How can data binning be performed in Python, and what are some examples?

Authored by

stats writer

July 1, 2024

Data binning is a data preprocessing technique used to group numerical data into discrete categories or bins. In Python, data binning can be performed using the "cut" function from the Pandas library. This function allows for the creation of bins based on specified cut points or by specifying the number of bins. For example, one could use data binning to categorize age data into groups such as "child," "teenager," "adult," and "elderly" based on age ranges. Another example could be binning sales data into categories such as "low," "medium," and "high" based on sales amounts. Data binning can help to simplify and analyze large datasets by reducing the number of distinct values and highlighting patterns within the data.

Perform Data Binning in Python (With Examples)

You can use the following basic syntax to perform data binning on a pandas DataFrame:

import pandas as pd

#perform binning with 3 bins
df = pd.qcut(df, q=3)

The following examples show how to use this syntax in practice with the following pandas DataFrame:

import pandas as pd

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

#view DataFrame
print(df)

points assists rebounds
0 4 2 7
1 4 5 7
2 7 4 4
3 8 7 6
4 12 7 3
5 13 8 8
6 15 5 9
7 18 4 9
8 22 5 12
9 23 11 11
10 23 13 8
11 25 8 9

Example 1: Perform Basic Data Binning

The following code shows how to perform data binning on the points variable using the function with specific break marks:

#perform data binning on *points* variable
df = pd.qcut(df, q=3)

#view updated DataFrame
print(df)

points assists rebounds points_bin
0 4 2 7 (3.999, 10.667]
1 4 5 7 (3.999, 10.667]
2 7 4 4 (3.999, 10.667]
3 8 7 6 (3.999, 10.667]
4 12 7 3 (10.667, 19.333]
5 13 8 8 (10.667, 19.333]
6 15 5 9 (10.667, 19.333]
7 18 4 9 (10.667, 19.333]
8 22 5 12 (19.333, 25.0]
9 23 11 11 (19.333, 25.0]
10 23 13 8 (19.333, 25.0]
11 25 8 9 (19.333, 25.0]

Notice that each row of the data frame has been placed in one of three bins based on the value in the points column.

We can use the value_counts() function to find how many rows have been placed in each bin:

#count frequency of each bin

```
df.value_counts()

(3.999, 10.667] 4
(10.667, 19.333] 4
(19.333, 25.0] 4
Name: points_bin, dtype: int64
```

We can see that each bin contains 4 observations.

Example 2: Perform Data Binning with Specific Quantiles

We can also perform data binning by using specific quantiles:

```
#perform data binning on points variable with specific quantiles
df = pd.qcut(df, q=)

#view updated DataFrame
print(df)

points assists rebounds points_bin
0 4 2 7 (3.999, 7.2]
1 4 5 7 (3.999, 7.2]
2 7 4 4 (3.999, 7.2]
3 8 7 6 (7.2, 12.4]
```

4 12 7 3 (7.2, 12.4]

5 13 8 8 (12.4, 16.8]

6 15 5 9 (12.4, 16.8]

7 18 4 9 (16.8, 22.8]

8 22 5 12 (16.8, 22.8]

9 23 11 11 (22.8, 25.0]

10 23 13 8 (22.8, 25.0]

11 25 8 9 (22.8, 25.0]

Example 3: Perform Data Binning with Labels

We can also perform data binning by using specific quantiles and specific labels:

```
#perform data binning on points variable with specific quantiles and labels
df = pd.qcut(df,
q=,
labels=)

#view updated DataFrame
print(df)
```

points assists rebounds points_bin

0 4 2 7 A

1 4 5 7 A

2 7 4 4 A

3 8 7 6 B

4 12 7 3 B

5 13 8 8 C

6 15 5 9 C

7 18 4 9 D

8 22 5 12 D

9 23 11 11 E

10 23 13 8 E

11 25 8 9 E

Additional Resources

The following tutorials explain how to perform other common tasks in pandas: