Categorical Encoding

Label Encoder

One hot Encoder

Importing Libraries

```
#importing the libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
```

This approach requires the category column to be of 'category' datatype. By default, a non-numerical column is of 'object' type.

Label-encoding is by using SciKit learn library.

```
# creating initial dataframe
bridge_types = ('Arch','Beam','Truss','Cantilever','Tied Arch','Suspension','Cable')
bridge_df = pd.DataFrame(bridge_types, columns=['Bridge_Types'])
# creating instance of labelencoder
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
bridge_df['Bridge_Types_Cat'] = labelencoder.fit_transform(bridge_df['Bridge_Types'])
bridge_df
```

	Bridge_Types	Bridge_Types_Cat
0	Arch	0
1	Beam	1
2	Truss	6

One-Hot Encoder

Though label encoding is straight but it has the disadvantage that the numeric values can be misinterpreted by algorithms as having some sort of hierarchy/order in them.

This ordering issue is addressed in another common alternative approach called 'One-Hot Encoding'.

In this strategy, each category value is converted into a new column and assigned a 1 or 0 (notation for true/false) value to the column.

OneHotEncoder from SciKit library only takes numerical categorical values, hence any value of string type should be label encoded before one hot encoded. So taking the dataframe from the previous example, we will apply OneHotEncoder on column Bridge_Types_Cat.

```
# creating instance of one-hot-encoder
enc = OneHotEncoder(handle_unknown='ignore')
# passing bridge-types-cat column (label encoded values of bridge_types)
enc_df = pd.DataFrame(enc.fit_transform(bridge_df[['Bridge_Types_Cat']]).toarray())
# merge with main df bridge_df on key values
bridge_df = bridge_df.join(enc_df)
bridge_df
```

	Bridge_Types	Bridge_Types_Cat	0	1	2	3	4	5	6
0	Arch	0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1	Beam	1	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2	Truss	6	0.0	0.0	0.0	0.0	0.0	0.0	1.0
3	Cantilever	3	0.0	0.0	0.0	1.0	0.0	0.0	0.0

Using dummies values approach:

This approach is more flexible because it allows encoding as many category columns as you would like and choose how to label the columns using a prefix.

Proper naming will make the rest of the analysis just a little bit easier.

```
# creating initial dataframe
bridge_types = ('Arch','Beam','Truss','Cantilever','Tied Arch','Suspension','Cable')
bridge_df = pd.DataFrame(bridge_types, columns=['Bridge_Types'])
# generate binary values using get_dummies
dum_df = pd.get_dummies(bridge_df, columns=["Bridge_Types"], prefix=["Type_is"] )
# merge with main df bridge_df on key values
bridge_df = bridge_df.join(dum_df)
bridge_df

```

	Bridge_Types	Type_is_Arch	Type_is_Beam	Type_is_Cable	Type_is_Cantilever	Type_is_Suspension	Type_is_Tied Arch
0	Arch	1	0	0	0	0	0
1	Beam	0	1	0	0	0	0

```
#reading the dataset
data=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/FODS2021/S9/insurance.csv")
data.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
encoder = OneHotEncoder(categories="auto" #learning the category automatically
                        ,drop="first", #creating k-1 category
                       sparse=False # this will return numpy array else it will return sparse matix
                        ,handle_unknown="error")  #helps to deal with unknown values

encoder.fit(data.drop(["age","bmi","charges"],axis=1))
```

```
OneHotEncoder(categories='auto', drop='first', dtype=<class 'numpy.float64'>,
              handle_unknown='error', sparse=False)
```

```
encoder.categories_
```

```
[array(['female', 'male'], dtype=object),
 array([0, 1, 2, 3, 4, 5]),
 array(['no', 'yes'], dtype=object),
 array(['northeast', 'northwest', 'southeast', 'southwest'], dtype=object)]
```

```
d1=encoder.transform(data.drop(["age","bmi","charges"],axis=1))
```

```
1 d1=encoder.transform(data.drop(["age","bmi","charges"],axis=1))
```

```
1 #converting into dataframe
2 variables = pd.DataFrame(d1,columns= encoder.get_feature_names(["sex","children","smoker","region"]))
```

```
1 #concatinating the two data
2 df= pd.concat([variables, data[["age","bmi","charges"]]],axis=1)
```

```
1 df
```

	sex_male	children_1	children_2	children_3	children_4	children_5	smoker_yes	region_northwest	region_southe
0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	
1	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
3	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
...	...	...	...	...	...	...	...	...	
1333	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	
1334	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1335	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1336	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1337	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	

1338 rows × 13 columns

```
1 import pandas as pd
2 from pandas import DataFrame
3 df = DataFrame({'category':['fruit','fruit','fruit','vegetables','vegetables','meat','meat'],
4                 'Origin':['United States','China','China','China','new Zealand','new Zealand','United States']
```

```
4                'Origin':['United States','China','China','China','new Zealand','new Zealand','United States'],
5                'fruit':['apple','pear','Strawberry','tomato','cucumber','Lamb','beef']})
6 df.head()
```

	category	Origin	fruit
0	fruit	United States	apple
1	fruit	China	pear
2	fruit	China	Strawberry
3	vegetables	China	tomato
4	vegetables	new Zealand	cucumber

```
1 df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/FODS2021/S9/indian_food.csv')
```

```
1 df
```

	name	ingredients	diet	prep_time	cook_time	flavor_profile	course
0	Balu shahi	Maida flour, yogurt, oil, sugar	vegetarian	45	25	sweet	dessert
1	Boondi	Gram flour, ghee, sugar	vegetarian	80	30	sweet	dessert
2	Gajar ka halwa	Carrots, milk, sugar, ghee, cashews, raisins	vegetarian	15	60	sweet	dessert
3	Ghevar	Flour, ghee, kewra, milk, clarified butter, su...	vegetarian	15	30	sweet	dessert
4	Gulab jamun	Milk powder, plain flour, baking powder, ghee,...	vegetarian	15	40	sweet	dessert

```
1 df.shape[1]
```

```
9
```

```
1 df.columns
```

```
Index(['name', 'ingredients', 'diet', 'prep_time', 'cook_time',
       'flavor_profile', 'course', 'state', 'region'],
      dtype='object')
```

Crosstab

Compute a simple cross-tabulation of two (or more) factors. By default computes a frequency table of the factors unless an array of values and an aggregation function are passed. implementing crostab on state & diet columns

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

Display how many vegetarian and nonvegetarian dishes are in each state

```
1 pd.crosstab(df['state'], df['diet'])
```

diet	non vegetarian	vegetarian
state		
-1	0	24
Andhra Pradesh	0	10
Assam	10	11
Bihar	0	3
Chhattisgarh	0	1
Goa	1	2
Gujarat	0	35
Haryana	0	1
Jammu & Kashmir	0	2
Karnataka	0	6
Kerala	1	7
Madhya Pradesh	0	2
Maharashtra	2	28
Manipur	1	1
NCT of Delhi	1	0
Nagaland	1	0
Odisha	0	7
Punjab	4	28
Rajasthan	0	6
Tamil Nadu	1	19
Telangana	1	4
Tripura	1	0

Tripura	1	0
Uttar Pradesh	0	9
Uttarakhand	0	1
West Bengal	5	19

```
pd.crosstab(df['state'], df['flavor_profile'])
```

flavor_profile	-1	bitter	sour	spicy	sweet
state					
-1	4	0	0	14	6
Andhra Pradesh	0	0	0	2	8
Assam	4	0	0	11	6
Bihar	0	0	0	2	1
Chhattisgarh	0	0	0	1	0
Goa	0	0	0	1	2
Gujarat	3	2	1	23	6

```
pd.crosstab(df['state'], df['course']).max()
```

```
2
```

Karnataka 1 0 0 2 3

```
pd.crosstab(df['state'], df['course']).max().mean()
```

```
15.5
```

```
pd.crosstab(df['state'], df['course']).max().min()
```

```
2
```

Kerala 3 3 0 1 3

```
pd.crosstab(df['state'], df['course']).max().median()
```

```
16.0
```

Punjab 2 2 0 25 3

```
pd.crosstab(df['state'], df['course']).max().std()
```

```
10.661457061146317
```

Telangana 0 0 0 1 4

```
pd.crosstab(df['state'], df['course']).max().var()
```

```
113.66666666666667
```

Uttarakhand	0	0	0	0	1
West Bengal	2	0	0	6	16

✓ 0s completed at 9:46 AM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.