Lesson 9

Python of ML

Step 1 import necessary python paskages

In [133]:
```
import pandas as pd
import seaborn as sns
import sklearn
```

Step 2 Read in the data set and understand the data

In [134]:
```
df = pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
# pd.read_csv('~/Desktop/Polaris/WA_Fn-UseC_-HR-Employee-Attrition.csv')
```

In [135]:
```
df.head()
```

Out[135]:

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

5 rows × 35 columns

In [136]:
```
# understand now many employee there
df.count()
```

Out[136]:
```
Age                         1470
Attrition                   1470
BusinessTravel              1470
DailyRate                   1470
Department                  1470
DistanceFromHome            1470
Education                   1470
EducationField              1470
EmployeeCount               1470
EmployeeNumber              1470
EnvironmentSatisfaction     1470
Gender                      1470
HourlyRate                  1470
JobInvolvement              1470
JobLevel                    1470
JobRole                     1470
JobSatisfaction             1470
MaritalStatus               1470
MonthlyIncome               1470
MonthlyRate                 1470
NumCompaniesWorked          1470
Over18                      1470
OverTime                    1470
PercentSalaryHike           1470
PerformanceRating           1470
RelationshipSatisfaction    1470
StandardHours               1470
StockOptionLevel            1470
TotalWorkingYears           1470
TrainingTimesLastYear       1470
WorkLifeBalance             1470
YearsAtCompany              1470
YearsInCurrentRole          1470
YearsSinceLastPromotion     1470
YearsWithCurrManager        1470
dtype: int64
```

In [137]:
```
df.columns
```

Out[137]:
```
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 [138]:
```python
# try to understand how the data distributed
df.describe()
```

Out[138]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	Environmen
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	

8 rows × 26 columns

In [139]:
```python
# look at the data, is it make sense or not? have a check
```

In [140]:
```
# try to look at data types of each column
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [141]:
```
# understand the meaning of the values of each column
df['Age']
```

Out[141]:
```
0       41
1       49
2       37
3       33
4       27
        ..
1465    36
1466    39
1467    27
1468    49
1469    34
Name: Age, Length: 1470, dtype: int64
```

In [142]:
```
# understand is there any types of travel in "BusinessTravel" column
set( df['BusinessTravel'] )
```

Out[142]:
```
{'Non-Travel', 'Travel_Frequently', 'Travel_Rarely'}
```

In [143]:
```
# for counting types of specific column
df['BusinessTravel'].value_counts()
```

Out[143]:
```
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
```

In [144]:
```
# want to know what specific values they have
df_obj = df.select_dtypes(include=['object'])
df_obj
```

Out[144]:

	Attrition	BusinessTravel	Department	EducationField	Gender	JobRole	MaritalStatus	Over18
0	Yes	Travel_Rarely	Sales	Life Sciences	Female	Sales Executive	Single	Y
1	No	Travel_Frequently	Research & Development	Life Sciences	Male	Research Scientist	Married	Y
2	Yes	Travel_Rarely	Research & Development	Other	Male	Laboratory Technician	Single	Y
3	No	Travel_Frequently	Research & Development	Life Sciences	Female	Research Scientist	Married	Y
4	No	Travel_Rarely	Research & Development	Medical	Male	Laboratory Technician	Married	Y
...	...	...	...	...	...	...	...	...
1465	No	Travel_Frequently	Research & Development	Medical	Male	Laboratory Technician	Married	Y
1466	No	Travel_Rarely	Research & Development	Medical	Male	Healthcare Representative	Married	Y
1467	No	Travel_Rarely	Research & Development	Life Sciences	Male	Manufacturing Director	Married	Y
1468	No	Travel_Frequently	Sales	Medical	Male	Sales Executive	Married	Y
1469	No	Travel_Rarely	Research & Development	Medical	Male	Laboratory Technician	Married	Y

1470 rows × 9 columns

In [145]:
```
# output the result of col in object type
# total 35 cols, 9 cols are types cols
```

In [146]:
```
# print out this table's cols
df_obj.columns
```

Out[146]:
```
Index(['Attrition', 'BusinessTravel', 'Department', 'EducationField', 'Gender',
       'JobRole', 'MaritalStatus', 'Over18', 'OverTime'],
      dtype='object')
```

In [147]:
```python
# if we dont want to look at BusinessTravel column, e.g. Travel_Rarely type, one by one,
# then we do below to print them out together
for col in list(df_obj.columns):
    print( df_obj[col].value_counts() )
    print(' ')
```

```
No     1233
Yes     237
Name: Attrition, dtype: int64

Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64

Research & Development    961
Sales                     446
Human Resources            63
Name: Department, dtype: int64

Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64

Male      882
Female    588
Name: Gender, dtype: int64

Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64

Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64

Y    1470
Name: Over18, dtype: int64

No     1054
Yes     416
Name: OverTime, dtype: int64
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Step 3 Exploratory data analysis(EDA)

what are the major factors that make people want to leave this job?

In [148]:
```
df.head()
```

Out[148]:

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

5 rows × 35 columns

In [149]:
```
df.describe().transpose() # to make it easier to look
```

Out[149]:

	count	mean	std	min	25%	50%	75%	max
Age	1470.0	36.923810	9.135373	18.0	30.00	36.0	43.00	60.0
DailyRate	1470.0	802.485714	403.509100	102.0	465.00	802.0	1157.00	1499.0
DistanceFromHome	1470.0	9.192517	8.106864	1.0	2.00	7.0	14.00	29.0
Education	1470.0	2.912925	1.024165	1.0	2.00	3.0	4.00	5.0
EmployeeCount	1470.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1.0
EmployeeNumber	1470.0	1024.865306	602.024335	1.0	491.25	1020.5	1555.75	2068.0
EnvironmentSatisfaction	1470.0	2.721769	1.093082	1.0	2.00	3.0	4.00	4.0
HourlyRate	1470.0	65.891156	20.329428	30.0	48.00	66.0	83.75	100.0
JobInvolvement	1470.0	2.729932	0.711561	1.0	2.00	3.0	3.00	4.0
JobLevel	1470.0	2.063946	1.106940	1.0	1.00	2.0	3.00	5.0
JobSatisfaction	1470.0	2.728571	1.102846	1.0	2.00	3.0	4.00	4.0
MonthlyIncome	1470.0	6502.931293	4707.956783	1009.0	2911.00	4919.0	8379.00	19999.0
MonthlyRate	1470.0	14313.103401	7117.786044	2094.0	8047.00	14235.5	20461.50	26999.0
NumCompaniesWorked	1470.0	2.693197	2.498009	0.0	1.00	2.0	4.00	9.0
PercentSalaryHike	1470.0	15.209524	3.659938	11.0	12.00	14.0	18.00	25.0
PerformanceRating	1470.0	3.153741	0.360824	3.0	3.00	3.0	3.00	4.0
RelationshipSatisfaction	1470.0	2.712245	1.081209	1.0	2.00	3.0	4.00	4.0
StandardHours	1470.0	80.000000	0.000000	80.0	80.00	80.0	80.00	80.0
StockOptionLevel	1470.0	0.793878	0.852077	0.0	0.00	1.0	1.00	3.0
TotalWorkingYears	1470.0	11.279592	7.780782	0.0	6.00	10.0	15.00	40.0
TrainingTimesLastYear	1470.0	2.799320	1.289271	0.0	2.00	3.0	3.00	6.0
WorkLifeBalance	1470.0	2.761224	0.706476	1.0	2.00	3.0	3.00	4.0
YearsAtCompany	1470.0	7.008163	6.126525	0.0	3.00	5.0	9.00	40.0
YearsInCurrentRole	1470.0	4.229252	3.623137	0.0	2.00	3.0	7.00	18.0
YearsSinceLastPromotion	1470.0	2.187755	3.222430	0.0	0.00	1.0	3.00	15.0
YearsWithCurrManager	1470.0	4.123129	3.568136	0.0	2.00	3.0	7.00	17.0

In [150]:
```
# what are the factors? maybe we look at for those who leave? who stay?
# then, maybe, what is their avg overtime working? (this is features)
df.groupby('Attrition').mean().transpose()
```

Out[150]:

Attrition	No	Yes
Age	37.561233	33.607595
DailyRate	812.504461	750.362869
DistanceFromHome	8.915653	10.632911
Education	2.927007	2.839662
EmployeeCount	1.000000	1.000000
EmployeeNumber	1027.656123	1010.345992
EnvironmentSatisfaction	2.771290	2.464135
HourlyRate	65.952149	65.573840
JobInvolvement	2.770479	2.518987
JobLevel	2.145985	1.637131
JobSatisfaction	2.778589	2.468354
MonthlyIncome	6832.739659	4787.092827
MonthlyRate	14265.779400	14559.308017
NumCompaniesWorked	2.645580	2.940928
PercentSalaryHike	15.231144	15.097046
PerformanceRating	3.153285	3.156118
RelationshipSatisfaction	2.733982	2.599156
StandardHours	80.000000	80.000000
StockOptionLevel	0.845093	0.527426
TotalWorkingYears	11.862936	8.244726
TrainingTimesLastYear	2.832928	2.624473
WorkLifeBalance	2.781022	2.658228
YearsAtCompany	7.369019	5.130802
YearsInCurrentRole	4.484185	2.902954
YearsSinceLastPromotion	2.234388	1.945148
YearsWithCurrManager	4.367397	2.852321

In [151]:

```python
# maybe for the first row, we can know the % of leace and stay on the job in diff age of empl
df_analysis = df.groupby('Attrition').mean().transpose()
df_analysis['abs_%dif'] = abs( df_analysis['No']-df_analysis['Yes'] ) / df_analysis['Yes']
df_analysis
```

Out[151]:

Attrition	No	Yes	abs_%dif
Age	37.561233	33.607595	0.117641
DailyRate	812.504461	750.362869	0.082815
DistanceFromHome	8.915653	10.632911	0.161504
Education	2.927007	2.839662	0.030759
EmployeeCount	1.000000	1.000000	0.000000
EmployeeNumber	1027.656123	1010.345992	0.017133
EnvironmentSatisfaction	2.771290	2.464135	0.124650
HourlyRate	65.952149	65.573840	0.005769
JobInvolvement	2.770479	2.518987	0.099838
JobLevel	2.145985	1.637131	0.310821
JobSatisfaction	2.778589	2.468354	0.125685
MonthlyIncome	6832.739659	4787.092827	0.427325
MonthlyRate	14265.779400	14559.308017	0.020161
NumCompaniesWorked	2.645580	2.940928	0.100427
PercentSalaryHike	15.231144	15.097046	0.008882
PerformanceRating	3.153285	3.156118	0.000898
RelationshipSatisfaction	2.733982	2.599156	0.051873
StandardHours	80.000000	80.000000	0.000000
StockOptionLevel	0.845093	0.527426	0.602297
TotalWorkingYears	11.862936	8.244726	0.438851
TrainingTimesLastYear	2.832928	2.624473	0.079427
WorkLifeBalance	2.781022	2.658228	0.046194
YearsAtCompany	7.369019	5.130802	0.436231
YearsInCurrentRole	4.484185	2.902954	0.544697
YearsSinceLastPromotion	2.234388	1.945148	0.148698
YearsWithCurrManager	4.367397	2.852321	0.531173

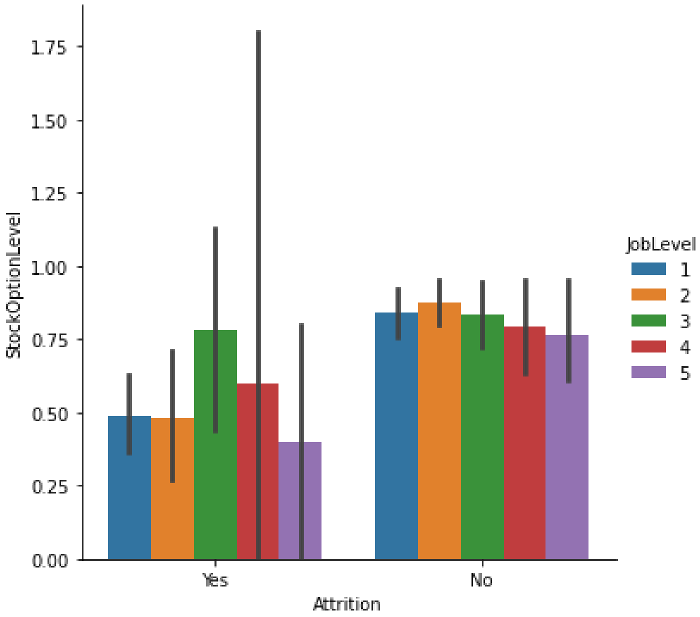
In [152]:
```
# then, we want to rank it
df_analysis.sort_values('abs_%dif',ascending=False) #descending order
```

Out[152]:

Attrition	No	Yes	abs_%dif
StockOptionLevel	0.845093	0.527426	0.602297
YearsInCurrentRole	4.484185	2.902954	0.544697
YearsWithCurrManager	4.367397	2.852321	0.531173
TotalWorkingYears	11.862936	8.244726	0.438851
YearsAtCompany	7.369019	5.130802	0.436231
MonthlyIncome	6832.739659	4787.092827	0.427325
JobLevel	2.145985	1.637131	0.310821
DistanceFromHome	8.915653	10.632911	0.161504
YearsSinceLastPromotion	2.234388	1.945148	0.148698
JobSatisfaction	2.778589	2.468354	0.125685
EnvironmentSatisfaction	2.771290	2.464135	0.124650
Age	37.561233	33.607595	0.117641
NumCompaniesWorked	2.645580	2.940928	0.100427
JobInvolvement	2.770479	2.518987	0.099838
DailyRate	812.504461	750.362869	0.082815
TrainingTimesLastYear	2.832928	2.624473	0.079427
RelationshipSatisfaction	2.733982	2.599156	0.051873
WorkLifeBalance	2.781022	2.658228	0.046194
Education	2.927007	2.839662	0.030759
MonthlyRate	14265.779400	14559.308017	0.020161
EmployeeNumber	1027.656123	1010.345992	0.017133
PercentSalaryHike	15.231144	15.097046	0.008882
HourlyRate	65.952149	65.573840	0.005769
PerformanceRating	3.153285	3.156118	0.000898
StandardHours	80.000000	80.000000	0.000000
EmployeeCount	1.000000	1.000000	0.000000

In [153]:
```
# now, we look at some to do analysis
# first, stock option
df['StockOptionLevel']
# want to see who stay who leave, the avg stock option level,
# use categorical plot: catplot # attrition:减员
sns.catplot( data=df, x='Attrition', y='StockOptionLevel', kind='bar', hue='JobLevel')
```

Out[153]:
```
<seaborn.axisgrid.FacetGrid at 0x1f8c60fa610>
```



In [154]:
```
# insight: for those who choose to leave this job (attrition==yes), job level 3 people,
# they have higher stock option level value, but still choose to leave the job
# the business insight is: maybe these job level 3 people have other problems,
# not related to stock option level,since we dont see this phenomenon on other job level
# For other 4 job levels, the lower the stock option level, the higher rate they choose to le
#                          the higher may stay.
# go to look at what other problems ~~~
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

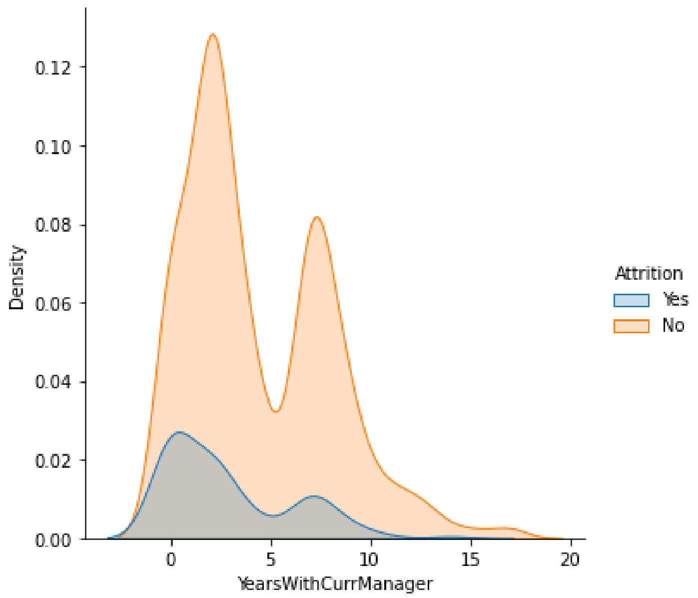
In [155]:
```python
# the other important feature is years with current manager
# we can see the new guys on this position have quite a short period of time,
# they leave this job with higher chance
```

In [156]:
```python
# first, know the dataset well, call out original dataset
df['YearsWithCurrManager']
```

Out[156]:
```
0       5
1       7
2       0
3       0
4       2
       ..
1465    3
1466    7
1467    3
1468    8
1469    2
Name: YearsWithCurrManager, Length: 1470, dtype: int64
```

In [157]:
```python
# for this case, we can either use distribution plot or density plot
# to show the difference in distribution,
# since diff years ~= diff distribution,
# histogram also can show the distribution on years 1 to 10 for leave & stay (hue='attrition'
# here, we use kde plot
sns.displot(data=df, x='YearsWithCurrManager', hue='Attrition', kind='kde', fill=True)
```

Out[157]: <seaborn.axisgrid.FacetGrid at 0x1f8c488d6a0>



In [158]:
```python
# these two distribution are not in the same scale, so there is another way to apply kde plot
```

In [159]:
```
sns.kdeplot( df [df['Attrition']=='Yes']['YearsWithCurrManager'], fill=True)
sns.kdeplot( df [df['Attrition']=='No']['YearsWithCurrManager'], fill=True)
# to put the graph overlay together, let them into same scale
```

Out[159]:
```
<AxesSubplot:xlabel='YearsWithCurrManager', ylabel='Density'>
```

In [160]:
```
# In the plot, we see that who choose to leave (blue area), the distribution is a little left
# that means the years with current manager are those younger, and tends to leave their jobs
# For those who  choose to stay their jobs, the distribution shows the age of managers are
# slightly higher than those who choose to leave.
# business insight: blue one, not overlapping, means that, for those new joiners of this job,
# there is high risk of churning from this current job.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [161]:
```python
# (third), look at another discrete variable
# Overtime
df['OverTime']
```

Out[161]:
```
0       Yes
1        No
2       Yes
3       Yes
4        No
       ... 
1465     No
1466     No
1467    Yes
1468     No
1469     No
Name: OverTime, Length: 1470, dtype: object
```

In [162]:
```python
# we want to know their attrition rate, so we need to turn yes/no to 1/0
# to calculate avg
df['Attrition_tag']=df.apply(lambda s:
                                    1 if s['Attrition']=='Yes'
                                      else 0,
                            axis=1)
# df
df[['Attrition','Attrition_tag']]
```

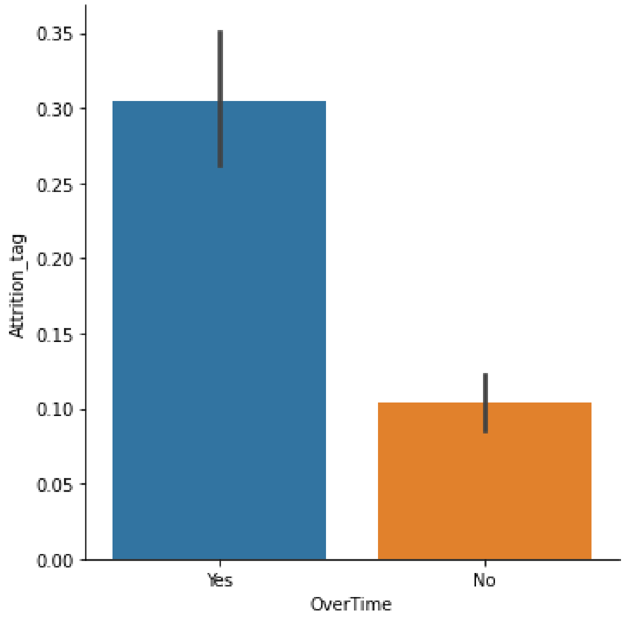
Out[162]:

	Attrition	Attrition_tag
0	Yes	1
1	No	0
2	Yes	1
3	No	0
4	No	0
...	...	...
1465	No	0
1466	No	0
1467	No	0
1468	No	0
1469	No	0

1470 rows × 2 columns

In [163]:
```
sns.catplot(data=df, x='OverTime', y='Attrition_tag', kind='bar')
```

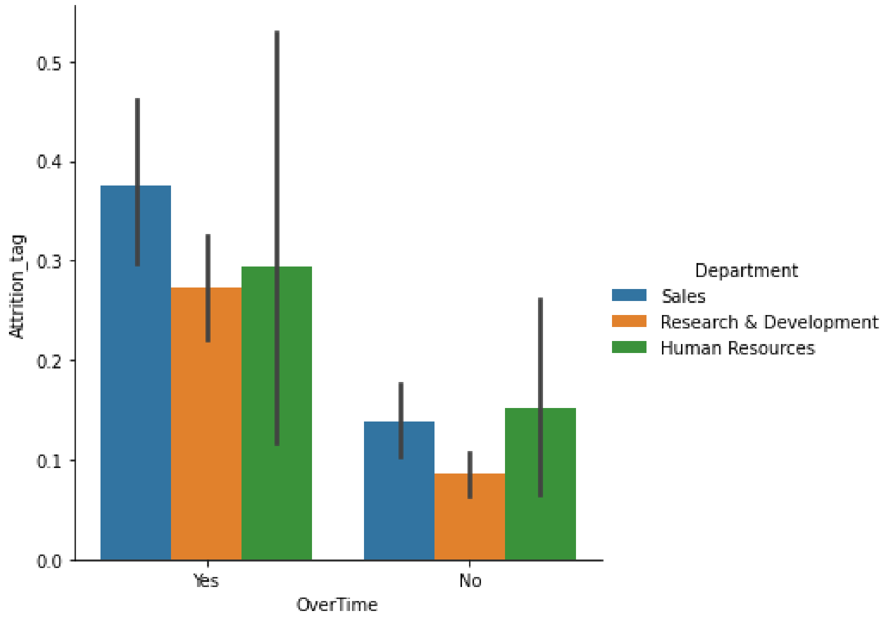
Out[163]: <seaborn.axisgrid.FacetGrid at 0x1f8c6138820>



In [164]:
```
# The plot shows the attrition rate of overtime people is high at 30%,
#                  others non-overtime worker is only about 10%
# business insight: overtime is an obvious cause for them to leave their jobs,
# director may do something to prevent this
```

In [165]:
```python
# then, we may want to see for different departments, is there behavior differently?
sns.catplot(data=df, x='OverTime', y='Attrition_tag', hue='Department', kind='bar')
```

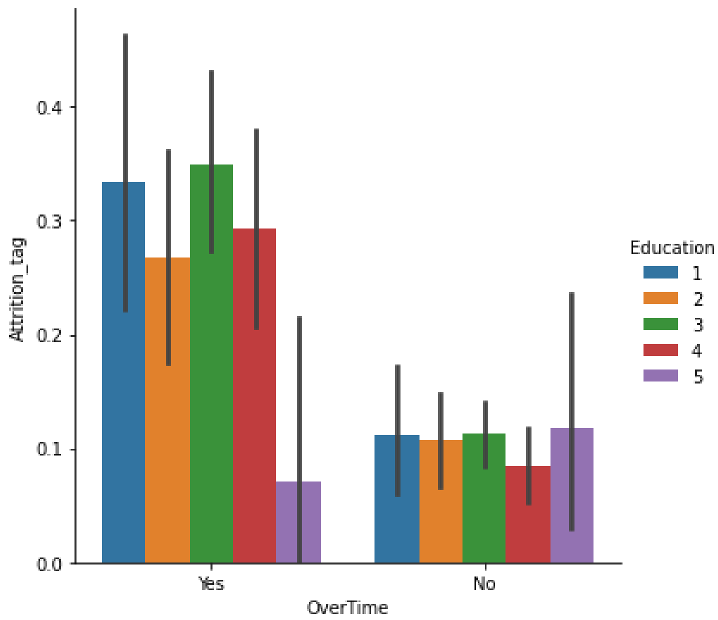
Out[165]: <seaborn.axisgrid.FacetGrid at 0x1f8c63821f0>



In [166]:
```python
# from the plot, we see that sales work overtime most, this makes their attrition rate higher
# For the research & development department, it doesnt seem to be so obvious.
# For the HR department, the attrition rate and non-attrition rate are not as different as ot
# business insight: HR employees leave their jobs,
# the reason behind are less related to working overtime.
```

In [167]:
```
# maybe different edu background, the attrition of overtime rate is ???
sns.catplot(data=df, x='OverTime', y='Attrition_tag', hue='Education', kind='bar')
```

Out[167]:
```
<seaborn.axisgrid.FacetGrid at 0x1f8c64f0b50>
```



In [168]:
```
# how about business travel??

# travel frequently, higher attrition rate in general
# for some roles, business travel are not concern to them, for HR, they seem to have much con
# ... --> to retake the talent in this company
```

In [169]:
```
# Those are the impacts of --> some factors of people's decision of leaving this job
#                               and so we can do this exploratory data analysis.
# Try to use different kinds of visualizations, different features of the graphs,
#                                             and have an understanding of these dataset.
# Generate some insights for company what they can do,
# and how to retain the talent in this company
```

In [ ]:

In [ ]:

In [ ]:

Step 4 Machine Learning Model

1. Do data preprocessing
2. Build ML models
3. Make prediction on test dataset and test the model preformance

In [170]:
```
# now, we already have some ideas on
# 'who are easily churned from this current job on segmentation level'
# e.g. HR people are more likely to leave this job
#      overtime people are more likely to leave this job
# these two are segmentation分割 level
# how predictions do on individual level?
# what is the prob.? what is the chance of leaving this job?
# This is advanced analytics == ML
```

For individual level, build a model to predict the probability of leaving current job

In [171]:
```
# 4.1 do data preprocessing
# since data format can be very messy, not suitable for ML algorithms to directly consume
df.head()
```

Out[171]:

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

5 rows × 36 columns

In [172]:
```
# take an example, why we need data poeprocessing,
# e.g.BusinessTravel, it is a string, which is not numerical values,
# data poeprocessing can make them suitable to ML job
# then there also some desciptions of some cols (some individuals) of dataset,

# For conducting ML,
# we need labels, which is a tag, yes or no, which means whether leave this job or not
# labels must be numerical values if it is a classification problem, (0/1) ,
# it can be just a value if it is a regression

# make sure all values are numerical values
# for business travel, we do as below:
```

In [173]:
```
# generate labels
```

In [174]:
```python
# one hot encoding, to convert all non-numerical features into numerical features
# one hot encoding, it is called pandas.get_dummies
df_test = pd.DataFrame( data=df, columns=['BusinessTravel'] )
df_test.head()
```

Out[174]:

	BusinessTravel
0	Travel_Rarely
1	Travel_Frequently
2	Travel_Rarely
3	Travel_Frequently
4	Travel_Rarely

In [175]:
```python
pd.get_dummies(df_test)
```

Out[175]:

	BusinessTravel_Non-Travel	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely
0	0	0	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
...	...	...	...
1465	0	1	0
1466	0	0	1
1467	0	0	1
1468	0	1	0
1469	0	0	1

1470 rows × 3 columns

In [176]:
```python
# then, we transform other features to matrix form
```

In [177]:
```python
df_modelling = df.drop('Attrition', 1)
df_modelling.head()
```

Out[177]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount
0	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	1
1	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1
2	37	Travel_Rarely	1373	Research & Development	2	2	Other	1
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1
4	27	Travel_Rarely	591	Research & Development	2	1	Medical	1

5 rows × 35 columns

In [178]:
```
df_modelling = pd.get_dummies(df_modelling)
df_modelling.head().transpose()
```

Out[178]:

	0	1	2	3	4
Age	41	49	37	33	27
DailyRate	1102	279	1373	1392	591
DistanceFromHome	1	8	2	3	2
Education	2	1	2	4	1
EmployeeCount	1	1	1	1	1
EmployeeNumber	1	2	4	5	7
EnvironmentSatisfaction	2	3	4	4	1
HourlyRate	94	61	92	56	40
JobInvolvement	3	2	2	3	3
JobLevel	2	2	1	1	1
JobSatisfaction	4	2	3	3	2
MonthlyIncome	5993	5130	2090	2909	3468
MonthlyRate	19479	24907	2396	23159	16632
NumCompaniesWorked	8	1	6	1	9
PercentSalaryHike	11	23	15	11	12
PerformanceRating	3	4	3	3	3
RelationshipSatisfaction	1	4	2	3	4
StandardHours	80	80	80	80	80
StockOptionLevel	0	1	0	0	1
TotalWorkingYears	8	10	7	8	6
TrainingTimesLastYear	0	3	3	3	3
WorkLifeBalance	1	3	3	3	3
YearsAtCompany	6	10	0	8	2
YearsInCurrentRole	4	7	0	7	2
YearsSinceLastPromotion	0	1	0	3	2
YearsWithCurrManager	5	7	0	0	2
Attrition_tag	1	0	1	0	0
BusinessTravel_Non-Travel	0	0	0	0	0
BusinessTravel_Travel_Frequently	0	1	0	1	0
BusinessTravel_Travel_Rarely	1	0	1	0	1
Department_Human Resources	0	0	0	0	0
Department_Research & Development	0	1	1	1	1
Department_Sales	1	0	0	0	0
EducationField_Human Resources	0	0	0	0	0
EducationField_Life Sciences	1	1	0	1	0
EducationField_Marketing	0	0	0	0	0
EducationField_Medical	0	0	0	0	1
EducationField_Other	0	0	1	0	0
EducationField_Technical Degree	0	0	0	0	0
Gender_Female	1	0	0	1	0
Gender_Male	0	1	1	0	1
JobRole_Healthcare Representative	0	0	0	0	0

	0	1	2	3	4
JobRole_Human Resources	0	0	0	0	0
JobRole_Laboratory Technician	0	0	1	0	1
JobRole_Manager	0	0	0	0	0
JobRole_Manufacturing Director	0	0	0	0	0
JobRole_Research Director	0	0	0	0	0
JobRole_Research Scientist	0	1	0	1	0
JobRole_Sales Executive	1	0	0	0	0
JobRole_Sales Representative	0	0	0	0	0
MaritalStatus_Divorced	0	0	0	0	0
MaritalStatus_Married	0	1	0	1	1
MaritalStatus_Single	1	0	1	0	0
Over18_Y	1	1	1	1	1
OverTime_No	0	1	0	0	1
OverTime_Yes	1	0	1	1	0

In [179]:
```
# do train test split
```

In [180]:
```
df['Age'].count() # 1470
```

Out[180]:
```
1470
```

In [181]:
```
# training set 70%, training machine learning models
# test set 30%, tset the model we have built to evaluate on the performance
```

In [182]:
```
# labels
x = df_modelling.drop('Attrition_tag', 1)
y = df_modelling['Attrition_tag']
```

In [183]:
```
x.head()
```

Out[183]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Ho
0	41	1102	1	2	1	1	2	
1	49	279	8	1	1	2	3	
2	37	1373	2	2	1	4	4	
3	33	1392	3	4	1	5	4	
4	27	591	2	1	1	7	1	

5 rows × 55 columns

In [184]:
```
y.head()
```

Out[184]:
```
0    1
1    0
2    1
3    0
4    0
Name: Attrition_tag, dtype: int64
```

In [185]:
```python
from sklearn.model_selection import train_test_split

# train_test_split is to split one data frame to two data frame
#                                                 according to a predefined ratio
```

In [186]:
```python
X_train, X_test, y_train, y_test = train_test_split( x, y, test_size=0.3, train_size=0.7 )
```

In [187]:
```python
X_train
```

Out[187]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction
49	35	1229	8	1	1	63	4
1101	32	824	5	2	1	1555	4
128	22	594	2	1	1	169	3
997	27	135	17	4	1	1405	4
789	44	1376	1	2	1	1098	2
...	...	...	...	...	...	...	...
1006	49	1475	28	2	1	1420	1
590	33	213	7	3	1	817	3
286	44	920	24	3	1	392	4
1024	47	359	2	4	1	1443	1
175	56	713	8	3	1	241	3

1029 rows × 55 columns

In [188]:
```python
X_test
```

Out[188]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction
544	47	217	3	3	1	746	4
417	40	1398	2	4	1	558	3
983	34	404	2	4	1	1383	3
65	55	836	8	3	1	84	4
575	54	376	19	4	1	799	4
...	...	...	...	...	...	...	...
1140	44	1313	7	3	1	1608	2
444	48	163	2	5	1	595	2
356	42	1332	2	4	1	477	1
1414	47	1180	25	3	1	1993	1
547	42	933	19	3	1	752	3

441 rows × 55 columns

In [189]:
```python
y_train
```
Out[189]:
```
49      0
1101    0
128     0
997     1
789     1
       ..
1006    1
590     0
286     1
1024    0
175     0
Name: Attrition_tag, Length: 1029, dtype: int64
```
In [190]:
```python
y_test
```
Out[190]:
```
544     0
417     0
983     0
65      0
575     0
       ..
1140    0
444     0
356     0
1414    0
547     1
Name: Attrition_tag, Length: 441, dtype: int64
```
In [191]:
```python
X_train['Age'].count()
```
Out[191]:
```
1029
```
In [192]:
```python
X_test['Age'].count()
```
Out[192]:
```
441
```
In [193]:
```python
# total (original) = x_train['Age'].count() + x_test['Age'].count()
# 1029 + 441
```
In [194]:
```python
# almost done -- data preprocessing
# x_train, x_test, y_train, y_test = train_test_split( x, y, test_size=0.3, train_size=0.7 )
# is a data frame that we prepare for building model
```
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [195]:
```python
## Build machine learning models
```

In [196]:
```python
# here we use random forest to build a ML model that predict the employees' attrition
```

In [197]:
```python
from sklearn.ensemble import RandomForestClassifier
```

In [198]:
```python
# defining the model to be used for training,
# *** max_depth= 3 or 5 for basic work, tesing ang changing values after
clf = RandomForestClassifier(max_depth=3)
```

In [199]:
```python
# input the training features and labels to train the model
clf.fit(X_train, y_train)  # two parameters for this fit
```

Out[199]:
```
RandomForestClassifier(max_depth=3)
```

In [200]:
```python
type(clf)  # clf is a model, but we cannot see what is inside now, wt info. inside
           #                  there is some complicated binary info. inside
```

Out[200]:
```
sklearn.ensemble._forest.RandomForestClassifier
```

In [ ]:
```python

```

In [201]:
```python
## Make prediction on test dataset and test the model preformance
```

In [202]:
```python
# model is ready
# we have X_test and y_test, but we ASSUME we only know the X_test
# then use it to predict the attrition of all these users,
#      compare with the y_test (the true label)
#      compare the accuracy and even auc to get uderstanding how accurate our model is
#                                              (the purpose of having this test set y_test
#              --> to test the model performance assume that we dont know the exact label)
```

In [203]:
```python
# model is ready & compare with the y_test (the true label)
# do predictions, clf we have
y_predict = clf.predict(X_test)
```

In [204]:
```python
y_predict
```

Out[204]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0], dtype=int64)
```

In [205]:
```python
# compare the y_predict, which is our prediction with the y_test,
# to evaluate on the accuracy of the modelling
# accuracy, confusion matrix, AUC
from sklearn.metrics import accuracy_score, precision_score, classification_report, confusion
```

In [206]:
```python
# accuracy
accuracy_score(y_test, y_predict)
```

Out[206]: 0.8367346938775511

In [207]:
```python
# confusion matrix
confusion_matrix(y_test, y_predict)
```

Out[207]:
```
array([[367,   0],
       [ 72,   2]], dtype=int64)
```

In [208]:
```python
# precision score
precision_score(y_test, y_predict, average='macro')
```

Out[208]: 0.917995444191344

In [209]:
```python
# AUC, 1 means perfect classification; 0.5 means random guess
```

In [210]:
```python
# first, go back to predict the prob.
y_prob = clf.predict_proba(X_test)
```

In [211]:
```python
y_prob  # y_prob shows the prob. being 0 or 1 while inputting to this roc curve,
        # we only need the prob. being 1, i.e.2nd col
```

Out[211]:
```
array([[0.86595354, 0.13404646],
       [0.90867342, 0.09132658],
       [0.89278317, 0.10721683],
       [0.86359322, 0.13640678],
       [0.86716493, 0.13283507],
       [0.8594494 , 0.1405506 ],
       [0.81388544, 0.18611456],
       [0.88451313, 0.11548687],
       [0.87814402, 0.12185598],
       [0.91907687, 0.08092313],
       [0.86018752, 0.13981248],
       [0.8410164 , 0.1589836 ],
       [0.91133067, 0.08866933],
       [0.90932658, 0.09067342],
       [0.89570447, 0.10429553],
       [0.91385646, 0.08614354],
       [0.80714651, 0.19285349],
       [0.84755643, 0.15244357],
       [0.90417799, 0.09582201],
```

In [212]:
```python
pred = y_prob[:,1]
pred
```

Out[212]:
```
array([0.13404646, 0.09132658, 0.10721683, 0.13640678, 0.13283507,
       0.1405506 , 0.18611456, 0.11548687, 0.12185598, 0.08092313,
       0.13981248, 0.1589836 , 0.08866933, 0.09067342, 0.10429553,
       0.08614354, 0.19285349, 0.15244357, 0.09582201, 0.17575492,
       0.13840078, 0.11190292, 0.17099475, 0.08772134, 0.0955093 ,
       0.12638733, 0.07083394, 0.07091455, 0.12215509, 0.24475597,
       0.13458955, 0.1259237 , 0.1055176 , 0.13054521, 0.257011  ,
       0.14344143, 0.19215894, 0.13942675, 0.15411547, 0.11071386,
       0.39688676, 0.06964711, 0.10959995, 0.07999001, 0.18619085,
       0.18230276, 0.06788407, 0.09167581, 0.08381336, 0.12317379,
       0.56217472, 0.15503336, 0.09753609, 0.40859939, 0.18466966,
       0.0686242 , 0.22064849, 0.26738714, 0.0722754 , 0.1187768 ,
       0.13527318, 0.14323223, 0.09563546, 0.14179391, 0.10503769,
       0.07566074, 0.17047731, 0.1223376 , 0.08070908, 0.15610384,
       0.15318554, 0.12987075, 0.112972  , 0.17164538, 0.14538373,
       0.13066199, 0.14326277, 0.09727115, 0.07990772, 0.16315019,
       0.1421768 , 0.18993914, 0.21497914, 0.08924901, 0.19198928,
       0.35611654, 0.1498767 , 0.13355691, 0.11735603, 0.15877578,
       0.1707702 , 0.09265355, 0.28960493, 0.07736208, 0.15993808,
       0.09352607, 0.08149653, 0.16550951, 0.16050289, 0.1806471 ,
       0.1808781 , 0.14379233, 0.27062536, 0.08386169, 0.2169655 ,
       0.14850449, 0.12028371, 0.08437629, 0.18579768, 0.10700483,
       0.14111707, 0.11137296, 0.09711177, 0.10364632, 0.09715243,
       0.30622393, 0.11397692, 0.11807778, 0.12639309, 0.09112674,
       0.14490383, 0.12666273, 0.09886894, 0.07567675, 0.11299404,
       0.08017902, 0.20288583, 0.24538205, 0.28932093, 0.18048066,
       0.07635398, 0.12995858, 0.20759646, 0.1055684 , 0.13269116,
       0.08494698, 0.196186  , 0.07689838, 0.09378883, 0.15132062,
       0.09270252, 0.10300146, 0.1222346 , 0.29949254, 0.08049744,
       0.19193518, 0.20875147, 0.10858373, 0.07941134, 0.12635382,
       0.11556494, 0.10380938, 0.08833151, 0.12762049, 0.12024225,
       0.22677276, 0.13093587, 0.2182216 , 0.13299469, 0.19450396,
       0.15751772, 0.24975914, 0.09350859, 0.14087251, 0.45113845,
       0.07784683, 0.09489807, 0.23853152, 0.32595872, 0.13158433,
       0.14497212, 0.10261452, 0.11986346, 0.1206024 , 0.11609361,
       0.14075758, 0.17541613, 0.14613178, 0.27846923, 0.1608504 ,
       0.10496046, 0.29118147, 0.15970054, 0.13371897, 0.10842305,
       0.10538615, 0.13927452, 0.14540123, 0.12812125, 0.1214549 ,
       0.12482455, 0.1404665 , 0.15158903, 0.28575636, 0.15690796,
       0.16509329, 0.12698645, 0.51322286, 0.10942556, 0.2145561 ,
       0.17198397, 0.20565768, 0.10151983, 0.17998034, 0.15537771,
       0.13869286, 0.09057686, 0.15360819, 0.20289046, 0.0856944 ,
       0.24257048, 0.37439771, 0.17143311, 0.08240617, 0.12619773,
       0.14795624, 0.19926192, 0.11051939, 0.15994312, 0.1371387 ,
       0.10573464, 0.10623851, 0.20738884, 0.12235309, 0.13639345,
       0.14083125, 0.14218066, 0.0811897 , 0.11512429, 0.087446  ,
       0.15089841, 0.26531542, 0.1340616 , 0.09156793, 0.2521624 ,
       0.18043154, 0.13350265, 0.13238648, 0.09659061, 0.10207815,
       0.10478297, 0.06673302, 0.14465441, 0.08248563, 0.08862069,
       0.16073202, 0.08350786, 0.1254342 , 0.16203856, 0.14354761,
       0.14592195, 0.19069554, 0.35236808, 0.1233117 , 0.11334863,
       0.07793665, 0.11759352, 0.10831299, 0.24679736, 0.21781198,
       0.17527191, 0.21839574, 0.18163679, 0.10013095, 0.10871124,
       0.11009323, 0.18303655, 0.1227979 , 0.07102975, 0.09879772,
       0.15567671, 0.14445193, 0.39186   , 0.15488906, 0.14447228,
       0.26236425, 0.11366167, 0.13593484, 0.23262209, 0.1045159 ,
       0.1262153 , 0.19996409, 0.16450969, 0.19285749, 0.17863237,
       0.08124579, 0.08143916, 0.12340556, 0.14631338, 0.09384868,
       0.13517713, 0.11857408, 0.10314382, 0.11462084, 0.08486531,
       0.14409866, 0.1598626 , 0.09640259, 0.0834663 , 0.18966979,
       0.24955541, 0.09374822, 0.19526749, 0.13861104, 0.30900737,
       0.18229232, 0.12566153, 0.15131443, 0.22391754, 0.12566924,
       0.15931076, 0.0732077 , 0.15819668, 0.1124318 , 0.13388231,
       0.08235557, 0.09581455, 0.1032827 , 0.10970459, 0.08985019,
       0.13752961, 0.3479358 , 0.24588248, 0.12609023, 0.15099828,
       0.17473951, 0.12439001, 0.12077683, 0.1527839 , 0.12792666,
       0.22372067, 0.29213145, 0.08648207, 0.13497147, 0.09212118,
```

```
       0.11153698, 0.098285  , 0.15165322, 0.45298158, 0.09970721,
       0.28518766, 0.09351676, 0.14668822, 0.07783329, 0.07379591,
       0.08541906, 0.12262829, 0.4205057 , 0.10947702, 0.10896189,
       0.09498043, 0.11352505, 0.14883984, 0.09420288, 0.14211942,
       0.08670099, 0.12250387, 0.09135524, 0.09937022, 0.11058624,
       0.12835555, 0.30616461, 0.21313432, 0.17685591, 0.31646052,
       0.13360609, 0.1034013 , 0.12520366, 0.20344412, 0.19791677,
       0.17545459, 0.09530972, 0.07434562, 0.27881733, 0.076851  ,
       0.30645641, 0.21246332, 0.1002544 , 0.1054672 , 0.09119428,
       0.06914465, 0.08767093, 0.11680467, 0.31584749, 0.17406027,
       0.2150377 , 0.1454668 , 0.10974006, 0.16088429, 0.22970713,
       0.31418236, 0.08551841, 0.29513905, 0.12863803, 0.07928706,
       0.26525298, 0.15267653, 0.08721603, 0.18553272, 0.09442895,
       0.24284111, 0.16819069, 0.36410807, 0.27812695, 0.20150458,
       0.13878706, 0.26039074, 0.16769068, 0.11026287, 0.14601669,
       0.09397093, 0.35499724, 0.08953765, 0.18315224, 0.10152202,
       0.10265868, 0.10949737, 0.12097004, 0.09496414, 0.07310614,
       0.20748964, 0.0736583 , 0.14775162, 0.0922919 , 0.08695814,
       0.16546813, 0.08835886, 0.17145428, 0.15322493, 0.11064691,
       0.22725014, 0.30047255, 0.13109199, 0.09607178, 0.11836993,
       0.10774611, 0.12881914, 0.09010179, 0.19172741, 0.12229373,
       0.2587578 ])
```

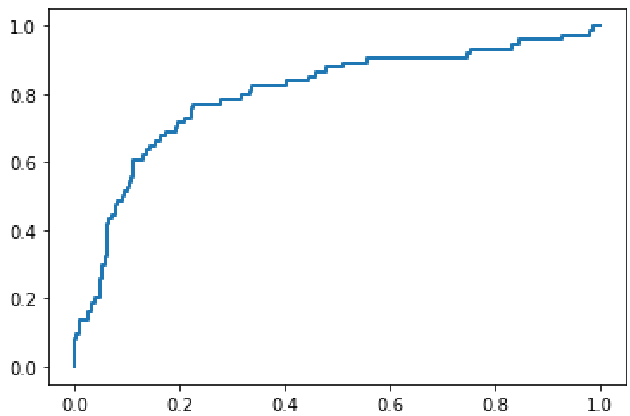
In [213]:
```
fpr, tpr, thresholds = roc_curve(y_test, pred)
auc(fpr, tpr)
```

Out[213]:
```
0.8006112379409382
```

In [214]:
```
# 0.78 > 0.5, it is random guess, there is a room to increase it.
```

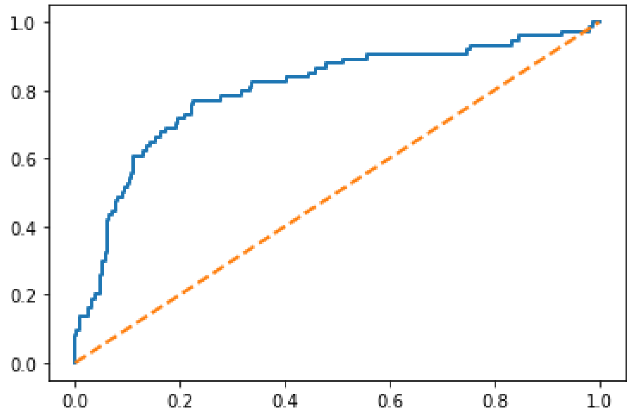
In [215]:
```
# try to plot auc curve

# we assign the auc(fpr, tpr) to a value called roc_auc lrl
import matplotlib.pyplot as plt
plt.plot(fpr, tpr, lw=2)
```

Out[215]:
```
[<matplotlib.lines.Line2D at 0x1f8c75a0a00>]
```



In [216]:
```
# area under this curve is 1 means it is a prefect segmentation
plt.plot(fpr, tpr, lw=2)
plt.plot([0,1], [0,1], lw=2, linestyle='--')  # this is roc_auc
```

Out[216]: [<matplotlib.lines.Line2D at 0x1f8c75f6e80>]



In [217]:
```
# this is a gd model because it is able to do some predictions
# and it performs much better than random guess
# we can improve it much more better
# look at auc score (b/c optimize auc score = improve the predictions)
# we can go back to 'building machine learning model' part,
#                                                      try another max_depth no. e.g. 8
# do this training again and run
```

In [ ]:

In [218]:
```
# we can save the model into a pico file
import pickle
pickle.dump(clf, open('model_att', 'wb'))
```

In [219]:
```
# next time, we can just load the model from files
loaded_model = pickle.load(open('model_att', 'rb'))
loaded_model
```

Out[219]: RandomForestClassifier(max_depth=3)

In [221]:
```python
# e.g.
loaded_model.predict_proba(X_test)
```

Out[221]:
```
array([[0.86595354, 0.13404646],
       [0.90867342, 0.09132658],
       [0.89278317, 0.10721683],
       [0.86359322, 0.13640678],
       [0.86716493, 0.13283507],
       [0.8594494 , 0.1405506 ],
       [0.81388544, 0.18611456],
       [0.88451313, 0.11548687],
       [0.87814402, 0.12185598],
       [0.91907687, 0.08092313],
       [0.86018752, 0.13981248],
       [0.8410164 , 0.1589836 ],
       [0.91133067, 0.08866933],
       [0.90932658, 0.09067342],
       [0.89570447, 0.10429553],
       [0.91385646, 0.08614354],
       [0.80714651, 0.19285349],
       [0.84755643, 0.15244357],
       [0.90417799, 0.09582201],
```

In [ ]:
```python

```

In [ ]:
```python

```