EMPLOYEE ATTRITION RATE ANALYSIS

DOMAIN – HR

Sowmya Siddanki
Project_0_8

Overview

THE MAIN OBJECTIVE OF THIS PROJECT IS TO IDENTIFY THE FACTORS INVOLVED IN ATTRITION OF EMPLOYEES FROM THE COMPANY>

Goals

1. This model and exploratory data analysis helps in identifying the factors for attrition
2. Help the HR team to find the reasons for attrition of employees.

Specifications

In this project instead of tableau I used a machine learning model to find the accuracy of the data.

Milestones

I. RadndomForest classifier

Random forest classifier has given us an accuracy of 90 percent.

IMPORTING THE REQUIRED MODULES

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,jaccard_score,f1_score,log_loss,confusion_matrix
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score, confusion_matrix, classification_report
```

READING THE DATASET AND PRINTING HEAD TO STUDY THE FEATURES OR ATTRIBUTES

```
df=pd.read_csv('/content/greendestination.csv')
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1
2	[illegible]	[illegible]	[illegible]	[illegible]	Research &	[illegible]	[illegible]	[illegible]	[illegible]
3	33	No	Travel_Frequently	1392	Research &	3	4	Life Sciences	1

5 rows × 35 columns

GETTING THE SHAPE OF THE DATASET

```
df.shape
```

```
(1470, 35)
```

THE GIVEN DATASET HAS 1470 EMPLOYEES AND 34 FEATURES WITH ONE TARGET VARIABLE "ATTRITION"

PRINTING THE INFO TO KNOW ABOUT THE INFORMATION OF THE DATASET LIKE KNOWING THE NULL COUNT AND DATATYPES OF THE ATTRIBUTES

```
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
```

THERE ARE NO NULL VALUES IN GIVEN DATA BUT THERE ARE MANY CATEGORICAL VALUES

CONVERTING THE DATA TO SUITABLE FORM USING DISPLAY METHODS

```
pd.options.display.float_format = '{:,.2f}'.format
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
df.describe(include='all')
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Employ
count	1,470.00	1470	1470	1,470.00	1470	1,470.00	1,470.00	1470	
unique	NaN	2	3	NaN	3	NaN	NaN	6	
top	NaN	No	Travel_Rarely	NaN	Research & Development	NaN	NaN	Life Sciences	
freq	NaN	1233	1043	NaN	961	NaN	NaN	606	
mean	36.92	NaN	NaN	802.49	NaN	9.19	2.91	NaN	
std	9.14	NaN	NaN	403.51	NaN	8.11	1.02	NaN	
min	18.00	NaN	NaN	102.00	NaN	1.00	1.00	NaN	
25%	30.00	NaN	NaN	465.00	NaN	2.00	2.00	NaN	
50%	36.00	NaN	NaN	802.00	NaN	7.00	3.00	NaN	
75%	43.00	NaN	NaN	1,157.00	NaN	14.00	4.00	NaN	
max	60.00	NaN	NaN	1,499.00	NaN	29.00	5.00	NaN	

DROPPING THE COLUMNS {EmployeeCount', 'EmployeeNumber', 'StandardHours','Over18} ##THOSE COLUMNS HAS SINGLE VALUE WHICH IS NOT USEFUL FOR OUR ANALYSIS

```
drop = ['EmployeeCount', 'EmployeeNumber', 'StandardHours','Over18']
df_drop = df.drop(drop, axis=1)
df_drop.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSat
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	

NOW SELECT THE TARGET ATTRITION AND CONVERT THE CATEGORICAL ATTRIBUTE TO NUMERICAL

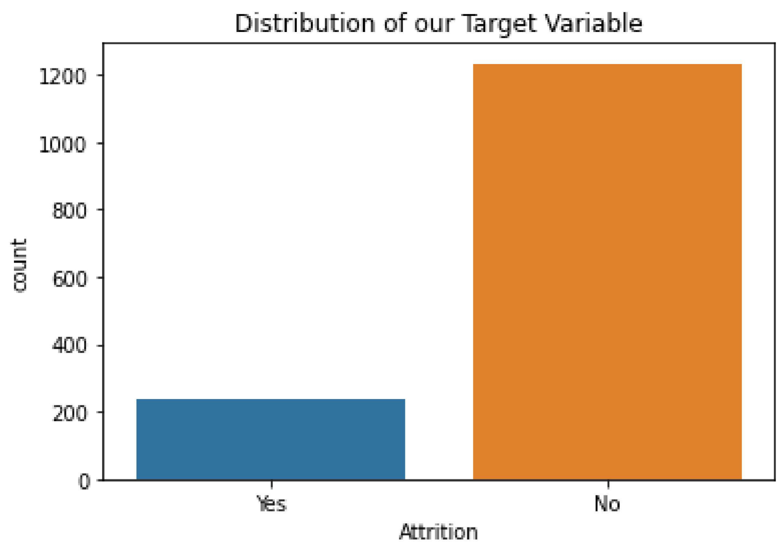
```
df_drop['target'] = df_drop['Attrition'].replace({'Yes':1,'No':0})
df_drop = df_drop.drop('Attrition',axis=1)
df_drop.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EnvironmentSatisfaction
0	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	2
1	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	3
2	37	Travel_Rarely	1373	Research & Development	2	2	Other	4
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	4
4	27	Travel_Rarely	591	Research & Development	2	1	Medical	1

PLOT SOME KINDS OF PLOTS TO LOOK THE VARIATIONS AND GAIN INSIGHTS FROM THE DATA

```
sns.countplot(x='Attrition',data=df)
```

```
plt.title('Distribution of our Target Variable')
plt.show()
```



DIVIDE THE COLUMNS TO NUMERICAL AND CATEGORICAL

```
num_cols = df_drop.select_dtypes(include=['float64', 'int64']).columns.tolist()
obj_cols = df_drop.select_dtypes(include=['object']).columns.tolist()
obj_cols
```

```
['BusinessTravel',
 'Department',
 'EducationField',
 'Gender',
 'JobRole',
 'MaritalStatus',
 'OverTime']
```

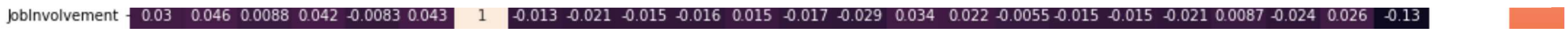
PLOT THE HEATMAP TO SEE THE CORRELATION IN NUMERICAL ATTRIBUTES

```
plt.figure(figsize=(20,20))
sns.heatmap(df_drop[num_cols].corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe77820efd0>
```

*DROP THE JOBLEVEL *

```
df_drop = df_drop.drop('JobLevel',axis=1)
```



GETTING THE CORRELATION OF TARGET WITH OTHER ATTRIBUTES

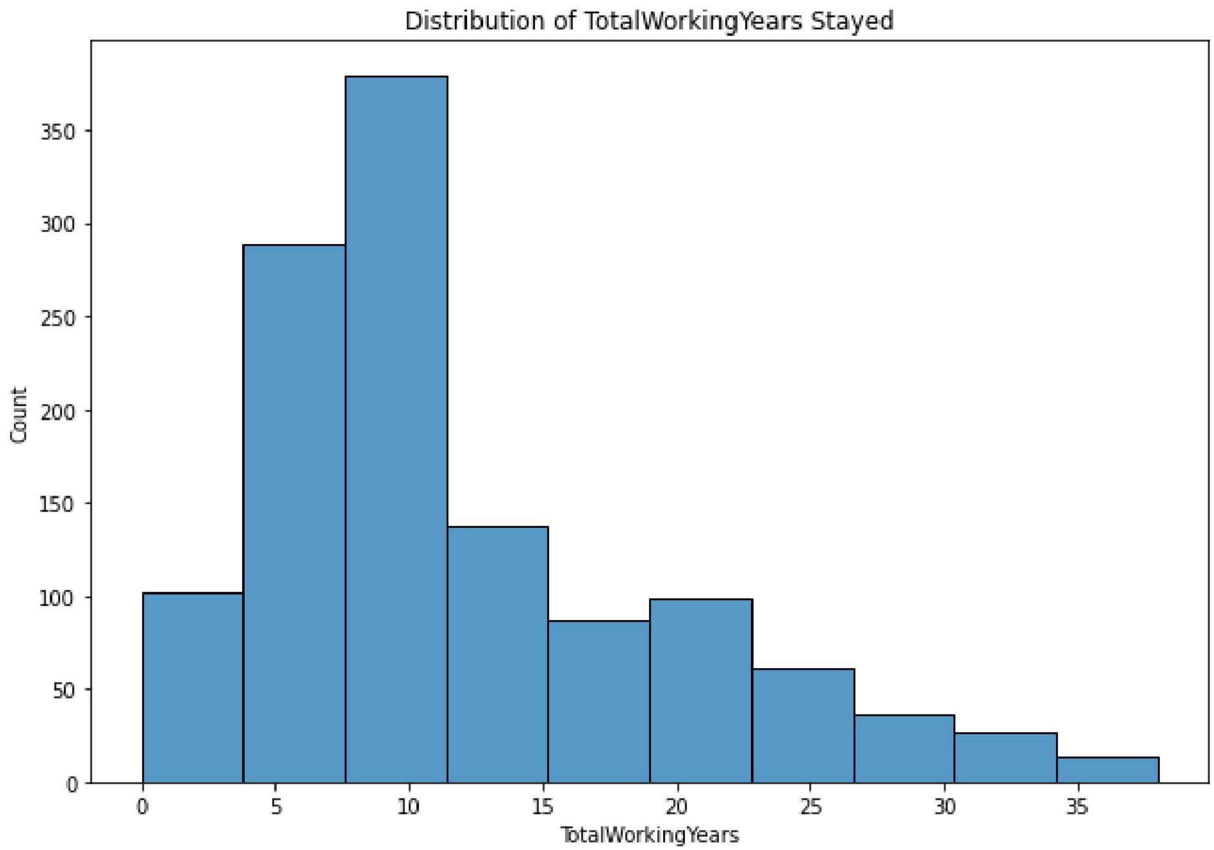
```
df_drop.corr()['target'].sort_values()
```

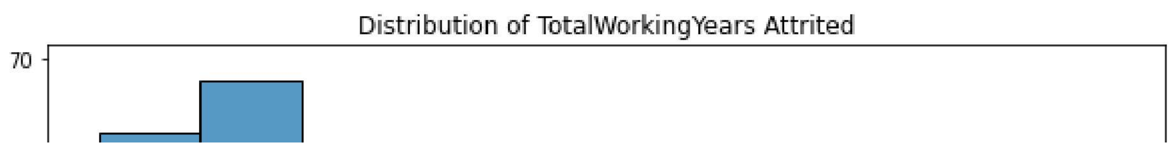
```
TotalWorkingYears          -0.17
YearsInCurrentRole         -0.16
MonthlyIncome              -0.16
Age                        -0.16
YearsWithCurrManager       -0.16
StockOptionLevel           -0.14
YearsAtCompany             -0.13
JobInvolvement             -0.13
JobSatisfaction            -0.10
EnvironmentSatisfaction    -0.10
WorkLifeBalance            -0.06
TrainingTimesLastYear      -0.06
DailyRate                  -0.06
RelationshipSatisfaction   -0.05
YearsSinceLastPromotion    -0.03
Education                  -0.03
PercentSalaryHike          -0.01
HourlyRate                 -0.01
PerformanceRating           0.00
MonthlyRate                 0.02
```

```
NumCompaniesWorked          0.04
DistanceFromHome            0.08
target                      1.00
Name: target, dtype: float64
```

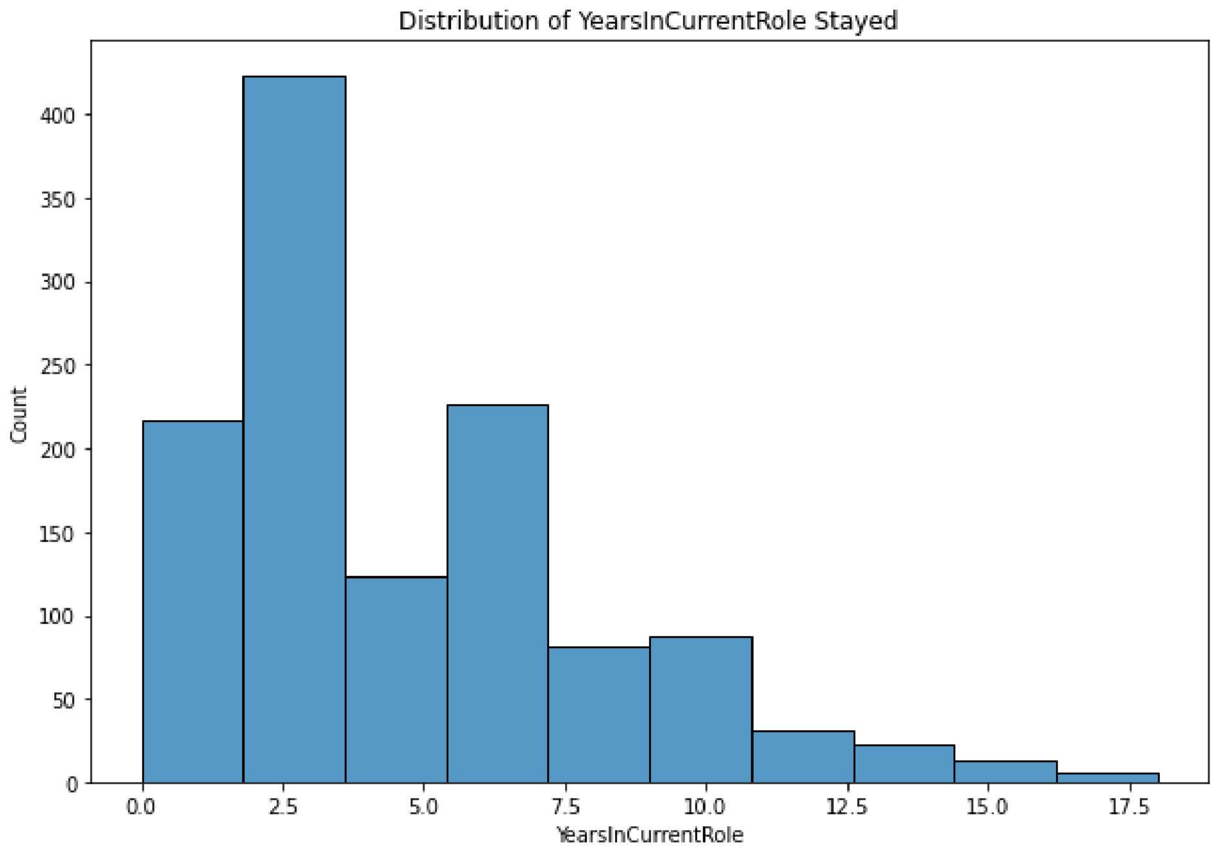
NOW LETS VIEW SOME DISTRIBUTIONS WITH SOME FEATURES TO OUR CORRELATION OF TARGET VARIABLE

```
def plot_dist(col,data=df):
    plt.figure(figsize=(10,15))
    plt.subplot(2,1,1)
    sns.histplot(data=data[data['Attrition'] == 'No'],x=col, bins=10)
    plt.title(f'Distribution of {col} Stayed')
    plt.subplot(2,1,2)
    sns.histplot(data=data[data['Attrition'] == 'Yes'],x=col, bins=10)
    plt.title(f'Distribution of {col} Attrited')
    plt.plot()
plot_dist('TotalWorkingYears')
```

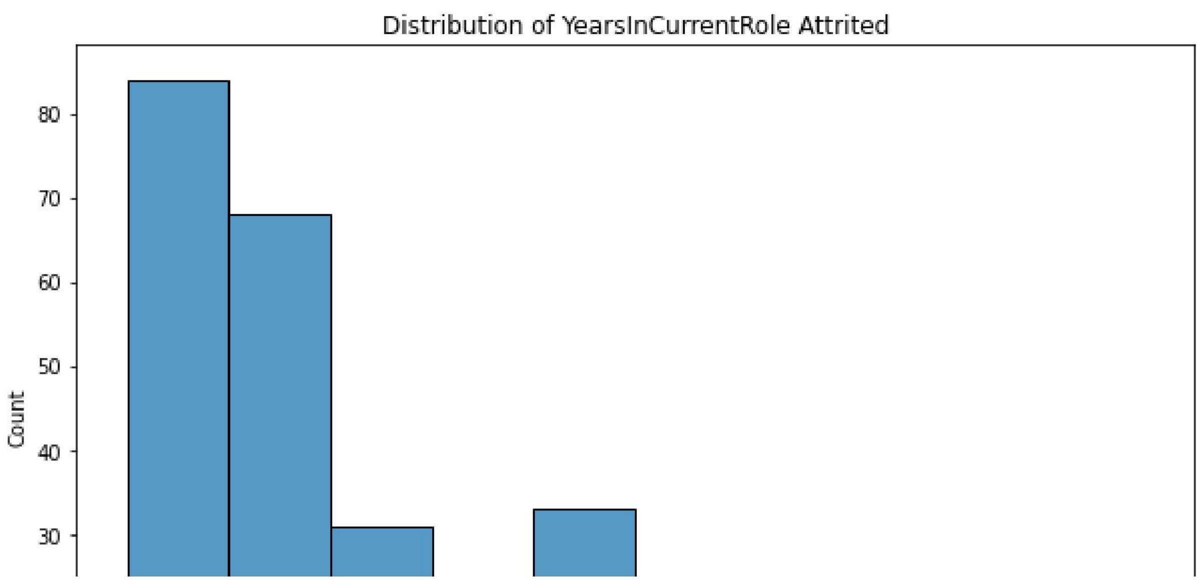





```
plot_dist('YearsInCurrentRole')
```

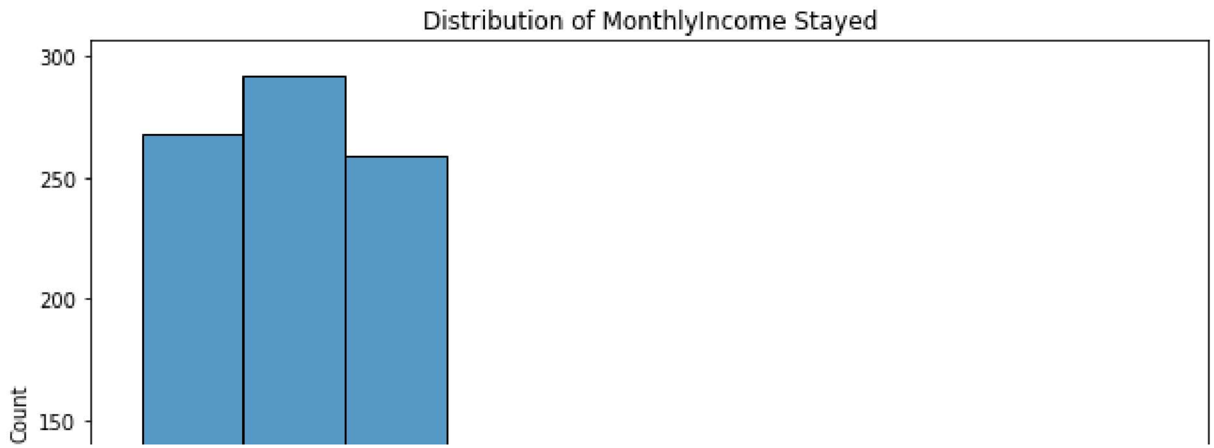


Distribution of YearsInCurrentRole Stayed
Count
YearsInCurrentRole



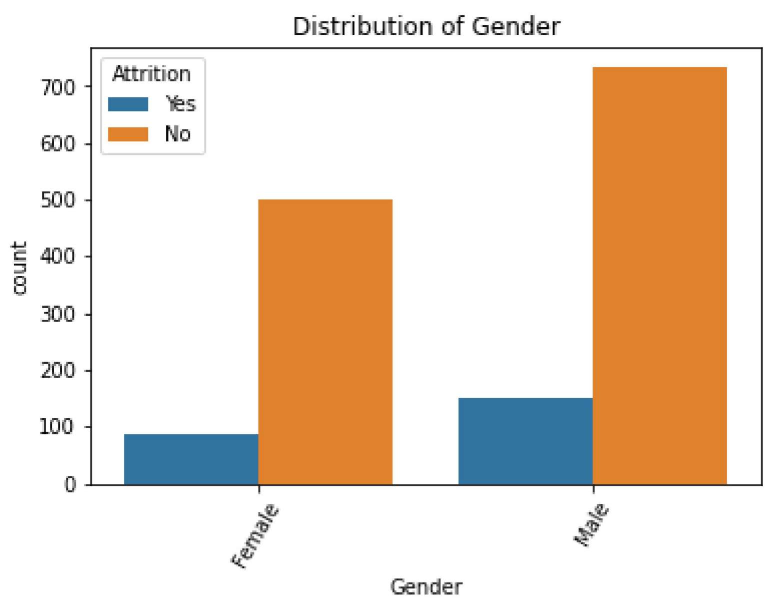
Distribution of YearsInCurrentRole Attrited
Count

```
plot_dist('MonthlyIncome')
```

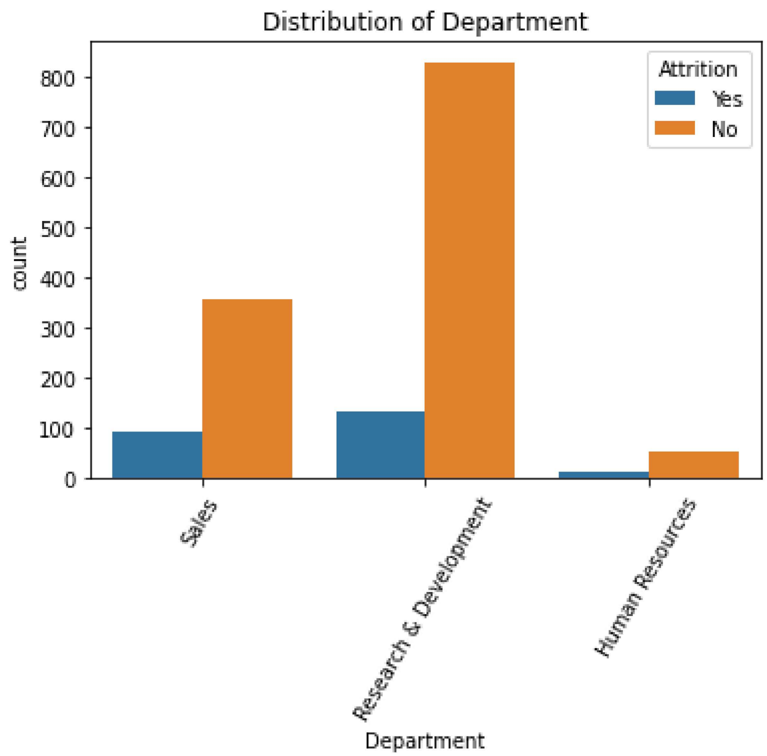



```
def plot_count(col,data=df):
    sns.countplot(x=col, data=data,hue='Attrition')
    plt.xticks(rotation=60)
    plt.title(f'Distribution of {col}')
    plt.show()
```

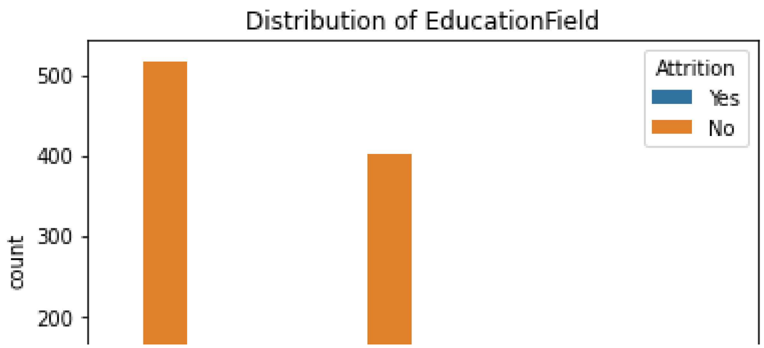
```
plot_count('Gender')
```



```
plot_count('Department')
```




```
plot_count('EducationField')
```



NOW LETS TRAIN AND TEST THE DATA

```
df_enc = pd.get_dummies(df_drop, columns=obj_cols,drop_first=True)
df_enc.head()
```

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobSatisfaction	M
0	41	1102	1	2	2	94	3	4	
1	49	279	8	1	3	61	2	2	
2	37	1373	2	2	4	92	2	3	
3	33	1392	3	4	4	56	3	3	
4	27	591	2	1	1	40	3	2	

```
y = df_enc['target']
X = df_enc.drop('target',axis=1)
y.value_counts()
```

```
0    1233
1     237
Name: target, dtype: int64
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X, y)
y_resampled.value_counts()
```

```
1    1233
0    1233
Name: target, dtype: int64
```

```
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, shuffle=True, test_size=0.3)
```

Support Vector Classifier

```
svc = SVC()
svc.fit(X_train, y_train)
svc_pred = svc.predict(X_test)
svc_acc = accuracy_score(y_test, svc_pred)
svc_f1 = f1_score(y_test, svc_pred)
print(f"Accuracy Score: {svc_acc}\nF1 Score {svc_f1}")
```

```
Accuracy Score: 0.6013513513513513
F1 Score 0.6406820950060902
```

SUPPORT VECTOR CLASSIFIER HAS THE ACCURACY OF 60 PERCENT

Decision Tree Classifier

```
tree = DecisionTreeClassifier()
tree.fit(X_train, y_train)
tree_pred = tree.predict(X_test)
tree_acc = accuracy_score(y_test, tree_pred)
```

```
tree_f1 = f1_score(y_test, tree_pred)
print(f"Accuracy Score: {tree_acc}\nF1 Score {tree_f1}")
```

```
Accuracy Score: 0.8027027027027027
F1 Score 0.8093994778067884
```

DECISION TREE CLASSIFIER HAS THE ACCURACY OF 80% SO LETS TRY OTHER CLASSIFIERS ALSO TO SEE IMPROVED ACCURACY

Random Forest Classifier

```
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
rfc_pred = rfc.predict(X_test)
rfc_acc = accuracy_score(y_test, rfc_pred)
rfc_f1 = f1_score(y_test, rfc_pred)
print(f"Accuracy Score: {rfc_acc}\nF1 Score {rfc_f1}")
```

```
Accuracy Score: 0.904054054054054
F1 Score 0.9031377899045021
```

```
recalling = recall_score(y_test, rfc_pred)
precision = precision_score(y_test, rfc_pred)
print(f"Precision score: {precision}\nRecall score: {recalling}")
```

```
Precision score: 0.927170868347339
Recall score: 0.8803191489361702
```

Random Forest Classifier IS GIVING US 90% OF ACCURACY .SO FROM THE ALL CLASSIFIERS RANDOM FOREST CLASSIFIER IS GIVING US THE BEST ACCURATE RESULTS

LET US SEE THE HYPERPARAMETER TUNING AND OBTAIN THE BEST SCORE FROM THE MODEL

```
from sklearn.model_selection import GridSearchCV
# define parameter grid
parameter_grid = {'n_estimators':[150,200,250],
             'max_depth': np.arange(10,20),
             'bootstrap': [True, False]}
# Create an instance of GridSearchCV
grid_search = GridSearchCV(rfc,parameter_grid,cv=5,scoring=('accuracy','recall','precision'),verbose=1,refit='accuracy')
```

```
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 60 candidates, totalling 300 fits
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'bootstrap': [True, False],
                         'max_depth': array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19]),
                         'n_estimators': [150, 200, 250]},
             refit='accuracy', scoring=('accuracy', 'recall', 'precision'),
             verbose=1)
```

```
print(f"Best score: {grid_search.best_score_}\nBest Params: {grid_search.best_params_}")
```

```
Best score: 0.9281628549886907
Best Params: {'bootstrap': False, 'max_depth': 15, 'n_estimators': 200}
```

```
model = RandomForestClassifier(bootstrap=False, max_depth=13, n_estimators=150)
# train our model
model.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=False, max_depth=13, n_estimators=150)
```

```
prediction = model.predict(X_test)
print(classification_report(y_test, prediction))
```

```
              precision    recall  f1-score   support

           0       0.88      0.95      0.91       364
           1       0.94      0.88      0.91       376
```

```
    accuracy                           0.91       740
   macro avg       0.91      0.91      0.91       740
weighted avg       0.91      0.91      0.91       740
```

So from the given data of green destinations I conclude that the random forest classifier is giving us the accurate results and from the exploratory data analysis I observed that employees with less salary less working years are opting for the attrition.

✓ 0s completed at 8:50 PM