How can I perform One-Hot Encoding in Python?

Authored by

stats writer

May 12, 2024

One-Hot Encoding is a commonly used technique in data preprocessing and machine learning tasks to convert categorical data into numerical data. It involves creating a binary vector for each category in a categorical variable, where the vector has a length equal to the total number of categories and contains only 0s and 1s. This allows categorical data to be represented in a way that can be easily processed by machine learning algorithms. In Python, One-Hot Encoding can be performed using the "get_dummies" function from the Pandas library or by using the "OneHotEncoder" class from the Scikit-learn library. Both methods involve converting categorical variables into dummy variables and merging them with the original data set. One-Hot Encoding is a crucial step in data preprocessing and can greatly improve the performance of machine learning models.

Perform One-Hot Encoding in Python

One-hot encoding is used to convert categorical variables into a format that can be readily used by .

The basic idea of one-hot encoding is to create new variables that take on values 0 and 1 to represent the original categorical values.

For example, the following image shows how we would perform one-hot encoding to convert a categorical variable that contains team names into new variables that contain only 0 and 1 values:

Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

One-Hot Encoded Data

Team_A	Team_B	Team_C	Points
1	0	0	25
1	0	0	12
0	1	0	15
0	1	0	14
0	1	0	19
0	1	0	23
0	0	1	25
0	0	1	29

The following step-by-step example shows how to perform one-hot encoding for this exact dataset in Python.

Step 1: Create the Data

First, let's create the following pandas DataFrame:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'team': ,
'points': })

#view DataFrame
print(df)
```

team points
0 A 25
1 A 12
2 B 15
3 B 14
4 B 19
5 B 23
6 C 25
7 C 29

Step 2: Perform One-Hot Encoding

Next, let's import the OneHotEncoder() function from the sklearn library and use it to perform one-hot encoding on the 'team' variable in the pandas DataFrame:

```
from sklearn.preprocessingimport OneHotEncoder

#creating instance of one-hot-encoder
encoder = OneHotEncoder(handle_unknown='ignore')

#perform one-hot encoding on 'team' column
encoder_df = pd.DataFrame(encoder.fit_transform(df]).toarray())
```

```
#merge one-hot encoded columns back with original DataFrame
final_df = df.join(encoder_df)

#view final df
print(final_df)

team points 0 1 2
0 A 25 1.0 0.0 0.0
1 A 12 1.0 0.0 0.0
2 B 15 0.0 1.0 0.0
3 B 14 0.0 1.0 0.0
4 B 19 0.0 1.0 0.0
5 B 23 0.0 1.0 0.0
6 C 25 0.0 0.0 1.0
7 C 29 0.0 0.0 1.0
```

Notice that three new columns were added to the DataFrame since the original 'team' column contained three unique values.

Note: You can find the complete documentation for the OneHotEncoder() function .

Step 3: Drop the Original Categorical Variable

Lastly, we can drop the original 'team' variable from the DataFrame since we no longer need it:

#drop 'team' column
final_df.drop('team', axis=1, inplace=True)

#view final df
print(final_df)

points 0 1 2
0 25 1.0 0.0 0.0
1 12 1.0 0.0 0.0
2 15 0.0 1.0 0.0
3 14 0.0 1.0 0.0
4 19 0.0 1.0 0.0
5 23 0.0 1.0 0.0
6 25 0.0 0.0 1.0
7 29 0.0 0.0 1.0

#rename columns
final_df.columns =

#view final dfprint(final_df)

points teamA teamB teamC

0 25 1.0 0.0 0.0

1 12 1.0 0.0 0.0

2 15 0.0 1.0 0.0

3 14 0.0 1.0 0.0

4 19 0.0 1.0 0.0

5 23 0.0 1.0 0.0

6 25 0.0 0.0 1.0

7 29 0.0 0.0 1.0

The one-hot encoding is complete and we can now feed this pandas DataFrame into any machine learning algorithm that we'd like.