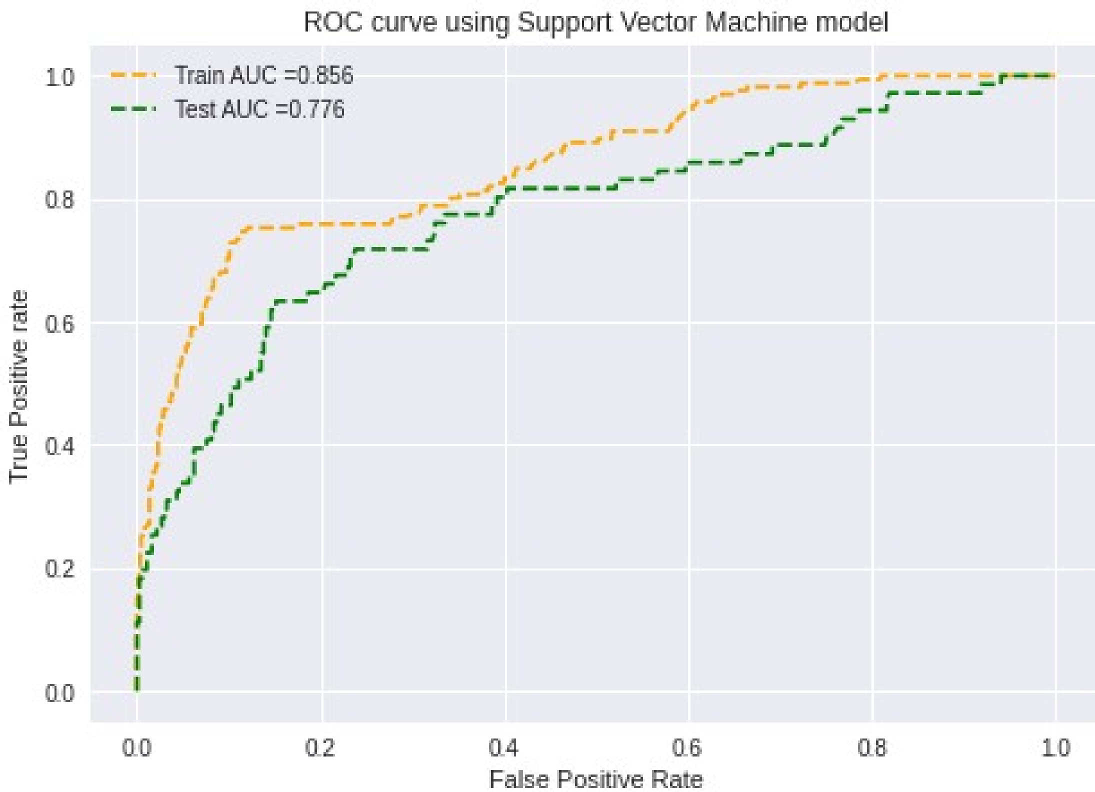
```
# best parameters for our Support Vector Machine model

best_C_SVM = 0.3
best_gamma_SVM = 0.1

print(" Best parameters for our Support Vector Machine model is :\n
Best C = {0}, Best Gamma = {1} "
       .format(best_C_SVM, best_gamma_SVM ))

model_SVM = SVC(C = best_C_SVM, gamma = best_gamma_SVM, kernel =
'poly', probability = True )

# fitting our model on Train data
model_SVM.fit(x_train_ft, y_train )

y_train_pred_SVM = model_SVM.predict_proba(x_train_ft)
y_test_pred_SVM = model_SVM.predict_proba(x_test_ft)
```

```
 Best parameters for our Support Vector Machine model is :
 Best C = 0.3, Best Gamma = 0.1
```

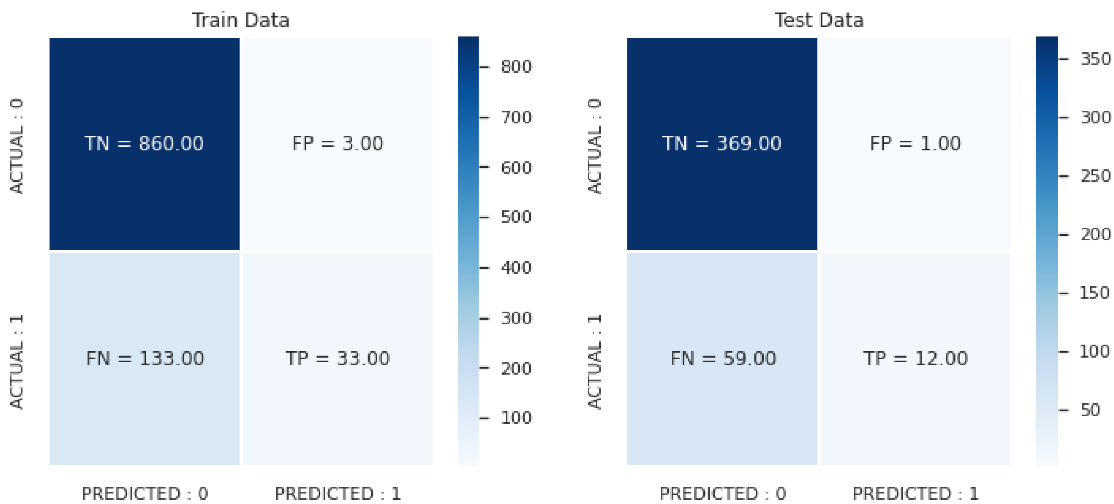
```
# values to be passed to evaluate our Model:
# model_name = Our Model name, model = classifier used used to
predict, y_train_pred, y_test_pred, x_train, x_test
evaluate_model ('Support Vector Machine', model_SVM, y_train_pred_SVM,
y_test_pred_SVM, x_train_ft, x_test_ft )
```

```
Train Accuracy : 0.8678328474246841
Test Accuracy : 0.8639455782312925


============================================================


Train Confusion Matrix:
[[860   3]
 [133  33]]
Test Confusion Matrix:
[[369   1]
 [ 59  12]]


============================================================


Classification report for our Model's Training data:
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.87      1.00      0.93       863
           1       0.92      0.20      0.33       166

    accuracy                           0.87      1029
   macro avg       0.89      0.60      0.63      1029
weighted avg       0.87      0.87      0.83      1029



============================================================


Classification report for our Model's Test data:
----------------------------------------------------
              precision    recall  f1-score   support

           0       0.86      1.00      0.92       370
           1       0.92      0.17      0.29        71

    accuracy                           0.86       441
   macro avg       0.89      0.58      0.61       441
weighted avg       0.87      0.86      0.82       441



============================================================
```

```
AUC scores for
Train data is : 0.8560534141199793  &
Test data is : 0.776341834792539


============================================================


Plotting AUC ROC curves for Train and Test Data
```




```
Plotting Train and Test Confusion matrices
```



Tabulating our results using Pretty table

```
# initializing a table
table = PrettyTable()

# adding title
table.title = "HR-Analytics Employee Attrition Prediction Modelling
Overview"
# adding fields
table.field_names = ["Model", "Best Hyper-Parameter", "Test AUC"]


# adding rows to our table
table.add_row(['Decision Trees', 'max_depth = 30 & min_samples_split =
50', '0.771'])
table.add_row(['Naive Bayes', 'alpha = 0.1' , '0.812'])
table.add_row(['Support Vector Machine', 'C = 0.3, gamma = 0.1 &
kernal = poly', '0.776'])

# printing the table
print(table)
```

```
+----------------------------------------------------------------------
--------+
|        HR-Analytics Employee Attrition Prediction Modelling Overview
|
+-------------------------+-----------------------------------------
+----------+
|         Model           |            Best Hyper-Parameter            |
Test AUC |
+-------------------------+-----------------------------------------
+----------+
|     Decision Trees      | max_depth = 30 & min_samples_split = 50 |
0.771    |
|      Naive Bayes        |                alpha = 0.1               |
```

```
0.812   |
| Support Vector Machine |   C = 0.3, gamma = 0.1 & kernal = poly  |
0.776   |
+-----------------------+--------------------------------------
+----------+
```

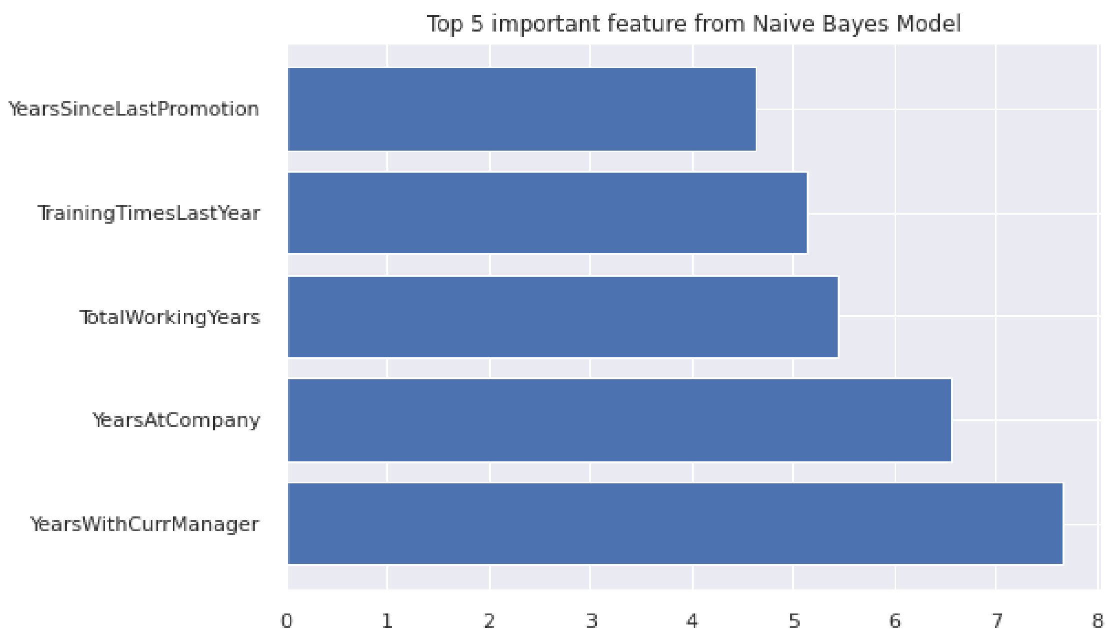
Feature importance

```python
# storing all the feature names inside a list
features_names = []

# adding each column / feature to our list of features
for col in df.columns:
    features_names.append(col)

features_names
```

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

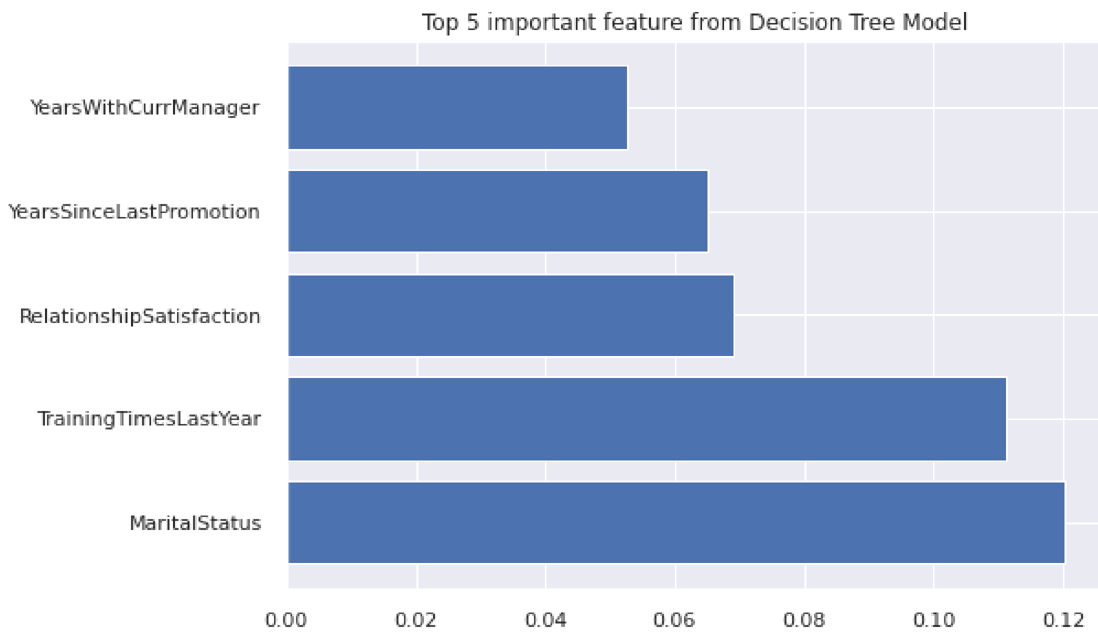
```python
''' Top feature from Naive Bayes Model '''
def f_importances(coef, names, top=-1):
    imp = coef
```

```
    imp, names = zip(*sorted(list(zip(imp, names))))

    # Show all features
    if top == -1:
        top = len(names)

    plt.barh(range(top), imp[::-1][0:top], align='center')
    plt.yticks(range(top), names[::-1][0:top])
    plt.title("Top 5 important feature from Naive Bayes Model")
    plt.show()


# Specify your top n features you want to visualize.
f_importances(model_NB.coef_[0], features_names, top = 5)
```




```
''' Top feature from Decision Tree Model '''
def f_importances(coef, names, top=-1):
    imp = coef
    imp, names = zip(*sorted(list(zip(imp, names))))

    # Show all features
    if top == -1:
        top = len(names)

    plt.barh(range(top), imp[::-1][0:top], align='center')
    plt.yticks(range(top), names[::-1][0:top])
    plt.title("Top 5 important feature from Decision Tree Model")
    plt.show()
```

```
# Specify your top n features you want to visualize.
f_importances(model_DT.feature_importances_, features_names, top = 5)
```



Observations from the performance of our Models:

1. Overvall, we have got good values of AUC for all the 3 models. But of all the 3, Naive Bayes model has got the highest AUC of nearly 0.812
2. We can also see that some of the the top 5 feature importance from Naive Bayes and Decision model are same.

Decision trees:

1. It can work well with both Categorical and Numerical variables. This is an example of a white box model, which closely mimics the human decision-making process. Feature selection happens automatically and unimportant features will not affect the result. The presence of features that depend on each other (multicollinearity) also will not affect the performance of our Model.

Support Vector Machines(SVM):

1. SVM with poly Kernal has perfomed little well than Decision tree model. We have only around 1.4k data points in our Data and if we have some more data then this would have performed even better than this. But this model will become inefficient if we use a large Dataset

Naive Bayes:

1. This model perfomed very well when compared to other 2 models and it is having a very less training time among the other ones. Performance of this model will be even high if most of the features are Categorical in nature, even though we have some categorical features but most of them are numerical values. And this models

assumes that all the features are independent which is the same thing for most of the features in our Data.