Data Collection

September 26, 2021

[ ]:
```
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

1 Acquiring Data

The datasets we have worked with so far have all been small enough that we have directly typed them in our code blocks. In reality, you'll need to bring in your data from an outside source.

This can be done in many ways. We'll explore a few in this lab and, we will also mention some methods that are out of scope for this course but often seen in the wild.

1.1 Uploading Data

If you have the data that you want to work with on your computer, you can work with it locally using Python, Jupyter, and/or many other tools. If you want to work with that data in Colab, you'll need to upload the data to Colab since Colab executes code on a virtual machine in the cloud, not locally on your computer.

The first step of uploading data is having the data on your local machine.

For this example we will use the famous Iris Dataset. There are many copies of this dataset on the internet. We'll use the version hosted by the University of California Irvine.

The direct link to the dataset is `https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data`.

Download `iris.data` now.

Once you have the data downloaded, you can upload the file to this Colab environment. To do that:

1. Click on the folder icon on the left of the Colab interface. This opens the 'Files' sidebar.
2. At the top of the sidebar there is an 'Upload' link. Click the link in order to open a file selctor.
3. Through the selector, find the `iris.data` file that you just downloaded.
4. Click 'Open' or 'OK' to confirm the upload.

You will see a warning about files not being saved. This is because the file is stored on a virtual machine in the cloud, and when that machine is turned off, all files in it are lost. For classes like this you should be fine. For longer projects or long-running model trainings, there are other ways and places to store your files. We'll get to those later in the course.

Let's see if you uploaded the file successfully.

Run the code block below

[ ]:
```
import pandas as pd

column_names = [
  'sepal length',
  'sepal width',
  'petal length',
  'petal width',
  'class'
]

pd.read_csv('iris.data', names=column_names)
```

[ ]:
```
     sepal length  sepal width  petal length  petal width           class
0             5.1          3.5           1.4          0.2     Iris-setosa
1             4.9          3.0           1.4          0.2     Iris-setosa
2             4.7          3.2           1.3          0.2     Iris-setosa
3             4.6          3.1           1.5          0.2     Iris-setosa
4             5.0          3.6           1.4          0.2     Iris-setosa
..            …            …             …            …             …
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica

[150 rows x 5 columns]
```

You should see a `DataFrame` containing information about iris flowers.

If you get an error, you likely uploaded the wrong file or uploaded the file to the wrong location.

By default Colab works in the `/content/` folder of the virtual machine. Most of the time this is invisible to you. However, when you uploaded the file, you might have hit the 'Parent Directory' link instead of the 'Upload' link since they are close to each other and both have "up arrow" icons. If you see a long list of folders instead of a single `sample_data` folder in the files list, then you hit

the 'Parent Directory' button. Unfortunately, the only way to redirect uploads to the correct folder is to restart your runtime.

1.2 Downloading With Python

If your data is hosted online, you can use Python to directly download the data and bypass the download/upload cycle mentioned in the last section.

One way to do this is to use the `urllib.request` library's `urlretrieve` method.

In the example below we request the `iris.names` file from UCI and then list the directory where it was downloaded.

[ ]:
```
import urllib.request
import os

urllib.request.urlretrieve(
    'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.names',
    'iris.names')

os.listdir()
```

[ ]:
```
['iris.names',
 'slides.pdf',
 'kaggle.json',
 'bridges.data.version2',
 'avocado.csv',
 'colab.ipynb',
 'iris.data',
 '.github',
 '.gitattributes',
 '.ipynb_checkpoints',
 '.git']
```

1.3 Downloading With Pandas

It is possible to download data directly into a Pandas `DataFrame`. We have used `read_csv` in previous labs to load files from disk. If you pass `read_csv`, a URL it will pull data from the internet and load that data into a `DataFrame` in one shot.

[ ]:
```
import pandas as pd

column_names = [
  'sepal length',
  'sepal width',
  'petal length',
  'petal width',
  'class'
]
```

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

pd.read_csv(url, names=column_names)
```

[ ]:

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
..	…	…	…	…	…
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

[150 rows x 5 columns]

1.4 Kaggle Data

Kaggle is a popular machine learning and data science educational playground. There are many interesting datasets hosted on Kaggle's datasets page.

If you navigate to a dataset you won't be able to download it until you create a Kaggle account.

We'll be using Kaggle in this course. Even outside of the course, Kaggle is a great place to learn and experiment with machine learning and data science, all while building your public machine learning and data science resume.

Log in to Kaggle now. Create a new account if you need to.

At this point you should have a Kaggle account. You can now download a dataset by clicking on the 'Download' link at the top of the information page for the dataset.

If you download the Oranges vs. Grapefruit dataset, you should now have a file called `oranges-vs-grapefruit.zip` on your computer.

Upload `oranges-vs-grapefruit.zip` to this lab.

Now that you have uploaded `oranges-vs-grapefruit.zip`, we can load it into a `DataFrame`.

```
import pandas as pd

pd.read_csv('oranges-vs-grapefruit.zip')
```

[ ]:

	name	diameter	weight	red	green	blue
0	orange	2.96	86.76	172	85	2
1	orange	3.91	88.05	166	78	3
2	orange	4.42	95.17	156	81	2

```
3          orange      4.47   95.60  163     81     4
4          orange      4.48   95.76  161     72     9
…           …       …      …   …    …    …
9995  grapefruit     15.35  253.89  149     77    20
9996  grapefruit     15.41  254.67  148     68     7
9997  grapefruit     15.59  256.50  168     82    20
9998  grapefruit     15.92  260.14  142     72    11
9999  grapefruit     16.45  261.51  152     74     2

[10000 rows x 6 columns]
```

Notice that the file that we loaded was `oranges-vs-grapefruit.zip`, which is a zip file, not a csv file. Zip files are 'compressed' files. We do this to save space. However, if you were to open `oranges-vs-grapefruit.zip` in a text editor, you wouldn't be able to read it. Lucky for us, `read_csv` knows what to do when it receives a compressed file.

Sometimes we will want to decompress a file before creating a `DataFrame`. Zip files can contain more than one file, so we might need to unzip our files and then load them individually into `DataFrame` objects.

To do this we use the `zipfile` library.

In the example below, we open the zip file and then extract all of the contained files into the current directory. We then list the directory and see that we now have a `citrus.csv` file, which is the uncompressed contents of `oranges-vs-grapefruit.zip`. This csv can then be loaded into a `DataFrame` directly.

[ ]:
```
import zipfile
import os

with zipfile.ZipFile('oranges-vs-grapefruit.zip','r') as z:
  z.extractall('./')

os.listdir()
```

[ ]:
```
['iris.names',
 'slides.pdf',
 'citrus.csv',
 'kaggle.json',
 'bridges.data.version2',
 'avocado.csv',
 'colab.ipynb',
 'iris.data',
 'oranges-vs-grapefruit.zip',
 '.github',
 '.gitattributes',
 '.ipynb_checkpoints',
 '.git']
```

Zip is one of many file compression formats, and it is actually more than just a compression format.

Remember when we mentioned above that a zip file might contain multiple files? The combining of one or more files is known as archiving. The reduction in size of files is known as compression. Zip is actually an archiving and compression algorithm.

You can find a list of similar types of algorithms on Wikipedia.

1.4.1 Direct Downloads

So far, we've been able to download data from Kaggle and then upload it to Colab. This involves downloading the entire dataset from Kaggle's servers onto your local machine and then uploading that dataset to the Colab server to actually process the data. For small datasets, this is reasonable. But for large datasets, this can quickly become a burden on your network connection and your device's storage space.

Kaggle offers an API that comes with a command line program that can help you download files directly from Kaggle to Colab, skipping over your local machine entirely.

Credentials In order to use the API, you'll need to "log in" to Kaggle. This is done using API credentials in the form of an API token.

To do that:

1. Navigate to the 'Account' tab of your user profile in Kaggle at `https://www.kaggle.com/<username>/account`.
2. Click the "Create New API Token" button. This will download a `kaggle.json` file containing your API credentials
3. Upload the `kaggle.json` file to this lab.

Warning: Keep your `kaggle.json` private! It contains information that will allow people to authenticate into Kaggle using your user account.

At this point you should have a `kaggle.json` file in the file list on the left. We can now download a dataset using the `kaggle` command and `datasets` subcommand:

[ ]:
```
! KAGGLE_CONFIG_DIR=/content/ kaggle datasets download joshmcadams/
↪oranges-vs-grapefruit
```

```
Traceback (most recent call last):
  File "/Users/josemartinez/opt/anaconda3/envs/data/bin/kaggle", line 7, in
<module>
    from kaggle.cli import main
  File "/Users/josemartinez/opt/anaconda3/envs/data/lib/python3.9/site-
packages/kaggle/__init__.py", line 19, in <module>
    from kaggle.api.kaggle_api_extended import KaggleApi
  File "/Users/josemartinez/opt/anaconda3/envs/data/lib/python3.9/site-
packages/kaggle/api/__init__.py", line 22, in <module>
    from kaggle.api.kaggle_api_extended import KaggleApi
  File "/Users/josemartinez/opt/anaconda3/envs/data/lib/python3.9/site-
packages/kaggle/api/kaggle_api_extended.py", line 84, in <module>
    class KaggleApi(KaggleApi):
  File "/Users/josemartinez/opt/anaconda3/envs/data/lib/python3.9/site-
```

```
packages/kaggle/api/kaggle_api_extended.py", line 102, in KaggleApi
    os.makedirs(config_dir)
  File "/Users/josemartinez/opt/anaconda3/envs/data/lib/python3.9/os.py", line
225, in makedirs
    mkdir(name, mode)
OSError: [Errno 30] Read-only file system: '/content/'
```

You should see text similar to:

```
Downloading oranges-vs-grapefruit.zip to /content
  0% 0.00/61.2k [00:00<?, ?B/s]
100% 61.2k/61.2k [00:00<00:00, 23.1MB/s]
```

If you do, then you successfully downloaded the dataset!

You might have also seen a warning like this:

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can ru
```

If so, it means that the `kaggle.json` file is readable by people other than you. This is probably okay since you are on a virtual machine by yourself. If you do want to fix the warning, run `chmod` as instructed.

[ ]:
```
! chmod 600 kaggle.json
```

You might also be wondering what that `KAGGLE_CONFIG_DIR=/content/` in front of the `kaggle` command was.

This is telling `kaggle` where to find your `kaggle.json` file. `kaggle` expects the file to be in `~/.kaggle/`. Since we didn't upload it there, `kaggle` can't find `kaggle.json` without us leading it to the correct folder.

If you want to not have to do this, move `kaggle.json`.

First make sure the directory exists.

[ ]:
```
! ls ~/.kaggle 2>/dev/null || mkdir ~/.kaggle
```

```
kaggle.json
```

And then move the file.

[ ]:
```
! mv kaggle.json ~/.kaggle
```

Now you can run the `kaggle` command without having to set the configuration directory.

[ ]:
```
! kaggle datasets download joshmcadams/oranges-vs-grapefruit
```

```
oranges-vs-grapefruit.zip: Skipping, found more recently modified local copy
(use --force to force download)
```

Note that you'll have to repeat this process every time your virtual machine resets. The setup will live through reloads though.

Keep a copy of your `kaggle.json` file on your local machine. Then, when you need to load your credentials into a colab, just:

1. Upload `kaggle.json`
2. Create a code block and run `! chmod 600 kaggle.json && (ls ~/.kaggle 2>/dev/null || mkdir ~/.kaggle) && mv kaggle.json ~/.kaggle/ && echo 'Done'`

1.5 Other Data Acquisition Methods

1.5.1 Databases

It is possible to interact with databases directly from Python and therefore from Colab and other notebook environments. Python has a standard database API and many toolkits that make interacting with databases easier.

If your data is stored in a database, you'll need to work with a database administrator to get access credentials and to understand the data and how it is stored.

1.5.2 APIs

Data can also be accessed by application programming interface (API). APIs provide a way for you to write Python code that interacts with another system in a well defined way.

For example, Twitter has an API that allows you to work with tweets. There are even abstraction layers like Tweepy that make working with the API even easier.

Every system has their own API with different methods and calling patterns. You'll hear terms like REST, SOAP, JSON and XML thrown around when talking about specific APIs.

2 Exercises

2.1 Exercise 1: Direct Download

Use Python to directly download the `bridges.data.version2` data file from the UCI Machine Learning Repository. Load the data into a Pandas `DataFrame` and `.describe()` that `DataFrame`.

Student Solution

[ ]:
```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

data_df = pd.read_csv('bridges.data.version2')
data_df.describe()
```

[ ]:
```
          E1    M    3  CRAFTS  HIGHWAY       ?    2    N  THROUGH   WOOD  \
count    107  107  107     107      107     107  107  107      107    107
unique   107    4   55       4        4       4    5    3        3      4
top       E2    A   28  MATURE  HIGHWAY  MEDIUM    2    G  THROUGH  STEEL
freq       1   49    5      54       70      48   60   80       86     79

          SHORT    S    WOOD.1
count       107  107       107
unique        4    4         8
top     MEDIUM    F  SIMPLE-T
freq         53   58        44
```

2.2 Exercise 2: Kaggle Download

Use the Kaggle API to download a dataset containing avocado prices in the US. Load the data into a Pandas `DataFrame` and describe the `DataFrame`.

Student Solution

```
[ ]: kaggle_df = pd.read_csv('avocado.csv')

kaggle_df.describe()
```

```
[ ]:          Unnamed: 0  AveragePrice  Total Volume          4046          4225  \
count  18249.000000  18249.000000  1.824900e+04  1.824900e+04  1.824900e+04
mean      24.232232      1.405978  8.506440e+05  2.930084e+05  2.951546e+05
std       15.481045      0.402677  3.453545e+06  1.264989e+06  1.204120e+06
min        0.000000      0.440000  8.456000e+01  0.000000e+00  0.000000e+00
25%       10.000000      1.100000  1.083858e+04  8.540700e+02  3.008780e+03
50%       24.000000      1.370000  1.073768e+05  8.645300e+03  2.906102e+04
75%       38.000000      1.660000  4.329623e+05  1.110202e+05  1.502069e+05
max       52.000000      3.250000  6.250565e+07  2.274362e+07  2.047057e+07

                4770    Total Bags    Small Bags    Large Bags    XLarge Bags  \
count   1.824900e+04  1.824900e+04  1.824900e+04  1.824900e+04   18249.000000
mean    2.283974e+04  2.396392e+05  1.821947e+05  5.433809e+04    3106.426507
std     1.074641e+05  9.862424e+05  7.461785e+05  2.439660e+05   17692.894652
min     0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00       0.000000
25%     0.000000e+00  5.088640e+03  2.849420e+03  1.274700e+02       0.000000
50%     1.849900e+02  3.974383e+04  2.636282e+04  2.647710e+03       0.000000
75%     6.243420e+03  1.107834e+05  8.333767e+04  2.202925e+04     132.500000
max     2.546439e+06  1.937313e+07  1.338459e+07  5.719097e+06  551693.650000

                year
count  18249.000000
mean    2016.147899
std        0.939938
min     2015.000000
25%     2015.000000
50%     2016.000000
75%     2017.000000
max     2018.000000
```