In [3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
path =r'C:\Users\aramaiah.ASUAD\Naresh_IT\MyDataScience\Data_Files\Visadataset.csv
visa_df=pd.read_csv(path)
visa_df.head(6)
```

Out[3]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_of_employ
0	EZYV01	Asia	High School	N	N	14
1	EZYV02	Asia	Master's	Y	N	2
2	EZYV03	Asia	Bachelor's	N	Y	44
3	EZYV04	Asia	Bachelor's	N	N	
4	EZYV05	Africa	Master's	Y	N	
5	EZYV06	Asia	Master's	Y	N	2

- In ML it is very imp to convert categorical data to numerical data
- Machine learnng models aer developed solely by mathematics
- Machine learning takes input in form of numbers only
- To convert we have some encoding techniques
- Label Encoder
    - map
    - np.where
    - using sklearn pacakge: LabelEncoder
- one hot encoder
    - using pandas package:pd.get_dummies

Map

- Before applying map method first get the unique labels of the column
- For example case_status is a cataegorical column
- It has two unique labels there
    - Certified
    - Denied
- Create a dictionary key as label, value as number
- d={'Certified':0,'Denied':1}
- This dictionary we need to map the case_status column

In [16]:

```
visa_df['case_status'].unique()
```

Out[16]:

```
array(['Denied', 'Certified'], dtype=object)
```

In [17]:
```
d={'Denied':1,'Certified':0}
visa_df['case_status']=visa_df['case_status'].map(d)
```

In [18]:
```
visa_df
```

Out[18]:

e	has_job_experience	requires_job_training	no_of_employees	yr_of_estab	region_of_employment	preva
l	N	N	14513	2007	West	
s	Y	N	2412	2002	Northeast	
s	N	Y	44444	2008	West	1
s	N	N	98	1897	West	
s	Y	N	1082	2005	South	1
.	...	...	...	...	...	
s	Y	Y	2601	2008	South	
l	Y	N	3274	2006	Northeast	2
s	Y	N	1121	1910	South	1
s	Y	Y	1918	1887	West	
s	Y	N	3195	1960	Midwest	

In [19]:
```
visa_df['continent'].unique()
```

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

In [10]:
```
c={'Asia':0,'Africa':1,'North America':2,'Europe':3,'South America':4,'Oceania':5}
visa_df['continent']=visa_df['continent'].map(c)
```

In [11]:
```
visa_df
```

Out[11]:

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

25480 rows × 12 columns

In [22]:
```
h={}
labels=visa_df['continent'].unique()
for i in range(len(labels)):
    h[labels[i]]=i
visa_df['continent']=visa_df['continent'].map(h)
visa_df
```

Out[22]:

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

25480 rows × 12 columns

In [23]:
```
# read the data
path =r'C:\Users\aramaiah.ASUAD\Naresh_IT\MyDataScience\Data_Files\Visadataset.csv
visa_df=pd.read_csv(path)
cat_cols=visa_df.select_dtypes(include='object').columns
cat_cols
```

Out[23]:
```
Index(['case_id', 'continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training', 'region_of_employment', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

In [24]:
```
cat_cols=visa_df.select_dtypes(include='object').columns
d={}
for j in cat_cols[1:]: #j=column
    labels=visa_df[j].unique()
    for i in range(len(labels)): #i=number
        d[labels[i]]=i
    visa_df[j]=visa_df[j].map(d)
visa_df
```

Out[24]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_o
0	EZYV01	0	0	0	0	
1	EZYV02	0	1	1	0	
2	EZYV03	0	2	0	1	
3	EZYV04	0	2	0	0	
4	EZYV05	1	1	1	0	
...	...	...	...	...	...	
25475	EZYV25476	0	2	1	1	
25476	EZYV25477	0	0	1	0	
25477	EZYV25478	0	1	1	0	
25478	EZYV25479	0	1	1	1	
25479	EZYV25480	0	2	1	0	

25480 rows × 12 columns

In [ ]:
```
# we always drop the id columns
# Id colmns never provide any infromation
```

Label Encoder

- Label Ecoder is the package available in sklearn
- scikit learn is the heart of ML
- READ THE PACKAGE
- SAVE THE PACKAGE
- APPLY FIT TRANSFORM

In [25]:

```
# Read the data again
path =r'C:\Users\aramaiah.ASUAD\Naresh_IT\MyDataScience\Data_Files\Visadataset.csv
visa_df=pd.read_csv(path)
```

In [27]:

```
from sklearn.preprocessing import LabelEncoder #read the pacakge
le=LabelEncoder() #save the package
visa_df['case_status']=le.fit_transform(visa_df['case_status']) #apply the fit tra
visa_df
```

Out[27]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_o
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	
...	...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	Y	
25476	EZYV25477	Asia	High School	Y	N	
25477	EZYV25478	Asia	Master's	Y	N	
25478	EZYV25479	Asia	Master's	Y	Y	
25479	EZYV25480	Asia	Bachelor's	Y	N	

25480 rows × 12 columns

In [28]:
```
from sklearn.preprocessing import LabelEncoder #read the pacakge
le=LabelEncoder() #save the package
for i in cat_cols:
    visa_df[i]=le.fit_transform(visa_df[i]) #apply the fit transform
visa_df
```

Out[28]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_of_er
0	0	1	2	0	0	
1	1	1	3	1	0	
2	2	1	0	0	1	
3	3	1	0	0	0	
4	4	0	3	1	0	
...	...	...	...	...	...	
25475	17204	1	0	1	1	
25476	17205	1	2	1	0	
25477	17206	1	3	1	0	
25478	17207	1	3	1	1	
25479	17209	1	0	1	0	

25480 rows × 12 columns

In [31]:
```
le.inverse_transform(visa_df['case_status'][:5])
```

Out[31]:
```
array([1, 0, 1, 1, 0])
```

In [30]:
```
visa_df['continent'][:5]
```

Out[30]:
```
0    1
1    1
2    1
3    1
4    0
Name: continent, dtype: int32
```

In [32]:
```
le.inverse_transform(visa_df['case_status'].values[:2])
```

Out[32]:
```
array([1, 0])
```

In [8]:
```
le.inverse_transform(visa_df['continent'])
```

Out[8]:
```
array(['Asia', 'Asia', 'Asia', ..., 'Asia', 'Asia', 'Asia'], dtype=object)
```

In [6]:
```
path =r'C:\Users\aramaiah.ASUAD\Naresh_IT\MyDataScience\Data_Files\Visadataset.csv
visa_df=pd.read_csv(path)
cat_cols=visa_df.select_dtypes(include='object').columns
cat_cols
from sklearn.preprocessing import LabelEncoder #read the pacakge
le=LabelEncoder() #save the package
visa_df['continent']=le.fit_transform(visa_df['continent'])
visa_df
```

Out[6]:

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

25480 rows × 12 columns

In [7]:
```
le.inverse_transform(visa_df['continent'])
```

Out[7]:
```
array(['Asia', 'Asia', 'Asia', ..., 'Asia', 'Asia', 'Asia'], dtype=object)
```

$np.where$

- np.where required 3 arguments
- condition
- True
- False
- It is applicable for Binary labels
- Case status has only two labels Certified and Denied
- if case status== Certified replace that as 0, otherwise 1

In [10]:
```
path =r'C:\Users\aramaiah.ASUAD\Naresh_IT\MyDataScience\Data_Files\Visadataset.csv
visa_df=pd.read_csv(path)
```

In [11]:
```
con=visa_df['case_status']=='Certified'
visa_df['case_status']=np.where(con,0,1)
visa_df
```

Out[11]:

e	has_job_experience	requires_job_training	no_of_employees	yr_of_estab	region_of_employment	prev
l	N	N	14513	2007	West	
s	Y	N	2412	2002	Northeast	
s	N	Y	44444	2008	West	1
s	N	N	98	1897	West	
s	Y	N	1082	2005	South	1
.	...	...	...	...	...	
s	Y	Y	2601	2008	South	
l	Y	N	3274	2006	Northeast	2
s	Y	N	1121	1910	South	1
s	Y	Y	1918	1887	West	
s	Y	N	3195	1960	Midwest	

One hot encoder

- one hot encoder name says at a time one will on and other will off
- for example case status has two columns has two labels
  - Certified
  - Denied
- When you apply one hot encding on case stauts , it creates two more extra columns
  - Case_status_Certified
  - Case_status-Denied

Case_status	Case_status_Certified	Case_status_Denied
Certified	1	0
Denied	0	1

Adavantagees

- When you develop ML model it is very important that the column should e independent to each other
- So here case status creating two extra columns
- Which are independednt to each other which means the row values at a time only one column has 1
- Columns are independent to each other
- Which means 90 degrees phase shift
- which means perpendicular to each other
- Which means orthogonal to each other

Disadvantage

- if a column has 100 unique lables , 100 new clumns will be created
- The data will become sparse, which means huge
- The processing time is more
- Coulmns are more means dimentions are more
- The memory consumption is more
- **Curse of Dimensionality**

pd.get_dummies

In [17]:

```
# Read teh data
path =r'C:\Users\aramaiah.ASUAD\Naresh_IT\MyDataScience\Data_Files\Visadataset.csv
visa_df=pd.read_csv(path)
pd.get_dummies(visa_df,columns=['case_status','education_of_employee'],dtype='int'
```

Out[17]:

Certified	case_status_Denied	education_of_employee_Bachelor's	education_of_employee_Doctorate	edu
0	1	0	0	
1	0	0	0	
0	1	1	0	
0	1	1	0	
1	0	0	0	
...	...	...	...	
1	0	1	0	
1	0	0	0	
1	0	0	0	
1	0	0	0	
1	0	1	0	

In [18]:

```
# Read teh data
path =r'C:\Users\aramaiah.ASUAD\Naresh_IT\MyDataScience\Data_Files\Visadataset.csv
visa_df=pd.read_csv(path)
# make sure to drop the id column
visa_df.drop('case_id',axis=1,inplace=True)
# When you dont provide the specific column it will take all the coulmn
pd.get_dummies(visa_df,dtype='int')
```

Out[18]:

_of_wage_Hour	unit_of_wage_Month	unit_of_wage_Week	unit_of_wage_Year	full_time_position_N	full_t
1	0	0	0	0	
0	0	0	1	0	
0	0	0	1	0	
0	0	0	1	0	
0	0	0	1	0	
...	...	...	...	...	
0	0	0	1	0	
0	0	0	1	0	
0	0	0	1	1	
0	0	0	1	0	
0	0	0	1	0	

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: