18.2.5

Data Processing

Now that you know what kind of data you want to work, it's time to meet the needs for your unsupervised model.

The next step is to move on from what you (the user) want to get out of your data and on to what the unsupervised model needs out of the data.

Recall that in the data selection step, you, as the user, are exploring the data to see what kind of insights and analysis you might glean. You reviewed the columns available and the data types stored, and determined if there were missing values.

For data processing, the focus is on making sure the data is set up for the unsupervised learning model, which requires the following:

- Null values are handled.
- Only numerical data is used.
- Values are scaled. In other words, data has been manipulated to ensure that the variance between the numbers won't skew results.

REWIND

Recall that when features have different scales, they can have a disproportionate impact on the model. The unscaled value could lead to messy graphs. Therefore, it is important to understand when to scale and normalize data. For example, if four columns of data are single digits, and the fifth column is in the millions, we would need to scale the fifth column to align the other four.

Let's return again to our list of questions.

Is the data in a format that can be passed into an unsupervised learning model?

We saw before that all our data had the correct type for each column; however, we know that our model can't have strings passed into it.

To make sure we can use our string data, we'll transform our strings of `Yes` and `No` from the Card Member column to `1` and `0`, respectively, by creating a function that will convert `Yes` to a `1` and anything else to `0`.

The function will then be run on the whole column with the `.apply` method, as shown below:

```
# Transform String column
def change_string(member):
    if member == "Yes":
        return 1
    else:
        return 0


df_shopping["Card Member"] = df_shopping["Card Member"].apply(change_string)
df_shopping.head()
```

	Card Member	Age	Annual Income	Spending Score (1-100)
0	1	19.0	15000	39.0
1	1	21.0	15000	81.0
2	0	20.0	16000	6.0
3	0	23.0	16000	77.0
4	0	31.0	17000	40.0

Also, there is one more thing you may notice about the data. The scale for Annual Income is much larger than all the other values in the dataset. We can adjust this format by dividing by 1,000 to rescale those data points, as shown below:

```
# Transform annual income
df_shopping["Annual Income"] = df_shopping["Annual Income"] / 1000
df_shopping.head()
```

	Genre	Age	Annual Income	Spending Score (1-100)
0	1	19	15.0	39
1	1	21	15.0	81
2	0	20	16.0	6
3	0	23	16.0	77
4	0	31	17.0	40

SKILL DRILL

Reformat the names of the columns so they contain no spaces or numbers.