Standardizing our Data

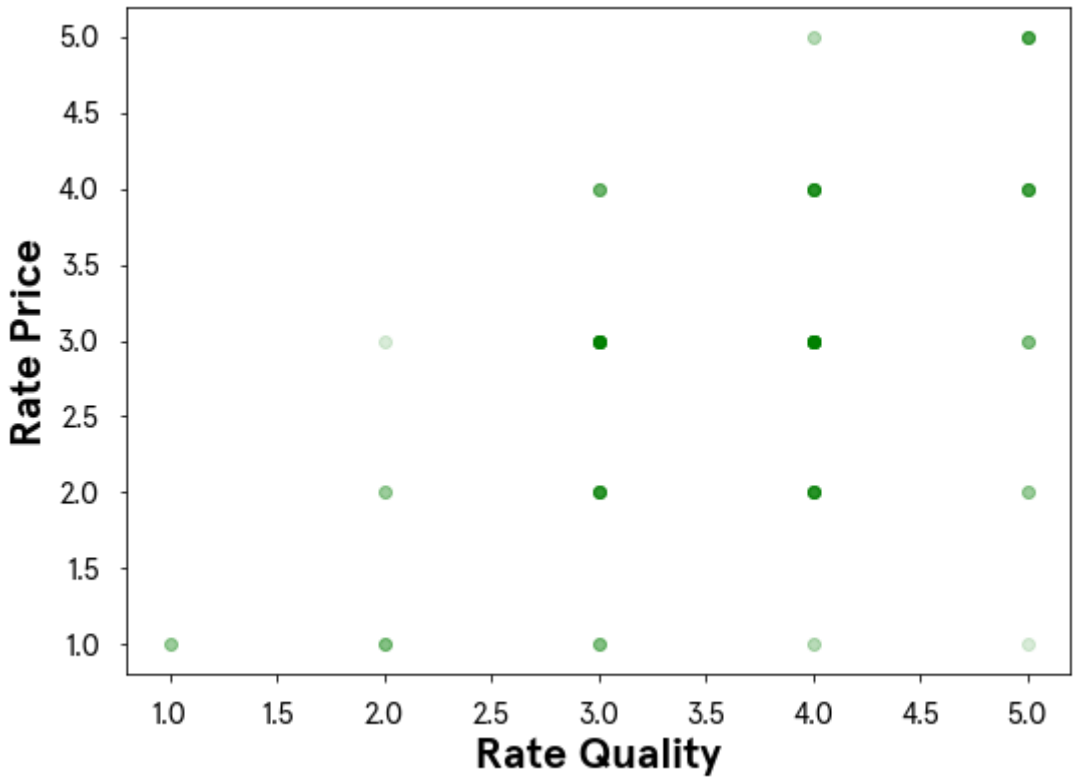
13 min

Excellent work with centering the age feature from our dataset! Now we'll take that concept one step further and discuss standardizing our data. *Standardization* (also known as *Z-Score normalization*) is when we center our data, then divide it by the standard deviation. Once we do that, our entire data set will have a mean of zero and a standard deviation of one. This allows all of our features to be on the same scale. How cool is that?

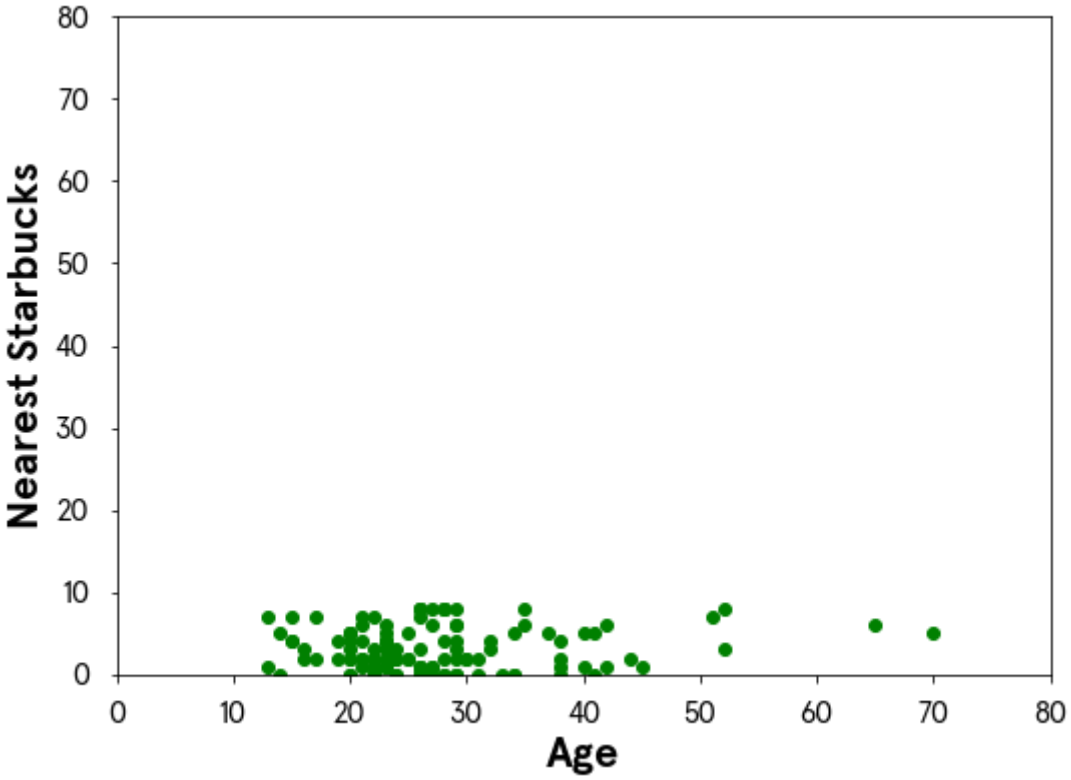
This step is critical because some machine learning models will treat all features equally regardless of their scale. You'll definitely want to standardize your data in the following situations:

- Before Principal Component Analysis
- Before using any clustering or distance based algorithm (think KMeans or DBSCAN)
- Before KNN
- Before performing regularization methods like LASSO and Ridge

If I wanted to see how customers rated quality vs. price, I could graph a scatter plot of those two features and easily see that customers tend to score those two questions closely. Notice the darker spots (meaning more data points are laying on top of one another) around the 3.0 for "Rate Quality" and 3.0 for "Rate Price" as an example. This insight was easy to see since our features are on the same one to five scale.



Now, what happens if I am working with features on two totally different scales? Perhaps the "customer age" and "how far they live from a Starbucks" feature? Let's take a look.



Woah! Looking at this, it is much more challenging to gain insight or even identify patterns within our data. This will be a similar experience for our machine learning models. That's why when we standardize our entire dataset, we tend to see a more robust model performance if all features are on the same scale.

Let's examine one feature to learn the mathematics that goes on when we standardize our data. The mathematical formula will look like this:

z=value−meanstdev*z=stdevvalue−mean*

We'll look at just one feature, where customers were asked how close they are to their nearest Starbucks, and follow that formula above. First, we will set our nearest_starbucks feature to its own variable and then find the mean and standard deviation. Then we can quickly standard our list following the formula above.

```
distance = coffee['nearest_starbucks']

#find the mean of our feature
distance_mean = np.mean(distance)

#find the standard deviation of our feature
distance_std_dev = np.std(distance)

#this will take each data point in distance subtract the mean, then divide by the standard deviation
distance_standardized = (distance - distance_mean) / distance_std_dev
```

We now have our distance feature standardized! Let's double-check by seeing what the mean and standard deviation of our array is.

```
# print what type distance_standardized is
print(type(distance_standardized))
#output = <class 'pandas.core.series.Series'>

#print the mean
print(np.mean(distance_standardized))
#output = 7.644158530205996e-17

#print the standard deviation
print(np.std(distance_standardized))
#output = 1.0000000000000013
```

Our outputs are basically mean = 0 and standard deviation = 1. Fantastic! Let's see what our age feature looks like standardized.

Instructions

1. Checkpoint 1 Passed

1.

Find the average of your variable ages and set the result to a variable called mean_age

Stuck? Get extra guidance

2. Checkpoint 2 Passed

2.

Find the standard deviation of your variable ages and set the result to a variable called std_dev_age

Stuck? Get extra guidance

3. Checkpoint 3 Passed

3.

Standardize our age column. Set the result to a variable called ages_standardized

Stuck? Get extra guidance

4. Checkpoint 4 Passed

4.

Print the mean and standard deviation of ages_standardized.

script.py

```
import pandas as pd

import numpy as np


coffee = pd.read_csv('starbucks_customers.csv')

ages = coffee['age']


## add code below

## set up your variables

mean_age = np.mean(ages)


## standardize ages

std_dev_age = np.std(ages)


## print the results

ages_standardized = (ages - mean_age) / std_dev_age


print(np.mean(ages_standardized))


print(np.std(ages_standardized))
```