DataEncoding

March 20, 2023

1 Data Encoding

1.1 OneHotEncoder

1.1.1 One hot encoding is a technique used in machine learning to represent categorical data as numerical vectors. Each category is assigned a unique binary value, with a 1 indicating the presence of that category and 0s indicating the absence of all other categories. This creates a sparse matrix where each row represents a data point and each column represents a category. One hot encoding is commonly used in natural language processing, where words are represented as one hot encoded vectors to be used as inputs to neural networks.

[1]:
```
import seaborn as sns
import pandas as pd
```

[2]:
```
df=sns.load_dataset("flights")
df
```

[2]:
```
     year month  passengers
0    1949   Jan         112
1    1949   Feb         118
2    1949   Mar         132
3    1949   Apr         129
4    1949   May         121
..    …     …           …
139  1960   Aug         606
140  1960   Sep         508
141  1960   Oct         461
142  1960   Nov         390
143  1960   Dec         432

[144 rows x 3 columns]
```

[3]:
```
years=pd.DataFrame(df["year"])
```

[4]:
```
years
```

[4]:
```
     year
0    1949
1    1949
2    1949
3    1949
4    1949
..    …
139  1960
140  1960
141  1960
142  1960
143  1960

[144 rows x 1 columns]
```

[5]:
```
from sklearn.preprocessing import OneHotEncoder
```

[6]:
```
one_hot=OneHotEncoder()
```

[7]:
```
ans=one_hot.fit_transform(years[["year"]]).toarray()
```

[8]:
```
ans
```

[8]:
```
array([[1., 0., 0., …, 0., 0., 0.],
       [1., 0., 0., …, 0., 0., 0.],
       [1., 0., 0., …, 0., 0., 0.],
       …,
       [0., 0., 0., …, 0., 0., 1.],
       [0., 0., 0., …, 0., 0., 1.],
       [0., 0., 0., …, 0., 0., 1.]])
```

[9]:
```
onehot=pd.DataFrame(ans,columns=one_hot.get_feature_names_out())
onehot
```

[9]:
```
     year_1949  year_1950  year_1951  year_1952  year_1953  year_1954  \
0          1.0        0.0        0.0        0.0        0.0        0.0
1          1.0        0.0        0.0        0.0        0.0        0.0
2          1.0        0.0        0.0        0.0        0.0        0.0
3          1.0        0.0        0.0        0.0        0.0        0.0
4          1.0        0.0        0.0        0.0        0.0        0.0
..         …          …          …          …          …          …
139        0.0        0.0        0.0        0.0        0.0        0.0
140        0.0        0.0        0.0        0.0        0.0        0.0
141        0.0        0.0        0.0        0.0        0.0        0.0
142        0.0        0.0        0.0        0.0        0.0        0.0
143        0.0        0.0        0.0        0.0        0.0        0.0
```

```
     year_1955  year_1956  year_1957  year_1958  year_1959  year_1960
0          0.0        0.0        0.0        0.0        0.0        0.0
1          0.0        0.0        0.0        0.0        0.0        0.0
2          0.0        0.0        0.0        0.0        0.0        0.0
3          0.0        0.0        0.0        0.0        0.0        0.0
4          0.0        0.0        0.0        0.0        0.0        0.0
..         …          …          …          …          …          …
139        0.0        0.0        0.0        0.0        0.0        1.0
140        0.0        0.0        0.0        0.0        0.0        1.0
141        0.0        0.0        0.0        0.0        0.0        1.0
142        0.0        0.0        0.0        0.0        0.0        1.0
143        0.0        0.0        0.0        0.0        0.0        1.0

[144 rows x 12 columns]
```

[10]:
```python
pd.concat([df,onehot],axis=1)
```

[10]:
```
     year month  passengers  year_1949  year_1950  year_1951  year_1952  \
0    1949   Jan         112        1.0        0.0        0.0        0.0
1    1949   Feb         118        1.0        0.0        0.0        0.0
2    1949   Mar         132        1.0        0.0        0.0        0.0
3    1949   Apr         129        1.0        0.0        0.0        0.0
4    1949   May         121        1.0        0.0        0.0        0.0
..    …     …           …          …          …          …          …
139  1960   Aug         606        0.0        0.0        0.0        0.0
140  1960   Sep         508        0.0        0.0        0.0        0.0
141  1960   Oct         461        0.0        0.0        0.0        0.0
142  1960   Nov         390        0.0        0.0        0.0        0.0
143  1960   Dec         432        0.0        0.0        0.0        0.0

     year_1953  year_1954  year_1955  year_1956  year_1957  year_1958  \
0          0.0        0.0        0.0        0.0        0.0        0.0
1          0.0        0.0        0.0        0.0        0.0        0.0
2          0.0        0.0        0.0        0.0        0.0        0.0
3          0.0        0.0        0.0        0.0        0.0        0.0
4          0.0        0.0        0.0        0.0        0.0        0.0
..         …          …          …          …          …          …
139        0.0        0.0        0.0        0.0        0.0        0.0
140        0.0        0.0        0.0        0.0        0.0        0.0
141        0.0        0.0        0.0        0.0        0.0        0.0
142        0.0        0.0        0.0        0.0        0.0        0.0
143        0.0        0.0        0.0        0.0        0.0        0.0

     year_1959  year_1960
0          0.0        0.0
1          0.0        0.0
2          0.0        0.0
```

```
3            0.0         0.0
4            0.0         0.0
..           ...         ...
139          0.0         1.0
140          0.0         1.0
141          0.0         1.0
142          0.0         1.0
143          0.0         1.0

[144 rows x 15 columns]
```

1.2 Label Encoding

1.2.1 Label encoding is a technique used in machine learning to convert categorical data into numerical data by assigning each unique category a numerical label. The labels are typically assigned in a sequential manner, with the first category receiving a label of 0, the second category receiving a label of 1, and so on. Label encoding is useful when working with algorithms that cannot handle categorical data directly, such as decision trees and random forests. However, it's important to note that label encoding can create an implicit ordinal relationship between the categories, which may not always be accurate or desirable.

[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]:
```
from sklearn.preprocessing import LabelEncoder
```

[17]:
```
label_encoder=LabelEncoder()
```

[18]:
```
df1=label_encoder.fit_transform(ans[["species"]])
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/_label.py:116:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[19]:
```
df1
```

[19]:
```
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])
```

[20]:
```
df2=pd.DataFrame(df1,columns=["code"])
```

[21]:
```
df2
```

[21]:

	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]

[22]:
```
pd.concat([df,df2],axis=1)
```

[22]:

	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 Ordinal Encoding

1.3.1 Ordinal encoding is a technique used in machine learning to convert categorical data into numerical data by assigning each unique category a numerical value based on its order or rank. Unlike label encoding, which assigns labels in a sequential manner, ordinal encoding assigns labels based on the relative order of the categories.

1.3.2 For example, if we have a categorical feature "Size" with the categories "Small", "Medium", and "Large", we can assign them labels of 0, 1, and 2, respectively, based on their order.

[23]:
```
# create a sample dataframe with an ordinal variable
df = pd.DataFrame({
    'size': ['small', 'medium', 'large', 'medium', 'small', 'large']
})
```

[24]:
```
df
```

[24]:

	size
0	small
1	medium
2	large
3	medium
4	small
5	large

[25]:
```
from sklearn.preprocessing import OrdinalEncoder
```

[26]:
```
ords=OrdinalEncoder(categories=[["large","medium","small"]])
```

[27]:
```
ans=ords.fit_transform(df[["size"]])
```

[28]:
```
answer=pd.DataFrame(ans,columns=["code"])
```

[29]:
```
answer
```

[29]:

	code
0	2.0
1	1.0
2	0.0
3	1.0
4	2.0
5	0.0

[30]:
```
pd.concat([df,answer],axis=1)
```

[30]:
```
     size  code
0   small   2.0
1  medium   1.0
2   large   0.0
3  medium   1.0
4   small   2.0
5   large   0.0
```

1.4 Target guided encoding

1.4.1 Target guided encoding is a technique used in machine learning to convert categorical data into numerical data by computing the mean or median target value for each category and assigning the resulting value as the label for that category. This method is similar to mean encoding or probability encoding, but it uses the target variable to guide the encoding process.

1.4.2 Target guided encoding can be useful when working with imbalanced datasets, as it can help to balance the distribution of the target variable across the categories. It can also be helpful in cases where the categories are highly correlated with the target variable, as it can capture this relationship more accurately than other encoding techniques.

[31]:
```
# Create a dictionary of sample data
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily'],
        'Age': [25, 30, 35, 40, 45],
        'Gender': ['Female', 'Male', 'Male', 'Male', 'Female'],
        'Income': [50000, 70000, 60000, 80000, 90000],
        'Education': ['Bachelors', 'Masters', 'Masters', 'PhD', 'PhD']}

# Convert the dictionary to a pandas DataFrame
df = pd.DataFrame(data)
```

[32]:
```
df
```

[32]:
```
      Name  Age  Gender  Income  Education
0    Alice   25  Female   50000  Bachelors
1      Bob   30    Male   70000    Masters
2  Charlie   35    Male   60000    Masters
3    David   40    Male   80000        PhD
4    Emily   45  Female   90000        PhD
```

[33]:
```
md=df.groupby("Education")["Income"].mean().to_dict()
```

[34]:
```
md
```

[34]:
```
{'Bachelors': 50000.0, 'Masters': 65000.0, 'PhD': 85000.0}
```

[35]:
```
df["Encoded"]=df["Education"].map(md)
```

[36]:

```
df
```

[36]:

	Name	Age	Gender	Income	Education	Encoded
0	Alice	25	Female	50000	Bachelors	50000.0
1	Bob	30	Male	70000	Masters	65000.0
2	Charlie	35	Male	60000	Masters	65000.0
3	David	40	Male	80000	PhD	85000.0
4	Emily	45	Female	90000	PhD	85000.0