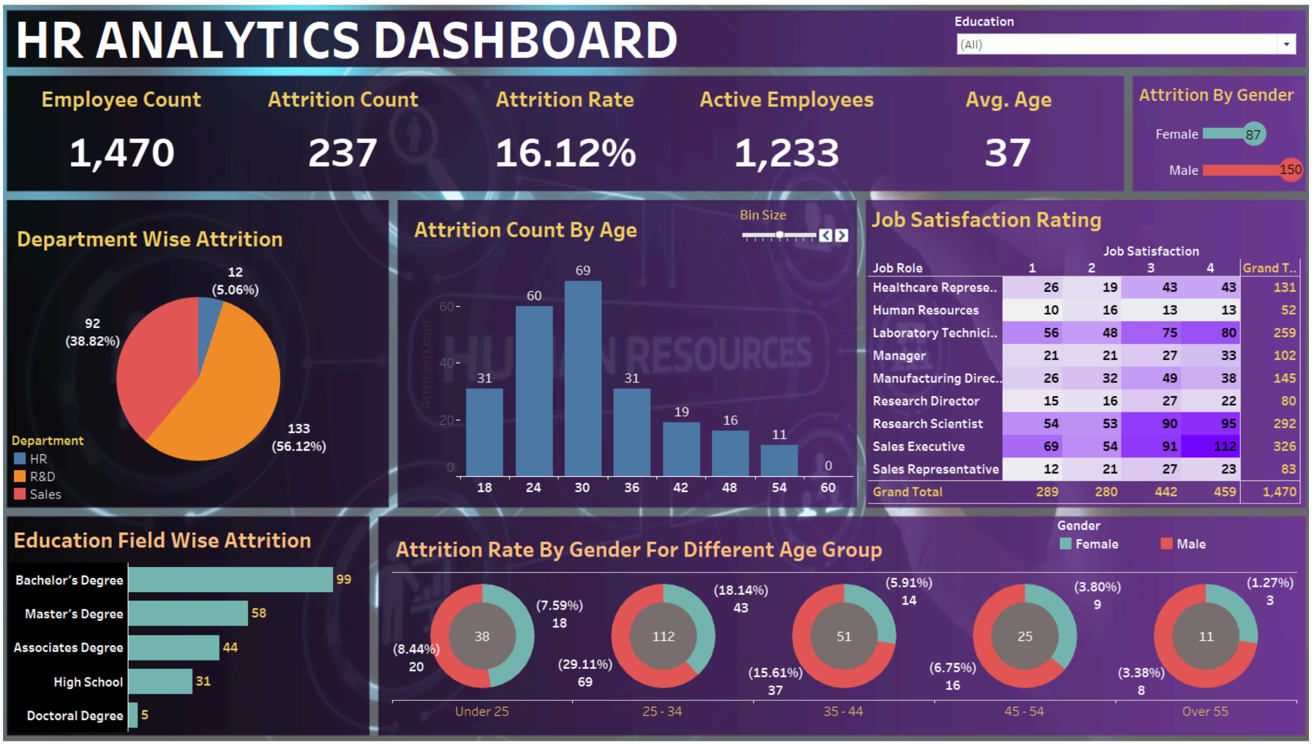
For my Workforce Analytics project, I am focusing on the available features to identify solutions for addressing attrition and forecasting trends to help stabilize attrition rates. My goal is to analyze these features to ensure smoother business operations and maintain a steady workforce flow.

This is a dashboard I created on TABLEAU to see the attrition at gender, department, education level etc. It also includes satisfaction ratings to delve deep into possible reasons for attrition



I have included step by step procedure for the project including, data cleaning, EDA, Visualization, Survival Analysis and Prediction using various models.

In [524…
```python
'''starting by importing the data and having a look at the various features gathered for the probl

# Importing the libraires
import pandas as pd
import numpy as np
attrition = pd.read_excel(r"D:\project\tableau\Final dataset Attrition.xlsx")
```

In [525…
```python
# Since the dataset is loaded we check a few details like
attrition.head(10)
```

Out[525…

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Gender	JobInvolvement	JobLevel
0	37	Yes	Travel_Rarely	Research & Development	2	Male	2	1
1	21	No	Travel_Rarely	Research & Development	15	Male	3	1
2	45	No	Travel_Rarely	Research & Development	6	Male	3	3
3	23	No	Travel_Rarely	Sales	2	Male	3	1
4	22	No	Travel_Rarely	Research & Development	15	Female	3	1
5	19	Yes	Travel_Rarely	Sales	22	Male	3	1
6	19	Yes	Travel_Frequently	Sales	1	Female	1	1
7	28	Yes	Travel_Rarely	Research & Development	2	Male	3	1
8	29	No	Travel_Rarely	Sales	2	Male	2	2
9	18	Yes	Travel_Rarely	Research & Development	3	Male	3	1

10 rows × 32 columns

In [526…

```
# Checking the columns wihtin the dataset
attrition.columns
```

Out[526…

```
Index(['Age', 'Attrition', 'BusinessTravel', 'Department', 'DistanceFromHome',
       'Gender', 'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked', 'OverTime',
       'PercentSalaryHike', 'PerformanceRating', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'YearsAtCompany',
       'YearsSinceLastPromotion', 'YearsWithCurrManager', 'Higher_Education',
       'Date_of_Hire', 'Date_of_termination', 'Status_of_leaving',
       'Mode_of_work', 'Leaves', 'Absenteeism', 'Work_accident',
       'Source_of_Hire', 'Job_mode'],
      dtype='object')
```

In [527…

```
# checking the dimensions of the dataset
attrition.shape
```

Out[527…

```
(1470, 32)
```

The dataset has the following features and the description are as follows

The dataset gathered has 1,470 nos of observations and the following 32 nos of features

1. "Age" = The age of the employee
2. "Attrition" = Whether the employee has attrited or not
3. "BusinessTravel" = Whether the employee used to travel for business or not
4. "Department" = Which department the employee was employed under

5. "DistanceFromHome" = The distance the employee travels to reach for job on a day to day basis
6. "Gender" = Gender of the employee
7. "JobInvolvement" = The involvement rating of an employee over the job handled
8. "JobLevel" = Level at which the employee is working
9. "JobRole" = The roles and resposibilites of the employee
10. "JobSatisfaction" = Satisfaction rating of the employee for the job
11. "MaritalStatus" = Marital status of the employee
12. "MonthlyIncome" = Monthly income of the employees
13. "NumCompaniesWorked" = Number of companies the employees has worked for
14. "OverTime" = Whether working Overtime or not
15. "PercentSalaryHike" = Percentage salary hike since their appointment in the company
16. "PerformanceRating" = Performance rating
17. "StockOptionLevel" = Level of opted for sharing the stock
18. "TotalWorkingYears" = Total years worked by the employees
19. "TrainingTimesLastYear" = How many trainings the employee has undergone
20. "YearsAtCompany" = Years spent at the present organisation
21. "YearsSinceLastPromotion" = Time gone in years since last promotion
22. "YearsWithCurrManager" = Years working under he current manager
23. "Higher_Education" = Higher education level of the employee
24. "Date_of_Hire" = Date of hire of the employee in the current organisation
25. "Date_of_termination" = Date of termination from the organisation
26. "Status_of_leaving" = Reason for leaving the organisation
27. "Mode_of_work" = WFH or WFO
28. "Leaves" = Total permitted leaves taken by the employee
29. "Absenteeism" = Total days absent for the employee
30. "Work_accident" = Work accident if any
31. "Source_of_hire" = Source of hire
32. "Job_Mode" = Working full time/ part or contractual

In [533…

```
attrition.describe()
```

Out[533…

	Age	DistanceFromHome	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	Nu
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	
mean	36.923810	9.192517	2.729932	2.063946	2.728571	6502.931293	
min	18.000000	1.000000	1.000000	1.000000	1.000000	1009.000000	
25%	30.000000	2.000000	2.000000	1.000000	2.000000	2911.000000	
50%	36.000000	7.000000	3.000000	2.000000	3.000000	4919.000000	
75%	43.000000	14.000000	3.000000	3.000000	4.000000	8379.000000	
max	60.000000	29.000000	4.000000	5.000000	4.000000	19999.000000	
std	9.135373	8.106864	0.711561	1.106940	1.102846	4707.956783	

In [535…

```
# Checking whether the datset has any missing values within
attrition.isna().sum()
```

Out[535…

```
Age                          0
Attrition                    0
BusinessTravel               0
Department                   0
DistanceFromHome             0
Gender                       0
JobInvolvement               0
JobLevel                     0
JobRole                      0
JobSatisfaction              0
MaritalStatus                0
MonthlyIncome                0
NumCompaniesWorked           0
OverTime                     0
PercentSalaryHike            0
PerformanceRating            0
StockOptionLevel             0
TotalWorkingYears            0
TrainingTimesLastYear        0
YearsAtCompany               0
YearsSinceLastPromotion      0
YearsWithCurrManager         0
Higher_Education             0
Date_of_Hire                 0
Date_of_termination       1470
Status_of_leaving            0
Mode_of_work                 0
Leaves                       0
Absenteeism                  0
Work_accident                0
Source_of_Hire               0
Job_mode                     0
dtype: int64
```

In [537…

```
# Category columns in the data
category_cols = ['Attrition', 'BusinessTravel', 'Department', 'Gender', 'JobRole', 'MaritalStatus'
```

In [539…

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
attrition[category_cols] = attrition[category_cols].apply(le.fit_transform)
attrition
```

Out[539…

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Gender	JobInvolvement	JobLevel
0	37	1	2	1	2	1	2	1
1	21	0	2	1	15	1	3	1
2	45	0	2	1	6	1	3	3
3	23	0	2	2	2	1	3	1
4	22	0	2	1	15	0	3	1
...	...	...	...	...	...	...	...	...
1465	52	0	2	2	3	1	2	4
1466	55	0	2	1	1	1	3	5
1467	55	0	2	2	26	1	2	5
1468	58	0	2	2	10	1	3	4
1469	58	1	2	1	23	0	3	3

1470 rows × 32 columns

In [541…

```
# removing/ dropping the columns passenger id, Name, ticket, cabin
attrition = attrition.drop(["Date_of_Hire", "Date_of_termination"], axis = 1)
attrition
```

Out[541…

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Gender	JobInvolvement	JobLevel
0	37	1	2	1	2	1	2	1
1	21	0	2	1	15	1	3	1
2	45	0	2	1	6	1	3	3
3	23	0	2	2	2	1	3	1
4	22	0	2	1	15	0	3	1
...	...	...	...	...	...	...	...	...
1465	52	0	2	2	3	1	2	4
1466	55	0	2	1	1	1	3	5
1467	55	0	2	2	26	1	2	5
1468	58	0	2	2	10	1	3	4
1469	58	1	2	1	23	0	3	3

1470 rows × 30 columns

In [543…

```
# Lets check out some visualisation to get the insights on the data
df_company = attrition

import seaborn as sns
import matplotlib.pyplot as plt
def stacked_plot(df, group, target):
    """
    Function to generate a stacked plots between two variables
    """
    fig, ax = plt.subplots(figsize = (6,4))
```

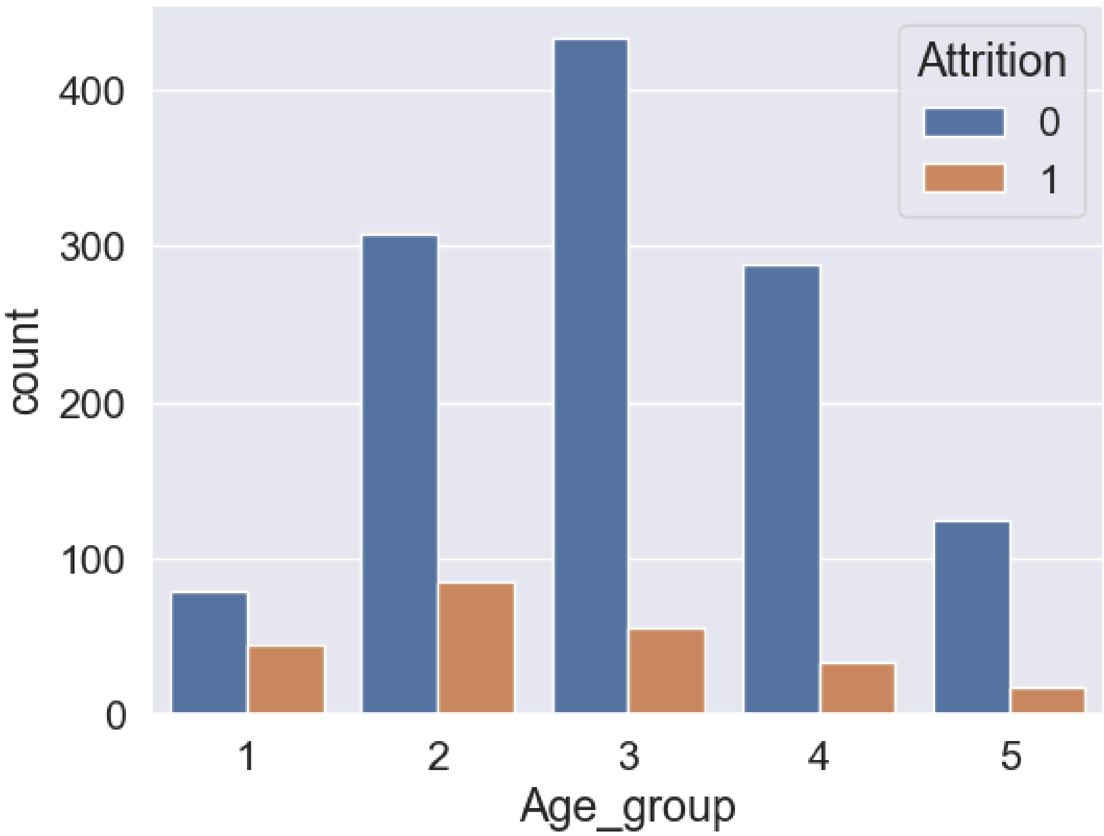
```
    temp_df = (df.groupby([group, target]).size()/df.groupby(group)[target].count()).reset_index()
    temp_df.plot(kind = 'bar', stacked = True, ax = ax, color = ["green", "darkred"])
    ax.xaxis.set_tick_params(rotation = 0)
    ax.set_xlabel(group)
    ax.set_ylabel('Attrition')
```

In [545…

```
def Age(a):
    if a <= 25:
        return 1
    elif a > 25 and a <= 32:
        return 2
    elif a > 32 and a <= 40:
        return 3
    elif a > 40 and a <= 50:
        return 4
    else:
        return 5

df_company["Age_group"] = df_company["Age"].apply(lambda x: Age(x))
df_company["Age_group"].value_counts()
sns.countplot(x = "Age_group", hue = "Attrition", data = df_company)
```

Out[545…

```
<Axes: xlabel='Age_group', ylabel='count'>
```



Having a look at the above plot which gives the relation between attrition and age group gives the insight that the employees in the age group of under 25 tend to move faster and the ones within 25 and 32 also

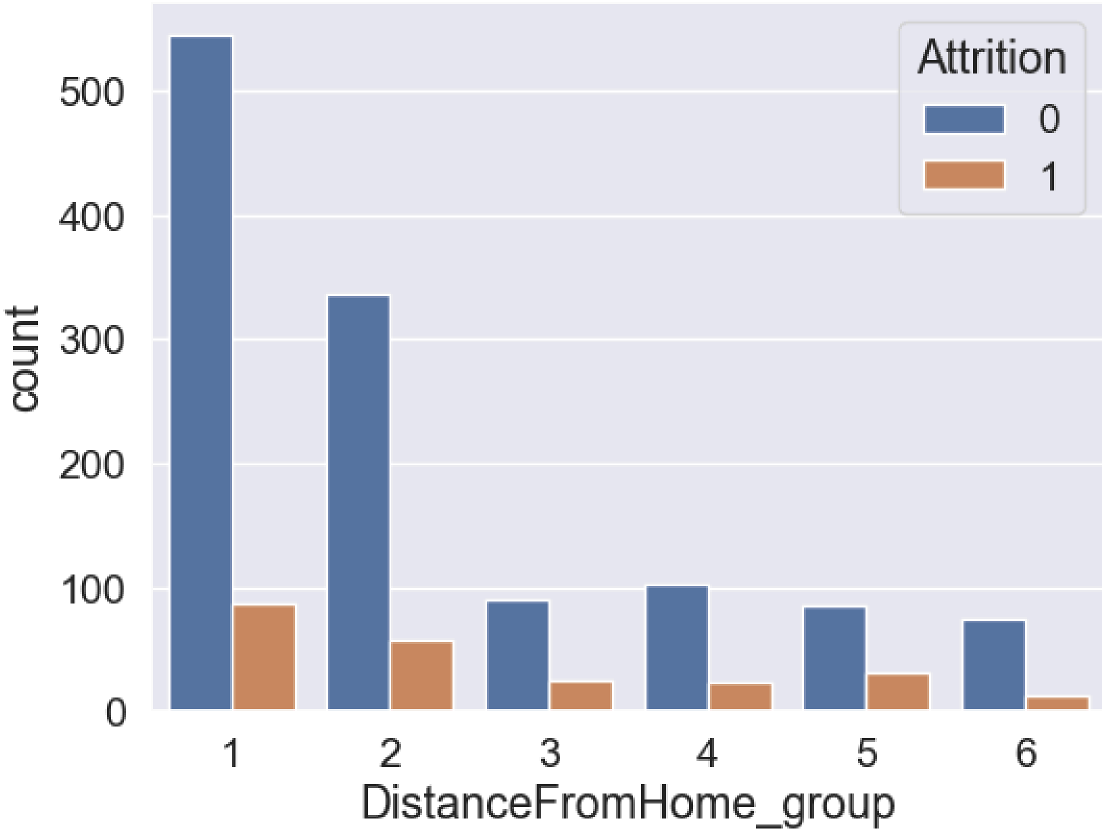
In [548…

```
def DistanceFromHome(d):
    if d <= 5:
        return 1
    elif d > 5 and d <= 10:
        return 2
    elif d > 10 and d <= 15:
        return 3
    elif d > 15 and d <= 20:
        return 4
    elif d > 20 and d <= 25:
```

```
        return 5
    else:
        return 6

df_company["DistanceFromHome_group"] = df_company["DistanceFromHome"].apply(lambda x: DistanceFrom
df_company["DistanceFromHome_group"].value_counts()
sns.countplot(x = "DistanceFromHome_group", hue="Attrition", data = df_company)

'''
Now taking the relation between attrition and Distance from home gives the insight that
the employees with a farther distance from home tend to take a decision to attrite quite obviously
'''
```

Out[548…

```
' \nNow taking the relation between attrition and Distance from home gives the insight that \nthe employees with a farther distance from home tend to take a decision to attrite quite obviousl
y.\n'
```



Now taking the relation between attrition and Distance from home gives the insight that the employees with a farther distance from home tend to take a decision to attrite quite obviously

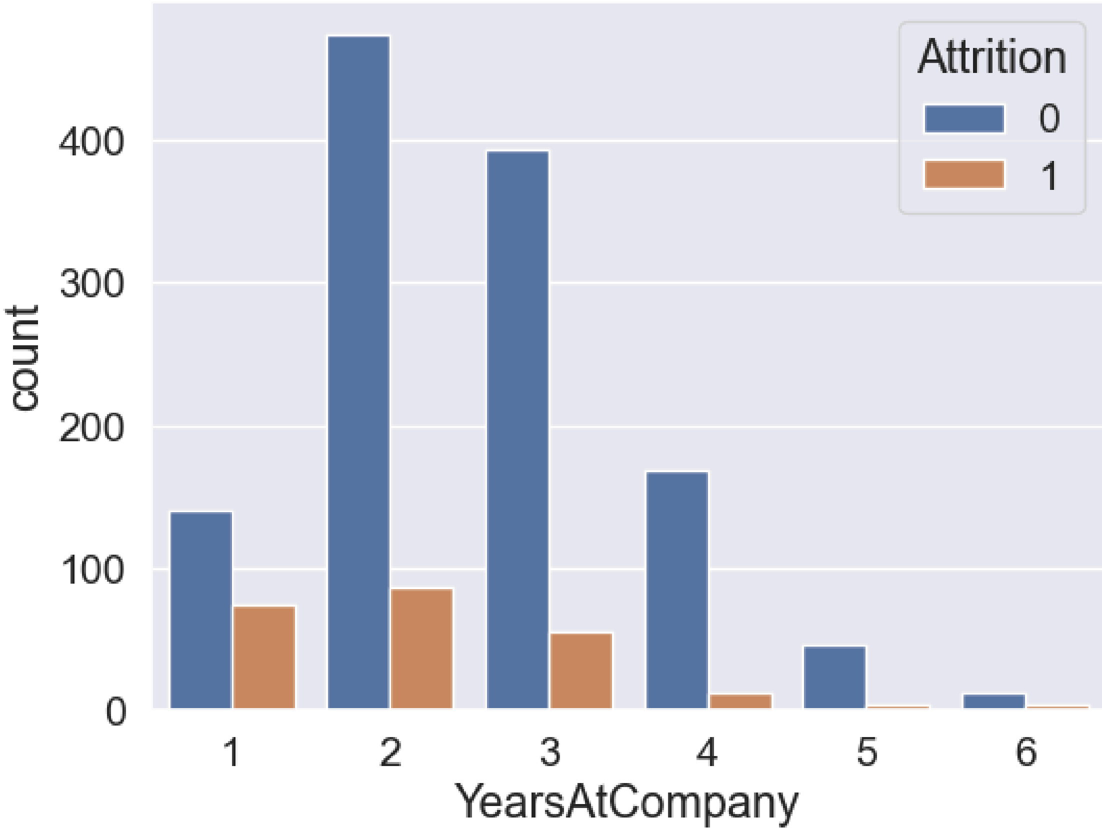
In [551…

```
def YearsAtCompany(t):
    if t <= 1:
        return 1
    elif t > 1 and t <= 5:
        return 2
    elif t > 5 and t <= 10:
        return 3
    elif t > 10 and t <= 20:
        return 4
    elif t > 20 and t <= 30:
        return 5
    else:
        return 6

df_company["YearsAtCompany"] = df_company["YearsAtCompany"].apply(lambda x:YearsAtCompany(x))
df_company["YearsAtCompany"].value_counts()
sns.countplot(x = "YearsAtCompany", hue = "Attrition", data = df_company)

'''
```

```
Now this interesting fact is very well known that the one year atrrition employees are
known as Jumpers but this does go against their profile, and then the most attritions
take place in the range of 1 to 5 years of employment.
'''
```

Out[551…

```
' \nNow this interesting fact is very well known that the one year atrrition employees are \nknow
n as Jumpers but this does go against their profile, and then the most attritions \ntake place in
the range of 1 to 5 years of employment.\n'
```



Now this interesting fact is very well known that the one year atrrition employees are known as Jumpers but this does go against their profile, and then the most attritions take place in the range of 1 to 5 years of employment.

In [554…

```
# df_company.to_excel(r"D:\project\tableau\Final dataset Attrition_final.xlsx")
```

Additionally we have to now normalize the data as the scale is not the same for all the variables. We will use minmax scaler for the job

In [558…

```
from sklearn.preprocessing import MinMaxScaler as mms
scale = mms()
attrition_mms = pd.DataFrame(scale.fit_transform(attrition.iloc[:,:]))
```

In [560…

```
attrition_mms.columns = attrition.columns
```

In [562…

```
attrition
```

Out[562…

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Gender	JobInvolvement	JobLevel
0	37	1	2	1	2	1	2	1
1	21	0	2	1	15	1	3	1
2	45	0	2	1	6	1	3	3
3	23	0	2	2	2	1	3	1
4	22	0	2	1	15	0	3	1
...	...	...	...	...	...	...	...	...
1465	52	0	2	2	3	1	2	4
1466	55	0	2	1	1	1	3	5
1467	55	0	2	2	26	1	2	5
1468	58	0	2	2	10	1	3	4
1469	58	1	2	1	23	0	3	3

1470 rows × 32 columns

In [564…

```
attrition = attrition.drop(attrition.iloc[:, 30:31], axis = 1)
```

In [566…

```
attrition
```

Out[566…

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Gender	JobInvolvement	JobLevel
0	37	1	2	1	2	1	2	1
1	21	0	2	1	15	1	3	1
2	45	0	2	1	6	1	3	3
3	23	0	2	2	2	1	3	1
4	22	0	2	1	15	0	3	1
...	...	...	...	...	...	...	...	...
1465	52	0	2	2	3	1	2	4
1466	55	0	2	1	1	1	3	5
1467	55	0	2	2	26	1	2	5
1468	58	0	2	2	10	1	3	4
1469	58	1	2	1	23	0	3	3

1470 rows × 31 columns

In [568…

```
# We check the correlation of the various features
attrition_mms.corr()
```

Out[568…

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Gender
Age	1.000000	-0.159205	0.024751	-0.031882	-0.001686	-0.036311
Attrition	-0.159205	1.000000	0.000074	0.063991	0.077924	0.029453
BusinessTravel	0.024751	0.000074	1.000000	-0.009044	-0.024469	-0.032981
Department	-0.031882	0.063991	-0.009044	1.000000	0.017225	-0.041583
DistanceFromHome	-0.001686	0.077924	-0.024469	0.017225	1.000000	-0.001851
Gender	-0.036311	0.029453	-0.032981	-0.041583	-0.001851	1.000000
JobInvolvement	0.029820	-0.130016	0.039062	-0.024586	0.008783	0.017960
JobLevel	0.509604	-0.169105	0.019311	0.101963	0.005303	-0.039403
JobRole	-0.122427	0.067151	0.002724	0.662431	-0.001015	-0.039723
JobSatisfaction	-0.004892	-0.103481	-0.033962	0.021001	-0.003669	0.033252
MaritalStatus	-0.095029	0.162070	0.024001	0.056073	-0.014437	-0.047183
MonthlyIncome	0.497855	-0.159840	0.034319	0.053130	-0.017014	-0.031858
NumCompaniesWorked	0.299635	0.043494	0.020875	-0.035882	-0.029251	-0.039147
OverTime	0.028062	0.246118	0.016543	0.007481	0.025514	-0.041924
PercentSalaryHike	0.003634	-0.013478	-0.029377	-0.007840	0.040235	0.002733
PerformanceRating	0.001904	0.002889	-0.026341	-0.024604	0.027110	-0.013859
StockOptionLevel	0.037510	-0.137145	-0.016727	-0.012193	0.044872	0.012716
TotalWorkingYears	0.680381	-0.171063	0.034226	-0.015762	0.004628	-0.046881
TrainingTimesLastYear	-0.019621	-0.059478	0.015240	0.036875	-0.036942	-0.038787
YearsAtCompany	0.260783	-0.171513	-0.021492	0.030146	-0.005582	-0.033069
YearsSinceLastPromotion	0.216513	-0.033019	-0.032591	0.040061	0.010029	-0.026985
YearsWithCurrManager	0.202089	-0.156199	-0.022636	0.034282	0.014406	-0.030599
Higher_Education	-0.000930	0.003642	-0.004724	0.049723	0.007394	0.035339
Status_of_leaving	-0.015250	0.020750	0.029387	-0.006956	0.003964	0.014051
Mode_of_work	0.009323	-0.006742	0.029590	0.010072	-0.029553	0.003336
Leaves	0.033811	-0.041820	-0.019584	0.000139	-0.022749	-0.024768
Absenteeism	-0.004628	-0.037867	-0.027932	-0.035409	0.024581	-0.031885
Work_accident	0.024869	0.009846	0.051351	-0.010932	-0.003409	-0.009442
Source_of_Hire	0.008830	0.004462	-0.024299	-0.007854	-0.030024	-0.043518
Job_mode	-0.030794	-0.055663	0.019918	0.028610	-0.021048	-0.016212
Age_group	0.962428	-0.164828	0.017509	-0.039766	0.000837	-0.037117
DistanceFromHome_group	0.008749	0.074065	-0.025894	0.011131	0.985209	-0.010878

32 rows × 32 columns

In [570…

```
attrition_mms = attrition_mms.drop(attrition_mms.iloc[:, 30:31], axis = 1)
corr_matrix = attrition_mms.corr()
```

```
(corr_matrix['Attrition'].sort_values(ascending = False))
```

Out[570…

```
Attrition                   1.000000
OverTime                    0.246118
MaritalStatus               0.162070
DistanceFromHome            0.077924
DistanceFromHome_group      0.074065
JobRole                     0.067151
Department                  0.063991
NumCompaniesWorked          0.043494
Gender                      0.029453
Status_of_leaving           0.020750
Work_accident               0.009846
Source_of_Hire              0.004462
Higher_Education            0.003642
PerformanceRating           0.002889
BusinessTravel              0.000074
Mode_of_work               -0.006742
PercentSalaryHike          -0.013478
YearsSinceLastPromotion    -0.033019
Absenteeism                -0.037867
Leaves                     -0.041820
Job_mode                   -0.055663
TrainingTimesLastYear      -0.059478
JobSatisfaction            -0.103481
JobInvolvement             -0.130016
StockOptionLevel           -0.137145
YearsWithCurrManager       -0.156199
Age                        -0.159205
MonthlyIncome              -0.159840
JobLevel                   -0.169105
TotalWorkingYears          -0.171063
YearsAtCompany             -0.171513
Name: Attrition, dtype: float64
```

We notice the correlation of various features and find that OverTime, Marital Status, DistanceFromHome and JobRole has the highest corelation with the Attririon

> EDA - Exploratory Data Analysis

In [574…

```
EDA = {"column": attrition_mms.columns,
       "mean": attrition_mms.mean(),
       "median": attrition_mms.median(),
       "mode": attrition_mms.mode(),
       "standard deviation": attrition_mms.std(),
       "variance": attrition_mms.var(),
       "skewness": attrition_mms.skew(),
       "kurtosis": attrition_mms.kurt()}
```

In [583…

```
EDA
```

```
Out[583… {'column': Index(['Age', 'Attrition', 'BusinessTravel', 'Department', 'DistanceFromHome',
        'Gender', 'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
        'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked', 'OverTime',
        'PercentSalaryHike', 'PerformanceRating', 'StockOptionLevel',
        'TotalWorkingYears', 'TrainingTimesLastYear', 'YearsAtCompany',
        'YearsSinceLastPromotion', 'YearsWithCurrManager', 'Higher_Education',
        'Status_of_leaving', 'Mode_of_work', 'Leaves', 'Absenteeism',
        'Work_accident', 'Source_of_Hire', 'Job_mode',
        'DistanceFromHome_group'],
       dtype='object'),
 'mean': Age                        0.450567
 Attrition                  0.161224
 BusinessTravel             0.803741
 Department                 0.630272
 DistanceFromHome           0.292590
 Gender                     0.600000
 JobInvolvement             0.576644
 JobLevel                   0.265986
 JobRole                    0.557313
 JobSatisfaction            0.576190
 MaritalStatus              0.548639
 MonthlyIncome              0.289307
 NumCompaniesWorked         0.299244
 OverTime                   0.282993
 PercentSalaryHike          0.300680
 PerformanceRating          0.153741
 StockOptionLevel           0.264626
 TotalWorkingYears          0.281990
 TrainingTimesLastYear      0.466553
 YearsAtCompany             0.309796
 YearsSinceLastPromotion    0.145850
 YearsWithCurrManager       0.242537
 Higher_Education           0.508844
 Status_of_leaving          0.497109
 Mode_of_work               0.522449
 Leaves                     0.513741
 Absenteeism                0.508390
 Work_accident              0.499320
 Source_of_Hire             0.501134
 Job_mode                   0.496259
 DistanceFromHome_group     0.258776
 dtype: float64,
 'median': Age                        0.428571
 Attrition                  0.000000
 BusinessTravel             1.000000
 Department                 0.500000
 DistanceFromHome           0.214286
 Gender                     1.000000
 JobInvolvement             0.666667
 JobLevel                   0.250000
 JobRole                    0.625000
 JobSatisfaction            0.666667
 MaritalStatus              0.500000
 MonthlyIncome              0.205898
 NumCompaniesWorked         0.222222
 OverTime                   0.000000
 PercentSalaryHike          0.214286
 PerformanceRating          0.000000
 StockOptionLevel           0.333333
 TotalWorkingYears          0.250000
 TrainingTimesLastYear      0.500000
 YearsAtCompany             0.200000
 YearsSinceLastPromotion    0.066667
 YearsWithCurrManager       0.176471
 Higher_Education           0.666667
 Status_of_leaving          0.500000
```

```
Mode_of_work              1.000000
Leaves                    0.600000
Absenteeism               0.666667
Work_accident             0.000000
Source_of_Hire            0.666667
Job_mode                  0.500000
DistanceFromHome_group    0.200000
dtype: float64,
'mode':          Age  Attrition  BusinessTravel  Department  DistanceFromHome  Gender  \
0  0.404762        0.0             1.0         0.5          0.035714     1.0

   JobInvolvement  JobLevel  JobRole  JobSatisfaction  ...  \
0        0.666667       0.0    0.875              1.0  ...

   YearsWithCurrManager  Higher_Education  Status_of_leaving  Mode_of_work  \
0              0.117647               1.0               0.25           1.0

   Leaves  Absenteeism  Work_accident  Source_of_Hire  Job_mode  \
0     0.8     0.333333            0.0        0.666667       0.5

   DistanceFromHome_group
0                     0.0

[1 rows x 31 columns],
'standard deviation': Age                         0.217509
Attrition                 0.367863
BusinessTravel            0.332727
Department                0.263896
DistanceFromHome          0.289531
Gender                    0.490065
JobInvolvement            0.237187
JobLevel                  0.276735
JobRole                   0.307728
JobSatisfaction           0.367615
MaritalStatus             0.365060
MonthlyIncome             0.247918
NumCompaniesWorked        0.277557
OverTime                  0.450606
PercentSalaryHike         0.261424
PerformanceRating         0.360824
StockOptionLevel          0.284026
TotalWorkingYears         0.194520
TrainingTimesLastYear     0.214878
YearsAtCompany            0.211705
YearsSinceLastPromotion   0.214829
YearsWithCurrManager      0.209890
Higher_Education          0.374724
Status_of_leaving         0.350945
Mode_of_work              0.499666
Leaves                    0.343234
Absenteeism               0.365952
Work_accident             0.500170
Source_of_Hire            0.372397
Job_mode                  0.402705
DistanceFromHome_group    0.311567
dtype: float64,
'variance': Age                         0.047310
Attrition                 0.135323
BusinessTravel            0.110708
Department                0.069641
DistanceFromHome          0.083828
Gender                    0.240163
JobInvolvement            0.056258
JobLevel                  0.076582
JobRole                   0.094696
JobSatisfaction           0.135141
```

```
MaritalStatus              0.133269
MonthlyIncome              0.061463
NumCompaniesWorked         0.077038
OverTime                   0.203046
PercentSalaryHike          0.068343
PerformanceRating          0.130194
StockOptionLevel           0.080671
TotalWorkingYears          0.037838
TrainingTimesLastYear      0.046173
YearsAtCompany             0.044819
YearsSinceLastPromotion    0.046151
YearsWithCurrManager       0.044054
Higher_Education           0.140418
Status_of_leaving          0.123162
Mode_of_work               0.249666
Leaves                     0.117810
Absenteeism                0.133921
Work_accident              0.250170
Source_of_Hire             0.138680
Job_mode                   0.162171
DistanceFromHome_group     0.097074
dtype: float64,
'skewness': Age                            0.413286
Attrition                  1.844366
BusinessTravel            -1.439006
Department                 0.172231
DistanceFromHome           0.958118
Gender                    -0.408665
JobInvolvement            -0.498419
JobLevel                   1.025401
JobRole                   -0.357270
JobSatisfaction           -0.329672
MaritalStatus             -0.152175
MonthlyIncome              1.369817
NumCompaniesWorked         1.026471
OverTime                   0.964489
PercentSalaryHike          0.821128
PerformanceRating          1.921883
StockOptionLevel           0.968980
TotalWorkingYears          1.117172
TrainingTimesLastYear      0.553124
YearsAtCompany             0.628168
YearsSinceLastPromotion    1.984290
YearsWithCurrManager       0.833451
Higher_Education          -0.024488
Status_of_leaving          0.029431
Mode_of_work              -0.089978
Leaves                    -0.087394
Absenteeism               -0.014666
Work_accident              0.002724
Source_of_Hire            -0.024668
Job_mode                   0.013560
DistanceFromHome_group     1.087165
dtype: float64,
'kurtosis': Age                           -0.404145
Attrition                  1.403594
BusinessTravel             0.702686
Department                -0.391435
DistanceFromHome          -0.224833
Gender                    -1.835492
JobInvolvement             0.270999
JobLevel                   0.399152
JobRole                   -1.192735
JobSatisfaction           -1.222193
MaritalStatus             -1.115037
MonthlyIncome              1.005233
```

```
NumCompaniesWorked          0.010214
OverTime                   -1.071221
PercentSalaryHike          -0.300598
PerformanceRating           1.695939
StockOptionLevel            0.364634
TotalWorkingYears           0.918270
TrainingTimesLastYear       0.494993
YearsAtCompany              0.357981
YearsSinceLastPromotion     3.612673
YearsWithCurrManager        0.171058
Higher_Education           -1.373634
Status_of_leaving          -1.283587
Mode_of_work               -1.994620
Leaves                     -1.284705
Absenteeism                -1.313232
Work_accident              -2.002719
Source_of_Hire             -1.357468
Job_mode                   -1.458127
DistanceFromHome_group     -0.040943
dtype: float64}
```

Now we try and visualise the factors that effect the attrtion most using the stacked plots as under.

Not only does it give a better understanding but the visuals help select the features better.

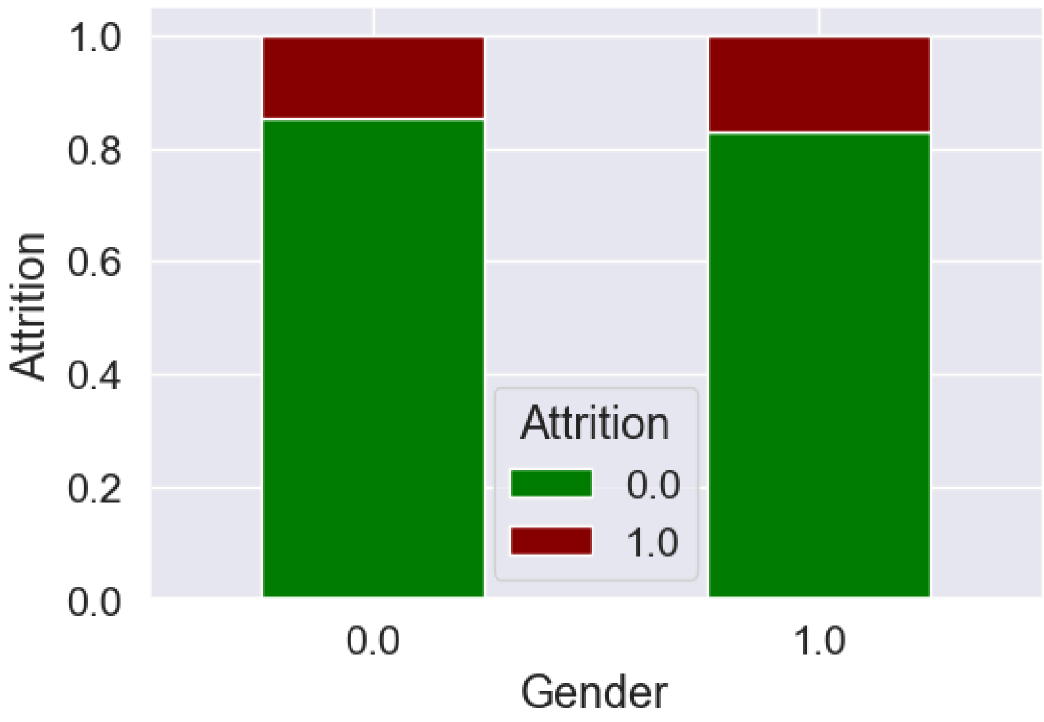
In [579…

```
df_company = attrition_mms
```

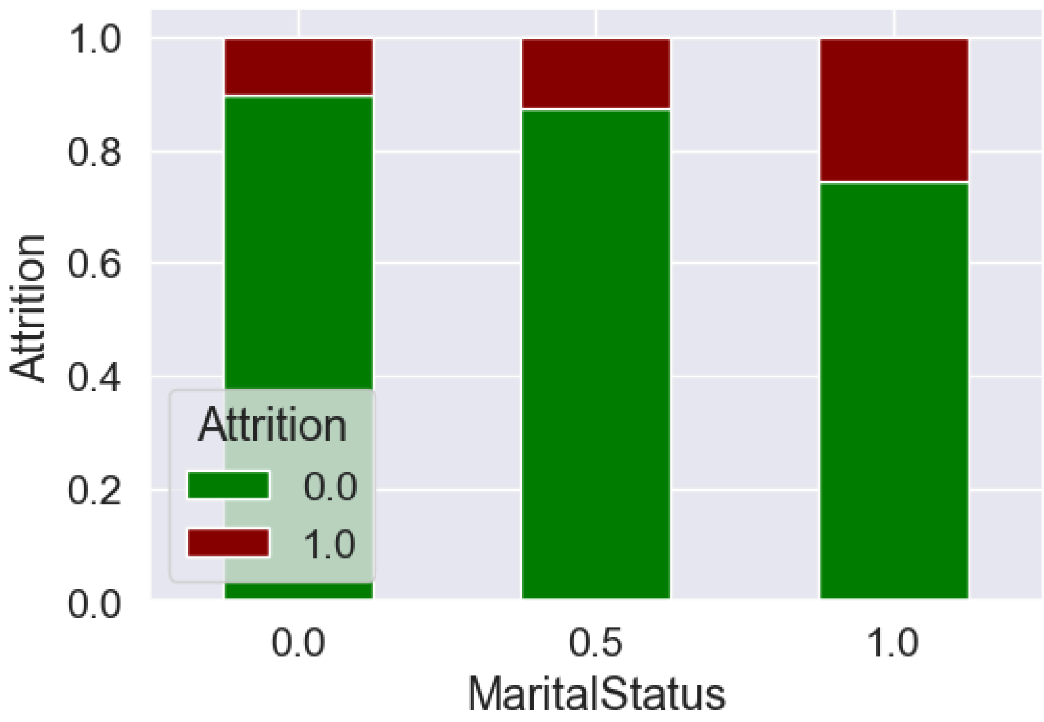
In [581…

```
stacked_plot(df_company, "Gender", "Attrition")
stacked_plot(df_company, "MaritalStatus", "Attrition")
stacked_plot(df_company, "BusinessTravel", "Attrition")
stacked_plot(df_company, "Department", "Attrition")
stacked_plot(df_company, "JobInvolvement", "Attrition")
stacked_plot(df_company, "JobRole", "Attrition")
stacked_plot(df_company, "JobLevel", "Attrition")
stacked_plot(df_company, "JobSatisfaction", "Attrition")
stacked_plot(df_company, "NumCompaniesWorked", "Attrition")
stacked_plot(df_company, "OverTime", "Attrition")
stacked_plot(df_company, "PercentSalaryHike", "Attrition")
stacked_plot(df_company, "PerformanceRating", "Attrition")
stacked_plot(df_company, "StockOptionLevel", "Attrition")
stacked_plot(df_company, "TrainingTimesLastYear", "Attrition")
stacked_plot(df_company, "Higher_Education", "Attrition")
stacked_plot(df_company, "Status_of_leaving", "Attrition")
stacked_plot(df_company, "Mode_of_work", "Attrition")
stacked_plot(df_company, "Leaves", "Attrition")
stacked_plot(df_company, "Absenteeism", "Attrition")
stacked_plot(df_company, "Work_accident", "Attrition")
stacked_plot(df_company, "Source_of_Hire", "Attrition")
stacked_plot(df_company, "Job_mode", "Attrition")
```

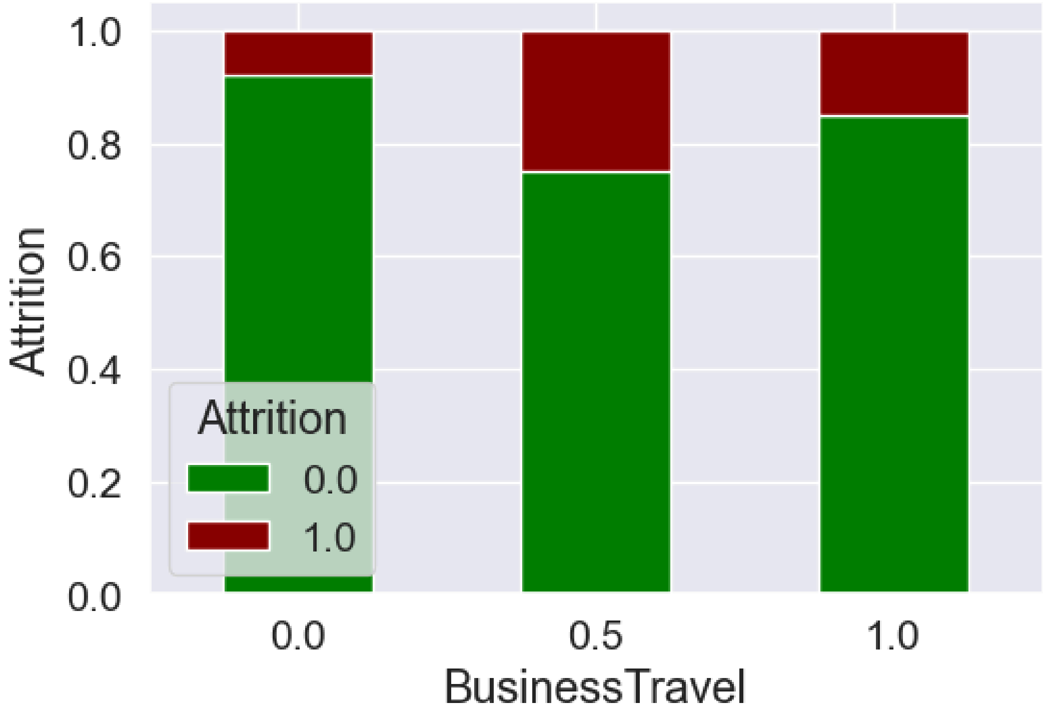
```
C:\Users\Rana\AppData\Local\Temp\ipykernel_22680\3410814390.py:10: RuntimeWarning: More than 20 fig
ures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) ar
e retained until explicitly closed and may consume too much memory. (To control this warning, see t
he rcParam `figure.max_open_warning`). Consider using `matplotlib.pyplot.close()`.
  fig, ax = plt.subplots(figsize = (6,4))
```



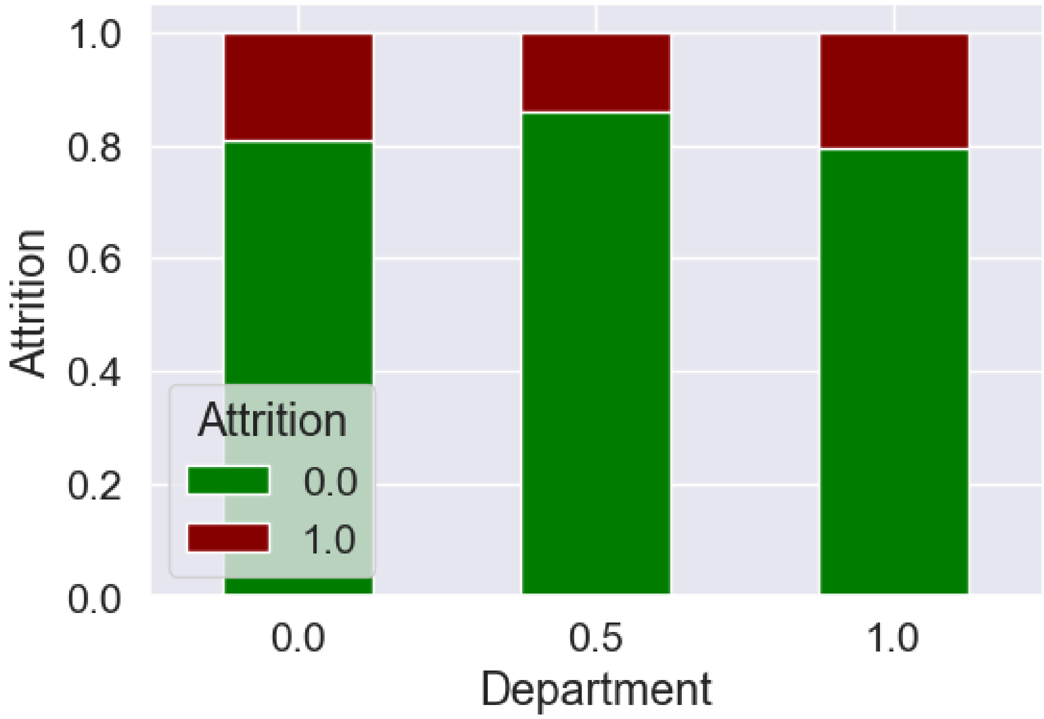
Attrition
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
Gender
Attrition
0.0
1.0



Attrition
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
MaritalStatus
Attrition
0.0
1.0

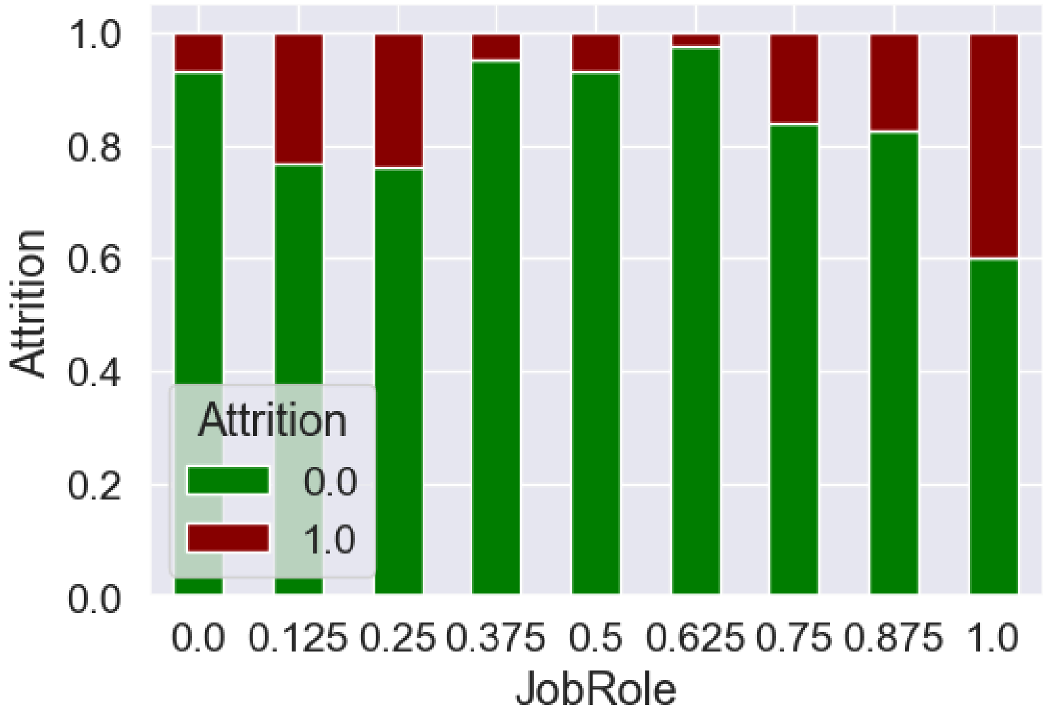


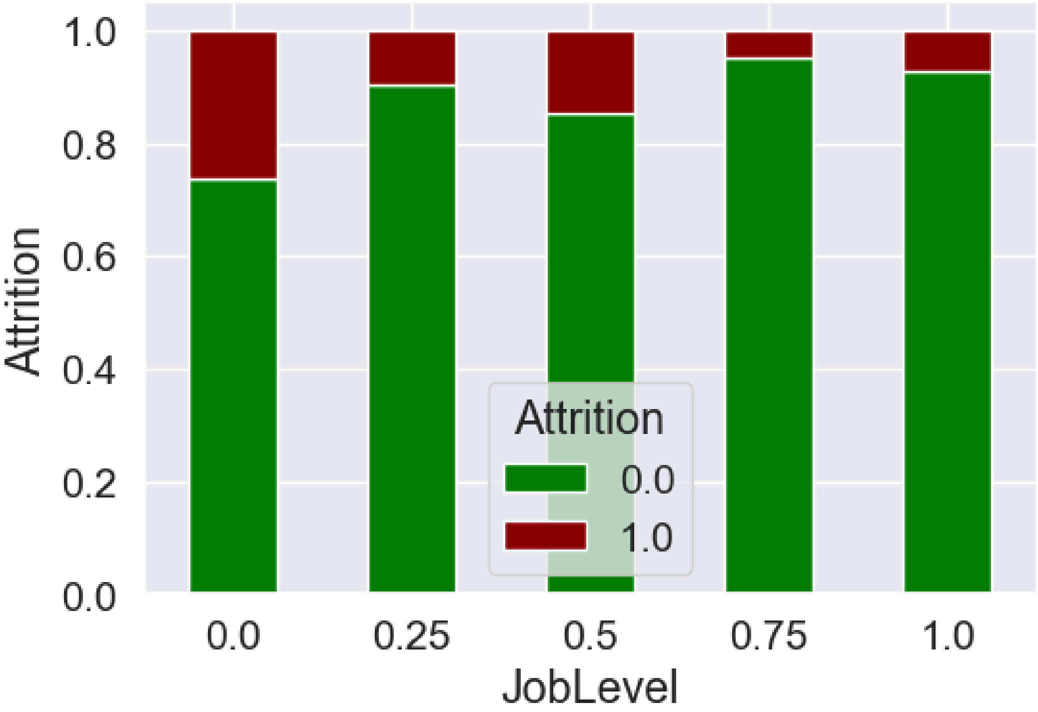
Attrition
BusinessTravel
Attrition
0.0
1.0



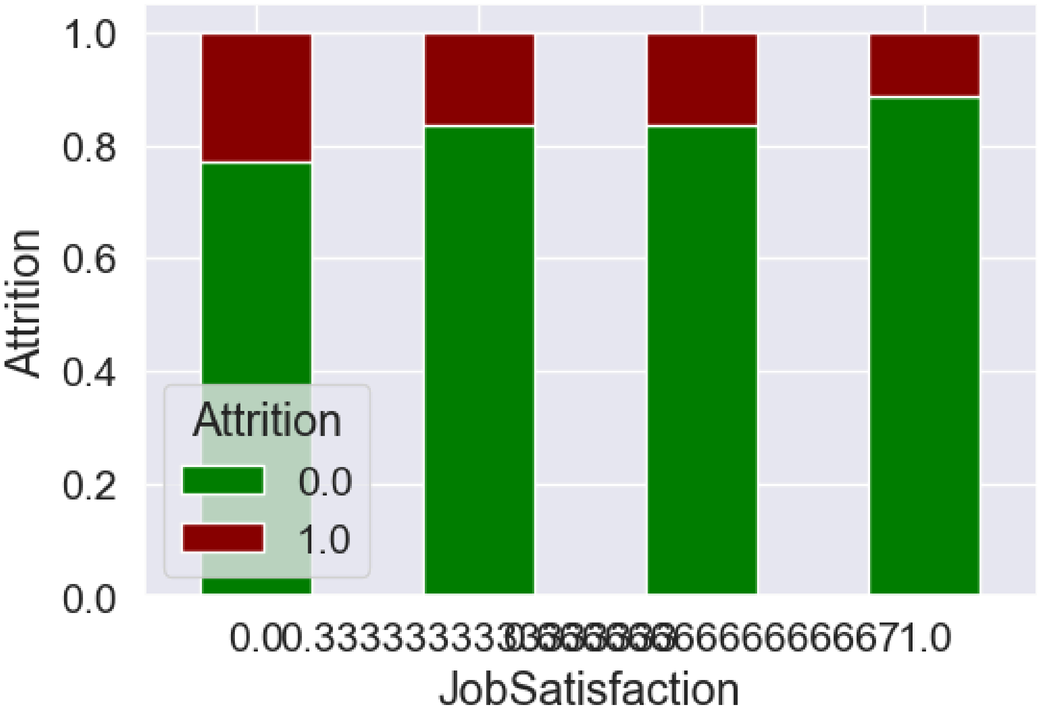
Attrition
Department
Attrition
0.0
1.0


Attrition
0.0
1.0
Attrition
1.0
0.8
0.6
0.4
0.2
0.0
0.00.3333333333333333666666666671.0
JobInvolvement

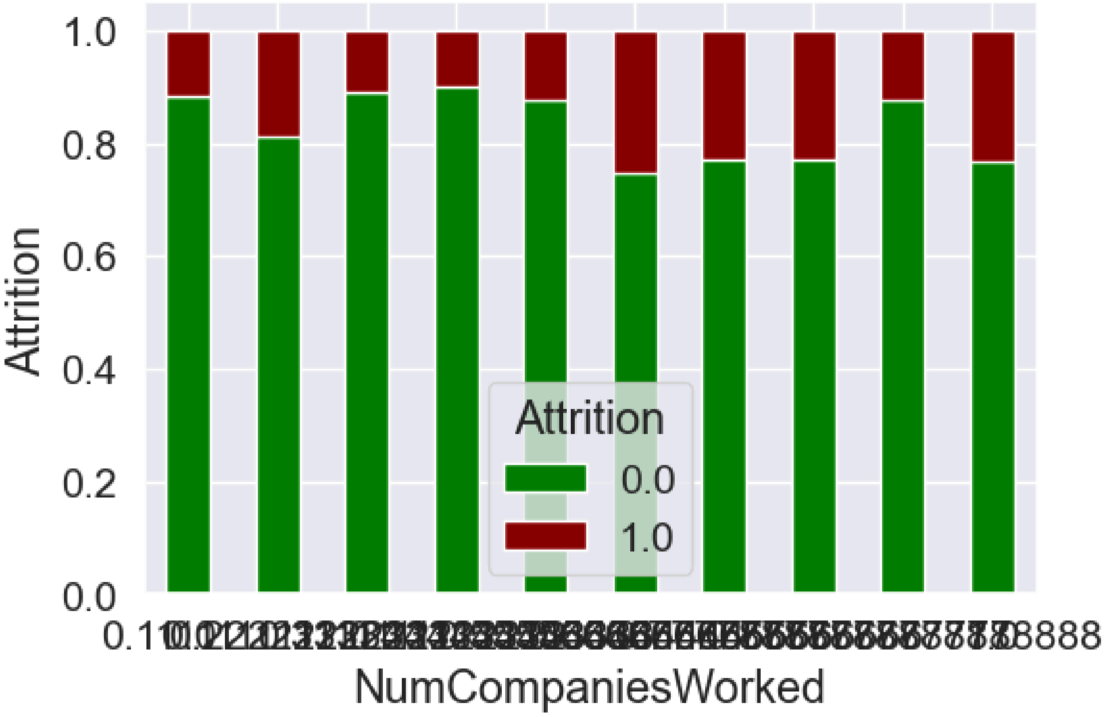

Attrition
0.0
1.0
Attrition
1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.125 0.25 0.375 0.5 0.625 0.75 0.875 1.0
JobRole

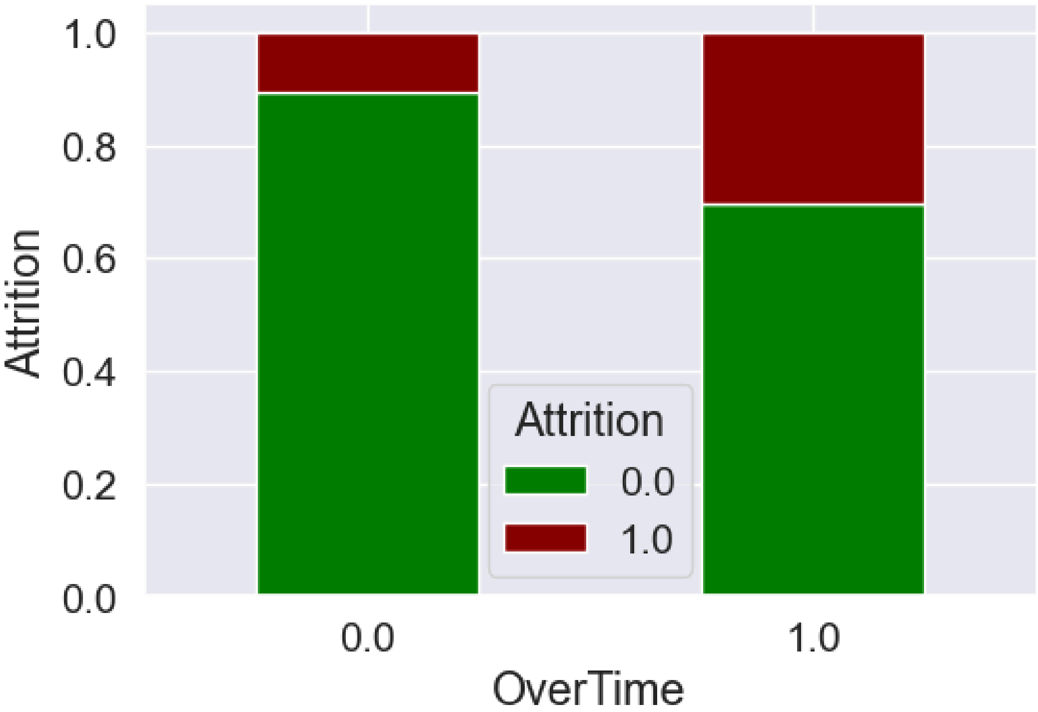


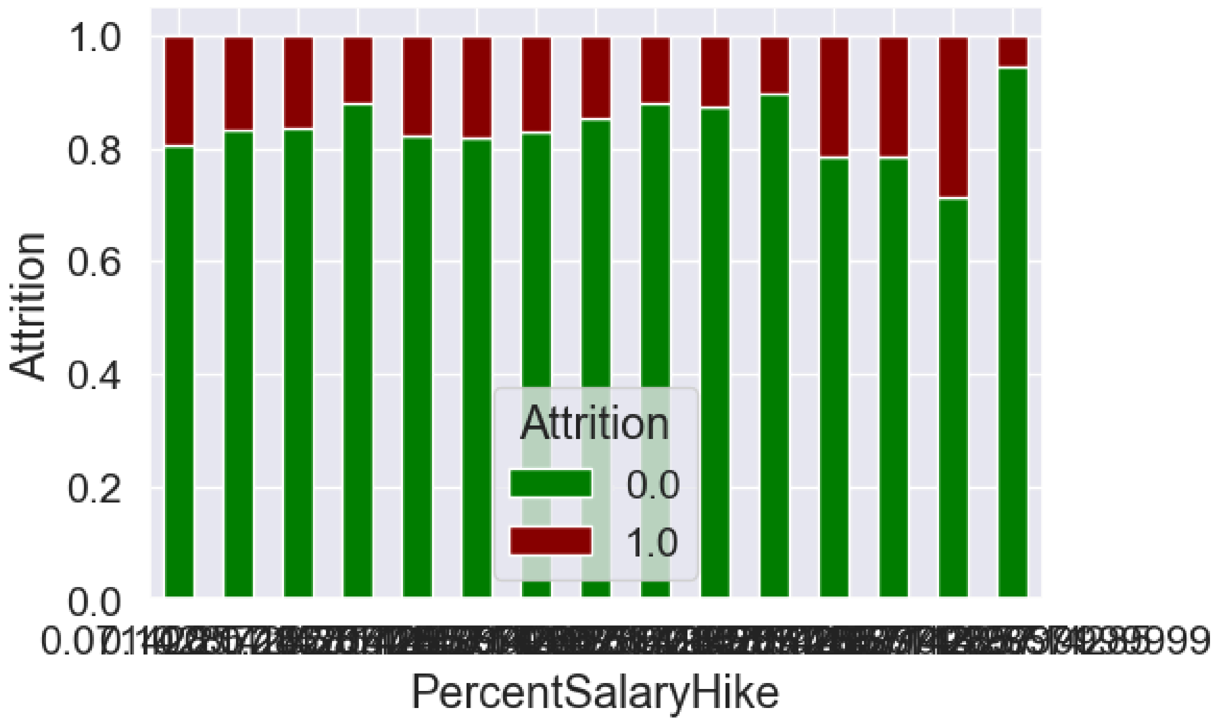
Attrition
1.0
0.8
0.6
0.4
0.2
0.0
Attrition
0.0
1.0
0.0
0.25
0.5
0.75
1.0
JobLevel

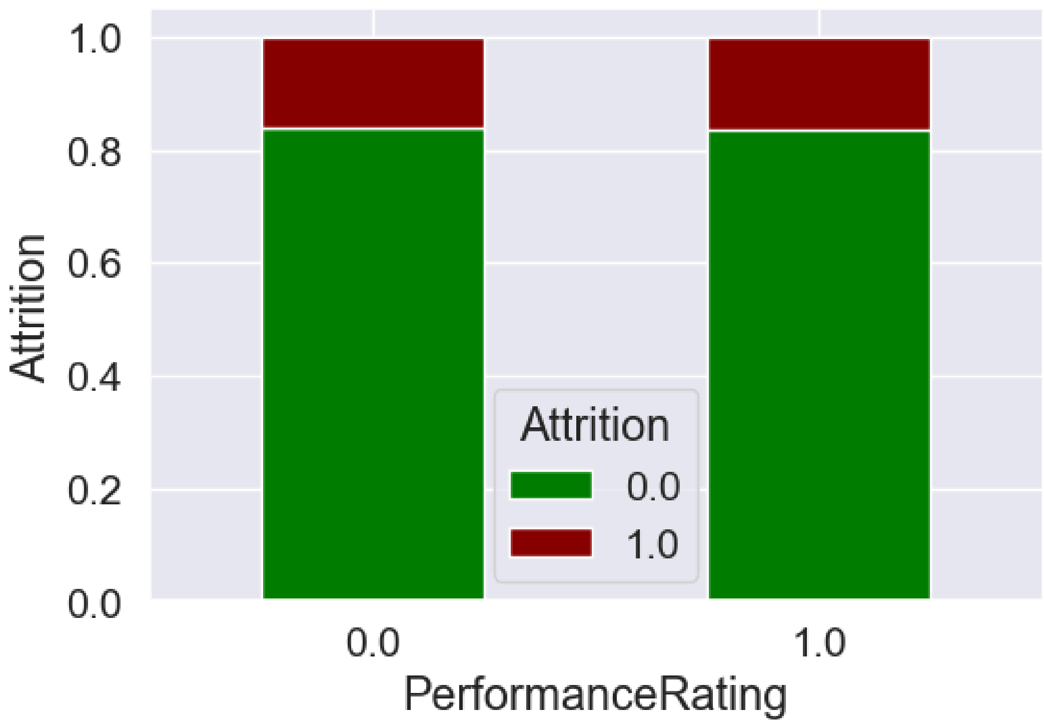


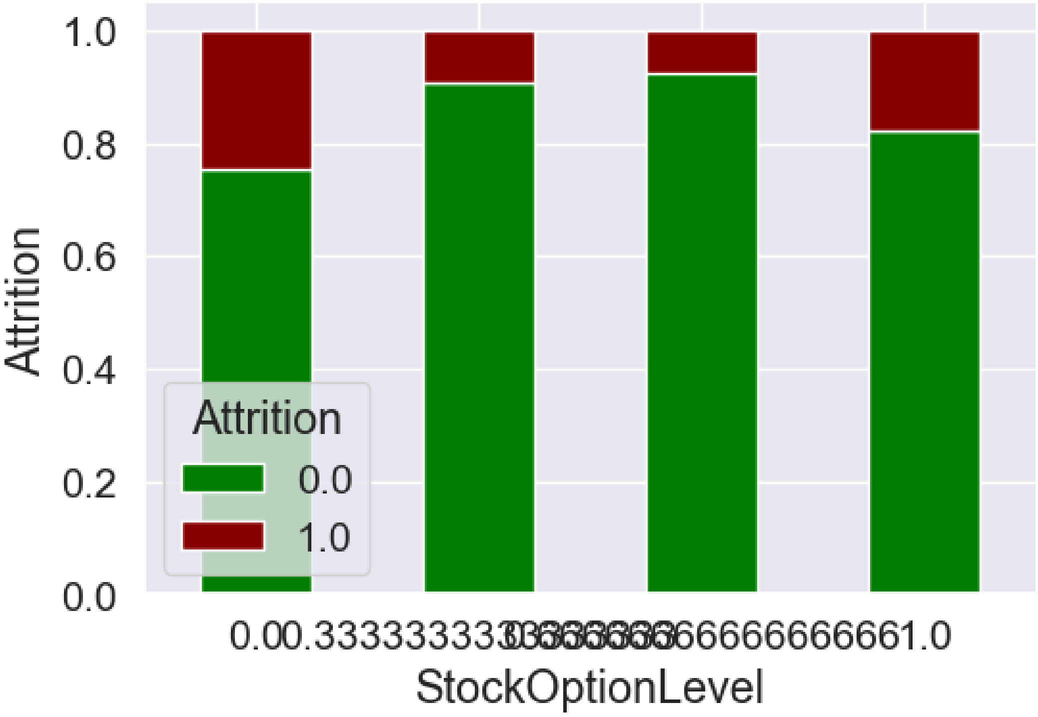
Attrition
1.0
0.8
0.6
0.4
0.2
0.0
Attrition
0.0
1.0
0.0
1.0
JobSatisfaction

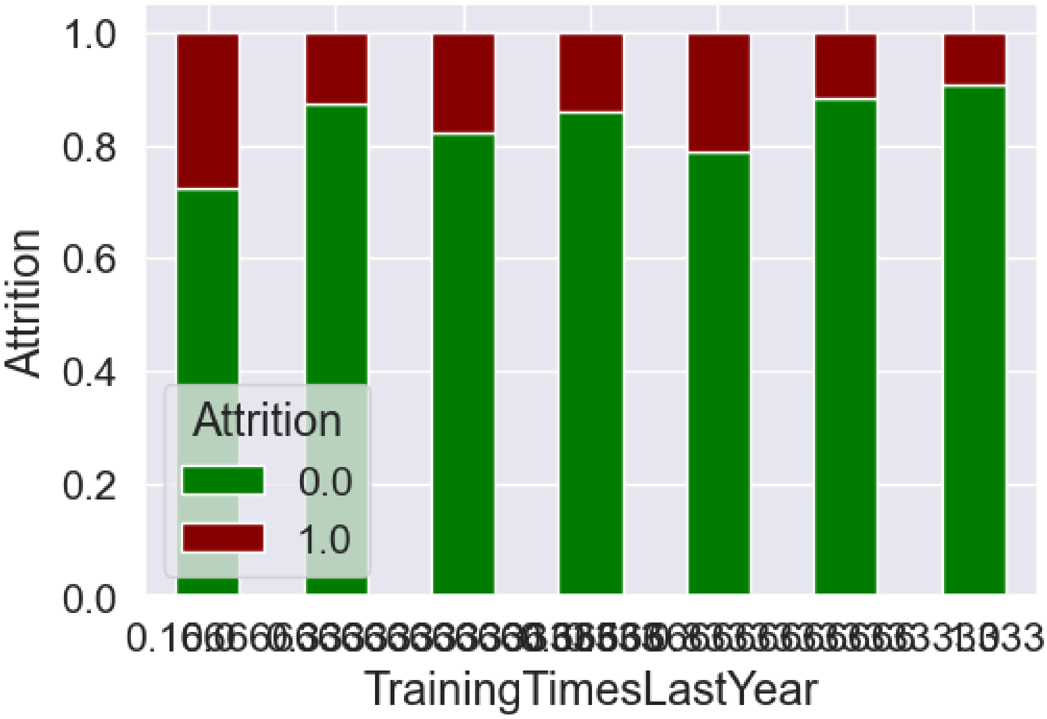

Attrition
1.0
0.8
0.6
0.4
0.2
0.0
Attrition
0.0
1.0
NumCompaniesWorked

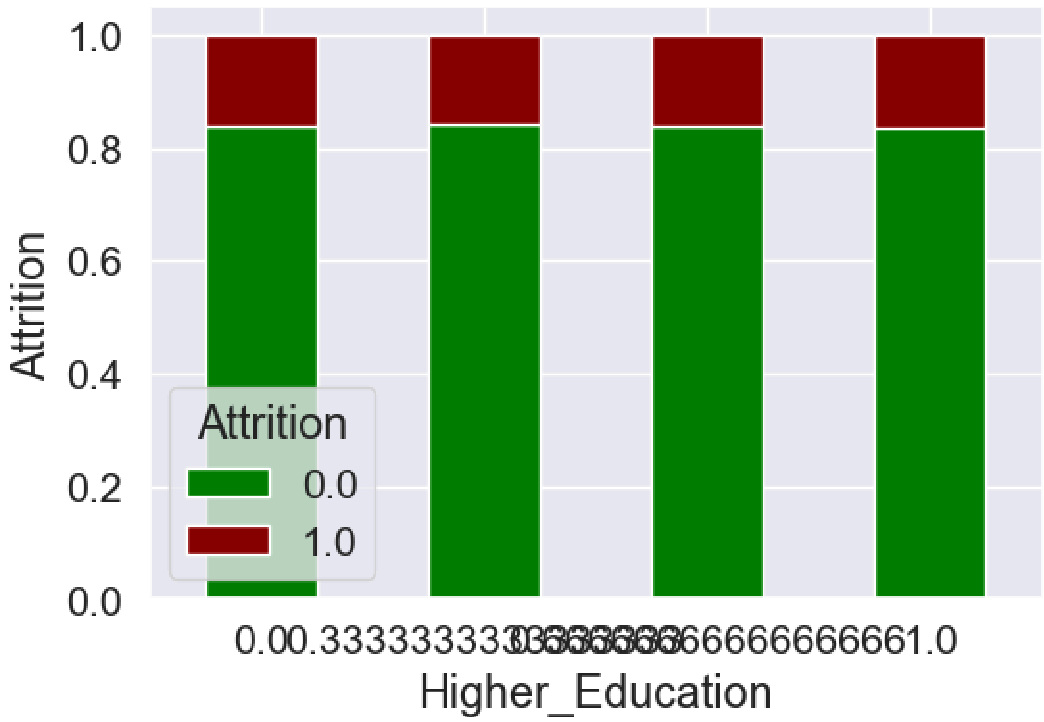

Attrition
1.0
0.8
0.6
0.4
0.2
0.0
Attrition
0.0
1.0
0.0
1.0
OverTime


1.0
0.8
0.6
0.4
0.2
0.0
Attrition
Attrition
0.0
1.0
PercentSalaryHike

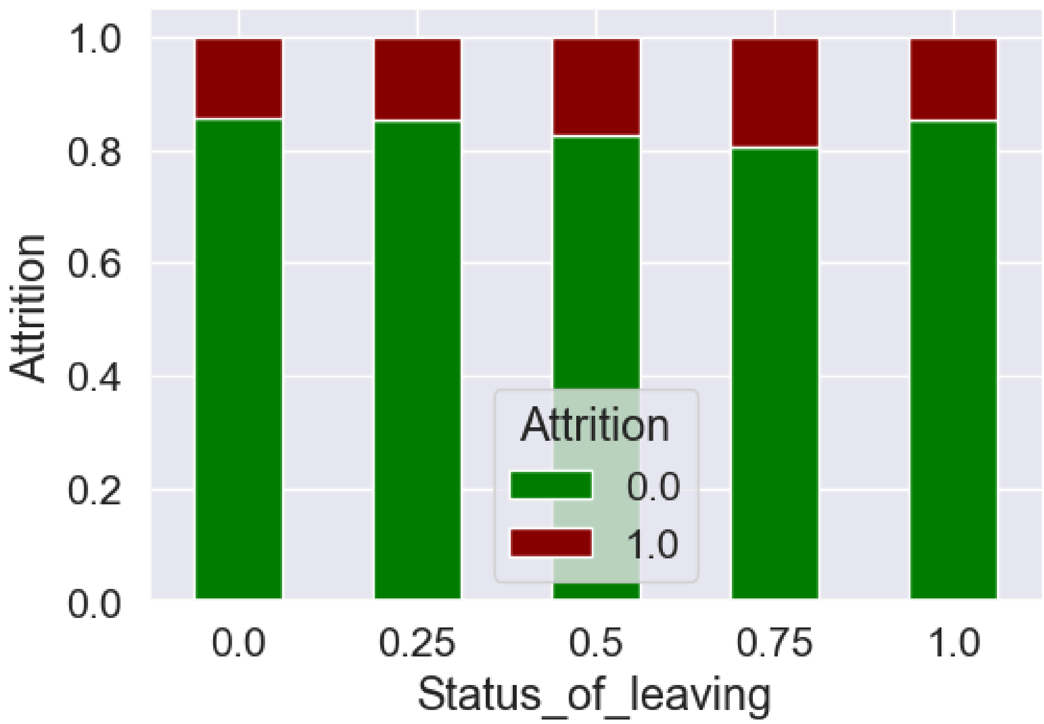

1.0
0.8
0.6
0.4
0.2
0.0
Attrition
Attrition
0.0
1.0
0.0
1.0
PerformanceRating


1.0
0.8
0.6
0.4
0.2
0.0
Attrition
Attrition
0.0
1.0
StockOptionLevel

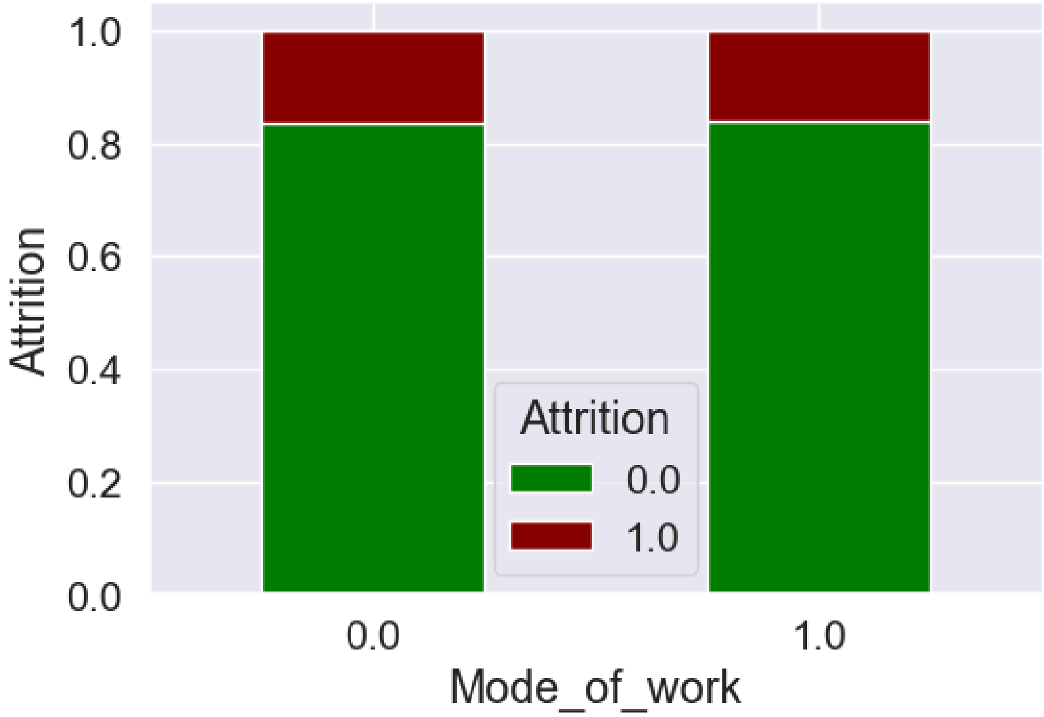

1.0
0.8
0.6
0.4
0.2
0.0
Attrition
Attrition
0.0
1.0
TrainingTimesLastYear

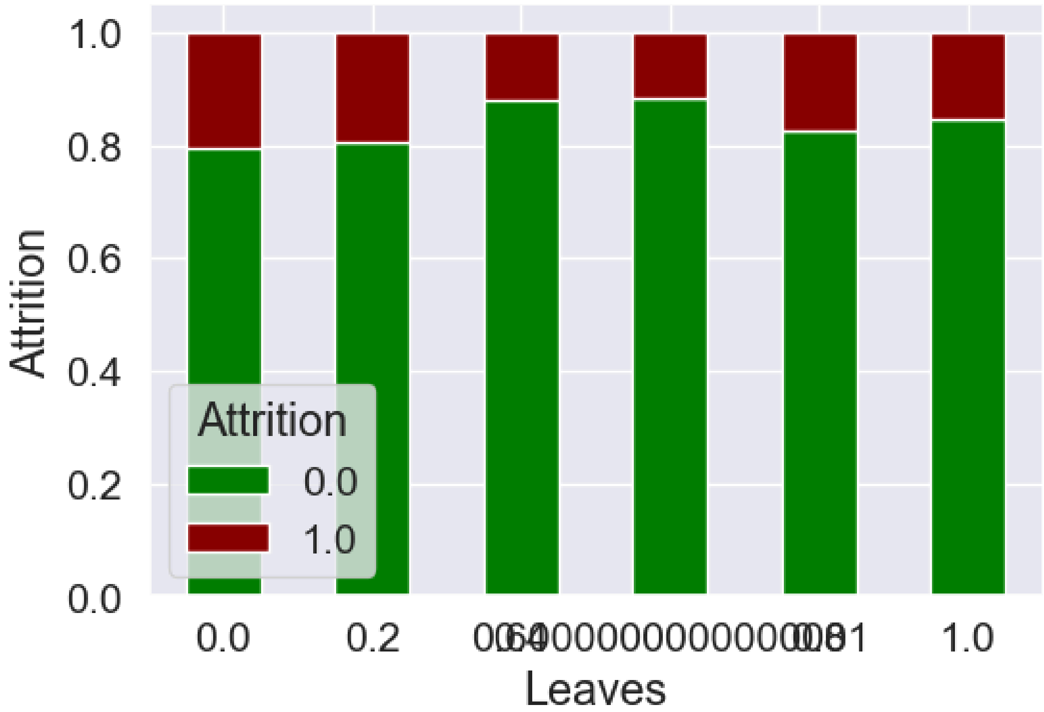


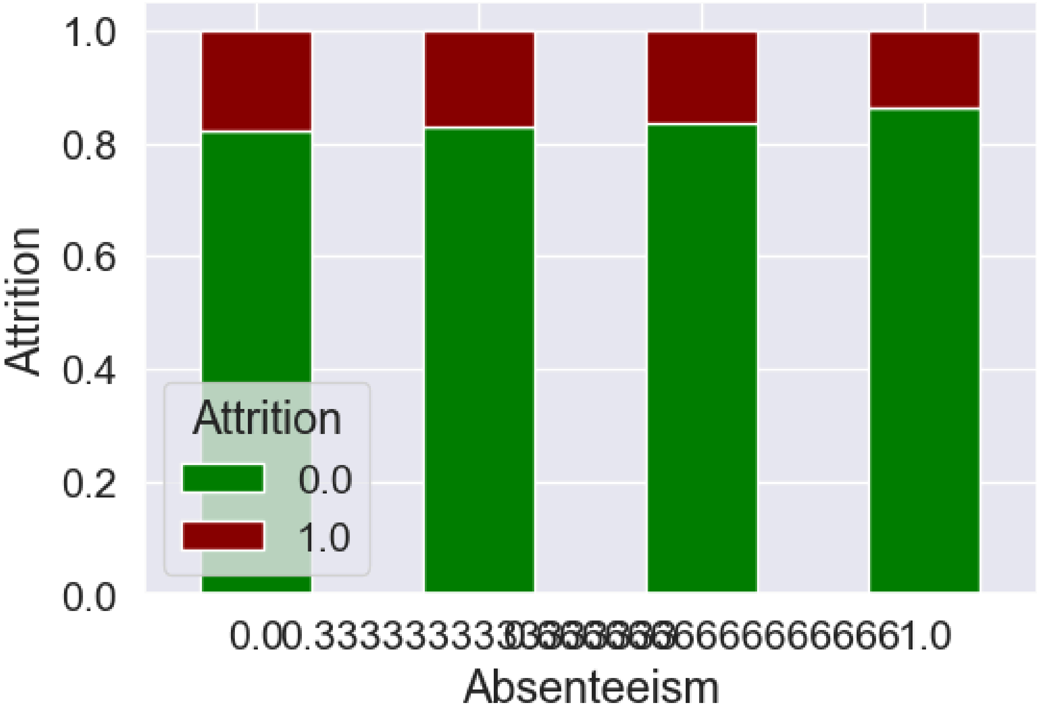
Attrition
1.0
0.8
0.6
0.4
0.2
0.0
Attrition
0.0
1.0
Higher_Education

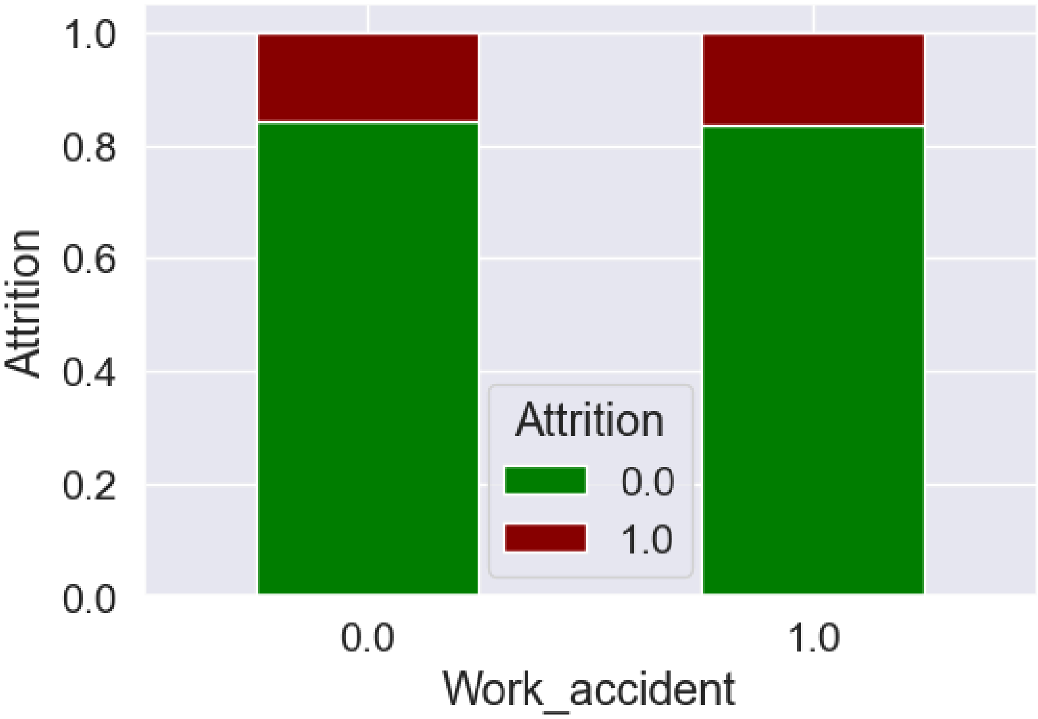


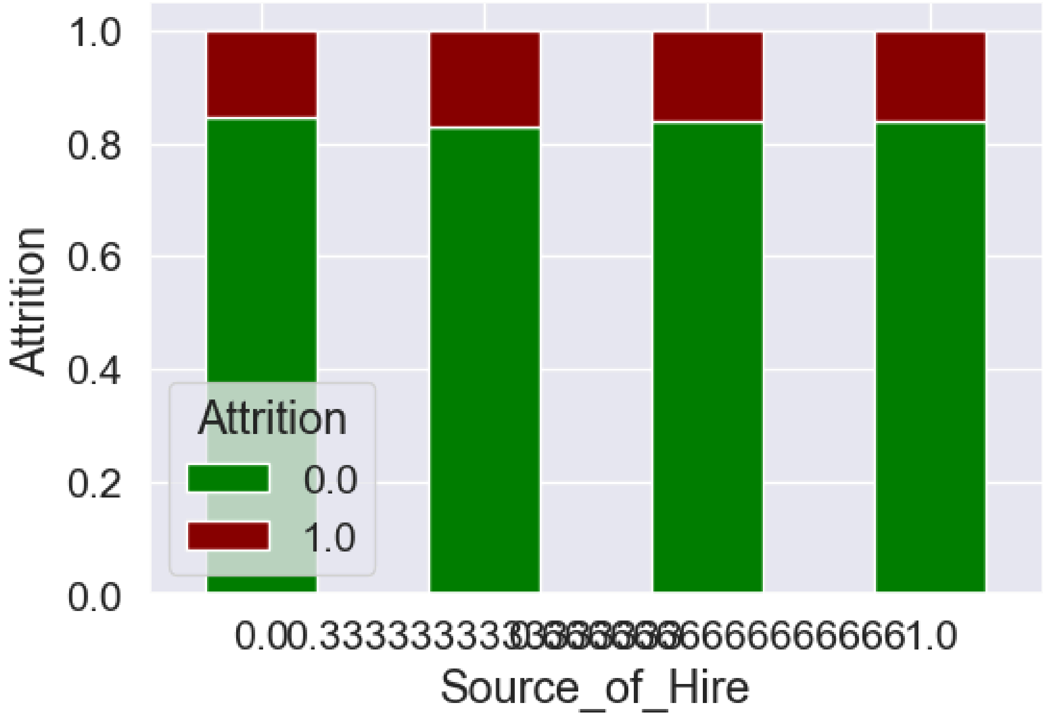
Attrition
1.0
0.8
0.6
0.4
0.2
0.0
Attrition
0.0
1.0
0.0
0.25
0.5
0.75
1.0
Status_of_leaving

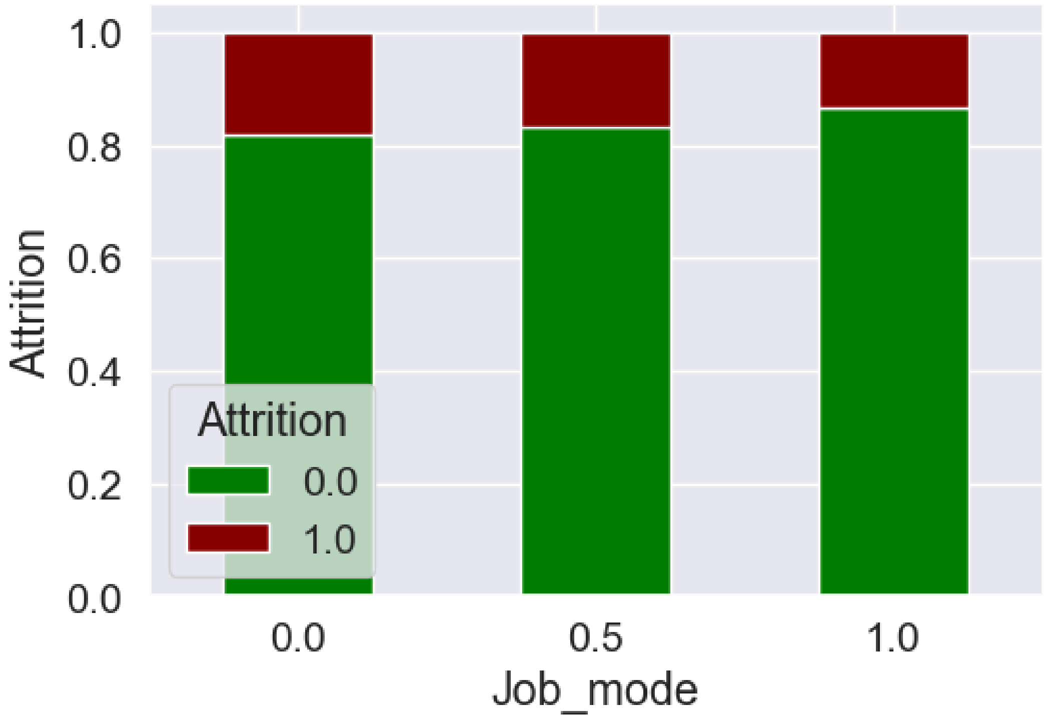

1.0
0.8
0.6
0.4
0.2
0.0
Attrition
Attrition
0.0
1.0
0.0
1.0
Mode_of_work


1.0
0.8
0.6
0.4
0.2
0.0
Attrition
Attrition
0.0
1.0
0.0
0.2
1.0
Leaves


1.0
0.8
0.6
0.4
0.2
0.0
Attrition
Attrition
0.0
1.0
Absenteeism

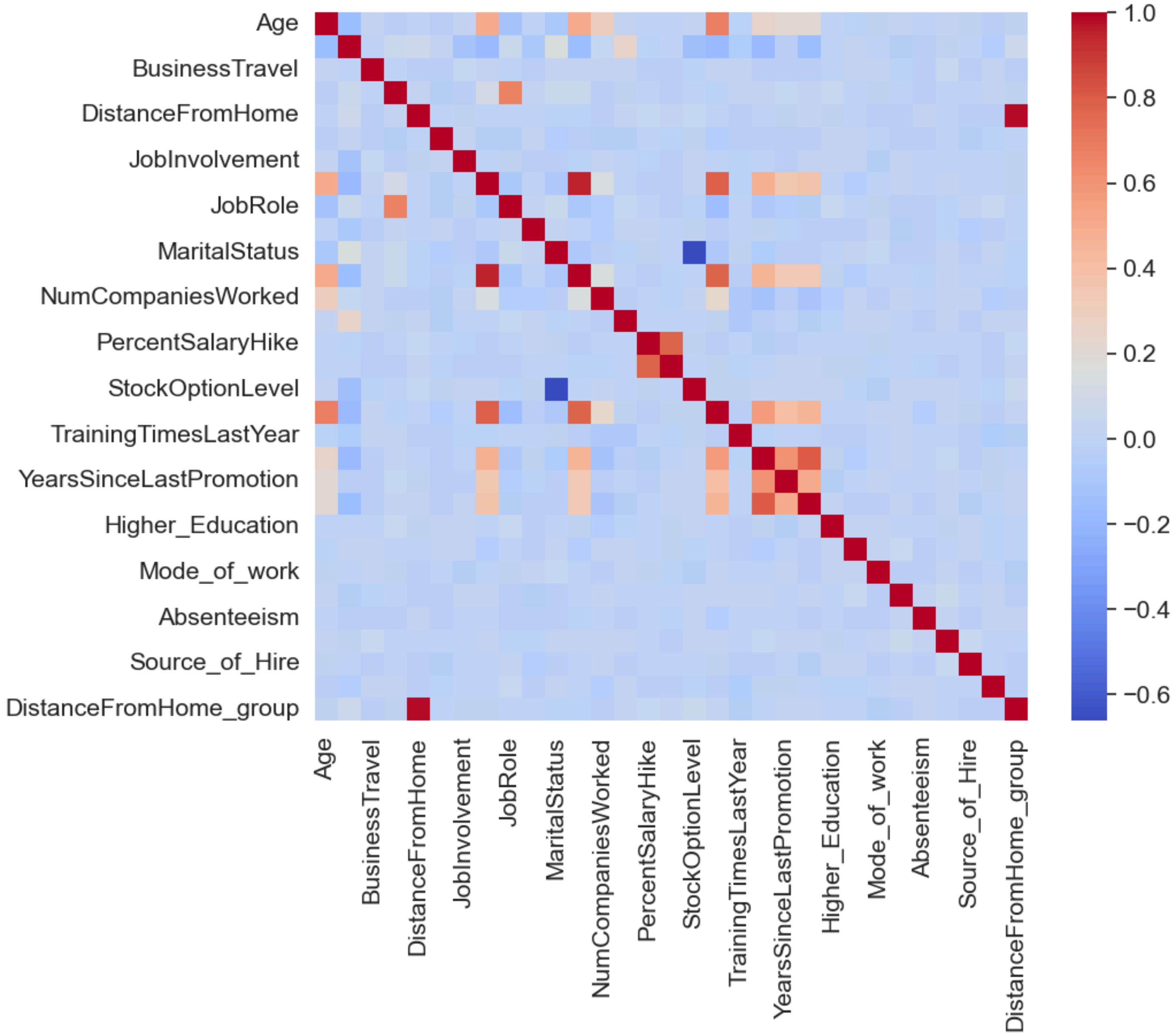

1.0
0.8
0.6
0.4
0.2
0.0
Attrition
Attrition
0.0
1.0
0.0
1.0
Work_accident



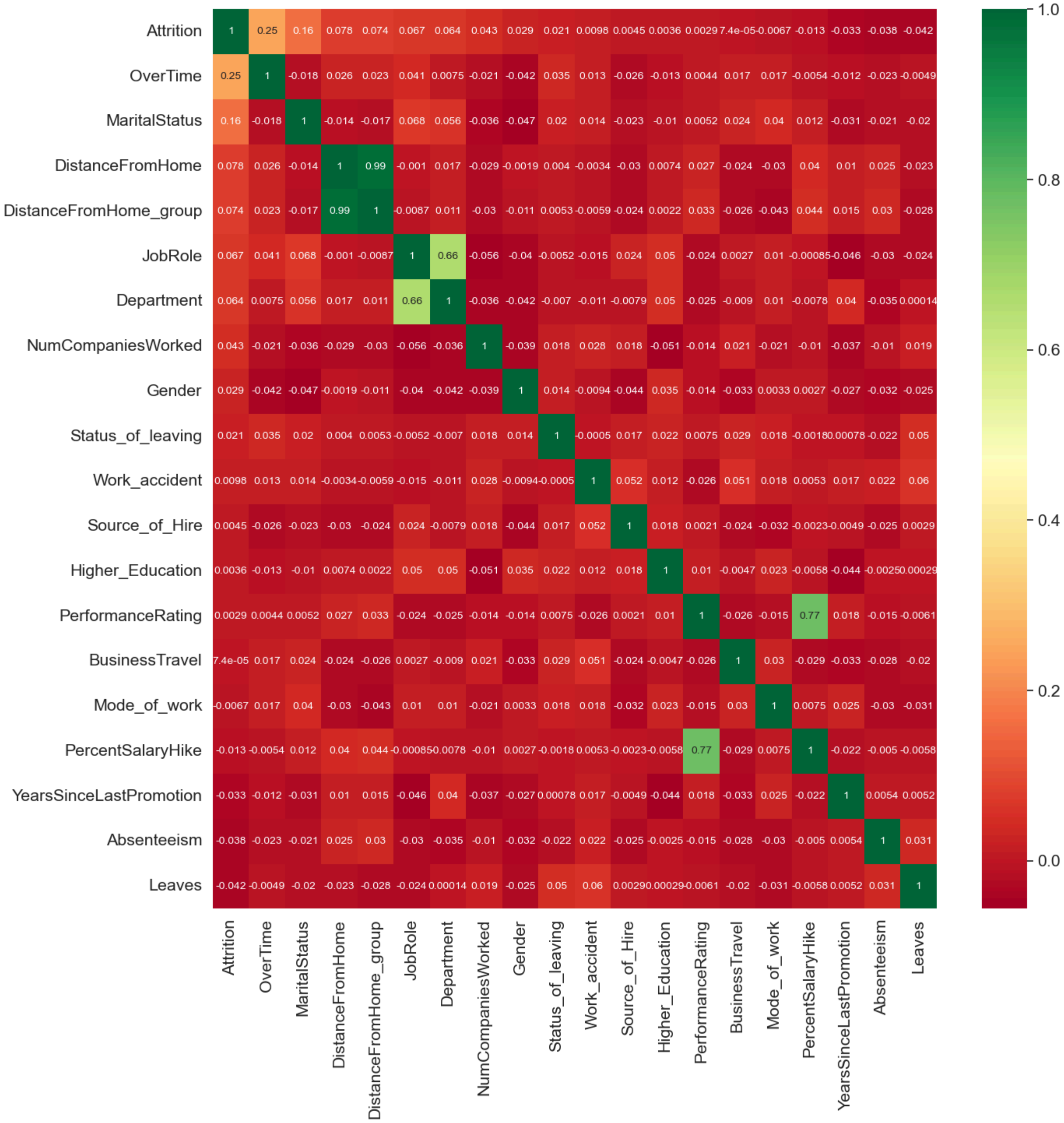


In [449…

```
################################################
# We plot the heat map to see the various relationships under correlation using the heatmap

plt.figure(figsize = (10,8))
sns.heatmap(df_company.corr(), annot = False, cmap = 'coolwarm')
plt.show()

# Checking the correlation coeficients and importance ordered
corr_attr = df_company.corr()
(corr_attr['Attrition'].sort_values(ascending = False))

col = df_company.corr().nlargest(20, "Attrition").Attrition.index
plt.figure(figsize=(15, 15))
sns.heatmap(df_company[col].corr(), annot = True, cmap = "RdYlGn", annot_kws = {"size":10})
```


Age
BusinessTravel
DistanceFromHome
JobInvolvement
JobRole
MaritalStatus
NumCompaniesWorked
PercentSalaryHike
StockOptionLevel
TrainingTimesLastYear
YearsSinceLastPromotion
Higher_Education
Mode_of_work
Absenteeism
Source_of_Hire
DistanceFromHome_group
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

Out[449… <Axes: >



We use now various features that are impactful on the attrition and try to check the survival analysis over them to determine the duration

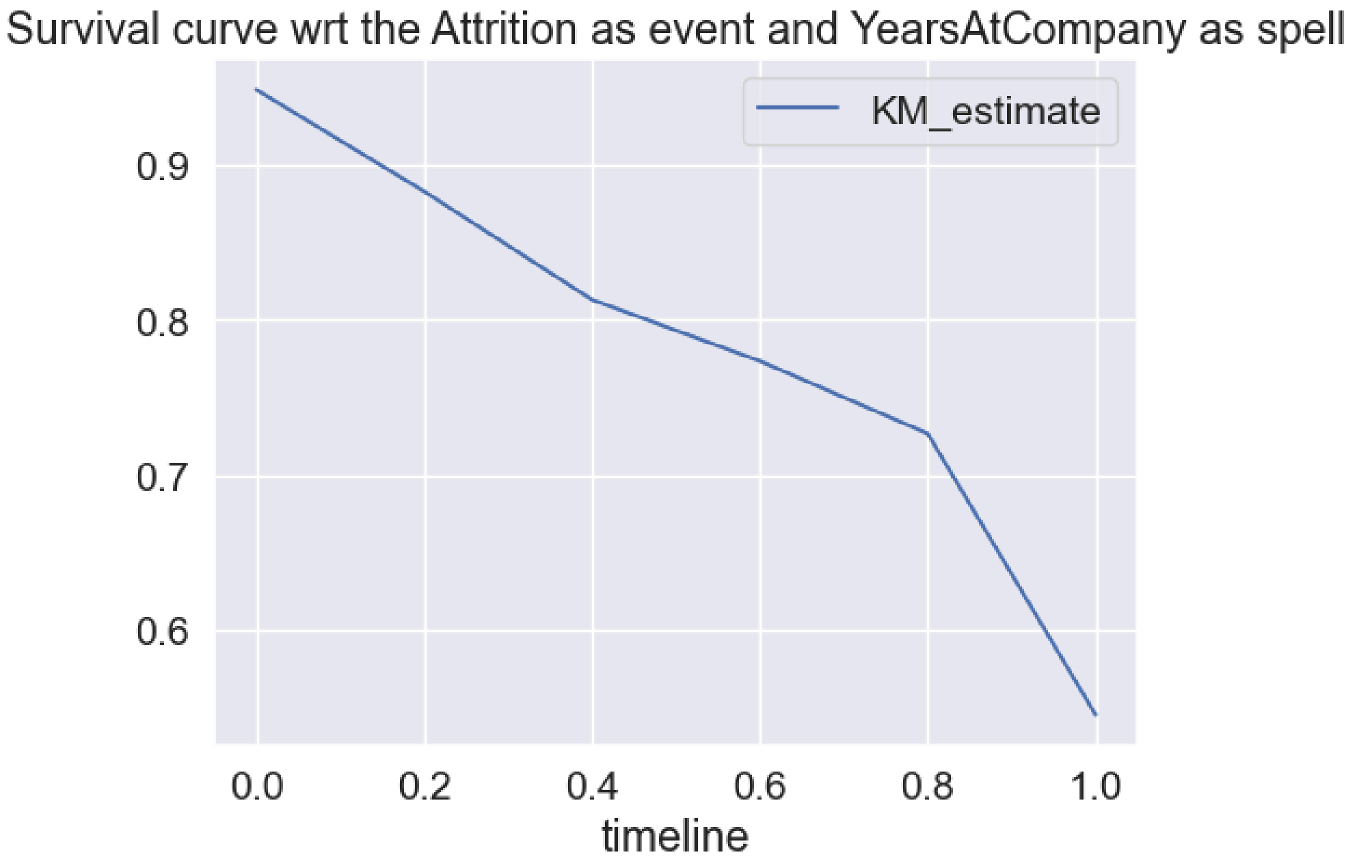
In [451…

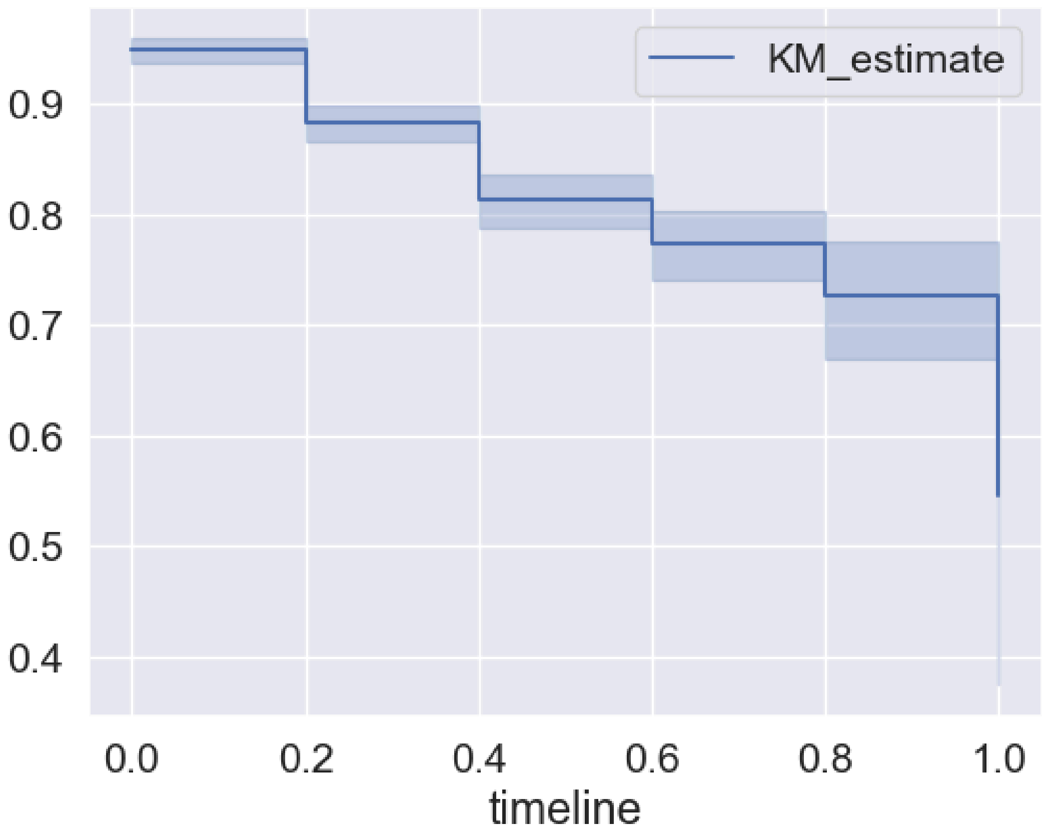
```
# !pip install lifelines
# import lifelines
```

In [452…

```
# Taking "YearsAtCompany" to be time spell
T = df.YearsAtCompany

# Importing the KaplanMeierFitter model to fit the survival analysis
from lifelines import KaplanMeierFitter
# Initiating the KaplanMeierFitter model
kmf = KaplanMeierFitter()
# Fitting KaplanMeierFitter model on Time and Events for Attrition
kmf.fit(durations = T, event_observed = df_company.Attrition)
# Time-line estimations plot
kmf.survival_function_.plot()
```

```python
plt.title('Survival curve wrt the Attrition as event and YearsAtCompany as spell')
plt.show()

# Print survival probabilities at each year
kmf.survival_function_

# Plot the survival function with confidence intervals
kmf.plot_survival_function()
plt.show()
```



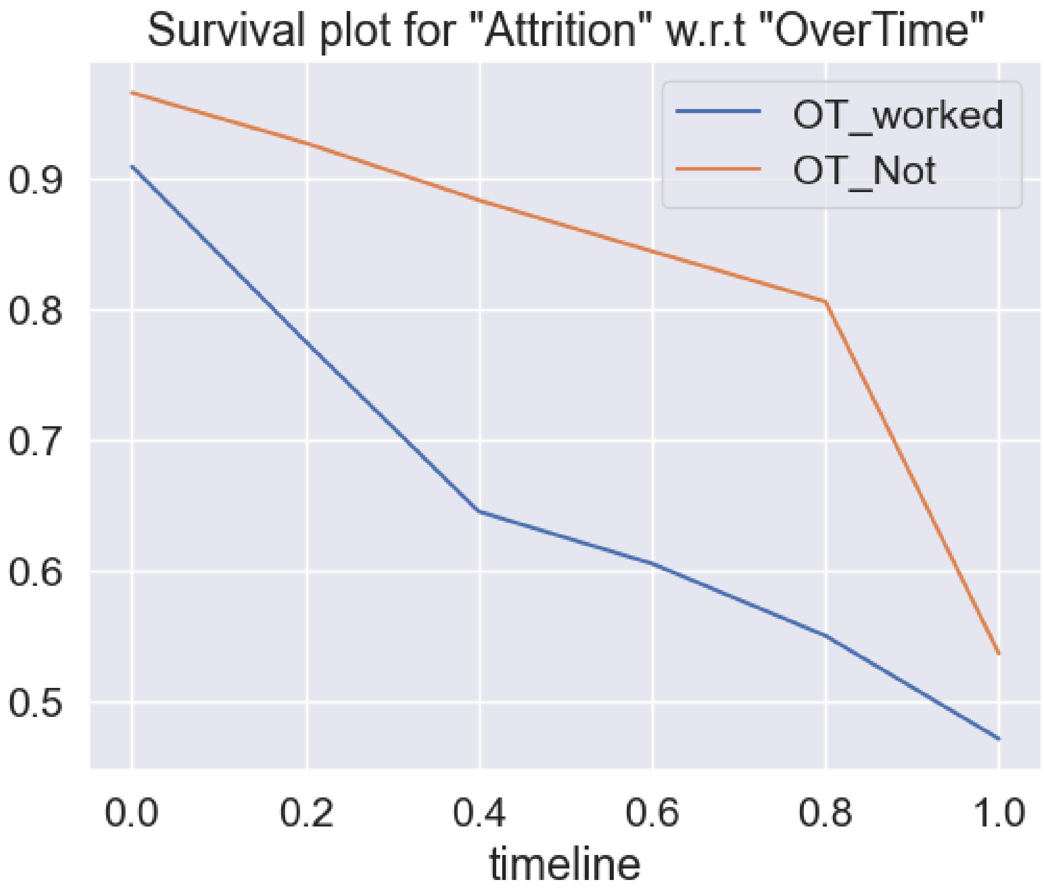


In [453…

```
################################################
# We try over Multiple groups with the event being "Attrition"
''' We first select the group to be OverTime'''
df_company.OverTime.value_counts()

OT_worked = df_company.OverTime == 1
OT_Not = df_company.OverTime == 0
# Applying KaplanMeierFitter model on Time and Events for the group "1"
kmf.fit(T[df_company.OverTime == 1], df_company.Attrition[df_company.OverTime == 1], label = 'OT_w
ax = kmf.survival_function_.plot()

# Applying KaplanMeierFitter model on Time and Events for the group "0"
kmf.fit(T[df_company.OverTime == 0], df_company.Attrition[df_company.OverTime == 0], label = 'OT_N
kmf.survival_function_.plot(ax=ax)
plt.title('Survival plot for "Attrition" w.r.t "OverTime"')
```

Out[453…

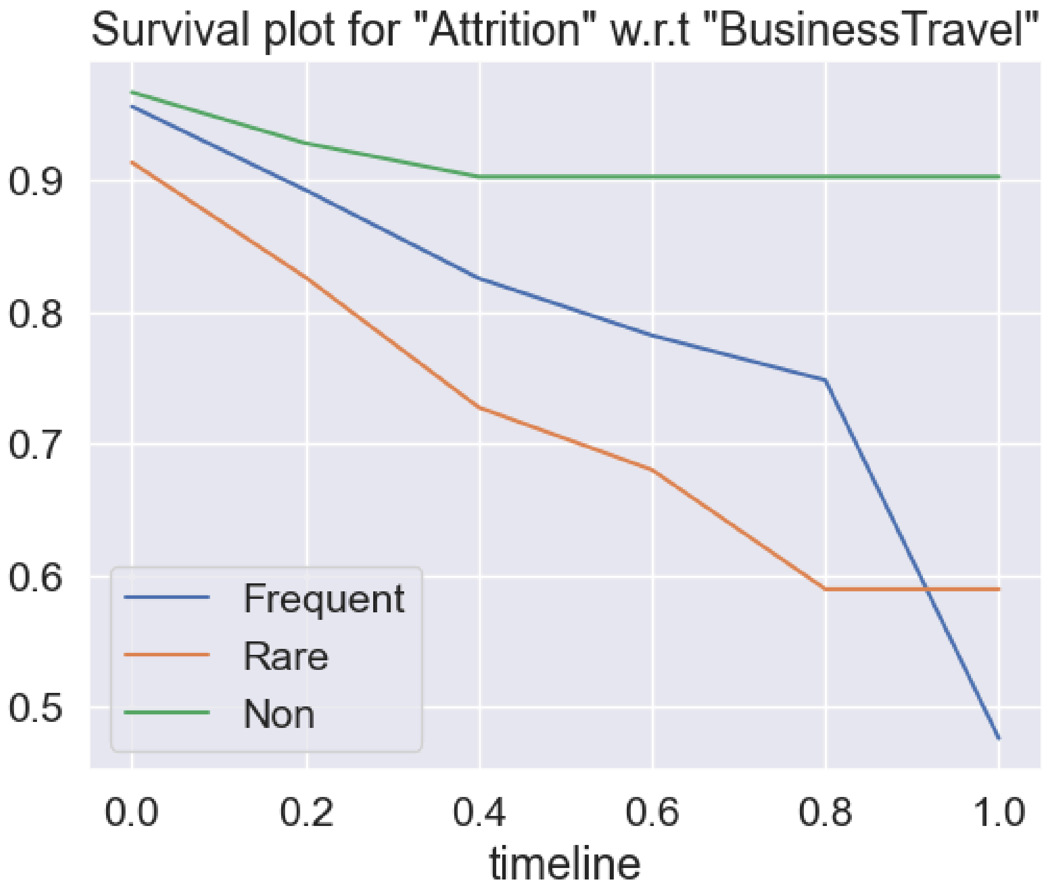
```
Text(0.5, 1.0, 'Survival plot for "Attrition" w.r.t "OverTime"')
```



In [454…

```
################################################
''' We now select the group to be BusinessTravel'''
df_company.BusinessTravel.value_counts()

Frequent = df_company.BusinessTravel == 1.00
Rare = df_company.BusinessTravel == 0.50
Non = df_company.BusinessTravel == 0.00
# Applying KaplanMeierFitter model on Time and Events for the group "1"
kmf.fit(T[df_company.BusinessTravel == 1], df_company.Attrition[df_company.BusinessTravel == 1], l
ax = kmf.survival_function_.plot()

# Applying KaplanMeierFitter model on Time and Events for the group "0.5"
kmf.fit(T[df_company.BusinessTravel == 0.5], df_company.Attrition[df_company.BusinessTravel == 0.5
kmf.survival_function_.plot(ax=ax)

# Applying KaplanMeierFitter model on Time and Events for the group "0"
kmf.fit(T[df_company.BusinessTravel == 0], df_company.Attrition[df_company.BusinessTravel == 0], l
kmf.survival_function_.plot(ax=ax)
plt.title('Survival plot for "Attrition" w.r.t "BusinessTravel"')
```

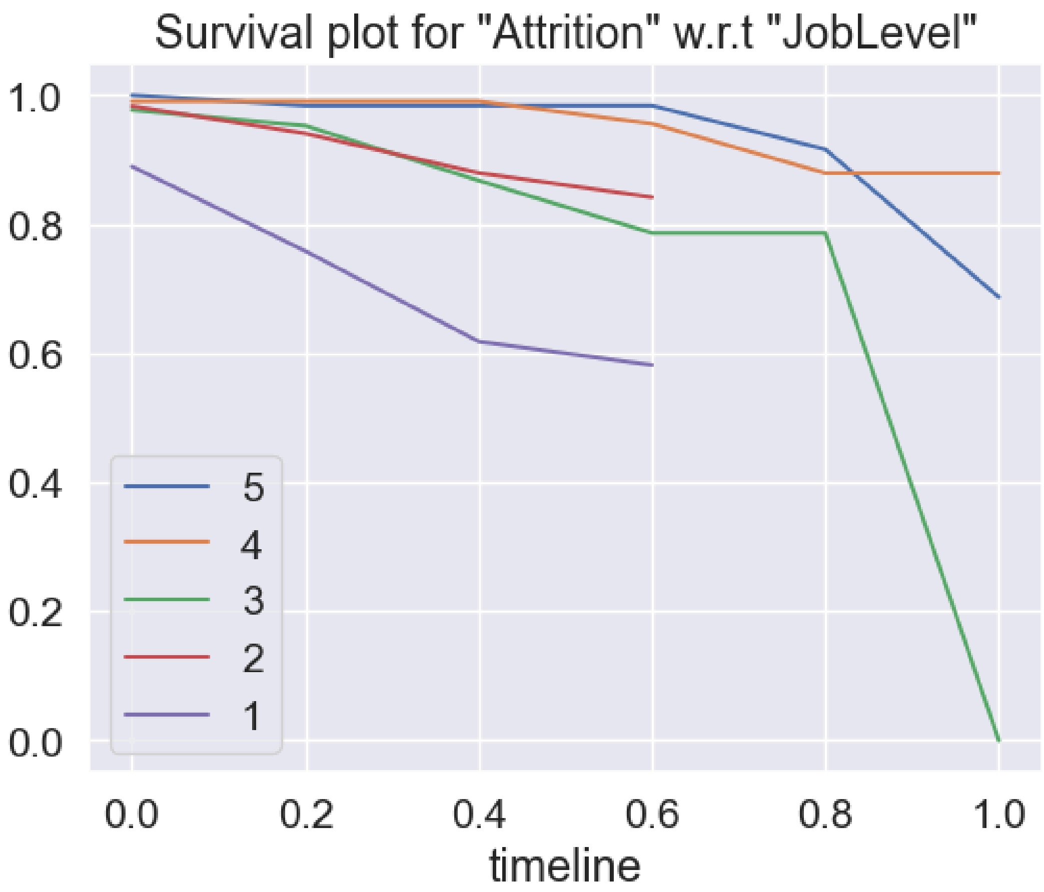
Out[454… Text(0.5, 1.0, 'Survival plot for "Attrition" w.r.t "BusinessTravel"')



In [455…

```
##############################################
''' We now select the group to be JobLevel'''
df_company.JobLevel.value_counts()


# Applying KaplanMeierFitter model on Time and Events for the group "1"
kmf.fit(T[df_company.JobLevel == 1], df_company.Attrition[df_company.JobLevel == 1], label = '5')
ax = kmf.survival_function_.plot()

# Applying KaplanMeierFitter model on Time and Events for the group "0.75"
kmf.fit(T[df_company.JobLevel == 0.75], df_company.Attrition[df_company.JobLevel == 0.75], label =
kmf.survival_function_.plot(ax=ax)

# Applying KaplanMeierFitter model on Time and Events for the group "0.50"
kmf.fit(T[df_company.JobLevel == 0.50], df_company.Attrition[df_company.JobLevel == 0.50], label =
kmf.survival_function_.plot(ax=ax)

# Applying KaplanMeierFitter model on Time and Events for the group "0.25"
kmf.fit(T[df_company.JobLevel == 0.25], df_company.Attrition[df_company.JobLevel == 0.25], label =
kmf.survival_function_.plot(ax=ax)

# Applying KaplanMeierFitter model on Time and Events for the group "0"
kmf.fit(T[df_company.JobLevel == 0], df_company.Attrition[df_company.JobLevel == 0], label = '1')
kmf.survival_function_.plot(ax=ax)
plt.title('Survival plot for "Attrition" w.r.t "JobLevel"')
```

Out[455… Text(0.5, 1.0, 'Survival plot for "Attrition" w.r.t "JobLevel"')



In [456…
```
df= pd.read_excel(r"D:\project\tableau\Final dataset Attrition_final.xlsx")
```

Splitting the Data and Building the Model

In [466…
```
#######################################################
'''
We start building the models for classification
We start by splitting the data into Train and test
'''
#######################################################

from sklearn.model_selection import train_test_split
df = df_company.iloc[:, 1]
df1 = df_company.drop('Attrition', axis = 1)
X = df1
Y = df
```

In [468…
```
# herein we split the data with test size kept as 15%
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.15, random_state = 40)
print(y_train.value_counts())
print(y_test.value_counts())
```

```
Attrition
0.0    1041
1.0     208
Name: count, dtype: int64
Attrition
0.0    192
1.0     29
Name: count, dtype: int64
```

Let us import the various libraries to build the models

In [471…

```
# We start building the models using the following regression models for classifying
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
```

Logistic Regression

In [474…

```
'''Logistic Regression'''
log = LogisticRegression()
```

In [476…

```
x_test = x_test.drop(x_test.iloc[:, 29:30], axis = 1)
x_test
```

Out[476…

	Age	BusinessTravel	Department	DistanceFromHome	Gender	JobInvolvement	JobLevel	JobRc
1456	0.761905	0.0	0.5	0.035714	1.0	0.666667	0.75	0.3
236	0.357143	1.0	0.5	0.142857	0.0	0.333333	0.00	0.7
70	0.023810	1.0	0.5	0.035714	1.0	0.333333	0.00	0.2
42	0.904762	1.0	1.0	0.357143	0.0	0.333333	0.25	0.8
454	0.119048	1.0	1.0	0.214286	1.0	0.666667	0.00	1.0
...	...	...	...	...	...	...	...	
1269	0.285714	1.0	0.5	0.000000	1.0	0.666667	0.00	0.2
348	0.976190	0.0	0.0	0.035714	0.0	0.333333	1.00	0.3
726	0.690476	1.0	0.0	0.892857	0.0	0.666667	1.00	0.3
1209	0.428571	1.0	1.0	0.321429	0.0	0.333333	0.25	0.8
192	0.261905	1.0	0.5	0.107143	0.0	0.333333	0.00	0.7

221 rows × 29 columns

In [478…

```
x_train = x_train.drop(x_train.iloc[:, 29:30], axis = 1)
x_train
```

Out[478…

	Age	BusinessTravel	Department	DistanceFromHome	Gender	JobInvolvement	JobLevel	JobRc
776	0.547619	1.0	1.0	0.000000	0.0	0.666667	0.25	0.8
281	0.357143	0.0	1.0	0.250000	0.0	0.333333	0.00	1.0
435	0.309524	1.0	0.0	0.250000	0.0	1.000000	0.00	0.1
1267	0.309524	1.0	1.0	1.000000	0.0	0.333333	0.25	0.8
323	0.666667	0.0	1.0	0.892857	1.0	0.333333	0.25	0.8
...	...	...	...	...	...	...	...	
1016	0.666667	1.0	1.0	0.321429	0.0	0.666667	0.50	0.8
165	0.214286	1.0	1.0	0.357143	0.0	1.000000	0.00	1.0
7	0.238095	1.0	0.5	0.035714	1.0	0.666667	0.00	0.2
219	0.404762	1.0	1.0	0.035714	0.0	0.666667	0.00	1.0
1350	0.833333	1.0	1.0	0.035714	1.0	0.666667	0.75	0.3

1249 rows × 29 columns

In [480…

```
log.fit(x_train, y_train)
log_acc = accuracy_score(y_test, log.predict(x_test))
print("Train Set Accuracy:"+str(accuracy_score(y_train, log.predict(x_train))*100))
print("Test Set Accuracy:"+str(accuracy_score(y_test, log.predict(x_test))*100))

plt.figure(figsize = (6,4))
df_ = pd.DataFrame(confusion_matrix(y_test, log.predict(x_test)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_, annot=True,annot_kws={"size": 16}, fmt='g')
plt.xlabel('Predicted Class')
plt.ylabel('Original Class')
plt.show()
```

```
Train Set Accuracy:87.02962369895917
Test Set Accuracy:88.23529411764706
```

Decision Tree

In [483…

```
'''Descision Tree'''
dec = DecisionTreeClassifier()
dec.fit(x_train, y_train)

dec_acc = accuracy_score(y_test, dec.predict(x_test))
print("Train test Accuracy:"+str(accuracy_score(y_train, dec.predict(x_train))*100))
print("Test Set Accuracy:"+str(accuracy_score(y_test, dec.predict(x_test))*100))

plt.figure(figsize = (6,4))
df_ = pd.DataFrame(confusion_matrix(y_test, dec.predict(x_test)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_, annot=True,annot_kws={"size": 16}, fmt='g')
plt.xlabel('Predicted Class')
plt.ylabel('Original Class')
plt.show()
```

```
Train test Accuracy:100.0
Test Set Accuracy:80.09049773755656
```

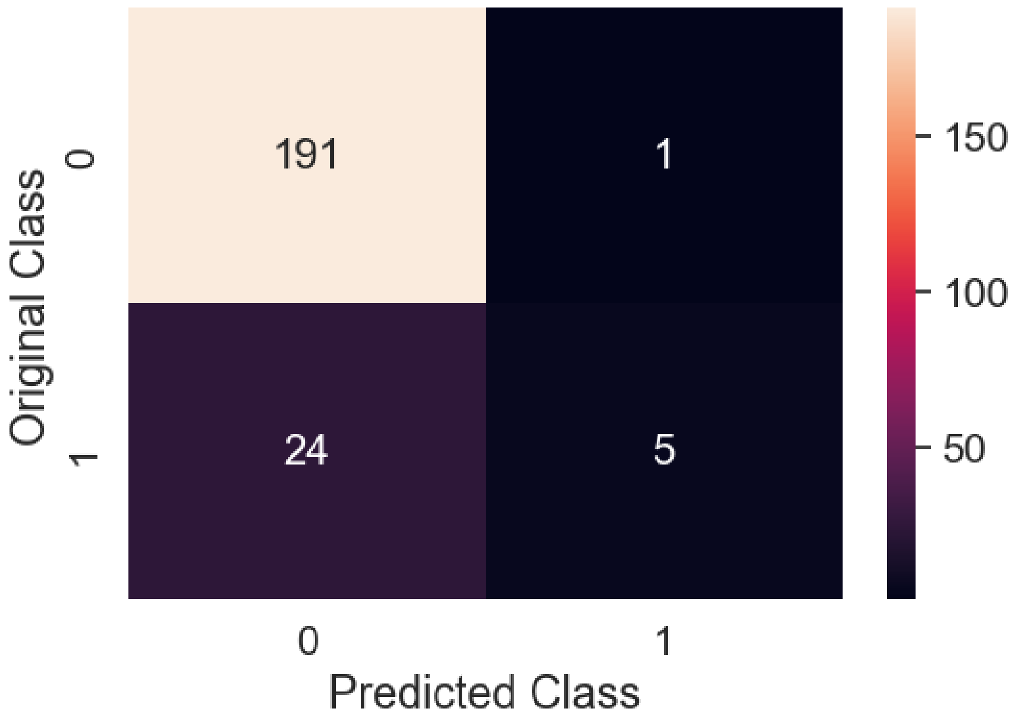
Random Forest

In [486…

```
"""**Random Forest**"""

r_for = RandomForestClassifier()
r_for.fit(x_train,y_train)

r_acc=accuracy_score(y_test,r_for.predict(x_test))

print("Train Set Accuracy:"+str(accuracy_score(y_train,r_for.predict(x_train))*100))
print("Test Set Accuracy:"+str(accuracy_score(y_test,r_for.predict(x_test))*100))

plt.figure(figsize=(6,4))
df_ = pd.DataFrame(confusion_matrix(y_test, r_for.predict(x_test)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_, annot=True,annot_kws={"size": 16}, fmt='g')
plt.xlabel('Predicted Class')
```

```
plt.ylabel('Original Class')
plt.show()
```

```
Train Set Accuracy:100.0
Test Set Accuracy:88.68778280542986
```

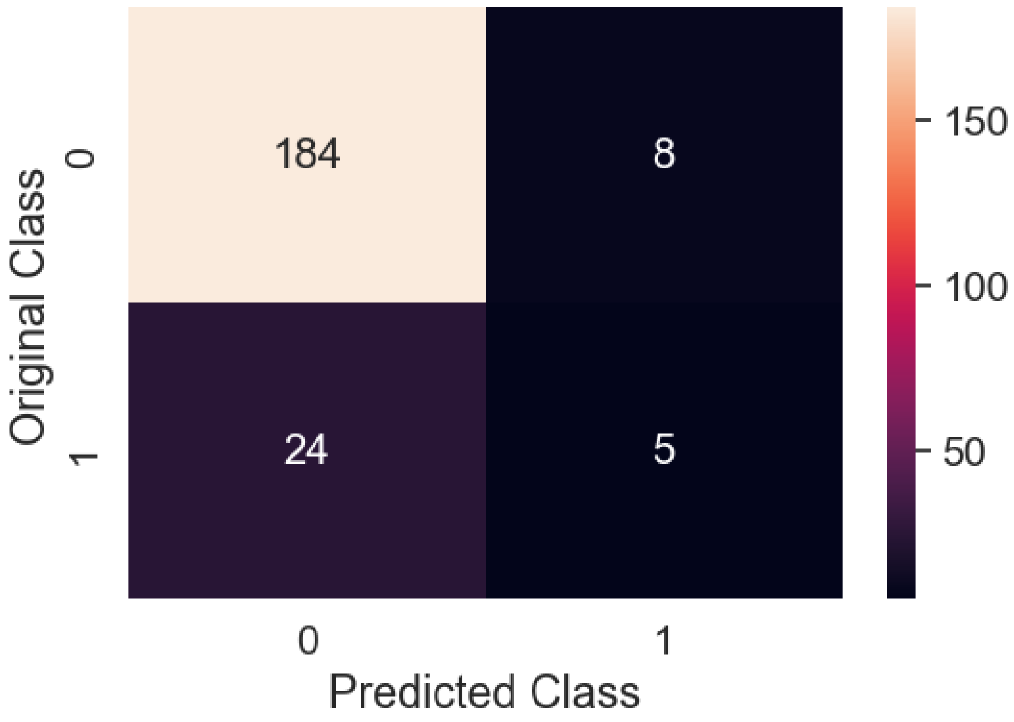


KNN - K Nearest Neighbours

In [489…

```
"""**K-NN**
"""

k_nei = KNeighborsClassifier()
k_nei.fit(x_train,y_train)

k_acc = accuracy_score(y_test,k_nei.predict(x_test))

print("Train set Accuracy:"+str(accuracy_score(y_train,k_nei.predict(x_train))*100))
print("Test Set Accuracy:"+str(accuracy_score(y_test,k_nei.predict(x_test))*100))

plt.figure(figsize=(6,4))
df_ = pd.DataFrame(confusion_matrix(y_test, k_nei.predict(x_test)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_, annot=True,annot_kws={"size": 16}, fmt='g')
plt.xlabel('Predicted Class')
plt.ylabel('Original Class')
plt.show()
```

```
Train set Accuracy:85.98879103282626
Test Set Accuracy:85.52036199095022
```

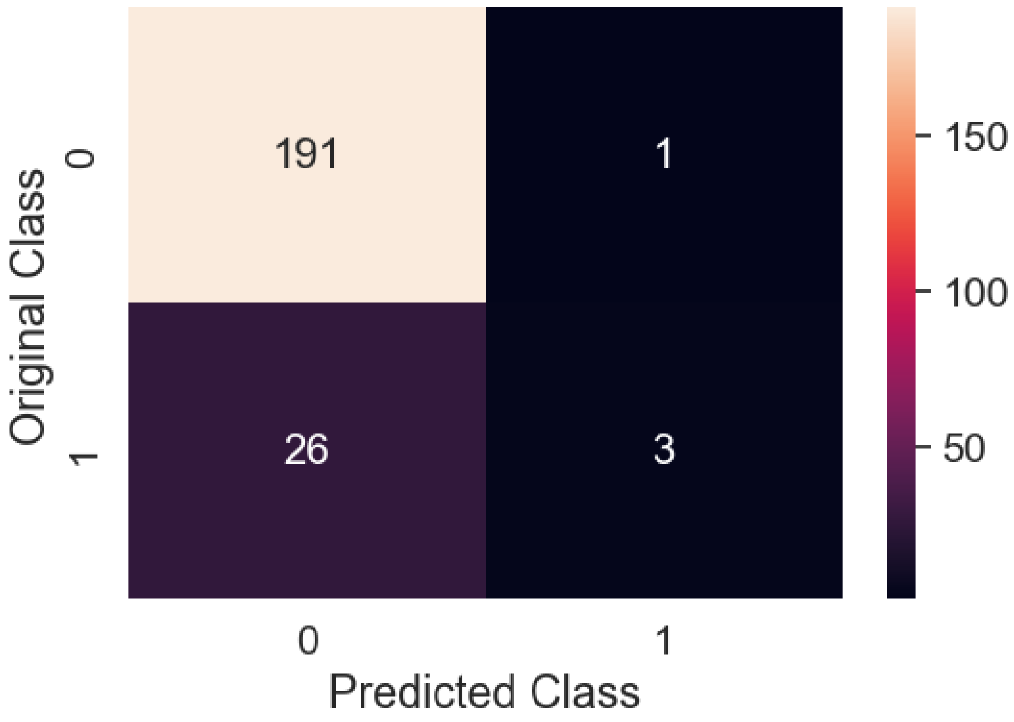


SVC

In [493…

```
"""**SVC**"""

s_vec = SVC()
s_vec.fit(x_train,y_train)

s_acc = accuracy_score(y_test,s_vec.predict(x_test))

print("Train set Accuracy:"+str(accuracy_score(y_train,s_vec.predict(x_train))*100))
print("Test Set Accuracy:"+str(accuracy_score(y_test,s_vec.predict(x_test))*100))

plt.figure(figsize=(6,4))
df_ = pd.DataFrame(confusion_matrix(y_test, s_vec.predict(x_test)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_, annot=True,annot_kws={"size": 16}, fmt='g')
plt.xlabel('Predicted Class')
plt.ylabel('Original Class')
plt.show()
```

```
Train set Accuracy:87.51000800640513
Test Set Accuracy:87.78280542986425
```



Gaussian Naive Bayes

In [496…

```
g_clf = GaussianNB()
g_clf.fit(x_train,y_train)

g_acc = accuracy_score(y_test,g_clf.predict(x_test))

print("Train set Accuracy:"+str(accuracy_score(y_train,g_clf.predict(x_train))*100))
print("Test Set Accuracy:"+str(accuracy_score(y_test,g_clf.predict(x_test))*100))

plt.figure(figsize=(6,4))
df_ = pd.DataFrame(confusion_matrix(y_test, g_clf.predict(x_test)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_, annot=True,annot_kws={"size": 16}, fmt='g')
plt.xlabel('Predicted Class')
plt.ylabel('Original Class')
plt.show()
```

```
Train set Accuracy:81.26501200960769
Test Set Accuracy:79.18552036199095
```

Comparing the models and checking the best accuracy result off the lot

In [499…

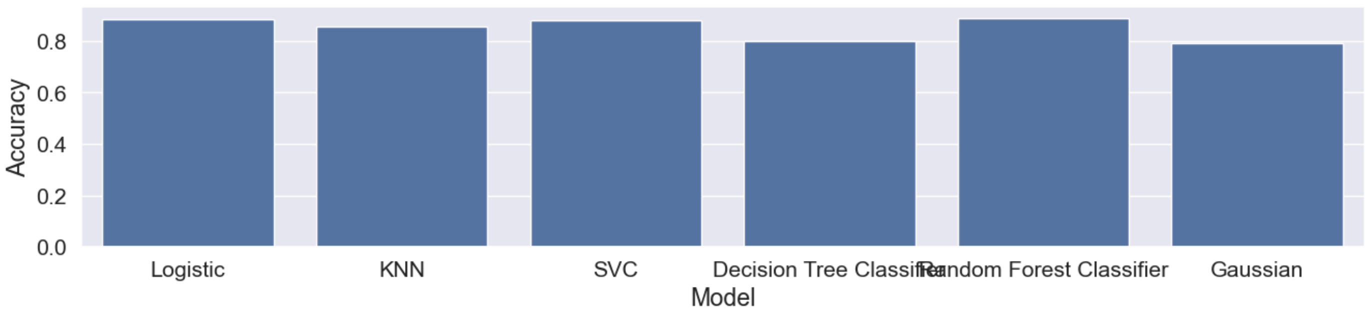
```
models = pd.DataFrame({'Model': ['Logistic', 'KNN', 'SVC', 'Decision Tree Classifier',
                       'Random Forest Classifier', 'Gaussian'],
                       'Accuracy': [ log_acc,k_acc, s_acc, dec_acc, r_acc, g_acc]})

models.sort_values(by = 'Accuracy', ascending = False)
```

Out[499…

	Model	Accuracy
4	Random Forest Classifier	0.886878
0	Logistic	0.882353
2	SVC	0.877828
1	KNN	0.855204
3	Decision Tree Classifier	0.800905
5	Gaussian	0.791855

In [504…

```
plt.figure(figsize = (16,3))
sns.barplot(x = 'Model', y = 'Accuracy', data = models)
plt.show()
```



We note that Random Forest Classifier gives the best result, hence we will go with the random forest as it less suceptable to overfitting than logistic regression

In [ ]: