# CS167: Machine Learning

Pandas Tutorial

Tuesday, February 6th, 2024

Drake UNIVERSITY

# Recap

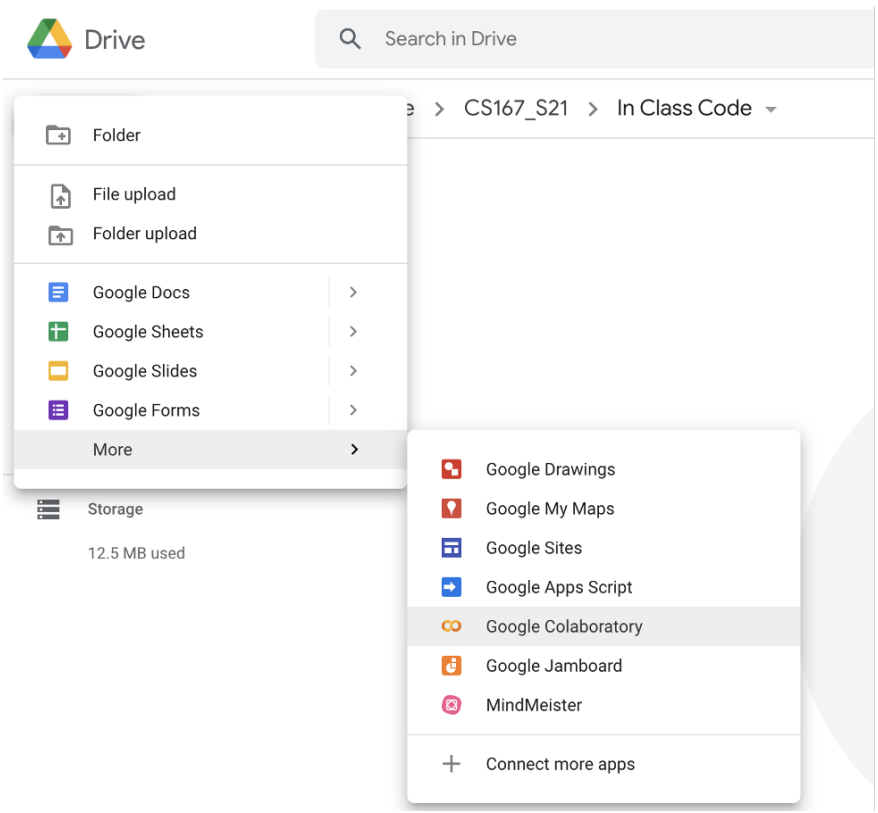
- Machine Learning variations
- Introduction to Google Colab
- Python Lab
- Accessing Data

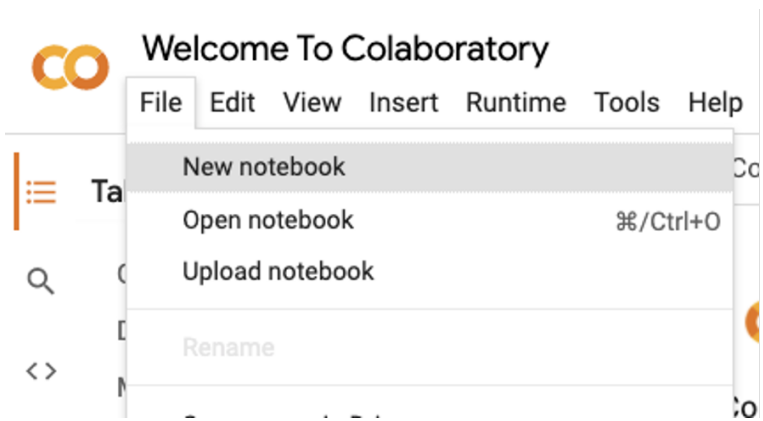
# Recap: Machine Learning Variations

- 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

# Recap: 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/



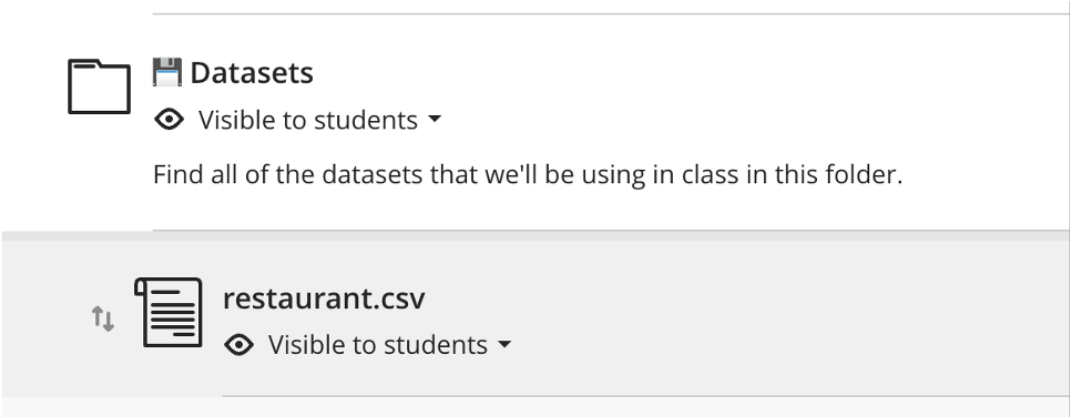
# Recap: Python Lab

- Make sure you give your notebook a name (maybe `Day01_notes.ipynb`), and save it to your CS167-Notes Github repository. Your workflow for the rest of class should look something like this:
  - you should have the `Day01_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!

# Recap: Accessing Data

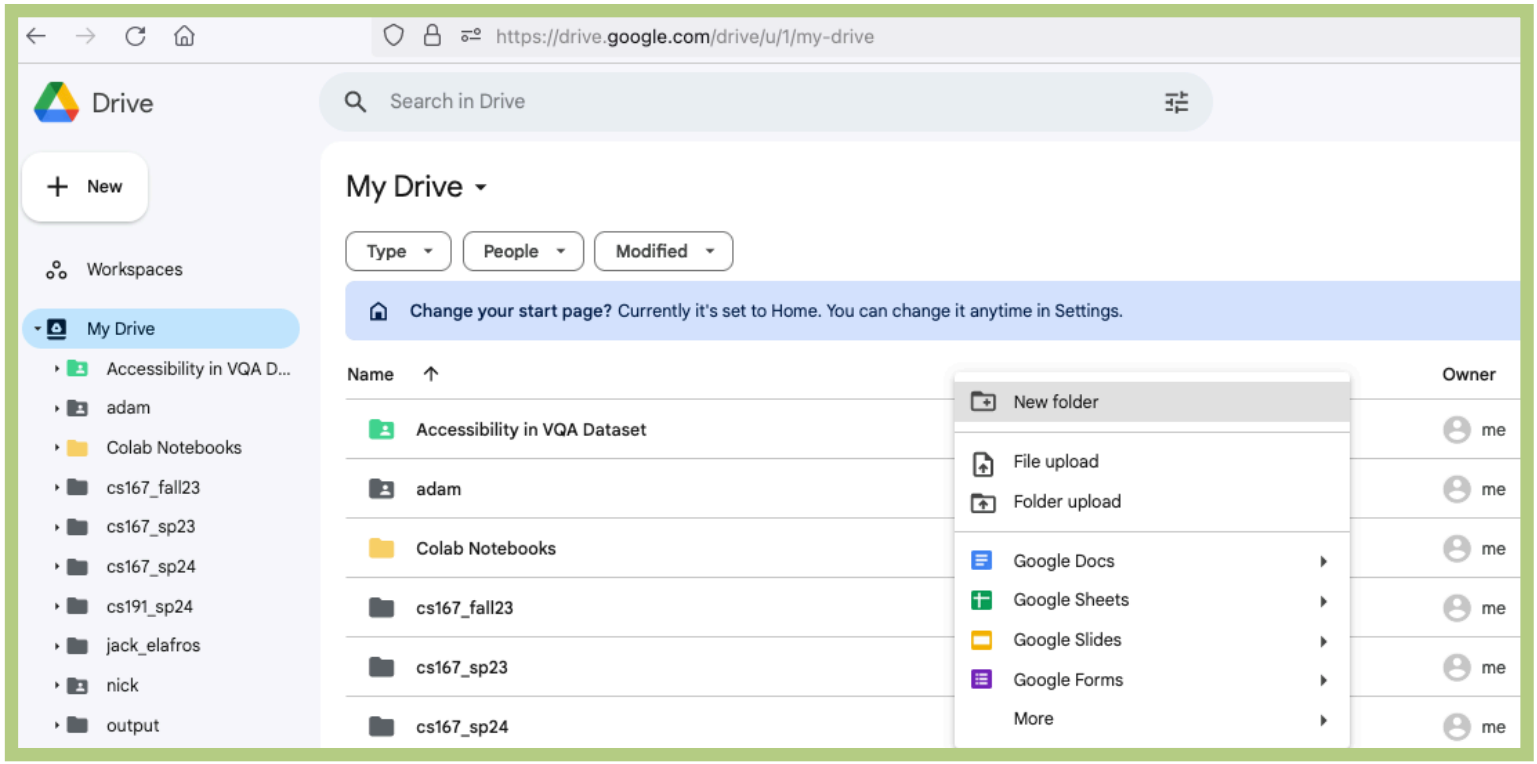
- Google Colab is a cloud-based tool, which means that we need to store our data in the cloud as well. We cannot simply reference our local data and expect it to work.

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



# Recap: Uploading File to Google Drive

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



# Recap: 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 …

# Today's Agenda

- Topics:
  - Introduction to Pandas (a library in Python)
  - Subsetting (Columns, Rows, or both) in a DataFrame

# Accessing Data

- **Pandas** is a super powerful Python data analysis library.
  - it's built on top of another powerful library called `numpy`

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

# Overview of Pandas Tutorial

- Overview of Pandas
  - Datatypes `DataFrame` and `Series`
  - helpful functions

- Other goals are as follows:
  - Select **columns** in DataFrames
  - Select **rows** in DataFrames
  - Select **subsets** of the DataFrame (both rows and columns)

# Pandas Datatypes: `DataFrame` and `Series`

- In `pandas`, there are two main datatypes
  - `DataFrame`
  - `Series`

# Pandas Datatypes: `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`

# Other ways of creating `DataFrame`

- The syntax for creating a `DataFrame` from scratch looks like this:
  - `pandas.DataFrame(data, index, columns)`

```
df = pd.DataFrame() # creates an empty DataFrame
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

# Other ways of creating `DataFrame`

- The syntax for creating a `DataFrame` from scratch looks like this:

```
[15] data = [10, 20, 30, 40, 50, 60]
     df_1 = pd.DataFrame(data, columns=['numbers'])
     print('size of the dataframe df_2', df_1.shape)
     df_1
```

```
size of the dataframe df_2 (6, 1)
```

| | numbers |
|---|---|
| 0 | 10 |
| 1 | 20 |
| 2 | 30 |
| 3 | 40 |
| 4 | 50 |
| 5 | 60 |

```
data = {'col1':[1,2,3], 'col2':[4,5,6], 'col3':[7,8,9]}
df_3 = pd.DataFrame(data)
print('size of the dataframe df_2', df_2.shape)
df_3
```

```
size of the dataframe df_2 (3, 3)
```

| | col1 | col2 | col3 |
|---|---|---|---|
| 0 | 1 | 4 | 7 |
| 1 | 2 | 5 | 8 |
| 2 | 3 | 6 | 9 |

# Creating DataFrame from 2d list

```python
# initialize list of lists
data = [['reza', 1], ['chris', 2], ['eric', 3]]

# Create the pandas DataFrame
df_3 = pd.DataFrame(data, columns=['name', 'score'])

# print dataframe.
df_3
```

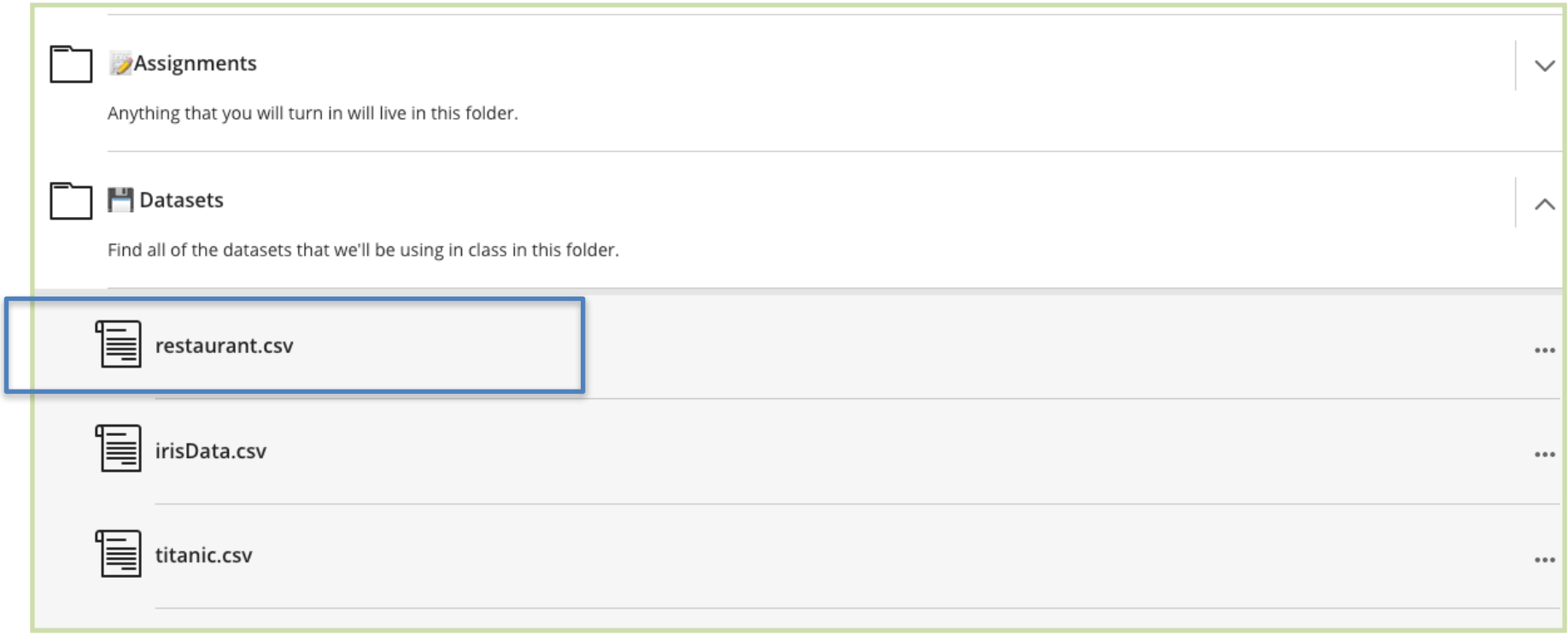
| | name | score |
|---|---|---|
| 0 | reza | 1 |
| 1 | chris | 2 |
| 2 | eric | 3 |

# Practice Time: Your Turn

- Try DataFrames on Google Colab for the next 10 minutes!

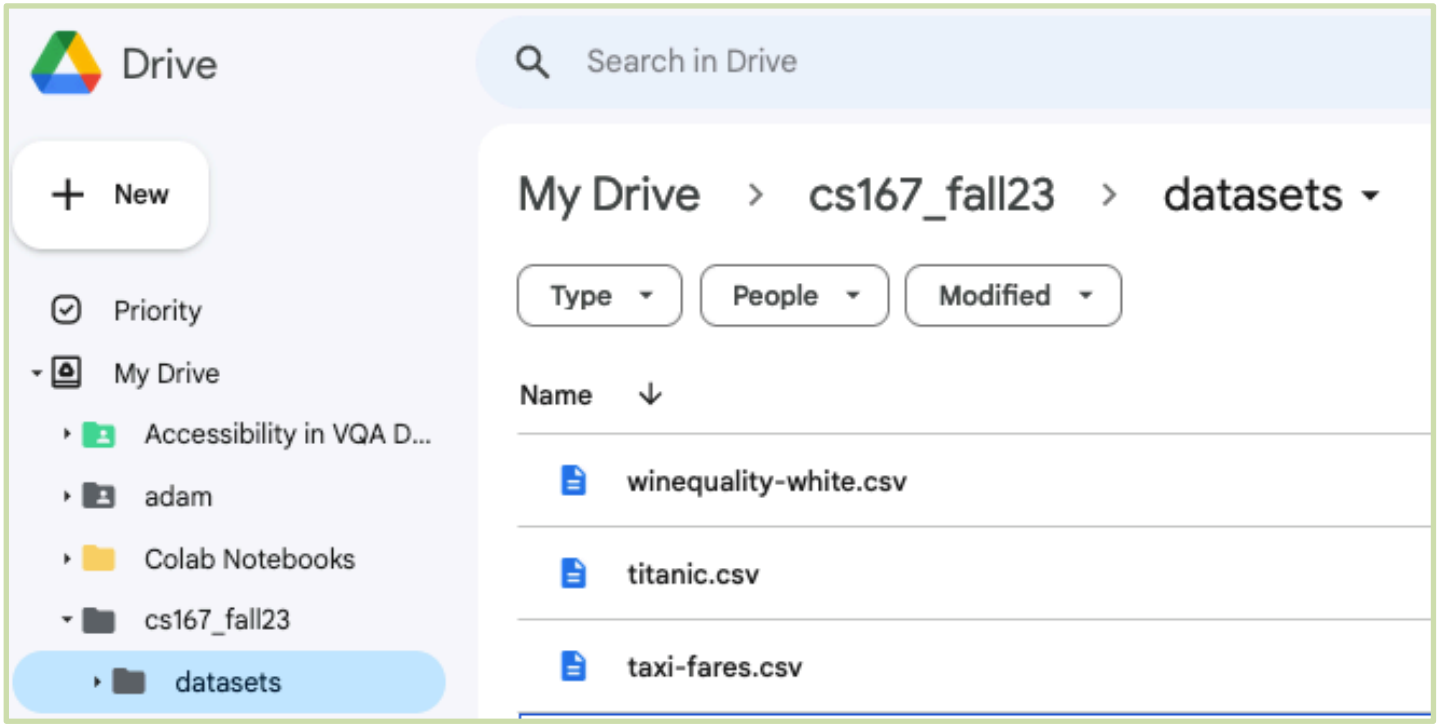
# Creating `DataFrame` by reading from a file

- Download a sample .csv file (eg, ‘restaurant.csv’) from Blackboard



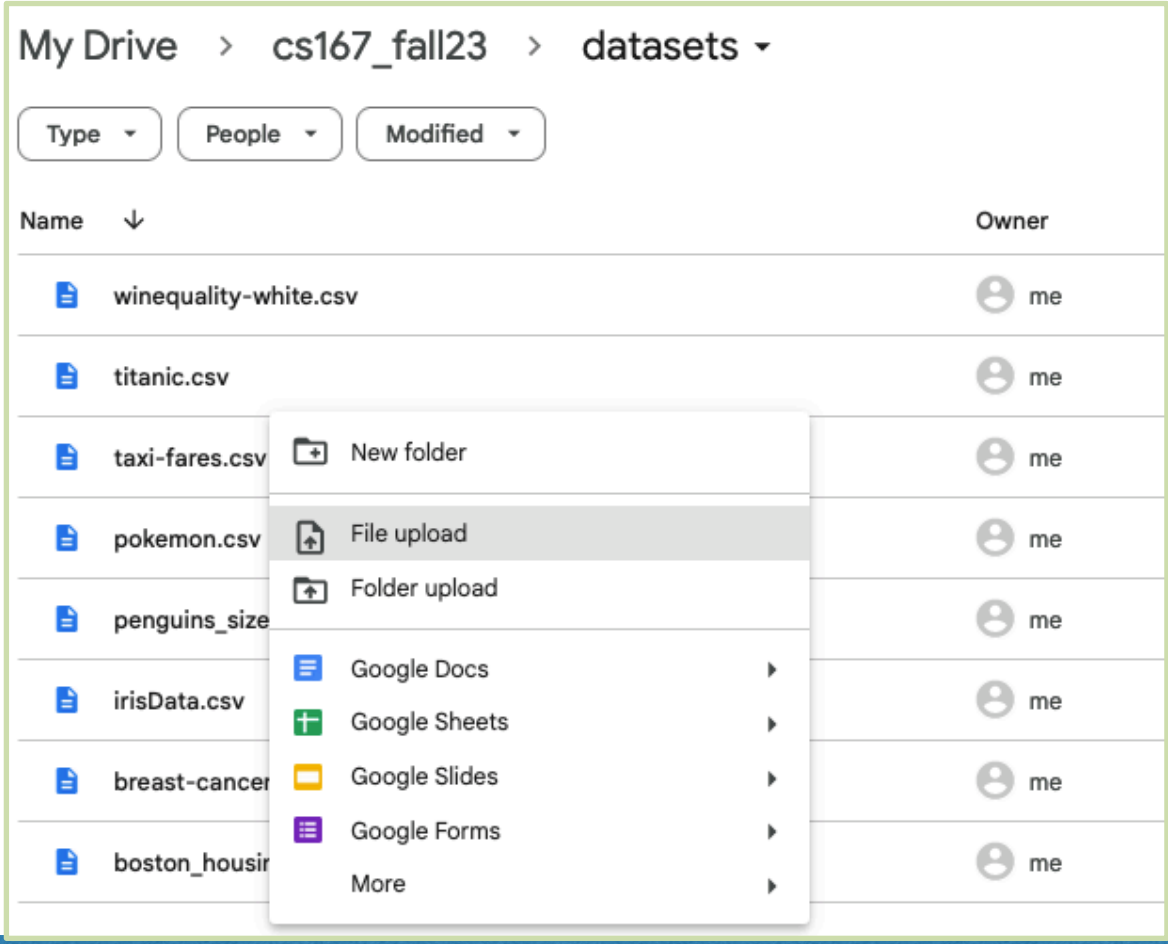
# Creating `DataFrame` by reading from a file

- Connect to your Google Drive and create a folder eg, ‘datasets’. For example, I have the following layout:



# Creating `DataFrame` by reading from a file

- Now upload this `restaurant.csv` on your Google Drive (use right-click on your mouse)



# Don’t forget to mount Google Drive

- 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')
```

- We did this last week

# Creating `DataFrame` by reading from a file

- You will be able to show the path of `restaurant.csv` on your Google Drive as follows:

```
#change this path to point to where your data is:
# if you're using colab it should be something like: '/content/drive/MyDrive/CS167/datasets/restaurant.csv'

import pandas as pd
path = '/content/drive/MyDrive/cs167_fall23/datasets/restaurant.csv'

restaurant_data = pd.read_csv(path)
print('data is a ', type(restaurant_data))
```

```
data is a  <class 'pandas.core.frame.DataFrame'>
```

# Practice Time: Your Turn

- Finish file uploading process on Google Drive and then read from your restaurant file for the next 10 minutes!

# 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.

```
[20] #change this path to point to where your data is:
     # if you're using colab it should be something like below:
     path = '/content/drive/MyDrive/cs167_fall23/datasets/restaurant.csv'

     # read the data from the csv file
     df_4 = pd.read_csv(path)

     # show the dataframe
     df_4.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 |

# Helpful Method Alert: `df.shape()`

- The `.shape()` method can be called on any `DataFrame`, and it will show the dimensions ie, *number of rows* and *number of columns*

```
[22] df_4.shape

     (12, 11)
```

# Column Names

- Want to see a list of all of the column names in your dataset? Try using `df.columns`

```
[24] col = df_4.columns
     col

     Index(['alt', 'bar', 'fri', 'hun', 'pat', 'price', 'rain', 'res', 'type',
            'est', 'target'],
           dtype='object')
```

- If there are no spaces in the name of a column, you can also reference it using dot notation like so: `df.type`

```
df_4.type

0      French
1        Thai
2      Burger
3        Thai
4      French
5     Italian
6      Burger
7        Thai
8      Burger
9     Italian
10       Thai
11     Burger
Name: type, dtype: object
```

# Practice Time: Your Turn

- Try these useful DataFrame methods on Google Colab  for the next 10 minute!

# 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`

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

```python
# load a new csv file 'titanic.csv'. you can find it on Blackboard under datasets module
path = '/content/drive/MyDrive/cs167_fall23/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 |

# 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

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

# Selecting Rows in DataFrames: `loc`

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

```
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 |

- 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**

```
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 |

- 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**

```
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 |

- 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**

```
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 |

```
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
```

# Practice Time: Your Turn

- Try `loc/iloc` on Google Colab!

# Pandas Datatypes: `DataFrame` and `Series`

- In `pandas`, there are two main datatypes
  - `DataFrame`
  - `Series`

# Pandas Datatypes: `Series`

- Pandas Documentation defines `Series` as:
  - `Series` are 1D arrays with axis labels
    - Each **row** in a DataFrame is a `Series`
    - Each **column** in a DataFrame is also a `Series`.

```
print(type(restaurant_data.iloc[0])) #the first row in the dataframe
print(type(restaurant_data['type'])) #the column 'type' from the dataframe
```

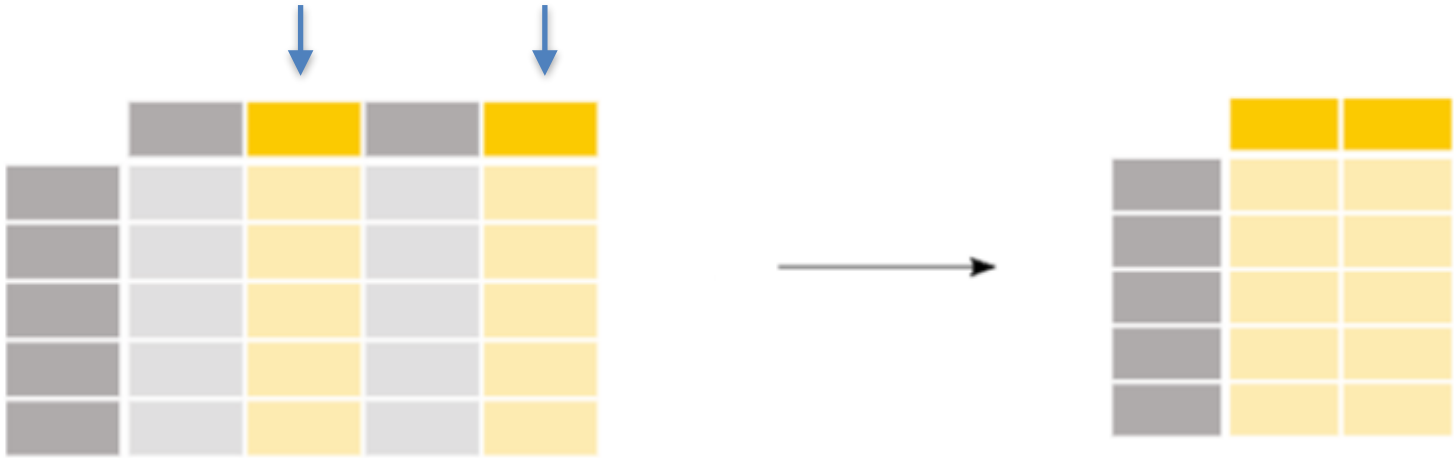
```
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
```

# Today's Agenda

- Topics:
  - Introduction to Pandas (a library in Python)
  - Subsetting (Columns, Rows, or both) in a DataFrame

# 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:

```
import pandas as pd
path = '/content/drive/MyDrive/cs167_fall23/datasets/restaurant.csv'
restaurant_data = pd.read_csv(path)
print('data is a ', type(restaurant_data))
restaurant_data.head()
```

```
data is a  <class 'pandas.core.frame.DataFrame'>
```

| | 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 = restaurant_data['price']
prices
```

# Subsetting Columns in a DataFrame

- So, if we wanted to look at the **alt, fri, pat** columns, we could do:

| | 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 = restaurant_data[[ 'alt','fri','pat']]
prices
```

| | alt | fri | pat |
|---|---|---|---|
| 0 | Yes | No | Some |
| 1 | Yes | No | Full |
| 2 | No | No | Some |
| 3 | Yes | Yes | Full |
| 4 | Yes | Yes | Full |
| 5 | No | No | Some |
| 6 | No | No | None |
| 7 | No | No | Some |
| 8 | No | Yes | Full |
| 9 | Yes | Yes | Full |
| 10 | No | No | None |
| 11 | Yes | Yes | Full |

# 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 |

```
col_subset = restaurant_data[['rain', 'hun', 'target']]
col_subset.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 |

```
reordered = col_subset[['hun', 'rain', 'target']]
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 |

# Group Exercise

- Download the Titanic Dataset from Blackboard (which we already did earlier), upload it to a spot in your GoogleDrive, and 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