```
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
df = pd.read_csv('/content/IBM Attrition Data.csv')
df.head()
```

```
   Age Attrition              Department  DistanceFromHome  Education
\
0   41       Yes                   Sales                 1          2

1   49        No  Research & Development                 8          1

2   37       Yes  Research & Development                 2          2

3   33        No  Research & Development                 3          4

4   27        No  Research & Development                 2          1


  EducationField  EnvironmentSatisfaction  JobSatisfaction
MaritalStatus  \
0  Life Sciences                        2                4
Single
1  Life Sciences                        3                2
Married
2          Other                        4                3
Single
3  Life Sciences                        4                3
Married
4        Medical                        1                2
Married

   MonthlyIncome  NumCompaniesWorked  WorkLifeBalance  YearsAtCompany

0           5993                   8                1               6

1           5130                   1                3              10

2           2090                   6                3               0

3           2909                   1                3               8

4           3468                   9                3               2
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      1470 non-null   int64
 1   Attrition                1470 non-null   object
 2   Department               1470 non-null   object
 3   DistanceFromHome         1470 non-null   int64
 4   Education                1470 non-null   int64
 5   EducationField           1470 non-null   object
 6   EnvironmentSatisfaction  1470 non-null   int64
 7   JobSatisfaction          1470 non-null   int64
 8   MaritalStatus            1470 non-null   object
 9   MonthlyIncome            1470 non-null   int64
 10  NumCompaniesWorked       1470 non-null   int64
 11  WorkLifeBalance          1470 non-null   int64
 12  YearsAtCompany           1470 non-null   int64
dtypes: int64(9), object(4)
memory usage: 149.4+ KB
```

```
df.describe().T
```

	count	mean	std	min	25%	\
Age	1470.0	36.923810	9.135373	18.0	30.0	
DistanceFromHome	1470.0	9.192517	8.106864	1.0	2.0	
Education	1470.0	2.912925	1.024165	1.0	2.0	
EnvironmentSatisfaction	1470.0	2.721769	1.093082	1.0	2.0	
JobSatisfaction	1470.0	2.728571	1.102846	1.0	2.0	
MonthlyIncome	1470.0	6502.931293	4707.956783	1009.0	2911.0	
NumCompaniesWorked	1470.0	2.693197	2.498009	0.0	1.0	
WorkLifeBalance	1470.0	2.761224	0.706476	1.0	2.0	
YearsAtCompany	1470.0	7.008163	6.126525	0.0	3.0	

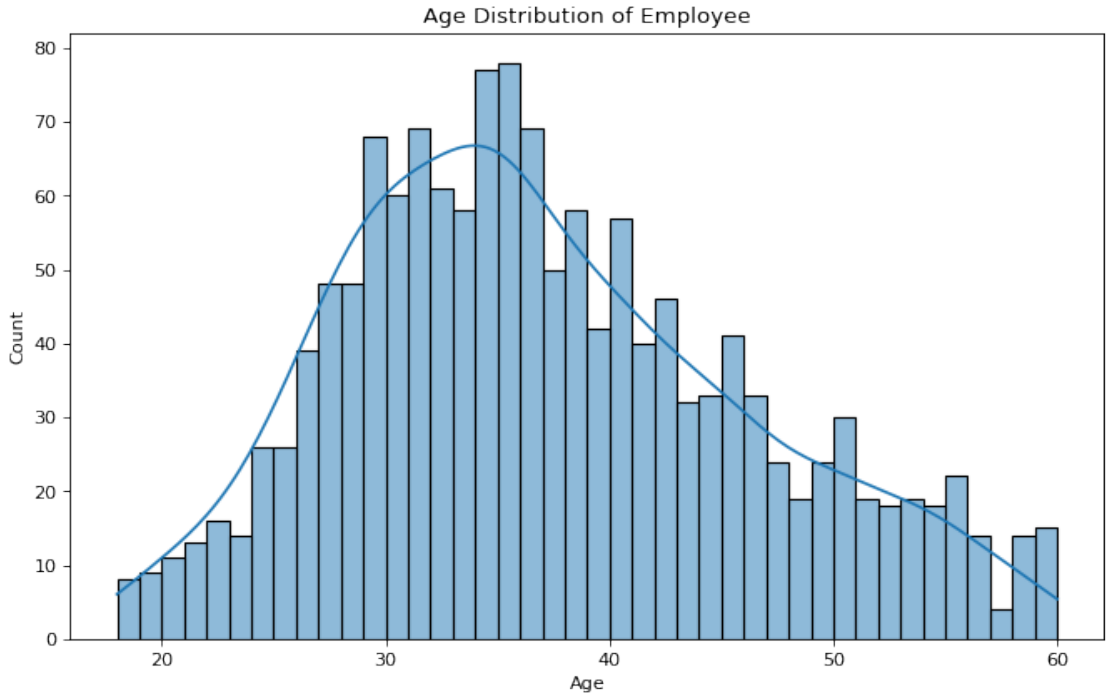
	50%	75%	max
Age	36.0	43.0	60.0
DistanceFromHome	7.0	14.0	29.0
Education	3.0	4.0	5.0
EnvironmentSatisfaction	3.0	4.0	4.0
JobSatisfaction	3.0	4.0	4.0

```
MonthlyIncome            4919.0  8379.0  19999.0
NumCompaniesWorked          2.0     4.0      9.0
WorkLifeBalance             3.0     3.0      4.0
YearsAtCompany              5.0     9.0     40.0
```

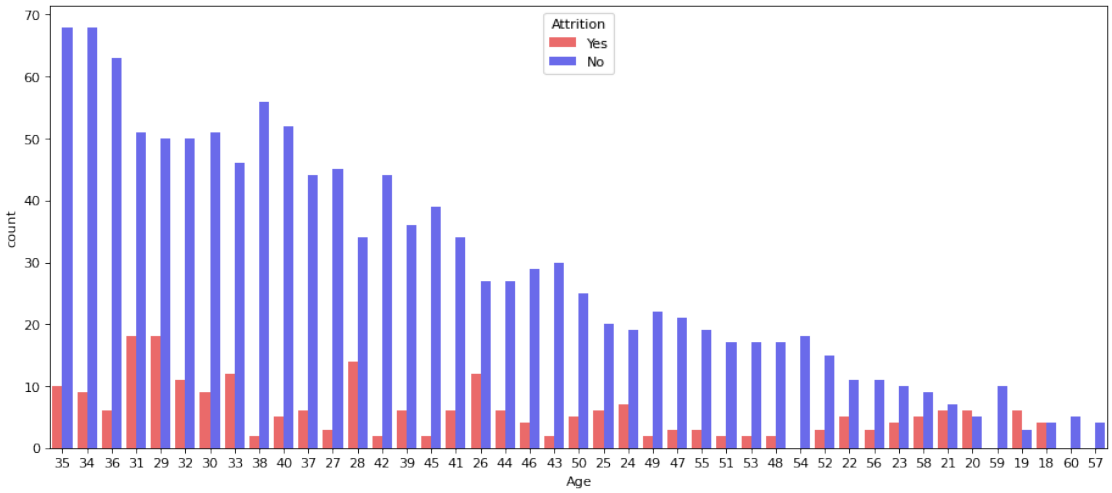
```
df.isnull().sum()
```

```
Age                         0
Attrition                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EnvironmentSatisfaction     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
NumCompaniesWorked          0
WorkLifeBalance             0
YearsAtCompany              0
dtype: int64
```

```
df = df[['Age', 'Attrition', 'Department', 'DistanceFromHome',
'Education', 'EducationField', 'EnvironmentSatisfaction',
         'JobSatisfaction','MaritalStatus', 'MonthlyIncome',
'NumCompaniesWorked','WorkLifeBalance', 'YearsAtCompany']]

# histogram for age

plt.figure(figsize=(10,6), dpi=80)
sns.histplot(data=df, x='Age', bins=42, kde=True).set_title('Age
Distribution of Employee');
```

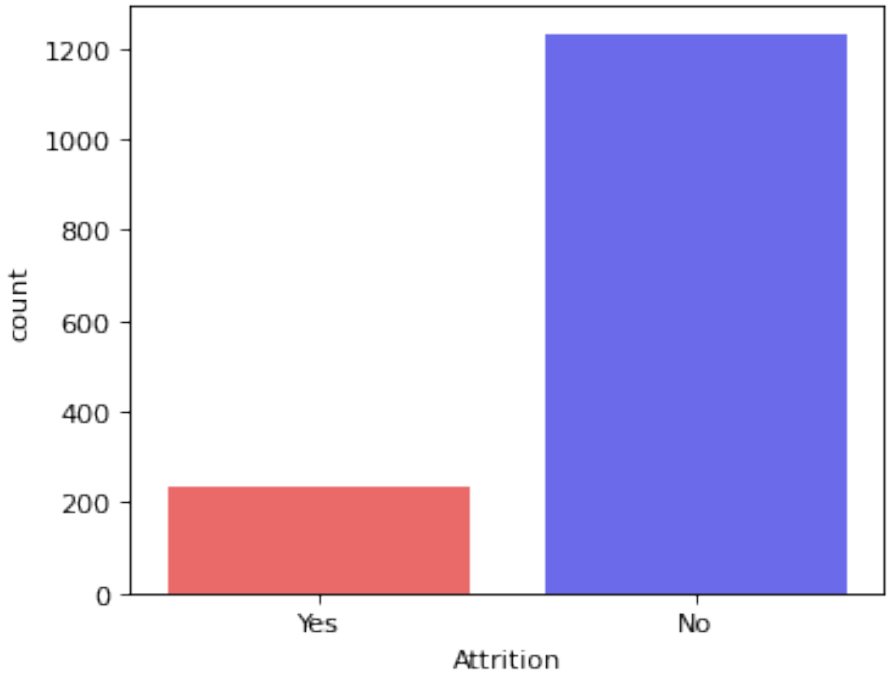



```
# print(df[(df['Attrition'] == 'Yes')].groupby('Age')
['Age'].count().sort_values(ascending=False))

plt.figure(figsize=(14,6), dpi=80)
sns.countplot(data=df, x='Age', hue='Attrition', order =
df['Age'].value_counts().index, palette='seismic_r').set_title
('Attrition by Age');
```

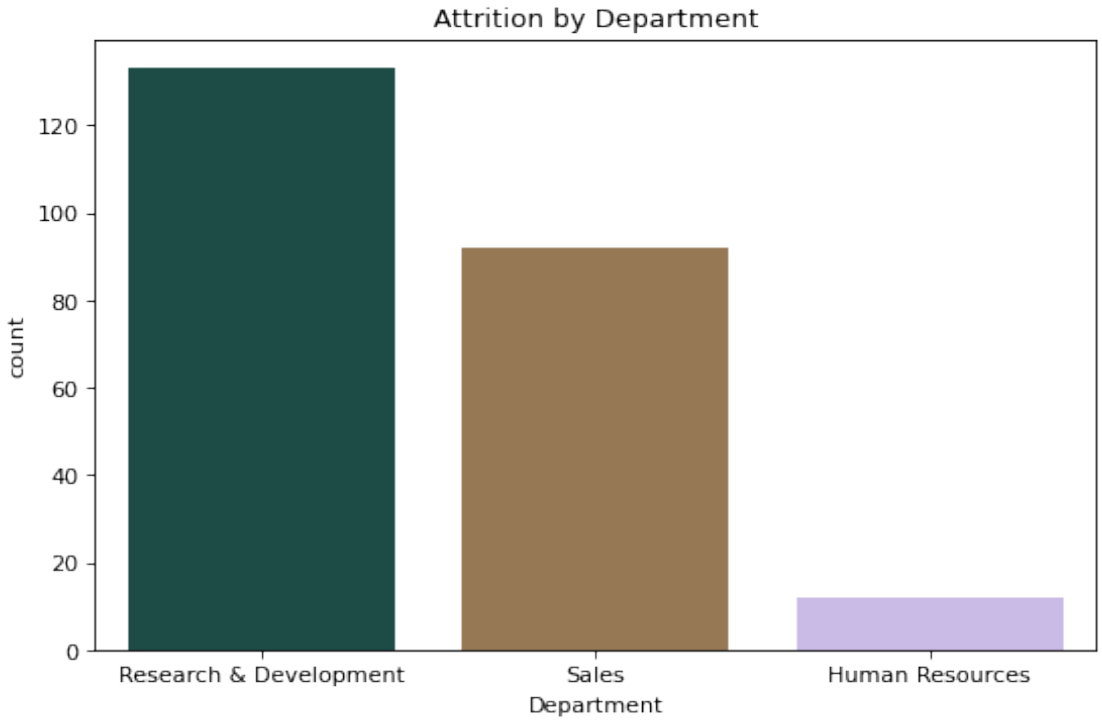



```
print(df.groupby('Attrition')['Attrition'].count())

plt.figure(figsize=(5,4), dpi=80)
sns.countplot(data=df, x='Attrition', palette='seismic_r');
```

```
Attrition
No     1233
Yes     237
Name: Attrition, dtype: int64
```




```
# print(df[(df['Attrition'] == 'Yes')].groupby('Department')
['Attrition'].count().sort_values(ascending=False))

plt.figure(figsize=(8,5), dpi=80)
sns.countplot(data=df[(df['Attrition'] == 'Yes')], x='Department',
palette='cubehelix', order = df['Department']
              .value_counts().index).set_title('Attrition by
Department');
```

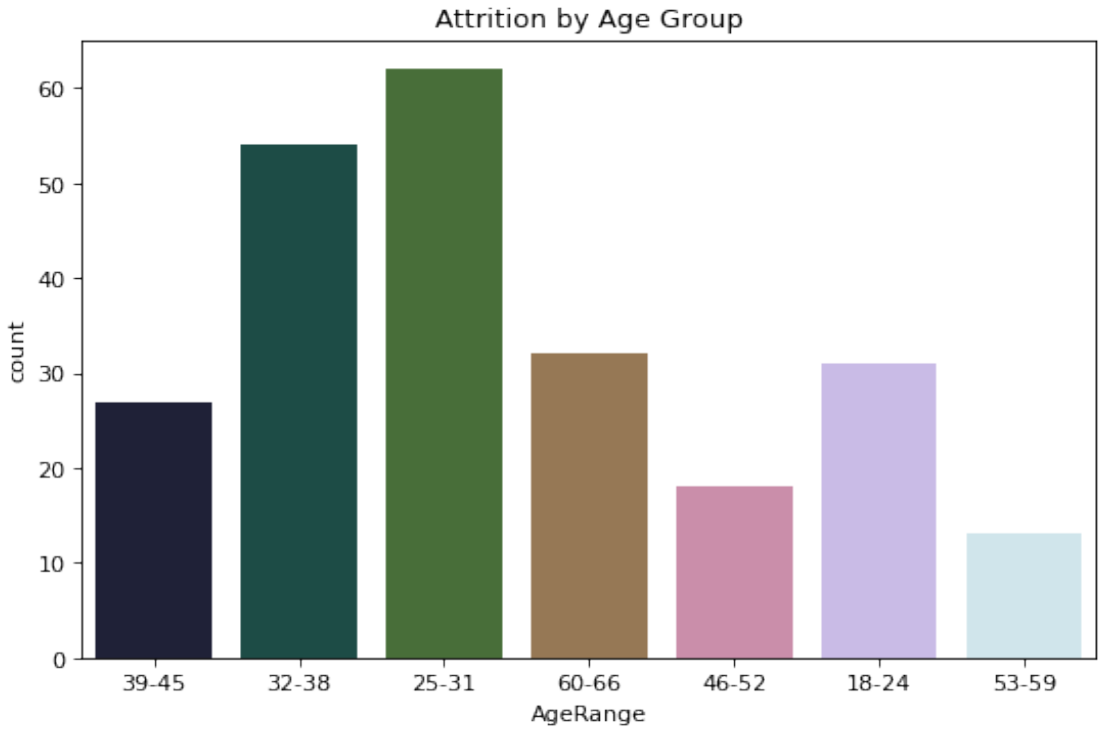



```
agerange = []
for age in df["Age"]:
    if age >= 18 and age < 24:
        agerange.append("18-24")
    elif age >= 25 and age < 31:
        agerange.append("25-31")
    elif age >= 32 and age < 38:
        agerange.append("32-38")
    elif age >= 39 and age < 45:
        agerange.append("39-45")
    elif age >= 46 and age < 52:
        agerange.append("46-52")
    elif age >= 53 and age < 59:
        agerange.append("53-59")
    else:
        agerange.append("60-66")

df["AgeRange"] = agerange
df.head()
```

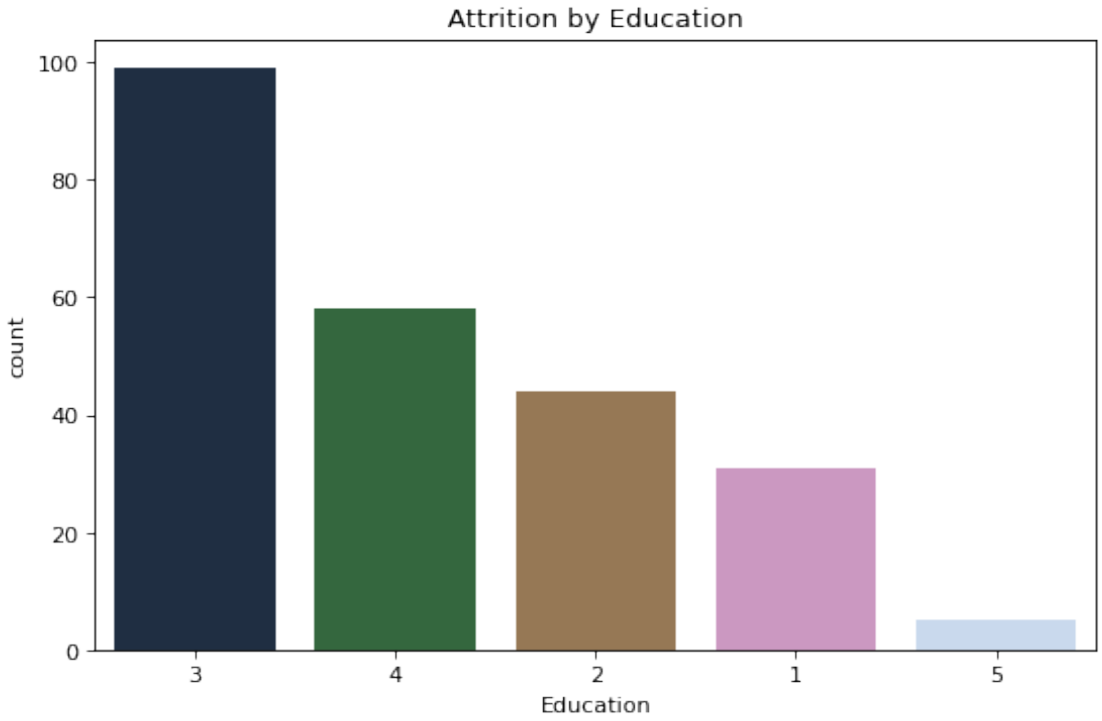
```
   Age Attrition              Department  DistanceFromHome  Education
\
0   41       Yes                   Sales                 1          2

1   49        No  Research & Development                 8          1

2   37       Yes  Research & Development                 2          2
```

```
3   33        No  Research & Development                 3          4

4   27        No  Research & Development                 2          1


  EducationField  EnvironmentSatisfaction  JobSatisfaction
MaritalStatus  \
0  Life Sciences                        2               4
Single
1  Life Sciences                        3               2
Married
2          Other                        4               3
Single
3  Life Sciences                        4               3
Married
4        Medical                        1               2
Married

   MonthlyIncome  NumCcompaniesWorked  WorkLifeBalance  YearsAtCompany
AgeRange
0           5993                    8                1               6
39-45
1           5130                    1                3              10
46-52
2           2090                    6                3               0
32-38
3           2909                    1                3               8
32-38
4           3468                    9                3               2
25-31
```

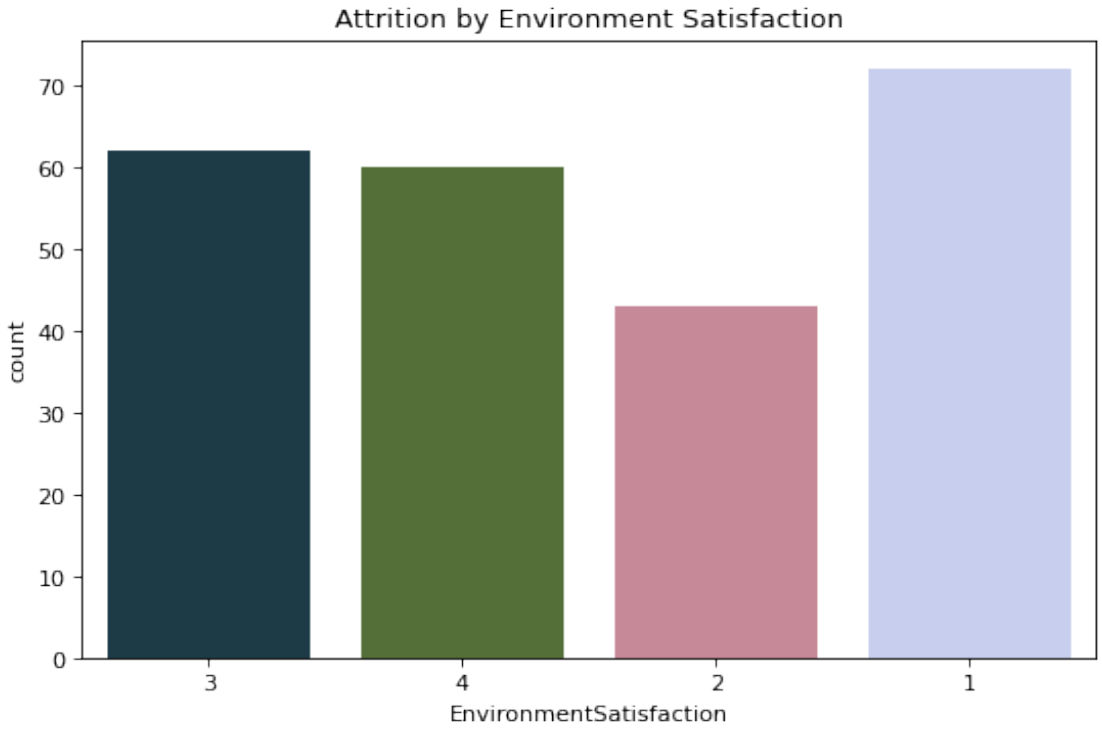
```
# print(df[(df['Attrition'] == 'Yes')].groupby('AgeRange')
['AgeRange'].count().sort_values(ascending=False))

plt.figure(figsize=(8,5), dpi=80)
sns.countplot(data=df[(df['Attrition'] == 'Yes')], x='AgeRange',
palette='cubehelix').set_title('Attrition by Age Group');
```



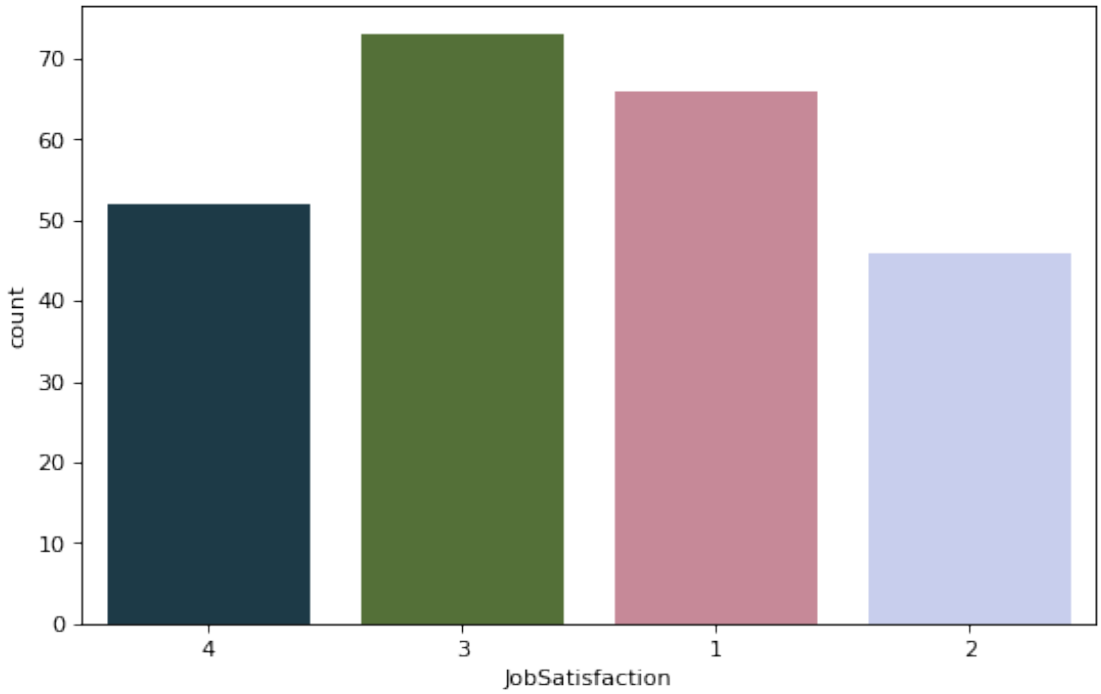

```
# print(df[(df['Attrition'] == 'Yes')].groupby('Education')
['Attrition'].count().sort_values(ascending=False))

plt.figure(figsize=(8,5),dpi=80)
sns.countplot(data=df[(df['Attrition'] == 'Yes')], x='Education',
order=df['Education'].value_counts().index,
              palette='cubehelix').set_title('Attrition by
Education');
```



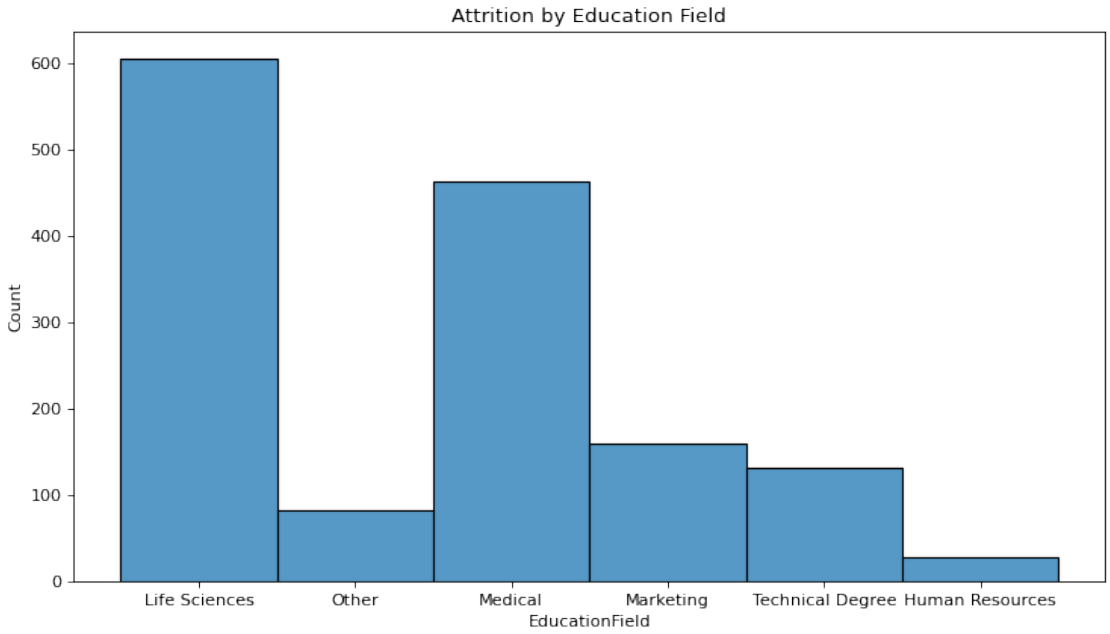

```
# print(df[(df['Attrition'] ==
'Yes')].groupby('EnvironmentSatisfaction')
['Attrition'].count().sort_values(ascending=False))

plt.figure(figsize=(8,5),dpi=80)
sns.countplot(data=df[(df['Attrition'] == 'Yes')],
x='EnvironmentSatisfaction', order=df['EnvironmentSatisfaction']
              .value_counts().index,
palette='cubehelix').set_title('Attrition by Environment
Satisfaction');
```




```
# print(df[(df['Attrition'] == 'Yes')].groupby('JobSatisfaction')
['Attrition'].count().sort_values(ascending=False))

plt.figure(figsize=(8,5),dpi=80)
sns.countplot(data=df[(df['Attrition'] == 'Yes')],
x='JobSatisfaction', order=df['JobSatisfaction'].value_counts().index,
palette='cubehelix');
```

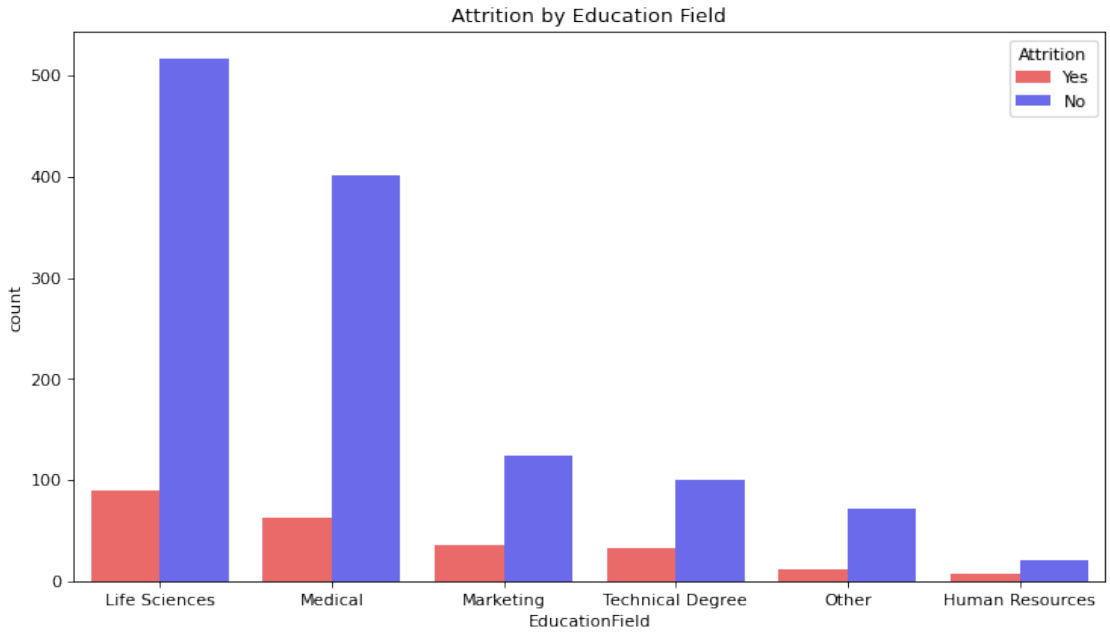



```
# print(df[(df['Attrition'] == 'Yes')].groupby('EducationField')
['EducationField'].count().sort_values(ascending=False))

plt.figure(figsize=(11,6), dpi=80)
sns.histplot(data=df, x='EducationField').set_title('Attrition by
Education Field');
```

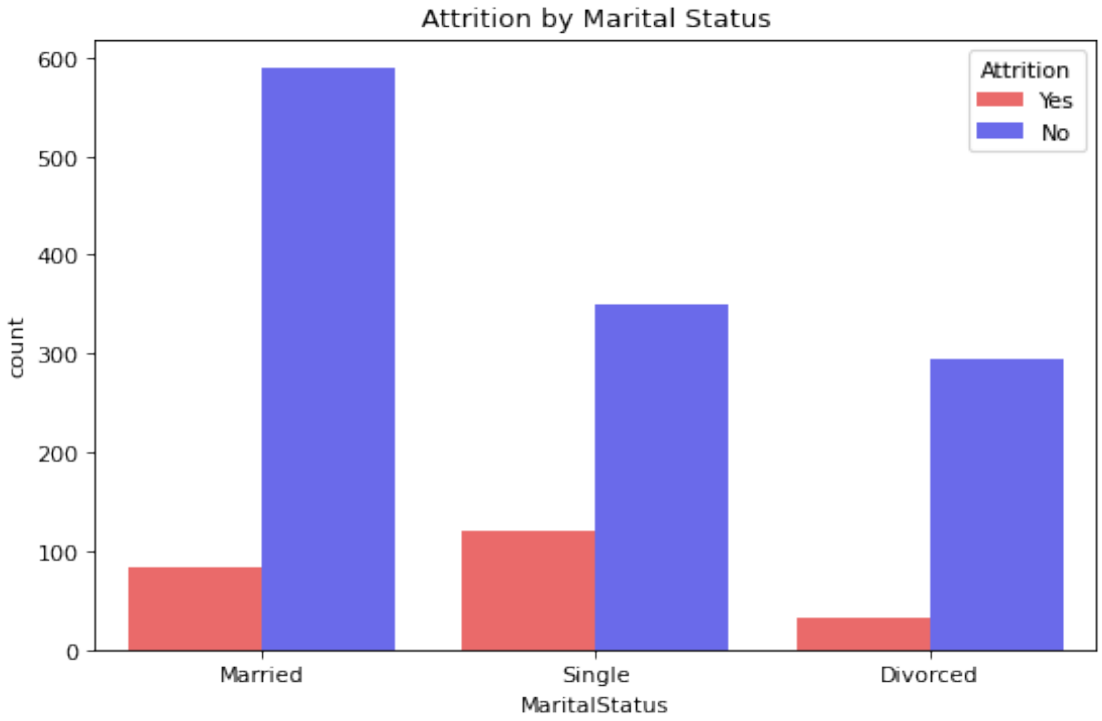



```
plt.figure(figsize=(11,6), dpi=80)
sns.countplot(data=df, x='EducationField', hue='Attrition', order =
```

```
df['EducationField'].value_counts().index,
palette='seismic_r').set_title('Attrition by Education Field');
```




```
# print(df[(df['Attrition'] == 'Yes')].groupby('MaritalStatus')
['Attrition'].count().sort_values(ascending=False))

plt.figure(figsize=(8,5),dpi=80)
sns.countplot(data=df, x='MaritalStatus', hue='Attrition',
order=df['MaritalStatus'].value_counts().index,
palette='seismic_r').set_title('Attrition by Marital Status');
```




```
df.dtypes
```

```
Age                         int64
Attrition                  object
Department                 object
DistanceFromHome            int64
Education                   int64
EducationField             object
EnvironmentSatisfaction     int64
JobSatisfaction             int64
MaritalStatus              object
MonthlyIncome               int64
NumCompaniesWorked          int64
WorkLifeBalance             int64
YearsAtCompany              int64
AgeRange                   object
dtype: object
```

```
df['Attrition'].replace('Yes', 1, inplace=True)
df['Attrition'].replace('No', 0, inplace=True)
df['Department'].replace('Human Resources', 1, inplace=True)
df['Department'].replace('Research & Development', 2, inplace=True)
df['Department'].replace('Sales', 3, inplace=True)
df['EducationField'].replace('Human Resources', 1, inplace=True)
df['EducationField'].replace('Life Sciences', 2, inplace=True)
df['EducationField'].replace('Marketing', 3, inplace=True)
df['EducationField'].replace('Medical', 4, inplace=True)
df['EducationField'].replace('Other', 5, inplace=True)
```

```
df['EducationField'].replace('Technical Degree', 6, inplace=True)
df['MaritalStatus'].replace('Divorced', 1, inplace=True)
df['MaritalStatus'].replace('Married', 2, inplace=True)
df['MaritalStatus'].replace('Single', 3, inplace=True)
df.dtypes
```

```
Age                         int64
Attrition                   int64
Department                  int64
DistanceFromHome            int64
Education                   int64
EducationField              int64
EnvironmentSatisfaction     int64
JobSatisfaction             int64
MaritalStatus               int64
MonthlyIncome               int64
NumCompaniesWorked          int64
WorkLifeBalance             int64
YearsAtCompany              int64
AgeRange                   object
dtype: object
```

```
x = df.drop(['Attrition', 'AgeRange', 'DistanceFromHome',
'NumCompaniesWorked', 'WorkLifeBalance'], axis=1)
y = df[['Attrition']]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr = lr.fit(x_train, y_train)
```

```
# check the accuracy on the training set
print('Accuracy =', lr.score(x_train, y_train)*100,'%');
```

```
Accuracy = 85.28911564625851 %
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/
validation.py:993: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
lr_y_pred = lr.predict(x_test)
pd.DataFrame(lr_y_pred).head()
```

```
   0
0  0
1  0
2  0
3  0
4  0
```

```
prob = lr.predict_proba(x_test)
print(prob)
```

```
[[0.88809852 0.11190148]
 [0.89226894 0.10773106]
 [0.8488027  0.1511973 ]
 [0.69837919 0.30162081]
 [0.88088139 0.11911861]
 [0.62575687 0.37424313]
 [0.89821967 0.10178033]
 [0.95841683 0.04158317]
 [0.95663234 0.04336766]
 [0.78545236 0.21454764]
 [0.97481259 0.02518741]
 [0.82986167 0.17013833]
 [0.96354114 0.03645886]
 [0.81984403 0.18015597]
 [0.96311734 0.03688266]
 [0.74506329 0.25493671]
 [0.93242633 0.06757367]
 [0.92948248 0.07051752]
 [0.59623262 0.40376738]
 [0.58572456 0.41427544]
 [0.82524584 0.17475416]
 [0.65385368 0.34614632]
 [0.84843423 0.15156577]
 [0.90997477 0.09002523]
 [0.93720237 0.06279763]
 [0.66199143 0.33800857]
 [0.77159759 0.22840241]
 [0.76855857 0.23144143]
 [0.73500245 0.26499755]
 [0.81781825 0.18218175]
 [0.95571046 0.04428954]
 [0.89030765 0.10969235]
 [0.80305538 0.19694462]
 [0.78057563 0.21942437]
 [0.94613964 0.05386036]
 [0.8834325  0.1165675 ]
 [0.93812612 0.06187388]
 [0.94954795 0.05045205]
 [0.94560644 0.05439356]
 [0.64164483 0.35835517]
 [0.95404185 0.04595815]
```

[0.9781094 0.0218906]
[0.84814739 0.15185261]
[0.6696115 0.3303885]
[0.8996602 0.1003398]
[0.90970153 0.09029847]
[0.98023719 0.01976281]
[0.70826188 0.29173812]
[0.82545918 0.17454082]
[0.90885626 0.09114374]
[0.9328453 0.0671547]
[0.9363322 0.0636678]
[0.73043809 0.26956191]
[0.91086454 0.08913546]
[0.59268692 0.40731308]
[0.86078304 0.13921696]
[0.72058174 0.27941826]
[0.90051487 0.09948513]
[0.92324132 0.07675868]
[0.91771847 0.08228153]
[0.93485959 0.06514041]
[0.8979041 0.1020959]
[0.83403928 0.16596072]
[0.94108873 0.05891127]
[0.93519502 0.06480498]
[0.94738434 0.05261566]
[0.67017395 0.32982605]
[0.89513686 0.10486314]
[0.80007229 0.19992771]
[0.95751043 0.04248957]
[0.69903388 0.30096612]
[0.83338095 0.16661905]
[0.94223601 0.05776399]
[0.92234062 0.07765938]
[0.86140545 0.13859455]
[0.82653811 0.17346189]
[0.71889706 0.28110294]
[0.69839086 0.30160914]
[0.72634463 0.27365537]
[0.79786947 0.20213053]
[0.78853346 0.21146654]
[0.71689301 0.28310699]
[0.91393149 0.08606851]
[0.94232534 0.05767466]
[0.86284129 0.13715871]
[0.72236796 0.27763204]
[0.91984516 0.08015484]
[0.94509883 0.05490117]
[0.93578696 0.06421304]
[0.63615017 0.36384983]
[0.96758662 0.03241338]

[0.93683774 0.06316226]
[0.98059875 0.01940125]
[0.92305113 0.07694887]
[0.94078302 0.05921698]
[0.72371331 0.27628669]
[0.76092803 0.23907197]
[0.9402585 0.0597415]
[0.64973681 0.35026319]
[0.91607898 0.08392102]
[0.96133708 0.03866292]
[0.91304925 0.08695075]
[0.71734922 0.28265078]
[0.89628649 0.10371351]
[0.80187119 0.19812881]
[0.89125967 0.10874033]
[0.69689125 0.30310875]
[0.79496217 0.20503783]
[0.49246166 0.50753834]
[0.95117197 0.04882803]
[0.88998408 0.11001592]
[0.86215911 0.13784089]
[0.82659331 0.17340669]
[0.87718541 0.12281459]
[0.83519833 0.16480167]
[0.93228706 0.06771294]
[0.98853433 0.01146567]
[0.83742158 0.16257842]
[0.49274621 0.50725379]
[0.6443019 0.3556981]
[0.57674576 0.42325424]
[0.81001533 0.18998467]
[0.83607945 0.16392055]
[0.73779687 0.26220313]
[0.96445353 0.03554647]
[0.9549095 0.0450905]
[0.79347655 0.20652345]
[0.65132641 0.34867359]
[0.8645664 0.1354336]
[0.93185821 0.06814179]
[0.66753994 0.33246006]
[0.94288554 0.05711446]
[0.96726992 0.03273008]
[0.69077679 0.30922321]
[0.95763888 0.04236112]
[0.82659373 0.17340627]
[0.94918039 0.05081961]
[0.90533058 0.09466942]
[0.84549264 0.15450736]
[0.94529312 0.05470688]
[0.78333606 0.21666394]

[0.83389635 0.16610365]
[0.86377566 0.13622434]
[0.83238893 0.16761107]
[0.81545821 0.18454179]
[0.8836396 0.1163604]
[0.97554459 0.02445541]
[0.9402062 0.0597938]
[0.60325005 0.39674995]
[0.73990891 0.26009109]
[0.93565253 0.06434747]
[0.89610011 0.10389989]
[0.89541789 0.10458211]
[0.92819316 0.07180684]
[0.83252926 0.16747074]
[0.90517395 0.09482605]
[0.56743654 0.43256346]
[0.68562358 0.31437642]
[0.81547531 0.18452469]
[0.90549715 0.09450285]
[0.43282189 0.56717811]
[0.79236893 0.20763107]
[0.80543318 0.19456682]
[0.95937628 0.04062372]
[0.67394922 0.32605078]
[0.85169967 0.14830033]
[0.74787463 0.25212537]
[0.90580772 0.09419228]
[0.81943873 0.18056127]
[0.89577185 0.10422815]
[0.59295462 0.40704538]
[0.86693542 0.13306458]
[0.65456241 0.34543759]
[0.83756719 0.16243281]
[0.6768482 0.3231518]
[0.67246876 0.32753124]
[0.78174991 0.21825009]
[0.81411091 0.18588909]
[0.93881221 0.06118779]
[0.74209864 0.25790136]
[0.85591055 0.14408945]
[0.73853474 0.26146526]
[0.79567807 0.20432193]
[0.91311673 0.08688327]
[0.83581888 0.16418112]
[0.87437616 0.12562384]
[0.68878316 0.31121684]
[0.96594618 0.03405382]
[0.83800253 0.16199747]
[0.63660772 0.36339228]
[0.99122066 0.00877934]

[0.88128425 0.11871575]
[0.77346607 0.22653393]
[0.82155158 0.17844842]
[0.98525857 0.01474143]
[0.89805266 0.10194734]
[0.81398005 0.18601995]
[0.80132133 0.19867867]
[0.94246099 0.05753901]
[0.94956502 0.05043498]
[0.93123006 0.06876994]
[0.96931523 0.03068477]
[0.92044628 0.07955372]
[0.94214493 0.05785507]
[0.92059077 0.07940923]
[0.85994405 0.14005595]
[0.82064884 0.17935116]
[0.9044545 0.0955455]
[0.98221216 0.01778784]
[0.87822041 0.12177959]
[0.78555539 0.21444461]
[0.74524687 0.25475313]
[0.8947117 0.1052883]
[0.86758209 0.13241791]
[0.98450866 0.01549134]
[0.8564952 0.1435048]
[0.88216847 0.11783153]
[0.7829534 0.2170466]
[0.93536485 0.06463515]
[0.85009236 0.14990764]
[0.79595791 0.20404209]
[0.72499891 0.27500109]
[0.6854314 0.3145686]
[0.91129426 0.08870574]
[0.74858669 0.25141331]
[0.96729834 0.03270166]
[0.95677469 0.04322531]
[0.94851614 0.05148386]
[0.87774874 0.12225126]
[0.93469067 0.06530933]
[0.92761283 0.07238717]
[0.95217291 0.04782709]
[0.87361931 0.12638069]
[0.94078607 0.05921393]
[0.89461636 0.10538364]
[0.94766321 0.05233679]
[0.57687648 0.42312352]
[0.41929862 0.58070138]
[0.89897076 0.10102924]
[0.87089862 0.12910138]
[0.86017195 0.13982805]

```
[0.8053956  0.1946044 ]
[0.97655918 0.02344082]
[0.88006957 0.11993043]
[0.91029747 0.08970253]
[0.73120952 0.26879048]
[0.8732278  0.1267722 ]
[0.74735941 0.25264059]
[0.89132353 0.10867647]
[0.54888214 0.45111786]
[0.89772849 0.10227151]
[0.47075167 0.52924833]
[0.95716786 0.04283214]
[0.76712959 0.23287041]
[0.85905403 0.14094597]
[0.44648047 0.55351953]
[0.99497421 0.00502579]
[0.78860143 0.21139857]
[0.98704607 0.01295393]
[0.83783768 0.16216232]
[0.98033205 0.01966795]
[0.83016527 0.16983473]
[0.93874792 0.06125208]
[0.88181999 0.11818001]
[0.87926618 0.12073382]
[0.93284049 0.06715951]
[0.91168595 0.08831405]
[0.85518468 0.14481532]
[0.94702183 0.05297817]
[0.7924446  0.2075554 ]
[0.78607208 0.21392792]
[0.62868343 0.37131657]
[0.91030575 0.08969425]
[0.83274787 0.16725213]
[0.8856958  0.1143042 ]
[0.85913255 0.14086745]
[0.94722362 0.05277638]
[0.77100378 0.22899622]
[0.76810974 0.23189026]
[0.8979711  0.1020289 ]
[0.57379527 0.42620473]
[0.92521966 0.07478034]
[0.94634093 0.05365907]
[0.81466654 0.18533346]
[0.9949775  0.0050225 ]
[0.97108584 0.02891416]
[0.9069691  0.0930309 ]
[0.80651785 0.19348215]
[0.88122739 0.11877261]
[0.83614243 0.16385757]
[0.97202813 0.02797187]
```

```
 [0.91894385 0.08105615]
 [0.87327822 0.12672178]
 [0.97649127 0.02350873]]
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

print('Test Accuracy Score:', accuracy_score(y_test, lr_y_pred)*100,
'%\n')

print('----------------------Classification
Report----------------------\n', classification_report(y_test,
lr_y_pred))
print('----------------------Confusion Matrix----------------------\
n', confusion_matrix(y_test, lr_y_pred))
```

```
Test Accuracy Score: 84.6938775510204 %

----------------------Classification Report----------------------
              precision    recall  f1-score   support

           0       0.85      0.99      0.92       249
           1       0.50      0.07      0.12        45

    accuracy                           0.85       294
   macro avg       0.68      0.53      0.52       294
weighted avg       0.80      0.85      0.79       294

----------------------Confusion Matrix----------------------
 [[246   3]
 [ 42   3]]
```

```
# add random values to check the proabily of attrition of the employee

pd.DataFrame(x_train).head()
```

```
          0         1         2         3         4         5
6  \
0  0.552819 -0.496162  0.092149  0.572068 -0.669091 -1.588515 -
0.118582
1 -0.435198 -0.496162  0.092149 -0.936344 -0.669091  1.147951
1.248598
2 -0.984097 -0.496162  0.092149  0.572068  0.246917 -1.588515 -
0.118582
3 -0.215639 -0.496162  0.092149 -0.936344  1.162925 -0.676360
1.248598
4 -0.544978 -0.496162  1.077303 -0.936344 -1.585099  1.147951 -
0.118582

          7         8
```

```
0 -0.208969 -1.155640
1  0.087370 -0.329380
2 -0.617879 -0.659884
3  0.435373  0.662132
4 -0.044313 -0.164128
```