Work Integrated Learning Programmes

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Data Mining Assignment - Employee Attrition Prediction

Group-70

Group Members:

- Jatinder Kumar Chaurasia (2021C104115)
- Swapnil Tiwari (2021C104116)
- Rahul Nareshrao Shastri (2021C104122)

Importing Libraries:

utilizing matplotlib , pandas,seaborn,sklearn, hvplot,numpy

In [1]:
```
!pip install seaborn
```

```
!pip install sklearn
!pip install numpy
!pip install hvplot
!pip install imblearn
```

```
Requirement already satisfied: seaborn in /Users/jatinder/opt/anaconda3/lib/py
thon3.9/site-packages (0.11.2)
Requirement already satisfied: matplotlib>=2.2 in /Users/jatinder/opt/anaconda
3/lib/python3.9/site-packages (from seaborn) (3.4.3)
Requirement already satisfied: scipy>=1.0 in /Users/jatinder/opt/anaconda3/li
b/python3.9/site-packages (from seaborn) (1.7.1)
Requirement already satisfied: numpy>=1.15 in /Users/jatinder/opt/anaconda3/li
b/python3.9/site-packages (from seaborn) (1.20.3)
Requirement already satisfied: pandas>=0.23 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from seaborn) (1.3.4)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/jatinder/opt/anacon
da3/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (1.3.1)
Requirement already satisfied: pyparsing>=2.2.1 in /Users/jatinder/opt/anacond
a3/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (3.0.4)
Requirement already satisfied: pillow>=6.2.0 in /Users/jatinder/opt/anaconda3/
lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (8.4.0)
Requirement already satisfied: python-dateutil>=2.7 in /Users/jatinder/opt/ana
conda3/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: six in /Users/jatinder/opt/anaconda3/lib/python
3.9/site-packages (from cycler>=0.10->matplotlib>=2.2->seaborn) (1.16.0)
Requirement already satisfied: pytz>=2017.3 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from pandas>=0.23->seaborn) (2021.3)
```

```
Requirement already satisfied: sklearn in /Users/jatinder/opt/anaconda3/lib/py
thon3.9/site-packages (0.0)
Requirement already satisfied: scikit-learn in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from sklearn) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/jatinder/opt/ana
conda3/lib/python3.9/site-packages (from scikit-learn->sklearn) (2.2.0)
Requirement already satisfied: numpy>=1.14.6 in /Users/jatinder/opt/anaconda3/
lib/python3.9/site-packages (from scikit-learn->sklearn) (1.20.3)
Requirement already satisfied: joblib>=0.11 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from scikit-learn->sklearn) (1.1.0)
Requirement already satisfied: scipy>=1.1.0 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from scikit-learn->sklearn) (1.7.1)
Requirement already satisfied: numpy in /Users/jatinder/opt/anaconda3/lib/pyth
on3.9/site-packages (1.20.3)
Requirement already satisfied: hvplot in /Users/jatinder/opt/anaconda3/lib/pyt
hon3.9/site-packages (0.7.3)
Requirement already satisfied: numpy>=1.15 in /Users/jatinder/opt/anaconda3/li
b/python3.9/site-packages (from hvplot) (1.20.3)
Requirement already satisfied: holoviews>=1.11.0 in /Users/jatinder/opt/anacon
da3/lib/python3.9/site-packages (from hvplot) (1.14.7)
Requirement already satisfied: colorcet>=2 in /Users/jatinder/opt/anaconda3/li
b/python3.9/site-packages (from hvplot) (3.0.0)
Requirement already satisfied: pandas in /Users/jatinder/opt/anaconda3/lib/pyt
hon3.9/site-packages (from hvplot) (1.3.4)
Requirement already satisfied: bokeh>=1.0.0 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from hvplot) (2.4.1)
Requirement already satisfied: packaging>=16.8 in /Users/jatinder/opt/anaconda
3/lib/python3.9/site-packages (from bokeh>=1.0.0->hvplot) (21.0)
Requirement already satisfied: typing-extensions>=3.10.0 in /Users/jatinder/op
```

```
t/anaconda3/lib/python3.9/site-packages (from bokeh>=1.0.0->hvplot) (3.10.0.2)
Requirement already satisfied: Jinja2>=2.9 in /Users/jatinder/opt/anaconda3/li
b/python3.9/site-packages (from bokeh>=1.0.0->hvplot) (2.11.3)
Requirement already satisfied: PyYAML>=3.10 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from bokeh>=1.0.0->hvplot) (6.0)
Requirement already satisfied: tornado>=5.1 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from bokeh>=1.0.0->hvplot) (6.1)
Requirement already satisfied: pillow>=7.1.0 in /Users/jatinder/opt/anaconda3/
lib/python3.9/site-packages (from bokeh>=1.0.0->hvplot) (8.4.0)
Requirement already satisfied: pyct>=0.4.4 in /Users/jatinder/opt/anaconda3/li
b/python3.9/site-packages (from colorcet>=2->hvplot) (0.4.8)
Requirement already satisfied: param>=1.7.0 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from colorcet>=2->hvplot) (1.12.0)
Requirement already satisfied: panel>=0.8.0 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from holoviews>=1.11.0->hvplot) (0.12.6)
Requirement already satisfied: pyviz-comms>=0.7.4 in /Users/jatinder/opt/anaco
nda3/lib/python3.9/site-packages (from holoviews>=1.11.0->hvplot) (2.1.0)
Requirement already satisfied: MarkupSafe>=0.23 in /Users/jatinder/opt/anacond
a3/lib/python3.9/site-packages (from Jinja2>=2.9->bokeh>=1.0.0->hvplot) (1.1.
1)
Requirement already satisfied: pyparsing>=2.0.2 in /Users/jatinder/opt/anacond
a3/lib/python3.9/site-packages (from packaging>=16.8->bokeh>=1.0.0->hvplot)
(3.0.4)
Requirement already satisfied: python-dateutil>=2.7.3 in /Users/jatinder/opt/a
naconda3/lib/python3.9/site-packages (from pandas->hvplot) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from pandas->hvplot) (2021.3)
Requirement already satisfied: markdown in /Users/jatinder/opt/anaconda3/lib/p
ython3.9/site-packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (3.3.6)
```

```
Requirement already satisfied: bleach in /Users/jatinder/opt/anaconda3/lib/pyth
on3.9/site-packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (4.0.0)
Requirement already satisfied: requests in /Users/jatinder/opt/anaconda3/lib/p
ython3.9/site-packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (2.26.0)
Requirement already satisfied: tqdm>=4.48.0 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (4.6
2.3)
Requirement already satisfied: six>=1.5 in /Users/jatinder/opt/anaconda3/lib/p
ython3.9/site-packages (from python-dateutil>=2.7.3->pandas->hvplot) (1.16.0)
Requirement already satisfied: webencodings in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from bleach->panel>=0.8.0->holoviews>=1.11.0->hvpl
ot) (0.5.1)
Requirement already satisfied: importlib-metadata>=4.4 in /Users/jatinder/opt/
anaconda3/lib/python3.9/site-packages (from markdown->panel>=0.8.0->holoviews>
=1.11.0->hvplot) (4.8.1)
Requirement already satisfied: zipp>=0.5 in /Users/jatinder/opt/anaconda3/lib/
python3.9/site-packages (from importlib-metadata>=4.4->markdown->panel>=0.8.0-
>holoviews>=1.11.0->hvplot) (3.6.0)
Requirement already satisfied: certifi>=2017.4.17 in /Users/jatinder/opt/anaco
nda3/lib/python3.9/site-packages (from requests->panel>=0.8.0->holoviews>=1.1
1.0->hvplot) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in /Users/jatinder/op
t/anaconda3/lib/python3.9/site-packages (from requests->panel>=0.8.0->holoview
s>=1.11.0->hvplot) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from requests->panel>=0.8.0->holoviews>=1.11.0->hv
plot) (3.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/jatinder/opt/an
aconda3/lib/python3.9/site-packages (from requests->panel>=0.8.0->holoviews>=
```

```
1.11.0->hvplot) (1.26.7)
Requirement already satisfied: imblearn in /Users/jatinder/opt/anaconda3/lib/p
ython3.9/site-packages (0.0)
Requirement already satisfied: imbalanced-learn in /Users/jatinder/opt/anacond
a3/lib/python3.9/site-packages (from imblearn) (0.9.0)
Requirement already satisfied: scipy>=1.1.0 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from imbalanced-learn->imblearn) (1.7.1)
Requirement already satisfied: numpy>=1.14.6 in /Users/jatinder/opt/anaconda3/
lib/python3.9/site-packages (from imbalanced-learn->imblearn) (1.20.3)
Requirement already satisfied: scikit-learn>=1.0.1 in /Users/jatinder/opt/anac
onda3/lib/python3.9/site-packages (from imbalanced-learn->imblearn) (1.0.2)
Requirement already satisfied: joblib>=0.11 in /Users/jatinder/opt/anaconda3/l
ib/python3.9/site-packages (from imbalanced-learn->imblearn) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/jatinder/opt/ana
conda3/lib/python3.9/site-packages (from imbalanced-learn->imblearn) (2.2.0)
```

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import hvplot.pandas  # noqa

### to split the training data and test data
from sklearn.model_selection import train_test_split,GridSearchCV,StratifiedK
### For encoding Categorical to numerical
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
#### Classfiers
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression

#### Classifiers Metrics
from sklearn.metrics import roc_curve,accuracy_score, roc_auc_score, precisio
```

In [3]:

```
import warnings
warnings.filterwarnings('ignore') # GridSearchCV create many warning - to ign

# pre-setting the NAN values;
df=pd.read_csv('../data/Attrition.csv',na_values=['NAN','NaN'])
```

Data Understanding

In [4]:

```
# checking the DatSet
df.shape
```

Out[4]: (1470, 33)

In [5]:

```
df.head()
```

Out[5]:

	Attrition	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Ec
0	Yes	41	Travel_Rarely	1102	Sales	1.0	2.0	
1	No	49	Travel_Frequently	279	Research & Development	8.0	1.0	
2	Yes	37	Travel_Rarely	1373	Research & Development	2.0	2.0	
3	No	33	Travel_Frequently	1392	Research & Development	3.0	4.0	
4	No	27	Travel_Rarely	591	Research & Development	2.0	1.0	

5 rows × 33 columns

In [6]:
```
# checking the data types of feature columns
df.dtypes
```

Out[6]:
```
Attrition                 object
Age                        int64
BusinessTravel            object
DailyRate                  int64
Department                object
DistanceFromHome         float64
```

```
Education                 float64
EducationField             object
EmployeeNumber              int64
EnvironmentSatisfaction     int64
Gender                     object
HourlyRate                float64
JobInvolvement              int64
JobLevel                    int64
JobRole                    object
JobSatisfaction             int64
MaritalStatus              object
MonthlyIncome             float64
MonthlyRate               float64
NumCompaniesWorked          int64
Over18                     object
OverTime                   object
PercentSalaryHike           int64
PerformanceRating           int64
RelationshipSatisfaction    int64
StockOptionLevel            int64
TotalWorkingYears           int64
TrainingTimesLastYear       int64
WorkLifeBalance             int64
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

In [7]:
```
# checking the NAN
df.isna().sum()
```

Out[7]:
```
Attrition                 0
Age                       0
BusinessTravel            0
DailyRate                 0
Department                0
DistanceFromHome          5
Education                 1
EducationField            0
EmployeeNumber            0
EnvironmentSatisfaction   0
Gender                    0
HourlyRate                5
JobInvolvement            0
JobLevel                  0
JobRole                   0
JobSatisfaction           0
MaritalStatus             0
MonthlyIncome             5
MonthlyRate               5
NumCompaniesWorked        0
Over18                    0
OverTime                  0
PercentSalaryHike         0
PerformanceRating         0
RelationshipSatisfaction  0
```

```
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

Data Exploration And Preprocessing

In [8]:
```
# checking if any cell is empty:
df.isnull().sum().head()
```

Out[8]:
```
Attrition          0
Age                0
BusinessTravel     0
DailyRate          0
Department         0
dtype: int64
```

In [9]:
```
# Removing Nan with their min
df['Education']=df['Education'].fillna(min(df['Education'].values.tolist()))
```

```
df['DistanceFromHome']=df['DistanceFromHome'].fillna(min(df['DistanceFromHome
# df['DistanceFromHome']=df['DistanceFromHome'].fillna(min(df['DistanceFromHo
# df['DistanceFromHome']=df['DistanceFromHome'].fillna(df['DistanceFromHome']

df['HourlyRate']=df['HourlyRate'].fillna(min(df['HourlyRate'].values.tolist()
df['MonthlyIncome']=df['MonthlyIncome'].fillna(min(df['MonthlyIncome'].values
df['MonthlyRate']=df['MonthlyRate'].fillna(min(df['MonthlyRate'].values.tolis
```

In [10]:

```
df.head(10)
```

Out[10]:

	Attrition	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Ec
0	Yes	41	Travel_Rarely	1102	Sales	1.0	2.0	
1	No	49	Travel_Frequently	279	Research & Development	8.0	1.0	
2	Yes	37	Travel_Rarely	1373	Research & Development	2.0	2.0	
3	No	33	Travel_Frequently	1392	Research & Development	3.0	4.0	
4	No	27	Travel_Rarely	591	Research & Development	2.0	1.0	
5	No	32	Travel_Frequently	1005	Research & Development	2.0	2.0	

	Attrition	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Ec
6	No	59	Travel_Rarely	1324	Research & Development	3.0	3.0	
7	No	30	Travel_Rarely	1358	Research & Development	24.0	1.0	
8	No	38	Travel_Frequently	216	Research & Development	23.0	3.0	
9	No	36	Travel_Rarely	1299	Research & Development	1.0	3.0	

10 rows × 33 columns

In [11]:
```
# rechecking if all the Nan got resolved
df.isna().sum()
```

Out[11]:
```
Attrition                 0
Age                       0
BusinessTravel            0
DailyRate                 0
Department                0
DistanceFromHome          0
Education                 0
EducationField            0
```

```
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

analysing the dataset values

In [12]:

```
# for i in range(33):
#     print(f'{df[df.columns[i]].name} : {df[df.columns[i]].unique()}')
datafr=[]
for i in df.columns:
    datafr.append([i, df[i].nunique(), df[i].drop_duplicates().values])
pd.DataFrame(datafr, columns = ['Features', 'Unique Number', 'Values'])
```

Out[12]:

	Features	Unique Number	Values
0	Attrition	2	[Yes, No]
1	Age	43	[41, 49, 37, 33, 27, 32, 59, 30, 38, 36, 35, 2...
2	BusinessTravel	3	[Travel_Rarely, Travel_Frequently, Non-Travel]
3	DailyRate	886	[1102, 279, 1373, 1392, 591, 1005, 1324, 1358,...
4	Department	3	[Sales, Research & Development, Human Resources]
5	DistanceFromHome	29	[1.0, 8.0, 2.0, 3.0, 24.0, 23.0, 16.0, 15.0, 2...
6	Education	5	[2.0, 1.0, 4.0, 3.0, 5.0]
7	EducationField	6	[Life Sciences, Other, Medical, Marketing, Tec...
8	EmployeeNumber	1470	[1, 2, 4, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16,...

	Features	Unique Number	Values
9	EnvironmentSatisfaction	4	[2, 3, 4, 1]
10	Gender	2	[Female, Male]
11	HourlyRate	71	[94.0, 61.0, 92.0, 56.0, 40.0, 79.0, 81.0, 67....
12	JobInvolvement	4	[3, 2, 4, 1]
13	JobLevel	5	[2, 1, 3, 4, 5]
14	JobRole	9	[Sales Executive, Research Scientist, Laborato...
15	JobSatisfaction	4	[4, 2, 3, 1]
16	MaritalStatus	3	[Single, Married, Divorced]
17	MonthlyIncome	1345	[5993.0, 5130.0, 2090.0, 2909.0, 3468.0, 3068....
18	MonthlyRate	1422	[19479.0, 24907.0, 2396.0, 23159.0, 16632.0, 1...
19	NumCompaniesWorked	10	[8, 1, 6, 9, 0, 4, 5, 2, 7, 3]
20	Over18	1	[Y]
21	OverTime	2	[Yes, No]
22	PercentSalaryHike	15	[11, 23, 15, 12, 13, 20, 22, 21, 17, 14, 16, 1...
23	PerformanceRating	2	[3, 4]
24	RelationshipSatisfaction	4	[1, 4, 2, 3]

	Features	Unique Number	Values
25	StockOptionLevel	4	[0, 1, 3, 2]
26	TotalWorkingYears	40	[8, 10, 7, 6, 12, 1, 17, 5, 3, 31, 13, 0, 26, ...
27	TrainingTimesLastYear	7	[0, 3, 2, 5, 1, 4, 6]
28	WorkLifeBalance	4	[1, 3, 2, 4]
29	YearsAtCompany	37	[6, 10, 0, 8, 2, 7, 1, 9, 5, 4, 25, 3, 12, 14,...
30	YearsInCurrentRole	19	[4, 7, 0, 2, 5, 9, 8, 3, 6, 13, 1, 15, 14, 16,...
31	YearsSinceLastPromotion	16	[0, 1, 3, 2, 7, 4, 8, 6, 5, 15, 9, 13, 12, 10,...
32	YearsWithCurrManager	18	[5, 7, 0, 2, 6, 8, 3, 11, 17, 1, 4, 12, 9, 10,...

In [13]:
```
data=df.Attrition.values.tolist()
df.Attrition.value_counts()
```

Out[13]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

Checking Missing Data if Any

In [14]:
```
df.isnull().sum().sort_values(ascending=False).head()
```

Out[14]:
```
Attrition                 0
MonthlyIncome             0
YearsSinceLastPromotion   0
YearsInCurrentRole        0
YearsAtCompany            0
dtype: int64
```

In [15]:
```
df.isna().sum().sort_values(ascending=False).head()
```

Out[15]:
```
Attrition                 0
MonthlyIncome             0
YearsSinceLastPromotion   0
YearsInCurrentRole        0
YearsAtCompany            0
dtype: int64
```

So there is no missing data ,let's check the data for any outliers and NAN text

In [16]:
```
df.describe()
```

Out[16]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber	Environ

	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber	Environ
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	
mean	36.923810	802.485714	9.136735	2.911565	1024.865306	
std	9.135373	403.509100	8.080581	1.025377	602.024335	
min	18.000000	102.000000	1.000000	1.000000	1.000000	
25%	30.000000	465.000000	2.000000	2.000000	491.250000	
50%	36.000000	802.000000	7.000000	3.000000	1020.500000	
75%	43.000000	1157.000000	14.000000	4.000000	1555.750000	
max	60.000000	1499.000000	29.000000	5.000000	2068.000000	

8 rows × 24 columns

1) Income related continuous features - Daily Rate, Hourly Rate, Monthly Income, Monthly Rate categorical features - Stock Option Level, Job Level, Job Role, Department 2) Job Satisfaction - Job Involvement, Environment Satisfaction, Job Satisfaction, Work Life Balance, RelationShip Satisfaction, Performance Rating, Distance From Home, Percent Salary Hike 3) Demographics - Age, Education, Ed. Field, Gender, Marital Status 4) Work-Place - Years At Company, Years In Current Role, Years Since Last Promotion, Years With Current Manager, Business Travel, Num Companies Worked, Over Time, Total Working Years,

Training Times Last Year and specific category with already redundant variables 5) Others - Emloyee Number, Employee Count, Over 18, Standard Hours

Removing the Redundant Columns

EmployeeNumber,Over18

As employeeno unique to each column and all employee are over 18
=> Now checking the distribution of datset

In [17]:
```
# removing two univariated features
df.drop(columns=['EmployeeNumber', 'Over18'], inplace=True)
```

In [18]:
```
df.shape
```

Out[18]:
```
(1470, 31)
```

In [19]:
```
# df.Attrition.replace(to_replace = dict(Yes = 1, No = 0), inplace = True)
```

In [20]:
```
df.head()
```

Out[20]:

	Attrition	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Ec

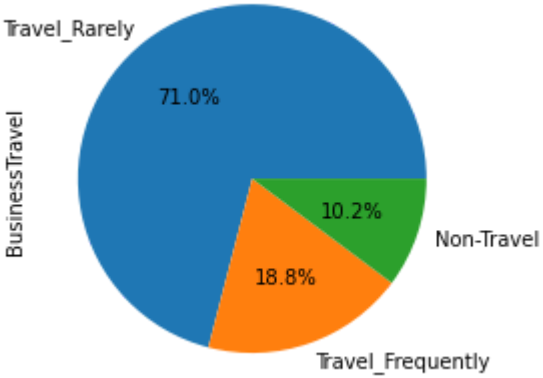
	Attrition	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Ec
0	Yes	41	Travel_Rarely	1102	Sales	1.0	2.0	
1	No	49	Travel_Frequently	279	Research & Development	8.0	1.0	
2	Yes	37	Travel_Rarely	1373	Research & Development	2.0	2.0	
3	No	33	Travel_Frequently	1392	Research & Development	3.0	4.0	
4	No	27	Travel_Rarely	591	Research & Development	2.0	1.0	

5 rows × 31 columns

Exploration Data Analysis - exploring the data and checking the distribution

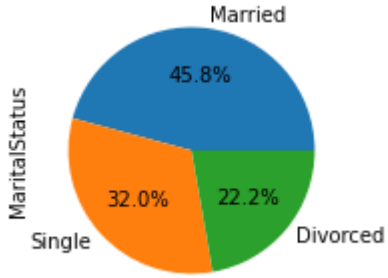
In [21]:
```
df['BusinessTravel'].value_counts().plot.pie(autopct='%1.1f%%')
```

Out[21]:
```
<AxesSubplot:ylabel='BusinessTravel'>
```



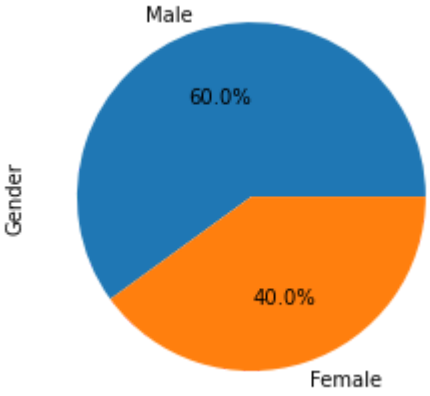
In [22]:
```
plt.figure(figsize=(23,13))
plt.subplot(1,1,1)
plt.subplot(2,2,2)
plt.subplot(3,3,3)
plt.subplot(4,3,3)
df.MaritalStatus.value_counts().plot.pie(autopct='%1.1f%%')
```

Out[22]:
```
<AxesSubplot:ylabel='MaritalStatus'>
```



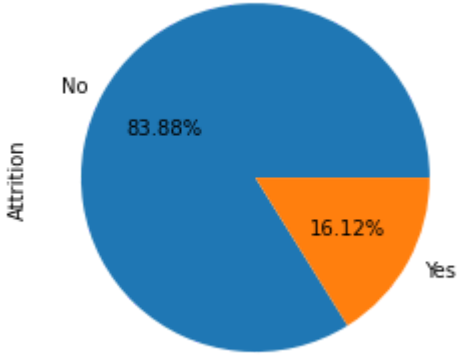
In [23]:
```
df.Gender.value_counts().plot.pie(autopct='%1.1f%%')
```

Out[23]:
```
<AxesSubplot:ylabel='Gender'>
```



In [24]:
```
df.Attrition.value_counts().plot.pie(autopct='%1.2f%%')
```

Out[24]:
```
<AxesSubplot:ylabel='Attrition'>
```



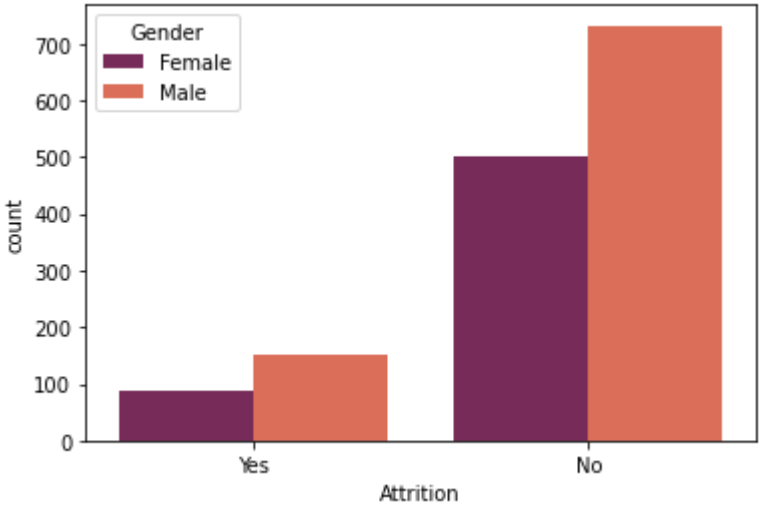
From the pie chart we are analyzing that around the rate of attrition is 16.12% and

---

From Here we are relating the columns with Attrition and checking which feature is more contributing to attrition

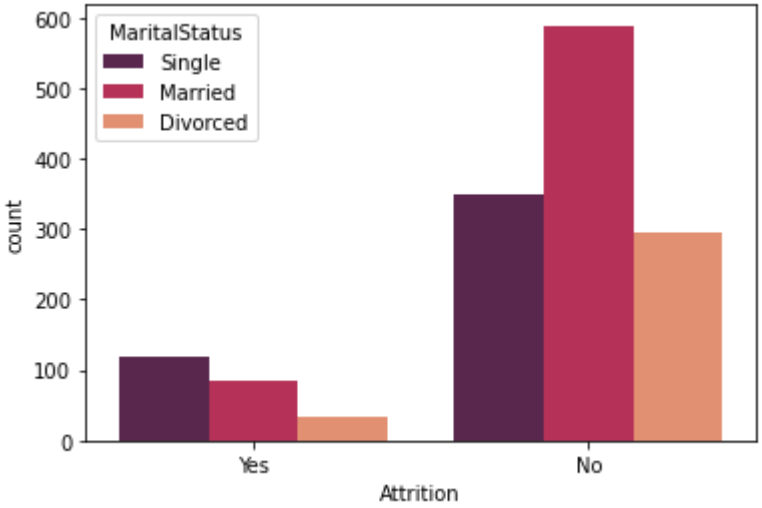
In [25]:
```
sns.countplot(x='Attrition', hue='Gender', palette = "rocket", data=df)
# Yes-1 No,-2
```

Out[25]:
```
<AxesSubplot:xlabel='Attrition', ylabel='count'>
```



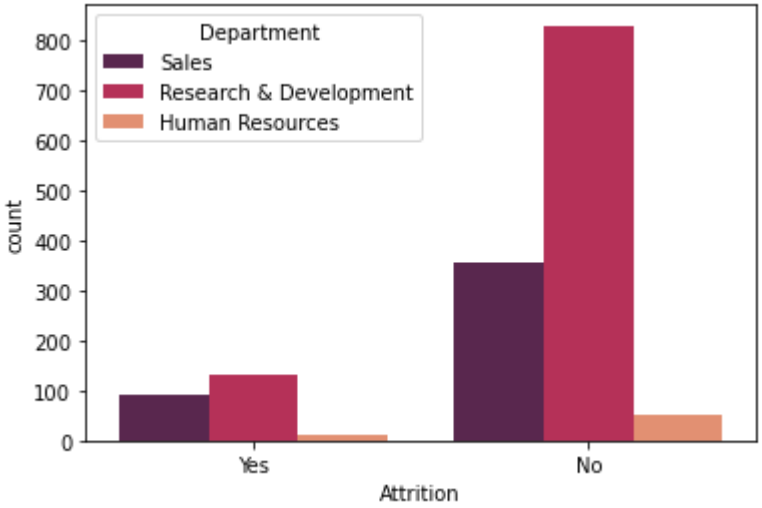
In [26]:
```
### as you can see the attrition is among male as compared to female

sns.countplot(x='Attrition', hue='MaritalStatus', palette = "rocket", data=df
```

Out[26]:
```
<AxesSubplot:xlabel='Attrition', ylabel='count'>
```



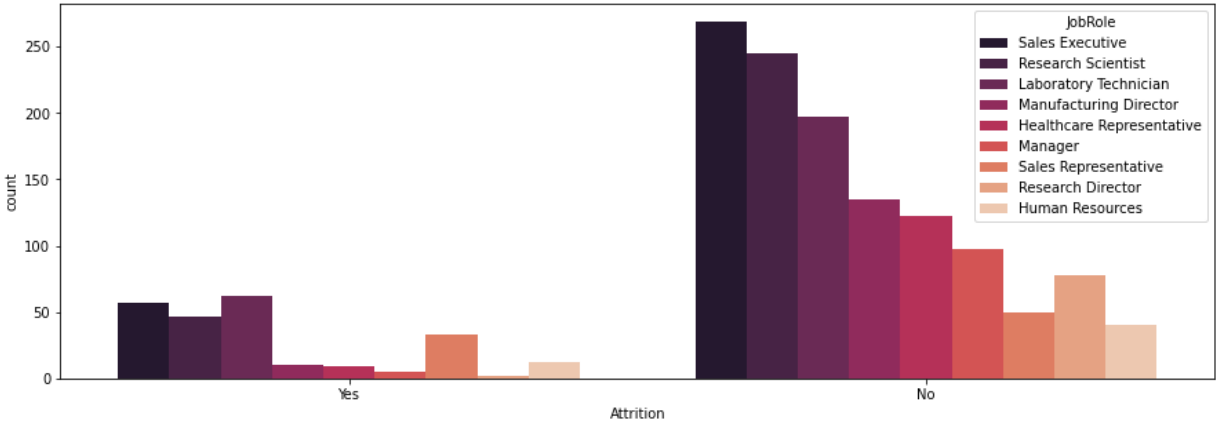
also it is greater among singles

In [27]:
```
sns.countplot(x='Attrition', hue='Department', palette = "rocket", data=df);
```

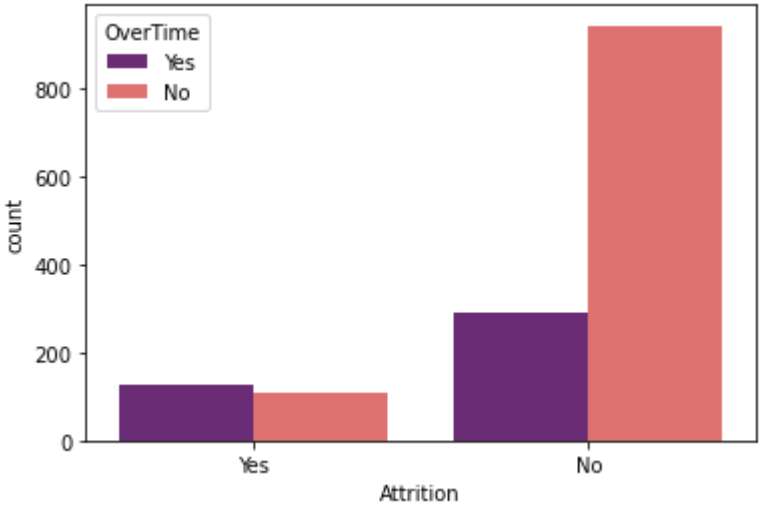


In [28]:

```
plt.figure(figsize=(15,5))
sns.countplot(x='Attrition', hue='JobRole', palette = "rocket", data=df);
```



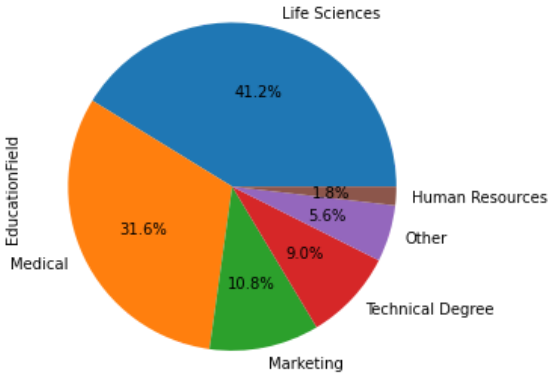
In [29]:
```
sns.countplot(x='Attrition', hue='OverTime', palette = "magma", data=df);
```

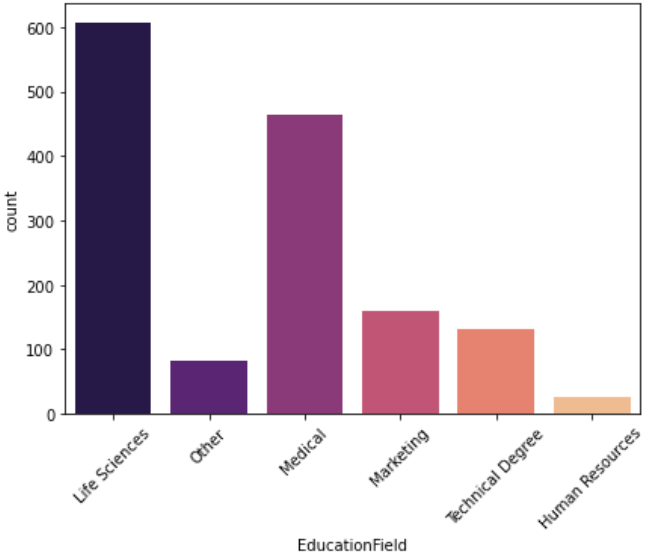


In [30]:
```
# sns.countplot(x='Attrition', hue='EducationField', palette = "magma", data=
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
df.EducationField.value_counts().plot.pie(autopct='%1.1f%%')
plt.subplot(1,2,2)
sns.countplot(x=df.EducationField,palette = "magma")
plt.xticks(rotation=45)
```

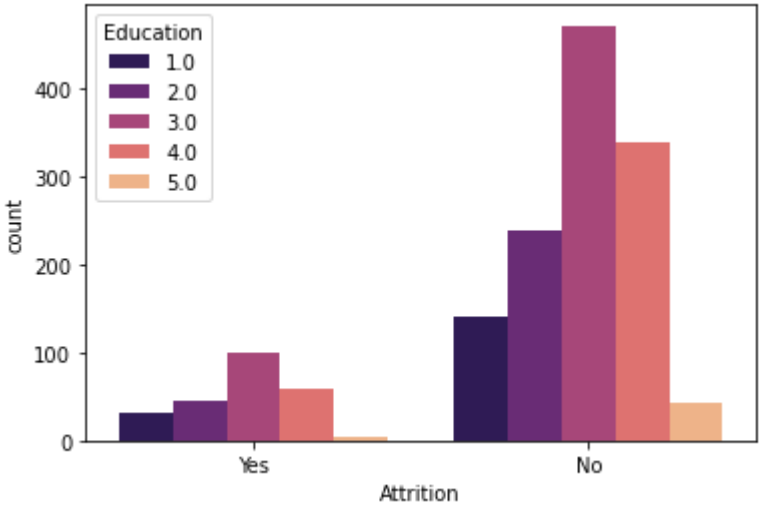
Out[30]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'Life Sciences'),
  Text(1, 0, 'Other'),
```

```
Text(2, 0, 'Medical'),
Text(3, 0, 'Marketing'),
Text(4, 0, 'Technical Degree'),
Text(5, 0, 'Human Resources')])
```

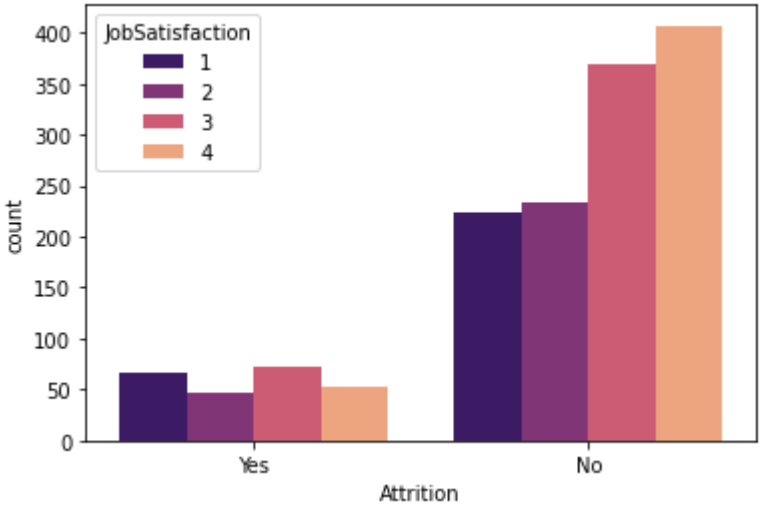




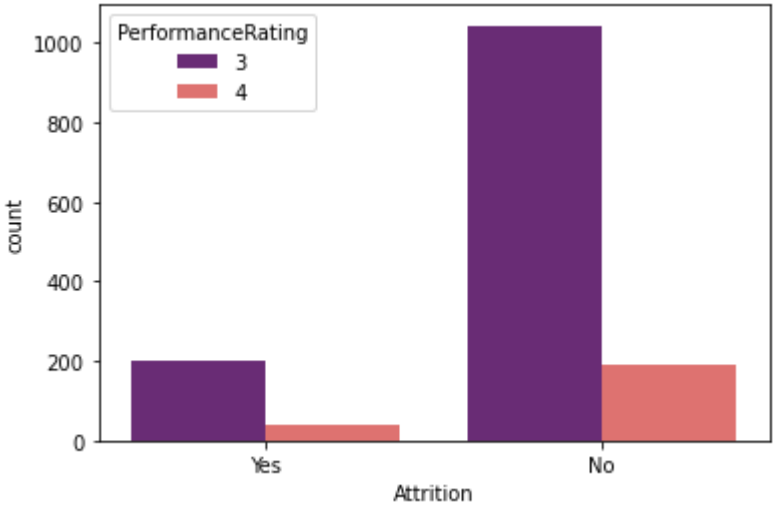
In [31]:
```
sns.countplot(x='Attrition', hue='Education', palette = "magma", data=df);
```



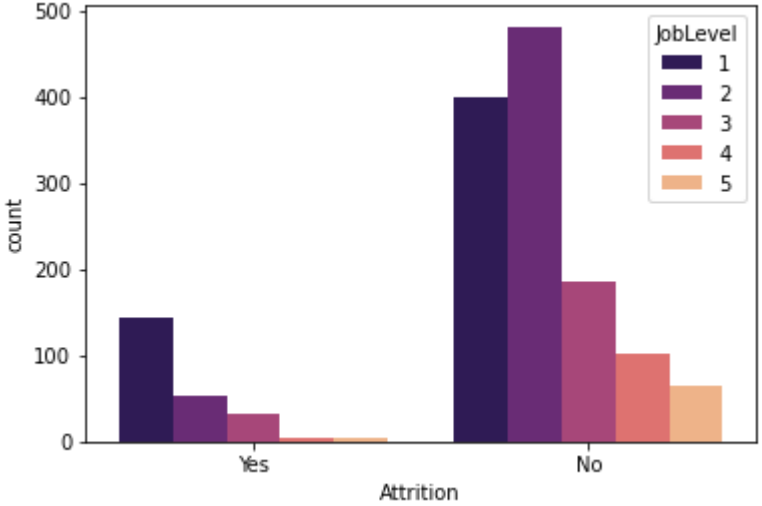
In [32]:
```
sns.countplot(x='Attrition', hue='JobSatisfaction', palette = "magma", data=d
```



In [33]:
```
sns.countplot(x='Attrition', hue='PerformanceRating', palette = "magma", data
```

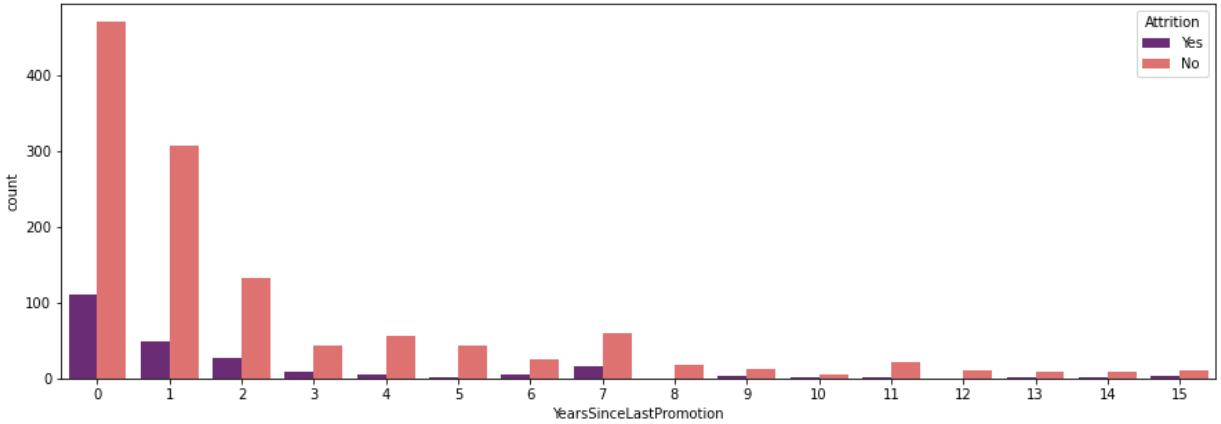


In [34]:
```
sns.countplot(x='Attrition', hue='JobLevel', palette = "magma", data=df);
```



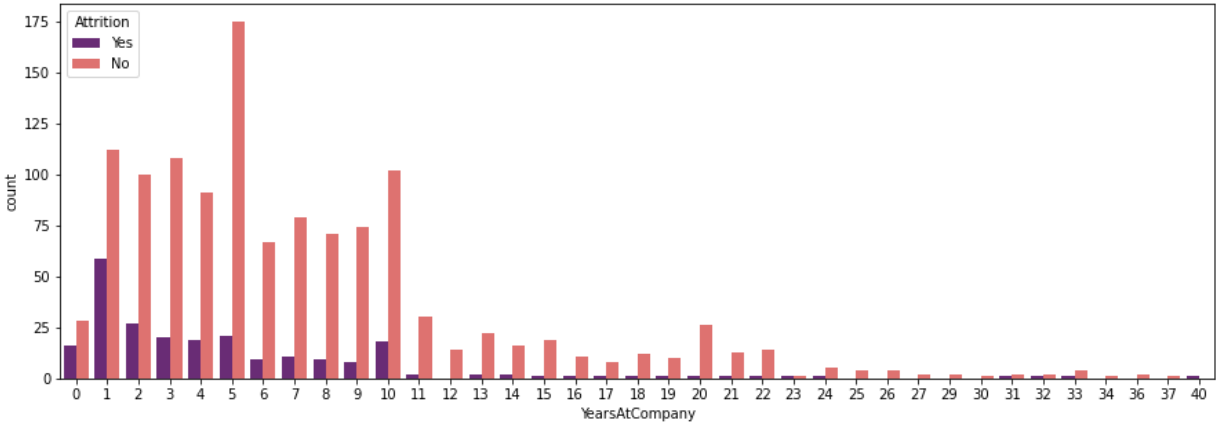
In [35]:

```
plt.figure(figsize=(15,5))
sns.countplot(x='YearsSinceLastPromotion', hue='Attrition', palette = "magma"
```



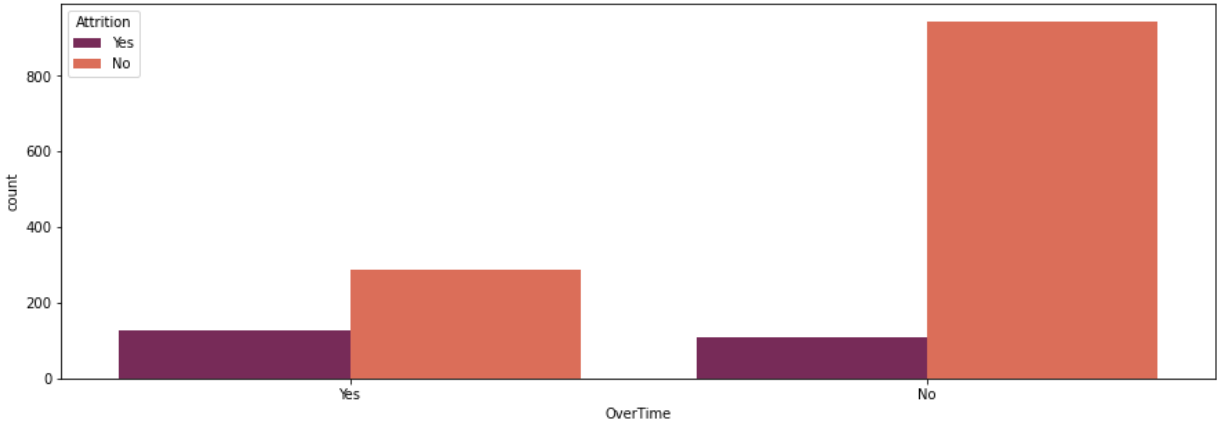
In [36]:

```
plt.figure(figsize=(15,5))
sns.countplot(x='YearsAtCompany', hue='Attrition', palette = "magma", data=df
```



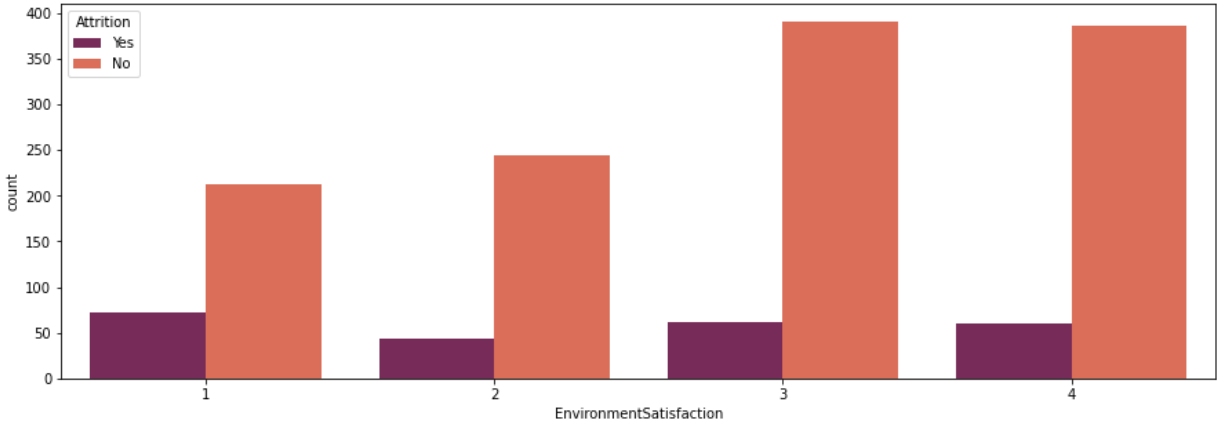
In [ ]:

In [37]:
```
plt.figure(figsize=(15,5))
sns.countplot(x='OverTime', hue='Attrition', palette = "rocket", data=df);
```



In [38]:

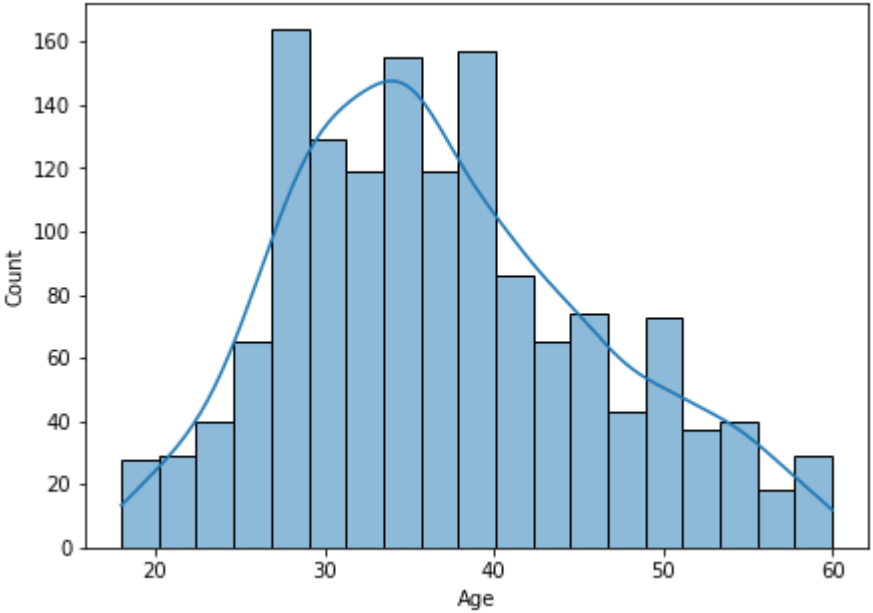
```
plt.figure(figsize=(15,5))
sns.countplot(x='EnvironmentSatisfaction', hue='Attrition', palette = "rocket
```



In [39]:

```
plt.figure(figsize=(7,5))
sns.histplot(df.Age,kde=True,palette='rocket')
print('Maximum',df.Age.max())
print('Minimum',df.Age.min())
```

```
Maximum 60
Minimum 18
```



Modeling

We are using two model for anylsis: Logistic Regression and Decision Tree Classifier
Logistic regression: Logistic regression is a statistical analysis method to predict a binary outcome,
Decision Tree Classifier: A Decision Tree is a supervised

Machine learning algorithm. It is used in both classification and regression algorithms

Confusion Matrix: to show the outcome believiability
we also comparing the accuracy of Models

In [40]:
```
# encoding categorical data to their numerical format for anlysis and correla
lenc = LabelEncoder()
df.Attrition = lenc.fit_transform(df.Attrition)
df.Department = lenc.fit_transform(df.Department)
df.EducationField = lenc.fit_transform(df.EducationField)
df.Gender = lenc.fit_transform(df.Gender)
df.JobRole = lenc.fit_transform(df.JobRole)
df.MaritalStatus = lenc.fit_transform(df.MaritalStatus)
df.BusinessTravel = lenc.fit_transform(df.BusinessTravel)
df.OverTime = lenc.fit_transform(df.OverTime)
```

In [41]:
```
df.head()
```

Out[41]:

	Attrition	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Edu
0	1	41	2	1102	2	1.0	2.0	
1	0	49	1	279	1	8.0	1.0	
2	1	37	2	1373	1	2.0	2.0	

	Attrition	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Edu
3	0	33	1	1392	1	3.0	4.0	
4	0	27	2	591	1	2.0	1.0	

5 rows × 31 columns

Feature Correlation

In [42]:
```
df.corr(method='pearson')
```

Out[42]:

	Attrition	Age	BusinessTravel	DailyRate	Department	Dista
Attrition	1.000000	-0.159205	0.000074	-0.056652	0.063991	
Age	-0.159205	1.000000	0.024751	0.010661	-0.031882	
BusinessTravel	0.000074	0.024751	1.000000	-0.004086	-0.009044	
DailyRate	-0.056652	0.010661	-0.004086	1.000000	0.007109	
Department	0.063991	-0.031882	-0.009044	0.007109	1.000000	
DistanceFromHome	0.080289	0.000722	-0.025079	-0.010000	0.016541	
Education	-0.030754	0.206614	0.003963	-0.017608	0.006127	

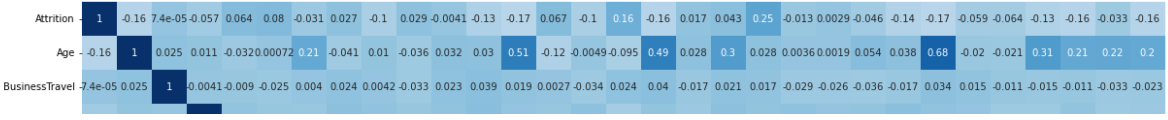
	Attrition	Age	BusinessTravel	DailyRate	Department	Dista
EducationField	0.026846	-0.040873	0.023724	0.037709	0.013720	
EnvironmentSatisfaction	-0.103369	0.010146	0.004174	0.018355	-0.019395	
Gender	0.029453	-0.036311	-0.032981	-0.011716	-0.041583	
HourlyRate	-0.004057	0.032393	0.023462	0.018696	-0.005693	
JobInvolvement	-0.130016	0.029820	0.039062	0.046135	-0.024586	
JobLevel	-0.169105	0.509604	0.019311	0.002966	0.101963	
JobRole	0.067151	-0.122427	0.002724	-0.009472	0.662431	
JobSatisfaction	-0.103481	-0.004892	-0.033962	0.030571	0.021001	
MaritalStatus	0.162070	-0.095029	0.024001	-0.069586	0.056073	
MonthlyIncome	-0.158107	0.494896	0.039559	0.007642	0.052898	
MonthlyRate	0.016647	0.028254	-0.016582	-0.029958	0.025373	
NumCompaniesWorked	0.043494	0.299635	0.020875	0.038153	-0.035882	
OverTime	0.246118	0.028062	0.016543	0.009135	0.007481	
PercentSalaryHike	-0.013478	0.003634	-0.029377	0.022704	-0.007840	
PerformanceRating	0.002889	0.001904	-0.026341	0.000473	-0.024604	
RelationshipSatisfaction	-0.045872	0.053535	-0.035986	0.007846	-0.022414	

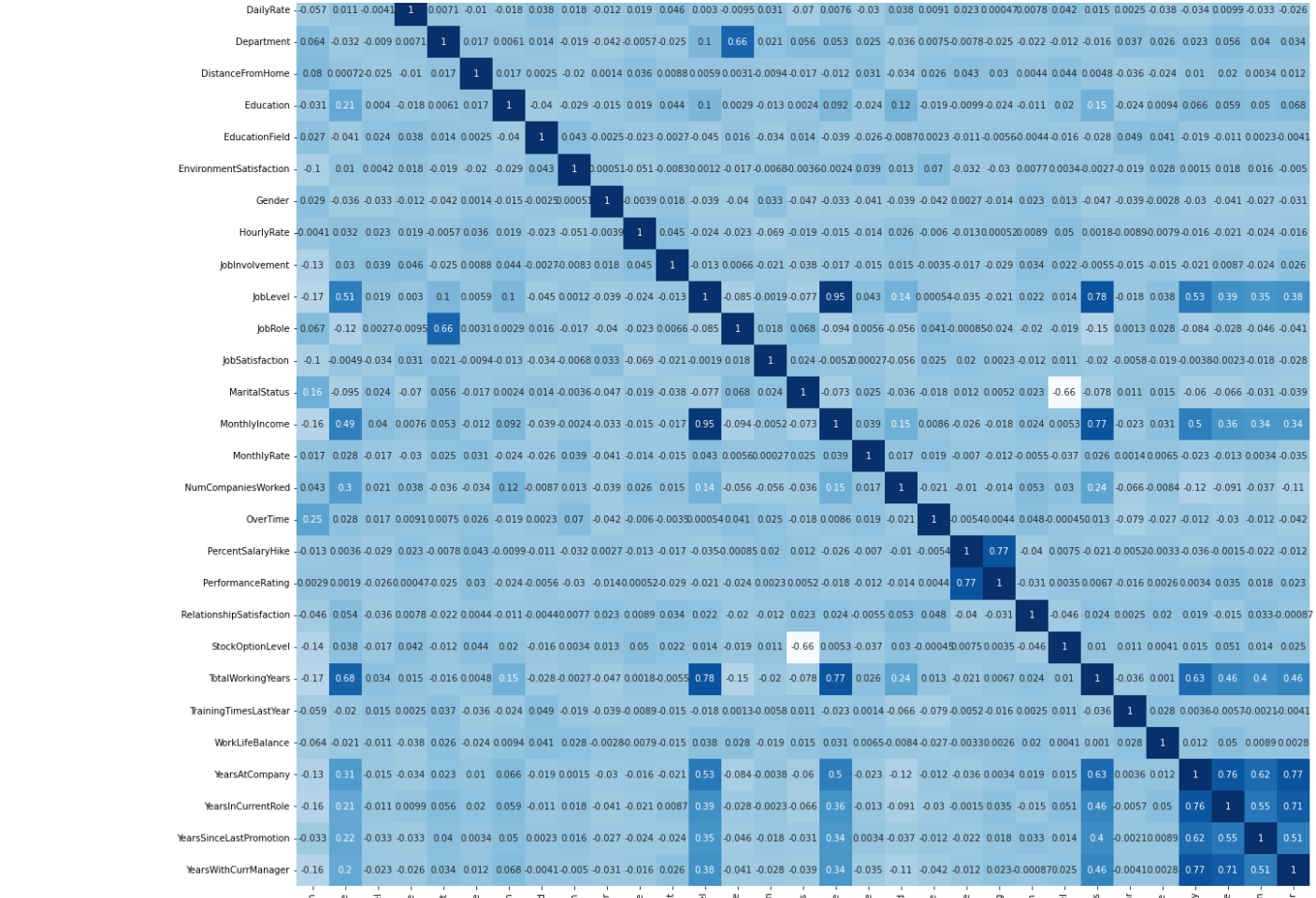
	Attrition	Age	BusinessTravel	DailyRate	Department	Dista
StockOptionLevel	-0.137145	0.037510	-0.016727	0.042143	-0.012193	
TotalWorkingYears	-0.171063	0.680381	0.034226	0.014515	-0.015762	
TrainingTimesLastYear	-0.059478	-0.019621	0.015240	0.002453	0.036875	
WorkLifeBalance	-0.063939	-0.021490	-0.011256	-0.037848	0.026383	
YearsAtCompany	-0.134392	0.311309	-0.014575	-0.034055	0.022920	
YearsInCurrentRole	-0.160545	0.212901	-0.011497	0.009932	0.056315	
YearsSinceLastPromotion	-0.033019	0.216513	-0.032591	-0.033229	0.040061	
YearsWithCurrManager	-0.156199	0.202089	-0.022636	-0.026363	0.034282	

31 rows × 31 columns

In [43]:
```
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(), annot=True, cmap="Blues",cbar=False)
```

Out[43]:
```
<AxesSubplot:>
```




DailyRate
Department
DistanceFromHome
Education
EducationField
EnvironmentSatisfaction
Gender
HourlyRate
JobInvolvement
JobLevel
JobRole
JobSatisfaction
MaritalStatus
MonthlyIncome
MonthlyRate
NumCompaniesWorked
OverTime
PercentSalaryHike
PerformanceRating
RelationshipSatisfaction
StockOptionLevel
TotalWorkingYears
TrainingTimesLastYear
WorkLifeBalance
YearsAtCompany
YearsInCurrentRole
YearsSinceLastPromotion
YearsWithCurrManager

From this heatmap plot we can see that

1. Attrition do have positive correlation with Age and MaritalStatus.
2. High corrleation B/w PerformanceRating and PerformanceSalaryHike
3. Features TotalWorkingYears,MonthlyIncome,JobLevel,YearsAtCompany,YearsAtCurrentRole,YearsSinc do have positive correlation/ highly related , so removing one of them will be affecting the other.

In [44]:
```
# checking the outliers on the dataset
def qr_outliers(col):
    outliers = []

    q1 = col.quantile(0.25)
    q3 = col.quantile(0.75)
    inter_qr = q3 - q1
    lower_limit = q1 - 1.5 * inter_qr
    upper_limit = q3 + 1.5 * inter_qr
    for val in col:
```

```
            if val > upper_limit or val < lower_limit:
                outliers.append(val)
        return outliers
```

In [45]:

```
col_outliers = []
for col in df.columns:
    if df[col].dtype == 'O' : continue
    else :
        outliers = qr_outliers(df[col])
        if outliers != []:
            col_outliers.append(col)
            print(col,':' ,outliers)
```

```
Attrition : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1]
MonthlyIncome : [19094.0, 18947.0, 19545.0, 18740.0, 18844.0, 18172.0, 17328.
0, 16959.0, 19537.0, 17181.0, 19926.0, 19033.0, 18722.0, 19999.0, 16792.0, 192
32.0, 19517.0, 19068.0, 19202.0, 19436.0, 16872.0, 19045.0, 19144.0, 17584.0,
18665.0, 17068.0, 19272.0, 18300.0, 16659.0, 19406.0, 19197.0, 19566.0, 18041.
0, 17046.0, 17861.0, 16835.0, 16595.0, 19502.0, 18200.0, 16627.0, 19513.0, 191
```

```
41.0, 19189.0, 16856.0, 19859.0, 18430.0, 17639.0, 16752.0, 19246.0, 17159.0,
17099.0, 17444.0, 17399.0, 19419.0, 18303.0, 19973.0, 19845.0, 17650.0, 19237.
0, 19627.0, 16756.0, 17665.0, 16885.0, 17465.0, 19626.0, 19943.0, 18606.0, 170
48.0, 17856.0, 19081.0, 17779.0, 19740.0, 18711.0, 18265.0, 18213.0, 18824.0,
18789.0, 19847.0, 19190.0, 18061.0, 17123.0, 16880.0, 17861.0, 19187.0, 19717.
0, 16799.0, 17328.0, 19701.0, 17169.0, 16598.0, 17007.0, 16606.0, 19586.0, 193
31.0, 19613.0, 17567.0, 19049.0, 19658.0, 17426.0, 17603.0, 16704.0, 19833.0,
19038.0, 19328.0, 19392.0, 19665.0, 16823.0, 17174.0, 17875.0, 19161.0, 19636.
0, 19431.0, 18880.0]
NumCompaniesWorked : [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
9, 9, 9, 9, 9, 9, 9]
PerformanceRating : [4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
StockOptionLevel : [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
TotalWorkingYears : [31, 29, 37, 38, 30, 40, 36, 34, 32, 33, 37, 30, 36, 31, 3
3, 32, 37, 31, 32, 32, 30, 34, 30, 40, 29, 35, 31, 33, 31, 29, 32, 30, 33, 30,
29, 31, 32, 33, 36, 34, 31, 36, 33, 31, 29, 33, 29, 32, 31, 35, 29, 32, 34, 3
6, 32, 30, 36, 29, 34, 37, 29, 29, 35]
```

```
TrainingTimesLastYear : [0, 5, 5, 5, 6, 5, 5, 5, 6, 6, 0, 0, 0, 5, 0, 5, 5, 5,
6, 6, 5, 0, 6, 5, 5, 0, 5, 5, 6, 5, 5, 5, 0, 5, 5, 5, 5, 6, 6, 5, 5, 5, 5, 0,
0, 5, 5, 5, 6, 6, 5, 0, 5, 0, 5, 5, 0, 6, 0, 5, 5, 6, 6, 5, 6, 5, 0, 5, 5, 5,
5, 0, 6, 5, 5, 5, 5, 6, 5, 5, 6, 5, 5, 5, 0, 5, 0, 5, 5, 6, 5, 6, 5, 0, 5, 5,
0, 6, 6, 5, 6, 0, 5, 0, 6, 6, 6, 6, 5, 5, 0, 5, 0, 0, 6, 0, 6, 5, 6, 5, 5, 0,
5, 6, 6, 5, 5, 0, 0, 6, 0, 0, 5, 0, 5, 6, 5, 5, 6, 6, 5, 5, 5, 5, 5, 6, 5, 6,
6, 0, 6, 6, 5, 5, 0, 0, 6, 6, 0, 5, 0, 0, 0, 0, 0, 5, 5, 6, 5, 5, 0, 5, 5, 0,
5, 5, 6, 5, 5, 5, 6, 5, 5, 5, 0, 0, 5, 5, 5, 5, 6, 0, 0, 6, 6, 6, 6, 5, 5, 5,
6, 5, 0, 5, 5, 6, 5, 6, 6, 5, 6, 6, 5, 0, 5, 5, 5, 5, 5, 0, 0, 0, 6, 5, 6, 6,
5, 6, 0, 6, 6, 5, 6, 6, 5, 5, 5, 0]
YearsAtCompany : [25, 22, 22, 27, 21, 22, 37, 25, 20, 40, 20, 24, 20, 24, 33,
20, 19, 22, 33, 24, 19, 21, 20, 36, 20, 20, 22, 24, 21, 21, 25, 21, 29, 20, 2
7, 20, 31, 32, 20, 20, 21, 22, 22, 34, 24, 26, 31, 20, 31, 26, 19, 21, 21, 32,
21, 19, 20, 22, 20, 21, 26, 20, 22, 24, 33, 29, 25, 21, 19, 19, 20, 19, 33, 1
9, 19, 20, 20, 20, 20, 20, 32, 20, 21, 33, 36, 26, 30, 22, 23, 23, 21, 21, 22,
22, 19, 22, 19, 22, 20, 20, 20, 22, 20, 20]
YearsInCurrentRole : [15, 16, 18, 15, 18, 17, 16, 15, 16, 15, 16, 16, 15, 16,
17, 15, 15, 15, 17, 17, 16]
YearsSinceLastPromotion : [8, 15, 8, 8, 9, 13, 12, 10, 11, 9, 12, 15, 15, 15,
9, 11, 11, 9, 12, 11, 15, 11, 10, 9, 11, 9, 8, 11, 11, 8, 13, 9, 9, 12, 10, 1
1, 15, 13, 9, 11, 10, 8, 8, 11, 9, 11, 12, 11, 14, 13, 14, 8, 11, 15, 10, 11,
11, 15, 11, 13, 11, 13, 15, 8, 13, 15, 11, 14, 15, 15, 9, 11, 9, 8, 9, 15, 11,
12, 9, 8, 10, 14, 8, 13, 13, 12, 14, 8, 8, 8, 14, 14, 8, 12, 13, 14, 14, 12, 1
1, 8, 11, 9, 12, 8, 9, 11, 9]
YearsWithCurrManager : [17, 15, 15, 15, 15, 17, 16, 17, 15, 17, 17, 17, 17, 1
6]
```

In [46]:
```
df.drop('Attrition', axis = 1).corrwith(df.Attrition).sort_values().hvplot.ba
```

Out[46]:

Modeling and Training the Dataset

splitting the dataset into test and train dataset

In [47]:
```
X = df.drop('Attrition', axis=1)
Y = df.Attrition

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, rand
X_train.shape
```

Out[47]:
```
(1029, 30)
```

In [48]:
```
Y_train.shape
```

Out[48]:
```
(1029,)
```

In [49]:
```
df.describe()
```

Out[49]:

	Attrition	Age	BusinessTravel	DailyRate	Department	DistanceFromHor
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.0000
mean	0.161224	36.923810	1.607483	802.485714	1.260544	9.1367
std	0.367863	9.135373	0.665455	403.509100	0.527792	8.0805
min	0.000000	18.000000	0.000000	102.000000	0.000000	1.0000
25%	0.000000	30.000000	1.000000	465.000000	1.000000	2.0000
50%	0.000000	36.000000	2.000000	802.000000	1.000000	7.0000
75%	0.000000	43.000000	2.000000	1157.000000	2.000000	14.0000
max	1.000000	60.000000	2.000000	1499.000000	2.000000	29.0000

8 rows × 31 columns

In [50]:

```
# checking if feature selection is working using variance
from sklearn.feature_selection import VarianceThreshold
selector = VarianceThreshold(1)
selector.fit(df)
df.columns[selector.get_support()]
```

Out[50]:

```
Index(['Age', 'DailyRate', 'DistanceFromHome', 'Education', 'EducationField',
       'EnvironmentSatisfaction', 'HourlyRate', 'JobLevel', 'JobRole',
```

```
       'JobSatisfaction', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorke
d',
       'PercentSalaryHike', 'RelationshipSatisfaction', 'TotalWorkingYears',
       'TrainingTimesLastYear', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager'],
      dtype='object')
```

In [51]:
```
##### Map of classifiers Models we are going to use in the dataset
```

In [52]:
```
classifiers = {
    'Logistic Regression' : LogisticRegression(),
    'Decision Tree Classifier': DecisionTreeClassifier(),
}

dtc = DecisionTreeClassifier()
dtc.fit(X_train,Y_train)
#Checking the accuracy score for train
y_train_pred = dtc.predict(X_train)
y_train_prob = dtc.predict_proba(X_train)[0,1]
accuracy_score(Y_train, y_train_pred)
```

Out[52]:
```
1.0
```

As you see the accuracy score is high seems to be problem of overfitting , we need to

balance/tune the data

We need to do hypertuning and use the best features for this model

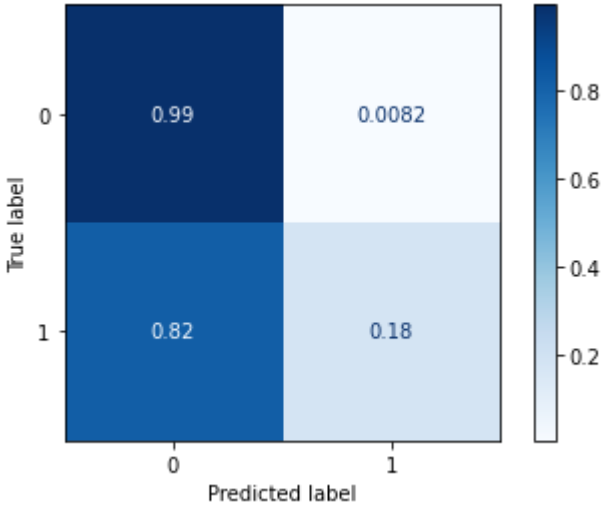
In [53]:

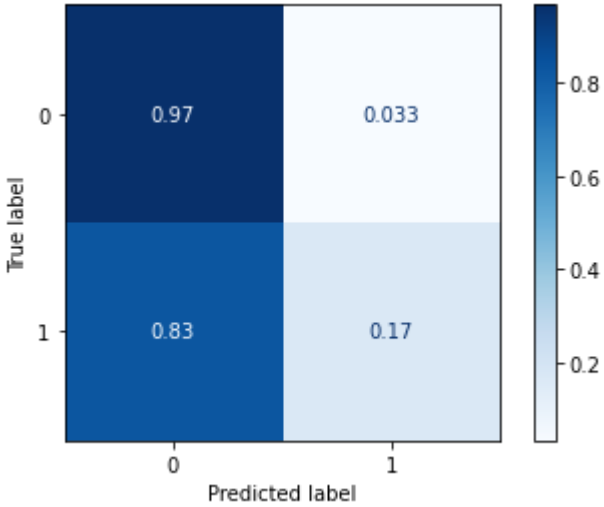
```
%%time

# using stratifiedKFold to improve the machine learning model speed/performan
skf = StratifiedKFold(n_splits = 5, random_state = None, shuffle = True)
data_cfg={}
parameters = {
    'Logistic Regression': {'penalty': ['l1', 'l2'], 'C':[0.01, 0.1, 1],'solv
    'Decision Tree Classifier': {'criterion': ['gini', 'entropy'], 'max_depth
}
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

model_scores={}

for name,params in parameters.items():
    classifier= classifiers.get(name)
    print(classifier)
    grid = GridSearchCV(classifier, params, cv = skf)
    grid.fit(X_train, Y_train)
    best_hyperparams = grid.best_params_
    best_estimator=grid.best_estimator_

    # grid.bestscore calculates the mean of r2 score across 5 folds of cross
    print(f'The best hyperparameters for: {name} : {best_hyperparams} with r2
```

```
    y_true, y_pred = Y_test, best_estimator.predict(X_test)

    y_train_pred = best_estimator.predict(X_train)
    y_train_prob = best_estimator.predict_proba(X_train)[0,1]
    print(f'Accuracy score for {name}: {accuracy_score(Y_train, y_train_pred)
    model_scores[name.replace(' ','')]=accuracy_score(Y_train, y_train_pred)
    y_test_pred = best_estimator.predict(X_test)
    y_test_prob = best_estimator.predict_proba(X_test)[:,1]
    print(f'ROC_AUC score for {name}: {roc_auc_score(Y_test, y_test_prob)}')
    print(f'Classification Report for {name}: \n {classification_report(y_tru
    nameFPR=name.replace(' ','') +'_false_positive_rate'
    nameTPR=name.replace(' ','') +'_true_positive_rate'
    nameTHR=name.replace(' ','') + '_thresholds'
    nameFPR,nameTPR,nameTHR = roc_curve(Y_test, y_test_prob)
    data=[nameFPR,nameTPR,nameTHR]
    data_cfg[name.replace(' ','')]=data
    plot_confusion_matrix(best_estimator, X_test, Y_test,normalize="true", cm
```

```
LogisticRegression()
The best hyperparameters for: Logistic Regression : {'C': 0.01, 'penalty': 'l
2', 'solver': 'liblinear'} with r2 score: 0.8697655695003552
Accuracy score for Logistic Regression: 0.8746355685131195
ROC_AUC score for Logistic Regression: 0.8411945197659483
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.85      0.99      0.92       364
           1       0.82      0.18      0.30        77
```

```
    accuracy                           0.85       441
   macro avg       0.84      0.59      0.61       441
weighted avg       0.85      0.85      0.81       441

DecisionTreeClassifier()
The best hyperparameters for: Decision Tree Classifier : {'criterion': 'entrop
y', 'max_depth': 4} with r2 score: 0.8445133791143735
Accuracy score for Decision Tree Classifier: 0.8794946550048591
ROC_AUC score for Decision Tree Classifier: 0.7269516198087627
Classification Report for Decision Tree Classifier:
              precision    recall  f1-score   support

           0       0.85      0.97      0.90       364
           1       0.52      0.17      0.25        77

    accuracy                           0.83       441
   macro avg       0.68      0.57      0.58       441
weighted avg       0.79      0.83      0.79       441

CPU times: user 2.66 s, sys: 780 ms, total: 3.44 s
Wall time: 1.13 s
```


0.99
0.0082
0.82
0.18
0
1
True label
0
1
Predicted label
0.8
0.6
0.4
0.2



As you see from the classifier the accuracy f1-score for Logistic Regression is high as compared to Decision Tree

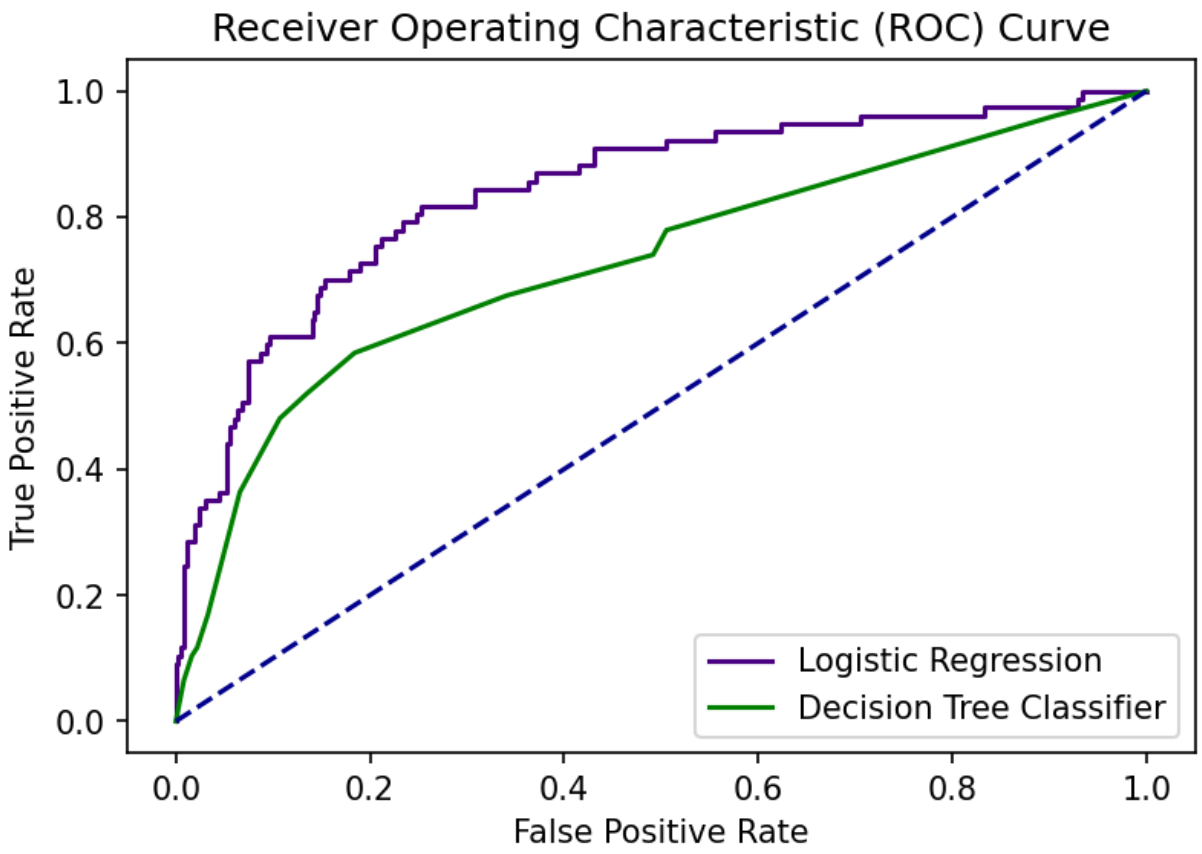
ROC curve (Receiver Operating Characteristic Curve) is a graph showing the performance of a classification model at all classification thresholds

In [54]:

```
plt.figure(dpi=150)
for name,_ in parameters.items():
    new_name=name.replace(' ','')
    data=data_cfg.get(new_name)
    # if 'Decision' in name
```

```
    color_d = 'green' if 'Decision' in name else 'indigo'
    plt.plot(data[0], data[1], color=color_d, label=f'{name}',animated=True,d

plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
```

Out[54]: <matplotlib.legend.Legend at 0x7fe104617ee0>


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Logistic Regression
Decision Tree Classifier

In [55]:

```
print(model_scores)
```

```
{'LogisticRegression': 0.8746355685131195, 'DecisionTreeClassifier': 0.8794946
550048591}
```

Model Comparison: comparing the accuracy of Logistic Regression and Decision Tree Classifier Models

In [56]:

```
model_compare=pd.DataFrame(model_scores,index=['accuracy'])
model_compare
```

Out[56]:

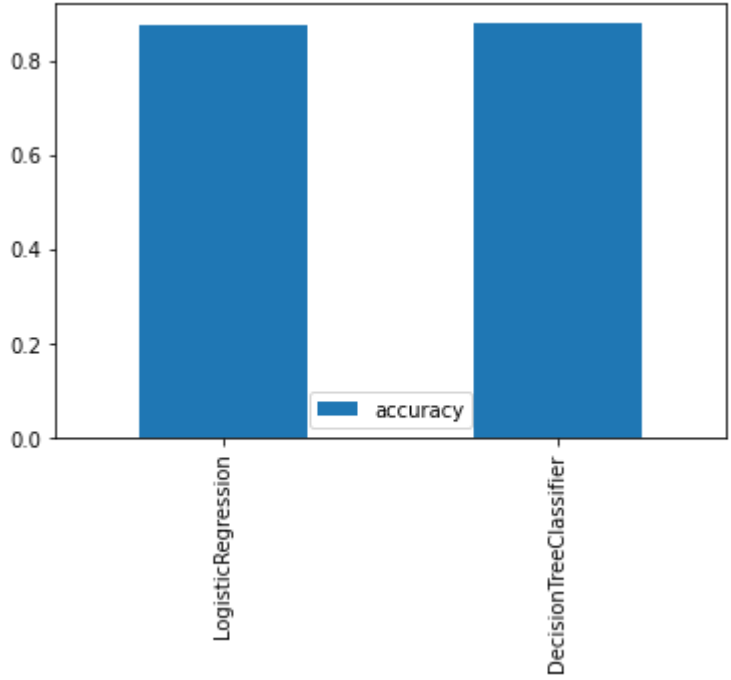
	LogisticRegression	DecisionTreeClassifier
accuracy	0.874636	0.879495

In [57]:

```
model_compare.T.plot(kind='bar') # (T is here for transpose)
```

Out[57]:

```
<AxesSubplot:>
```



We found that Logistic Regresssion is the best model for predicting Attrition after comparing accuracy against Decision Tree.

In [ ]: