In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]: data = pd.read_csv("D:\Unified_Internship\Project 3 - Employee Attrition\Attrition data.csv")

In [4]: data.head()

Out[4]:

	EmployeeID	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Gender	...	TotalWorkingYears	TrainingTimesLastYear	YearsAtCompany	Y
0	1	51	No	Travel_Rarely	Sales	6	2	Life Sciences	1	Female	...	1.0	6	1	
1	2	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	1	Female	...	6.0	3	5	
2	3	32	No	Travel_Frequently	Research & Development	17	4	Other	1	Male	...	5.0	2	5	
3	4	38	No	Non-Travel	Research & Development	2	5	Life Sciences	1	Male	...	13.0	5	8	
4	5	32	No	Travel_Rarely	Research & Development	10	1	Medical	1	Male	...	9.0	2	6	

5 rows × 29 columns

In [5]: data.shape

Out[5]: (4410, 29)

In [6]: data.Attrition.value_counts()

Out[6]:
```
No    3699
Yes    711
Name: Attrition, dtype: int64
```

In [7]: data.describe()

Out[7]:

	EmployeeID	Age	DistanceFromHome	Education	EmployeeCount	JobLevel	MonthlyIncome	NumCompaniesWorked	PercentSalaryHike	StandardHours	...	TotalWorkingYears	Train
count	4410.000000	4410.000000	4410.000000	4410.000000	4410.0	4410.000000	4410.000000	4391.000000	4410.000000	4410.0	...	4401.000000	
mean	2205.500000	36.923810	9.192517	2.912925	1.0	2.063946	65029.312925	2.694836	15.209524	80.0	...	11.279936	
std	1273.201673	9.133301	8.105026	1.023450	0.0	1.106689	47568.888559	2.498887	3.659160	0.0	...	7.782222	
min	1.000000	18.000000	1.000000	1.000000	1.0	1.000000	10090.000000	0.000000	11.000000	80.0	...	0.000000	
25%	1103.250000	30.000000	2.000000	2.000000	1.0	1.000000	29110.000000	1.000000	12.000000	80.0	...	6.000000	
50%	2205.500000	36.000000	7.000000	3.000000	1.0	2.000000	49190.000000	2.000000	14.000000	80.0	...	10.000000	
75%	3307.750000	43.000000	14.000000	4.000000	1.0	3.000000	83600.000000	4.000000	18.000000	80.0	...	15.000000	
max	4410.000000	60.000000	29.000000	5.000000	1.0	5.000000	199990.000000	9.000000	25.000000	80.0	...	40.000000	

5 rows × 21 columns

In [8]: data.isnull().sum()

Out[8]:
```
EmployeeID                  0
Age                         0
Attrition                   0
BusinessTravel              0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
Gender                      0
JobLevel                    0
JobRole                     0
MaritalStatus               0
MonthlyIncome               0
NumCompaniesWorked         19
Over18                      0
PercentSalaryHike           0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           9
TrainingTimesLastYear       0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
EnvironmentSatisfaction    25
JobSatisfaction            20
WorkLifeBalance            38
JobInvolvement              0
PerformanceRating           0
dtype: int64
```

In [9]: attrition_count = pd.DataFrame(df['Attrition'].value_counts())

In [10]: attrition_count

Out[10]:

	Attrition
No	3825
Yes	585

In [11]: plt.pie(attrition_count['Attrition'], labels = ['No' , 'Yes'], explode = (0,2,0))

Out[11]:
```
([<matplotlib.patches.Wedge at 0x20b41a5d30>,
  <matplotlib.patches.Wedge at 0x20b41a2f3fc>],
 [Text(-1.1200000341997374, 0.032967077135401, 'No'),
  Text(0.9611145182968274, -0.6348117791385214, 'Yes')])
```

In [12]: df.drop(['EmployeeCount'], axis = 1)

Out[12]:

	EmployeeID	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	Gender	JobLevel	...	TotalWorkingYears	TrainingTimesLastYear	YearsAtCompany	Years
0	1	51	No	Travel_Rarely	Sales	6	2	Life Sciences	Female	1	...	1.0	6	1	
1	2	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	Female	1	...	6.0	3	5	
2	3	32	No	Travel_Frequently	Research & Development	17	4	Other	Male	4	...	5.0	2	5	
3	4	38	No	Non-Travel	Research & Development	2	5	Life Sciences	Male	3	...	13.0	5	8	
4	5	32	No	Travel_Rarely	Research & Development	10	1	Medical	Male	1	...	9.0	2	6	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4404	4405	29	No	Travel_Rarely	Sales	4	3	Other	Female	2	...	6.0	2	6	
4405	4406	42	No	Travel_Rarely	Research & Development	5	4	Medical	Female	1	...	10.0	5	3	
4406	4407	29	No	Travel_Rarely	Research & Development	2	4	Medical	Male	1	...	10.0	2	3	
4407	4408	26	No	Travel_Rarely	Research & Development	25	2	Life Sciences	Male	2	...	5.0	4	4	
4408	4409	42	No	Travel_Rarely	Sales	18	2	Medical	Male	1	...	10.0	2	9	

4200 rows × 28 columns

In [13]: attrition_dummies = pd.get_dummies(df['Attrition'])

In [14]: attrition_dummies.head()

Out[14]:

	No	Yes
0	1	0
1	0	1
2	1	0
3	1	0
4	1	0

In [15]: df = pd.concat([df, attrition_dummies], axis =1)

In [16]: df.head()

Out[16]:

	EmployeeID	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Gender	JobLevel	...	YearsAtCompany	YearsSinceLastPromotion	YearsWithCurrMan
0	1	51	No	Travel_Rarely	Sales	6	2	Life Sciences	1	Female	1	...	1	0	
1	2	31	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	1	Female	1	...	5	1	
2	3	32	No	Travel_Frequently	Research & Development	17	4	Other	1	Male	4	...	5	0	
3	4	38	No	Non-Travel	Research & Development	2	5	Life Sciences	1	Male	3	...	8	7	
4	5	32	No	Travel_Rarely	Research & Development	10	1	Medical	1	Male	1	...	6	0	

5 rows × 31 columns

In [17]: df = df.drop(['Attrition', 'No'], axis = 1)

In [18]: df.head()

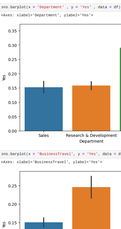
Out[18]:

	EmployeeID	Age	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Gender	JobLevel	...	TrainingTimesLastYear	YearsAtCompany	YearsSinceLastProm
0	1	51	Travel_Rarely	Sales	6	2	Life Sciences	1	Female	1	...	6	1	
1	2	31	Travel_Frequently	Research & Development	10	1	Life Sciences	1	Female	1	...	3	5	
2	3	32	Travel_Frequently	Research & Development	17	4	Other	1	Male	4	...	2	5	
3	4	38	Non-Travel	Research & Development	2	5	Life Sciences	1	Male	3	...	5	8	
4	5	32	Travel_Rarely	Research & Development	10	1	Medical	1	Male	1	...	2	6	

5 rows × 29 columns

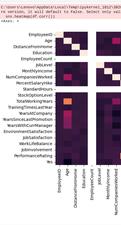
In [19]: sns.barplot(x = 'Gender', y = 'Yes', data = df)

Out[19]: <Axes: xlabel='Gender', ylabel='Yes'>



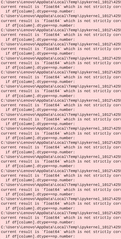
In [20]: sns.barplot(x = 'Department', y = 'Yes', data = df)

Out[20]: <Axes: xlabel='Department', ylabel='Yes'>



In [21]: sns.barplot(x = 'BusinessTravel', y = 'Yes', data = df)

Out[21]: <Axes: xlabel='BusinessTravel', ylabel='Yes'>



In [22]: pt.figure(figsize = (20,6))
sns.heatmap(df.corr())

```
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\3929928256.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a fu
ture version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr())
```

Out[22]: <Axes: >

In [23]: df = df.drop(['Age', 'JobLevel'], axis= 1)

Data preprocessing

converting string columns into integers

In [24]:
```
from sklearn.preprocessing import LabelEncoder
for column in df.columns:
    if df[column].dtype==np.number:
        continue
    else:
        df[column]=LabelEncoder().fit_transform(df[column])
```

```
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
C:\Users\ramon\AppData\Local\Temp\ipykernel_10127\4209844400.py:3: DeprecationWarning: Converting np.inexact or np.floating to a dtype is deprecated. The
current result is float64 which is not strictly correct.
  if df[column].dtype==np.number:
```

Model Building

In [25]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
```

In [26]:
```
x = df.drop(['Yes'], axis = 1)
y = df['Yes']
```

In [27]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 0)

In [28]: x_train.head()

Out[28]:

	EmployeeID	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Gender	JobRole	MaritalStatus	...	TotalWorkingYears	TrainingTimesLastYear	YearsAtCo
1384	1343	1	2	24	2	2	0	0	5	2	...	13.0	3	
4254	4146	2	2	3	3	1	0	0	0	2	...	1.0	3	
3644	3550	2	2	17	1	1	0	1	4	0	...	10.0	2	
1080	1047	2	1	11	3	1	0	0	7	2	...	1.0	3	
883	854	2	1	22	2	1	0	1	4	0	...	6.0	2	

5 rows × 26 columns

In [29]: rf.fit(x_train, y_train)

Out[29]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

In [30]: rf.score(x_train, y_train)

Out[30]: 0.9908710963455153

Predicting for x_test

In [31]: pred = rf.predict(x_test)

In [32]: from sklearn.metrics import accuracy_score

In [33]: accuracy_score(y_test, pred)

Out[33]: 0.9508952408928

Accuracy for Tested Data = 95.96%

In [34]: |