# Assignment - 8

Title - Perform Encoding categorical feature on given dataset

Theory

Overview

- Understand what is categorical Data Encoding
- Learn different encoding technique & when to use them

Introduction

The performace of a machine learning model not only depends on the model & the hyper parameters but also on how we process & feed different types of variables to the model. Since most machine learning models only accept numerical variables, preprocessing the categorical variables becomes a necessary step. We need to convert these categorical variables to no such that the model is able to understand & extract valuable information. A typical data scientist spends 70-80% of his time cleaning and preparing the data. And converting categorical data is an ~~preparing the data. And so~~ an unavoidable activity. It not only elevates the model quality but also helps in better feature engineering.

Categorical data

Since we are going to be working on categorical variables, here is a quick refresher Categorical variables are usually represented as 'strings or 'categories' & are finite in no., Here ar ex.

1. The city where a person lives: Delhi, Mumbai, etc
2. The grades of a student : A+, A, B+, B, B-, etc

In above ex. the variables only have definite possible values. there are 2 kind of categorical data

- Ordinal data : The categories have an inherent order.

. Nominal Data : The categories donot have an inherent order

In ordinal data, while endoding, one should retain the information regarding the order in which the category is provided.

While encoding Nominal data, we have to consider the presence or absence of a feature. In such as case, no notion of order is present.

For encoding categorical data, we have a python package category-encoders

Label Encoding or ordinal Encoding - We use this categorical data encoding techique when categorical feature is ordinal. In this case, retaining the order is important, Hence encording should reflect the sequence.
In label encoding, each label is converted into an integer value, We will create a variable that contains the categories representing the education qualification of a person.

One Hot Encoding
We use this categorical data encoding technique when the features are nominal (do not have any order) In one hot encoding, for each level of a category. feature, we create a new variable, Each category is mapped with a binary variable containing, either 0 or 1. Here, 0 represents the absence, & 1 represents the presence of the category. These newly created binary features are known as Dummy variables. The no. of dummy variables depends on the levels present in the categorical variables This might sound complicated.

Dummy Encoding

Dummy coding scheme is similar to one-hot encoding. This categorical data encoding method transforms the categorical variables into set of

binary variables. The dummy encoding is a small improvement over one-hot-encoding. Dummy encoding uses N-1 features to represent N labels/categories.

Drawbacks of one-Hot & Dummy Encoding

One hot encoding & dummy encoder are two powerful & effective encoding schemes. They are also very popular among the data scientists, But may not be as effective when

1. A large no. of levels are present in data. If there are multiple categories in a feature variable in such a case we need a similar no. of dummy variables to encode the data.
2. If we have multiple categorical features in the dataset similar situation will occur & again we will end to have several binary feature each representing the categorical feature & their multiple categories eg. a dataset having 10 or more categorical columns.

In both the above cases, these two encoding schemes introduce sparsity in the data set ie. several columns having 0s & a few of them having 1s
Also they might lead to a Dummy variable trap. It is a phenomenon where features are highly correlated Due to the massive increase in data set, coding slows down the learning of model along with deteriorating the overall performance that ultimately makes the model compatationally expensive

Effect Encoding:

This encoding technique is also known as Deviation Encoding or Sum Encoding. Effect encoding is almost similar to dummy encoding, with a little difference. In dummy coding, we use 0 & 1 to represent the data but in effect encoding, we use three values i.e 1, 0 & -1.

Hash Encoder -

To understand Hash encoding it is necessary to know about Hashing. Hashing is the transformation of arbitrary size input in the form of fixed-size value. We use hashing algorithm to perform hashing operation ie to generate the hash value of an input. Further, hashing is a one way process.

Hashing has several application like data retrieval, checking data corruption & in data encryption also, we have multiple has function ~~are~~ available for ex. Message Digest (MD, MD2, MD5), Secure Hash Function (SHA0, HA1, SHA2) & Many more.

Just like one-hot encoding, the Hash encoder represents categorical features using the new dimension, Here, the user can fix no. of dimensions after transformation using n-component argument.

Since, Hashing tranforms the data in lesser dimensions it may lead to loss of information. Another issue faced by hashing encoder is the collision. Since, here, a large no of features are depicted into lesser dimensions, hence multiple values can be represented by same hash value, this is known as a collision

Moreover, hashing encoders have been very successful in some kaggle competitions. It is great to toy if the data set has high cardinality features.

Binary Encoding:

It is a compitation of Hash encoding & one Hot encoding. In this encoding scheme, then categorical feature is 1st converted into numerical using an ordinal encoder. Then the no. are transformed in the binary no. After that binary value is split into different columns.

Binary encoding is a memory efficient encoding scheme as it uses fewer features than one-hot encoding. Further, it reduces the curse of dimensionality for data with high cardinality.

Base N Encoding:

In numeral system, the Base or the radix is the no of digits or a

Combination of digits & letters used to represent the no. The most common base we use in our life is 10 or decimal system as there we use 10 unique digits, ie 0 to 9 to represent all the no. Another widely used system is binary. i.e. the base is 2. It uses 0 & 1 ie. 2 digits to express all nos.

For Binary encoding, the Base is 2 which means it converts from. In the case when categories are more and binary encoding is not able to handle the dimensionality then we can use a larger base such as 4 or 8.

Base N encoding technique further reduces the no of feature required to efficiently represent the data & improving memory usage. The default Base for Base N is 2 which is equivalent to Binary encoding.

Target Encoding -
Target encoding is a Bayesian encoding technique Bayesian encoders use information from dependent / target variables to encode ~~we calculate~~ the categorical data.
In target encoding, we calculate the mean of the target variable for each calculate the categorical data.
In target encoding, we calculate the mean of the target variable for each category & replace the category variable with the mean value. In the case of categorical target variables, the posterior probability of the target replaces each category.
We perform target encoding for train data only and code the test data using results obtained from the training dataset. Although, a very efficient coding system, it has following issue responsible for deteriorating the model performance -
1. It can lead to target leakage or overfitting. to address overfitting we can use different technique.

1. In the leave one out encoding, the current target value is reduced from the overall mean of the target to avoid leakage.

2. In another method, we may introduce some Gaussian noise in the target statistics. The value of this noise is hyperparameter to the model.

2. The second issue, we may face is the improper distribution of categories in train & test data. In such case the categories may assume extreme values. Therefore the target means for category are mixed with the marginal mean of target.

Conclusion:
To summarize, encoding categorical data is an unavoidable part of feature engineering. It is more important to know what encoding scheme we should use. Having into consideration the data set we are working with & model we are going to use. We have seen various encoding techniques along with their issue & suitable use cases.

Installing python package category_encoders

In [1]:
```
pip install category_encoders
```

```
Requirement already satisfied: category_encoders in c:\users\prati\anaconda3\lib\site-pa
ckages (2.3.0)
Requirement already satisfied: numpy>=1.14.0 in c:\users\prati\anaconda3\lib\site-packag
es (from category_encoders) (1.19.2)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\prati\anaconda3\lib\site
-packages (from category_encoders) (0.23.2)
Requirement already satisfied: patsy>=0.5.1 in c:\users\prati\anaconda3\lib\site-package
s (from category_encoders) (0.5.1)
Requirement already satisfied: pandas>=0.21.1 in c:\users\prati\anaconda3\lib\site-packa
ges (from category_encoders) (1.1.3)
Requirement already satisfied: scipy>=1.0.0 in c:\users\prati\anaconda3\lib\site-package
s (from category_encoders) (1.5.2)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\prati\anaconda3\lib\site-p
ackages (from category_encoders) (0.12.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\prati\anaconda3\lib\site
-packages (from scikit-learn>=0.20.0->category_encoders) (2.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\prati\anaconda3\lib\site-package
s (from scikit-learn>=0.20.0->category_encoders) (0.17.0)
Requirement already satisfied: six in c:\users\prati\anaconda3\lib\site-packages (from p
atsy>=0.5.1->category_encoders) (1.15.0)
Requirement already satisfied: pytz>=2017.2 in c:\users\prati\anaconda3\lib\site-package
s (from pandas>=0.21.1->category_encoders) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\prati\anaconda3\lib\si
te-packages (from pandas>=0.21.1->category_encoders) (2.8.1)
Note: you may need to restart the kernel to use updated packages.
```

**Label Encoding or Ordinal Encoding**

In [2]:
```
import category_encoders as ce
import pandas as pd
train_df=pd.DataFrame({'Degree':['High school','Masters','Diploma','Bachelors','Bachelo

# create object of Ordinalencoding
encoder= ce.OrdinalEncoder(cols=['Degree'],return_df=True,
                           mapping=[{'col':'Degree',
'mapping':{'None':0,'High school':1,'Diploma':2,'Bachelors':3,'Masters':4,'phd':5}}])

#Original data
train_df
```

Out[2]:

| | Degree |
|---|---|
| **0** | High school |
| **1** | Masters |
| **2** | Diploma |
| **3** | Bachelors |
| **4** | Bachelors |
| **5** | Masters |
| **6** | Phd |
| **7** | High school |
| **8** | High school |

In [3]:
```
#fit and transform train data
encoder.fit_transform(train_df)
```

Out[3]:

| | Degree |
|---|---|
| **0** | 1.0 |
| **1** | 4.0 |
| **2** | 2.0 |
| **3** | 3.0 |
| **4** | 3.0 |
| **5** | 4.0 |
| **6** | -1.0 |
| **7** | 1.0 |
| **8** | 1.0 |

### One Hot Encoding

In [4]:
```
import category_encoders as ce
import pandas as pd
data=pd.DataFrame({'City':[
'Delhi','Mumbai','Hydrabad','Chennai','Bangalore','Delhi','Hydrabad','Bangalore','Delhi
]})

#Create object for one-hot encoding
encoder=ce.OneHotEncoder(cols='City',handle_unknown='return_nan',return_df=True,use_cat

#Original Data
data
```

Out[4]:

| | City |
|---|---|
| **0** | Delhi |
| **1** | Mumbai |
| **2** | Hydrabad |
| **3** | Chennai |
| **4** | Bangalore |
| **5** | Delhi |
| **6** | Hydrabad |
| **7** | Bangalore |
| **8** | Delhi |

In [5]:
```
#Fit and transform Data
data_encoded = encoder.fit_transform(data)
data_encoded
```

Out[5]:

| | City_Delhi | City_Mumbai | City_Hydrabad | City_Chennai | City_Bangalore |
|---|---|---|---|---|---|
| **0** | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| **2** | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| **3** | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **5** | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **6** | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| **7** | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **8** | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |

### Dummy Encoding

In [6]:
```
import category_encoders as ce
import pandas as pd
data=pd.DataFrame({'City':[
'Delhi','Mumbai','Hydrabad','Chennai','Bangalore','Delhi','Hydrabad','Bangalore','Delhi
]})
#Original Data
data
```

Out[6]:

| | City |
|---|---|
| **0** | Delhi |
| **1** | Mumbai |
| **2** | Hydrabad |
| **3** | Chennai |
| **4** | Bangalore |
| **5** | Delhi |
| **6** | Hydrabad |
| **7** | Bangalore |
| **8** | Delhi |

In [7]:
```
#encode the data
data_encoded=pd.get_dummies(data=data,drop_first=True)
data_encoded
```

Out[7]:

| | City_Chennai | City_Delhi | City_Hydrabad | City_Mumbai |
|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 0 |
| **1** | 0 | 0 | 0 | 1 |
| **2** | 0 | 0 | 1 | 0 |
| **3** | 1 | 0 | 0 | 0 |

| | City_Chennai | City_Delhi | City_Hydrabad | City_Mumbai |
|---|---|---|---|---|
| **4** | 0 | 0 | 0 | 0 |
| **5** | 0 | 1 | 0 | 0 |
| **6** | 0 | 0 | 1 | 0 |
| **7** | 0 | 0 | 0 | 0 |
| **8** | 0 | 1 | 0 | 0 |

### Effect Encoding:

In [8]:
```
import category_encoders as ce
import pandas as pd
data=pd.DataFrame({'City':[
'Delhi','Mumbai','Hydrabad','Chennai','Bangalore','Delhi','Hydrabad','Bangalore','Delhi

encoder=ce.sum_coding.SumEncoder(cols='City',verbose=False)

#Original Data
data
```

Out[8]:

| | City |
|---|---|
| **0** | Delhi |
| **1** | Mumbai |
| **2** | Hydrabad |
| **3** | Chennai |
| **4** | Bangalore |
| **5** | Delhi |
| **6** | Hydrabad |
| **7** | Bangalore |
| **8** | Delhi |

In [9]:
```
encoder.fit_transform(data)
```

Out[9]:

| | intercept | City_0 | City_1 | City_2 | City_3 |
|---|---|---|---|---|---|
| **0** | 1 | 1.0 | 0.0 | 0.0 | 0.0 |
| **1** | 1 | 0.0 | 1.0 | 0.0 | 0.0 |
| **2** | 1 | 0.0 | 0.0 | 1.0 | 0.0 |
| **3** | 1 | 0.0 | 0.0 | 0.0 | 1.0 |
| **4** | 1 | -1.0 | -1.0 | -1.0 | -1.0 |
| **5** | 1 | 1.0 | 0.0 | 0.0 | 0.0 |
| **6** | 1 | 0.0 | 0.0 | 1.0 | 0.0 |
| **7** | 1 | -1.0 | -1.0 | -1.0 | -1.0 |

| | intercept | City_0 | City_1 | City_2 | City_3 |
|---|---|---|---|---|---|
| **8** | 1 | 1.0 | 0.0 | 0.0 | 0.0 |

### Hash Encoder

In [10]:
```
import category_encoders as ce
import pandas as pd

#Create the dataframe
data=pd.DataFrame({'Month':['January','April','March','April','Februay','June','July','
#Create object for hash encoder
encoder=ce.HashingEncoder(cols='Month',n_components=6)
data
```

Out[10]:

| | Month |
|---|---|
| **0** | January |
| **1** | April |
| **2** | March |
| **3** | April |
| **4** | Februay |
| **5** | June |
| **6** | July |
| **7** | June |
| **8** | September |

In [11]:
```
#Fit and Transform Data
encoder.fit_transform(data)
```

Out[11]:

| | col_0 | col_1 | col_2 | col_3 | col_4 | col_5 |
|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 1 | 0 |
| **1** | 0 | 0 | 0 | 1 | 0 | 0 |
| **2** | 0 | 0 | 0 | 0 | 1 | 0 |
| **3** | 0 | 0 | 0 | 1 | 0 | 0 |
| **4** | 0 | 0 | 0 | 1 | 0 | 0 |
| **5** | 0 | 1 | 0 | 0 | 0 | 0 |
| **6** | 1 | 0 | 0 | 0 | 0 | 0 |
| **7** | 0 | 1 | 0 | 0 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 | 1 | 0 |

### Binary Encoding

In [12]:
```
#Import the libraries
```

```
import category_encoders as ce
import pandas as pd

#Create the Dataframe
data=pd.DataFrame({'City':['Delhi','Mumbai','Hyderabad','Chennai','Bangalore','Delhi','

#Create object for binary encoding
encoder= ce.BinaryEncoder(cols=['City'],return_df=True)

#Original Data
data
```

Out[12]:

| | City |
|---|---|
| **0** | Delhi |
| **1** | Mumbai |
| **2** | Hyderabad |
| **3** | Chennai |
| **4** | Bangalore |
| **5** | Delhi |
| **6** | Hyderabad |
| **7** | Mumbai |
| **8** | Agra |

In [13]:
```
data_encoded = encoder.fit_transform(data)
data_encoded
```

Out[13]:

| | City_0 | City_1 | City_2 |
|---|---|---|---|
| **0** | 0 | 0 | 1 |
| **1** | 0 | 1 | 0 |
| **2** | 0 | 1 | 1 |
| **3** | 1 | 0 | 0 |
| **4** | 1 | 0 | 1 |
| **5** | 0 | 0 | 1 |
| **6** | 0 | 1 | 1 |
| **7** | 0 | 1 | 0 |
| **8** | 1 | 1 | 0 |

### Base N Encoding

In [14]:
```
#Import the libraries
import category_encoders as ce
import pandas as pd

#Create the dataframe
data=pd.DataFrame({'City':['Delhi','Mumbai','Hyderabad','Chennai','Bangalore','Delhi','
```

```
#Create an object for Base N Encoding
encoder= ce.BaseNEncoder(cols=['City'],return_df=True,base=5)

#Original Data
data
```

Out[14]:

| | City |
|---|---|
| **0** | Delhi |
| **1** | Mumbai |
| **2** | Hyderabad |
| **3** | Chennai |
| **4** | Bangalore |
| **5** | Delhi |
| **6** | Hyderabad |
| **7** | Mumbai |
| **8** | Agra |

In [15]:
```
#Fit and Transform Data
data_encoded=encoder.fit_transform(data)
data_encoded
```

Out[15]:

| | City_0 | City_1 |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 0 | 2 |
| **2** | 0 | 3 |
| **3** | 0 | 4 |
| **4** | 1 | 0 |
| **5** | 0 | 1 |
| **6** | 0 | 3 |
| **7** | 0 | 2 |
| **8** | 1 | 1 |

### Target Encoding

In [16]:
```
#import the libraries
import pandas as pd
import category_encoders as ce

#Create the Dataframe
data=pd.DataFrame({'class':['A,','B','C','B','C','A','A','A'],'Marks':[50,30,70,80,45,9

#Create target encoding object
encoder=ce.TargetEncoder(cols='class')
```

```
#Original Data
data
```

Out[16]:

| | class | Marks |
|---|---|---|
| **0** | A, | 50 |
| **1** | B | 30 |
| **2** | C | 70 |
| **3** | B | 80 |
| **4** | C | 45 |
| **5** | A | 97 |
| **6** | A | 80 |
| **7** | A | 68 |

In [17]:

```
#Fit and Transform Train Data
encoder.fit_transform(data['class'],data['Marks'])
```

Out[17]:

| | class |
|---|---|
| **0** | 65.000000 |
| **1** | 57.689414 |
| **2** | 59.517061 |
| **3** | 57.689414 |
| **4** | 59.517061 |
| **5** | 79.679951 |
| **6** | 79.679951 |
| **7** | 79.679951 |

In [ ]: