Code

```
!unzip ibm-hr-analytics-attrition-dataset.zip
```

```
Archive:  ibm-hr-analytics-attrition-dataset.zip
  inflating: WA_Fn-UseC_-HR-Employee-Attrition.csv
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import pylab as pl
from sklearn.metrics import roc_curve, auc
from sklearn import metrics
from sklearn import feature_selection
import itertools
from sklearn.model_selection import cross_val_score
import warnings
import seaborn as sns
warnings.filterwarnings("ignore")
```

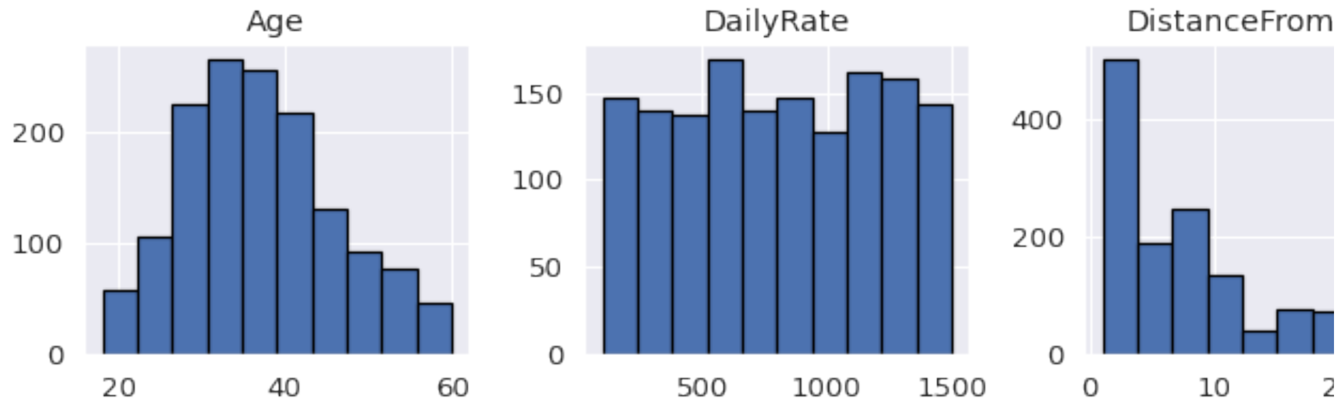
```
df = pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
df.head()
```

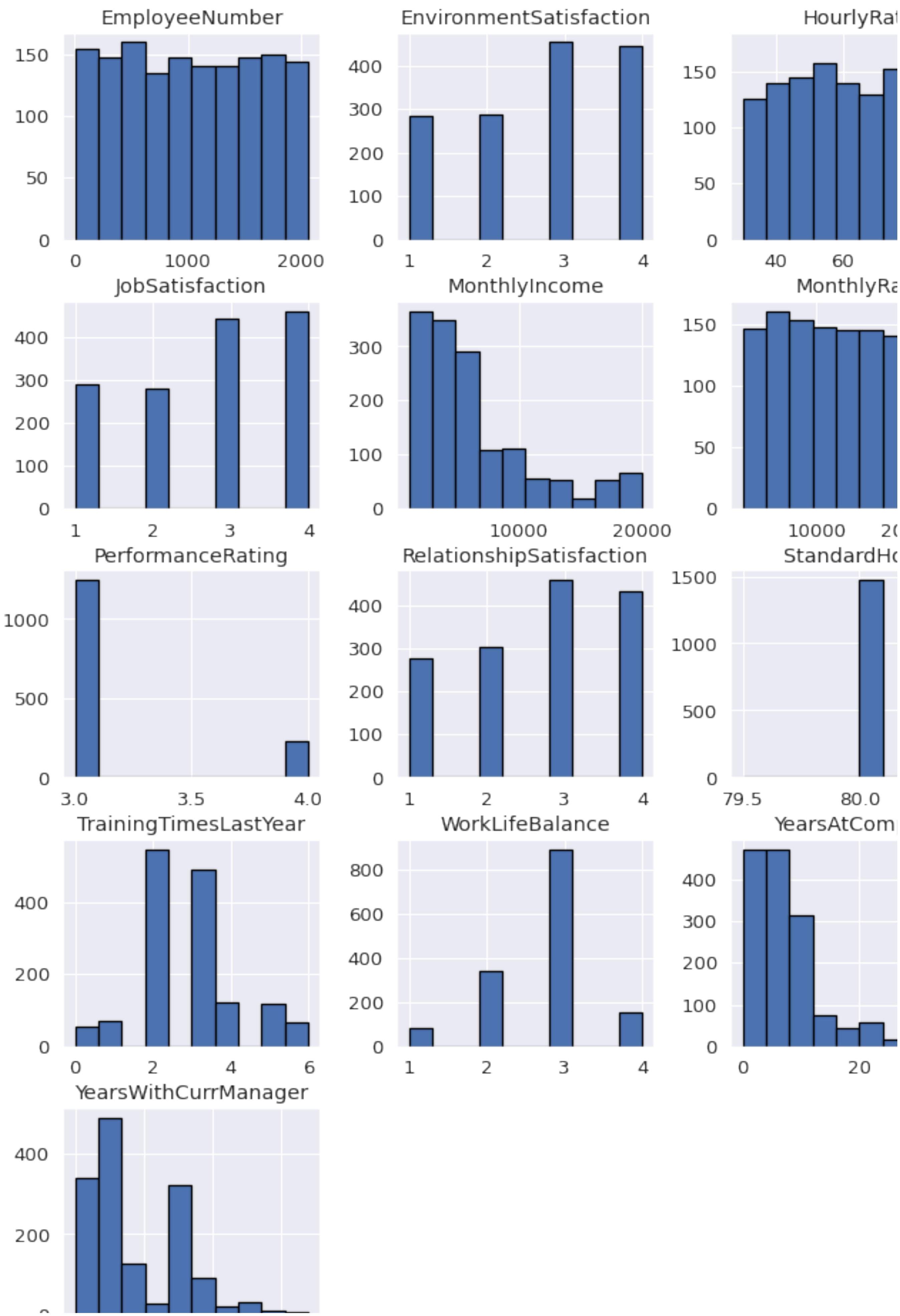
	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	E
0	41	Yes	Travel_Rarely	1102	Sales	1	
1	49	No	Travel_Frequently	279	Research & Development	8	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	
4	27	No	Travel_Rarely	591	Research & Development	2	

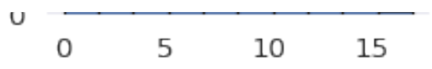
```
df.isnull().sum() #To check whether there are any missing values
```

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

```
df.hist(edgecolor='black', linewidth=1.2, figsize=(20, 20));
```




EmployeeNumber
150
100
50
0
0
1000
2000
EnvironmentSatisfaction
400
300
200
100
0
1
2
3
4
HourlyRat
150
100
50
0
40
60
JobSatisfaction
400
300
200
100
0
1
2
3
4
MonthlyIncome
300
200
100
0
10000
20000
MonthlyRa
150
100
50
0
10000
20
PerformanceRating
1000
500
0
3.0
3.5
4.0
RelationshipSatisfaction
400
300
200
100
0
1
2
3
4
StandardHo
1500
1000
500
0
79.5
80.0
TrainingTimesLastYear
400
200
0
0
2
4
6
WorkLifeBalance
800
600
400
200
0
1
2
3
4
YearsAtCom
400
300
200
100
0
0
20
YearsWithCurrManager
400
200



As we see that standard deviation for EmployeeCount, StandardHours, Over18 is near zero so w
Indentification number which won't be helpful in prediction so we will also drop it.

```
df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis="co
```

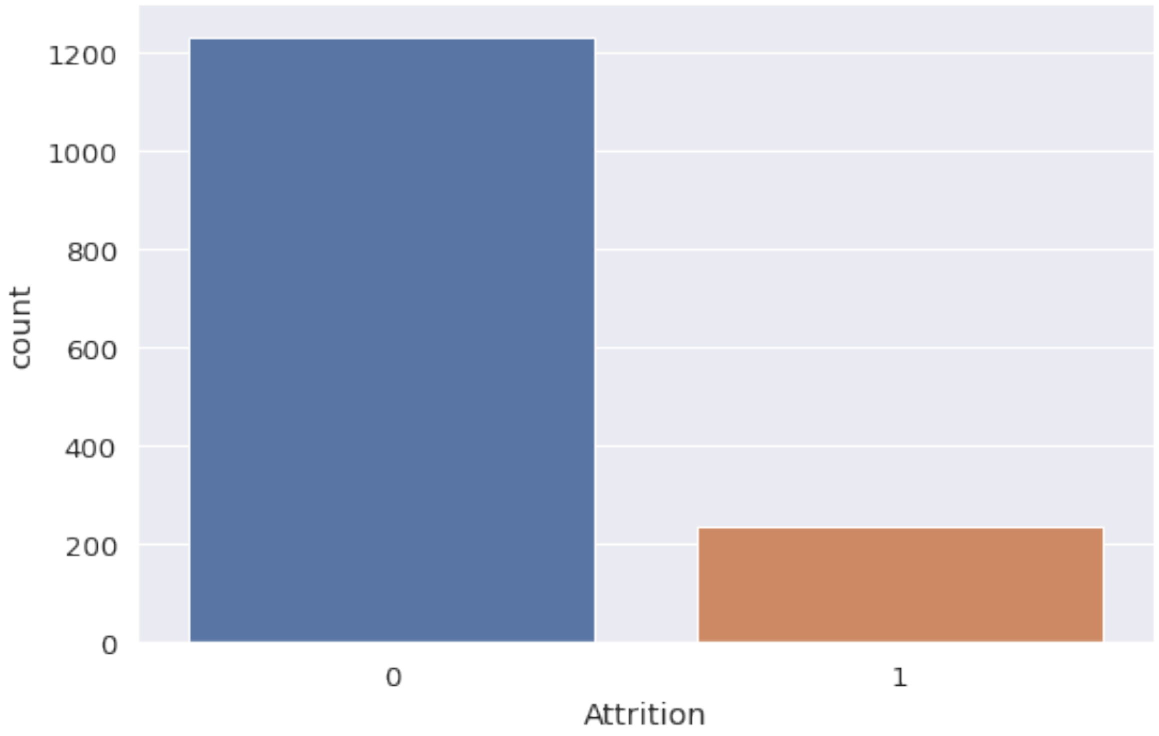
```
categorical_col = []
for column in df.columns:
    if df[column].dtype == object and len(df[column].unique()) <= 50:
        categorical_col.append(column)
        print(f"{column} : {df[column].unique()}")
        print("====================================")
```

```
Attrition : ['Yes' 'No']
====================================
BusinessTravel : ['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
====================================
Department : ['Sales' 'Research & Development' 'Human Resources']
====================================
EducationField : ['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical D
 'Human Resources']
====================================
Gender : ['Female' 'Male']
====================================
JobRole : ['Sales Executive' 'Research Scientist' 'Laboratory Technician'
 'Manufacturing Director' 'Healthcare Representative' 'Manager'
 'Sales Representative' 'Research Director' 'Human Resources']
====================================
MaritalStatus : ['Single' 'Married' 'Divorced']
====================================
OverTime : ['Yes' 'No']
====================================
```

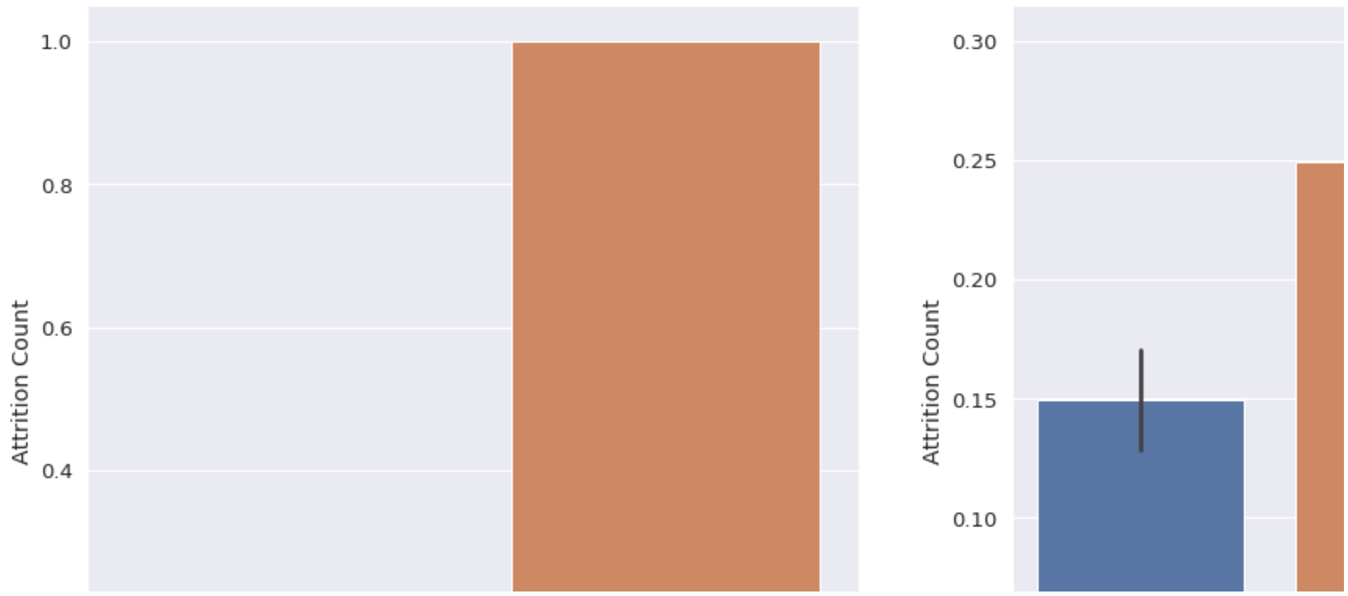
```
df['Attrition'] = df.Attrition.astype("category").cat.codes
```

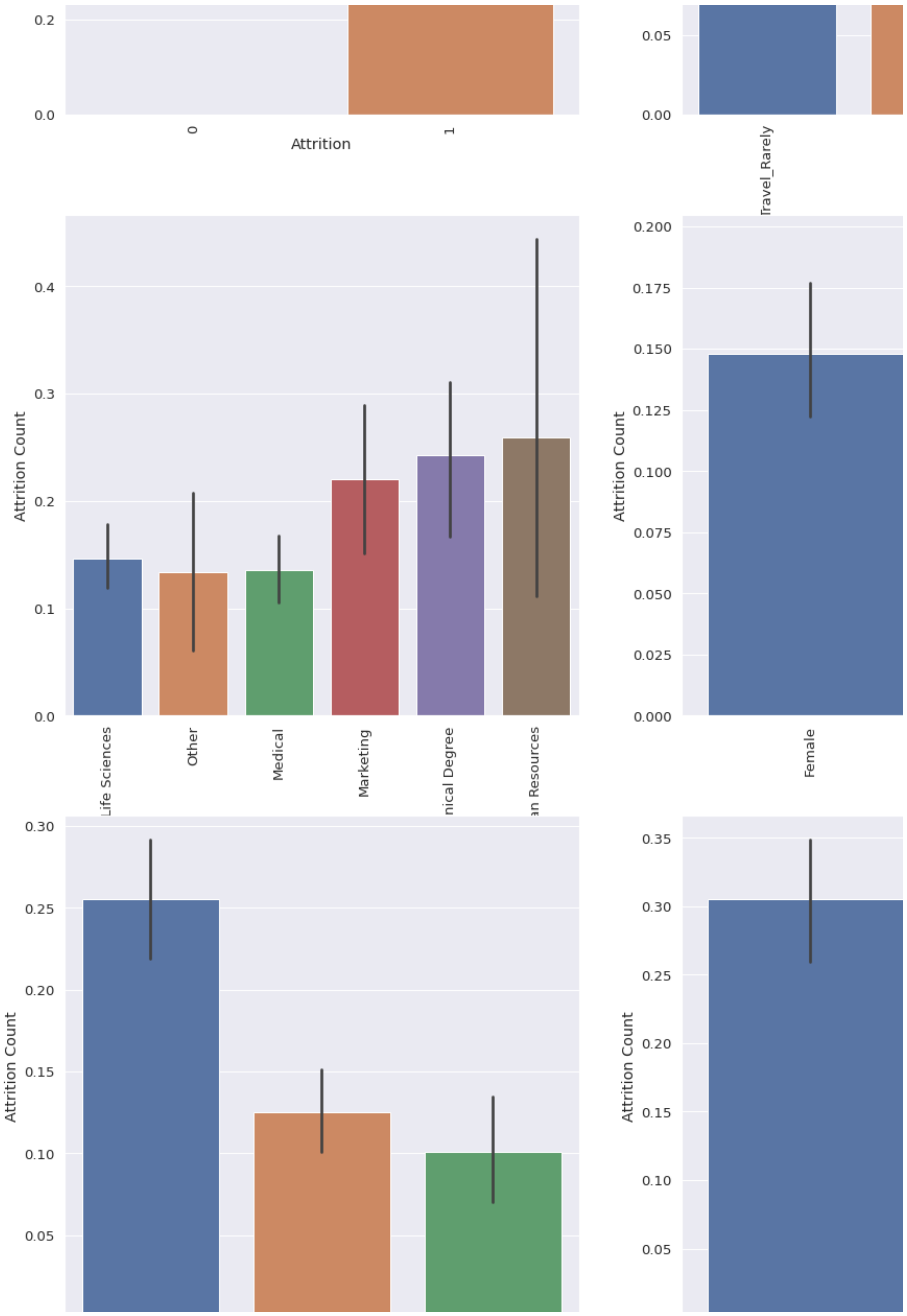
```
def plot_cat(attr,labels=None):
    if(attr=='JobRole'):
        sns.factorplot(data=df,kind='count',size=5,aspect=3,x=attr)
        return

    sns.factorplot(data=df,kind='count',size=5,aspect=1.5,x=attr)
```

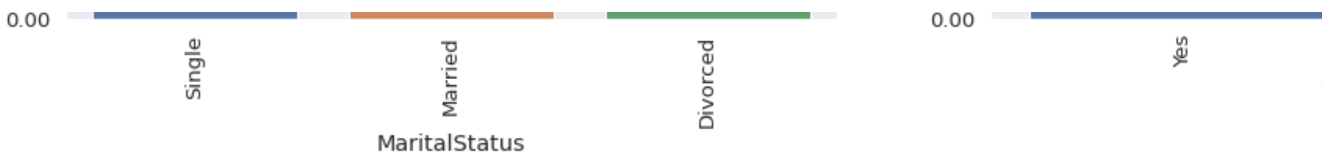
```
plot_cat('Attrition')
```



```
# Plotting how every feature correlate with the "target"
sns.set(font_scale=1.2)
plt.figure(figsize=(30, 30))

for i, column in enumerate(categorical_col, 1):
    plt.subplot(3, 3, i)
    g = sns.barplot(x=f"{column}", y='Attrition', data=df)
    g.set_xticklabels(g.get_xticklabels(), rotation=90)
    plt.ylabel('Attrition Count')
    plt.xlabel(f'{column}')
```




0.2
0.0
0
1
Attrition
0.05
0.00
Travel_Rarely
0.4
0.3
0.2
0.1
0.0
Attrition Count
Life Sciences
Other
Medical
Marketing
nical Degree
an Resources
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Attrition Count
Female
0.30
0.25
0.20
0.15
0.10
0.05
Attrition Count
0.35
0.30
0.25
0.20
0.15
0.10
0.05
Attrition Count



Conclusions

BusinessTravel : The workers who travel alot are more likely to quit then other employees.

Department : The worker in Research & Development are more likely to stay then the workers or

EducationField : The workers with Human Resources and Technical Degree are more likely to qu educations.

Gender : The Male are more likely to quit.

JobRole : The workers in Laboratory Technician, Sales Representative, and Human Resources a positions.

MaritalStatus : The workers who have Single marital status are more likely to quit the Married, a

OverTime : The workers who work more hours are likely to quit then others.

```
plt.figure(figsize=(30, 30))
sns.heatmap(df.corr(), annot=True, cmap="RdYlGn", annot_kws={"size":15})
```

```
categorical_col.remove('Attrition')
```

```
# Transform categorical data into dummies
# categorical_col.remove("Attrition")
# data = pd.get_dummies(df, columns=categorical_col)
# data.info()
from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()
for column in categorical_col:
    df[column] = label.fit_transform(df[column])
```

```
X = df.drop('Attrition', axis=1)
y = df.Attrition
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, r

def print_score(clf, X, y, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        print("\nTrain Result:\n===========================================")
        print(f"accuracy score: {accuracy_score(y_train, pred):.4f}\n")
        train_accuracy.append(accuracy_score(y_train, pred))
        print(f"Classification Report: \n \tPrecision: {precision_score(y_train,
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, clf.predict(X_tr

    elif train==False:
        pred = clf.predict(X_test)
        print("\nTest Result:\n===========================================")
        test_accuracy.append(accuracy_score(y_test, pred))
        print(f"accuracy score: {accuracy_score(y_test, pred)}\n")
        print(f"Classification Report: \n \tPrecision: {precision_score(y_test,
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")
        scores = cross_val_score(clf, X, y, cv=10)
        cv_accuracy.append(scores.mean())
        cv_devia.append(scores.std()*2)
        print(f"10-fold Cross Validation Accuracy: %0.2f (+/- %0.2f)" % (scores.


def plot_ROC(model, X_test, y_test):
  probs = model.predict_proba(X_test)
  preds = probs[:,1]
  fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
  roc_auc = metrics.auc(fpr, tpr)
  roc_accuracy.append(roc_auc)
  # method I: plt
  plt.title('Receiver Operating Characteristic')
  plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
  plt.legend(loc = 'lower right')
  plt.plot([0, 1], [0, 1],'r--')
  plt.xlim([0, 1])
  plt.ylim([0, 1])
  plt.ylabel('True Positive Rate')
  plt.xlabel('False Positive Rate')
  plt.show()
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_
```

```
train_accuracy = []
test_accuracy = []
cv_accuracy = []
cv_devia = []
roc_accuracy = []
```

Decision Tree

↳ 5 cells hidden

Random Forest

↳ 7 cells hidden

Logistic Regression

↳ 3 cells hidden

SVM

↳ 7 cells hidden