- Before Model Development
- We will do EDA : Exploratry data analysis
  - Features(Columns) analysis
    - Categorical , Numerical , Outlier analysis, Bivariate, Multivariate
    - We will get the insights
  - Feature Engineering
    - we will modify the data
    - Models developed by maths
    - Encoding : Convert categorical data to Numerical Variables
    - Scaling : All the numerical columns makes under one scale
    - Transformations : All the maths developed by assumption data follows Normal distribution
      - Convert skewed distribution to Normal
  - Feature selection

In [2]:
```
# import the packages
# read the data
# divide into numerical and categorical
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

file_path=r'C:\Users\omkar\OneDrive\Documents\Gen_AI\Data_files\Visadataset.csv'
visa_df=pd.read_csv(file_path)

cat=visa_df.select_dtypes(include='object').columns
num=visa_df.select_dtypes(exclude='object').columns
```

In [4]:
```
visa_df['case_status']
```

Out[4]:
```
0           Denied
1        Certified
2           Denied
3           Denied
4        Certified
           ...
25475    Certified
25476    Certified
25477    Certified
25478    Certified
25479    Certified
Name: case_status, Length: 25480, dtype: object
```

map method

- create a dictionary Lables as keys, numbers a values
- in python index start with 0
- Case status has two lables we need two values 0 and 1
- Certified becomes 0
- Denied becomes 1

In [7]:
```
visa_df['case_status'].nunique()
```

Out[7]:
```
2
```

In [10]:
```
visa_df['case_status'].unique()
```

Out[10]:
```
array(['Denied', 'Certified'], dtype=object)
```

In [12]:
```
d={'Certified':0,'Denied':1}
visa_df['case_status']=visa_df['case_status'].map(d)
```

In [14]:
```
visa_df
```

Out[14]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_t
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 [24]:
```
# idea
# how many lables that many numbers start 0 end with len
# empty dictionary
d={}
d['Certified']=0
```

```
d['Denied']=1
d
```

Out[24]: {'Certified': 0, 'Denied': 1}

In [34]:
```
visa_df=pd.read_csv(file_path)
d={}
lables=visa_df['case_status'].unique()
for i in range(len(lables)):
    d[lables[i]]=i

print(d)
visa_df['case_status']=visa_df['case_status'].map(d)
```

```
{'Denied': 0, 'Certified': 1}
```

In [36]:
```
visa_df
```

Out[36]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_t
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 [38]:
```
visa_df=pd.read_csv(file_path)
lables=visa_df['case_status'].unique()
d={lables[i]:i for i in range(len(lables))}
visa_df['case_status']=visa_df['case_status'].map(d)
```

```
{'Denied': 0, 'Certified': 1}
```

In [44]:
```
visa_df=pd.read_csv(file_path)
for i in cat[1:]:
    lables=visa_df[i].unique()
    d={lables[i]:i for i in range(len(lables))}
    visa_df[i]=visa_df[i].map(d)
```

In [46]:
```
visa_df
```

Out[46]:

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

25480 rows × 12 columns

LabelEncoder

- Sickit-learn(sklearn)
- sklearn library is a heart of machine learning in Python
- MLlib is heart of machine learning in pyspark
- sklearn (Library)
  - preprocess (Class)
    - LabelEncoder (Method)
- fit : calcuate something
- fit_transform : we are modify the values because of calculated one

In [50]:
```
# read the data again
visa_df=pd.read_csv(file_path)
```

In [56]:
```
from sklearn.preprocessing import LabelEncoder # Read the method
le=LabelEncoder() # save the method
le.fit_transform(visa_df['case_status']) # apply fit transform
```

Out[56]:
```
array([1, 0, 1, ..., 0, 0, 0])
```

In [ ]:
```
Sir , when to use fit # when to teach
```