learn-co-curriculum / dsc-python-essentials-lab Public

0 stars 45 forks

Star Watch

Code Issues Pull requests Actions Projects Security Insights

solution

This branch is 4 commits ahead, 5 commits behind master. Contribute

hoffm386 update negative and neutral ratings … on Jul 13, 2021 5

View code

Python Essentials - Cumulative Lab

Introduction

Congratulations, you made it through the new content for the first section of the prework! This cumulative lab will help you review and practice everything you've learned in this section — "putting it all together" into an analysis with real-world data.

Objectives

You will be able to:

- Recall the data types covered so far
- Practice extracting information from a nested dataset
- Practice generating insights with conditional logic

Your Task: Analyze Amazon Review Data

For this lab we are going to be working with data collected by Computer Science researchers at the University of California, San Diego. Their full paper citation is here:

> **Justifying recommendations using distantly-labeled reviews and fined-grained aspects** Jianmo Ni, Jiacheng Li, Julian McAuley Empirical Methods in Natural Language Processing (EMNLP), 2019 pdf

We are using a cleaned-up sample version of their full dataset, which contains over 200 million reviews of products on Amazon.com. Specifically, a subset of reviews from the Home and Kitchen category about coffee-related products.

Photo by Karl Fredrickson on Unsplash

Requirements

1. Data Summary

Write code that describes the number of records (dictionaries) in the dataset, as well as the features (keys) contained in each record.

2. Review Selection

Create a variable `review_index` that can be changed, so that your code will select any review from the collection in order to print out information.

3. Review Summary

a. Positive, Negative, or Neutral

Print out information describing whether a review's rating is positive, negative, or neutral.

b. Review Year

Extract just the year from the review date

c. BONUS: Image

Use Jupyter Notebook functionality to display the first image from the review

Data Summary

In the cell below, we've opened up the dataset and loaded it into a list of dictionaries called `reviews` .

```
import json
with open("coffee_product_reviews.json") as f:
    reviews = json.load(f)
type(reviews)
```

```
list
```

In the cell below, delete `None` and replace it with appropriate code so the info printout is correct

(In other words: when you're done, you should have code to find the number of entries in the `reviews` list, i.e. the size or length of the list)

```
num_reviews = len(reviews)
```

```
print("The coffee product review dataset contains {} reviews".format(num_reviews))
```

```
The coffee product review dataset contains 86 reviews
```

Ok, so now we know how many records we are working with! Let's investigate what each record looks like. In the cell below, replace `None` with appropriate code to select the first review

```
first_review = reviews[0]
first_review
```

```
{'rating': 5.0,
 'reviewer_name': 'Sns073194',
 'product_id': 'B00004RFRV',
 'review_title': 'Perfect cafsito every time',
 'review_time': '03 11, 2018',
 'images': ['https://images-na.ssl-images-
amazon.com/images/I/71d2cQEgJsL._SY88.jpg'],
 'styles': {'Size:': ' 6-Cup', 'Color:': ' Silver'}}
```

Now we can check out the keys:

```
first_review.keys()
```

```
dict_keys(['rating', 'reviewer_name', 'product_id', 'review_title',
'review_time', 'images', 'styles'])
```

Before looking at the answer below, try to identify: **what data type are all of these keys?**

.

.

.

Answer: They are all strings. We can tell because they are surrounded by single quotes, e.g. `'rating'` *)*

Now let's look at the values:

```
first_review.values()
```

```
dict_values([5.0, 'Sns073194', 'B00004RFRV', 'Perfect cafsito every time', '03
11, 2018', ['https://images-na.ssl-images-
amazon.com/images/I/71d2cQEgJsL._SY88.jpg'], {'Size:': ' 6-Cup', 'Color:': '
Silver'}])
```

Before looking at the answer below, try to identify: **what data type are all of these values?**

.

.

.

Answer: We have a mix of values

- *The first one,* `5.0` *, is a float (no quotes or brackets, decimal place at the end)*
- *The next four are strings*
- *The fifth is a list. We can tell because it's surrounded by square brackets* `[]` *. Inside that list is a string.*

README.md

dictionary.

Finally, let's look at the first five reviews, all at once:

```
for index in range(5):
    print(reviews[index])
```

```
{'rating': 5.0, 'reviewer_name': 'Sns073194', 'product_id': 'B00004RFRV',
'review_title': 'Perfect cafsito every time', 'review_time': '03 11, 2018',
'images': ['https://images-na.ssl-images-
amazon.com/images/I/71d2cQEgJsL._SY88.jpg'], 'styles': {'Size:': ' 6-Cup',
'Color:': ' Silver'}}
{'rating': 5.0, 'reviewer_name': 'Maverick', 'product_id': 'B00004RFRV',
'review_title': 'Delicious results from a wonderfully simple Bialetti Moka
Express!', 'review_time': '12 3, 2017', 'images': ['https://images-na.ssl-images-
amazon.com/images/I/61NG30sWdJL._SY88.jpg'], 'styles': {'Size:': ' 1-Cup',
'Color:': ' Silver'}}
{'rating': 5.0, 'reviewer_name': 'Karen', 'product_id': 'B00004RFRV',
'review_title': 'Bialetti is the Best!', 'review_time': '11 12, 2017', 'images':
['https://images-na.ssl-images-amazon.com/images/I/81+XxFRGyBL._SY88.jpg'],
'styles': {'Size:': ' 12-Cup', 'Color:': ' Silver'}}
{'rating': 5.0, 'reviewer_name': 'Feles (muy Mala)', 'product_id': 'B00004RFRV',
'review_title': 'Awesome portion control for one person!', 'review_time': '08 5,
2017', 'images': ['https://images-na.ssl-images-
amazon.com/images/I/71BcwbkGyfL._SY88.jpg'], 'styles': {'Size:': ' 6-Cup',
'Color:': ' Purple'}}
{'rating': 1.0, 'reviewer_name': 'EJ', 'product_id': 'B00004RFRV',
'review_title': 'Rusted spots everywhere fresh out the box...nasty',
'review_time': '06 4, 2017', 'images': ['https://images-na.ssl-images-
amazon.com/images/I/71Dbr6X0bYL._SY88.jpg'], 'styles': {'Size:': ' 9-Cup',
'Color:': ' Silver'}}
```

It looks like each review has the same structure as the first one.

Edit the string below to describe what we've learned about the dataset so far:

```
"""
For this analysis, we are using a dataset collected from Amazon.com by UCSD research

Each record represents a product review of a coffee-related product

There are a total of 86 records

Each record has 7 keys, all of which are type string

The values associated with these keys have mixed types: float, string, list, and dic
"""
```

Review Selection

Now that we have a general sense of what is contained in our dataset, let's implement a system for a user to be able to query for an individual record. For now, assume that the user can edit the value of a variable in this Jupyter Notebook.

In the cell below, create a variable called `review_index` and set it to the value of `2`

(Why are we bothering to use a variable, if we're just "hard-coding" it to 2? Because it's helpful to practice *parameterizing* our code, i.e. using variables that can have their values substituted rather than using the values directly.)

```
review_index = 2
```

Now let's use that review index to create a variable `selected_review` that extracts the relevant review dictionary from the list of review dictionaries

```
selected_review = reviews[review_index]
selected_review
```

```
{'rating': 5.0,
 'reviewer_name': 'Karen',
 'product_id': 'B00004RFRV',
 'review_title': 'Bialetti is the Best!',
 'review_time': '11 12, 2017',
 'images': ['https://images-na.ssl-images-
amazon.com/images/I/81+XxFRGyBL._SY88.jpg'],
 'styles': {'Size:': ' 12-Cup', 'Color:': ' Silver'}}
```

Review Summary

So far we have investigated the structure of our data, and written reusable code to extract a single review from the list. This allowed us to practice identifying data types and extracting information from nested lists and dictionaries.

Now it's time to practice two other key skills: **conditionals** and **string parsing** (and optionally, learn how to display images with Python code in a Jupyter Notebook).

We'll do this by **writing code to summarize a given review dictionary** in a more user-friendly way than the original raw dictionary format, practicing some data cleaning along the way.

Positive, Negative, or Neutral

Using conditionals, let's display whether a given review is positive, negative, or neutral based on the value associated with the `rating` key. We'll use the following definitions:

- Positive: `rating` value of 4 or 5 (out of 5)
- Neutral: `rating` value of 3 (out of 5)
- Negative: `rating` value of 1 or 2 (out of 5)

Once you've found that value, print out: `This is a <blank> review` where `<blank>` is replaced with either `positive`, `negative`, or `neutral`.

For example, with the current selection, the rating is 5.0, so we should print `This is a positive review`.

First, let's extract the rating from the `selected_review` variable:

```
selected_rating = selected_review["rating"]
selected_rating
```

```
5.0
```

Now, in the cell below, write code using `selected_rating`, `if`, `elif` and `else` so that when `selected_rating` changes value, it will print out the right `This is a <blank> review` statement.

(Again, since the current value of `selected_rating` == 5.0, your code should print out `This is a positive review`, but your code should be able to print different statements when `selected_rating` changes value!)

```
if selected_rating >= 4:
    print("This is a positive review")
elif selected_rating <= 2:
    print("This is a negative review")
else:
    print("This is a neutral review")
```

```
This is a positive review
```

Ok, now that this worked for a single example, let's try it out on a few others.

```
review_index = 4
selected_review = reviews[review_index]
selected_rating = selected_review["rating"]
```

Paste your code from above to analyze the new selection. This one should say it's a negative review.

```
if selected_rating >= 4:
    print("This is a positive review")
elif selected_rating <= 2:
    print("This is a negative review")
else:
    print("This is a neutral review")
```

```
This is a negative review
```

Let's try one more, which should say it's a neutral review.

```
review_index = 47
selected_review = reviews[review_index]
selected_rating = selected_review["rating"]



if selected_rating >= 4:
    print("This is a positive review")
elif selected_rating <= 2:
    print("This is a negative review")
else:
    print("This is a neutral review")
```

```
This is a neutral review
```

Great! We just practiced using conditionals to make a more user-friendly summary

Review Year

While it may be less exciting than building machine learning models, a significant part of data science is data cleaning. Lets start to practice some data cleaning skills with the `review_time` key-value pairs.

For the rest of this lab, we'll go ahead and set up three variables to represent the positive, negative, and neutral examples above.

(Don't worry too much about this syntax; it uses "unpacking" and "list comprehensions", which we haven't covered yet.)

```
selected_review_indices = (2, 4, 47)
positive_review, negative_review, neutral_review = [reviews[i] for i in selected_rev
```

Now let's extract the `review_time` value from the positive review:

```
positive_review_time = positive_review["review_time"]
positive_review_time
```

```
'11 12, 2017'
```

Ok, it looks like this is is a string showing the month, the day, and then the year that the review was written. Write code to extract the last 4 characters of the string, then convert it into an integer

```
positive_review_year = int(positive_review_time[-4:])
positive_review_year
```

```
2017
```

```
type(positive_review_year)
```

```
int
```

Repeat the same logic for the negative review and the neutral review

```
negative_review_time = negative_review["review_time"]
negative_review_year = int(negative_review_time[-4:])
negative_review_year
```

```
2017
```

```
neutral_review_time = neutral_review["review_time"]
neutral_review_year = int(neutral_review_time[-4:])
neutral_review_year
```

```
2015
```

Bonus: Images

(You can skip past this section if you want — this content will not be assessed.)

One of the reasons Jupyter Notebooks are such a powerful data science tool is that they allow you to do a bit of web development without learning a new language beyond Python and Markdown. This entire notebook is just a complicated web page, and the information you've been printing out so far means you are creating dynamic web elements with your code!

In addition to displaying the output of cells as strings of data, we can actually use Python to display images. We'll use the `Image` class from the `display` submodule of the `IPython` library, which is kind of like using the built-in `print` function to write text. There are a lot of other options in the `display` submodule that we won't cover, but you can read about them here.

```
from IPython.display import Image
```

Here is an example of using the `Image` class with a hard-coded image, pulled from the documentation:

```
Image('http://www.google.fr/images/srpr/logo3w.png')
```

Recall that our data contains links to images:

```
positive_review
```

```
{'rating': 5.0,
 'reviewer_name': 'Karen',
 'product_id': 'B00004RFRV',
 'review_title': 'Bialetti is the Best!',
 'review_time': '11 12, 2017',
 'images': ['https://images-na.ssl-images-
amazon.com/images/I/81+XxFRGyBL._SY88.jpg'],
 'styles': {'Size:': ' 12-Cup', 'Color:': ' Silver'}}
```

The image link is a string, contained in a list, associated with the key `images`, so we'll extract it like this:

```
positive_review_image_url = positive_review['images'][0]
positive_review_image_url
```

```
'https://images-na.ssl-images-amazon.com/images/I/81+XxFRGyBL._SY88.jpg'
```

Now we can plug that into the `Image` tool:

```
Image(positive_review_image_url)
```

The same goes for the negative and neutral reviews:

```
negative_review_image_url = negative_review['images'][0]
Image(negative_review_image_url)
```

```
neutral_review_image_url = neutral_review['images'][0]
Image(neutral_review_image_url)
```

Bringing It All Together

Now that we have individually extracted all of the relevant pieces, let's display a complete review summary based on a specified `review_index`

We'll use index `2` (the positive review) as an example, although someone using your notebook should be able to change just the value of `review_index` and the rest should automatically update.

```
review_index = 2
```

The complete summary for index `2` should look like this:

```
"Bialetti is the Best!"
This was a positive review written by Karen in 2017.
```

Optionally, it can also show the first associated image.

```
# Extract review from list of reviews
selected_review = reviews[review_index]

# Extract title
selected_review_title = selected_review["review_title"]

# Extract rating and format as positive, negative, or neutral
selected_rating = selected_review["rating"]
if selected_rating >= 4:
    selected_sentiment = "positive"
elif selected_rating <= 2:
    selected_sentiment = "negative"
else:
    selected_sentiment = "neutral"

# Extract author
selected_author = selected_review["reviewer_name"]

# Extract year (doesn't need to be int for this use case)
selected_year = selected_review["review_time"][-4:]

print(f'''"{selected_review_title}"
This was a {selected_sentiment} review written by {selected_author} in {selected_yea

Image(selected_review["images"][0])
```

```
"Bialetti is the Best!"
This was a positive review written by Karen in 2017.
```

Conclusion

In this cumulative lab, you practiced some of the skills you've learned so far using real-world data. Starting from a nested list of dictionaries (which also contained other lists and dictionaries), you were able to extract and transform data into a new format.

Releases

No releases published

Packages

No packages published

Contributors 2

hoffm386 Erin R Hoffman

ismayc Chester Ismay

Languages

● **Jupyter Notebook** 100.0%