Name: Ankit Sharma

Email: kumarankitx022@gmail.com

Mob no.: +91-7677241423

ASSIGNMENT – 03

COLUMN TRANSFORMER:

Applies transformers to columns of an array or pandas DataFrame.

This estimator allows different columns or column subsets of the input to be transformed separately and the features generated by each transformer will be concatenated to form a single feature space. This is useful for heterogeneous or columnar data, to combine several feature extraction mechanisms or transformations into a single transformer.

The ColumnTransformer constructor takes quite a few arguments,

- **Name**: a name for the column transformer, which will make setting of parameters and searching of the transformer easy.
- **Transformer**: here we're supposed to provide an estimator. We can also just "passthrough" or "drop" if we want. But since we're encoding the data we'll use the OneHotEncoder here. Remember that the estimator you use here needs to support fit and transform.
- **Column(s)**: the list of columns which you want to be transformed.

Creating an object for ColumnTransformer:

```
cOlumnTransformer = ColumnTransformer([('encoder', OneHotEncoder(), [0])], remainder = 'passthrough').
```

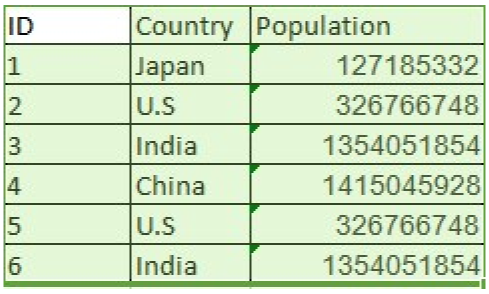
As you can see from the snippet above, we'll name the transformer simply "encoder." We're using the OneHotEncoder() constructor to provide a new instance as the estimator. And then we're specifying that only the first column has to be transformed.

ONE HOT ENCODER (OneHotEncoder):

What one hot encoding does is, it takes a column which has categorical data, which has been label encoded or column transformed and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.

It simply creates additional features based on the number of unique values in the categorical feature. Every unique value in the category will be added as a feature. In other words, it creates dummy variables making the value in the category as features.

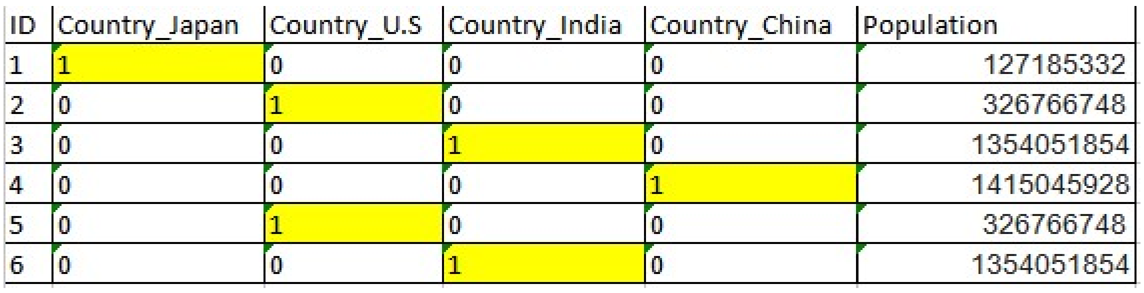
Consider below example:



ID	Country	Population
1	Japan	127185332
2	U.S	326766748
3	India	1354051854
4	China	1415045928
5	U.S	326766748
6	India	1354051854

In our example, on applying OneHotEncoder we'll get four new columns, one for each country- Japan, U.S, India, and China.

For rows which have the first column value as Japan, the 'Japan' column will have a '1' and the other three columns will have '0's. Similarly, for rows which have the first column value as the U.S, the 'U.s' column will have a '1' and the other three columns will have '0's and so on.



ID	Country_Japan	Country_U.S	Country_India	Country_China	Population
1	1	0	0	0	127185332
2	0	1	0	0	326766748
3	0	0	1	0	1354051854
4	0	0	0	1	1415045928
5	0	1	0	0	326766748
6	0	0	1	0	1354051854

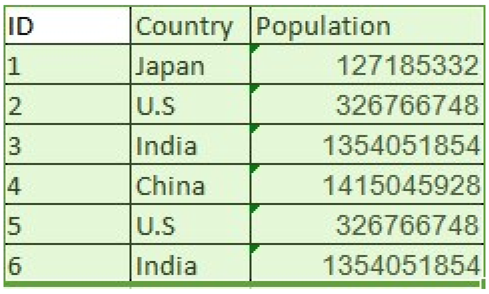
LABEL ENCODER:

Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

Label Encoding in Python can be achieved using Sklearn Library. Sklearn provides a very efficient tool for encoding the levels of categorical features into numeric values. LabelEncoder encode labels with a value between 0 and n_classes-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier.

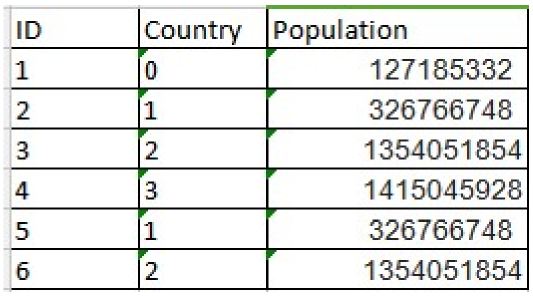
- Encode target labels with value between 0 and n_classes-1.
- This transformer should be used to encode target values, *i.e.* y, and not the input X.

Consider below example:



ID	Country	Population
1	Japan	127185332
2	U.S	326766748
3	India	1354051854
4	China	1415045928
5	U.S	326766748
6	India	1354051854

This time we will try encoding this data using Label Encoder. On applying Label Encoder we will get a result as seen below,



ID	Country	Population
1	0	127185332
2	1	326766748
3	2	1354051854
4	3	1415045928
5	1	326766748
6	2	1354051854

That's all label encoding is about. But depending on the data, label encoding introduces a new problem. For example, we have encoded a set of country names into numerical data. This is actually categorical data and there is no relation, of any kind, between the rows.

The problem here is since there are different numbers in the same column, the model will misunderstand the data to be in some kind of order, 0 < 1 <2.

The model may derive a correlation like as the country number increases the population increases but this clearly may not be the scenario in some other data or the prediction set. To overcome this problem, we use One Hot Encoder.

When to use a Label Encoding vs. One Hot Encoding:

This question generally depends on your dataset and the model which you wish to apply. But still, a few points to note before choosing the right encoding technique for your model:

We apply One-Hot Encoding when:

1. The categorical feature is **not ordinal** (like the countries above)
2. The number of categorical features is less so one-hot encoding can be effectively applied

We apply Label Encoding when:

1. The categorical feature is **ordinal** (like Jr. kg, Sr. kg, Primary school, high school)
2. The number of categories is quite large as one-hot encoding can lead to high memory consumption