```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/WA_Fn-UseC_-HR-Employee-Attrition.csv')
df
```

```
df.head()
```

```
#Here output column contains values yes/or, so which is required to encode for the
#correlation and heat map to see the relation b/w o/p feature and i/p features
df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	StandardHours	Stoc
0	41	1	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	80	
1	49	0	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	80	
2	37	1	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	80	
3	33	0	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3	80	
4	27	0	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4	80	

5 rows × 35 columns

```
df.tail()
```

```
df.isna().sum()
```

```
df.dtypes
```

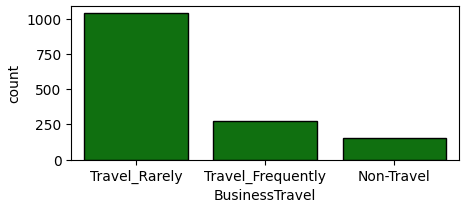
```
df['Attrition'].value_counts() #Highly unbalanced data
```

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

```
df['BusinessTravel'].value_counts()
```

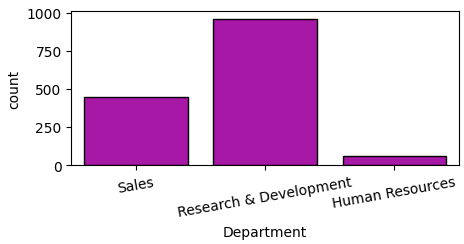
```
plt.figure(figsize=(5,2))
sns.countplot(x='BusinessTravel',data=df,color='g',edgecolor='k')
```

```
<Axes: xlabel='BusinessTravel', ylabel='count'>
```



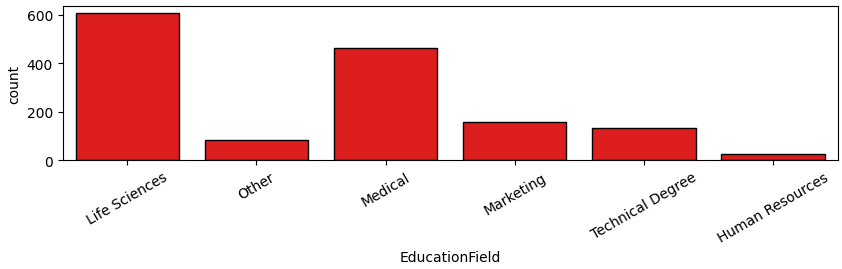

```
plt.figure(figsize=(5,2))
sns.countplot(x='Department',data=df,color='m',edgecolor='k')
plt.xticks(rotation=10)
```

```
([0, 1, 2],
 [Text(0, 0, 'Sales'),
  Text(1, 0, 'Research & Development'),
  Text(2, 0, 'Human Resources')])
```



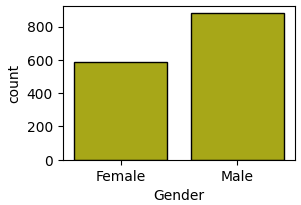

```
plt.figure(figsize=(10,2))
sns.countplot(x='EducationField',data=df,color='r',edgecolor='k')
plt.xticks(rotation=30)
```

```
([0, 1, 2, 3, 4, 5],
 [Text(0, 0, 'Life Sciences'),
  Text(1, 0, 'Other'),
  Text(2, 0, 'Medical'),
  Text(3, 0, 'Marketing'),
  Text(4, 0, 'Technical Degree'),
  Text(5, 0, 'Human Resources')])
```



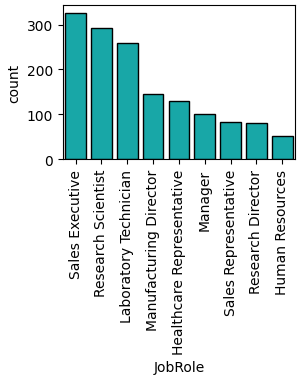

```
plt.figure(figsize=(3,2))
sns.countplot(x='Gender',data=df,color='y',edgecolor='k')
```

```
<Axes: xlabel='Gender', ylabel='count'>
```



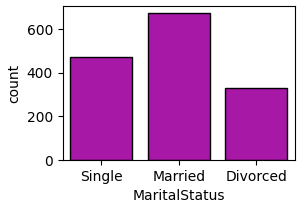

```
plt.figure(figsize=(3,2))
sns.countplot(x='JobRole',data=df,color='c',edgecolor='k')
plt.xticks(rotation='vertical')
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8],
 [Text(0, 0, 'Sales Executive'),
  Text(1, 0, 'Research Scientist'),
  Text(2, 0, 'Laboratory Technician'),
  Text(3, 0, 'Manufacturing Director'),
  Text(4, 0, 'Healthcare Representative'),
  Text(5, 0, 'Manager'),
  Text(6, 0, 'Sales Representative'),
  Text(7, 0, 'Research Director'),
  Text(8, 0, 'Human Resources')])
```



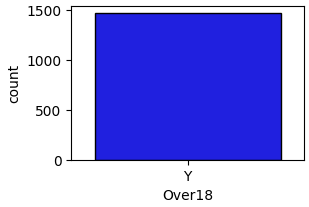

```
plt.figure(figsize=(3,2))
sns.countplot(x='MaritalStatus',data=df,color='m',edgecolor='k')
```

```
<Axes: xlabel='MaritalStatus', ylabel='count'>
```



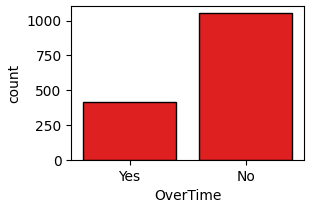

```
plt.figure(figsize=(3,2))
sns.countplot(x='Over18',data=df,color='b',edgecolor='k')
```

```
<Axes: xlabel='Over18', ylabel='count'>
```




```
plt.figure(figsize=(3,2))
sns.countplot(x='OverTime',data=df,color='r',edgecolor='k')
```

```
<Axes: xlabel='OverTime', ylabel='count'>
```




```
print(df['EmployeeCount'].unique())
print(df['StandardHours'].unique())
```

```
[1]
[80]
```

```
corr=df.corr()
corr
```

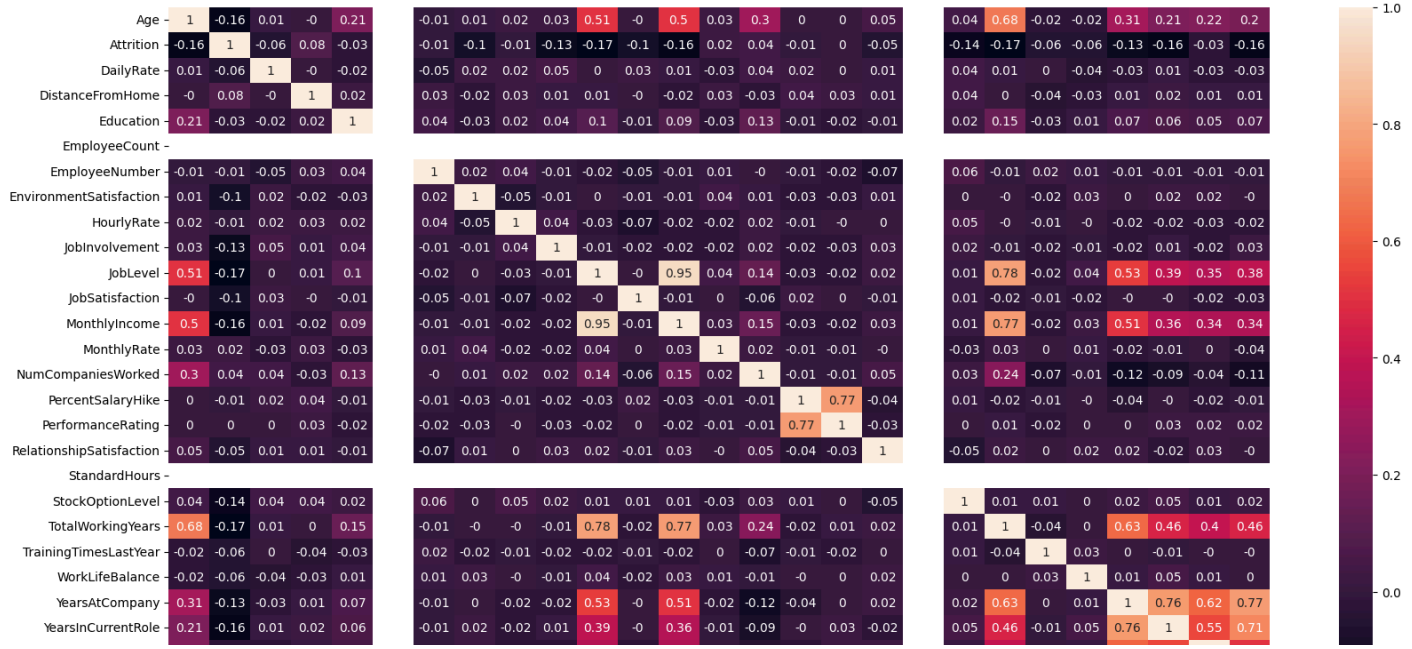
```
<ipython-input-843-7d5195e2bf4d>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid c
  corr=df.corr()
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	...	Relationsh
Age	1.000000	-0.159205	0.010661	-0.001686	0.208034	NaN	-0.010145	0.010146	0.024287	0.029820	...	
Attrition	-0.159205	1.000000	-0.056652	0.077924	-0.031373	NaN	-0.010577	-0.103369	-0.006846	-0.130016	...	
DailyRate	0.010661	-0.056652	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.018355	0.023381	0.046135	...	
DistanceFromHome	-0.001686	0.077924	-0.004985	1.000000	0.021042	NaN	0.032916	-0.016075	0.031131	0.008783	...	
Education	0.208034	-0.031373	-0.016806	0.021042	1.000000	NaN	0.042070	-0.027128	0.016775	0.042438	...	
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
EmployeeNumber	-0.010145	-0.010577	-0.050990	0.032916	0.042070	NaN	1.000000	0.017621	0.035179	-0.006888	...	
EnvironmentSatisfaction	0.010146	-0.103369	0.018355	-0.016075	-0.027128	NaN	0.017621	1.000000	-0.049857	-0.008278	...	
HourlyRate	0.024287	-0.006846	0.023381	0.031131	0.016775	NaN	0.035179	-0.049857	1.000000	0.042861	...	
JobInvolvement	0.029820	-0.130016	0.046135	0.008783	0.042438	NaN	-0.006888	-0.008278	0.042861	1.000000	...	
JobLevel	0.509604	-0.169105	0.002966	0.005303	0.101589	NaN	-0.018519	0.001212	-0.027853	-0.012630	...	
JobSatisfaction	-0.004892	-0.103481	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.006784	-0.071335	-0.021476	...	
MonthlyIncome	0.497855	-0.159840	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.006259	-0.015794	-0.015271	...	
MonthlyRate	0.028051	0.015170	-0.032182	0.027473	-0.026084	NaN	0.012648	0.037600	-0.015297	-0.016322	...	
NumCompaniesWorked	0.299635	0.043494	0.038153	-0.029251	0.126317	NaN	-0.001251	0.012594	0.022157	0.015012	...	
PercentSalaryHike	0.003634	-0.013478	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.031701	-0.009062	-0.017205	...	
PerformanceRating	0.001904	0.002889	0.000473	0.027110	-0.024539	NaN	-0.020359	-0.029548	-0.002172	-0.029071	...	
RelationshipSatisfaction	0.053535	-0.045872	0.007846	0.006557	-0.009118	NaN	-0.069861	0.007665	0.001330	0.034297	...	
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
StockOptionLevel	0.037510	-0.137145	0.042143	0.044872	0.018422	NaN	0.062227	0.003432	0.050263	0.021523	...	
TotalWorkingYears	0.680381	-0.171063	0.014515	0.004628	0.148280	NaN	-0.014365	-0.002693	-0.002334	-0.005533	...	
TrainingTimesLastYear	-0.019621	-0.059478	0.002453	-0.036942	-0.025100	NaN	0.023603	-0.019359	-0.008548	-0.015338	...	
WorkLifeBalance	-0.021490	-0.063939	-0.037848	-0.026556	0.009819	NaN	0.010309	0.027627	-0.004607	-0.014617	...	
YearsAtCompany	0.311309	-0.134392	-0.034055	0.009508	0.069114	NaN	-0.011240	0.001458	-0.019582	-0.021355	...	
YearsInCurrentRole	0.212901	-0.160545	0.009932	0.018845	0.060236	NaN	-0.008416	0.018007	-0.024106	0.008717	...	
YearsSinceLastPromotion	0.216513	-0.033019	-0.033229	0.010029	0.054254	NaN	-0.009019	0.016194	-0.026716	-0.024184	...	
YearsWithCurrManager	0.202089	-0.156199	-0.026363	0.014406	0.069065	NaN	-0.009197	-0.004999	-0.020123	0.025976	...	

27 rows × 27 columns

```
plt.figure(figsize=(20,10))
sns.heatmap(corr.round(2),annot=True)
```

<Axes: >




```
#Feature Selection
corr_pairs=[]

for i in range(len(corr.columns)):
  for j in range(i):
    if corr.iloc[i,j]>0.90:
      corr_pairs.append((corr.columns[i],corr.columns[j],corr.iloc[i,j]))
corr_pairs
```

```
[('MonthlyIncome', 'JobLevel', 0.9502999134798473)]
```

```
df['MonthlyIncome'].corr(df['Attrition'])# get correlation values of features in the corr_pairs
```

```
-0.15983958238498835
```

```
df['JobLevel'].corr(df['Attrition'])
```

```
-0.16910475093102642
```

```
corr.Attrition
#from above coor_pairs we've to choose one feature and drop the other inorder to avoid duplicate features in the data.
#from 'MonthlyIncome',  correlation pair Joblevel is dropped as which have less corrulation with output feature  as compared with Mont
```

```
df1=pd.get_dummies(df[['BusinessTravel','Department','EducationField','Gender','JobRole',
                       'MaritalStatus','OverTime']],drop_first=True)
df1
```

```
dfe=pd.concat([df,df1],axis=1)
dfe
```

```
dfe.columns
```

```
#Since,over 18,EmployeeCount',StandardHours consists of only one value('y') and 1 respectively, it can be dropped.
#from 'MonthlyIncome',  correlation pair Joblevel is dropped as which have less corrulation with output feature  as compared with Monthly income.
dfe.drop(['EmployeeCount','StandardHours', 'BusinessTravel','Department','EducationField','Gender','JobRole',
         'MaritalStatus','OverTime','Over18','JobLevel'],axis=1,inplace=True)
dfe
```

```
des=dfe.describe()
des
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobSatisfaction	...	JobRole_Laboratory Technician
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	...	1470.000000
mean	36.923810	0.161224	802.485714	9.192517	2.912925	1024.865306	2.721769	65.891156	2.729932	2.728571	...	0.176190
std	9.135373	0.367863	403.509100	8.106864	1.024165	602.024335	1.093082	20.329428	0.711561	1.102846	...	0.381112
min	18.000000	0.000000	102.000000	1.000000	1.000000	1.000000	1.000000	30.000000	1.000000	1.000000	...	0.000000
25%	30.000000	0.000000	465.000000	2.000000	2.000000	491.250000	2.000000	48.000000	2.000000	2.000000	...	0.000000
50%	36.000000	0.000000	802.000000	7.000000	3.000000	1020.500000	3.000000	66.000000	3.000000	3.000000	...	0.000000
75%	43.000000	0.000000	1157.000000	14.000000	4.000000	1555.750000	4.000000	83.750000	3.000000	4.000000	...	0.000000
max	60.000000	1.000000	1499.000000	29.000000	5.000000	2068.000000	4.000000	100.000000	4.000000	4.000000	...	1.000000

8 rows × 45 columns

```
#Boxplot
for i in dfe.columns:
  plt.subplots(1, sharex=True)
  sns.boxplot(dfe[i], linewidth= 1.0)
```

```
sns.boxplot(data=dfe).set(title="Box Plot of Employee Attrition & Performance")
plt.xticks(rotation='vertical')
```

```
features= dfe.columns[dfe.columns != 'Attrition']
features
```

```
#Outlier removal by capping the oupper and lower bound datas
def iqr_rem(df,cols):
  for col in cols:
    q1=df[col].quantile(0.25)
    q3=df[col].quantile(0.75)
    IQR=q3-q1
    upper_bound=q3+(1.5*IQR)
    lower_bound=q3-(1.5*IQR)
    dfe[col]=np.where(dfe[col]>upper_bound,upper_bound,np.where(dfe[col]<lower_bound,lower_bound,df[col]))
```

```
iqr_rem(dfe,features)#caping the outliers
```

```
#check whether outliers has removed or not
sns.boxplot(data=dfe).set(title="Box Plot of Employee Attrition & Performance")
plt.xticks(rotation='vertical')
```

```
x=dfe.drop(['Attrition'],axis=1)
x.values
```

```
y=dfe['Attrition']
y.values
```

```
array([1, 0, 1, ..., 0, 0, 0])
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
x_train.values
```

```
y_train.values
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
from sklearn.preprocessing import StandardScaler
norm=StandardScaler()
norm.fit(x_train)
x_train=norm.transform(x_train)
x_test=norm.transform(x_test)
x_train
```

```
from sklearn.naive_bayes import BernoulliNB
model=BernoulliNB()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
```

```
array([0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0])
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,ConfusionMatrixDisplay,classification_report
print("Score is",accuracy_score(y_test,y_pred))
cm=confusion_matrix(y_test,y_pred)
print(cm)
print(classification_report(y_test,y_pred))
labels=['0','1']
cmd=ConfusionMatrixDisplay(cm,display_labels=labels)
cmd.plot()
plt.title("Confusion matrix Display")
```

```
Score is 0.8367346938775511
[[336  28]
 [ 44  33]]
              precision    recall  f1-score   support

           0       0.88      0.92      0.90       364
           1       0.54      0.43      0.48        77

    accuracy                           0.84       441
   macro avg       0.71      0.68      0.69       441
weighted avg       0.82      0.84      0.83       441

Text(0.5, 1.0, 'Confusion matrix Display')
```