Convert categorical data

- Value replacing
- Label encoding
- One-hot encoding
- Binary encoding

1. Value replacing

Predefine number

In [1]:
```
import pandas as pd
df = pd.DataFrame(data = { 'Job':['Engineer', 'Sale', 'Marketing', 'Finance', 'HR'] ,
                           'Salary':[20000,30000,15000,20000,15000] } )
```

In [2]:
```
df
```

Out[2]:

	Job	Salary
0	Engineer	20000
1	Sale	30000
2	Marketing	15000
3	Finance	20000
4	HR	15000

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Job     5 non-null      object
 1   Salary  5 non-null      int64 
dtypes: int64(1), object(1)
memory usage: 208.0+ bytes
```

In [4]:
```
mapping = { 'Job' : {'Engineer': 101, 'Sale': 102, 'Marketing': 103, 'Finance': 201, 'HR': 202}}
```

In [5]:
```
df_map = df.copy()
df_map.replace(mapping, inplace=True)
```

In [6]:
```
df_map
```

Out[6]:

	Job	Salary
0	101	20000
1	102	30000
2	103	15000
3	201	20000
4	202	15000

In [7]:
```
df_map.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Job     5 non-null      int64
 1   Salary  5 non-null      int64
dtypes: int64(2)
memory usage: 208.0 bytes
```

In [8]:
```
# Change data type to operate faster
df_map['Job'] = df_map['Job'].astype('category')
```

In [9]:
```
df_map.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype   
---  ------  --------------  -----   
 0   Job     5 non-null      category
 1   Salary  5 non-null      int64   
dtypes: category(1), int64(1)
memory usage: 385.0 bytes
```

Auto-number

In [10]:
```
import pandas as pd
df = pd.DataFrame(data = { 'Job':['Engineer', 'Sale', 'Marketing', 'Finance', 'HR'] ,
                           'Salary':[20000,30000,15000,20000,15000] } )
```

In [11]:
```
labels = df['Job']
mapping = {'Job' : {k: v for k,v in zip(labels,list(range(1,len(labels)+1)))}}
```

In [12]:
```
list(range(1,len(labels)+1))
```

Out[12]:
```
[1, 2, 3, 4, 5]
```

In [13]:
```
mapping
```

Out[13]:
```
{'Job': {'Engineer': 1, 'Sale': 2, 'Marketing': 3, 'Finance': 4, 'HR': 5}}
```

In [14]:
```
# Example of zip function
z = zip( ['a','b','c'] , [2,4,9] )
print(tuple(z))
```

```
(('a', 2), ('b', 4), ('c', 9))
```

In [15]:
```
df_map = df.copy()
df_map.replace(mapping, inplace=True)
```

In [16]:
```
df_map
```

Out[16]:

	Job	Salary
0	1	20000
1	2	30000
2	3	15000
3	4	20000
4	5	15000

In [ ]:
```

```

2. Label Encoding

- Numerical labels are always between 0 and n_categories-1

Built-in function in dataframe

In [17]:
```
import pandas as pd
df = pd.DataFrame(data = { 'Job':['Engineer', 'Sale', 'Marketing', 'Finance', 'HR'] ,
                           'Salary':[20000,30000,15000,20000,15000] } )
```

In [18]:
```
df_label = df.copy()
```

In [19]:
```
df_label['Job'] =df_label['Job'].astype('category')
df_label['Job'] = df_label['Job'].cat.codes
```

In [20]:
```
df_label
```

Out[20]:

	Job	Salary
0	0	20000
1	4	30000
2	3	15000
3	1	20000
4	2	15000

Built-in function in 'sklearn'

In [21]:
```
import pandas as pd
df = pd.DataFrame(data = { 'Job':['Engineer', 'Sale', 'Marketing', 'Finance', 'HR'] ,
                           'Salary':[20000,30000,15000,20000,15000] } )
```

In [22]:
```
df_label = df.copy()
```

In [23]:
```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
df_label['Job'] = label.fit_transform(df_label['Job'])
```

In [24]:
```
df_label
```

Out[24]:

	Job	Salary
0	0	20000
1	4	30000
2	3	15000
3	1	20000
4	2	15000

Built-in function in Numpy

(In the case of only two categories)

In [25]:
```
import pandas as pd
df = pd.DataFrame(data = { 'Sex':['M', 'F', 'M', 'M', 'F'] ,
                           'Height':[170,165,168,165,161] } )
```

In [26]:
```
df
```

Out[26]:

	Sex	Height
0	M	170
1	F	165
2	M	168
3	M	165
4	F	161

In [27]:
```
df_label = df.copy()
```

In [28]:
```
import numpy as np
df_label['Sex'] = np.where(df_label['Sex'].str.contains('M'), 1, 0)
```

In [29]:
```
df_label
```

Out[29]:

	Sex	Height
0	1	170
1	0	165
2	1	168
3	1	165
4	0	161

3. One-hot encoding

- Each category value will be a new column
- No weighting value

Built-in function in Pandas

In [30]:
```
import pandas as pd
df = pd.DataFrame(data = { 'Job':['Engineer', 'Sale', 'Marketing', 'Finance', 'HR'] ,
                           'Salary':[20000,30000,15000,20000,15000] } )
```

In [31]:
```
df_label = df.copy()
```

In [32]:
```
df_label = pd.get_dummies(df_label, columns=['Job'], prefix = ['Label'])
```

In [33]:
```
df_label
```

Out[33]:

	Salary	Label_Engineer	Label_Finance	Label_HR	Label_Marketing	Label_Sale
0	20000	1	0	0	0	0
1	30000	0	0	0	0	1
2	15000	0	0	0	1	0
3	20000	0	1	0	0	0
4	15000	0	0	1	0	0

Built-in function in 'sklearn'

In [34]:
```
import pandas as pd
df = pd.DataFrame(data = { 'Job':['Engineer', 'Sale', 'Marketing', 'Finance', 'HR'] ,
                           'Salary':[20000,30000,15000,20000,15000] } )
```

In [35]:
```
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
lb_results = lb.fit_transform(df['Job'])
df_label = pd.DataFrame(lb_results, columns=lb.classes_)
```

In [36]:
```
lb_results
```

Out[36]:
```
array([[1, 0, 0, 0, 0],
       [0, 0, 0, 0, 1],
       [0, 0, 0, 1, 0],
       [0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0]])
```

In [37]:
```
df_label
```

Out[37]:

	Engineer	Finance	HR	Marketing	Sale
0	1	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	1	0	0	0
4	0	0	1	0	0

In [38]:
```
df_label['Salary'] = df['Salary']
```

In [39]:
```
df_label
```

Out[39]:

	Engineer	Finance	HR	Marketing	Sale	Salary
0	1	0	0	0	0	20000
1	0	0	0	0	1	30000
2	0	0	0	1	0	15000
3	0	1	0	0	0	20000
4	0	0	1	0	0	15000

4. Binary encoding

- There are fewer dimensions than the One-hot encoding

In [40]:
```
import pandas as pd
df = pd.DataFrame(data = { 'Job':['Engineer', 'Sale', 'Marketing', 'Finance', 'HR'] ,
                           'Salary':[20000,30000,15000,20000,15000] } )
```

In [41]:
```
df
```

Out[41]:

	Job	Salary
0	Engineer	20000
1	Sale	30000
2	Marketing	15000
3	Finance	20000
4	HR	15000

In [42]:
```
df_label = df.copy()
```

In [43]:
```
#pip install category_encoders
```

In [44]:
```
import category_encoders as ce
encoder = ce.BinaryEncoder(cols=['Job'])
df_label = encoder.fit_transform(df_label)
```

In [45]:
```
df_label
```

Out[45]:

	Job_0	Job_1	Job_2	Salary
0	0	0	1	20000
1	0	1	0	30000
2	0	1	1	15000
3	1	0	0	20000
4	1	0	1	15000

In [ ]:

In [ ]: