8.2.1 Extract the Wikipedia Movies JSON

Wikipedia has a ton of information about movies, including budgets and box office returns, cast and crew, production and distribution, and so much more. Luckily, one of Britta's coworkers created a script to go through a list of movies on Wikipedia from 1990 to 2018 and extract the data from the sidebar into a JSON. Unfortunately, her coworker can't find the script anymore and just has the JSON file. We'll need to load in the JSON file into a Pandas DataFrame.

GITHUB

Create a new GitHub repository named "Movies-ETL." Then navigate to your class folder and clone your new repo within the folder.

Download the Wikipedia JSON file to your class folder.

wikipedia-movies.json (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_8/wikipedia-movies.json)

Then, activate your coding environment following the steps for your operating system below.

Check out the macOS instructions below, or jump to the **Windows instructions**.

---

macOS

While the JSON file is downloading, follow these steps:

1. Open a new terminal window.
2. Navigate to your class folder.
3. Activate the PythonData environment.
4. Start the Jupyter Notebook server.

> **REWIND**
>
> ---
>
> Remember, the command to activate the PythonData environment is `conda activate PythonData`. The command to start up the Jupyter Notebook server is `jupyter notebook`.

Windows

While the JSON file is downloading, follow these steps.

1. Open the Anaconda Prompt (PythonData).
2. Navigate to your class folder.

3. Start the Jupyter Notebook server.

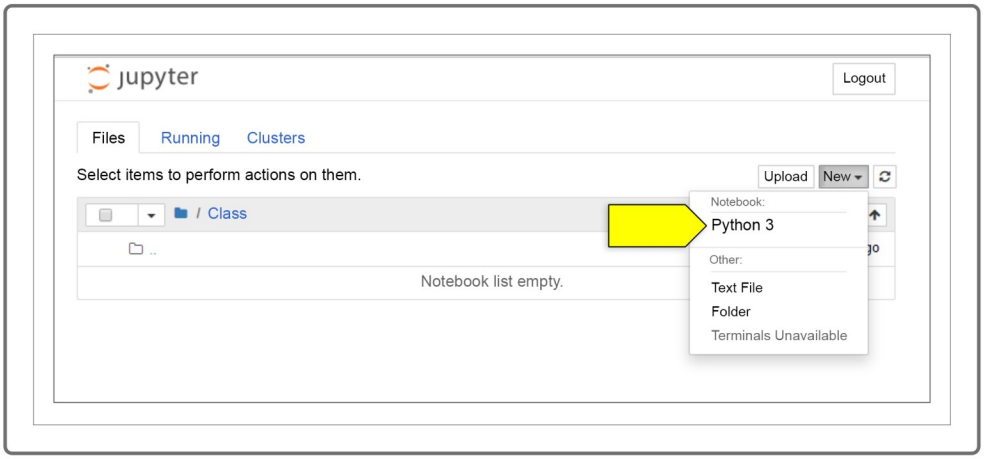
REWIND

Recall that the command to start the Jupyter Notebook server is `jupyter notebook`.

Once Jupyter Notebook is up and running, create a new Python 3 notebook.

REWIND

Remember, the New dropdown menu to create a new notebook is in the top right of the Jupyter page.



Eventually, we'll want to create an automated pipeline, which Jupyter Notebooks aren't suited for. But first, we'll need to do some exploratory data analysis, as Jupyter Notebooks are great for exploring data. Then we can copy the code we've created to a Python script.

Find the File

Now that our notebook is up and running and we've downloaded the Wikipedia movie data, we can start writing some code.

> **REWIND**
>
> ---
>
> Remember, if you're going to use any Python dependencies, it's best to import them all at the beginning. As you write your code, if you learn that you need more dependencies, add them to the import statements at the top of your code.

To start, we need to import three dependencies:

1. JSON library to extract the Wikipedia data
2. Pandas library to create DataFrames
3. NumPy library for converting data types

Import all three dependencies in the first cell with the following code:

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

In the next cell, we want to import the Wikipedia JSON file. To make our lives easier, define a variable `file_dir` for the directory that's holding our data. The exact file path will depend on the directory in which you've saved your data. Here's what our code looks like:

```
file_dir = 'C://Users/Username/DataBootcamp/'
```

Now, if you want to open a file in your directory, you can use an f-string (see below) instead of having to type out the whole directory every time. If you move your files, you only need to update the `file_dir` variable.

```
f'{file_dir}filename'
```

You may be tempted to try to read the JSON files directly into a Pandas DataFrame. While technically that may be possible, the `read_json` method that comes built into the Pandas library only works well for data that is already clean—for example, when the JSON data has every field filled in every time it is returned. We call data like this "flat."

Most data you'll work with in real life won't come to you in a flat format. One great thing about the JSON format is it's really flexible, and it can handle raw, messy data. But if you try to read raw, messy JSON data directly into a DataFrame, the DataFrame will be a mess too. It's very difficult to find and fix corrupted data in messy DataFrames, and it's also difficult to consolidate columns without headaches.

As you may have guessed, the type of data we get from doing a scrape of Wikipedia is pretty messy, so it's easier to load the raw JSON as a list of dictionaries before converting it to a DataFrame.

Load the JSON into a List of Dictionaries

To load the raw JSON into a list of dictionaries, we will use the `load()` method.

REWIND

Remember, when opening files in Python, we want to use the `with` statement to handle the file resource.

Using the `with` statement, open the Wikipedia JSON file to be read into the variable `file`, and use `json.load()` to save the data to a new variable.

```
with open(f'{file_dir}/wikipedia-movies.json', mode='r') as file:
    wiki_movies_raw = json.load(file)
```

Here, `wiki_movies_raw` is now a list of dicts. Before we take a look at the data, we should check how many records were pulled in. We can use the `len()` function (see below), which returns 7,311 records.

```
len(wiki_movies_raw)
```

PAUSE

Is 7,311 a reasonable number of records? We just want to make sure that we don't have an outlandishly large or small number. If we do, there's potentially something seriously wrong with the data that needs to be investigated before moving on.

Show Answer

Also, we should always take a look at a few individual records just to make sure that the data didn't come in horribly garbled. With a DataFrame, we'd do this with the `head()` and `tail()` methods, but with a list of dicts, we need to inspect the records directly.

REWIND

Remember, since we're working with a list, we'll use index slices to select specific chunks of `wiki_movies_raw` to inspect directly. This is also a great use case for **negative index** slices.

To see the first five records, use the following:

```
# First 5 records
wiki_movies_raw[:5]
```

To see the last five records, use the following:

```
# Last 5 records
```

```
wiki_movies_raw[-5:]
```

It's always a good idea to check records in the middle as well. Choose a number somewhere around the halfway mark and look at a handful of records after that index.

```
# Some records in the middle
wiki_movies_raw[3600:3605]
```

If everything looks good, congratulations! You're halfway through the Extract step. Now we'll load in the Kaggle data.

© 2020 - 2022 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.