- It is very important convert categorical data to numerical data
- Because all ML models developed by using mathamatics
- If you try to implement ML algorithm development using Cat columns it will fail
- Methods:
  - Manually by usin map method
  - One hot encoder : pd.get_dummies
  - Label encode : Sickit learn package

Import required packages

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Read the data

In [2]:
```
file_location="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh I
visa_df=pd.read_csv(file_location)
visa_df.head()
```

Out[2]:

	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	

Method-1

map

- First consider one categorical column
- Observe how many unique lables are there
- Assume here we are taking case_status column
- case_status two unique lables are there
  - Denied
  - Certified
- Create a dictionary by providing key:value
- Here the key= Denied and the value= 1
- key= Certified and the value = 0
- dictionary will looks : {'Certified':0,'Denied':1}

In [3]:
```
visa_df['case_status'].unique()
```

Out[3]:
```
array(['Denied', 'Certified'], dtype=object)
```

In [4]:
```
d1={'Certified':0,'Denied':1}
visa_df['case_status'].map(d1)

# After mapping, in which column you want apply the data
# You want to create a new column
# you want to use old columns

visa_df['case_status']=visa_df['case_status'].map(d1)
```

In [5]:
```
visa_df.head()
```

Out[5]:

	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	

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

Out[7]:
```
'Asia'
```

In [11]:
```
d1={}
labels=visa_df['continent'].unique()
for i in range(len(labels)):
    d1[labels[i]]=i

visa_df['continent']=visa_df['continent'].map(d1)
visa_df.head()
```

Out[11]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_
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	

Methdod-2

np. where

In [12]:
```
# Read the data again
file_location="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh I
visa_df=pd.read_csv(file_location)
visa_df.head()
```

Out[12]:

	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	

- np.where is a kind of if-else
- np.where is useful for binary condition
- which means it is useful for the columns having two labels
- np.where will take 3 parameters
  - condition
  - True value
  - False value
- I want to replace case status column with 1 when it is Denied
- Otherwise it is zero
- con=visa_df['case_status']=='Denied'
- True vaue =1 False value =0
- np.where(con,True value,False value)

In [14]:
```
con=visa_df['case_status']=='Denied'
visa_df['case_status']=np.where(con,1,0)
visa_df.head()
```

Out[14]:

	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	

Method-3

$one - hot - encoder$

pd.getdummies()

- one hot encoder means at a time only one will be ON , Other is OFF
- ON represents with 1, and OFF represents with 0
- One hot encoder creates new extra columns based on lables in the columns
- For example case_status has two unique labels

- If you apply one hot encoder on case_status it creates two extra columns
  - case_status_Denied
  - case_status_Certified

Case status	case_status_Denied	case_status_Certified
Denied	1	0
Certified	0	1

Advantage

- One hot encoder gives independancy between the variables
- It is one of the important property before apply ML models
- Variables are independent each other
- Variables are perpendicular each other
- Variables are Orthogonal each other
- Variables are 90 degrees phase shift

Disadvantage

- Assume that a column has 100 unique labels
- If we apply one hot encoder it will create 100 New columns
- Your data matrix become sparse
- We required More time and more memory to compute the calculations
- This is called **Curse of Dimensionality**

In [19]:
```
# Read the data again
file_location="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh I
visa_df=pd.read_csv(file_location)
visa_df.head()
```

Out[19]:

	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	

In [20]:
```
# Read the data
file_location="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh 
visa_df=pd.read_csv(file_location)

# Drop the ID befor apply One hot encoder
visa_df.drop('case_id',axis=1,inplace=True)

# Apply one hot encoder
pd.get_dummies(visa_df,dtype='int')
```

Out[20]:

	no_of_employees	yr_of_estab	prevailing_wage	continent_Africa	continent_Asia	cont
0	14513	2007	592.2029	0	1	
1	2412	2002	83425.6500	0	1	
2	44444	2008	122996.8600	0	1	
3	98	1897	83434.0300	0	1	
4	1082	2005	149907.3900	1	0	
...	...	...	...	...	...	
25475	2601	2008	77092.5700	0	1	
25476	3274	2006	279174.7900	0	1	
25477	1121	1910	146298.8500	0	1	
25478	1918	1887	86154.7700	0	1	
25479	3195	1960	70876.9100	0	1	

25480 rows × 30 columns

Method-4

Label Encoder

- we already implemented map method by our own
- The same map method implemented as Python package way
- Package name: **sklearn**
- Method name: **LabelEncoder**
- Sklearn package approach is very easy
  - Step-1: Read the package
  - Step-2: Save the package
  - Step-3: Apply fit transform

In [26]:
```
file_location="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh 
visa_df=pd.read_csv(file_location)
print(visa_df['continent'].values[:10])

# Read the package
from sklearn.preprocessing import LabelEncoder
# save the package
le=LabelEncoder()
# Apply fit transform
visa_df['continent']=le.fit_transform(visa_df['continent'])
visa_df.head()
```

```
['Asia' 'Asia' 'Asia' 'Asia' 'Africa' 'Asia' 'Asia' 'North America' 'Asia'
 'Europe']
```

Out[26]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_
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	

In [32]:
```
file_location="C:\\Users\\omkar\\OneDrive\\Documents\\Data science\\Naresh 
visa_df=pd.read_csv(file_location)

# Read the package
from sklearn.preprocessing import LabelEncoder
# save the package
le=LabelEncoder()
# Apply fit transform
# we cant directly pass the data
# Do the loop
cat_cols=visa_df.select_dtypes(include='object').columns

for i in cat_cols:
    visa_df[i]=le.fit_transform(visa_df[i])

visa_df.head()
```

```
['Asia' 'Asia' 'Asia' 'Asia' 'Africa' 'Asia' 'Asia' 'North America' 'Asia'
 'Europe']
```

Out[32]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_
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	

inverse transform

In [36]:
```
print(visa_df['case_status'][:5])  # transformed one
# we are trying to check what is 1 and 0
# le is LabelEncoder we already defined it
le.inverse_transform(visa_df['case_status'])[:5]
```

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

Out[36]:
```
array(['Denied', 'Certified', 'Denied', 'Denied', 'Certified'],
      dtype=object)
```

- Map
- np.where
- one hot enoder pd.getdummies()
- LableEncoder sklearn

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: