Categorical Variable

- We all know our machine learning algorithm not properly work on categorical data (object), so we need to convert categorical to numerical variable.
- Before converting we 1st know types of categorical variable.

There are two types of Categorical Variable

```
1. Ordinal - These  variable are meaningfully orderd
2. Nominal - Here no intrensic order of the label
```

- Example of Ordinal variables: (All are in orders)

```
- High, Medium, Low
- Strongly agree, Agree, Neutral, Disagree, Strongly Disagree.
- Excellent, Okay, Bad
```

- Few examples as below for Nominal variable: (here no order)

```
- Red, Yellow, Pink, Blue
- Singapore, Japan, USA, India, Korea
- Cow, Dog, Cat, Snake
```

There are various technique to Handle categorical variable

- here i provided top 5 technique that use widly

1) One Hot Encoding (used both ordinal and nominal)

2) get_dummy (used both ordinal and nominal)

3) Label Encoding (used only ordinal)

4) mapping by replace function (used both ordinal and nominal)

Now lets implement All encoding method

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
data = {"Temp":['hot','cold','very hot','warm','hot','warm','warm','hot','hot','cold'], #or
         "Color":['red','yellow','black','black','red','white','black','yellow','black','wh
df = pd.DataFrame(data)
df.head(5)
```

Out[2]:

	Temp	Color
0	hot	red
1	cold	yellow
2	very hot	black
3	warm	black
4	hot	red

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Temp    10 non-null     object
 1   Color   10 non-null     object
dtypes: object(2)
memory usage: 288.0+ bytes
```

1. One Hot Encoding

- Scikit-learn has OneHotEncoder for this purpose, but it does not create an additional feature column.

In [4]:

```
df_ohc = df.copy()
df_ohc.head(5)
```

Out[4]:

	Temp	Color
0	hot	red
1	cold	yellow
2	very hot	black
3	warm	black
4	hot	red

In [5]:

```
from sklearn.preprocessing import OneHotEncoder
ohc = OneHotEncoder()
ohe = ohc.fit_transform(df_ohc.Temp.values.reshape(-1,1)).toarray()
dfOnehot = pd.DataFrame(ohe,columns=["Temp_"+str(ohc.categories_[0][i])
                                     for i in range (len(ohc.categories_[0]))
                                     ])
df_ohc = pd.concat([df_ohc,dfOnehot],axis=1)
df_ohc.head(5)
```

Out[5]:

	Temp	Color	Temp_cold	Temp_hot	Temp_very hot	Temp_warm
0	hot	red	0.0	1.0	0.0	0.0
1	cold	yellow	1.0	0.0	0.0	0.0
2	very hot	black	0.0	0.0	1.0	0.0
3	warm	black	0.0	0.0	0.0	1.0
4	hot	red	0.0	1.0	0.0	0.0

here i apply one hot encoding i only "Temp" , you can apply this any categorical variable irrespective of ordinal or nominal

2. get_dummies

- In this method, we map each category to a vector that contains 1 and 0 denoting the presence or absence of the feature. The number of vectors depends on the number of categories for features. This method produces a lot of columns that slows down the learning significantly if the number of the category is very high for the feature.
- this is very easy than sklearn onehot encoding. it just one line code.
- Pandas has get_dummies function, which is quite easy to use. For the sample data-frame code would be as below:

In [6]:

```
df_dummy = df.copy()
```

In [7]:

```
dfdummy = pd.get_dummies(df['Temp'],prefix="Temp_")
df_dummy = pd.concat([df_dummy,dfdummy],axis=1)
df_dummy.head(5)
```

Out[7]:

	Temp	Color	Temp__cold	Temp__hot	Temp__very hot	Temp__warm
0	hot	red	0	1	0	0
1	cold	yellow	1	0	0	0
2	very hot	black	0	0	1	0
3	warm	black	0	0	0	1
4	hot	red	0	1	0	0

- you compare sklearn onehotencoding and pandas getdummy, which is easy for you use.
- in this way you can apply onehot and getdummy to color variable

3. Label Encoding

- In this encoding, each category is assigned a value from 1 through N (here N is the number of categories for the feature. One major issue with this approach is there is no relation or order between these classes, but the algorithm might consider them as some order, or there is some relationship. In below example it may look like (Cold<Hot<Very Hot<Warm….0 < 1 < 2 < 3)
- Point to remember it only apply in ordinal variable.
- Scikit-learn code for the data-frame as follows:

In [8]:

```
df_label = df.copy()
from sklearn.preprocessing import LabelEncoder
lbe = LabelEncoder()
df_label['temp_label_encode'] = lbe.fit_transform(df_label.Temp)
df_label.head(5)
```

Out[8]:

	Temp	Color	temp_label_encode
0	hot	red	1
1	cold	yellow	0
2	very hot	black	2
3	warm	black	3
4	hot	red	1

Here see all our temp variable are convert according order.

4. Using Mapping by replace function

- here we replace the categorical variable with some numeric

In [9]:

```
# here i encode color column
df_mapp = df.copy()
df_mapp['Color'].value_counts()
```

Out[9]:

```
black     4
red       2
yellow    2
white     2
Name: Color, dtype: int64
```

In [10]:

```
df_mapp['Color_encode_map'] = df_mapp['Color'].replace(("black","white","red","yellow"),(0,
df_mapp.head(5)
```

Out[10]:

	Temp	Color	Color_encode_map
0	hot	red	2
1	cold	yellow	3
2	very hot	black	0
3	warm	black	0
4	hot	red	2

see here i encode black:0, white:1,red:2 and yellow:3.