CS506 Programming for Computing

HOS06E– Data Transformation

06/06/2020 Created by Apiwat Chuaphan

3/28/2022 Revised by Cedric Huang

03/08/2023 Reviewed by Maram Elwakeel

School of Technology & Computing (STC) @ City University of Seattle (CityU)

Before You Start

- Version numbers may not match with the most current version at the time of writing. If given the option to choose between stable release (long-term support) or most recent, please select the stable release rather than the beta-testing version.
- There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
- For your working directory, use your course number. This tutorial may use a different course number as an example.
- All the steps and concepts in this tutorial are from the textbook, so if you encounter problems in this tutorial, please try to read and compare the textbook to solve the problem. If you still can't solve the problem, please feel free to contact your course TA.
- Avoid copy-pasting code from the book or the GitHub repository. Instead, type out the code yourself. Resort to copy-pasting only when you are stuck and find that things are not working as expected.

Learning Outcomes

- Understand data transformation
- Understand data discretization and bin
- Understand the difference between cut and qcut

Resources

- Pandas Documentation: https://pandas.pydata.org/docs/user_guide/index.html
- Pandas.qcut(): https://pandas.pydata.org/docs/reference/api/pandas.qcut.html
- Pandas.cut(): https://pandas.pydata.org/docs/reference/api/pandas.cut.html
- Pandas.tranform(): https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.transform.html?highlight=transform#pandas.DataFrame.transform

Section 1 - Data Transformation

1) The process of changing the format, structure, or values of data so that the result may be more efficient and easier to understand. This involves converting data from one structure to another so you can integrate it with different applications.
2) With Pandas, we can perform transformation easily, we did some of transformations in previous HOPs, we will instead use transform function in this module
3) Open Jupyter Notebook:
    i. Create a new file named `data_transformation.ipynb` under Module folder
    ii. Create a DataFrame with the following rows and columns.

```
import numpy as np
import pandas as pd
```

```
trans = pd.DataFrame({"A":[12, 4, 5, None, 1],
                      "B":[7, 2, 54, 3, None],
                      "C":[20, 16, 11, 3, 8],
                      "D":[14, 3, None, 2, 6]},
                      index=['Row_1', 'Row_2', 'Row_3', 'Row_4', 'Row_5'])
trans
```

	A	B	C	D
Row_1	12.0	7.0	20	14.0
Row_2	4.0	2.0	16	3.0
Row_3	5.0	54.0	11	NaN
Row_4	NaN	3.0	3	2.0
Row_5	1.0	NaN	8	6.0

4) Now we will use transform() function to add 10 to each element of the data frame.

i. DataFrame.transform call func on self-producing a DataFrame with transformed values. Produced DataFrame will have same axis length as self.

```
result = trans.transform(lambda x : x + 10)
result
```

	A	B	C	D
Row_1	22.0	17.0	30	24.0
Row_2	14.0	12.0	26	13.0
Row_3	15.0	64.0	21	NaN
Row_4	NaN	13.0	13	12.0
Row_5	11.0	NaN	18	16.0

Notice that we used an anonymous function (lambda) to add 10 to each value. So, all non-empty values have been added successfully.

Section 2 - Data Discretization

1) Defined as a process of converting continuous data attribute values into a finite set of intervals with minimal loss of information. For example, you have height data and want to discretize it to 0 and 1 interval depending on if the height is below or above a certain value of height.
2) Open Jupyter Notebook:
3) Within the same file, create a numpy array with 10 random integers between 10 and 200.

```
# 10 random numbers from 10-200
x = np.random.randint(10, 200, size=10)
x
```

```
array([144,  88, 180, 157,  96,  29, 123, 128,  40, 133])
```

4) Numpy provides some of the functions you can perform discretize with, one is using digitize() for doing so.
5) In this step, we will discretize where 50 is the threshold to divide the data into two categories: one less than 50 and ones above 50.

```
np.digitize(x,bins=[50])

array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1])
```

- The bins argument is a list and so we can specify multiple binning or discretizing conditions.
- As you can see, there are only two numbers lower than 50 represented as 0. There are more than 50 represented as 1.

6) Let's say if we want to have 3 categories, how many values should we specify in the bins array? You got it, 2 values.

```
np.digitize(x, bins=[50, 100])

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

- We have what we wanted. There are 3 categories here: 0, 1, and 2. 0 category has the values of less than 50, the 1 category has the value of less than 100 and category 3 has the value of more than 100.

7) Now let's use a function from Pandas called cut() to discretize our data.

8) It does the same thing as what Numpy's digitize() does, but the way it work is a bit different. Let's first create a DataFrame with random number from provius.

```
df = pd.DataFrame({"height": x})
df
```

	height
0	144
1	88
2	180

- We created 10 values in height columns with index 0-9.

9) Categorize the height variable into four categories using Pandas cut function.

```
# create new column 'binned' with interval
df['binned'] = pd.cut(x=df['height'], bins=[0, 25, 50, 100, 200])
df
```

	height	binned
0	144	(100, 200]
1	88	(50, 100]
2	180	(100, 200]
3	157	(100, 200]
4	96	(50, 100]
5	29	(25, 50]
6	123	(100, 200]
7	128	(100, 200]
8	40	(25, 50]
9	133	(100, 200]

- The height values between 0 and 25 are in one category, height between 25 and 50 are in the second category, 50-100 in third category, and 100-200 are in the fourth category.

10) We also can label them as follows.

```
df['bin_label'] = pd.cut(x = df['height'],
                         bins = [0, 25, 50, 100, 200],
                         labels = [1, 2, 3, 4])

df
```

	height	binned	bin_label
0	144	(100, 200]	4
1	88	(50, 100]	3
2	180	(100, 200]	4
3	157	(100, 200]	4
4	96	(50, 100]	3
5	29	(25, 50]	2
6	123	(100, 200]	4
7	128	(100, 200]	4
8	40	(25, 50]	2
9	133	(100, 200]	4

- As my random data does not have values below 25 so there is no category 1 in this example. Your data may be different.
- You can change labels to string instead of number.

11) Pandas also have another function called qcut() which discretize variable into equal-sized buckets.

You only need to tell the function the number of quantiles, pandas will figure out how to bin that data.

```
pd.qcut(df['height'], q=5)
```

```
0    (130.0, 146.6]
1     (78.4, 112.2]
2    (146.6, 180.0]
3    (146.6, 180.0]
4     (78.4, 112.2]
5    (28.999, 78.4]
6    (112.2, 130.0]
7    (112.2, 130.0]
8    (28.999, 78.4]
9    (130.0, 146.6]
Name: height, dtype: category
Categories (5, interval[float64]): [(28.999, 78.4] < (78.4, 112.2] < (112.2, 130.0] < (130.0, 146.6] < (146.6, 180.0]]
```

- Notice that qcut discretized the data into equal size but the interval is not equal. Binned into 5 groups with 2 members each. Unlike what cut function does which is no guarantee about the distribution of items in each bin, but the interval pretty much the same for all bin.

	Spacing between 2 Bins	Frequency of Samples in Each Bins
cut	Equal Spacing	Different
qcut	Unequal Spacing	Same

12) Let's look at a familiar example below of how to use discretize with data.

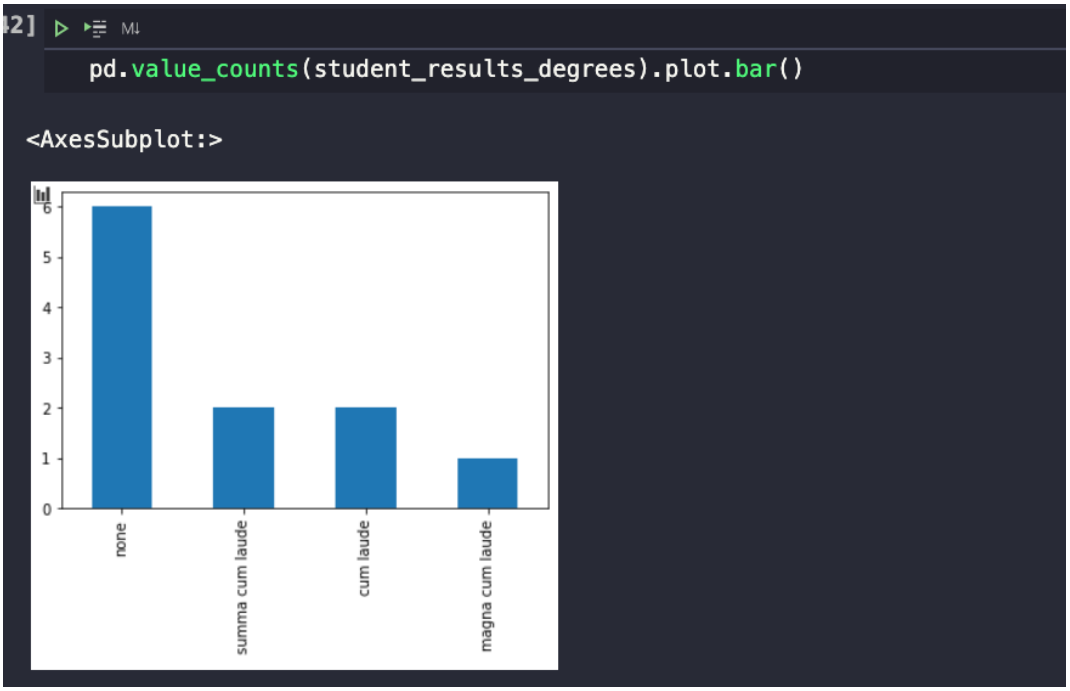
```
degrees = ["none", "cum laude", "magna cum laude", "summa cum laude"]
student_results = [3.93, 3.24, 2.80, 2.83, 3.91, 3.698, 3.731, 3.25, 3.24, 3.82, 3.22]
student_results.sort(reverse=True)

student_results_degrees = pd.cut(student_results, [0, 3.6, 3.8, 3.9, 4.0], labels=degrees)
honor = pd.DataFrame({'grades': student_results,
                      'honors': student_results_degrees})
honor
```

	grades	honors
0	3.930	summa cum laude
1	3.910	summa cum laude
2	3.820	magna cum laude
3	3.731	cum laude
4	3.698	cum laude
5	3.250	none
6	3.240	none
7	3.240	none
8	3.220	none
9	2.830	none
10	2.800	none

- Line number 3, we sorted for easier to see in each category. We categorized what grades belong to which categories. Then, we put the data into DataFrame.

13) If you are a visual person, you need to see the chart. You can do that with a basic plotting provided by pandas like the below image.

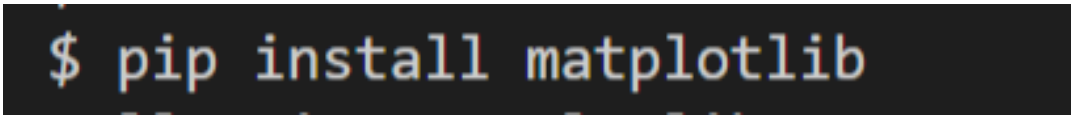
14) We will do a bar plot based on the number of students in each group.



Note: In case you have the following error: "matplotlib is required for plotting"

Then do the following step:

15) In the terminal, type the following to install matplotlib library and rerun your code



16) If you want to see what value_counts() does, remove .plot.bar()

17) Save your Jupyter Notebook with all Output