**Problem statement:**

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?

**Data analysis:**

Reading the Csv File. And setting the display option to show all columns and rows

**df = pd.read\_csv("WA\_Fn-UseC\_-HR-Employee-Attrition.csv")**

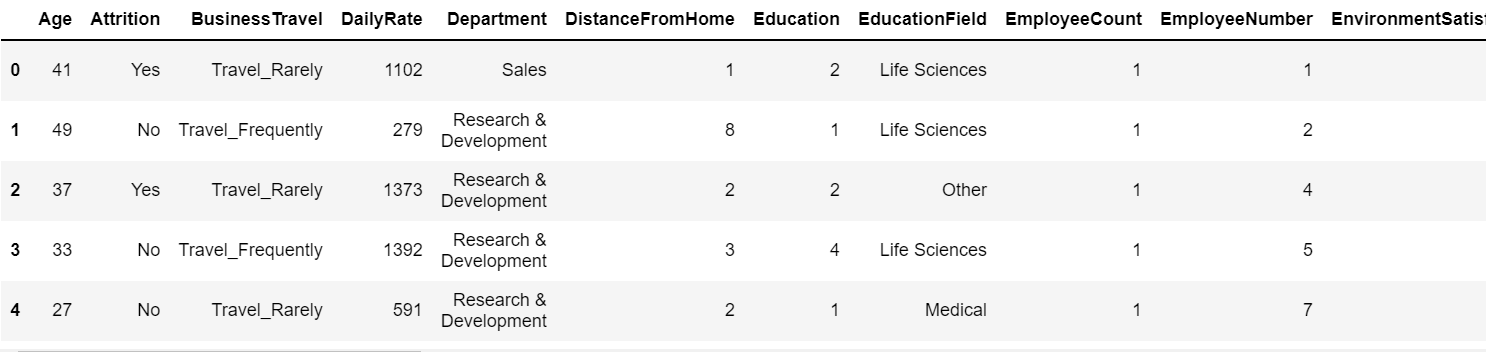
**pd.set\_option("display.max\_rows",None)**

**pd.set\_option("display.max\_columns",None)**

**Code:**

**df.head()**

**Output:**

****

**Code:**

**df.shape**

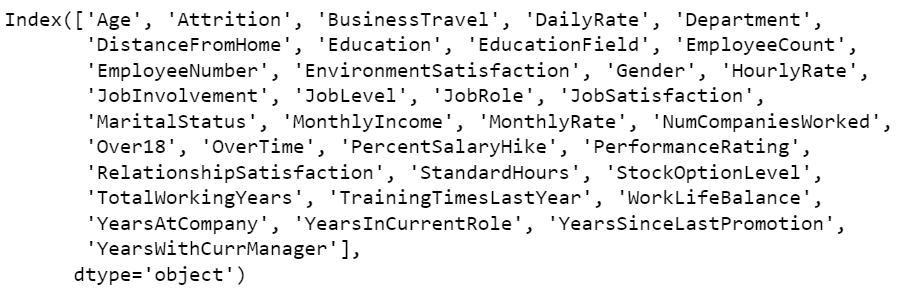
**Output:**

****

**Code:**

**df.columns**

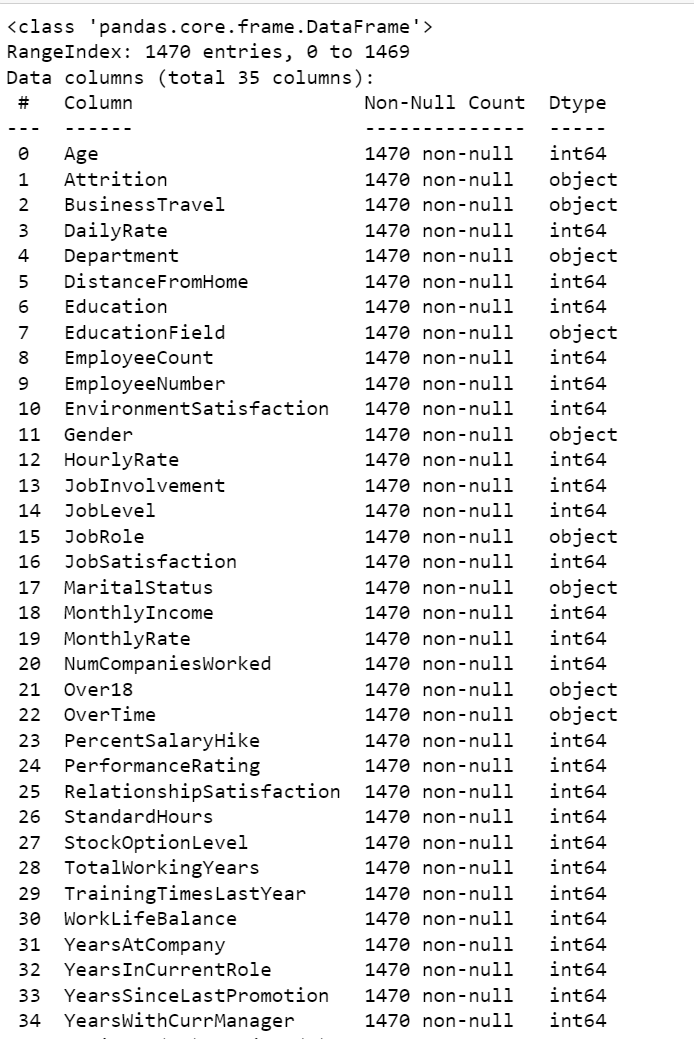
**Output:**

****

**Code:**

**df.info()**

**Output:**

****

**Observation:**

We can see the Null value count and Data types. there are no null values in data.

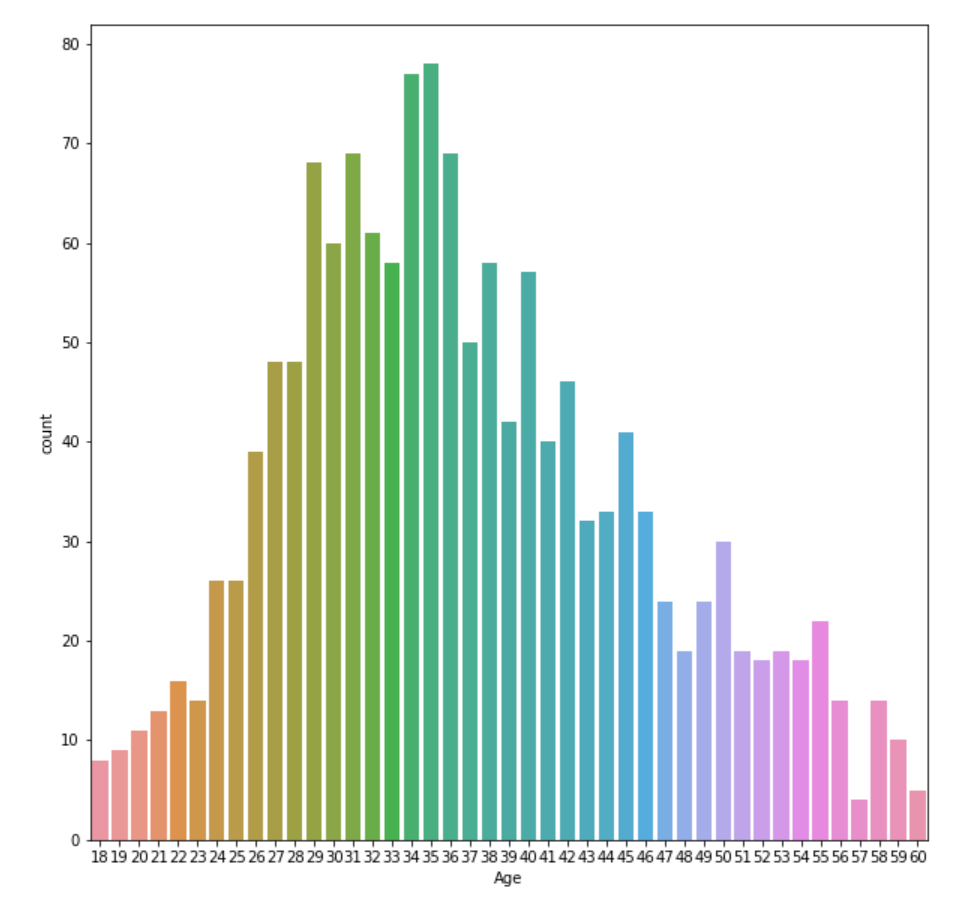
**Univariate analysis:**

**Code:**

**sns.countplot(df["Age"])**

**plt.show()**

**Output:**

****

**Observation:**

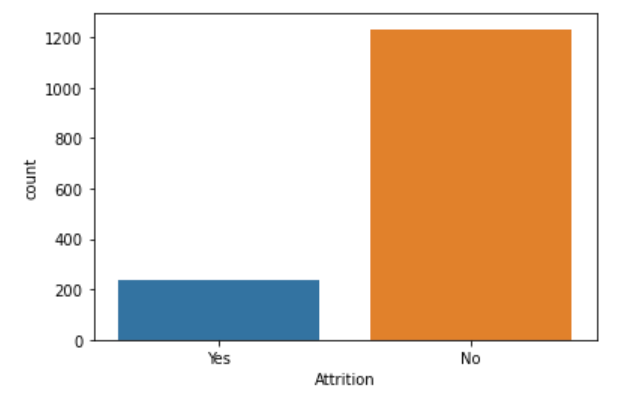
Age group is higher between 27 to 45

**Code:**

**sns.countplot(df["Attrition"])**

**plt.show()**

**Output:**

****

**Observation:**

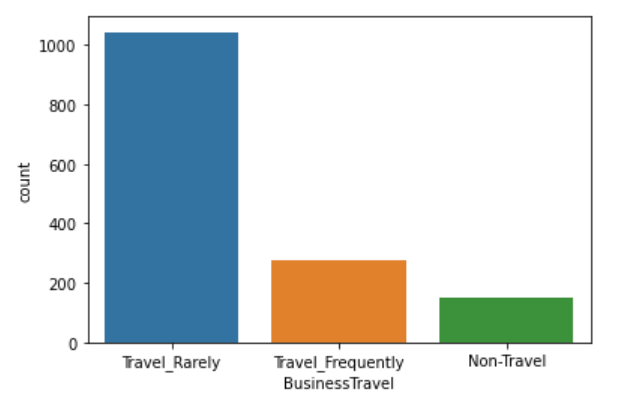
Attrition target variable is imbalanced so it needs to be balanced

**Code:**

**sns.countplot(df["BusinessTravel"])**

**plt.show()**

**Output:**

****

**Observation:**

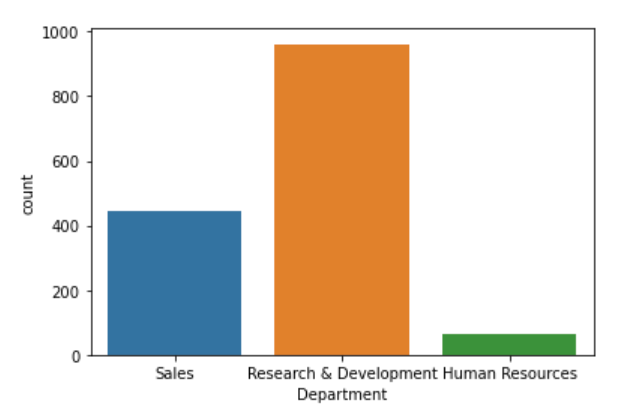
Business Travel has Travel rarely highly ranked. it shows most jobs don’t need frequent travels

**Code:**

**sns.countplot(df["Department"])**

**plt.show()**

**Output:**



**Observation:**

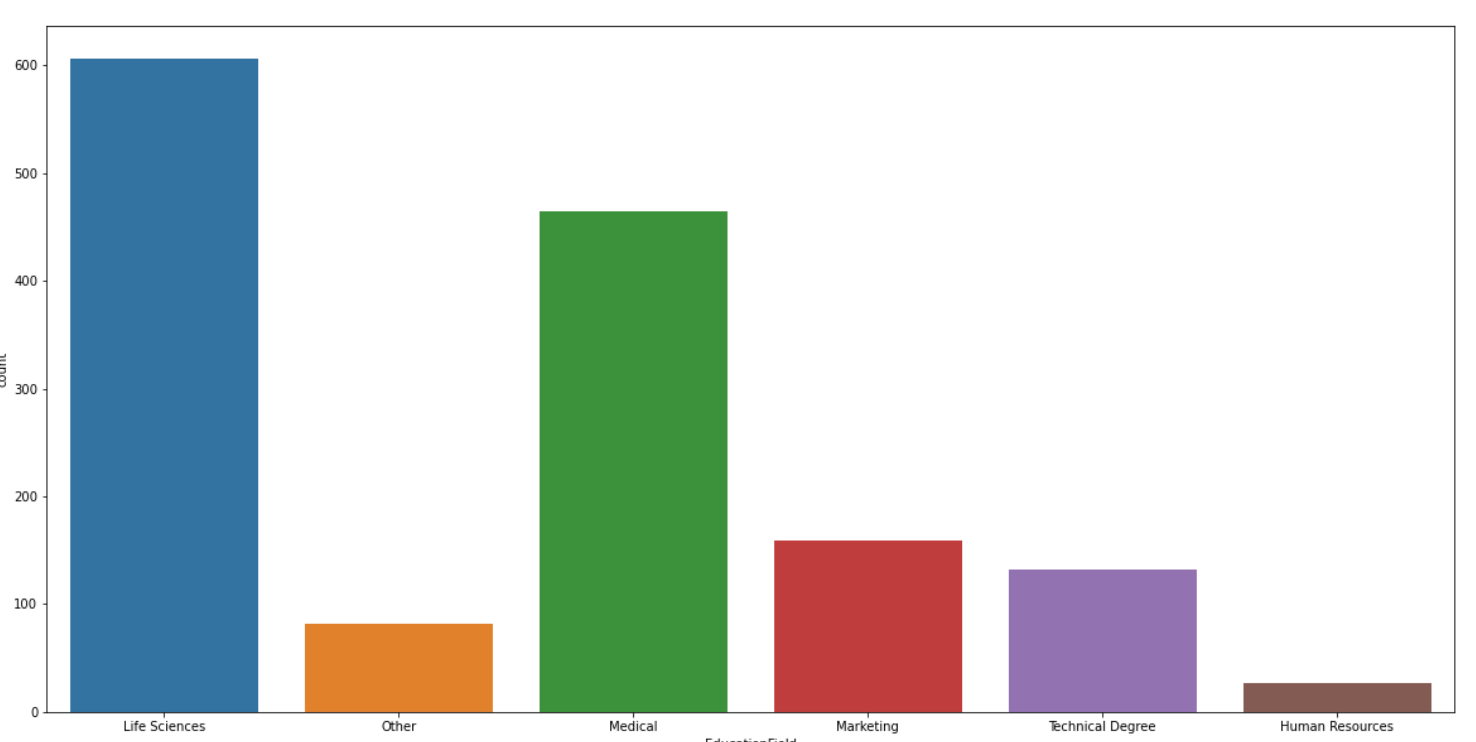
Research & Development Departments have higher counts in this data

**Code:**

**plt.figure(figsize=(20,10))**

**sns.countplot(df["EducationField"])**

**Output:**

****

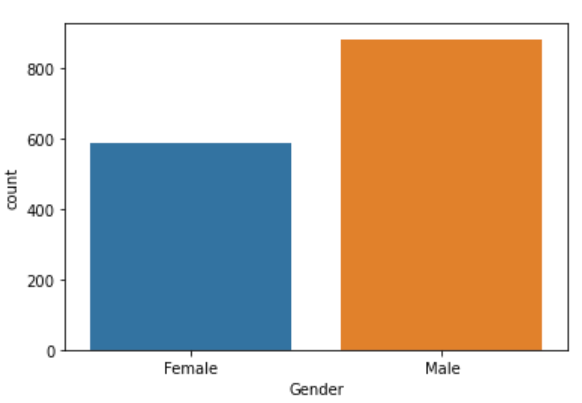
**Observation:**

We can can see life sciences count is higher. people mostly have education in life science

**Code:**

**sns.countplot(df["Gender"])**

**Output:**

****

**Observation:**

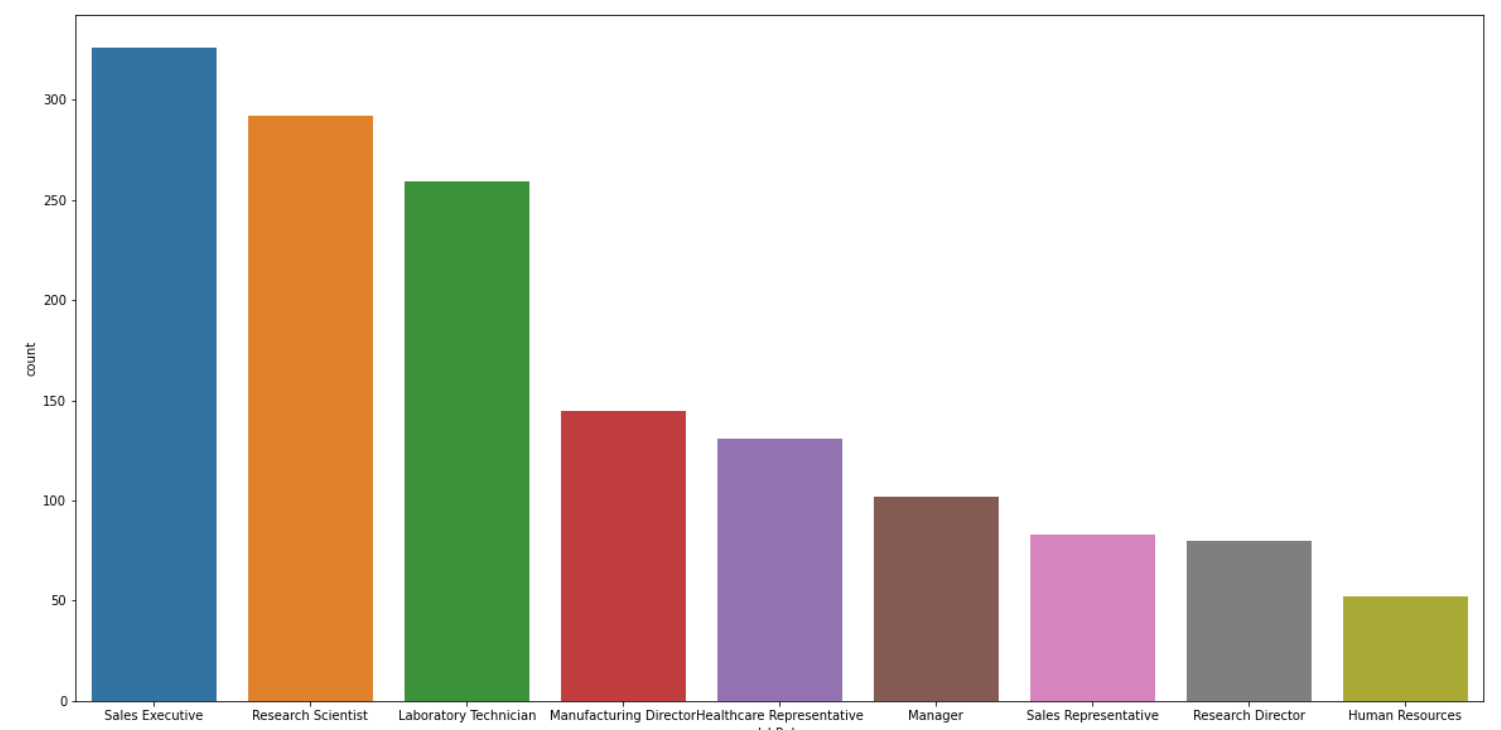
Female workforce is less according to this data

**Code:**

**plt.figure(figsize=(20,10))**

**sns.countplot(df["JobRole"])**

**Output:**

****

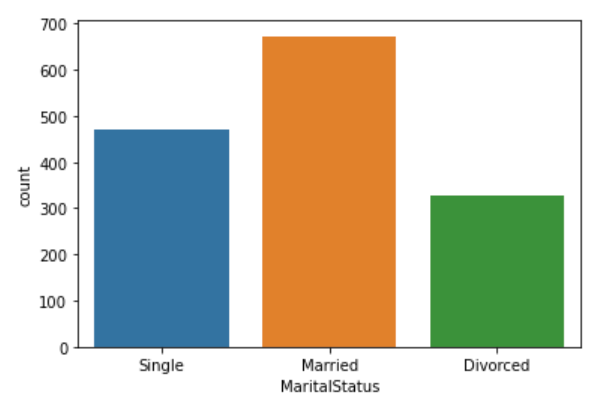
**Observation:**

Sales executive job role is higher followed by research scientist

**Code:**

**sns.countplot(df["MaritalStatus"])**

**Output:**

****

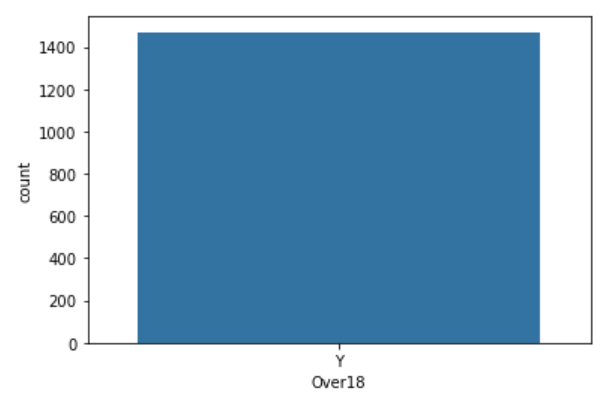
**Observation:**

Most of the workforce is married . we have also seen higher age group is between 27 to 45 so it makes sense

**Code:**

**sns.countplot(df["Over18"])**

**Output:**

****

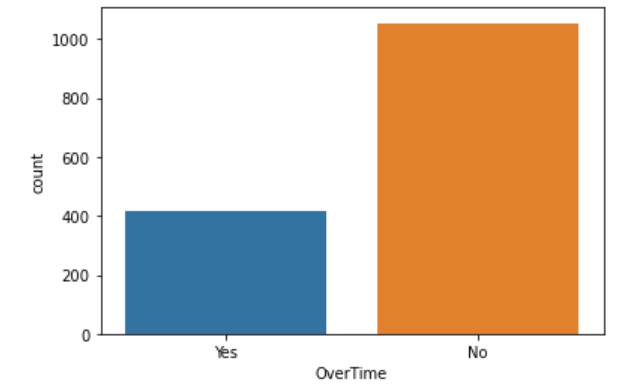
**Observation:**

almost all working people are above 18

**Code:**

**sns.countplot(df["OverTime"])**

**Output:**

****

**Observation:**

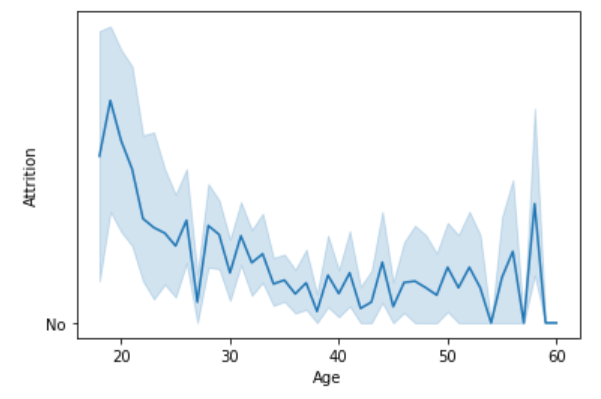
Most jobs seem’s don’t have overtime

**Bivariate Analysis:**

**Code:**

**sns.lineplot(df["Age"],df["Attrition"])**

**Output:**

****

**Observations:**

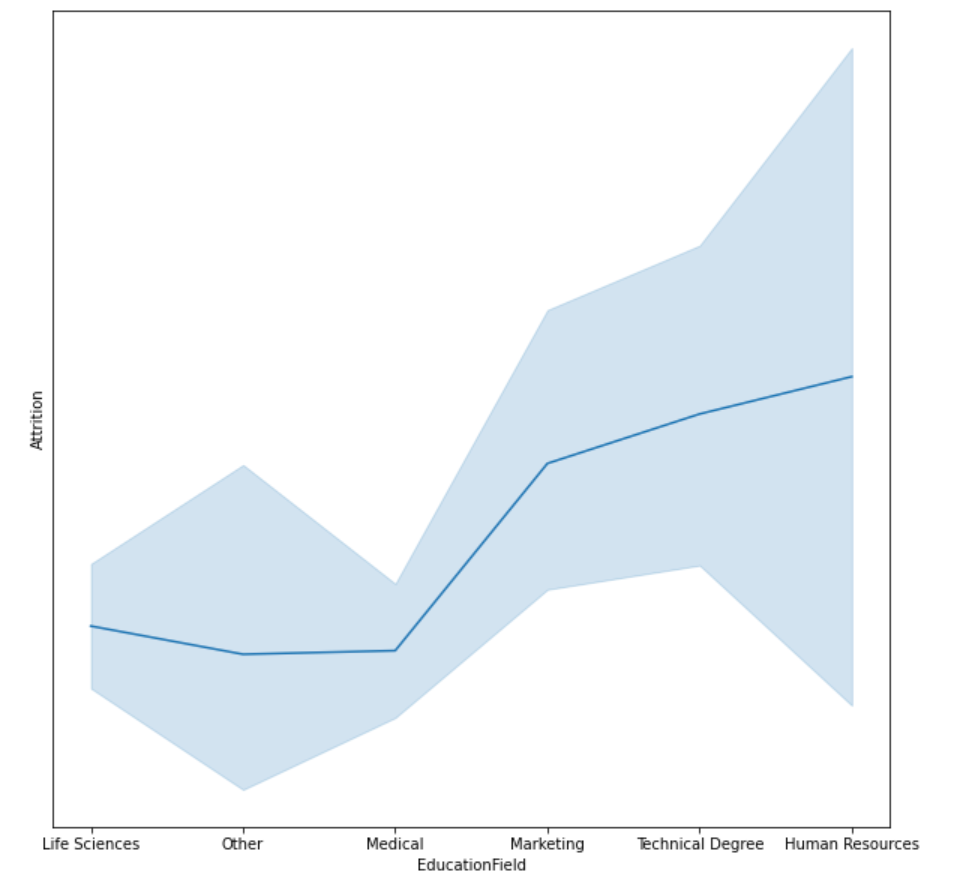
it seems people below age group 20 have high attrition rate

**Code:**

**plt.figure(figsize=(10,10))**

**sns.lineplot(df["EducationField"],df["Attrition"])**

**Output:**

****

**Observation:**

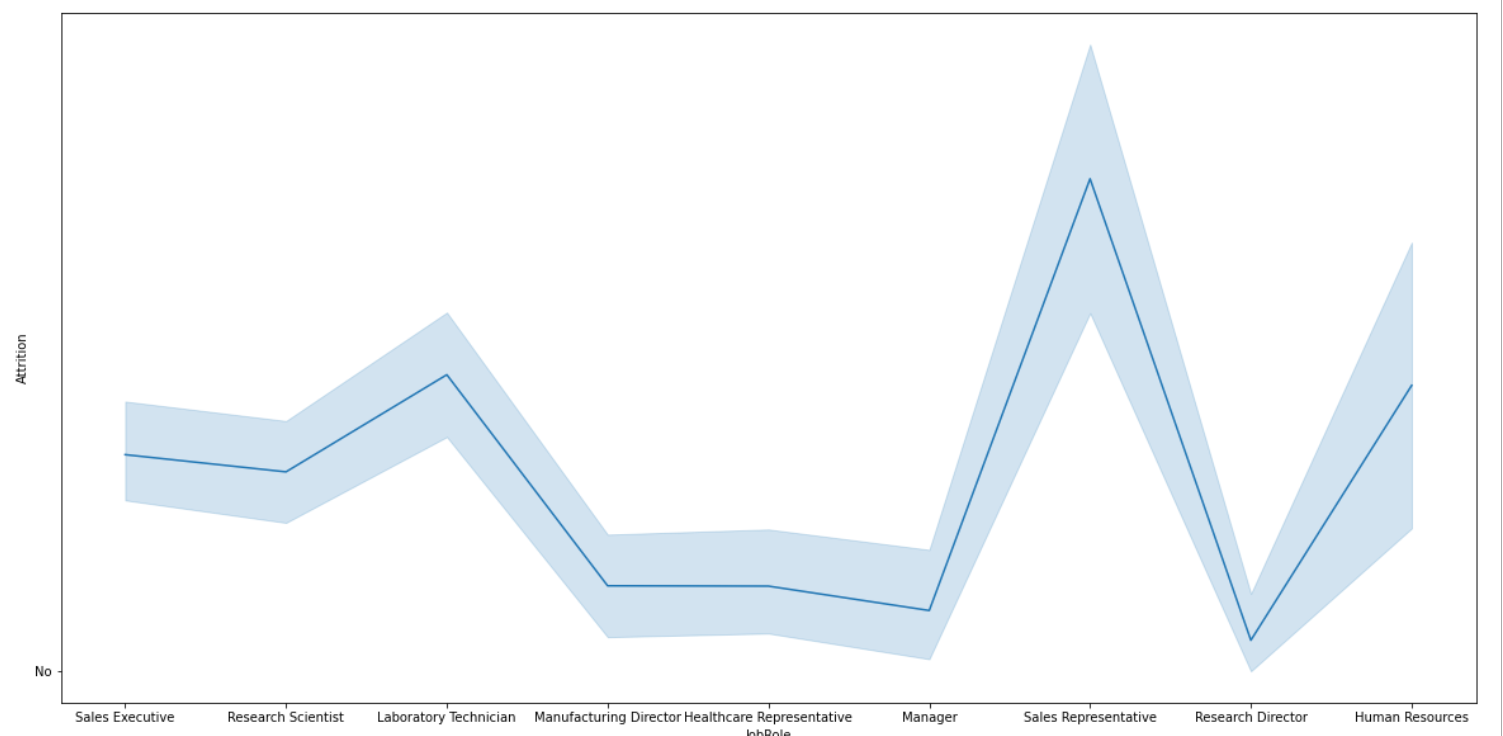
Attrition is higher for Human resource and Technical Degree

**Code:**

**plt.figure(figsize=(20,10))**

**sns.lineplot(df["JobRole"],df["Attrition"])**

**Output:**

****

**Observation:**

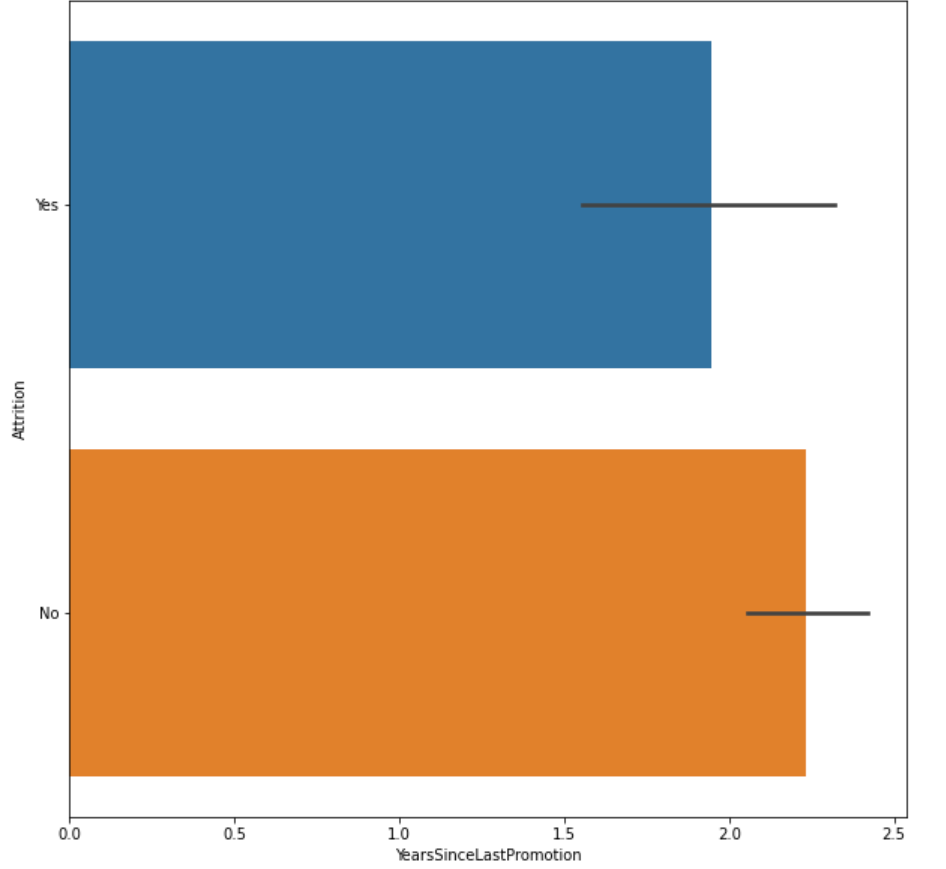
sales representative jobs have higher Attrition

**Code:**

**plt.figure(figsize=(10,10))**

**sns.barplot(df["YearsSinceLastPromotion"],df["Attrition"])**

**Output:**

****

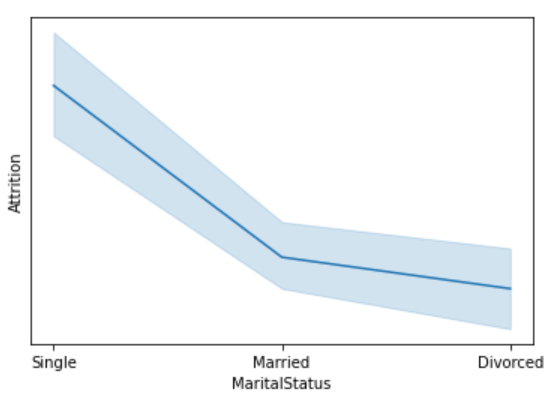
**Observation:**

Attrition is higher since YearsSinceLastPromotion increases

**Code:**

**sns.lineplot(df["MaritalStatus"],df["Attrition"])**

**Output:**

****

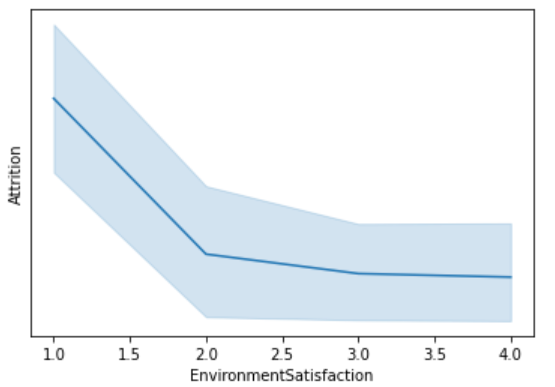
**Observation:**

single people have high Attrition.

**Code:**

**sns.lineplot(df["EnvironmentSatisfaction"],df["Attrition"])**

**Output:**



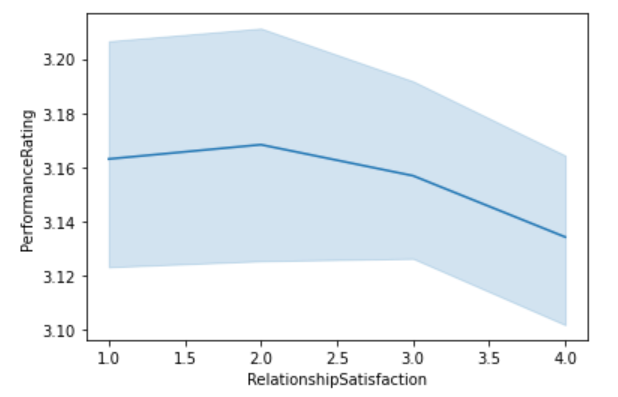
**Observation:**

As Environment Satisfaction increase attrition decrease. people tend to leave when they don’t adapt

**Code:**

**sns.lineplot(df["RelationshipSatisfaction"],df["PerformanceRating"])’**

**Output:**



**Observation:**

Performance rating decrease with increase in relationship satisfaction

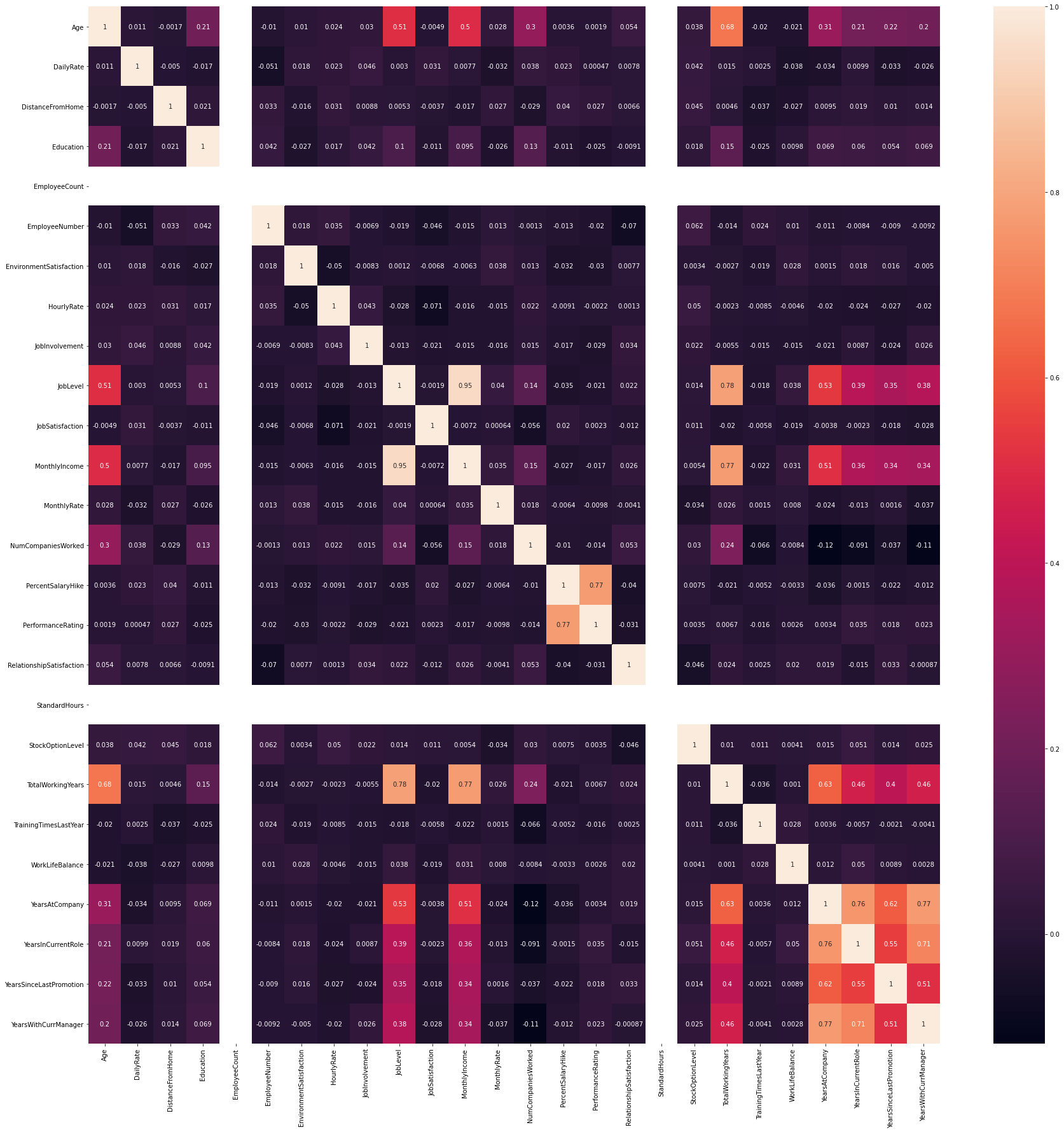
**Correlation:**

**Code:**

**plt.figure(figsize=(30,30))**

**sns.heatmap(df.corr(),annot=True)**

**Output**

****

**Observation:**

Percent Salary Hike is Highly Correlated to Performance Rating, Monthly Income is Highly Correlated to Total working years. Percent Salary Hike is highly correlated to Performance Rating.

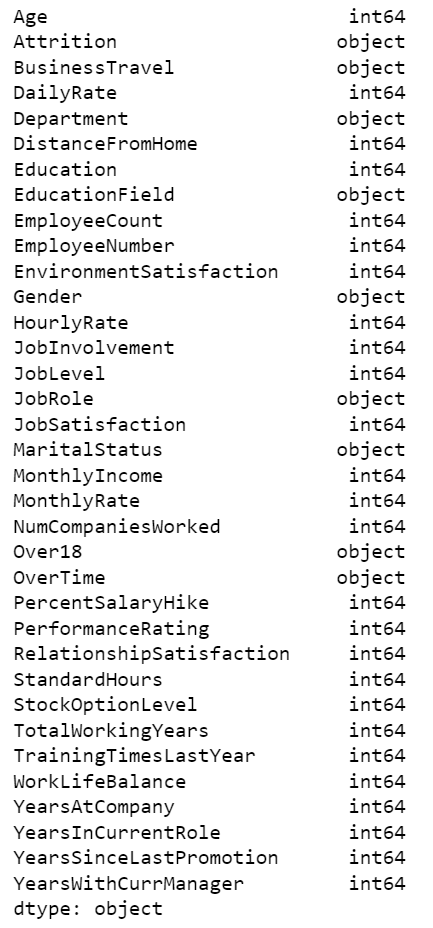
**Encoding:**

The Categorical data are not ordinal in Nature So we are going to use get dummies method to encode them.

**Code:**

**df.dtypes**

**Output:**



**Observation:**

We can see Some columns are Object Data so we are going to encode them

**Code:**

enc = ["BusinessTravel","Department","EducationField","Gender","JobRole","MaritalStatus","Over18","OverTime"]

a = pd.get\_dummies(df[["BusinessTravel","Department","EducationField","Gender","JobRole","MaritalStatus","Over18","OverTime"]])

**Observation:**

I am creating a list of categorical data without target variable and using pd.get\_dummies to encode them. And storing them in variable name a.

**Code:**

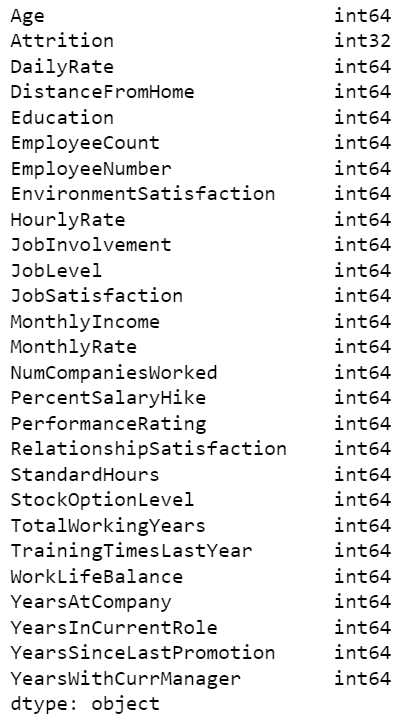
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df["Attrition"] = le.fit\_transform(df["Attrition"].values.reshape(-1,1))

df.drop(["BusinessTravel","Department","EducationField","Gender","JobRole","MaritalStatus","Over18","OverTime"],axis=1,inplace=True)

**Output:**



**Observation:**

We are using label encoder for target variable and we can see all categorical data are converted to Numbers.

**Skewness:**

**Code:**

dfs = df.drop(["Attrition"],axis=1)

**Observation:**

I am dropping Target Variable and creating a new data frame for checking skewness

**Code:**

dfs.plot(kind="hist",subplots=True,layout=(6,6),figsize=(20,20))

plt.show()

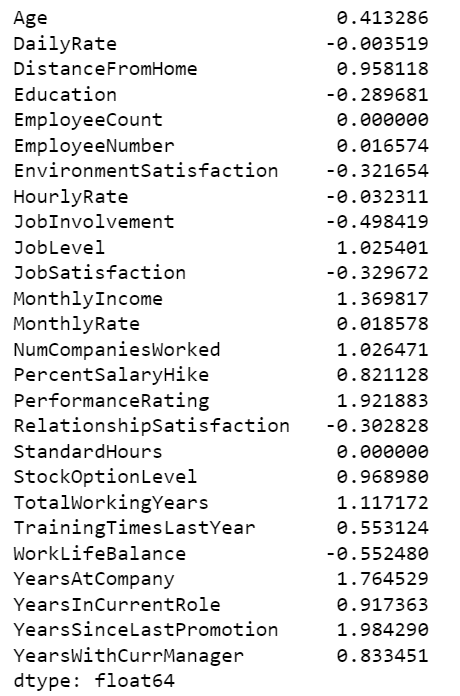
**Output:**



**Code:**

dfs.skew()

**Output:**



**Observation:**

Lot of features have skewness so proceeding with power transform for skewness removal.

**Code:**

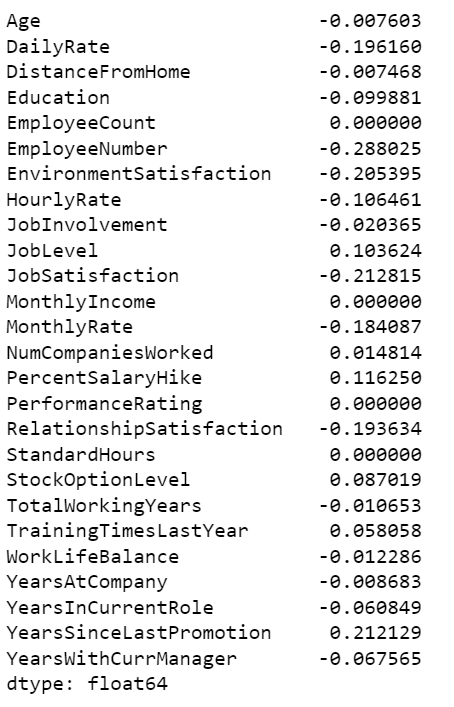
from sklearn.preprocessing import power\_transform

pe = power\_transform(dfs)

dfs1 = pd.DataFrame(pe,columns = dfs.columns)

dfs1.skew()

**Output:**



**Observation:**

skewness is removed

**Outliers:**

**Code:**

from scipy.stats import zscore

z = np.abs(zscore(dfs1))

print(np.where(z>3))

**Output:**



Observation:

No outliers are found

**SMOTE:**

Code:

df1 = pd.DataFrame(a)

dfv = dfs1.join(df1)

dfv.shape

**Output**



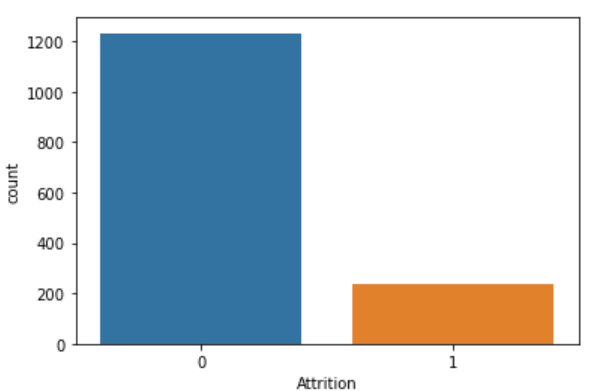
**Observation:**

I am joining the Skewness removed data and Categorical Dataframe.

**Code:**

sns.countplot(df["Attrition"])

Output:



**Observation:**

Our Target variable has imbalace so we use smote to balance them

**Code:**

from imblearn.over\_sampling import SMOTE

dx = dfv

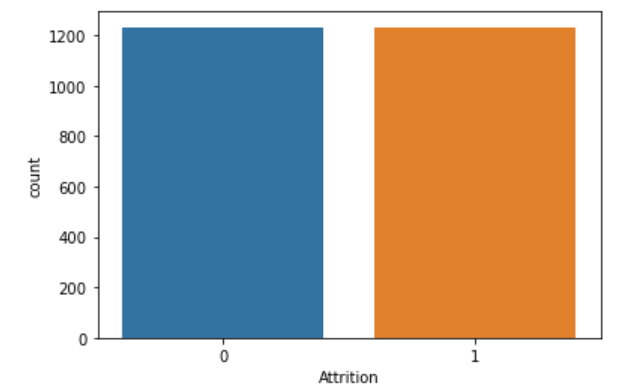
dy = df["Attrition"]

sm = SMOTE()

sx,sy = sm.fit\_resample(dx,dy)

sns.countplot(sy)

**Output:**



**Observation:**

we have balanced data using SMOTE

**Standard Scaler:**

**Code:**

from sklearn.preprocessing import StandardScaler

sca = StandardScaler()

s = sca.fit\_transform(x)

x1= pd.DataFrame(s,columns=x.columns)

**Observation:**

We have Scaled the Data Which is not categorical and stored it in dataframe

**PCA:**

Now Joining the Scaled Data with Categroical Data.

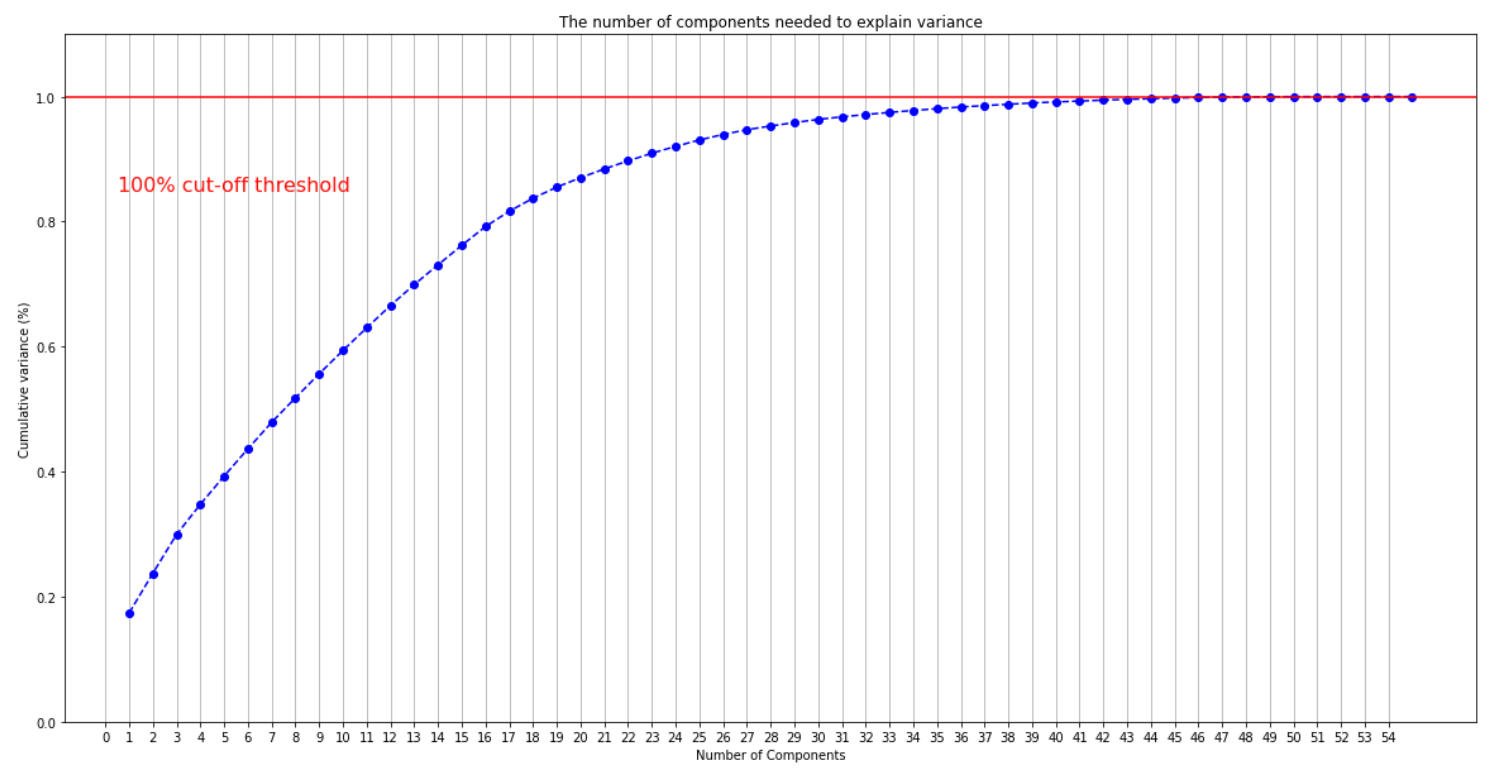
**Code:**

p=x1.join(sx.iloc[:,26:])

**Code:**



**Output:**



**Observation:**

We can with 46 Columns we can Retain all information without any Loss.

**Code:**

pca = PCA(n\_components=46)

x\_final = pca.fit\_transform(p)

x\_final.shape

Output:



**Observation:**

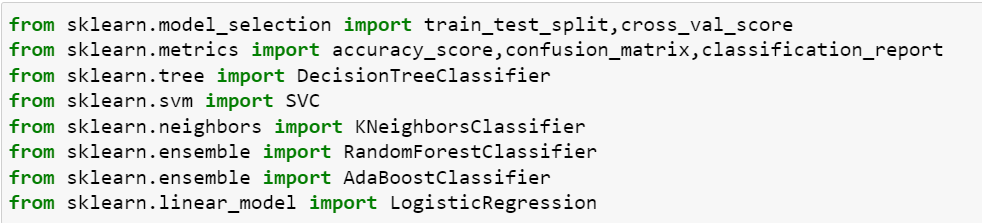
We have converted them to 46 Columns. PCA will handle Multicollinearity so no need to check for them.

**EDA Concluding Remarks:**

**We have processed the data checked for null values, encoded categorical data, with outliers removed skewness, checked for outliers, balanced the target variables and finally reduced the columns using PCA.**

**BUILDING MACHINE LEARNING MODEL:**

**Code:**

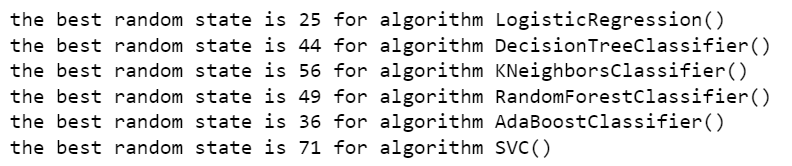
Observation:

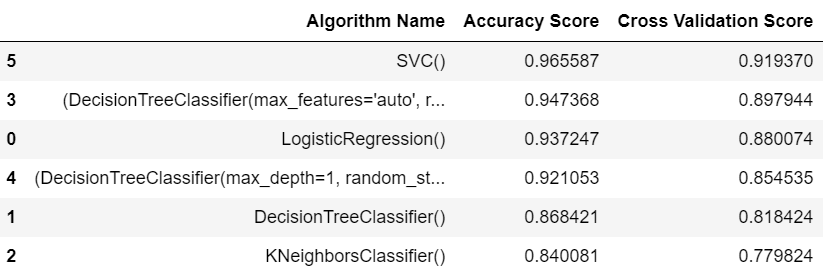
Importing all Libraries and functions required for Model Training and testing.

**Code:**



**Output:**





**Observation:**

This code prints the best random state for algorithm and ranks them in Data Frame. As we can see Support Vector Classifier performed best compared to all.

**Hyperparameters Tuning**

**Code:**



**Output:**

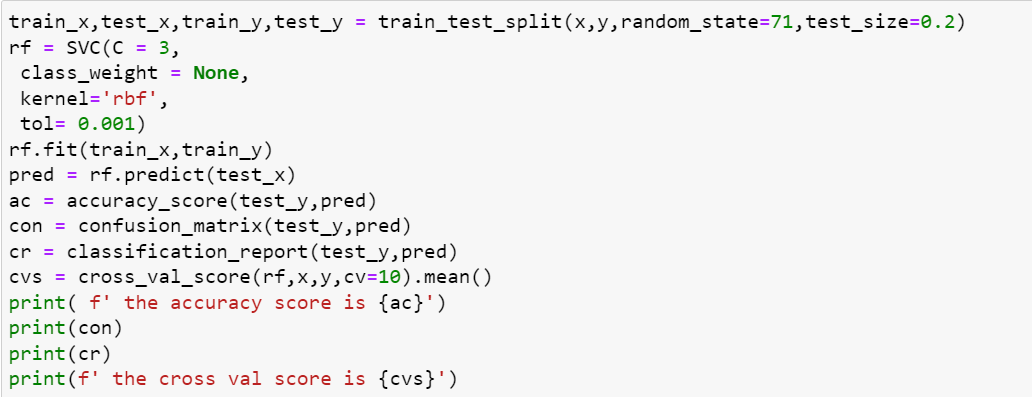


**Observation:**

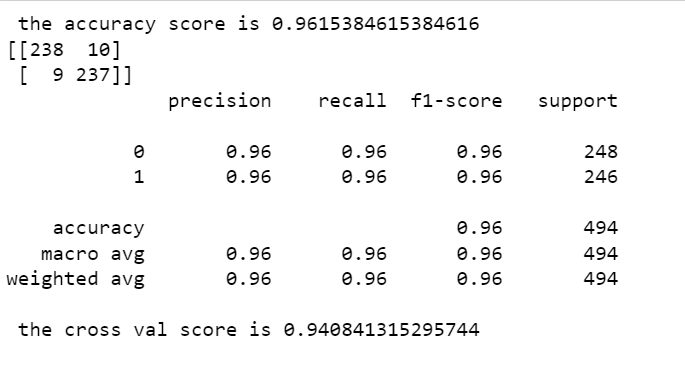
Now we got best parameters using Grid Search CV

Model Training:

**Code:**



**Output:**

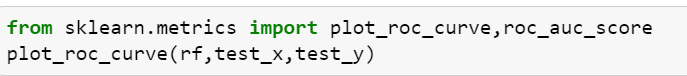


**Observation:**

We can see Hyper Parameter Tuning Boosted our Cross Validation Score from 91% to 94%

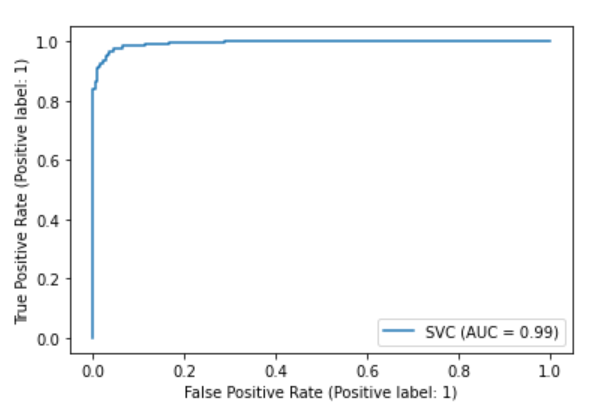
**AUC-ROC Curve:**

**Code:**





**Output:**

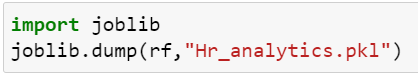




**Observation:**

We got Good AUC – ROC score of 96%

**Saving the Model:**



**CONCLUSION:**

**We have conducted the Data Analysis on Employee Attrition Dataset. Built the best machine learning model using hyper parameter tuning.**