Import Libraries

In [1]:

```
import warnings
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from pandas import DataFrame
from statsmodels.formula.api import ols
from scipy.stats import chi2_contingency
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
```

Read Data

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
import pandas as pd
A = pd.read_csv('HR-Employee-Attrition-Table 1.csv')
```

Profile

In [4]:

```
A.head(3)
```

Out[4]:

	Attrition	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount
0	1	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	1
1	0	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1
2	1	37	Travel_Rarely	1373	Research & Development	2	2	Other	1

3 rows × 35 columns

In [5]:

```
A.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Attrition                 1470 non-null   int64 
 1   Age                       1470 non-null   int64 
 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(27), object(8)
memory usage: 402.1+ KB
```

> ## Missing Data

In [6]:

```
A.isna().sum()
```

Out[6]:

```
Attrition                   0
Age                         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
```

We can see data need not to replace any value but still Replacer and Preprocessing function we will create

In [7]:

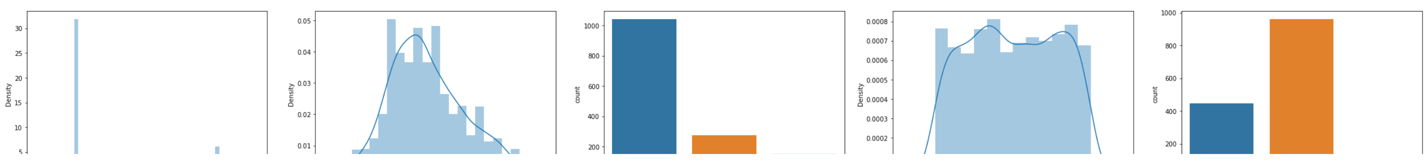
```python
def replacer(df):
    T=pd.DataFrame(df.isna().sum(), columns=['misval'])
    Q = T[T.misval > 0]
    for i in Q.index:
        if df[i].dtypes == "object":
            mode = df[i].mode()[0]
            df[i]=df[i].fillna(mode)
        else:
            mean = round(df[i].mean(), 2)
            df[i] = df[i].fillna(mean)
```

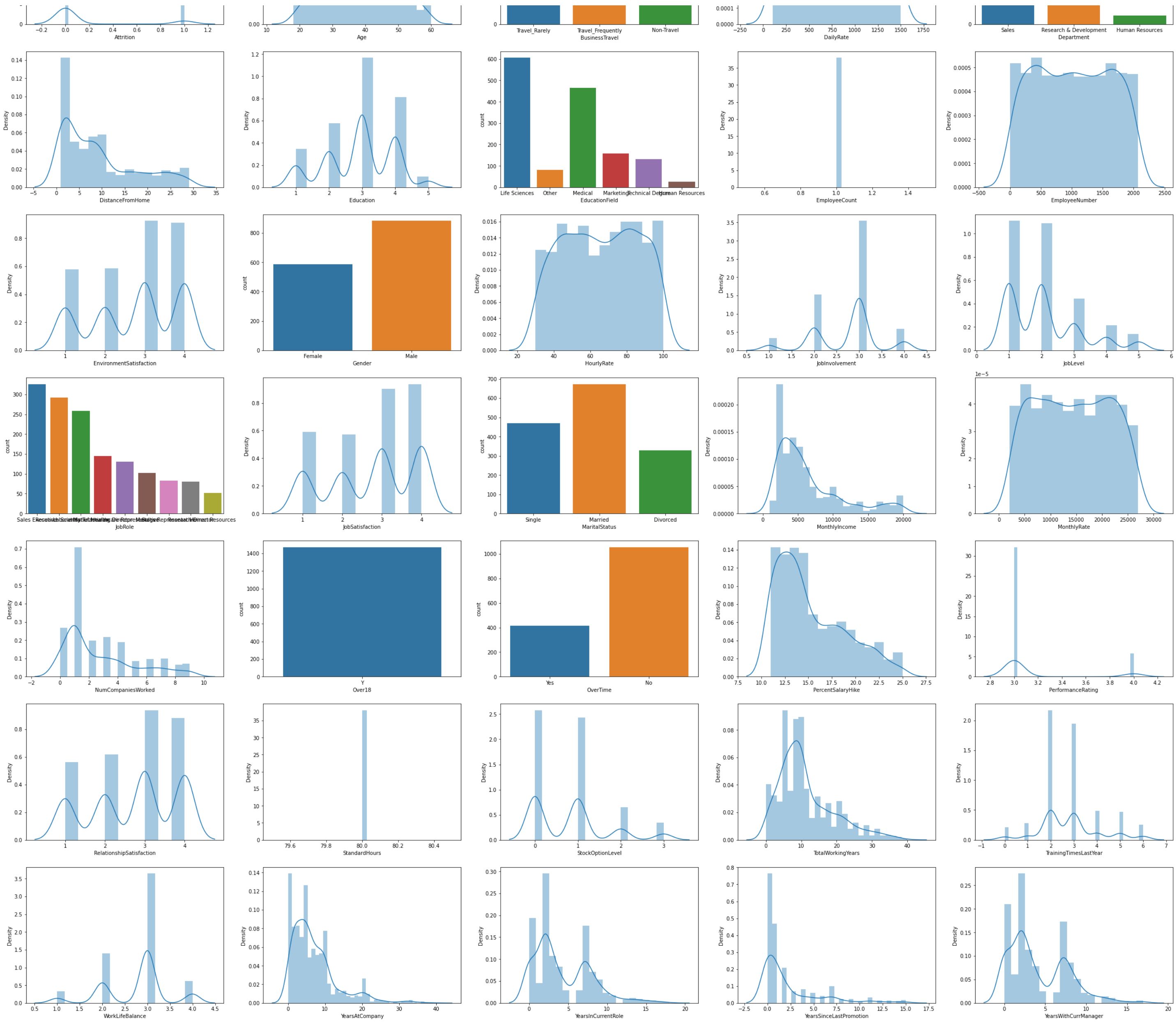
> # Exploratory Data Analysis

> # 1. Univariate Analysis

In [8]:

```python
x = 1
plt.figure(figsize=(40,40))
for i in A.columns:
    if A[i].dtypes == 'object':
        plt.subplot(7,5,x)
        sns.countplot(A[i])
        x = x + 1
    else:
        plt.subplot(7,5,x)
        sns.distplot(A[i])
        x = x+1
```





> **Above you can see the metrics of univariate Analysis. It Shows the over all data in the visuallized pictures. But afterall some of the features are overlapping to show the picture of data. So I take the step by step visualization to get the better idea. Some of the features are looking good so will take only which i think it will affect in our data.**

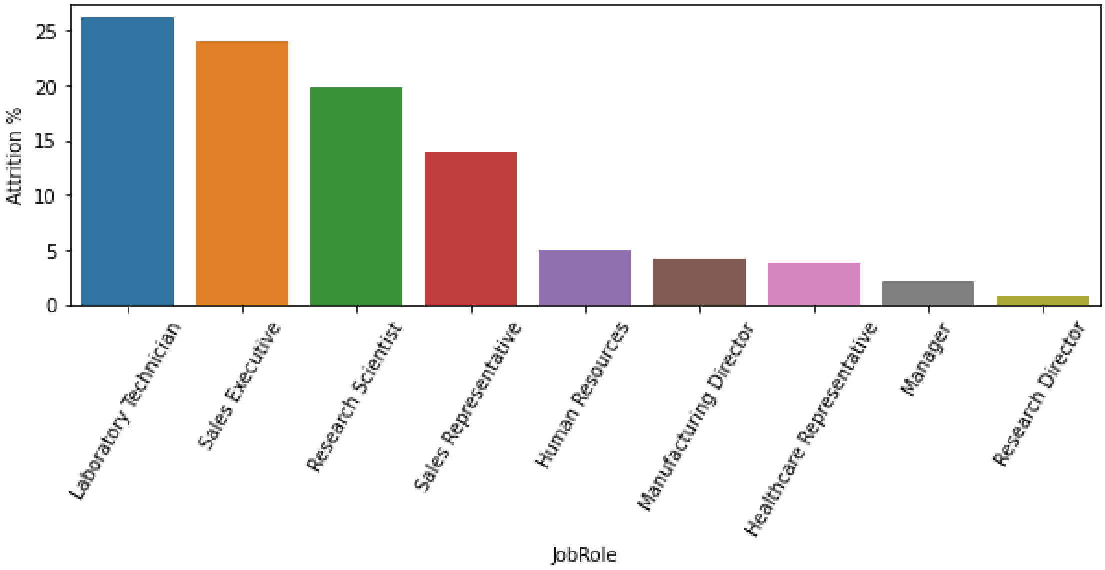
In [9]:

```
df = A[["JobRole", "Attrition"]] # Taken Two Features
JobRole_Att = df.groupby('JobRole').sum() # Sum of both the feature
Total_Sum_Att = JobRole_Att['Attrition'].sum()
JobRole_Att['JobRole'] = list(JobRole_Att.index) # Creating list to save index of the act
ual feature
JobRole_Att.index = range(len(JobRole_Att)) # Doing Reindexing
JobRole_Att['Attrition %'] = JobRole_Att['Attrition']*100/Total_Sum_Att
## Created columns for attrition percentage on the basis of jobRole.
plt.rcParams['figure.figsize'] = (10,3)
sns.barplot(data = JobRole_Att, x = 'JobRole', y = 'Attrition %', order=JobRole_Att.sort
_values('Attrition %',ascending = False).JobRole)
plt.xticks(rotation = 60)
```

Out[9]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(0, 0, 'Laboratory Technician'),
  Text(1, 0, 'Sales Executive'),
  Text(2, 0, 'Research Scientist'),
  Text(3, 0, 'Sales Representative'),
```

```
 Text(4, 0, 'Human Resources'),
 Text(5, 0, 'Manufacturing Director'),
 Text(6, 0, 'Healthcare Representative'),
 Text(7, 0, 'Manager'),
 Text(8, 0, 'Research Director')])
```



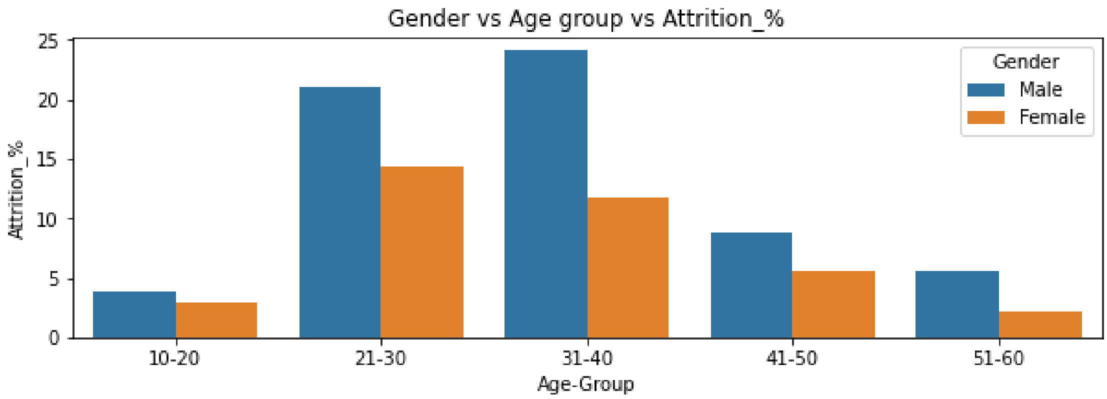
> **It appears that Attrition % is highest among Sales Executive (24%), closely followed by Laboratory (26%) Technician, followed by Research Scientist (19%). These three job roles have suffered more compared to other job roles. Lowest Attrition % is observed for the role of Research Director (1%). The general trend observed here is that bigger the role is in an organization, less are the chances of being released from the company.**

In [10]:

```
Age_Gender = A[["Age", "Gender", 'Attrition']] # Compare Age Gender basis attrition
grp_Age = [[10,20],[21,30],[31,40],[41,50],[51,60]] ## Taking the ages
Ages_Res = []
Total_Sum_Att = Age_Gender['Attrition'].sum() ## COntains sum for this latest DataFrame
for Grp in grp_Age: ## For Loop is running on the ages and
    Df = Age_Gender[(Age_Gender['Age'] >= Grp[0]) & (Age_Gender['Age'] <= Grp[1])]
    Df_Age = Df[["Age",'Attrition']].groupby('Age').sum()
    for Gen in ['Male', "Female"]: # Running Loop in gender basis
        Gen_Df = Df[Df['Gender']==Gen]
        Ages_Res.append([str(Grp[0])+'-'+str(Grp[1]), Gen_Df['Attrition'].sum()*100/Tota
l_Sum_Att, Gen])
Age_Grp_Df = pd.DataFrame(data=Ages_Res, columns=['Age-Group', "Attrition_%", 'Gender'])
sns.barplot(data=Age_Grp_Df, x = 'Age-Group', y = "Attrition_%", hue="Gender")
plt.title('Gender vs Age group vs Attrition_%')
```

Out[10]:

```
Text(0.5, 1.0, 'Gender vs Age group vs Attrition_%')
```



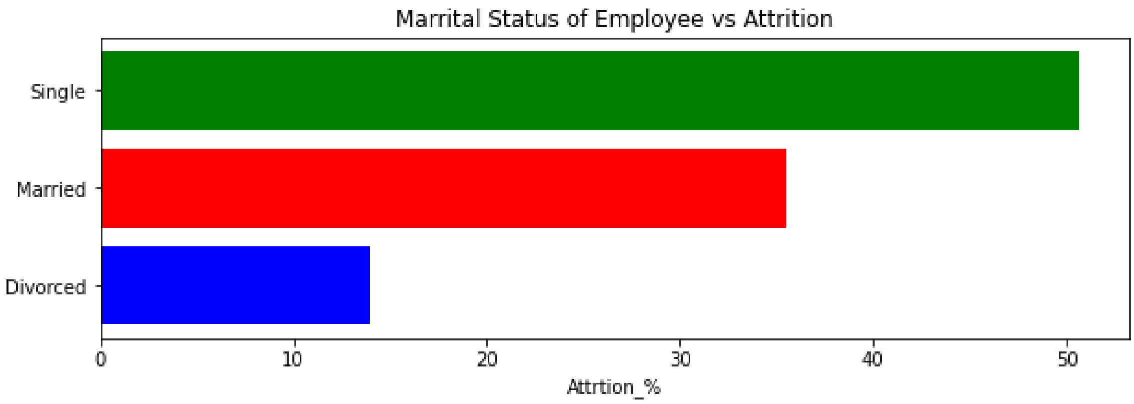
> **So, we see that the age-group of 21-30 suffers the highest attrition (about 23% in Male, 13% in Female), which makes sense since majority of them are freshers in company starting straight after finishing from the college. They are the dispensable resources due to lack of skills, and are let go first if the cash flow of the company starts falling due to external factors. For 10-20 age group its less because very less no. of people start their corporate career at the age of 19 or 20. For higher age group, the attrition % starts falling due to high amount of exposure, experience leads to firm grip in their positioning and role in the company.**

In [11]:

```
Marriage_data = A[['MaritalStatus', 'Attrition']]
Total_Sum_Att = Marriage_data['Attrition'].sum()
Married_Att = Marriage_data[['MaritalStatus','Attrition']].groupby("MaritalStatus").sum(
)*100/Total_Sum_Att
plt.barh(list(Married_Att.index), list(Married_Att['Attrition']), color = ['Blue', "Red"
,"Green" ])
plt.xlabel('Attrtion_%')
plt.title("Marrital Status of Employee vs Attrition")
```

Out[11]:

```
Text(0.5, 1.0, 'Marrital Status of Employee vs Attrition')
```



> **It appears that Bachelor Employees had the highest attrition rate while Divorced Employees had lowest. This is quite related to Age vs Attrition graph where we saw that age group of 21-30 were affected from job loss the most. Generally, these lower age groups fall under Bachelors category, so the higher attrition rate is justified due to lack of required skillsets and experience in Coporate World. Married and Divorced Employees belong to higher age groups who have plenty of corporate and business exposure, leading to lower attrition rate**

In [ ]:

```

```

In [ ]:

```

```

> `Define X and Y`

Categorical and Continous Separator

In [12]:

```
Y = A[["Attrition"]]
```

```python
Y = A[["Attrition"]]
X = A.drop(labels=["Attrition"],axis=1)
cat = []
con = []
for i in X.columns:
    if(X[i].dtypes == "object"):
        cat.append(i)
    else:
        con.append(i)
```

> ## 2. Anova(Analysis of Variance)Find relationship Bw Cat and Con

In [13]:

```python
def ANOVA(df,cat,con):
    from pandas import DataFrame
    from statsmodels.formula.api import ols
    rel = con + " ~ " + cat
    model = ols(rel,df).fit()
    from statsmodels.stats.anova import anova_lm
    anova_results = anova_lm(model)
    Q = DataFrame(anova_results)
    a = Q['PR(>F)'][cat]
    return round(a,3)
```

In [14]:

```python
imp_con_cols = []
for i in con:
    print("--------Attrition vs",i,"----------")
    x = ANOVA(A,"Attrition",i)
    print(x)
    if(x < 0.05):
        imp_con_cols.append(i)
```

```
--------Attrition vs Age ----------
0.0
--------Attrition vs DailyRate ----------
0.03
--------Attrition vs DistanceFromHome ----------
0.003
--------Attrition vs Education ----------
0.229
--------Attrition vs EmployeeCount ----------
0.403
--------Attrition vs EmployeeNumber ----------
0.685
--------Attrition vs EnvironmentSatisfaction ----------
0.0
--------Attrition vs HourlyRate ----------
0.793
--------Attrition vs JobInvolvement ----------
0.0
--------Attrition vs JobLevel ----------
0.0
--------Attrition vs JobSatisfaction ----------
0.0
--------Attrition vs MonthlyIncome ----------
0.0
--------Attrition vs MonthlyRate ----------
0.561
--------Attrition vs NumCompaniesWorked ----------
0.096
--------Attrition vs PercentSalaryHike ----------
0.606
--------Attrition vs PerformanceRating ----------
0.012
```

```
0.912
--------Attrition vs RelationshipSatisfaction ----------
0.079
--------Attrition vs StandardHours ----------
0.212
--------Attrition vs StockOptionLevel ----------
0.0
--------Attrition vs TotalWorkingYears ----------
0.0
--------Attrition vs TrainingTimesLastYear ----------
0.023
--------Attrition vs WorkLifeBalance ----------
0.014
--------Attrition vs YearsAtCompany ----------
0.0
--------Attrition vs YearsInCurrentRole ----------
0.0
--------Attrition vs YearsSinceLastPromotion ----------
0.206
--------Attrition vs YearsWithCurrManager ----------
0.0
```

Created new list for saving the important columns, Using anova

In [15]:

```
imp_con_cols
```

Out[15]:

```
['Age',
 'DailyRate',
 'DistanceFromHome',
 'EnvironmentSatisfaction',
 'JobInvolvement',
 'JobLevel',
 'JobSatisfaction',
 'MonthlyIncome',
 'StockOptionLevel',
 'TotalWorkingYears',
 'TrainingTimesLastYear',
 'WorkLifeBalance',
 'YearsAtCompany',
 'YearsInCurrentRole',
 'YearsWithCurrManager']
```

> ## 3. Crosstab for Find the relation bw cat vs cat

In [16]:

```
for i in cat:
    print("----------Attrition vs",i,"----------\n")
    print(pd.crosstab(A.Attrition,A[i]))
    print("\n")
```

```
----------Attrition vs BusinessTravel ----------

BusinessTravel  Non-Travel  Travel_Frequently  Travel_Rarely
Attrition
0                      138                208            887
1                       12                 69            156


----------Attrition vs Department ----------

Department   Human Resources  Research & Development  Sales
```

```
Attrition
0                        51                      828    354
1                        12                      133     92


----------Attrition vs EducationField ----------

EducationField  Human Resources  Life Sciences  Marketing  Medical  Other  \
Attrition
0                            20            517        124      401     71
1                             7             89         35       63     11

EducationField  Technical Degree
Attrition
0                            100
1                             32


----------Attrition vs Gender ----------

Gender     Female  Male
Attrition
0             501   732
1              87   150


----------Attrition vs JobRole ----------

JobRole    Healthcare Representative  Human Resources  Laboratory Technician  \
Attrition
0                                122               40                    197
1                                  9               12                     62

JobRole    Manager  Manufacturing Director  Research Director  \
Attrition
0               97                     135                 78
1                5                      10                  2

JobRole    Research Scientist  Sales Executive  Sales Representative
Attrition
0                         245              269                    50
1                          47               57                    33


----------Attrition vs MaritalStatus ----------

MaritalStatus  Divorced  Married  Single
Attrition
0                   294      589     350
1                    33       84     120


----------Attrition vs Over18 ----------

Over18        Y
Attrition
0          1233
1           237


----------Attrition vs OverTime ----------

OverTime    No  Yes
Attrition
0          944  289
1          110  127
```

> `4. Chisquare Contingency`
>
> **If the pvalue is very near to zero means it will be count as a best predictor**

In [17]:

```
from scipy.stats import chi2_contingency
for i in cat:
    ct = pd.crosstab(A.Attrition,A[i],normalize="index")
    a,b,c,d = chi2_contingency(ct)
    print(i,round(b,4))
```

```
BusinessTravel 0.9712
Department 0.9871
EducationField 1.0
Gender 0.163
JobRole 1.0
MaritalStatus 0.9479
Over18 1.0
OverTime 0.3101
```

Extend two columns which is fine to get a prediction

Extend is used to add columns in alphabetcial order

In [18]:

```
imp_con_cols.extend(["OverTime","Gender"])
imp_con_cols
```

Out[18]:

```
['Age',
 'DailyRate',
 'DistanceFromHome',
 'EnvironmentSatisfaction',
 'JobInvolvement',
 'JobLevel',
 'JobSatisfaction',
 'MonthlyIncome',
 'StockOptionLevel',
 'TotalWorkingYears',
 'TrainingTimesLastYear',
 'WorkLifeBalance',
 'YearsAtCompany',
 'YearsInCurrentRole',
 'YearsWithCurrManager',
 'OverTime',
 'Gender']
```

In [ ]:

```

```

> `Preprocessing`

In [19]:

```
def preprocessing(X):
    import pandas as pd
    cat = []
    con = []
    for i in X.columns:
```

```
        if(X[i].dtypes =='object'):
            cat.append(i)
        else:
            con.append(i)

    X1 = pd.get_dummies(X[cat])
    from sklearn.preprocessing import StandardScaler
    ss = StandardScaler()
    X2 = pd.DataFrame(ss.fit_transform(X[con]),columns=con)

    X3 = X2.join(X1)
    return X3
```

In [20]:

```
X[imp_con_cols]
```

Out[20]:

	Age	DailyRate	DistanceFromHome	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome
0	41	1102	1	2	3	2	4	599
1	49	279	8	3	2	2	2	513
2	37	1373	2	4	2	1	3	209
3	33	1392	3	4	3	1	3	290
4	27	591	2	1	3	1	2	346
...	...	...	...	...	...	...	...	..
1465	36	884	23	3	4	2	4	257
1466	39	613	6	4	2	3	1	999
1467	27	155	4	2	4	2	2	614
1468	49	1023	2	4	2	2	2	539
1469	34	628	8	2	4	2	3	440

1470 rows × 17 columns

In [21]:

```
from PM8 import preprocessing
Xnew = preprocessing(X[imp_con_cols])
Xnew.head(4)
```

Out[21]:

	Age	DailyRate	DistanceFromHome	EnvironmentSatisfaction	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncon
0	0.446350	0.742527	-1.010909	-0.660531	0.379672	-0.057788	1.153254	-0.1083
1	1.322365	-1.297775	-0.147150	0.254625	-1.026167	-0.057788	-0.660853	-0.2917
2	0.008343	1.414363	-0.887515	1.169781	-1.026167	-0.961486	0.246200	-0.9376
3	-0.429664	1.461466	-0.764121	1.169781	0.379672	-0.961486	0.246200	-0.7636

> `Split the data in training and testing set`

In [22]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(Xnew,Y,test_size=0.2,random_state=21)
```

> # Create classification models:
>
> 1. Logistic Regression
> 2. DTC
> 3. RF
> 4. ADB

importing all algos which i am gonna use there

In [23]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
```

In [24]:

```
def model_basic(mobj):
    model = mobj.fit(xtrain,ytrain)
    pred_tr = model.predict(xtrain)
    tr_acc = accuracy_score(ytrain,pred_tr)
    pred_ts = model.predict(xtest)
    ts_acc = accuracy_score(ytest,pred_ts)
    return round(tr_acc,3),round(ts_acc,3),mobj
```

In [25]:

```
def grid_cvtune(mobj,xtrain,ytrain, tp, cv):
    from sklearn.model_selection import GridSearchCV
    cv = GridSearchCV(mobj,tp, scoring="accuracy", cv = cv)
    cvmodel = cv.fit(xtrain,ytrain)
    bestval =cvmodel.best_params_
    return bestval
```

In [26]:

```
def model(mobj):
    try:
        model = mobj.fit(xtrain,ytrain)
        pred_tr = model.predict(xtrain)
        tr_acc = accuracy_score(ytrain,pred_tr)
        pred_ts = model.predict(xtest)
        ts_acc = accuracy_score(ytest,pred_ts)
        p = pd.DataFrame([mobj,round(tr_acc,3),round(ts_acc,3)]).T
        p.columns = ["Algorithm", "Train Acc", "Test Acc"]
    except TypeError:
        mobj = str(mobj)
        p = pd.DataFrame([mobj,round(tr_acc,3),round(ts_acc,3)]).T
        p.columns = ["Algorithm", "Train Acc", "Test Acc"]
    return p
```

> # 1. Logistic Regression

In [27]:

```
lr = LogisticRegression()
lr_model = model(lr)
```

In [28]:

```
lr_model
```

Out[28]:

	Algorithm	Train Acc	Test Acc
0	LogisticRegression()	0.862	0.878

2. Decision Tree

In [29]:

```
Q = []
for i in range(2,20):
    dtc = DecisionTreeClassifier(random_state=21,max_depth=i)
    Q.append(model_basic(dtc))
Q
```

Out[29]:

```
[(0.843, 0.85, DecisionTreeClassifier(max_depth=2, random_state=21)),
 (0.861, 0.864, DecisionTreeClassifier(max_depth=3, random_state=21)),
 (0.874, 0.844, DecisionTreeClassifier(max_depth=4, random_state=21)),
 (0.895, 0.847, DecisionTreeClassifier(max_depth=5, random_state=21)),
 (0.914, 0.854, DecisionTreeClassifier(max_depth=6, random_state=21)),
 (0.929, 0.84, DecisionTreeClassifier(max_depth=7, random_state=21)),
 (0.94, 0.827, DecisionTreeClassifier(max_depth=8, random_state=21)),
 (0.952, 0.786, DecisionTreeClassifier(max_depth=9, random_state=21)),
 (0.969, 0.82, DecisionTreeClassifier(max_depth=10, random_state=21)),
 (0.982, 0.806, DecisionTreeClassifier(max_depth=11, random_state=21)),
 (0.99, 0.806, DecisionTreeClassifier(max_depth=12, random_state=21)),
 (0.995, 0.793, DecisionTreeClassifier(max_depth=13, random_state=21)),
 (0.998, 0.796, DecisionTreeClassifier(max_depth=14, random_state=21)),
 (0.999, 0.796, DecisionTreeClassifier(max_depth=15, random_state=21)),
 (1.0, 0.796, DecisionTreeClassifier(max_depth=16, random_state=21)),
 (1.0, 0.796, DecisionTreeClassifier(max_depth=17, random_state=21)),
 (1.0, 0.796, DecisionTreeClassifier(max_depth=18, random_state=21)),
 (1.0, 0.796, DecisionTreeClassifier(max_depth=19, random_state=21))]
```

In [30]:

```
Q = []
for i in range(2,20):
    dtc = DecisionTreeClassifier(random_state=21,min_samples_leaf=i)
    Q.append(model(dtc))
Q
```

Out[30]:

```
[                                           Algorithm Train Acc Test Acc
 0  DecisionTreeClassifier(min_samples_leaf=2, ran...      0.966     0.827,
                                            Algorithm Train Acc Test Acc
 0  DecisionTreeClassifier(min_samples_leaf=3, ran...      0.945     0.827,
                                            Algorithm Train Acc Test Acc
 0  DecisionTreeClassifier(min_samples_leaf=4, ran...      0.929      0.83,
                                            Algorithm Train Acc Test Acc
 0  DecisionTreeClassifier(min_samples_leaf=5, ran...      0.918     0.796,
                                            Algorithm Train Acc Test Acc
 0  DecisionTreeClassifier(min_samples_leaf=6, ran...      0.906     0.813,
```

```
0   DecisionTreeClassifier(min_samples_leaf=6, ran...      0.906     0.813,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=7, ran...      0.903     0.796,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=8, ran...      0.901     0.806,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=9, ran...       0.89     0.813,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=10, ra...      0.883     0.816,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=11, ra...      0.883     0.816,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=12, ra...      0.879     0.813,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=13, ra...      0.877     0.823,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=14, ra...      0.874      0.83,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=15, ra...      0.872     0.833,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=16, ra...       0.87      0.83,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=17, ra...      0.869      0.83,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=18, ra...      0.869      0.83,
                                          Algorithm Train Acc Test Acc
0   DecisionTreeClassifier(min_samples_leaf=19, ra...      0.869      0.83]
```

min sample split pruning params

In [31]:

```
tp = {"min_samples_split":range(1,20,1)}
grid_cvtune(dtc,xtrain,ytrain, tp, 5)
```

Out[31]:

```
{'min_samples_split': 2}
```

In [32]:

```
dtc = DecisionTreeClassifier(random_state=21,min_samples_split=18)
dtc_model = model(dtc)
dtc_model
```

Out[32]:

	Algorithm	Train Acc	Test Acc
0	**DecisionTreeClassifier(min_samples_split=18, r...**	**0.918**	**0.816**

max depth pruning params

In [33]:

```
tp = {"max_depth":range(1,20,1)}
grid_cvtune(dtc,xtrain,ytrain, tp, 5)
```

Out[33]:

```
{'max_depth': 2}
```

In [34]:

```
dtc = DecisionTreeClassifier(random_state=21,max_depth=2)
model(dtc)
```

Out[34]:

Algorithm	Train Acc	Test Acc

	Algorithm	Train Acc	Test Acc
0	DecisionTreeClassifier(max_depth=2, random_sta...	0.843	0.85

3. Random Forest

In [35]:

```
rf = RandomForestClassifier(random_state=21,n_estimators=11)
tp = {"n_estimators":range(1,20,1)}
grid_cvtune(rf,xtrain,ytrain, tp, 5)
```

Out[35]:

```
{'n_estimators': 18}
```

In [36]:

```
rf = RandomForestClassifier(random_state=21,n_estimators=18)
rf_model = model(rf)
```

In [37]:

```
rf_model
```

Out[37]:

	Algorithm	Train Acc	Test Acc
0	RandomForestClassifier(n_estimators=18, random...	0.993	0.867

4. Adaboost

In [38]:

```
adb = AdaBoostClassifier(dtc, n_estimators=10, random_state=42)
tp = {"n_estimators":range(1,20,1)}
grid_cvtune(adb,xtrain,ytrain, tp, 5)
```

Out[38]:

```
{'n_estimators': 7}
```

In [39]:

```
adb = AdaBoostClassifier(dtc, n_estimators=7, random_state=42)
adb_model = model(adb)
adb_model
```

Out[39]:

	Algorithm	Train Acc	Test Acc
0	AdaBoostClassifier(base_estimator=DecisionTree...	0.879	0.847

In [40]:

```
all_algo = pd.concat([lr_model, rf_model, dtc_model, adb_model ])
all_algo.index = range(4)
```

In [41]:

```
all_algo
```

Out[41]:

	Algorithm	Train Acc	Test Acc
0	LogisticRegression()	0.862	0.878
1	RandomForestClassifier(n_estimators=18, random...	0.993	0.867
2	DecisionTreeClassifier(min_samples_split=18, r...	0.918	0.816
3	AdaBoostClassifier(base_estimator=DecisionTree...	0.879	0.847

In these all algorithmns we are getting only 99% training accuracy and 88% testing accuracy. By using Random Forest Classifier on the other hand we got logistic regression is also giving good accuracy.

In [45]:

```
!pip install -U nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\user\appdata\roaming\python\python38
\site-packages (6.5.3)
Requirement already satisfied: traitlets>=5.0 in c:\users\user\anaconda3\lib\site-package
s (from nbconvert) (5.0.5)
Requirement already satisfied: packaging in c:\users\user\anaconda3\lib\site-packages (fr
om nbconvert) (20.9)
Requirement already satisfied: bleach in c:\users\user\anaconda3\lib\site-packages (from
nbconvert) (3.3.0)
Requirement already satisfied: defusedxml in c:\users\user\anaconda3\lib\site-packages (f
rom nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=3.0 in c:\users\user\appdata\roaming\python\python
38\site-packages (from nbconvert) (3.1.2)
Requirement already satisfied: pygments>=2.4.1 in c:\users\user\anaconda3\lib\site-packag
es (from nbconvert) (2.8.1)
Requirement already satisfied: jupyter-core>=4.7 in c:\users\user\anaconda3\lib\site-pack
ages (from nbconvert) (4.7.1)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\user\anaconda3\lib\site-p
ackages (from nbconvert) (1.4.3)
Requirement already satisfied: tinycss2 in c:\users\user\appdata\roaming\python\python38\
site-packages (from nbconvert) (1.1.1)
Requirement already satisfied: jupyterlab-pygments in c:\users\user\anaconda3\lib\site-pa
ckages (from nbconvert) (0.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\user\appdata\roaming\python\py
thon38\site-packages (from nbconvert) (2.1.1)
Requirement already satisfied: lxml in c:\users\user\anaconda3\lib\site-packages (from nb
convert) (4.6.3)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\user\anaconda3\lib\site-packag
es (from nbconvert) (0.5.3)
Requirement already satisfied: beautifulsoup4 in c:\users\user\anaconda3\lib\site-package
s (from nbconvert) (4.9.3)
Requirement already satisfied: nbformat>=5.1 in c:\users\user\anaconda3\lib\site-packages
(from nbconvert) (5.1.3)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\user\anaconda3\lib\site-pac
kages (from nbconvert) (0.3)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\user\anaconda3\lib\site-pack
ages (from nbconvert) (0.8.4)
Requirement already satisfied: pywin32>=1.0 in c:\users\user\anaconda3\lib\site-packages
(from jupyter-core>=4.7->nbconvert) (227)
Requirement already satisfied: nest-asyncio in c:\users\user\anaconda3\lib\site-packages
(from nbclient>=0.5.0->nbconvert) (1.5.1)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\user\anaconda3\lib\site-
packages (from nbclient>=0.5.0->nbconvert) (6.1.12)
Requirement already satisfied: async-generator in c:\users\user\anaconda3\lib\site-packag
es (from nbclient>=0.5.0->nbconvert) (1.10)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\user\anaconda3\lib\site-p
ackages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (2.8.1)
Requirement already satisfied: pyzmq>=13 in c:\users\user\anaconda3\lib\site-packages (fr
om jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (20.0.0)
Requirement already satisfied: tornado>=4.1 in c:\users\user\anaconda3\lib\site-packages
(from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert) (6.1)
Requirement already satisfied: ipython-genutils in c:\users\user\anaconda3\lib\site-packa
ges (from nbformat>=5.1->nbconvert) (0.2.0)
```

```
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\users\user\anaconda3\lib\sit
e-packages (from nbformat>=5.1->nbconvert) (3.2.0)
Requirement already satisfied: six>=1.11.0 in c:\users\user\anaconda3\lib\site-packages (
from jsonschema!=2.5.0,>=2.4->nbformat>=5.1->nbconvert) (1.15.0)
Requirement already satisfied: setuptools in c:\users\user\anaconda3\lib\site-packages (f
rom jsonschema!=2.5.0,>=2.4->nbformat>=5.1->nbconvert) (52.0.0.post20210125)
Requirement already satisfied: attrs>=17.4.0 in c:\users\user\anaconda3\lib\site-packages
(from jsonschema!=2.5.0,>=2.4->nbformat>=5.1->nbconvert) (20.3.0)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\users\user\anaconda3\lib\site-pac
kages (from jsonschema!=2.5.0,>=2.4->nbformat>=5.1->nbconvert) (0.17.3)
Requirement already satisfied: soupsieve>1.2 in c:\users\user\anaconda3\lib\site-packages
(from beautifulsoup4->nbconvert) (2.2.1)
Requirement already satisfied: webencodings in c:\users\user\anaconda3\lib\site-packages
(from bleach->nbconvert) (0.5.1)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\user\anaconda3\lib\site-packa
ges (from packaging->nbconvert) (2.4.7)
```

In [ ]: