In [16]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
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 [17]:
```
data = pd.read_csv("Hr.csv")
data.head(5)
```

Out[17]:

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRole	Business
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 [18]:
```
data.shape
```

Out[18]:
```
(1200, 28)
```

In [19]:
```
data.columns
```

Out[19]:
```
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 [20]:
```
data.info()
```

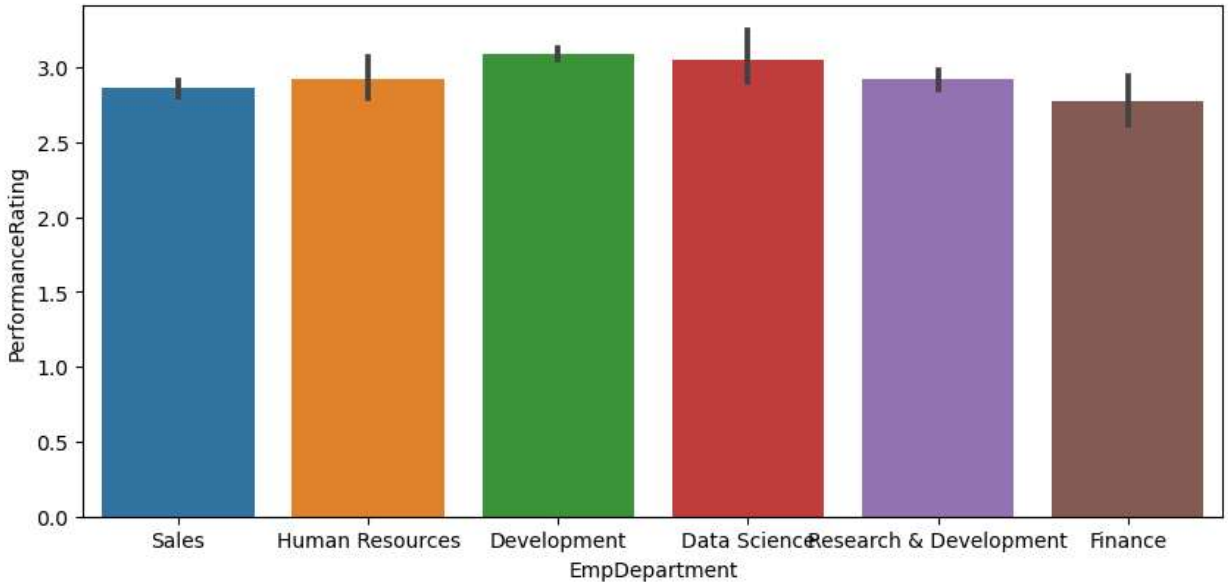
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 28 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   EmpNumber                     1200 non-null   object
 1   Age                           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
```

In [21]:
```
dept = data.iloc[:, [5,27]].copy()
dept_per = dept
dept_per.groupby(by='EmpDepartment')['PerformanceRating'].mean()
```

Out[21]:
```
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 [23]:
```
plt.figure(figsize=(10,4.5))
sns.barplot(x = dept_per['EmpDepartment'],y = dept_per['PerformanceRating'])
```

Out[23]:
```
<Axes: xlabel='EmpDepartment', ylabel='PerformanceRating'>
```

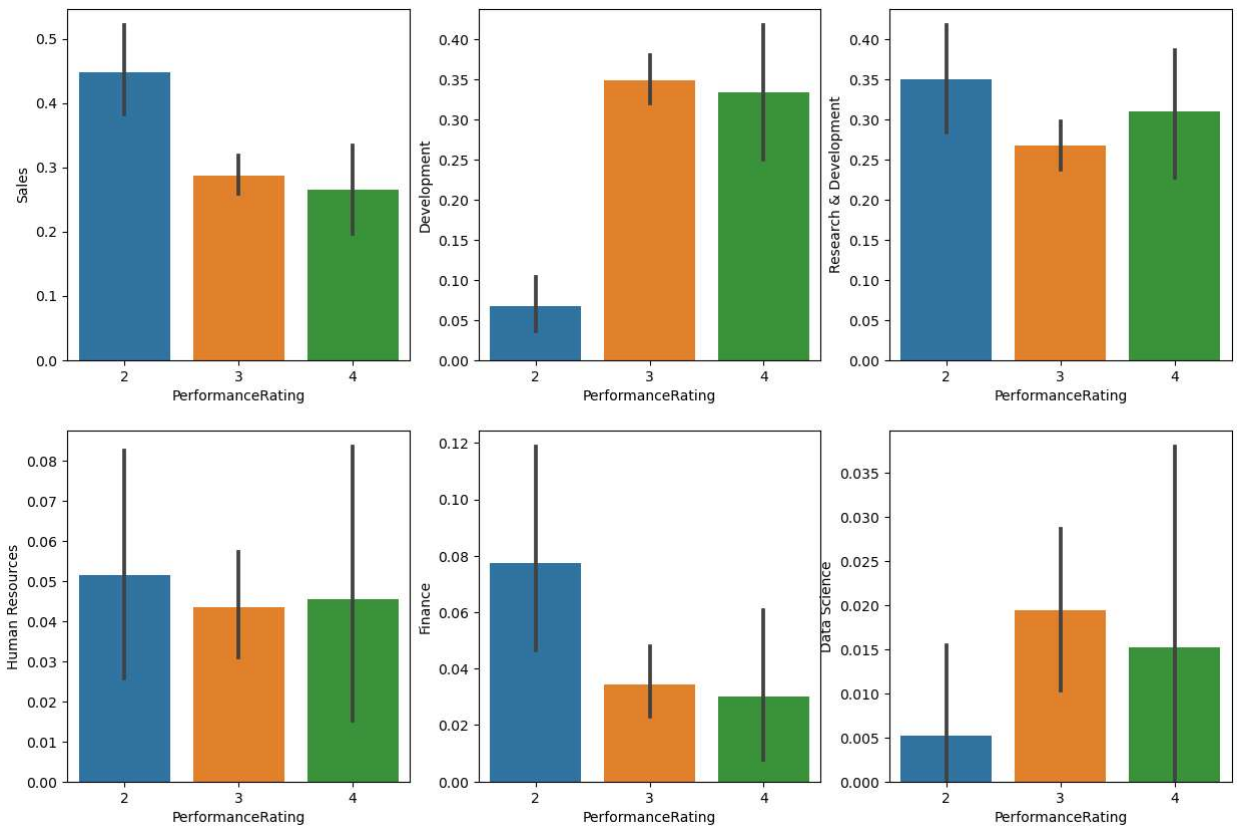


In [24]:
```
dept_per.groupby(by='EmpDepartment')['PerformanceRating'].value_counts()
```

Out[24]:
```
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 [25]:
```
department = pd.get_dummies(dept_per['EmpDepartment'])
performance = pd.DataFrame(dept_per['PerformanceRating'])
dept_rating = pd.concat([department, performance], axis = 1)
```

In [28]:
```
plt.figure(figsize=(15,10))
plt.subplot(2,3,1)
sns.barplot(x= dept_rating['PerformanceRating'],y =dept_rating['Sales'])
plt.subplot(2,3,2)
sns.barplot(x= dept_rating['PerformanceRating'],y=dept_rating['Development'])
plt.subplot(2,3,3)
sns.barplot(x = dept_rating['PerformanceRating'],y=dept_rating['Research & Development
plt.subplot(2,3,4)
sns.barplot(x= dept_rating['PerformanceRating'],y=dept_rating['Human Resources'])
plt.subplot(2,3,5)
sns.barplot(x=dept_rating['PerformanceRating'],y=dept_rating['Finance'])
plt.subplot(2,3,6)
sns.barplot(x=dept_rating['PerformanceRating'],y=dept_rating['Data Science'])
plt.show()
```



In [29]:
```
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[29]:

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRole	Business
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 [30]:
```
data.corr()
```

Out[30]:

	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment
Age	1.000000	-0.040107	-0.055905	-0.098368	-0.000104
Gender	-0.040107	1.000000	0.009922	-0.042169	-0.010925
EducationBackground	-0.055905	0.009922	1.000000	-0.001097	-0.026874
MaritalStatus	-0.098368	-0.042169	-0.001097	1.000000	0.067272
EmpDepartment	-0.000104	-0.010925	-0.026874	0.067272	1.000000
EmpJobRole	-0.037665	0.011332	-0.012325	0.038023	0.568973
BusinessTravelFrequency	0.040579	-0.043608	0.012382	0.028520	-0.045233
DistanceFromHome	0.020937	-0.001507	-0.013919	-0.019148	0.007707
EmpEducationLevel	0.207313	-0.022960	-0.047978	0.026737	0.019175
EmpEnvironmentSatisfaction	0.013814	0.000033	0.045028	-0.032467	-0.019237
EmpHourlyRate	0.062867	0.002218	-0.030234	-0.013540	0.003957
EmpJobInvolvement	0.027216	0.010949	-0.025505	-0.043355	-0.076988
EmpJobLevel	0.509139	-0.050685	-0.056338	-0.087359	0.100526
EmpJobSatisfaction	-0.002436	0.024680	-0.030977	0.044593	0.007150
NumCompaniesWorked	0.284408	-0.036675	-0.032879	-0.030095	-0.033950
OverTime	0.051910	-0.038410	0.007046	-0.022833	-0.026841
EmpLastSalaryHikePercent	-0.006105	-0.005319	-0.009788	0.010128	-0.012661
EmpRelationshipSatisfaction	0.049749	0.030707	0.005652	0.026410	-0.050286
TotalWorkExperienceInYears	0.680886	-0.061055	-0.027929	-0.093537	0.016065
TrainingTimesLastYear	-0.016053	-0.057654	0.051596	0.026045	0.016438
EmpWorkLifeBalance	-0.019563	0.015793	0.022890	0.014154	0.068875
ExperienceYearsAtThisCompany	0.318852	-0.030392	-0.009887	-0.075728	0.047677
ExperienceYearsInCurrentRole	0.217163	-0.031823	-0.003215	-0.076663	0.069602
YearsSinceLastPromotion	0.228199	-0.021575	0.014277	-0.052951	0.052315
YearsWithCurrManager	0.205098	-0.036643	0.002767	-0.061908	0.033850
Attrition	-0.189317	0.035758	0.027161	0.162969	0.048006
PerformanceRating	-0.040164	-0.001780	0.005607	0.024172	-0.162615

27 rows × 27 columns

In [31]:
```
data.drop(['EmpNumber'],inplace=True,axis=1)
data.head(5)
```

Out[31]:

	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRole	BusinessTravelFrequen
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 [32]:
```
y = data.PerformanceRating
#X = data.iloc[:,0:-1] All predictors were selected it resulted in dropping of accuracy
X = data.iloc[:,[4,5,9,16,20,21,22,23,24]] # Taking only variables with correlation co
X.head()
```

Out[32]:

	EmpDepartment	EmpJobRole	EmpEnvironmentSatisfaction	EmpLastSalaryHikePercent	EmpWorkLifeBala
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 [33]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=10)
# Standardization technique is used
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [34]:
```
X_train.shape
```

Out[34]: (840, 9)

In [35]:
```
X_test.shape
```

Out[35]: (360, 9)

In [36]:
```
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[36]:
```
GridSearchCV
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')
estimator: RandomForestClassifier
RandomForestClassifier
```

In [37]:
```
model_gridrf.best_params_
```

Out[37]:
```
{'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 4}
```

In [38]:
```
# Predicting the model
y_predict_rf = model_gridrf.predict(X_test)
```

In [39]:
```
# 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 [40]:
```
confusion_matrix(y_test,y_predict_rf)
```

Out[40]:
```
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:

Environment Satisfaction

Last Salary Hike Percent

Worklife Balance. (This means that if these factors increases, Performance Rating will increase.)

On the other hand, the features that are negatively correlated are:

Years Since Last Promotion

Experience Years at this Company

Experience years in Current Role

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 [ ]: