Data Encoding

May 21, 2023

1 1) Data Encoding

- Encoding is a technique of converting categorical variables into numerical values so that it could be easily fitted to a machine learning model.

1.1 1.1) Nominal/One Hot Encoding

[1]:
```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
```

[2]:
```
df=pd.DataFrame({'color':['red','blue','green','red','blue']})
```

[3]:
```
df
```

[3]:
```
   color
0    red
1   blue
2  green
3    red
4   blue
```

[4]:
```
encoder=OneHotEncoder()
```

[5]:
```
ans=encoder.fit_transform(df[['color']]).toarray()
```

[6]:
```
ans
```

[6]:
```
array([[0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.]])
```

[7]:
```
onehot=pd.DataFrame(ans,columns=encoder.get_feature_names_out())
```

[8]:
```
onehot
```

```
[8]:    color_blue  color_green  color_red
     0         0.0          0.0        1.0
     1         1.0          0.0        0.0
     2         0.0          1.0        0.0
     3         0.0          0.0        1.0
     4         1.0          0.0        0.0
```

```
[9]: pd.concat([df,onehot],axis=1)
```

```
[9]:    color  color_blue  color_green  color_red
     0    red         0.0          0.0        1.0
     1   blue         1.0          0.0        0.0
     2  green         0.0          1.0        0.0
     3    red         0.0          0.0        1.0
     4   blue         1.0          0.0        0.0
```

1.2 1.2) Label Encoding

```
[10]: import numpy as np
      import seaborn as sns
      from sklearn.preprocessing import LabelEncoder
      import warnings
      warnings.filterwarnings('ignore')
```

```
[11]: df=sns.load_dataset('iris')
```

```
[12]: df
```

```
[12]:      sepal_length  sepal_width  petal_length  petal_width    species
      0             5.1          3.5           1.4          0.2     setosa
      1             4.9          3.0           1.4          0.2     setosa
      2             4.7          3.2           1.3          0.2     setosa
      3             4.6          3.1           1.5          0.2     setosa
      4             5.0          3.6           1.4          0.2     setosa
      ..            …            …             …            …          …
      145           6.7          3.0           5.2          2.3  virginica
      146           6.3          2.5           5.0          1.9  virginica
      147           6.5          3.0           5.2          2.0  virginica
      148           6.2          3.4           5.4          2.3  virginica
      149           5.9          3.0           5.1          1.8  virginica

      [150 rows x 5 columns]
```

```
[13]: species=df['species']
      species
```

```
[13]: 0         setosa
      1         setosa
      2         setosa
      3         setosa
      4         setosa
               …
      145    virginica
      146    virginica
      147    virginica
      148    virginica
      149    virginica
      Name: species, Length: 150, dtype: object
```

```
[14]: ans=pd.DataFrame(species)
```

```
[15]: ans
```

```
[15]:        species
      0       setosa
      1       setosa
      2       setosa
      3       setosa
      4       setosa
      ..         …
      145  virginica
      146  virginica
      147  virginica
      148  virginica
      149  virginica

      [150 rows x 1 columns]
```

```
[16]: encoder=LabelEncoder()
```

```
[17]: df1=encoder.fit_transform(ans[['species']])
```

```
[18]: df1
```

```
[18]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
[19]: df2=pd.DataFrame(df1,columns=["Code"])
```

[20]:
```
df2
```

[20]:
```
     Code
0       0
1       0
2       0
3       0
4       0
..    …
145     2
146     2
147     2
148     2
149     2

[150 rows x 1 columns]
```

[21]:
```
pd.concat([df,df2],axis=1)
```

[21]:
```
     sepal_length  sepal_width  petal_length  petal_width    species  Code
0             5.1          3.5           1.4          0.2     setosa     0
1             4.9          3.0           1.4          0.2     setosa     0
2             4.7          3.2           1.3          0.2     setosa     0
3             4.6          3.1           1.5          0.2     setosa     0
4             5.0          3.6           1.4          0.2     setosa     0
..            …            …           …           …        …   …
145           6.7          3.0           5.2          2.3  virginica     2
146           6.3          2.5           5.0          1.9  virginica     2
147           6.5          3.0           5.2          2.0  virginica     2
148           6.2          3.4           5.4          2.3  virginica     2
149           5.9          3.0           5.1          1.8  virginica     2

[150 rows x 6 columns]
```

1.3 1.3) Ordinal Encoding

[22]:
```
df=pd.DataFrame({
'size':['small','medium','large','medium','small','large']})
```

[23]:
```
df
```

[23]:
```
     size
0   small
1  medium
2   large
3  medium
4   small
```

```
5   large
```

[24]:
```
from sklearn.preprocessing import OrdinalEncoder
```

[25]:
```
ords=OrdinalEncoder(categories=[['small','medium','large']])
```

[26]:
```
ans=ords.fit_transform(df[['size']])
```

[27]:
```
df1=pd.DataFrame(ans,columns=['code'])
```

[28]:
```
df1
```

[28]:
```
   code
0   0.0
1   1.0
2   2.0
3   1.0
4   0.0
5   2.0
```

[29]:
```
pd.concat([df,df1],axis=1)
```

[29]:
```
     size  code
0   small   0.0
1  medium   1.0
2   large   2.0
3  medium   1.0
4   small   0.0
5   large   2.0
```

1.4 1.4) Target Guided Ordinal Encoding

[30]:
```
import pandas as pd

# create a sample dataframe with a categorical variable and a target variable
df = pd.DataFrame({
    'city': ['New York', 'London', 'Paris', 'Tokyo', 'New York', 'Paris'],
    'price': [200, 150, 300, 250, 180, 320]
})
```

[31]:
```
## calculate the mean price for each city
mean_price=df.groupby('city')['price'].mean().to_dict()
mean_price
```

[31]:
```
{'London': 150.0, 'New York': 190.0, 'Paris': 310.0, 'Tokyo': 250.0}
```

[32]:

```python
## replace each city with its mean price
df['city_encoded']=df['city'].map(mean_price)
```

[33]:

```python
df
```

[33]:

```
       city  price  city_encoded
0  New York    200         190.0
1    London    150         150.0
2     Paris    300         310.0
3     Tokyo    250         250.0
4  New York    180         190.0
5     Paris    320         310.0
```