In [1]:
```
# Import packages
```

In [2]:
```
# Read the data

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


file_path="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh IT\\
visa_df=pd.read_csv(file_path)
visa_df
```

Out[2]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

In Machine learning algorithms will develop models by using maths
Maths allows only numbers
So it is very important , you need to pass numerical data only
So we need to convert categorical data to numerical data
For that we have encoding methods
Encoding
- Lable encoder
  - map method
  - np.where
  - LableEncoder package from sklearn
- One hot encoder
  - pd.get_dummies()

������h���� − 1 : ������

Read any categorical column: case_status
Check how many unique lables are there
Create a dictionary with those unique lables as keys by providing a number as values

In [5]:

```
############ read the data
################################
file_path="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh IT\\
visa_df=pd.read_csv(file_path)
visa_df

################################Map#######
```

```
#########################
visa_df['case_status'].unique() # 2
dict1={'Certified':0,'Denied':1}
visa_df['case_status'].map(dict1)

# do you want overwrite existed column
# do you want create a new column
```

Out[5]:

```
0 1
1 0
2 1
3 1
4 0
  ..
25475 0
25476 0
25477 0
25478 0
25479 0
Name: case_status, Length: 25480, dtype: int64
```

Create a new column

In [7]:

```
############ read the data
################################
file_path="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh IT\\
visa_df=pd.read_csv(file_path)
visa_df
```

```
################################Map#######
#########################
visa_df['case_status'].unique() # 2
dict1={'Certified':0,'Denied':1}
visa_df['case_status_num']=visa_df['case_status'].map(dict1)
```



```
visa_df
```

In [8]:

Out[8]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	...
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 13 columns

drop case_status_num

In [9]:
```
tatus_num',  axis=1,
 inplace=True)
```

In [10]:
```
visa_df
```

```
visa_df.drop('case_s
```

Out[10]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	...
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

Overwrite on same column(preferable)

In [11]:
```
############# read the data
################################
file_path="C:\\Users\\omkar\\OneDrive\\Doc
uments\\Data science\\Naresh IT\\
```

In [12]:
```
visa_df=pd.read_csv(file_path)
visa_df
```

Out[12]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	...
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

In [14]:
```
visa_df['continent'].uni
que()
```

In [19]:
```
################################Map########
#########################
visa_df['case_status'].unique() # 2
dict1={'Certified':0,'Denied':1}
visa_df['case_status']=visa_df['case_statu
s'].map(dict1)

# in the map method inplace=True is not
there

visa_df
file_path="C:\\Users\\omkar\\OneDrive\\Doc
uments\\Data science\\Naresh IT\\
visa_df=pd.read_csv(file_path)
visa_df

{'Certified':0,'Denied':
1}
{'Asia':0,'Africa':1,'No
rth America':2,
'Europe':3,'South
America':4,'Oceania'}
visa_df['continent'
].unique()
```

Out[14]:
```
array(['Asia', 'Africa', 'North America', 'Europe', 'South America',
       'Oceania'], dtype=object)
```

Out[19]:
```
array(['Asia', 'Africa', 'North America', 'Europe', 'South America',
       'Oceania'], dtype=object)
```

```
                        dict1
```

In [24]:

```
lables=visa_df['contine
nt'].unique()
num=len(visa_df['contin
ent'].unique())
dict1={}
for i in range(num):
 dict1[lables[i]]=i
```

```
lables=visa_df['contine
nt'].unique()
num=len(visa_df['contin
ent'].unique())
{lables[i]:i for i in
range(num)}
```

Out[24]:

```
{'Asia': 0,
 'Africa': 1,
 'North America': 2,
 'Europe': 3,
 'South America': 4,
 'Oceania': 5}
```

������ℎ����– 2 : ����. ��ℎ�������()

In [25]:

```
################ Read the data
again###################
```

```
file_path="C:\\Users\\omkar\\OneDrive\\Doc
uments\\Data science\\Naresh IT\\
visa_df=pd.read_csv(file_path)
visa_df
```

Out[25]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	...
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

np.where() is applicable for binary conditions
which means it is applicable only for two lables
np.where(condition,Truevalue,False_value)

For example case_status has two lables
condition: =='Certified'
True value: Replace all certified values with 0
False value: Replace all denied values with 1

In [27]:
```
con=visa_df['case_status
']=='Certified'
```

In [29]:
```
visa_df['case_status']=n
p.where(con,0,1)

visa_df.head()
```

Out[29]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_
0	EZYV01	Asia	High School	N	N	
1	EZYV02	Asia	Master's	Y	N	
2	EZYV03	Asia	Bachelor's	N	Y	
3	EZYV04	Asia	Bachelor's	N	N	
4	EZYV05	Africa	Master's	Y	N	

LableEncoder is a method from sklearn
Under sklearn we have sub modules
One of the submodule: preprocessing
Any sklearn packages we have only 3 steps
Step-1: Read the package
Step-2: Save the package
Step-3: Apply fit transfotm

In [40]: In [43]:

```
########################## Read the data again ######################

file_path="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh IT\\
visa_df=pd.read_csv(file_path)
print(visa_df[['continent','case_status']].head(10))

############################## LableEncoder################### # step-1
from sklearn.preprocessing import LabelEncoder
# step-2
le=LabelEncoder()
#step=3:
visa_df['case_status']=le.fit_transform(visa_df['case_status'])
visa_df['continent']=le.fit_transform(visa_df['continent'])
print(visa_df[['continent','case_status']].head(10))
```

```
 continent case_status
0 Asia Denied
1 Asia Certified
2 Asia Denied
3 Asia Denied
4 Africa Certified
5 Asia Certified
```

```
6 Asia Certified
7 North America Denied
8 Asia Certified
9 Europe Certified
 continent case_status
0 1 1
1 1 0
2 1 1
3 1 1
4 0 0
5 1 0
6 1 0
7 3 1
8 1 0
9 2 0
```

```
print(visa_df['continent'][:5])
le.inverse_transform(visa_df['continent'])
```

```
0 1
1 1
2 1
3 1
4 0
Name: continent, dtype: int32
```

Out[43]: array(['Asia', 'Asia', 'Asia', ..., 'Asia', 'Asia', 'Asia'], dtype=object)

fit and transform two different definations

age: 1 2 3 4 5

new age : by adding each observation with mean value: x+mean

new age: ========== >
Transform 1+3 2+3 3+3 4+3 5+3

```
import random
random.randint(10,20)
```

In [30]:
mean= 1+2+3+4+5/5=3 ========== > fit

Out[30]: 12

```
import randint
randint(10,20)
```

In [32]:

```
from random
```

Out[32]: 19

In [1]:
```
# Import packages
# Read the data
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

file_path="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh IT\\
visa_df=pd.read_csv(file_path)
visa_df
```

Out[2]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	...
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

One hot encoder

one hot encoder means at a time only one will be ON(1/True), others are OFF(0/False)
Supposer case status has two unique lables
Certified and denied
In one hot encoder it will create two new columns
like below

other

Orthogonality means 90 degress phase shift

90 degress phase shift means perpendicular each other

Perpendicular means independent each other

Independent means no relation

It is very important property that is to avoid relation between varaibes

Draw back:

Assume that you have 100 unique lables for a column

So it will create 100 new columns

To process 100 columns we require more

memory,more time,more hardaware

pd.get_dummies

In [6]:

```
################ Read the data
##############################
file_path="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh IT\\
visa_df=pd.read_csv(file_path)
```

case_status certified denied

certified 1 0

denied 0 1

```
pd.get_dummies(visa_df,
 columns=['case_status'],
 dtype='int')
```

Advantage

One hot encoder new columns are orthogonal each

Out[6]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	...
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 13 columns

In [ ]:

In [ ]:

In [ ]:

In [ ]: