Dharavath Ramdas

Github link: https://github.com/dharavathramdas101 (https://github.com/dharavathramdas101)
linkedin link: https://www.linkedin.com/in/dharavath-ramdas-a283aa213/ (https://www.linkedin.com/in/dharavath-ramdas-a283aa213/)

Label Encoder and OneHot Encoder in Python

Understanding Label Encoder, OneHot Encoder and Pandas dummy variables with examples

Machine Learning algorithms understand the numbers and not texts. Hence, all the “text” columns must be converted into “numerical” columns to make it understandable for the algorithm.

This is the story of transforming labels or categorical or text values into numbers or numerical values. In simple words,

Encoding is the process of transforming words into numbers

In Python, OneHot Encoding and Lebel Encoding are two methods for encoding the categorical columns into numerical columns. And these are part of one of the most commonly used Python library: Scikit-Learn

But wait, you don’t want to import Scikit-Learn in your notebook ??

No problem at all, ⚡ Pandas comes for your help.

Let us dive into this story of converting categorical variables into numerical ones so that ML algorithm understands it.

Categorical Data

Any dataset contains multiple columns containing numerical as well as categorical values.

In [22]:

```
data = {'Type of Column':['Numerical','Categorical'],"Data type":['int,Float','Category,Object']
        }
pd.DataFrame(data)
```

Out[22]:

	Type of Column	Data type
0	Numerical	int,Float
1	Categorical	Category,Object

In [ ]:

```
Categorical variables represent types of data which may be divided into groups. It has a limited and usually fixed number of possible valu

But, if this data is encoded into numerical values then only it can be processed in a machine learning algorithm.

Let us consider the below example to understand the encoding in a simple way.
```

In [6]:

```
import pandas as pd
countries = ["Germany","India","UK","Egypt","Iran"]
continents = ["Europe","Asia","Europe","Africa","Asia"]
code = ["G","I","U","E","N"]
d={"country": countries, "continent":continents, "code":code}
df = pd.DataFrame(d)
```

In [7]:

```
df
```

Out[7]:

	country	continent	code
0	Germany	Europe	G
1	India	Asia	I
2	UK	Europe	U
3	Egypt	Africa	E
4	Iran	Asia	N

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   country    5 non-null      object
 1   continent  5 non-null      object
 2   code       5 non-null      object
dtypes: object(3)
memory usage: 248.0+ bytes
```

Converting the data type of column “code” from object to the category

In [9]:

```
df['code'] = df.code.astype('category')
```

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype   
---  ------     --------------  -----   
 0   country    5 non-null      object  
 1   continent  5 non-null      object  
 2   code       5 non-null      category
dtypes: category(1), object(2)
memory usage: 425.0+ bytes
```

In [ ]:

With this example let us understand the encoding process.

Label Encoding in Python

Label encoding is a simple and straight forward approach. This converts each value in a categorical column into a numerical valu
e. Each value in a categorical column is called Label.

Label encoding: Assign a unique integer to each label based on alphabetical order

Let me show you how Label encoding works in python with the same above example,

In [11]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df["labeled_continent"] = le.fit_transform(df["continent"])
df
```

Out[11]:

	country	continent	code	labeled_continent
0	Germany	Europe	G	2
1	India	Asia	I	1
2	UK	Europe	U	2
3	Egypt	Africa	E	0
4	Iran	Asia	N	1

the labels in column continent will be converted into numbers and will be stored in the new column – labeled_continent

The output

In more simple words, labels are arranged in alphabetical order and a unique index is assigned to each label starting from 0.

In [18]:

```
data = {"Labels":["Labels after encoding"],"Africa":[0],"Asia":[1],"Europe":[2]}
```

In [19]:

```
pd.DataFrame(data)
```

Out[19]:

	Labels	Africa	Asia	Europe
0	Labels after encoding	0	1	2

All looks good ?? Worked well ⁉

Here jumps in the problem with Label encoding. It uses numbers in a sequence that introduces a comparison between the labels. In the above example, the labels in the column continent do not have an order or rank. But after label encoding, these labels are or dered in an alphabetical manner. because of these numbers, a machine learning model can interpret this ordering as Europe > Asia > Africa

To overcome this ordering problem with Label Encoding, OneHot Encoding comes into the picture.

OneHot Encoding in Python

In OneHot encoding, a binary column is created for each label in a column. Here each label is transformed into a new column or ne w feature and assigned 1 (Hot) or 0 (Cold) value.

Let me show you an example first to understand the above statement,

In [20]:

```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder()
df3 = pd.DataFrame(ohe.fit_transform(df[["continent"]]).toarray())
df_new=pd.concat([df,df3],axis=1)
df_new
```

Out[20]:

	country	continent	code	labeled_continent	0	1	2
0	Germany	Europe	G	2	0.0	0.0	1.0
1	India	Asia	I	1	0.0	1.0	0.0
2	UK	Europe	U	2	0.0	0.0	1.0
3	Egypt	Africa	E	0	1.0	0.0	0.0
4	Iran	Asia	N	1	0.0	1.0	0.0

In this scenario, the last three columns are the result of OneHot Encoding. Labels Africa, Asia, and Europe have been encoded as 0, 1, 2 respectively. OneHot encoding transforms these labels into columns. hence, looking at the last 3 columns, we have 3 label s → 3 columns

OneHot Encoding: In a single row only one Label is Hot

In a particular row, only one label has a value of 1 and all other labels have a value of 0. Before feeding such an encoded datas et into a machine learning model few more transformations can be done as given in OneHot Encoding

pandas.get_dummies() in Python

OneHot encoding can be implemented in a simpler way and without importing Scikit-Learn.

⚡ Yess !! Pandas is your friend here. This simple function pandas.get_dummies() will quickly transform all the labels from sp ecified column into individual binary columns

In [21]:

```
df2=pd.get_dummies(df[["continent"]])
df_new=pd.concat([df,df2],axis=1)
df_new
```

Out[21]:

	country	continent	code	labeled_continent	continent_Africa	continent_Asia	continent_Europe
0	Germany	Europe	G	2	0	0	1
1	India	Asia	I	1	0	1	0
2	UK	Europe	U	2	0	0	1
3	Egypt	Africa	E	0	1	0	0
4	Iran	Asia	N	1	0	1	0

The last 3 columns of above DataFrame are the same as observed in OneHot Encoding.

pandas.get_dummies() generates dummy variables for each label in the column continent. Hence, continent_Africa, continent_Asia, a nd continent_Europe are the dummy binary variables for the labels Africa, Asia, and Europe respectively.

In [ ]: