12_ML - Label Encoder

12.1 Label encoding on Nominal data

In [3]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
```

In [5]:
```
df = pd.DataFrame({'name':['Rashid', 'Lion', 'Computer', 'Gym', 'Plant']})
df
```

Out[5]:

	name
0	Rashid
1	Lion
2	Computer
3	Gym
4	Plant

In [7]:
```
le = LabelEncoder()
df['en_name'] = le.fit_transform(df['name'])
```

In [8]:
```
df
```

Out[8]:

	name	en_name
0	Rashid	4
1	Lion	2
2	Computer	0
3	Gym	1
4	Plant	3

Now work on real time data

In [9]:
```
dataset = pd.read_csv('loan.csv')
```

In [10]:
```
dataset.head(3)
```

Out[10]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	

In [14]:
```
# To check number of data
dataset['Property_Area'].unique()
```

Out[14]:
```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

In [12]:
```
la = LabelEncoder()
la.fit(dataset['Property_Area'])
```

Out[12]:
```
▾ LabelEncoder
LabelEncoder()
```

In [13]:
```
la.transform(dataset['Property_Area'])
```

Out[13]:
```
array([2, 0, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2,
       1, 0, 1, 1, 1, 2, 2, 1, 2, 2, 0, 1, 0, 2, 2, 1, 2, 1, 2, 2, 2, 1,
       2, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 0, 2, 2, 2, 2, 0, 0, 1, 1,
       2, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1,
       2, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 2, 2, 0, 2, 1,
       2, 1, 0, 1, 1, 0, 1, 2, 0, 2, 0, 1, 1, 1, 0, 0, 0, 0, 2, 0, 2, 2,
       1, 1, 1, 1, 0, 2, 1, 0, 0, 2, 1, 1, 2, 1, 2, 2, 0, 1, 0, 0, 2, 0,
       2, 1, 0, 2, 0, 1, 1, 2, 1, 0, 2, 0, 0, 0, 1, 1, 0, 2, 0, 1, 1, 0,
       0, 1, 1, 2, 2, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 2, 1, 0, 1, 0, 2,
       1, 2, 1, 1, 2, 2, 1, 1, 2, 0, 2, 1, 1, 1, 2, 0, 2, 1, 0, 1, 1, 1,
       2, 1, 1, 1, 1, 0, 2, 1, 1, 0, 1, 0, 0, 1, 1, 0, 2, 2, 0, 1, 0, 2,
       2, 0, 1, 2, 2, 2, 1, 2, 1, 2, 0, 1, 2, 0, 0, 2, 0, 1, 2, 1, 1, 0,
       1, 0, 1, 2, 0, 2, 2, 2, 0, 1, 1, 1, 1, 2, 1, 0, 2, 1, 2, 2, 0, 0,
       1, 0, 1, 0, 0, 1, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1, 0, 2, 0, 2, 0, 2,
       0, 0, 1, 1, 0, 0, 0, 2, 1, 2, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 2, 2,
       2, 1, 2, 2, 2, 1, 0, 0, 2, 1, 0, 0, 2, 1, 0, 1, 0, 2, 1, 0, 1, 0,
       0, 0, 1, 2, 0, 2, 2, 1, 1, 1, 2, 2, 0, 0, 1, 0, 1, 0, 1, 1, 0, 2,
       2, 2, 0, 1, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 0, 0, 0, 2, 1, 2, 1,
       2, 2, 0, 1, 2, 0, 1, 1, 0, 1, 2, 0, 1, 0, 1, 2, 0, 0, 1, 2, 2, 2,
       0, 1, 0, 2, 2, 2, 1, 0, 0, 1, 0, 2, 1, 0, 1, 1, 2, 1, 1, 2, 2, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 2, 0, 0, 1, 1, 2, 2, 0, 1, 1, 2,
       0, 1, 1, 0, 2, 1, 1, 2, 1, 0, 1, 2, 0, 0, 1, 1, 1, 2, 0, 0, 1, 1,
       1, 0, 0, 2, 1, 2, 1, 2, 0, 1, 0, 1, 0, 2, 1, 0, 0, 1, 1, 0, 1, 0,
       2, 2, 2, 2, 0, 1, 2, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 2, 0, 1, 2, 0, 2, 1, 0, 0, 1, 1, 1, 2, 1, 0, 1, 0, 1, 0,
       0, 0, 2, 2, 0, 1, 2, 1, 1, 1, 1, 1, 0, 1, 2, 0, 2, 0, 2, 2, 2, 2,
       2, 1, 1, 2, 1, 2, 0, 2, 1, 2, 1, 0, 0, 0, 2, 1, 1, 1, 1, 1, 1, 0,
       2, 0, 0, 1, 0, 2, 2, 0, 2, 0, 1, 2, 1, 0, 0, 0, 0, 2, 2, 1])
```

In [15]:
```
# to replace the property data with encoding data
dataset['Property_Area'] = la.transform(dataset['Property_Area'])
```

In [18]:
```
dataset.head(3)
```

Out[18]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	

12.1 Label encoding on Ordinal data

12.1.1 Ordincal encoding through Cyclic Line

In [21]:
```
dfo = pd.DataFrame({'size': ['s', 'm', 'l', 'xl', 'xxl', 's', 's', 's', 'xl', 'm',
dfo.head(3)
```

Out[21]:

	size
0	s
1	m
2	l

In [22]:
```
ord_data = [['s', 'm', 'l', 'xl', 'xxl']]
```

In [24]:
```
from sklearn.preprocessing import OrdinalEncoder
```

In [25]:
```
# oe = OrdinalEncoder() : This will encode the data alphabatically
oe = OrdinalEncoder(categories=ord_data)
oe.fit(dfo[['size']])
```

Out[25]:

OrdinalEncoder

```
OrdinalEncoder(categories=[['s', 'm', 'l', 'xl', 'xxl']])
```

In [27]:
```
oe.transform(dfo[['size']])
```

Out[27]:
```
array([[0.],
       [1.],
       [2.],
       [3.],
       [4.],
       [0.],
       [0.],
       [0.],
       [3.],
       [1.],
       [2.]])
```

In [29]:
```
dfo['size_en'] = oe.transform(dfo[['size']])
dfo
```

Out[29]:

	size	size_en
0	s	0.0
1	m	1.0
2	l	2.0
3	xl	3.0
4	xxl	4.0
5	s	0.0
6	s	0.0
7	s	0.0
8	xl	3.0
9	m	1.0
10	l	2.0

12.1.2 Ordincal encoding through Map function

In [30]:
```
# In map function, you can manually assign numbers to each data type, for example
# You can assign any number
ord_data1 = {'s':0, 'm':1, 'l':2, 'xl':3, 'xxl':4}
```

In [32]:
```
dfo['size'].map(ord_data1)
```

Out[32]:
```
0     0
1     1
2     2
3     3
4     4
5     0
6     0
7     0
8     3
9     1
10    2
Name: size, dtype: int64
```

In [33]:
```
dfo['size_en_map'] = dfo['size'].map(ord_data1)
dfo
```

Out[33]:

	size	size_en	size_en_map
0	s	0.0	0
1	m	1.0	1
2	l	2.0	2
3	xl	3.0	3
4	xxl	4.0	4
5	s	0.0	0
6	s	0.0	0
7	s	0.0	0
8	xl	3.0	3
9	m	1.0	1
10	l	2.0	2

12.2 Perform Ordinal Encoding on big data

In [35]:
```
dataset = pd.read_csv('loan.csv')
```

In [36]:
```
dataset.head(3)
```

Out[36]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	

In [37]:
```
dataset['Property_Area'].unique()
```

Out[37]:
```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

In [40]:
```
# if there is nan (missing data)m, you can fill it
# if the data is categorical then you should do mode filling
dataset['Property_Area'].fillna(dataset['Property_Area'].mode()[0], inplace=True)
```

In [42]:
```
en_data_loan = [['Urban', 'Rural', 'Semiurban']]
```

In [45]:
```
oen = OrdinalEncoder(categories=en_data_loan)
oen.fit_transform(dataset[['Property_Area']])
```

Out[45]:
```
array([[0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [2.],
       [0.],
       [2.],
       [0.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [2.],
       [1.],
       [2.],
       [2.],
       [2.],
       [0.],
       [0.],
       [2.],
       [0.],
       [0.],
       [1.],
       [2.],
       [1.],
       [0.],
       [0.],
       [2.],
       [0.],
       [2.],
       [0.],
       [0.],
       [0.],
       [2.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [2.],
       [2.],
       [2.],
       [2.],
       [0.],
       [0.],
       [2.],
```

[2.],
[1.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[2.],
[2.],
[0.],
[0.],
[0.],
[2.],
[0.],
[2.],
[2.],
[2.],
[2.],
[0.],
[0.],
[0.],
[2.],
[2.],
[2.],
[2.],
[0.],
[2.],
[0.],
[2.],
[2.],
[2.],
[0.],
[2.],
[2.],
[2.],
[0.],
[2.],
[2.],
[2.],
[2.],
[2.],
[0.],
[2.],
[0.],
[2.],
[0.],
[0.],
[0.],
[1.],
[0.],
[2.],
[0.],
[2.],

[1.],
[2.],
[2.],
[1.],
[2.],
[0.],
[1.],
[0.],
[1.],
[2.],
[2.],
[2.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[0.],
[2.],
[2.],
[2.],
[2.],
[1.],
[0.],
[2.],
[1.],
[1.],
[0.],
[2.],
[2.],
[0.],
[2.],
[0.],
[0.],
[1.],
[2.],
[1.],
[1.],
[0.],
[1.],
[0.],
[2.],
[1.],
[0.],
[1.],
[2.],
[2.],
[0.],
[2.],
[1.],
[0.],
[1.],
[1.],
[1.],

[2.],
[2.],
[1.],
[0.],
[1.],
[2.],
[2.],
[1.],
[1.],
[2.],
[2.],
[0.],
[0.],
[1.],
[2.],
[2.],
[2.],
[2.],
[1.],
[1.],
[1.],
[1.],
[1.],
[2.],
[0.],
[2.],
[1.],
[2.],
[1.],
[0.],
[2.],
[0.],
[2.],
[2.],
[0.],
[0.],
[2.],
[2.],
[0.],
[1.],
[0.],
[2.],
[2.],
[2.],
[0.],
[1.],
[0.],
[2.],
[1.],
[2.],
[2.],
[2.],
[0.],
[2.],
[2.],
[2.],

```
[2.],
[1.],
[0.],
[2.],
[2.],
[1.],
[2.],
[1.],
[1.],
[2.],
[2.],
[1.],
[0.],
[0.],
[1.],
[2.],
[1.],
[0.],
[0.],
[1.],
[2.],
[0.],
[0.],
[0.],
[2.],
[0.],
[2.],
[0.],
[1.],
[2.],
[0.],
[1.],
[1.],
[0.],
[1.],
[2.],
[0.],
[2.],
[2.],
[1.],
[2.],
[1.],
[2.],
[0.],
[1.],
[0.],
[0.],
[0.],
[1.],
[2.],
[2.],
[2.],
[2.],
[0.],
[2.],
[1.],
```

[0.],
[2.],
[0.],
[0.],
[1.],
[1.],
[2.],
[1.],
[2.],
[1.],
[1.],
[2.],
[0.],
[0.],
[2.],
[0.],
[2.],
[0.],
[1.],
[0.],
[0.],
[2.],
[1.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[2.],
[2.],
[1.],
[1.],
[1.],
[0.],
[2.],
[0.],
[2.],
[1.],
[2.],
[2.],
[1.],
[1.],
[1.],
[1.],
[1.],
[2.],
[0.],
[0.],
[0.],
[2.],
[0.],
[0.],
[0.],
[2.],

[1.],
[1.],
[0.],
[2.],
[1.],
[1.],
[0.],
[2.],
[1.],
[2.],
[1.],
[0.],
[2.],
[1.],
[2.],
[1.],
[1.],
[1.],
[2.],
[0.],
[1.],
[0.],
[0.],
[2.],
[2.],
[2.],
[0.],
[0.],
[1.],
[1.],
[2.],
[1.],
[2.],
[1.],
[2.],
[2.],
[1.],
[0.],
[0.],
[0.],
[1.],
[2.],
[0.],
[0.],
[2.],
[2.],
[0.],
[0.],
[0.],
[0.],
[2.],
[0.],
[0.],
[1.],
[1.],
[1.],

```
[0.],
[2.],
[0.],
[2.],
[0.],
[0.],
[1.],
[2.],
[0.],
[1.],
[2.],
[2.],
[1.],
[2.],
[0.],
[1.],
[2.],
[1.],
[2.],
[0.],
[1.],
[1.],
[2.],
[0.],
[0.],
[0.],
[1.],
[2.],
[1.],
[0.],
[0.],
[0.],
[2.],
[1.],
[1.],
[2.],
[1.],
[0.],
[2.],
[1.],
[2.],
[2.],
[0.],
[2.],
[2.],
[0.],
[0.],
[1.],
[2.],
[1.],
[2.],
[2.],
[1.],
[1.],
[1.],
[1.],
```

```
[1.],
[2.],
[1.],
[0.],
[1.],
[1.],
[2.],
[2.],
[0.],
[0.],
[1.],
[2.],
[2.],
[0.],
[1.],
[2.],
[2.],
[1.],
[0.],
[2.],
[2.],
[0.],
[2.],
[1.],
[2.],
[0.],
[1.],
[1.],
[2.],
[2.],
[2.],
[0.],
[1.],
[1.],
[2.],
[2.],
[2.],
[1.],
[1.],
[0.],
[2.],
[0.],
[2.],
[0.],
[1.],
[2.],
[1.],
[2.],
[1.],
[0.],
[2.],
[1.],
[1.],
[2.],
[2.],
[1.],
```

```
[2.],
[1.],
[0.],
[0.],
[0.],
[0.],
[1.],
[2.],
[0.],
[2.],
[1.],
[1.],
[2.],
[2.],
[2.],
[1.],
[2.],
[2.],
[1.],
[1.],
[2.],
[1.],
[2.],
[2.],
[2.],
[2.],
[1.],
[0.],
[1.],
[2.],
[0.],
[1.],
[0.],
[2.],
[1.],
[1.],
[2.],
[2.],
[2.],
[0.],
[2.],
[1.],
[2.],
[1.],
[2.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[2.],
[0.],
[2.],
[2.],
[2.],
```

```
[2.],
[2.],
[1.],
[2.],
[0.],
[1.],
[0.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[2.],
[2.],
[0.],
[2.],
[0.],
[1.],
[0.],
[2.],
[0.],
[2.],
[1.],
[1.],
[1.],
[0.],
[2.],
[2.],
[2.],
[2.],
[2.],
[2.],
[1.],
[0.],
[1.],
[1.],
[2.],
[1.],
[0.],
[0.],
[1.],
[0.],
[1.],
[2.],
[0.],
[2.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[2.]])
```

In [47]:
```
dataset['Property_Area'] = oen.fit_transform(dataset[['Property_Area']])
```

In [49]:
```
dataset.head(3)
```

Out[49]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	