Title :- Expert System - Employee performance evaluation

In [1]:
```
# Importing the necessary libraries
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
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Importing Raw data

In [3]:
```
# Importing the csv file
data = pd.read_csv('INX_Future_Inc_Employee_Performance_CDS_Project2_Data_V1.8.csv')
```

Exploratory Data Analysis

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

```
(1200, 28)
```

In [5]:
```
data.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')
```

In [6]:
```
data.head()
```

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

5 rows × 28 columns

In [7]:
```
# Looking for missing data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1200 entries, 0 to 1199
Data columns (total 28 columns):
EmpNumber                      1200 non-null object
Age                            1200 non-null int64
Gender                         1200 non-null object
EducationBackground            1200 non-null object
MaritalStatus                  1200 non-null object
EmpDepartment                  1200 non-null object
EmpJobRole                     1200 non-null object
BusinessTravelFrequency        1200 non-null object
DistanceFromHome               1200 non-null int64
EmpEducationLevel              1200 non-null int64
EmpEnvironmentSatisfaction     1200 non-null int64
EmpHourlyRate                  1200 non-null int64
EmpJobInvolvement              1200 non-null int64
EmpJobLevel                    1200 non-null int64
EmpJobSatisfaction             1200 non-null int64
NumCompaniesWorked             1200 non-null int64
OverTime                       1200 non-null object
EmpLastSalaryHikePercent       1200 non-null int64
EmpRelationshipSatisfaction    1200 non-null int64
TotalWorkExperienceInYears     1200 non-null int64
TrainingTimesLastYear          1200 non-null int64
EmpWorkLifeBalance             1200 non-null int64
ExperienceYearsAtThisCompany   1200 non-null int64
ExperienceYearsInCurrentRole   1200 non-null int64
YearsSinceLastPromotion        1200 non-null int64
YearsWithCurrManager           1200 non-null int64
Attrition                      1200 non-null object
PerformanceRating              1200 non-null int64
dtypes: int64(19), object(9)
memory usage: 262.6+ KB
```

Analysis of Department wise Perfomance

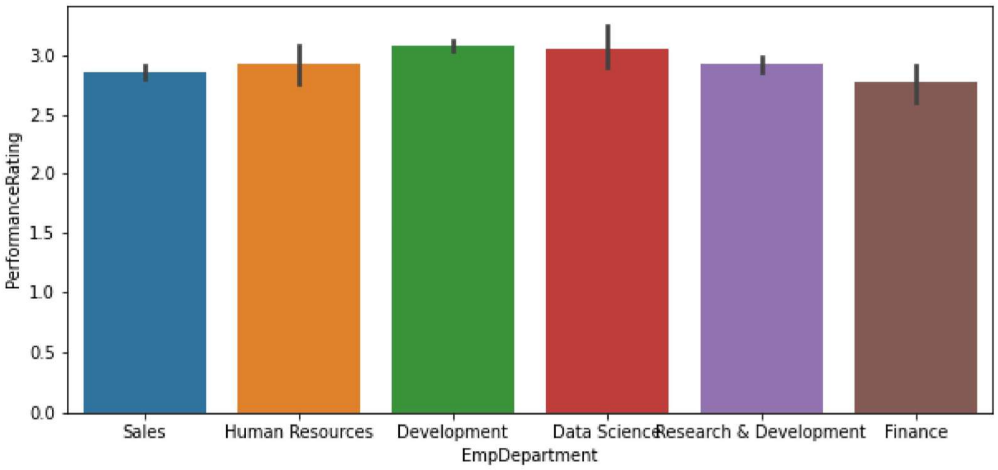
In [8]:
```
# A new pandas Dataframe is created to analyze department wise performance as asked.
dept = data.iloc[:,[5,27]].copy()
dept_per = dept.copy()
```

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

```
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'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x17e0b405fc8>
```



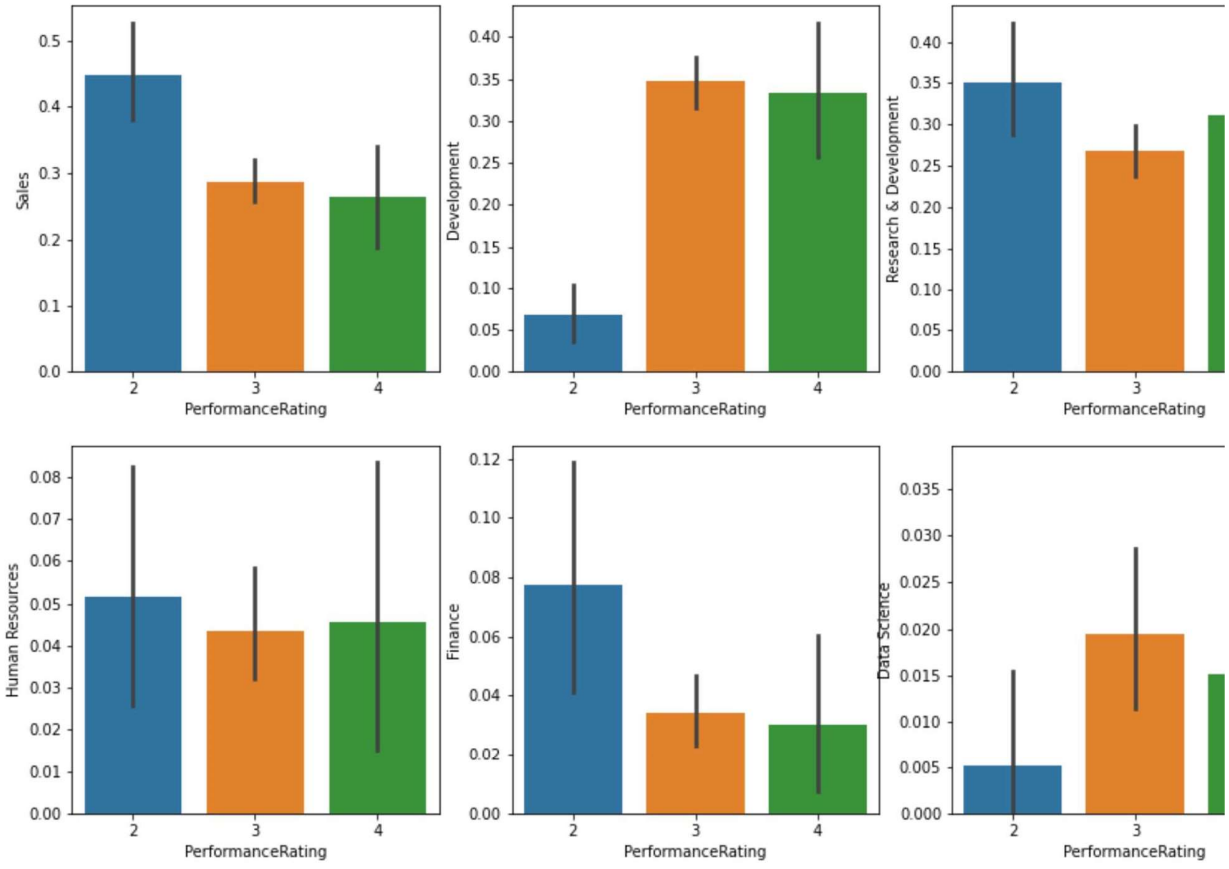
In [11]:
```
# Analyze each department separately
dept_per.groupby(by='EmpDepartment')['PerformanceRating'].value_counts()
```

```
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 [13]:

```
# 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/ Data Munging

In [14]:

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

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

5 rows × 28 columns

Feature Selection

In [15]:
```
# Finding out the correlation coeffecient to find out which predictors are significant.
data.corr()
```

	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	Emp
Age	1.000000	-0.040107	-0.055905	-0.098368	-0.000104	-0.037
Gender	-0.040107	1.000000	0.009922	-0.042169	-0.010925	0.011
EducationBackground	-0.055905	0.009922	1.000000	-0.001097	-0.026874	-0.012
MaritalStatus	-0.098368	-0.042169	-0.001097	1.000000	0.067272	0.038
EmpDepartment	-0.000104	-0.010925	-0.026874	0.067272	1.000000	0.568
EmpJobRole	-0.037665	0.011332	-0.012325	0.038023	0.568973	1.000
BusinessTravelFrequency	0.040579	-0.043608	0.012382	0.028520	-0.045233	-0.086
DistanceFromHome	0.020937	-0.001507	-0.013919	-0.019148	0.007707	0.022
EmpEducationLevel	0.207313	-0.022960	-0.047978	0.026737	0.019175	-0.016
EmpEnvironmentSatisfaction	0.013814	0.000033	0.045028	-0.032467	-0.019237	0.044
EmpHourlyRate	0.062867	0.002218	-0.030234	-0.013540	0.003957	-0.016
EmpJobInvolvement	0.027216	0.010949	-0.025505	-0.043355	-0.076988	-0.008
EmpJobLevel	0.509139	-0.050685	-0.056338	-0.087359	0.100526	0.004
EmpJobSatisfaction	-0.002436	0.024680	-0.030977	0.044593	0.007150	0.032
NumCompaniesWorked	0.284408	-0.036675	-0.032879	-0.030095	-0.033950	-0.009
OverTime	0.051910	-0.038410	0.007046	-0.022833	-0.026841	0.015
EmpLastSalaryHikePercent	-0.006105	-0.005319	-0.009788	0.010128	-0.012661	0.005
EmpRelationshipSatisfaction	0.049749	0.030707	0.005652	0.026410	-0.050286	-0.043
TotalWorkExperienceInYears	0.680886	-0.061055	-0.027929	-0.093537	0.016065	-0.049
TrainingTimesLastYear	-0.016053	-0.057654	0.051596	0.026045	0.016438	0.004
EmpWorkLifeBalance	-0.019563	0.015793	0.022890	0.014154	0.068875	-0.007
ExperienceYearsAtThisCompany	0.318852	-0.030392	-0.009887	-0.075728	0.047677	-0.009
ExperienceYearsInCurrentRole	0.217163	-0.031823	-0.003215	-0.076663	0.069602	0.019
YearsSinceLastPromotion	0.228199	-0.021575	0.014277	-0.052951	0.052315	0.012
YearsWithCurrManager	0.205098	-0.036643	0.002767	-0.061908	0.033850	-0.004
Attrition	-0.189317	0.035758	0.027161	0.162969	0.048006	0.037
PerformanceRating	-0.040164	-0.001780	0.005607	0.024172	-0.162615	-0.096

27 rows × 27 columns

In [16]:
```
# Dropping the first columns as it is of no use for analysis.
data.drop(['EmpNumber'],inplace=True,axis=1)
```

In [17]:
```
data.head()
```

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

5 rows × 27 columns

In [18]:
```
# 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 accuracy.
X = data.iloc[:,[4,5,9,16,20,21,22,23,24]] # Taking only variables with correlation coeffeci
X.head()
```

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

In [19]:
```
# Splitting into train and test for calculating the accuracy
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=42)
```

In [20]:
```
# Standardization technique is used
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [21]:
```
X_train.shape
```

```
(900, 9)
```

In [22]:
```
X_test.shape
```

```
(300, 9)
```

Model

We have used Support Vector Machine to calculate the accuracy and found out that gives an accura

Support Vector Machine

In [23]:
```
# Training the model
from sklearn.svm import SVC
rbf_svc = SVC(kernel='rbf', C=100, random_state=42).fit(X_train,y_train)
```

In [24]:
```
# Predicting the model
y_predict_svm = rbf_svc.predict(X_test)
```

In [25]:
```
# Finding accuracy, precision, recall and confusion matrix
print(accuracy_score(y_test,y_predict_svm))
print(classification_report(y_test,y_predict_svm))
```

```
0.85
              precision    recall  f1-score   support

           2       0.68      0.76      0.72        37
           3       0.92      0.89      0.90       232
           4       0.60      0.68      0.64        31

    accuracy                           0.85       300
   macro avg       0.73      0.77      0.75       300
weighted avg       0.86      0.85      0.85       300
```

In [26]:
```
confusion_matrix(y_test,y_predict_svm)
```

```
array([[ 28,   9,   0],
       [ 12, 206,  14],
       [  1,   9,  21]], dtype=int64)
```