# HR Analytics Project

## Problem definition

Employee turn-over is a costly problem for companies. 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. How does Attrition affect companies? and how does HR Analytics help in analyzing attrition.

## 1.Data Analysis

In this case study, a HR dataset was sourced from IBM HR Analytics Employee Attrition & Performance which contains employee data for 1,470 employees with various information about the employees.

### 1.1 Import library and dataset

First step is to import library such as pandas, numpy, seaborn and matplotlib. We need to create CSV file for train data and test data.

Import train dataset as per the below steps.

```
In [3]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import warnings
        warnings.filterwarnings('ignore')
```

```
In [4]: df=pd.read_csv('hr.csv')
```

```
In [5]: df
```

**1. df.shape**: Represent the number of rows and column in the dataset. In our dataset we have 1470 rows and 35 columns.

**1.3 df.column**: Represent the names of the columns present. We have 34 columns.

```
df.columns
```

```
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')
```

**1.4 df.info**: Represent the data type of each column.

```
df.info()
```

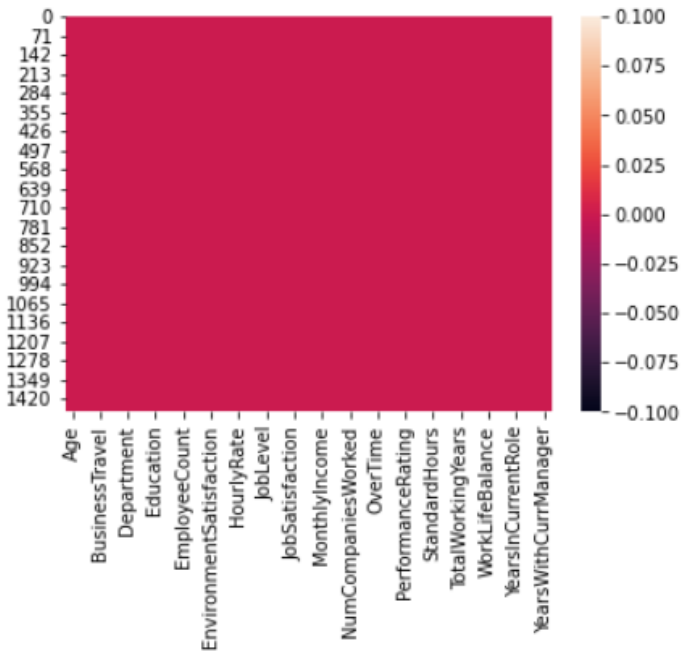
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 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
```

## 2. Exploratory Data Analysis

### 2.1 Check null values

For the given dataset null value present needs to be checked and sum of the null value in each column will be found. Less number of values missed, we can drop those rows. More number of values missed then we can replace numeric value with mean and characters with mode.

sns.heatmap (df.is null()) is used for visual representation of null values. In our dataset, we do not have any null values.




## 2.2 Statistics summary

This can be found by df.describe(), where we get to find the mean, median, count, standard deviation and other statistics information. Our dataset has 34 columns part of it shown below.

```
df.describe()
```

| | Age | Attrition | BusinessTravel | DailyRate | Department | DistanceFromHome | Education | EducationField | EmployeeCount | EmployeeNumbe |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.0 | 1470.00000( |
| **mean** | 36.923810 | 0.161224 | 1.607483 | 802.485714 | 1.260544 | 9.192517 | 2.912925 | 2.247619 | 1.0 | 1024.86530( |
| **std** | 9.135373 | 0.367863 | 0.665455 | 403.509100 | 0.527792 | 8.106864 | 1.024165 | 1.331369 | 0.0 | 602.02433 |
| **min** | 18.000000 | 0.000000 | 0.000000 | 102.000000 | 0.000000 | 1.000000 | 1.000000 | 0.000000 | 1.0 | 1.00000( |
| **25%** | 30.000000 | 0.000000 | 1.000000 | 465.000000 | 1.000000 | 2.000000 | 2.000000 | 1.000000 | 1.0 | 491.25000( |
| **50%** | 36.000000 | 0.000000 | 2.000000 | 802.000000 | 1.000000 | 7.000000 | 3.000000 | 2.000000 | 1.0 | 1020.50000( |
| **75%** | 43.000000 | 0.000000 | 2.000000 | 1157.000000 | 2.000000 | 14.000000 | 4.000000 | 3.000000 | 1.0 | 1555.75000( |
| **max** | 60.000000 | 1.000000 | 2.000000 | 1499.000000 | 2.000000 | 29.000000 | 5.000000 | 5.000000 | 1.0 | 2068.00000( |

8 rows × 34 columns

From the above table we can infer

- Age distribution is a slightly right-skewed normal distribution with majority between 25 and 45 years old. Min is 18 and Max is 60.
- Employee Count and Standard Hours are constant values for all employees. They're likely to be redundant features.
- Difference between 75% and max is high for monthly income, total working years, years at company, years since last promotion, years with current manager data might have outliers
- Daily rate min 102 and max is 1499

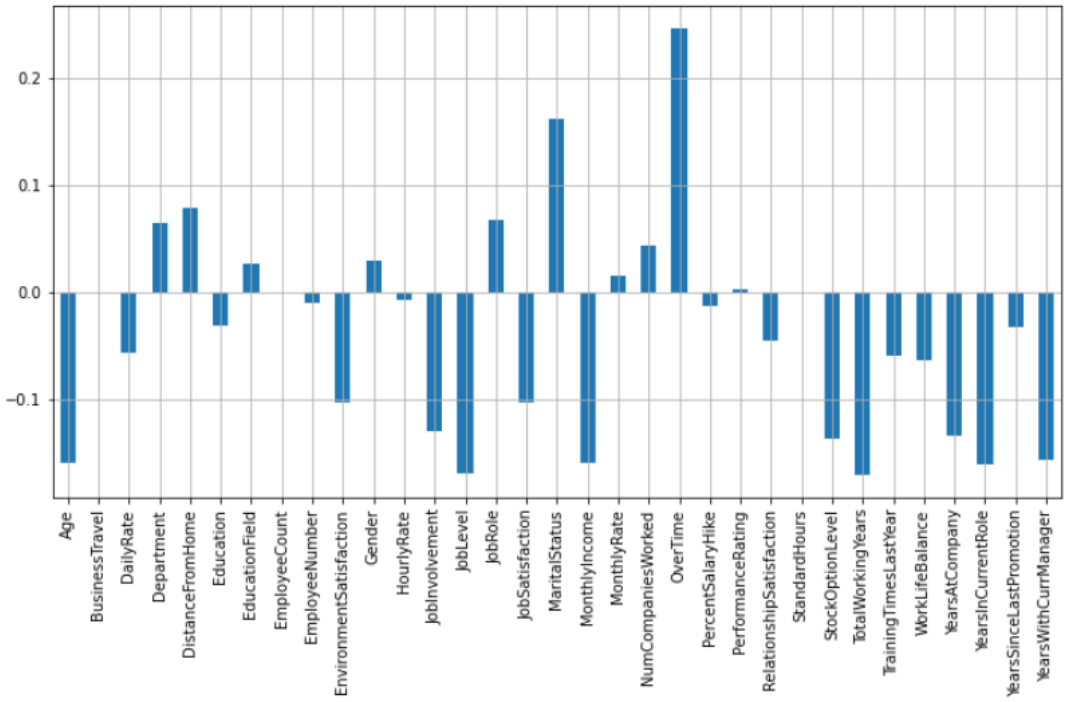
### 2.3 Correlation

Correlation is used to find how each variable is related with the target variable. It will positively, negatively or neutral correlated. Lighter shade towards the top are positively correlated and darker shared towards the end are negatively correlated.

In our dataset we take attrition as target variable. ID column can be dropped as it is not integer type. Correlation and outliers can be done only for integer data. Below charts is used to explain the correlation.

```
In [13]: plt.figure(figsize=(12,6))
         df.drop('Attrition',axis=1).corrwith(df['Attrition']).plot(kind='bar',grid=True)
```

```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x21de5f5f8e0>
```




From the above correlation chart, we can understand

- Overtime, Marital status, job role, distance from home and department are positively correlated with attrition
- Environment satisfaction, job involvement, job level, monthly income, stocking option level, total working years, years in current role and years with current manager are negatively correlated with

### 2.4 Label encoding

Encoding method is used to encode categorical labels with numerical values. There are two main methods One-Hot-Encoding and Label-Encoder. Label Encoding is very simple, and it involves converting each value in a column to a number. In one-Hot Encoder each category value is converted into a new column and assigned a 1 or 0 (notation for true/false) value to the column.

In our dataset, we have 9 columns of categorical data and we use label encoding method to convert to numeric values.

```
print(df["Department"].unique())
print(df["BusinessTravel"].unique())
print(df["EducationField"].unique())
print(df["Gender"].unique())
print(df["Attrition"].unique())
print(df["JobRole"].unique())
print(df["MaritalStatus"].unique())
print(df["OverTime"].unique())
print(df["Attrition"].unique())
```

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

```
#change columns to 0 and 1
from sklearn import preprocessing
le=preprocessing.LabelEncoder()
list1=['Department','BusinessTravel','EducationField','Gender','Attrition','JobRole','MaritalStatus','OverTime',"Attrition"]
for val in list1:
    df[val]=le.fit_transform(df[val].astype(str))
```

### 2.5 Outliers

Outliers are those extreme values which lies beyond 3 and -3. Visualization of outliers can be seen by boxplot where point's lies either above the max or lower the min. Outliers will be removed by Z score method.

In our dataset, Employee count, standard hours and over18 columns can be dropped as the value is same for entire column. we identified 92 rows as outliers and removed. Given dataset have 1470 rows and 35 columns, after removing outliers we have 1378 rows and 32 columns.

```
df=df.drop(["EmployeeCount","StandardHours"],axis=1)
```

```
df=df.drop(["Over18"],axis=1)
```

```
#Remove outliers
from scipy.stats import zscore
z=np.abs(zscore(df))
threshold=3
print(np.where(z>3))
data=df[(z<3).all(axis=1)]
data
```

### 2.6. Skewness

Skewness occurs because data is not of symmetric distribution it would be either towards right as right skewed or towards left as left skewed. We use power transform, boxcox and log method to treat the dataset for highly skewed data.

- skewness is between -0.5- 0.5, the data is symmetrical.
- If the skewness is between -1 -0.5 and 0.5-1 the data are moderately skewed.
- If the skewness is less than -1 or greater than 1 the data are highly skewed.

As per the below skew value, data is not skewed so we do not have to apply any technique.

```
#Skewness
data.skew()
```

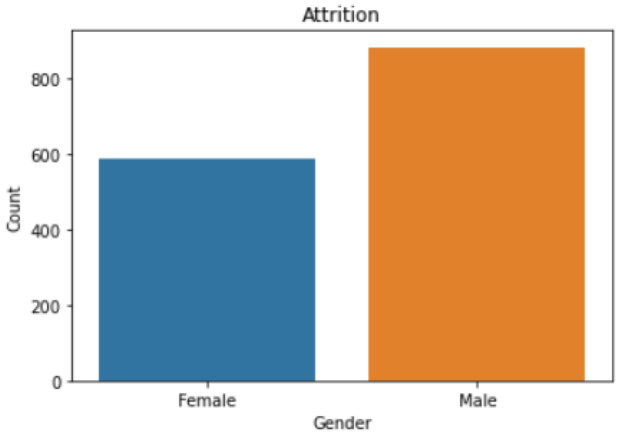
```
Age                         0.472280
Attrition                   1.805983
BusinessTravel             -1.426774
DailyRate                  -0.017078
Department                  0.183919
DistanceFromHome            0.954752
Education                  -0.289024
EducationField              0.544868
EmployeeNumber              0.018931
EnvironmentSatisfaction    -0.325285
Gender                     -0.417296
HourlyRate                 -0.030481
JobInvolvement             -0.501401
JobLevel                    1.126075
JobRole                    -0.386843
JobSatisfaction            -0.345612
MaritalStatus              -0.160952
MonthlyIncome               1.544770
MonthlyRate                 0.030596
NumCompaniesWorked          1.037715
OverTime                    0.954751
PercentSalaryHike           0.800592
PerformanceRating           1.931566
RelationshipSatisfaction   -0.295686
StockOptionLevel            0.962332
TotalWorkingYears           1.034487
TrainingTimesLastYear       0.577614
WorkLifeBalance            -0.557100
YearsAtCompany              1.248623
YearsInCurrentRole          0.726675
YearsSinceLastPromotion     1.756335
YearsWithCurrManager        0.694506
dtype: float64
```

## 3. Univariant and Bivariant analysis

### 3.1 Univariant analysis

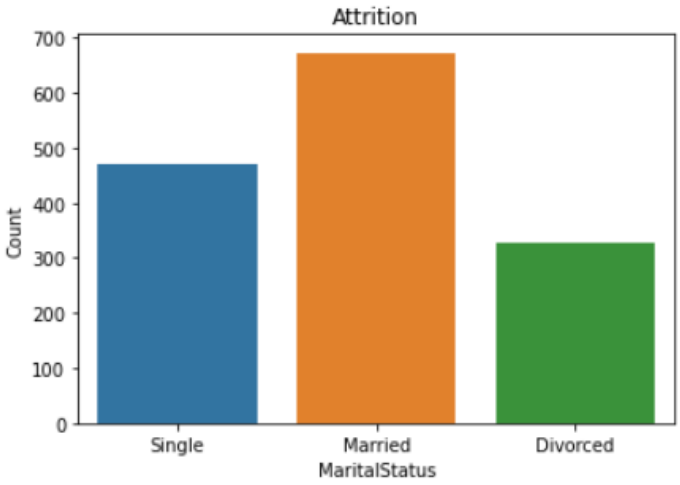
We use Univariant analysis to find the count for attrition, department, business travel, gender, education field, job role, overtime.

#### 3.1.1 Gender



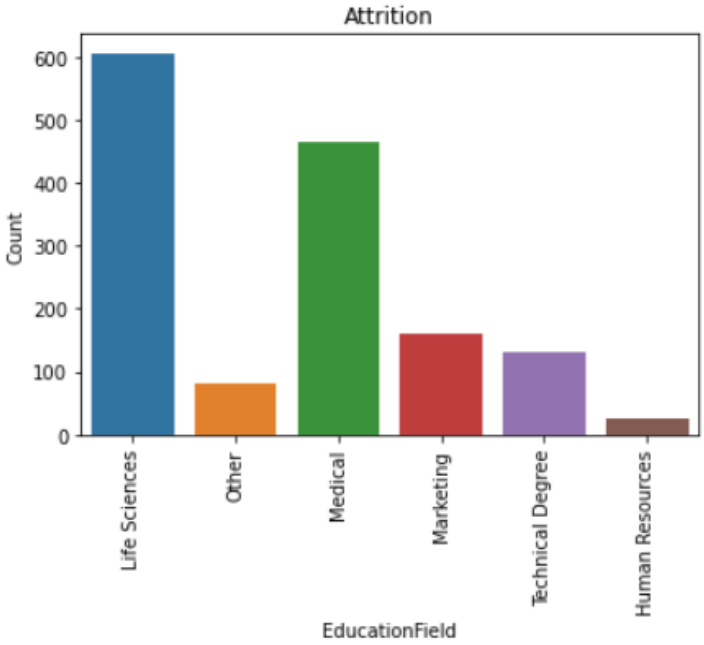

From the above barchart we get to know 882 males and 588 females. Count of male is higher than female.

#### 3.1.2 Marital Status



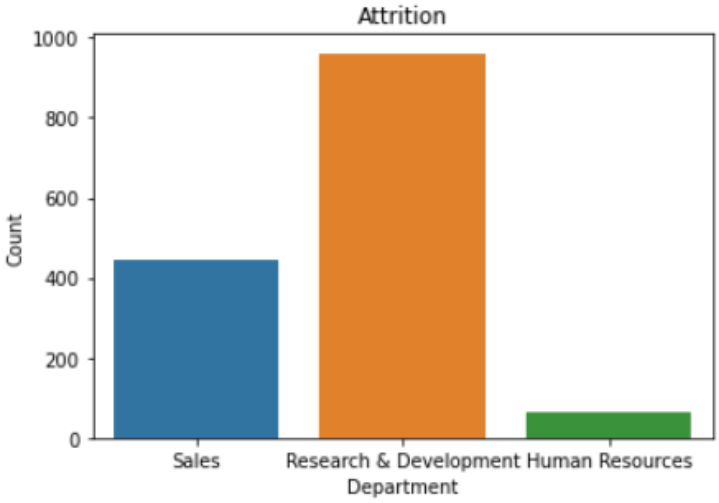

From the above barchart we get to know Married has 673, and single 470 and Divorced 371. Number of married is more.

### 3.1.3 Education field



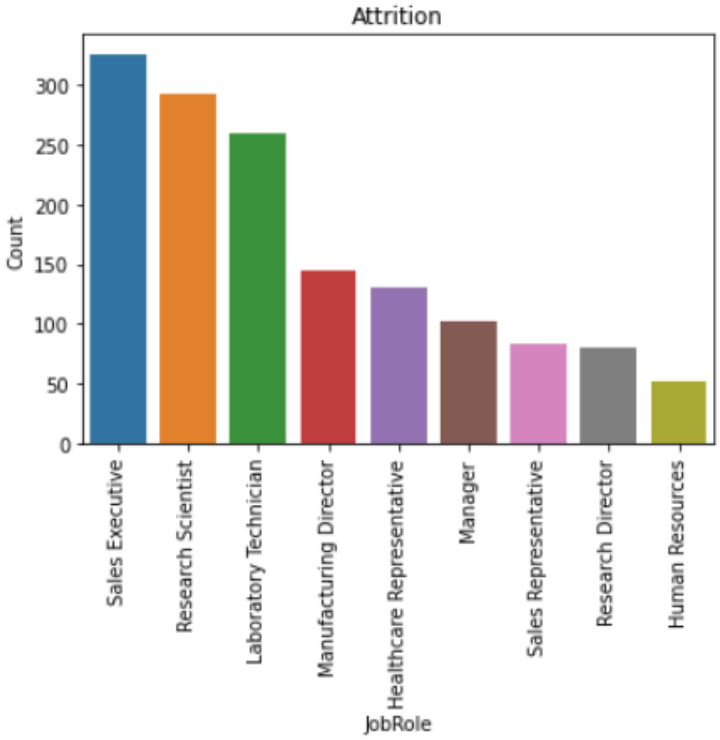

From the above barchart we get to know there are six different types of education field. The highest number are in Life science which has 606 and medical which has 464.

### 3.1.4 Department



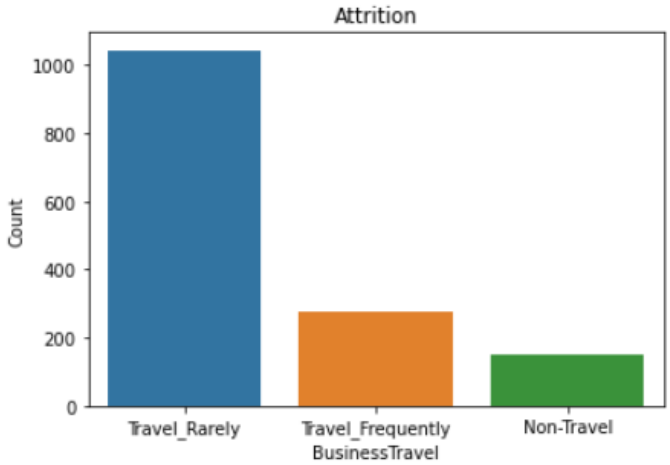

From the above barchart we get to know there are three department. RnD has the highest number 961, followed by sales 446 and HR 63.

### 3.1.5 Job Role



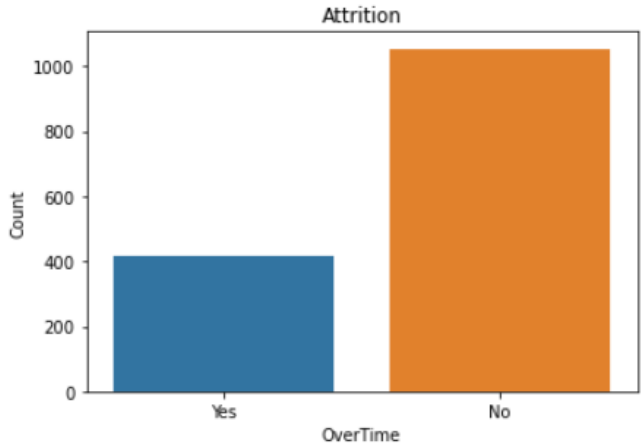

From the above barchart we get to know there are nine types of job role. Highest number are in sales executive and research scientistist and laboratory technician. Others are in small numbers when compared to tothers.

### 3.1.6 Business Travel



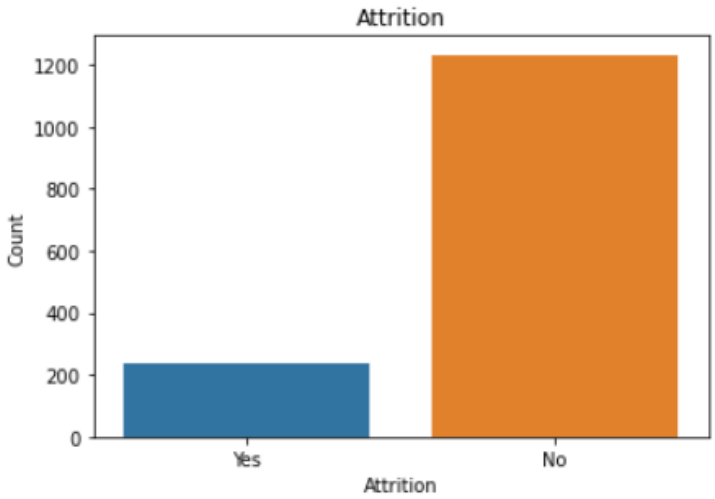

From the above barchart, Travel rarely has the highest number 1043. Travel frequently and non travel are less in number.

### 3.1.7 Overtime




From the above barchart, no overtime has 1054 and overtime has 416.

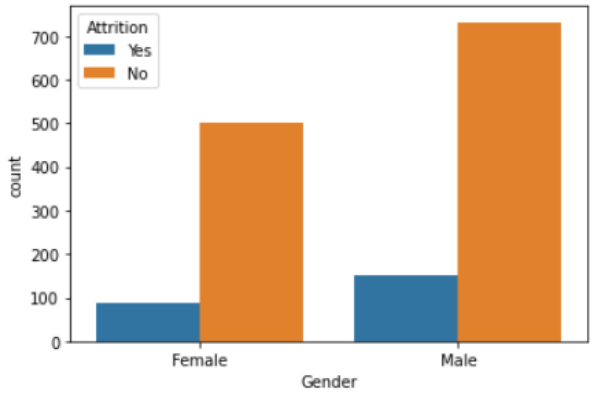
### 3.1.8 Attrition




From the above barchart, no attrition has 1233 and attrition has 237.

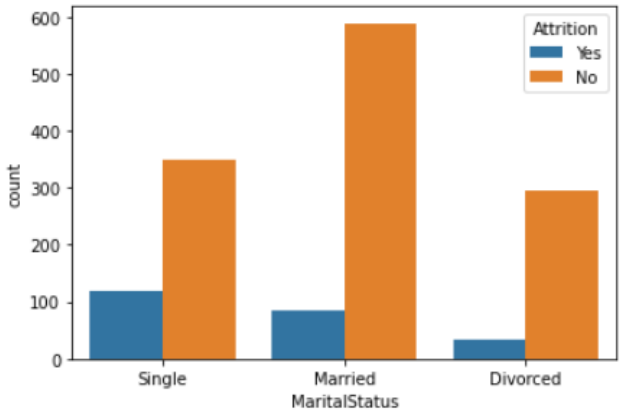
### 3.2 Bivariant Analysis:

### 3.2.1 Gender vs Attrition



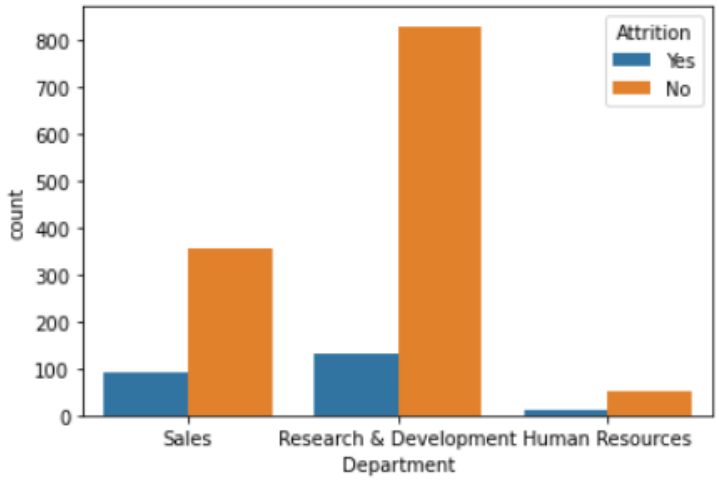

From the above chart we can infer that male has more chances of attrition 17% when compared to women 14%.

### 3.2.2 Marital status vs Attrition



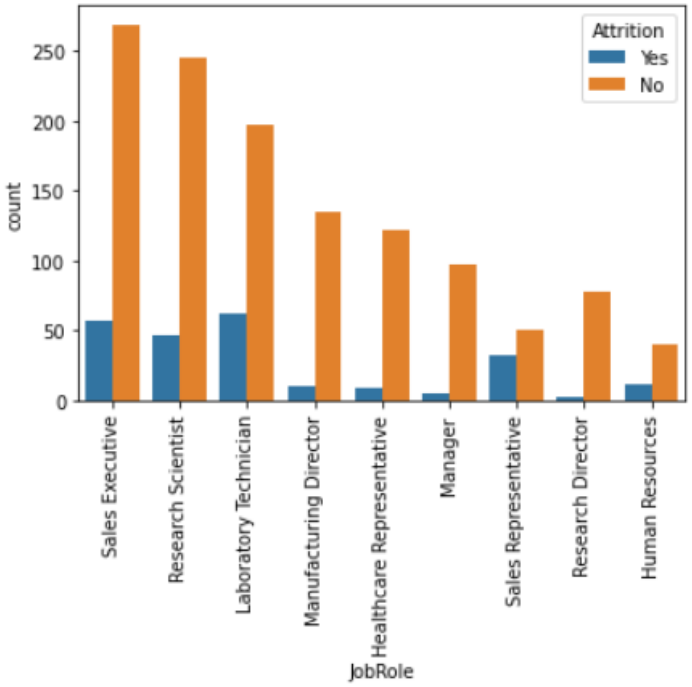

From the above chart we can infer that attrition is more for singles which is 25.5% when compared to married 12.4% and divorced 10%.

### 3.2.3 Department vs attrition



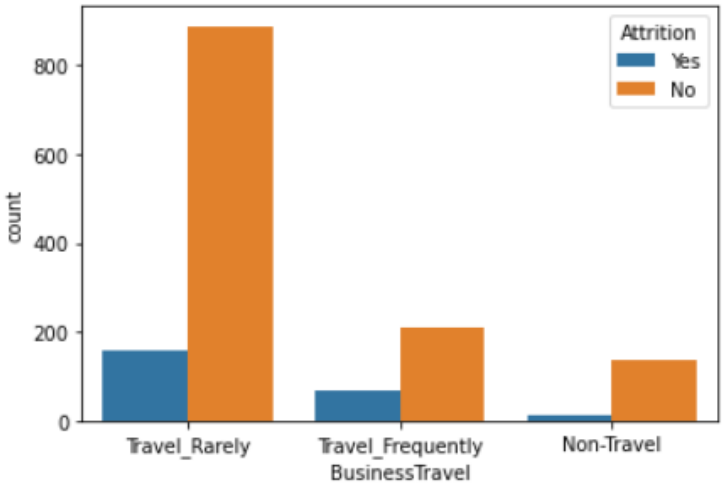

From the above chart we can infer that sales team has more chances of attrition 26%,when compared to research and developemnt 19% and human resource 13%.

### 3.2.4 Job role vs Attrition



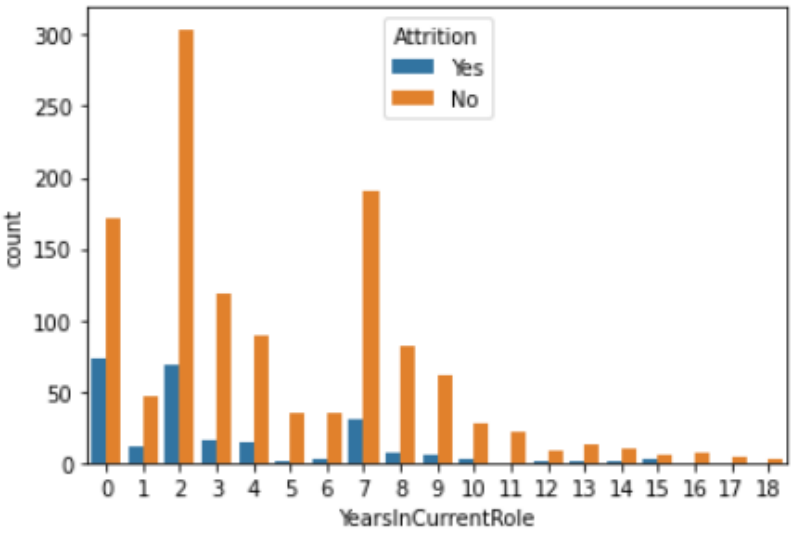

From the above chart we can infer that job role of sales representative and sales executive has highest attrition percentage when compared to other such as research scientist 16% and laboratory technician 14%.

### 3.2.5 BusinessTravel vs Attrition




From the above chart, travel frequently has more chances for attrition 24% when compared to travel rarely 14.9% and non travel 8%

### 3.2.6 Years in current role vs Attrition:




From the above barchart, we can infer the maximum number of attrition are there for freshers and below two years.

Similar analysis are done for other columns.

## 4. Train/Test split

Train/Test split procedure is used to evaluate the performance of various algorithms and used to make prediction on data.

First, we divide the data as input (X) and output (Y). Input consists of all columns except attrition and output consists of target variable (attrition). We use standard scaler technique on input data, to make it as normal distribution with mean 0 and standard deviation 1.

On X and Y data we split again with respect to train and test (x_train, x_test, y_train, y_test). Split percentage is of our choice. I have taken train as 70% and test as 30%. Train Dataset is used to train the model and Test data is used to predict the output when input is provided, and result is compared to the expected output.

We have 1387 rows and 31 columns. On splitting data with train and test we have:

- x_train.shape consists of 970 rows and 31 columns
- x_test.shape consists of 417 rows and 31 columns
- y_train.shape consists of 970 rows and 1 columns
- y_test.shape consists of 417 rows and 1 columns

## 5. Building model

Our target variable is binary data which has type 0 as not attrition and 1 as attrition. Hence, I go with binary classification model.

### 5.1 Import library

a. From sklearn.metrics- Import accuracy score, confusion matrix, classification report
b. From sklearn.ensemble- Import Logistic regression, AdaBoost Classifier, Random Forest Classifier, Gradient Boosting Classifier, Bagging Classifier, Extra Trees Classifier
c. From sklearn-Import Decision Tree Classifier, SVC, GaussianNB, KNeighbors Classifier

### 5.2 Classification metrics

1. Accuracy score-is calculated as true results among the total number of cases examined.
2. Confusion matrix-The matrix compares the actual target values with those predicted and diagonal will represent the error.
3. Classification report- consists of precision, recall and f1 score**.** Precision is calculated by (TP/TP+FP). Recall is calculated by (TP/TP+FN) and f1-score- is calculated by 2*((P*R)/(P+R)).

   On comparing the results on various algorithms, Best model will be chosen.

### 5.3. Classification and boosting method

#### 5.3.1 Classification Method

Classification algorithm is applied on the train data. We have used Logistic Regression, Decision Tree Classifier, GaussianNB, KNN and SVC.

- Logistic Regression: is used when the dependent variable is categorical. It is used because the relationship between the discrete variable and a predictor is non-linear.
- Decision Tree Classifier: Is a flowchart tree structure where internal node represents attribute, branch represent decision rule and leaf node represents outcome. We start at the tree root and split the data.
- Naïve Bayes: is based on Bayes' Theorem. Bayes Formula (P(H/X)= P(X/H)P(H) / P(X)). It has GaussianNB and MultinomialNB method.
- KNN: stands for knearest neighbor. Distance is calculated by Euclidian distance. Three steps are followed calculate distance, find the closest neighbors, vote for labels.
- SVC: Support vector machine. We perform classification by finding the hyper-plane that differentiates the two classes. SVC has technique called kernel trick (These functions takes low dimensional input space and transform to higher dimensional space). Kernel type Linear, poly and RBF**.**

#### 5.3.2 Boosting Method

Ensemble technique is used on algorithms to improve the accuracy score and get better results. We have used AdaBoost Classifier, RandomForest Classifier, Gradient Boosting Classifier, Extratree Classifier and bagging Classifier.

In Bagging method, multiple models (weak learners) are trained independently from each other in parallel way and all output is combined to get result. Random forest classifier and bagging classifier are the example of bagging method.

In Boosting method Output From one model is boosted and given as input for next model. It is Sequential method. Ada boost and Gradient boost are example of boosting method.

| **Algorithm** | **Accuracy Score** |
|---|---|
| **Logistic Regression** | **82.7** |
| DecisionTreeClassifier | **80** |
| GaussianNB | **72.9** |
| KNeighborsClassifie | **79.8** |
| SVC | 82.7 |
| AdaBoostClassifier | **85.6** |
| RandomForestClassifier | **84.8** |
| GradientBoostingClassifier | **83.9** |
| ExtraTreesClassifier | **84.4** |
| BaggingClassifier | **85.1** |

Based on the result obtained for accuracy score and f1 score, AdaBoost classifier has the highest score.

### 5.4 Cross validation

Cross validation (CV) is one of the techniques used to test the effectiveness of a machine learning models. K-Fold CV is where a given data set is split into a K number of sections. We have used K value to be 5. Data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds has been used as the testing set.

Our model has an average accuracy of 86% with a standard deviation of 12 %. The standard deviation shows how precise the estimates are.

| Algorithm | Cross validation Score |
|---|---|
| Logistic Regression | **84.1** |
| DecisionTreeClassifier | **77.8** |
| GaussianNB | **77.7** |
| KNeighborsClassifier | **81.5** |
| SVC | **83.4** |
| AdaBoostClassifier | **85.7** |
| RandomForestClassifier | **85.2** |
| GradientBoostingClassifier | **86.1** |
| ExtraTreesClassifier | **84.4** |
| BaggingClassifier | **85.3** |

Based on the result obtained, Gradient boosting and AdaBoost classifier has the highest score. We will try to increase its performance by hyper tuning the parameter for gradient boost and adaboost.

### 5.5 GridsearchCV

GridsearchCV is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. We pass in few parameters such as learning rate and n_estimator. Among the values passed in dictionary optimal values is taken. We have obtained best parameters of learning_rate as 0.1 and n_estimators as 400 for AdaBoost and criterion as mse for Gradient boost classifier.
Upon the results obtained for both, AdaBoost is selected as best model with accuracy of 85.6 and f1 score as 83.

```
#GridsearchCV for Adaboost
from sklearn.model_selection import GridSearchCV
ab=AdaBoostClassifier()
parameters={'n_estimators':[400,500,600,700],'learning_rate':[1,10,0.1]}
clf=GridSearchCV(ab,parameters)
clf.fit(x,y)
print('best parameters',clf.best_params_)
```

```
best parameters {'learning_rate': 0.1, 'n_estimators': 400}
```

```
#GridsearchCV for Gradientboost
from sklearn.model_selection import GridSearchCV
gb=GradientBoostingClassifier()
parameters={'criterion':['friedman_mse','mse','mae']}
clf=GridSearchCV(gb,parameters)
clf.fit(x,y)
print('best parameters',clf.best_params_)
```

```
best parameters {'criterion': 'mse'}
```

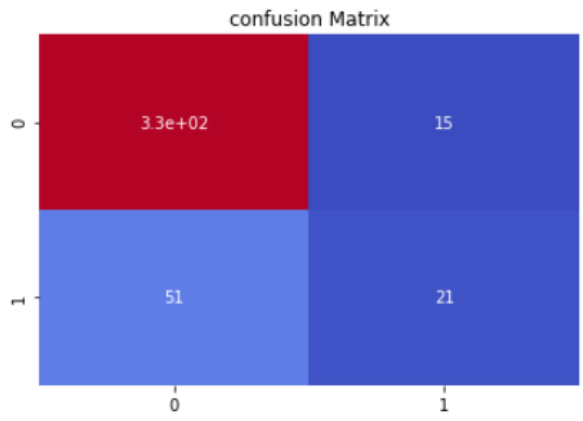
### 5.6 Best model

Based on the result obtained on GridsearchCV, model with those parameters will be saved. We can use the model to predict the output. AdaBoost classifier is chosen as best model. We have obtained the below results:

- accuracy score:85.6
- Precision-84
- Recall-86
- f1 score-83
- Confusion Matrix

```
cm=confusion_matrix(y_test,predm)
sns.heatmap(cm,annot=True,cbar=False,cmap='coolwarm')

plt.title("confusion Matrix")
plt.show()
```



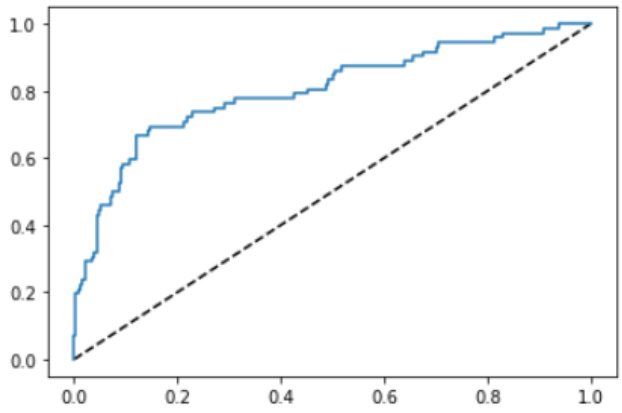
## 6. AUC-ROC Curve

ROC is a probability curve and AUC represent the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC score, the better the model is. ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis. TPR defines how many correct positive results occur among all positive samples available during the test. FPR defines how many incorrect positive results occur among all negative samples available during the test.

From the graph below, we have obtained AUC score as 62.1

```
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr,label='LR')
```

```
[<matplotlib.lines.Line2D at 0x21de93e5a60>]
```




```
auc_score=roc_auc_score(y_test,abc.predict(x_test))
print(auc_score)
```

```
0.6217995169082126
```

## 7. Conclusion

The stronger indicators of attrition which are highly correlated are:

- Overtime: People who work overtime are more likely to leave the company.
- Marital status: Single has more chances to leave.
- Distance from home: People who live further from home are more likely to leave the company.
- Years at company: Less number of years worked at company has more chances to leave.

Hence steps have to be taken reduce attrition.