CS167: Machine Learning

Notebook and Google Colab
Pandas library introduction

Wednesday, August 26th, 2026

Alimoor Reza | Associate Professor | CS Department | Drake University

Recap

- Course introduction and syllabus
- Briefly talked about ML topics
- Other logistics

Recap: An Example: Classifying my dog

- Imagine we want to classify which image depicts a specific dog we want to identify
- Our training data might look something like this:

Recap: An Example: Classifying my dog

- Then, when we have some new pictures of my dogs, the idea is that we can make a **prediction** based on previous data as to whether it is **Zoey** or **Ace** in the photo.

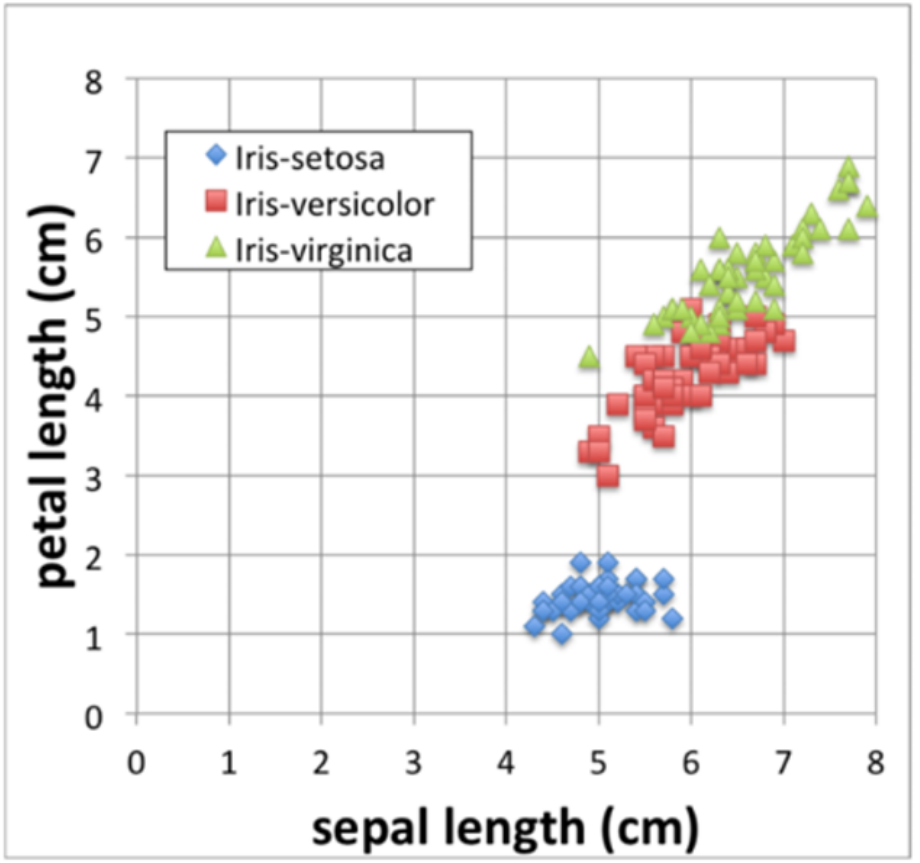
Another Example: What species of Iris is this?

- Use the poll link to answer this question

Imagine you found this beautiful flower while on a walk and took the following measurements:

5.1 cm petal length
7.2 cm sepal length

What species do you think it is?



Machine Learning Variations

- We are going to learn about a lot of different types of machine learning in CS167. Here are a few categories to look out for:

  - **classification:** identify which category it goes in, eg, *'Spam or ham?', 'Eric or Tim?', 'Fish, amphibian, reptile, bird, or mammal'*
  - **regression**: real-valued labels eg, price of Bitcoin, tomorrow's temperature, etc.
  - **supervised learning:** data has labels, goal is to predict the labels of new instance
  - **unsupervised learning:** data does not have a label, the goal is to analyze/ cluster the examples
  - **other issues:** missing data, sequential data, outlier anomaly detection, and many more

Group Exercise

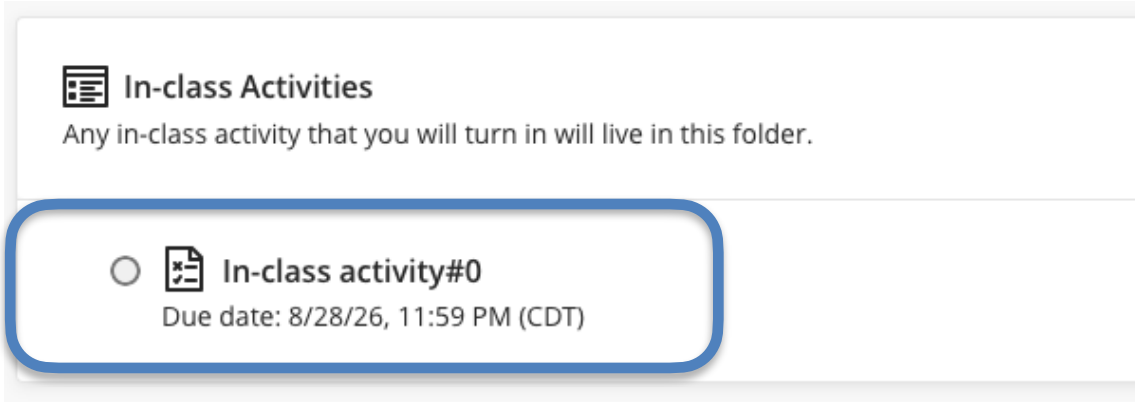
- Group Exercise:
  - Take 2 minutes to talk to your neighbors about the following:
    - Come up with as many examples as you can of ways you interact with machine learning on a day-to-day basis.
    - Submit your answers to the following Google form:
      - https://forms.gle/w19WLdLkmEpnQ1HE9

In-class activity#0

- Blackboard Task
  - Try to finish **In-class activity#0** before next lecture but definitely by Friday

In-class activity#0

- Blackboard Task
  - Finish In-class activity#0 before Friday



New Content

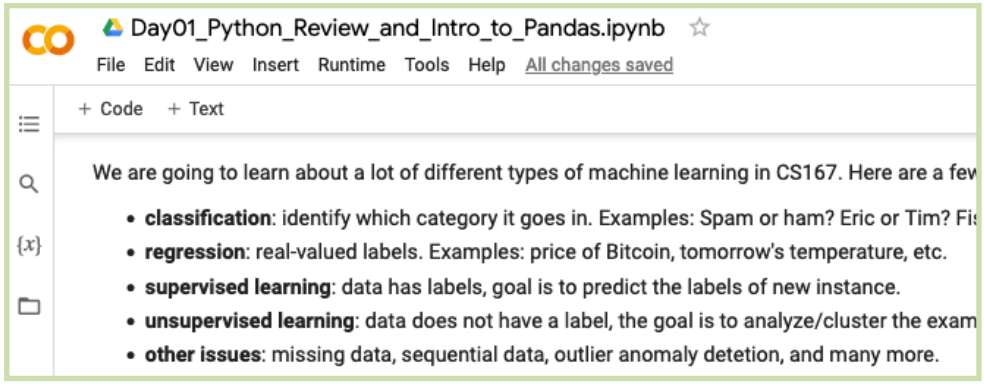
- Introduction to Google Colab
- Python Lab
- Introduction to Pandas (a library in Python)

Introduction to Google Colab

- What is Google colab?

  - Colab is a great browser-based tool for developing iPython Notebooks. Colab allows anybody to write and execute Python code through the browser, and is especially well suited to machine learning, data analysis and education.

  - It makes it really easy to open up someone else's notebook and run it yourself. It also makes it easy to annotate and explain what is happening in the code using Markdown cells.



Other similar tool: Jupyter Notebook

- Google Colab
  - Access to GPUs (Graphics Processing Units), which are very helpful for working with deep learning models. We will explore this later in the semester.
  - Pre-installed packages are available, but you can also install additional ones.
  - A cloud-based service provided by Google that runs on their servers.

vs.

- Jupyter Notebook
  - you can run your code offline (without an internet connection).
  - you can install your Python packages and have complete control over the libraries.
  - **Additional hassle:** you need to install Python and Jupyter in your machine/laptop

Google Colab Demonstration

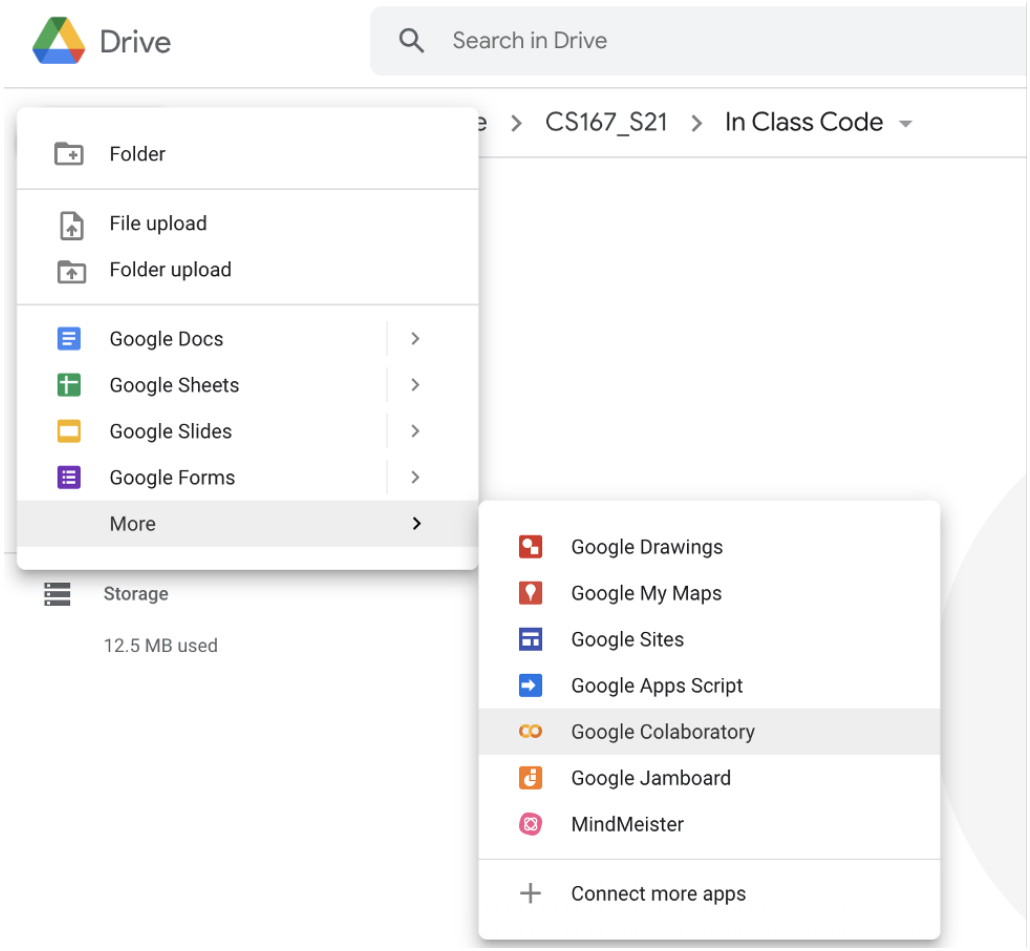
- Switch to Google Colab demo …

Let's get started!

- First step: create a new notebook
- There are two ways to do this:
  - From Google Drive: https://drive.google.com/
  - From Colab: https://colab.research.google.com/

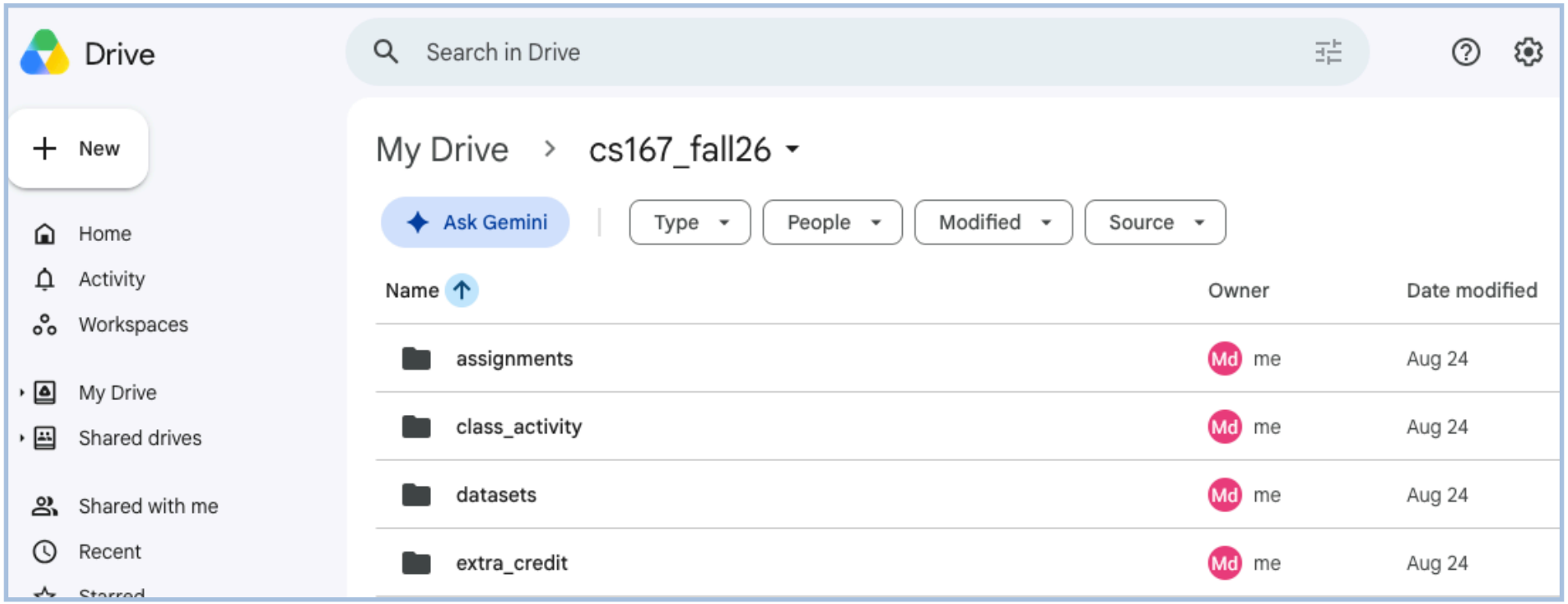
Create a new notebook

- From Google Drive: https://drive.google.com/
- You may need to 'connect more apps', and select Google Colaboratory.



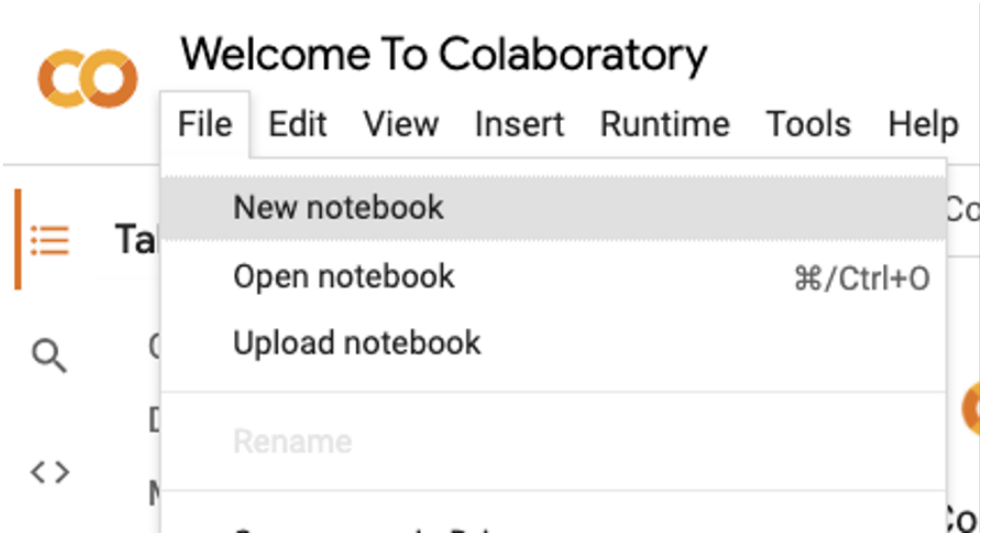
Create a new notebook: via Colab

- **Location of the saved file:** The notebook will be saved whatever location you want eg, I picked cs167_fall26' directory on my Google Drive



Create a new notebook

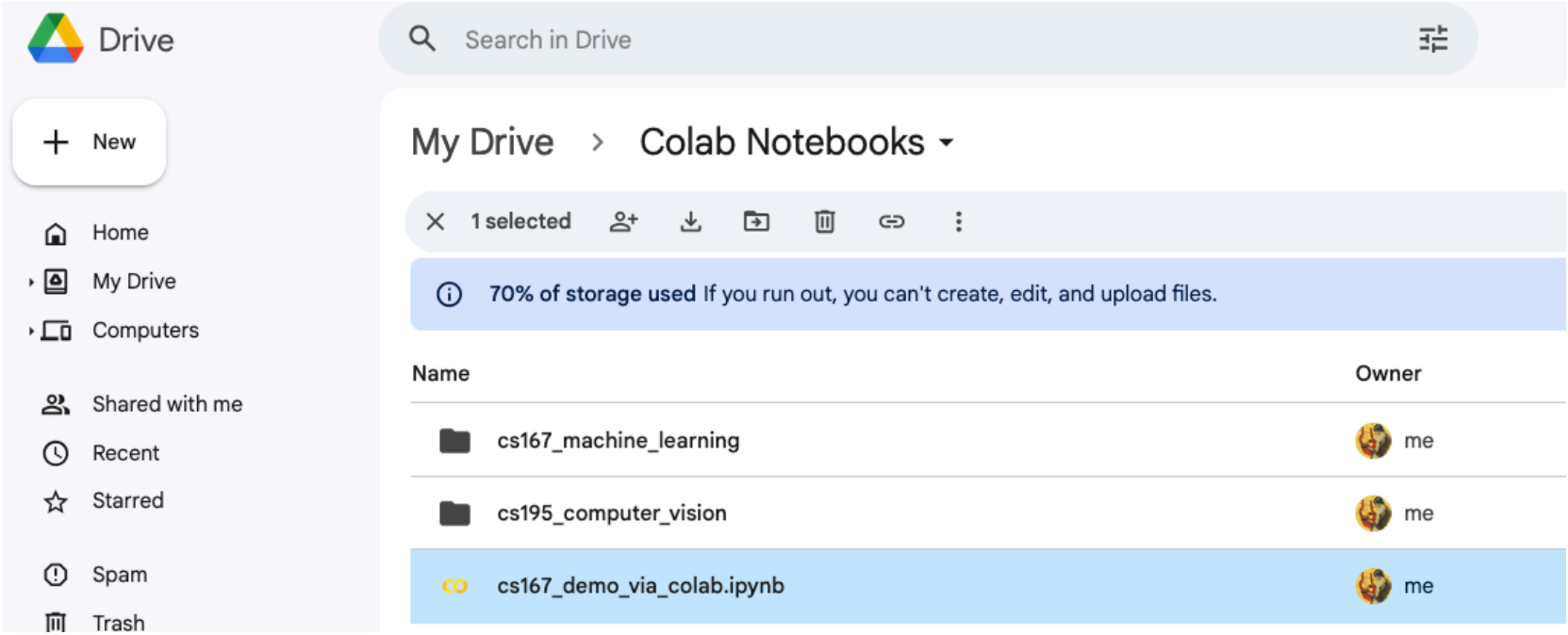
- From Colab: https://colab.research.google.com/



- Location of the saved file: The notebook will be saved in the 'Colab' directory on your Google Drive

Create a new notebook: via Colab

- **Location of the saved file:** The notebook will be saved inside the 'Colab Notebooks' directory on your Google Drive



Today's Agenda

- Introduction to Google Colab
- Python Lab
- Introduction to Pandas (a library in Python)

Let’s do some Python activities on Colab

- On Blackboard, you should find a file called **`Day02_Notes.txt`** This is the outline for today's Python Refresher course.

- For the rest of the class period, here's what I'm expecting:
  - Please don't just copy, paste, and run the code…
    - take a second before running it and discuss with your group what you think is going to happen. This really helps us to understand where our underlying assumptions are incorrect.

Let’s do some Python activities on Colab

- For the rest of the class period, here's what I'm expecting:
  - When the outcome is different than you expected, dig in, and try to figure out why. If you're not sure why, raise your hand, and I'll come help explain
    - what does this difference say about how Python works?
  - Answer the questions from the slides in text cells in your notebook.
    - if something was of interest, add a text cell and describe what was of note.

Wait, what am I supposed to be doing?

- Make sure you give your notebook a name (maybe `Day02_notes.ipynb`), and save it to your CS167-Notes directory. Your workflow for the rest of class should look something like this:

  - you should have the `Day02_Notes.txt` file open, as well as your Colab Notebook.
  - Copy a section of text from the .txt file and paste it into a new cell in your Colab Notebook.
  - Take a minute and look over the code and predict what will happen. Some cells have specific instructions as to what you should be trying to predict.
  - Run the cell, and see if your prediction was correct.
    - If so, great! Move on.
    - If not, even better--you get to dig into why your expectations were different than how it actually worked, which is a great opportunity to learn something new :)

  - Move on to the next cell and repeat!

Wait, what am I supposed to be doing?

- For the Python review, I'm going to let you all go through it at your own pace, but I will interrupt and go over some of the `pandas` material when I notice people ready to get into it. I'll be walking around to provide moral support, nudges in the (hopefully) correct direction
    - Feel free to flag me down if you get stuck, or don't understand how/why something is working.

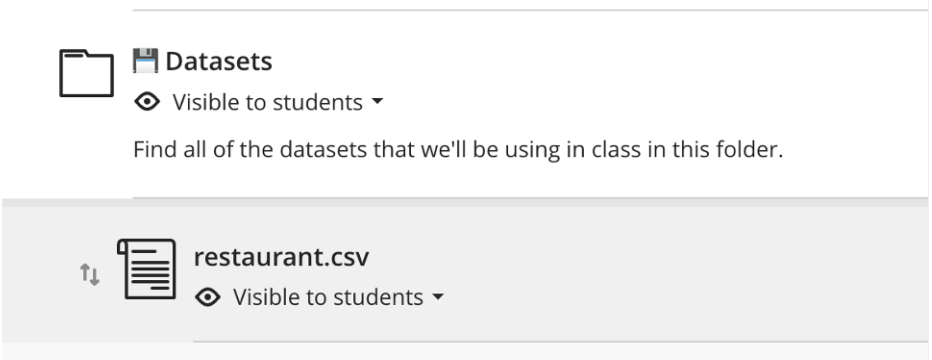
Today's Agenda

- Introduction to Google Colab
- Python Lab
- Introduction to Pandas (a library in Python)

Accessing Data

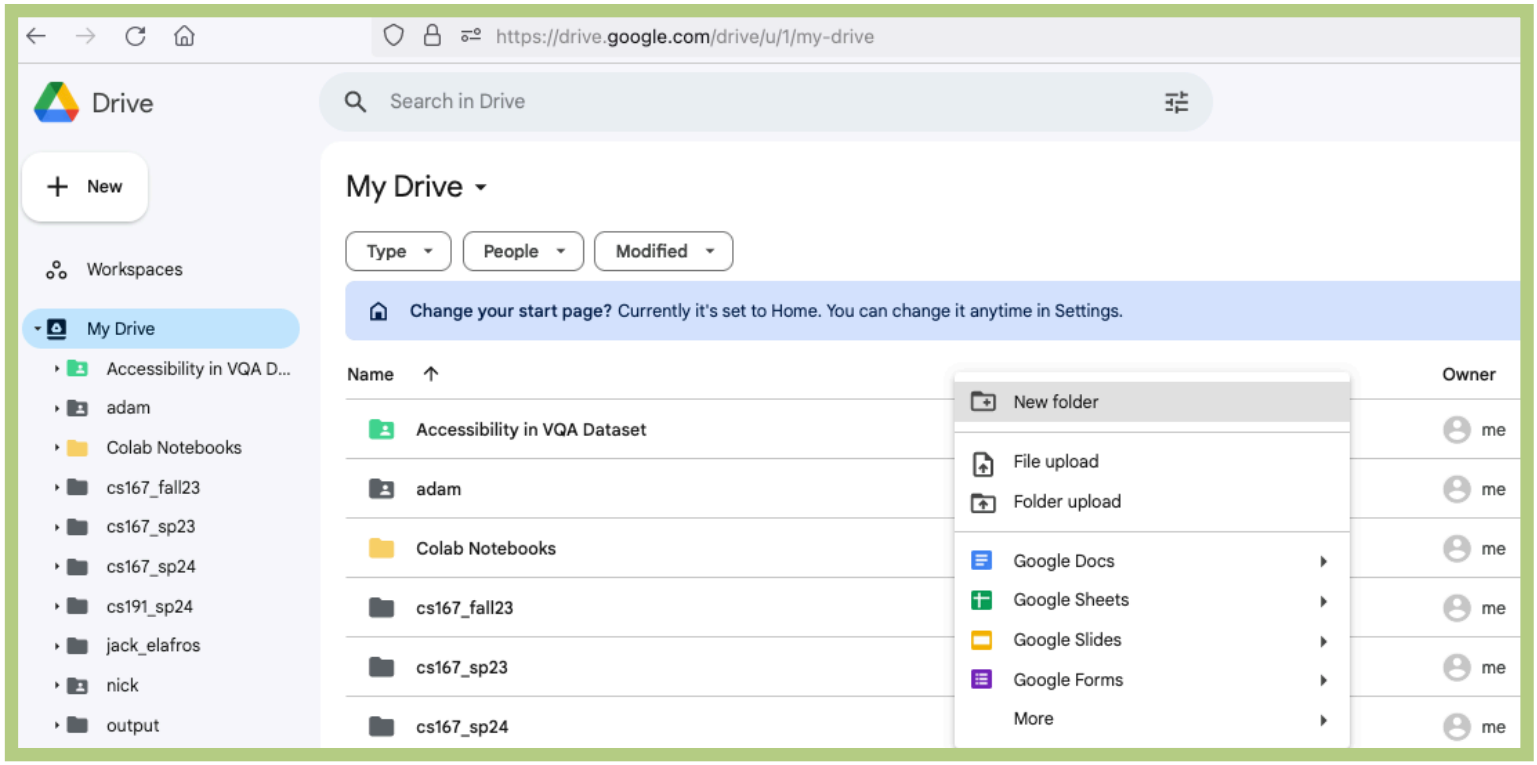
- Google Colab is a cloud-based tool, which means that we need to store our data in the cloud as well. We cannot simply reference our local data and expect it to work.

- Go ahead and download the `"restaurant.csv"` file from Blackboard. It is in the Datasets folder.



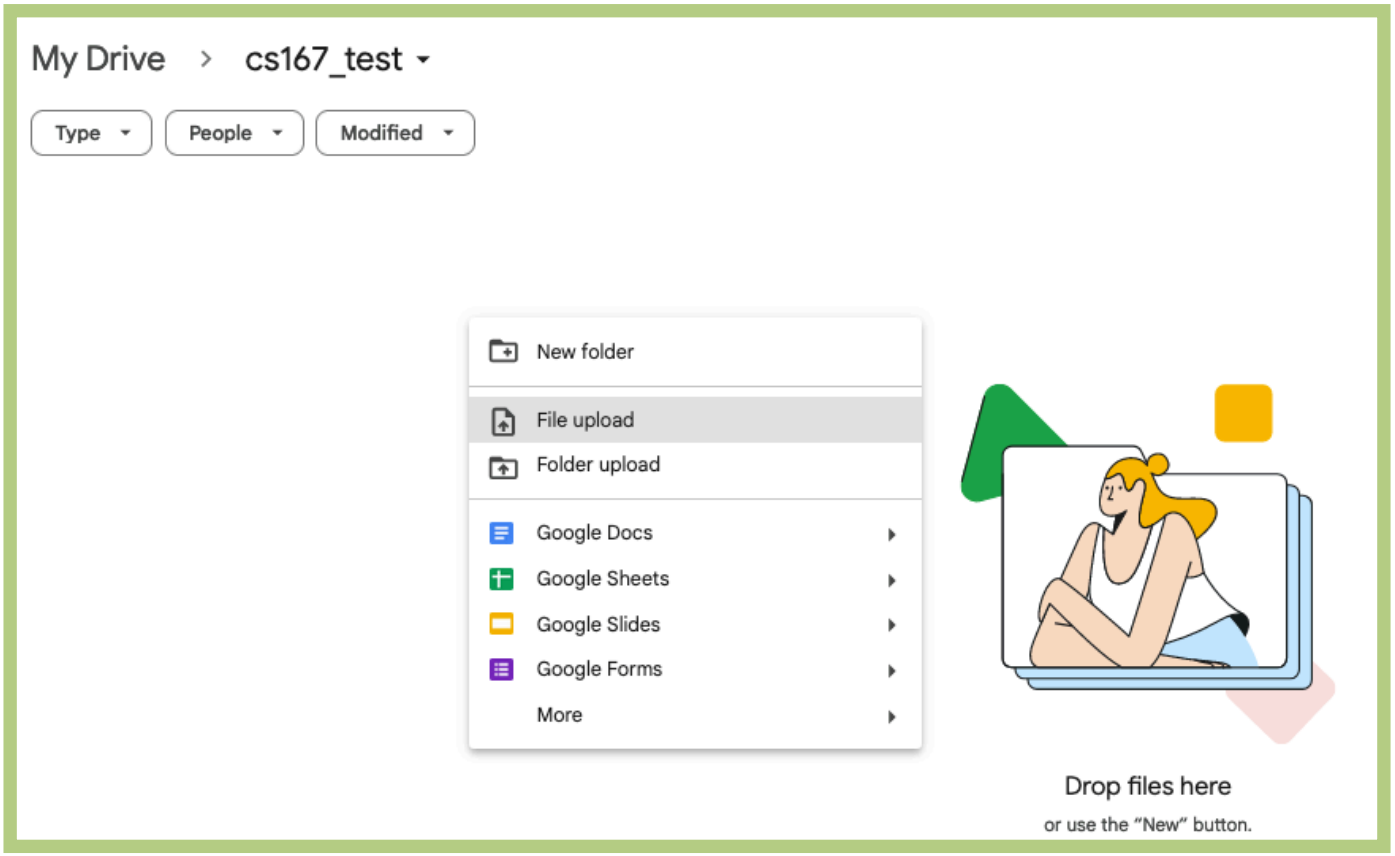
Uploading File to Google Drive

- Upload the `"restaurant.csv"` to your Google Drive.
  - First go to: drive.google.com
  - Then, create a directory/folder (by right-clicking your mouse) as shown below:



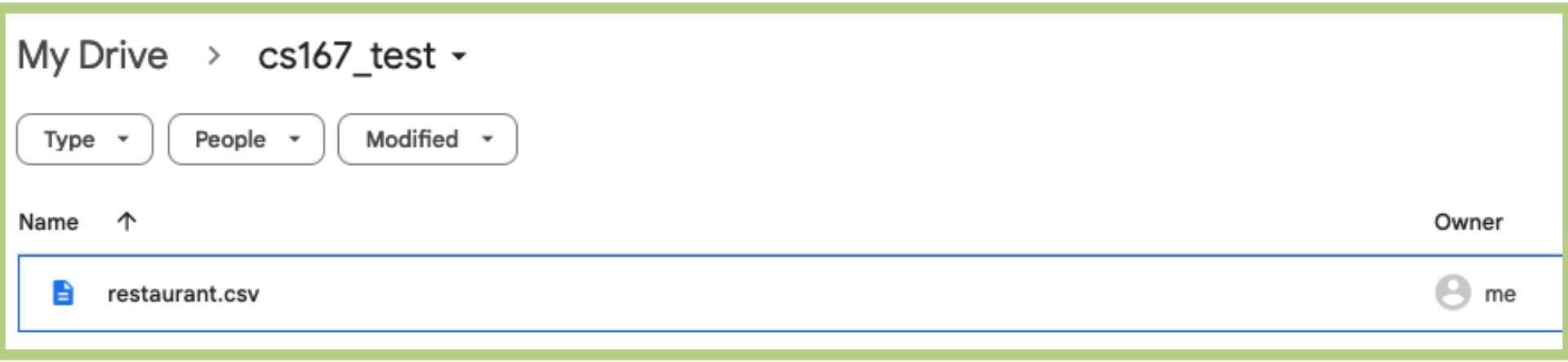
Uploading File to Google Drive

- Upload the `"restaurant.csv"` to your Google Drive.
  - Go to the directory (eg, ‘cs167_test’) and then click the ‘File upload’ option as follows:



Uploading File to Google Drive

- Upload the `"restaurant.csv"` to your Google Drive.
  - Result of your upload looks as follows:



Accessing Data

- To access this file in Google Colab, you'll need a little bit of code.

```
[ ]  # The first step is to mount your Google Drive to your Colab account.
     #You will be asked to authorize Colab to access your Google Drive. Follow the steps they lead you through.

     #this will only work in Google Colab.

     from google.colab import drive
     drive.mount('/content/drive')
```

- Do a demonstration …

Accessing Data

- `pandas` is a super powerful Python data analysis library.
  - built on top of another powerful library called `numpy`
  - `Fun fact:` "Pan-da" name is derived from the term "**panel data**"

- `pandas` should already be installed when you use Google Colab. If you see `In [*]` next to a cell, it means your computer is working on the task

Creating a DataFrame

- There are two main datatypes in `pandas`, the first of which is a `DataFrame`

- Pandas Documentation defines `DataFrames` as:

  - *'Two-dimensional, size-mutable, potentially heterogeneous tabular data'*

  - basically, think of `DataFrames` as our excel sheets — two dimensional, tabular data
  - Each column has a name, and you can use these names to filter and create subsets of data
  - often, you'll see `DataFrames` abbreviated to `df`

DataFrame Demonstration

```
[ ]  #in colab, your path should look something like this:
     path = '/content/drive/MyDrive/CS167/datasets/restaurant.csv' #'dataset/restaurant.csv'
     data = pd.read_csv(path)
     print("data is a ", type(data))
```

Helpful Method Alert: `df.head()`

- The `.head()` method can be called on any DataFrame, and by default will display the first 5 lines rows of the data, as well as the names of the columns.
  - if you want it to display more than 5 rows, you can provide a number as an argument to the method.

- In iPython notebooks, whatever you leave at the end of a cell will automatically output.

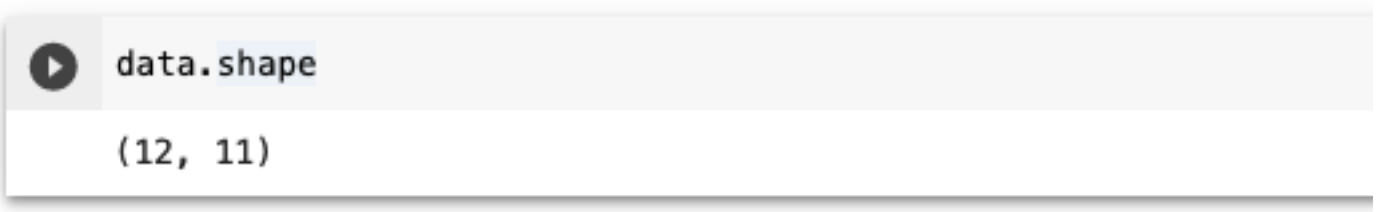
- So, when you put those two facts together, you get this nifty functionality:

```
[ ] data.head()
```

	alt	bar	fri	hun	pat	price	rain	res	type	est	target
0	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
1	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
2	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
3	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes
4	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No

DataFrame Shape

- Want to know the dimensions of your DataFrame? Use `df.shape`



Today's Agenda

- Introduction to Google Colab
- Python Lab
- Introduction to Pandas (a library in Python)
  - Rows in a DataFrame
  - Subsetting (Columns, Rows, or both) in a DataFrame
    - Select subset of Columns in a DataFrame
    - Select subset of Rows in a DataFrame
    - Select **subsets** of the DataFrame (both rows and columns)

Selecting Rows in DataFrames using `loc` and `iloc`:

- Simply put:
  - `loc` gets DataFrame rows and columns by **labels/names**
  - `iloc` gets DataFrame rows and columns by **index/position**

Selecting Rows in DataFrames: `loc`

- Let's read the dataset and try `loc` to get items by rows **labels/names**

```
# load a new csv file 'titanic.csv'. you can find it on Blackboard under datasets module
path = '/content/drive/MyDrive/cs167_fall24/datasets/titanic.csv'

# read the file into a dataframe
df_titanic = pd.read_csv(path)
print('data.shape: ', df_titanic.shape)
df_titanic.head()
```

```
data.shape:  (891, 15)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

labels/names (not indices)

Selecting Rows in DataFrames: `loc`

- `loc` gets DataFrame rows and columns by **labels/names**
- Let's take a subset of titanic and try to use `loc`:

```
[51] subset = df_titanic.loc[800:805] # since it's a label, it will take rows labeled 800, 801, 802, 803, 804, and 805.
     print(subset)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
800	0	2	male	34.00	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
801	1	2	female	31.00	1	1	26.2500	S	Second	woman	False	NaN	Southampton	yes	False
802	1	1	male	11.00	1	2	120.0000	S	First	child	False	B	Southampton	yes	False
803	1	3	male	0.42	0	1	8.5167	C	Third	child	False	NaN	Cherbourg	yes	False
804	1	3	male	27.00	0	0	6.9750	S	Third	man	True	NaN	Southampton	yes	True
805	0	3	male	31.00	0	0	7.7750	S	Third	man	True	NaN	Southampton	no	True

labels/names (not indices)

ALERT: `print(subset)` shows all 6 rows

Selecting Rows in DataFrames: `loc`

- `loc` gets DataFrame rows and columns by **labels/names**
- Let's take a subset of titanic and try to use `loc` and `iloc`:

```
subset = df_titanic.loc[800:805]
subset.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
800	0	2	male	34.00	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
801	1	2	female	31.00	1	1	26.2500	S	Second	woman	False	NaN	Southampton	yes	False
802	1	1	male	11.00	1	2	120.0000	S	First	child	False	B	Southampton	yes	False
803	1	3	male	0.42	0	1	8.5167	C	Third	child	False	NaN	Cherbourg	yes	False
804	1	3	male	27.00	0	0	6.9750	S	Third	man	True	NaN	Southampton	yes	True

labels/names (not indices)

ALERT: `subset.head()` only shows the first 5 rows.

Selecting Rows in DataFrames: `loc`

- `loc` gets DataFrame rows and columns by **labels/names**

```
[51] subset = df_titanic.loc[800:805] # since it's a label, it will take rows labeled 800, 801, 802, 803, 804, and 805.
     print(subset)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
800	0	2	male	34.00	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
801	1	2	female	31.00	1	1	26.2500	S	Second	woman	False	NaN	Southampton	yes	False
802	1	1	male	11.00	1	2	120.0000	S	First	child	False	B	Southampton	yes	False
803	1	3	male	0.42	0	1	8.5167	C	Third	child	False	NaN	Cherbourg	yes	False
804	1	3	male	27.00	0	0	6.9750	S	Third	man	True	NaN	Southampton	yes	True
805	0	3	male	31.00	0	0	7.7750	S	Third	man	True	NaN	Southampton	no	True

- What would happen if I do the following?

```
subset.loc[800]
```

```
survived                0
pclass                  2
sex                  male
age                  34.0
sibsp                   0
parch                   0
fare                 13.0
embarked                S
class              Second
who                   man
adult_male           True
deck                  NaN
embark_town    Southampton
alive                  no
alone                True
Name: 800, dtype: object
```

Selecting Rows in DataFrames: `loc`

- `loc` gets DataFrame rows and columns by **labels/names**

```
[51] subset = df_titanic.loc[800:805] # since it's a label, it will take rows labeled 800, 801, 802, 803, 804, and 805.
     print(subset)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
800	0	2	male	34.00	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
801	1	2	female	31.00	1	1	26.2500	S	Second	woman	False	NaN	Southampton	yes	False
802	1	1	male	11.00	1	2	120.0000	S	First	child	False	B	Southampton	yes	False
803	1	3	male	0.42	0	1	8.5167	C	Third	child	False	NaN	Cherbourg	yes	False
804	1	3	male	27.00	0	0	6.9750	S	Third	man	True	NaN	Southampton	yes	True
805	0	3	male	31.00	0	0	7.7750	S	Third	man	True	NaN	Southampton	no	True

- What would happen if I do the following?

```
subset.loc[805]
```

```
survived                0
pclass                  3
sex                  male
age                  31.0
sibsp                   0
parch                   0
fare                7.775
embarked                S
class               Third
who                   man
adult_male           True
deck                  NaN
embark_town    Southampton
alive                   no
alone                 True
Name: 805, dtype: object
```

Selecting Rows in DataFrames: `loc`

- `loc` gets DataFrame rows and columns by **labels/names**

```
[51] subset = df_titanic.loc[800:805] # since it's a label, it will take rows labeled 800, 801, 802, 803, 804, and 805.
     print(subset)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
800	0	2	male	34.00	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
801	1	2	female	31.00	1	1	26.2500	S	Second	woman	False	NaN	Southampton	yes	False
802	1	1	male	11.00	1	2	120.0000	S	First	child	False	B	Southampton	yes	False
803	1	3	male	0.42	0	1	8.5167	C	Third	child	False	NaN	Cherbourg	yes	False
804	1	3	male	27.00	0	0	6.9750	S	Third	man	True	NaN	Southampton	yes	True
805	0	3	male	31.00	0	0	7.7750	S	Third	man	True	NaN	Southampton	no	True

- What would happen if I do the following?

```
subset.loc[806] #
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/range.py in get_loc(self, key, method, tolerance)
    390                 try:
--> 391                     return self._range.index(new_key)
    392                 except ValueError as err:

ValueError: 806 is not in range

The above exception was the direct cause of the following exception:
```

Selecting Rows in DataFrames: `iloc`

- `iloc` gets DataFrame rows and columns by **index/position**

```
[51] subset = df_titanic.loc[800:805] # since it's a label, it will take rows labeled 800, 801, 802, 803, 804, and 805.
     print(subset)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
800	0	2	male	34.00	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
801	1	2	female	31.00	1	1	26.2500	S	Second	woman	False	NaN	Southampton	yes	False
802	1	1	male	11.00	1	2	120.0000	S	First	child	False	B	Southampton	yes	False
803	1	3	male	0.42	0	1	8.5167	C	Third	child	False	NaN	Cherbourg	yes	False
804	1	3	male	27.00	0	0	6.9750	S	Third	man	True	NaN	Southampton	yes	True
805	0	3	male	31.00	0	0	7.7750	S	Third	man	True	NaN	Southampton	no	True

```
subset.iloc[0]  #works
```

```
survived                0
pclass                  2
sex                  male
age                  34.0
sibsp                   0
parch                   0
fare                 13.0
embarked                S
class              Second
who                   man
adult_male           True
deck                  NaN
embark_town    Southampton
alive                  no
alone                True
Name: 800, dtype: object
```

```
subset.iloc[1]  #works
```

```
survived                1
pclass                  2
sex                female
age                  31.0
sibsp                   1
parch                   1
fare                26.25
embarked                S
class              Second
who                 woman
adult_male          False
deck                  NaN
embark_town    Southampton
alive                  yes
alone                False
Name: 801, dtype: object
```

```
subset.iloc[5]  #works
```

```
survived                0
pclass                  3
sex                  male
age                  31.0
sibsp                   0
parch                   0
fare                7.775
embarked                S
class              Third
who                   man
adult_male           True
deck                  NaN
embark_town    Southampton
alive                   no
alone                True
Name: 805, dtype: object
```

Exercise:
Practice Time, Your Turn

- Try `loc/iloc` on your Google Colab notebook using different name or indices!

Today's Agenda

- Introduction to Google Colab
- Python Lab
- Introduction to Pandas (a library in Python)
  - Rows in a DataFrame
  - Subsetting (Columns, Rows, or both) in a DataFrame
    - Select subset of Columns in a DataFrame
    - Select subset of Rows in a DataFrame
    - Select subsets of the DataFrame (both rows and columns)

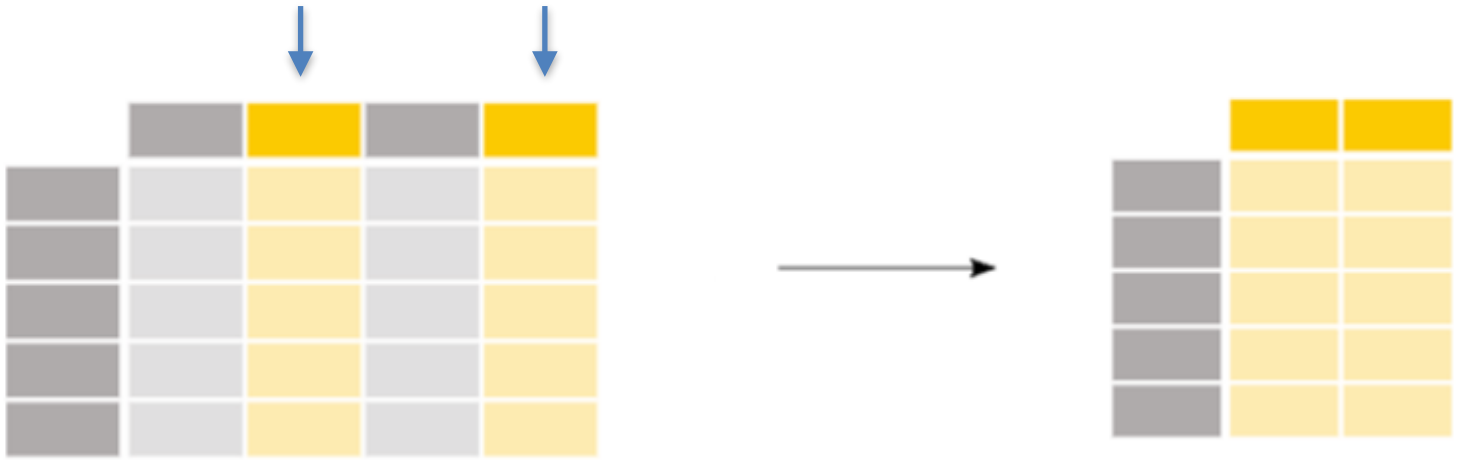
Today's Agenda

- Rows in a DataFrame

- Subsetting (Columns, Rows, or both) in a DataFrame
  - Select subset of Columns in a DataFrame
  - Select subset of Rows in a DataFrame
  - Select **subsets** of the DataFrame (both rows and columns)

Subsetting Columns in a DataFrame

- Why might we want a subset of the columns of a DataFrame?



Subsetting Columns in a DataFrame

- Sometimes you don't need all of the columns and just want to work with a **subset** of the columns of the original dataset.

- Other times, you may want to reorder the columns in your dataset.

- Here's how you would do either of those: The syntax for subsetting columns from a DataFrame (`df`) is:

  - One column: `df['column_name']`

  - Multiple columns: `df[['column1', 'column2', 'target']]`

Subsetting Columns in a DataFrame

- So, if we wanted to look at the **`price`** column, we could do:

```
#you should be able to run this without any issue.
import pandas as pd

path = "/content/drive/MyDrive/cs167_sp26/datasets/restaurant.csv"
df_rest = pd.read_csv(path)
print(df_rest)
```

```
...     alt  bar  fri  hun   pat price rain  res     type    est target
0   Yes   No   No  Yes  Some   $$$   No  Yes   French   0-10    Yes
1   Yes   No   No  Yes  Full     $   No   No     Thai  30-60     No
2    No  Yes   No   No  Some     $   No   No   Burger   0-10    Yes
3   Yes   No  Yes  Yes  Full     $   No   No     Thai  10-30    Yes
4   Yes   No  Yes   No  Full   $$$   No  Yes   French    >60     No
5    No  Yes   No  Yes  Some    $$  Yes  Yes  Italian   0-10    Yes
6    No  Yes   No   No   NaN     $  Yes   No   Burger   0-10     No
7    No   No   No  Yes  Some    $$  Yes  Yes     Thai   0-10    Yes
8    No  Yes  Yes   No  Full     $  Yes   No   Burger    >60     No
9   Yes  Yes  Yes  Yes  Full   $$$   No  Yes  Italian  10-30     No
10   No   No   No   No   NaN     $   No   No     Thai   0-10     No
11  Yes  Yes  Yes  Yes  Full     $   No   No   Burger  30-60    Yes
```

```
prices = df_rest['price']
prices.head()
```

	price
0	$$$
1	$
2	$
3	$
4	$$$

Subsetting Columns in a DataFrame

- Imagine you want to only work with '**rain**', '**hun**', '**target**'

	alt	bar	fri	hun	pat	price	rain	res	type	est	target
0	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
1	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
2	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
3	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes
4	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No

```
prices = df_rest[ ['rain', 'hun', 'target'] ]
prices.head()
```

	rain	hun	target
0	No	Yes	Yes
1	No	Yes	No
2	No	No	Yes
3	No	Yes	Yes
4	No	No	No

Subsetting Columns in a DataFrame

- Re-order your new subset so that **`rain`** `and` **`hun`** `switched`

	alt	bar	fri	hun	pat	price	rain	res	type	est	target
0	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
1	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
2	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
3	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes
4	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No

```
prices_reordered = df_rest[ ['hun', 'rain', 'target'] ]
prices_reordered.head()
```

	hun	rain	target
0	Yes	No	Yes
1	Yes	No	No
2	No	No	Yes
3	Yes	No	Yes
4	No	No	No

Exercise:
Practice Time, Your Turn

💬 Group Exercise:

Download the Titanic Dataset from Blackboard, upload it to a spot in your GoogleDrive; you might have already done so by now.

See if you can make the following subsets:

- make a subset called `ages` that holds the ages of the passengers on the titanic
- create a subset called `titanic_subset` with the columns `survived`, `deck`, `sex`, and `age`, in that order.

```
[ ]  # your code
     # ...
```