You are predicting the attrition rate with HR Analytics. Attrition rate prediction means predicting which employees are likely to leave the company in near future. Prediction of the exits will help the company prepare for their replacements within the available time frame.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
df=pd.read_csv('dataset.csv')
df.head()
```

Out[4]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	F
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns

```
In [3]: df.columns

Out[3]: Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
               'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
               'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
               'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
               'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
               'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
               'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
               'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
               'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
               'YearsWithCurrManager'],
              dtype='object')

In [5]: df=df.drop(['MonthlyRate','EmployeeCount','EmployeeNumber','Gender','StandardHours','DailyRate','HourlyRate','Over18'],

In [6]: df.columns

Out[6]: Index(['Age', 'Attrition', 'BusinessTravel', 'Department', 'DistanceFromHome',
               'Education', 'EducationField', 'EnvironmentSatisfaction',
               'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
               'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked', 'OverTime',
               'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction',
               'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
               'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
               'YearsSinceLastPromotion', 'YearsWithCurrManager'],
              dtype='object')

In [7]: df.shape

Out[7]: (1470, 27)

In [7]: ###so,we have information of 1470 employees.
```

In [8]:
```python
for j in df.columns:
    print(j,':',df[j].value_counts())

    print('-'*45)
    print('-'*45)
```

```
12     198
15     101
18      89
17      82
16      78
19      76
22      56
20      55
21      48
23      28
24      21
25      18
Name: PercentSalaryHike, dtype: int64
---------------------------------------------
---------------------------------------------
PerformanceRating : 3    1244
4     226
Name: PerformanceRating, dtype: int64
---------------------------------------------
```

In [9]:
```python
df.columns
```

Out[9]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'Department', 'DistanceFromHome',
       'Education', 'EducationField', 'EnvironmentSatisfaction',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked', 'OverTime',
       'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager'],
      dtype='object')
```

In [10]:
```
df[['RelationshipSatisfaction','JobSatisfaction','WorkLifeBalance','EnvironmentSatisfaction','JobInvolvement']]
```

Out[10]:

	RelationshipSatisfaction	JobSatisfaction	WorkLifeBalance	EnvironmentSatisfaction	JobInvolvement
0	1	4	1	2	3
1	4	2	3	3	2
2	2	3	3	4	2
3	3	3	3	4	3
4	4	2	3	1	3
...	...	...	...	...	...
1465	3	4	3	3	4
1466	1	1	3	4	2
1467	2	2	3	2	4
1468	4	2	2	4	2
1469	1	3	4	2	4

1470 rows × 5 columns

In [15]:
```
df['avg_satifaction']=(df['RelationshipSatisfaction']+df['JobSatisfaction']+df['WorkLifeBalance']+df['EnvironmentSatis
```

In [16]:
```
df['avg_satifaction']
```

Out[16]:
```
0       2.2
1       2.8
2       2.8
3       3.2
4       2.6
       ...
1465    3.4
1466    2.2
1467    2.6
1468    2.8
1469    2.8
Name: avg_satifaction, Length: 1470, dtype: float64
```

In [17]:
```
def satisfaction(df):
    if df['avg_satifaction']>2.5:
        return 0
    else:
        return 1
```

In [18]:
```
df['satif']=df.apply(lambda df:satisfaction(df),axis=1)
```

In [19]:
```
df['satif']
```

Out[19]:
```
0       1
1       0
2       0
3       0
4       0
       ..
1465    0
1466    1
1467    0
1468    0
1469    0
Name: satif, Length: 1470, dtype: int64
```

In [20]:
```
df['DistanceFromHome'].mean()
```

Out[20]:
```
9.19251700680272
```

In [21]:
```
df['TrainingTimesLastYear'].mean()
```

Out[21]:
```
2.7993197278911564
```

In [22]:
```
### Two important variables-'training times last year' and 'distancefrom home':

def dist_train(df):
    if df['DistanceFromHome']>9.19 and df['TrainingTimesLastYear']<2.8:
        return 1
    else:
        return 0
```

In [23]:
```
df['dist_train']=df.apply(lambda df:dist_train(df),axis=1)
```

In [24]:
```
df.columns
```

Out[24]:
```
Index(['Age', 'Attrition', 'BusinessTravel', 'Department', 'DistanceFromHome',
       'Education', 'EducationField', 'EnvironmentSatisfaction',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'NumCompaniesWorked', 'OverTime',
       'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager', 'avg_satifaction',
       'satif', 'dist_train'],
      dtype='object')
```

In [25]:
```
df=df.drop(['RelationshipSatisfaction','WorkLifeBalance','JobSatisfaction','JobInvolvement','EnvironmentSatisfaction',
```

In [26]:
```
df.shape
```

Out[26]:
```
(1470, 22)
```

In [ ]:
```
###we managed to do a feature selection-feature transformation-feature elimination and reduced the feature count by 13
```