In [33]:
```
#Name: Himanshu Vijay Kapse
#Class: TE-A Roll no: COTA62
#AI_LAB_ASSIGNMENT_06
```

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:
```
data = pd.read_csv('Hr.csv')
```

In [3]:
```
data.shape
```

Out[3]:
```
(1200, 28)
```

In [4]:
```
data.columns
```

Out[4]:
```
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')
```

In [5]:
```
data.head()
```

Out[5]:

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRole	Bu
0	E1001000	32	Male	Marketing	Single	Sales	Sales Executive	
1	E1001006	47	Male	Marketing	Single	Sales	Sales Executive	
2	E1001007	40	Male	Life Sciences	Married	Sales	Sales Executive	
3	E1001009	41	Male	Human Resources	Divorced	Human Resources	Manager	
4	E1001010	60	Male	Marketing	Single	Sales	Sales Executive	

5 rows × 28 columns

In [6]:
```
# Looking for missing data
data.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
```

#Data Visualisation

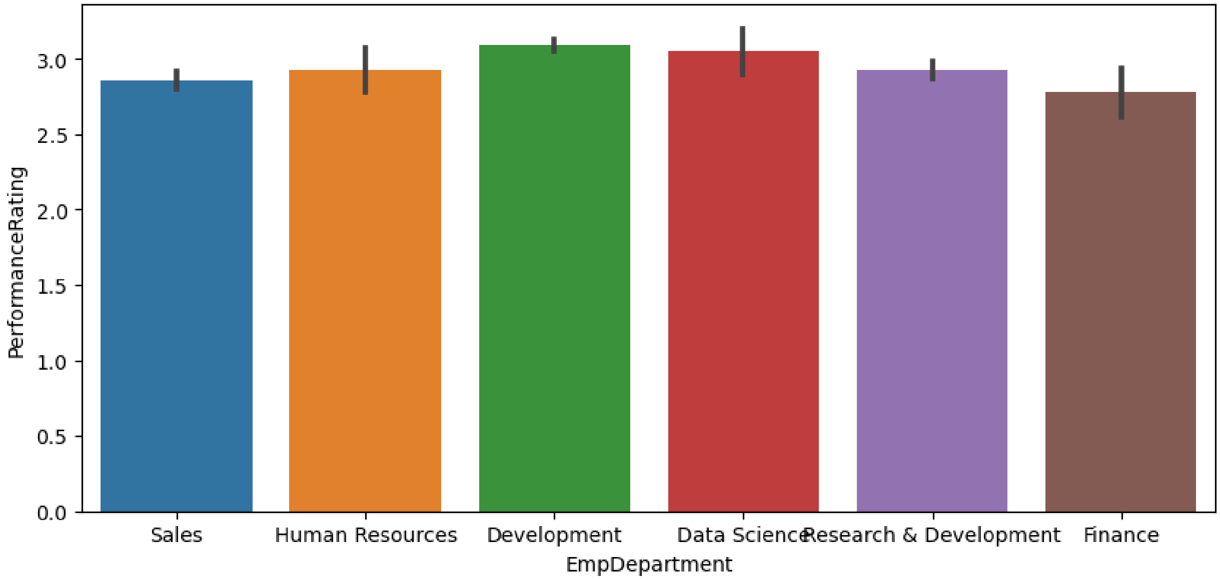
In [8]:
```
# A new pandas Dataframe is created to analyze department wise performance as asked.
dept = data.iloc[:,[5,27]].copy()
dept_per = dept.copy()
```

In [9]:
```
# Finding out the mean performance of all the departments and plotting its bar graph u
dept_per.groupby(by='EmpDepartment')['PerformanceRating'].mean()
```

Out[9]:
```
EmpDepartment
Data Science              3.050000
Development               3.085873
Finance                   2.775510
Human Resources           2.925926
Research & Development    2.921283
Sales                     2.860590
Name: PerformanceRating, dtype: float64
```

In [10]:
```
plt.figure(figsize=(10,4.5))
sns.barplot(dept_per['EmpDepartment'],dept_per['PerformanceRating'])
```

Out[10]:
```
<AxesSubplot:xlabel='EmpDepartment', ylabel='PerformanceRating'>
```



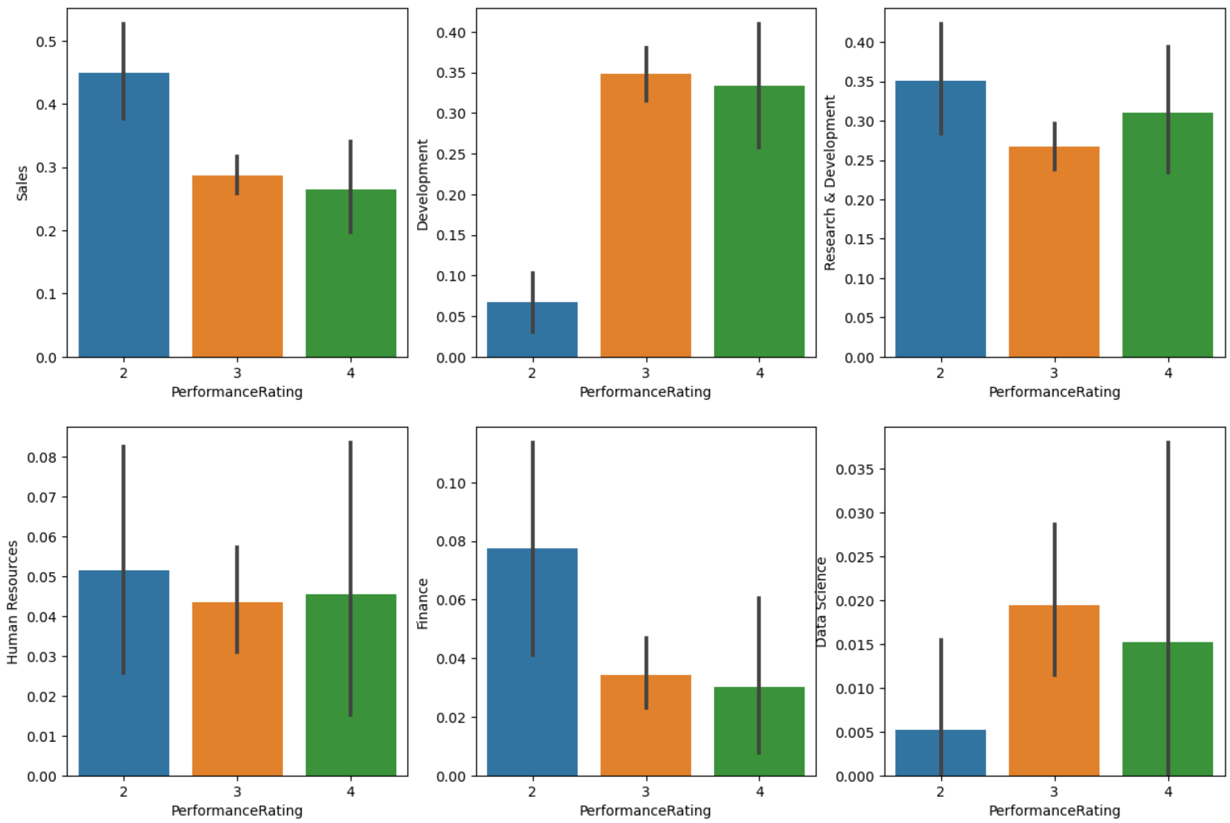
In [11]:
```
# Analyze each department separately
dept_per.groupby(by='EmpDepartment')['PerformanceRating'].value_counts()
```

Out[11]:
```
EmpDepartment           PerformanceRating
Data Science            3                     17
                        4                      2
                        2                      1
Development             3                    304
                        4                     44
                        2                     13
Finance                 3                     30
                        2                     15
                        4                      4
Human Resources         3                     38
                        2                     10
                        4                      6
Research & Development  3                    234
                        2                     68
                        4                     41
Sales                   3                    251
                        2                     87
                        4                     35
Name: PerformanceRating, dtype: int64
```

In [12]:
```
# Creating a new dataframe to analyze each department separately
department = pd.get_dummies(dept_per['EmpDepartment'])
performance = pd.DataFrame(dept_per['PerformanceRating'])
dept_rating = pd.concat([department,performance],axis=1)
```

In [14]:
```
# Plotting a separate bar graph for performance of each department using seaborn
plt.figure(figsize=(15,10))
plt.subplot(2,3,1)
sns.barplot(dept_rating['PerformanceRating'],dept_rating['Sales'])
plt.subplot(2,3,2)
sns.barplot(dept_rating['PerformanceRating'],dept_rating['Development'])
```

```
plt.subplot(2,3,3)
sns.barplot(dept_rating['PerformanceRating'],dept_rating['Research & Development'])
plt.subplot(2,3,4)
sns.barplot(dept_rating['PerformanceRating'],dept_rating['Human Resources'])
plt.subplot(2,3,5)
sns.barplot(dept_rating['PerformanceRating'],dept_rating['Finance'])
plt.subplot(2,3,6)
sns.barplot(dept_rating['PerformanceRating'],dept_rating['Data Science'])
plt.show()
```



Data Processing

In [15]:

```
# Encoding all the ordinal columns and creating a dummy variable for them to see if th
enc = LabelEncoder()
for i in (2,3,4,5,6,7,16,26):
 data.iloc[:,i] = enc.fit_transform(data.iloc[:,i])
data.head()
```

Out[15]:

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRole	Bu
0	E1001000	32	1	2	2	5	13	
1	E1001006	47	1	2	2	5	13	
2	E1001007	40	1	1	1	5	13	
3	E1001009	41	1	0	0	3	8	
4	E1001010	60	1	2	2	5	13	

5 rows × 28 columns

In [16]:

```
# Finding out the correlation coeffecient to find out which predictors are significant
data.corr()
```

Out[16]:

	Age	Gender	EducationBackground	MaritalStatus	EmpDepartr
Age	1.000000	-0.040107	-0.055905	-0.098368	-0.00
Gender	-0.040107	1.000000	0.009922	-0.042169	-0.01
EducationBackground	-0.055905	0.009922	1.000000	-0.001097	-0.02
MaritalStatus	-0.098368	-0.042169	-0.001097	1.000000	0.06
EmpDepartment	-0.000104	-0.010925	-0.026874	0.067272	1.00
EmpJobRole	-0.037665	0.011332	-0.012325	0.038023	0.56
BusinessTravelFrequency	0.040579	-0.043608	0.012382	0.028520	-0.04
DistanceFromHome	0.020937	-0.001507	-0.013919	-0.019148	0.00
EmpEducationLevel	0.207313	-0.022960	-0.047978	0.026737	0.01
EmpEnvironmentSatisfaction	0.013814	0.000033	0.045028	-0.032467	-0.01
EmpHourlyRate	0.062867	0.002218	-0.030234	-0.013540	0.00
EmpJobInvolvement	0.027216	0.010949	-0.025505	-0.043355	-0.07
EmpJobLevel	0.509139	-0.050685	-0.056338	-0.087359	0.10
EmpJobSatisfaction	-0.002436	0.024680	-0.030977	0.044593	0.00
NumCompaniesWorked	0.284408	-0.036675	-0.032879	-0.030095	-0.03
OverTime	0.051910	-0.038410	0.007046	-0.022833	-0.02
EmpLastSalaryHikePercent	-0.006105	-0.005319	-0.009788	0.010128	-0.01
EmpRelationshipSatisfaction	0.049749	0.030707	0.005652	0.026410	-0.05
TotalWorkExperienceInYears	0.680886	-0.061055	-0.027929	-0.093537	0.01
TrainingTimesLastYear	-0.016053	-0.057654	0.051596	0.026045	0.01
EmpWorkLifeBalance	-0.019563	0.015793	0.022890	0.014154	0.06
ExperienceYearsAtThisCompany	0.318852	-0.030392	-0.009887	-0.075728	0.04
ExperienceYearsInCurrentRole	0.217163	-0.031823	-0.003215	-0.076663	0.06
YearsSinceLastPromotion	0.228199	-0.021575	0.014277	-0.052951	0.05
YearsWithCurrManager	0.205098	-0.036643	0.002767	-0.061908	0.03
Attrition	-0.189317	0.035758	0.027161	0.162969	0.04
PerformanceRating	-0.040164	-0.001780	0.005607	0.024172	-0.16

27 rows × 27 columns

In [17]:

```
# Dropping the first columns as it is of no use for analysis.
data.drop(['EmpNumber'],inplace=True,axis=1)
```

In [18]:

```
data.head()
```

Out[18]:

	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRole	BusinessTravelFre
0	32	1	2	2	5	13	
1	47	1	2	2	5	13	
2	40	1	1	1	5	13	
3	41	1	0	0	3	8	
4	60	1	2	2	5	13	

5 rows × 27 columns

In [19]:

```
# Here we have selected only the important columns
y = data.PerformanceRating
#X = data.iloc[:,0:-1] All predictors were selected it resulted in dropping of accurac
X = data.iloc[:,[4,5,9,16,20,21,22,23,24]] # Taking only variables with correlation co
X.head()
```

Out[19]:

	EmpDepartment	EmpJobRole	EmpEnvironmentSatisfaction	EmpLastSalaryHikePercent	EmpWorkLif
0	5	13	4	12	
1	5	13	4	12	
2	5	13	4	21	
3	3	8	2	15	
4	5	13	1	14	

In [20]:

```
# Splitting into train and test for calculating the accuracy
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=10)
```

In [21]:

```
# Standardization technique is used
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [22]:

```
X_train.shape
```

Out[22]:

```
(840, 9)
```

In [23]:

```
X_test.shape
```

Out[23]:

```
(360, 9)
```

Model: Random Forest with GridSearchCV

In [28]:

```
# Training the model
from sklearn.ensemble import RandomForestClassifier
classifier_rfg=RandomForestClassifier(random_state=33,n_estimators=23)
parameters=[{'min_samples_split':[2,3,4,5],'criterion':['gini','entropy'],'min_samples
```

```
model_gridrf=GridSearchCV(estimator=classifier_rfg, param_grid=parameters, scoring='ac
model_gridrf.fit(X_train,y_train)
```

Out[28]:
```
GridSearchCV(estimator=RandomForestClassifier(n_estimators=23, random_state=33),
             param_grid=[{'criterion': ['gini', 'entropy'],
                          'min_samples_leaf': [1, 2, 3],
                          'min_samples_split': [2, 3, 4, 5]}],
             scoring='accuracy')
```

In [29]:
```
model_gridrf.best_params_
```

Out[29]:
```
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 4}
```

In [30]:
```
# Predicting the model
y_predict_rf = model_gridrf.predict(X_test)
```

In [31]:
```
# Finding accuracy, precision, recall and confusion matrix
print(accuracy_score(y_test,y_predict_rf))
print(classification_report(y_test,y_predict_rf))
```

```
0.9333333333333333
              precision    recall  f1-score   support

           2       0.90      0.89      0.90        63
           3       0.95      0.97      0.96       264
           4       0.83      0.76      0.79        33

    accuracy                           0.93       360
   macro avg       0.90      0.87      0.88       360
weighted avg       0.93      0.93      0.93       360
```

In [32]:
```
confusion_matrix(y_test,y_predict_rf)
```

Out[32]:
```
array([[ 56,   7,   0],
       [  4, 255,   5],
       [  2,   6,  25]], dtype=int64)
```

You can see the model has 93.05% Accuracy.

The features that are positively correlated are:

1. Environment Satisfaction
2. Last Salary Hike Percent
3. Worklife Balance. This means that if these factors increases, Performance Rating will increase. On the other hand, the features that are negatively correlated are:
4. Years Since Last Promotion
5. Experience Years at this Company
6. Experience years in Current Role
7. Years with Current Manager. This means that if these factors increases, Performance Rating will go down.

Conclusion: The company should provide a better environment as it increases the performance drastically. The company should increase the salary of the employee from time to time and help them maintain a worklife balance, shuffling the manager from time to time will also affect performance.

```
In [ ]:
```