In [1]:

```
import pandas as pd

df=pd.DataFrame({'Salary':[25000,48000,71000,85000,90000,55000],
                 'City':['Bengaluru','Delhi','Hyderabad','Bengaluru','Hyderabad','Bengaluru
                'Gender':['Male','Female','Female','Female','Male','Male'],
                'Exp':[1,3,5,6,9,None]})
df
```

Out[1]:

	Salary	City	Gender	Exp
0	25000	Bengaluru	Male	1.0
1	48000	Delhi	Female	3.0
2	71000	Hyderabad	Female	5.0
3	85000	Bengaluru	Female	6.0
4	90000	Hyderabad	Male	9.0
5	55000	Bengaluru	Male	NaN

Encoder and Imputers

In [2]:

```
from sklearn.preprocessing import LabelEncoder
```

In [3]:

```
lab_enc= LabelEncoder()
```

In [4]:

```
df2=lab_enc.fit_transform(df['City'])
pd.Series(df2)
```

Out[4]:

```
0    0
1    1
2    2
3    0
4    2
5    0
dtype: int32
```

In [5]:

```
df['City']=df2
df
```

Out[5]:

	Salary	City	Gender	Exp
0	25000	0	Male	1.0
1	48000	1	Female	3.0
2	71000	2	Female	5.0
3	85000	0	Female	6.0
4	90000	2	Male	9.0
5	55000	0	Male	NaN

In [7]:

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import make_column_transformer
```

In [8]:

```
ohe=OneHotEncoder()
si=SimpleImputer()
```

In [9]:

```
import pandas as pd

df=pd.DataFrame({'Salary':[25000,48000,71000,85000,90000,55000],
                 'City':['Bengaluru','Delhi','Hyderabad','Bengaluru','Hyderabad','Bengaluru
                'Gender':['Male','Female','Female','Female','Male','Male'],
                'Exp':[1,3,5,6,9,None]})
df
```

Out[9]:

	Salary	City	Gender	Exp
0	25000	Bengaluru	Male	1.0
1	48000	Delhi	Female	3.0
2	71000	Hyderabad	Female	5.0
3	85000	Bengaluru	Female	6.0
4	90000	Hyderabad	Male	9.0
5	55000	Bengaluru	Male	NaN

In [10]:

```
ct=make_column_transformer(
   (ohe,['City','Gender']),
   (si,['Exp']),
   remainder='passthrough') # 'passthrough' to keep all other columns
```

In [11]:

```
encoded=pd.DataFrame(ct.fit_transform(df))
encoded
```

Out[11]:

	0	1	2	3	4	5	6
0	1.0	0.0	0.0	0.0	1.0	1.0	25000.0
1	0.0	1.0	0.0	1.0	0.0	3.0	48000.0
2	0.0	0.0	1.0	1.0	0.0	5.0	71000.0
3	1.0	0.0	0.0	1.0	0.0	6.0	85000.0
4	0.0	0.0	1.0	0.0	1.0	9.0	90000.0
5	1.0	0.0	0.0	0.0	1.0	4.8	55000.0

In [12]:

```
#Rename the columns as per your choice

encoded = pd.DataFrame(ct.fit_transform(df),columns=['City_Bengaluru','City_Delhi','City_Hy
```

In [13]:

```
encoded
```

Out[13]:

	City_Bengaluru	City_Delhi	City_Hyd	Gender_male	Gender_Female	Exp	Salary
0	1.0	0.0	0.0	0.0	1.0	1.0	25000.0
1	0.0	1.0	0.0	1.0	0.0	3.0	48000.0
2	0.0	0.0	1.0	1.0	0.0	5.0	71000.0
3	1.0	0.0	0.0	1.0	0.0	6.0	85000.0
4	0.0	0.0	1.0	0.0	1.0	9.0	90000.0
5	1.0	0.0	0.0	0.0	1.0	4.8	55000.0

In [14]:

```
# Original Data set
df
```

Out[14]:

	Salary	City	Gender	Exp
0	25000	Bengaluru	Male	1.0
1	48000	Delhi	Female	3.0
2	71000	Hyderabad	Female	5.0
3	85000	Bengaluru	Female	6.0
4	90000	Hyderabad	Male	9.0
5	55000	Bengaluru	Male	NaN

Get_Dummies

one hot Encoding and get_dummies almost equal.Major difference is if you want to reduce(drop_first=True) the column size of the data set you can use get_dummies

OHE does not add variable names to your data frame but get_dummies add variable names.

sometimes having more columns might overfit the model

In [16]:

```
df1=pd.get_dummies(df[['City','Gender']])
df1
```

Out[16]:

	City_Bengaluru	City_Delhi	City_Hyderabad	Gender_Female	Gender_Male
0	1	0	0	0	1
1	0	1	0	1	0
2	0	0	1	1	0
3	1	0	0	1	0
4	0	0	1	0	1
5	1	0	0	0	1

In [17]:

```
df1=pd.get_dummies(df[['City','Gender']],drop_first=True)
df1
```

Out[17]:

	City_Delhi	City_Hyderabad	Gender_Male
0	0	0	1
1	1	0	0
2	0	1	0
3	0	0	0
4	0	1	1
5	0	0	1

Ordinal Encoder

In [18]:

```
from sklearn.preprocessing import OrdinalEncoder
```

In [20]:

```
import pandas as pd

Employee =pd.DataFrame({'Position':['SE','Manager','Team Lead','SSE'],
                        'Project':['A','B','C','D'],
                        'Salary':[25000,85000,71000,48000]})

Employee
```

Out[20]:

	Position	Project	Salary
0	SE	A	25000
1	Manager	B	85000
2	Team Lead	C	71000
3	SSE	D	48000

In [21]:

```
ord_enc=OrdinalEncoder(categories=[['SE','SSE','Team Lead','Manager'],['A','B','C','D']])
Encoded_df=ord_enc.fit_transform(Employee[['Position','Project']])
```

In [22]:

```
Encoded_df
```

Out[22]:

```
array([[0., 0.],
       [3., 1.],
       [2., 2.],
       [1., 3.]])
```

Binary Encoder

In [23]:

```
import pandas as pd

df=pd.DataFrame({'Cat_data':['A','B','C','D','E','F','G','H','I','A','A','D']})
df
```

Out[23]:

	Cat_data
0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H
8	I
9	A
10	A
11	D

In [27]:

```
pip install category_encoders
```

```
Collecting category_encodersNote: you may need to restart the kernel to use
updated packages.

  Downloading category_encoders-2.4.1-py2.py3-none-any.whl (80 kB)
Requirement already satisfied: patsy>=0.5.1 in c:\users\user\anaconda3\lib\s
ite-packages (from category_encoders) (0.5.2)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\user\anaconda3
\lib\site-packages (from category_encoders) (0.12.2)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\user\anacond
a3\lib\site-packages (from category_encoders) (0.24.2)
Requirement already satisfied: pandas>=0.21.1 in c:\users\user\anaconda3\lib
\site-packages (from category_encoders) (1.3.4)
Requirement already satisfied: scipy>=1.0.0 in c:\users\user\anaconda3\lib\s
ite-packages (from category_encoders) (1.7.1)
Requirement already satisfied: numpy>=1.14.0 in c:\users\user\anaconda3\lib
\site-packages (from category_encoders) (1.20.3)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\user\anaco
nda3\lib\site-packages (from pandas>=0.21.1->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\user\anaconda3\lib\s
ite-packages (from pandas>=0.21.1->category_encoders) (2021.3)
Requirement already satisfied: six in c:\users\user\anaconda3\lib\site-packa
ges (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=0.11 in c:\users\user\anaconda3\lib\s
ite-packages (from scikit-learn>=0.20.0->category_encoders) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\user\anacond
a3\lib\site-packages (from scikit-learn>=0.20.0->category_encoders) (2.2.0)
Installing collected packages: category-encoders
Successfully installed category-encoders-2.4.1
```

In [30]:

```
from category_encoders import BinaryEncoder
from sklearn.preprocessing import OneHotEncoder
```

In [31]:

```
bi_enc=BinaryEncoder()
```

In [32]:

```
df_bi=bi_enc.fit_transform(df)
df_bi
```

Out[32]:

	Cat_data_0	Cat_data_1	Cat_data_2	Cat_data_3
0	0	0	0	1
1	0	0	1	0
2	0	0	1	1
3	0	1	0	0
4	0	1	0	1
5	0	1	1	0
6	0	1	1	1
7	1	0	0	0
8	1	0	0	1
9	0	0	0	1
10	0	0	0	1
11	0	1	0	0

Comparing with OneHotEncoder

In [33]:

```
ohe=OneHotEncoder(sparse=False)
ohe.fit_transform(df[['Cat_data']])
```

Out[33]:

```
array([[1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0.]])
```

Knn Imputer

In [34]:

```
import pandas as pd

df=pd.DataFrame({'Salary':[25000,48000,71000,85000,90000,55000],
                 'City':['Bengaluru','Delhi','Hyderabad','Bengaluru','Hyderabad','Bengaluru
                'Gender':['Male','Female','Female','Female','Male','Male'],
                'Exp':[1,3,5,6,9,None]})
df
```

Out[34]:

	Salary	City	Gender	Exp
0	25000	Bengaluru	Male	1.0
1	48000	Delhi	Female	3.0
2	71000	Hyderabad	Female	5.0
3	85000	Bengaluru	Female	6.0
4	90000	Hyderabad	Male	9.0
5	55000	Bengaluru	Male	NaN

In [35]:

```
#knn imputer will try to find the relation with other columns and impute the data according
# In this case Age NaN is depending on the similarity with Fare columns

from sklearn.impute import KNNImputer
```

In [37]:

```
knnimp =KNNImputer(n_neighbors=3)
knn_imp=pd.DataFrame(knnimp.fit_transform(df[['Salary','Exp']]))

knn_imp
```

Out[37]:

	0	1
0	25000.0	1.0
1	48000.0	3.0
2	71000.0	5.0
3	85000.0	6.0
4	90000.0	9.0
5	55000.0	3.0

Iterative Imputer

This method treat other columns (which doesnot have nulls as feature and train on them and trat Null column as label. Finally it will predict the NaN data and impute. Its just likr regression problem. Here Null column is label

In [38]:

```
# Before usin Iterative  Imputer, we need to enable it using below code.
from sklearn.experimental import enable_iterative_imputer

# import Iterative Imputer

from sklearn.impute import IterativeImputer
```

In [39]:

```
import pandas as pd

df=pd.DataFrame({'Salary':[25000,48000,71000,85000,90000,55000],
                 'City':['Bengaluru','Delhi','Hyderabad','Bengaluru','Hyderabad','Bengaluru
                'Gender':['Male','Female','Female','Female','Male','Male'],
                'Exp':[1,3,5,6,9,None]})
df
```

Out[39]:

	Salary	City	Gender	Exp
0	25000	Bengaluru	Male	1.0
1	48000	Delhi	Female	3.0
2	71000	Hyderabad	Female	5.0
3	85000	Bengaluru	Female	6.0
4	90000	Hyderabad	Male	9.0
5	55000	Bengaluru	Male	NaN

In [40]:

```
iter_impute = IterativeImputer()
ite_imp=pd.DataFrame(iter_impute.fit_transform(df[['Salary','Exp']]),columns=['Salary','Exp
ite_imp
```

Out[40]:

	Salary	Exp
0	25000.0	1.000000
1	48000.0	3.000000
2	71000.0	5.000000
3	85000.0	6.000000
4	90000.0	9.000000
5	55000.0	3.864759

In [ ]:

```

```