feature-encoding

August 1, 2024

0.1 FEATURE ENCODING

Feature encoding in machine learning refers to the process of transforming categorical data into a numerical format that can be used by machine learning algorithms. Most algorithms require numerical input, so categorical variables (such as labels or text) need to be encoded into numbers to be used effectively

[2]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, OrdinalEncoder
from category_encoders import BinaryEncoder, CountEncoder
```

[3]:
```
df = pd.read_excel('/kaggle/input/train-2441656/train.xlsx')
```

Feature transformation:

Encoding categorical features (e.g., ‘Sex’, ‘Embarked’) using appropriate techniques (e.g., one-hot encoding, label encoding). MentionING the Top 5 Categorical Encoding Techniques and also listing out the Major differences between them with the most suitable scenarios where we can use them.

[10]:
```
#creating custom functions for␣
↪OneHotEncode,LabelEncoder,OrdinalEncoder,BinaryEncoder, CountEncoderFunction

saved_econders={}
df.dropna(subset=['Sex', 'Embarked'],inplace=True)
features = df[['Sex', 'Embarked']]

def OneHotEncoderFunction(features_to_encode):
    encoder = OneHotEncoder(sparse_output=False)
    encoder.fit(features_to_encode)
    saved_econders['OneHotEncoder_'+'_'.join(features_to_encode.columns)] =␣
↪encoder

def LabelEncoderFunction(features_to_encode):
    list_encoder = []
    for col in features_to_encode.columns:
        encoder = LabelEncoder()
        encoder.fit(features_to_encode[col])
        saved_econders['LabelEncoder_'+ str(col)] = encoder
```

```python
def OrdinalEncoderFunction(features_to_encode):
    categories = []
    for col in features_to_encode.columns:
        categories.append(sorted(list(features_to_encode[col].unique())))
    encoder = OrdinalEncoder(categories=categories)
    encoder.fit(features_to_encode)
    saved_econders['OrdinalEncoder_'+'_'.join(features_to_encode.columns)] =␣
 ↪encoder

def BinaryEncoderFunction(features_to_encode):
    encoder = BinaryEncoder(cols=features_to_encode.columns)
    encoder.fit(features_to_encode)
    saved_econders['BinaryEncoder_'+'_'.join(features_to_encode.columns)] =␣
 ↪encoder

def CountEncoderFunction(features_to_encode):
    encoder = CountEncoder(cols=features_to_encode.columns)
    encoder.fit(features_to_encode)
    saved_econders['CountEncoder_'+'_'.join(features_to_encode.columns)] =␣
 ↪encoder

encoder_functions =␣
 ↪[OneHotEncoderFunction,OrdinalEncoderFunction,BinaryEncoderFunction,CountEncoderFunction]

for fun in encoder_functions:
    fun(features)

saved_econders
```

[10]:
```
{'OneHotEncoder_Sex_Embarked': OneHotEncoder(sparse_output=False),
 'OrdinalEncoder_Sex_Embarked': OrdinalEncoder(categories=[['female', 'male'],
['C', 'Q', 'S']]),
 'BinaryEncoder_Sex_Embarked': BinaryEncoder(cols=Index(['Sex', 'Embarked'],
dtype='object'),
               mapping=[{'col': 'Sex',
                         'mapping':     Sex_0  Sex_1
  1      0      1
  2      1      0
 -1      0      0
 -2      0      0},
                        {'col': 'Embarked',
                         'mapping':     Embarked_0  Embarked_1
  1           0           1
  2           1           0
  3           1           1
 -1           0           0
```

```
 -2           0           0}]),
 'CountEncoder_Sex_Embarked': CountEncoder(cols=Index(['Sex', 'Embarked'],
dtype='object'),
              combine_min_nan_groups=True)}
```

[11]:
```
# Ecoding the features using custom functions

encoded_dfs = {}

for encoder_name, encoder in saved_econders.items():
    if 'OneHotEncoder' in encoder_name:
        transformed_data = encoder.transform(features)
        columns = encoder.get_feature_names_out(features.columns)
        encoded_df = pd.DataFrame(transformed_data, columns=columns)

    elif 'LabelEncoder' in encoder_name:
        column = encoder_name.split('_')[-1]
        transformed_data = encoder.transform(features[column])
        encoded_df = pd.DataFrame(transformed_data, columns=[column +␣
↪'_encoded'])

    elif 'OrdinalEncoder' in encoder_name:
        transformed_data = encoder.transform(features)
        columns = features.columns + '_ordinal'
        encoded_df = pd.DataFrame(transformed_data, columns=columns)

    elif 'BinaryEncoder' in encoder_name:
        transformed_data = encoder.transform(features)
        columns = transformed_data.columns
        encoded_df = pd.DataFrame(transformed_data, columns=columns)

    elif 'CountEncoder' in encoder_name:
        transformed_data = encoder.transform(features)
        columns = transformed_data.columns
        encoded_df = pd.DataFrame(transformed_data, columns=columns)

    # Storing the DataFrame in a dictionary for later use
    encoded_dfs[encoder_name] = encoded_df

# Displaying the DataFrames
for encoder_name, df in encoded_dfs.items():
    print(f"\n{encoder_name}:\n", df)
```

```
OneHotEncoder_Sex_Embarked:
      Sex_female  Sex_male  Embarked_C  Embarked_Q  Embarked_S
0           0.0       1.0         0.0         0.0         1.0
1           1.0       0.0         1.0         0.0         0.0
```

```
2           1.0       0.0         0.0         0.0         1.0
3           1.0       0.0         0.0         0.0         1.0
4           0.0       1.0         0.0         0.0         1.0
..          …       …         …         …         …
884         0.0       1.0         0.0         0.0         1.0
885         1.0       0.0         0.0         0.0         1.0
886         1.0       0.0         0.0         0.0         1.0
887         0.0       1.0         1.0         0.0         0.0
888         0.0       1.0         0.0         1.0         0.0

[889 rows x 5 columns]

OrdinalEncoder_Sex_Embarked:
      Sex_ordinal  Embarked_ordinal
0            1.0               2.0
1            0.0               0.0
2            0.0               2.0
3            0.0               2.0
4            1.0               2.0
..           …               …
884          1.0               2.0
885          0.0               2.0
886          0.0               2.0
887          1.0               0.0
888          1.0               1.0

[889 rows x 2 columns]

BinaryEncoder_Sex_Embarked:
      Sex_0  Sex_1  Embarked_0  Embarked_1
0         0      1           0           1
1         1      0           1           0
2         1      0           0           1
3         1      0           0           1
4         0      1           0           1
..      …    …         …         …
886       0      1           0           1
887       1      0           0           1
888       1      0           0           1
889       0      1           1           0
890       0      1           1           1

[889 rows x 4 columns]

CountEncoder_Sex_Embarked:
      Sex  Embarked
0    577       644
1    312       168
```

```
2    312       644
3    312       644
4    577       644
..   …       …
886  577       644
887  312       644
888  312       644
889  577       168
890  577        77

[889 rows x 2 columns]
```

0.1.1 Top 5 Categorical Encoding

1. 'OneHotEncoder'
2. 'LabelEncoder'
3. 'OrdinalEncoder'
4. 'BinaryEncoder'
5. 'CountEncoder'

Major differences between them with the most suitable scenarios where we can use them.

1. 'OneHotEncoder' Converts each unique category level into a separate binary column (0/1). Prevents assumptions about ordinal relationships.Provides a complete representation of categories. Can increase dimensionality significantly with many unique categories, leading to sparse matrices.

   suitable scenarios Suitable for algorithms that don't assume order among categories (e.g., linear regression, neural networks) Nominal data without an inherent order (e.g., colors, gender).

2. 'LabelEncoder' Encodes categories as integers from 0 to n-1. Assumes ordinal relationship between categories, which may not be suitable for nominal data Simple and efficient. Maintains order for ordinal data.

   suitable scenarios Ordinal data where categories have a clear order. Suitable for algorithms that can handle numerical values directly (e.g., decision trees, random forests).

3. 'OrdinalEncoder' Encodes categories as integers based on a specified order. Maintains specified order for ordinal data.Suitable for models needing ordinal information. Assumes ordinal relationship, which may not apply to all datasets. Requires specifying category order.

   suitable scenarios Ordinal data where categories have a clear hierarchy or ranking (e.g., education levels, satisfaction ratings).

4. 'BinaryEncoder' Encodes categories into binary digits, reducing the number of columns compared to OneHotEncoder. Reduces dimensionality compared to OneHotEncoder.Less sparse than OneHotEncoder for high-cardinality data. More complex to interpret compared to OneHotEncoder. Assumes no inherent order among categories.

   suitable scenarios Nominal data with many categories, reducing dimensionality while preserving information. Suitable for large datasets where OneHotEncoding would be too sparse.

5. ‘CountEncoder’ Replaces categories with their corresponding frequency counts. Reduces dimensionality. Captures information about category frequency. May lose some categorical information.- Assumes categories with higher frequency are more important.

   suitable scenarios High-cardinality categorical data. Suitable for tree-based algorithms that handle numerical features well (e.g., decision trees, random forests).