Encoding

Converting Discrete Categorical Variable to Discrete Numerical Variable

There are 2 types of categorical variables

- 1.Nominal (Ex : Item Type)
- 2.Ordinal (Ex : Outlet_Size (Small,Medium,High))

In [1]:
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df = pd.read_csv("homeprices.csv")
df
```

Out[2]:

	town	area	price
0	Chennai	2600	5500000
1	Chennai	3000	5650000
2	Chennai	3200	6100000
3	Chennai	3600	6800000
4	Bangalore	2600	5850000
5	Bangalore	2800	6150000
6	Bangalore	3300	6500000
7	Bangalore	3600	7100000
8	Hyderabad	2600	5750000
9	Hyderabad	2900	6000000
10	Hyderabad	3100	6200000
11	Hyderabad	3600	6950000

To convert Nominal Categorical into Numeric

- 1.get dummies using pandas
- 2.One hot encoding using sklearn

pd.get_dummies() ---> Nominal Variable encoding using pandas

In [3]:
```
dummies = pd.get_dummies(df.town)
dummies
```

Out[3]:

	Bangalore	Chennai	Hyderabad
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	1	0	0
5	1	0	0
6	1	0	0
7	1	0	0
8	0	0	1
9	0	0	1
10	0	0	1
11	0	0	1

In [4]:
```
df_dummies= pd.concat([df,dummies],axis='columns')
df_dummies
```

Out[4]:

	town	area	price	Bangalore	Chennai	Hyderabad
0	Chennai	2600	5500000	0	1	0
1	Chennai	3000	5650000	0	1	0
2	Chennai	3200	6100000	0	1	0
3	Chennai	3600	6800000	0	1	0
4	Bangalore	2600	5850000	1	0	0
5	Bangalore	2800	6150000	1	0	0
6	Bangalore	3300	6500000	1	0	0
7	Bangalore	3600	7100000	1	0	0
8	Hyderabad	2600	5750000	0	0	1
9	Hyderabad	2900	6000000	0	0	1
10	Hyderabad	3100	6200000	0	0	1
11	Hyderabad	3600	6950000	0	0	1

In [5]:
```
df_dummies.drop('town',axis='columns',inplace=True)
df_dummies
```

Out[5]:

	area	price	Bangalore	Chennai	Hyderabad
0	2600	5500000	0	1	0
1	3000	5650000	0	1	0
2	3200	6100000	0	1	0
3	3600	6800000	0	1	0
4	2600	5850000	1	0	0
5	2800	6150000	1	0	0
6	3300	6500000	1	0	0
7	3600	7100000	1	0	0
8	2600	5750000	0	0	1
9	2900	6000000	0	0	1
10	3100	6200000	0	0	1
11	3600	6950000	0	0	1

Dummy Variable Trap

When you can derive one variable from other variables, they are known to be multi-colinear. Here if you know values of california and georgia then you can easily infer value of new jersey state, i.e. california=0 and georgia=0. There for these state variables are called to be multi-colinear. In this situation linear regression won't work as expected. Hence you need to drop one column.

NOTE: sklearn library takes care of dummy variable trap hence even if you don't drop one of the state columns it is going to work, however we should make a habit of taking care of dummy variable trap ourselves just in case library that you are using is not handling this for you

In [6]:
```
df_dummies.drop('Chennai',axis='columns',inplace=True)
df_dummies
```

Out[6]:

	area	price	Bangalore	Hyderabad
0	2600	5500000	0	0
1	3000	5650000	0	0
2	3200	6100000	0	0
3	3600	6800000	0	0
4	2600	5850000	1	0
5	2800	6150000	1	0
6	3300	6500000	1	0
7	3600	7100000	1	0
8	2600	5750000	0	1
9	2900	6000000	0	1
10	3100	6200000	0	1
11	3600	6950000	0	1

Everything in a single line

In [7]:
```
df_dum = pd.get_dummies(df,drop_first=True)
df_dum
```

Out[7]:

	area	price	town_Chennai	town_Hyderabad
0	2600	5500000	1	0
1	3000	5650000	1	0
2	3200	6100000	1	0
3	3600	6800000	1	0
4	2600	5850000	0	0
5	2800	6150000	0	0
6	3300	6500000	0	0
7	3600	7100000	0	0
8	2600	5750000	0	1
9	2900	6000000	0	1
10	3100	6200000	0	1
11	3600	6950000	0	1

OneHotEncoder ---> Nominal Variable encoding using sklearn

In [8]:
```
### OneHotEncoding: We use the OneHotEncoder from sklearn library
from sklearn.preprocessing import OneHotEncoder

### Call the function
enc = OneHotEncoder(drop='first')

### fit_transform
enc.fit_transform(df[['town']])
```

Out[8]:
```
<12x2 sparse matrix of type '<class 'numpy.float64'>'
        with 8 stored elements in Compressed Sparse Row format>
```

In [9]:
```
### save to an array
enc_array = enc.fit_transform(df[['town']]).toarray()

### convert to a dataframe
enc_df = pd.DataFrame(enc_array)
```

In [10]:
```
# merge with main df
df_ohe = pd.concat([df,enc_df],axis='columns')

# drop the original variable
df_ohe.drop('town',axis='columns',inplace=True)

#finally
df_ohe
```

Out[10]:

	area	price	0	1
0	2600	5500000	1.0	0.0
1	3000	5650000	1.0	0.0
2	3200	6100000	1.0	0.0
3	3600	6800000	1.0	0.0
4	2600	5850000	0.0	0.0
5	2800	6150000	0.0	0.0
6	3300	6500000	0.0	0.0
7	3600	7100000	0.0	0.0
8	2600	5750000	0.0	1.0
9	2900	6000000	0.0	1.0
10	3100	6200000	0.0	1.0
11	3600	6950000	0.0	1.0

To convert Ordinal Categorical into Numeric

- 1.map using pandas
- 2.label Encoder using sklearn
- 3.Ordinal Encoder using sklearn

LabelEncoder ---> Ordinal Variable encoding using sklearn

- 1. convert to numeric as per alphabetical order
- 2. used for binary category variable

In [11]:
```
dfle=df.copy()
dfle
```

Out[11]:

	town	area	price
0	Chennai	2600	5500000
1	Chennai	3000	5650000
2	Chennai	3200	6100000
3	Chennai	3600	6800000
4	Bangalore	2600	5850000
5	Bangalore	2800	6150000
6	Bangalore	3300	6500000
7	Bangalore	3600	7100000
8	Hyderabad	2600	5750000
9	Hyderabad	2900	6000000
10	Hyderabad	3100	6200000
11	Hyderabad	3600	6950000

In [12]:
```
### import from sklearn library
from sklearn.preprocessing import LabelEncoder

### Call the function
le = LabelEncoder()

### fit_transform
dfle.town = le.fit_transform(dfle.town)

dfle
```

Out[12]:

	town	area	price
0	1	2600	5500000
1	1	3000	5650000
2	1	3200	6100000
3	1	3600	6800000
4	0	2600	5850000
5	0	2800	6150000
6	0	3300	6500000
7	0	3600	7100000
8	2	2600	5750000
9	2	2900	6000000
10	2	3100	6200000
11	2	3600	6950000

OrdinalEncoder ---> Ordinal Variable encoding using sklearn

- 1. convert to numeric as per given order (ascending order) in the function
- 2. used for multi category variable

In [13]:
```
df_oe = df.copy()

### import from sklearn library
from sklearn.preprocessing import OrdinalEncoder

### Call the function
oe = OrdinalEncoder(categories=[['Bangalore','Hyderabad','Chennai']])

### fit_transform
df_oe.town = oe.fit_transform(df_oe[["town"]])

df_oe
```

Out[13]:

	town	area	price
0	2.0	2600	5500000
1	2.0	3000	5650000
2	2.0	3200	6100000
3	2.0	3600	6800000
4	0.0	2600	5850000
5	0.0	2800	6150000
6	0.0	3300	6500000
7	0.0	3600	7100000
8	1.0	2600	5750000
9	1.0	2900	6000000
10	1.0	3100	6200000
11	1.0	3600	6950000

map() ---> Ordinal Variable encoding using pandas

- 1. convert to numeric as per your choice
- 2. used for multi category variable

In [14]:
```
df_m = df.copy()

df_m['town'] = df_m['town'].map({'Chennai':0 ,'Bangalore': 25,'Hyderabad': 10})
df_m
```

Out[14]:

	town	area	price
0	0	2600	5500000
1	0	3000	5650000
2	0	3200	6100000
3	0	3600	6800000
4	25	2600	5850000
5	25	2800	6150000
6	25	3300	6500000
7	25	3600	7100000
8	10	2600	5750000
9	10	2900	6000000
10	10	3100	6200000
11	10	3600	6950000