HR Analytics Project

1. Problem Definition.
2. Data Analysis.
3. EDA Concluding Remark.
4. Pre-Processing Pipeline.
5. Building Machine Learning Models.
6. Concluding Remarks.

Presented by-

Shailja Tiwari

Data Trained

Batch-1840

1. Problem Definition.

Theoretical overview:

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

HR Analytics:

- HR Analytics is also known as people analytics
- Human resource analytics (HR analytics) is an area in the field of analytics that refers to applying analytic processes to the human resource department of an organization in the hope of improving employee performance and therefore getting a better return on investment.
-

- HR analytics does not just deal with gathering data on employee efficiency. Instead, it aims to provide insight into each process by gathering data and then using it to make relevant decisions about how to improve these processes.

Attrition in HR:

- Attrition is the departure of employees from the organization for any reason (voluntary or involuntary), including resignation, termination, death or retirement.

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

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

Attrition affecting Companies:

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

Data Source:

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

Problem Statement:

We have to analyse the given data and apply all the domain technical knowledge in finding the factors that affect the employee attrition rate in the various organisation.

2. Data Analysis.

Data analysis refers to the process of inspecting, cleansing, transforming, and modelling data with the aim of discovering useful insights and information from the data which can be used for business developments and easy representation of the raw data to the non tech group and decision makers of any organization.

In data analytics and data science, there are four main types of analysis:

- Descriptive.
- Diagnostic.
- predictive
- prescriptive.

Various steps are involved in the data analysis process. These steps include importing all the important libraries, loading the dataset in the form of DataFrame by the help of Pandas library present in Python.

Importing some of the important Libraries

n [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Loading the Dataset.

n [2]:
```
df = pd.read_csv(r'C:\Users\shailja tiwari\Downloads\WA_Fn-UseC_-HR-Employee-Attrition.csv')
```

n [3]:
```
df
```

ut[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	StandardHo
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061	...	3	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062	...	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064	...	2	
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065	...	4	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	2068	...	1	

1470 rows × 35 columns

n [4]:
```
df.shape
```

ut[4]:
```
(1470, 35)
```

- So in our dataset we have total 1470 number of rows and 35 columns are present.

```
df.shape
(1470, 35)
```

Dataframe Description:

Dataset is divided in two parts input (Independent/Feature) variables and output(dependent/Label/Target) variables.

Feature Columns:

- Age
- 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

Label column:

- Attrition

- Feature columns also called as input or independent column has mixed type of data i.e continuous and categorical.

- Label/target column also called as output or dependent variable has numerical data.

Checking for the null or missing values in the dataset for proper model building and prediction the data set must contain non null values.

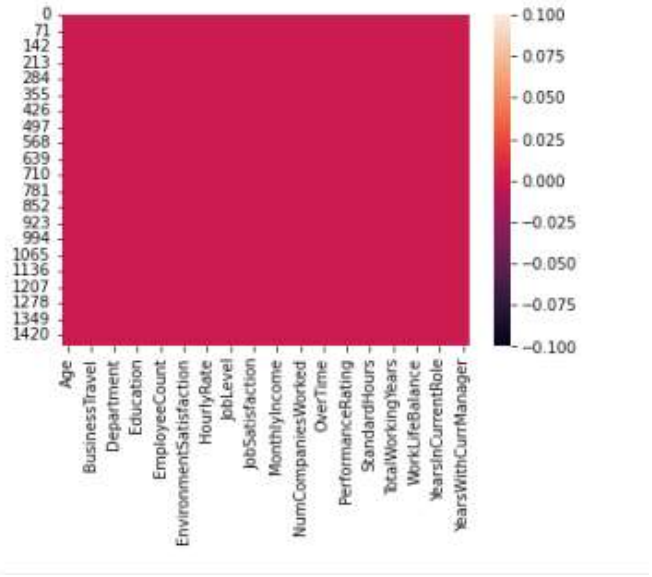
We will visualise the same with the help of heatmap and code.

df.isnull().sum()

There are no null values in the dataset.

```
In [7]: sns.heatmap(df.isna())
        # no null values present in the datset
```

Out[7]: <AxesSubplot:>



Getting the information about datatypes present in dataset.

df.info()

```
[9]: 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
```

Establishing the Statistical description of the dataset.

df.describe()

```
]: df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	...	Relationsh
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	...	
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	...	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	...	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	...	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	...	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	...	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	...	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	...	

8 rows × 26 columns

- In most columns mean and 50% are similar in value, and mean is greater than standard deviation, indicating a symmetric distribution.
- TotalWorkingYears,YearsAtCompany,YearsInCurrentRole,YearsSinceLastPromotion,YearsWithCurrManager have a big difference between 75% and max indicating the presence of outliers.
- Due to higher standard deviation skewness may be present in the dataset.

3. Exploratory Data Analysis (EDA).

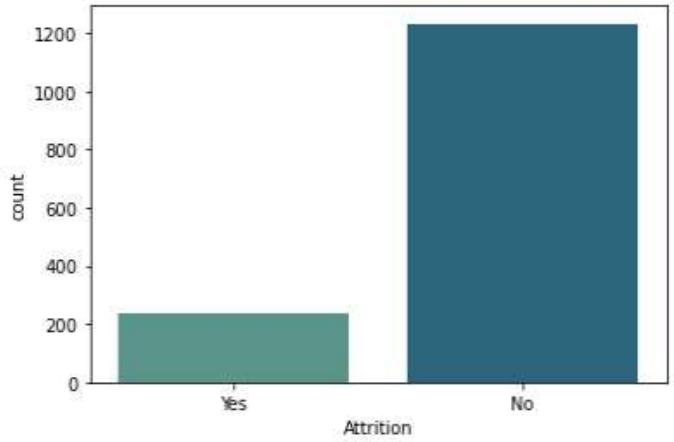
EDA is the method of analysing data using visual methods. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

TYPES OF EXPLORATORY DATA ANALYSIS:

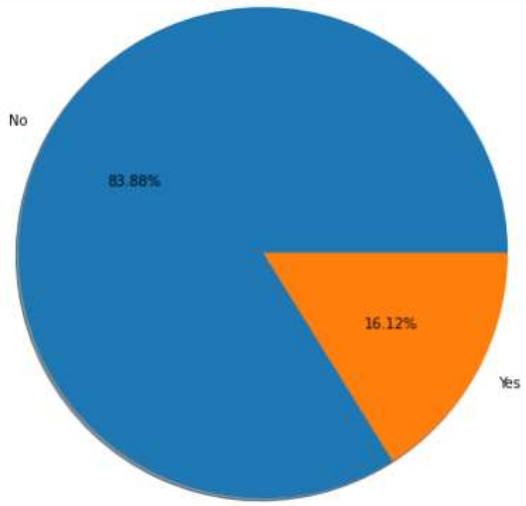
- Univariate Analysis
- Bivariate Analysis
- Multivariate Analysis

Univariate Analysis

```
.6]: sns.countplot(df['Attrition'], palette="crest")
.6]: <AxesSubplot:xlabel='Attrition', ylabel='count'>
```




```
In [17]: labels = 'No','Yes'
fig, ax = plt.subplots()
ax.pie(df['Attrition'].value_counts(),labels = labels,radius =2,autopct = '%1.2f%%', shadow=True)
plt.show()
```

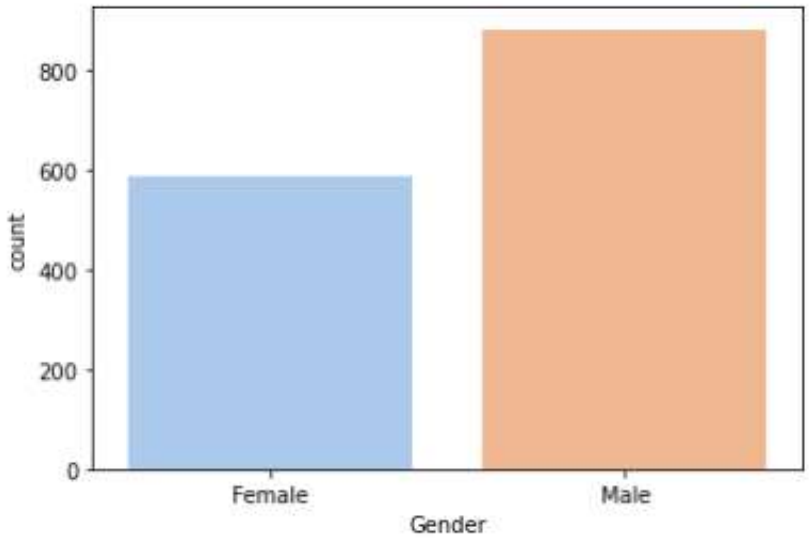


Class

'No' = 83.88% of total values

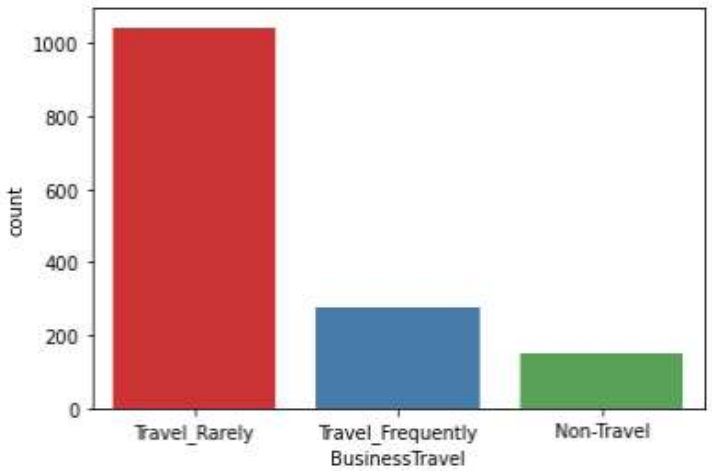
'Yes'= 16.12% of total values

```
[18]: df['Gender'].value_counts()

[18]: Male      882
      Female    588
      Name: Gender, dtype: int64

[19]: sns.countplot(df['Gender'], palette="pastel")

[19]: <AxesSubplot:xlabel='Gender', ylabel='count'>
```



Female employees are less than Male employees

```
[20]: df['BusinessTravel'].value_counts()

[20]: Travel_Rarely        1043
      Travel_Frequently     277
      Non-Travel            150
      Name: BusinessTravel, dtype: int64

[21]: sns.countplot(df['BusinessTravel'], palette="Set1")

[21]: <AxesSubplot:xlabel='BusinessTravel', ylabel='count'>
```



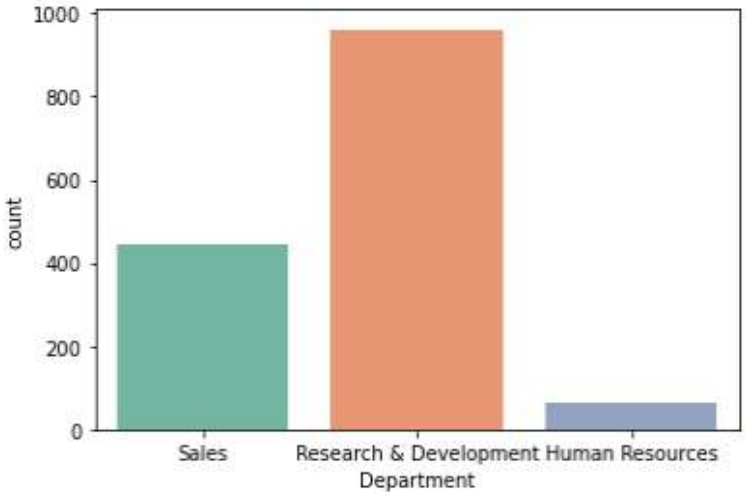
Most employees Travel Rarely

```
[22]: df['Department'].value_counts()
```

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

```
[23]: sns.countplot(df['Department'], palette="Set2")
```

```
[23]: <AxesSubplot:xlabel='Department', ylabel='count'>
```



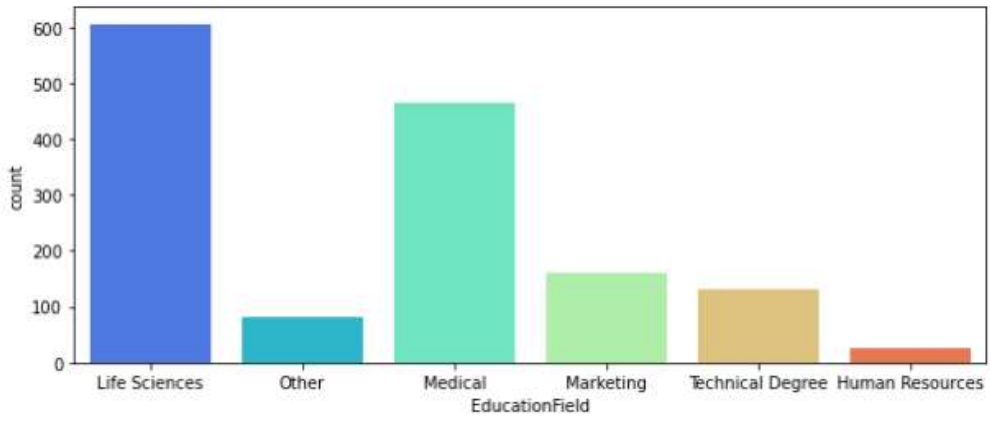
Most employees belong to Research and Development.

```
In [24]: df['EducationField'].value_counts()
```

```
Out[24]: Life Sciences       606
         Medical             464
         Marketing           159
         Technical Degree    132
         Other                82
         Human Resources      27
         Name: EducationField, dtype: int64
```

```
In [25]: plt.figure(figsize=(10,4),facecolor='white')
         sns.countplot(df['EducationField'], palette="rainbow")
```

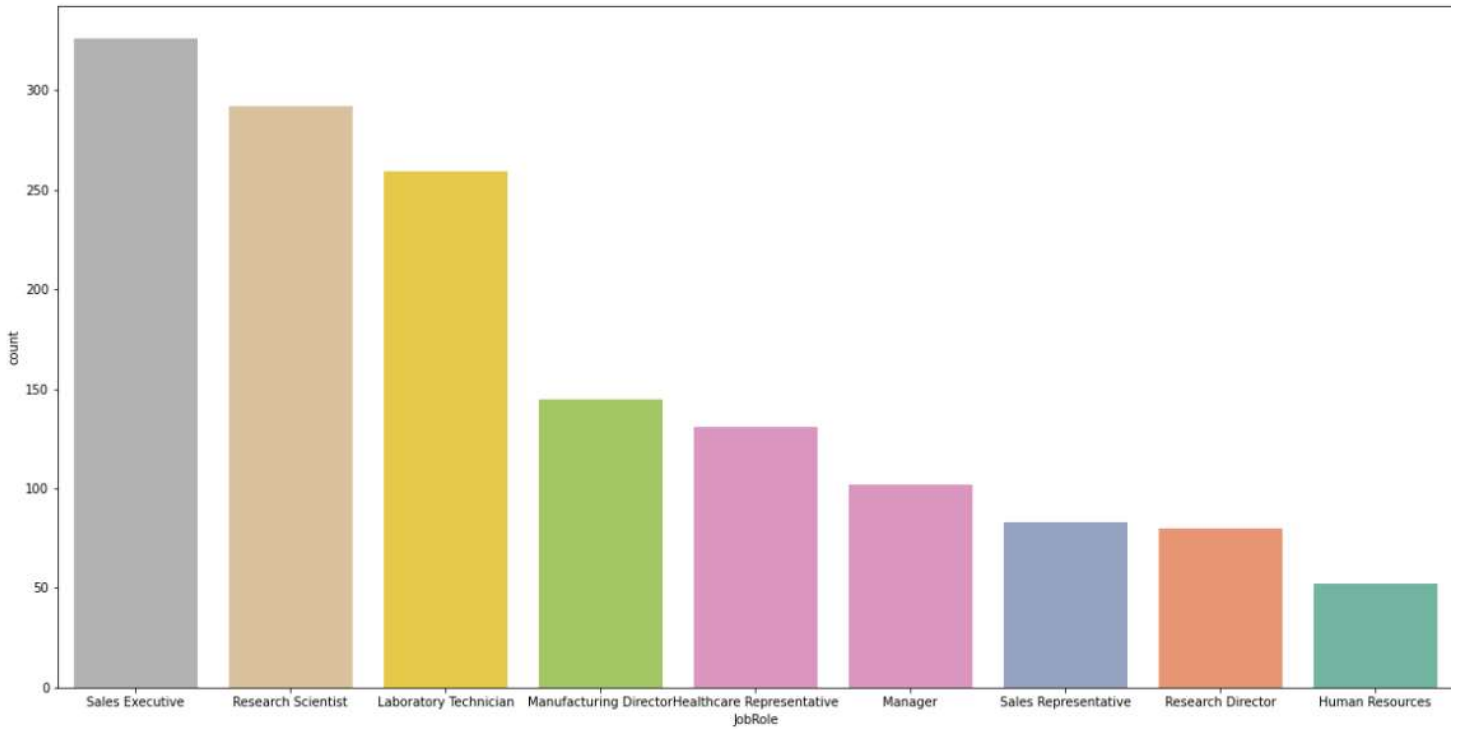
```
Out[25]: <AxesSubplot:xlabel='EducationField', ylabel='count'>
```



Life Sciences is the most common Education field which is found in this dataset.

```
plt.figure(figsize=(20,10),facecolor='white')
sns.countplot(df['JobRole'], palette="Set2_r")
```

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



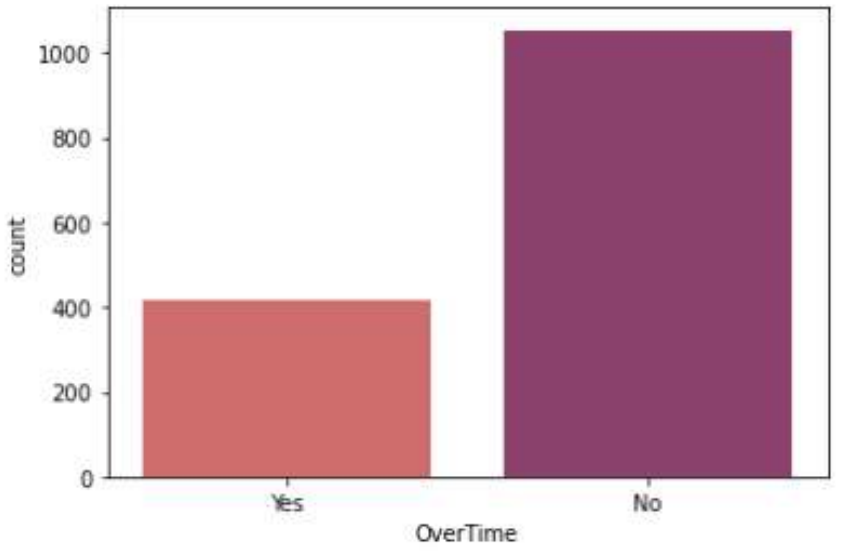
Highest number of Employees are Sales Executives, followed by Research Scientist and Laboratory Technicians

```
df['OverTime'].value_counts()
```

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

```
sns.countplot(df['OverTime'], palette="flare")
```

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



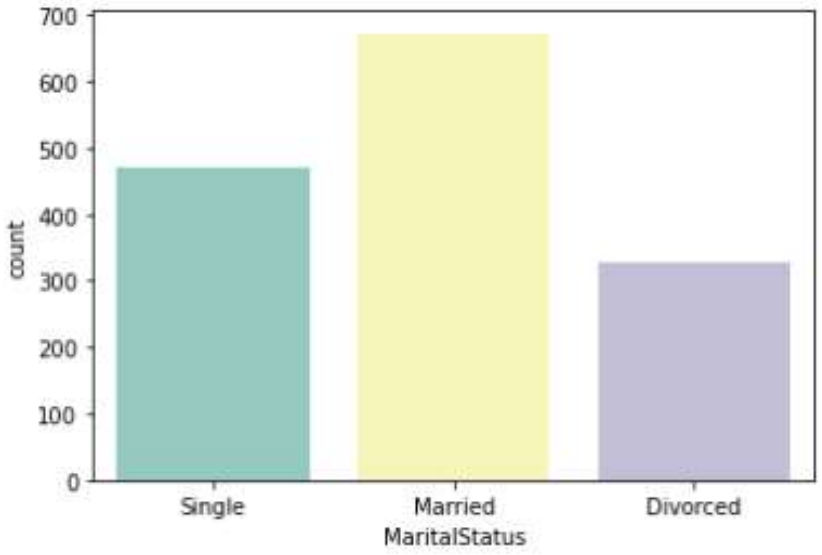
Maximum employees don't over time.

```
df['MaritalStatus'].value_counts()
```

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

```
sns.countplot(df['MaritalStatus'], palette="Set3")
```

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

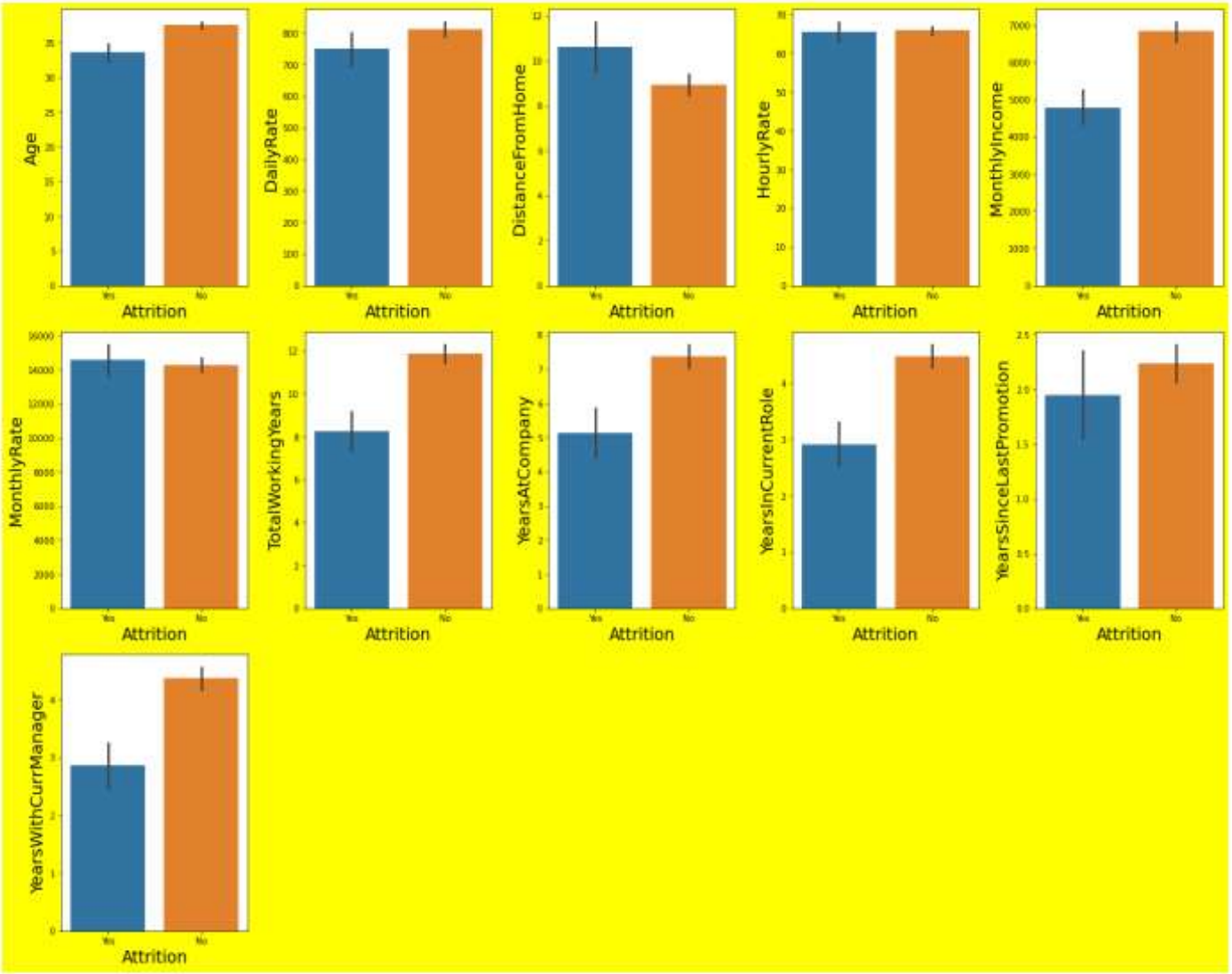


Maximum number of employees are married.

Interpreting Relationship between Dependent Variable and Independent Variables

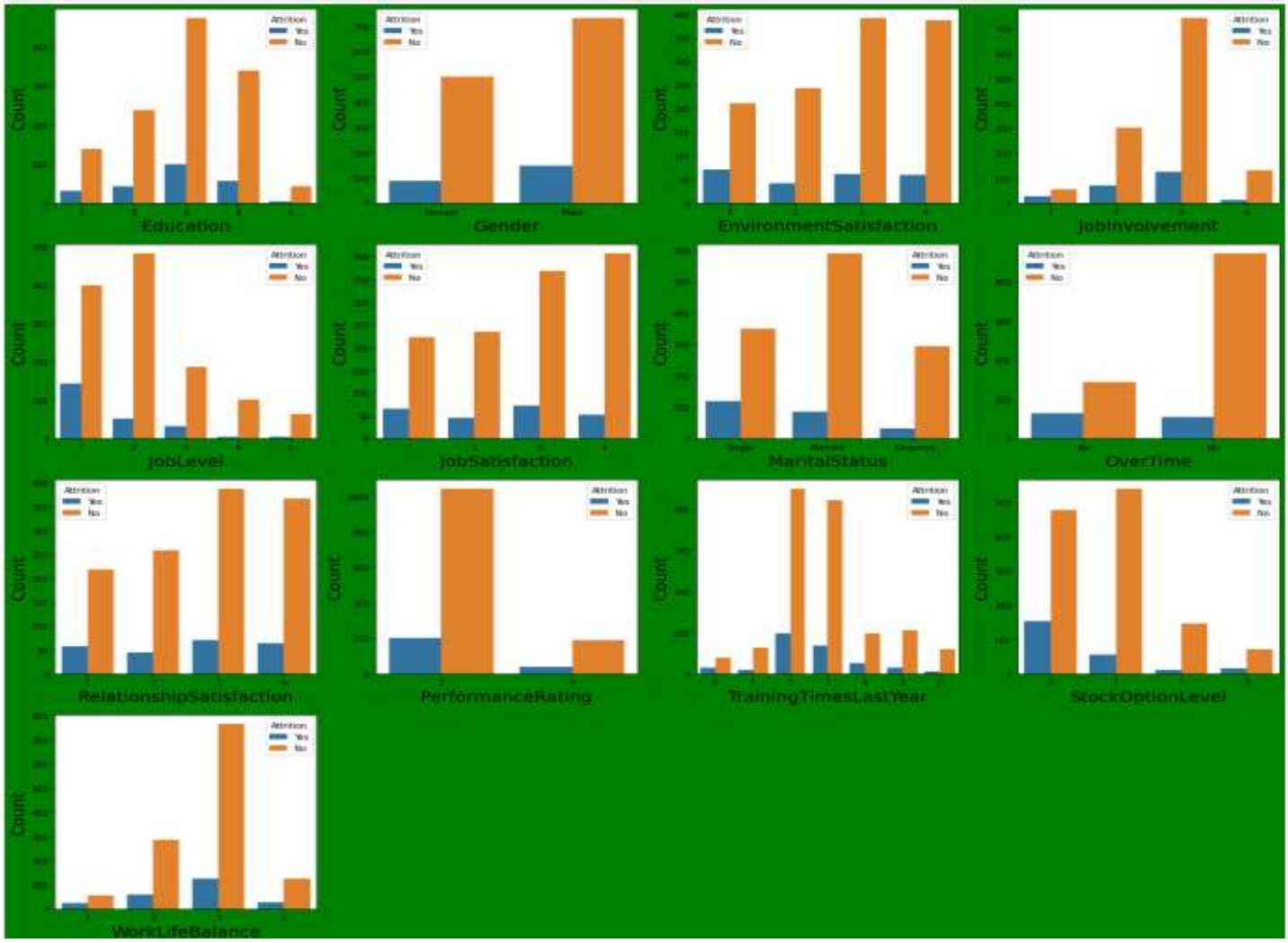
- **Bivariate and Multivariate Analysis**

[22]:
```
plt.figure(figsize=(20,25),facecolor='yellow')
plotnum=1
y = df['Attrition']
X = df[['Age','DailyRate','DistanceFromHome','HourlyRate','MonthlyIncome','MonthlyRate','TotalWorkingYears','YearsAtCompany','YearsInCurrentRole','
for col in X:
    if plotnum<=15:
        plt.subplot(5,5,plotnum)
        sns.barplot(y,X[col])
        plt.xlabel('Attrition',fontsize=19)
        plt.ylabel(col,fontsize=20)
    plotnum+=1
plt.tight_layout()
```



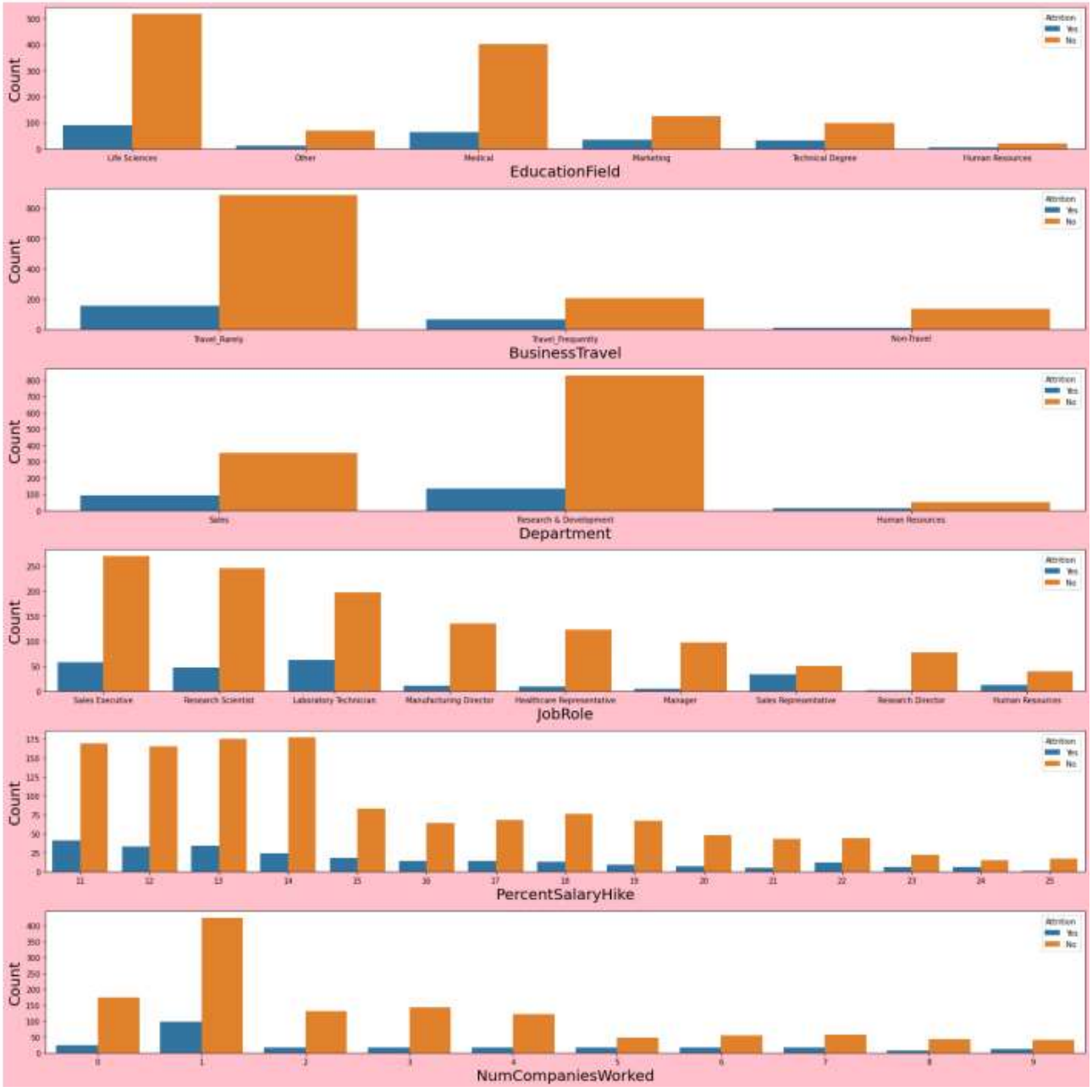
'Attrition' vs Categorical Data Columns

```
plt.figure(figsize=(20,21),facecolor='green')
plotnum=1
X = df[['Education','Gender','EnvironmentSatisfaction','JobInvolvement','JobLevel','JobSatisfaction','MaritalStatus','OverTime','RelationshipSatisf
y = df['Attrition']
for col in X:
    if plotnum<=23:
        plt.subplot(5,4,plotnum)
        sns.countplot(X[col],hue =y)
        plt.xlabel(col,fontsize=20)
        plt.ylabel('Count',fontsize=20)
    plotnum+=1
plt.tight_layout()
```



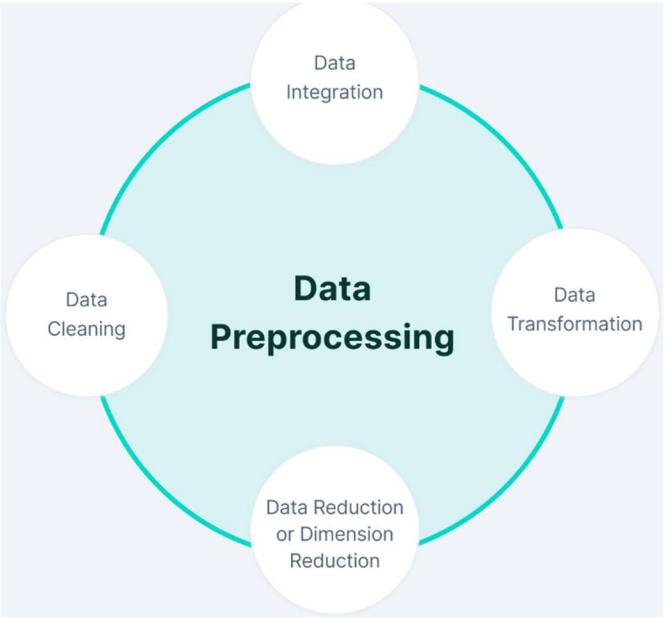
- Employees with Education '3' and '4' are retained the most.
- Employees with Education '3' mostly leave.
- Employees with EducationFields Life Sciences and Medical are retained the most.
- Employees with EducationFields Life Sciences mostly leave
- The number of male employees retained is a bit greater than the number of female employees retained.
- Male employees also contribute the most to Attrition.
- Employees with greater Environment Satisfaction are retained. While Employees with lowest Environment Satisfaction contribute the most to Attrition.
- Employees with Job Involvement of 3 are retained the most.

```
plt.figure(figsize=(20,20),facecolor='pink')
plotnum=1
X = df[['EducationField','BusinessTravel','Department','JobRole','PercentSalaryHike','NumCompaniesWorked']]
y = df['Attrition']
for col in X:
    if plotnum<=23:
        plt.subplot(6,1,plotnum)
        sns.countplot(X[col],hue =y)
        plt.xlabel(col,fontsize=20)
        plt.ylabel('Count',fontsize=20)
    plotnum+=1
plt.tight_layout()
```



- Employees who Travel rarely are retained the most.
- Employees belonging to Reseach and Development are retained the most followed by Sales.
- While Employees belonging to Reseach and Development also contribute the most to Attrition.

4. Pre-Processing Pipeline.

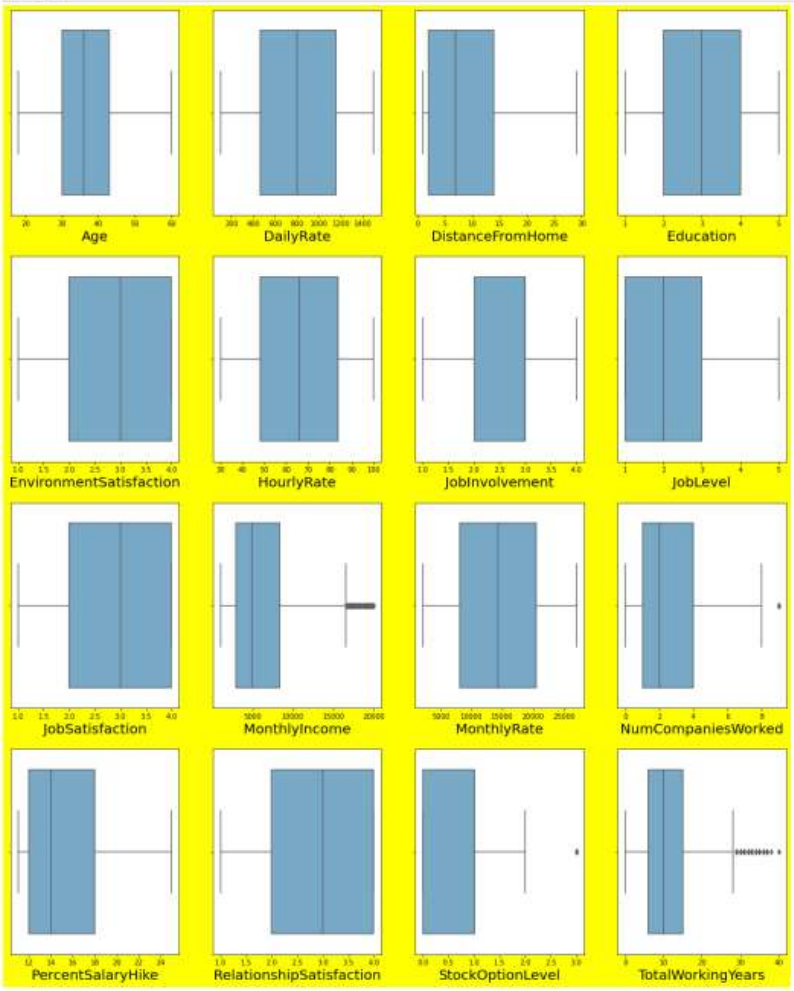


Outliers visualisation and removal

Use of Boxplot for visualising and checking outliers

Boxplot Visualization for checking the Outliers

```
plt.figure(figsize=(20,25),facecolor='yellow')
plotnum=1
for col in X.drop(columns = ['BusinessTravel','Department','PerformanceRating','EducationField','Gender','JobRole','MaritalStatus','OverTime']):
    if plotnum<=16:
        plt.subplot(4,4,plotnum)
        sns.boxplot(X[col], palette='Blues', orient='v')
        plt.xlabel(col,fontsize=20)
    plotnum+=1
plt.show()
```



An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. During the model building and for precise and accurate predictions these normal values are removed.

Outliers are stragglers extremely high or extremely low values. For example, if you were measuring children's nose length, your average value might be thrown off if Pinocchio was in the class.

1. Removing Outliers using Interquartile Range or IQR

The IQR describes the middle 50% of values when ordered from lowest to highest. The IQR is then the difference between Third quartile and First quartile.
IQR=Q3-Q1

2. Removing Outliers using Z score method

- A Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values.
- Z-score is measured in terms of standard deviations from the mean.
- If a Z-score is 0, it indicates that the data point's score is identical to the mean score.
- A Z-score of 1.0 would indicate a value that is one standard deviation from the mean.
- Z-scores may be positive or negative, with a positive value indicating the score is above the mean and a negative score indicating it is below the mean.

```
from scipy.stats import zscore zscor = zscore(df2) z_score_abs = np.abs(zscor)
df3 = df2[(z_score_abs < 3).all(axis=1)]
#taking 3 as threshold value
```

Checking for skewness in the data set

- The skewness is a measure of symmetry or asymmetry of data distribution, and kurtosis measures whether data is heavy-tailed or light-tailed in a normal distribution.

- Data can be positive-skewed (data-pushed towards the right side) or negative-skewed (data-pushed towards the left side).
- It is the asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right.

Effect of skewness on the dataset

- Skewed data degrades the model's ability (especially regression based models) to describe typical cases as it has to deal with rare cases on extreme values.

To check for skew in data:

```
df.skew().sort_values(ascending=False)
```

Reducing skewness in the dataset and for receiving a normal distribution

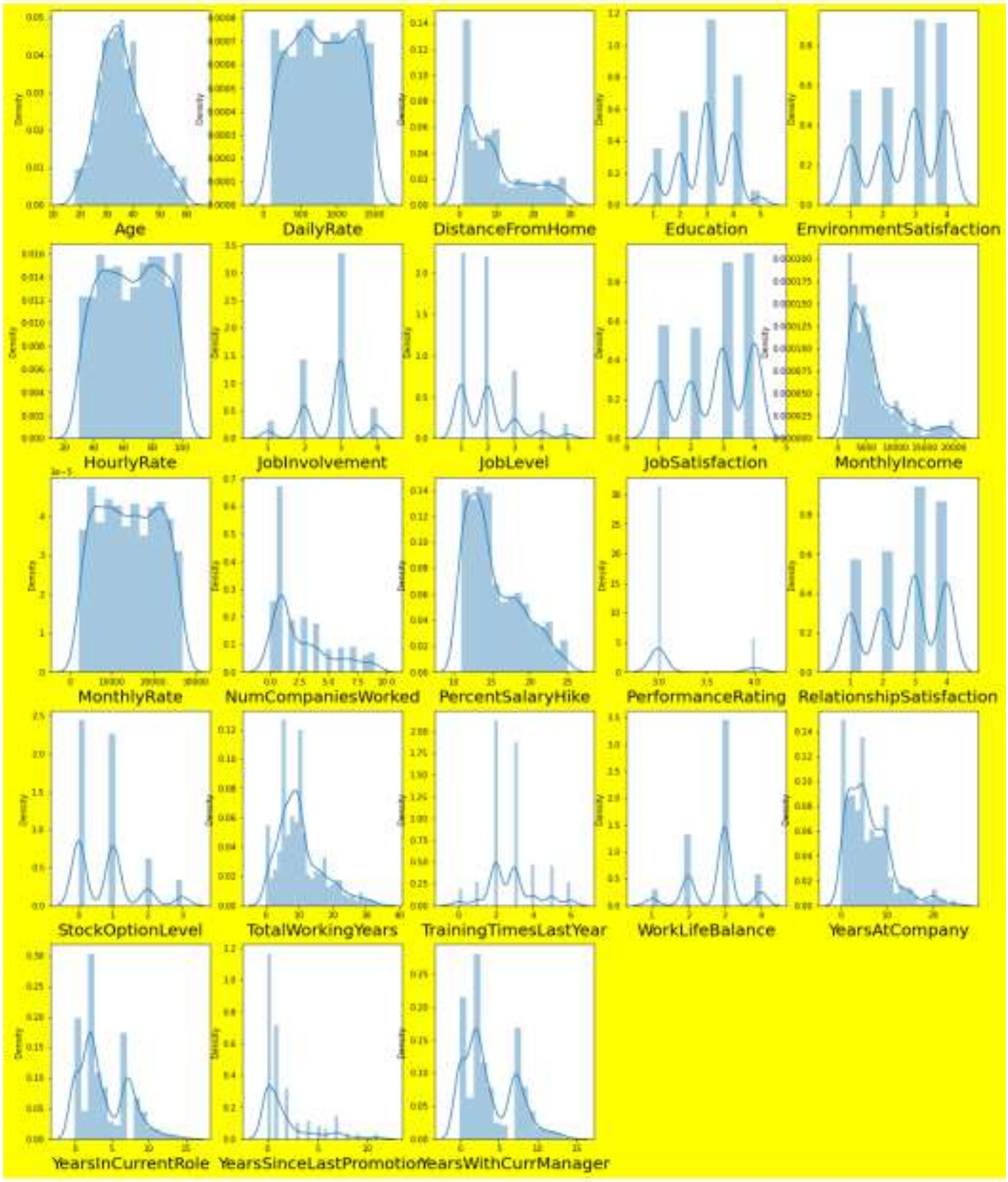
Using Power transformer technique

```
from sklearn.preprocessing import PowerTransformer

powtrans= PowerTransformer(method='yeo-johnson', standardize=True)

df4 = X[['DistanceFromHome', 'MonthlyIncome', 'TotalWorkingYears',
'YearsAtCompany','PercentSalaryHike','YearsWithCurrManager','YearsInCurrentRole' ]]

transformed= powtrans.fit_transform(df4)

transformed = pd.DataFrame(transformed, columns=df4.columns)


transformed.skew()
```

Visualisation using distribution plot.

Checking the skewness level with the Distribution plots

```
plt.figure(figsize=(20,25),facecolor='yellow')
plotnum=1
for col in X.drop(columns = ['BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','OverTime']):
    if plotnum<=35:
        plt.subplot(5,5,plotnum)
        sns.distplot(X[col])
        plt.xlabel(col,fontsize=20)
    plotnum+=1
plt.show()
```



Further Reduction in skewness using PowerTransformer

Normal distribution of skewed column

Skewness check after Transformation

```
plt.figure(figsize=(20,25),facecolor='teal')
plotnum=1
for col in X.drop(columns = ['BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','OverTime']):
    if plotnum<=25:
        plt.subplot(5,5,plotnum)
        sns.distplot(X[col])
        plt.xlabel(col,fontsize=20)
    plotnum+=1
plt.show()
```

Encoding the dataset using Label Encoder

Label Encoding or Ordinal Encoding

- We use this categorical data encoding technique when the categorical feature is ordinal. In this case, retaining the order is important. Hence encoding should reflect the sequence.
- In Label encoding, each label is converted into an integer value. We will create a variable that contains the categories representing the education qualification of a person.

  X[['BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','OverTime']].nunique()
- df_train_transformed = encoder.fit_transform(train_df)

Encoding using get_dummies()

- dumm=pd.get_dummies(X[['BusinessTravel','Department','EducationField','Gender','JobRole','MaritalStatus','OverTime']],drop_first = False)
- dumm

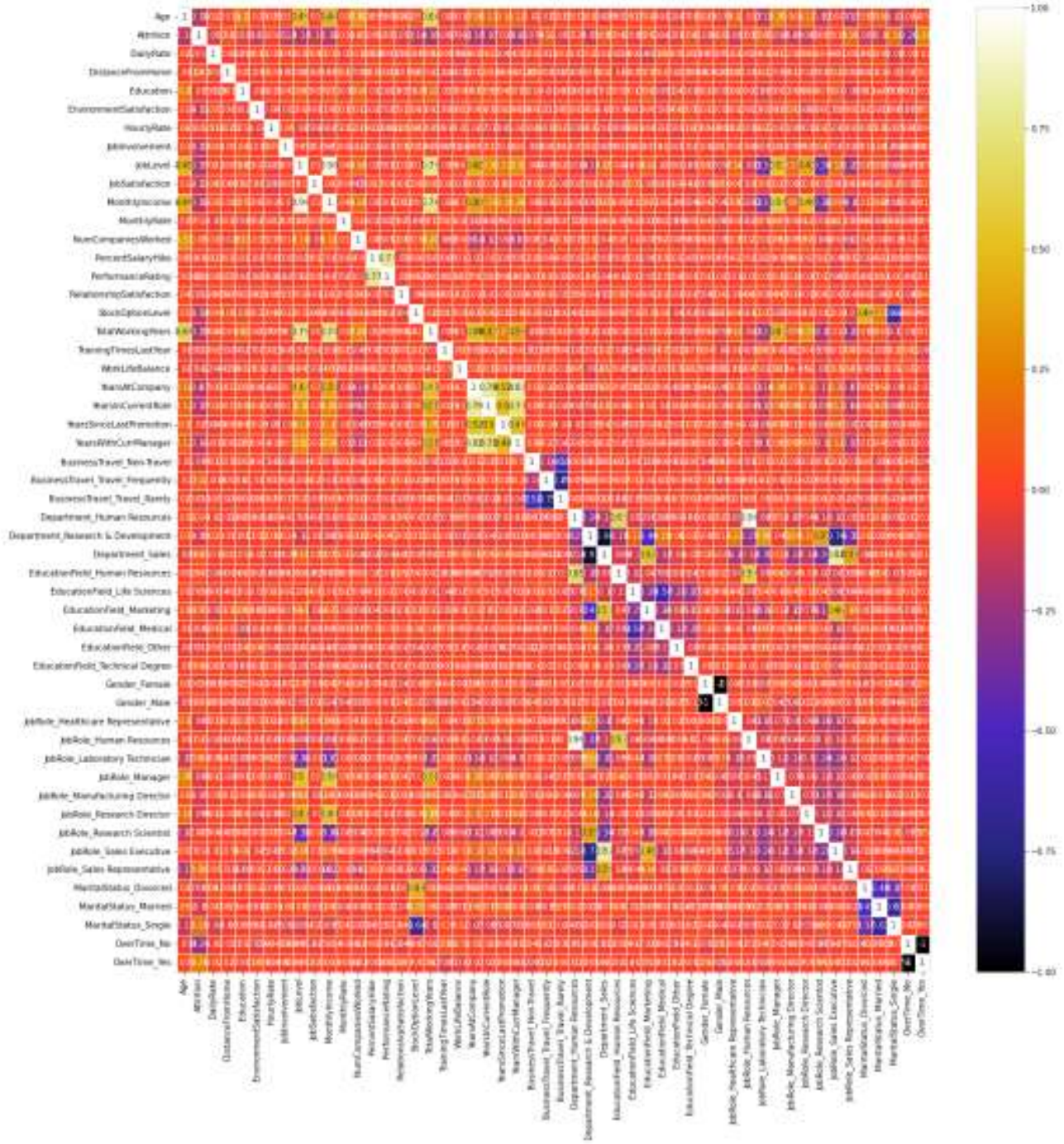
correlation matrix

- Correlation is an indication about the changes between two variables.
- A matrix is an array of numbers arranged in rows and columns.
- A correlation matrix is simply a table showing the correlation coefficients between variables. Here, the variables are represented in the first row, and in the first column

Establishing the correlation between various columns

- h_corr = df.corr()
- h_corr

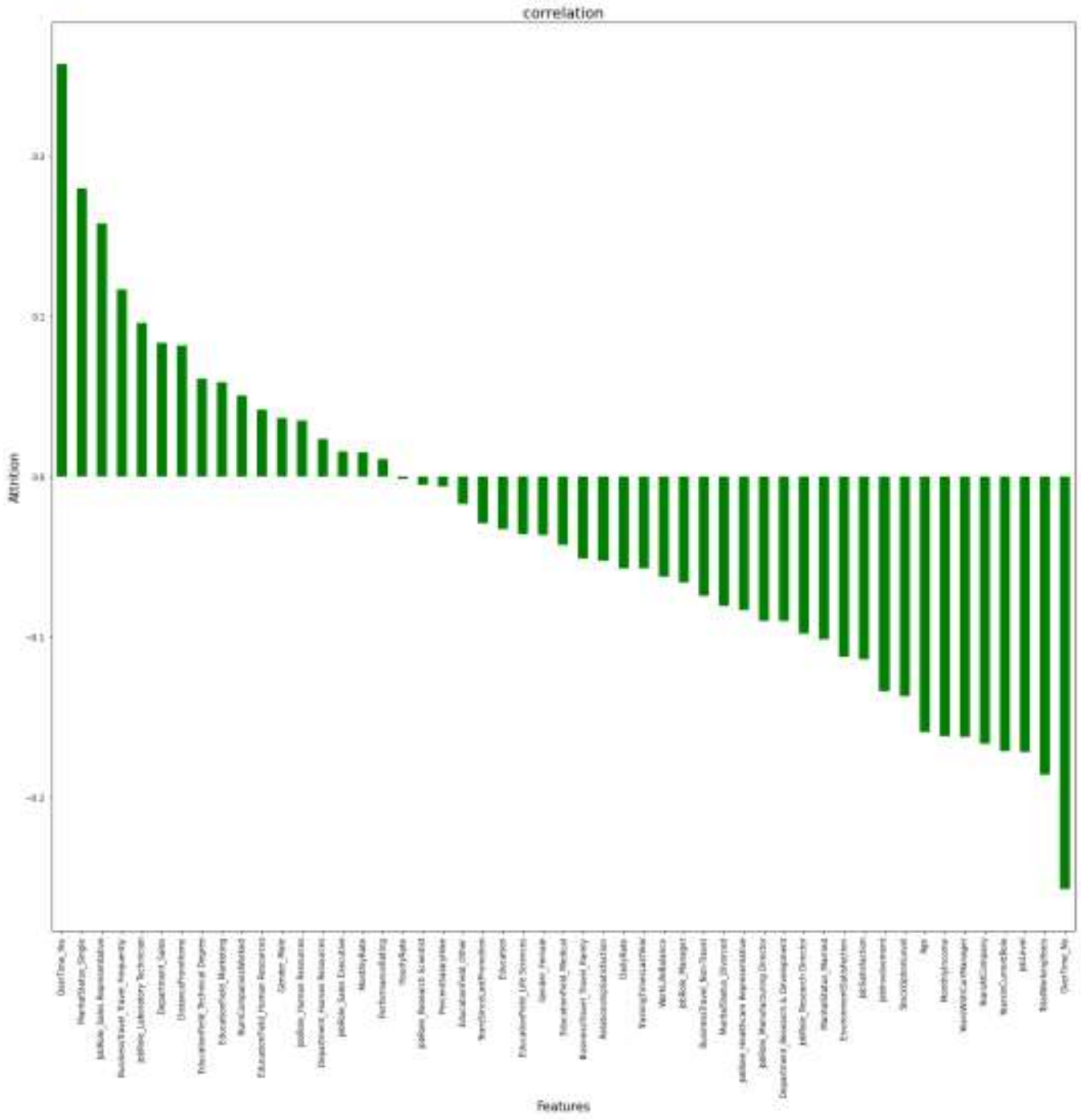
```
plt.figure(figsize=(20,21))
sns.heatmap(h_corr,annot=True,linewidth=1, cmap='CMRmap')
plt.show()
```



Visualizing correlation of feature columns with label column.

- plt.figure(figsize = (22,20))
- df.corr()['Attrition'].sort_values(ascending = False).drop(['Attrition']).plot(kind='bar',color = 'g')
- plt.xlabel('Features',fontsize=15) plt.ylabel('Attrition',fontsize=15) plt.title('correlation',fontsize = 18)
- plt.show()

```
plt.figure(figsize = (22,20))
df.corr()['Attrition'].sort_values(ascending = False).drop(['Attrition']).plot(kind='bar',color = 'g')
plt.xlabel('Features',fontsize=15)
plt.ylabel('Attrition',fontsize=15)
plt.title('correlation',fontsize = 18)
plt.show()
```



Feature Scaling using Standard Scaler

- Feature Scaling is a technique of bringing down the values of all the independent features of our dataset on the same scale. Feature selection helps to do calculations in algorithms very quickly. It is the important stage of data pre-processing.

Feature Scaling using Standard Scaler

Standard Scaler removes the mean and scales the data to the unit variance. Standard Scaler comes into play when the characteristics of the input dataset differ greatly between their ranges, or simply when they are measured in different units of measure.

- from sklearn.preprocessing import StandardScaler
- scaler= StandardScaler()
- scaled_X = scaler.fit_transform(X)
- scaled_X

Multicollinearity

Multicollinearity is a well-known challenge in multiple regression. The term refers to the high correlation between two or more explanatory variables, i.e. predictors.

According to Graham's study, multicollinearity in multiple regression leads to:

- Inaccurate parameter estimates,
- Decreased power
- Exclusion of significant predictors

Dealing with Multicollinearity

- You can deal with multicollinearity by

- dropping highly correlated variables
- extracting new features with Principal Component Analysis (PCA)

Multicollinearity using Variance Inflation Factor

- from statsmodels.stats.outliers_influence import variance_inflation_factor
- vif = pd.DataFrame()
- vif["Features"] = X.columns vif['vif'] = [variance_inflation_factor(scaled_X,i) for i in range(scaled_X.shape[1])]
- vif

Selecting best features based on their scores and dropping a highly colinear columns.

- **X_best = X.drop(columns=['MonthlyIncome','YearsSinceLastPromotion','Department_Human Resources','EducationField_Other','JobRole_Sales Executive',' MonthlyIncome]**
- **x_best**

5. Building Machine Learning Models.

Machine Learning

A machine learning model is defined as a mathematical representation of the output of the training process.

Machine learning is the study of different algorithms that can improve automatically through experience & old data and build the model.

A machine learning model is similar to computer software designed to recognize patterns or behaviours based on previous experience or data.

The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.

Classification of Machine Learning Models:

Based on different business goals and data sets, there are three learning models for algorithms. Each machine learning algorithm settles into one of the three models:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Supervised Learning is further divided into two categories:

- Classification
- Regression

Unsupervised Learning is also divided into below categories:

- Clustering
- Association Rule
- Dimensionality Reduction

Since the Dataset is the classification problem as the Label data is non-numerical data Classification Model Building

Classification

Classification models are the second type of Supervised Learning techniques, which are used to generate conclusions from observed values in the categorical form. For example, the classification model can identify if the email is spam or not; a buyer will purchase the product or not, etc. Classification algorithms are used to predict two classes and categorize the output into different groups.

Classification Model Building

- from sklearn.model_selection import train_test_split
- from sklearn.metrics import accuracy_score

Finding the best random state

- from sklearn.linear_model import LogisticRegression
- maxAcc = 0
- maxRS=0
- for i in range(1,100):
- x_train,x_test,y_train,y_test = train_test_split(scaled_x_best,y,test_size = .22, random_state = i,stratify = y)
- modRF = LogisticRegression()
- modRF.fit(x_train,y_train)
- pred = modRF.predict(x_test)
- acc = accuracy_score(y_test,pred)
- if acc>maxAcc:
- maxAcc=acc
- maxRS=i
- print(f"Best Accuracy is: {maxAcc} on random_state: {maxRS}")

```
x_train,x_test,y_train,y_test = train_test_split(scaled_x_best,y,test_size = .22, stratify = y,random_state = 71) #since class imbalance exists
```

```
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
from sklearn.metrics import classification_report
from sklearn.metrics import plot_roc_curve
```

Importing all the neccessary Algorithms

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
```

```
#  Calling  the Instances
RFC = RandomForestClassifier(n_estimators=600,random_state=71,criterion='gini', max_depth=4)
LOGR= LogisticRegression(solver='liblinear')
SV =  SVC(kernel='linear')
KNN = KNeighborsClassifier()
```

Training the Models

```
### training Random forest classifier
RFC.fit(x_train,y_train)
```

```
RandomForestClassifier(max_depth=4, n_estimators=600, random_state=71)
```

```
# Training  support vector classifier
SV.fit(x_train,y_train)
```

```
SVC(kernel='linear')
```

```
# Training logistic regression classifier
LOGR.fit(x_train,y_train)
```

```
LogisticRegression(solver='liblinear')
```

```
# Training KNN  classifier
KNN.fit(x_train,y_train)
```

```
KNeighborsClassifier()
```

```
from sklearn.metrics import accuracy_score
```

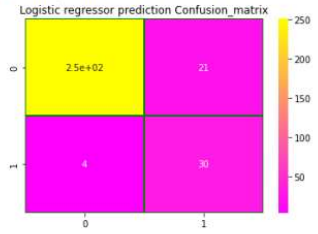
Logistic Regression Model

```
#Report  from Logistic  regression
y_pred_log= LOGR.predict(x_test)
print('this is  the accuracy_score', accuracy_score(y_pred_log,y_test))
print('this is  the confusion_matrix:', confusion_matrix(y_pred_log,y_test))
print('this is  the  classification_report', classification_report(y_pred_log,y_test))
```

```
this is  the accuracy_score 0.9183006535947712
this is  the confusion_matrix: [[251  21]
 [  4  30]]
this is  the  classification_report              precision    recall  f1-score   support

           0       0.98      0.92      0.95       272
           1       0.59      0.88      0.71        34

    accuracy                           0.92       306
   macro avg       0.79      0.90      0.83       306
weighted avg       0.94      0.92      0.93       306
```

```
sns.heatmap(confusion_matrix(y_pred_log,y_test), annot=True, cmap='spring',linewidth=0.1, linecolor='green')
plt.title("Logistic regressor prediction Confusion_matrix")
print('accuracy_score is :' , 0.9183006535947712)
```

```
accuracy_score is : 0.9183006535947712
```



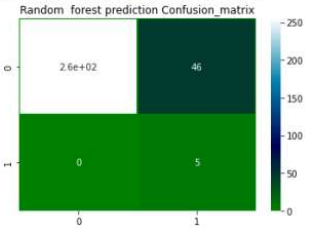
Random Forest Classifier Model

```
RFCpred = RFC.predict(x_test)
print('this is  the accuracy_score', accuracy_score(RFCpred,y_test))
print('this is  the confusion_matrix:', confusion_matrix(RFCpred,y_test))
print('this is  the  classification_report', classification_report(RFCpred,y_test))
```

```
this is  the accuracy_score 0.8496732026143791
this is  the confusion_matrix: [[255  46]
 [  0   5]]
this is  the  classification_report              precision    recall  f1-score   support

           0       1.00      0.85      0.92       301
           1       0.10      1.00      0.18         5

    accuracy                           0.85       306
   macro avg       0.55      0.92      0.55       306
weighted avg       0.99      0.85      0.91       306
```

```
sns.heatmap(confusion_matrix(RFCpred,y_test), annot=True, cmap='ocean',linewidth=0.1, linecolor='green')
plt.title("Random  forest prediction Confusion_matrix")
print('accuracy_score is :' , 0.8496732026143791)
```

```
accuracy_score is : 0.8496732026143791
```



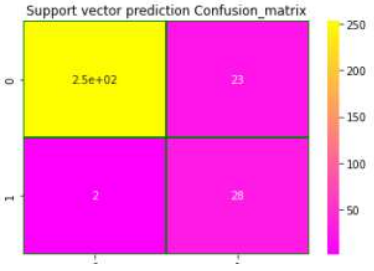
Support vector machine Classifier Model

```
SVpred = SV.predict(x_test)
print('this is  the accuracy_score', accuracy_score(SVpred,y_test))
print('this is  the confusion_matrix:', confusion_matrix(SVpred,y_test))
print('this is  the  classification_report', classification_report(SVpred,y_test))
```

```
this is  the accuracy_score 0.9183006535947712
this is  the confusion_matrix: [[253  23]
 [  2  28]]
this is  the  classification_report              precision    recall  f1-score   support

           0       0.99      0.92      0.95       276
           1       0.55      0.93      0.69        30

    accuracy                           0.92       306
   macro avg       0.77      0.93      0.82       306
weighted avg       0.95      0.92      0.93       306
```

```
sns.heatmap(confusion_matrix(SVpred,y_test), annot=True, cmap='spring',linewidth=0.1, linecolor='green')
plt.title("Support vector prediction Confusion_matrix")
print('accuracy_score is :' ,0.9183006535947712)
```

```
accuracy_score is : 0.9183006535947712
```



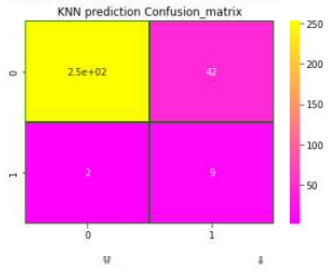
K Nearest Neighbors Classifier Model

In [115]:
```
KNNpred = KNN.predict(x_test)
print('this is  the accuracy_score', accuracy_score(KNNpred,y_test))
print('this is  the confusion_matrix:', confusion_matrix(KNNpred,y_test))
print('this is  the  classification_report', classification_report(KNNpred,y_test))
```

```
this is  the accuracy_score 0.8562091503267973
this is  the confusion_matrix: [[253  42]
 [  2   9]]
this is  the  classification_report              precision    recall  f1-score   support

           0       0.99      0.86      0.92       295
           1       0.18      0.82      0.29        11

    accuracy                           0.86       306
   macro avg       0.58      0.84      0.61       306
weighted avg       0.96      0.86      0.90       306
```

In [116]:
```
sns.heatmap(confusion_matrix(KNNpred,y_test), annot=True, cmap='spring',linewidth=0.1, linecolor='green')
plt.title("KNN prediction Confusion_matrix")
print('accuracy_score is :' ,0.8562091503267973)
```

```
accuracy_score is : 0.8562091503267973
```



Model Cross Validation

K-Fold CV

In [121]:
```
from sklearn.model_selection import cross_val_score as cvs
```

Logistic Regression

In [122]:
```
print(cvs(LOGR,scaled_x_best,y,cv=9).mean())
```

```
0.8831727412372573
```

Random Forest Classifier

In [123]:
```
print(cvs(RFC,scaled_x_best,y,cv=9).mean())
```

```
0.8449890611180934
```

SVM Classifier

In [124]:
```
print(cvs(SV,scaled_x_best,y,cv=9).mean())
```

```
0.8824652050458502
```

K Nearest Neighbours Classifier

In [125]:
```
print(cvs(KNN,scaled_x_best,y,cv=9).mean())
```

```
0.8392217101894521
```

Based on comparing Accuracy Score results with Cross Validation results, it is determined that Logistic regression and the SVM model are the best models a same in the terms of Accuracy .

ROC AUC Scores

Logistic Regression

In [126]:
```
roc_auc_score(y_test,y_pred_log)
```

Out[126]:
```
0.7862745098039217
```

Random Forest Classifier

In [127]:
```
roc_auc_score(y_test,RFCpred)
```

Out[127]:
```
0.5490196078431373
```

SV Classifier

In [128]:
```
roc_auc_score(y_test,SVpred)
```

Out[128]:
```
0.7705882352941177
```

Logistic Regression performing the best and the is followed by the Support Vector Machine

Hyper Parameter Tuning Of Models

136
```
from sklearn.model_selection import GridSearchCV
```

Logistic Regression hypertune

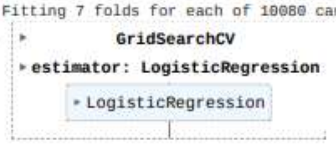
137
```
parameter = {'C':[0.001,0.01,0.1,0,1,1.1],
             'penalty':['l1', 'l2', 'elasticnet', 'none'],
             'dual': [True,False],
             'fit_intercept':[True,False],
             'random_state':[1,2,3,5,10,20,45],
             'solver':['liblinear'],
             'max_iter':[100,200,300,400,500],
             'multi_class':['auto', 'ovr', 'multinomial']}
```

138
```
GridCV = GridSearchCV(LogisticRegression(),parameter,cv=7,n_jobs = -1,verbose=2)
```

139
```
GridCV.fit(x_train,y_train)
```

```
Fitting 7 folds for each of 10080 candidates, totalling 70560 fits
```

139]:



140
```
GridCV.best_params_
```

140]:
```
{'C': 0.1,
 'dual': True,
 'fit_intercept': True,
 'max_iter': 100,
 'multi_class': 'auto',
 'penalty': 'l2',
 'random_state': 1,
 'solver': 'liblinear'}
```

141
```
Best_mod1 = LogisticRegression(C = 0.1,dual = True,fit_intercept = True,max_iter = 100,multi_class = 'auto', penalty = '
Best_mod1.fit(x_train,y_train)
lrpred = Best_mod1.predict(x_test)
acc = accuracy_score(y_test,lrpred)
print(acc*100)
```

```
91.83006535947712
```

As logistic regression gave us the better validation score

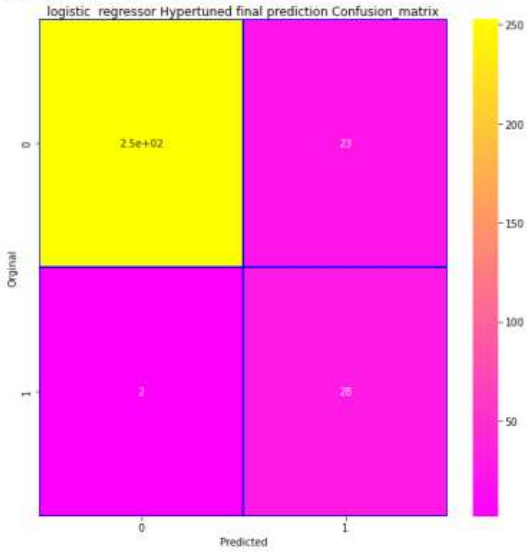
also gave better AUC score than all

Thus we are hypertuning Logistic only for the getting good predtion results

Logistic Regression has an accuracy score of 91%

13
```
plt.figure(figsize=(9,9))
sns.heatmap(confusion_matrix(lrpred,y_test), annot=True, cmap='spring',linewidth=0.1, linecolor='blue')
plt.title("logistic  regressor Hypertuned final prediction Confusion_matrix",pad=True)
plt.xlabel('Predicted')
plt.ylabel('Orginal')
print('accuracy_score is :' , 91.83006535947712)
```

```
accuracy_score is : 91.83006535947712
```



Saving The Model

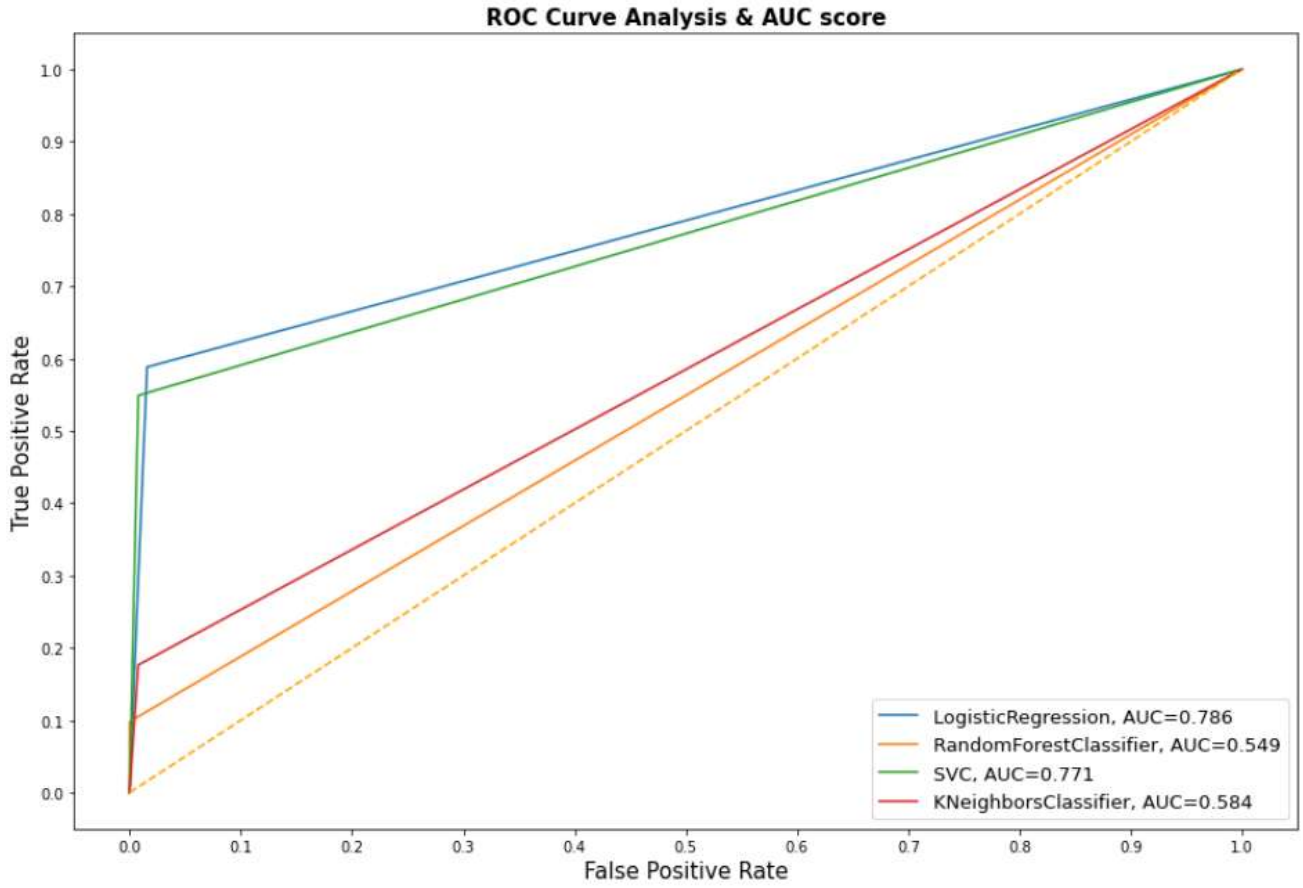
14
```
import joblib
joblib.dump(Best_mod1,"BestModelHR_analytics.pkl")
```

14]:
```
['BestModelHR_analytics.pkl']
```

Best Accuracy is: 0.9183006535947712 on random_state: 71

As logistic regression gave us the better validation score also gave better AUC score than all Thus we are hypertuning Logistic only for the getting good prediction results

Logistic Regression has an accuracy score of 91%



Logistic Regression performing the best and the is followed by the Support Vector Machine

Loading The Model

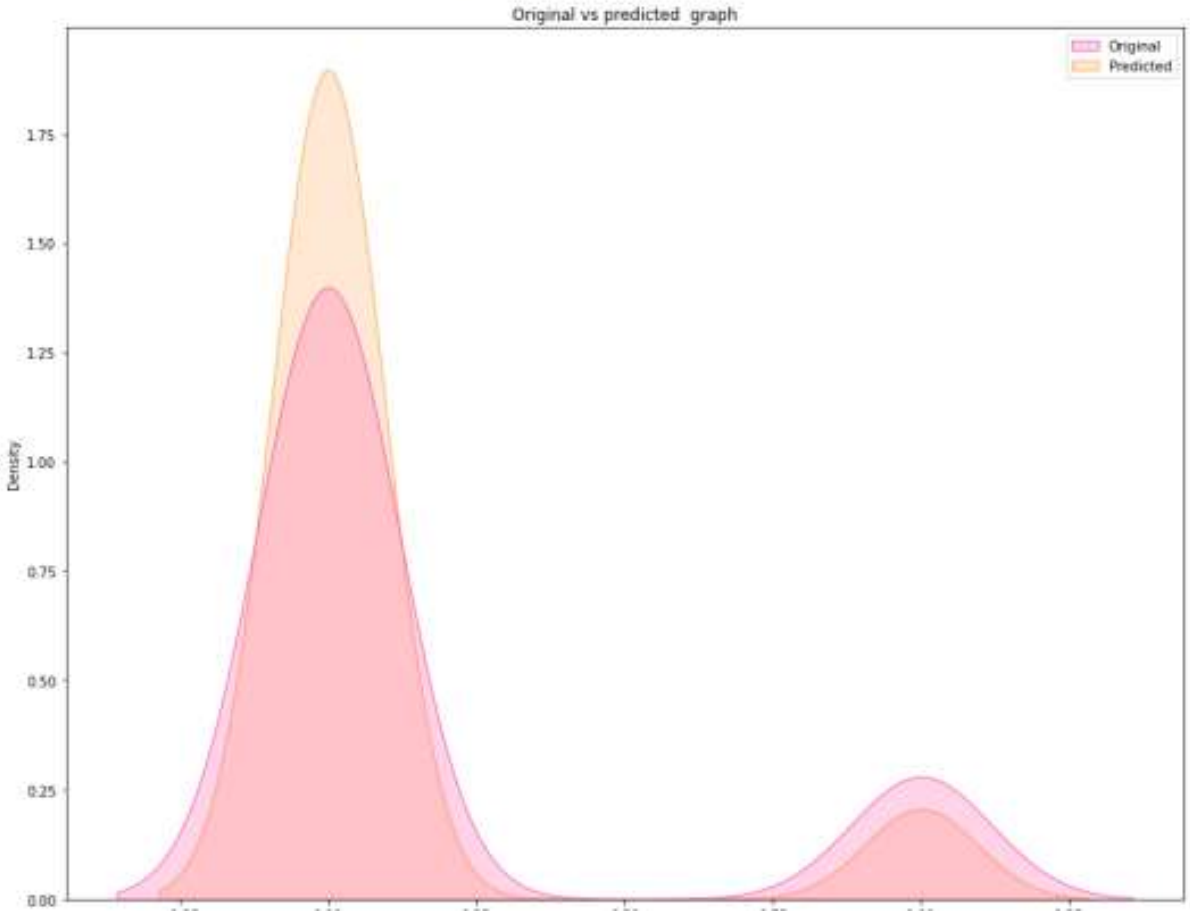
```
import numpy as np
a = np.array(y_test)
predicted = np.array(Best_mod1.predict(x_test))
Analytics_model = pd.DataFrame({"Original":a,"Predicted":predicted},index= range(len(a)))
Analytics_model
```

	Original	Predicted
0	0	0
1	0	0
2	1	0
3	0	0
4	0	0
...	...	...
301	0	0
302	0	0
303	0	0
304	0	0
305	0	0

306 rows × 2 columns

```
plt.figure(figsize=(15,12))
sns.kdeplot(data=Analytics_model, palette='spring',gridsize=900, shade=True)
plt.title('Original vs predicted  graph')
```

```
Text(0.5, 1.0, 'Original vs predicted  graph')
```



6. Concluding Remarks/Key Findings.

- Feature column contains mixed type of data i.e. continuous and categorical. Target/label column contains categorical data
- There are no null values in the dataset
- Target column have 2 unique values "Yes", "No"
- There are 3 columns with one value present in them i.e. employee count, over 18, standard hours
- In most columns mean and 50% are similar in value, and mean is greater than standard deviation, indicating a symmetric distribution.
- TotalWorkingYears,YearsAtCompany,YearsInCurrentRole,YearsSinceLastPromotion,YearsWithCurrManager have a big difference between 75% and max indicating the presence of outliers. as the std deviation is high so there are chances that - the dataset has the skewness present in it
- 2 unique categorical values in the Label column / target variable.
- Class- 'No' : Has 1233 values 'Yes' : Has 237 values
- There is class imbalance in the Target Data
- Class 'No' = 83.88% of total values 'Yes'= 16.12% of total values
- Classes are imbalanced
- Female employees are much less than Male employees
- Most employees Travel Rarely.
- Most employees belong to Research and Development.
- Life Sciences is the most common Education field among this dataset.
- Highest number of Employees are Sales Executives, followed by Research Scientist and Laboratory Technicians.
- Maximum number of employees are married.
- Maximum employees don't over time
- Ages under 35 are contributing the most to Attrition.
- High number of people who are 35 and above have been retained.
- In terms of DailyRate, there isn't much of a difference in number between those who left and those who did not, however attrition is higher amongst those with lower DailyRate
- Distance from home greater than 10 is a deciding factor for Attrition
- HourlyRate doesn't seem to contribute to Attrition.
- Those employees whose MonthlyIncome is below 5000 are contributing to Attrition
- MonthlyRate doesn't have a big effect on Attrition
- Employees having Total working years under 9 are mostly contributing to Attrition
- Employees having worked 6 years at the company or less have more impact on Attrition.
- Employees having gone up to 2 years since last promotion have higher Attrition.
- Employees having worked under current manager for 3 years and less have high Attrition.
- Employees with Education '3' and '4' are retained the most.
- Employees with Education '3' mostly leave.
- Employees with EducationFields Life Sciences and Medical are retained the most.
- Employees with EducationFields Life Sciences mostly leave.
- Employees who Travel rarely are retained the most.
- Employees belonging to Research and Development are retained the most followed by Sales. While Employees belonging to Research and Development also contribute the most to Attrition.
- The number of male employees retained is a bit greater than the number of female employees retained. Male employees also contribute the most to Attrition.
- Employees with greater Environment Satisfaction are retained. While Employees with lowest Environment Satisfaction contribute the most to Attrition.
- Employees with Job Involvement of 3 are retained the most.

- Employees with Job level 3 and 2 are retained the most. While employees with job Level 1 contribute the most to Attrition.
- Sales Executives,Research Scientists, Laboratory Technicians are amongst the most retained employees. Sales Executives and Laboratory Technicians also contribute more to attrition than Research Scientists.
- Employees with Highest Job Satisfaction of 4 and 3 are amongst the most retained employees.
- More Married Employees are retained than single and divorced.
- Employees who don't do overtime are retained the most.
- Employees who have worked in 1 and 0 companies before are retained more than those with work history in multiple companies. While employees who have worked in 1 company before are the most to leave.
- Employees with lowest Salary percentage hike contribute most to Attrition.
- Employees with Highest Relationship Satisfaction of 3 and 4 are those who are retained the most.
- Employees with Performance Rating 3 are retained a lot more than those with Performance Rating 4
- Employees with Training Times of 2 and 3 Last Year are retained the most.
- Employees with StockOption level 1 are retained the most. While Employees with stock optoion level 0 contribute most to Attrition.
- Employees with worklife balance of 3 are retained the most.

- OverTime_Yes,Marital_status_Single,JobRole_Laboratory_Technician,BusinessTravel_Travel_Frequently,Department_Sales,DistanceFromHome,EducationField have the highest positive correlation with Attrition.

- OverTime_No,TotalWorkingYears,JobLevel,YearsAtCompany,YearWithCurrManager,Age,StockOptionLevel,JobInvolvement,JobSatisfaction have the highest negative correlation with Attrition.

- we can see that in this dataset we have more of inverse correlation .
- Based on comparing Accuracy Score results with Cross Validation results, it is determined that Logistic regression and the SVM model are the best models as both are performing the same in the terms of Accuracy .
- Logistic Regression performing the best and the is followed by the Support Vector Machine
- As logistic regression gave us the better validation score
- also gave better AUC score than all
- Thus, we are hypertuning Logistic only for the getting good prediction results
- Logistic Regression has an accuracy score of 91%.