Attrition:- ***company losing its customer base***

Attrition is a process in which the workforce dwindles at a company, following a period in which a number of people retire or resign, and are not replaced.

- A reduction in staff due to attrition is often called a hiring freeze and is seen as a less disruptive way to trim the workforce and reduce payroll than layoffs
- In this NoteBook our Aim will be to analyze the datasets completely wrt each and feature and find the reasin behind Attrition of Employees.
- And what the top factors which lead to employee attrition?

Description about the data

- Age: A period of employee life, measured by years from birth.
- Attrition: The departure of employees from the organization.
- BusinessTravel: Did the employee travel on a business trip or not.
- DailyRate: Employee salary for the period is divided by the amount of calendar days in the period.
- Department: In which department the Employee working.
- DistanceFromHome: How far the Employee live from the office location.
- Education: In education 1 means 'Below College', 2 means 'College', 3 means 'Bachelor', 4 means 'Master', 5 means 'Doctor'
- EducationField: In which field Employee complete his education.
- EmployeeCount: How many employee working in a department
- EmployeeNumber: An Employee Number is a unique number that has been assigned to each current and former State employee and elected official in the Position and Personnel DataBase (PPDB).
- Job involvement: Is the degree to which an employee identifies with their work and actively participates in it where 1 means 'Low', 2 means 'Medium', 3 means 'High', 4 means 'Very High'
- JobLevel: Job levels, also known as job grades and classifications, set the responsibility level and expectations of roles at your organization. They may be further defined by impact, seniority, knowledge, skills, or job title, and are often associated with a pay band. The way you structure your job levels should be dictated by the needs of your unique organization and teams.
- JobRole: What is the jobrole of an employee.
- JobSatisfaction: Employee job satisfaction rate where, 1 means 'Low', 2 means 'Medium', 3 means 'High', 4 means 'Very High'
- MaritalStatus: Marital status of the employee.
- MonthlyIncome: total monetary value paid by the organization to an employee.
- MonthlyRate: The per-day wage of the employee.
- NumCompaniesWorked: Before joining this organization how many organizations employee worked.
- Over18: Is the employee age over than 18 or not.
- OverTime: A Employee works more than 9 hours in any day or for more than 48 hours in any week.
- PercentSalaryHike:
- PerformanceRating 1 'Low' 2 'Good' 3 'Excellent' 4 'Outstanding'
- EnvironmentSatisfaction 1 'Low' 2 'Medium' 3 'High' 4 'Very High'
- RelationshipSatisfaction 1 'Low' 2 'Medium' 3 'High' 4 'Very High'
- StandardHours: Is the number of hours of production time that should have been used during an working period.
- StockOptionLevel: Employee stock options, also known as ESOs, are stock options in the company’s stock granted by an employer to certain employees. Typically they are granted to those in management or officer-level positions. Stock

options give the employee the right to buy a certain amount of stock at a specific price, during a specific period of time. Options typically have expiration dates as well, by which the options must have been exercised, otherwise they will become worthless.

- TotalWorkingYears: Total years the employee working in any organization
- TrainingTimesLastYear: Last year how many times employee took training session.
- WorkLifeBalance 1 'Bad' 2 'Good' 3 'Better' 4 'Best'
- YearsAtCompany: How many years the employee working in the current organization
- YearsInCurrentRole: How many years the employee working in the current position
- YearsSinceLastPromotion: How many years the employee working in the current position after promotion
- YearsWithCurrManager: How many years the employee working under the current manager

Some Python Libraries

In the first place, Let's define some libraries to help us in the manipulation the data set, such as `pandas`, `numpy`, `matplotlib`, `seaborn`. In this tutorial, we are implementing a Logistic Regression with `sikit-learn`. The goal here is to be as simple as possible! So to help you with this task, we implementing the Logistic regression using ready-made libraries and their functinality.

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

Get the Data

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

Basic Data Exploration

- This is an Important Step in Data Science and Machine Learning to ensure about the columns, and rows present.
- First, we will check the shape of the dataset
- Second, we will check the head, tail, and sample of the datasets
- Third, we will check the Data Description
- Then, we will check the Data Types of the columns present in the data.

```
data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Employee
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	

```
pd.set_option('display.max_columns',None)
```

```
data.head()
```


	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Employee
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	


```
data.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
```

```
data.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfa
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.0(
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.72
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.09
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.0(
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.0(
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.0(
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.0(
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.0(

Observations

- we only have int and string data types features. there is no feature with float. 26 features are numerical and 9 features are categorical
- Attrition in out target value which has no missing value. But, the quantity of data of emp having Attrition is less compared to employees whoch do not have Attrition.
- It's very good that we are having a complete dataset, there is no any missing values in dataset.

Check Duplicates

```
print(data.duplicated().value_counts())
data.drop_duplicates(inplace = True)
print(len(data))
```

```
False    1470
Name: count, dtype: int64
1470
```

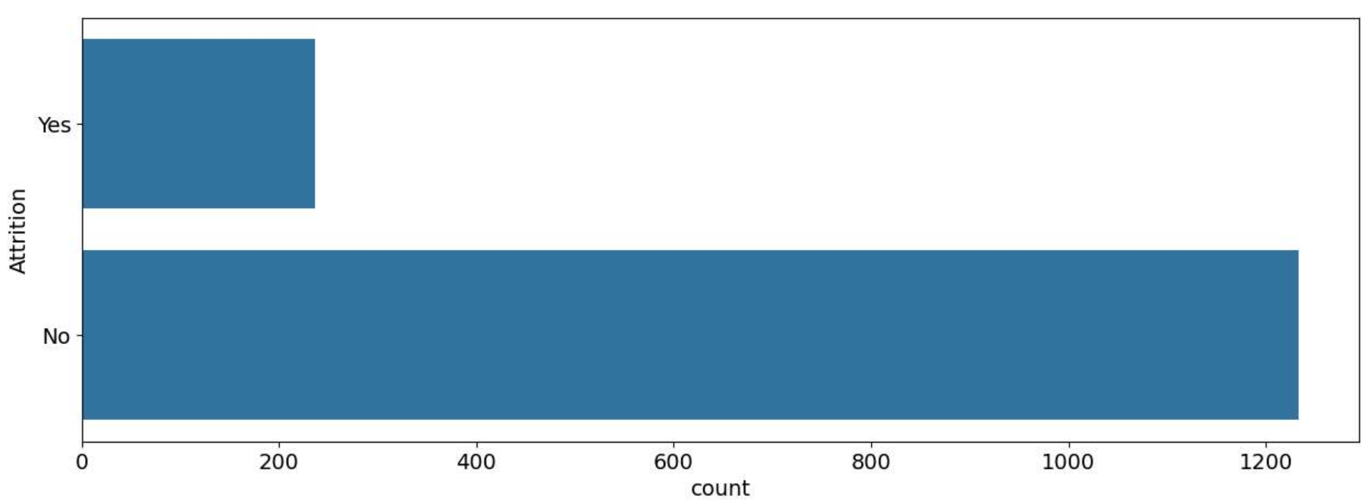
Checking missing value

```
data.isnull().sum()
```

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

Target Variable

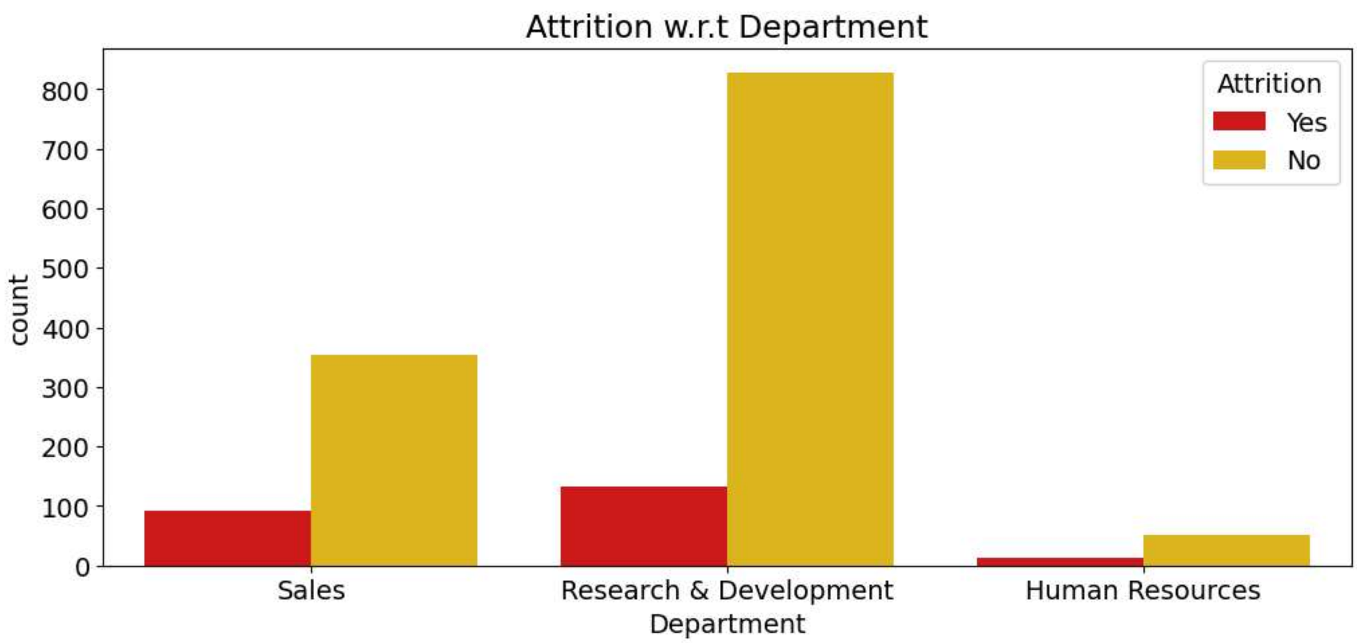
```
plt.figure(figsize=(15,5))
plt.rc("font", size=14)
sns.countplot(y ='Attrition',data=data)
plt.show()
```



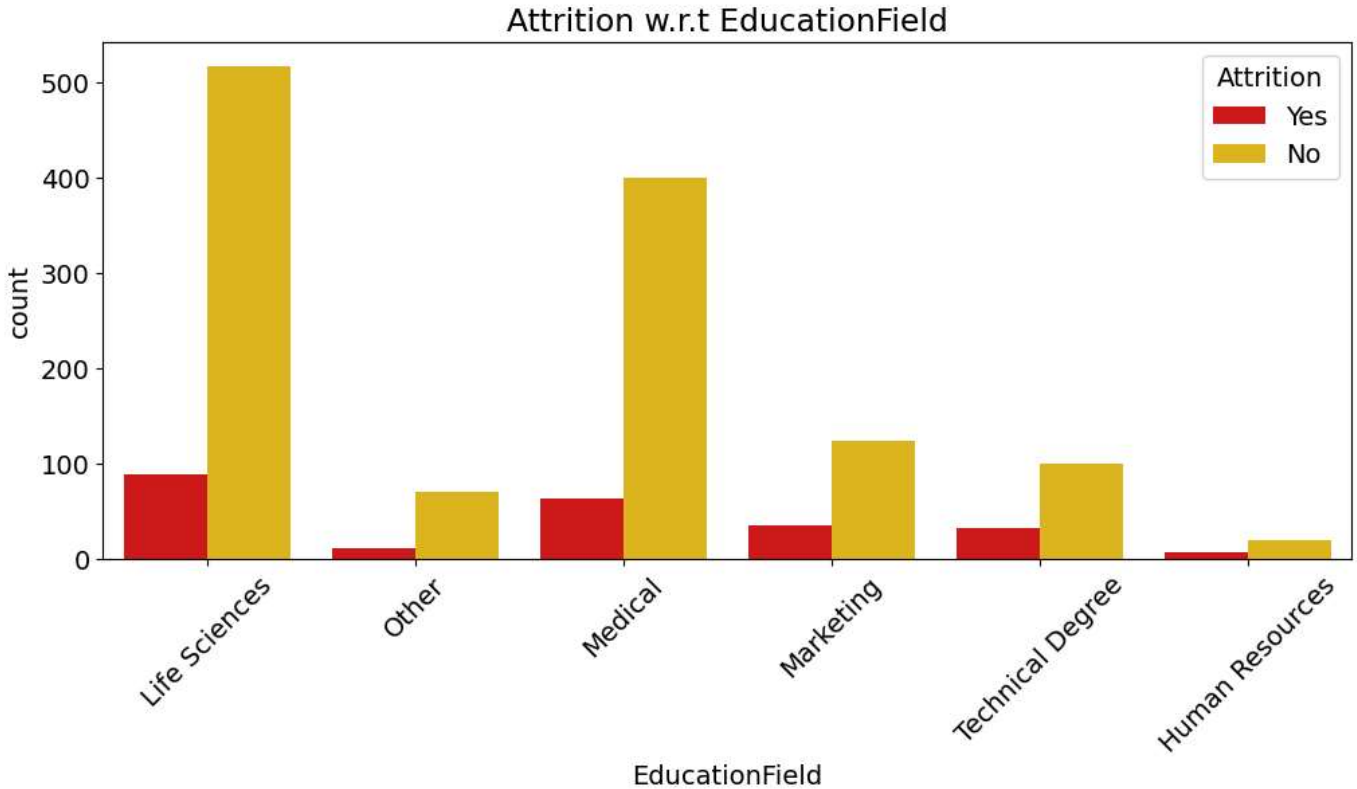
Over here we noticed that the Target column is Highly Imbalanced, we need to balance the data by using some Statistical Methods.

Exploratory Data Analysis

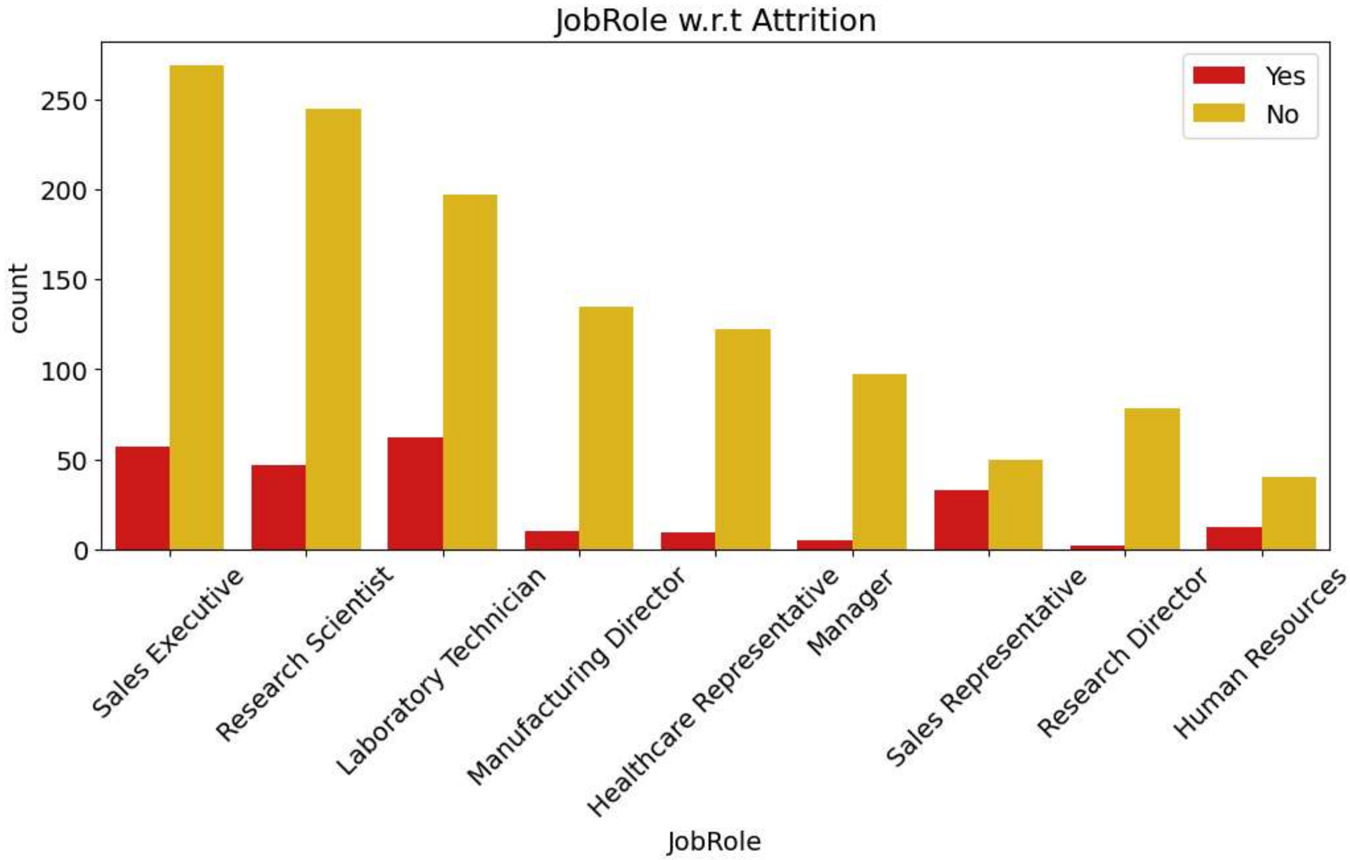
```
# Department wrt Attrition
plt.figure(figsize=(12,5))
sns.countplot(x='Department',hue='Attrition', data=data, palette='hot')
plt.title("Attrition w.r.t Department")
plt.show()
```



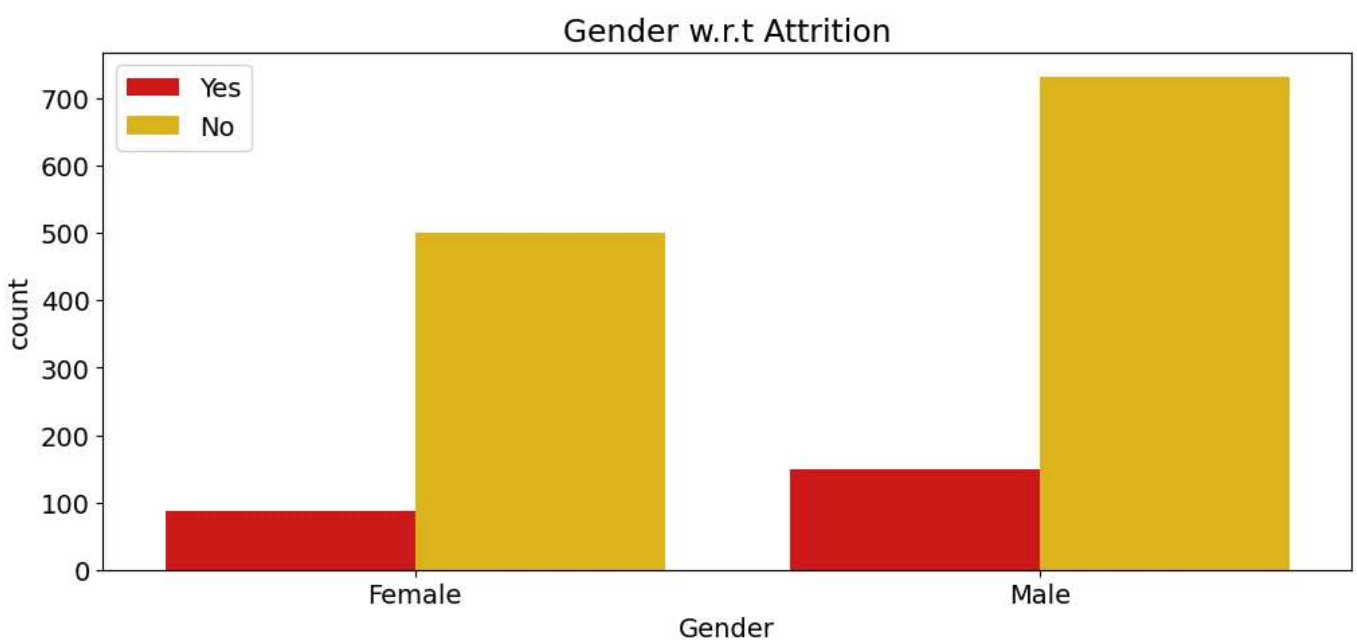

```
# Department wrt Attrition
plt.figure(figsize=(12,5))
sns.countplot(x='EducationField',hue='Attrition', data=data, palette='hot')
plt.title("Attrition w.r.t EducationField")
plt.xticks(rotation=45)
plt.show()
```




```
# let's see at which post most people are leaving the jobs
# JobRole
plt.figure(figsize=(12,5))
sns.countplot(x='JobRole',hue='Attrition', data=data, palette='hot')
plt.title("JobRole w.r.t Attrition")
plt.legend(loc='best')
plt.xticks(rotation=45)
plt.show()
```



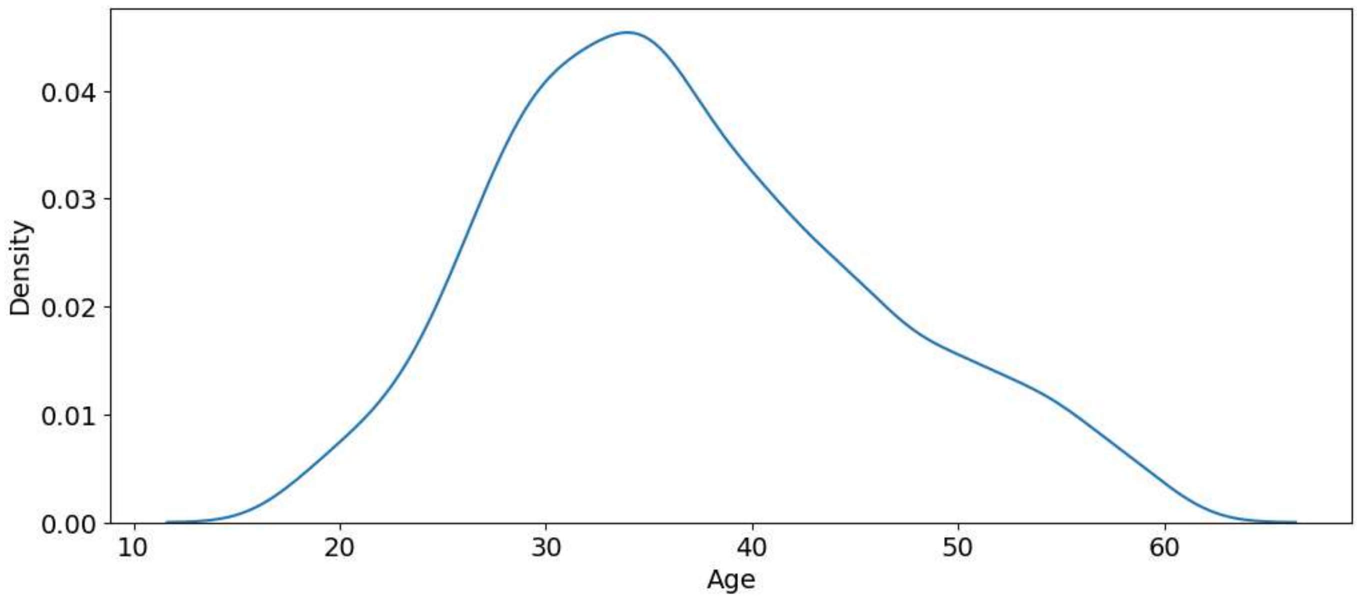

```
# most male of female employes Attriate
# Department wrt Attrition
plt.figure(figsize=(12,5))
sns.countplot(x='Gender',hue='Attrition', data=data, palette='hot')
plt.title("Gender w.r.t Attrition")
plt.legend(loc='best')
plt.show()
```



OBSERVATIONS

- Employees working in R&D department are more, but employees from sales department or at position like sales executive,sale Representative leaves the job early.
- Males are more under Attrition then Females

```
# distribution of age
plt.figure(figsize=(12,5))
sns.distplot(data['Age'],hist=False)
plt.show()
```

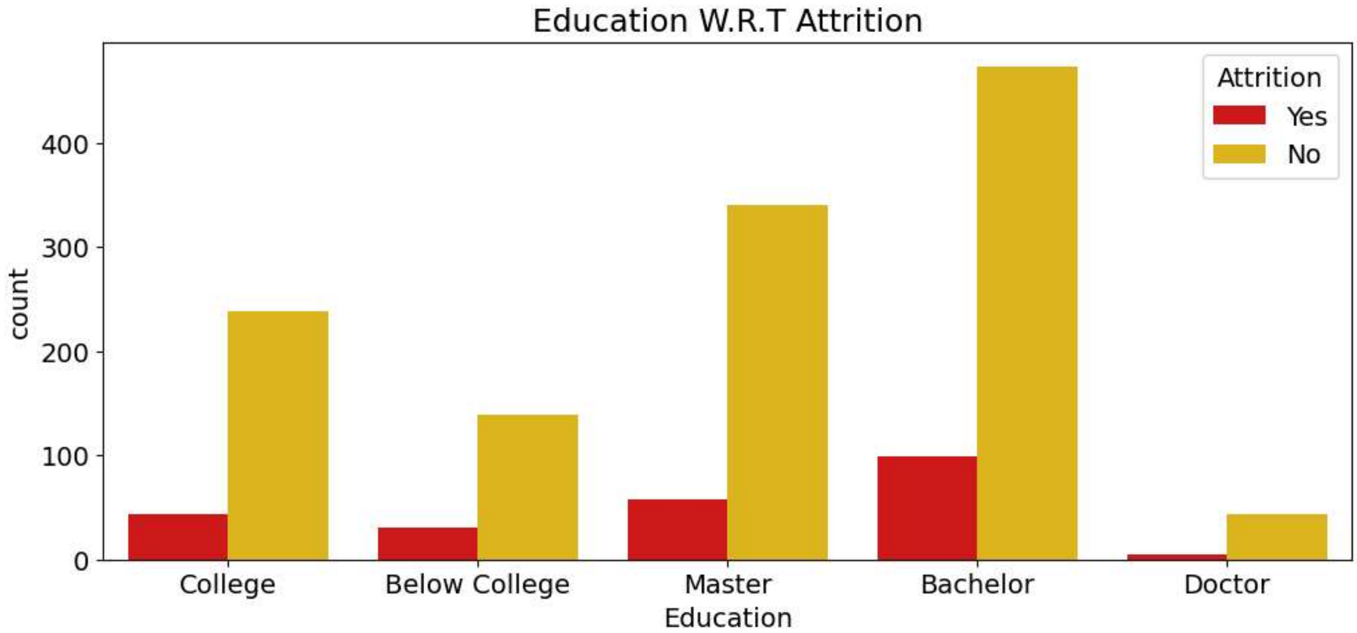


- Age column is very well normalized, most of employees are age between 25 to 40.
- we are having some of the numerical columns which are lebel encoded for us, they are ordinal labels, so let's have a look at them first

```
ordinal_features = ['Education','EnvironmentSatisfaction','JobInvolvement','JobSatisfaction',
                    'PerformanceRating','RelationshipSatisfaction','WorkLifeBalance']
data[ordinal_features].head()
```

	Education	EnvironmentSatisfaction	JobInvolvement	JobSatisfaction	PerformanceRating	RelationshipSatisfact
0	2	2	3	4	3	
1	1	3	2	2	4	
2	2	4	2	3	3	
3	4	4	3	3	3	
4	1	1	3	2	3	

```
edu_map = {1 :'Below College', 2: 'College', 3 :'Bachelor', 4 :'Master', 5: 'Doctor'}
plt.figure(figsize=(12,5))
sns.countplot(x=data['Education'].map(edu_map), hue='Attrition', data=data, palette='hot')
plt.title("Education W.R.T Attrition")
plt.show()
```



OBSERVATIONS

- Employees from Bachelor are more, then from Masters background. Attrition wrt to bachelor can be seem more because they have more and more expectation from companies and it will be interesting to see the reason behind this in this dataset.

Label Encodeing

In machine learning, we usually deal with datasets that contain multiple labels in one or more than one columns. These labels can be in the form of words or numbers. To make the data understandable or in human-readable form, the training data is often labelled in words.

```
# Target Variable(Attrition)
data['Attrition'] = data['Attrition'].replace({'No':0,'Yes':1})


#encode binary variables
data['OverTime'] = data['OverTime'].map({'No':0,'Yes':1})
data['Gender'] = data['Gender'].map({'Male':0,'Female':1})


# encode categorical columns which are ordinal, use labelEncoding
# apply Label encoder to df_categorical
from sklearn.preprocessing import LabelEncoder
encoding_cols=['BusinessTravel','Department','EducationField','JobRole','MaritalStatus']
label_encoders = {}
for column in encoding_cols:
    label_encoders[column] = LabelEncoder()
    data[column] = label_encoders[column].fit_transform(data[column])


# look at the final data
data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCo
0	41	1	2	1102	2	1	2	1	
1	49	0	1	279	1	8	1	1	
2	37	1	2	1373	1	2	2	4	
3	33	0	1	1392	1	3	4	1	
4	27	0	2	591	1	2	1	3	

Machine Learning: Splitting the data into Training and Testing sample

We dont use the full data for creating the model. Some data is randomly selected and kept aside for checking how good the model is. This is known as Testing Data and the remaining data is called Training data on which the model is built. Typically 70% of data is used as Training data and the rest 30% is used as Tesing data.

```
X = data.drop(['Attrition','Over18'], axis=1)
y = data['Attrition'].values
```

Resampling

Resampling is the method that consists of drawing repeated samples from the original data samples. The method of Resampling is a nonparametric method of statistical inference Oversampling and undersampling in data analysis are techniques used to adjust the class distribution of a data set. These terms are used both in statistical sampling, survey design methodology and in machine learning. Oversampling and undersampling are opposite and roughly equivalent techniques

- We are going to use Over Sampling.
- We will not use Under Sampling to avoid data loss.

```
from collections import Counter
from imblearn.over_sampling import RandomOverSampler
print(Counter(y))
rus = RandomOverSampler(random_state = 42)
X_over, y_over = rus.fit_resample(X,y)
print(Counter(y_over))
```

```
Counter({0: 1233, 1: 237})
Counter({1: 1233, 0: 1233})
```

```
# Split the data into training and testing set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_over, y_over, test_size=0.2, random_state=42)
```

```
# Sanity check for the sampled data
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(1972, 33)
(1972,)
(494, 33)
(494,)
```

Logistic Regression in Machine Learning

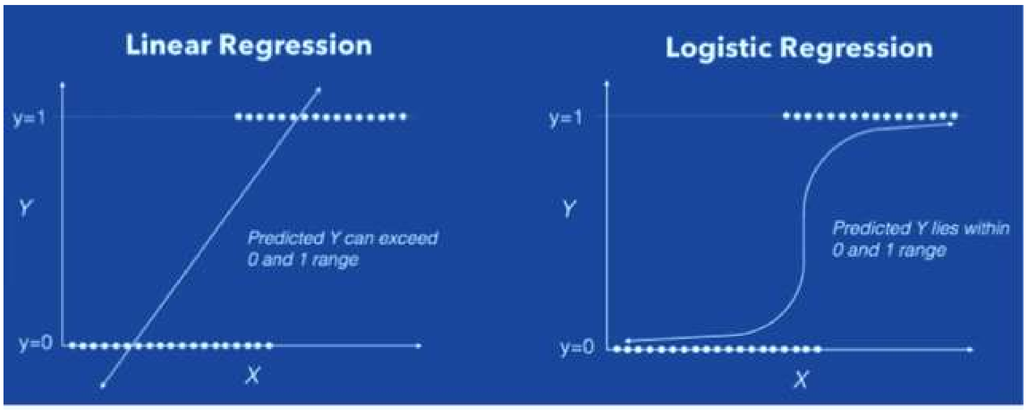
Logistic Regression is used for predicting a category, specially the Binary categories(Yes/No , 0/1).

For example, whether to approve a loan or not (Yes/No)? Which group does this customer belong to (Silver/Gold/Platinum)? etc.

When there are only two outcomes in Target Variable it is known as Binomial Logistic Regression.

If there are more than two outcomes in Target Variable it is known as Multinomial Logistic Regression.

If the outcomes in Target Variable are ordinal and there is a natural ordering in the values (eg. Small< Medium< Large) then it is known as Ordinal Logistic Regression.



$$y = \beta_0 + \beta_1 * (P1) + \beta_2 * (P2) + \beta_3 * (P3) \ldots$$

$$\mathbf{P(1) = e^{\wedge}y / (1 + e^{\wedge}y)}$$

Logistic regression is based on logit function logit(x) = log(x / (1 − x))

The output is a value between 0 to 1. It is the probability of an event's occurrence.

E.g. There is an 80% chance that the loan application is good, approve it.

The coefficients β0, β1, β2, β3… are found using Maximum Likelihood Estimation Technique. Basically, if the Target Variable's value (y) is 1, then the probability of one "P(1)" should be as close to 1 as possible and the probability of zero "P(0)" should be as close to 0 as possible. Find those coefficients which satisfy both the conditions.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, roc_curve, roc_auc_score
```

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
prediction=logreg.predict(X_test)
cnf_matrix = confusion_matrix(y_test,prediction)
print("Accuracy Score -", accuracy_score(y_test , prediction))
```

```
Accuracy Score - 0.6417004048582996
```