Naveen_joon_Lab8_IBM

April 20, 2020

1 IBM HR Analytics Employee Attrition & Performance.

1.1 1) Basic analysis and Visualization

In [3]:
```
# data visualisation and manipulation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
import missingno as msno

#configure
# sets matplotlib to inline and displays graphs below the corressponding cell.
% matplotlib inline
style.use('fivethirtyeight')
sns.set(style='whitegrid',color_codes=True)

# Ignore  the warnings
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')

#import the necessary modelling algos.
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB

#model selection
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score,precision_score,recall_score,confusion_matr
```

```
from sklearn.model_selection import GridSearchCV

from imblearn.over_sampling import SMOTE

#preprocess.
from sklearn.preprocessing import MinMaxScaler,StandardScaler,Imputer,LabelEncoder,OneH
```

1.2 Reading the data from a CSV file

In [4]:
```
df=pd.read_csv(r'WA_Fn-UseC_-HR-Employee-Attrition.csv')
```

In [5]:
```
df.head()
```

Out[5]:
```
   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  \
0           ...                                  1            80
1           ...                                  4            80
2           ...                                  2            80
3           ...                                  3            80
4           ...                                  4            80

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

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

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

[5 rows x 35 columns]
```

In [6]: `df.shape`

Out[6]: `(1470, 35)`

In [7]: `df.columns`

```
Out[7]: Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
               'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
               'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
               'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
               'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
               'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
               'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
               'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
               'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
               'YearsWithCurrManager'],
              dtype='object')
```

1.3 Missing Values Treatment

In [8]: `df.info()   # no null or Nan values.`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
Age                         1470 non-null int64
Attrition                   1470 non-null object
BusinessTravel              1470 non-null object
DailyRate                   1470 non-null int64
Department                  1470 non-null object
DistanceFromHome            1470 non-null int64
Education                   1470 non-null int64
EducationField              1470 non-null object
EmployeeCount               1470 non-null int64
EmployeeNumber              1470 non-null int64
EnvironmentSatisfaction     1470 non-null int64
Gender                      1470 non-null object
HourlyRate                  1470 non-null int64
JobInvolvement              1470 non-null int64
JobLevel                    1470 non-null int64
```

```
JobRole                     1470 non-null object
JobSatisfaction             1470 non-null int64
MaritalStatus               1470 non-null object
MonthlyIncome               1470 non-null int64
MonthlyRate                 1470 non-null int64
NumCompaniesWorked          1470 non-null int64
Over18                      1470 non-null object
OverTime                    1470 non-null object
PercentSalaryHike           1470 non-null int64
PerformanceRating           1470 non-null int64
RelationshipSatisfaction    1470 non-null int64
StandardHours               1470 non-null int64
StockOptionLevel            1470 non-null int64
TotalWorkingYears           1470 non-null int64
TrainingTimesLastYear       1470 non-null int64
WorkLifeBalance             1470 non-null int64
YearsAtCompany              1470 non-null int64
YearsInCurrentRole          1470 non-null int64
YearsSinceLastPromotion     1470 non-null int64
YearsWithCurrManager        1470 non-null int64
dtypes: int64(26), object(9)
memory usage: 402.0+ KB
```

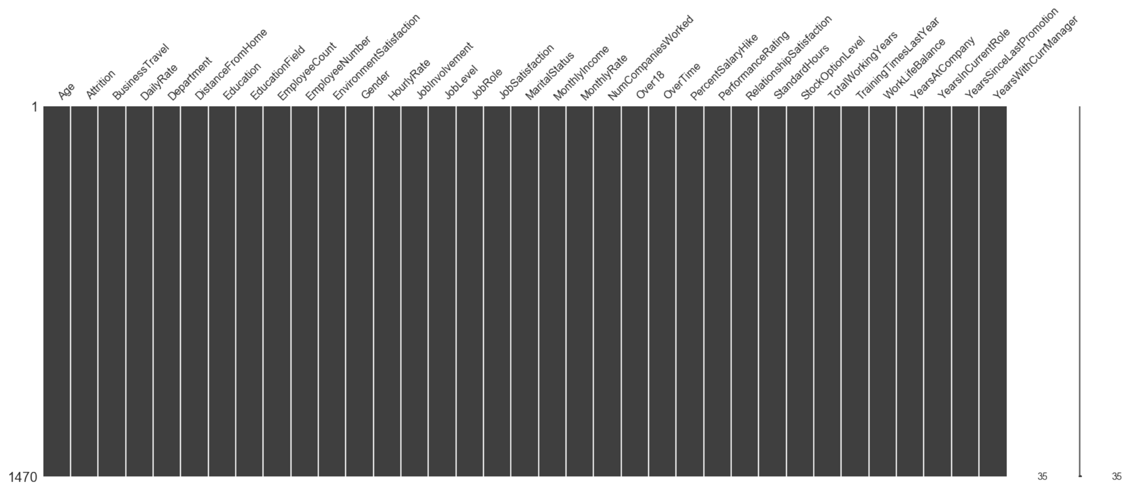
In [9]:
```
df.isnull().sum()
```

Out[9]:
```
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
```

In [10]: `msno.matrix(df)`

Out[10]: `<matplotlib.axes._subplots.AxesSubplot at 0x195011d5208>`



In [11]: `df.columns`

```
Out[11]: Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

```
In [12]: df.head()
```

```
Out[12]:    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  \
         0           ...                                  1            80
         1           ...                                  4            80
         2           ...                                  2            80
         3           ...                                  3            80
         4           ...                                  4            80

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

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

            YearsWithCurrManager
         0                     5
         1                     7
         2                     0
         3                     0
         4                     2

         [5 rows x 35 columns]
```

'Attrition' of the employee which can be either a Yes or a No. Hence this is a Binary Classification problem.

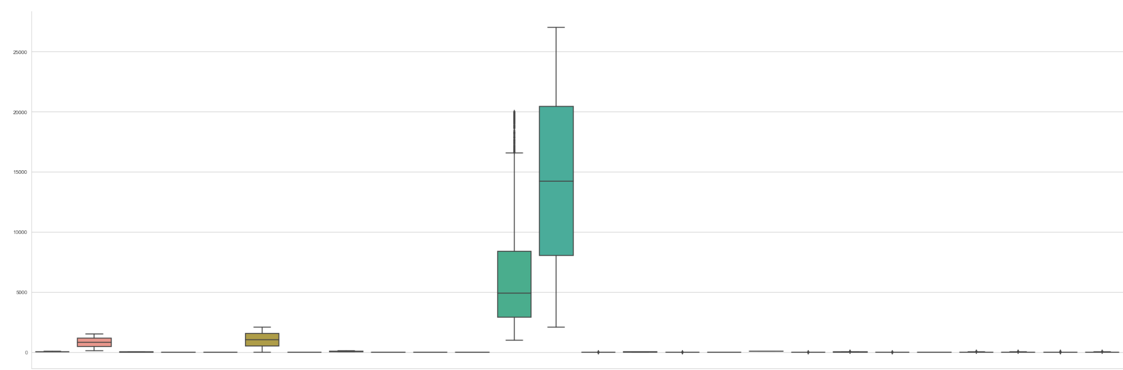
In [13]: df.describe()

```
Out[13]:               Age    DailyRate  DistanceFromHome    Education  EmployeeCount  \
         count  1470.000000  1470.000000       1470.000000  1470.000000         1470.0
         mean     36.923810   802.485714          9.192517     2.912925            1.0
         std       9.135373   403.509100          8.106864     1.024165            0.0
         min      18.000000   102.000000          1.000000     1.000000            1.0
         25%      30.000000   465.000000          2.000000     2.000000            1.0
         50%      36.000000   802.000000          7.000000     3.000000            1.0
         75%      43.000000  1157.000000         14.000000     4.000000            1.0
         max      60.000000  1499.000000         29.000000     5.000000            1.0

                EmployeeNumber  EnvironmentSatisfaction   HourlyRate  JobInvolvement  \
         count     1470.000000              1470.000000  1470.000000     1470.000000
         mean      1024.865306                 2.721769    65.891156        2.729932
         std        602.024335                 1.093082    20.329428        0.711561
         min          1.000000                 1.000000    30.000000        1.000000
         25%        491.250000                 2.000000    48.000000        2.000000
         50%       1020.500000                 3.000000    66.000000        3.000000
         75%       1555.750000                 4.000000    83.750000        3.000000
         max       2068.000000                 4.000000   100.000000        4.000000

                   JobLevel          ...           RelationshipSatisfaction  \
         count  1470.000000          ...                        1470.000000
         mean      2.063946          ...                           2.712245
         std       1.106940          ...                           1.081209
         min       1.000000          ...                           1.000000
         25%       1.000000          ...                           2.000000
         50%       2.000000          ...                           3.000000
         75%       3.000000          ...                           4.000000
         max       5.000000          ...                           4.000000

                StandardHours  StockOptionLevel  TotalWorkingYears  \
         count         1470.0       1470.000000        1470.000000
         mean            80.0          0.793878          11.279592
         std              0.0          0.852077           7.780782
         min             80.0          0.000000           0.000000
         25%             80.0          0.000000           6.000000
         50%             80.0          1.000000          10.000000
         75%             80.0          1.000000          15.000000
         max             80.0          3.000000          40.000000

                TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany  \
         count            1470.000000      1470.000000     1470.000000
         mean                2.799320         2.761224        7.008163
         std                 1.289271         0.706476        6.126525
         min                 0.000000         1.000000        0.000000
         25%                 2.000000         2.000000        3.000000
```

```
50%                 3.000000         3.000000        5.000000
75%                 3.000000         3.000000        9.000000
max                 6.000000         4.000000       40.000000

       YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
count         1470.000000              1470.000000           1470.000000
mean             4.229252                 2.187755              4.123129
std              3.623137                 3.222430              3.568136
min              0.000000                 0.000000              0.000000
25%              2.000000                 0.000000              2.000000
50%              3.000000                 1.000000              3.000000
75%              7.000000                 3.000000              7.000000
max             18.000000                15.000000             17.000000

[8 rows x 26 columns]
```

In [14]:
```
sns.factorplot(data=df,kind='box',size=10,aspect=3)
```

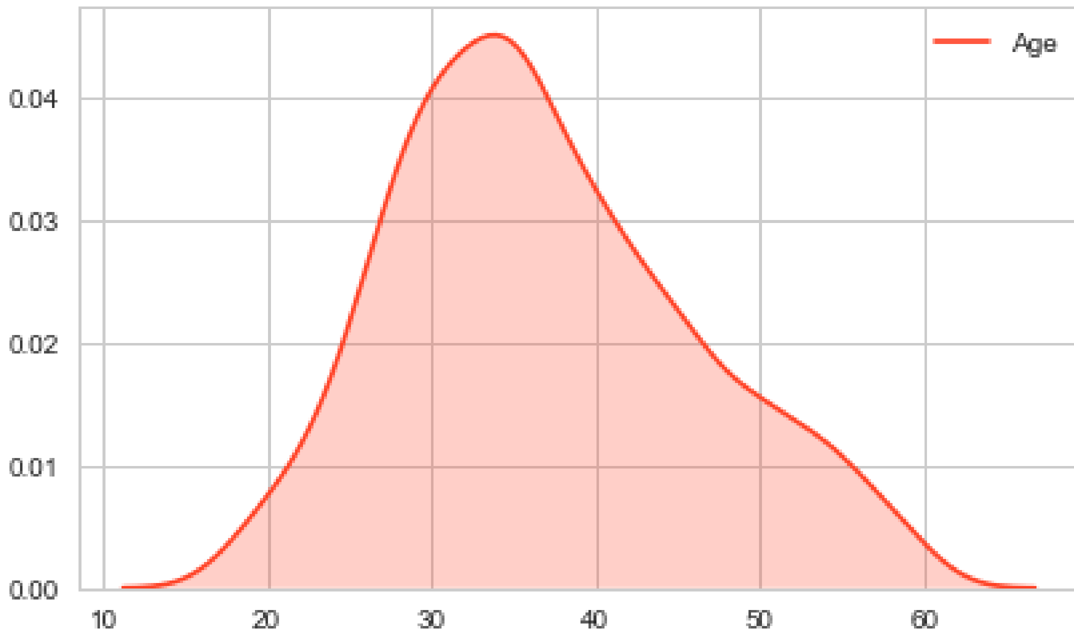
Out[14]: <seaborn.axisgrid.FacetGrid at 0x1950103e080>



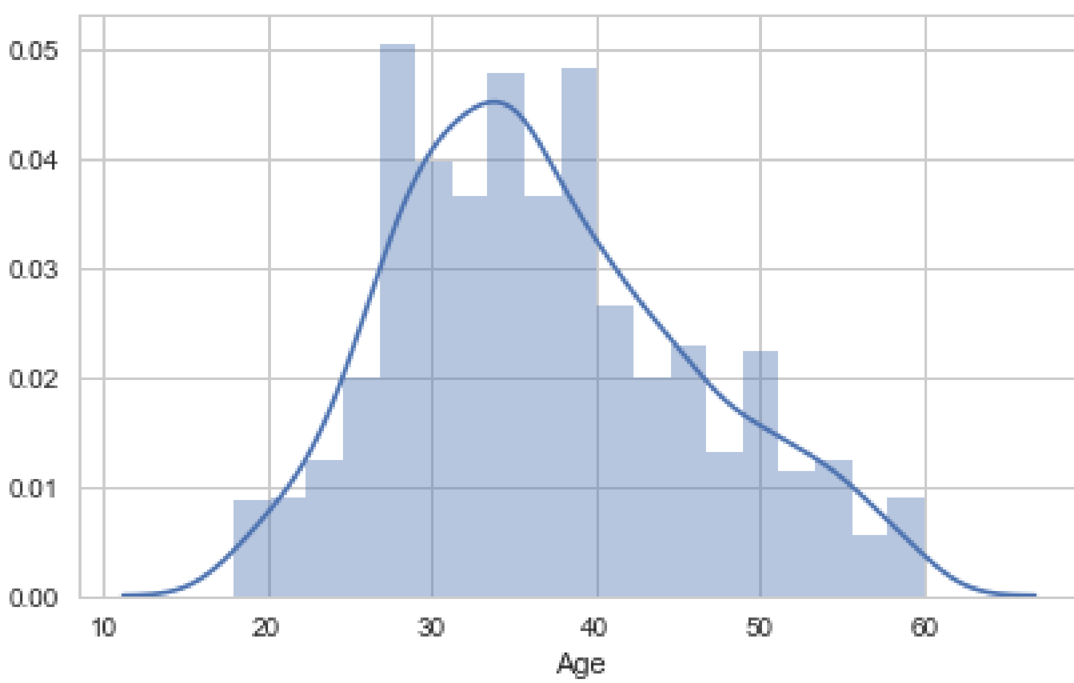
Note that all the features have pretty different scales and so plotting a boxplot is not a good idea.

In [15]:
```
sns.kdeplot(df['Age'],shade=True,color='#ff4125')
```

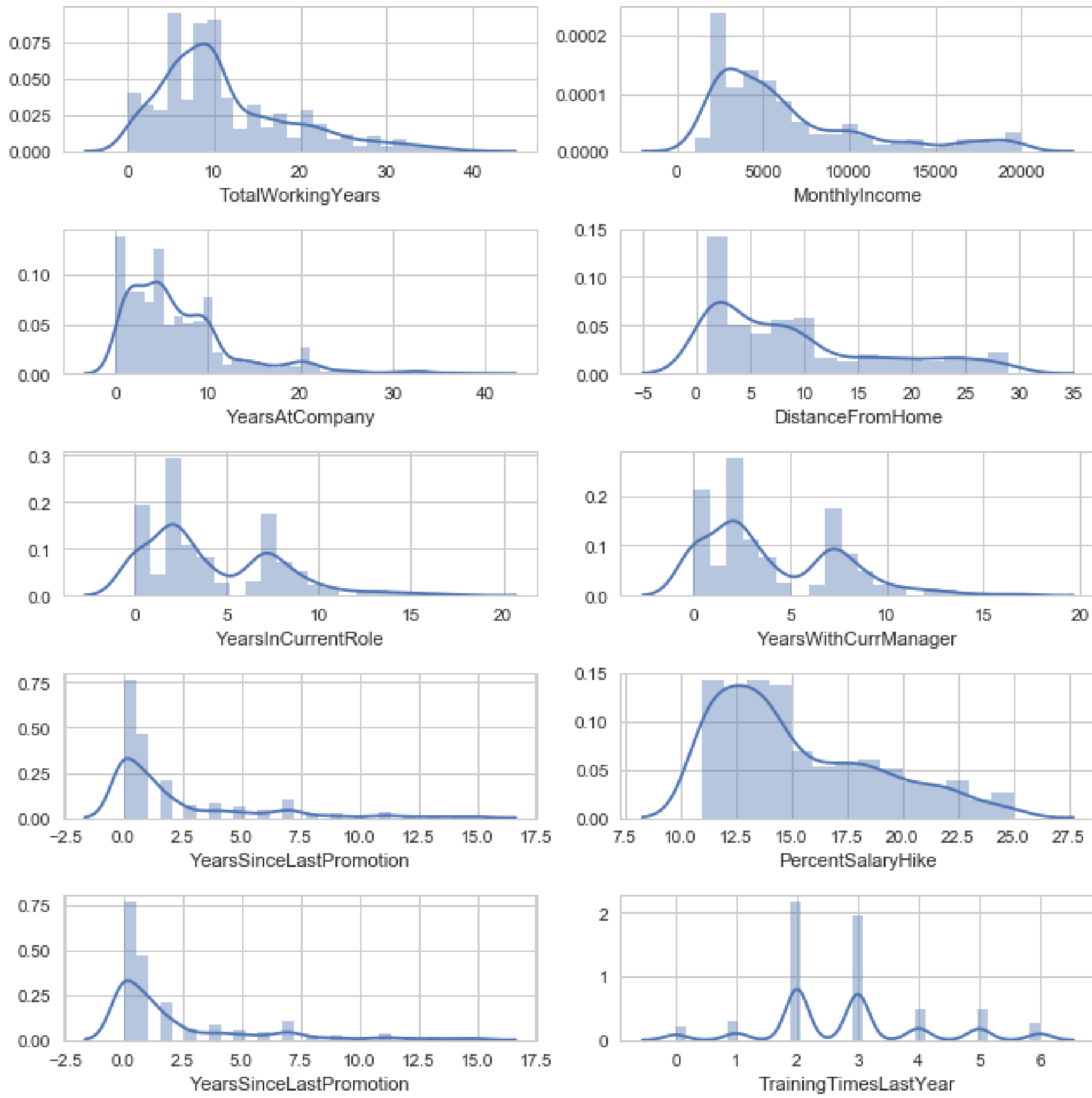
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x19501808b38>




```
In [16]: sns.distplot(df['Age']);

C:\Users\HP\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'normed
  warnings.warn("The 'normed' kwarg is deprecated, and has been "

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1950185b9e8>
```



In [17]:
```
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')

fig,ax = plt.subplots(5,2, figsize=(9,9))
sns.distplot(df['TotalWorkingYears'], ax = ax[0,0])
sns.distplot(df['MonthlyIncome'], ax = ax[0,1])
sns.distplot(df['YearsAtCompany'], ax = ax[1,0])
sns.distplot(df['DistanceFromHome'], ax = ax[1,1])
sns.distplot(df['YearsInCurrentRole'], ax = ax[2,0])
sns.distplot(df['YearsWithCurrManager'], ax = ax[2,1])
sns.distplot(df['YearsSinceLastPromotion'], ax = ax[3,0])
sns.distplot(df['PercentSalaryHike'], ax = ax[3,1])
sns.distplot(df['YearsSinceLastPromotion'], ax = ax[4,0])
sns.distplot(df['TrainingTimesLastYear'], ax = ax[4,1])
plt.tight_layout()
plt.show()
```



Let us now analyze the various categorical features.

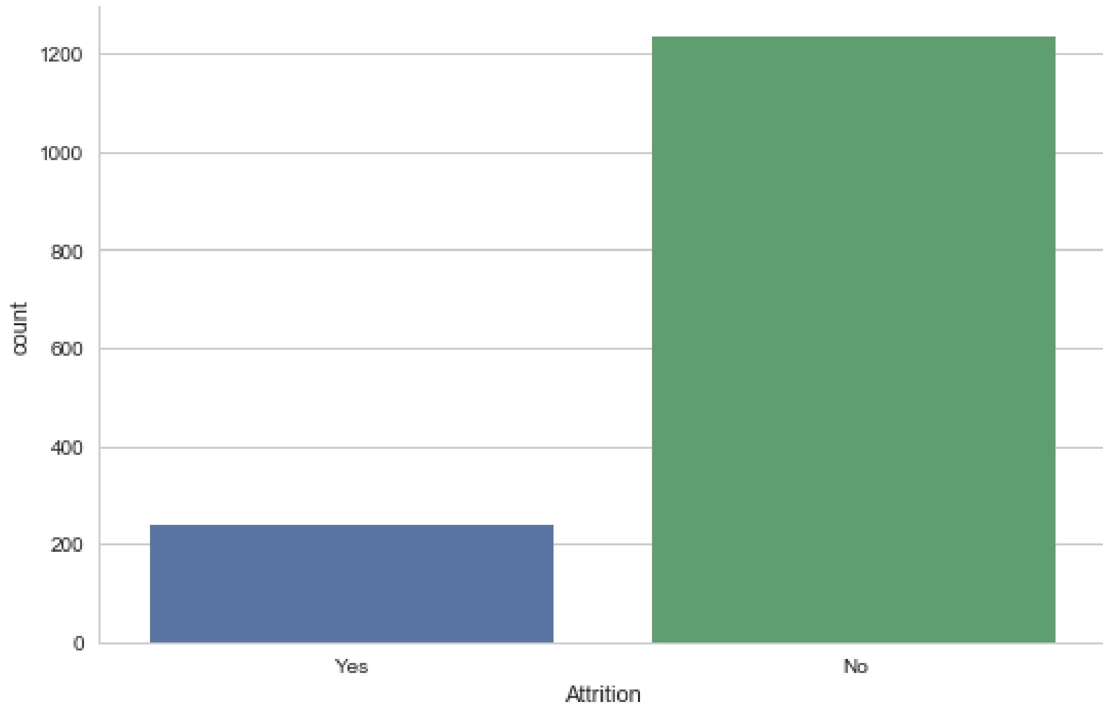
```
In [18]: cat_df=df.select_dtypes(include='object')
```

```
In [19]: cat_df.columns
```

```
Out[19]: Index(['Attrition', 'BusinessTravel', 'Department', 'EducationField', 'Gender',
               'JobRole', 'MaritalStatus', 'Over18', 'OverTime'],
              dtype='object')
```

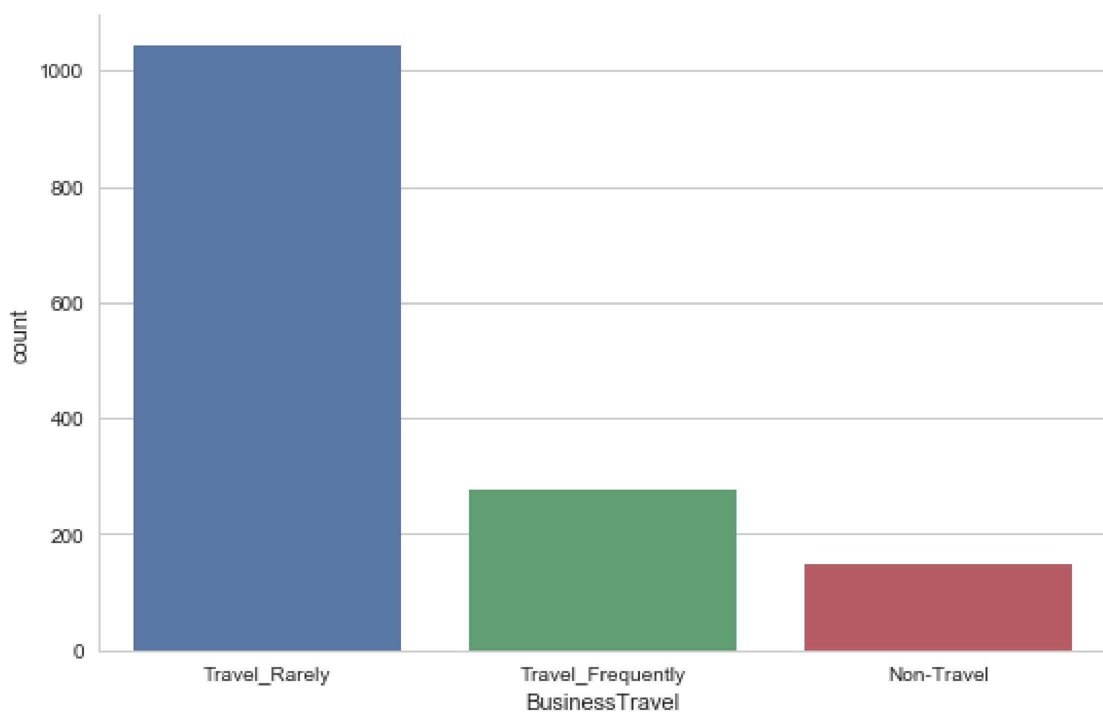
```
In [20]: def plot_cat(attr,labels=None):
             if(attr=='JobRole'):
                 sns.factorplot(data=df,kind='count',size=5,aspect=3,x=attr)
                 return

             sns.factorplot(data=df,kind='count',size=5,aspect=1.5,x=attr)
```

```
In [21]: plot_cat('Attrition')
```

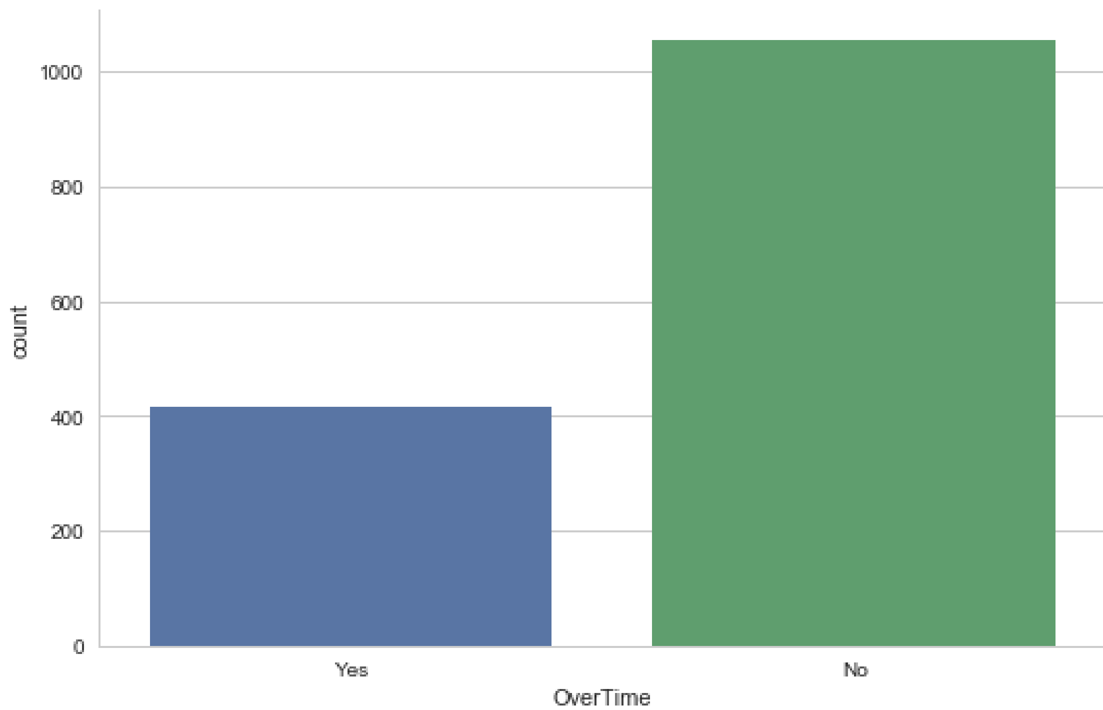


Let us now similalry analyze other categorical features.

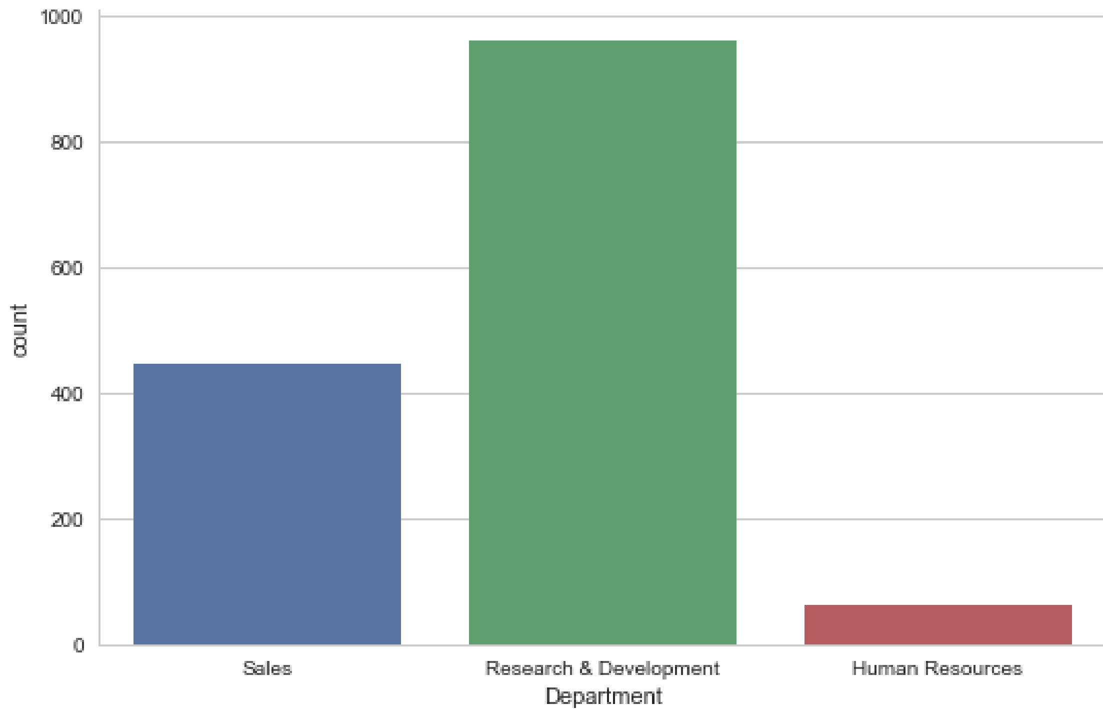
```
In [22]: plot_cat('BusinessTravel')
```



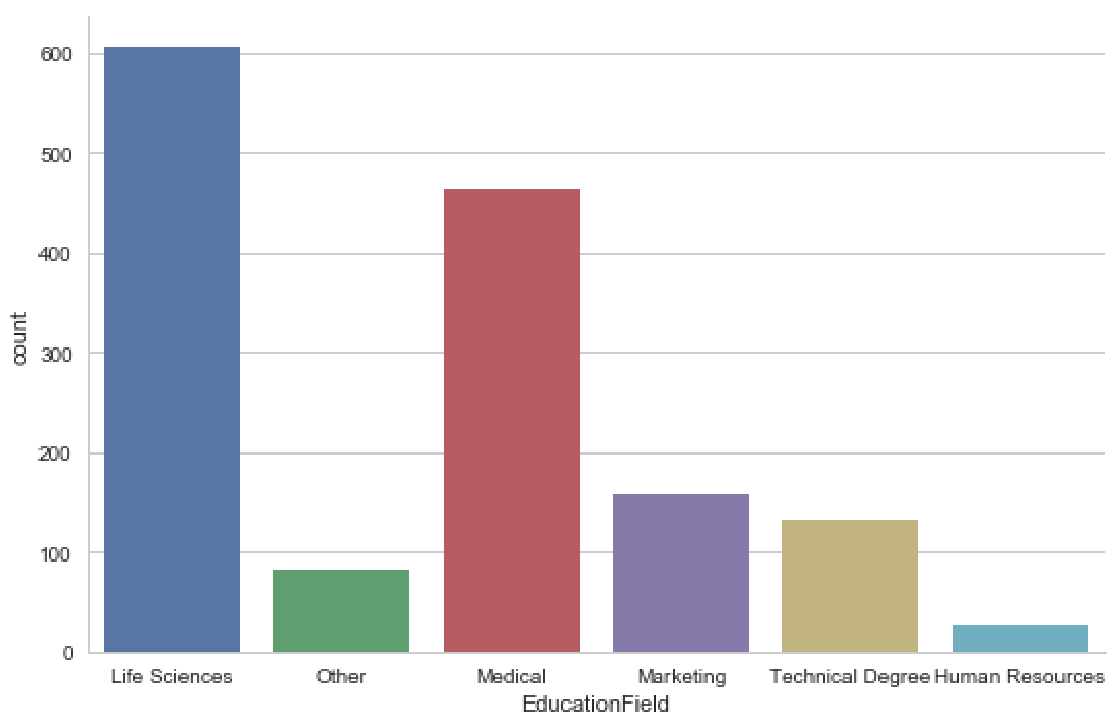
```
In [23]: plot_cat('OverTime')
```



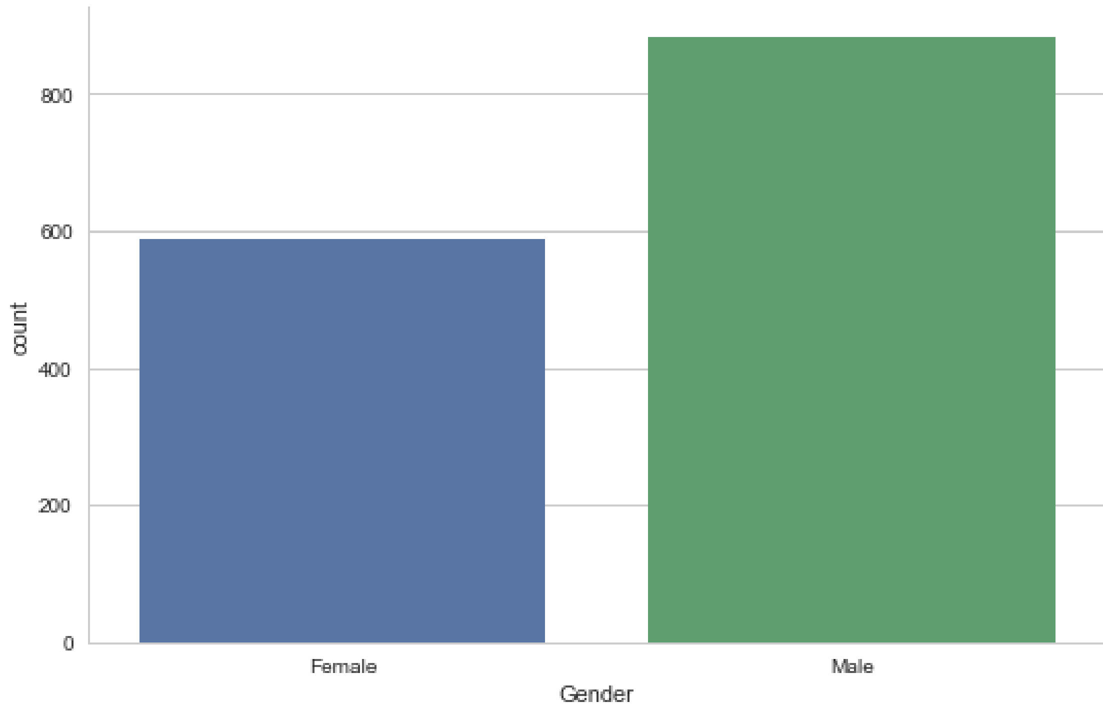

```
In [24]: plot_cat('Department')
```




```
In [25]: plot_cat('EducationField')
```

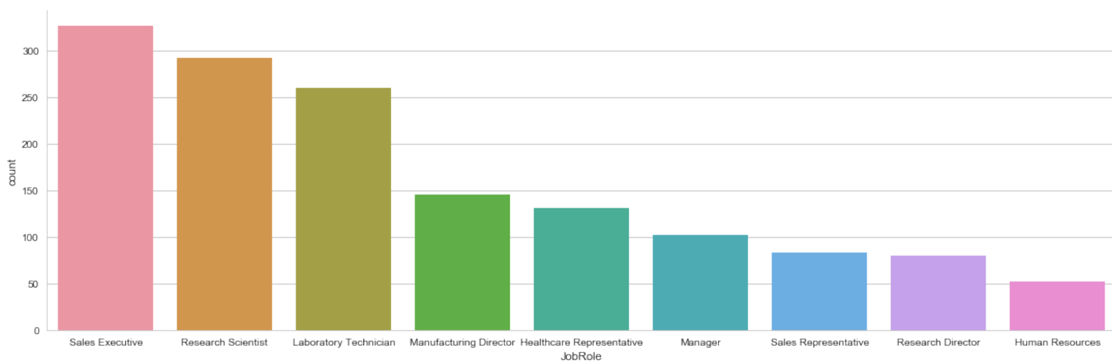


```
In [26]: plot_cat('Gender')
```



Note that males are presnt in higher number.

```
In [27]: plot_cat('JobRole')
```

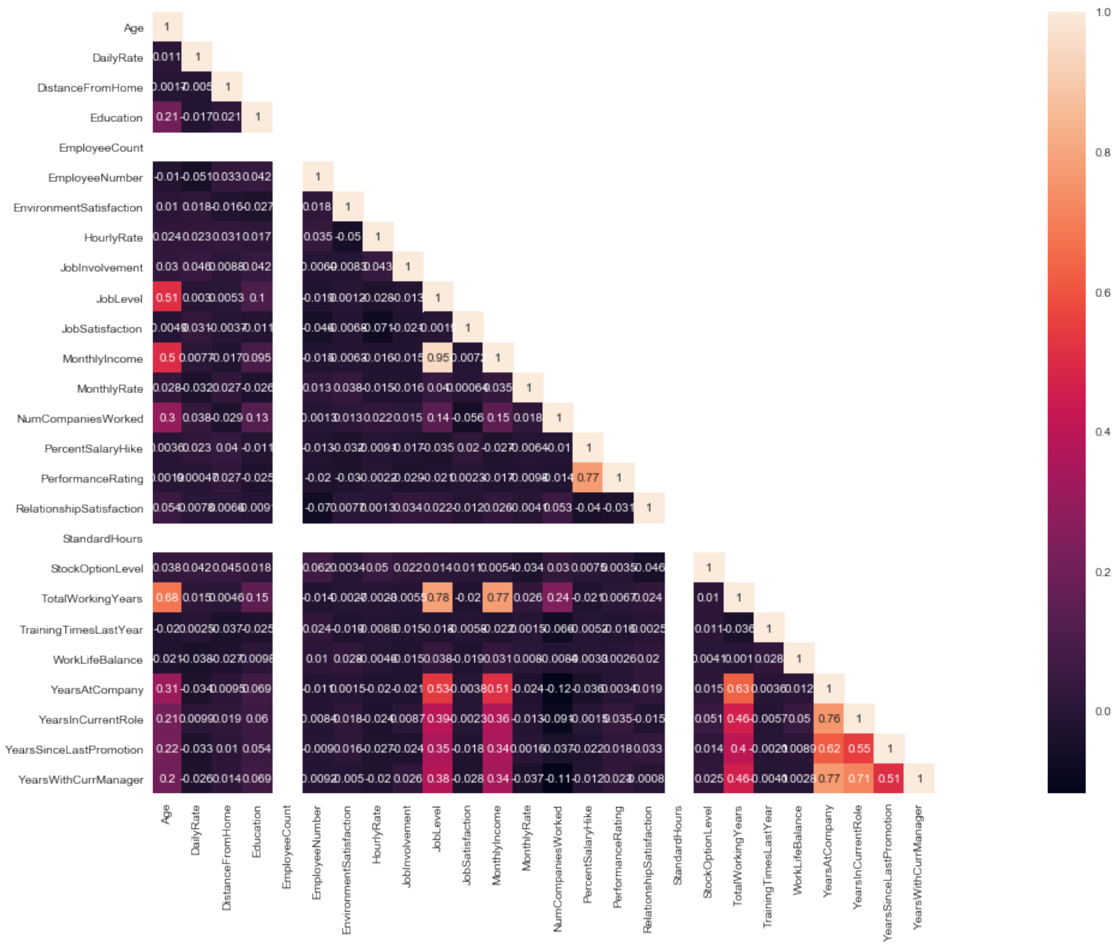


1.4 Corelation b/w Features

```
In [30]: #corelation matrix.
         cor_mat= df.corr()
         mask = np.array(cor_mat)
```

```
mask[np.tril_indices_from(mask)] = False
fig=plt.gcf()
fig.set_size_inches(30,12)
sns.heatmap(data=cor_mat,mask=mask,square=True,annot=True,cbar=True)
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x195044db710>



Note that we can drop some highly corelated features as they add redundancy to the model

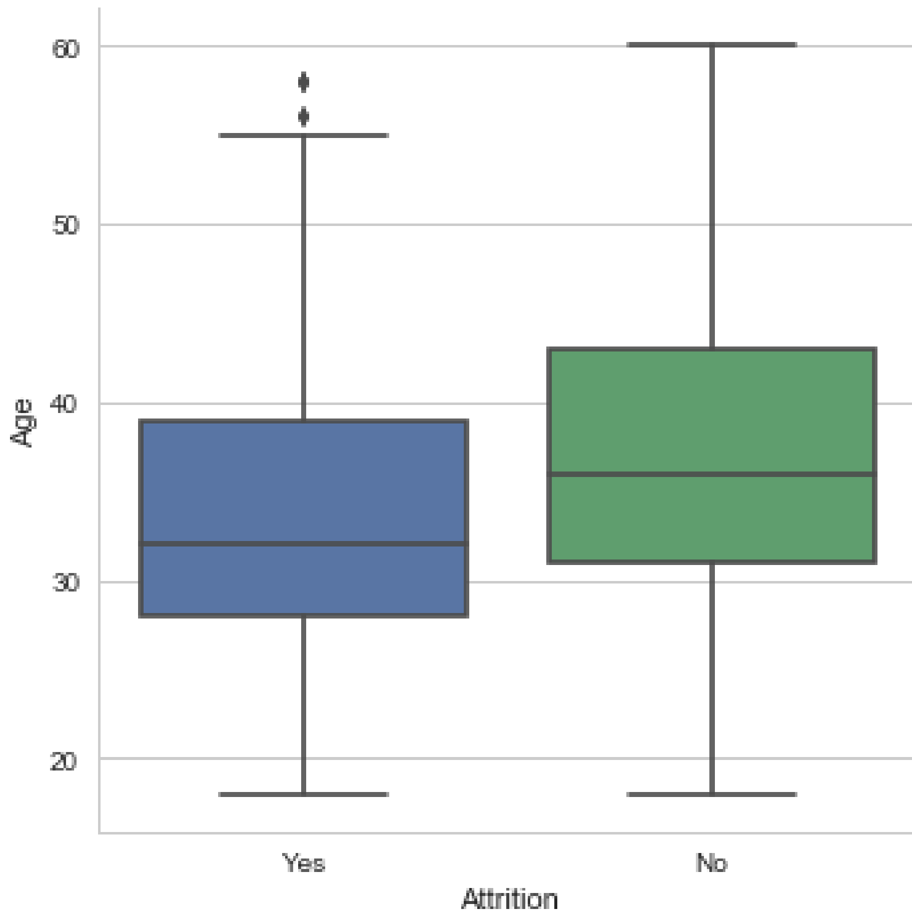
In [31]: df.columns

```
Out[31]: Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
```

```
            'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
            'YearsWithCurrManager'],
           dtype='object')
```

In [32]:
```
sns.factorplot(data=df,y='Age',x='Attrition',size=5,aspect=1,kind='box')
```

Out[32]:
```
<seaborn.axisgrid.FacetGrid at 0x195044ed4e0>
```

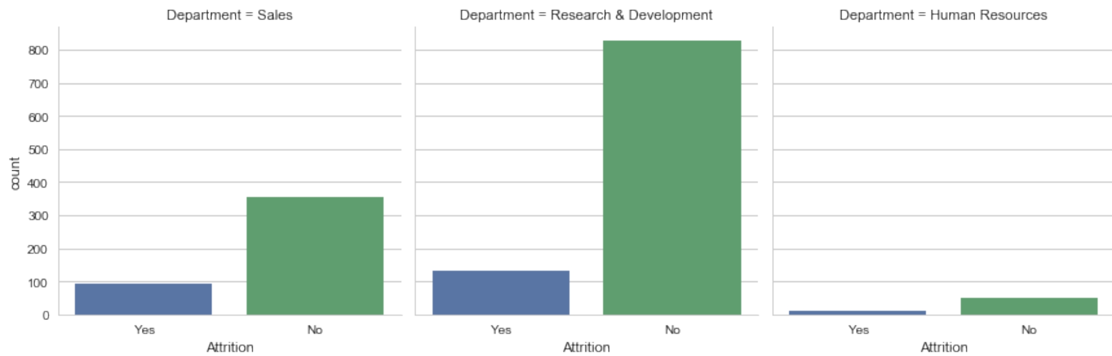


In [33]:
```
df.Department.value_counts()
```

Out[33]:
```
Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64
```

In [34]:
```
sns.factorplot(data=df,kind='count',x='Attrition',col='Department')
```

Out[34]:
```
<seaborn.axisgrid.FacetGrid at 0x19502e4c9b0>
```




```
In [35]: pd.crosstab(columns=[df.Attrition],index=[df.Department],margins=True,normalize='inde
```

```
Out[35]: Attrition                     No       Yes
         Department
         Human Resources         0.809524  0.190476
         Research & Development  0.861602  0.138398
         Sales                   0.793722  0.206278
         All                     0.838776  0.161224
```

About 81 % of the people in HR dont want to leave the organisation and only 19 % want to leave.

```
In [36]: pd.crosstab(columns=[df.Attrition],index=[df.Gender],margins=True,normalize='index')
```

```
Out[36]: Attrition        No       Yes
         Gender
         Female     0.852041  0.147959
         Male       0.829932  0.170068
         All        0.838776  0.161224
```

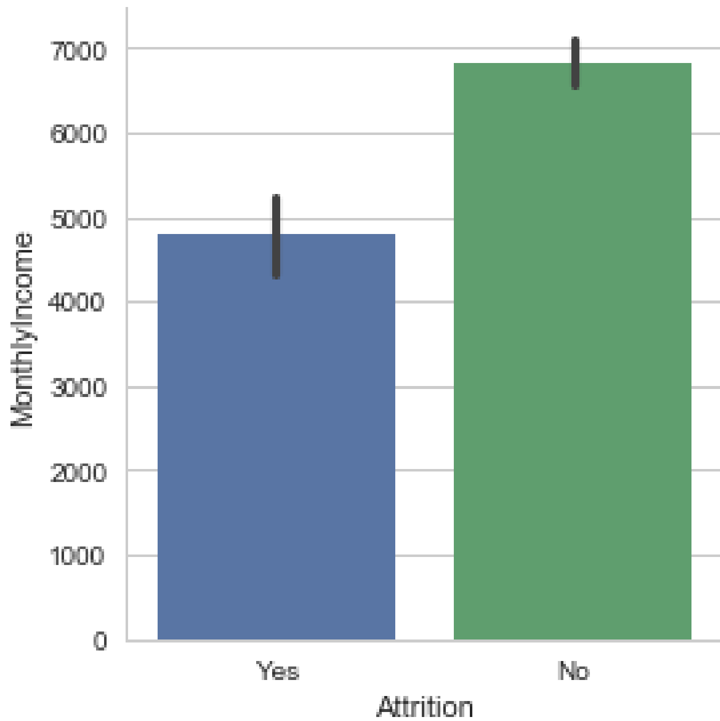
About 85 % of females want to stay in the organisation while only 15 % want to leave the organisation.

```
In [37]: pd.crosstab(columns=[df.Attrition],index=[df.JobLevel],margins=True,normalize='index'
```

```
Out[37]: Attrition        No       Yes
         JobLevel
         1          0.736648  0.263352
         2          0.902622  0.097378
         3          0.853211  0.146789
         4          0.952830  0.047170
         5          0.927536  0.072464
         All        0.838776  0.161224
```

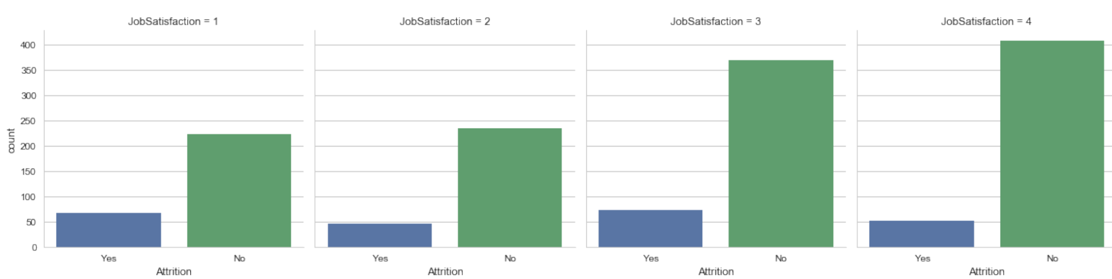
```
In [38]: sns.factorplot(data=df,kind='bar',x='Attrition',y='MonthlyIncome')
```

Out[38]: <seaborn.axisgrid.FacetGrid at 0x1950456de48>



In [39]: `sns.factorplot(data=df,kind='count',x='Attrition',col='JobSatisfaction')`

Out[39]: <seaborn.axisgrid.FacetGrid at 0x195045d36d8>



In [40]: `pd.crosstab(columns=[df.Attrition],index=[df.JobSatisfaction],margins=True,normalize=`

Out[40]:

Attrition	No	Yes
JobSatisfaction		
1	0.771626	0.228374
2	0.835714	0.164286
3	0.834842	0.165158
4	0.886710	0.113290
All	0.838776	0.161224

In [41]: pd.crosstab(columns=[df.Attrition],index=[df.EnvironmentSatisfaction],margins=True,no

```
Out[41]: Attrition                      No       Yes
         EnvironmentSatisfaction
         1                        0.746479  0.253521
         2                        0.850174  0.149826
         3                        0.863135  0.136865
         4                        0.865471  0.134529
         All                      0.838776  0.161224
```

In [42]: pd.crosstab(columns=[df.Attrition],index=[df.JobInvolvement],margins=True,normalize='

```
Out[42]: Attrition             No       Yes
         JobInvolvement
         1               0.662651  0.337349
         2               0.810667  0.189333
         3               0.855991  0.144009
         4               0.909722  0.090278
         All             0.838776  0.161224
```

In [43]: pd.crosstab(columns=[df.Attrition],index=[df.WorkLifeBalance],margins=True,normalize=

```
Out[43]: Attrition              No       Yes
         WorkLifeBalance
         1                0.687500  0.312500
         2                0.831395  0.168605
         3                0.857783  0.142217
         4                0.823529  0.176471
         All              0.838776  0.161224
```

In [44]: pd.crosstab(columns=[df.Attrition],index=[df.RelationshipSatisfaction],margins=True,no

```
Out[44]: Attrition                       No       Yes
         RelationshipSatisfaction
         1                         0.793478  0.206522
         2                         0.851485  0.148515
         3                         0.845316  0.154684
         4                         0.851852  0.148148
         All                       0.838776  0.161224
```

Notice that I have plotted just some of the important features against out 'Target' variable i.e. Attrition in our case. Similarly we can plot other features against the 'Target' variable and analye the trends i.e. how the feature effects the 'Target' variable.

1.5 Feature Selection

```
In [45]: df.drop(['BusinessTravel','DailyRate','EmployeeCount','EmployeeNumber','HourlyRate','I
                   ,'NumCompaniesWorked','Over18','StandardHours', 'StockOptionLevel','Training
```

1.6 Feature Encoding

I have used the Label Encoder from the scikit library to encode all the categorical features.

```
In [46]: def transform(feature):
             le=LabelEncoder()
             df[feature]=le.fit_transform(df[feature])
             print(le.classes_)
```

```
In [47]: cat_df=df.select_dtypes(include='object')
         cat_df.columns
```

```
Out[47]: Index(['Attrition', 'Department', 'EducationField', 'Gender', 'JobRole',
                'MaritalStatus', 'OverTime'],
               dtype='object')
```

```
In [48]: for col in cat_df.columns:
             transform(col)
```

```
['No' 'Yes']
['Human Resources' 'Research & Development' 'Sales']
['Human Resources' 'Life Sciences' 'Marketing' 'Medical' 'Other'
 'Technical Degree']
['Female' 'Male']
['Healthcare Representative' 'Human Resources' 'Laboratory Technician'
 'Manager' 'Manufacturing Director' 'Research Director'
 'Research Scientist' 'Sales Executive' 'Sales Representative']
['Divorced' 'Married' 'Single']
['No' 'Yes']
```

```
In [49]: df.head() # just to verify.
```

```
Out[49]:    Age  Attrition  Department  DistanceFromHome  Education  EducationField  \
         0   41          1           2                 1          2               1
         1   49          0           1                 8          1               1
         2   37          1           1                 2          2               4
         3   33          0           1                 3          4               1
         4   27          0           1                 2          1               3

            EnvironmentSatisfaction  Gender  JobInvolvement  JobLevel  \
         0                        2       0               3         2
         1                        3       1               2         2
         2                        4       1               2         1
         3                        4       0               3         1
         4                        1       1               3         1

                     ...          OverTime  PercentSalaryHike  PerformanceRating  \
```

```
0          ...                 1               11                3
1          ...                 0               23                4
2          ...                 1               15                3
3          ...                 1               11                3
4          ...                 0               12                3

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

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

   YearsWithCurrManager
0                     5
1                     7
2                     0
3                     0
4                     2

[5 rows x 24 columns]
```

1.7 Feature Scaling.

The scikit library provides various types of scalers including MinMax Scaler and the Standard-Scaler. Below I have used the StandardScaler to scale the data.

```
In [50]: scaler=StandardScaler()
         scaled_df=scaler.fit_transform(df.drop('Attrition',axis=1))
         X=scaled_df
         Y=df['Attrition'].as_matrix()
```

1.8 Splitting the data into training and validation sets

```
In [51]: x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.25,random_state=42)
```

```
In [52]: oversampler=SMOTE(random_state=42)
         x_train_smote,  y_train_smote = oversampler.fit_sample(x_train,y_train)
```

1.9 2. Using the Right Evaluation Metric

Another important point while dealing with the imbalanced classes is the choice of right evaluation metrics.

Note that accuracy is not a good choice. This is because since the data is skewed even an algorithm classifying the target as that belonging to the majority class at all times will achieve a very high accuracy.

Hence in these type of cases we may use other metrics such as --> 'Precision'-- (true positives)/(true positives+false positives) 'Recall'-- (true positives)/(true positives+false negatives) 'F1 Score'-- The harmonic mean of 'precision' and 'recall' 'AUC ROC'-- ROC curve is a plot between 'senstivity' (Recall) and '1-specificity' (Specificity=Precision) 'Confusion Matrix'-- Plot the entire confusion matrix

1.10 3. Building A Model & Making Predictions

In [53]:
```
def compare(model):
    clf=model
    clf.fit(x_train_smote,y_train_smote)
    pred=clf.predict(x_test)

    # Calculating various metrics
    acc.append(accuracy_score(pred,y_test))
    prec.append(precision_score(pred,y_test))
    rec.append(recall_score(pred,y_test))
    auroc.append(roc_auc_score(pred,y_test))
```

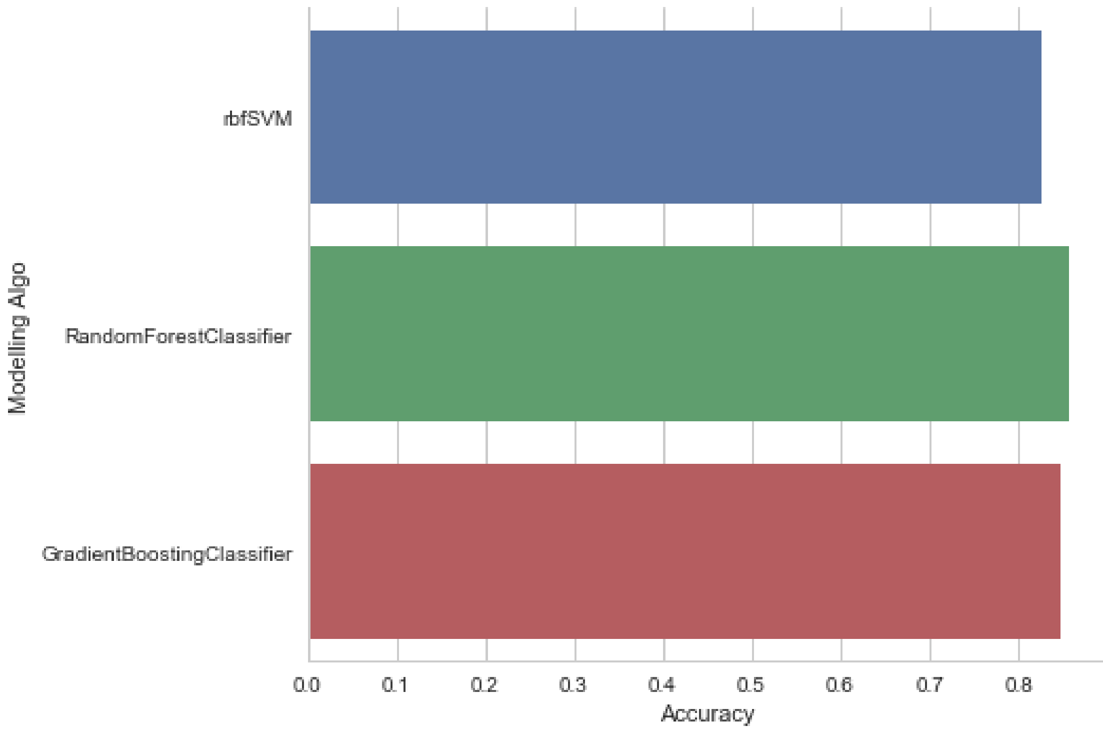
In [54]:
```
acc=[]
prec=[]
rec=[]
auroc=[]
models=[SVC(kernel='rbf'),RandomForestClassifier(),GradientBoostingClassifier()]
model_names=['rbfSVM','RandomForestClassifier','GradientBoostingClassifier']

for model in range(len(models)):
    compare(models[model])

d={'Modelling Algo':model_names,'Accuracy':acc,'Precision':prec,'Recall':rec,'Area Und
met_df=pd.DataFrame(d)
met_df
```

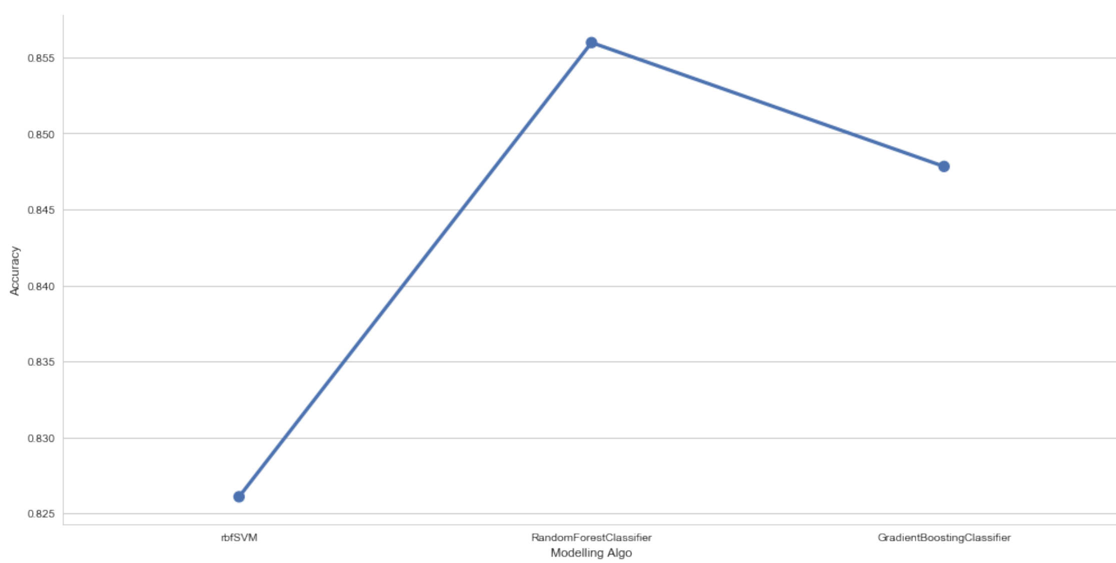
Out[54]:
```
                Modelling Algo  Accuracy  Precision    Recall  \
0                       rbfSVM  0.826087   0.479167  0.370968
1       RandomForestClassifier  0.855978   0.187500  0.391304
2   GradientBoostingClassifier  0.847826   0.291667  0.388889

   Area Under ROC Curve
0              0.644634
1              0.639130
2              0.643240
```

1.11 Conclusion: Comparing Different Models

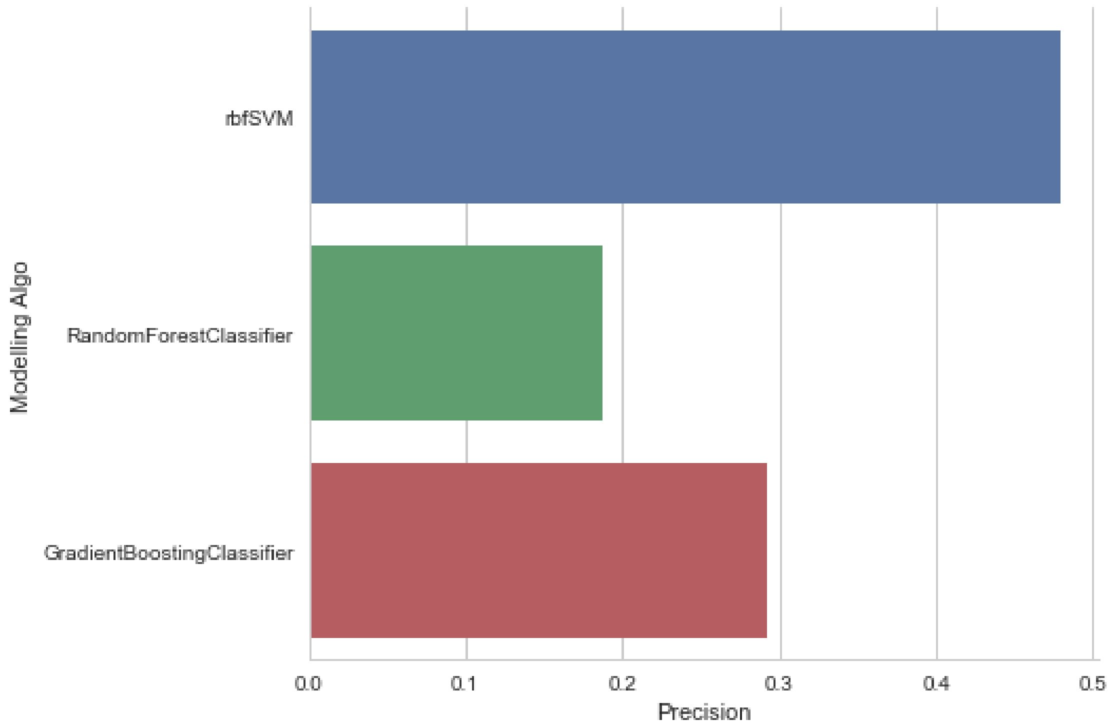
```
In [55]: def comp_models(met_df,metric):
             sns.factorplot(data=met_df,x=metric,y='Modelling Algo',size=5,aspect=1.5,kind='ba
             sns.factorplot(data=met_df,y=metric,x='Modelling Algo',size=7,aspect=2,kind='poin
```

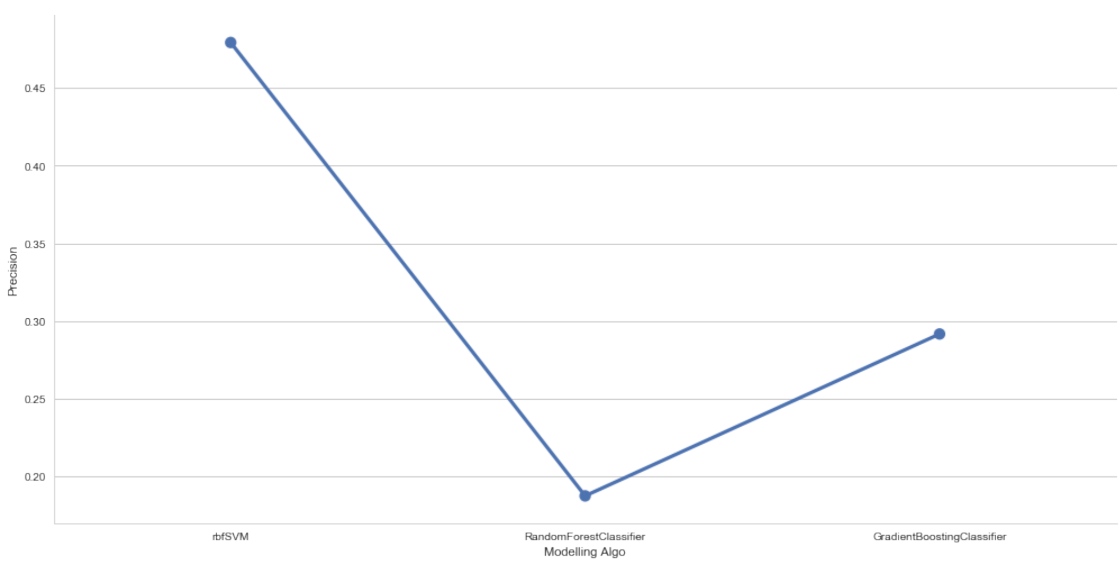
```
In [56]: comp_models(met_df,'Accuracy')
```



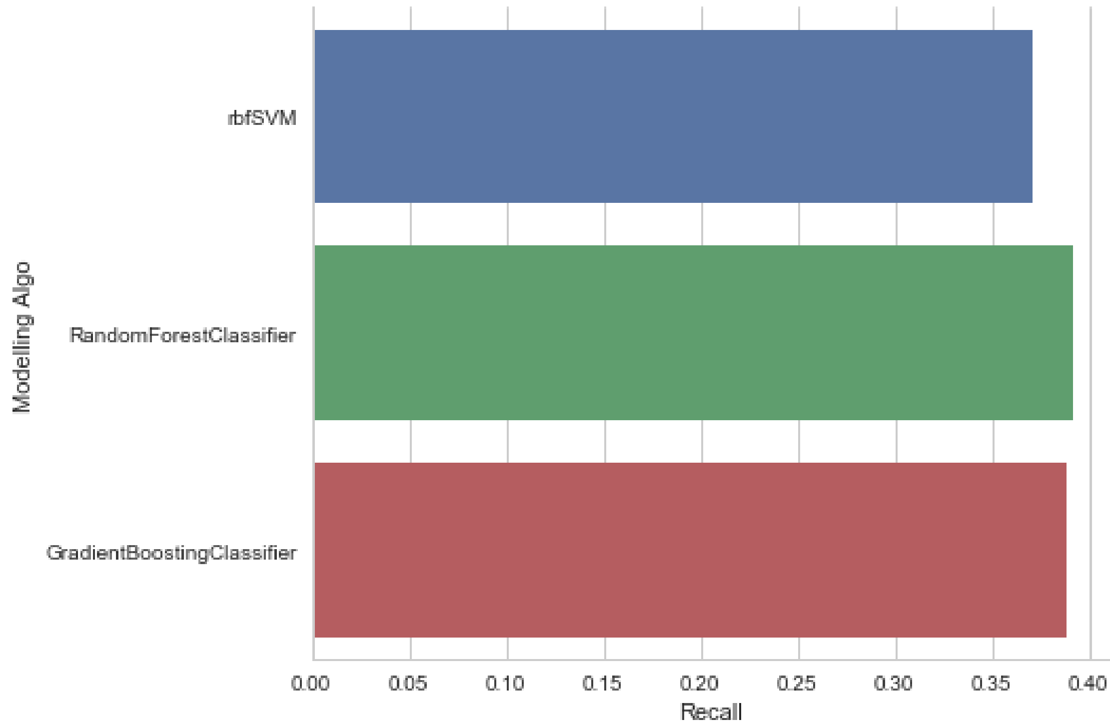


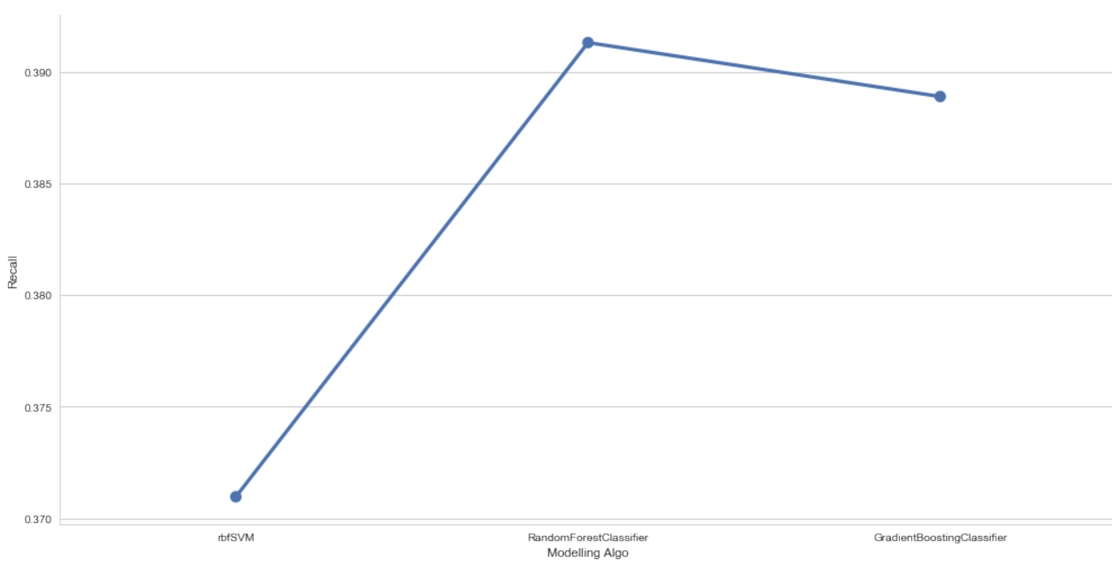
```
In [57]: comp_models(met_df,'Precision')
```



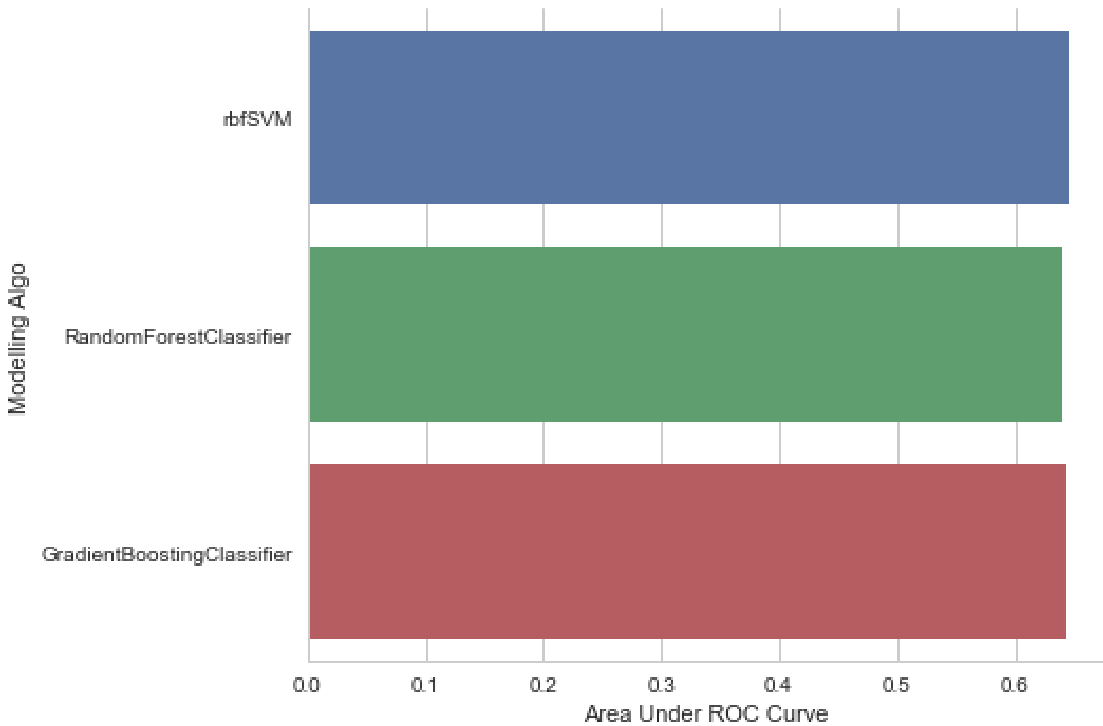


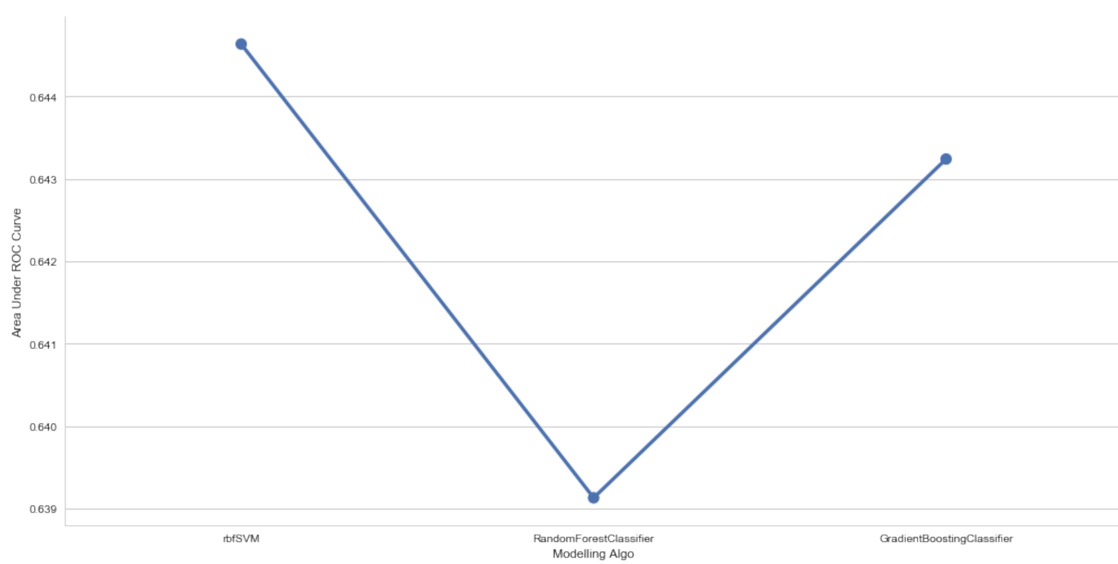
```
In [58]: comp_models(met_df,'Recall')
```






```
In [59]: comp_models(met_df,'Area Under ROC Curve')
```





The above data frame and the visualizations summarize the resuts after training different models on the given dataset.