Merging and Data Cleaning

Data Boot Camp

Lesson 4.3

Class Objectives

By the end of today's class, you will be able to:

Merge DataFrames and distinguish between inner, outer, left, and right merges.

Slice data by using the `cut()` method, and create new values based on a series of bins.

Fix Python/Pandas bugs within Jupyter Notebook.

Use Google to explore additional Pandas functionality.

Instructor Demonstration

Merging DataFrames

Merging DataFrames

What's Merging?

Sometimes, an analyst will receive data split across multiple tables and sources.

Working across multiple tables is error-prone and confusing.

Merging is the process of combining two tables based on shared data.

Shared data can be an identical column in both tables or a shared index.

In Pandas, we can merge separate DataFrames by using the `pd.merge()` method.

Merging DataFrames: Inner Joins

An inner join is the default method for combining DataFrames by using `pd.merge()`. It only returns data whose values match. Rows that do not include matching data will be dropped from the combined DataFrame.

```python
# Merge two DataFrames using an inner join
merge_df = pd.merge(info_df, items_df, on="customer_id")
merge_df
```

	customer_id	name	email	item	cost
0	112	John	jman@gmail	chips	4.5
1	403	Kelly	kelly@aol.com	soda	3.0
2	999	Sam	sports@school.edu	Laptop	900.0
3	543	April	April@yahoo.com	TV	600.0

Merging DataFrames: Outer Joins

Outer joins combine the DataFrames whether or not any of the rows match. They must be declared as a parameter within the `pd.merge()` method by using the syntax `how="outer"`.

```python
# Merge two DataFrames using an outer join
merge_df = pd.merge(info_df, items_df, on="customer_id", how="outer")
merge_df
```

	customer_id	name	email	item	cost
0	112	John	jman@gmail	chips	4.5
1	403	Kelly	kelly@aol.com	soda	3.0
2	999	Sam	sports@school.edu	Laptop	900.0
3	543	April	April@yahoo.com	TV	600.0
4	123	Bobbo	HeyImBobbo@msn.com	NaN	NaN
5	654	NaN	NaN	Cooler	150.0

Merging DataFrames: Right and Left Joins

These joins protect the data contained within one DataFrame, like an outer join does, while also dropping the rows with null data from the other DataFrame.

```python
# Merge two DataFrames using a left join
merge_df = pd.merge(info_df, items_df, on="customer_id", how="left")
merge_df
```

	customer_id	name	email	item	cost
0	112	John	jman@gmail	chips	4.5
1	403	Kelly	kelly@aol.com	soda	3.0
2	999	Sam	sports@school.edu	Laptop	900.0
3	543	April	April@yahoo.com	TV	600.0
4	123	Bobbo	HeyImBobbo@msn.com	NaN	NaN

```python
# Merge two DataFrames using a right join
merge_df = pd.merge(info_df, items_df, on="customer_id", how="right")
merge_df
```

	customer_id	name	email	item	cost
0	403	Kelly	kelly@aol.com	soda	3.0
1	112	John	jman@gmail	chips	4.5
2	543	April	April@yahoo.com	TV	600.0
3	999	Sam	sports@school.edu	Laptop	900.0
4	654	NaN	NaN	Cooler	150.0

Activity: Census Merging

In this activity, you will merge the two Census datasets that we created in the last class, and then do a calculation and sort the values.

Suggested Time:

15 minutes

Time's Up! Let's Review.

Instructor Demonstration

Binning Data

The binning method places values into groups to enable more vigorous dataset customization.

Binning Data: `pd.cut()`

Use `pd.cut()` when you need to segment and sort data values into bins. This function is also useful for going from a continuous variable to a categorical variable.

```python
# Create the bins in which Data will be held
# Bins are 0, 59.9, 69.9, 79.9, 89.9, 100.
bins = [0, 59.9, 69.9, 79.9, 89.9, 100]

# Create the names for the five bins
group_names = ["F", "D", "C", "B", "A"]
```

```python
# Slice the data and place it into bins
test_scores_df["Test Score Summary"] = pd.cut(test_scores_df["Test Score"],
                                               bins, labels=group_names,
                                               include_lowest=True)
test_scores_df
```

	Class	Name	Test Score	Test Score Summary
0	Oct	Cyndy	90	A
1	Oct	Logan	59	F
2	Jan	Laci	72	C
3	Jan	Elmer	88	B
4	Oct	Crystle	98	A
5	Jan	Emmie	60	D

Binning Data

Binning is so powerful because, after creating and applying these bins, we can group the DataFrame according to those values, and then conduct a higher-level analysis.

```
# Creating a group based off of the bins
test_scores_df = test_scores_df.groupby("Test Score Summary")
test_scores_df.max()
```

Test Score Summary	Class	Name	Test Score
F	Oct	Logan	59
D	Jan	Emmie	60
C	Jan	Laci	72
B	Jan	Elmer	88
A	Oct	Cyndy	98

Activity: Binning Movies

In this activity, you will test your binning skills by creating bins for movies based on their IMDb user vote count.

Suggested Time:

25 minutes

Time's Up! Let's Review.

Instructor Demonstration

Mapping

Similar to Excel's number formats, Pandas unlocks the same functionality by using the `df.map()` method, which allows users to style entire columns at once.

Mapping

`df[<COLUMN>].map(<FORMAT STRING>.format)` enables users to modify the styling of an entire column.

To convert values into a typical dollar format, use `"${:.2f}"`. This places a dollar sign before the value, which has been rounded to two decimal places.

Using `"{:,}"` will split a number up so that it uses comma notation.

```python
# Use Map to format all the columns
file_df["INCOME"] = file_df["INCOME"].map("${:,.2f}".format)
file_df["COSTS"] = file_df["COSTS"].map("${:,.2f}".format)
file_df["PERCENT30"] = (file_df["PERCENT30"]*100).map("{:.1f}%".format)
file_df["PERCENT3050"] = (file_df["PERCENT3050"]*100).map("{:.1f}%".format)
file_df["PERCENT50"] = (file_df["PERCENT50"]*100).map("{:.1f}%".format)
file_df["PERCENT_NODATA"] = (file_df["PERCENT_NODATA"]*100).map("{:.1f}%".format)
file_df["PERCENT_NOBURDEN"] = (file_df["PERCENT_NOBURDEN"]*100).map("{:.1f}%".format)
file_df["TOTAL"] = file_df["TOTAL"].map("{:,}".format)
file_df.head()
```

N3050	BURDEN50	NODATA	NOBURDEN	PERCENT30	PERCENT3050	PERCENT50	PERCENT_NODATA	PERCENT_NOBURDEN	INCOME	COSTS	ObjectId
0	0	0	0	nan%	nan%	nan%	nan%	nan%	$nan	$nan	1
0	0	0	0	nan%	nan%	nan%	nan%	nan%	$nan	$nan	2
4488	1167	0	28209	16.7%	13.3%	3.4%	0.0%	83.3%	$146,287.71	$2,473.83	3
360	157	0	1201	30.1%	21.0%	9.1%	0.0%	69.9%	$147,017.51	$2,508.57	4
436	68	0	3199	13.6%	11.8%	1.8%	0.0%	86.4%	$161,444.76	$2,873.53	5

Mapping

Format mapping only really works once. It will return errors if the same code is run multiple times without restarting the kernel. Therefore, formatting is usually applied near the end of an application.

It will also format NaN values, so it is a good idea to run a `.fillna()` or `.dropna()` to avoid formatting null values.

Format mapping also can change the data type of a column, so all calculations should be handled before modifying the formatting.

```
# Mapping has changed the datatypes of the columns to strings
file_df.dtypes
```

```
YEAR                  int64
AMI                  object
RACE                 object
TENURE               object
AGE                  object
TOTAL                object
BURDEN30              int64
BURDEN3050            int64
BURDEN50              int64
NODATA                int64
NOBURDEN              int64
PERCENT30            object
PERCENT3050          object
PERCENT50            object
PERCENT_NODATA       object
PERCENT_NOBURDEN     object
INCOME               object
COSTS                object
```

Activity: Crowdfunding Cleaning

In this activity, you will take a dataset similar to your first homework, clean it up, and format it.

Suggested Time:

30 minutes

Time's Up! Let's Review.

Break

Instructor Demonstration

Introduction to Bug Fixing

Bug fixing

An error is returned as the application attempts to collect the average value within the `Percentage` column.

```python
# Get the average percentage of veterans living within 75 miles of a cemetery
veterans_df["Percentage"].mean()
```

```
---------------------------------------------------------------------
ValueError                               Traceback (most recent call last)
/opt/anaconda3/envs/dev/lib/python3.10/site-packages/pandas/core/nanops.py in _ensure_numeric(x)
   1621          try:
-> 1622              x = float(x)
   1623          except (TypeError, ValueError):

ValueError: could not convert string to float: '72%74%75%80%83%84%88%89%90%90%89%'

During handling of the above exception, another exception occurred:
```

Bug fixing

The first step: Keep calm.

Bugs happen all the time, and they are rarely the end of the world. In fact, most bugs that you'll encounter are simple enough to solve as long as you know how and where to look for the solution.

Bug fixing

The second step: Figure out what the bug is and where it's located.

- Jupyter Notebook makes it easy to find the erroneous block of code because the error will always be returned in the space following the erroneous cell.

- Unfortunately, Pandas is not known for returning clearly understandable error text. In fact, it often returns large blocks of text that can easily confuse those who do not know the library's underlying code. The line following `KeyError:` is generally a good starting point.

- For example, the text following `ValueError:` within the current code lets the programmer know that Pandas cannot convert the string values in the `Percentage` column to floats.

  ```
  ValueError: could not convert string to float:
  ```

- If the error text isn't entirely clear, it can be helpful to print out variables/columns to the console to uncover the bug's location. For example, printing out the `Percentage` series lets the programmer know that the data type of this Series is an object and not a float.

Bug fixing

The third step: Research the error online to find solutions.

- The key part to this step is coming up with an accurate way to describe the bug, which can take multiple attempts, but it is a skill that will develop over time.

- Google is the programmer's best friend, as typing in a description of the bug will often bring up links to possible solutions. If not, simply alter the search a bit until a solution is discovered.

> 🔍   Pandas cannot convert string to float   🎤

- This particular problem requires the code to drop the percentages within the `Percentage` column, so the search could be more specific and add that information.

> 🔍   Pandas cannot convert string to float percentages   🎤

Activity: Bug Fixing Bonanza

In this activity, you will be provided with a Pandas project containing TONS of bugs. Your job is to take the application and fix it up so that it works properly.

Suggested Time:

35 minutes

Time's Up! Let's Review.

Questions?