

Name Of The Intern : Sarthak Irappa Gadge

Title Of The Project : Employee Attrition Analysis

Technologies : Data Science

Domain : Human Resource

Importing the Dependencies

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import seaborn as sns
import matplotlib.pyplot as plt
```

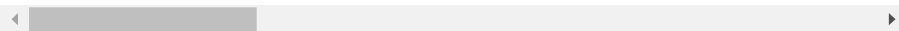
Data Collection and Processing

```
# Loading the csv data to Pandas DataFrame
df = pd.read_csv('/content/Attrition data.csv')
```

```
# Print the first 5 rows of the dataset
df.head()
```

	EmployeeID	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education
0	1	51	No	Travel_Rarely	Sales	6	2
1	2	31	Yes	Travel_Frequently	Research & Development	10	1
2	3	32	No	Travel_Frequently	Research & Development	17	4
3	4	38	No	Non-Travel	Research & Development	2	5
4	5	32	No	Travel_Rarely	Research & Development	10	1

5 rows × 29 columns



```
#Number of rows and columns in the dataset
df.shape
```

```
(4410, 29)
```

```
#statistical measures about the data
df.describe()
```

```
EmployeeID      Age  DistanceFromHome  Education  EmployeeCount  JobLevel  MonthlyIncome
# Checking the missing values
df.isnull().sum()

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
YearsSinceLastPromotion 0
YearsWithCurrManager 0
EnvironmentSatisfaction 25
JobSatisfaction  20
WorkLifeBalance  38
JobInvolvement   0
PerformanceRating 0
dtype: int64
```

```
attrition_count = pd.DataFrame(df['Attrition'].value_counts())
```

```
attrition_count
```

Attrition	
No	3699
Yes	711

```
plt.pie(attrition_count['Attrition'] , labels = ['No' , 'Yes'] , explode = (0.2,0))
```

```
([<matplotlib.patches.Wedge at 0x7fa6d61a9490>,
 <matplotlib.patches.Wedge at 0x7fa6d61a96a0>],
 [Text(-1.136781068348268, 0.6306574368426737, 'No'),
  Text(0.961891673217765, -0.5336332157899548, 'Yes')])
```



Splitting

```
df.drop(['EmployeeID' , 'Age'] , axis = 1)
```

	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCour
0	No	Travel_Rarely	Sales	6	2	Life Sciences	
1	Yes	Travel_Frequently	Research & Development	10	1	Life Sciences	
2	No	Travel_Frequently	Research & Development	17	4	Other	
3	No	Non-Travel	Research & Development	2	5	Life Sciences	
4	No	Travel_Rarely	Research & Development	10	1	Medical	
...	
4405	No	Travel_Rarely	Research & Development	5	4	Medical	
4406	No	Travel_Rarely	Research & Development	2	4	Medical	
4407	No	Travel_Rarely	Research & Development	25	2	Life Sciences	
4408	No	Travel_Rarely	Sales	18	2	Medical	
4409	No	Travel_Rarely	Research & Development	28	3	Medical	

4410 rows × 27 columns

```
attrition_dummies = pd.get_dummies(df['Attrition'])
attrition_dummies.head()
```

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

Concaneting The DataFrame & attrition_dummies

```
df = pd.concat([df, attrition_dummies] , axis = 1)
df.head
```

	<bound method NDFrame.head of	EmployeeID	Age	Attrition	BusinessTravel	Department	\
0	1 51	No	Travel_Rarely		Sales		
1	2 31	Yes	Travel_Frequently		Research & Development		
2	3 32	No	Travel_Frequently		Research & Development		
3	4 38	No	Non-Travel		Research & Development		
4	5 32	No	Travel_Rarely		Research & Development		
...
4405	4406 42	No	Travel_Rarely		Research & Development		
4406	4407 29	No	Travel_Rarely		Research & Development		
4407	4408 25	No	Travel_Rarely		Research & Development		
4408	4409 42	No	Travel_Rarely		Sales		
4409	4410 40	No	Travel_Rarely		Research & Development		
	DistanceFromHome	Education	EducationField	EmployeeCount	Gender	...	\
0	6	2	Life Sciences	1	Female	...	
1	10	1	Life Sciences	1	Female	...	
2	17	4	Other	1	Male	...	
3	2	5	Life Sciences	1	Male	...	
4	10	1	Medical	1	Male	...	
...
4405	5	4	Medical	1	Female	...	
4406	2	4	Medical	1	Male	...	

4407	25	2	Life Sciences	1	Male	...
4408	18	2	Medical	1	Male	...
4409	28	3	Medical	1	Male	...
	YearsAtCompany	YearsSinceLastPromotion	YearsWithCurrManager	\		
0	1	0	0			
1	5	1	4			
2	5	0	3			
3	8	7	5			
4	6	0	4			
...			
4405	3	0	2			
4406	3	0	2			
4407	4	1	2			
4408	9	7	8			
4409	21	3	9			

	EnvironmentSatisfaction	JobSatisfaction	WorkLifeBalance	\	
0	3.0	4.0	2.0		
1	3.0	2.0	4.0		
2	2.0	2.0	1.0		
3	4.0	4.0	3.0		
4	4.0	1.0	3.0		
...		
4405	4.0	1.0	3.0		
4406	4.0	4.0	3.0		
4407	1.0	3.0	3.0		
4408	4.0	1.0	3.0		
4409	1.0	3.0	NaN		

	JobInvolvement	PerformanceRating	No	Yes
0	3	3	1	0
1	2	4	0	1
2	3	3	1	0
3	2	3	1	0
4	2	2	1	0

Splitting

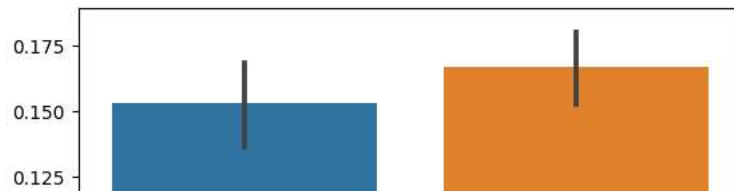
```
df = df.drop(['Attrition' , 'No'] , axis = 1)
df.head()
```

	EmployeeID	Age	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeC
0	1	51	Travel_Rarely	Sales	6	2	Life Sciences	
1	2	31	Travel_Frequently	Research & Development	10	1	Life Sciences	
2	3	32	Travel_Frequently	Research & Development	17	4	Other	
3	4	38	Non-Travel	Research & Development	2	5	Life Sciences	
4	5	32	Travel_Rarely	Research & Development	10	1	Medical	

5 rows × 29 columns

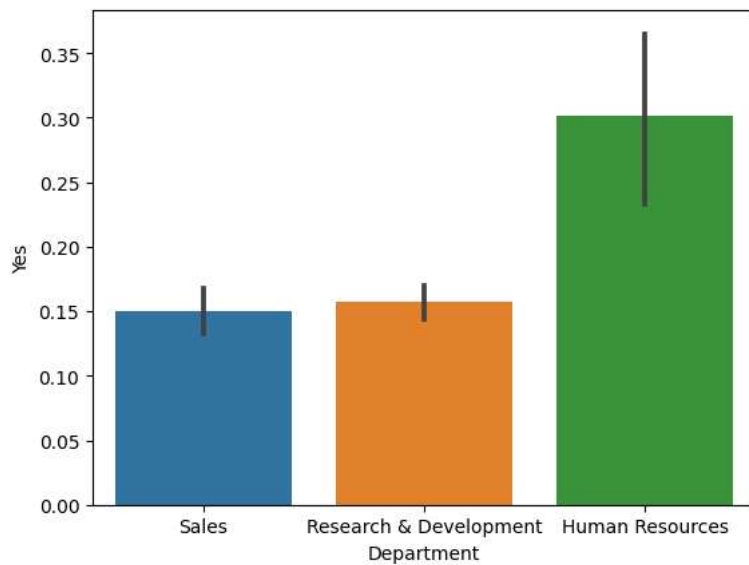
```
sns.barplot(x = 'Gender' , y = 'Yes', data = df)
```

<Axes: xlabel='Gender', ylabel='Yes'>



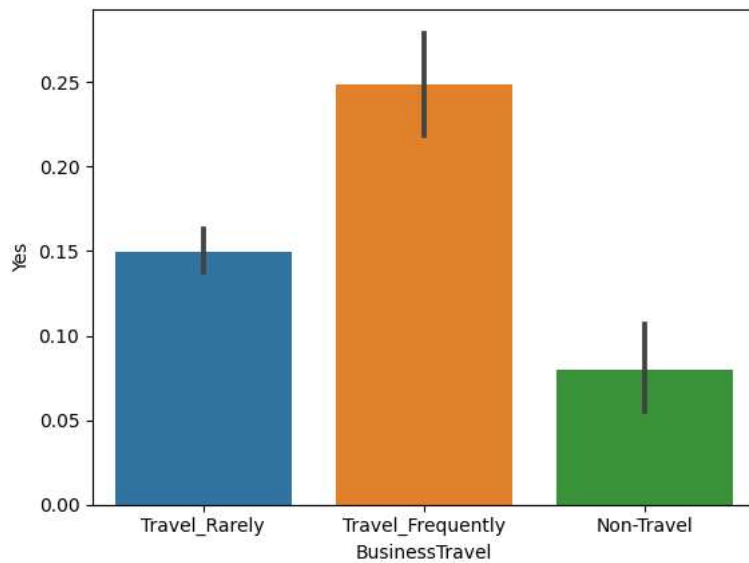
```
sns.barplot(x = 'Department', y = 'Yes', data = df)
```

<Axes: xlabel='Department', ylabel='Yes'>

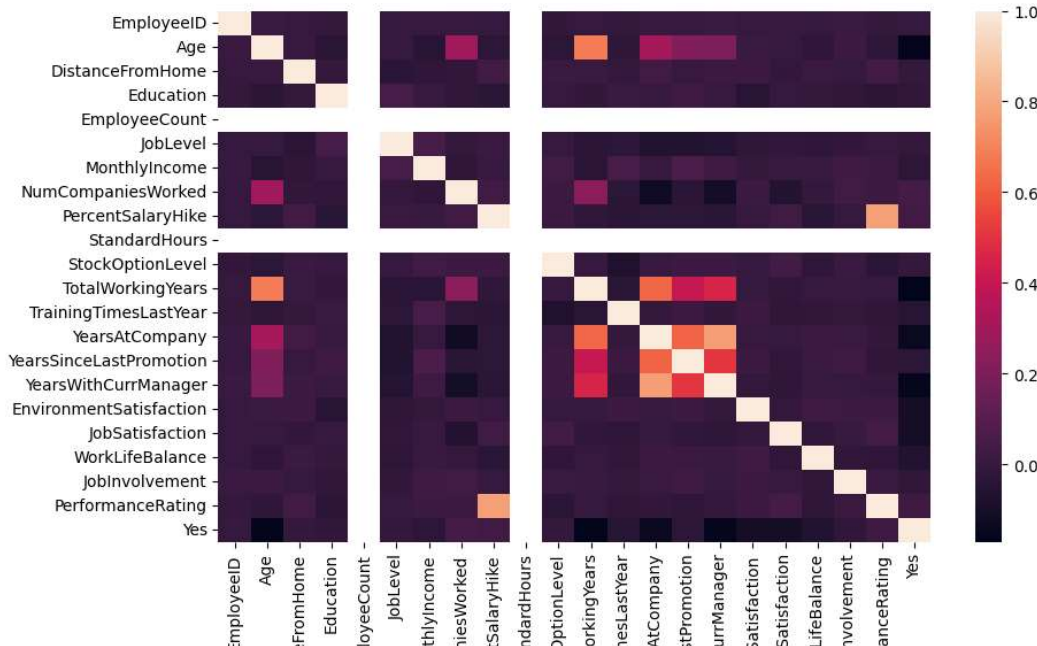


```
sns.barplot(x = 'BusinessTravel', y = 'Yes', data = df)
```

<Axes: xlabel='BusinessTravel', ylabel='Yes'>



```
plt.figure(figsize = (10,6))
sns.heatmap(df.corr())
```



Year	Area	Population	Population Density
1990	1	100	100
1990	2	200	200
1990	3	300	300
1990	4	400	400
1990	5	500	500
1990	6	600	600
1990	7	700	700
1990	8	800	800
1990	9	900	900
1990	10	1000	1000
1990	11	1100	1100
1990	12	1200	1200
1990	13	1300	1300
1990	14	1400	1400
1990	15	1500	1500
1990	16	1600	1600
1990	17	1700	1700
1990	18	1800	1800
1990	19	1900	1900
1990	20	2000	2000
1990	21	2100	2100
1990	22	2200	2200
1990	23	2300	2300
1990	24	2400	2400
1990	25	2500	2500
1990	26	2600	2600
1990	27	2700	2700
1990	28	2800	2800
1990	29	2900	2900
1990	30	3000	3000
1990	31	3100	3100
1990	32	3200	3200
1990	33	3300	3300
1990	34	3400	3400
1990	35	3500	3500
1990	36	3600	3600
1990	37	3700	3700
1990	38	3800	3800
1990	39	3900	3900
1990	40	4000	4000
1990	41	4100	4100
1990	42	4200	4200
1990	43	4300	4300
1990	44	4400	4400
1990	45	4500	4500
1990	46	4600	4600
1990	47	4700	4700
1990	48	4800	4800
1990	49	4900	4900
1990	50	5000	5000
1990	51	5100	5100
1990	52	5200	5200
1990	53	5300	5300
1990	54	5400	5400
1990	55	5500	5500
1990	56	5600	5600
1990	57	5700	5700
1990	58	5800	5800
1990	59	5900	5900
1990	60	6000	6000
1990	61	6100	6100
1990	62	6200	6200
1990	63	6300	6300
1990	64	6400	6400
1990	65	6500	6500
1990	66	6600	6600
1990	67	6700	6700
1990	68	6800	6800
1990	69	6900	6900
1990	70	7000	7000
1990	71	7100	7100
1990	72	7200	7200
1990	73	7300	7300
1990	74	7400	7400
1990	75	7500	7500
1990	76	7600	7600
1990	77	7700	7700
1990	78	7800	7800
1990	79	7900	7900
1990	80	8000	8000
1990	81	8100	8100
1990	82	8200	8200
1990	83	8300	8300
1990	84	8400	8400
1990	85	8500	8500
1990	86	8600	8600
1990	87	8700	8700
1990	88	8800	8800
1990	89	8900	8900
1990	90	9000	9000
1990	91		

```
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])
```

[illegible]

```
if df[column].dtype==np.number:
<ipython-input-23-b2b38dba6098>:3: DeprecationWarning: Converting `np.inexact` or `np.floating` to a dtype is deprecated. The current re
if df[column].dtype==np.number:

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)

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

Splitting The Data into Training Data & Test Data

```
x_train, x_test , y_train, y_test = train_test_split(x,y, test_size = 0.3, random_state = 0)

x_train.head()
```

	EmployeeID	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeCount
1087	1087	2	2	1	0	5	
1621	1621	1	2	9	2	2	
1804	1804	2	1	5	3	4	
744	744	1	1	5	3	3	
686	686	0	1	0	2	3	

5 rows × 26 columns