

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

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

```
Out[4]: (1200, 28)
```

```
In [5]: data.columns
```

```
Out[5]: Index([u'EmpNumber', u'Age', u'Gender', u'EducationBackground',
              u'MaritalStatus', u'EmpDepartment', u'EmpJobRole',
              u'BusinessTravelFrequency', u'DistanceFromHome', u'EmpEducationLevel',
              u'EmpEnvironmentSatisfaction', u'EmpHourlyRate', u'EmpJobInvolvement',
              u'EmpJobLevel', u'EmpJobSatisfaction', u'NumCompaniesWorked',
              u'Overtime', u'EmpLastSalaryHikePercent',
              u'EmpRelationshipSatisfaction', u'TotalWorkExperienceInYears',
              u'TrainingTimesLastYear', u'EmpWorkLifeBalance',
              u'ExperienceYearsAtThisCompany', u'ExperienceYearsInCurrentRole',
              u'YearsSinceLastPromotion', u'YearsWithCurrManager', u'Attrition',
              u'PerformanceRating'],
              dtype='object')
```

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

Out[6]:

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

5 rows × 28 columns

In [7]: `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
```

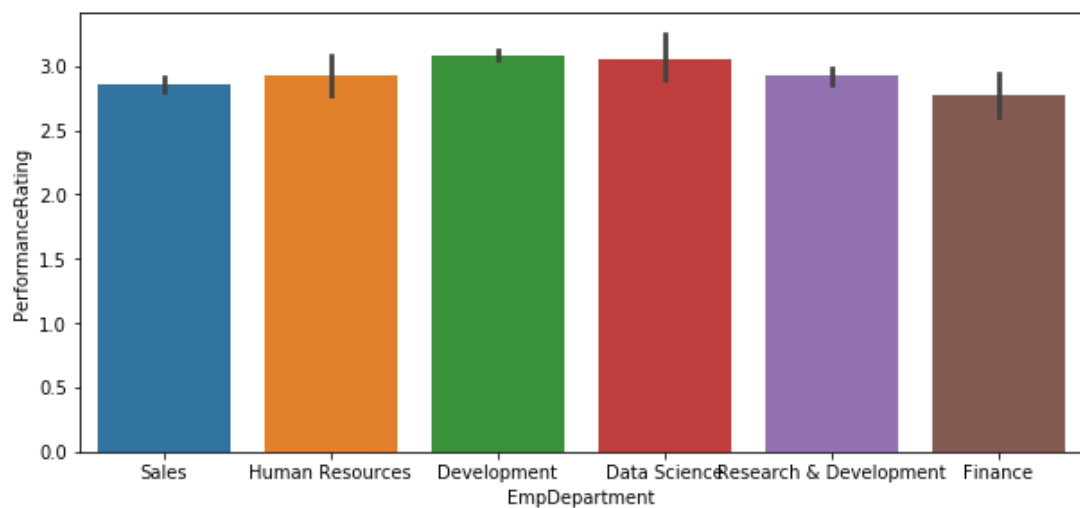
```
In [8]: dept = data.iloc[:,[5,27]].copy()  
dept_per = dept.copy()
```

```
In [9]: 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]: <matplotlib.axes._subplots.AxesSubplot at 0x7f078c105a50>
```

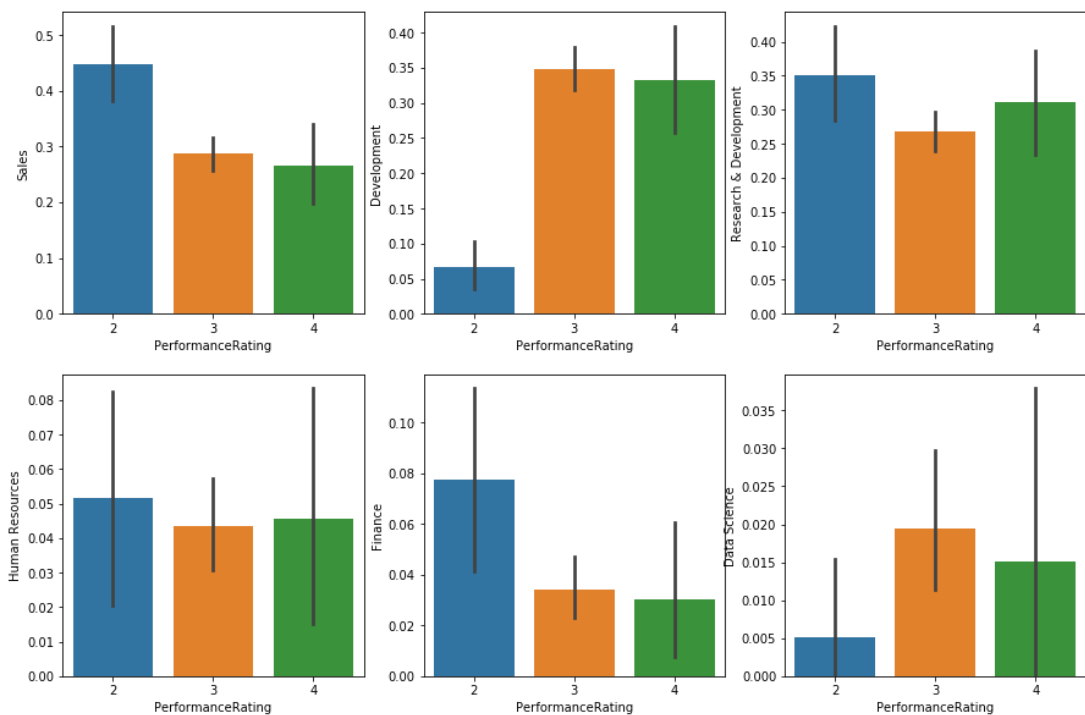


```
In [11]: dept_per.groupby(by='EmpDepartment')['PerformanceRating'].value_counts()
```

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

```
In [14]: 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['Develo
pment'])
plt.subplot(2,3,3)
sns.barplot(dept_rating['PerformanceRating'],dept_rating['Resear
ch & 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['Financ
e'])
plt.subplot(2,3,6)
sns.barplot(dept_rating['PerformanceRating'],dept_rating['Data S
cience'])
plt.show()
```



```
In [16]: #Data Processing
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[16]:

	EmpNumber	Age	Gender	EducationBackground	MaritalStatus	EmpDepartme
0	E1001000	32	1	2	2	5
1	E1001006	47	1	2	2	5
2	E1001007	40	1	1	1	5
3	E1001009	41	1	0	0	3
4	E1001010	60	1	2	2	5

5 rows × 28 columns

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

Out[17]:

	Age	Gender	EducationBackground	Ma
Age	1.000000	-0.040107	-0.055905	-0.0
Gender	-0.040107	1.000000	0.009922	-0.0
EducationBackground	-0.055905	0.009922	1.000000	-0.0
MaritalStatus	-0.098368	-0.042169	-0.001097	1.0
EmpDepartment	-0.000104	-0.010925	-0.026874	0.0
EmpJobRole	-0.037665	0.011332	-0.012325	0.0
BusinessTravelFrequency	0.040579	-0.043608	0.012382	0.0
DistanceFromHome	0.020937	-0.001507	-0.013919	-0.0
EmpEducationLevel	0.207313	-0.022960	-0.047978	0.0
EmpEnvironmentSatisfaction	0.013814	0.000033	0.045028	-0.0
EmpHourlyRate	0.062867	0.002218	-0.030234	-0.0
EmpJobInvolvement	0.027216	0.010949	-0.025505	-0.0
EmpJobLevel	0.509139	-0.050685	-0.056338	-0.0
EmpJobSatisfaction	-0.002436	0.024680	-0.030977	0.0
NumCompaniesWorked	0.284408	-0.036675	-0.032879	-0.0
OverTime	0.051910	-0.038410	0.007046	-0.0
EmpLastSalaryHikePercent	-0.006105	-0.005319	-0.009788	0.0
EmpRelationshipSatisfaction	0.049749	0.030707	0.005652	0.0
TotalWorkExperienceInYears	0.680886	-0.061055	-0.027929	-0.0
TrainingTimesLastYear	-0.016053	-0.057654	0.051596	0.0
EmpWorkLifeBalance	-0.019563	0.015793	0.022890	0.0
ExperienceYearsAtThisCompany	0.318852	-0.030392	-0.009887	-0.0
ExperienceYearsInCurrentRole	0.217163	-0.031823	-0.003215	-0.0
YearsSinceLastPromotion	0.228199	-0.021575	0.014277	-0.0
YearsWithCurrManager	0.205098	-0.036643	0.002767	-0.0
Attrition	-0.189317	0.035758	0.027161	0.1
PerformanceRating	-0.040164	-0.001780	0.005607	0.0

27 rows × 27 columns

In [18]: `data.drop(['EmpNumber'], inplace=True, axis=1)`

In [19]: `data.head()`

Out[19]:

	Age	Gender	EducationBackground	MaritalStatus	EmpDepartment	EmpJobRole
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 × 7 columns

In [40]: `y=data.PerformanceRating  
X=data.iloc[:,[4,5,9,16,20,21,22,23,24]]  
X.head()`

Out[40]:

	EmpDepartment	EmpJobRole	EmpEnvironmentSatisfaction	EmpLastSalaryHil
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 [41]: `X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=10)`

In [42]: `sc=StandardScaler()  
X_train=sc.fit_transform(X_train)  
X_test=sc.transform(X_test)`

In [43]: `X_train.shape`

Out[43]: (840, 9)

In [44]: `X_test.shape`

Out[44]: (360, 9)



```
In [46]: 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_leaf':[1,2,3], 'min_samples_split':[2,3,4,5]}]
model_gridrf=GridSearchCV(estimator=classifier_rfg, param_grid=parameters, scoring='accuracy')
model_gridrf.fit(X_train,y_train)
```

```
Out[46]: GridSearchCV(cv=None, error_score='raise',
                    estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=23, n_jobs=1,
                    oob_score=False, random_state=33, verbose=0, warm_start=False),
                    fit_params=None, iid=True, n_jobs=1,
                    param_grid=[{'min_samples_split': [2, 3, 4, 5], 'criterion': ['gini', 'entropy'], 'min_samples_leaf': [1, 2, 3]}],
                    pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                    scoring='accuracy', verbose=0)
```

```
In [47]: model_gridrf.best_params_
```

```
Out[47]: {'criterion': 'entropy', 'min_samples_leaf': 2, 'min_samples_split': 2}
```

```
In [48]: y_predict_rf = model_gridrf.predict(X_test)
```

```
In [49]: print(accuracy_score(y_test,y_predict_rf))
print(classification_report(y_test,y_predict_rf))
```

```
0.9305555555555556
```

	precision	recall	f1-score	support
2	0.92	0.89	0.90	63
3	0.94	0.97	0.96	264
4	0.83	0.73	0.77	33
avg / total	0.93	0.93	0.93	360

```
In [50]: confusion_matrix(y_test,y_predict_rf)
```

```
Out[50]: array([[ 56,   7,   0],
                [  4, 255,   5],
                [  1,   8,  24]])
```