Employee Performance Rating

```
!pip install --upgrade scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Collecting scikit-learn
  Downloading scikit_learn-1.4.1.post1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.1/12.1 MB 36.4 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0,>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.25.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.4.0)
Installing collected packages: scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
    Uninstalling scikit-learn-1.2.2:
      Successfully uninstalled scikit-learn-1.2.2
Successfully installed scikit-learn-1.4.1.post1
```

```
import sklearn
print(sklearn.__version__)
```

```
1.4.1.post1
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
#Load the data
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
df=pd.read_excel('INX_Future_Inc_Employee_Performance_CDS_Project2_Data_V1.8.xls')
```

```
df
```

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRole	BusinessTravelFrequency	DistanceFromHo
0	E1001000	32	Male	Marketing	Single	Sales	Sales Executive	Travel_Rarely	
1	E1001006	47	Male	Marketing	Single	Sales	Sales Executive	Travel_Rarely	
2	E1001007	40	Male	Life Sciences	Married	Sales	Sales Executive	Travel_Frequently	
3	E1001009	41	Male	Human Resources	Divorced	Human Resources	Manager	Travel_Rarely	
4	E1001010	60	Male	Marketing	Single	Sales	Sales Executive	Travel_Rarely	
...	...	...	...	...	...	...	...	...	
1195	E100992	27	Female	Medical	Divorced	Sales	Sales Executive	Travel_Frequently	
1196	E100993	37	Male	Life Sciences	Single	Development	Senior Developer	Travel_Rarely	
1197	E100994	50	Male	Medical	Married	Development	Senior Developer	Travel_Rarely	
1198	E100995	34	Female	Medical	Single	Data Science	Data Scientist	Travel_Rarely	
1199	E100998	24	Female	Life Sciences	Single	Sales	Sales Executive	Travel_Rarely	

1200 rows × 28 columns

```
df.head()
```

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobR
0	E1001000	32	Male	Marketing	Single	Sales	Sa Execu
1	E1001006	47	Male	Marketing	Single	Sales	Sa Execu
2	E1001007	40	Male	Life Sciences	Married	Sales	Sa Execu
3	E1001009	41	Male	Human Resources	Divorced	Human Resources	Mana
4	E1001010	60	Male	Marketing	Single	Sales	Sa Execu

5 rows × 28 columns

```
df.tail()
```

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJ
1195	E100992	27	Female	Medical	Divorced	Sales	Ex
1196	E100993	37	Male	Life Sciences	Single	Development	De
1197	E100994	50	Male	Medical	Married	Development	De
1198	E100995	34	Female	Medical	Single	Data Science	S
1199	E100998	24	Female	Life Sciences	Single	Sales	Ex

5 rows × 28 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 28 columns):
 #   Column                        Non-Null Count  Dtype 
---  ------                        --------------  ----- 
 0   EmpNumber                     1200 non-null   object
 1   Age                           1200 non-null   int64 
 2   Gender                        1200 non-null   object
 3   EducationBackground           1200 non-null   object
 4   MaritalStatus                 1200 non-null   object
 5   EmpDepartment                 1200 non-null   object
 6   EmpJobRole                    1200 non-null   object
 7   BusinessTravelFrequency       1200 non-null   object
 8   DistanceFromHome              1200 non-null   int64 
 9   EmpEducationLevel             1200 non-null   int64 
 10  EmpEnvironmentSatisfaction    1200 non-null   int64 
 11  EmpHourlyRate                 1200 non-null   int64 
 12  EmpJobInvolvement             1200 non-null   int64 
 13  EmpJobLevel                   1200 non-null   int64 
 14  EmpJobSatisfaction            1200 non-null   int64 
 15  NumCompaniesWorked            1200 non-null   int64 
 16  OverTime                      1200 non-null   object
 17  EmpLastSalaryHikePercent      1200 non-null   int64 
 18  EmpRelationshipSatisfaction   1200 non-null   int64 
 19  TotalWorkExperienceInYears    1200 non-null   int64 
 20  TrainingTimesLastYear         1200 non-null   int64 
 21  EmpWorkLifeBalance            1200 non-null   int64 
 22  ExperienceYearsAtThisCompany  1200 non-null   int64 
 23  ExperienceYearsInCurrentRole  1200 non-null   int64 
 24  YearsSinceLastPromotion       1200 non-null   int64 
 25  YearsWithCurrManager          1200 non-null   int64 
 26  Attrition                     1200 non-null   object
 27  PerformanceRating             1200 non-null   int64 
dtypes: int64(19), object(9)
memory usage: 262.6+ KB
```

We have 1200 Rows and 28 columns in which 19 columns have int datatype and 9 have object datatype and memory usages by this data is 262.6+ KB with no null values present

```
df.describe()
```

	Age	DistanceFromHome	EmpEducationLevel	EmpEnvironmentSatisfaction
count	1200.000000	1200.000000	1200.00000	1200.000000
mean	36.918333	9.165833	2.89250	2.715833
std	9.087289	8.176636	1.04412	1.090599
min	18.000000	1.000000	1.00000	1.000000
25%	30.000000	2.000000	2.00000	2.000000
50%	36.000000	7.000000	3.00000	3.000000
75%	43.000000	14.000000	4.00000	4.000000
max	60.000000	29.000000	5.00000	4.000000

```
df.columns
```

```
Index(['EmpNumber', 'Age', 'Gender', 'EducationBackground', 'MaritalStatus',
       'EmpDepartment', 'EmpJobRole', 'BusinessTravelFrequency',
       'DistanceFromHome', 'EmpEducationLevel', 'EmpEnvironmentSatisfaction',
       'EmpHourlyRate', 'EmpJobInvolvement', 'EmpJobLevel',
       'EmpJobSatisfaction', 'NumCompaniesWorked', 'OverTime',
       'EmpLastSalaryHikePercent', 'EmpRelationshipSatisfaction',
       'TotalWorkExperienceInYears', 'TrainingTimesLastYear',
       'EmpWorkLifeBalance', 'ExperienceYearsAtThisCompany',
       'ExperienceYearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager', 'Attrition', 'PerformanceRating'],
      dtype='object')
```

```
for i in df:
    print(i)
    print(df[i].unique())
```

```
EmpNumber
['E1001000' 'E1001006' 'E1001007' ... 'E100994' 'E100995' 'E100998']
Age
[32 47 40 41 60 27 50 28 36 38 44 30 29 42 34 39 56 53 35 52 33 25 45 23
 26 54 37 24 49 55 43 51 22 31 58 20 21 48 19 18 59 46 57]
Gender
['Male' 'Female']
EducationBackground
['Marketing' 'Life Sciences' 'Human Resources' 'Medical' 'Other'
 'Technical Degree']
MaritalStatus
['Single' 'Married' 'Divorced']
EmpDepartment
['Sales' 'Human Resources' 'Development' 'Data Science'
 'Research & Development' 'Finance']
EmpJobRole
['Sales Executive' 'Manager' 'Developer' 'Sales Representative'
 'Human Resources' 'Senior Developer' 'Data Scientist'
 'Senior Manager R&D' 'Laboratory Technician' 'Manufacturing Director'
 'Research Scientist' 'Healthcare Representative' 'Research Director'
 'Manager R&D' 'Finance Manager' 'Technical Architect' 'Business Analyst'
 'Technical Lead' 'Delivery Manager']
BusinessTravelFrequency
['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
DistanceFromHome
[10 14  5 16  8  1 24  3 27 19  9  2  7  6 26 11 23 13 29 12  4 25 21 18
 20 17 22 28 15]
EmpEducationLevel
[3 4 2 5 1]
EmpEnvironmentSatisfaction
[4 2 1 3]
EmpHourlyRate
[ 55  42  48  73  84  32  54  67  63  81  49  99  57  96  44  86  83  61
  80  31  46  79  77  60  76  41  71  66  38  72  95  82  75  93  64  36
  69  33  35  40  43  51  52  98  45  37  94  59  88  50  65  53  56  78
  74  91  62  87  68  34  70 100  92  39  97  47  90  85  30  58  89]
EmpJobInvolvement
[3 2 1 4]
EmpJobLevel
[2 3 5 1 4]
EmpJobSatisfaction
[4 1 2 3]
NumCompaniesWorked
[1 2 5 3 8 7 9 4 6 0]
OverTime
['No' 'Yes']
EmpLastSalaryHikePercent
[12 21 15 14 13 23 11 20 19 17 18 22 16 24 25]
EmpRelationshipSatisfaction
```

```
[4 3 2 1]
TotalWorkExperienceInYears
[10 20 23  9  4 28  1  7 12 30  5  2 19 16 34  6  8 11 17  3 14 26 13 22
  0 29 18 35 33 31 24 15 32 21 27 25 36 37 38 40]
TrainingTimesLastYear
[2 1 4 5 6 3 0]
EmpWorkLifeBalance
[2 3 4 1]
```

Check the null values present or not

```
df.isnull().sum()/len(df)*100
```

```
EmpNumber                       0.0
Age                             0.0
Gender                          0.0
EducationBackground             0.0
MaritalStatus                   0.0
EmpDepartment                   0.0
EmpJobRole                      0.0
BusinessTravelFrequency         0.0
DistanceFromHome                0.0
EmpEducationLevel               0.0
EmpEnvironmentSatisfaction      0.0
EmpHourlyRate                   0.0
EmpJobInvolvement               0.0
EmpJobLevel                     0.0
EmpJobSatisfaction              0.0
NumCompaniesWorked              0.0
OverTime                        0.0
EmpLastSalaryHikePercent        0.0
EmpRelationshipSatisfaction     0.0
TotalWorkExperienceInYears      0.0
TrainingTimesLastYear           0.0
EmpWorkLifeBalance              0.0
ExperienceYearsAtThisCompany    0.0
ExperienceYearsInCurrentRole    0.0
YearsSinceLastPromotion         0.0
YearsWithCurrManager            0.0
Attrition                       0.0
PerformanceRating               0.0
dtype: float64
```

```
df.drop('EmpNumber',axis=1,inplace=True)
```

separating cat and num columns

```
cat_col=df.select_dtypes(object)
cat_col
```

	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRole	Business
0	Male	Marketing	Single	Sales	Sales Executive	
1	Male	Marketing	Single	Sales	Sales Executive	
2	Male	Life Sciences	Married	Sales	Sales Executive	
3	Male	Human Resources	Divorced	Human Resources	Manager	
4	Male	Marketing	Single	Sales	Sales Executive	
...	...	...	...	...	...	
1195	Female	Medical	Divorced	Sales	Sales Executive	
1196	Male	Life Sciences	Single	Development	Senior Developer	

```
num_col=df.select_dtypes([int,float])
num_col
```

	Age	DistanceFromHome	EmpEducationLevel	EmpEnvironmentSatisfaction	EmpHourly
0	32	10	3	4	
1	47	14	4	4	
2	40	5	4	4	
3	41	10	4	2	
4	60	16	4	1	
...	...	...	...	...	
1195	27	3	1	4	
1196	37	10	2	4	
1197	50	28	1	4	
1198	34	9	3	4	
1199	24	3	2	1	

1200 rows × 19 columns

```
for i in cat_col:
    print(i)
    print(cat_col[i].unique())
```

```
Gender
['Male' 'Female']
EducationBackground
['Marketing' 'Life Sciences' 'Human Resources' 'Medical' 'Other'
 'Technical Degree']
MaritalStatus
['Single' 'Married' 'Divorced']
EmpDepartment
['Sales' 'Human Resources' 'Development' 'Data Science'
 'Research & Development' 'Finance']
EmpJobRole
['Sales Executive' 'Manager' 'Developer' 'Sales Representative'
 'Human Resources' 'Senior Developer' 'Data Scientist'
 'Senior Manager R&D' 'Laboratory Technician' 'Manufacturing Director'
 'Research Scientist' 'Healthcare Representative' 'Research Director'
 'Manager R&D' 'Finance Manager' 'Technical Architect' 'Business Analyst'
 'Technical Lead' 'Delivery Manager']
BusinessTravelFrequency
['Travel_Rarely' 'Travel_Frequently' 'Non-Travel']
OverTime
['No' 'Yes']
Attrition
['No' 'Yes']
```

```
for i in num_col:
    print(i)
    print(num_col[i].unique())
```

```
Age
[32 47 40 41 60 27 50 28 36 38 44 30 29 42 34 39 56 53 35 52 33 25 45 23
 26 54 37 24 49 55 43 51 22 31 58 20 21 48 19 18 59 46 57]
DistanceFromHome
[10 14  5 16  8  1 24  3 27 19  9  2  7  6 26 11 23 13 29 12  4 25 21 18
 20 17 22 28 15]
EmpEducationLevel
[3 4 2 5 1]
EmpEnvironmentSatisfaction
[4 2 1 3]
EmpHourlyRate
[ 55  42  48  73  84  32  54  67  63  81  49  99  57  96  44  86  83  61
  80  31  46  79  77  60  76  41  71  66  38  72  95  82  75  93  64  36
  69  33  35  40  43  51  52  98  45  37  94  59  88  50  65  53  56  78
  74  91  62  87  68  34  70 100  92  39  97  47  90  85  30  58  89]
EmpJobInvolvement
[3 2 1 4]
EmpJobLevel
[2 3 5 1 4]
EmpJobSatisfaction
[4 1 2 3]
NumCompaniesWorked
[1 2 5 3 8 7 9 4 6 0]
EmpLastSalaryHikePercent
[12 21 15 14 13 23 11 20 19 17 18 22 16 24 25]
EmpRelationshipSatisfaction
[4 3 2 1]
TotalWorkExperienceInYears
[10 20 23  9  4 28  1  7 12 30  5  2 19 16 34  6  8 11 17  3 14 26 13 22
  0 29 18 35 33 31 24 15 32 21 27 25 36 37 38 40]
```

```
TrainingTimesLastYear
[2 1 4 5 6 3 0]
EmpWorkLifeBalance
[2 3 4 1]
ExperienceYearsAtThisCompany
[10  7 18 21  2  9  8  1  5 22  4  0 34  6 15  3 13 26 20 12 31 16 11 14
 17 19 32 24 33 29 25 36 30 23 27 37 40]
ExperienceYearsInCurrentRole
[ 7 13  6  2  0  1  3  8  9  5  4 14 12 10 16 11 15 17 18]
YearsSinceLastPromotion
[ 0  1 12  2  3 11  7  4  5 14  8 15  6 10 13  9]
YearsWithCurrManager
[ 8  7 12  6  2  5  0  4 13  3  1 16  9 17 11 10 14 15]
PerformanceRating
[3 4 2]
```

Encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
for i in cat_col:
    cat_col[i]=le.fit_transform(cat_col[i])
    print(cat_col[i].unique())
    print(le.classes_)
```

```
[1 0]
['Female' 'Male']
[2 1 0 3 4 5]
['Human Resources' 'Life Sciences' 'Marketing' 'Medical' 'Other'
 'Technical Degree']
[2 1 0]
['Divorced' 'Married' 'Single']
[5 3 1 0 4 2]
['Data Science' 'Development' 'Finance' 'Human Resources'
 'Research & Development' 'Sales']
[13  8  3 14  6 15  1 16  7 10 12  5 11  9  4 17  0 18  2]
['Business Analyst' 'Data Scientist' 'Delivery Manager' 'Developer'
 'Finance Manager' 'Healthcare Representative' 'Human Resources'
 'Laboratory Technician' 'Manager' 'Manager R&D' 'Manufacturing Director'
 'Research Director' 'Research Scientist' 'Sales Executive'
 'Sales Representative' 'Senior Developer' 'Senior Manager R&D'
 'Technical Architect' 'Technical Lead']
[2 1 0]
['Non-Travel' 'Travel_Frequently' 'Travel_Rarely']
[0 1]
['No' 'Yes']
[0 1]
['No' 'Yes']
```

```
new_df=pd.concat([cat_col,num_col],axis=1)
```

```
new_df
```

	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRole	Business
0	1	2	2	5	13	
1	1	2	2	5	13	
2	1	1	1	5	13	
3	1	0	0	3	8	
4	1	2	2	5	13	
...	...	...	...	...	...	
1195	0	3	0	5	13	
1196	1	1	2	1	15	
1197	1	3	1	1	15	
1198	0	3	2	0	1	
1199	0	1	2	5	13	

1200 rows × 27 columns

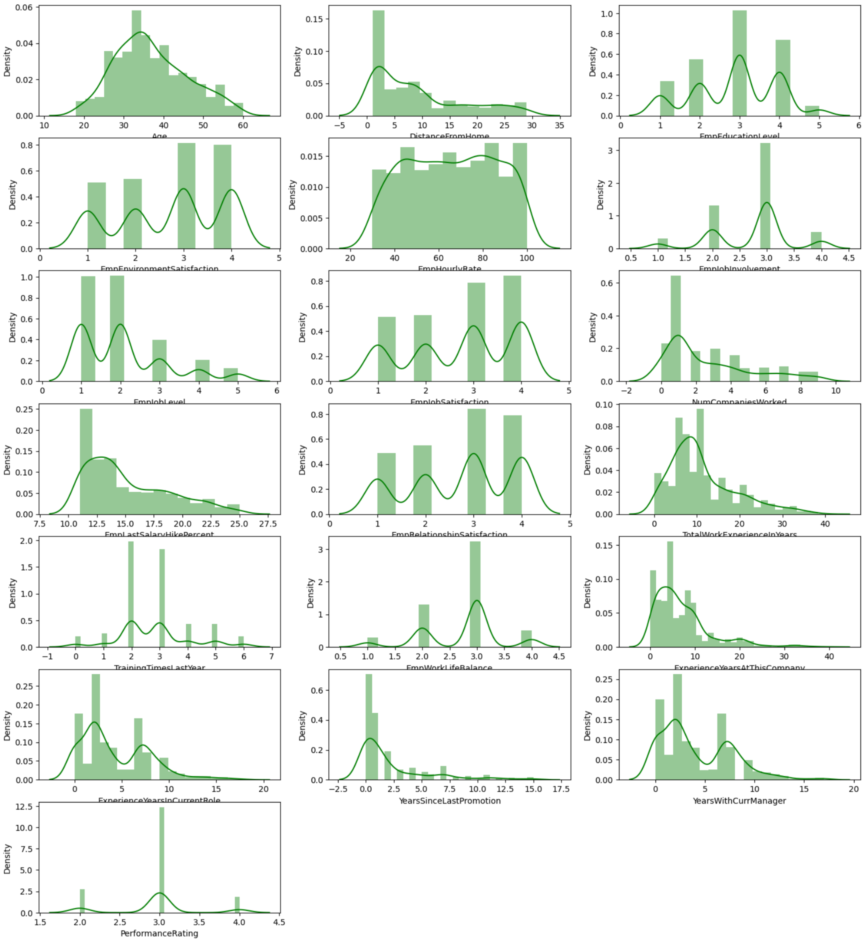
```
new_df.columns
```

```
Index(['Gender', 'EducationBackground', 'MaritalStatus', 'EmpDepartment',
       'EmpJobRole', 'BusinessTravelFrequency', 'OverTime', 'Attrition', 'Age',
       'DistanceFromHome', 'EmpEducationLevel', 'EmpEnvironmentSatisfaction',
       'EmpHourlyRate', 'EmpJobInvolvement', 'EmpJobLevel',
       'EmpJobSatisfaction', 'NumCompaniesWorked', 'EmpLastSalaryHikePercent',
       'EmpRelationshipSatisfaction', 'TotalWorkExperienceInYears',
       'TrainingTimesLastYear', 'EmpWorkLifeBalance',
       'ExperienceYearsAtThisCompany', 'ExperienceYearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager', 'PerformanceRating'],
      dtype='object')
```

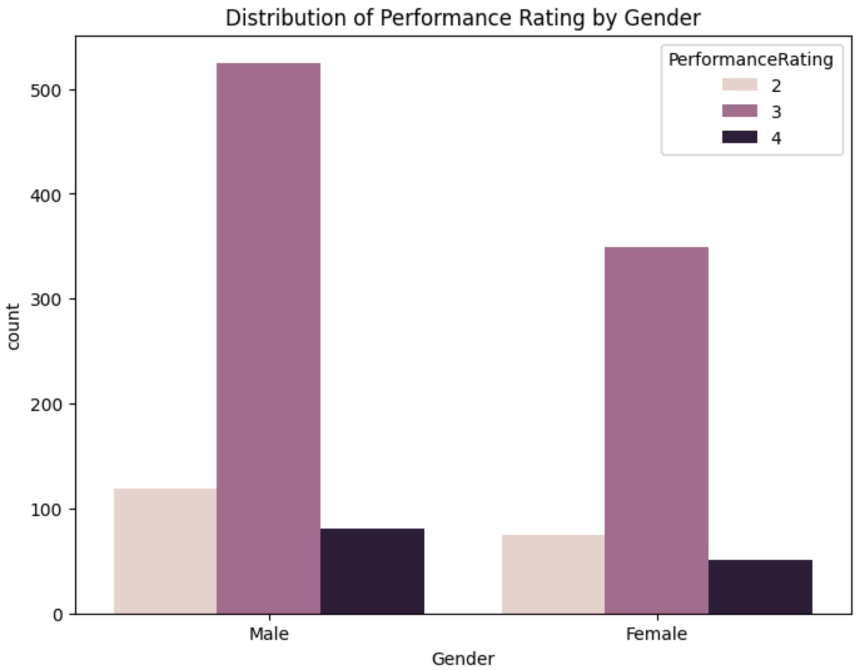
EDA

Numerical columns distribution

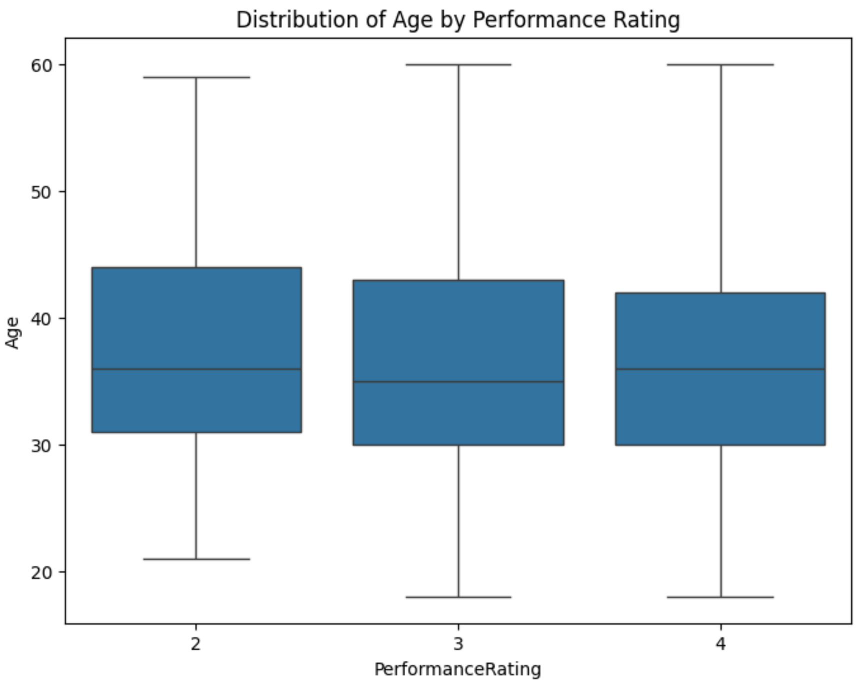
```
plt.figure(figsize= (18,20))
count=1
for i in num_col:

    plt.subplot(7,3,count)
    sns.distplot(num_col[i],color='green')
    count+=1
plt.show()
```


Age
DistanceFromHome
EmpEducationLevel
EmpEnvironmentSatisfaction
EmpHourlyRate
EmpJobInvolvement
EmpJobLevel
EmpJobSatisfaction
NumCompaniesWorked
EmpLastSalaryHikePercent
EmpRelationshipSatisfaction
TotalWorkExperienceInYears
TrainingTimesLastYear
EmpWorkLifeBalance
ExperienceYearsAtThisCompany
ExperienceYearsInCurrentRole
YearsSinceLastPromotion
YearsWithCurrManager
PerformanceRating
Density

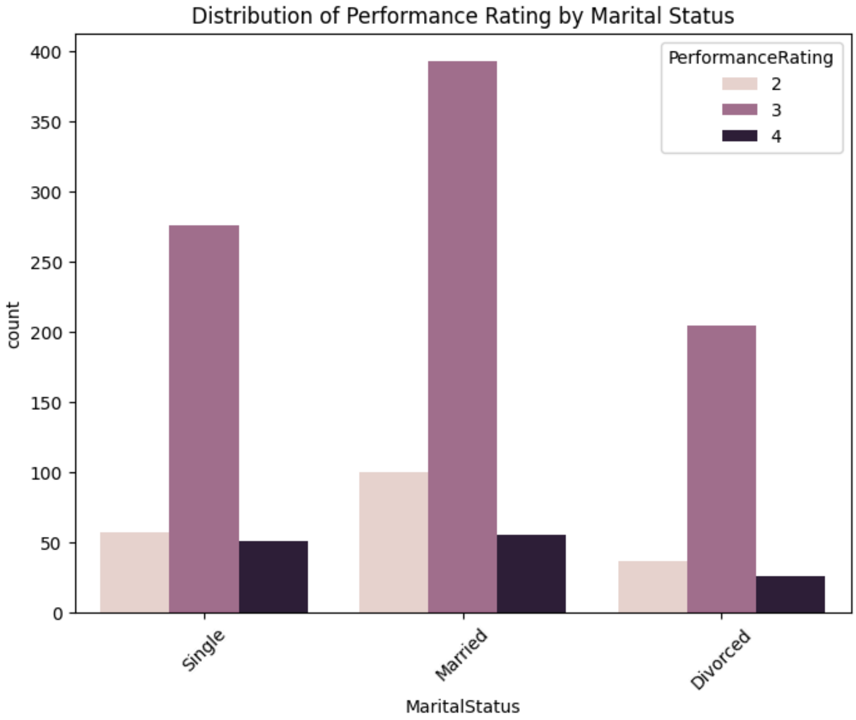
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Gender', hue='PerformanceRating')
plt.title('Distribution of Performance Rating by Gender')
plt.show()
```




```
# Example for 'Age' by Performance Rating
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='PerformanceRating', y='Age')
plt.title('Distribution of Age by Performance Rating')
plt.show()
```

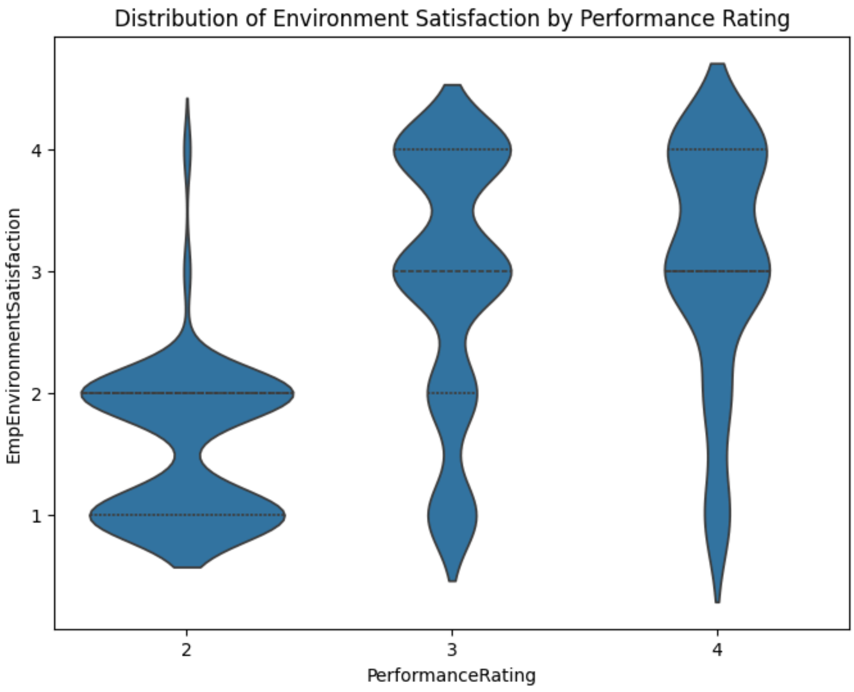



```
# Example for 'MaritalStatus' by Performance Rating
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='MaritalStatus', hue='PerformanceRating')
plt.title('Distribution of Performance Rating by Marital Status')
plt.xticks(rotation=45)
plt.show()
```



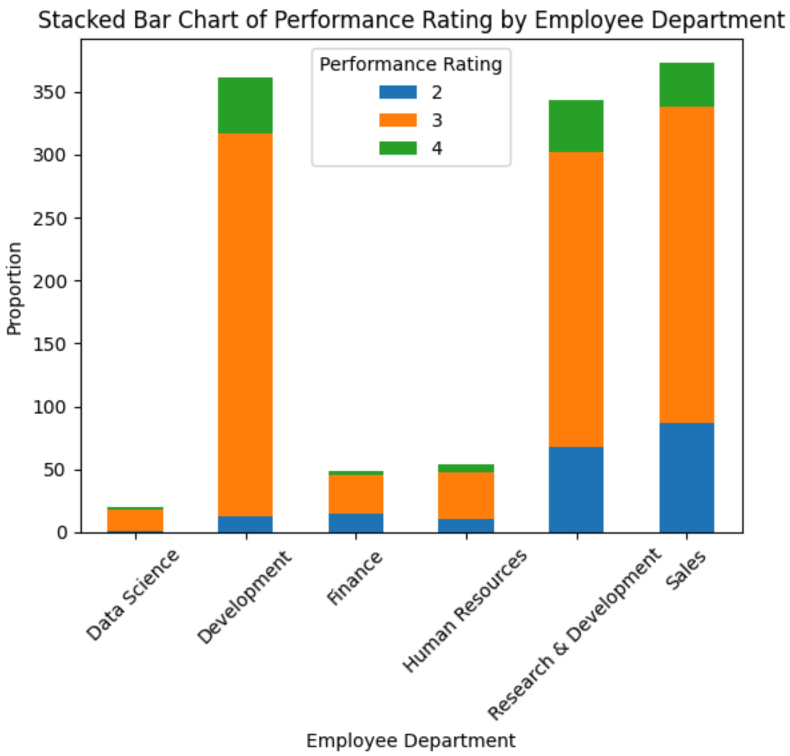

```
# Example for 'TotalWorkExperienceInYears'
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='TotalWorkExperienceInYears', hue='PerformanceRating', bins=20, kde=True)
plt.title('Distribution of Total Work Experience by Performance Rating')
plt.show()
```


```
# Example for 'EmpEnvironmentSatisfaction'
plt.figure(figsize=(8, 6))
sns.violinplot(data=df, x='PerformanceRating', y='EmpEnvironmentSatisfaction', inner='quart')
plt.title('Distribution of Environment Satisfaction by Performance Rating')
plt.show()
```

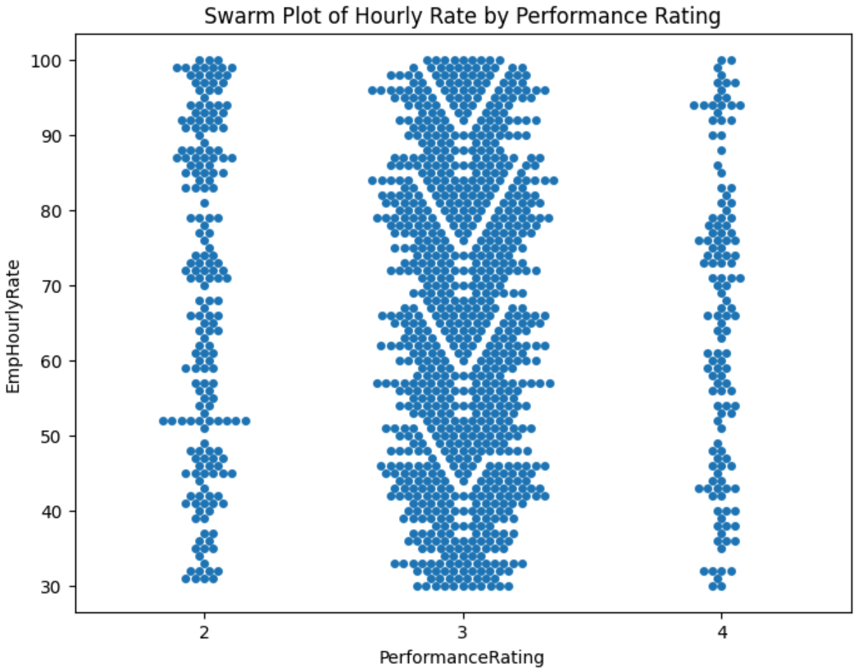



```
# Example for 'EmpDepartment' stacked bar chart
plt.figure(figsize=(10, 6))
cross_tab = pd.crosstab(df['EmpDepartment'], df['PerformanceRating'])
cross_tab.plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Performance Rating by Employee Department')
plt.xlabel('Employee Department')
plt.ylabel('Proportion')
plt.legend(title='Performance Rating')
plt.xticks(rotation=45)
plt.show()
```

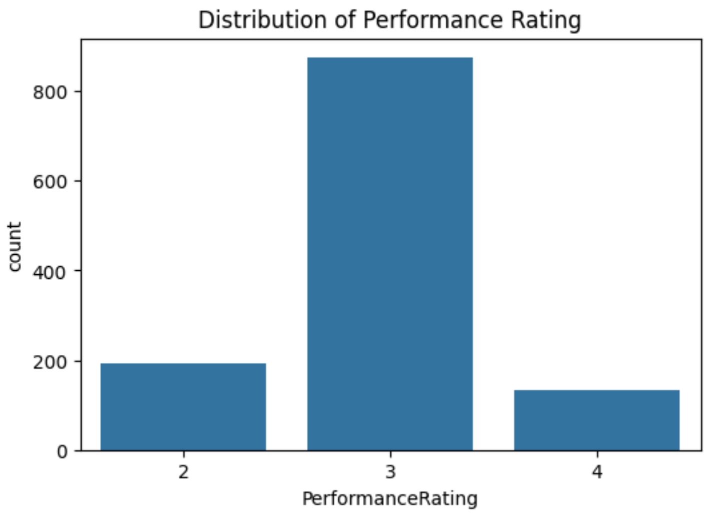
```
<Figure size 1000x600 with 0 Axes>
```



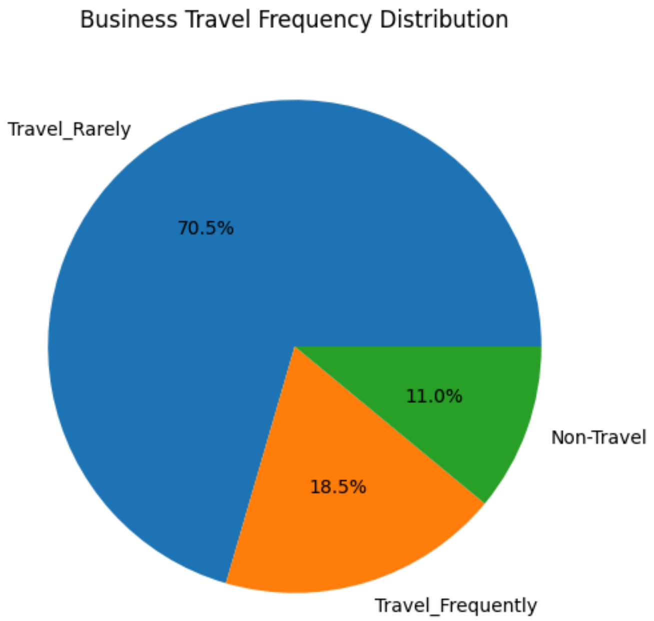

```
# Example for 'EmpHourlyRate'
plt.figure(figsize=(8, 6))
sns.swarmplot(data=df, x='PerformanceRating', y='EmpHourlyRate')
plt.title('Swarm Plot of Hourly Rate by Performance Rating')
plt.show()
```



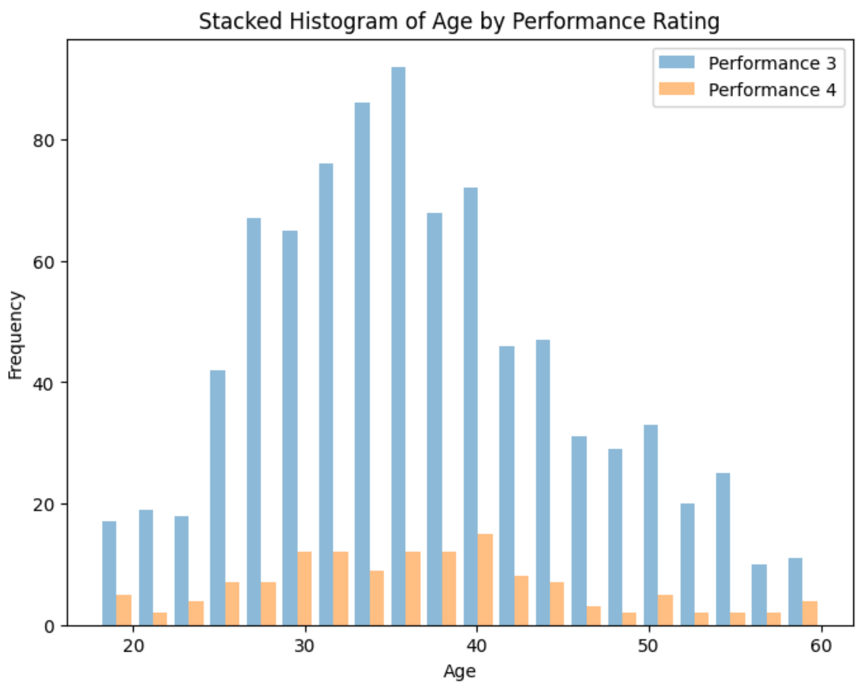

```
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='PerformanceRating')
plt.title('Distribution of Performance Rating')
plt.show()
```




```
# Example for 'BusinessTravelFrequency'
plt.figure(figsize=(6, 6))
df['BusinessTravelFrequency'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Business Travel Frequency Distribution')
plt.ylabel('')
plt.show()
```




```
# Example for 'Age' stacked histogram
plt.figure(figsize=(8, 6))
plt.hist([df[df['PerformanceRating'] == 3]['Age'], df[df['PerformanceRating'] == 4]['Age']],
         bins=20, alpha=0.5, label=['Performance 3', 'Performance 4'])
plt.title('Stacked Histogram of Age by Performance Rating')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

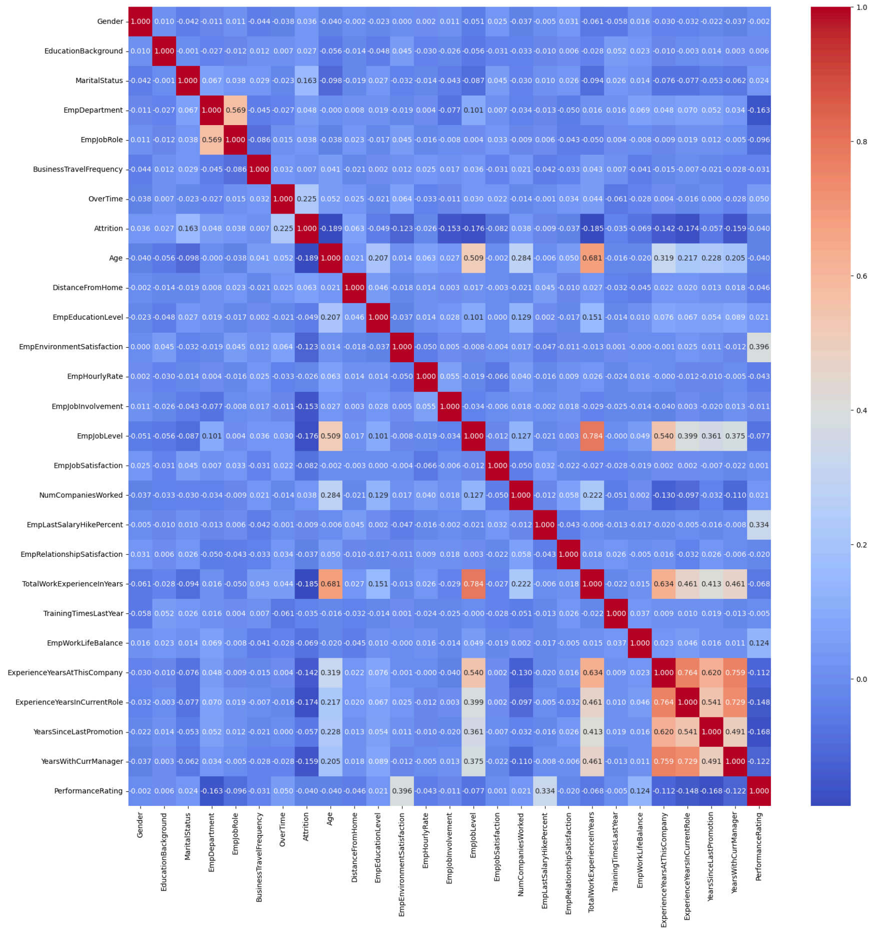



```
from wordcloud import WordCloud

text = " ".join(df['EmpJobRole'])
wordcloud = WordCloud(width=800, height=400, background_color='White').generate(text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Word Cloud of Job Roles')
plt.show()
```

Feature Selection

```
plt.figure(figsize=(20,20))
sns.heatmap(new_df.corr(),annot=True,cmap='coolwarm',fmt=".3f")
```

<Axes: >

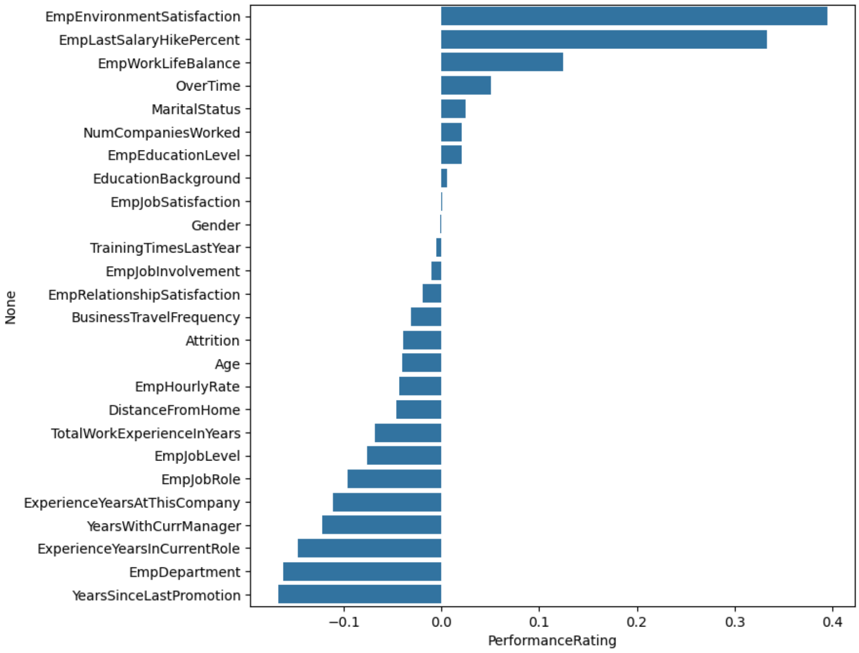


```
corr=new_df.corr()['PerformanceRating'].reset_index()
corr.sort_values('PerformanceRating',ascending=False)
```

	index	PerformanceRating
26	PerformanceRating	1.000000
11	EmpEnvironmentSatisfaction	0.395561
17	EmpLastSalaryHikePercent	0.333722
21	EmpWorkLifeBalance	0.124429
6	OverTime	0.050206
2	MaritalStatus	0.024172
16	NumCompaniesWorked	0.020980
10	EmpEducationLevel	0.020529
1	EducationBackground	0.005607
15	EmpJobSatisfaction	0.000606
0	Gender	-0.001780
20	TrainingTimesLastYear	-0.005443
13	EmpJobInvolvement	-0.010539
18	EmpRelationshipSatisfaction	-0.019502
5	BusinessTravelFrequency	-0.031025
7	Attrition	-0.039796
8	Age	-0.040164
12	EmpHourlyRate	-0.043116
9	DistanceFromHome	-0.046142
19	TotalWorkExperienceInYears	-0.068141
14	EmpJobLevel	-0.076632
4	EmpJobRole	-0.096209
22	ExperienceYearsAtThisCompany	-0.111645
25	YearsWithCurrManager	-0.122313
23	ExperienceYearsInCurrentRole	-0.147638
3	EmpDepartment	-0.162615
24	YearsSinceLastPromotion	-0.167629

```
corelation = pd.DataFrame(new_df.corr())
corelation = pd.DataFrame(corelation['PerformanceRating'])
corelation=corelation.sort_values('PerformanceRating',ascending=False)
indices_to_remove = ['PerformanceRating']
corelation = corelation.drop(indices_to_remove)
plt.figure(figsize=(8,8))
sns.barplot(x=corelation['PerformanceRating'],y=corelation.index)
```

```
<Axes: xlabel='PerformanceRating', ylabel='None'>
```




```
new_df.columns
```

```
Index(['Gender', 'EducationBackground', 'MaritalStatus', 'EmpDepartment',
       'EmpJobRole', 'BusinessTravelFrequency', 'OverTime', 'Attrition', 'Age',
       'DistanceFromHome', 'EmpEducationLevel', 'EmpEnvironmentSatisfaction',
       'EmpHourlyRate', 'EmpJobInvolvement', 'EmpJobLevel',
       'EmpJobSatisfaction', 'NumCompaniesWorked', 'EmpLastSalaryHikePercent',
       'EmpRelationshipSatisfaction', 'TotalWorkExperienceInYears',
       'TrainingTimesLastYear', 'EmpWorkLifeBalance',
       'ExperienceYearsAtThisCompany', 'ExperienceYearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager', 'PerformanceRating'],
      dtype='object')
```

```
pr=new_df[['EmpEnvironmentSatisfaction','EmpLastSalaryHikePercent','YearsSinceLastPromotion','EmpDepartment','ExperienceYearsInCurrentRo
```

```
pr
```

	EmpEnvironmentSatisfaction	EmpLastSalaryHikePercent	YearsSinceLastPromotion
0	4	12	0
1	4	12	1
2	4	21	1
3	2	15	12
4	1	14	2
...	...	...	...
1195	4	20	0
1196	4	17	0
1197	4	11	3
1198	4	14	7
1199	1	14	2

1200 rows × 9 columns

```
pr.columns
```

```
Index(['EmpEnvironmentSatisfaction', 'EmpLastSalaryHikePercent',
       'YearsSinceLastPromotion', 'EmpDepartment',
       'ExperienceYearsInCurrentRole', 'EmpHourlyRate', 'EmpJobRole',
       'TotalWorkExperienceInYears', 'PerformanceRating'],
      dtype='object')
```

1. **EmpEnvironmentSatisfaction**: This column likely represents the level of employee satisfaction with their work environment. It may contain values indicating how satisfied employees are with factors like office conditions, workspace, or workplace culture.
2. **EmpLastSalaryHikePercent**: This column probably represents the percentage increase in an employee's salary during their most recent salary hike or raise. It measures the extent to which an employee's salary has increased.
3. **YearsSinceLastPromotion**: This column likely indicates the number of years that have passed since an employee's last promotion within the organization. It measures the time interval between promotions.
4. **EmpDepartment**: This column may represent the department or division within the organization where the employee works. It could include values such as "Sales," "Marketing," "Human Resources," etc.
5. **ExperienceYearsInCurrentRole**: This column probably represents the number of years an employee has spent in their current role or position within the company. It measures the duration of time an employee has held their current job title.
6. **EmpHourlyRate**: This column may indicate the hourly wage or pay rate of employees. It represents the amount an employee earns per hour of work.
7. **EmpJobRole**: This column likely represents the job role or title of employees within the organization. It could include values such as "Manager," "Engineer," "Analyst," etc.
8. **TotalWorkExperienceInYears**: This column may represent the total number of years of work experience an employee has, possibly including work experience both within and outside the current organization.
9. **PerformanceRating**: This column is likely the target variable or label for your predictive model. It represents the performance rating assigned to employees. It could be a categorical variable with values such as "2," "3," and "4," indicating different levels of performance.

```
for i in pr:
    print(i)
    print(pr[i].unique())
```

```
EmpEnvironmentSatisfaction
[4 2 1 3]
EmpLastSalaryHikePercent
[12 21 15 14 13 23 11 20 19 17 18 22 16 24 25]
YearsSinceLastPromotion
[ 0  1 12  2  3 11  7  4  5 14  8 15  6 10 13  9]
EmpDepartment
[5 3 1 0 4 2]
ExperienceYearsInCurrentRole
[ 7 13  6  2  0  1  3  8  9  5  4 14 12 10 16 11 15 17 18]
EmpHourlyRate
[ 55  42  48  73  84  32  54  67  63  81  49  99  57  96  44  86  83  61
  80  31  46  79  77  60  76  41  71  66  38  72  95  82  75  93  64  36
  69  33  35  40  43  51  52  98  45  37  94  59  88  50  65  53  56  78
  74  91  62  87  68  34  70 100  92  39  97  47  90  85  30  58  89]
EmpJobRole
[13  8  3 14  6 15  1 16  7 10 12  5 11  9  4 17  0 18  2]
TotalWorkExperienceInYears
[10 20 23  9  4 28  1  7 12 30  5  2 19 16 34  6  8 11 17  3 14 26 13 22
  0 29 18 35 33 31 24 15 32 21 27 25 36 37 38 40]
PerformanceRating
[3 4 2]
```

```
pr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   EmpEnvironmentSatisfaction    1200 non-null   int64
 1   EmpLastSalaryHikePercent      1200 non-null   int64
 2   YearsSinceLastPromotion       1200 non-null   int64
 3   EmpDepartment                 1200 non-null   int64
 4   ExperienceYearsInCurrentRole  1200 non-null   int64
 5   EmpHourlyRate                 1200 non-null   int64
 6   EmpJobRole                    1200 non-null   int64
 7   TotalWorkExperienceInYears    1200 non-null   int64
 8   PerformanceRating             1200 non-null   int64
dtypes: int64(9)
memory usage: 84.5 KB
```

split training and testing data

```
from sklearn.model_selection import train_test_split
```

```
X=pr.drop(['PerformanceRating'],axis=1)
X
```

	EmpEnvironmentSatisfaction	EmpLastSalaryHikePercent	YearsSinceLastPromotion
0	4	12	0
1	4	12	1
2	4	21	1
3	2	15	12
4	1	14	2
...	...	...	...
1195	4	20	0
1196	4	17	0
1197	4	11	3
1198	4	14	7
1199	1	14	2

1200 rows × 8 columns

```
y=pr['PerformanceRating']
y
```

```
0       3
1       3
2       4
3       3
4       3
       ..
1195    4
1196    3
1197    3
1198    3
1199    2
Name: PerformanceRating, Length: 1200, dtype: int64
```

```
Xtrain,Xtest,ytrain,ytest=train_test_split(X,y,test_size=0.2,random_state=42)
```

```
Xtrain.shape ,Xtest.shape
```

```
((960, 8), (240, 8))
```

```
ytrain.shape , ytest.shape
```

```
((960,), (240,))
```

Scaling

```
from sklearn.preprocessing import StandardScaler
se=StandardScaler()
```

```
from sklearn.preprocessing import StandardScaler
se=StandardScaler()
Xtrain=se.fit_transform(Xtrain)
Xtest=se.fit_transform(Xtest)
```

Training models

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import accuracy_score,classification_report
from sklearn.metrics import roc_auc_score,roc_curve


knn=KNeighborsClassifier(n_neighbors=3)
lr=LogisticRegression()
dt=DecisionTreeClassifier()
ra=RandomForestClassifier()
ad=AdaBoostClassifier()
svm=SVC(probability=True)
gau=GaussianNB()
bag=BaggingClassifier()
Gr=GradientBoostingClassifier()



Training_score= []
Testing_score= []
def model_building(model):
    model.fit(Xtrain, ytrain)
    ytrain_pred= model.predict(Xtrain)
    ytest_pred= model.predict(Xtest)
    a= accuracy_score(ytrain, ytrain_pred)
    b= accuracy_score(ytest, ytest_pred)
    Training_score.append(a)
    Testing_score.append(b)
    print(model)
    print("Train Data\n", accuracy_score(ytrain,ytrain_pred))
    print("Test Data\n", accuracy_score(ytest,ytest_pred))
```

```
model_building(knn)
```

```
KNeighborsClassifier(n_neighbors=3)
Train Data
 0.88125
Test Data
 0.8708333333333333
```

```
model_building(lr)
```

```
LogisticRegression()
Train Data
 0.8197916666666667
Test Data
 0.8291666666666667
```

```
model_building(dt)
```

```
DecisionTreeClassifier()
Train Data
 1.0
Test Data
 0.8833333333333333
```

```
model_building(ra)
```

```
RandomForestClassifier()
Train Data
 1.0
Test Data
 0.9375
```

```
model_building(ad)
```

```
AdaBoostClassifier()
Train Data
 0.8864583333333333
Test Data
 0.8833333333333333
```

```
model_building(svm)
```

```
SVC(probability=True)
Train Data
 0.896875
Test Data
 0.8791666666666667
```

```
model_building(gau)
```

```
GaussianNB()
Train Data
 0.8229166666666666
Test Data
 0.8083333333333333
```

```
model_building(bag)
```

```
BaggingClassifier()
Train Data
 0.9885416666666667
Test Data
 0.9166666666666666
```

```
model_building(Gr)
```

```
GradientBoostingClassifier()
Train Data
 0.971875
Test Data
 0.9291666666666667
```

```
Models= ["k-Nearest Neighbors","Logistic Regression" ,"Decision Tree Classifier", "Random forest Classifier" ,
        "Ada-Boosting Classifier","svm","GaussianNB","Bagging Classifier", "Gradiant- Bossting Classifier"]
```

```
new_df1 = pd.DataFrame({"Algorithms":Models,
                   "Training Score":Training_score,
                   "Testing Score":Testing_score,})
```

```
new_df1
```

	Algorithms	Training Score	Testing Score
0	k-Nearest Neighbors	0.881250	0.870833
1	Logistic Regression	0.819792	0.829167
2	Decision Tree Classifier	1.000000	0.883333
3	Random forest Classifier	1.000000	0.937500
4	Ada-Boosting Classifier	0.886458	0.883333
5	svm	0.896875	0.879167
6	GaussianNB	0.822917	0.808333
7	Bagging Classifier	0.988542	0.916667
8	Gradiant- Bossting Classifier	0.971875	0.929167

Hypertunning

Random forest

```
from sklearn.model_selection import RandomizedSearchCV
```

```
ra=RandomForestClassifier()
```

```
random_forest_params = {
    'n_estimators': [25,50,75,100],
    'max_depth': [2, 3, 5, 10, 20],
    'min_samples_leaf': [5, 10, 20, 50, 100],
    'min_samples_split': [2, 5, 10],
    'criterion': ["gini", "entropy"],
    'max_features': ['auto', 'sqrt'],
    'bootstrap': [True, False],
    'class_weight' : ["balanced", "balanced_subsample"]

}
```

```
ra_reg=RandomizedSearchCV(ra,param_distributions=random_forest_params,random_state=42,scoring='accuracy',cv=5,n_jobs=-1)
```

```
model_building(ra_reg)
```

```
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'class_weight': ['balanced',
                                                         'balanced_subsample'],
                                        'criterion': ['gini', 'entropy'],
                                        'max_depth': [2, 3, 5, 10, 20],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [5, 10, 20, 50,
                                                             100],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [25, 50, 75, 100]},
                   random_state=42, scoring='accuracy')
Train Data
 0.9052083333333333
Test Data
 0.9083333333333333
```

```
ra_reg.best_params_
```

```
{'n_estimators': 25,
 'min_samples_split': 2,
 'min_samples_leaf': 20,
 'max_features': 'sqrt',
 'max_depth': 10,
 'criterion': 'entropy',
 'class_weight': 'balanced',
 'bootstrap': False}
```

gaussianNB

```
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import GaussianNB
```

```
gau=GaussianNB()
```

```
param_grid = {
    'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5],
    'priors': [None, [0.2, 0.8], [0.5, 0.5], [0.8, 0.2]]
}
```

```
gau_reg=GridSearchCV(gau,param_grid=param_grid,cv=5,n_jobs=-1,scoring='accuracy')
```

```
gau_reg.fit(Xtrain,ytrain)
```

```
▸ GridSearchCV
▸ estimator: GaussianNB
  ▸ GaussianNB
```

```
model_building(gau_reg)
```

```
GridSearchCV(cv=5, estimator=GaussianNB(), n_jobs=-1,
             param_grid={'priors': [None, [0.2, 0.8], [0.5, 0.5], [0.8, 0.2]],
                         'var_smoothing': [1e-09, 1e-08, 1e-07, 1e-06, 1e-05]},
             scoring='accuracy')
Train Data
```

```
 0.8229166666666666
Test Data
 0.8083333333333333
```

SVM

```
svm=SVC(probability=True)
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'gamma': ['scale', 'auto', 0.1, 1],
    'degree': [2, 3, 4],
    'class_weight' : ["balanced", "balanced_subsample"]
}
```

```
svm_reg = RandomizedSearchCV(svm, param_distributions=param_grid, n_iter=10, cv=5,n_jobs=-1)
```

```
model_building(svm_reg)
```

```
RandomizedSearchCV(cv=5, estimator=SVC(probability=True), n_jobs=-1,
                   param_distributions={'C': [0.1, 1, 10],
                                        'class_weight': ['balanced',
                                                         'balanced_subsample'],
                                        'degree': [2, 3, 4],
                                        'gamma': ['scale', 'auto', 0.1, 1],
                                        'kernel': ['linear', 'poly', 'rbf',
                                                   'sigmoid']})
Train Data
 0.8416666666666667
Test Data
 0.8166666666666667
```

Gradient Boosting

```
param_grid_classification = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 5, 10],
    'subsample': [0.8, 0.9, 1.0],
    'max_features': ['auto', 'sqrt', 'log2', None],
    'loss': ['deviance', 'exponential']
}
```

```
gr_reg=RandomizedSearchCV(Gr,param_distributions=param_grid_classification,random_state=42,scoring='accuracy',cv=5,n_jobs=-1)
```