About this Book

Contents

Resources

- Glossary
- References on Python
- How Tos
- Cheatsheet
- Data Sources
- General Tips and Resources
- How to Study in this class
- Getting Help with Programming
- Terminals and Environments
- Getting Organized for class
- Advice from FA2020 Students
- Advice from FA2021 Students
- Letters to Future students

Welcome to the course manual for CSC310 at URI with Professor Brown.

This class meets MWF 2-2:50pm in Ranger 302.

This website will contain the syllabus, class notes, and other reference material for the class.

Course Calendar on BrightSpace

> **Tip**
>
> subscribe to that calendar in your favorite calendar application

Navigating the Sections

The Syllabus section has logistical operations for the course broken down into sections. You can also read straight through by starting in the first one and navigating to the next section using the arrow navigation at the end of the page.

This site is a resource for the course. We do not follow a text book for this course, but all notes from class are posted in the notes section, accessible on the left hand side menu, visible on large screens and in the menu on mobile.

The resources section has links and short posts that provide more context and explanation. Content in this section is for the most part not strictly the material that you'll be graded on, but it is often material that will help you understand and grow as a programmer and data scientist.

Reading each page

All class notes can be downloaded in multiple formats, including as a notebook. Some pages of the syllabus and resources are also notebooks, if you want to see behind the curtain of how I manage the course information.

> **Try it Yourself**
>
> Notes will have exercises marked like this

> **Question from Class**
>
> Questions that are asked in class, but unanswered at that time will be answered in the notes and marked with a box like this. Long answers will be in the main notes

> **Further reading**
>
> Notes that are mostly links to background and context will be highlighted like this. These are optional, but will mostly help you understand code excerpts they relate to.

> **Hint**
>
> Both notes and assignment pages will have hints from time to time. Pay attention to these on the notes, they'll typically relate to things that will appear in the assignment.

> **Think Ahead**
>
> Think ahead boxes will guide you to start thinking about what can go into your portfolio to build on the material at hand.

> **Question from class**
>
> Questions that are asked in class, but unanswered at that time will be answered in the notes and marked with a box like this. Short questions will be in the margin note

About

About this course

Data science exists at the intersection of computer science, statistics, and machine learning. That means writing programs to access and manipulate data so that it becomes available for analysis using statistical and machine learning techniques is at the core of data science. Data scientists use their data and analytical ability to find and interpret rich data sources; manage large amounts of data despite hardware, software, and bandwidth constraints; merge data sources; ensure consistency of datasets; create visualizations to aid in understanding data; build mathematical models using the data; and present and communicate the data insights/findings.

This course provides a survey of data science. Topics include data driven programming in Python; data sets, file formats and meta-data; descriptive statistics, data visualization, and foundations of predictive data modeling and machine learning; accessing web data and databases; distributed data management. You will work on weekly substantial programming problems such as accessing data in database and visualize it or build machine learning models of a given data set.

Basic programming skills (CSC201 or CSC211) are a prerequisite to this course. This course is a prerequisite course to machine learning, where you learn how machine learning algorithms work. In this course, we will start with a very fast review of basic programming ideas, since you've already done that before. We will learn how to *use* machine learning algorithms to do data science, but not how to *build* machine learning algorithms, we'll use packages that implement the algorithms for us.

About this semester

This semester is a lot of new things for all of us. This course will be completely online all semester, so we will get to use a single instructional format all semester, including when all campus activities move remote after Thanksgiving. I recognize that those last two weeks of the semester may change your obligations with siblings, parents, work, etc. In light of that, we will cover all of the most important topics and you will have the opportunity to achieve all of the course learning outcomes before Thanksgiving. The material in the last two weeks of the semester will be more advanced, likely interesting and definitely useful material, but if your ability to participate in class is less at that time, it will not hurt your grade.

About this syllabus

This syllabus is a *living* document and accessible from BrightSpace, as a pdf for download directly online at rhodyprog4ds.github.io/BrownFall20/syllabus. If you choose to download a copy of it, note that it is only a copy. You can get notification of changes from GitHub by "watching" the repository. You can view the date of changes and exactly what changes were made on the Github commits page.

Creating an issue on the repository is also a good way to ask questions about anything in the course it will prompt additions and expand the FAQ section.

About your instructor

Name: Dr. Sarah Brown Office hours: TBA via zoom, link in BrightSpace

Dr. Brown is an Assistant Professor of Computer Science, who does research on how social context changes machine learning. Dr. Brown earned a PhD in Electrical Engineering from Northeastern University, completed a postdoctoral fellowship at University of California Berkeley, and worked as a postdoctoral research associate at Brown University before joining URI. At Brown University, Dr. Brown taught the Data and Society course for the Master's in Data Science Program.

The best way to contact me is e-mail or by dropping into my office hours. Please include `[CSC310]` or `[DSP310]` in the subject line of your email along with the topic of your message. This is important, because your messages are important, but I also get a lot of e-mail. Consider these a cheat code to my inbox: I have setup a filter that will flag your e-mail if you use one of those in the subject to ensure that I see it. I rarely check e-mail between 6pm and 9am, on weekends or holidays. You might see me post or send things during these hours, but I will not reliably see emails that arrive during those hours.

> **Note**
>
> Whether you use CSC or DSP does not matter.

Tools and Resources

We will use a variety of tools to conduct class and to facilitate your programming. You will need a computer with Linux, MacOS, or Windows. It is unlikely that a tablet will be able to do all of the things required in this course. A Chromebook may work, especially with developer tools turned on. Ask Dr. Brown if you need help getting access to an adequate computer.

All of the tools and resources below are either:

- paid for by URI **OR**
- freely available online.

BrightSpace

This will be the central location from which you can access all other materials. Any links that are for private discussion among those enrolled in the course will be available only from our course Brightspace site.

This is also where your grades will appear and how I will post announcements.

For announcements, you can customize how you receive them.

> **Important**
>
> TL;DR [1]
>
> - check Brightspace
> - Log in to Prismia Chat
> - Make a GitHub Account
> - Install Python
> - Install Git

Prismia chat

Our class link for Prismia chat is available on Brightspace. We will use this for chatting and in-class understanding checks.

On Prismia, all students see the instructor's messages, but only the Instructor and TA see student responses.

> **Note**
>
> Seeing the BrightSpace site requires loging in with your URI SSO and being enrolled in the course

Course Website

The course manual will have content including the class policies, scheduling, class notes, assignment information, and additional resources. This will be linked from Brightspace and available publicly online at rhodyprog4ds.github.io/BrownFall22/. Links to the course reference text and code documentation will also be included here in the assignments and class notes.

GitHub

You will need a GitHub Account. If you do not already have one, please create one by the first day of class. If you have one, but have not used it recently, you may need to update your password and login credentials as the Authentication rules changed over the summer. In order to use the command line with https, you will need to create a Personal Access Token for each device you use. In order to use the command line with SSH, set up your public key.

Programming Environment

This a programming course, so you will need a programming environment. In order to complete assignments you need the items listed in the requirements list. The easiest way to meet these requirements is to follow the recommendations below. I will provide instruction assuming that you have followed the recommendations.

Requirements:

- Python with scientific computing packages (numpy, scipy, jupyter, pandas, seaborn, sklearn)
- Git
- A web browser compatible with Jupyter Notebooks

> **Warning**
>
> Everything in this class will be tested with the up to date (or otherwise specified) version of Jupyter Notebooks. Google Colab is similar, but not the same, and some things may not work there. It is an okay backup, but should not be your primary work environment.

Recommendation:

- Install python via Anaconda
- if you use Windows, install Git with GitBash (video instructions).
- if you use MacOS, install Git with the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this by trying to run git from the Terminal the very first time.`git --version`
- if you use Chrome OS, follow these instructions:

1. Find Linux (Beta) in your settings and turn that on.
2. Once the download finishes a Linux terminal will open, then enter the commands: sudo apt-get update and sudo apt-get upgrade. These commands will ensure you are up to date.
3. Install tmux with:

```
sudo apt -t stretch-backports install tmux
```

4. Next you will install nodejs, to do this, use the following commands:

```
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash
sudo apt-get install -y nodejs
sudo apt-get install -y build-essential.
```

5. Next install Anaconda's Python from the website provided by the instructor and use the top download link under the Linux options.
6. You will then see a .sh file in your downloads, move this into your Linux files.
7. Make sure you are in your home directory (something like home/YOURUSERNAME), do this by using the `pwd` command.
8. Use the `bash` command followed by the file name of the installer you just downloaded to start the installation.
9. Next you will add Anaconda to your Linux PATH, do this by using the `vim .bashrc` command to enter the .bashrc file, then add the `export PATH=/home/YOURUSERNAME/anaconda3/bin/:$PATH` line. This can be placed at the end of the file.
10. Once that is inserted you may close and save the file, to do this hold escape and type `:x`, then press enter. After doing that you will be returned to the terminal where you will then type the source .bashrc command.
11. Next, use the `jupyter notebook –generate-config` command to generate a Jupyter Notebook.
12. Then just type `jupyter lab` and a Jupyter Notebook should open up.

Optional:

- Text Editor: you may want a text editor outside of the Jupyter environment. Jupyter can edit markdown files (that you'll need for your portfolio), in browser, but it is more common to use a text editor like Atom or Sublime for this purpose.

Video install instructions for Anaconda:

- Windows
- Mac

On Mac, to install python via environment, this article may be helpful

- I don't have a video for linux, but it's a little more straight forward.

Textbook

The texts for this class are references and for context and will not be a source of assignments. Both are available free online, but are also relatively affordable if you want a hard copy.

Think Like a Data Scientist

It will be a helpful reference and you may be directed there for answers to questions or alternate explanations ot topics.

Python for Data Science is available free online:

Zoom (backup only, Fall 2021 is in person)

This is where we will meet if for any reason we cannot be in person. You will find the link to class zoom sessions on Brightspace.

URI provides all faculty, staff, and students with a paid Zoom account. It *can* run in your browser or on a mobile device, but you will be able to participate in class best if you download the Zoom client on your computer. Please log in and configure your account. Please add a photo of yourself to your account so that we can still see your likeness in some form when your camera is off. You may also wish to use a virtual background and you are welcome to do so.

Class will be interactive, so if you cannot be in a quiet place at class time, headphones with a built in microphone are strongly recommended.

For help, you can access the instructions provided by IT.

---

[1] Too long; didn't read.

> **Note**
>
> all Git instructions will be given as instructions for the command line interface and GitHub specific instructions via the web interface. You may choose to use GitHub desktop or built in IDE tools, but the instructional team may not be able to help.

> **A tip from Dr. Brown**
>
> I use atom, but I decided to use it by downloading both Atom and Sublime and trying different things in each for a week. I liked Atom better after that and I've stuck with it since. I used Atom to write all of the content in this syllabus. VScode will also work, if needed

Data Science Achievements

In this course there are 5 learning outcomes that I expect you to achieve by the end of the semester. To get there, you'll focus on 15 smaller achievements that will be the basis of your grade. This section will describe how the topics covered, the learning outcomes, and the achievements are covered over time. In the next section, you'll see how these achievements turn into grades.

Learning Outcomes

By the end of the semester

1. (process) Describe the process of data science, define each phase, and identify standard tools
2. (data) Access and combine data in multiple formats for analysis
3. (exploratory) Perform exploratory data analyses including descriptive statistics and visualization
4. (modeling) Select models for data by applying and evaluating mutiple models to a single dataset
5. (communicate) Communicate solutions to problems with data in common industry formats

We will build your skill in the `process` and `communicate` outcomes over the whole semester. The middle three skills will correspond roughly to the content taught for each of the first three portfolio checks.

Schedule

The course will meet MWF 2-2:50pm in Ranger 302. Every class will include participatory live coding (instructor types code while explaining, students follow along)) instruction and small exercises for you to progress toward level 1 achievements of the new skills introduced in class that day.

Each Assignment will have a deadline posted on the page. Portfolio deadlines will be announced at least 2 weeks in advance.

week	topics	skills
1	[admin, python review]	process
2	Loading data, Python review	[access, prepare, summarize]
3	Exploratory Data Analysis	[summarize, visualize]
4	Data Cleaning	[prepare, summarize, visualize]
5	Databases, Merging DataFrames	[access, construct, summarize]
6	Modeling, classification performance metrics, cross validation	[evaluate]
7	Naive Bayes, decision trees	[classification, evaluate]
8	Regression	[regression, evaluate]
9	Clustering	[clustering, evaluate]
10	SVM, parameter tuning	[optimize, tools]
11	KNN, Model comparison	[compare, tools]
12	Text Analysis	[unstructured]
13	Images Analysis	[unstructured, tools]
14	Deep Learning	[tools, compare]

Achievement Definitions

The table below describes how your participation, assignments, and portfolios will be assessed to earn each achievement. The keyword for each skill is a short name that will be used to refer to skills throughout the course materials; the full description of the skill is in this table.

Note

On the Course Calendar on BrightSpace page you can get a feed link to add to the calendar of your choice by clicking on the subscribe (star) button on the top right of the page. Class is for 1 hour there because of Brightspace/zoom integration limitations, but that calendar includes the zoom link.

keyword	skill	Level 1	Level 2	Level 3
python	pythonic code writing	python code that mostly runs, occasional pep8 adherance	python code that reliably runs, frequent pep8 adherance	reliable, efficient, pythonic code that consistently adheres to pep8
process	describe data science as a process	Identify basic components of data science	Describe and define each stage of the data science process	Compare different ways that data science can facilitate decision making
access	access data in multiple formats	load data from at least one format; identify the most common data formats	Load data for processing from the most common formats; Compare and constrast most common formats	access data from both common and uncommon formats and identify best practices for formats in different contexts
construct	construct datasets from multiple sources	identify what should happen to merge datasets or when they can be merged	apply basic merges	merge data that is not automatically aligned
summarize	Summarize and describe data	Describe the shape and structure of a dataset in basic terms	compute summary statndard statistics of a whole dataset and grouped data	Compute and interpret various summary statistics of subsets of data
visualize	Visualize data	identify plot types, generate basic plots from pandas	generate multiple plot types with complete labeling with pandas and seaborn	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters
prepare	prepare data for analysis	identify if data is or is not ready for analysis, potential problems with data	apply data reshaping, cleaning, and filtering as directed	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received
evaluate	Evaluate model performance	Explain basic performance metrics for different data science tasks	Apply and interpret basic model evaluation metrics to a held out test set	Evaluate a model with multiple metrics and cross validation
classification	Apply classification	identify and describe what classification is, apply pre-fit classification models	fit, apply, and interpret preselected classification model to a dataset	fit and apply classification models and select appropriate classification models for different contexts
regression	Apply Regression	identify what data that can be used for regression looks like	fit and interpret linear regression models	fit and explain regrularized or nonlinear regression
clustering	Clustering	describe what clustering is	apply basic clustering	apply multiple clustering techniques, and interpret results
optimize	Optimize model parameters	Identify when model parameters need to be optimized	Optimize basic model parameters such as model order	Select optimal parameters based of mutiple quanttiateve criteria and automate parameter tuning
compare	compare models	Qualitatively compare model classes	Compare model classes in specific terms and fit models in terms of traditional model performance metrics	Evaluate tradeoffs between different model comparison types
representation	Choose representations and transform data	Identify options for representing text and categorical data in many contexts	Apply at least one representation to transform unstructured or inappropriately data for model fitting or summarizing	apply transformations in different contexts OR compare and contrast multiple representations a single type of data in terms of model performance
workflow	use industry standard data science tools and workflows to solve data science problems	Solve well strucutred fully specified problems with a single tool pipeline	Solve well-strucutred, open-ended problems, apply common structure to learn new features of standard tools	Independently scope and solve realistic data science problems OR independently learn releated tools and describe strengths and weakensses of common tools

Assignments and Skills

Using the keywords from the table above, this table shows which assignments you will be able to demonstrate which skills and the total number of assignments that assess each skill. This is the number of opportunities you have to earn Level 2 and still preserve 2 chances to earn Level 3 for each skill.

keyword	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	# Assignments
python	1	1	0	1	1	0	0	0	0	0	0	0	0	4
process	1	0	0	0	0	1	1	1	1	1	1	0	0	7
access	0	1	1	1	1	0	0	0	0	0	0	0	0	4
construct	0	0	0	0	1	0	1	1	0	0	0	0	0	3
summarize	0	0	1	1	1	1	1	1	1	1	1	1	1	11
visualize	0	0	1	1	0	1	1	1	1	1	1	1	1	10
prepare	0	0	0	1	1	0	0	0	0	0	0	0	0	2
evaluate	0	0	0	0	0	1	1	1	0	1	1	0	0	5
classification	0	0	0	0	0	0	1	0	0	1	0	0	0	2
regression	0	0	0	0	0	0	0	1	0	0	1	0	0	2
clustering	0	0	0	0	0	0	0	0	1	0	1	0	0	2
optimize	0	0	0	0	0	0	0	0	0	1	1	0	0	2
compare	0	0	0	0	0	0	0	0	0	0	1	0	1	2
representation	0	0	0	0	0	0	0	0	0	0	0	1	1	2
workflow	0	0	0	0	0	0	0	0	0	1	1	1	1	4

> **Warning**
>
> **process** achievements are accumulated a little slower. Prior to portfolio check 1, only level 1 can be earned. Portfolio check 1 is the first chance to earn level 2 for process, then level 3 can be earned on portfolio check 2 or later.

Portfolios and Skills

The objective of your portfolio submissions is to earn Level 3 achievements. The following table shows what Level 3 looks like for each skill and identifies which portfolio submissions you can earn that Level 3 in that skill.

keyword	Level 3	P1	P2	P3	P4
python	reliable, efficient, pythonic code that consistently adheres to pep8	1	1	0	1
process	Compare different ways that data science can facilitate decision making	0	1	1	1
access	access data from both common and uncommon formats and identify best practices for formats in different contexts	1	1	0	1
construct	merge data that is not automatically aligned	1	1	0	1
summarize	Compute and interpret various summary statistics of subsets of data	1	1	0	1
visualize	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters	1	1	0	1
prepare	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received	1	1	0	1
evaluate	Evaluate a model with multiple metrics and cross validation	0	1	1	1
classification	fit and apply classification models and select appropriate classification models for different contexts	0	1	1	1
regression	fit and explain regrularized or nonlinear regression	0	1	1	1
clustering	apply multiple clustering techniques, and interpret results	0	1	1	1
optimize	Select optimal parameters based of mutiple quanttiateve criteria and automate parameter tuning	0	0	1	1
compare	Evaluate tradeoffs between different model comparison types	0	0	1	1
representation	apply transformations in different contexts OR compare and contrast multiple representations a single type of data in terms of model performance	0	0	1	1
workflow	Independently scope and solve realistic data science problems OR independently learn releated tools and describe strengths and weakensses of common tools	0	0	1	1

Detailed Checklists

python-level1

python code that mostly runs, occasional pep8 adherance

- [ ] logical use of control structures
- [ ] callable functions
- [ ] correct calls to functions
- [ ] correct use of variables
- [ ] use of logical operators

python-level2

python code that reliably runs, frequent pep8 adherance

- [ ] descriptive variable names
- [ ] pythonic loops
- [ ] efficient use of return vs side effects in functions
- [ ] correct, effective use of builtin python iterable types (lists & dictionaries)

python-level3

reliable, efficient, pythonic code that consistently adheres to pep8

- [ ] pep8 adherant variable, file, class, and function names
- [ ] effective use of multi-paradigm abilities for efficiency gains
- [ ] easy to read code that adheres to readability over other rules

process-level1

Identify basic components of data science

- [ ] identify component disciplines OR
- [ ] idenitfy phases

process-level2

Describe and define each stage of the data science process

- [ ] correctly defines stages
- [ ] identifies stages in use
- [ ] describes general goals as well as a specific processes

process-level3

Compare different ways that data science can facilitate decision making

- [ ] describes exceptions to process and iteration in process
- [ ] connects choices at one phase to impacts in other phases
- [ ] connects data science steps to real world decisions

access-level1

load data from at least one format; identify the most common data formats

- [ ] use at least one pandas `read_` function correctly
- [ ] name common types
- [ ] describe the structure of common types

access-level2

Load data for processing from the most common formats; Compare and constrast most common formats

- [ ] load data from at least two of (.csv, .tsv, .dat, database, .json)
- [ ] describe advantages and disadvantages of most commone types
- [ ] descive how most common types are different

access-level3

access data from both common and uncommon formats and identify best practices for formats in different contexts

- [ ] load data from at least 1 uncommon format
- [ ] describe when one format is better than another

construct-level1

identify what should happen to merge datasets or when they can be merged

- [ ] identify what the structure of a merged dataset should be (size, shape, columns)
- [ ] idenitfy when datasets can or cannot be merged

construct-level2

apply basic merges

- [ ] use 3 different types of merges
- [ ] choose the right type of merge for realistic scenarios

construct-level3

merge data that is not automatically aligned

- [ ] manipulate data to make it mergable
- [ ] identify how to combine data from many sources to answer a question
- [ ] implement stesp to combine data from multiple sources

summarize-level1

Describe the shape and structure of a dataset in basic terms

- [ ] use attributes to produce a description of a dataset
- [ ] display parts of a dataset

summarize-level2

compute and interpret summary standard statistics of a whole dataset and grouped data

- [ ] compute descriptive statistics on whole datasets
- [ ] apply individual statistics to datasets
- [ ] group data by a categorical variable for analysis
- [ ] apply split-apply-combine paradigm to analyze data
- [ ] interprete statistics on whole datasets
- [ ] interpret statistics on subsets of data

summarize-level3

Compute and interpret various summary statistics of subsets of data

- [ ] produce custom aggregation tables to summarize datasets
- [ ] compute multivariate summary statistics by grouping
- [ ] compute custom cacluations on datasets

visualize-level1

identify plot types, generate basic plots from pandas

- [ ] generate at least two types of plots with pandas
- [ ] identify plot types by name
- [ ] interpret basic information from plots

visualize-level2

generate multiple plot types with complete labeling with pandas and seaborn

- [ ] generate at least 3 types of plots
- [ ] use correct, complete, legible labeling on plots
- [ ] plot using both pandas and seaborn
- [ ] interpret multiple types of plots to draw conclusions

visualize-level3

generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters

- [ ] use at least two libraries to plot
- [ ] generate figures with subplots
- [ ] customize the display of a plot to be publication ready
- [ ] interpret plot types and explain them for novices
- [ ] choose appopriate plot types to convey information
- [ ] explain why plotting common best practices are effective

prepare-level1

identify if data is or is not ready for analysis, potential problems with data

- [ ] identify problems in a dataset
- [ ] anticipate how potential data setups will interfere with analysis
- [ ] describe the structure of tidy data
- [ ] label data as tidy or not

prepare-level2

apply data reshaping, cleaning, and filtering as directed

- [ ] reshape data to be analyzable as directed
- [ ] filter data as directed
- [ ] rename columns as directed
- [ ] rename values to make data more analyzable
- [ ] handle missing values in at least two ways
- [ ] transform data to tidy format

prepare-level3

apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received

- [ ] identify issues in a dataset and correctly implement solutions
- [ ] convert varialbe representation by changing types
- [ ] change variable representation using one hot encoding

evaluate-level1

Explain basic performance metrics for different data science tasks

- [ ] define at least two performance metrics
- [ ] describe how those metrics compare or compete

evaluate-level2

Apply and interpret basic model evaluation metrics to a held out test set

- [ ] apply at least three performance metrics to models
- [ ] apply metrics to subsets of data
- [ ] apply disparity metrics
- [ ] interpret at least three metrics

evaluate-level3

Evaluate a model with multiple metrics and cross validation

- [ ] explain cross validation
- [ ] explain importance of held out test and validation data
- [ ] describe why cross vaidation is important
- [ ] idenitfy appropriate metrics for different types of modeling tasks
- [ ] use multiple metriccs together to create a more complete description of a model's performance

classification-level1

identify and describe what classification is, apply pre-fit classification models

- [ ] describe what classification is
- [ ] describe what a dataset must look like for classifcation
- [ ] identify appliations of classifcation in the real world
- [ ] describe set up for a classification problem (tes,train)

classification-level2

fit, apply, and interpret preselected classification model to a dataset

- [ ] split data for training and testing
- [ ] fit a classification model
- [ ] apply a classification model to obtain predictions
- [ ] interpret the predictions of a classification model
- [ ] examine parameters of at least one fit classifier to explain how the prediction is made

- differentiate between model fitting and generating predictions
- evaluate how model parameters impact model performance

classification-level3

fit and apply classification models and select appropriate classification models for different contexts

- choose appropriate classifiers based on application context
- explain how at least 3 different classifiers make predictions
- evaluate how model parameters impact model performance and justify choices when tradeoffs are necessary

regression-level1

identify what data that can be used for regression looks like

- identify data that is/not appropriate for regression
- describe univariate linear regression
- identify appliations of regression in the real world

regression-level2

fit and interpret linear regression models

- split data for training and testing
- fit univariate linear regression models
- interpret linear regression models
- fit multivariate linear regression models

regression-level3

fit and explain regrularized or nonlinear regression

- fit nonlinear or regrularized regression models
- interpret and explain nonlinear or regrularized regresion models

clustering-level1

describe what clustering is

- differentiate clustering from classification and regression
- identify appliations of clustering in the real world

clustering-level2

apply basic clustering

- fit Kmeans
- interpret kmeans
- evaluate clustering models

clustering-level3

apply multiple clustering techniques, and interpret results

- apply at least two clustering techniques
- explain the differences between two clustering models

optimize-level1

Identify when model parameters need to be optimized

- identify when parameters might impact model performance

optimize-level2

Optimize basic model parameters such as model order

- automatically optimize multiple parameters
- evaluate potential tradeoffs
- interpret optimization results in context

optimize-level3

Select optimal parameters based of mutiple quanttiateve criteria and automate parameter tuning

- optimize models based on multiple metrics
- describe when one model vs another is most appropriate

compare-level1

Qualitatively compare model classes

- compare models within the same task on complexity

compare-level2

Compare model classes in specific terms and fit models in terms of traditional model performance metrics

- compare models in multiple terms
- interpret cross model comparisons in context

compare-level3

Evaluate tradeoffs between different model comparison types

- compare models on multiple criteria
- compare optimized models
- jointly interpret optimization result and compare models
- compare models on quanttiateve and qualitative measures

representation-level1

Identify options for representing text and categorical data in many contexts

- describe the basic goals for changing the representation of data

representation-level2

Apply at least one representation to transform unstructured or inappropriately data for model fitting or summarizing

- transform text or image data for use with ML

representation-level3

apply transformations in different contexts OR compare and contrast multiple representations a single type of data in terms of model performance

- transform both text and image data for use in ml
- evaluate the impact of representation on model performance

workflow-level1

Solve well strucutred fully specified problems with a single tool pipeline

- pseudocode out the steps to answer basic data science questions

workflow-level2

Solve well-strucutred, open-ended problems, apply common structure to learn new features of standard tools

- plan and execute answering real questions to an open ended question
- describe the necessary steps and tools

workflow-level3

Independently scope and solve realistic data science problems OR independently learn releated tools and describe strengths and weakensses of common tools

- scope and solve realistic data science problems
- compare different data science tool stacks

Grading

This section of the syllabus describes the principles and mechanics of the grading for the course. This course will be graded on a basis of a set of *skills* (described in detail the next section of the syllabus). This is in contrast to more common grading on a basis of points earned through assignments.

Principles of Grading

Learning happens through practice and feedback. My goal as a teacher is for you to learn. The grading in this course is based on your learning of the material, rather than your completion of the activities that are assigned.

This course is designed to encourage you to work steadily at learning the material and demonstrating your new knowledge. There are no single points of failure, where you lose points that cannot be recovered. Also, you cannot cram anything one time and then forget it. The material will build and you have to demonstrate that you retained things.

- Earning a C in this class means you have a general understanding of Data Science and could participate in a basic conversation about all of the topics we cover. I expect everyone to reach this level.
- Earning a B means that you could solve simple data science problems on your own and complete parts of more complex problems as instructed by, for example, a supervisor in an internship or entry level job. This is a very accessible goal, it does not require you to get anything on the first try or to explore topics on your own. I expect most students to reach this level.
- Earning an A means that you could solve moderately complex problems independently and discus the quality of others' data science solutions. This class will be challenging, it requires you to explore topics a little deeper than we cover them in class, but unlike typical grading it does not require all of your assignments to be near perfect.

Grading this way also is more amenable to the fact that there are correct and incorrect ways to do things, but there is not always a single correct answer to a realistic data science problem. Your work will be assessed on whether or not it demonstrates your learning of the targeted skills. You will also receive feedback on how to improve.

How it works

There are 15 skills that you will be graded on in this course. While learning these skills, you will work through a progression of learning. Your grade will be based on earning 45 achievements that are organized into 15 skill groups with 3 levels for each.

These map onto letter grades roughly as follows:

- If you achieve level 1 in all of the skills, you will earn at least a C in the course.
- To earn a B, you must earn all of the level 1 and level 2 achievements.
- To earn an A, you must earn all of the achievements.

You will have at least three opportunities to earn every level 2 achievement. You will have at least two opportunities to earn every level 3 achievement. You will have three *types* of opportunities to demonstrate your current skill level: participation, assignments, and a portfolio.

Each level of achievement corresponds to a phase in your learning of the skill:

- To earn level 1 achievements, you will need to demonstrate basic awareness of the required concepts and know approximately what to do, but you may need specific instructions of which things to do or to look up examples to modify every step of the way. You can earn level 1 achievements in class, assignments, or portfolio submissions.
- To earn level 2 achievements you will need to demonstrate understanding of the concepts and the ability to apply them with instruction after earning the level 1 achievement for that skill. You can earn level 2 achievements in assignments or portfolio submissions.
- To earn level 3 achievements you will be required to consistently execute each skill and demonstrate deep understanding of the course material, after achieving level 2 in that skill. You can earn level 3 achievements only through your portfolio submissions.

For each skill these are defined in the Achievement Definition Table

Participation

While attending synchronous class sessions, there will be understanding checks and in class exercises. Completing in class exercises and correctly answering questions in class can earn level 1 achievements. In class questions will be administered through the classroom chat platform Prismia.chat; these records will be used to update your skill progression.

Office Hours

If you miss questions during class, you can make up level 1 achievements in office hours in the following two weeks. You can earn up to 2 level 1 achievements in a single visit to office hours. To earn them in office hours, you will be asked similar questions, but have the opportunity to answer verbally.

Assignments

For your learning to progress and earn level 2 achievements, you must practice with the skills outside of class time.

There will be an assignment each week. Assignments will be a chance to analyze data using the new skills and they will build so that you can get continuous practice with the skills. After your assignment is reviewed, you will get qualitative feedback on your work, and an assessment of your demonstration of the targeted skills. In an assignment, you can earn level 1 achievements in the designated skills by including an outline, you can earn level 2 by completing the analysis as prescribed.

> **Warning**
>
> If you will skip an assignment, please accept the GitHub assignment and then close the Feedback pull request with a comment. This way we can make sure that you have support you need.

Assignments are also an opportunity to get a head start on your portfolio. You can propose extensions of the analysis and get feedback on that proposal before you implement it. This way you can have more guidance on what goes in your portfolio and do the work for it continually.

Feedback on assignments is designed to be a two way *conversation* about your work to help you become a better data science programmer. Reading and acknowledging (reply or emoji reaction) your feedback is required. There is a limit of 2 assignments worth of unacknowledged feedback. If you have 2 unacknowledged assignments, your future assignments will not get feedback, they will be considered unsubmitted.

Portfolio Checks

To earn level 3 achievements, you will build a portfolio consisting of reflections, challenge problems, and longer analyses over the course of the semester. You will submit your portfolio for review 4 times. The first two will cover the skills taught up until 1 week before the submission deadline.

The third and fourth portfolio checks will cover all of the skills. The fourth will be due during finals. This means that, if you have earned all of your targeted achievements by the 3rd portfolio check, you do not need to submit the fourth one.

Portfolio prompts will be given throughout the class, some will be structured questions, others may be questions that arise in class, for which there is not time to answer.

TLDR

You *could* earn a C through in class participation alone, if you make nearly zero mistakes. To earn a B, you must complete assignments and participate in class. To earn an A you must participate, complete assignments, and build a portfolio.

Detailed mechanics

The table below shows the minimum number of skills at each level to earn each letter grade.

letter grade	Level 3	Level 2	Level 1
A	15	15	15
A-	10	15	15
B+	5	15	15
B	0	15	15
B-	0	10	15
C+	0	5	15
C	0	0	15
C-	0	0	10
D+	0	0	5
D	0	0	3

For example, if you achieve level 2 on all of the skills and level 3 on 7 skills, that will be a B+.

If you achieve level 3 on 14 of the skills, but only level 1 on one of the skills, that will be a B-, because the minimum number of level 2 achievements for a B is 15. In this scenario the total number of achievements is 14 at level 3, 14 at level 2 and 15 at level 3, because you have to earn achievements within a skill in sequence.

> **Note**
>
> In this example, you will have also achieved level 1 on all of the skills, because it is a prerequisite to level 2.

The letter grade can be computed as follows

Late work

Late assignments will not be graded. Every skill will be assessed through more than one assignment, so missing assignments occasionally not necessarily hurt your grade. If you do not submit any assignments that cover a given skill, you may earn the level 2 achievement in that skill through a portfolio check, but you will not be able to earn the level 3 achievement in that skill. If you submit work that is not complete, however, it will be assessed and receive feedback. Submitting pseudocode or code with errors and comments about what you have tried could earn a level 1 achievement. Additionally, most assignments cover multiple skills, so partially completing the assignment may earn level 2 for one, but not all. Submitting *something* even if it is not perfect is important to keeping conversation open and getting feedback and help continuously.

> **Note**
>
> You may visit office hours to discuss assignments that you did not complete on time to get feedback and check your own understanding, but they will not count toward skill demonstration.

Building your Data Science Portfolio should be an ongoing process, where you commit work to your portfolio frequently. If something comes up and you cannot finish all that you would like assessed by the deadline, open an `Extension Request` issue on your repository.

In this issue, include:

1. A new deadline proposal
2. What additional work you plan to add
3. Why the extension is important to your learning
4. Why the extension will not hinder your ability to complete the next assignment on time.

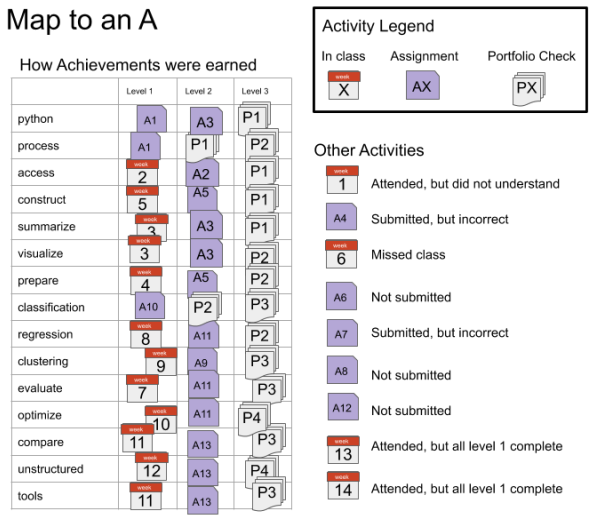
This request should be no more than 7 sentences.

Portfolio due dates will be announced well in advance. You should spend some time working on it each week, applying what you've learned so far and building on the feedback on previous assignments.

Grading Examples

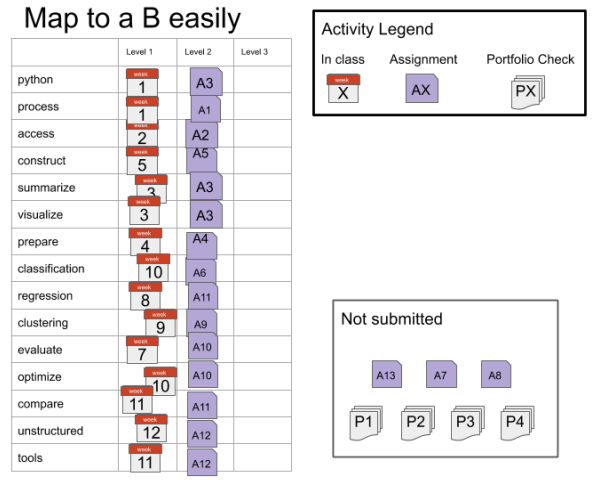
If you always attend and get everything correct, you will earn and A and you won't need to submit the 4th portfolio check or assignment 13.

Getting an A Without Perfection



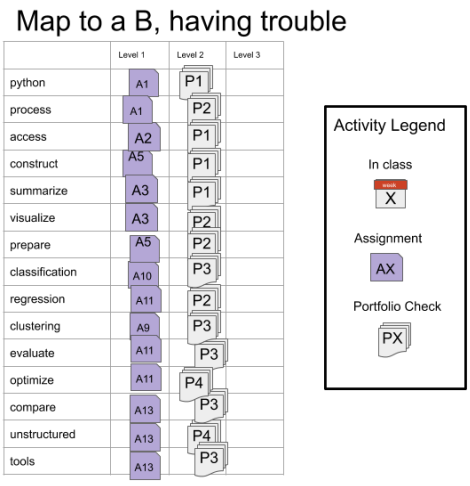
In this example the student made several mistakes, but still earned an A. This is the advantage to this grading scheme. For the `python`, `process`, and `classification` skills, the level 1 achievements were earned on assignments, not in class. For the `process` and `classification` skills, the level 2 achievements were not earned on assignments, only on portfolio checks, but they were earned on the first portfolio of those skills, so the level 3 achievements were earned on the second portfolio check for that skill. This student's fourth portfolio only demonstrated two skills: `optimize` and `unstructured`. It included only 1 analysis, a text analysis with optimizing the parameters of the model. Assignments 4 and 7 were both submitted, but didn't earn any achievements, the student got feedback though, that they were able to apply in later assignments to earn the achievements. The student missed class week 6 and chose to not submit assignment 6 and use week 7 to catch up. The student had too much work in another class and chose to skip assignment 8. The student tried assignment 12, but didn't finish it on time, so it was not graded, but the student visited office hours to understand and be sure to earn the level 2 `unstructured` achievement on assignment 13.

Getting a B with minimal work



In this example, the student earned all level 1 achievements in class and all level 2 on assignments. This student was content with getting a B and chose to not submit a portfolio.

Getting a B while having trouble



In this example, the student struggled to understand in class and on assignments. Assignments were submitted that showed some understanding, but all had some serious mistakes, so only level 1 achievements were earned from assignments. The student wanted to get a B and worked hard to get the level 2 achievements on the portfolio checks.

Grading Policies

Late Work

Late assignments will not be graded. Every skill will be assessed through more than one assignment, so missing assignments occasionally not necessarily hurt your grade. If you do not submit any assignments that cover a given skill, you may earn the level 2 achievement in that skill through a portfolio check, but you will not be able to earn the level 3 achievement in that skill. If you submit work that is not complete, however, it will be assessed and receive feedback. Submitting pseudocode or code with errors and comments about what you have tried could earn a level 1 achievement. Additionally, most assignments cover multiple skills, so partially completing the assignment may earn level 2 for one, but not all. Submitting *something* even if it is not perfect is important to keeping conversation open and getting feedback and help continuously.

> **Note**
>
> You may visit office hours to discuss assignments that you did not complete on time to get feedback and check your own understanding, but they will not count toward skill demonstration.

Building your Data Science Portfolio should be an ongoing process, where you commit work to your portfolio frequently. If something comes up and you cannot finish all that you would like assessed by the deadline, open an `Extension Request` issue on your repository.

In this issue, include:

1. A new deadline proposal
2. What additional work you plan to add
3. Why the extension is important to your learning
4. Why the extension will not hinder your ability to complete the next assignment on time.

This request should be no more than 7 sentences.

Portfolio due dates will be announced well in advance and prompts for it will be released weekly. You should spend some time working on it each week, applying what you've learned so far, from the feedback on previous assignments.

Regrading

Re-request a review on your Feedback Pull request.

For general questions, post on the conversation tab of your Feedback PR with your request.

For specific questions, reply to a specifc comment.

If you think we missed *where* you did something, add a comment on that line (on the code tab of the PR, click the plus (+) next to the line) and then post on the conversation tab with an overview of what you're requestion and tag @brownsarahm

Collaboration

You may talk to other students about the general approach or ask clarifying questions about instructions by posting to the GitHub discussions for our course.

You may only view one another's code, when explicitly instructed to share for peer review, and only *via GitHub* by adding a classmate as a collaborator. If you do not have permission to share your repository or an assignment is not created as a team assignment, then you may not collaborate on that assignment at the code level.

Support

Academic Enhancement Center

Academic Enhancement Center (for undergraduate courses): Located in Roosevelt Hall, the AEC offers free face-to-face and web-based services to undergraduate students seeking academic support. Peer tutoring is available for STEM-related courses by appointment online and in-person. The Writing Center offers peer tutoring focused on supporting undergraduate writers at any stage of a writing assignment. The UCS160 course and academic skills consultations offer students strategies and activities aimed at improving their studying and test-taking skills. Complete details about each of these programs, up-to-date schedules, contact information and self-service study resources are all available on the AEC website.

- **STEM Tutoring** helps students navigate 100 and 200 level math, chemistry, physics, biology, and other select STEM courses. The STEM Tutoring program offers free online and limited in-person peer-tutoring this fall. Undergraduates in introductory STEM courses have a variety of small group times to choose from and can select occasional or weekly appointments. Appointments and locations will be visible in the TutorTrac system on September 14th, 2020. The TutorTrac application is available through URI Microsoft 365 single sign-on and by visiting aec.uri.edu. More detailed information and instructions can be found on the AEC tutoring page.
- **Academic Skills Development** resources helps students plan work, manage time, and study more effectively. In Fall 2020, all Academic Skills and Strategies programming are offered both online and in-person. UCS160: Success in Higher Education is a one-credit course on developing a more effective approach to studying. Academic Consultations are 30-minute, 1 to 1 appointments that students can schedule on Starfish with Dr. David Hayes to address individual academic issues. Study Your Way to Success is a self-guided web portal connecting students to tips and strategies on studying and time management related topics. For more information on these programs, visit the Academic Skills Page or contact Dr. Hayes directly at davidhayes@uri.edu.
- The **Undergraduate Writing Center** provides free writing support to students in any class, at any stage of the writing process: from understanding an assignment and brainstorming ideas, to developing, organizing, and revising a draft. Fall 2020 services are offered through two online options: 1) real-time synchronous appointments with a peer consultant (25- and 50-minute slots, available Sunday - Friday), and 2) written asynchronous consultations with a 24-hour turn-around response time (available Monday - Friday). Synchronous appointments are video-based, with audio, chat, document-sharing, and live captioning capabilities, to meet a range of accessibility needs. View the synchronous and asynchronous schedules and book online, visit uri.mywconline.com.

General URI Policies

COVID/Viral Illness Precautions Statement

The University is committed to delivering its educational mission while protecting the health and safety of our community. As members of the URI community, students are required to comply with standards of conduct and take precautions to keep themselves and others safe.

> **Important**
>
> Masks are required in all classrooms, laboratories, and spaces where direct academic instruction and research are taking place, unless the instructor or staff member expressly waives that requirement.

> **Warning**
>
> not waived

We strongly recommend surgical or higher grade masks where face coverings are required. Masks should be properly worn, well-fitting, and high quality.

Students who do not comply with the classroom/lab masking requirement will be asked to leave class and will be reported through the Student Conduct process.

Students who are experiencing symptoms of viral illness should NOT go to class/work. Those who test positive for COVID-19 should follow the isolation guidelines from the Rhode Island Department of Health and CDC.

Anti-Bias Statement:

We respect the rights and dignity of each individual and group. We reject prejudice and intolerance, and we work to understand differences. We believe that equity and inclusion are critical components for campus community members to thrive. If you are a target or a witness of a bias incident, you are encouraged to submit a report to the URI Bias Response Team at www.uri.edu/brt. There you will also find people and resources to help.

Disability Services for Students Statement:

Your access in this course is important. Please send me your Disability Services for Students (DSS) accommodation letter early in the semester so that we have adequate time to discuss and arrange your approved academic accommodations. If you have not yet established services through DSS, please contact them to engage in a confidential conversation about the process for requesting reasonable accommodations in the classroom. DSS can be reached by calling: 401-874-2098, visiting: web.uri.edu/disability, or emailing: dss@etal.uri.edu. We are available to meet with students enrolled in Kingston as well as Providence courses.

Academic Honesty

Students are expected to be honest in all academic work. A student's name on any written work, quiz or exam shall be regarded as assurance that the work is the result of the student's own independent thought and study. Work should be stated in the student's own words, properly attributed to its source. Students have an obligation to know how to quote, paraphrase, summarize, cite and reference the work of others with integrity. The following are examples of academic dishonesty.

- Using material, directly or paraphrasing, from published sources (print or electronic) without appropriate citation
- Claiming disproportionate credit for work not done independently
- Unauthorized possession or access to exams
- Unauthorized communication during exams
- Unauthorized use of another's work or preparing work for another student
- Taking an exam for another student
- Altering or attempting to alter grades
- The use of notes or electronic devices to gain an unauthorized advantage during exams
- Fabricating or falsifying facts, data or references
- Facilitating or aiding another's academic dishonesty
- Submitting the same paper for more than one course without prior approval from the instructors

URI COVID-19 Statement

The University is committed to delivering its educational mission while protecting the health and safety of our community. While the university has worked to create a healthy learning environment for all, it is up to all of us to ensure our campus stays that way.

As members of the URI community, students are required to comply with standards of conduct and take precautions to keep themselves and others safe. Visit web.uri.edu/coronavirus/ for the latest information about the URI COVID-19 response.

- Universal indoor masking is required by all community members, on all campuses, regardless of vaccination status. If the universal mask mandate is discontinued during the semester, students who have an approved exemption and are not fully vaccinated will need to continue to wear a mask indoors and maintain physical distance.
- Students who are experiencing symptoms of illness should not come to class. Please stay in your home/room and notify URI Health Services via phone at 401-874-2246.
- If you are already on campus and start to feel ill, go home/back to your room and self-isolate. Notify URI Health Services via phone immediately at 401-874-2246.

If you are unable to attend class, please notify me at brownsarahm@uri.edu. We will work together to ensure that course instruction and work is completed for the semester.

Help Hours and Course Communications

We have several different ways to communicate in this course. This section summarizes them

Help Hours

```
/tmp/ipykernel_2545/2146052215.py:1: FutureWarning: this method is deprecated in
favour of `Styler.hide(axis="index")`
  help_df.style.hide_index()
```

Day	Time	Location	Host
Tuesday	11am-12pm	139 Tyler Hall	Aiden
Wednesday	11am-12pm	139 Tyler Hall	Aiden
Wednesday	7pm-8:30pm	Zoom	Dr. Brown
Thursday	11am-12pm	139 Tyler Hall	Aiden
Friday	11am-12pm	139 Tyler Hall	Aiden
Friday	3:30-4:30	134 Tyler Hall	Dr. Brown

To reach out, By usage

```
/tmp/ipykernel_2545/3027175348.py:2: FutureWarning: this method is deprecated in
favour of `Styler.hide(axis="index")`
  display(HTML(df.style.hide_index()._repr_html_()))
```

usage	platform	area	note
in class	prismia	chat	outside of class time this is not monitored closely
any time	prismia	download transcript	use after class to get preliminary notes eg if you miss a class
private questions to your assignment	github	issue on assignment repo	eg bugs in your code"
for general questions that can help others	github	issue on course website	eg what the instructions of an assignment mean or questions about the syllabus
to share resources or ask general questions in a semi-private forum	github	discussion on community repo	include links in your portfolio
matters that don't fit into another category	e-mail	to brownsarahm@uri.edu	remember to include `[CSC310]` or `[DSP310]` (note `verbatim` no space)

Note

e-mail is last because it's not collaborative; other platforms allow us (Proessor + TA) to collaborate on who responds to things more easily.

By Platform

Use e-mail for

```
/tmp/ipykernel_2545/2135006347.py:3: FutureWarning: this method is deprecated in
favour of `Styler.hide(axis="index")`
  display(HTML(data.drop(columns='platform').style.hide_index()._repr_html_()))
```

usage	area	note
matters that don't fit into another category	to brownsarahm@uri.edu	remember to include `[CSC310]` or `[DSP310]` (note `verbatim` no space)

Use github for

```
/tmp/ipykernel_2545/2135006347.py:3: FutureWarning: this method is deprecated in
favour of `Styler.hide(axis="index")`
  display(HTML(data.drop(columns='platform').style.hide_index()._repr_html_()))
```

usage	area	note
private questions to your assignment	issue on assignment repo	eg bugs in your code"
for general questions that can help others	issue on course website	eg what the instructions of an assignment mean or questions about the syllabus
to share resources or ask general questions in a semi-private forum	discussion on community repo	include links in your portfolio

Use prismia for

```
/tmp/ipykernel_2545/2135006347.py:3: FutureWarning: this method is deprecated in
favour of `Styler.hide(axis="index")`
  display(HTML(data.drop(columns='platform').style.hide_index()._repr_html_()))
```

usage	area	note
in class	chat	outside of class time this is not monitored closely
any time	download transcript	use after class to get preliminary notes eg if you miss a class

Tips

For assignment help

- **send in advance, leave time for a response** I check e-mail/github a small number of times per day, during work hours, almost exclusively. You might see me post to this site, post to BrightSpace, or comment on your assignments outside of my normal working hours, but I will not reliably see emails that arrive during those hours. This means that it is important to start assignments early.

Using issues

- use issues for content directly related to assignments. If you push your code to the repository and then open an issue, I can see your code and your question at the same time and download it to run it if I need to debug it
- use issues for questions about this syllabus or class notes. At the top right there's a GitHub logo that allows you to open a issue (for a question) or suggest an edit (eg if you think there's a typo or you find an additional helpful resource related to something)

For E-email

- use e-mail for general inquiries or notifications
- Please include `[CSC310]` or `[DSP310]` in the subject line of your email along with the topic of your message. This is important, because your messages are important, but I also get a lot of e-mail. Consider these a cheat code to my inbox: I have setup a filter that will flag your e-mail if you use one of those in the subject to ensure that I see it.

Note

Whether you use CSC or DSP does not matter.

1. Welcome to Programming to Data Science

1.1. Prismia Chat

We will use these to monitor your participation in class and to gather information. Features:

- instructor only
- reply to you directly
- share responses for all

1.2. How this class will work

Participatory Live Coding

What is a topic you want to use data to learn about?

Debugging is both technical and a soft skill

1.3. Programming for Data Science vs other Programming

The audience is different, so the form is different.

In Data Science our product is more often a report than a program.

> **Note**
>
> Also, in data science we are *using code* to interact with data, instead of having a plan in advance

So programming for data science is more like *writing* it has a narrative flow and is made to be seen more than some other programming thaat you may have done.

> **Warning**
>
> Sometimes there will be points in the notes that were not made in class due to time or in response questions that came at the end of class.

1.4. Get Organized!

In this class you will have many separate folders that your work is in. The separate folders are *required* because we will use GitHub for submission.

I recommend you make a folder for this class and make all of your other folders inside that.

Create a separate notes folder in there too. We will be writing code each class, that you should keep your own notes.

> **Important**
>
> If you made your notebook in a location other than where you want it to be, you can move it like any other file using Finder on mac or File Explorer on Windows.

1.5. Jupyter Notebooks

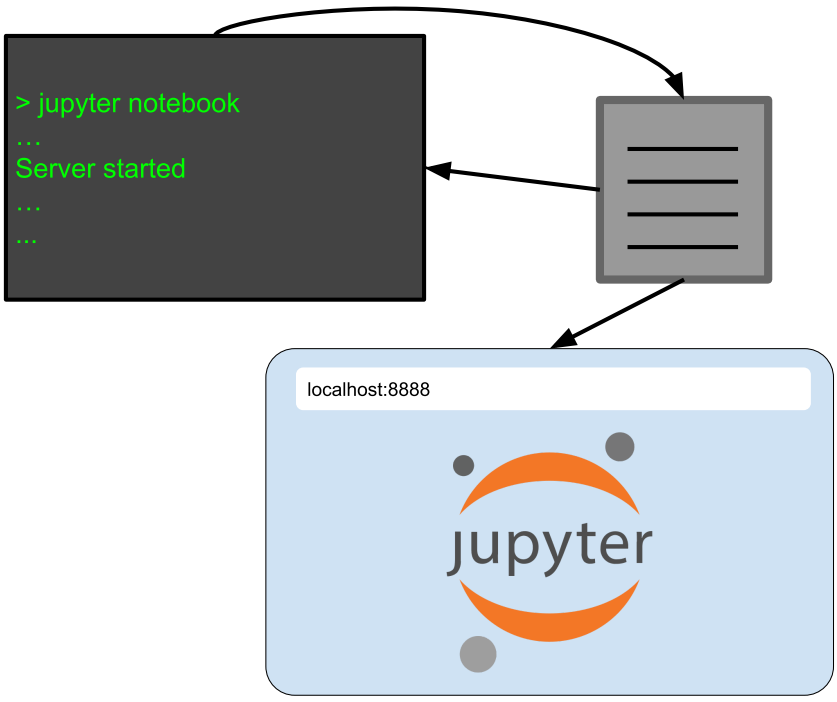
Launch a `jupyter notebook` server:

- on Windows, use anaconda terminal
- on Mac/Linux, use terminal

```
cd path/to/where/you/save/notes
jupyter notebook
```

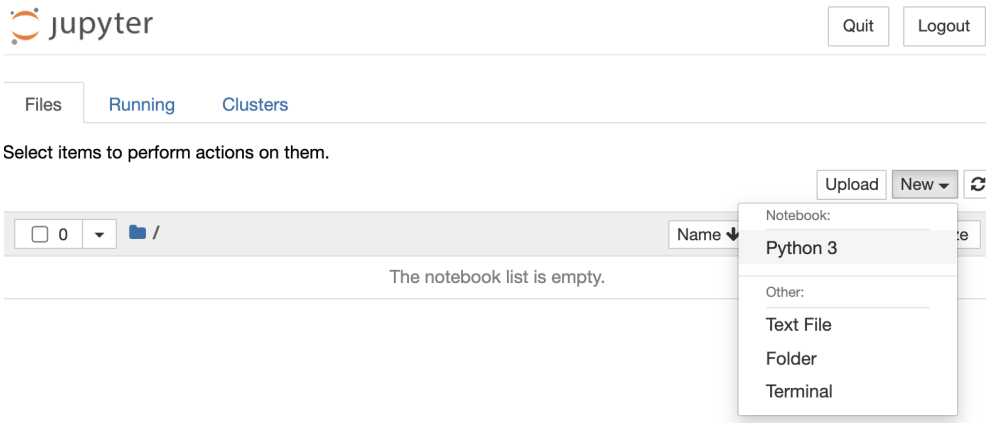
1.5.1. What just happened?

- launched a local web server
- opened a new browser tab pointed to it

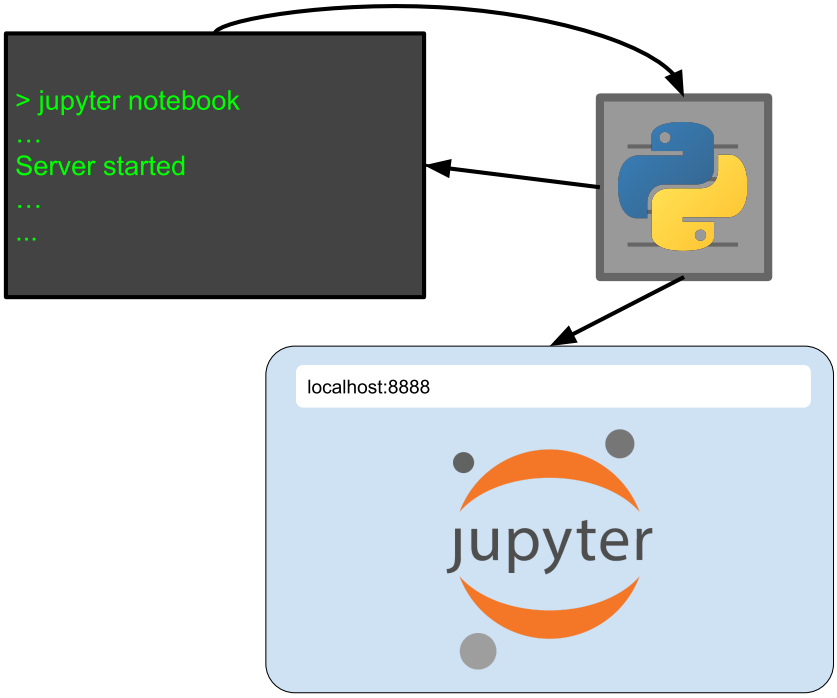


1.5.2. Start a Notebook

Go to the new menu in the top right and choose Python 3

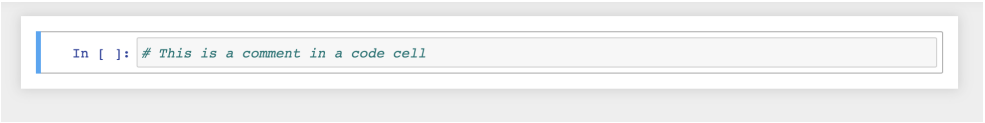


Now, it starts a python kernel on the webserver

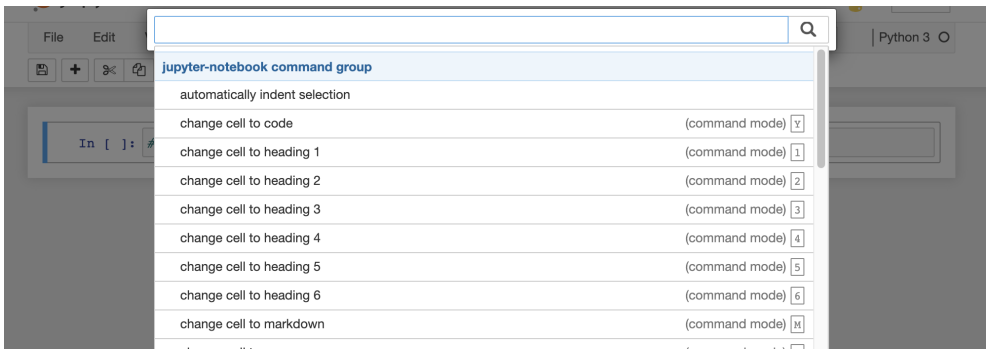


1.5.3. A jupyter notebook tour

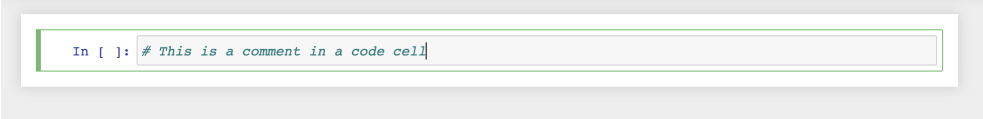
A Jupyter notebook has two modes. When you first open, it is in command mode. The border is blue in command mode.



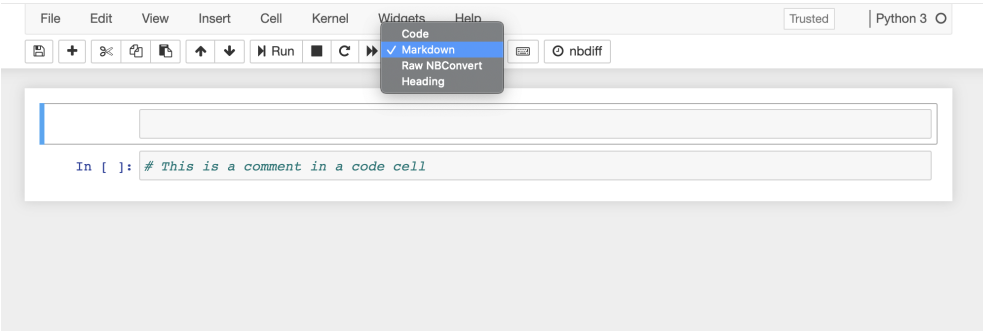
When you press a key in command mode it works like a shortcut. For example `p` shows the command search menu.



If you press `enter` (or `return`) or click on the highlighted cell, which is the boxes we can type in, it changes to edit mode. The border is green in edit mode



There are two type of cells that we will used: code and markdown. You can change that in command mode with `y` for code and `m` for markdown or on the cell type menu at the top of the notebook.



++

This is a markdown cell

- we can make
- itemized lists of
- bullet points

1. and we can make nubmered
2. lists, and not have to worry
3. about renumbering them
4. if we add a step in the middle later

1.5.4. Notebook Reminders

Blue border is command mode, green border is edit mode

use Escape to get to command mode

Common command mode actions:

- m: switch cell to markdown
- y: switch cell to code
- a: add a cell above
- b: add a cell below
- c: copy cell
- v: paste the cell
- 0 + 0: restart kernel
- p: command menu

use enter/return to get to edit mode

In code cells, we can use a python interpreter, for example as a calculator.

```
4+6
```

```
10
```

It prints out the last line of code that it ran, even though it executes all of them

```
name = 'sarah'
4+5
name *3
```

```
'sarahsarahsarah'
```

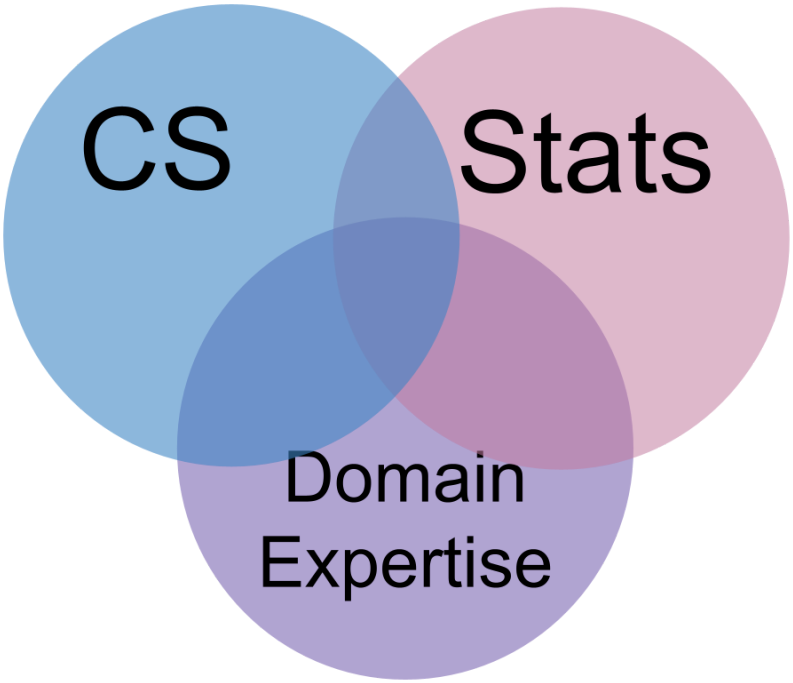
1.6. Getting Help in Jupyter

When your cursor is inside the `()` of a function if you hodl the shift key and press tab it will open a popup with information.

Python has a `print` function and we can use the help in jupyter to learn about how to use it in different ways.

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```



1.7. Course Administration

We will use GitHub for all course administration.

- join to discuss
- go to discussions and introduce yourself
- course website
- Main page with links

1.8. Prepare for the next class

- Read carefully the syllabus section of the course website
- skim the rest of the course website
- Bring questions about how the class will work to class on Friday.
- Review Git & GitHub Fundamentals
- Bring git/github questions on Friday.
- Begin reading chapter 1 of think like a data scientist (finish in time for it to help you with the assignment due Sunday night)

On Friday we will start with a review of the syllabus. You will answer an ungraded quiz to confirm that you understand and I'll answer all of your questions. Then we will do a little bit with Git/GitHub and start your first assignment in class.

Think like a data scientist is written for practitioners; not as a text book for a class. It does not have a lot of prerequisite background, but the sections of it that I assign will help you build a better mental picture of what doing Data Science about. In chapter 1, focus most on sections 1.1, 1.3, and 1.7.

Only the first assignment will be due this fast, it's a short review and setup assignment. It's due quickly so that we know that you have everything set up and the prerequisite material before we start new material next week.

1.9. Questions after class

1.9.1. Grading

1.9.1.1. How do the portfolios work? what are the 1/2/3 achievement levels? How is gradint structured?

Read the syllabus carefully and we will discuss on Friday

1.9.1.2. Will there be any group work?

Not in the regular sense of collaborative and shared grade. There will be optional collaboration opportunities and in class discussion/troubleshooting together.

1.9.2. Uses for what we cover

1.9.2.1. How in depth does Data Science go, and what can it be used for in the industry?

The basic ideas here can be used for any tabular data in industry exactly as we will cover them. At the end of the semester, we will see in less detail how to work with text and images, with a focus on translating what you learned on tabular data (because its low dimensional and easier to see/faster to process) to more complex data.

We'll talk more about this on Friday and every time we use a new dataset in class.

1.9.2.2. What is the difference between data analytics and data science? Are analysts and scientists jobs different?

This is a hard question. It varies company to company. It is, however, a topic a lot of Data Science Bloggers write about. If you find some you like, share them on the discussion board or submit a PR.

1.9.3. Logistics

1.9.3.1. What is the main site

this is it

1.9.3.2. do these prismia chats stay up after class or go away?

They persist. You can scroll back or get a transcript by clicking the `>` in the top left, then the 3 bar menu icon and then “Get Transcript From Class”

1.9.4. Tools

1.9.4.1. Can we open jupyter notebook server without the terminal?

Yes, you can, but I do not use it that way and they change how that works from time to time, so I can only troubleshoot with you via the terminal.

1.9.4.2. Which IDE to use? Can we use VSCode?

Your will be required to submit jupyter notebooks that are compatible with some other jupyter related tools I use to process them for grading.

1.9.4.3. Will we all have a single server to have various notebooks for assignments, or will we be required to make different servers for other assignments?

The jupyter notebook web server is something that you will start and stop many times, each working session you'll stop it. You can run multiple in parallel or use one and open multiple notebooks.

1.9.4.4. Will we do any web scraping in this class?

Yes

1.9.4.5. Will we be able to choose our own data sets when doing assignments?

Yes, with some requirements, on most assignments.

2. Reading Docstrings, Object Inspection, and Loading Data

2.1. Programming is a Practice

Python has a `print` function and we can use the help in jupyter to learn about how to use it in different ways.

Given this code excerpt, how could you print out “Sarah_Brown”?

```
first = 'Sarah'
last = 'Brown'
```

We can print the docstring out, as a whole instead of using the shfit + tab to view it.

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

The first line says that it can take multiple values, because it says `value, ..., sep`

It also has a keyword argument (must be used like `argument=value` and has a default) described as `sep=' '`. This means that by default it adds a space as above.

```
print(first,last)
```

```
Sarah Brown
```

```
print(first,last,sep='_')
```

```
Sarah_Brown
```

```
type(first)
```

```
str
```

```
def compute_grade(num_level1,num_level2,num_level3):
    '''
    Computes a grade for CSC/DSP310 from numbers of achievements at each level

    Parameters:
    -----------
    num_level1 : int
      number of level 1 achievements earned
    num_level2 : int
      number of level 2 achievements earned
    num_level3 : int
      number of level 3 achievements earned

    Returns:
    --------
    letter_grade : string
      letter grade with modifier (+/-)
    '''
    if num_level1 == 15:
        if num_level2 == 15:
            if num_level3 == 15:
                grade = 'A'
            elif num_level3 >= 10:
                grade = 'A-'
            elif num_level3 >=5:
                grade = 'B+'
            else:
                grade = 'B'
        elif num_level2 >=10:
            grade = 'B-'
        elif num_level2 >=5:
            grade = 'C+'
        else:
            grade = 'C'
    elif num_level1 >= 10:
        grade = 'C-'
    elif num_level1 >= 5:
        grade = 'D+'
    elif num_level1 >=3:
        grade = 'D'
    else:
        grade = 'F'


    return grade
```

```
type(compute_grade)
```

```
function
```

```
help(compute_grade())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [8], line 1
----> 1 help(compute_grade())

TypeError: compute_grade() missing 3 required positional arguments: 'num_level1',
'num_level2', and 'num_level3'
```

2.1.1. Why inspection in code?

Some IDEs give you GUI based tools to inspect objects. We are going to do it programmatically inline with our analyses for two reasons.

(minor, logistical) it helps make for good notes (most importantly) it helps build habits of data science

2.1.2. Investigating how doc strings work

We can see how the docstring impacts help and how exactly it has to be formatted to become a docstring

```
def ex_1(a):
    print(a)
```

```
help(ex_1())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [10], line 1
----> 1 help(ex_1())

TypeError: ex_1() missing 1 required positional argument: 'a'
```

```
def ex_2(a):
    #nis htis a docstring?
    print(a)
```

```
help(ex_2())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [12], line 1
----> 1 help(ex_2())

TypeError: ex_2() missing 1 required positional argument: 'a'
```

```
def ex_3(a):
    ''' this is a docstring'''
    #nis htis a docstring?
    print(a)
```

```
help(ex_3())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [14], line 1
----> 1 help(ex_3())

TypeError: ex_3() missing 1 required positional argument: 'a'
```

```
def ex_4(a):
    """this is a docstring"""
    #nis htis a docstring?
    print(a)
```

```
help(ex_4())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [16], line 1
----> 1 help(ex_4())

TypeError: ex_4() missing 1 required positional argument: 'a'
```

Tip

In python, PEP 257 says how to write a docstring, but it is very broad.

In Data Science, numpydoc style docstrings are popular.

- Pandas follows numpydoc
- [Numpy uses it]
- Scipy follows numpydoc

2.2. Coffee Data

Structured data is easier to work with than other data.

We're going to focus on tabular data for now. At the end of the course, we'll examine images, which are structured, but more complex and text, which is much less structured.

We're going to use a dataset about coffee quality today.

How was this dataset collected?

- reviews added to DB
- then scraped

Where did it come from?

- coffee Quality Institute's trained reviewers.

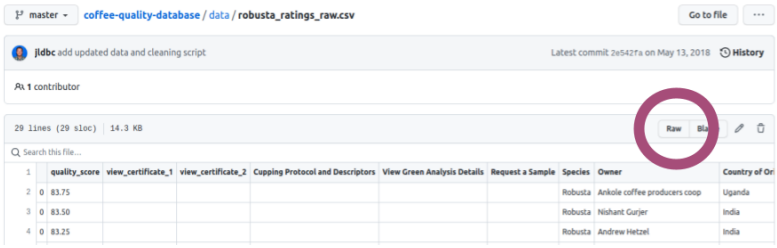
what format is it provided in?

- csv (Comma Separated Values)

what other information is in this repository?

- the code to scrape and clean the data
- the data before cleaning

Get raw url for the dataset click on the raw button on the csv page, then copy the url.



We'll save that url as a variable to work with it.

```
coffee_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/arabica_data_cleaned.csv'
```

2.3. Loading in Data

We will use a library called Pandas

```
import pandas as pd
```

- the `import` keyword is used for loading packages
- `pandas` is the name of the package that is installed
- `as` keyword allows us to assign an alias (nickname)
- `pd` is the typical alias for pandas

We can read data in using the read csv file

```
pd.read_csv(coffee_data_url)
```

Important

It's important to always know where data came from and how it was collected.

This helps you know what is is useful for and what its limitations are.

An important research article on documenting datasets for machine learning is called Datasheets for Datasets these researchers also did a follow up study to better understand how practitioner use datasheets and decide how to use data.

Note

If this type of research is interesting to you, let me know!

Further Reading

the pandas package overview can be a good read if reading more about things helps you understand

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Defects	Ex
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	0	A
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	1	A
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN	NaN	1600 - 1800 m	...	NaN	0	M
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN	yidnekachew debessa coffee plantation	1800-2200	...	Green	2	25
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	2	A
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1306	1307	Arabica	juan carlos garcia lopez	Mexico	el centenario	NaN	la esperanza, municipio juchique de ferrer, ve...	1104328663	terra mia	900	...	None	20	Se 17
1307	1308	Arabica	myriam kaplan-pasternak	Haiti	200 farms	NaN	coeb koperativ ekselsyo basen (350 members)	NaN	haiti coffee	~350m	...	Blue-Green	16	M
1308	1309	Arabica	exportadora atlantic, s.a.	Nicaragua	finca las marías	017-053-0211/ 017-053-0212	beneficio atlantic condega	017-053-0211/ 017-053-0212	exportadora atlantic s.a	1100	...	Green	5	J
1309	1310	Arabica	juan luis alvarado romero	Guatemala	finca el limon	NaN	beneficio serben	11/853/165	unicafe	4650	...	Green	4	M
1310	1312	Arabica	bismarck castro	Honduras	los hicaques	103	cigrah s.a de c.v.	13-111-053	cigrah s.a de c.v	1400	...	Green	2	Aj

1311 rows × 44 columns

This read in the data and prints it out because it is the last line on the cell. If we do something else after, it will read it in, but not print it out

```
pd.read_csv(coffee_data_url)
print(first)
```

```
Sarah
```

In order to use it, we save the output to a variable.

```
coffee_data = pd.read_csv(coffee_data_url)
```

Then we can check the type.

```
type(coffee_data)
```

```
pandas.core.frame.DataFrame
```

This is a new type that is provided by the pandas library. Notice this uses the full libary name, not the alias, because this comes from the code for the library itself, not our current code where pandas as a nickname.

```
coffee_df = pd.read_csv(coffee_data_url)
```

```
coffee_df
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Defects	Ex
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	0	A
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	1	A
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN	NaN	1600 - 1800 m	...	NaN	0	M
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN	yidnekachew debessa coffee plantation	1800-2200	...	Green	2	25
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	2	A
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1306	1307	Arabica	juan carlos garcia lopez	Mexico	el centenario	NaN	la esperanza, municipio juchique de ferrer, ve...	1104328663	terra mia	900	...	None	20	Se 17
1307	1308	Arabica	myriam kaplan-pasternak	Haiti	200 farms	NaN	coeb koperativ ekselsyo basen (350 members)	NaN	haiti coffee	~350m	...	Blue-Green	16	M
1308	1309	Arabica	exportadora atlantic, s.a.	Nicaragua	finca las marías	017-053-0211/ 017-053-0212	beneficio atlantic condega	017-053-0211/ 017-053-0212	exportadora atlantic s.a	1100	...	Green	5	J
1309	1310	Arabica	juan luis alvarado romero	Guatemala	finca el limon	NaN	beneficio serben	11/853/165	unicafe	4650	...	Green	4	M
1310	1312	Arabica	bismarck castro	Honduras	los hicaques	103	cigrah s.a de c.v.	13-111-053	cigrah s.a de c.v	1400	...	Green	2	A

1311 rows × 44 columns

If you're curious about something, try it out, see what happens. We're going to use a lot of code inspection tools during class. These are helpful both for understanding what's going on, but the advantage to knowing how to get this information programmatically even though a different IDE would give you inspection tools is that it helps you treat your code as data.

2.4. Good Code is always relative

> **Important**
>
> I added this section as notes, that was not in class today. I said similar things last week, but this includes more references and context.

In programming for data science, we are often trying to tell a story.

> **Try it yourself**
>
> How might this goal change your code for this class relative to other code you have written or could imagine writing?

Python is a fully open source project and as such is governed by community standards and conventions.

> **Try it yourself**
>
> Find PEP8 (note that following it is part of earning python achievements)

The documentation for the full language is online too.

Guido van Rossum was the first main developer and wrote essays about python too.

it's pretty popular

2.5. Questions After Class

2.5.1. About the Course

2.5.1.1. Will we further go over how to achieve level 3 achievements with more specificity?

Right now, you are still only able to earn level 1s and then with assignment 2 you can start earning level 2s. After that, it will make morse sense to be able to talk about portfolios.

2.5.1.2. How do portfolio checks work?

At a logistical level, you add files to your portfolio repository by a specific check date and then I grade them.

2.5.1.3. how much stats will we do in this class?

Only a little bit. We wil do some modeling of data and compute basic statistics, but we will not cover the underlying concepts of statistics, much. However if you know some statistics, you will be able to extend what we cover to use them.

2.5.1.4. Is there a rate at which we need to complete skill checks if we fall behind?

Follow the table on the Achievements page for which assignments and portfolio checks have which achievements eligible. Some assignments you can earn only 1-2 achievements others you can earn 4-5 level 2s. It is not recommended that you skip early chances because there are future chances, though. However, sometimes if you have an achievement already you can skip a section of an assignment.

2.5.2. About Jupyter

2.5.2.1. Can you interact with those data tables that we put on jupyter today in real time?

Yes, we can manipulate the data, but we read a copy in. We are not manipulating the version that was on GitHub.

2.5.2.2. Will we eventually learn how to filter data in order to separate different names of data, or perform mathematical operations on the datasets?

Yes, all!

2.5.2.3. Can you show how to launch a book another day after we saved it and come back to it?

When you launch your server on the “home” tab you can click on the file name of a previously saved notebook to work on it again.

2.5.2.4. can you just put anything in the docstring?

Yes, technically, but you should follow good code style guidelines.

2.5.2.5. What would happen if we just call ‘pd.read_csv(coffe_data_url)’ instead of storing it in a variable and then call the variable?

It would print it out, but then you don’t have a variable, so you would have to read it in again to be able to manipulate it.

2.5.2.6. What is considered scraped data?

Data that is pulled from websites automatically. We will do some web scraping starting this week.

2.5.3. Questions we will answer in the rest of this week

- How to view a more detailed look of the data instead of it only showing the first and last few columns
- Could we retrieve data in all formats with one function?
- Can we access data like a 2d array?
- What is the function to check the unique values in a column?

3. Data Frames and other iterables

Today, we’re going to explore DataFrames in greater detail. We’ll continue using that same coffee dataset.

```
import pandas as pd
```

```
coffee_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_data_cleaned.csv'
```

```
coffee_df =pd.read_csv(coffee_data_url)
```

Important

A reason to use Jupyter is that it formats the output to be more readable. Compare the view of the DataFrame with jupyter and without.

Jupyter uses the object’s `to_html` method if it exists, where the `print` function casts the object to a string.

```
coffee_df
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Defects
0	1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	1488	...	Green	2
1	2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi royale	3170	...	NaN	2
2	3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuraman estate	1000m	...	Green	0
3	4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof ltd	1212	...	Green	7
4	5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	0	katuka development trust ltd	1200-1300	...	Green	3
5	6	Robusta	andrew hetzel	India	NaN	NaN	(self)	NaN	cafemakers, llc	3000'	...	Green	0
6	7	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	...	Green	0
7	8	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethuraman estate	14/1148/2017/18	kaapi royale	3140	...	Bluish-Green	0
8	9	Robusta	nishant gurjer	India	sethuraman estate	RKR	sethuraman estate	14/1148/2016/17	kaapi royale	1000	...	Green	0
9	10	Robusta	ugacof	Uganda	ishaka	NaN	nsubuga umar	0	ugacof ltd	900-1300	...	Green	6
10	11	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof ltd	1095	...	Green	1
11	12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuraman estate	14/1148/2016/12	kaapi royale	1000	...	Green	0
12	13	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	...	Green	1
13	14	Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN	NaN	0	kasozi coffee farmers association	1367	...	Green	7
14	15	Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	ankole coffee producers coop union ltd	0	ankole coffee producers coop	1488	...	Green	2
15	16	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuraman estate	1000m	...	Green	0
16	17	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	750m	...	Blue-Green	0
17	18	Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	kawacom	0	kawacom uganda ltd	1600	...	Green	1
18	19	Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nitubaasa	0	nitubaasa ltd	1745	...	Green	2
19	20	Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	mannya coffee project	0	mannya coffee project	1200	...	Green	1
20	21	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	...	Bluish-Green	1
21	22	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	750m	...	Green	0
22	23	Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	3000'	...	Green	0
23	24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	NaN	robustasa	NaN	...	Blue-Green	1
24	25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	NaN	robustasa	40	...	Blue-Green	0
25	26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN	global opportunity fund	795 meters	...	NaN	6
26	27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	cafe politico	NaN	...	Green	1
27	28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN	cafe politico	NaN	...	None	9

28 rows × 44 columns

```
print(coffee_df)
```

	Unnamed: 0	Species	Owner	Country.of.Origin	\
0	1	Robusta	ankole coffee producers coop	Uganda	
1	2	Robusta	nishant gurjer	India	
2	3	Robusta	andrew hetzel	India	
3	4	Robusta	ugacof	Uganda	
4	5	Robusta	katuka development trust ltd	Uganda	
5	6	Robusta	andrew hetzel	India	
6	7	Robusta	andrew hetzel	India	
7	8	Robusta	nishant gurjer	India	
8	9	Robusta	nishant gurjer	India	
9	10	Robusta	ugacof	Uganda	
10	11	Robusta	ugacof	Uganda	
11	12	Robusta	nishant gurjer	India	
12	13	Robusta	andrew hetzel	India	
13	14	Robusta	kasozi coffee farmers association	Uganda	

```
14          15  Robusta       ankole coffee producers coop            Uganda
15          16  Robusta                     andrew hetzel             India
16          17  Robusta                     andrew hetzel             India
17          18  Robusta                kawacom uganda ltd            Uganda
18          19  Robusta                    nitubaasa ltd            Uganda
19          20  Robusta              mannya coffee project            Uganda
20          21  Robusta                     andrew hetzel             India
21          22  Robusta                     andrew hetzel             India
22          23  Robusta                     andrew hetzel     United States
23          24  Robusta                       luis robles           Ecuador
24          25  Robusta                       luis robles           Ecuador
25          26  Robusta                       james moore     United States
26          27  Robusta                     cafe politico             India
27          28  Robusta                     cafe politico           Vietnam

                            Farm.Name Lot.Number  \
0       kyangundu cooperative society        NaN
1      sethuraman estate kaapi royale         25
2                 sethuraman estate        NaN
3                ugacof project area        NaN
4   katikamu capca farmers association        NaN
5                                NaN        NaN
6                 sethuraman estates        NaN
7      sethuraman estate kaapi royale          7
8                 sethuraman estate        RKR
9                           ishaka        NaN
10               ugacof project area        NaN
11     sethuraman estate kaapi royale      RC AB
12                sethuraman estates        NaN
13             kasozi coffee farmers        NaN
14            kyangundu coop society        NaN
15                 sethuraman estate        NaN
16                sethuraman estates        NaN
17                          bushenyi        NaN
18   kigezi coffee farmers association        NaN
19              mannya coffee project        NaN
20                sethuraman estates        NaN
21                sethuraman estates        NaN
22                sethuraman estates        NaN
23                        robustasa   Lavado 1
24                        robustasa   Lavado 3
25                   fazenda cazengo        NaN
26                               NaN        NaN
27                               NaN        NaN

                                    Mill         ICO.Number  \
0                 ankole coffee producers                  0
1                       sethuraman estate    14/1148/2017/21
2                                     NaN               0000
3                                  ugacof                  0
4                katuka development trust                  0
5                                 (self)                NaN
6                                     NaN                NaN
7                       sethuraman estate    14/1148/2017/18
8                       sethuraman estate    14/1148/2016/17
9                            nsubuga umar                  0
10                                 ugacof                  0
11                      sethuraman estate    14/1148/2016/12
12                                    NaN                NaN
13                                    NaN                  0
14  ankole coffee producers coop union ltd                  0
15                                    NaN               0000
16                     sethuraman estates                NaN
17                                kawacom                  0
18                               nitubaasa                  0
19                  mannya coffee project                  0
20                                    NaN                NaN
21                     sethuraman estates                NaN
22                     sethuraman estates                NaN
23                            our own lab                NaN
24                        own laboratory                 NaN
25                          cafe cazengo                 NaN
26                                    NaN  14-1118-2014-0087
27                                    NaN                NaN

                            Company    Altitude  ...         Color  \
0       ankole coffee producers coop        1488  ...         Green
1                  kaapi royale             3170  ...           NaN
2                sethuraman estate       1000m  ...         Green
3                  ugacof ltd            1212  ...         Green
4       katuka development trust ltd   1200-1300  ...         Green
5                  cafemakers, llc        3000'  ...         Green
6                      cafemakers         750m  ...         Green
7                  kaapi royale           3140  ...  Bluish-Green
8                  kaapi royale           1000  ...         Green
9                    ugacof ltd     900-1300  ...         Green
10                   ugacof ltd          1095  ...         Green
11                 kaapi royale           1000  ...         Green
12                   cafemakers          750m  ...         Green
13  kasozi coffee farmers association        1367  ...         Green
14      ankole coffee producers coop        1488  ...         Green
15               sethuraman estate       1000m  ...         Green
16                 cafemakers, llc         750m  ...    Blue-Green
17               kawacom uganda ltd        1600  ...         Green
18                  nitubaasa ltd        1745  ...         Green
19             mannya coffee project        1200  ...         Green
20                     cafemakers         750m  ...  Bluish-Green
21                 cafemakers, llc         750m  ...         Green
22                 cafemakers, llc        3000'  ...         Green
23                       robustasa          NaN  ...    Blue-Green
24                       robustasa           40  ...    Blue-Green
25           global opportunity fund  795 meters  ...           NaN
26                    cafe politico          NaN  ...         Green
27                    cafe politico          NaN  ...          None

   Category.Two.Defects           Expiration  \
0                     2      June 26th, 2015
1                     2   October 31st, 2018
2                     0     April 29th, 2016
3                     7      July 14th, 2015
4                     3      June 26th, 2015
5                     0  February 28th, 2013
6                     0       May 15th, 2015
7                     0   October 25th, 2018
8                     0    August 17th, 2017
9                     6     August 5th, 2015
10                    1      June 26th, 2015
11                    0    August 23rd, 2017
12                    1       May 19th, 2015
13                    7      July 14th, 2015
14                    2      July 14th, 2015
15                    0     April 29th, 2016
16                    0       June 3rd, 2014
17                    1      June 27th, 2015
18                    2      June 27th, 2015
19                    1      June 27th, 2015
20                    1       May 19th, 2015
21                    0      June 20th, 2014
22                    0  February 28th, 2013
23                    1   January 18th, 2017
24                    0   January 18th, 2017
```

```
25                  6  December 23rd, 2015
26                  1    August 25th, 2015
27                  9    August 25th, 2015

                   Certification.Body  \
0   Uganda Coffee Development Authority
1          Specialty Coffee Association
2          Specialty Coffee Association
3   Uganda Coffee Development Authority
4   Uganda Coffee Development Authority
5          Specialty Coffee Association
6          Specialty Coffee Association
7          Specialty Coffee Association
8          Specialty Coffee Association
9   Uganda Coffee Development Authority
10  Uganda Coffee Development Authority
11         Specialty Coffee Association
12         Specialty Coffee Association
13  Uganda Coffee Development Authority
14  Uganda Coffee Development Authority
15         Specialty Coffee Association
16         Specialty Coffee Association
17  Uganda Coffee Development Authority
18  Uganda Coffee Development Authority
19  Uganda Coffee Development Authority
20         Specialty Coffee Association
21         Specialty Coffee Association
22         Specialty Coffee Association
23         Specialty Coffee Association
24         Specialty Coffee Association
25         Specialty Coffee Association
26         Specialty Coffee Association
27         Specialty Coffee Association

                     Certification.Address  \
0   e36d0270932c3b657e96b7b0278dfd85dc0fe743
1   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
2   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
3   e36d0270932c3b657e96b7b0278dfd85dc0fe743
4   e36d0270932c3b657e96b7b0278dfd85dc0fe743
5   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
6   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
7   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
8   ff7c18ad303d4b603ac3f8cff7e611ffc735e720
9   e36d0270932c3b657e96b7b0278dfd85dc0fe743
10  e36d0270932c3b657e96b7b0278dfd85dc0fe743
11  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
12  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
13  e36d0270932c3b657e96b7b0278dfd85dc0fe743
14  e36d0270932c3b657e96b7b0278dfd85dc0fe743
15  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
16  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
17  e36d0270932c3b657e96b7b0278dfd85dc0fe743
18  e36d0270932c3b657e96b7b0278dfd85dc0fe743
19  e36d0270932c3b657e96b7b0278dfd85dc0fe743
20  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
21  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
22  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
23  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
24  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
25  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
26  ff7c18ad303d4b603ac3f8cff7e611ffc735e720
27  ff7c18ad303d4b603ac3f8cff7e611ffc735e720

                     Certification.Contact unit_of_measurement  \
0   03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
1   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
2   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
3   03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
4   03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
5   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
6   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
7   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
8   352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
9   03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
10  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
11  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
12  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
13  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
14  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
15  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
16  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
17  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
18  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
19  03077a1c6bac60e6f514691634a7f6eb5c85aae8                   m
20  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
21  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
22  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
23  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
24  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
25  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
26  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m
27  352d0cf7f3e9be14dad7df644ad65efc27605ae2                   m

   altitude_low_meters altitude_high_meters altitude_mean_meters
0               1488.0               1488.0                1488.0
1               3170.0               3170.0                3170.0
2               1000.0               1000.0                1000.0
3               1212.0               1212.0                1212.0
4               1200.0               1300.0                1250.0
5               3000.0               3000.0                3000.0
6                750.0                750.0                 750.0
7               3140.0               3140.0                3140.0
8               1000.0               1000.0                1000.0
9                900.0               1300.0                1100.0
10              1095.0               1095.0                1095.0
11              1000.0               1000.0                1000.0
12               750.0                750.0                 750.0
13              1367.0               1367.0                1367.0
14              1488.0               1488.0                1488.0
15              1000.0               1000.0                1000.0
16               750.0                750.0                 750.0
17              1600.0               1600.0                1600.0
18              1745.0               1745.0                1745.0
19              1200.0               1200.0                1200.0
20               750.0                750.0                 750.0
21               750.0                750.0                 750.0
22              3000.0               3000.0                3000.0
23                 NaN                  NaN                   NaN
24                40.0                 40.0                  40.0
25               795.0                795.0                 795.0
26                 NaN                  NaN                   NaN
27                 NaN                  NaN                   NaN

[28 rows x 44 columns]
```

3.1. Examining the Structure of a Data Frame

I told you this was a DataFrame, but we can check with type.

```
type(coffee_df)
```

```
pandas.core.frame.DataFrame
```

We can also see that the DataFrame type comes from the `pandas` library, without the library loaded this type does not exist.

We can also exmaine its parts. It consists of several; first the column headings

```
coffee_df.columns
```

```
Index(['Unnamed: 0', 'Species', 'Owner', 'Country.of.Origin', 'Farm.Name',
       'Lot.Number', 'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region',
       'Producer', 'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner',
       'Harvest.Year', 'Grading.Date', 'Owner.1', 'Variety',
       'Processing.Method', 'Fragrance...Aroma', 'Flavor', 'Aftertaste',
       'Salt...Acid', 'Bitter...Sweet', 'Mouthfeel', 'Uniform.Cup',
       'Clean.Cup', 'Balance', 'Cupper.Points', 'Total.Cup.Points', 'Moisture',
       'Category.One.Defects', 'Quakers', 'Color', 'Category.Two.Defects',
       'Expiration', 'Certification.Body', 'Certification.Address',
       'Certification.Contact', 'unit_of_measurement', 'altitude_low_meters',
       'altitude_high_meters', 'altitude_mean_meters'],
      dtype='object')
```

These are a special type called Index

```
type(coffee_df.columns)
```

```
pandas.core.indexes.base.Index
```

These are still iterable, much like python lists.

and it stores the data

```
coffee_df.values
```

```
array([[1, 'Robusta', 'ankole coffee producers coop', ..., 1488.0,
        1488.0, 1488.0],
       [2, 'Robusta', 'nishant gurjer', ..., 3170.0, 3170.0, 3170.0],
       [3, 'Robusta', 'andrew hetzel', ..., 1000.0, 1000.0, 1000.0],
       ...,
       [26, 'Robusta', 'james moore', ..., 795.0, 795.0, 795.0],
       [27, 'Robusta', 'cafe politico', ..., nan, nan, nan],
       [28, 'Robusta', 'cafe politico', ..., nan, nan, nan]], dtype=object)
```

It also has an index (first column, visually) but it is special because this is how you can index the data.

```
coffee_df.index
```

```
RangeIndex(start=0, stop=28, step=1)
```

Right now this is an autogenerated index, but we can also use the `index_col` parameter to set that up front.

```
coffee_df = pd.read_csv(coffee_data_url,index_col=0)
coffee_df
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Color	Category.Two.Defects
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	1488	sheema south western	...	Green	2
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi royale	3170	chikmagalur karnataka indua	...	NaN	2
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuraman estate	1000m	chikmagalur	...	Green	0
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof ltd	1212	central	...	Green	7
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	0	katuka development trust ltd	1200-1300	luwero central region	...	Green	3
6	Robusta	andrew hetzel	India	NaN	NaN	(self)	NaN	cafemakers, llc	3000'	chikmagalur	...	Green	0
7	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	chikmagalur	...	Green	0
8	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethuraman estate	14/1148/2017/18	kaapi royale	3140	chikmagalur karnataka india	...	Bluish-Green	0
9	Robusta	nishant gurjer	India	sethuraman estate	RKR	sethuraman estate	14/1148/2016/17	kaapi royale	1000	chikmagalur karnataka	...	Green	0
10	Robusta	ugacof	Uganda	ishaka	NaN	nsubuga umar	0	ugacof ltd	900-1300	western	...	Green	6
11	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof ltd	1095	iganga namadrope eastern	...	Green	1
12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuraman estate	14/1148/2016/12	kaapi royale	1000	chikmagalur karnataka	...	Green	0
13	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	chikmagalur	...	Green	1
14	Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN	NaN	0	kasozi coffee farmers association	1367	eastern	...	Green	7
15	Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	ankole coffee producers coop union ltd	0	ankole coffee producers coop	1488	south western	...	Green	2
16	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuraman estate	1000m	chikmagalur	...	Green	0
17	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	750m	chikmagalur	...	Blue-Green	0
18	Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	kawacom	0	kawacom uganda ltd	1600	western	...	Green	1
19	Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nitubaasa	0	nitubaasa ltd	1745	western	...	Green	2
20	Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	mannya coffee project	0	mannya coffee project	1200	southern	...	Green	1
21	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN	NaN	cafemakers	750m	chikmagalur	...	Bluish-Green	1
22	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	750m	chikmagalur	...	Green	0
23	Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethuraman estates	NaN	cafemakers, llc	3000'	chikmagalur	...	Green	0
24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	NaN	robustasa	NaN	san juan, playas	...	Blue-Green	1
25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	NaN	robustasa	40	san juan, playas	...	Blue-Green	0
26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN	global opportunity fund	795 meters	kwanza norte province, angola	...	NaN	6
27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	cafe politico	NaN	NaN	...	Green	1
28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN	cafe politico	NaN	NaN	...	None	9

28 rows × 43 columns

```
coffee_df.index
```

```
Int64Index([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
            18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28],
           dtype='int64')
```

Now it's neater

3.2. Extracting Parts of Data Frames

We can look at the first 5 rows with `head`

```
coffee_df.head()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Color	Category.Two.Defects	E
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	1488	sheema south western	...	Green	2	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi royale	3170	chikmagalur karnataka indua	...	NaN	2	
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	0000	sethuraman estate	1000m	chikmagalur	...	Green	0	
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	0	ugacof ltd	1212	central	...	Green	7	
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	0	katuka development trust ltd	1200-1300	luwero central region	...	Green	3	

5 rows × 43 columns

and the last 5 with `tail`

```
coffee_df.tail()
```

> **Try it yourself**
>
> How can you look at the first 3 or last 2 rows?

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Color	Category.Two.Defects	Expiration	Cer
24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab	NaN	robustasa	NaN	san juan, playas	...	Blue-Green	1	January 18th, 2017	
25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory	NaN	robustasa	40	san juan, playas	...	Blue-Green	0	January 18th, 2017	
26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN	global opportunity fund	795 meters	kwanza norte province, angola	...	NaN	6	December 23rd, 2015	
27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087	cafe politico	NaN	NaN	...	Green	1	August 25th, 2015	
28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN	cafe politico	NaN	NaN	...	None	9	August 25th, 2015	

5 rows × 43 columns

the shape of a DataFrame is an attribute

```
coffee_df.shape
```

```
(28, 43)
```

```
len(coffee_df)
```

```
28
```

We can pick out columns by name.

```
coffee_df['Species']
```

```
1     Robusta
2     Robusta
3     Robusta
4     Robusta
5     Robusta
6     Robusta
7     Robusta
8     Robusta
9     Robusta
10    Robusta
11    Robusta
12    Robusta
13    Robusta
14    Robusta
15    Robusta
16    Robusta
17    Robusta
18    Robusta
19    Robusta
20    Robusta
21    Robusta
22    Robusta
23    Robusta
24    Robusta
25    Robusta
26    Robusta
27    Robusta
28    Robusta
Name: Species, dtype: object
```

> **Important**
>
> We did not do this step in class

We can pick out rows with `loc`

```
coffee_df.loc[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/indexes/base.py:3803, in Index.get_loc(self, key, method,
tolerance)
   3802 try:
-> 3803     return self._engine.get_loc(casted_key)
   3804 except KeyError as err:

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/_libs/index.pyx:138, in pandas._libs.index.IndexEngine.get_loc()

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/_libs/index.pyx:165, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:2263, in
pandas._libs.hashtable.Int64HashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:2273, in
pandas._libs.hashtable.Int64HashTable.get_item()

KeyError: 0

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

KeyError                                  Traceback (most recent call last)
Cell In [18], line 1
----> 1 coffee_df.loc[0]

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/indexing.py:1073, in _LocationIndexer.__getitem__(self, key)
   1070 axis = self.axis or 0
   1072 maybe_callable = com.apply_if_callable(key, self.obj)
-> 1073 return self._getitem_axis(maybe_callable, axis=axis)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/indexing.py:1312, in _LocIndexer._getitem_axis(self, key,
axis)
   1310 # fall thru to straight lookup
   1311 self._validate_key(key, axis)
-> 1312 return self._get_label(key, axis=axis)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/indexing.py:1260, in _LocIndexer._get_label(self, label,
axis)
   1258 def _get_label(self, label, axis: int):
   1259     # GH#5567 this will fail if the label is not present in the axis.
-> 1260     return self.obj.xs(label, axis=axis)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/generic.py:4056, in NDFrame.xs(self, key, axis, level,
drop_level)
   4054             new_index = index[loc]
   4055 else:
-> 4056     loc = index.get_loc(key)
   4058     if isinstance(loc, np.ndarray):
   4059         if loc.dtype == np.bool_:

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/indexes/base.py:3805, in Index.get_loc(self, key, method,
tolerance)
   3803     return self._engine.get_loc(casted_key)
   3804 except KeyError as err:
-> 3805     raise KeyError(key) from err
   3806 except TypeError:
   3807     # If we have a listlike key, _check_indexing_error will raise
   3808     #  InvalidIndexError. Otherwise we fall through and re-raise
   3809     #  the TypeError.
   3810     self._check_indexing_error(key)

KeyError: 0
```

3.3. Reading data from websites

We'll first read from the course website.

> **Note**
>
> This is our first bit of web scraping! We will do more, but for very structured data it can be this easy

```
comm_url =
'https://rhodyprog4ds.github.io/BrownFall22/syllabus/communication.html#'
```

So far, we've read data in from a .csv file with `pd.read_csv` and created a DataFrame with the constructor `pd.DataFrame` using a dictionary. Pandas provides many interfaces for reading in data. They're described on the Pandas IO page.

We can use the `read_html` method to read from this page. We know that it has multiple tables on the page, and from the help, we know that it will return a list of DataFrames.

```
df_list = pd.read_html(comm_url)
```

We can also verify what it returns

```
type(df_list)
```

```
list
```

We can index with `[]` to pick one item from the list and verify that it is a DataFrame.

```
type(df_list[0])
```

```
pandas.core.frame.DataFrame
```

3.4. Pythonic Loops

In Python, loops do not require an iterator variable. It has an interable object and a loop variable.

```
for loop_variable in iterable_object:
    # loop body
```

> **Note**
>
> Using the documentation for a library (and the base language) is totally expected and normal part of programming. That's what you should use as your primary source for questions in this class. Other sources can become outdated pretty quickly as the language changes, but most of the libraries we'll use have processes in place to ensure that their own documentation gets updated at the same time the code does.

> **Warning**
>
> If you use other sources and get advised to solutions that are deprecated you may not earn achievements for that work.

the `loop_variable` takes on the value of each item in the `iterable_object` each time it goes through, in order. Writing loops this way makes them more compact and more readable, this is more like English. For example:

```
name = 'sarah'
for letter in name:
    print(letter.upper())
```

```
S
A
R
A
H
```

It is best to name variables so that the loop variable makes sense as an item from the iterable. For example, names have letters in them, and an item in `df_list` makes sense as `df`.

```
for df in df_list:
    print(df.shape)
```

```
(6, 4)
(6, 4)
(1, 3)
(3, 3)
(2, 3)
```

3.5. Types Solution

> **Warning**
>
> I am using bad variable names here `a`, `b` ,... because these are only as options for a question and we will not use them again

```
a = [char for char in 'abcde']
a
```

This is called a list comprehension. It allows you to build a list using a for loop all in one step.

```
['a', 'b', 'c', 'd', 'e']
```

```
type(a)
```

```
list
```

```
b = {char:i for i, char in enumerate('abcde')}
b
```

This is called a dictionary comprehension. It allows you to build a dictionary using a for loop all in one step.

```
{'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4}
```

```
type(b)
```

```
dict
```

```
c = ('a','b','c','d','e')
c
```

```
('a', 'b', 'c', 'd', 'e')
```

```
type(c)
```

```
tuple
```

```
d = 'a b c d e'.split('')
d
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [31], line 1
----> 1 d = 'a b c d e'.split('')
      2 d

ValueError: empty separator
```

```
type(d)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [32], line 1
----> 1 type(d)

NameError: name 'd' is not defined
```

3.6. Questions After Class

3.6.1. what is a dictionary in python?

a dictionary is a datatype from base python that stores key, value pairs.

For example

```
prof_info = {'first':'Sarah', 'last':'Brown', 'title':'Dr.'}
prof_info
```

```
{'first': 'Sarah', 'last': 'Brown', 'title': 'Dr.'}
```

We can use the keys to index in and get the values out

```
prof_info['title']
```

```
'Dr.'
```

Even though we will mostly use DataFrame, dictionaries and other base python types are important. Dictionaries are very powerful they can hold whole functions in them. For example, the Python language does not have a switch case (which can be used for handling many if/else cases) but instead dictionaries can be used for that.

> **Further Reading**
>
> You can read more about the details of data types in Pandas in the documentation

3.6.2. How to see unique values in a column

We will get to this soon! We got the first part, picking out a single column to look at, we will see the method for that probably on Monday, but maybe on Friday.

4. Iterables and Indexing

4.1. Logistics

4.1.1. Assignment

On Assignment 2, there's a script to run to prepare it for grading.

> **Important**
>
> I missed a file in the template, so please create a file called requirements.txt that has the following contents.
>
> ```
> jupyter-book
> pyppeteer
> pandas
> jupytext
> ```

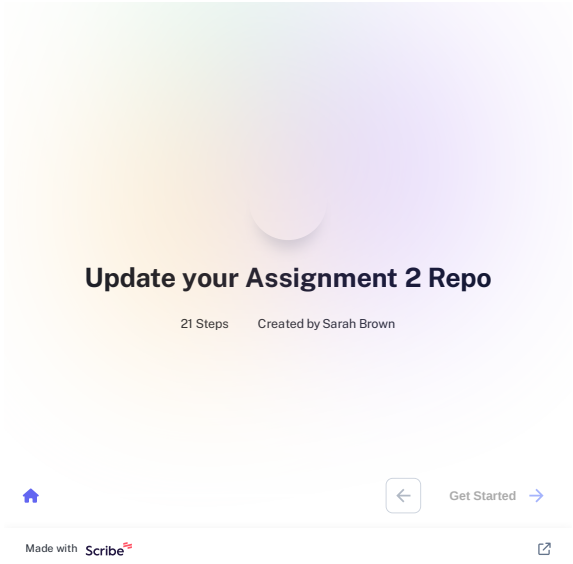
Correct script file (for .githu/workflows/submit.yml)

```
name: Prepare and Submit
on:
  workflow_dispatch

jobs:
  generatereport:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    # Install dependencies
    - name: Set up Python 3.9
      uses: actions/setup-python@v1
      with:
        python-version: 3.9

    # install dependencies
    - name: dependencies
      run: pip install -r requirements.txt

    # generate the report
    - name: Convert notebooks to md for reading
      run: |
        jupytext --to myst *.ipynb
        jupytext --to myst */*.ipynb

    # compile the pdf
    - name: Build a basic html to pdf
      run: |
        jupyter-book build *.ipynb --builder pdfhtml

    # start a pull request for it
    - name: Create Pull Request
      uses: peter-evans/create-pull-request@v4
      with:
        commit-message: 'convert to md'
        draft: false
        branch: published
        base: main
        title: Submission
```

Add this to `.gitignore`

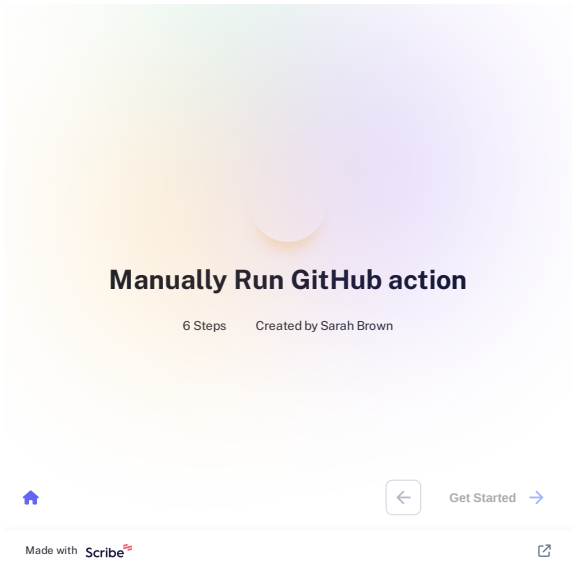
```
_build/
```

How to apply these updates:



full view

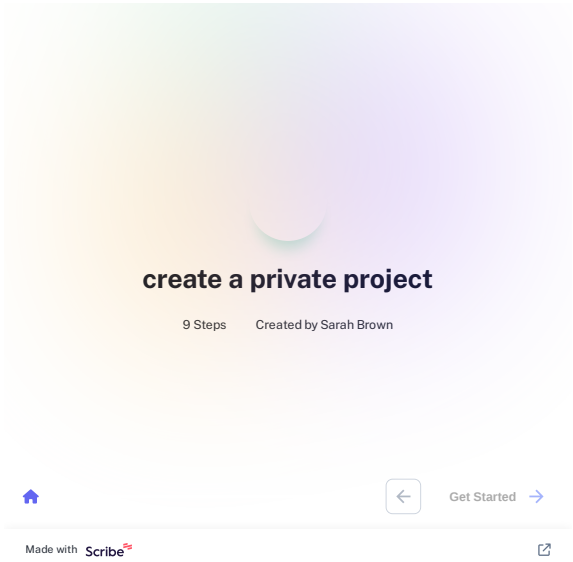
To run the script



see instructions in a full page

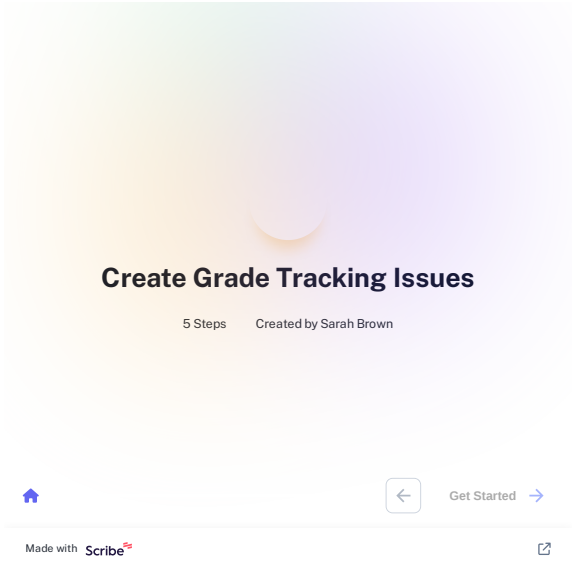
4.1.2. Grade Tracking

Create a Project board



Then create your repo and run the manual workflow on that actions tab to create issues:

> **Note**
>
> Are these scribe screenshots helpful? do you want more of them?



4.2. Lists of lists

```
import pandas as pd
```

Recall how indexing works with negagive numbers

```
topics = ['what is data science', 'jupyter', 'conditional','functions', 'lists',
'dictionaries','pandas' ]
```

```
topics[-1]
```

```
'pandas'
```

A list we built together:

```
# You can remain anonymous (this page & the notes will be fully public)
# by attributing it to a celebrity or psuedonym, but include *some* sort of
attribution
sentence_list = [
"The class is just starting to feel settled for me. - Dr. Brown",
"",
"Hello, I like sushi! - ",
"Why squared  aka the mask - is a computer science student."
"Data science is fun",
"Hello my fellow gaymers - Sun Tzu",
"Soccer is a sport -Obama",
"Hello, I love pizza - Bear",
"This class is CSC/DSP 310. - Student",
"It is 2:21pm -",
"Pizza conquers all- Beetlejuice"
"",
"ayyy whaddup wit it - frankie",
"This is a sentence - George W Bush",
"Steam is the best place to play videogames change my mind. - Todd Howard",
"This is a hello -",
"Hello how are you -",
"The monkey likes bananas. - A banana",
"",
"Just type a random sentence - Rosa Parks",
"",
"I love CSC. - Everyone",
"",
"The quick brown fox jumps over the lazy dog - Brendan Chadwick",
"I like computers - David",
"",
"The fitness gram pacer test is a multi aerobic capacity test - Matt 3",
"Sally sells seashells by the seashore. - Narrator",
"I would like to take a nap. - Tom Cruise,"
]
```

We can confirm this is a list

```
type(sentence_list)
```

```
list
```

We can use a list comprehension to remove the empty ones.

```
sentences_clean = [s for s in sentence_list if len(s)>1]
sentences_clean
```

```
['The class is just starting to feel settled for me. - Dr. Brown',
 'Hello, I like sushi! - ',
 'Why squared  aka the mask - is a computer science student.Data science is fun',
 'Hello my fellow gaymers - Sun Tzu',
 'Soccer is a sport -Obama',
 'Hello, I love pizza - Bear',
 'This class is CSC/DSP 310. - Student',
 'It is 2:21pm -',
 'Pizza conquers all- Beetlejuice',
 'ayyy whaddup wit it - frankie',
 'This is a sentence - George W Bush',
 'Steam is the best place to play videogames change my mind. - Todd Howard',
 'This is a hello -',
 'Hello how are you -',
 'The monkey likes bananas. - A banana',
 'Just type a random sentence - Rosa Parks',
 'I love CSC. - Everyone',
 'The quick brown fox jumps over the lazy dog - Brendan Chadwick',
 'I like computers - David',
 'The fitness gram pacer test is a multi aerobic capacity test - Matt 3',
 'Sally sells seashells by the seashore. - Narrator',
 'I would like to take a nap. - Tom Cruise,']
```

We can use the `split` method on a string to make a list of lists

```
sentence_data = [s.split('-') for s in sentences_clean]
sentence_data
```

```
[['The class is just starting to feel settled for me. ', ' Dr. Brown'],
 ['Hello, I like sushi! ', ' '],
 ['Why squared  aka the mask ',
  ' is a computer science student.Data science is fun'],
 ['Hello my fellow gaymers ', ' Sun Tzu'],
 ['Soccer is a sport ', 'Obama'],
 ['Hello, I love pizza ', ' Bear'],
 ['This class is CSC/DSP 310. ', ' Student'],
 ['It is 2:21pm ', ''],
 ['Pizza conquers all', ' Beetlejuice'],
 ['ayyy whaddup wit it ', ' frankie'],
 ['This is a sentence ', ' George W Bush'],
 ['Steam is the best place to play videogames change my mind. ',
  ' Todd Howard'],
 ['This is a hello ', ''],
 ['Hello how are you ', ''],
 ['The monkey likes bananas. ', ' A banana'],
 ['Just type a random sentence ', ' Rosa Parks'],
 ['I love CSC. ', ' Everyone'],
 ['The quick brown fox jumps over the lazy dog ', ' Brendan Chadwick'],
 ['I like computers ', ' David'],
 ['The fitness gram pacer test is a multi aerobic capacity test ', ' Matt 3'],
 ['Sally sells seashells by the seashore. ', ' Narrator'],
 ['I would like to take a nap. ', ' Tom Cruise,']]
```

then we can use the DataFrame Constructor)

```
sentence_df = pd.DataFrame(data= sentence_data, columns=
['sentence','attribution'])
```

We can use head by default for 5

```
sentence_df.head()
```

	sentence	attribution
0	The class is just starting to feel settled for...	Dr. Brown
1	Hello, I like sushi!	
2	Why squared aka the mask	is a computer science student.Data science is...
3	Hello my fellow gaymers	Sun Tzu
4	Soccer is a sport	Obama

or pass a nubmer to get a different number of rows

```
sentence_df.head(3)
```

	sentence	attribution
0	The class is just starting to feel settled for...	Dr. Brown
1	Hello, I like sushi!	
2	Why squared aka the mask	is a computer science student.Data science is...

the `loc` property can index on rows or columns, but is rows by deault

```
sentence_df.loc[3]
```

```
sentence       Hello my fellow gaymers
attribution                   Sun Tzu
Name: 3, dtype: object
```

We can select a range, like in base python with a colon

```
sentence_df.loc[3:5]
```

	sentence	attribution
3	Hello my fellow gaymers	Sun Tzu
4	Soccer is a sport	Obama
5	Hello, I love pizza	Bear

4.3. Loading a json

We can load a json using the `read_json` method the same way we used the `read_csv`

```
rhodyprog4ds_gh_events_url = 'https://api.github.com/orgs/rhodyprog4ds/events'
```

```
pd.read_json(rhodyprog4ds_gh_events_url)
```

	id	type	actor	repo	payload	public	created_at	org
0	25320514276	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11704698397, 'size': 1, 'distinct_...	True	2022-11-18 03:09:50+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
1	25144088977	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11613190412, 'size': 1, 'distinct_...	True	2022-11-10 02:48:15+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
2	25144043727	ReleaseEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'published', 'release': {'url': 'ht...	True	2022-11-10 02:44:25+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
3	25144027183	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'ref': 'c27', 'ref_type': 'tag', 'master_bran...	True	2022-11-10 02:43:00+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
4	25144017610	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11613152649, 'size': 2, 'distinct_...	True	2022-11-10 02:42:09+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
5	25142512510	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11612387217, 'size': 1, 'distinct_...	True	2022-11-10 00:49:41+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
6	25142364425	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11612318098, 'size': 3, 'distinct_...	True	2022-11-10 00:40:43+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
7	25139682863	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11611012581, 'size': 1, 'distinct_...	True	2022-11-09 21:54:16+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
8	25139577301	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11610961795, 'size': 1, 'distinct_...	True	2022-11-09 21:48:21+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
9	25139071066	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11610711539, 'size': 1, 'distinct_...	True	2022-11-09 21:18:56+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
10	25138999879	IssuesEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'closed', 'issue': {'url': 'https:/...	True	2022-11-09 21:14:50+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
11	25138969075	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11610660531, 'size': 1, 'distinct_...	True	2022-11-09 21:13:04+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
12	25138947856	IssuesEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'closed', 'issue': {'url': 'https:/...	True	2022-11-09 21:11:50+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
13	25115458774	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11599023856, 'size': 1, 'distinct_...	True	2022-11-09 01:17:52+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
14	25087626486	ReleaseEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'published', 'release': {'url': 'ht...	True	2022-11-08 01:56:11+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
15	25087620296	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'ref': 'c26', 'ref_type': 'tag', 'master_bran...	True	2022-11-08 01:55:44+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
16	25087612809	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11585346623, 'size': 1, 'distinct_...	True	2022-11-08 01:55:10+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
17	25043076741	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11560187220, 'size': 1, 'distinct_...	True	2022-11-05 00:02:20+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
18	25043031795	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11560160777, 'size': 1, 'distinct_...	True	2022-11-04 23:56:23+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
19	25042967428	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11560123632, 'size': 1, 'distinct_...	True	2022-11-04 23:47:14+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
20	25042941563	ReleaseEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'published', 'release': {'url': 'ht...	True	2022-11-04 23:43:41+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...

	id	type	actor	repo	payload	public	created_at	org
21	25042936184	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'ref': 'c25', 'ref_type': 'tag', 'master_bran...	True	2022-11-04 23:42:56+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
22	25042926798	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11560100188, 'size': 2, 'distinct_...	True	2022-11-04 23:41:37+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
23	24997307055	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11536809007, 'size': 1, 'distinct_...	True	2022-11-03 02:32:54+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
24	24997291157	IssuesEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'opened', 'issue': {'url': 'https:/...	True	2022-11-03 02:31:13+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
25	24997284566	ReleaseEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'published', 'release': {'url': 'ht...	True	2022-11-03 02:30:32+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
26	24997260517	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'ref': 'c24', 'ref_type': 'tag', 'master_bran...	True	2022-11-03 02:27:33+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
27	24997253357	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11536777920, 'size': 1, 'distinct_...	True	2022-11-03 02:26:49+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
28	24997193960	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 297149047, 'name': 'rhodyprog4ds/assign...	{'push_id': 11536743852, 'size': 1, 'distinct_...	True	2022-11-03 02:20:34+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
29	24995548550	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11535803728, 'size': 1, 'distinct_...	True	2022-11-02 23:34:09+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...

4.4. Working with your repo offline

> **Warning**
>
> This was not done in class and is optional

4.4.1. Authenticate on Mac

On macOS install GitHub CLI

```
gh auth login
```

use defualts and choose to log in via browser.

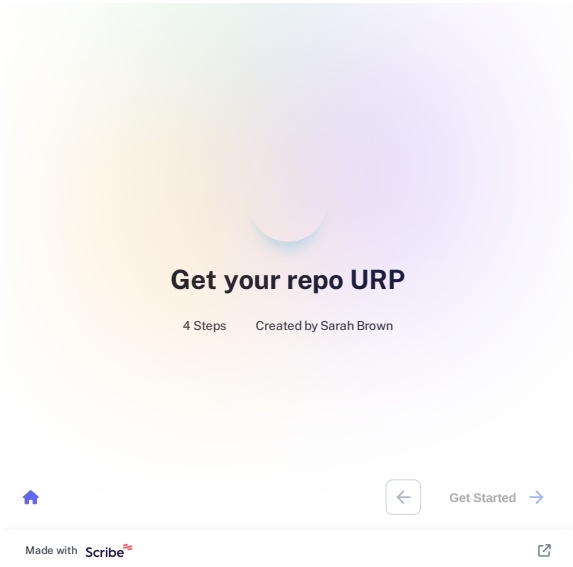
4.4.2. Authenticate on Windows

On winows install GitBash

Then try to do the clone step and GitBash will help you authenticate

4.4.3. Work offline

Get your repo URL:



cd to where you want to save

```
cd prog4ds
git clone https://github.com/rhodyprog4ds/02-loading-data-brownsarahm.git
```

work in the new folder that creates

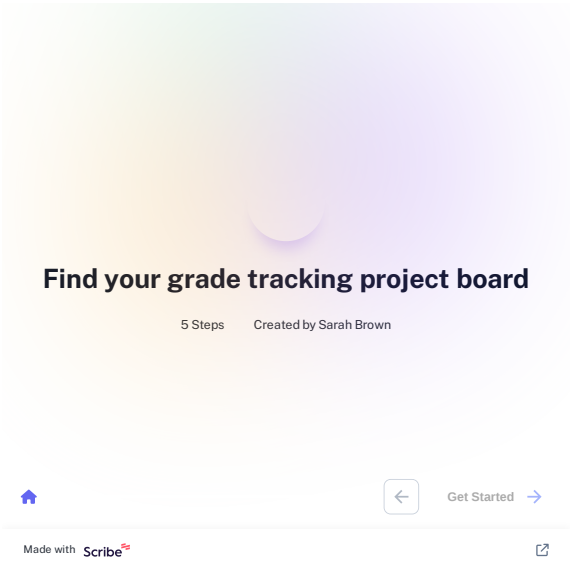
When you want to Save

```
git add .
git commit -m 'describe the work you did'
git push
```

4.5. Questions After Class

4.5.1. Logistics

4.5.1.1. where do we find the grading page?



4.5.2. Assignemnt

4.5.2.1. what are the keys needed on the dictionaries for the assignment?

See the `datasets.py` file in the template repo

4.5.2.2. do we have to accept assignment 2 anywhere and if so how

Yes on the assignment page. The link says “accept the assignment”

4.5.2.3. For the purposes of the assignment should we download it locally to work with our notebooks?

Read the instructions carefully on the assignment. It tells you exactly what to do.

4.5.3. Content

4.5.3.1. How do you locate a specific row and column from a dataframe?

`.loc` accepts both, using a comma to separate. The docs for `loc` have lots of examples.

4.5.3.2. with data sets if there is an error with formatting and we can modify the original how would we fix it

Download to a copy where you can edit.

4.5.3.3. Can you use .loc to pull out multiple rows that aren’t next to each other. For example, if I wanted to view rows 3, 8 and 12

Yes, to select multiple nonconsecutive, you pass a list. The docs for `loc` have lots of examples.

4.5.3.4. how can we iterate through dictionaries

dictionaries have a `.items()` method that pops off tuples of th key and value.

> **Warning**
>
> the assignment does not ask you to iterate through a dictionary object, but over a list of dictionaries

4.5.3.5. What is the main differnce between JSON and csv files; does one allocate more memory / store larger sets?

The main difference is the structure. JSON can hold nested data. For example look at the GitHub data that we read in in class.

4.5.3.6. what exactly does json mean / do

json is a data file format. It is an acronym for JavaScript Object Notation. It’s a popular format for internet content.

4.5.3.7. are nested lists the only way to create DataFrames in python

nested lists are not the only way to create pandas DataFrames, you can also do that from a Dictionary. see in the docs for the constructor.

4.5.3.8. How do nested loops work in the jupyter notebook

Python constructs other than display items work just as they do in any other interpretter in a jupyter notebook. The list comprehension that we saw today also works in base Python. You can nest list comphrehenions in different ways depending on your goal.

5. Getting Started with Exploratory Data Analysis

Now we get to start actual data science!

Our goal this week is to explore the actual data, the values, in a dataset to get a basic idea of what is in the data.

Summarizing and Visualizing Data are **very** important

- People cannot interpret high dimensional or large samples quickly
- Important in EDA to help you make decisions about the rest of your analysis
- Important in how you report your results
- Summaries are similar calculations to performance metrics we will see later
- visualizations are often essential in debugging models

THEREFORE

- You have a lot of chances to earn summarize and visualize
- we will be picky when we assess if you earned them or not

```
import pandas as pd
```

5.1. Staying Organized

- See the new File structure section.
- Be sure to accept assignments and close the feedback PR if you will not work on them

5.2. Loading Data and Examining Structure

```
coffee_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_data_cleaned.csv'
coffee_df = pd.read_csv(coffee_data_url,index_col=0)
```

So far, we've loaded data in a few different ways and then we've examined DataFrames as a data structure, looking at what different attributes they have and what some of the methods are, and how to get data into them.

```
coffee_df.head(2)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Color	Category.Two.Defects	Expirat
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	0	ankole coffee producers coop	1488	sheema south western	...	Green	2	June 2 2
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/2017/21	kaapi royale	3170	chikmagalur karnataka indua	...	NaN	2	Octo 31st, 2

2 rows × 43 columns

From here we can see a few sample values of most of the column and we can be sure that the data loaded correctly.

> **Try it Yourself**
>
> What other tools have we learned to examine a DataFrame and when might you use them?

We can also get more structural information with the **`info`** method.

More information on this can also be found in the dtypes attribute. Including that the type is the most general if there are multiple types in the column.

```
coffee_df.info()
```

> **Hint**
>
> There is also a related **`select_dtypes`** method.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28 entries, 1 to 28
Data columns (total 43 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Species                28 non-null     object 
 1   Owner                  28 non-null     object 
 2   Country.of.Origin      28 non-null     object 
 3   Farm.Name              25 non-null     object 
 4   Lot.Number             6 non-null      object 
 5   Mill                   20 non-null     object 
 6   ICO.Number             17 non-null     object 
 7   Company                28 non-null     object 
 8   Altitude               25 non-null     object 
 9   Region                 26 non-null     object 
 10  Producer               26 non-null     object 
 11  Number.of.Bags         28 non-null     int64  
 12  Bag.Weight             28 non-null     object 
 13  In.Country.Partner     28 non-null     object 
 14  Harvest.Year           28 non-null     int64  
 15  Grading.Date           28 non-null     object 
 16  Owner.1                28 non-null     object 
 17  Variety                3 non-null      object 
 18  Processing.Method      10 non-null     object 
 19  Fragrance...Aroma      28 non-null     float64
 20  Flavor                 28 non-null     float64
 21  Aftertaste             28 non-null     float64
 22  Salt...Acid            28 non-null     float64
 23  Bitter...Sweet         28 non-null     float64
 24  Mouthfeel              28 non-null     float64
 25  Uniform.Cup            28 non-null     float64
 26  Clean.Cup              28 non-null     float64
 27  Balance                28 non-null     float64
 28  Cupper.Points          28 non-null     float64
 29  Total.Cup.Points       28 non-null     float64
 30  Moisture               28 non-null     float64
 31  Category.One.Defects   28 non-null     int64  
 32  Quakers                28 non-null     int64  
 33  Color                  26 non-null     object 
 34  Category.Two.Defects   28 non-null     int64  
 35  Expiration             28 non-null     object 
 36  Certification.Body     28 non-null     object 
 37  Certification.Address  28 non-null     object 
 38  Certification.Contact  28 non-null     object 
 39  unit_of_measurement    28 non-null     object 
 40  altitude_low_meters    25 non-null     float64
 41  altitude_high_meters   25 non-null     float64
 42  altitude_mean_meters   25 non-null     float64
dtypes: float64(15), int64(5), object(23)
memory usage: 9.6+ KB
```

5.3. Summary Statistics

Now, we can actually start to analyze the data itself.

The **`describe`** method provides us with a set of summary statistics that broadly describe the data overall.

```
coffee_df.describe()
```

	Number.of.Bags	Harvest.Year	Fragrance...Aroma	Flavor	Aftertaste	Salt...Acid	Bitter...Sweet	Mouthfeel	Uniform.Cup	Clean.Cup	Balance	Cupper.Points
count	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000
mean	168.000000	2013.964286	7.702500	7.630714	7.559643	7.657143	7.675714	7.506786	9.904286	9.928214	7.541786	7.761429
std	143.226317	1.346660	0.296156	0.303656	0.342469	0.261773	0.317063	0.725152	0.238753	0.211030	0.526076	0.330507
min	1.000000	2012.000000	6.750000	6.670000	6.500000	6.830000	6.670000	5.080000	9.330000	9.330000	5.250000	6.920000
25%	1.000000	2013.000000	7.580000	7.560000	7.397500	7.560000	7.580000	7.500000	10.000000	10.000000	7.500000	7.580000
50%	170.000000	2014.000000	7.670000	7.710000	7.670000	7.710000	7.750000	7.670000	10.000000	10.000000	7.670000	7.830000
75%	320.000000	2015.000000	7.920000	7.830000	7.770000	7.830000	7.830000	7.830000	10.000000	10.000000	7.830000	7.920000
max	320.000000	2017.000000	8.330000	8.080000	7.920000	8.000000	8.420000	8.250000	10.000000	10.000000	8.000000	8.580000

From this, we can draw several conclusions. FOr example straightforward ones like:

- the smallest number of bags rated is 1 and at least 25% of the coffees rates only had 1 bag
- the first ratings included were 2012 and last in 2017 (min & max)
- the mean Mouthfeel was 7.5
- Category One defects are not very common (the 75th% is 0)

Or more nuanced ones that compare across variables like

- the raters scored coffee higher on Uniformity.Cup and Clean.Cup than other scores (mean score; only on the ones that seem to have a scale of up to 8/10)
- the coffee varied more in Mouthfeel and Balance that most other scores (the std; only on the ones that seem to have a scale of of up to 8/10)
- there are 3 ratings with no altitude (count of other variables is 28; alt is 25

And these all give us a sense of the values and the distribution or spread fo the data in each column.

We can use the descriptive statistics on individual columns as well.

```
coffee_df['Balance'].describe()
```

```
count    28.000000
mean      7.541786
std       0.526076
min       5.250000
25%       7.500000
50%       7.670000
75%       7.830000
max       8.000000
Name: Balance, dtype: float64
```

To dig in on what the quantiles really mean, we can compute one manually.

First, we sort the data, then for the 25%, we select the point in index 6 because becaues there are 28 values.

```
balance_sorted = coffee_df['Balance'].sort_values().values
```

What value of `x` to pick out 25%

```
x = 6
balance_sorted[x]
```

```
7.5
```

We can also extract each of the statistics that the `describe` method calculates individually, by name. The quantiles are tricky, we cannot just `.25%()` to get the 25% percentile, we have to use the `quantile` method and pass it a value between 0 and 1.

```
coffee_df['Flavor'].quantile(.8)
```

```
7.83
```

Calculate the mean of `Aftertaste`

```
coffee_df['Aftertaste'].mean()
```

```
7.559642857142856
```

5.4. Describing Nonnumerical Variables

There are different columns in the describe than the the whole dataset:

```
coffee_df.columns
```

```
Index(['Species', 'Owner', 'Country.of.Origin', 'Farm.Name', 'Lot.Number',
       'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region', 'Producer',
       'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner', 'Harvest.Year',
       'Grading.Date', 'Owner.1', 'Variety', 'Processing.Method',
       'Fragrance...Aroma', 'Flavor', 'Aftertaste', 'Salt...Acid',
       'Bitter...Sweet', 'Mouthfeel', 'Uniform.Cup', 'Clean.Cup', 'Balance',
       'Cupper.Points', 'Total.Cup.Points', 'Moisture', 'Category.One.Defects',
       'Quakers', 'Color', 'Category.Two.Defects', 'Expiration',
       'Certification.Body', 'Certification.Address', 'Certification.Contact',
       'unit_of_measurement', 'altitude_low_meters', 'altitude_high_meters',
       'altitude_mean_meters'],
      dtype='object')
```

```
coffee_df.describe().columns
```

```
Index(['Number.of.Bags', 'Harvest.Year', 'Fragrance...Aroma', 'Flavor',
       'Aftertaste', 'Salt...Acid', 'Bitter...Sweet', 'Mouthfeel',
       'Uniform.Cup', 'Clean.Cup', 'Balance', 'Cupper.Points',
       'Total.Cup.Points', 'Moisture', 'Category.One.Defects', 'Quakers',
       'Category.Two.Defects', 'altitude_low_meters', 'altitude_high_meters',
       'altitude_mean_meters'],
      dtype='object')
```

We can get the prevalence of each one with `value_counts`

```
coffee_df['Color'].value_counts()
```

```
Green           20
Blue-Green       3
Bluish-Green     2
None             1
Name: Color, dtype: int64
```

Try it Yourself

Note `value_counts` does not count the `NaN` values, but `count` counts all of the not missing values and the shape of the DataFrame is the total number of rows. How can you get the number of missing Colors?

Describe only operates on the numerical columns, but we might want to know about the others. We can get the number of each value with `value_counts`

```
coffee_df['Country.of.Origin'].value_counts()
```

```
India            13
Uganda           10
United States     2
Ecuador           2
Vietnam           1
Name: Country.of.Origin, dtype: int64
```

We can get the name of the most common country out of this Series using `idmax`

```
coffee_df['Country.of.Origin'].value_counts().idxmax()
```

```
'India'
```

Or see only how many different values with the related:

```
coffee_df['Country.of.Origin'].nunique()
```

```
5
```

We can also use the mode function, which works on both numerical or nonnumerical

```
coffee_df['Country.of.Origin'].mode()
```

```
0    India
Name: Country.of.Origin, dtype: object
```

further reading

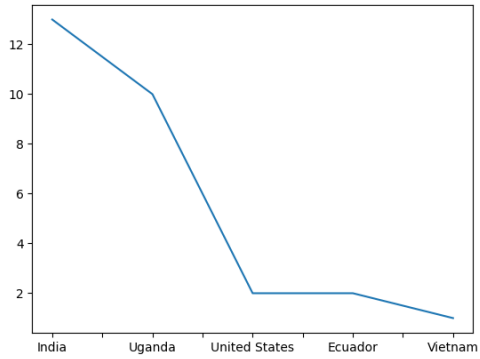
On the documentation page for describe the "See Also" shows the links to the documentation of most of the individual functions. This is a good way to learn about other things, or find something when you are not quite sure what it would be named. Go to a function thats similar to what you want and then look at the related functions.

5.5. Basic Plotting

Pandas give us basic plotting capability built right into DataFrames.

```
coffee_df['Country.of.Origin'].value_counts().plot()
```

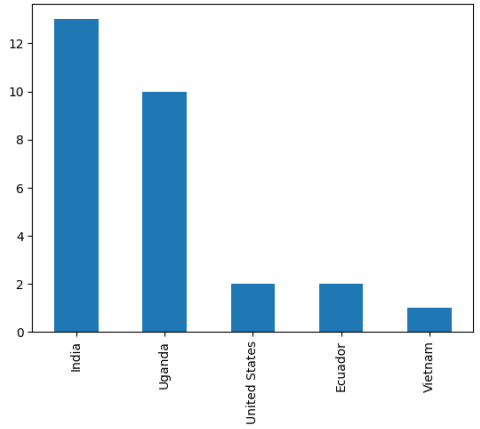
```
<AxesSubplot: >
```



It defaults to a line graph, which is not very informative in this case, so we can use the `kind` parameter to change it to a bar graph.

```
coffee_df['Country.of.Origin'].value_counts().plot(kind='bar')
```

```
<AxesSubplot: >
```



5.6. Matching Questions to Summary Statistics

When we brainstorm more questions that we can answer with summary statistics or that we might want to ask of this data, the ones that come to mind are often actually dependent on two variables. For example are two scores correlated? or what country has the highest rated coffe on `Flavor`?

We'll come back to more detailed questions like this on Friday, but to start we can look at how numerical variables vary by categorical (nonnumerical) variables by grouping the data.

Above we saw which country had the most ratings (remember one row is one rating), but what if we wanted to see the mean number of bags per country?

```
coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].mean()
```

```
Country.of.Origin
Ecuador            1.000000
India            230.076923
Uganda           160.900000
United States     50.500000
Vietnam            1.000000
Name: Number.of.Bags, dtype: float64
```

> **Important**
>
> This data is only about coffee that was rated by a particular agency it is not economic data, so we cannot, for example conclude which country *produces* the amount of data. If we had economic dataset, a `Number.of.Bags` columns's mean would tell us exactly that, but the context of the dataset defines what a row means and therefore how we can interpret the **every single statistic** we calculate.

5.7. Questions after class

5.7.1. Can arrays be used in Jupyter Notebook?

Jupyter runs a fully powered python interpreter, so all python can work inside it.

5.7.1.1. why did coffee_df['Country.of.Origin'].max() say vietnam?

That applies a the `max` function to the `'Country.of.Origin'` column, without counting how many times the values occur.

5.7.1.2. Why, in the groupby function the first column is in between parenthesis nd the second one between brackets?

The inside parenthesis is the parameter of the groupby function, we could instead do it this way:

```
coffee_grouped = coffee_df.groupby('Country.of.Origin')
```

Then we can use `coffee_grouped` for different things

```
coffee_grouped.describe()
```

	Number.of.Bags								Harvest.Year		...	altitude_high_meters		altitude_mean_meters			
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	mir
Country.of.Origin																	
Ecuador	2.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1.0	2.0	2016.000000	...	40.00	40.0	1.0	40.000000	NaN	4
India	13.0	230.076923	110.280296	1.0	140.00	300.0	320.00	320.0	13.0	2014.384615	...	1500.00	3170.0	12.0	1421.666667	1021.022957	75
Uganda	10.0	160.900000	163.690392	1.0	2.25	160.0	320.00	320.0	10.0	2013.300000	...	1488.00	1745.0	10.0	1354.500000	220.042546	109
United States	2.0	50.500000	70.003571	1.0	25.75	50.5	75.25	100.0	2.0	2013.000000	...	2448.75	3000.0	2.0	1897.500000	1559.170453	79
Vietnam	1.0	1.000000	NaN	1.0	1.00	1.0	1.00	1.0	1.0	2013.000000	...	NaN	NaN	0.0	NaN	NaN	N

5 rows × 160 columns

or as we did before

```
coffee_grouped['Number.of.Bags'].mean()
```

```
Country.of.Origin
Ecuador            1.000000
India            230.076923
Uganda           160.900000
United States     50.500000
Vietnam            1.000000
Name: Number.of.Bags, dtype: float64
```

The second one is to index (see last week's notes for more on indexing) the DataFrame and pick out one column. It makes the DataFrame have fewer columns before applying mean to the whole object.

We could make a smaller DataFrame first, then group, then mean

```
coffee_country_bags_df= coffee_df[['Number.of.Bags','Country.of.Origin']]
coffee_country_bags_df.head()
```

	Number.of.Bags	Country.of.Origin
1	300	Uganda
2	320	India
3	300	India
4	320	Uganda
5	1	Uganda

There are two sets of square brackets because putting a comma it would try to index along rows with one and columns with the other, to select multiple in one dimension you need to pass a list.

```
type(['Number.of.Bags','Country.of.Origin'])
```

```
list
```

If you do not put square `[]` or curly `{}` brackets around items, Python implicitly treats them as if there are parenthesis`()`

```
type(('Number.of.Bags','Country.of.Origin'))
```

```
tuple
```

and tuples are treated separately .

```
coffee_country_bags_df.groupby('Country.of.Origin').mean()
```

	Number.of.Bags
Country.of.Origin	
Ecuador	1.000000
India	230.076923
Uganda	160.900000
United States	50.500000
Vietnam	1.000000

5.7.1.3. A quick recap of how .info() counts non-null and dtype for each column would be great. Thanks!

above

5.7.1.4. Can you plot a graph that uses two columns in a dataset?

Yes, we will see that on Wednesday.

5.7.1.5. Are there any other data types as important as dataframes in pandas?

DataFrames are the main type provided by pandas, there are also Series which is conceptually a single column, groupby objects which is basically a list of (DataFrame,label) tuples, and some indexing, but all of these exist to support creating and operating on DataFrames.

5.7.1.6. Will we be utilizing any other packages /libraries mentioned than the ones for visualizing data that we talked about today?

We will use pandas for basic plots and seaborn for more advanced plots. There are other libraries for visualization like plotly for interactive plots, but those are more advanced than we need or for specialized use cases. However, the way these type of libraries are designed, is that the special use case ones are for when the basic ones do not do what you need and they often have common patterns.

5.7.1.7. When displaying two categories like we did at the end could you measure a different thing for each category? Like count for one and mean for the other?

Yes! We will likely not do this in class, but it is a small extension from what we have covered and uses the pandas aggregrate method.

> **Hint**
>
> This is expected for visualize level 3

5.7.1.8. How in depth will we go with graphing data and looking at it systemically to find patterns?

At some level, that is what we sill do for the rest of the semester. We will not cover everything there is to know about graphing, there are whole courses on data visualization. Most of the pattern-finding we will do is machine learning.

5.7.2. Assignment

5.7.2.1. For assignment 2, how do I find appropriate datasets online?

I collated the ones that I recommend on the datasets page. Then find something that is of interest to you.

5.7.2.2. How do you import a .py file from GitHub? Is there a way to do that without downloading it locally?

It does need to be local in order to import it as a module.

Ideally, you will work with all of your assignment repos locally either following the instructions in the notes from last class

6. Visualization

```
import pandas as pd
```

6.1. A note on viewing output

When we create a variable and then put that on the last line of a cell, jupyter *displays* it.

```
name = 'sarah'
name
```

```
'sarah'
```

How it displays it depends on the type

```
type(name)
```

```
str
```

For a string, it uses `print`

```
print(name)
```

```
sarah
```

so this and the one above look the same. For objects that have a `_repr_html_` method, jupyter uses that, and then your brownser uses html to render the object in a more visually appealing way.

6.2. Review from Monday

We will load the robusta data briefly again.

```
robusta_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_data_cleaned.csv'
```

Is the robusta coffee's Mouthfeel or Aftertaste rated more consistently

```
robusta_df = pd.read_csv(robusta_data_url)
robusta_df.describe()
```

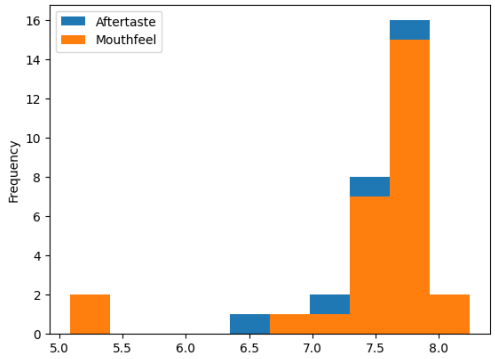
	Unnamed: 0	Number.of.Bags	Harvest.Year	Fragrance...Aroma	Flavor	Aftertaste	Salt...Acid	Bitter...Sweet	Mouthfeel	Uniform.Cup	...	Balance	Cupper.Poin
count	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	28.000000	...	28.000000	28.0000
mean	14.500000	168.000000	2013.964286	7.702500	7.630714	7.559643	7.657143	7.675714	7.506786	9.904286	...	7.541786	7.7614
std	8.225975	143.226317	1.346660	0.296156	0.303656	0.342469	0.261773	0.317063	0.725152	0.238753	...	0.526076	0.3305
min	1.000000	1.000000	2012.000000	6.750000	6.670000	6.500000	6.830000	6.670000	5.080000	9.330000	...	5.250000	6.9200
25%	7.750000	1.000000	2013.000000	7.580000	7.560000	7.397500	7.560000	7.580000	7.500000	10.000000	...	7.500000	7.5800
50%	14.500000	170.000000	2014.000000	7.670000	7.710000	7.670000	7.710000	7.750000	7.670000	10.000000	...	7.670000	7.8300
75%	21.250000	320.000000	2015.000000	7.920000	7.830000	7.770000	7.830000	7.830000	7.830000	10.000000	...	7.830000	7.9200
max	28.000000	320.000000	2017.000000	8.330000	8.080000	7.920000	8.000000	8.420000	8.250000	10.000000	...	8.000000	8.5800

8 rows × 21 columns

from the lower `std` we can see that Aftertaste is more consistently rated.

```
robusta_df[['Aftertaste','Mouthfeel']].plot(kind='hist')
```

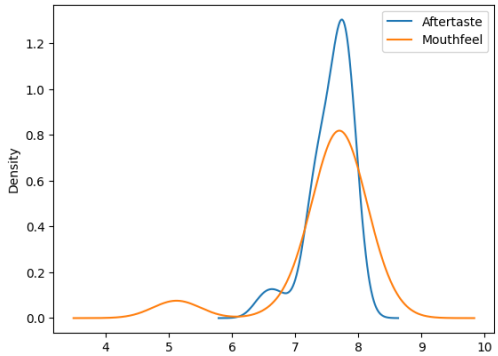
```
<AxesSubplot: ylabel='Frequency'>
```



We can change the kind, for example to a Kernel Density Estimate. This approximates the distribution of the data, you can think of it rougly like a smoothed out histogram.

```
robusta_df[['Aftertaste','Mouthfeel']].plot(kind='kde')
```

```
<AxesSubplot: ylabel='Density'>
```



This version makess it more visually clear that the the Aftertaste is more consistently, but it also helps us see that that might not be the whole story. Both have a second smaller bump, so the overall std might not be the best measure.

> **Question from class**
>
> Why do we need two sets of brackets?

It tries to use them to index in multiple ways instead.

```
robusta_df['Aftertaste','Mouthfeel']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/indexes/base.py:3803, in Index.get_loc(self, key, method,
tolerance)
   3802 try:
-> 3803     return self._engine.get_loc(casted_key)
   3804 except KeyError as err:

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/_libs/index.pyx:138, in pandas._libs.index.IndexEngine.get_loc()

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/_libs/index.pyx:165, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:5745, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:5753, in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('Aftertaste', 'Mouthfeel')

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

KeyError                                  Traceback (most recent call last)
Cell In [9], line 1
----> 1 robusta_df['Aftertaste','Mouthfeel']

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:3804, in DataFrame.__getitem__(self, key)
   3802 if self.columns.nlevels > 1:
   3803     return self._getitem_multilevel(key)
-> 3804 indexer = self.columns.get_loc(key)
   3805 if is_integer(indexer):
   3806     indexer = [indexer]

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/indexes/base.py:3805, in Index.get_loc(self, key, method,
tolerance)
   3803     return self._engine.get_loc(casted_key)
   3804 except KeyError as err:
-> 3805     raise KeyError(key) from err
   3806 except TypeError:
   3807     # If we have a listlike key, _check_indexing_error will raise
   3808     #  InvalidIndexError. Otherwise we fall through and re-raise
   3809     #  the TypeError.
   3810     self._check_indexing_error(key)

KeyError: ('Aftertaste', 'Mouthfeel')
```

It tries to look for a multiindex, but we do not have one so it fails. THe second square brackets, makes it a list of names to use and pandas looks for them sequentially.

6.3. Comparing two datasets

we're going to work with the arabica data today, because it's a little bigger and more interesting for plotting

```
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/arabica_data_cleaned.csv'
arabica_df = pd.read_csv(arabica_data_url,index_col=0)
```

It is mostly the same columns as the robusta data

```
arabica_df.columns
```

```
Index(['Species', 'Owner', 'Country.of.Origin', 'Farm.Name', 'Lot.Number',
       'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region', 'Producer',
       'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner', 'Harvest.Year',
       'Grading.Date', 'Owner.1', 'Variety', 'Processing.Method', 'Aroma',
       'Flavor', 'Aftertaste', 'Acidity', 'Body', 'Balance', 'Uniformity',
       'Clean.Cup', 'Sweetness', 'Cupper.Points', 'Total.Cup.Points',
       'Moisture', 'Category.One.Defects', 'Quakers', 'Color',
       'Category.Two.Defects', 'Expiration', 'Certification.Body',
       'Certification.Address', 'Certification.Contact', 'unit_of_measurement',
       'altitude_low_meters', 'altitude_high_meters', 'altitude_mean_meters'],
      dtype='object')
```

```
robusta_df.columns
```

```
Index(['Unnamed: 0', 'Species', 'Owner', 'Country.of.Origin', 'Farm.Name',
       'Lot.Number', 'Mill', 'ICO.Number', 'Company', 'Altitude', 'Region',
       'Producer', 'Number.of.Bags', 'Bag.Weight', 'In.Country.Partner',
       'Harvest.Year', 'Grading.Date', 'Owner.1', 'Variety',
       'Processing.Method', 'Fragrance...Aroma', 'Flavor', 'Aftertaste',
       'Salt...Acid', 'Bitter...Sweet', 'Mouthfeel', 'Uniform.Cup',
       'Clean.Cup', 'Balance', 'Cupper.Points', 'Total.Cup.Points', 'Moisture',
       'Category.One.Defects', 'Quakers', 'Color', 'Category.Two.Defects',
       'Expiration', 'Certification.Body', 'Certification.Address',
       'Certification.Contact', 'unit_of_measurement', 'altitude_low_meters',
       'altitude_high_meters', 'altitude_mean_meters'],
      dtype='object')
```

but it has a lot more rows

```
len(arabica_df) - len(robusta_df)
```

```
1283
```

```
arabica_df.shape, robusta_df.shape
```

```
((1311, 43), (28, 44))
```

6.4. Plotting in Python

- matplotlib: low level plotting tools
- seaborn: high level plotting with opinionated defaults
- ggplot: plotting based on the ggplot library in R.

Pandas and seaborn use matplotlib under the hood.

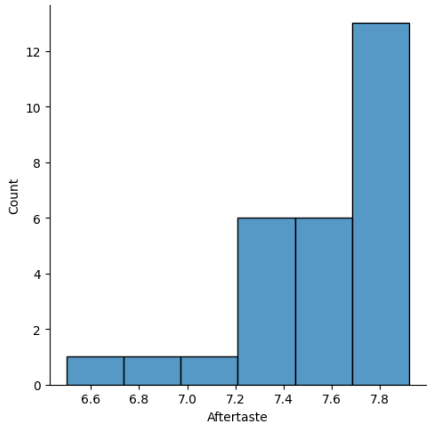
Seaborn and ggplot both assume the data is set up as a DataFrame. Getting started with seaborn is the simplest, so we'll use that.

> **Think Ahead**
>
> Learning ggplot is a way to earn level 3 for visualize

```
import seaborn as sns
```

```
sns.displot(data = robusta_df, x='Aftertaste')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f31576119d0>
```




```
sns.displot(data = robusta_df, x='Moutfeel')
```

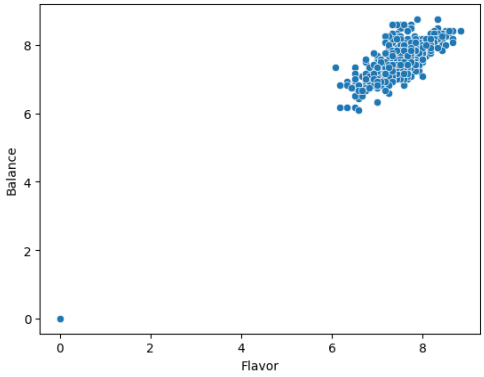
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [17], line 1
----> 1 sns.displot(data = robusta_df, x='Moutfeel')

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/distributions.py:2125, in displot(data, x, y, hue, row, col,
weights, kind, rug, rug_kws, log_scale, legend, palette, hue_order, hue_norm,
color, col_wrap, row_order, col_order, height, aspect, facet_kws, **kwargs)
   2111 def displot(
   2112     data=None, *,
   2113     # Vector variables
   (...)
   2122     **kwargs,
   2123 ):
-> 2125     p = _DistributionFacetPlotter(
   2126         data=data,
   2127         variables=_DistributionFacetPlotter.get_semantics(locals())
   2128     )
   2130     p.map_hue(palette=palette, order=hue_order, norm=hue_norm)
   2132     _check_argument("kind", ["hist", "kde", "ecdf"], kind)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/distributions.py:113, in _DistributionPlotter.__init__(self,
data, variables)
    107 def __init__(
    108     self,
    109     data=None,
    110     variables={},
    111 ):
--> 113     super().__init__(data=data, variables=variables)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:640, in VectorPlotter.__init__(self, data, variables)
    635 # var_ordered is relevant only for categorical axis variables, and may
    636 # be better handled by an internal axis information object that tracks
    637 # such information and is set up by the scale_* methods. The analogous
    638 # information for numeric axes would be information about log scales.
    639 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 640 self.assign_variables(data, variables)
    642 for var, cls in self._semantic_mappings.items():
    643
    644     # Create the mapping function
    645     map_func = partial(cls.map, plotter=self)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:701, in VectorPlotter.assign_variables(self, data,
variables)
    699 else:
    700     self.input_format = "long"
--> 701     plot_data, variables = self._assign_variables_longform(
    702         data, **variables,
    703     )
    705 self.plot_data = plot_data
    706 self.variables = variables

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:938, in
VectorPlotter._assign_variables_longform(self, data, **kwargs)
    933 elif isinstance(val, (str, bytes)):
    934
    935     # This looks like a column name but we don't know what it means!
    937     err = f"Could not interpret value `{val}` for parameter `{key}`"
--> 938     raise ValueError(err)
    940 else:
    941
    942     # Otherwise, assume the value is itself data
    943
    944     # Raise when data object is present and a vector can't matched
    945     if isinstance(data, pd.DataFrame) and not isinstance(val, pd.Series):

ValueError: Could not interpret value `Moutfeel` for parameter `x`
```

to plot these together with seaborn we have to transform the DataFrame, so we will see that next year.

6.4.1. how are flavor and balance related?

```
sns.scatterplot(data=arabica_df,x='Flavor',y='Balance')
```

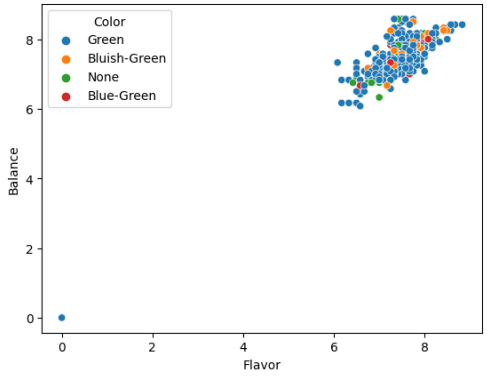
```
<AxesSubplot: xlabel='Flavor', ylabel='Balance'>
```



But now we have more power to investigate more relationships in the data.

```
sns.scatterplot(data=arabica_df,x='Flavor',y='Balance',hue='Color')
```

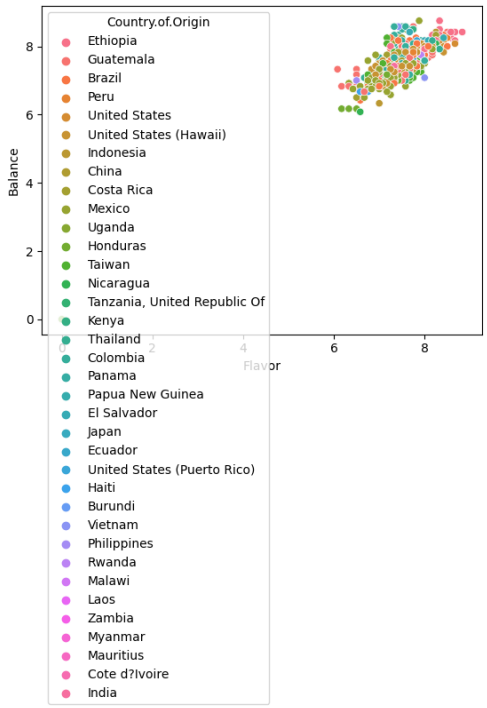
```
<AxesSubplot: xlabel='Flavor', ylabel='Balance'>
```



From this we can see that the color doesn't appear to be related to the flavor or balance scores, but that the flavor and balacne are related.

```
sns.scatterplot(data=arabica_df,x='Flavor',y='Balance',
                hue='Country.of.Origin')
```

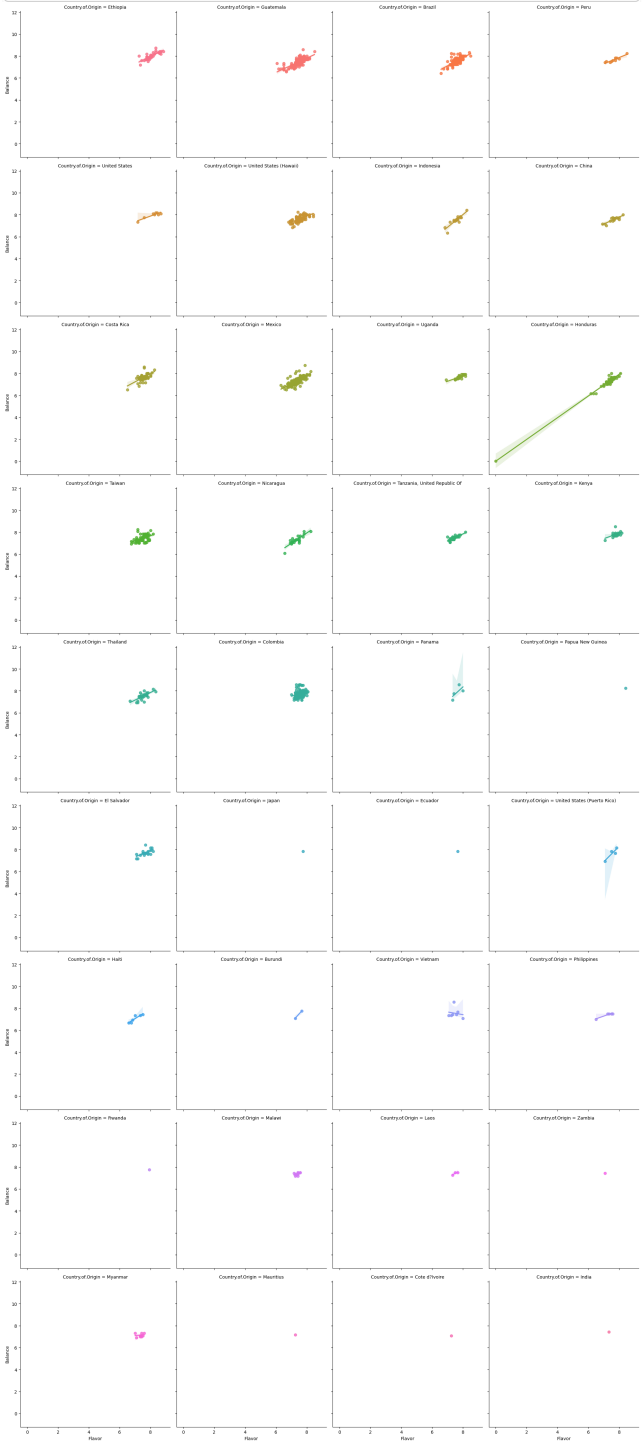
```
<AxesSubplot: xlabel='Flavor', ylabel='Balance'>
```



We can also break this apart. `lmplot` is a higher level plotting function so it allows us to create grids of plots and by default also includes a regression line.

```
sns.lmplot(data=arabica_df,x='Flavor',y='Balance',
                hue='Country.of.Origin',col='Country.of.Origin',
           col_wrap=4)
```

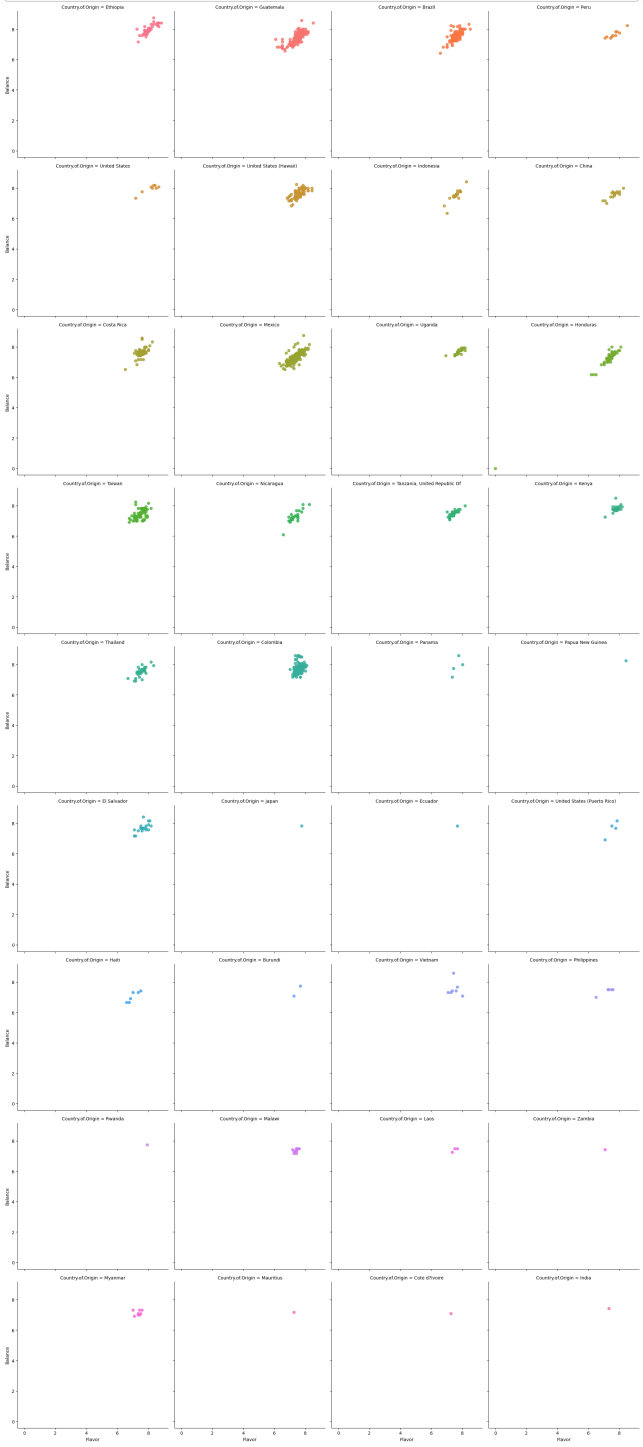
```
<seaborn.axisgrid.FacetGrid at 0x7f3156b76190>
```



If we were not interested in the regression line, we could turn that off for now, with `,fit_reg=False`.

```
sns.lmplot(data=arabica_df,x='Flavor',y='Balance',
                hue='Country.of.Origin',col='Country.of.Origin',
           col_wrap=4,fit_reg=False)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f314cb024f0>
```



6.5. Things you want to know more about

6.5.1. Things we will touch Friday

- more parameters and capabilities for seaborn
- how to analyze a graph

6.5.2. Things that you could study for level 3

- The little details about the plot functions
- More things that you can do with seaborn, like how to manipulate the scatterplots more
- More ways to create unique plots and charts (we'll give you a bit more depth but all the way here will be a level 3 task)
- One thing I want to learn based on what's learned so far is if we can compare graphs explicitly.

6.5.3. Things we will do later

6.5.3.1. Plotting data from multiple datasets on the same graphs (considering they share the same data points)

This will require being able to combine datasets into a single DataFrame, because a single plot funciton will require

6.5.4. Graphing lines of best fit onto graphs (but not just linear models, quadratic or other types as well)

We will learn how to fit models in general when we get to regression, but ultimately quadratic and other polynomial plots will also be a level 3 exploration you can do. We will get you to the point in class of all the pieces, but you will have to swap in some alternative parametrs

6.5.5. How to remove outliers or other data.

When we saw indexing we got really close to this, and we will do more when we clean data next week.

7. More EDA

```
import pandas as pd
import seaborn as sns
sns.set_theme(palette='colorblind')
```

```
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/arabica_data_cleaned.csv'
robusta_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/robusta_data_cleaned.csv'
```

```
coffee_df = pd.read_csv(arabica_data_url,index_col=0)
```

7.1. Which of the following is distributed most similarly to Sweetness?

```
scores_of_interest = ['Flavor','Balance','Aroma','Body',
                      'Uniformity','Aftertaste','Sweetness']
```

We can answer this statistically, technically, but it will be easiest to do this looking ad a plot.

First step is to subset the data:

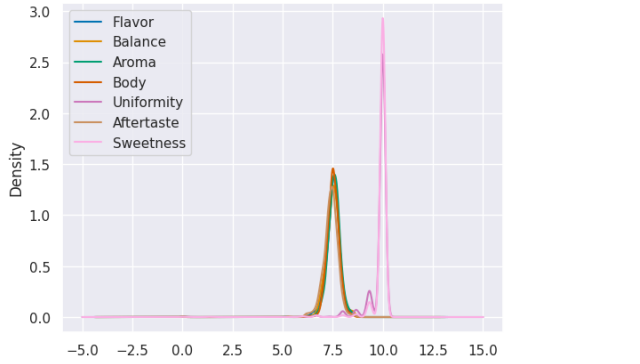
```
coffee_df[scores_of_interest].head(2)
```

	Flavor	Balance	Aroma	Body	Uniformity	Aftertaste	Sweetness
1	8.83	8.42	8.67	8.50	10.0	8.67	10.0
2	8.67	8.42	8.75	8.42	10.0	8.50	10.0

Then we produce a kde plot

```
coffee_df[scores_of_interest].plot(kind='kde')
```

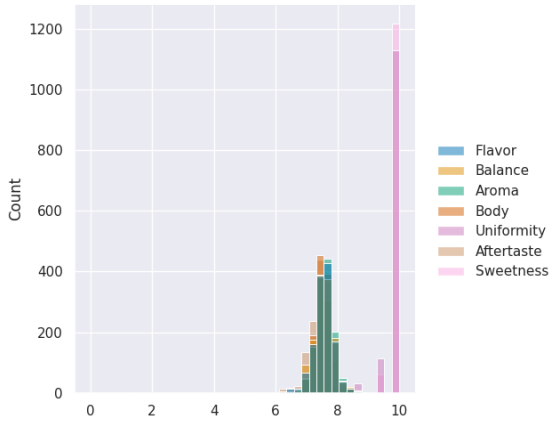
```
<AxesSubplot: ylabel='Density'>
```



We could also do it with seaborn

```
sns.displot(data=coffee_df[scores_of_interest])
```

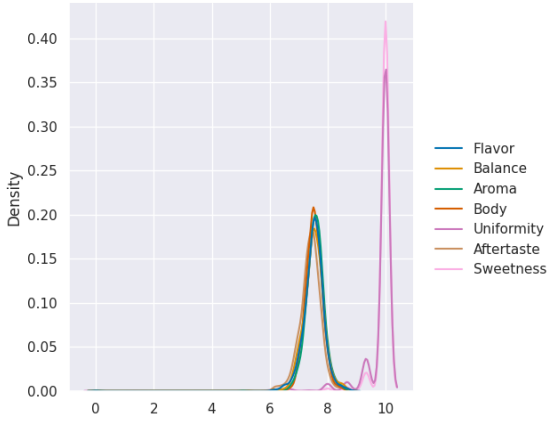
```
<seaborn.axisgrid.FacetGrid at 0x7f0b75959b80>
```



the default is not as helpful as using

```
sns.displot(data=coffee_df[scores_of_interest],kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0b7556fd30>
```



If we forget the parameter `kind`, we get its default value.

```
print('\n'.join(sns.displot.__doc__.split('\n')[5:10]))
```

```
``kind`` parameter selects the approach to use:

- :func:`histplot` (with ``kind="hist"``; the default)
- :func:`kdeplot` (with ``kind="kde"``)
- :func:`ecdfplot` (with ``kind="ecdf"``; univariate-only)
```

> **Note**
>
> If you show this excerpt, you'll see how I was able to select only a subset of the docstring to display in the notebook, programmatically. You're not required to know how to do it, but if you're curious, you can see.

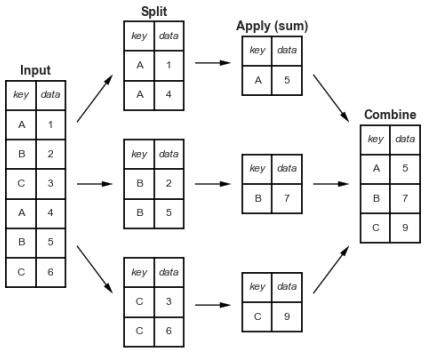
7.2. Summarizing with multiple variables

So, we can summarize data now, but the summaries we have done so far have treated each variable one at a time. The most interesting patterns are in often in how multiple variables interact. We'll do some modeling that looks at multivariate functions of data in a few weeks, but for now, we do a little more with summary statistics.

On Monday, we saw how to see how many reviews there were per country, using

```
total_per_country = coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()
```

What just happened?



Groupby splits the whole dataframe into parts where each part has the same value for `Country.of.Origin` and then after that, we extracted the `Number.of.Bags` column, took the sum (within each separate group) and then put it all back together in one table (in this case, a `Series` becuase we picked one variable out)

7.2.1. How does Groupby Work?

> **Important**
>
> This is more details with code examples on how the groupby works. If you want to run this code for yourself, use the download icon at the top right to download these notes as a notebook.

We can view this by saving the groupby object as a variable and exploring it.

```
country_grouped = coffee_df.groupby('Country.of.Origin')

country_grouped
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f0b79c81f10>
```

Trying to look at it without applying additional functions, just tells us the type. But, it's iterable, so we can loop over.

```
for country,df in country_grouped:
    print(type(country), type(df))
```

```
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
<class 'str'> <class 'pandas.core.frame.DataFrame'>
```

We could manually compute things using the data structure, if needed, though using pandas functionality will usually do what we want. For example:

```
bag_total_dict = {}

for country,df in country_grouped:
    tot_bags =  df['Number.of.Bags'].sum()
    bag_total_dict[country] = tot_bags

pd.DataFrame.from_dict(bag_total_dict, orient='index',
                           columns = ['Number.of.Bags.Sum'])
```

Note

I used this feature to build the separate view of the communication channels on this website. You can view that source using the github icon on that page.

Note

I tried putting this dictionary into the dataframe for display purposes using the regular constructor and got an error, so I googled about making one from a dictionary to get the docs, which is how I learned about the `from_dict` method and its `orient` parameter which solved my problems.

	Number.of.Bags.Sum
Brazil	30534
Burundi	520
China	55
Colombia	41204
Costa Rica	10354
Cote d?Ivoire	2
Ecuador	1
El Salvador	4449
Ethiopia	11761
Guatemala	36868
Haiti	390
Honduras	13167
India	20
Indonesia	1658
Japan	20
Kenya	3971
Laos	81
Malawi	557
Mauritius	1
Mexico	24140
Myanmar	10
Nicaragua	6406
Panama	537
Papua New Guinea	7
Peru	2336
Philippines	259
Rwanda	150
Taiwan	1914
Tanzania, United Republic Of	3760
Thailand	1310
Uganda	3868
United States	361
United States (Hawaii)	833
United States (Puerto Rico)	71
Vietnam	10
Zambia	13

is the same as what we did before

7.3. Sorting DataFrames

We saved the totals, so we can check what type that is.

```
type(total_per_country)
```

```
pandas.core.series.Series
```

It's a pandas series, so we can use the `sort_values` method.

```
total_per_country.sort_values(ascending=False)
```

```
Country.of.Origin
Colombia                        41204
Guatemala                       36868
Brazil                          30534
Mexico                          24140
Honduras                        13167
Ethiopia                        11761
Costa Rica                      10354
Nicaragua                        6406
El Salvador                      4449
Kenya                            3971
Uganda                           3868
Tanzania, United Republic Of     3760
Peru                             2336
Taiwan                           1914
Indonesia                        1658
Thailand                         1310
United States (Hawaii)            833
Malawi                            557
Panama                            537
Burundi                           520
Haiti                             390
United States                     361
Philippines                       259
Rwanda                            150
Laos                               81
United States (Puerto Rico)        71
China                              55
India                              20
Japan                              20
Zambia                             13
Myanmar                            10
Vietnam                            10
Papua New Guinea                    7
Cote d?Ivoire                       2
Ecuador                             1
Mauritius                           1
Name: Number.of.Bags, dtype: int64
```

Try it yourself

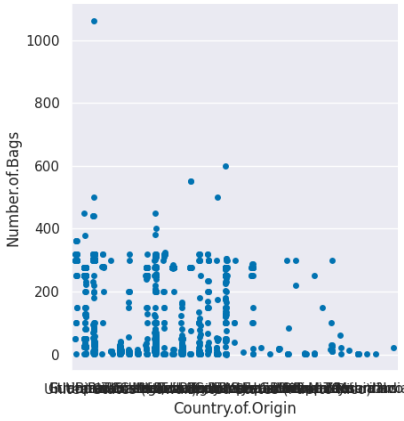
There is another sort method, `sort_index` what would that one do?

7.4. More plot types

We can also look at the distributions

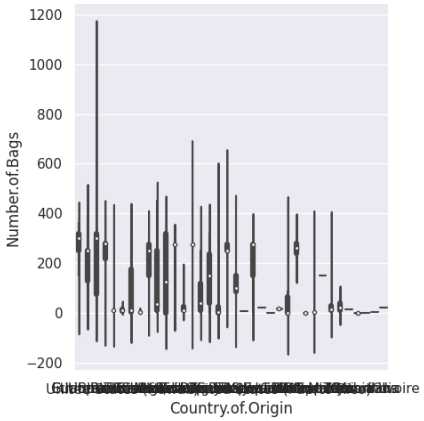
```
sns.catplot(data=coffee_df,x='Country.of.Origin',y='Number.of.Bags')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0b7584ee80>
```



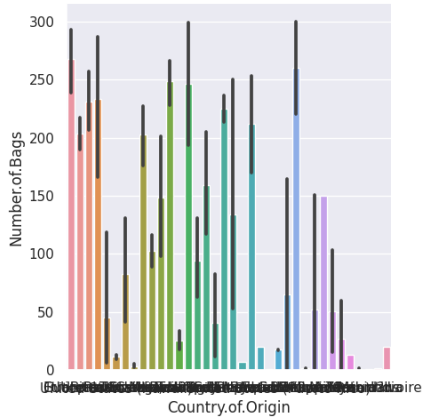

```
sns.catplot(data=coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='violin')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0b75342ac0>
```



```
sns.catplot(data=coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0b75368820>
```



7.5. How can we pick the top 10 countries out?

> **Note**
>
> We can use `head(10)` here too.

```
total_per_country.sort_values(ascending=False)[:10]
```

```
Country.of.Origin
Colombia       41204
Guatemala      36868
Brazil         30534
Mexico         24140
Honduras       13167
Ethiopia       11761
Costa Rica     10354
Nicaragua       6406
El Salvador     4449
Kenya           3971
Name: Number.of.Bags, dtype: int64
```

```
type(total_per_country.sort_values(ascending=False)[:10])
```

```
pandas.core.series.Series
```

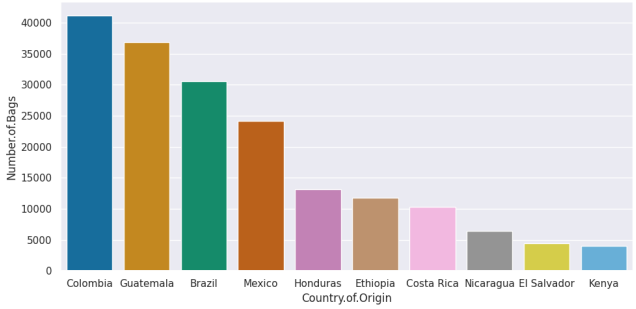
this is a series, but we can change it to a DataFrame with `reset_index()` this will also add a new index column but a side effect is that it becomes a DataFrame again

```
top_total_df = total_per_country.sort_values(ascending=False)[:10].reset_index()
top_total_df
```

	Country.of.Origin	Number.of.Bags
0	Colombia	41204
1	Guatemala	36868
2	Brazil	30534
3	Mexico	24140
4	Honduras	13167
5	Ethiopia	11761
6	Costa Rica	10354
7	Nicaragua	6406
8	El Salvador	4449
9	Kenya	3971

```
sns.catplot(data=top_total_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar',aspect=2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0b7584e880>
```



this got the number of countries fewer, bu they were still hard to read because they were small and still overlapping, so I added `aspect=2` to make it 2x as wide as tall and give it more space. We can also use a similar strategy to pull out just these country names and get all the data for those.

```
top_countries = total_per_country.sort_values(ascending=False)[:10].index
```

```
coffee_df[coffee_df['Country.of.Origin'].isin(top_countries)].head(2)
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	Region	...	Color	Category.Two.Defects	Expiration	Certifica
1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	guji-hambela	...	Green	0	April 3rd, 2016	Develo
2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	guji-hambela	...	Green	1	April 3rd, 2016	Develo

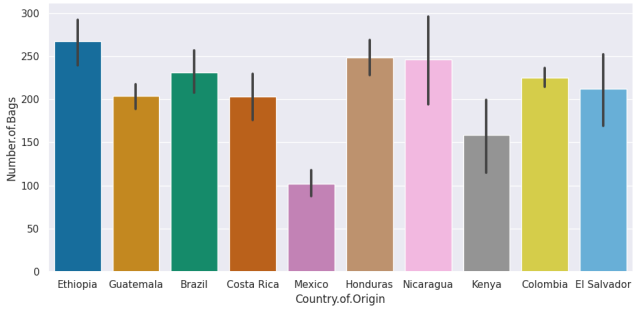
2 rows × 43 columns

I forgot the `isin` method in class and had to look that up, I more often need to filter by numerical columns or selecting a single value.

```
top_coffee_df = coffee_df[coffee_df['Country.of.Origin'].isin(top_countries)]
```

```
sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar',aspect=2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0b752e9be0>
```




```
top_coffee_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 952 entries, 1 to 1312
Data columns (total 43 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Species                952 non-null    object
 1   Owner                  945 non-null    object
 2   Country.of.Origin      952 non-null    object
 3   Farm.Name              694 non-null    object
 4   Lot.Number             205 non-null    object
 5   Mill                   760 non-null    object
 6   ICO.Number             908 non-null    object
 7   Company                789 non-null    object
 8   Altitude               833 non-null    object
 9   Region                 919 non-null    object
 10  Producer               812 non-null    object
 11  Number.of.Bags         952 non-null    int64
 12  Bag.Weight             952 non-null    object
 13  In.Country.Partner     952 non-null    object
 14  Harvest.Year           932 non-null    object
 15  Grading.Date           952 non-null    object
 16  Owner.1                945 non-null    object
 17  Variety                824 non-null    object
 18  Processing.Method      850 non-null    object
 19  Aroma                  952 non-null    float64
 20  Flavor                 952 non-null    float64
 21  Aftertaste             952 non-null    float64
 22  Acidity                952 non-null    float64
 23  Body                   952 non-null    float64
 24  Balance                952 non-null    float64
 25  Uniformity             952 non-null    float64
 26  Clean.Cup              952 non-null    float64
 27  Sweetness              952 non-null    float64
 28  Cupper.Points          952 non-null    float64
 29  Total.Cup.Points       952 non-null    float64
 30  Moisture               952 non-null    float64
 31  Category.One.Defects   952 non-null    int64
 32  Quakers                951 non-null    float64
 33  Color                  808 non-null    object
 34  Category.Two.Defects   952 non-null    int64
 35  Expiration             952 non-null    object
 36  Certification.Body     952 non-null    object
 37  Certification.Address  952 non-null    object
 38  Certification.Contact  952 non-null    object
 39  unit_of_measurement    952 non-null    object
 40  altitude_low_meters    830 non-null    float64
 41  altitude_high_meters   830 non-null    float64
 42  altitude_mean_meters   830 non-null    float64
dtypes: float64(16), int64(3), object(24)
memory usage: 327.2+ KB
```

```
sns.displot(top_coffee_df, x='Sweetness', kind='kde',
col='Country.of.Origin',col_wrap=3)
```

We'll see next week how to manipulate the dataset so that we can plot multiple scores this way.

Variable types and data types Related but not the same.

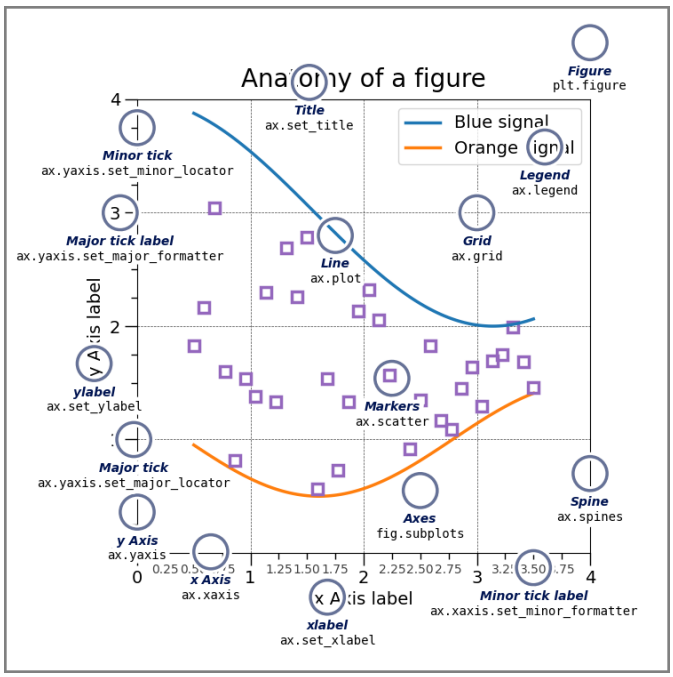
7.6. General Plotting Ideas

There are lots of type of plots, we saw the basic patterns of how to use them and we've used a few types, but we cannot (and do not need to) go through every single type. There are general patterns that you can use that will help you think about what type of plot you might want and help you understand them to be able to customize plots.

[Seaborn's main goal is opinionated defaults and flexible customization](https://seaborn.pydata.org/tutorial/introduction.html#opinionated-defaults-and-flexible-customization

7.6.1. Anatomy of a figure

First is the matplotlib structure of a figure. BOth pandas and seaborn and other plotting libraries use matplotlib. Matplotlib was used in visualizing the first Black hole.



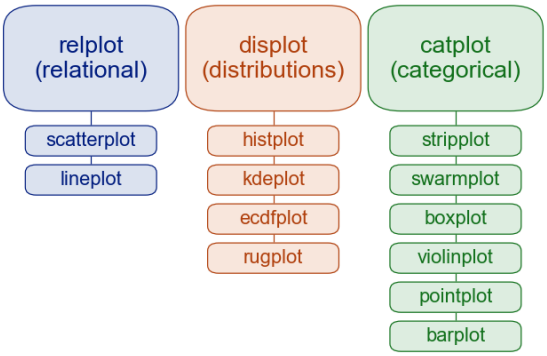
This is a lot of information, but these are good to know things. THe most important is the figure and the axes.

> **Try it Yourself**
>
> Make sure you can explain what is a figure and what are axes in your own words and why that distinction matters. Discuss in office hours if you are unsure.

that image was drawn with code and that page explains more.

7.6.2. Plotting Function types in Seaborn

Seaborn has two *levels* or groups of plotting functions. Figure and axes. Figure level fucntions can plot with subplots.



This is from thie overivew section of the official seaborn tutorial. It also includes a comparison of figure vs axes plotting.

The official introduction is also a good read.

7.6.3. More

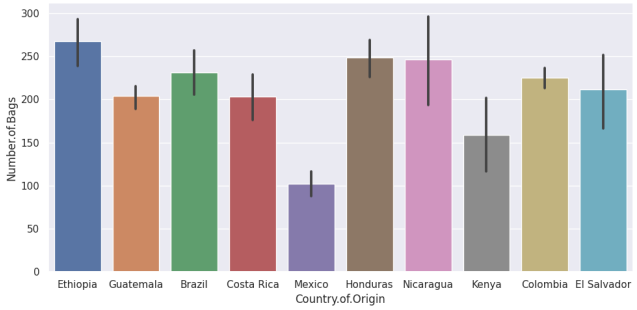
The seaborn gallery and matplotlib gallery are nice to look at too.

7.7. Styling Plots

```
sns.set_theme()
```

```
sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar',aspect=2)
```

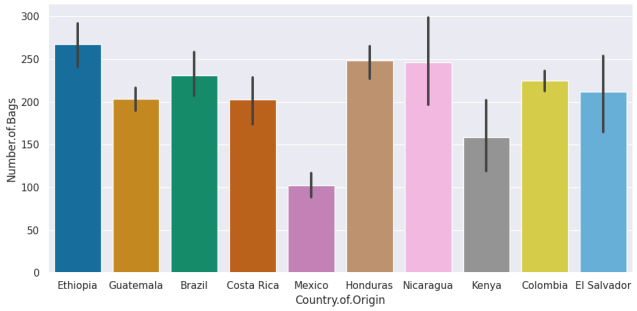
```
<seaborn.axisgrid.FacetGrid at 0x7f0b73c1ef70>
```



This by default styles the plots to be more visually appealing

```
sns.set_theme(palette='colorblind')
sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar',aspect=2)
```

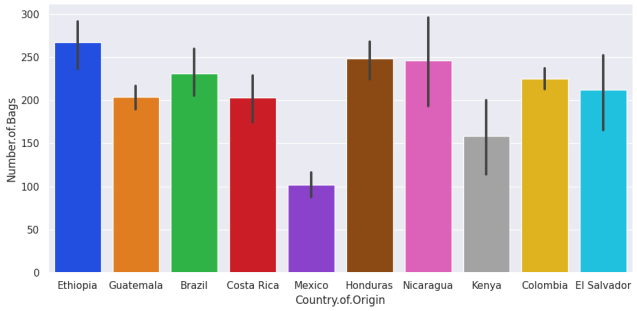
```
<seaborn.axisgrid.FacetGrid at 0x7f0b70044f10>
```



the colorblind palette is more distinguishable under a variety fo colorblindness types. for more

```
sns.set_theme(palette='bright')
sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
            kind='bar',aspect=2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0b6bf46430>
```

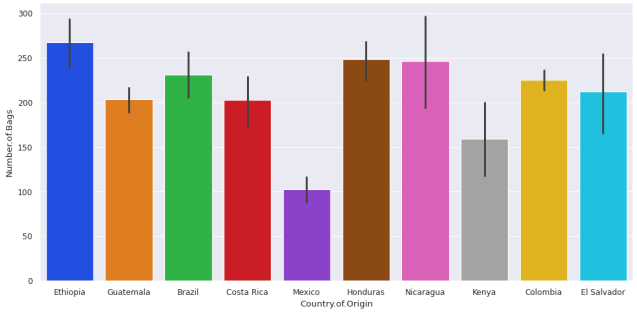


Colorblind is a good default, but you can choose others that yo like more too.

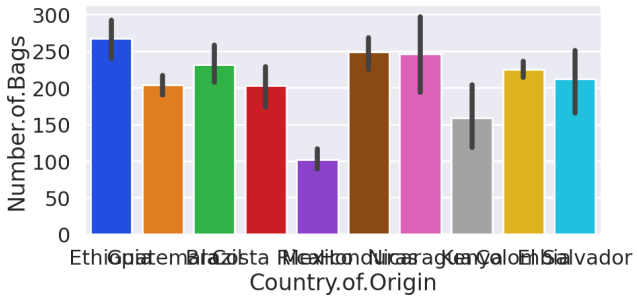
[more on colors]

to prepare plots for posters vs printing, etc you can use:

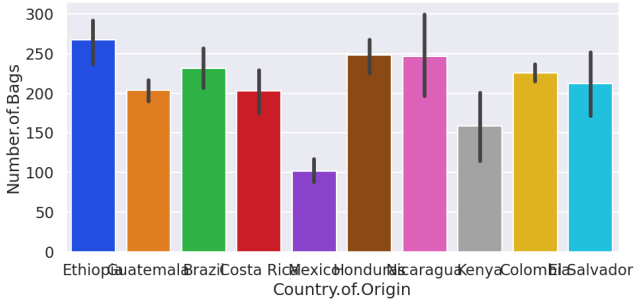
```
with sns.plotting_context('paper'):
    sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
           kind='bar',aspect =2)
```



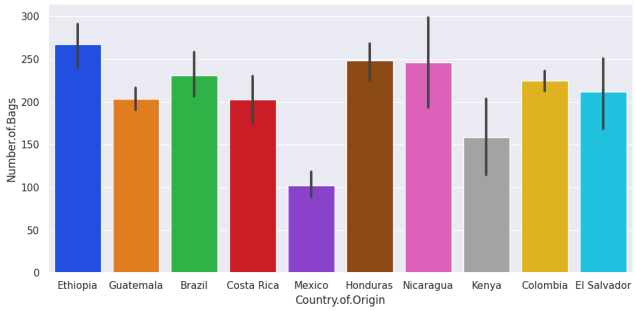

```
with sns.plotting_context('poster'):
    sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
           kind='bar',aspect =2)
```




```
with sns.plotting_context('talk'):
    sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
           kind='bar',aspect =2)
```




```
with sns.plotting_context('notebook'):
    sns.catplot(data=top_coffee_df,x='Country.of.Origin',y='Number.of.Bags',
           kind='bar',aspect =2)
```



It's not perfect, but it gives you a good starting point.

7.8. More Practice

- Make a table thats total number of bags and mean and count of scored for each of the variables in the `scores_of_interest` list.
- Make a bar chart of the mean score for each variable `scores_of_interest` grouped by country.

7.9. Questions

> **Note**
>
> Submit questions as Issues

7.10. Questions After Class

8. Intro to Data Cleaning

This week, we'll be cleaning data.

Cleaning data is **labor intensive** and requires making *subjective* choices.
We'll focus on, and assess you on, manipulating data correctly, making reasonable choices, and documenting the choices you make carefully.

We'll focus on the programming tools that get used in cleaning data in class this week:

- reshaping data
- handling missing or incorrect values
- renaming columns

```
import pandas as pd
import seaborn as sns

# sns.set_theme(pallete='colorblind')
```

8.1. Tidy Data

Read in the three csv files described below and store them in a list of dataFrames

```
url_base = 'https://raw.githubusercontent.com/rhodyprog4ds/rhodyds/main/data/'

datasets = ['study_a.csv','study_b.csv','study_c.csv']
```

```
df_list = [pd.read_csv(url_base + current) for current in datasets]
```

```
type(df_list)
```

```
list
```

```
type(df_list[0])
```

```
pandas.core.frame.DataFrame
```

```
df_list[0]
```

	name	treatmenta	treatmentb
0	John Smith	-	2
1	Jane Doe	16	11
2	Mary Johnson	3	1

```
df_list[1]
```

	intervention	John Smith	Jane Doe	Mary Johnson
0	treatmenta	-	16	3
1	treatmentb	2	11	1

```
df_list[2]
```

	person	treatment	result
0	John Smith	a	-
1	Jane Doe	a	16
2	Mary Johnson	a	3
3	John Smith	b	2
4	Jane Doe	b	11
5	Mary Johnson	b	1

These three all show the same data, but let's say we have two goals:

- find the average effect per person across treatments
- find the average effect per treatment across people

This works differently for these three versions.

```
df_list[0].mean()
```

```
/tmp/ipykernel_1934/2274987639.py:1: FutureWarning: The default value of
numeric_only in DataFrame.mean is deprecated. In a future version, it will default
to False. In addition, specifying 'numeric_only=None' is deprecated. Select only
valid columns or specify the value of numeric_only to silence this warning.
  df_list[0].mean()
```

```
treatmenta   -54.333333
treatmentb     4.666667
dtype: float64
```

we get the average per treatment, but to get the average per person, we have to go across rows, which we can do here, but doesn't work as well with plotting

```
df_list[0].mean(axis=1)
```

```
/tmp/ipykernel_1934/1371725361.py:1: FutureWarning: Dropping of nuisance columns
in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future
version this will raise TypeError.  Select only valid columns before calling the
reduction.
  df_list[0].mean(axis=1)
```

```
0     2.0
1    11.0
2     1.0
dtype: float64
```

and this is not well labeled.

Let's try the next one.

```
df_list[1].mean()
```

```
/tmp/ipykernel_1934/1105758803.py:1: FutureWarning: The default value of
numeric_only in DataFrame.mean is deprecated. In a future version, it will default
to False. In addition, specifying 'numeric_only=None' is deprecated. Select only
valid columns or specify the value of numeric_only to silence this warning.
  df_list[1].mean()
```

```
John Smith      -1.0
Jane Doe        13.5
Mary Johnson     2.0
dtype: float64
```

Now we get the average per person, but what about per treatment? again we have to go across rows instead.

```
df_list[1].mean(axis=1)
```

```
/tmp/ipykernel_1934/1112167831.py:1: FutureWarning: Dropping of nuisance columns
in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future
version this will raise TypeError.  Select only valid columns before calling the
reduction.
  df_list[1].mean(axis=1)
```

```
0    9.5
1    6.0
dtype: float64
```

For the third one, however, we can use groupby

```
df_list[2].groupby('person').mean()
```

```
/tmp/ipykernel_1934/906930014.py:1: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  df_list[2].groupby('person').mean()
```

	result
person	
Jane Doe	805.5
John Smith	-1.0
Mary Johnson	15.5

```
df_list[2].groupby('treatment').mean()
```

```
/tmp/ipykernel_1934/2480310521.py:1: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  df_list[2].groupby('treatment').mean()
```

treatment	result
a	-54.333333
b	703.666667

The original Tidy Data paper is worth reading to build a deeper understanding of these ideas.

8.2. Tidying Data

Let's reshape the first one to match the tidy one. First, we will save it to a DataFrame, this makes things easier to read and enables us to use the built in help in jupyter, because it can't check types too many levels into a data structure.

```
treat_df = df_list[0]
```

Let's look at it again, so we can see

```
treat_df.head()
```

	name	treatmenta	treatmentb
0	John Smith	-	2
1	Jane Doe	16	11
2	Mary Johnson	3	1

```
df_list[0].columns
```

```
Index(['name', 'treatmenta', 'treatmentb'], dtype='object')
```

```
df_a = df_list[0]
```

```
df_a.melt(id_vars=['name'],value_vars=['treatmenta','treatmentb'],
          value_name='result',var_name='treatment')
```

	name	treatment	result
0	John Smith	treatmenta	-
1	Jane Doe	treatmenta	16
2	Mary Johnson	treatmenta	3
3	John Smith	treatmentb	2
4	Jane Doe	treatmentb	11
5	Mary Johnson	treatmentb	1

When we melt a dataset:

- the `id_vars` stay as columns
- the data from the `value_vars` columns become the values in the `value` column
- the column names from the `value_vars` become the values in the `variable` column
- we can rename the value and the variable columns.

Let's do it for our coffee data:

```
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-
database/master/data/arabica_data_cleaned.csv'
```

```
coffee_df = pd.read_csv(arabica_data_url)
scores_of_interest = ['Balance','Aroma','Body','Aftertaste']
```

```
coffee_tall = coffee_df.melt(id_vars=['Country.of.Origin','Color'],
                             value_vars=scores_of_interest
                ,var_name = 'Score')
coffee_tall.head()
```

	Country.of.Origin	Color	Score	value
0	Ethiopia	Green	Balance	8.42
1	Ethiopia	Green	Balance	8.42
2	Guatemala	NaN	Balance	8.42
3	Ethiopia	Green	Balance	8.25
4	Ethiopia	Green	Balance	8.33

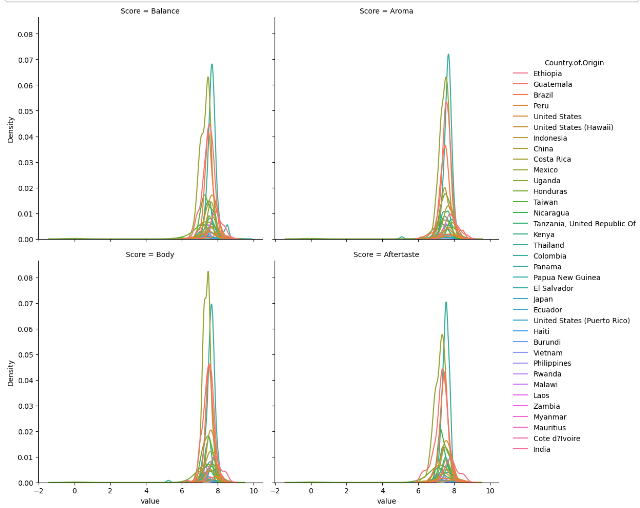
Notice that the actual column names inside the `scores_of_interest` variable become the values in the variable column.

This one we can plot in more ways:

```
sns.displot(data=coffee_tall, x='value',hue='Country.of.Origin',
            col='Score',kind='kde',col_wrap=2)
```

```
/tmp/ipykernel_1934/3603277008.py:1: UserWarning: Dataset has 0 variance; skipping
density estimate. Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffee_tall, x='value',hue='Country.of.Origin',
/tmp/ipykernel_1934/3603277008.py:1: UserWarning: Dataset has 0 variance; skipping
density estimate. Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffee_tall, x='value',hue='Country.of.Origin',
/tmp/ipykernel_1934/3603277008.py:1: UserWarning: Dataset has 0 variance; skipping
density estimate. Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffee_tall, x='value',hue='Country.of.Origin',
/tmp/ipykernel_1934/3603277008.py:1: UserWarning: Dataset has 0 variance; skipping
density estimate. Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffee_tall, x='value',hue='Country.of.Origin',
/tmp/ipykernel_1934/3603277008.py:1: UserWarning: Dataset has 0 variance; skipping
density estimate. Pass `warn_singular=False` to disable this warning.
  sns.displot(data=coffee_tall, x='value',hue='Country.of.Origin',
```

```
<seaborn.axisgrid.FacetGrid at 0x7f2fb7b89c40>
```



8.3. Filtering Data

```
high_prod = coffee_df[coffee_df['Number.of.Bags']>250]
high_prod.shape
```

```
(368, 44)
```

```
coffee_df.shape
```

```
(1311, 44)
```

We see that filters and reduces. We can use any boolean expression in the square brackets.

```
top_balance = coffee_df[coffee_df['Balance']>coffee_df['Balance'].quantile(.75)]
top_balance.shape
```

```
(252, 44)
```

We can confirm that we got only the top 25% of balance scores:

```
top_balance.describe()
```

	Unnamed: 0	Number.of.Bags	Aroma	Flavor	Aftertaste	Acidity	Body	Balance	Uniformity	Clean.Cup	Sweetness	Cupper.Points	Total
count	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	252.000000	
mean	273.535714	153.337302	7.808889	7.837659	7.734167	7.824881	7.780159	8.003095	9.885278	9.896667	9.885952	7.865714	
std	284.249040	126.498576	0.355319	0.318172	0.302481	0.320253	0.317712	0.213056	0.363303	0.470514	0.380433	0.383738	
min	1.000000	0.000000	5.080000	7.170000	6.920000	7.080000	5.250000	7.830000	6.670000	5.330000	6.670000	5.170000	
25%	65.750000	12.750000	7.647500	7.670000	7.500000	7.670000	7.670000	7.830000	10.000000	10.000000	10.000000	7.670000	
50%	171.500000	165.500000	7.780000	7.830000	7.750000	7.830000	7.750000	7.920000	10.000000	10.000000	10.000000	7.830000	
75%	378.250000	275.000000	8.000000	8.000000	7.920000	8.000000	7.920000	8.080000	10.000000	10.000000	10.000000	8.080000	
max	1260.000000	360.000000	8.750000	8.830000	8.670000	8.750000	8.580000	8.750000	10.000000	10.000000	10.000000	9.250000	

We can also use the `isin` method to filter by comparing to an iterable type

```
total_per_country = coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()
top_countries = total_per_country.sort_values(ascending=False)[:10].index
top_coffee_df = coffee_df[coffee_df['Country.of.Origin'].isin(top_countries)]
```

8.4. Questions after class

8.4.1. In the data we had about treatment a and treatment b. Say there was another column that provided information about the age of the people. Could we create another value variable to or would we put it in the value_vars list along with treatment a and treatmentb?

We can have multiple variables as the `id_vars` so that the dataset would have 4 columns instead of 3.

8.4.2. Can we clean data within any row of the data as long as there is a column name for it?

Yes

8.4.3. How to clean larger datasets

Everything we will learn will work in any context that you can load the dataset into RAM on the computer you are working on; or process it in batches.

8.4.4. with very large data sets, how can you tell id there are missing values or incorrect types present?

We use ways of checking the values, like the `info` method to check the whole column.

8.4.5. Are there any prebuilt methods to identify and remove extreme outliers?

There may be, that is a good thing to look in the documentation for. However, typically the definition of an outlier is best made within context, so the filtering strategy that we just used would be the way to remove them, after doign some EDA.

8.4.6. Is there a specific metric to which a dataset must meet to consider it 'cleaned'?

It's "clean" when it will work for the analysis you want to do. This means clean enough for one analysis may not be enough for another goal. Domain expertise is alwasy important.

8.4.7. Are there any packages we may utilize that help us clean data more effectively?

8.4.8. Things we will do later this week:

- adding more data to a DataFrame in jupyter notebook?
- replace values
- deal with NaN values in a dataset?

8.4.9. Next week

- What is the difference between different types of merging data frames (inner, outer, left, right, etc)?

9. Fixing Data representations

9.1. Admin

- next assignment posted tonight.

9.1.1. a4 TLDR:

- Review how your new dataset manipulation skills could have helped you in A2 or A3.
- clean provided (in the template) datasets
  - do tiny EDA to show complete;
  - add more EDA to get summarize and visualize
  - clean a specific dataset for access level 2 or python level 2 also; do both if you need all 3 achievments (prepare, access, python)
- Study some examples and notice patterns in data clearning

9.1.2. Portfolio update

- merge in your work from assignment 1
- portoflio will be due in ~2 weeks start planning
- read the instructions and example ideas; there will be time for q&a on Friday
- you can also create an outline and get feedback on your plan using an issue on your portflio repo

> **Note**
>
> The portfolio is open ended intentionally. I want you to learn these skills a little bit deeper than you have for the assignments, which is a little bit deeper than we cover in class and show me that you learned. I do *not* want you to spend too much effort guessing what I want. I will look at what you submit for evidence of learning and assess that. There is not a single "right" answer.

9.2. Review Filtering

```
import pandas as pd
```

We can also filter using the `isin` method to compare each item in a column to a list

```
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-
database/master/data/arabica_data_cleaned.csv'
# load the data
coffee_df = pd.read_csv(arabica_data_url)
# get total bags per country
bags_per_country_df = coffee_df.groupby('Country.of.Origin')
['Number.of.Bags'].sum()

# sort descending, keep only the top 10 and pick out only the country names
top_bags_country_list = bags_per_country_df.sort_values(ascending=False)
[:10].index

# filter the original data for only the countries in the top list
top_coffee_df =
coffee_df[coffee_df['Country.of.Origin'].isin(top_bags_country_list)]
```

> **Note**
>
> You can see more about groupby in notes from last week.

Yes this is a Series, not a data Frame, but a DataFrame is built of Series, so this is a small error in naming. `top_bags_country_list` is also an `Index` not a `list` but the names need to give an idea, not necessarily be precise so my choices are okay, but could be better.

```
type(coffee_df['Number.of.Bags'])
```

```
pandas.core.series.Series
```

```
type(coffee_df.loc[1])
```

```
pandas.core.series.Series
```

We can look at the final result

```
top_coffee_df.head()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Color	Category.Two.Defects	Expiratio
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	0	April 3r 201
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	1	April 3r 201
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	NaN	NaN	1600 - 1800 m	...	NaN	0	May 31s 201
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	NaN	yidnekachew debessa coffee plantation	1800-2200	...	Green	2	Marc 25th, 201
4	5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	Green	2	April 3r 201

5 rows × 44 columns

```
top_coffee_df.shape, coffee_df.shape
```

```
((952, 44), (1311, 44))
```

9.3. Fixing a Column

In tidy data each column is exactly one variable. Let's look at the `Bag.Weight`

```
coffee_df['Bag.Weight'].sample(10)
```

```
257       60 kg
104     100 lbs
325       70 kg
1189       1 kg
3         60 kg
35         1 kg
1064      69 kg
270       70 kg
1274       1 kg
73        60 kg
Name: Bag.Weight, dtype: object
```

This is actually two pieces of information, the value measured and the units used. In addition to the fact that there are multiple units, even if we could convert them, we cannot do math on these because they're strings. (notice it says "object" as the type)

Series have a `str` attribute so we can apply base python string methods to each value in a column.

```
coffee_df['Bag.Weight'].str
```

```
<pandas.core.strings.accessor.StringMethods at 0x7f95ade8a220>
```

What we want is to split it

```
coffee_df['Bag.Weight'].str.split(' ').sample(10)
```

```
93     [100, lbs]
245      [69, kg]
279       [2, kg]
476      [69, kg]
806      [10, kg]
891       [1, kg]
908      [60, kg]
748       [1, kg]
955      [60, kg]
218      [59, kg]
Name: Bag.Weight, dtype: object
```

Since this looks good, we can save it to a variable.

```
split_bw = coffee_df['Bag.Weight'].str.split(' ')
```

We still have one problem, each element contains a list.

```
type(split_bw[0])
```

```
list
```

What we want is a `DataFrame` with two columns, one of the first value and one of the second value for each list.

A `DataFrame` is build of Series which are each row (and each column) the lists we have are each the content that we want to put in one Series and then stack them all together.

To do this, we can cast each list to a `Series` using the `apply` method which automatically stacks Series or DataFrames back together after applying a function to each row (or column).

Recall that in python, we can use types as functions to cast

```
num = '2'
```

```
type(int(num))
```

```
int
```

```
type(num)
```

```
str
```

So, back to our real problem:

```
split_bw.apply(pd.Series)
```

	0	1
0	60	kg
1	60	kg
2	1	NaN
3	60	kg
4	60	kg
...	...	...
1306	1	kg
1307	2	kg
1308	69	kg
1309	1	kg
1310	69	kg

1311 rows × 2 columns

This looks good, but these columns are not very informative, so we can rename them.

Rename can take many different inputs and be applied on lots of parts, but we will use it with a dictionary that serves as a mapping (like the mathematical sense of mapping; like a function). It will change the column with each key to the value.

```
split_df = split_bw.apply(pd.Series).rename(columns={0:'Weight',1:'Units'})
split_df
```

	Weight	Units
0	60	kg
1	60	kg
2	1	NaN
3	60	kg
4	60	kg
...	...	...
1306	1	kg
1307	2	kg
1308	69	kg
1309	1	kg
1310	69	kg

1311 rows × 2 columns

This is good, but it's only the one column. We can use concat to put them together.

```
pd.concat([coffee_df,split_df],axis=1).head(2)
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Expiration	Certification.Body	
0	1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	April 3rd, 2016	METAD Agricultural Development plc	309fcf77415a3661
1	2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2015	metad agricultural developmet plc	1950-2200	...	April 3rd, 2016	METAD Agricultural Development plc	309fcf77415a3661

2 rows × 46 columns

Why `axis=1` again?

Let's look at the other one.

```
bad_concat = pd.concat([coffee_df,split_df],axis=0)
bad_concat.shape
```

```
(2622, 46)
```

It has double the rows

> **Important**
>
> checking the shape is the first thing to do to see if you did the right one

and the bottom of it most columns are NaN (null, missing)

```
bad_concat.tail()
```

	Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Number	Company	Altitude	...	Expiration	Certification.Body	Certification.Add
1306	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
1307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
1308	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
1310	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	

5 rows × 46 columns

9.4. Unpacking Jsons

```
rhodyprog4ds_gh_events_url = 'https://api.github.com/orgs/rhodyprog4ds/events'
```

```
course_gh_df = pd.read_json(rhodyprog4ds_gh_events_url)
course_gh_df.head()
```

	id	type	actor	repo	payload	public	created_at	org
0	25320534974	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11704710069, 'size': 1, 'distinct_...	True	2022-11-18 03:11:38+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
1	25320514276	PushEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11704698397, 'size': 1, 'distinct_...	True	2022-11-18 03:09:50+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
2	25144088977	PushEvent	{'id': 41898282, 'login': 'github-actions[bot]...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'push_id': 11613190412, 'size': 1, 'distinct_...	True	2022-11-10 02:48:15+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
3	25144043727	ReleaseEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'action': 'published', 'release': {'url': 'ht...	True	2022-11-10 02:44:25+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...
4	25144027183	CreateEvent	{'id': 10656079, 'login': 'brownsarahm', 'disp...	{'id': 532028859, 'name': 'rhodyprog4ds/BrownF...	{'ref': 'c27', 'ref_type': 'tag', 'master_bran...	True	2022-11-10 02:43:00+00:00	{'id': 69595187, 'login': 'rhodyprog4ds', 'gra...

We want to transform each one of those from a dictionary like thing into a row in a data frame.

```
type(course_gh_df['actor'])
```

```
pandas.core.series.Series
```

Recall, that base python types can be used as function, to cast an object from type to another.

```
5
```

```
5
```

```
type(5)
```

```
int
```

```
str(5)
```

```
'5'
```

To unpack one column we can cast each element of the column to a series and then stack them back together.

First, let's look at one row of one column

```
course_gh_df['actor'][0]
```

```
{'id': 10656079,
 'login': 'brownsarahm',
 'display_login': 'brownsarahm',
 'gravatar_id': '',
 'url': 'https://api.github.com/users/brownsarahm',
 'avatar_url': 'https://avatars.githubusercontent.com/u/10656079?'}
```

Now let's cast it to a Series

```
pd.Series(course_gh_df['actor'][0])
```

```
id                                                10656079
login                                           brownsarahm
display_login                                   brownsarahm
gravatar_id
url                       https://api.github.com/users/brownsarahm
avatar_url       https://avatars.githubusercontent.com/u/10656079?
dtype: object
```

What we want is to do this over and over and stack them.

The `apply` method does this for us, in one compact step.

```
course_gh_df['actor'].apply(pd.Series)
```

	id	login	display_login	gravatar_id	url	avatar_url
0	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
1	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
2	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?
3	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
4	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
5	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
6	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?
7	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
8	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?
9	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
10	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?
11	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
12	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
13	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
14	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?
15	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
16	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
17	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
18	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?
19	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
20	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?
21	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
22	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
23	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
24	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?
25	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
26	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
27	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
28	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?
29	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?

How can we do this for all of the columns and put them back together after?

First, let's make a list of the columns we need to convert.

```
js_cols = ['actor','repo','payload','org']
```

When we use `.apply(pd.Series)` we get a a DataFrame.

```
type(course_gh_df['actor'].apply(pd.Series))
```

```
pandas.core.frame.DataFrame
```

`pd.concat` takes a list of DataFrames and puts the together in one DataFrame.

to illustrate, it's nice to make small dataFrames.

```
df1 = pd.DataFrame([[1,2,3],[3,4,7]], columns = ['A','B','t'])
df2 = pd.DataFrame([[10,20,30],[30,40,70]], columns = ['AA','BB','t'])
df1
```

	A	B	t
0	1	2	3
1	3	4	7

```
df2
```

	AA	BB	t
0	10	20	30
1	30	40	70

If we use concat with the default settings, it stacks them vertically and aligns any columns that have the same name.

```
pd.concat([df1,df2])
```

	A	B	t	AA	BB
0	1.0	2.0	3	NaN	NaN
1	3.0	4.0	7	NaN	NaN
0	NaN	NaN	30	10.0	20.0
1	NaN	NaN	70	30.0	40.0

So, since the original DataFrames were both 2 rows with 3 columns each, with one column name appearing in both, we end up with a new DataFrame with shape (4,5) and it fills with `NaN` in the top right and the bottom left.

```
pd.concat([df1,df2]).shape
```

```
(4, 5)
```

We can use the `axis` parameter to tell it how to combine them. The default is `axis=0`, but `axis=1` will combine along rows.

```
pd.concat([df1,df2], axis =1)
```

	A	B	t	AA	BB	t
0	1	2	3	10	20	30
1	3	4	7	30	40	70

So now we get no NaN values, because both DataFrames have the same number of rows and the same index.

```
df1.index == df2.index
```

```
array([ True,  True])
```

and we have a total of 6 columns and 2 rows.

```
pd.concat([df1,df2], axis =1).shape
```

```
(2, 6)
```

Back to our gh data, we want to make a list of DataFrames where each DataFrame corresponds to one of the columns in the original DataFrame, but unpacked and then stack them horizontally (`axis=1`) because each DataFrame in the list is based on the same original DataFrame, they again have the same index.

```
pd.concat([course_gh_df[cur_col].apply(pd.Series) for cur_col in js_cols],
          axis=1)
```

	id	login	display_login	gravatar_id	url	avatar_url	id	
0	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
1	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
2	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	rhodyprog
3	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
4	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
5	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
6	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	rhodyprog
7	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
8	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	rhodyprog
9	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
10	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	rhodyprog
11	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
12	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
13	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
14	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	rhodyprog
15	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
16	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
17	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
18	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	rhodyprog
19	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
20	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	rhodyprog
21	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
22	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
23	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
24	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	rhodyprog
25	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
26	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
27	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
28	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	rhodyprog
29	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	297149047	rhodyprog4ds/assi

30 rows × 28 columns

> **Try it Yourself**
>
> examine the list of DataFrames to see what structure they share and do not share

In this case we get the same 30 rows, beacuse that's what the API gave us and turned our 4 columns from `js_cols` into 26 columns.

```
pd.concat([course_gh_df[cur_col].apply(pd.Series) for cur_col in js_cols],
          axis=1).shape
```

```
(30, 28)
```

If we had used the default, we'd end up with 120 rows (30*4) and we have only 19 columns, because there are subfield names that are shared across the original columns. (eg most have an `id`)

```
pd.concat([course_gh_df[cur_col].apply(pd.Series) for cur_col in js_cols],
          axis=0).shape
```

```
(120, 21)
```

> **Try it yourself**
>
> How could you anticipate how many are shared?

we might want to rename the new columns so that they have the original column name prepended to the new name. This will help us distinguish between the different `id` columns

pandas has a `rename` method for this.

and this is another job for lambdas.

```
pd.concat([course_gh_df[cur_col].apply(pd.Series).rename(columns = lambda c:
cur_col + '_' +c)
           for cur_col in js_cols],
          axis=1)
```

	actor_id	actor_login	actor_display_login	actor_gravatar_id	actor_url	actor_avatar_url	repo_id	
0	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
1	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
2	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
3	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
4	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
5	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
6	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
7	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
8	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
9	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
10	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
11	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
12	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
13	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
14	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
15	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
16	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
17	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
18	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
19	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
20	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
21	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
22	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
23	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
24	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
25	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
26	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
27	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
28	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
29	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	297149047	rhody

30 rows × 28 columns

the `rename` method's `column` parameter can take a lambda defined inline, which is helpful, because we want that function to take one parameter (the current columnt name) and do the same thing to all of the columns within a single DataFrame, but to prepend a different thing for each DataFrame

```
pd.concat([course_gh_df[cur_col].apply(pd.Series).rename(columns = lambda c:
cur_col + '_' +c)
           for cur_col in js_cols],
          axis=1)
```

	actor_id	actor_login	actor_display_login	actor_gravatar_id	actor_url	actor_avatar_url	repo_id	
0	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
1	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
2	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
3	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
4	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
5	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
6	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
7	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
8	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
9	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
10	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
11	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
12	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
13	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
14	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
15	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
16	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
17	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
18	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
19	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
20	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
21	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
22	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
23	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
24	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatars.githubusercontent.com/u/41898282?	532028859	
25	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
26	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
27	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
28	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	532028859	
29	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatars.githubusercontent.com/u/10656079?	297149047	rhody

30 rows × 28 columns

So now, we have the unpacked columns with good column names, but we lost the columns that were originally good.

How can we append the new columns to the old ones? First we can make a DataFrame that's just the columns not on the list that we're goign to expand.

```
course_gf_df_good = course_gh_df[[col for col in
          course_gh_df.columns if not(col in js_cols)]]
course_gf_df_good
```

	id	type	public	created_at
0	25320534974	PushEvent	True	2022-11-18 03:11:38+00:00
1	25320514276	PushEvent	True	2022-11-18 03:09:50+00:00
2	25144088977	PushEvent	True	2022-11-10 02:48:15+00:00
3	25144043727	ReleaseEvent	True	2022-11-10 02:44:25+00:00
4	25144027183	CreateEvent	True	2022-11-10 02:43:00+00:00
5	25144017610	PushEvent	True	2022-11-10 02:42:09+00:00
6	25142512510	PushEvent	True	2022-11-10 00:49:41+00:00
7	25142364425	PushEvent	True	2022-11-10 00:40:43+00:00
8	25139682863	PushEvent	True	2022-11-09 21:54:16+00:00
9	25139577301	PushEvent	True	2022-11-09 21:48:21+00:00
10	25139071066	PushEvent	True	2022-11-09 21:18:56+00:00
11	25138999879	IssuesEvent	True	2022-11-09 21:14:50+00:00
12	25138969075	PushEvent	True	2022-11-09 21:13:04+00:00
13	25138947856	IssuesEvent	True	2022-11-09 21:11:50+00:00
14	25115458774	PushEvent	True	2022-11-09 01:17:52+00:00
15	25087626486	ReleaseEvent	True	2022-11-08 01:56:11+00:00
16	25087620296	CreateEvent	True	2022-11-08 01:55:44+00:00
17	25087612809	PushEvent	True	2022-11-08 01:55:10+00:00
18	25043076741	PushEvent	True	2022-11-05 00:02:20+00:00
19	25043031795	PushEvent	True	2022-11-04 23:56:23+00:00
20	25042967428	PushEvent	True	2022-11-04 23:47:14+00:00
21	25042941563	ReleaseEvent	True	2022-11-04 23:43:41+00:00
22	25042936184	CreateEvent	True	2022-11-04 23:42:56+00:00
23	25042926798	PushEvent	True	2022-11-04 23:41:37+00:00
24	24997307055	PushEvent	True	2022-11-03 02:32:54+00:00
25	24997291157	IssuesEvent	True	2022-11-03 02:31:13+00:00
26	24997284566	ReleaseEvent	True	2022-11-03 02:30:32+00:00
27	24997260517	CreateEvent	True	2022-11-03 02:27:33+00:00
28	24997253357	PushEvent	True	2022-11-03 02:26:49+00:00
29	24997193960	PushEvent	True	2022-11-03 02:20:34+00:00

Then we can prepend that to the list that we pass to `concat`. We have to put it in a list first, then use + to do that.

```
pd.concat([course_gf_df_good]+[course_gh_df[col].apply(pd.Series,).rename(
  columns= lambda i_col: col + '_' + i_col )
      for col in js_cols],axis=1)
```

	id	type	public	created_at	actor_id	actor_login	actor_display_login	actor_gravatar_id	actor_url	
0	25320534974	PushEvent	True	2022-11-18 03:11:38+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
1	25320514276	PushEvent	True	2022-11-18 03:09:50+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
2	25144088977	PushEvent	True	2022-11-10 02:48:15+00:00	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatar
3	25144043727	ReleaseEvent	True	2022-11-10 02:44:25+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
4	25144027183	CreateEvent	True	2022-11-10 02:43:00+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
5	25144017610	PushEvent	True	2022-11-10 02:42:09+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
6	25142512510	PushEvent	True	2022-11-10 00:49:41+00:00	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatar
7	25142364425	PushEvent	True	2022-11-10 00:40:43+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
8	25139682863	PushEvent	True	2022-11-09 21:54:16+00:00	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatar
9	25139577301	PushEvent	True	2022-11-09 21:48:21+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
10	25139071066	PushEvent	True	2022-11-09 21:18:56+00:00	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatar
11	25138999879	IssuesEvent	True	2022-11-09 21:14:50+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
12	25138969075	PushEvent	True	2022-11-09 21:13:04+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
13	25138947856	IssuesEvent	True	2022-11-09 21:11:50+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
14	25115458774	PushEvent	True	2022-11-09 01:17:52+00:00	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatar
15	25087626486	ReleaseEvent	True	2022-11-08 01:56:11+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
16	25087620296	CreateEvent	True	2022-11-08 01:55:44+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
17	25087612809	PushEvent	True	2022-11-08 01:55:10+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
18	25043076741	PushEvent	True	2022-11-05 00:02:20+00:00	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatar
19	25043031795	PushEvent	True	2022-11-04 23:56:23+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
20	25042967428	PushEvent	True	2022-11-04 23:47:14+00:00	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatar
21	25042941563	ReleaseEvent	True	2022-11-04 23:43:41+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
22	25042936184	CreateEvent	True	2022-11-04 23:42:56+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
23	25042926798	PushEvent	True	2022-11-04 23:41:37+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
24	24997307055	PushEvent	True	2022-11-03 02:32:54+00:00	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions[bot]	https://avatar
25	24997291157	IssuesEvent	True	2022-11-03 02:31:13+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
26	24997284566	ReleaseEvent	True	2022-11-03 02:30:32+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
27	24997260517	CreateEvent	True	2022-11-03 02:27:33+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
28	24997253357	PushEvent	True	2022-11-03 02:26:49+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar
29	24997193960	PushEvent	True	2022-11-03 02:20:34+00:00	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm	https://avatar

30 rows × 32 columns

10. Cleaning Data: fixing values

So far, we've dealt with structural issues in data. but there's a lot more to cleaning.

Today, we'll deal with how to fix the values within the data.

Examples of how values can be represented poorly:

Stanford Policy Lab Open Policing Project data readme Propublica Machine Bias the "How we acquired data" section

> **Hint**
>
> you can treat this one proejct as multiple cleaned datasets and study it more carefully for A4.

10.1. Admin

- watch this repo to get announcements
- grading will be updated over the weekend!
- remember to accept a4

> **Hacktoberfest**
>
> learn about Hacktoberfest](https://hacktoberfest.com/)

> **Important**
>
> Use the Submit workflow on your Actions tab for each assignment

```
import pandas as pd
import seaborn as sns
import numpy as np #
na_toy_df = pd.DataFrame(data = [[1,3,4,5],[2 ,6, np.nan,4]])

# make plots look nicer and increase font size
sns.set_theme(font_scale=2)
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-
database/master/data/arabica_data_cleaned.csv'

coffee_df = pd.read_csv(arabica_data_url)


rhodyprog4ds_gh_events_url = 'https://api.github.com/orgs/rhodyprog4ds/events'
course_gh_df = pd.read_json(rhodyprog4ds_gh_events_url)
```

10.2. Missing Values

Dealing with missing data is a whole research area. There isn't one solution.

in 2020 there was a whole workshop on missing

one organizer is the main developer of sci-kit learn the ML package we will use soon. In a 2020 invited talk he listed more automatic handling as an active area of research and a development goal for sklearn.

There are also many classic approaches both when training and when applying models.

example application in breast cancer detection

Even in pandas, dealing with missing values is under experimentation as to how to represent it symbolically

Missing values even causes the datatypes to change

That said, there are are om Pandas gives a few basic tools:

- dropna
- fillna

Dropping is a good choice when you otherwise have a lot of data and the data is missing at random.

Dropping can be risky if it's not missing at random. For example, if we saw in the coffee data that one of the scores was missing for all of the rows from one country, or even just missing more often in one country, that could bias our results.

Filling can be good if you know how to fill reasonably, but don't have data to spare by dropping. For example

- you can approximate with another column
- you can approximate with that column from other rows

whatever you do, document it

10.2.1. Finding Missing Values

Our skill in summarixing and getting overviews of data helps us know when we have a problem.

```
coffee_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1311 entries, 0 to 1310
Data columns (total 44 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             1311 non-null   int64
 1   Species                1311 non-null   object
 2   Owner                  1304 non-null   object
 3   Country.of.Origin      1310 non-null   object
 4   Farm.Name              955 non-null    object
 5   Lot.Number             270 non-null    object
 6   Mill                   1001 non-null   object
 7   ICO.Number             1165 non-null   object
 8   Company                1102 non-null   object
 9   Altitude               1088 non-null   object
 10  Region                 1254 non-null   object
 11  Producer               1081 non-null   object
 12  Number.of.Bags         1311 non-null   int64
 13  Bag.Weight             1311 non-null   object
 14  In.Country.Partner     1311 non-null   object
 15  Harvest.Year           1264 non-null   object
 16  Grading.Date           1311 non-null   object
 17  Owner.1                1304 non-null   object
 18  Variety                1110 non-null   object
 19  Processing.Method      1159 non-null   object
 20  Aroma                  1311 non-null   float64
 21  Flavor                 1311 non-null   float64
 22  Aftertaste             1311 non-null   float64
 23  Acidity                1311 non-null   float64
 24  Body                   1311 non-null   float64
 25  Balance                1311 non-null   float64
 26  Uniformity             1311 non-null   float64
 27  Clean.Cup              1311 non-null   float64
 28  Sweetness              1311 non-null   float64
 29  Cupper.Points          1311 non-null   float64
 30  Total.Cup.Points       1311 non-null   float64
 31  Moisture               1311 non-null   float64
 32  Category.One.Defects   1311 non-null   int64
 33  Quakers                1310 non-null   float64
 34  Color                  1095 non-null   object
 35  Category.Two.Defects   1311 non-null   int64
 36  Expiration             1311 non-null   object
 37  Certification.Body     1311 non-null   object
 38  Certification.Address  1311 non-null   object
 39  Certification.Contact  1311 non-null   object
 40  unit_of_measurement    1311 non-null   object
 41  altitude_low_meters    1084 non-null   float64
 42  altitude_high_meters   1084 non-null   float64
 43  altitude_mean_meters   1084 non-null   float64
dtypes: float64(16), int64(4), object(24)
memory usage: 450.8+ KB
```

10.2.2. Example Filling

The 'Lot.Number' has a lot of NaN values, how can we explore it?

We can look at the type:

```
coffee_df['Lot.Number'].dtype
```

```
dtype('O')
```

And we can look at the value counts.

```
coffee_df['Lot.Number'].value_counts()
```

```
1                             18
020/17                         6
019/17                         5
2                              3
102                            3
                              ..
11/23/0696                     1
3-59-2318                      1
8885                           1
5055                           1
017-053-0211/ 017-053-0212     1
Name: Lot.Number, Length: 221, dtype: int64
```

We see that a lot are '1', maybe we know that when the data was collected, if the Farm only has one lot, some people recorded '1' and others left it as missing. So we could fill in with 1:

```
coffee_df['Lot.Number'].fillna(1).head(10)
```

```
0    1
1    1
2    1
3    1
4    1
5    1
6    1
7    1
8    1
9    1
Name: Lot.Number, dtype: object
```

Note that even after we called `fillna` we display it again and the original data is unchanged.

```
coffee_df['Lot.Number'].head(3)
```

```
0    NaN
1    NaN
2    NaN
Name: Lot.Number, dtype: object
```

To save the filled in column we have a few choices:

- use the `inplace` parameter. This doesn't offer performance advantages, but does It still copies the object, but then reassigns the pointer. Its under discussion to deprecate
- write to a new DataFrame
- add a column

We'll use adding a column:

```
coffee_df['lot_number_clean'] = coffee_df['Lot.Number'].fillna('1')
```

```
coffee_df['lot_number_clean'].value_counts()
```

```
1                             1059
020/17                           6
019/17                           5
102                              3
103                              3
                              ...
3-59-2318                        1
8885                             1
5055                             1
MCCFWXA15/16                     1
017-053-0211/ 017-053-0212       1
Name: lot_number_clean, Length: 221, dtype: int64
```

10.2.3. Example Dropping

To illustrate how `dropna` works, we'll use the `shape` method after each call.
Dropna is going to drop either rows or columns. So we

```
coffee_df.shape
```

```
(1311, 45)
```

By default, it drops any row with one or more `NaN` values.

```
coffee_df.dropna().shape
```

```
(130, 45)
```

We could instead tell it to only drop rows with `NaN` in a subset of the columns. We might do this if we were interested in studying the impact of altitude on the ratings .

```
coffee_df.dropna(subset=['altitude_low_meters']).shape
```

```
(1084, 45)
```

In the Open Policing Project Data Summary we saw that they made a summary information that showed which variables had at least 70% not missing values. We can similarly choose to keep only variables that have more than a specific threshold of data, using the `thresh` parameter and `axis=1` to drop along columns.

The `thresh` parameter requires an `int` not a fraction of the rows, so we can save the shape values to variables and then use that

```
nrows, ncols = coffee_df.shape
coffee_df.dropna(thresh =.7*nrows,axis=1).shape
```

```
(1311, 44)
```

This dataset is actually in pretty good shape, but if we use a more stringent threshold it drops more columns.

```
nrows, ncols = coffee_df.shape
coffee_df.dropna(thresh =.85*nrows,axis=1).shape
```

```
(1311, 34)
```

10.3. Inconsistent values

This was one of the things that many of you anticipated or had observed. A useful way to investigate for this, is to use `value_counts` and sort them alphabetically by the values from the original data, so that similar ones will be consecutive in the list. Once we have the `value_counts()` Series, the values from the `coffee_df` become the index, so we use `sort_index`.

Let's look at the `In.Country.Partner` column

```
coffee_df['In.Country.Partner'].value_counts().sort_index()
```

10.4. We can see there's only one `Blossom Valley International\n` but 58 `Blossom Valley International`, the former is likely a typo, especially since `\n` is a special character for a newline. Similarly, with 'Specialty Coffee Ass' and 'Specialty Coffee Association'.

This is another job for dictionaries, we make one with the value to replace as the key and the value to insert as the value.

```
partner_corrections = {'Blossom Valley International\n':'Blossom Valley International',
  'Specialty Coffee Ass':'Specialty Coffee Association'}
coffee_df['in_country_partner_clean'] = coffee_df['In.Country.Partner'].replace(
  to_replace=partner_corrections)
coffee_df['in_country_partner_clean'].value_counts().sort_index()
```

10.5. Fixing data at load time

Some of the different parameters in `read_csv` can also fix how it reads in data.

For example `header` can make somethign like this:

Ethnicity	Asian				Black				Hispanic			
Gender	Female		Male		Female		Male		Female		Male	
	count	mean	count	mean	count	mean	count	mean	count	mean	count	m
Age Cohort												
0 - 5	6	1544.33333							18	1431.33333	26	1366
51 +												

read in correctly.

10.6. A Cleaning Data Recipe

not everything possible, but good enough for this course

1. Can you use parameters to read the data in better?
2. Fix the index and column headers (making these easier to use makes the rest easier)
3. Is the data strucutred well?
4. Are there missing values?
5. Do the datatypes match what you expect by looking at the head or a sample?
6. Are categorical variables represented in usable way?
7. Does your analysis require filtering or augmenting the data?

10.7. Further reading

Instead of more practice with these manipulations, below are more examples of cleaning data to see how these types of manipulations get used.
Your goal here is not to memorize every possible thing, but to build a general idea of what good data looks like and good habits for cleaning data and keeping it reproducible.

- Cleaning the Adult Dataset
- All Shades

Also here are some tips on general data management and organization.

This article is a comprehensive discussion of data cleaning.

10.8. Questions

> **Important**
>
> I will add these later. I have a deadline tomorrow.

11. Building Datasets from Multiple Sources

focus this week is on how to programmatically combine sources of data

We will start by looking at combining multiple tabular data formats and see how to get data from other sources.

11.1. Self- assessment, what's your plan

Take a few minutes to think about the following questions and make a few notes for yourself whereever you need them to be (a planner, calendar, etc). Share one takeaway or question you have below when you're done.

1. What achievements have you earned?
2. Does BrightSpace seem accurate to what you've done?
3. If not, e-mail brownsarahm with [CSC310] or [DSP310] in the subject and specific details about what you think is missing and why
4. Are you on track to earn the grade you want in this class?
5. If not, what will you need to do (respond more in class, submit more assignments, use your portfolio to catch up) to get back on track?
6. If you are on track and you want to earn above a B, take a minute to think about your portfolio. (tip: post an idea as an issue to get early feedback and help shaping your idea)

11.2. Logistics

Check your earned achievements. See the instructions that are now saved for future reference on the left side.

11.3. Merges

```
import pandas as pd
```

I created a folder with datasets we use in class

```
course_data_url =
'https://raw.githubusercontent.com/rhodyprog4ds/rhodyds/main/data/'
```

We can load in two data sets of player information.

```
df_18 = pd.read_csv(course_data_url+ '2018-players.csv')
df_19 = pd.read_csv(course_data_url+ '2019-players.csv')
```

and take a peek at each

```
df_18.head(2)
```

	TEAM_ID	PLAYER_ID	SEASON
0	1610612761	202695	2018
1	1610612761	1627783	2018

Important

Remember `columns` is an attribute, so it does not need `()`

```
df_19.columns
```

```
Index(['PLAYER_NAME', 'TEAM_ID', 'PLAYER_ID', 'SEASON'], dtype='object')
```

Let's make note of the shape of each

```
df_18.shape, df_19.shape
```

```
((748, 3), (626, 4))
```

11.4. What if we want to analyze them together?

We can combine them vertically:

```
pd.concat([df_18,df_19]).shape
```

```
(1374, 4)
```

Note that this has the maximum number of columns (because both had some overlapping columns) and the total number of rows.

11.5. How can we find which players changed teams?

To do this we want to have one player column and a column with each year's team.

We can use a merge to do that.

```
pd.merge(df_18,df_19,).head(2)
```

	TEAM_ID	PLAYER_ID	SEASON	PLAYER_NAME

Note

the `head` here is only to help us see more at a time

if we merge them without any parameters, it tries to merge on all shared columns. We want to merge them using the `PLAYER_ID` column though, we would say hat we are "merging on player ID" and we use the `on` parameter to do it

```
pd.merge(df_18,df_19, on = 'PLAYER_ID').head(2)
```

	TEAM_ID_x	PLAYER_ID	SEASON_x	PLAYER_NAME	TEAM_ID_y	SEASON_y
0	1610612761	202695	2018	Kawhi Leonard	1610612746	2019
1	1610612761	1627783	2018	Pascal Siakam	1610612761	2019

Since there are other columns that appear in both DataFrames, they get a suffis, which by default is `x` or `y`, we can specify them though.

```
pd.merge(df_18,df_19, on = 'PLAYER_ID',
        suffixes=('_2018','_2019')).head(2)
```

	TEAM_ID_2018	PLAYER_ID	SEASON_2018	PLAYER_NAME	TEAM_ID_2019	SEASON_2019
0	1610612761	202695	2018	Kawhi Leonard	1610612746	2019
1	1610612761	1627783	2018	Pascal Siakam	1610612761	2019

We also told it what to append to any column names that match, but are not the same across both datasets.

```
df_18.head(1)
```

	TEAM_ID	PLAYER_ID	SEASON
0	1610612761	202695	2018

```
df_19.head(1)
```

	PLAYER_NAME	TEAM_ID	PLAYER_ID	SEASON
0	Royce O'Neale	1610612762	1626220	2019

```
df1819 = pd.merge(df_18,df_19, on = 'PLAYER_ID',
        suffixes=('_2018','_2019'))
```

```
df1819.shape
```

```
(538, 6)
```

By default, this uses an *inner* merge, so we get the players that are in both datasets only.

```
df1819.head()
```

	TEAM_ID_2018	PLAYER_ID	SEASON_2018	PLAYER_NAME	TEAM_ID_2019	SEASON_2019
0	1610612761	202695	2018	Kawhi Leonard	1610612746	2019
1	1610612761	1627783	2018	Pascal Siakam	1610612761	2019
2	1610612761	201188	2018	Marc Gasol	1610612761	2019
3	1610612763	201188	2018	Marc Gasol	1610612761	2019
4	1610612761	201980	2018	Danny Green	1610612747	2019

Some players still appear twice, because they were in one of the datsets twice, this happens when a player plays for two team in one season.

11.6. Which players played in 2018, but not 2019?

We have different types of merges, inner is both, out is either. Left and right keep all the rows of one dataFrame. We can use left with `df_18` as the left DataFrame to see which players played only in 18.

```
pd.merge(df_18,df_19,on='PLAYER_ID', how='left',suffixes=('_2018','_2019')).tail()
```

	TEAM_ID_2018	PLAYER_ID	SEASON_2018	PLAYER_NAME	TEAM_ID_2019	SEASON_2019
749	1610612752	1629246	2018	NaN	NaN	NaN
750	1610612748	1629159	2018	NaN	NaN	NaN
751	1610612762	1629163	2018	NaN	NaN	NaN
752	1610612743	1629150	2018	NaN	NaN	NaN
753	1610612738	1629167	2018	NaN	NaN	NaN

To pick out those rows:

```
df_18_only = pd.merge(df_18,df_19,on='PLAYER_ID', how='left', suffixes=
('_2018','_2019'))
df_18_only[df_18_only['SEASON_2019'].isna()]
```

	TEAM_ID_2018	PLAYER_ID	SEASON_2018	PLAYER_NAME	TEAM_ID_2019	SEASON_2019
9	1610612761	202391	2018	NaN	NaN	NaN
11	1610612761	201975	2018	NaN	NaN	NaN
18	1610612744	101106	2018	NaN	NaN	NaN
23	1610612744	2733	2018	NaN	NaN	NaN
24	1610612744	201973	2018	NaN	NaN	NaN
...	...	...	...	...	...	...
749	1610612752	1629246	2018	NaN	NaN	NaN
750	1610612748	1629159	2018	NaN	NaN	NaN
751	1610612762	1629163	2018	NaN	NaN	NaN
752	1610612743	1629150	2018	NaN	NaN	NaN
753	1610612738	1629167	2018	NaN	NaN	NaN

216 rows × 6 columns

```
df_18_left = pd.merge(df_18,df_19.drop(columns=['SEASON'])
                                  ,on='PLAYER_ID', how='left', suffixes=
('_2018','_2019'))
df_18_only = df_18_only[df_18_only['TEAM_ID_2019'].isna()]
df_18_only.head()
```

	TEAM_ID_2018	PLAYER_ID	SEASON_2018	PLAYER_NAME	TEAM_ID_2019	SEASON_2019
9	1610612761	202391	2018	NaN	NaN	NaN
11	1610612761	201975	2018	NaN	NaN	NaN
18	1610612744	101106	2018	NaN	NaN	NaN
23	1610612744	2733	2018	NaN	NaN	NaN
24	1610612744	201973	2018	NaN	NaN	NaN

```
n_18_only, _ = df_18_only.drop_duplicates(subset=['PLAYER_ID']).shape
n_18_only
```

```
178
```

11.7. Which players played for the same team both seasons?

```
df_same_team = pd.merge(df_18,df_19, on = ['PLAYER_ID','TEAM_ID'],)
df_same_team.head()
```

	TEAM_ID	PLAYER_ID	SEASON_x	PLAYER_NAME	SEASON_y
0	1610612761	1627783	2018	Pascal Siakam	2019
1	1610612761	201188	2018	Marc Gasol	2019
2	1610612761	200768	2018	Kyle Lowry	2019
3	1610612761	1627832	2018	Fred VanVleet	2019
4	1610612761	201586	2018	Serge Ibaka	2019

In this case, the suffix only applies to season, but they're not telling us much, so we can clean it up using `drop` before we merge.

```
df_18_only_clean =
pd.merge(df_18.drop(columns='SEASON'),df_19.drop(columns='SEASON'), on =
['PLAYER_ID','TEAM_ID'],)
df_18_only_clean.head()
```

	TEAM_ID	PLAYER_ID	PLAYER_NAME
0	1610612761	1627783	Pascal Siakam
1	1610612761	201188	Marc Gasol
2	1610612761	200768	Kyle Lowry
3	1610612761	1627832	Fred VanVleet
4	1610612761	201586	Serge Ibaka

```
df_18_only_clean.shape
```

```
(263, 3)
```

```
df_18_only_clean.drop_duplicates(subset=['PLAYER_ID']).shape
```

```
(263, 3)
```

We do not need to drop the duplicates in this case becaue we merged on two columns and there were no actual duplicates in the original dataset.

11.8. Visualizing merges

```
left = pd.DataFrame(
    {
        "key": ["K0", "K1", "K2", "K4"],
        "A": ["A0", "A1", "A2", "A4"],
        "B": ["B0", "B1", "B2", "B4"],
    }
)


right = pd.DataFrame(
    {
        "key": ["K0", "K1", "K2", "K3"],
        "C": ["C0", "C1", "C2", "C3"],
        "D": ["D0", "D1", "D2", "D3"],
    }
)
```

```
left
```

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K4	A4	B4

```
right
```

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

```
pd.merge(left, right, how='inner',)
```

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2

```
pd.merge(left, right, how='outer')
```

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K4	A4	B4	NaN	NaN
4	K3	NaN	NaN	C3	D3

```
pd.merge(left, right, how='left')
```

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K4	A4	B4	NaN	NaN

```
pd.merge(left, right, how='right')
```

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	NaN	NaN	C3	D3

We can merge on multiple columns too/

```
left = pd.DataFrame(
    {
        "key1": ["K0", "K0", "K1", "K2"],
        "key2": ["K0", "K1", "K0", "K1"],
        "A": ["A0", "A1", "A2", "A3"],
        "B": ["B0", "B1", "B2", "B3"],
    }
)


right = pd.DataFrame(
    {
        "key1": ["K0", "K1", "K1", "K2"],
        "key2": ["K0", "K0", "K0", "K0"],
        "C": ["C0", "C1", "C2", "C3"],
        "D": ["D0", "D1", "D2", "D3"],
    }
)
```

```
left
```

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

```
right
```

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

```
pd.merge(left, right, on=["key1", "key2"])
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

```
pd.merge(left, right, on=["key1", "key2"],how='outer')
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN
5	K2	K0	NaN	NaN	C3	D3

```
pd.merge(left, right,on = 'key1' )
```

	key1	key2_x	A	B	key2_y	C	D
0	K0	K0	A0	B0	K0	C0	D0
1	K0	K1	A1	B1	K0	C0	D0
2	K1	K0	A2	B2	K0	C1	D1
3	K1	K0	A2	B2	K0	C2	D2
4	K2	K1	A3	B3	K0	C3	D3

```
left = pd.DataFrame({"A": [1, 2], "B": [2, 2]})
right = pd.DataFrame({"A": [4, 5, 6], "B": [2, 2, 2]})
```

```
left
```

	A	B
0	1	2
1	2	2

```
right
```

	A	B
0	4	2
1	5	2
2	6	2

```
pd.merge(left, right, on="B", how="outer")
```

	A_x	B	A_y
0	1	2	4
1	1	2	5
2	1	2	6
3	2	2	4
4	2	2	5
5	2	2	6

11.9. Questions After Class

> **Important**
>
> several questions were easiest to answer within the narrative of the notes above.

11.9.1. How do I represent NaN as a variable? (Ex. df1819['TEAM_ID_y'==NaN]…something like that?)

For that specific case, you can use the `isna` method as I did above, but if you need a NaN constant otherwise pandas provides

```
pd.NA
```

```
<NA>
```

and numpy provides

```
import numpy as np
np.nan
```

```
nan
```

Watch out though because they are not the same:

```
np.nan == pd.NA
```

```
<NA>
```

and this cannot even assert

```
assert pd.NA == np.nan
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [44], line 1
----> 1 assert pd.NA == np.nan

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/_libs/missing.pyx:382, in pandas._libs.missing.NAType.__bool__()

TypeError: boolean value of NA is ambiguous
```

The `pandas` `pd.isna` method is robust, and it knows how `numpy` works (because it is built on top and imports).

```
pd.isna(np.nan)
```

```
True
```

```
pd.isna(pd.NA)
```

```
True
```

However, `numpy` does not know `pandas`

```
np.isnan(pd.NA)
```

```
<NA>
```

and returns a value that cannot be used for indexing

11.9.2. Is there an easier way to look at the changes with merge?

Any of the tools we have to look at a DataFrame will work. I think using a small DataFrame is the best way to get a good idea of how they each work, so I included that. Overall, checking the shape gives a broad view and then comparing values gives the detials.

11.9.3. Do merges also delete any other overlapping columns other than the one they are told to consolidate on?

No they copy them all

11.9.4. in pd.merge(p18,p19, on='PLAYER_ID',how='____').shape is p18 and p19 supposed to be df_18 and df_19?

Yes, sorry, I read them in using different names than were in my notes

11.9.5. Learning more about accessing data from databases

We will access databases on Wednesday

11.9.6. How to merge data that does not line up as well as these two datasets?

We will see how to merge on columns that have the same data, but different columns this week. When there is no columns that match value to value, you have to transform and add new columns, combining what we learned about cleaning data to make them line up somehow. For example, maybe one dataset has dates like `YYYY-MM-DD` and the other data set has only months like "January". You could split the whole date into three columns and transform the strings to numbers or numbers to strings by treating the columns as dates

11.9.7. Why did we merged on 'PLAYER_ID'?

We were interested in linking based on the players.

11.9.8. What is the second best source after documentation? For ex: you found a parameter that might do what you want it to in pandas, but where do you go after to make sure? I'm not sure if I ran it, I'd specifically know if it "worked"

Ideally, you should know what you are looking for to know if it worked or not. If you are not sure what you are looking for, that is a good time to try to think that through, or attend office hours. Next is to create an issue on your assignment repo if the question might reveal a solution.

11.9.9. I think I need to understand how GitHub works because I'm following the steps of the assignments and I feel like I'm doing something wrong every time I have to work on GitHub.

If you are getting feedback, you are close enough. However, I am offering an extra session on GitHub on Friday.

12. Web Scraping

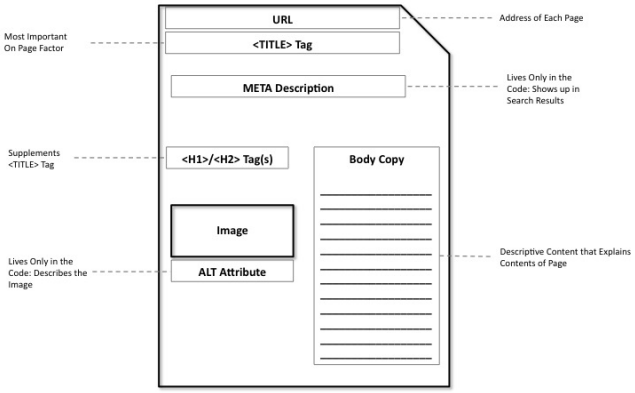
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

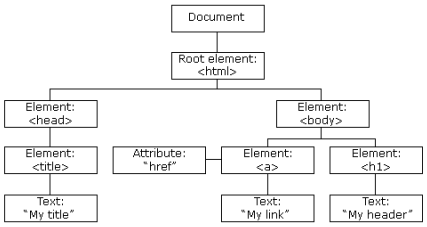
> **Warning**
>
> If it says it cannot load one of the libraries, use pip inside your notebook to install, then restart your kernel (Kernel menu, choose restart)
>
> ```
> pip install beautifulsoup4
> ```

We have that `read_html` can get content from an actual website, not a data file that is hosted somewhere on the internet, that takes tables on a website and returns a list of DataFrames.

12.1. Everything is Data

For the purpose of this class, it is best to think of the content on a web page like a datastructure.





there are tags `<>` that define the structure, and these can be further classified with `classes`

12.2. Scraping a URI website

We're going to create a DataFrame about URI CS & Statistics Faculty.

from the people page of the department website.

We can inspect the page to check that it's well structured

> **Warning**
>
> With great power comes great responsibility.
>
> - always check the robots.txt
> - do not do things that the owner says not to do
> - government websites are typically safe

We'lls ave the URL for easy use

```
cs_people_url = 'https://web.uri.edu/cs/people/'
```

Then we can use the `requests` library to make a call to the internet. It actually gets back a response object which has a lot of extra information. For today we only need the `content` from the page which is an attrtibute of that object

```
cs_people_html = requests.get(cs_people_url).content
```

This is raw:

```
type(cs_people_html)
```

```
bytes
```

```
cs_people_html
```

b'\n<!DOCTYPE html>\n<html lang="en-US">\n\t\n<head>\n<meta charset="UTF-
8">\n<meta name="viewport" content="width=device-width, initial-scale=1">\n<link
rel="profile" href="http://gmpg.org/xfn/11">\n\n<title>People – Department
of Computer Science and Statistics</title>\n\r\n\t <meta name=\'robots\'
content=\'max-image-preview:large\' />\n<link rel=\'dns-prefetch\'
href=\'//s.w.org\' />\n<link rel="alternate" type="application/rss+xml"
title="Department of Computer Science and Statistics » Feed"
href="https://web.uri.edu/cs/feed/" />\n<link rel="alternate"
type="application/rss+xml" title="Department of Computer Science and Statistics
» Comments Feed" href="https://web.uri.edu/cs/comments/feed/"
/>\n\t\t<script type="text/javascript">\n\t\t\twindow._wpemojiSettings =
{"baseUrl":"https:\\/\\/s.w.org\\/images\\/core\\/emoji\\/13.0.1\\/72x72\\/","ext"
:".png","svgUrl":"https:\\/\\/s.w.org\\/images\\/core\\/emoji\\/13.0.1\\/svg\\/","
svgExt":".svg","source":{"concatemoji":"https:\\/\\/web.uri.edu\\/cs\\/wp-
includes\\/js\\/wp-emoji-release.min.js?ver=5.7.1"}};\n\t\t\t!function(e,a,t){var
n,r,o,i=a.createElement("canvas"),p=i.getContext&&i.getContext("2d");function
s(e,t){var
a=String.fromCharCode;p.clearRect(0,0,i.width,i.height),p.fillText(a.apply(this,e)
,0,0);e=i.toDataURL();return
p.clearRect(0,0,i.width,i.height),p.fillText(a.apply(this,t),0,0),e===i.toDataURL(
)}function c(e){var
t=a.createElement("script");t.src=e,t.defer=t.type="text/javascript",a.getElements
ByTagName("head")[0].appendChild(t)}for(o=Array("flag","emoji"),t.supports=
{everything:!0,everythingExceptFlag:!0},r=0;r<o.length;r++)t.supports[o[r]]=functi
on(e){if(!p||!p.fillText)return!1;switch(p.textBaseline="top",p.font="600 32px
Arial",e){case"flag":return s([127987,65039,8205,9895,65039],
[127987,65039,8203,9895,65039])?!1:!s([55356,56826,55356,56819],
[55356,56826,8203,55356,56819])&&!s([55356,57332,56128,56423,56128,56418,56128,564
21,56128,56430,56128,56423,56128,56447],

[55356,57332,8203,56128,56423,8203,56128,56418,8203,56128,56421,8203,56128,56430,8203,56128,56423,8203,56128,56447]);case"emoji":return!s([55357,56424,8205,55356,57212],[55357,56424,8203,55356,57212])}return!1}(o[r]),t.supports.everything=t.supports.everything&&t.supports[o[r]],"flag"!==o[r]&&(t.supports.everythingExceptFlag=t.supports.everythingExceptFlag&&t.supports[o[r]]);t.supports.everythingExceptFlag=t.supports.everythingExceptFlag&&!t.supports.flag,t.DOMReady=!1,t.readyCallback=function(){t.DOMReady=!0},t.supports.everything||(n=function(){t.readyCallback()},a.addEventListener?(a.addEventListener("DOMContentLoaded",n,!1),e.addEventListener("load",n,!1)):(e.attachEvent("onload",n),a.attachEvent("onreadystatechange",function(){"complete"===a.readyState&&t.readyCallback()})),(n=t.source||{}).concatemoji?c(n.concatemoji):n.wpemoji&&n.twemoji&&(c(n.twemoji),c(n.wpemoji)))}(window,document,window._wpemojiSettings);\n\t\t</script>\n\t\t<style type="text/css">\nimg.wp-smiley,\nimg.emoji {\n\tdisplay: inline !important;\n\tborder: none !important;\n\tbox-shadow: none !important;\n\theight: 1em !important;\n\twidth: 1em !important;\n\tmargin: 0 .07em !important;\n\tvertical-align: -0.1em !important;\n\tbackground: none !important;\n\tpadding: 0 !important;\n}\n</style>\n\t<link rel=\'stylesheet\' id=\'wp-block-library-css\' href=\'https://web.uri.edu/cs/wp-includes/css/dist/block-library/style.min.css?ver=5.7.1\' type=\'text/css\' media=\'all\' />\n<link rel=\'stylesheet\' id=\'uricl-css-css\' href=\'https://web.uri.edu/cs/wp-content/plugins/uri-component-library/css/cl.built.css?ver=5.0.2\' type=\'text/css\' media=\'all\' />\n<link rel=\'stylesheet\' id=\'uri-kuali-styles-css\' href=\'https://web.uri.edu/cs/wp-content/plugins/uri-kuali/css/uri-kuali.css?ver=5.7.1\' type=\'text/css\' media=\'all\' />\n<link rel=\'stylesheet\' id=\'uri-courses-styles-css\' href=\'https://web.uri.edu/cs/wp-content/plugins/uri-courses/assets/courses.css?ver=5.7.1\' type=\'text/css\' media=\'all\' />\n<link rel=\'stylesheet\' id=\'uri-people-styles-css\' href=\'https://web.uri.edu/cs/wp-content/plugins/uri-people-tool/assets/people.css?ver=5.7.1\' type=\'text/css\' media=\'all\' />\n<link rel=\'stylesheet\' id=\'wp-lightbox-2.min.css-css\' href=\'https://web.uri.edu/cs/wp-content/plugins/wp-lightbox-2/styles/lightbox.min.css?ver=1.3.4\' type=\'text/css\' media=\'all\' />\n<link rel=\'stylesheet\' id=\'uri-modern-style-css\' href=\'https://web.uri.edu/cs/wp-content/themes/uri-modern/style.css?ver=2.4.4\' type=\'text/css\' media=\'all\' />\n<link rel=\'stylesheet\' id=\'tablepress-default-css\' href=\'https://web.uri.edu/cs/wp-content/plugins/tablepress/css/default.min.css?ver=1.13\' type=\'text/css\' media=\'all\' />\n<script type=\'text/javascript\' src=\'https://web.uri.edu/cs/wp-includes/js/jquery/jquery.min.js?ver=3.5.1\' id=\'jquery-core-js\'></script>\n<script type=\'text/javascript\' src=\'https://web.uri.edu/cs/wp-includes/js/jquery/jquery-migrate.min.js?ver=3.3.2\' id=\'jquery-migrate-js\'></script>\n<link rel="https://api.w.org/" href="https://web.uri.edu/cs/wp-json/" /><link rel="alternate" type="application/json" href="https://web.uri.edu/cs/wp-json/wp/v2/pages/629" />\n<link rel="EditURI" type="application/rsd+xml" title="RSD" href="https://web.uri.edu/cs/xmlrpc.php?rsd" />\n<link rel="wlwmanifest" type="application/wlwmanifest+xml" href="https://web.uri.edu/cs/wp-includes/wlwmanifest.xml" /> \n<meta name="generator" content="WordPress 5.7.1" />\n<link rel="canonical" href="https://web.uri.edu/cs/people/" />\n<link rel=\'shortlink\' href=\'https://web.uri.edu/cs/?p=629\' />\n<link rel="alternate" type="application/json+oembed" href="https://web.uri.edu/cs/wp-json/oembed/1.0/embed?url=https%3A%2F%2Fweb.uri.edu%2Fcs%2Fpeople%2F" />\n<link rel="alternate" type="text/xml+oembed" href="https://web.uri.edu/cs/wp-json/oembed/1.0/embed?url=https%3A%2F%2Fweb.uri.edu%2Fcs%2Fpeople%2F&format=xml" />\n<meta name="description" content="People Full-time Faculty Adjunct Faculty and Limited Join Appointments" />\n<meta name="twitter:card" content="summary_large_image" />\n<meta name="twitter:site" content="@universityofri" />\n<meta name="twitter:creator" content="@universityofri" />\n<meta property="og:url" content="https://web.uri.edu/cs/people/" />\n<meta property="og:title" content="People" />\n<meta property="og:description" content="People Full-time Faculty Adjunct Faculty and Limited Join Appointments" />\n<meta property="og:image" content="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/logo-wordmark.png" />\n\t\t\t\t\t<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({\'gtm.start\':\nnew Date().getTime(),event:\'gtm.js\'});var f=d.getElementsByTagName(s)[0],\nj=d.createElement(s),dl=l!=\'dataLayer\'?\'&l=\'+l:\'\';j.async=true;j.src=\n\'https://www.googletagmanager.com/gtm.js?id=\'+i+dl;f.parentNode.insertBefore(j,f);\n})(window,document,\'script\',\'dataLayer\',\'GTM-K5GL9W\');</script>\n\t\t\t<style type="text/css" id="wp-custom-css">\n\t\t\t.button-list .cl-button {\n\tmargin-bottom: 1rem;\n}\n\n#calendar .date,#calendar-wrap header {\n\ttext-align:center\n}\n#calendar {\n\twidth:100%\n}\n#calendar a {\n\tcolor:#005eff;\n\ttext-decoration:none\n}\n#calendar ul {\n\tlist-style:none;\n\tpadding:0;\n\tmargin:0;\n\twidth:100%\n}\n#calendar li {\n\tdisplay:block;\n\tfloat:left;\n\twidth:14.342%;\n\tpadding:5px;\n\tbox-sizing:border-box;\n\tborder:1px solid #ccc;\n\tmargin-right:-1px;\n\tmargin-bottom:-1px\n}\n#calendar ul.weekdays {\n\theight:40px;\n\tbackground:#002147\n}\n#calendar ul.weekdays li {\n\ttext-align:center;\n\ttext-transform:uppercase;\n\tline-height:1.2;\n\tborder:none!important;\n\tpadding:.75rem .5rem;\n\tcolor:#fff;\n\tfont-size:.9rem\n}\n#calendar ul.days.mobile {\n\tdisplay:none\n}\n#calendar .days li {\n\theight:15rem\n}\n#calendar .days.events-zero li {\n\theight:10rem\n}\n#calendar .days.events-one li {\n\theight:10.75rem\n}\n#calendar .days.events-two li {\n\theight:11.5rem\n}\n#calendar .days.events-three li {\n\theight:15rem\n}\n#calendar .date {\n\tmargin-bottom:5px;\n\tpadding:4px;\n\tcolor:#000;\npadding-right: 1rem;\n\tfloat:right;\n}\n#calendar .event {\n\tclear:both;\n\tdisplay:block;\n\tfont-size: 1rem;\n\tfont-family: hind,arial,sans-serif;\n\tborder-radius:30px;\n\tpadding:.4rem .25rem .25rem 5rem;\n\tmargin-top:5px;\n\tmargin-bottom:5px;\n\tline-height:1.75;\n\tbackground:#e4f2f2;\n\ttext-decoration:none;\n\tfloat:left;\n\tposition:relative;\n\twidth:calc(100% * 4.68)\n}\n#calendar .event.four-day{\n\t\twidth:calc(76% * 4.68)\n}\n#calendar .event.mobile {\n\tdisplay:none\n}\n#calendar .event.scratch {\n\tbackground:#E2D58B;\n}\n#calendar .event.lego {\n\tbackground:#E1EDE8;\n}\n#calendar .event.datacience {\n\tbackground:#FCB97D;\n}\n#calendar .event.girlstech {\n\tbackground:#C2CFB2;\n}\n#calendar .event.games {\n\tbackground:#F79365;\n}\n#calendar .event.programming {\n\tbackground:#BCDCE0;\n}\n#calendar .event.webdesign {\n\tbackground:#E09FA2;\n}\n#calendar .event-desc {\n\tcolor:#666;\n\tmargin:3px 0 7px;\n\ttext-decoration:none;\n\tdisplay:inline-block\n}\n#calendar .event-time {\n\tmargin:3px 0 7px 5px;\n\ttext-decoration:none;\n\tdisplay:inline-block\n}\n#calendar .other-month {\n\tbackground:#f5f5f5;\n\tcolor:#666\n}\n#desc {\n\tdisplay:none;\n\tbackground:#eee;\n\tbox-shadow: 0 0 4px #999;\n\tpadding:10px;\n\t\tcolor: #000;\n}\n#calendar .event-desc:hover+#desc {\n\tdisplay:block;\n\tposition:absolute;\n\tz-index:2\n}\n.days+header h1 {\n\tpadding-top:2rem;\n\tclear: left\n}\n@media(max-width:768px) {\n\t#calendar .event img,#calendar .other-month,#calendar .weekdays {\n\t\tdisplay:none\n\t}\n\t#calendar .event {\n\t\twidth:calc(100% - 2em);\n\t\tpadding-left:20px;\n\t\tpadding-right:10px\n\t}\n\t#calendar .event.mobile,#calendar ul.days.mobile {\n\t\tdisplay:block\n\t}\n\t#calendar li {\n\t\theight:auto!important;\n\t\tborder:1px solid #ededed;\n\t\twidth:100%;\n\t\tpadding:10px;\n\t\tmargin-bottom:-1px\n\t}\n\t#calendar .date {\n\t\tfloat:none\n\t}\n\n}\n.peopleitem.has-thumbnail img {\n\twidth: 200px;\n}\n\t\t</style>\n\t\t\t\n<!-- Favicons -->\n<link rel="mask-icon" href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/safari-pinned-tab.svg" color="#005eff">\n<link rel="icon" type="image/png" href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/favicon.png">\n<link rel="apple-touch-icon" href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/apple-touch-icon.png">\n<link rel="apple-touch-icon" sizes="180x180" href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/apple-touch-icon-180x180.png">\n\n\n</head>\n\t\n<body class="page-template-default page page-id-629 page-parent group-blog ln-people">\n\t\n<noscript><iframe src="https://www.googletagmanager.com/ns.html?id=GTM-K5GL9W" height="0" width="0" style="display:none;visibility:hidden"></iframe></noscript>\t\n<div id="page"

class="site">\n\tSkip to content\n\n\n\t \n\t\t\n\t<header id="brandbar" class="site-header" role="banner">\n\n\t\t <img src="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/logo-print.png" width="120px" alt="University of Rhode Island"> \n\n\t\t \n\t\t\t<input type="checkbox" id="gsform-toggle" role="presentation" aria-label="Toggle visibility of the search box.">\n\t\t\t<label for="gsform-toggle" id="gsform">Search</label>\n\t\t\t<form id="gs" method="get" action="https://www.uri.edu/search" name="global_general_search_form">\n\t\t\t\t<input type="hidden" name="cx" value="016863979916529535900:17qai8akniu" />\n\t\t\t\t<input type="hidden" name="cof" value="FORID:11" />\n\t\t\t\t<label id="gs-query-label" for="gs-query">Searchbox</label>\n\t\t\t\t<input role="searchbox" name="q" id="gs-query" value="" type="text" placeholder="Search" />\n\t\t\t\t<input type="submit" id="gs-submit" class="searchsubmit" name="searchsubmit" value="Search" />\n\t\t\t</form>\n\t\t \n\n\t \n\t\t \n\t\t\t University of Rhode Island \n\n\t\t\t \n\t\t\t\t<input type="checkbox" id="gateways-toggle" role="presentation" aria-label="Open the audience gateways menu when browsing on mobile">\n\t\t\t\t<label for="gateways-toggle" id="gateways-label">You</label>\n\t\t\t\t<ul id="gateways-menu" role="menu">\n\t\t\t\t\t
Future Students
\n\t\t\t\t\t
Students
\n\t\t\t\t\t
Faculty
\n\t\t\t\t\t
Staff
\n\t\t\t\t\t
Parents and Families
\n\t\t\t\t\t
Alumni
\n\t\t\t\t\t
Community
\n\t\t\t\t
\n\t\t\t \n\n\t\t \n\t\t \n\n\t</header>
<!-- #brandbar -->\n\t\t\n\t<header id="siteheader">\n\n\t\t\n<div id="sitebanner" \n class="light">\n\t\n\t \n\t\t<div id="sb-background-image" \n\t\t style="background-image:url(https://web.uri.edu/cs/files/cropped-Big-Data.jpg)"> \n\t\t \n\t \n\t\n\t \n\n\t\t\n \n\t\t\t\n\t<h1 class="site-title">\n\t\t\n\t\t\tDepartment of Computer Science and Statistics\t\t\n\t</h1>\n\t\t\t<h2 class="site-description">College of Arts and Sciences</h2>\n\t\n \n\t\t\n\t\t \n\t\t\t\t\t\t \n\t\t\n\t <!-- #sitebranding -->\n\n
<!-- #sitebanner -->\n\n\t\t \n\t\t\t\n\n<nav id="breadcrumbs" aria-label="Breadcrumb">\n\t
URI
Arts and Sciences
Department of Computer Science and Statistics
<li aria-current="page">People
</nav>\n \n\t\t\t\n \n\t<section class="cl-wrapper cl-menu-wrapper"> <ul id="menu-navigation" class="cl-menu-list cl-menu-list-no-js"><li id="menu-item-8243" class="menu-item menu-item-type-custom menu-item-object-custom menu-item-home menu-item-8243">Home
\n<li id="menu-item-8255" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8255">About
\n<li id="menu-item-8806" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8806">Academics
\n<li id="menu-item-8257" class="menu-item menu-item-type-post_type menu-item-object-page current-menu-item page_item page-item-629 current_page_item menu-item-8257">People
\n<li id="menu-item-9661" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-9661">Research
\n<li id="menu-item-10871" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-10871">News and Events
\n<li id="menu-item-8267" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8267">Contact
\n </section> \n\t\t\t \n\t\t\t\n\t\t\n\t\t</header>\n\t \n\t\n\t \n\t\t\n\t\n\n\t<main id="main" class="site-main" role="main">\n\n\t\t\n<article id="post-629" class="post-629 page type-page status-publish hentry">\n\t\n\t \n\t\t<h1>People</h1>\n<section class="cl-wrapper cl-menu-wrapper"> <ul id="menu-people" class="cl-menu-list cl-menu-list-no-js"><li id="menu-item-8737" class="menu-item menu-item-type-post_type menu-item-object-page current-menu-item page_item page-item-629 current_page_item menu-item-8737">Faculty
\n<li id="menu-item-8746" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8746">Staff
\n<li id="menu-item-11844" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-11844">Faculty Emeriti
\n </section>\n<h2>Full-time Faculty</h2>\n \n \n\t<header>\n\t\t \n\t\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t\t\t\t
\n\t\t\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t\t\t\t\t<h3 class="p-name">Marco Alvarez</h3>\n\t\t\t\n\t\t\t\t\t \n\t</header>\n\t \n\n\t\t Associate Professor | Director of Graduate Studies \n\t\t\n\t\t\t\t Computer Science \n\t\t\n\t\t\n\t\t\t\t 401.874.5009 – malvarez@uri.edu \n\t\t\n\t \n\t \n \n \n\t<header>\n\t\t \n\t\t\t\t\t\t\t\t\t\t\t<h3 class="p-name">Samantha Armenti</h3>\n\t\t\t\n\t\t\t\t\t \n\t</header>\n\t \n\n\t\t Assistant Teaching Professor \n\t\t\n\t\t\t\t Computer Science \n\t\t\n\t\t\n\t\t\t\t sarmenti@uri.edu \n\t\t\n\t \n\t \n \n \n\t<header>\n\t\t \n\t\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t\t\t\t\n\t\t\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t\t\t\t\t<h3 class="p-name">Sarah Brown</h3>\n\t\t\t\n\t\t\t\t\t \n\t</header>\n\t \n\n\t\t Assistant Professor \n\t\t\n\t\t\t\t Computer Science \n\t\t\n\t\t\n\t\t\t\t brownsarahm@uri.edu \n\t\t\n\t\t\t\t \n\t \n \n \n\t<header>\n\t\t \n\t\t\t\t\t\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t\t\t\t<a href="https://web.uri.edu/cs/meet/michael-conti/"><img width="2560" height="2475" src="https://web.uri.edu/cs/files/mike-conti-scaled.jpg" class="u-photo wp-post-image" alt="" loading="lazy" srcset="https://web.uri.edu/cs/files/mike-conti-scaled.jpg 2560w, https://web.uri.edu/cs/files/mike-conti-300x290.jpg 300w, https://web.uri.edu/cs/files/mike-conti-1024x990.jpg 1024w, https://web.uri.edu/cs/files/mike-conti-768x743.jpg 768w,

```
https://web.uri.edu/cs/files/mike-conti-1536x1485.jpg 1536w,
https://web.uri.edu/cs/files/mike-conti-2048x1980.jpg 2048w,
https://web.uri.edu/cs/files/mike-conti-364x352.jpg 364w,
https://web.uri.edu/cs/files/mike-conti-500x483.jpg 500w,
https://web.uri.edu/cs/files/mike-conti-1000x967.jpg 1000w,
https://web.uri.edu/cs/files/mike-conti-1280x1238.jpg 1280w,
https://web.uri.edu/cs/files/mike-conti-2000x1934.jpg 2000w" sizes="(max-width:
2560px) 100vw, 2560px" /></a>\n\t\t\t\t\t\t\t</figure>\n\t\t\n\t\t\t\t\t\t\t<h3
class="p-name"><a href="https://web.uri.edu/cs/meet/michael-conti/">Michael
Conti</a></h3>\n\t\t\t\n\t\t\t\n\t\t</div>\n\t</header>\n\t<div
class="inside">\n\n\t\t<p class="people-title p-job-title">Assistant Teaching
Professor</p>\n\t\t\n\t\t\t\t<p class="people-department">Computer
Science</p>\n\t\t\n\t\t\n\t\t\t\t<p class="people-misc"><a class="u-email"
href="mailto:michaelconti@uri.edu ">michaelconti@uri.edu </a></p>\n\t\t\n\t\t<div
style="clear:both;"></div>\n\t</div>\n</div>\n<div class="peopleitem h-card has-
thumbnail">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t<a
href="https://web.uri.edu/cs/meet/noah-daniels/"><img width="219" height="219"
src="https://web.uri.edu/cs/files/noah-daniels-web.png" class="u-photo  wp-post-
image" alt="" loading="lazy" srcset="https://web.uri.edu/cs/files/noah-daniels-
web.png 219w, https://web.uri.edu/cs/files/noah-daniels-web-150x150.png 150w"
sizes="(max-width: 219px) 100vw, 219px" />
</a>\n\t\t\t\t\t</figure>\n\t\t\n\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/noah-daniels/">Noah Daniels</a>
</h3>\n\t\t\n\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Assistant Professor</p>\n\t\t\n\t\t\t<p
class="people-department">Computer Science</p>\n\t\t\n\t\t\n\t\t\t<p
class="people-misc"><a class="u-email"
href="mailto:noah_daniels@uri.edu">noah_daniels@uri.edu</a></p>\n\t\t\n\t<div
style="clear:both;"></div>\n\t</div>\n</div>\n<div class="peopleitem h-card has-
thumbnail">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t<a
href="https://web.uri.edu/cs/meet/lisa-dipippo/"><img width="125" height="126"
src="https://web.uri.edu/cs/files/lisa-dipippo-web.jpg" class="u-photo  wp-post-
image" alt="" loading="lazy" />
</a>\n\t\t\t\t\t</figure>\n\t\t\n\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/lisa-dipippo/">Lisa DiPippo</a>
</h3>\n\t\t\n\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Professor | Chair</p>\n\t\t\n\t\t\t<p
class="people-department">Computer Science</p>\n\t\t\n\t\t\n\t\t\t<p
class="people-misc"><a class="u-email"
href="mailto:ldipippo@uri.edu">ldipippo@uri.edu</a></p>\n\t\t\n\t<div
style="clear:both;"></div>\n\t</div>\n</div>\n<div class="peopleitem h-card has-
thumbnail">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t<a
href="https://web.uri.edu/cs/meet/victor-fay-wolfe/"><img width="125" height="125"
src="https://web.uri.edu/cs/files/victor-fay-wolfe-web.jpg" class="u-photo  wp-
post-image" alt="" loading="lazy" />
</a>\n\t\t\t\t\t</figure>\n\t\t\n\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/victor-fay-wolfe/">Victor Fay-Wolfe</a>
</h3>\n\t\t\n\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Professor</p>\n\t\t\n\t\t\t<p class="people-
department">Computer Science</p>\n\t\t\n\t\t\n\t\t\t<p class="people-misc"><a
class="u-email" href="mailto:vfaywolfe@uri.edu">vfaywolfe@uri.edu</a>
</p>\n\t\t\n\t<div style="clear:both;"></div>\n\t</div>\n</div>\n<div
class="peopleitem h-card ">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/lutz-hamel/">Lutz Hamel</a>
</h3>\n\t\t\n\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Associate Professor </p>\n\t\t\n\t\t\t<p
class="people-department">Computer Science</p>\n\t\t\n\t\t\n\t\t\t<p
class="people-misc"><a class="u-email"
href="mailto:lutzhamel@uri.edu">lutzhamel@uri.edu</a></p>\n\t\t\n\t<div
style="clear:both;"></div>\n\t</div>\n</div>\n<div class="peopleitem h-card has-
thumbnail">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t<a
href="https://web.uri.edu/cs/meet/abdeltawab-hendawi/"><img width="1000"
height="885" src="https://web.uri.edu/cs/files/0-1.jpeg" class="u-photo  wp-post-
image" alt="" loading="lazy" srcset="https://web.uri.edu/cs/files/0-1.jpeg 1000w,
https://web.uri.edu/cs/files/0-1-300x266.jpeg 300w,
https://web.uri.edu/cs/files/0-1-768x680.jpeg 768w,
https://web.uri.edu/cs/files/0-1-364x322.jpeg 364w,
https://web.uri.edu/cs/files/0-1-500x443.jpeg 500w" sizes="(max-width: 1000px)
100vw, 1000px" /></a>\n\t\t\t\t\t\t\t</figure>\n\t\t\n\t\t\t\t\t\t\t<h3
class="p-name"><a href="https://web.uri.edu/cs/meet/abdeltawab-
hendawi/">Abdeltawab Hendawi</a>
</h3>\n\t\t\n\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Assistant Professor</p>\n\t\t\n\t\t\t<p
class="people-department">Data Science | Computer
Science</p>\n\t\t\n\t\t\n\t\t\t\t<p class="people-misc"><span class="p-
tel">401.874.5738</span> &ndash; <a class="u-email"
href="mailto:hendawi@uri.edu">hendawi@uri.edu</a></p>\n\t\t\n\t<div
style="clear:both;"></div>\n\t</div>\n</div>\n<div class="peopleitem h-card has-
thumbnail">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t<a
href="https://web.uri.edu/cs/meet/jean-yves-herve/"><img width="300" height="299"
src="https://web.uri.edu/cs/files/jean-yves-herve-web.jpg" class="u-photo  wp-
post-image" alt="" loading="lazy" srcset="https://web.uri.edu/cs/files/jean-yves-
herve-web.jpg 300w, https://web.uri.edu/cs/files/jean-yves-herve-web-150x150.jpg
150w" sizes="(max-width: 300px) 100vw, 300px" />
</a>\n\t\t\t\t\t</figure>\n\t\t\n\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/jean-yves-herve/">Jean-Yves Herv\xc3\xa9</a>
</h3>\n\t\t\n\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Associate Professor</p>\n\t\t\n\t\t\t<p
class="people-department">Computer Science</p>\n\t\t\n\t\t\n\t\t\t<p
class="people-misc"><a class="u-email"
href="mailto:jyh@cs.uri.edu">jyh@cs.uri.edu</a></p>\n\t\t\n\t<div
style="clear:both;"></div>\n\t</div>\n</div>\n<div class="peopleitem h-card has-
thumbnail">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t<a
href="https://web.uri.edu/cs/meet/natallia-katenka/"><img width="200" height="200"
src="https://web.uri.edu/cs/files/natalia-katenka-web.jpg" class="u-photo  wp-
post-image" alt="" loading="lazy" srcset="https://web.uri.edu/cs/files/natalia-
katenka-web.jpg 200w, https://web.uri.edu/cs/files/natalia-katenka-web-150x150.jpg
150w" sizes="(max-width: 200px) 100vw, 200px" />
</a>\n\t\t\t\t\t</figure>\n\t\t\n\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/natallia-katenka/">Natallia Katenka</a>
</h3>\n\t\t\n\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Associate Professor | Director of Undergraduate
Studies</p>\n\t\t\n\t\t\t<p class="people-
department">Statistics</p>\n\t\t\n\t\t\n\t\t\t<p class="people-misc"><a
class="u-email" href="mailto:nkatenka@uri.edu">nkatenka@uri.edu</a>
</p>\n\t\t\n\t<div style="clear:both;"></div>\n\t</div>\n</div>\n<div
class="peopleitem h-card ">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/soheyb-kouider/">Soheyb Kouider</a>
</h3>\n\t\t\n\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Associate Teaching
Professor</p>\n\t\t\n\t\t\t<p class="people-
department">Statistics</p>\n\t\t\n\t\t\n\t\t\t<p class="people-misc"><span
class="p-tel">401.874.2562</span> &ndash; <a class="u-email"
href="mailto:soheyb@uri.edu">soheyb@uri.edu</a></p>\n\t\t\n\t<div
style="clear:both;"></div>\n\t</div>\n</div>\n<div class="peopleitem h-card has-
thumbnail">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t<a
href="https://web.uri.edu/cs/meet/edmund-lamagna/"><img width="150" height="150"
src="https://web.uri.edu/cs/files/edmund-lamagna-web.jpg" class="u-photo  wp-post-
image" alt="" loading="lazy" />
</a>\n\t\t\t\t\t</figure>\n\t\t\n\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/edmund-lamagna/">Edmund Lamagna</a>
</h3>\n\t\t\n\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
```

class="people-title p-job-title">Professor \n\t\t\n\t\t\t\t Computer Science \n\t\t\n\t\t\n\t\t\t\t eal@cs.uri.edu
\n\t\t\n\t\t \n\t \n \n \n\t<header>\n\t\t \n\t\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t\n\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t<h3 class="p-name">Indrani Mandal
</h3>\n\t\t\t\n\t\t\t\n\t\t \n\t</header>\n\t \n\n\t\t Assistant Teaching Professor \n\t\t\n\t\t\t Computer Science \n\t\t\n\t\t\n\t\t\t indrani_mandal@uri.edu
\n\t\t\n\t \n\t \n \n \n\t<header>\n\t\t \n\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t
\n\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t<h3 class="p-name">Gavino Puggioni
</h3>\n\t\t\t\n\t\t\t\n\t\t \n\t</header>\n\t \n\n\t\t Associate Professor | Statistics Section Head | Director of Graduate Studies \n\t\t\n\t\t\t Statistics \n\t\t\n\t\t\n\t\t\t 401.874.4388 – gpuggioni@uri.edu \n\t\t\n\t\t \n\t \n \n \n\t<header>\n\t\t \n\t\t\t\t\t\t\t<h3 class="p-name">Jonathan Schrader</h3>\n\t\t\t\n\t\t\t\n\t\t \n\t</header>\n\t \n\n\t\t Assistant Teaching Professor \n\t\t\n\t\t\t Computer Science \n\t\t\n\t\t\n\t\t\t jonathan.schrader@uri.edu
\n\t\t\n\t \n\t \n \n \n\t<header>\n\t\t \n\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t\n\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t<h3 class="p-name">Krishna Venkatasubramanian
</h3>\n\t\t\t\n\t\t\t\n\t\t \n\t</header>\n\t \n\n\t\t Assistant Professor \n\t\t\n\t\t\t Computer Science \n\t\t\n\t\t\n\t\t\t krish@uri.edu \n\t\t\n\t\t \n\t \n \n \n\t<header>\n\t\t \n\t\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t\n\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t<h3 class="p-name">Jing Wu
</h3>\n\t\t\t\n\t\t\t\n\t\t \n\t</header>\n\t \n\n\t\t Associate Professor \n\t\t\n\t\t\t Statistics \n\t\t\n\t\t\n\t\t\t 401.874.4504 – jing_wu@uri.edu \n\t\t\n\t\t \n\t \n \n \n\t<header>\n\t\t \n\t\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t
\n\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t<h3 class="p-name">Yichi Zhang
</h3>\n\t\t\t\n\t\t\t\n\t\t \n\t</header>\n\t \n\n\t\t Assistant Professor \n\t\t\n\t\t\t Statistics \n\t\t\n\t\t\n\t\t\t yichizhang@uri.edu
\n\t\t\n\t\t \n\t \n \n \n\t<header>\n\t\t \n\t\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t
\n\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t<h3 class="p-name">Guangyu Zhu
</h3>\n\t\t\t\n\t\t\t\n\t\t \n\t</header>\n\t \n\n\t\t Assistant Professor \n\t\t\n\t\t\t Statistics \n\t\t\n\t\t\n\t\t\t guangyuzhu@uri.edu
\n\t\t\n\t\t
\n\t \n \n \n \n<h2>Adjunct Faculty and Limited Join Appointments</h2>\n \n \n\t<header>\n\t\t \n\t\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t\n\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t<h3 class="p-name">Ashley Buchanan</h3>\n\t\t\t\n\t\t\t\n\t\t \n\t</header>\n\t<div

```
class="inside">\n\n\t\t<p class="people-title p-job-title">Limited Joint
Appointment</p>\n\t\t\n\t\t\t\t<p class="people-
department">Biostatistics</p>\n\t\t\n\t\t\n\t\t\t\t\t<p class="people-misc"><span
class="p-tel">401.874.4739</span> &ndash; <a class="u-email"
href="mailto:buchanan@uri.edu">buchanan@uri.edu</a></p>\n\t\t\n\t\t<div
style="clear:both;"></div>\n\t</div>\n</div>\n<div class="peopleitem h-card has-
thumbnail">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t\t<a
href="https://web.uri.edu/cs/meet/nina-kajiji/"><img width="125" height="125"
src="https://web.uri.edu/cs/files/nina-kajiji-web.jpg" class="u-photo  wp-post-
image" alt="" loading="lazy" />
</a>\n\t\t\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/nina-kajiji/">Nina Kajiji</a>
</h3>\n\t\t\t\n\t\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Adjunct Associate Professor</p>\n\t\t\n\t\t\t\t<p
class="people-department">Computer Science</p>\n\t\t\n\t\t\n\t\t\t\t\t<p
class="people-misc"><a class="u-email" href="mailto:nina@uri.edu">nina@uri.edu</a>
</p>\n\t\t\n\t\t<div style="clear:both;"></div>\n\t</div>\n</div>\n<div
class="peopleitem h-card has-thumbnail">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t\t<a
href="https://web.uri.edu/cs/meet/rachel-schwartz/"><img width="120" height="120"
src="https://web.uri.edu/cs/files/Rachel-Schwartz-web.jpg" class="u-photo  wp-
post-image" alt="" loading="lazy" />
</a>\n\t\t\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/rachel-schwartz/">Rachel Schwartz</a>
</h3>\n\t\t\t\n\t\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Assistant Professor \xe2\x80\x93 Limited Joint
Appointment</p>\n\t\t\n\t\t\t\t<p class="people-department">Biological
Sciences</p>\n\t\t\n\t\t\n\t\t\t\t\t<p class="people-misc"><span class="p-
tel">401.874.5404</span> &ndash; <a class="u-email"
href="mailto:rsschwartz@uri.edu">rsschwartz@uri.edu</a></p>\n\t\t\n\t\t<div
style="clear:both;"></div>\n\t</div>\n</div>\n<div class="peopleitem h-card has-
thumbnail">\n\t<header>\n\t\t<div
class="header">\n\t\t\t\t\t\t<figure>\n\t\t\t\t\t\t\t\t\t<a
href="https://web.uri.edu/cs/meet/ying-zhang/"><img width="249" height="250"
src="https://web.uri.edu/cs/files/ying-zhang-web.jpg" class="u-photo  wp-post-
image" alt="" loading="lazy" srcset="https://web.uri.edu/cs/files/ying-zhang-
web.jpg 249w, https://web.uri.edu/cs/files/ying-zhang-web-150x150.jpg 150w"
sizes="(max-width: 249px) 100vw, 249px" />
</a>\n\t\t\t\t\t\t\t</figure>\n\t\t\t\n\t\t\t\t\t\t\t<h3 class="p-name"><a
href="https://web.uri.edu/cs/meet/ying-zhang/">Ying Zhang</a>
</h3>\n\t\t\t\n\t\t\t\n\t\t</div>\n\t</header>\n\t<div class="inside">\n\n\t\t<p
class="people-title p-job-title">Assistant Professor \xe2\x80\x93 Limited Joint
Appointment</p>\n\t\t\n\t\t\t\t<p class="people-department">Computer
Science</p>\n\t\t\n\t\t\n\t\t\t\t\t<p class="people-misc"><span class="p-
tel">401.874.4915</span> &ndash; <a class="u-email"
href="mailto:yingzhang@uri.edu">yingzhang@uri.edu</a></p>\n\t\t\n\t\t<div
style="clear:both;"></div>\n\t</div>\n</div>\n</div>\n<p>\n\t</div><!-- .entry-
content -->\n\n\t</article><!-- #post-## -->\n\n\t</main><!-- #main --
>\n\n\n\t</div><!-- #content -->\n\n\t\n\t<div id="actionbar-wrapper">\n\t\t<div
id="actionbar" role="menu">\n\t\t\t<a href="https://www.uri.edu/connect/?
utm_campaign=request-info&#038;utm_source=action-bar&#038;utm_medium=web"
id="action-connect" role="menuitem"><span role="presentation" title="Learn more
about URI: Get in touch"></span>Connect</a><a href="https://www.uri.edu/apply"
id="action-apply" role="menuitem"><span role="presentation"></span>Apply</a><a
href="https://www.uri.edu/tour" id="action-tour" role="menuitem"><span
role="presentation"></span>Tour</a><a href="https://www.uri.edu/give" id="action-
give" role="menuitem"><span role="presentation">
</span>Give</a>\t\t</div>\n\t</div><!-- #actionbar-wrapper -->\n\n\t<footer
id="globalfooter">\n\t\t<div id="basement">\n\t\t\t<div
id="storagebins">\n\t\t\t\t<div id="sb-university">\n\t\t\t\t\t<input id="sb-
university-toggle" type="radio" name="storagebin" value="university"
role="presentation" checked>\n\t\t\t\t\t<label for="sb-university-toggle" aria-
label="Open the University footer menu when browsing on mobile.">
<span>University</span></label>\n\t\t\t\t\t<ul role="menu" aria-label="The
University footer menu.">\n\t\t\t\t\t<li><a
href="https://www.uri.edu/about/leadership/" role="menuitem">Leadership</a>
</li>\n\t\t\t\t\t<li><a href="https://web.uri.edu/diversity/"
role="menuitem">Diversity and Inclusion</a></li>\n\t\t\t\t\t<li><a
href="https://web.uri.edu/global/" role="menuitem">Global</a>
</li>\n\t\t\t\t\t<li><a href="https://www.uri.edu/about/campuses/"
role="menuitem">Campuses</a></li>\n\t\t\t\t\t<li><a
href="https://www.uri.edu/safety/" role="menuitem">Safety</a>
</li>\n\t\t\t\t</ul>\n\t\t\t</div>\n\t\t\t<div id="sb-campus-
life">\n\t\t\t\t<input id="sb-campus-life-toggle" type="radio" name="storagebin"
value="campus-life" role="presentation">\n\t\t\t\t<label for="sb-campus-life-
toggle" aria-label="Open the Campus Life footer menu when browsing on mobile.">
<span>Campus Life</span></label>\n\t\t\t\t<ul role="menu" aria-label="The Campus
Life footer menu.">\n\t\t\t\t<li><a href="https://web.uri.edu/housing/"
role="menuitem">Housing</a></li>\n\t\t\t\t<li><a
href="https://web.uri.edu/dining/" role="menuitem">Dining</a>
</li>\n\t\t\t\t<li><a href="https://www.uri.edu/athletics/"
role="menuitem">Athletics and Recreation</a></li>\n\t\t\t\t<li><a
href="https://www.uri.edu/campus-life/health-and-wellness/" role="menuitem">Health
and Wellness</a></li>\n\t\t\t\t<li><a href="https://events.uri.edu"
role="menuitem">Events</a></li>\n\t\t\t\t</ul>\n\t\t\t</div>\n\t\t<div
id="sb-academics">\n\t\t\t\t<input id="sb-academics-toggle" type="radio"
name="storagebin" value="academics" role="presentation">\n\t\t\t\t<label
for="sb-academics-toggle" aria-label="Open the Academics footer menu when browsing
on mobile."><span>Academics</span></label>\n\t\t\t\t<ul role="menu" aria-
label="The Academics footer menu.">\n\t\t\t\t<li><a
href="https://www.uri.edu/academics/" role="menuitem">Undergraduate</a>
</li>\n\t\t\t\t<li><a href="https://web.uri.edu/graduate-school/"
role="menuitem">Graduate</a></li>\n\t\t\t\t<li><a
href="https://web.uri.edu/advising/" role="menuitem">Advising</a>
</li>\n\t\t\t\t<li><a href="https://web.uri.edu/library/"
role="menuitem">Libraries</a></li>\n\t\t\t\t<li><a
href="https://web.uri.edu/career/students/" role="menuitem">Internships</a>
</li>\n\t\t\t\t</ul>\n\t\t\t</div>\n\t\t</div>\n\t\t<div
id="gimmicks">\n\t\t\t<!-- Tides Widget -->\n\t\t\t<div class="uri-tides-
widget darkmode" data-station="8454049" " data-height="22"><span class="status">
</span></div>\n\t\t\t\t<hr>\n\t\t\t\n\t\t\t<!-- Social Media Component
-->\n\t\t\t<aside class="cl-wrapper cl-social-wrapper"><ul class="cl-social
light  "><li><a href="https://www.facebook.com/universityofri" class="cl-social-
facebook" title="Facebook">Facebook</a></li><li><a
href="https://www.instagram.com/universityofri/" class="cl-social-instagram"
title="Instagram">Instagram</a></li><li><a
href="https://twitter.com/universityofri" class="cl-social-twitter"
title="Twitter">Twitter</a></li><li><a
href="https://www.youtube.com/user/UniversityOfRI" class="cl-social-youtube"
title="YouTube">YouTube</a></li></ul></aside>\t\t\t</div>\n\t\t</div>\n\t\t<div
id="tagline"></div>\n\t\t<div id="legal">\n\t\t\t<p>Copyright &copy; <a
class="subtle" href="http://www.uri.edu/">University of Rhode Island</a> |
University of Rhode Island, Kingston, RI 02881, USA |
1.401.874.1000</p>\n\t\t\t<p>URI is an equal opportunity employer committed to the
principles of affirmative action.  <a class="jobs"
href="https://jobs.uri.edu/">Work at URI</a></p>\n\t\t</div>\n\t</footer><!--
#globalfooter -->\n</div><!-- #page -->\n\n<link rel=\'stylesheet\' id=\'uri-
tides-css-css\'  href=\'https://web.uri.edu/cs/wp-content/plugins/uri-tides-
1.2/css/tides.css?ver=5.7.1\' type=\'text/css\' media=\'all\' />\n<script
type=\'text/javascript\' src=\'https://web.uri.edu/cs/wp-content/plugins/uri-
component-library/js/cl.built.js?ver=20221118\' id=\'uricl-js-js\'>
</script>\n<script type=\'text/javascript\' id=\'wp-jquery-lightbox-js-
extra\'>\n/* <![CDATA[ */\nvar JQLBSettings =
{"fitToScreen":"0","resizeSpeed":"400","displayDownloadLink":"0","navbarOnTop":"0"
,"loopImages":"","resizeCenter":"","marginSize":"","linkTarget":"","help":"","prev
LinkTitle":"previous image","nextLinkTitle":"next image","prevLinkText":"\\u00ab
Previous","nextLinkText":"Next \\u00bb","closeTitle":"close image
gallery","image":"Image ","of":" of
","download":"Download","jqlb_overlay_opacity":"80","jqlb_overlay_color":"#000000"
,"jqlb_overlay_close":"1","jqlb_border_width":"10","jqlb_border_color":"#ffffff","
```

```
jqlb_border_radius":"0","jqlb_image_info_background_transparency":"100","jqlb_imag
e_info_bg_color":"#ffffff","jqlb_image_info_text_color":"#000000","jqlb_image_info
_text_fontsize":"10","jqlb_show_text_for_image":"1","jqlb_next_image_title":"next
image","jqlb_previous_image_title":"previous
image","jqlb_next_button_image":"https:\\/\\/web.uri.edu\\/cs\\/wp-
content\\/plugins\\/wp-lightbox-
2\\/styles\\/images\\/next.gif","jqlb_previous_button_image":"https:\\/\\/web.uri.
edu\\/cs\\/wp-content\\/plugins\\/wp-lightbox-
2\\/styles\\/images\\/prev.gif","jqlb_maximum_width":"","jqlb_maximum_height":"","
jqlb_show_close_button":"1","jqlb_close_image_title":"close image
gallery","jqlb_close_image_max_heght":"22","jqlb_image_for_close_lightbox":"https:
\\/\\/web.uri.edu\\/cs\\/wp-content\\/plugins\\/wp-lightbox-
2\\/styles\\/images\\/closelabel.gif","jqlb_keyboard_navigation":"1","jqlb_popup_s
ize_fix":"0"};\n/* ]]> */\n</script>\n<script type=\'text/javascript\'
src=\'https://web.uri.edu/cs/wp-content/plugins/wp-lightbox-2/js/dist/wp-lightbox-
2.min.js?ver=1.3.4.1\' id=\'wp-jquery-lightbox-js\'></script>\n<script
type=\'text/javascript\' src=\'https://web.uri.edu/cs/wp-content/themes/uri-
modern/js/navigation.js?ver=2.4.4\' id=\'uri-modern-navigation-js\'>
</script>\n<script type=\'text/javascript\' src=\'https://web.uri.edu/cs/wp-
content/themes/uri-modern/js/smoothscroll.min.js?ver=2.4.4\' id=\'uri-modern-
smoothscroll-js\'></script>\n<script type=\'text/javascript\'
src=\'https://web.uri.edu/cs/wp-content/themes/uri-modern/js/skip-link-focus-
fix.js?ver=2.4.4\' id=\'uri-modern-skip-link-focus-fix-js\'></script>\n<script
type=\'text/javascript\' id=\'uri-modern-scripts-js-extra\'>\n/* <![CDATA[ */\nvar
URIMODERN = {"base":"https:\\/\\/web.uri.edu\\/cs","path":
{"page":"https:\\/\\/web.uri.edu\\/cs\\/people\\/","theme":"https:\\/\\/web.uri.ed
u\\/cs\\/wp-content\\/themes\\/uri-
modern","themes":"https:\\/\\/web.uri.edu\\/cs\\/wp-
content\\/themes","plugins":"https:\\/\\/web.uri.edu\\/cs\\/wp-
content\\/plugins"},"theme":{"name":"URI
Modern","version":"2.4.4","textDomain":"uri"},"is":
{"404":false,"childTheme":false,"admin":false},"features":[]};\n/* ]]>
*/\n</script>\n<script type=\'text/javascript\' src=\'https://web.uri.edu/cs/wp-
content/themes/uri-modern/js/script.min.js?ver=2.4.4\' id=\'uri-modern-scripts-
js\'></script>\n<script type=\'text/javascript\' src=\'https://web.uri.edu/cs/wp-
content/plugins/page-links-to/dist/new-tab.js?ver=3.3.5\' id=\'page-links-to-js\'>
</script>\n<script type=\'text/javascript\' src=\'https://web.uri.edu/cs/wp-
includes/js/wp-embed.min.js?ver=5.7.1\' id=\'wp-embed-js\'></script>\n<script
type=\'text/javascript\' id=\'uri-tides-js-extra\'>\n/* <![CDATA[ */\nvar tides =
{"temperature":{"metadata":{"id":"8454049","name":"Quonset
Point","lat":"41.5868","lon":"-71.4110"},"data":[{"t":"2022-11-18
03:00","v":"54.7","f":"0,0,0"}]},"tide":{"predictions":[{"t":"2022-11-17
00:30","v":"0.940","type":"L"},{"t":"2022-11-17 06:52","v":"2.938","type":"H"},
{"t":"2022-11-17 12:24","v":"0.945","type":"L"},{"t":"2022-11-17
19:03","v":"3.020","type":"H"},{"t":"2022-11-18 01:34","v":"0.798","type":"L"},
{"t":"2022-11-18 07:42","v":"3.107","type":"H"},{"t":"2022-11-18
13:39","v":"0.810","type":"L"},{"t":"2022-11-18 19:55","v":"3.081","type":"H"},
{"t":"2022-11-19 02:18","v":"0.584","type":"L"},{"t":"2022-11-19
08:35","v":"3.368","type":"H"},{"t":"2022-11-19 14:40","v":"0.576","type":"L"},
{"t":"2022-11-19 20:51","v":"3.202","type":"H"},{"t":"2022-11-20
02:55","v":"0.329","type":"L"},{"t":"2022-11-20 09:28","v":"3.714","type":"H"},
{"t":"2022-11-20 15:30","v":"0.305","type":"L"},{"t":"2022-11-20
21:46","v":"3.385","type":"H"}]},"date":"1668741118","expires_on":"1668741418"};\n
/* ]]> */\n</script>\n<script type=\'text/javascript\'
src=\'https://web.uri.edu/cs/wp-content/plugins/uri-tides-1.2/js/tides.js?
ver=5.7.1\' id=\'uri-tides-js\'></script>\n<!--
\n######################################
####################################\n######################################
#|================================|#\n################VoV##################
#|___    ___         __  ___  __ |#\n###############/ _ \\################     #|
| |_|  |  |\\ | |/  |_)  |  /__ |#\n###############V==)/\\################     #| |
| | _|_ | \\| |\\  |_) _|_ \\_| |#\n##############/=/ ||=|###############     #|
_    _   _        |#\n#########o(___/=/  _______)o#########     #|      \\    / |_
| \\ / \\       |#\n############/=/#| |=/################     #|       \\/\\/  |_
|_/ \\_/       |#\n############|=|#| |##################     #|
|#\n#############\\=\\| |##################     #|           .~~~~~~.
|#\n##############A=| |##################     #|         ~~~~######~~
|#\n######o#########| |\\########o########     #|         ~~~~#########~~
|#\n######|````A####| |=\\##A````|#######     #|         ~~~~~#######~~
|#\n######| _ \\####V   \\=\\##/ _ |#######     #|         ~~~~~###~~#~~~
|#\n######A/#s `s==\\___/=/s` s#\\A#######     #|         ~~~~~##~~~~~
|#\n##########s. `\\=====/  .s###########     #|___________`-~~~~#~-
`___________|#\n###########|\'.__```__.\'#############     #|############_-====-_
###########|#\n###########|=|###`##################     #|##########/`    .
`\\##########|#\n############\\_\\######################     #|#########/   {-||-}
\\#########|#\n##############V######################     #|#########(   --||--
)#########|#\n####|``|############################     #|#########\\ =\\\\-\\/-//=
/#########|#\n####|  |  \\V/  _ \\\\____ \\V/ __ \\####     #|##########\\.  ====
./##########|#\n####|   Y  |  (_) |  |_) |  ___/####     #|###########`-______-
`##########|#\n####|___|__/\\____/|  ___/A\\____)####
#|################################|#\n##################|_|###############
#|################################|#\n####################################
###################################\n|================================|
|================================|\n|`                              `|     |`
`|\n|    THE STATE OF RHODE ISLAND     |     |  THE UNIVERSITY OF RHODE ISLAND
|\n|             ~ 1636 ~             |     |             ~ 1892 ~
|\n|.                               .|     |.
.|\n|================================|
|================================|\n####################################
###################################\n-->\n\n</body>\n</html>\n'
```

But we do not need to manually write search tools, that's what **BeautifulSoup** is for.

```
cs_people = BeautifulSoup(cs_people_html,'html.parser')
```

In this object we can use any tag from the file and get back the first instance

```
cs_people.a
```

```
<a class="skip-link screen-reader-text" href="#content">Skip to content</a>
```

```
cs_people.h3
```

```
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/marco-alvarez/">Marco
Alvarez</a></h3>
```

More helpful is the `find_all` method we wnat to find all `div` tags that are "peopleitem" class. We decided this by inspecting the code on the website.

```
cs_people.find_all("div","peopleitem")
```

```
[<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/marco-alvarez/"><img alt="" class="u-photo
wp-post-image" height="120" loading="lazy"
src="https://web.uri.edu/cs/files/marco-alvarez.png" width="120"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/marco-alvarez/">Marco
Alvarez</a></h3>
```

```
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Associate Professor | Director of Graduate
Studies</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><span class="p-tel">401.874.5009</span> – <a class="u-
email" href="mailto:malvarez@uri.edu">malvarez@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card">
 <header>
 <div class="header">
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/samantha-
armenti/">Samantha Armenti</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Teaching Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email" href="mailto:sarmenti@uri.edu
">sarmenti@uri.edu </a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/sarah-brown/"><img alt="" class="u-photo wp-
post-image" height="300" loading="lazy" sizes="(max-width: 300px) 100vw, 300px"
src="https://web.uri.edu/cs/files/Sarah-Brown.png"
srcset="https://web.uri.edu/cs/files/Sarah-Brown.png 300w,
https://web.uri.edu/cs/files/Sarah-Brown-150x150.png 150w" width="300"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/sarah-brown/">Sarah
Brown</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email"
href="mailto:brownsarahm@uri.edu">brownsarahm@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/michael-conti/"><img alt="" class="u-photo
wp-post-image" height="2475" loading="lazy" sizes="(max-width: 2560px) 100vw,
2560px" src="https://web.uri.edu/cs/files/mike-conti-scaled.jpg"
srcset="https://web.uri.edu/cs/files/mike-conti-scaled.jpg 2560w,
https://web.uri.edu/cs/files/mike-conti-300x290.jpg 300w,
https://web.uri.edu/cs/files/mike-conti-1024x990.jpg 1024w,
https://web.uri.edu/cs/files/mike-conti-768x743.jpg 768w,
https://web.uri.edu/cs/files/mike-conti-1536x1485.jpg 1536w,
https://web.uri.edu/cs/files/mike-conti-2048x1980.jpg 2048w,
https://web.uri.edu/cs/files/mike-conti-364x352.jpg 364w,
https://web.uri.edu/cs/files/mike-conti-500x483.jpg 500w,
https://web.uri.edu/cs/files/mike-conti-1000x967.jpg 1000w,
https://web.uri.edu/cs/files/mike-conti-1280x1238.jpg 1280w,
https://web.uri.edu/cs/files/mike-conti-2000x1934.jpg 2000w" width="2560"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/michael-conti/">Michael
Conti</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Teaching Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email" href="mailto:michaelconti@uri.edu
">michaelconti@uri.edu </a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/noah-daniels/"><img alt="" class="u-photo
wp-post-image" height="219" loading="lazy" sizes="(max-width: 219px) 100vw, 219px"
src="https://web.uri.edu/cs/files/noah-daniels-web.png"
srcset="https://web.uri.edu/cs/files/noah-daniels-web.png 219w,
https://web.uri.edu/cs/files/noah-daniels-web-150x150.png 150w" width="219"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/noah-daniels/">Noah
Daniels</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email"
href="mailto:noah_daniels@uri.edu">noah_daniels@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/lisa-dipippo/"><img alt="" class="u-photo
wp-post-image" height="126" loading="lazy" src="https://web.uri.edu/cs/files/lisa-
dipippo-web.jpg" width="125"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/lisa-dipippo/">Lisa
DiPippo</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Professor | Chair</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email"
href="mailto:ldipippo@uri.edu">ldipippo@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/victor-fay-wolfe/"><img alt="" class="u-
photo wp-post-image" height="125" loading="lazy"
src="https://web.uri.edu/cs/files/victor-fay-wolfe-web.jpg" width="125"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/victor-fay-wolfe/">Victor
Fay-Wolfe</a></h3>
```

```
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email"
href="mailto:vfaywolfe@uri.edu">vfaywolfe@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card">
 <header>
 <div class="header">
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/lutz-hamel/">Lutz
Hamel</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Associate Professor </p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email"
href="mailto:lutzhamel@uri.edu">lutzhamel@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/abdeltawab-hendawi/"><img alt="" class="u-
photo wp-post-image" height="885" loading="lazy" sizes="(max-width: 1000px) 100vw,
1000px" src="https://web.uri.edu/cs/files/0-1.jpeg"
srcset="https://web.uri.edu/cs/files/0-1.jpeg 1000w,
https://web.uri.edu/cs/files/0-1-300x266.jpeg 300w,
https://web.uri.edu/cs/files/0-1-768x680.jpeg 768w,
https://web.uri.edu/cs/files/0-1-364x322.jpeg 364w,
https://web.uri.edu/cs/files/0-1-500x443.jpeg 500w" width="1000"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/abdeltawab-
hendawi/">Abdeltawab Hendawi</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Professor</p>
 <p class="people-department">Data Science | Computer Science</p>
 <p class="people-misc"><span class="p-tel">401.874.5738</span> – <a class="u-
email" href="mailto:hendawi@uri.edu">hendawi@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/jean-yves-herve/"><img alt="" class="u-photo
wp-post-image" height="299" loading="lazy" sizes="(max-width: 300px) 100vw, 300px"
src="https://web.uri.edu/cs/files/jean-yves-herve-web.jpg"
srcset="https://web.uri.edu/cs/files/jean-yves-herve-web.jpg 300w,
https://web.uri.edu/cs/files/jean-yves-herve-web-150x150.jpg 150w" width="300"/>
</a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/jean-yves-herve/">Jean-
Yves Hervé</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Associate Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email"
href="mailto:jyh@cs.uri.edu">jyh@cs.uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/natallia-katenka/"><img alt="" class="u-
photo wp-post-image" height="200" loading="lazy" sizes="(max-width: 200px) 100vw,
200px" src="https://web.uri.edu/cs/files/natalia-katenka-web.jpg"
srcset="https://web.uri.edu/cs/files/natalia-katenka-web.jpg 200w,
https://web.uri.edu/cs/files/natalia-katenka-web-150x150.jpg 150w" width="200"/>
</a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/natallia-
katenka/">Natallia Katenka</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Associate Professor | Director of
Undergraduate Studies</p>
 <p class="people-department">Statistics</p>
 <p class="people-misc"><a class="u-email"
href="mailto:nkatenka@uri.edu">nkatenka@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card">
 <header>
 <div class="header">
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/soheyb-kouider/">Soheyb
Kouider</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Associate Teaching Professor</p>
 <p class="people-department">Statistics</p>
 <p class="people-misc"><span class="p-tel">401.874.2562</span> – <a class="u-
email" href="mailto:soheyb@uri.edu">soheyb@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/edmund-lamagna/"><img alt="" class="u-photo
wp-post-image" height="150" loading="lazy"
src="https://web.uri.edu/cs/files/edmund-lamagna-web.jpg" width="150"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/edmund-lamagna/">Edmund
Lamagna</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email"
href="mailto:eal@cs.uri.edu">eal@cs.uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
```

```
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/indrani-mandal/"><img alt="" class="u-photo
wp-post-image" height="280" loading="lazy" sizes="(max-width: 278px) 100vw, 278px"
src="https://web.uri.edu/cs/files/0.jpeg"
srcset="https://web.uri.edu/cs/files/0.jpeg 278w, https://web.uri.edu/cs/files/0-
150x150.jpeg 150w" width="278"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/indrani-mandal/">Indrani
Mandal</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Teaching Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email" href="mailto:indrani_mandal@uri.edu
">indrani_mandal@uri.edu </a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/gavino-puggioni/"><img alt="" class="u-photo
wp-post-image" height="1600" loading="lazy" sizes="(max-width: 1600px) 100vw,
1600px" src="https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381.jpg"
srcset="https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381.jpg 1600w,
https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381-300x300.jpg 300w,
https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381-1024x1024.jpg 1024w,
https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381-150x150.jpg 150w,
https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381-768x768.jpg 768w,
https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381-1536x1536.jpg 1536w,
https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381-364x364.jpg 364w,
https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381-500x500.jpg 500w,
https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381-1000x1000.jpg 1000w,
https://web.uri.edu/cs/files/Gavino_headshot1-e1610666088381-1280x1280.jpg 1280w"
width="1600"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/gavino-puggioni/">Gavino
Puggioni</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Associate Professor |  Statistics Section
Head | Director of Graduate Studies</p>
 <p class="people-department">Statistics</p>
 <p class="people-misc"><span class="p-tel">401.874.4388</span> – <a class="u-
email" href="mailto:gpuggioni@uri.edu">gpuggioni@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card">
 <header>
 <div class="header">
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/jonathan-
schrader/">Jonathan Schrader</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Teaching Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email"
href="mailto:jonathan.schrader@uri.edu">jonathan.schrader@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/krishna-venkatasubramanian/"><img alt=""
class="u-photo wp-post-image" height="2703" loading="lazy" sizes="(max-width:
2437px) 100vw, 2437px" src="https://web.uri.edu/cs/files/KV_biopic_2.jpg"
srcset="https://web.uri.edu/cs/files/KV_biopic_2.jpg 2437w,
https://web.uri.edu/cs/files/KV_biopic_2-270x300.jpg 270w,
https://web.uri.edu/cs/files/KV_biopic_2-768x852.jpg 768w,
https://web.uri.edu/cs/files/KV_biopic_2-923x1024.jpg 923w,
https://web.uri.edu/cs/files/KV_biopic_2-364x404.jpg 364w,
https://web.uri.edu/cs/files/KV_biopic_2-500x555.jpg 500w,
https://web.uri.edu/cs/files/KV_biopic_2-1000x1109.jpg 1000w,
https://web.uri.edu/cs/files/KV_biopic_2-1280x1420.jpg 1280w,
https://web.uri.edu/cs/files/KV_biopic_2-2000x2218.jpg 2000w" width="2437"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/krishna-
venkatasubramanian/">Krishna Venkatasubramanian</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email"
href="mailto:krish@uri.edu">krish@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/jing-wu/"><img alt="" class="u-photo wp-
post-image" height="200" loading="lazy" sizes="(max-width: 200px) 100vw, 200px"
src="https://web.uri.edu/cs/files/jing-wu-web.jpg"
srcset="https://web.uri.edu/cs/files/jing-wu-web.jpg 200w,
https://web.uri.edu/cs/files/jing-wu-web-150x150.jpg 150w" width="200"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/jing-wu/">Jing Wu</a>
</h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Associate Professor</p>
 <p class="people-department">Statistics</p>
 <p class="people-misc"><span class="p-tel">401.874.4504</span> – <a class="u-
email" href="mailto:jing_wu@uri.edu">jing_wu@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/yichi-zhang/"><img alt="" class="u-photo wp-
post-image" height="240" loading="lazy" sizes="(max-width: 240px) 100vw, 240px"
src="https://web.uri.edu/cs/files/yichi-zhang-web.jpg"
srcset="https://web.uri.edu/cs/files/yichi-zhang-web.jpg 240w,
https://web.uri.edu/cs/files/yichi-zhang-web-150x150.jpg 150w" width="240"/></a>
 </figure>
```

```
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/yichi-zhang/">Yichi
Zhang</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Professor </p>
 <p class="people-department">Statistics</p>
 <p class="people-misc"><a class="u-email"
href="mailto:yichizhang@uri.edu">yichizhang@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/guangyu-zhu/"><img alt="" class="u-photo wp-
post-image" height="200" loading="lazy" sizes="(max-width: 200px) 100vw, 200px"
src="https://web.uri.edu/cs/files/Guangyu-Zhu-web.jpg"
srcset="https://web.uri.edu/cs/files/Guangyu-Zhu-web.jpg 200w,
https://web.uri.edu/cs/files/Guangyu-Zhu-web-150x150.jpg 150w" width="200"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/guangyu-zhu/">Guangyu
Zhu</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Professor</p>
 <p class="people-department">Statistics</p>
 <p class="people-misc"><a class="u-email"
href="mailto:guangyuzhu@uri.edu">guangyuzhu@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/ashley-buchanan/"><img alt="" class="u-photo
wp-post-image" height="966" loading="lazy" sizes="(max-width: 966px) 100vw, 966px"
src="https://web.uri.edu/cs/files/Ashley-Buchanan.jpg"
srcset="https://web.uri.edu/cs/files/Ashley-Buchanan.jpg 966w,
https://web.uri.edu/cs/files/Ashley-Buchanan-300x300.jpg 300w,
https://web.uri.edu/cs/files/Ashley-Buchanan-150x150.jpg 150w,
https://web.uri.edu/cs/files/Ashley-Buchanan-768x768.jpg 768w,
https://web.uri.edu/cs/files/Ashley-Buchanan-364x364.jpg 364w,
https://web.uri.edu/cs/files/Ashley-Buchanan-500x500.jpg 500w" width="966"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/ashley-buchanan/">Ashley
Buchanan</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Limited Joint Appointment</p>
 <p class="people-department">Biostatistics</p>
 <p class="people-misc"><span class="p-tel">401.874.4739</span> – <a class="u-
email" href="mailto:buchanan@uri.edu">buchanan@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/nina-kajiji/"><img alt="" class="u-photo wp-
post-image" height="125" loading="lazy" src="https://web.uri.edu/cs/files/nina-
kajiji-web.jpg" width="125"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/nina-kajiji/">Nina
Kajiji</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Adjunct Associate Professor</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><a class="u-email"
href="mailto:nina@uri.edu">nina@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/rachel-schwartz/"><img alt="" class="u-photo
wp-post-image" height="120" loading="lazy"
src="https://web.uri.edu/cs/files/Rachel-Schwartz-web.jpg" width="120"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/rachel-schwartz/">Rachel
Schwartz</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Professor – Limited Joint
Appointment</p>
 <p class="people-department">Biological Sciences</p>
 <p class="people-misc"><span class="p-tel">401.874.5404</span> – <a class="u-
email" href="mailto:rsschwartz@uri.edu">rsschwartz@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>,
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>
 <a href="https://web.uri.edu/cs/meet/ying-zhang/"><img alt="" class="u-photo wp-
post-image" height="250" loading="lazy" sizes="(max-width: 249px) 100vw, 249px"
src="https://web.uri.edu/cs/files/ying-zhang-web.jpg"
srcset="https://web.uri.edu/cs/files/ying-zhang-web.jpg 249w,
https://web.uri.edu/cs/files/ying-zhang-web-150x150.jpg 150w" width="249"/></a>
 </figure>
 <h3 class="p-name"><a href="https://web.uri.edu/cs/meet/ying-zhang/">Ying
Zhang</a></h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">Assistant Professor – Limited Joint
Appointment</p>
 <p class="people-department">Computer Science</p>
 <p class="people-misc"><span class="p-tel">401.874.4915</span> – <a class="u-
email" href="mailto:yingzhang@uri.edu">yingzhang@uri.edu</a></p>
 <div style="clear:both;"></div>
 </div>
 </div>]
```

this is a long, object and we can see it looks iterable ([at the start)

> **Important**
>
> answer to questions about searching from the docs

We could search all tags for the class attribute like this:

```
cs_people.find_all(class="peopleitem")[0]
```

```
  Cell In [10], line 1
    cs_people.find_all(class="peopleitem")[0]
                       ^
SyntaxError: invalid syntax
```

We can also look at only the first instance

```
cs_people.find_all("peopleitem")[0]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In [11], line 1
----> 1 cs_people.find_all("peopleitem")[0]

IndexError: list index out of range
```

We notice that the name is inside a `<h3>` tag with class `p-name` and then inside an a tag after that. We also know from looking at the overall page that there are lots of other a tags, so we do not want to search all of those.

```
names = [n.a.string for n in cs_people.find_all("h3","p-name")]
names
```

```
['Marco Alvarez',
 'Samantha Armenti',
 'Sarah Brown',
 'Michael Conti',
 'Noah Daniels',
 'Lisa DiPippo',
 'Victor Fay-Wolfe',
 'Lutz Hamel',
 'Abdeltawab Hendawi',
 'Jean-Yves Hervé',
 'Natallia Katenka',
 'Soheyb Kouider',
 'Edmund Lamagna',
 'Indrani Mandal',
 'Gavino Puggioni',
 'Jonathan Schrader',
 'Krishna Venkatasubramanian',
 'Jing Wu',
 'Yichi Zhang',
 'Guangyu Zhu',
 'Ashley Buchanan',
 'Nina Kajiji',
 'Rachel Schwartz',
 'Ying Zhang']
```

How to pull out the titles for each person (eg Assitatn Teaching Professor, Associate Professor)

```
titles = [t.string for t in cs_people.find_all("p","people-title")]
```

```
titles
```

```
['Associate Professor | Director of Graduate Studies',
 'Assistant Teaching Professor',
 'Assistant Professor',
 'Assistant Teaching Professor',
 'Assistant Professor',
 'Professor | Chair',
 'Professor',
 'Associate Professor ',
 'Assistant Professor',
 'Associate Professor',
 'Associate Professor | Director of Undergraduate Studies',
 'Associate Teaching Professor',
 'Professor',
 'Assistant Teaching Professor',
 'Associate Professor |  Statistics Section Head | Director of Graduate Studies',
 'Assistant Teaching Professor',
 'Assistant Professor',
 'Associate Professor',
 'Assistant Professor ',
 'Assistant Professor',
 'Limited Joint Appointment',
 'Adjunct Associate Professor',
 'Assistant Professor – Limited Joint Appointment',
 'Assistant Professor – Limited Joint Appointment']
```

We can pull out two more things, the people-department indicates who is CS & who is Statistics.

```
disciplines = [d.string for d in cs_people.find_all("p",'people-department')]
emails = [e.string for e in cs_people.find_all("a",'u-email')]
```

```
css_fac_df = pd.DataFrame({'name':names, 'title':titles,'e-mails':emails,
'discipline':disciplines})
css_fac_df
```

	name	title	e-mails	discipline
0	Marco Alvarez	Associate Professor | Director of Graduate Stu...	malvarez@uri.edu	Computer Science
1	Samantha Armenti	Assistant Teaching Professor	sarmenti@uri.edu	Computer Science
2	Sarah Brown	Assistant Professor	brownsarahm@uri.edu	Computer Science
3	Michael Conti	Assistant Teaching Professor	michaelconti@uri.edu	Computer Science
4	Noah Daniels	Assistant Professor	noah_daniels@uri.edu	Computer Science
5	Lisa DiPippo	Professor | Chair	ldipippo@uri.edu	Computer Science
6	Victor Fay-Wolfe	Professor	vfaywolfe@uri.edu	Computer Science
7	Lutz Hamel	Associate Professor	lutzhamel@uri.edu	Computer Science
8	Abdeltawab Hendawi	Assistant Professor	hendawi@uri.edu	Data Science | Computer Science
9	Jean-Yves Hervé	Associate Professor	jyh@cs.uri.edu	Computer Science
10	Natallia Katenka	Associate Professor | Director of Undergraduat...	nkatenka@uri.edu	Statistics
11	Soheyb Kouider	Associate Teaching Professor	soheyb@uri.edu	Statistics
12	Edmund Lamagna	Professor	eal@cs.uri.edu	Computer Science
13	Indrani Mandal	Assistant Teaching Professor	indrani_mandal@uri.edu	Computer Science
14	Gavino Puggioni	Associate Professor | Statistics Section Head...	gpuggioni@uri.edu	Statistics
15	Jonathan Schrader	Assistant Teaching Professor	jonathan.schrader@uri.edu	Computer Science
16	Krishna Venkatasubramanian	Assistant Professor	krish@uri.edu	Computer Science
17	Jing Wu	Associate Professor	jing_wu@uri.edu	Statistics
18	Yichi Zhang	Assistant Professor	yichizhang@uri.edu	Statistics
19	Guangyu Zhu	Assistant Professor	guangyuzhu@uri.edu	Statistics
20	Ashley Buchanan	Limited Joint Appointment	buchanan@uri.edu	Biostatistics
21	Nina Kajiji	Adjunct Associate Professor	nina@uri.edu	Computer Science
22	Rachel Schwartz	Assistant Professor – Limited Joint Appointment	rsschwartz@uri.edu	Biological Sciences
23	Ying Zhang	Assistant Professor – Limited Joint Appointment	yingzhang@uri.edu	Computer Science

12.2.1. Do you have any recommendations for learning html/css for the purpose of webscraping?

For webscraping only, careful examination of the page for patterns will be mostly enough. Rely on the nesting that the inspect or an IDE does. The following simple facts about website design will mostly be enough:

- tags are written like `<tagname>`
- classes are used to group similar tags and written like `class = list of classes separated by spaces`
- ids are used to "name" specific instances of tags like `id=section-qa`
- the other values inside tags are called attributes (class, id, href are attributes)
- the `<a>` tag is for links and it written like `<a href="url">link text</a>`

I will look for some examples, though.

12.3. Is there a way to search just one person with that method?

We definitely can after pulling all the names, but the scraping is really for pulling the structure and reformatting batches of information more than finding a single thing. For finding one thing, there are better tools.

12.3.1. what all the little commands in the inspection of a website are for?

12.3.2. How do you know what to use to iterate through values in the HTML?

I knew I wanted the to look at each person, that was the information I wanted from the page, from visual inspection. Then I inspected on one of the boxes about a person and noticed that they were a `div` tag with `peopleitem` class. I actually decided just the way we did it in class.

In the card for one person, I found each piece of information I wanted and then looked for the closest tag. I also tried things that did not work before class to work out an example that would work, but whe you are learning new things, that is how it works. Letting the computer tell you that you made a mistake is totally fine! Try things and experiment. Errors are okay.

12.3.3. How to use the find_all feature for just the class name?

use it as a keyword parameter like `class=`

12.3.4. Can we get more practice webscraping and using the .find_all() function?

Try pulling additional information from this page (like the webpage for each person) and the information from that page.

12.3.5. Will making a DataFrame via web scraping be use the DataFrame constructor, or is there more to it?

Use the constructor !

13. Getting Data from Databases

> **Important**
>
> Your first portfolio check is due October 17th. Submitting something for this check is important. You can treat it like a draft to get feedback and then you will be able to revise for the next one to improve your work if needed.

13.1. What is a Database?

A common attitude in Data Science is:

> If your data fits in memory there is no advantage to putting it in a database: it will only be slower and more frustrating. — Hadley Wickham

Businesses and research organizations nearly always have too much data to feasibly work without a database. Instead, they use different tools which are designed to scale to very large amounts of data. These tools are largely databases like Snowflake or Google's BigQuery and distributed computing frameworks like Apache Spark.

> **Warning**
>
> We are going to focus on the case of getting data out of a Database so that you can use it and making sure you know what a Database is.
>
> You could spend a whole semester on databases:
>
> - CSC436 covers how to implement them in detail (recommended, but requires CSC212)
> - BAI456 only how to use them (counts for DS majors, but if you want to understand them deeper, the CSC one is recommended)

For the purpose of this class the key attributes of a database are:

- it is a collection of tables
- the data is accessed live from disk (not RAM)
- you send a query to the database to get the data (or your answer)

Databases can be designed in many different ways. For examples two popular ones.

- SQLite is optimized for transactional workloads, which means a high volume of requests that involving inserting or reading a couple things. This is good for eg a webserver.
- DuckDB is optimized for analytical workloads, which means a small number of requests that each require reading many records in the database. This is better for eg: data science

Experimenting with DuckDB is a way to earn construct level 3

13.2. Accessing a Database from Python

We will use pandas again, as well as the `request` module from the `urllib` package and `sqlite3`.

Off the shelf, pandas cannot read databased by default. We'll use the `sqlite3` library, but there are others, depending on the type of database.

```
import pandas as pd
from urllib import request
import sqlite3
```

First we need to download the database to work with it.

```
request.urlretrieve('https://github.com/rhodyprog4ds/rhodyds/raw/main/data/nba1819.db',
   'nba1819.db')
```

```
('nba1819.db', <http.client.HTTPMessage at 0x7f798904acd0>)
```

Next, we set up a connection, that links the the notebook to the database. To use it, we add a cursor.

```
conn = sqlite3.connect('nba1819.db')
cursor = conn.cursor()
```

We can use execute to pass SQL queries through the cursor to the database.

```
cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
```

```
<sqlite3.Cursor at 0x7f79890b4a40>
```

Then we use `fetchall` to get the the results of the query.

```
cursor.fetchall()
```

```
[('teams',),
 ('conferences',),
 ('playerGameStats2018',),
 ('playerGameStats2019',),
 ('teamGameStats2018',),
 ('teamGameStats2019',),
 ('playerTeams2018',),
 ('playerTeams2019',),
 ('teamDailyRankings2018',),
 ('teamDailyRankings2019',),
 ('playerNames',)]
```

If we fetch again, there is nothing to fetch. Fetch pulls what was queued by execute.

```
cursor.fetchall()
```

```
[]
```

We can run another query with execute then fetch that result. This query gives us the column names.

The schema of a database is the description of its setup and layout. The * means to get all.

```
cursor.execute("SELECT * FROM INFORMATION_SCHEMA.COLUMNS ")
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
Cell In [7], line 1
----> 1 cursor.execute("SELECT * FROM INFORMATION_SCHEMA.COLUMNS ")

OperationalError: no such table: INFORMATION_SCHEMA.COLUMNS
```

Then we use `fetchall` to get the the results of the query.

```
cursor.fetchall()
```

```
[]
```

13.3. Querying with pandas

We can use `pd.read_sql` to send queries, get the result sand transform them into a DataFrame all at once

```
pd.read_sql("SELECT * FROM teams",conn)
```

	index	LEAGUE_ID	TEAM_ID	MIN_YEAR	MAX_YEAR	ABBREVIATION	NICKNAME	YEARFOUNDED	CITY	ARENA	ARENACAPACITY	OWNE
0	0	0	1610612737	1949	2019	ATL	Hawks	1949	Atlanta	State Farm Arena	18729.0	Tony Ressl
1	1	0	1610612738	1946	2019	BOS	Celtics	1946	Boston	TD Garden	18624.0	W Grousbe
2	2	0	1610612740	2002	2019	NOP	Pelicans	2002	New Orleans	Smoothie King Center	NaN	Tom Bensi
3	3	0	1610612741	1966	2019	CHI	Bulls	1966	Chicago	United Center	21711.0	Jer Reinsd
4	4	0	1610612742	1980	2019	DAL	Mavericks	1980	Dallas	American Airlines Center	19200.0	Mark Cub
5	5	0	1610612743	1976	2019	DEN	Nuggets	1976	Denver	Pepsi Center	19099.0	Stan Kroen
6	6	0	1610612745	1967	2019	HOU	Rockets	1967	Houston	Toyota Center	18104.0	Tilm Fertil
7	7	0	1610612746	1970	2019	LAC	Clippers	1970	Los Angeles	Staples Center	19060.0	Steve Ballm
8	8	0	1610612747	1948	2019	LAL	Lakers	1948	Los Angeles	Staples Center	19060.0	Jerry Bu Family Tru
9	9	0	1610612748	1988	2019	MIA	Heat	1988	Miami	AmericanAirlines Arena	19600.0	Micky Aris
10	10	0	1610612749	1968	2019	MIL	Bucks	1968	Milwaukee	Fiserv Forum	17500.0	Wesl Ede & Marc Las
11	11	0	1610612750	1989	2019	MIN	Timberwolves	1989	Minnesota	Target Center	19356.0	Glen Tayl
12	12	0	1610612751	1976	2019	BKN	Nets	1976	Brooklyn	Barclays Center	NaN	Joe Ts
13	13	0	1610612752	1946	2019	NYK	Knicks	1946	New York	Madison Square Garden	19763.0	Cablevisi (Jam Dola
14	14	0	1610612753	1989	2019	ORL	Magic	1989	Orlando	Amway Center	0.0	Rick DeV
15	15	0	1610612754	1976	2019	IND	Pacers	1976	Indiana	Bankers Life Fieldhouse	18345.0	Herb Sim
16	16	0	1610612755	1949	2019	PHI	76ers	1949	Philadelphia	Wells Fargo Center	NaN	Joshua Har
17	17	0	1610612756	1968	2019	PHX	Suns	1968	Phoenix	Talking Stick Resort Arena	NaN	Robert Sarv
18	18	0	1610612757	1970	2019	POR	Trail Blazers	1970	Portland	Moda Center	19980.0	Paul All
19	19	0	1610612758	1948	2019	SAC	Kings	1948	Sacramento	Golden 1 Center	17500.0	Viv Ranadi
20	20	0	1610612759	1976	2019	SAS	Spurs	1976	San Antonio	AT&T Center	18694.0	Peter H
21	21	0	1610612760	1967	2019	OKC	Thunder	1967	Oklahoma City	Chesapeake Energy Arena	19163.0	Clay Benn
22	22	0	1610612761	1995	2019	TOR	Raptors	1995	Toronto	Scotiabank Arena	19800.0	Maple Le Sports a Entertainme
23	23	0	1610612762	1974	2019	UTA	Jazz	1974	Utah	Vivint Smart Home Arena	20148.0	Greg Mill
24	24	0	1610612763	1995	2019	MEM	Grizzlies	1995	Memphis	FedExForum	18119.0	Robert Pe
25	25	0	1610612764	1961	2019	WAS	Wizards	1961	Washington	Capital One Arena	20647.0	Ted Leons
26	26	0	1610612765	1948	2019	DET	Pistons	1948	Detroit	Little Caesars Arena	21000.0	Tom Gor
27	27	0	1610612766	1988	2019	CHA	Hornets	1988	Charlotte	Spectrum Center	19026.0	Micha Jord
28	28	0	1610612739	1970	2019	CLE	Cavaliers	1970	Cleveland	Quicken Loans Arena	20562.0	Dan Gilb
29	29	0	1610612744	1946	2019	GSW	Warriors	1946	Golden State	Chase Center	19596.0	Joe Lac

We can use * to get all of the columns and `LIMIT` to reduce the number of rows.

```
pd.read_sql(con=conn,sql="SELECT * FROM teams LIMIT 3")
```

	index	LEAGUE_ID	TEAM_ID	MIN_YEAR	MAX_YEAR	ABBREVIATION	NICKNAME	YEARFOUNDED	CITY	ARENA	ARENACAPACITY	OWNER	GENERALM
0	0	0	1610612737	1949	2019	ATL	Hawks	1949	Atlanta	State Farm Arena	18729.0	Tony Ressler	Travis
1	1	0	1610612738	1946	2019	BOS	Celtics	1946	Boston	TD Garden	18624.0	Wyc Grousbeck	Dar
2	2	0	1610612740	2002	2019	NOP	Pelicans	2002	New Orleans	Smoothie King Center	NaN	Tom Benson	Trajan

13.3.1. Which player was traded the most during the 2018 season? How many times?

Note that the NBA Data is a little complicated for the questions we were asking because there is one row in players per team a played for per season. SO if a player was traded (changed teams), they are in there multiple times.

```
pd.read_sql("SELECT * FROM playerTeams2018 LIMIT 1",conn)
```

	index	TEAM_ID	PLAYER_ID
0	0	1610612761	202695

```
p18 =pd.read_sql("SELECT PLAYER_ID FROM playerTeams2018 ",conn)
```

Then we can use value counts

```
p18.value_counts().sort_values(ascending=False).head(10)
```

```
PLAYER_ID
1629150    4
202325     3
203092     3
201160     3
202328     3
1626150    3
1628393    3
202083     3
202692     3
203477     3
dtype: int64
```

and we can get the player's name from the player name **remember our first query told us all the tables**

```
pd.read_sql("SELECT PLAYER_NAME FROM playerNames WHERE PLAYER_ID = 1629150",conn)
```

	PLAYER_NAME
0	Emanuel Terry

13.3.2. Using multiple merges

In the NBA, there are 30 teams organized into two conferences: East and West; the `conferences` table has the columns `TEAM_ID` and `CONFERENCE`

build a Dataframe that could answer the question:

Did more players who changed teams from the 2018 season to the 2019 season stay in the same conferences or switch conferences?

> **Warning**
>
> I am going to fill this in later: get notified by commenting on the issue

```
# get conferences

# get 2018 players

# add conferences to 2018 players wiht a left merge on teams


# get 2019 players

# add conference

# inner merge the two tables and add suffixes to conference

# subset the dataframe with only rows where 18 & 19 conference are diff
```

> **Important**
>
> Remmber you have a choice to focus on web scraping *or* databases for assignment 5.
> You can then do the other or learn more about the same one for your portfolio.

13.4. Questions After class

13.4.1. What other SQL 'keywords' in the queries are there? ex: SELECT, FROM, WHERE

quick reference

13.4.2. what will you talk about on Monday

13.4.3. Is it viable to utilize sqlite for all big datasets? For example genomics data?

I am going to ask around more about the common formats for genomics data. I know that sometimes it is provided in plain text files, but it is not used by loading it all into RAM, for example this lesson works with the raw genomics data using bash scripts.

13.4.4. How to merge data together with sqlite?

You can use the same *types* of merges we have seen directly in the database by passign them as your query.

13.4.5. Do you need to put SQL code in caps?

> **Important**
>
> I have put one hot encoding in my notes to come back to it in two weeks during classificaiton, but we are giving you credit on A4 even if that is missed. I covered the apply and splitting a column differntly this year that led to fewer of you getting *very* confused, but also meant we did not get to one hot encoding, but I missed that it was still in a4.

14. Intro to Machine Learning: Evaluation

This week we are going to start learning about machine learning.

We are going to do this by looking at how to tell if machine learning has worked.

This is because:

- you have to check if one worked after you build one
- if you do not check carefully, it might only sometimes work
- gives you a chance to learn *only* evaluation instead of evaluation + an ML task

We are going to do this by auditing an algorithm that was built with machine learning.

14.1. What is ML?

First, what is an Algorithm?

> An algorithm is a set of ordered steps to complete a task.

Note that when people outside of CS talk about algorithms that impact people's lives these are often *not written directly by people* anymore. THey are often the result of machine learning.

In machine learning, people write an algorithm for how to write an algorithm based on data. This often comes in the form of a statitistical model of some sort

When we *do* machine learning, this can also be called:

- data mining
- pattern recognition
- modeling

because we are looking for patterns in the data and typically then planning to use those patterns to make predictions or automate a task.

Each of these terms does have slightly different meanings and usage, but sometimes they're used close to exchangeably.

14.2. Evaluating Algorithms: Propublica's COMPAS Audit

We are going to replicate the audit from ProPublica Machine Bias

Propublica started the COMPAS Debate with the article Machine Bias. With their article, they also released details of their methodology and their data and code. This presents a real data set that can be used for research on how data is used in a criminal justice setting without researchers having to perform their own requests for information, so it has been used and reused a lot of times.

14.3. Propublica COMPAS Data

The dataset consists of COMPAS scores assigned to defendants over two years 2013-2014 in Broward County, Florida, it was released by Propublica in a GitHub Repository. These scores are determined by a proprietary algorithm designed to evaluate a persons recidivism risk - the likelihood that they will reoffend. Risk scoring algorithms are widely used by judges to inform their sentencing and bail decisions in the criminal justice system in the United States.

The journalists collected, for each person arreste din 2013 and 2014:

- basic demographics
- details about what they were charged with and priors
- the COMPAS score assigned to them
- if they had actually been re-arrested within 2 years of their arrest

This means that we have what the COMPAS algorithm predicted (in the form of a score from 1-10) and what actually happened (re-arrested or not). We can then measure how well the algorithm worked, in practice, in the real world.

```
import pandas as pd
from sklearn import metrics
import seaborn as sns
```

We're going to work with a cleaned copy of the data released by Propublica that also has a minimal subset of features.

```
compas_clean_url = 'https://raw.githubusercontent.com/ml4sts/outreach-
compas/main/data/compas_c.csv'
compas_df = pd.read_csv(compas_clean_url,index_col = 'id')
```

```
compas_df.head()
```

id	age	c_charge_degree	race	age_cat	score_text	sex	priors_count	days_b_screening_arrest	decile_score	is_recid	two_year_recid	c_jail_in	c_jail_out	l
3	34	F	African-American	25 - 45	Low	Male	0	-1.0	3	1	1	2013-01-26 03:45:27	2013-02-05 05:36:53	
4	24	F	African-American	Less than 25	Low	Male	4	-1.0	4	1	1	2013-04-13 04:58:34	2013-04-14 07:02:04	
8	41	F	Caucasian	25 - 45	Medium	Male	14	-1.0	6	1	1	2014-02-18 05:08:24	2014-02-24 12:18:30	
10	39	M	Caucasian	25 - 45	Low	Female	0	-1.0	1	0	0	2014-03-15 05:35:34	2014-03-18 04:28:46	
14	27	F	Caucasian	25 - 45	Low	Male	0	-1.0	4	0	0	2013-11-25 06:31:06	2013-11-26 08:26:57	

Here is an explanation of these features:

- `age`: defendant's age
- `c_charge_degree`: degree charged (Misdemeanor of Felony)
- `race`: defendant's race
- `age_cat`: defendant's age quantized in "less than 25", "25-45", or "over 45"
- `score_text`: COMPAS score: 'low'(1 to 5), 'medium' (5 to 7), and 'high' (8 to 10).
- `sex`: defendant's gender
- `priors_count`: number of prior charges
- `days_b_screening_arrest`: number of days between charge date and arrest where defendant was screened for compas score
- `decile_score`: COMPAS score from 1 to 10 (low risk to high risk)
- `is_recid`: if the defendant recidivized
- `two_year_recid`: if the defendant within two years
- `c_jail_in`: date defendant was imprisoned
- `c_jail_out`: date defendant was released from jail
- `length_of_stay`: length of jail stay

14.4. One-hot Encoding

We will audit first to see how good the algorithm is by treating the predictions as either high or not high. One way we can get to that point is to transform the `score_text` column from one column with three values, to 3 binary columns.

```
compas_df = pd.get_dummies(compas_df,columns=['score_text'])
```

```
compas_df.head()
```

id	age	c_charge_degree	race	age_cat	sex	priors_count	days_b_screening_arrest	decile_score	is_recid	two_year_recid	c_jail_in	c_jail_out	length_of_sta
3	34	F	African-American	25 - 45	Male	0	-1.0	3	1	1	2013-01-26 03:45:27	2013-02-05 05:36:53	1
4	24	F	African-American	Less than 25	Male	4	-1.0	4	1	1	2013-04-13 04:58:34	2013-04-14 07:02:04	
8	41	F	Caucasian	25 - 45	Male	14	-1.0	6	1	1	2014-02-18 05:08:24	2014-02-24 12:18:30	
10	39	M	Caucasian	25 - 45	Female	0	-1.0	1	0	0	2014-03-15 05:35:34	2014-03-18 04:28:46	
14	27	F	Caucasian	25 - 45	Male	0	-1.0	4	0	0	2013-11-25 06:31:06	2013-11-26 08:26:57	

Note the last 3 columns

14.5. Performance Metrics in sklearn

The first thing we usually check is the accuracy: the percentage of all samples that are correct.

```
metrics.accuracy_score(compas_df['two_year_recid'],compas_df['score_text_High'])
```

```
0.6288366805608185
```

However this does not tell us anything about what *types* of mistakes the algorithm made. The type of mistake often matters in terms of how we trust or deploy an algorithm. We use a confusion matrix to describe the performance in more detail.

> **Note**
>
> the wikipedia page for confusion matrix is a really good reference

A confusion matrix counts the number of samples of each *true* category that wre predicted to be in each category. In this case we have a binary prediction problem: people either are re-arrested (truth) or not and were given a high score or not(prediction). In binary problems we adopt a common language of labeling one outcome/predicted value positive and the other negative. We do this not based on the social value of the outcome, but on the numerical encoding.

In this data, being re-arrested is indicated by a 1 in the `two_year_recid` column, so this is the *positive class* and not being re-arrested is 0, so the *negative class*. Similarly a high score is 1, so that's the *positive prediction* and not high is 0, so that is the a *negative prediction*.

> **Note**
>
> these terms can be used in any sort of detection problem, whether machine learning is used or not

`sklearn.metrics` provides a `[confusion matrix](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)` function that we can use.

```
metrics.confusion_matrix(compas_df['two_year_recid'],compas_df['score_text_High'])
```

```
array([[2523,  272],
       [1687,  796]])
```

Since this is binary problem we have 4 possible outcomes:

- true negatives($C_{0,0}$): did not get a high score and were not re-arrested
- false negatives($C_{1,0}$):: did not get a high score and were re-arrested
- false positives($C_{0,1}$):: got a high score and were not re-arrested
- true positives($C_{1,1}$):: got a high score and were re-arrested

With these we can revisit accuracy:

$$A = \frac{C_{0,0} + C_{1,1}}{C_{0,0} + C_{1,0} + C_{0,1} + C_{1,1}}$$

and we can define new scores. Two common ones in CS are recall and precision.

Recall is:

$$R = \frac{C_{1,1}}{C_{1,0} + C_{1,1}}$$

```
metrics.recall_score(compas_df['two_year_recid'],compas_df['score_text_High'])
```

```
0.3205799436165928
```

That is, among the truly positive class how many were correctly predicted? In COMPAS, it's the percentage of the re-arrested people who got a high score.

Precision is $$P = \frac{C_{1,1}}{C_{0,1}+C_{1,1}}$$

```
metrics.precision_score(compas_df['two_year_recid'],compas_df['score_text_High'])
```

```
0.7453183520599251
```

That is, among the positive predictions, what percentage was correct. In COPMAS, that is among the people who got a high score, what percentage were re-arrested.

> **Important**
>
> Install install aif360 before class Friday

14.6. Questions after class

All were clarifying details that I expanded above.

15. Performance Metrics continued

15.1. Logistics

- A6 posted ASAP! (I promise it will be straightforward in terms of the actual code; you do need to carefully interpret though)
- a5 grading is behind, but if it's not done by early tomorrow (eg 10am) then I'll extend the portfolio deadline accordingly
- Mid Semester Feedback

15.2. Completing the COMPAS Audit

Let's start where we left off, plus some additional imports

```
import pandas as pd
from sklearn import metrics as skmetrics
from aif360 import metrics as fairmetrics
from aif360.datasets import BinaryLabelDataset
import seaborn as sns

compas_clean_url = 'https://raw.githubusercontent.com/ml4sts/outreach-
compas/main/data/compas_c.csv'
compas_df = pd.read_csv(compas_clean_url,index_col = 'id')

compas_df = pd.get_dummies(compas_df,columns=['score_text'],)
```

> **Note**
>
> I used a tag on this cell in the notes so that the output with warning is not shown here.

```
WARNING:root:No module named 'tempeh': LawSchoolGPADataset will be unavailable. To
install, run:
pip install 'aif360[LawSchoolGPA]'
```

```
WARNING:root:No module named 'tensorflow': AdversarialDebiasing will be
unavailable. To install, run:
pip install 'aif360[AdversarialDebiasing]'
```

```
WARNING:root:No module named 'tensorflow': AdversarialDebiasing will be
unavailable. To install, run:
pip install 'aif360[AdversarialDebiasing]'
```

```
WARNING:root:No module named 'fairlearn': ExponentiatedGradientReduction will be
unavailable. To install, run:
pip install 'aif360[Reductions]'
```

```
WARNING:root:No module named 'fairlearn': GridSearchReduction will be unavailable.
To install, run:
pip install 'aif360[Reductions]'
```

```
WARNING:root:No module named 'fairlearn': GridSearchReduction will be unavailable.
To install, run:
pip install 'aif360[Reductions]'
```

Warning

We'll get a warning which is **okay** but if you run again it will go away.

to review:

```
compas_df.head()
```

id	age	c_charge_degree	race	age_cat	sex	priors_count	days_b_screening_arrest	decile_score	is_recid	two_year_recid	c_jail_in	c_jail_out	length_of_sta
3	34	F	African-American	25 - 45	Male	0	-1.0	3	1	1	2013-01-26 03:45:27	2013-02-05 05:36:53	1
4	24	F	African-American	Less than 25	Male	4	-1.0	4	1	1	2013-04-13 04:58:34	2013-04-14 07:02:04	
8	41	F	Caucasian	25 - 45	Male	14	-1.0	6	1	1	2014-02-18 05:08:24	2014-02-24 12:18:30	
10	39	M	Caucasian	25 - 45	Female	0	-1.0	1	0	0	2014-03-15 05:35:34	2014-03-18 04:28:46	
14	27	F	Caucasian	25 - 45	Male	0	-1.0	4	0	0	2013-11-25 06:31:06	2013-11-26 08:26:57	

Notice today we imported the sklearn.metrics module with an alias.

```
skmetrics.accuracy_score(compas_df['two_year_recid'],compas_df['score_text_High'])
```

```
0.6288366805608185
```

More common is to use medium or high to check accuracy (or not low) we can calulate tihs by either summing two or inverting. We'll do it as not low for now, to review using apply.

Try it Yourself

A good exercise to review data manipulation is to try creating the `score_text_MedHigh` column by adding the other two together (because medium or high if they're booleans is the same as medium + high if they're ints)

```
int_not = lambda a:int(not(a))
compas_df['score_text_MedHigh'] = compas_df['score_text_Low'].apply(int_not)
skmetrics.accuracy_score(compas_df['two_year_recid'],
            compas_df['score_text_MedHigh'])
```

```
0.6582038651004168
```

We can see this gives us a slightly higher score, but still not that great.

the `int_not lambda` is a function:

```
type(int_not)
```

```
function
```

it is equivalent as the following, but a more compact notation.

```
def int_not_f(a):
    return int(not(a))
```

It flips a 0 to a 1

```
int_not(0)
```

```
1
```

and th eother way

```
int_not(1)
```

```
0
```

```
compas_df.groupby('score_text_Medium')['decile_score'].min()
```

```
score_text_Medium
0    1
1    5
Name: decile_score, dtype: int64
```

```
compas_race = compas_df.groupby('race')
```

15.3. Per Group scores with groupby

To groupby and then do the score, we can use a lambda again, with apply

```
acc_fx = lambda d: skmetrics.accuracy_score(d['two_year_recid'],
             d['score_text_MedHigh'])

compas_race.apply(acc_fx,)
```

```
race
African-American    0.649134
Caucasian           0.671897
dtype: float64
```

In this case it gives a series, but with `reset_index` we can make it a DataFrame and then rename the column to label it as accuracy.

```
compas_race.apply(acc_fx,).reset_index().rename(columns={0:'accuracy'})
```

	race	accuracy
0	African-American	0.649134
1	Caucasian	0.671897

That lambda + apply is equivalent to:

```
race_acc = []
for race, rdf in compas_race:
    acc = skmetrics.accuracy_score(rdf['two_year_recid'],
             rdf['score_text_MedHigh'])
    race_acc.append([race,acc])

pd.DataFrame(race_acc, columns =['race','accuracy'])
```

	race	accuracy
0	African-American	0.649134
1	Caucasian	0.671897

15.4. Using AIF360

The AIF360 package implements fairness metrics, some of which are derived from metrics we have seen and some others. the documentation has the full list in a summary table with English explanations and details with most equations.

However, it has a few requirements:

- its constructor takes two `BinaryLabelDataset` objects
- these objects must be the same except for the label column
- the constructor for `BinaryLabelDataset` only accepts all numerical DataFrames

So, we have some preparation to do.

First, we'll make a numerical copy of the `compas_df` columns that we need. The only nonnumerical column that we need is race, wo we'll make a `dict` to replace that/

```
race_num_map = {r:i for i,r, in enumerate(compas_df['race'].value_counts().index)}
race_num_map
```

```
{'African-American': 0, 'Caucasian': 1}
```

and here we select columns and replace the values

```
required_cols = ['race','two_year_recid','score_text_MedHigh']
num_compas = compas_df[required_cols].replace(race_num_map)
num_compas.head(2)
```

	race	two_year_recid	score_text_MedHigh
id			
3	0	1	0
4	0	1	0

Next we will make two versions, one with race & the ground truth and ht eother with race & the predictions. It's easiest to drop the column we don't want.

```
num_compas_true = num_compas.drop(columns=['score_text_MedHigh'])
num_compas_pred = num_compas.drop(columns=['two_year_recid'])
```

Now we make the **`BinaryLabelDataset`** objects, this type comes from AIF360 too. Basically, it is a DataFrame with extra attributes; some specific and some inherited from **`StructuredDataset`**.

```
# here we want actual favorable outcome
broward_true = BinaryLabelDataset(0,1,df = num_compas_true,
         label_names= ['two_year_recid'],
        protected_attribute_names=['race'])
compas_predictions = BinaryLabelDataset(0,1,df = num_compas_pred,
         label_names= ['score_text_MedHigh'],
        protected_attribute_names=['race'])
```

Try it Yourself

remember, you can inspect *any* object using the `__dict__` attribute

This type also has an `ignore_fields` column for when comparisons are made, since the requirement is that only the *content* of the label column is different, but in our case also the label names are different, we have to tell it that that's okay.

```
compas_predictions.ignore_fields.add('label_names')
broward_true.ignore_fields.add('label_names')
```

Now, we can instantiate our metric object:

```
compas_fair_scorer = fairmetrics.ClassificationMetric(broward_true,
                        compas_predictions,
                unprivileged_groups=[{'race':0}],
               privileged_groups = [{'race':1}])
```

And finally we can compute! First, we can verify that we get the same accuracy as before

```
compas_fair_scorer.accuracy()
```

```
0.6582038651004168
```

For the aif360 metrics, they have one parameter, `privleged` with a defautl value of `None` when it's none it computes th ewhole dataset. When `True` it compues only the priveleged group.

```
compas_fair_scorer.accuracy(True)
```

```
0.6718972895863052
```

Here that is Caucasion people.

When `False` it's the unpriveleged group, here African American

```
compas_fair_scorer.accuracy(False)
```

```
0.6491338582677165
```

We can also compute other scores. Many fairness scores are ratios of the un priveleged group's score to the privleged group's score.

In Disparate Impact the ratio is of the positive outcome, independent of the predictor. So this is the ratio of the % of Black people not rearrested to % of white people rearrested.

```
compas_fair_scorer.disparate_impact()
```

```
0.6336457196581771
```

The courts use an "80%" rule saying that if this ratio is above .8 for things like employment, it's close enough. T

```
compas_fair_scorer.error_rate_ratio()
```

```
1.0693789798014377
```

We can also do ratios of the scores. This is where the journalists found bias.

```
compas_fair_scorer.false_positive_rate_ratio()
```

```
0.5737241916634204
```

Black people were given a low score and then re-arrested only a little more than half as often as white people. (White people were give an low score and rearrested almost twice as often)

```
compas_fair_scorer.false_negative_rate_ratio()
```

```
1.9232342111919953
```

Black people were given a high score and not rearrested almost twice as often as white people.

So while the accuracy was similar (see error rate ratio) for Black and White people; the algorithm makes the opposite types of errors.

After the journalists published the piece, the people who made COMPAS countered with a technical report, arguing that that the journalists had measured fairness incorrectly.

The journalists two measures false positive rate and false negative rate use the true outcomes as the denominator.

The COMPAS creators argued that the model should be evaluated in terms of if a given score means the same thing across races; using the prediction as the denominator.

```
compas_fair_scorer.false_omission_rate_ratio()
```

```
0.8649767923408909
```

```
compas_fair_scorer.false_discovery_rate_ratio()
```

```
1.2118532033913119
```

On these two metrics, the ratio is closer to 1 and much less disparate.

The creators thought it was important for the score to mean the same thing for every person assigned a score. The journalists thought it was more important for the algorithm to have the same impact of different groups of people.
Ideally, we would like the score to both mean the same thing for different people and to have the same impact.

Researchers established that these are mutually exclusive, provably. We cannot have both, so it is very important to think about what the performance metrics mean and how your algorithm will be used in order to choose how to prepare a model. We will train models starting next week, but knowing these goals in advance is essential.

Importantly, this is not a statistical, computational choice that data can answer for us. This is about *human* values (and to some extent the law; certain domains have legal protections that require a specific condition).

The Fair Machine Learning book's classificaiton Chapter has a section on relationships between criteria with the proofs.

Important

We used ProPublica's COMPAS dataset to replicate (parts of, with different tools) their analysis. That is, they collected the dataset in order to audit the COMPAS algorithm and we used it for the same purpose (and to learn model evaluation). This dataset is not designed for *training* models, even though it has been used as such many times. This is not the best way to use this dataset and for future assignments I do not recommend using this dataset.

15.5. Portfolio Reminder

If you do not need level 3s to be happy with your grade for the course (eg you want a B) and you have all the achievements so far you can skip the portfolio submission. If you do not need level 3 and you are not on track, you should submit to get caught up. This can be (and is advised to be) reflective revisions of past assignment(s).

If you need level 3 achievements for your desired grade, then you can pick a subset of the eligible skills(or all) and add *new* work that shows that you have learned those skills according to the level 3 checklists. the ideas page has example formats for that new work.

Note

If you are interested in fairness in ML, that is what my research is. Aiden has been working on a project with me since summer and I'll be taking new students in the spring. Students who have completed this course are excellent candidates to join my lab. Let me know if you are interested!

15.6. Questions After Class

Today's questions were only clarifying, so hopefully re-reading the notes is enough. If not, post a question as an issue!

16. Modeling and Naive Bayes

Important

Remember to give feedback on the course so far. This feedback helps me make adjustments to the course if needed. Even if things are good for you right now, letting me know helps me keep emphasis on the things that are helping.

The form requires you to be logged into you URI Google account, but does not share your e-mail with me in the results. It is required so that people not at URI do not complete the form.

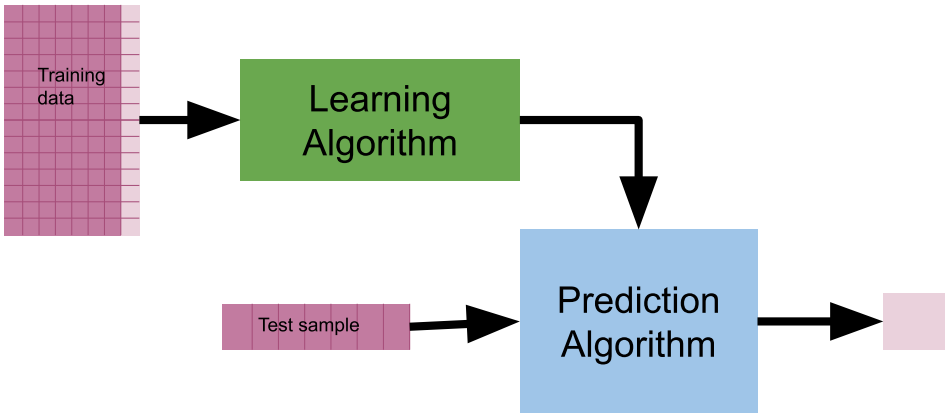
We're going to approach machine learning from the perspective of *modeling* for a few reasons:

- model based machine learning streamlines understanding the big picture
- the model way of interpreting it aligns well with using sklearn
- thinking in terms of models aligns with incorporating domain expertise, as in our data science definition

this paper by Christopher M. Bishop, a senior ML researcher who also wrote one of a the widely preferred graduate level ML textbooks, details advantages of a model based perspective and a more mathematical version of a model based approach to machine learning. He is a co-author on an introductory text book Model Based ML

In CSC461: Machine Learning, you can encounter an *algorithm* focused approach to machine learning, but I think having the model based perspective first helps you avoid common pitfalls.

Remmeber our overview of ML:



16.1. What is a Model?

A model is a simplified representation of some part of the world. A famous quote about models is:

> All models are wrong, but some are useful –George Box[^wiki]

In machine learning, we use models, that are generally *statistical* models.

A statistical model is a mathematical model that embodies a set of statistical assumptions concerning the generation of sample data (and similar data from a larger population). A statistical model represents, often in considerably idealized form, the data-generating process wikipedia

read more in theModel Based Machine Learning Book

16.2. Models in Machine Learning

Starting from a dataset, we first make an additional designation about how we will use the different variables (columns). We will call most of them the *features*, which we denote mathematically with $\mathbf{X}$ and we'll choose one to be the *target* or *labels*, denoted by $\mathbf{y}$.

The core assumption for just about all machine learning is that there exists some function f so that for the ith sample

$$y_i = f(\mathbf{x}_i)$$

i would be the index of a DataFrame

Example models are (informally):

- we can describe the data as a set of blobs
- we can describe the rule to separate classes as a flow chart
- we can describe the rule to separate as a curved line

16.3. Types of Machine Learning

Then with different additional assumptions we get different types of machine learning:

- if both features ($\mathbf{X}$) and target ($\mathbf{y}$) are observed (contained in our dataset) it's **supervised learning** code
- if only the features ($\mathbf{X}$) are observed, it's **unsupervised learning** code

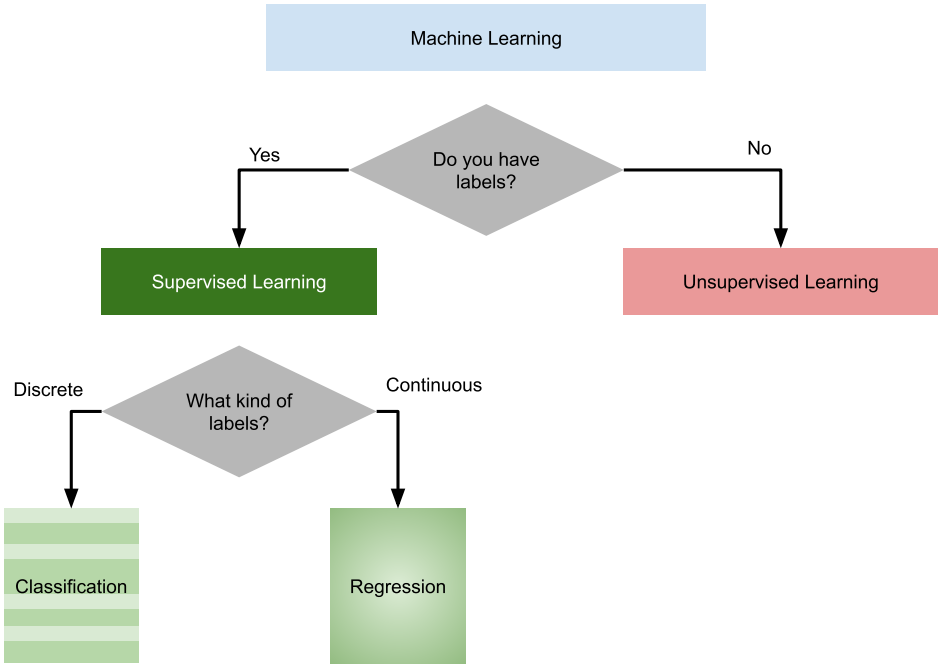
sup
–>

16.4. Supervised Learning

we'll focus on supervised learning first. we can take that same core assumption and use it with additional information about our target variable to determine learning **task** we are working to do.

$$y_i = f(\mathbf{x}_i)$$

- if y_i are discrete (eg flower species) we are doing **classification**
- if y_i are continuous (eg height) we are doing **regression**



Further Reading

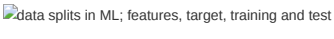
sklearn provides a popular flowchart for choosing a specific model

16.5. Machine Learning Pipeline

To do machine learning we start with **training data** which we put as input to the **learning algorithm**. A learning algorithm might be a generic optimization procedure or a specialized procedure for a specific model. The learning algorithm outputs a trained **model** or the parameters of the model. When we deploy a model we pair the **fit model** with a **prediction algorithm** or **decision** algorithm to evaluate a new sample in the world.

In experimenting and design, we need **testing data** to evaluate how well our learning algorithm understood the world. We need to use previously unseen data, because if we don't we can't tell if the prediction algorithm is using a rule that the learning algorithm produced or just looking up from a lookup table the result. This can be thought of like the difference between memorization and understanding.

When the model does well on the training data, but not on test data, we say that it does not generalize well.



16.6. Iris Dataset

```
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score
iris_df = sns.load_dataset('iris')
```

We're trying to build an automatic flower classifier that, for measurements of a new flower returns the predicted species. To do this, we have a DataFrame with columns for species, petal width, petal length, sepal length, and sepal width. The species is what type of flower it is the petal and sepal are parts of the flower.

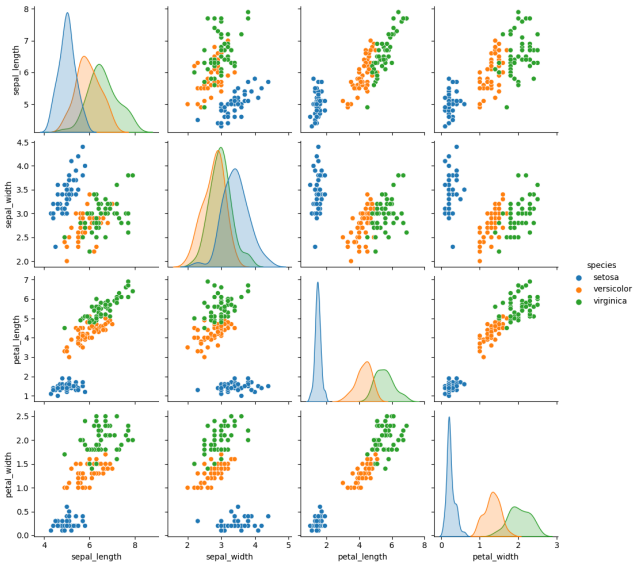
```
iris_df.head(1)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa

We can look at this data using a pair plot. It plots each pair of numerical variables in a grid of scatterplots and on the diagonal (where it would be a variable with itself) shows the distribution of that variable.

```
sns.pairplot(data=iris_df,hue='species')
```

```
<seaborn.axisgrid.PairGrid at 0x7fd9dd67f640>
```



This data is reasonably **separable** beacuse the different species (indicated with colors in the plot) do not overlap much. We see that the features are distributed sort of like a normal, or Gaussian, distribution. In 2D a Gaussian distribution is like a hill, so we expect to see more points near the center and fewer on the edge of circle-ish blobs. These blobs are slightly live ovals, but not too skew.

16.7. Creating test and train

To do machine learning, we split the data both sample wise (rows if tidy) and variable-wise (columns if tidy). First, we'll designate the columns to use as features and as the target.

The features are the input that we wish to use to predict the target.

```
feature_vars = ['sepal_length', 'sepal_width','petal_length', 'petal_width',]
target_var = 'species'
```

Next, we'll use a sklearn function to split the data randomly into test and train portions.

```
X_train, X_test, y_train, y_test =
train_test_split(iris_df[feature_vars],iris_df[target_var],random_state=0)
```

> **Note**
>
> Here i set the random state. This means that the site will always have the same result even when this notebook is run over and over again.
>
> Try downloading it (or adding `random_state` to your own code) and running it on your own computer.

We can see by default how many samples it puts in each set

```
X_train.shape
```

```
(112, 4)
```

```
X_test.shape
```

```
(38, 4)
```

We can also see that it picks a random subset by the index:

```
X_train.head()
```

	sepal_length	sepal_width	petal_length	petal_width
61	5.9	3.0	4.2	1.5
92	5.8	2.6	4.0	1.2
112	6.8	3.0	5.5	2.1
2	4.7	3.2	1.3	0.2
141	6.9	3.1	5.1	2.3

16.8. Instantiating our Model Object

This is the *model*. In `sklearn` they call these objects estimator. All estimators have a similar usage. First we instantiate the object and set any *hyperparameters*.

Instantiating the object says we are assuming a particular type of model. In this case Gaussian Naive Bayes. This sets several assumptions in one form:

- we assume data are Gaussian (normally) distributed
- the features are uncorrelated/independent (Naive)
- the best way to predict is to find the highest probability (Bayes)

this is one example of a Bayes Estimator

```
gnb = GaussianNB()
```

At this point the object is not very interesting

```
gnb.__dict__
```

```
{'priors': None, 'var_smoothing': 1e-09}
```

The fit method uses the data to learn the model's parameters. In this case, a Gaussian distribution is characterized by a mean and variance; so the GNB classifier is characterized by one mean and one variance for each class (in 4d, like our data)

```
gnb.fit(X_train,y_train)
```

```
▾ GaussianNB
GaussianNB()
```

The attributes of the estimator object (`gbn`) describe the data (eg the class list) and the model's parameters. The `theta_` (θ) represents the mean and the `sigma_` (σ) represents the variance of the distributions.

```
gnb.__dict__
```

```
{'priors': None,
 'var_smoothing': 1e-09,
 'classes_': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 'feature_names_in_': array(['sepal_length', 'sepal_width', 'petal_length',
'petal_width'],
       dtype=object),
 'n_features_in_': 4,
 'epsilon_': 3.2135586734693885e-09,
 'theta_': array([[4.9972973 , 3.38918919, 1.45405405, 0.24054054],
        [5.91764706, 2.75882353, 4.19117647, 1.30882353],
        [6.66341463, 2.9902439 , 5.58292683, 2.03902439]]),
 'var_': array([[0.12242513, 0.14474799, 0.01978087, 0.01159971],
        [0.2649827 , 0.11124568, 0.22139274, 0.0408045 ],
        [0.4071981 , 0.11453897, 0.30483046, 0.06579417]]),
 'class_count_': array([37., 34., 41.]),
 'class_prior_': array([0.33035714, 0.30357143, 0.36607143])}
```

Once we fit, we can predict

```
y_pred = gnb.predict(X_test)
y_pred
```

```
array(['virginica', 'versicolor', 'setosa', 'virginica', 'setosa',
       'virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'virginica', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'setosa', 'versicolor', 'versicolor', 'setosa',
       'setosa', 'virginica', 'versicolor', 'setosa', 'setosa',
       'virginica', 'setosa', 'setosa', 'versicolor', 'versicolor',
       'setosa', 'virginica', 'versicolor', 'setosa', 'virginica',
       'virginica', 'versicolor', 'setosa', 'versicolor'], dtype='<U10')
```

we get one prediciton for each sample.

Estimator objects also have a score method. If the estimator is a classifier, that score is accuracy. We will see that for other types of estimators it is different types.

```
gnb.score(X_test,y_test)
```

```
1.0
```

We can also compute a confusion matrix as we did last week (see those notes). Now it is 3x3 though because we have 3 species instead of two outcomes like the COMPAS predictions.

```
confusion_matrix(y_test,y_pred)
```

```
array([[13,  0,  0],
       [ 0, 16,  0],
       [ 0,  0,  9]])
```

16.9. Questions After Class

16.9.1. is there any extra material where we can learn a better understanding oh how to do the predictions and models

The Scikit Learn User Guide is a good place, as is the Model Based ML

16.9.2. Are there any good introductions to ScikitLearn that you are aware of?

Scikit Learn User Guide is the best one and they have a large example gallery.

16.9.3. What is a confusion matrix?

Notes from last wed go over that.

16.9.4. Can you use machine learning for any type of data?

Yes the features for example could be an image instead of four numbers. It could also be text. The basic ideas are the same for more complex data, so we are going to spend a lot of time building your understanding of what ML *is* on simple data. Past students have successfully applied ML in more complex data after this course because once you have a good understanding of the core ideas, applying it to other forms of data is easier to learn on your own.

16.9.5. Can we check how well the model did using the y_test df?

we could compare them directly or using `score` that does.

```
y_pred == y_test
```

Further Reading

All of this is beyond the scope of this course, but may be of interest The Scikit Learn User Guide is a really good place to learn the details of machine learning. It is high quality documentation from both a statistical and computer science perspective of every element of the library.

The sklearn API describes how the library is structured and organized. Because the library is so popular (and it's pretty well architected from a software perspective as well) if you are developing new machine learning techniques it's good to make them sklearn compatible.

For example, IBM's AIF360 is a package for doing fair machine learning which has a [sklearn compatible interface(https://aif360.readthedocs.io/en/latest/modules/sklea
Scikit Learn documentation also includes a related projects page.

```
114    True
62     True
33     True
107    True
7      True
100    True
40     True
86     True
76     True
71     True
134    True
51     True
73     True
54     True
63     True
37     True
78     True
90     True
45     True
16     True
121    True
66     True
24     True
8      True
126    True
22     True
44     True
97     True
93     True
26     True
137    True
84     True
27     True
127    True
132    True
59     True
18     True
83     True
Name: species, dtype: bool
```

```
sum(y_pred == y_test)/len(y_test)
```

```
1.0
```

```
gnb.score(X_test,y_test)
```

```
1.0
```

We can also use any of the other metrics we saw, we'll practice more on Wednesday

16.9.6. I want to know more about the the test_train_split() function

the docs are a good place to start.

16.9.7. Could we use the comparison stuff we've been learning to test the ML algorithm?

Yes!!

16.9.8. How should we set the random set we had for x_test to a specific group?

the **`random_state`** parameter will fix it.

> **Note**
>
> If I misunderstood this question, post an issue

16.9.9. Not a question but we're using the iris dataset in my programming with R class too!

The iris dataset is a very popular easy to use dataset. It is good for this case because it fits the criteria of a simple classifier. It comes from the UC Irvine Machine Learning Repository a large collection of good datasets for machine learning, you can learn more about it on its page there.

17. Classification with Naive Bayes

> **Note**
>
> This is typically not required but can fix issues. Mine was caused because I installed a package that reverted the version of my jupyter. This was the same problem that caused the notes to not render correctly.
>
> The solution to this is to use environments more carefully.

```
%matplotlib inline
```

We'll start with the same dataset as Monday.

```
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
iris_df = sns.load_dataset('iris')
```

```
# dataset vars:
# ,
feature_vars = ['petal_width', 'sepal_length', 'sepal_width','petal_length']
target_var = 'species'
X_train, X_test, y_train, y_test =
train_test_split(iris_df[feature_vars],iris_df[target_var],random_state=0)
```

Using the `random_state` makes it so that we get the same "random" set each type we run the code. see docs

```
X_train.head()
```

	petal_width	sepal_length	sepal_width	petal_length
61	1.5	5.9	3.0	4.2
92	1.2	5.8	2.6	4.0
112	2.1	6.8	3.0	5.5
2	0.2	4.7	3.2	1.3
141	2.3	6.9	3.1	5.1

```
iris_df.head(1)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa

```
X_train.shape, iris_df.shape
```

```
((112, 4), (150, 5))
```

```
112/150
```

```
0.7466666666666667
```

We can see that we get ~75% of the samples in the training set by default. We could also see this from the docstring.

Again we will instantiate the object.

```
gnb = GaussianNB()
```

```
gnb.__dict__
```

```
{'priors': None, 'var_smoothing': 1e-09}
```

```
gnb.fit(X_train,y_train)
```

```
▾ GaussianNB
GaussianNB()
```

we fit the Gaussian Naive Bayes, it computes a mean θ and variance σ and adds them to model parameters in attributes `gnb.theta_, gnb.sigma_`.

```
gnb.__dict__
```

```
{'priors': None,
 'var_smoothing': 1e-09,
 'classes_': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 'feature_names_in_': array(['petal_width', 'sepal_length', 'sepal_width',
'petal_length'],
       dtype=object),
 'n_features_in_': 4,
 'epsilon_': 3.2135586734693885e-09,
 'theta_': array([[0.24054054, 4.9972973 , 3.38918919, 1.45405405],
        [1.30882353, 5.91764706, 2.75882353, 4.19117647],
        [2.03902439, 6.66341463, 2.9902439 , 5.58292683]]),
 'var_': array([[0.01159971, 0.12242513, 0.14474799, 0.01978087],
        [0.0408045 , 0.2649827 , 0.11124568, 0.22139274],
        [0.06579417, 0.4071981 , 0.11453897, 0.30483046]]),
 'class_count_': array([37., 34., 41.]),
 'class_prior_': array([0.33035714, 0.30357143, 0.36607143])}
```

The attributes of the estimator object (`gbn`) describe the data (eg the class list) and the model's parameters. The `theta_` (θ) represents the mean and the `sigma_` (σ) represents the variance of the distributions.

> **Try it Yourself**
>
> Could you use what we learned about EDA to find the mean and variance of each feature for each species of flower?

Because the GaussianNB classifier calculates parameters that describe the assumed distribuiton of the data is is called a generative classifier. From a generative classifer, we can generate synthetic data that is from the distribution the classifer learned. If this data looks like our real data, then the model assumptions fit well.

> **Warning**
>
> the details of this math are not required understanding, but this describes the following block of code

To do this, we extract the mean and variance parameters from the model (`gnb.theta_,gnb.sigma_`) and `zip` them together to create an iterable object that in each iteration returns one value from each list (`for th, sig in zip(gnb.theta_,gnb.sigma_)`). We do this inside of a list comprehension and for each `th,sig` where `th` is from `gnb.theta_` and `sig` is from `gnb.sigma_` we use `np.random.multivariate_normal` to get 20 samples. In a general multivariate normal distribution the second parameter is actually a covariance matrix. This describes both the variance of each individual feature and the correlation of the features. Since Naive Bayes is Naive it assumes the features are independent or have 0 correlation. So, to create the matrix from the vector of variances we multiply by `np.eye(4)` which is the identity matrix or a matrix with 1 on the diagonal and 0 elsewhere. Finally we stack the groups for each species together with `np.concatenate` (like `pd.concat` but works on numpy objects and `np.random.multivariate_normal` returns numpy arrays not data frames) and put all of that in a DataFrame using the feature names as the columns.

> **Hint**
>
> to try understanding this block of code, try extracting pieces of it and running each piece individually. I built it up piece by piece adn then wrapped them all together.

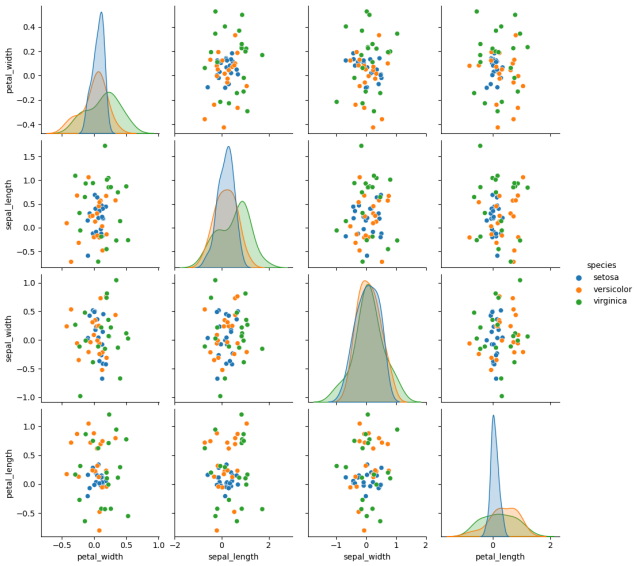
Then we add a species column, by repeating each species 20 times `[c]*N for c in gnb.classes_` and then unpack that into a single list instead of as list of lists.

```
N = 20
gnb_df = pd.DataFrame(np.concatenate([np.random.multivariate_normal(th,
sig*np.eye(4),N)
         for th, sig in zip(gnb.var_,gnb.sigma_)]),
         columns = feature_vars)
gnb_df['species'] = [ci for cl in [[c]*N for c in gnb.classes_] for ci in cl]
```

```
/opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/sklearn/utils/deprecation.py:103: FutureWarning: Attribute `sigma_` was
deprecated in 1.0 and will be removed in1.2. Use `var_` instead.
  warnings.warn(msg, category=FutureWarning)
```

```
sns.pairplot(data =gnb_df, hue='species')
```

```
<seaborn.axisgrid.PairGrid at 0x7fde38c88670>
```



This one looks pretty close to the actual data. The biggest difference is that these data are all in uniformly circular-ish blobs and the ones above are not. That means that the naive assumption doesn't hold perfectly on this data.

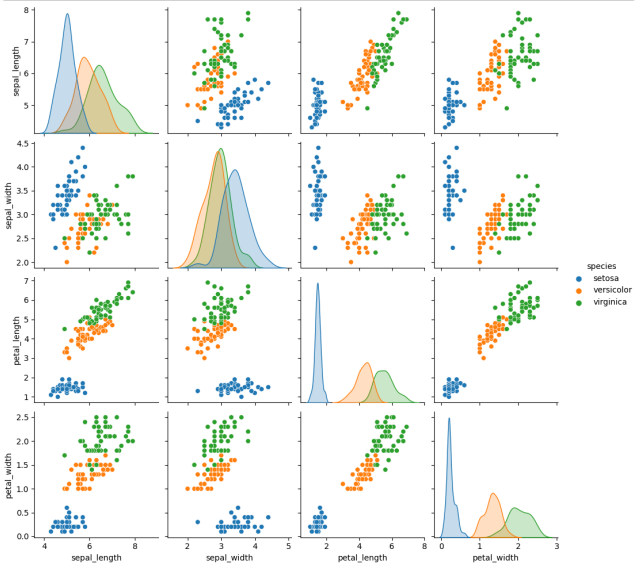
Further Reading

There are other classifiers that use roughly the same model but don't make the naive assumption. The more flexible is called Quadratic Discriminant Analysis.

- mathematical formulation for both
- QDA code
- LDA code

```
sns.pairplot(data=iris_df,hue='species')
```

```
<seaborn.axisgrid.PairGrid at 0x7fde332b38b0>
```



When we use the predict method, it uses those parameters to calculate the likelihood of the sample according to a Gaussian distribution (normal) for each class and then calculates the probability of the sample belonging to each class and returns the one with the highest probability.

```
y_pred = gnb.predict(X_test)
```

We can check the performance by comparing manually

```
sum(y_pred ==y_test)/len(y_test)
```

```
1.0
```

Or by scoring it.

```
gnb.score(X_test,y_test)
```

```
1.0
```

```
metrics.confusion_matrix(y_test,y_pred)
```

```
array([[13,  0,  0],
       [ 0, 16,  0],
       [ 0,  0,  9]])
```

17.1. Interpretting probabilities

```
gnb.predict_proba(X_test)
```

```
array([[2.05841140e-233, 1.23816844e-006, 9.99998762e-001],
       [1.76139943e-084, 9.99998414e-001, 1.58647449e-006],
       [1.00000000e+000, 1.48308613e-018, 1.73234612e-027],
       [6.96767669e-312, 5.33743814e-007, 9.99999466e-001],
       [1.00000000e+000, 9.33944060e-017, 1.22124682e-026],
       [4.94065646e-324, 6.57075840e-011, 1.00000000e+000],
       [1.00000000e+000, 1.05531886e-016, 1.55777574e-026],
       [2.45560284e-149, 7.80950359e-001, 2.19049641e-001],
       [4.01160627e-153, 9.10103555e-001, 8.98964447e-002],
       [1.46667004e-094, 9.99887821e-001, 1.12179234e-004],
       [5.29999917e-215, 4.59787449e-001, 5.40212551e-001],
       [4.93479766e-134, 9.46482991e-001, 5.35170089e-002],
       [5.23735688e-135, 9.98906155e-001, 1.09384481e-003],
       [4.97057521e-142, 9.50340361e-001, 4.96596389e-002],
       [9.11315109e-143, 9.87982897e-001, 1.20171030e-002],
       [1.00000000e+000, 7.81797826e-019, 1.29694954e-028],
       [3.86310964e-133, 9.87665084e-001, 1.23349155e-002],
       [2.27343573e-113, 9.99940331e-001, 5.96690955e-005],
       [1.00000000e+000, 1.80007196e-015, 9.14666201e-026],
       [1.00000000e+000, 1.30351394e-015, 8.42776899e-025],
       [4.66537803e-188, 1.18626155e-002, 9.88137385e-001],
       [1.02677291e-131, 9.92205279e-001, 7.79472050e-003],
       [1.00000000e+000, 6.61341173e-013, 1.42044069e-022],
       [1.00000000e+000, 9.98321355e-017, 3.50690661e-027],
       [2.27898063e-170, 1.61227371e-001, 8.38772629e-001],
       [1.00000000e+000, 2.29415652e-018, 2.54202512e-028],
       [1.00000000e+000, 5.99780345e-011, 5.24260178e-020],
       [1.62676386e-112, 9.99340062e-001, 6.59938068e-004],
       [2.23238199e-047, 9.99999965e-001, 3.47984452e-008],
       [1.00000000e+000, 1.95773682e-013, 4.10256723e-023],
       [3.52965800e-228, 1.15450262e-003, 9.98845497e-001],
       [3.20480410e-131, 9.93956330e-001, 6.04366979e-003],
       [1.00000000e+000, 1.14714843e-016, 2.17310302e-026],
       [3.34423817e-177, 8.43422262e-002, 9.15657774e-001],
       [5.60348582e-264, 1.03689515e-006, 9.99998963e-001],
       [7.48035097e-091, 9.99950155e-001, 4.98452400e-005],
       [1.00000000e+000, 1.80571225e-013, 1.83435499e-022],
       [8.97496247e-182, 5.65567226e-001, 4.34432774e-001]])
```

These are hard to interpret as is, one option is to plot them

```
# make the prbabilities into a dataframe labeled with classes & make the index a
separate column
prob_df = pd.DataFrame(data = gnb.predict_proba(X_test), columns = gnb.classes_
).reset_index()
# add the predictions
prob_df['predicted_species'] = y_pred
prob_df['true_species'] = y_test.values
# for plotting, make a column that combines the index & prediction
pred_text = lambda r: str( r['index']) + ',' + r['predicted_species']
prob_df['i,pred'] = prob_df.apply(pred_text,axis=1)
# same for ground truth
true_text = lambda r: str( r['index']) + ',' + r['true_species']
prob_df['correct'] = prob_df['predicted_species'] == prob_df['true_species']
# a dd a column for which are correct
prob_df['i,true'] = prob_df.apply(true_text,axis=1)
prob_df_melted = prob_df.melt(id_vars =[ 'index',
'predicted_species','true_species','i,pred','i,true','correct'],value_vars =
gnb.classes_,
                              var_name = target_var, value_name = 'probability')
prob_df_melted.head()
```

	index	predicted_species	true_species	i,pred	i,true	correct	species	probability
0	0	virginica	virginica	0,virginica	0,virginica	True	setosa	2.058411e-233
1	1	versicolor	versicolor	1,versicolor	1,versicolor	True	setosa	1.761399e-84
2	2	setosa	setosa	2,setosa	2,setosa	True	setosa	1.000000e+00
3	3	virginica	virginica	3,virginica	3,virginica	True	setosa	6.967677e-312
4	4	setosa	setosa	4,setosa	4,setosa	True	setosa	1.000000e+00

Now we have a data frame where each rown is one the probability of one sample belonging to one class. So there's a total of `number_of_samples*number_of_classes` rows

```
prob_df_melted.shape
```

```
(114, 8)
```

```
len(y_pred)*len(gnb.classes_)
```

```
114
```

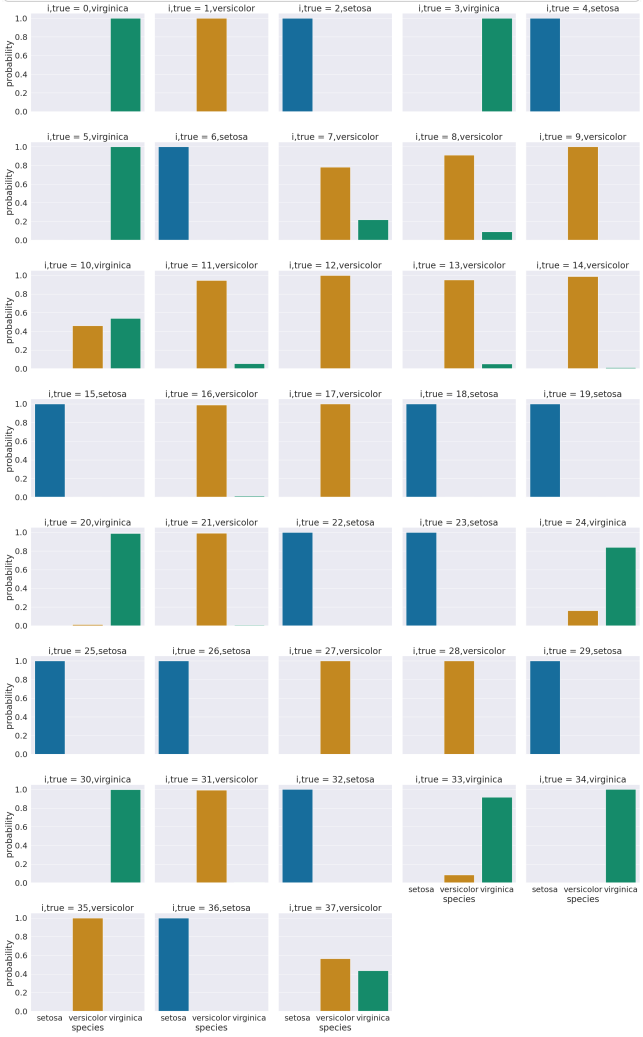
One way to look at these is to, for each sample in the test set, make a bar chart of the probability it belongs to each class. We added to the data frame information so that we can plot this with the true class in the title using `col = 'i,true'`

```
sns.set_theme(font_scale=2, palette= "colorblind")
# plot a bar graph for each point labeled with the prediction
sns.catplot(data =prob_df_melted, x = 'species', y='probability' ,col ='i,true',
            col_wrap=5,kind='bar')
```

Tip

I used `set_theme` to change both the fond size and the color palette. Seaborn has a detailed guide for choosing colors. The `colorblind` palette uses colors that are distinguishable under most common forms of color blindness.

```
<seaborn.axisgrid.FacetGrid at 0x7fde2bfbfc10>
```



We see that most sampples have nearly all of their probability mass (all probabiilties in a distribution sum (or integrate if continuous) to 1, but a few samples are not.

17.2. Questions After Class

17.2.1. What are the train and test datasets?

They are subsets of the original dataset.

For example if I use the index of the test set to pick a set of rows from the original dataset

```
iris_df.iloc[X_test.index[:5]]
```

	sepal_length	sepal_width	petal_length	petal_width	species
114	5.8	2.8	5.1	2.4	virginica
62	6.0	2.2	4.0	1.0	versicolor
33	5.5	4.2	1.4	0.2	setosa
107	7.3	2.9	6.3	1.8	virginica
7	5.0	3.4	1.5	0.2	setosa

and compare it to the actual test set

```
X_test.head()
```

	petal_width	sepal_length	sepal_width	petal_length
114	2.4	5.8	2.8	5.1
62	1.0	6.0	2.2	4.0
33	0.2	5.5	4.2	1.4
107	1.8	7.3	2.9	6.3
7	0.2	5.0	3.4	1.5

the columns are out of order because of how we created the train test splits, but the data are the same.

We split the data so that we can be sure that the algorithm is generalizing what it learned from the test set.

17.2.2. y_pred = gnb.predict(X_test). What is this predicting?

This uses the model to find the highest probability class for each sample in the test set.

17.2.3. What is the best number of samples to use to have the most accurate predictions?

That will vary and is one of the things you will experiment with in assignment 7.

17.2.4. need more clarification on the confusion_matrix. not sure what its doing if we already proved y_pred and y_test are the same

In this case, it does basically repeat that, except it does it per species (class). Notice that all of the numbers are on the diagonal of the matrix.

17.2.5. Can we use this machine learning to predict MIC data of peptides based off their sequences and similarly output data for new synthetic AMPS?

Possibly, but maybe a different model. I would need to know more about the dat here and I am not a biologist. We can discuss this in office hours.

17.2.6. If more samples of data increases accuracy of the model, is there ever a point where adding more samples of data does not increase accuracy?

More samples increases the reliability of the estimate, and generally increases the accuracy, but not always. We will see different ways that more data does not increase over the next few weeks. You will also experiment with this in your next assignment (I give you the steps, you interpret the results)

18. Decision Trees

```
%matplotlib inline
```

```
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn import tree
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
sns.set(palette='colorblind') # this improves contrast

from sklearn.metrics import confusion_matrix, classification_report
```

18.1. Let's look at a toy dataset

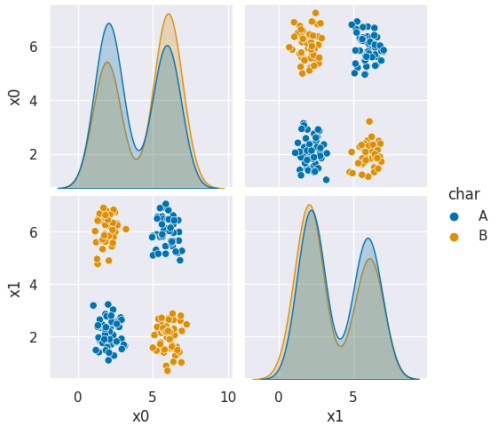
Using a toy dataset here shows an easy to see challenge for the classifier that we have seen so far. Real datasets will be hard in different ways, and since they're higher dimensional, it's harder to visualize the cause.

```
corner_data = 'https://raw.githubusercontent.com/rhodyprog4ds/06-naive-
bayes/f425ba121cc0c4dd8bcaa7ebb2ff0b40b0b03bff/data/dataset6.csv'
df6= pd.read_csv(corner_data,usecols=[1,2,3])
```

```
gnb = GaussianNB()
```

```
sns.pairplot(data=df6, hue='char',hue_order=['A','B'])
```

```
<seaborn.axisgrid.PairGrid at 0x7f1016a80340>
```



As we can see in this dataset, these classes are quite separated.

```
X_train, X_test, y_train, y_test = train_test_split(df6[['x0','x1']],
                                                    df6['char'],
                                                    random_state = 4)
```

```
gnb.fit(X_train,y_train)
gnb.score(X_test,y_test)
```

```
0.72
```

But we do not get a very good classification score.

To see why, we can look at what it learned.

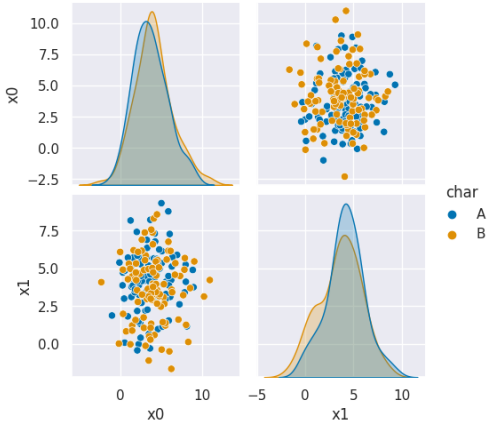
```
gnb.__dict__
```

```
{'priors': None,
 'var_smoothing': 1e-09,
 'classes_': array(['A', 'B'], dtype='<U1'),
 'feature_names_in_': array(['x0', 'x1'], dtype=object),
 'n_features_in_': 2,
 'epsilon_': 4.2942498888888889e-09,
 'theta_': array([[3.91910256, 3.9624359 ],
        [4.42861111, 3.54222222]]),
 'var_': array([[4.0355774 , 3.99742099],
        [4.43948696, 4.14491451]]),
 'class_count_': array([78., 72.]),
 'class_prior_': array([0.52, 0.48])}
```

```
N = 100
gnb_df = pd.DataFrame(np.concatenate([np.random.multivariate_normal(th,
sig*np.eye(2),N)
         for th, sig in zip(gnb.theta_,gnb.sigma_)]),
         columns = ['x0','x1'])
gnb_df['char'] = [ci for cl in [[c]*N for c in gnb.classes_] for ci in cl]

sns.pairplot(data =gnb_df, hue='char',hue_order=['A','B'])
```

```
/opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/sklearn/utils/deprecation.py:103: FutureWarning: Attribute `sigma_` was
deprecated in 1.0 and will be removed in1.2. Use `var_` instead.
  warnings.warn(msg, category=FutureWarning)
```

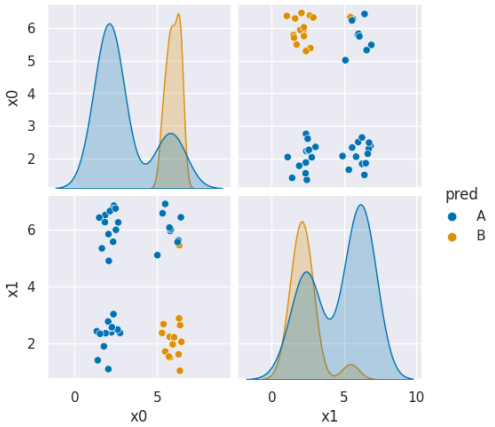
```
<seaborn.axisgrid.PairGrid at 0x7f10077a0280>
```



This does not look much like the data and it's hard to tell which is higher at any given point in the 2D space. We know though, that it has missed the mark. We can also look at the actual predictions.

```
df6pred = X_test.copy()
df6pred['pred'] =gnb.predict(X_test)

sns.pairplot(df6pred, hue = 'pred',hue_order=['A','B'])
```

```
<seaborn.axisgrid.PairGrid at 0x7f10044e8fa0>
```



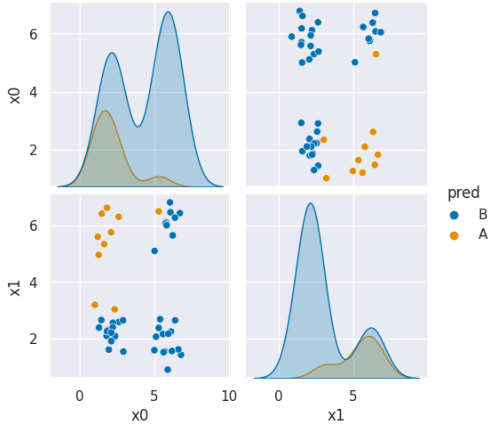
This makes it more clear. It basically learns one group that covers 3 blobs and only 1 for the second color. If we train again with a different random seed, it makes a different specific error, but basically the same idea.

```
X_train, X_test, y_train, y_test = train_test_split(df6[['x0','x1']],
                                   df6['char'], random_state = 5)
gnb.fit(X_train,y_train)
gnb.score(X_test,y_test)
```

```
0.34
```

```
df6pred = X_test.copy()
df6pred['pred'] =gnb.predict(X_test)

sns.pairplot(df6pred, hue = 'pred')
```

```
<seaborn.axisgrid.PairGrid at 0x7f10076fcb50>
```

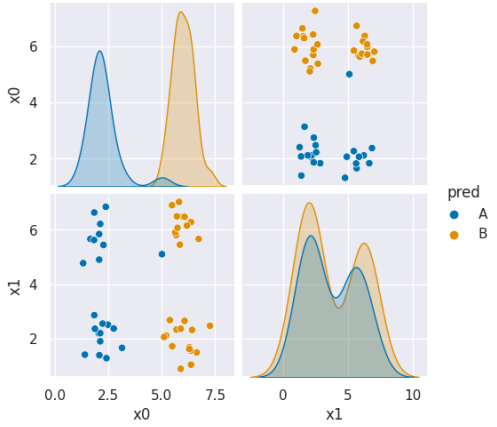



```
X_train, X_test, y_train, y_test = train_test_split(df6[['x0','x1']],
                                                    df6['char'], random_state = 7)
gnb.fit(X_train,y_train)
gnb.score(X_test,y_test)
```

```
0.58
```

```
df6pred = X_test.copy()
df6pred['pred'] =gnb.predict(X_test)

sns.pairplot(df6pred, hue = 'pred')
```

```
<seaborn.axisgrid.PairGrid at 0x7f1016a80e50>
```



If you try this again, split, fit, plot, it will learn different decisions, but always at least about 25% of the data will have to be classified incorrectly.

18.2. Decision Trees

This data does not fit the assumptions of the Niave Bayes model, but a decision tree has a different rule. It can be more complex, but for the scikit learn one relies on splitting the data at a series of points along one axis at a time.

It is a **discriminative** model, because it describes how to discriminate (in the sense of differentiate) between the classes.

```
dt = tree.DecisionTreeClassifier()
```

```
dt.fit(X_train,y_train)
```

```
▼ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
dt.score(X_test,y_test)
```

```
1.0
```

The sklearn estimator objects (that corresond to different models) all have the same API, so the `fit`, `predict`, and `score` methods are the same as above. We will see this also in regression and clustering. What each method does in terms of the specific calculations will vary depending on the model, but they're always there.

the `tree` module also allows you to plot the tree to examine it.

```
plt.figure(figsize=(15,20))
tree.plot_tree(dt, rounded =True, class_names = ['A','B'],
      proportion=True, filled =True, impurity=False,fontsize=10);
```

18.3. Setting Classifier Parameters

The decision tree we had above has a lot more layers than we would expect. This is really simple data so we still got perfect classification. However, the more complex the model, the more risk that it will learn something noisy about the training data that doesn't hold up in the test set.

Fortunately, we can control the parameters to make it find a simpler decision boundary.

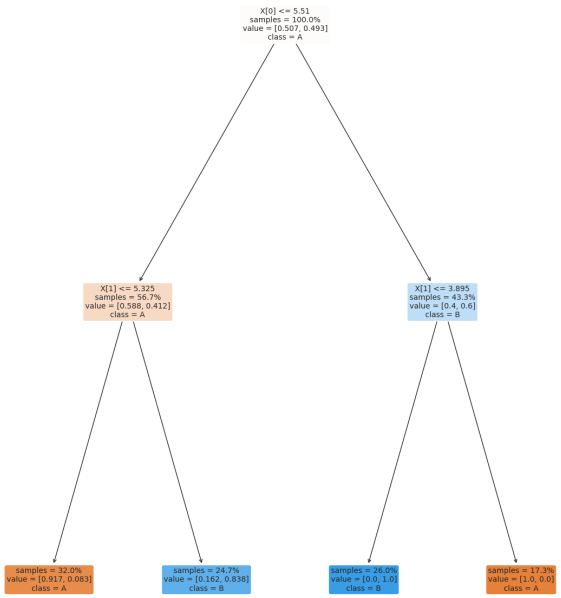
```
dt2 = tree.DecisionTreeClassifier(max_depth=2)
dt2.fit(X_train,y_train)
```

```
▼ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2)
```

```
plt.figure(figsize=(15,20))
tree.plot_tree(dt2, rounded =True, class_names = ['A','B'],
      proportion=True, filled =True, impurity=False,fontsize=10);
```

Note

On the iris dataset, the sklearn docs include a diagram showing the decision boundary You should be able to modify this for another classifier.




```
dt2.score(X_test,y_test)
```

```
0.86
```

We might need to play with different parameters to get it just how we want it. A simpler model is better because it will be more reliable in general.

```
dt2.score(X_test,y_test)
```

```
0.86
```

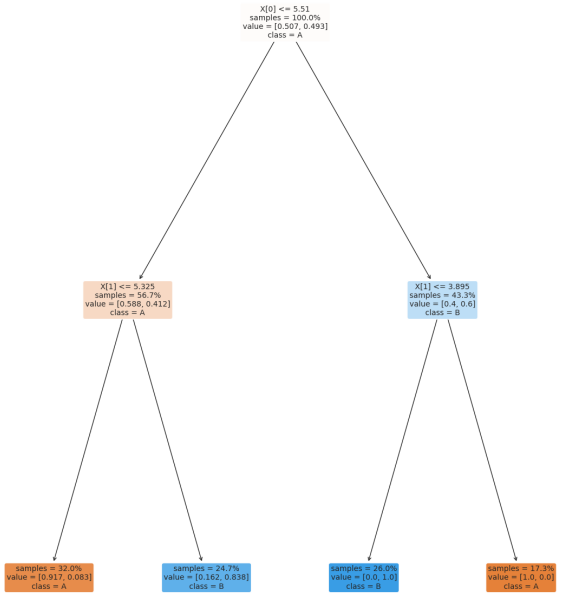
```
dt2_20 = tree.DecisionTreeClassifier(max_depth=2)
dt2_20.fit(X_train,y_train)
```

```
▼ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2)
```

```
dt2_20.score(X_test, y_test)
```

```
0.86
```

```
plt.figure(figsize=(15,20))
tree.plot_tree(dt2_20, rounded =True, class_names = ['A','B'],
      proportion=True, filled =True, impurity=False,fontsize=10);
```



18.4. Questions After Class

18.4.1. What do the dots and lines represent in the graph?

The scatter plots, each dot is one sample (row from the DataFrame), the different sub plots are different views of the data, determined by different pairs of variables.

The lines are the density, when it's high it means values are likely , when it's low it means they are unlikely.

18.4.2. What doe the GNB model do?

When we fit, it learns a description of the data. When we predict it uses that description to predict the probability that each test sample belongs to each of the classes and returns the most probable one. See the last notes for a visualization of the probabilities.

18.4.3. Is xtrain ytrain, xtest ytest the same every time?

Generally no, with the `random_state` variable set on a given data they will be the same subset every time, but without it would get different rows. For example:

```
X_train, X_test, y_train, y_test = train_test_split(df6[['x0','x1']],
                                                    df6['char'],
                                                    random_state = 4)
X_train.head()
```

	x0	x1
110	5.91	2.39
154	6.07	6.47
16	6.84	2.61
19	6.27	1.99
2	2.27	5.44

```
X_train, X_test, y_train, y_test = train_test_split(df6[['x0','x1']],
                                                    df6['char'],
                                                    random_state = 4)
X_train.head()
```

	x0	x1
110	5.91	2.39
154	6.07	6.47
16	6.84	2.61
19	6.27	1.99
2	2.27	5.44

with random state, we get the same samples each time.

```
X_train, X_test, y_train, y_test = train_test_split(df6[['x0','x1']],
                                                    df6['char'])
X_train.head()
```

	x0	x1
65	2.12	6.21
103	3.06	1.69
191	1.87	2.36
112	2.03	2.77
150	6.87	2.80

```
X_train, X_test, y_train, y_test = train_test_split(df6[['x0','x1']],
                                                    df6['char'])
X_train.head()
```

	x0	x1
52	6.05	2.70
58	1.44	1.86
128	5.49	1.72
11	2.75	2.37
116	5.74	6.07

without, we get different ones.

18.4.4. I still want to know why it is worse to have a model that gets 100% accuracy with more depth to the decision tree flowchart than to have a model with less accuracy and less decisions in the decision tree.

This accuracy is only on one set of training data, we might not want the big accuracy drop we saw here ,but we could do other things to improve it.

19. Feedback & Regression

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
import pandas as pd
sns.set_theme(font_scale=2,palette='colorblind')

data_url =
'https://raw.githubusercontent.com/rhodyprog4ds/rhodyds/main/data/310_data_22-
midsem.csv'
```

19.1. Feedback from you

Here is the feedback from the survey that you completed with feedback on the course.

```
feedback_df_raw = pd.read_csv(data_url)
```

```
feedback_df_raw.head()
```

	How much do you think you've learned so far this semester?	How much of the material that's been taught do you feel you understand?	How do you think the achievements you've earned so far align with your understanding?	Rank the following as what you feel the grading (when you do/not earn achievements) so far actually reflects about your performance in the class [How well I follow instructions]	Rank the following as what you feel the grading (when you do/not earn achievements) so far actually reflects about your performance in the class [What I understand about the material]	Rank the following as what you feel the grading (when you do/not earn achievements) so far actually reflects about your performance in the class [How much effort I put into assignments]	How fair do you think the amount each of the following is reflected in the grading [How well I follow instructions]	How fair do you think the amount each of the following is reflected in the grading [What I understand about the material]	How fair do you think the amount each of the following is reflected in the grading [How much effort I put into assignments]	Which of the following have you done to support your learning outside of class time?
0	4	4	I think they reflect my understanding well	Reflected perfectly in the grading	Reflected perfectly in the grading	Reflected perfectly in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	read the notes online, experimenting with the ...
1	5	4	I think they reflect my understanding well	Reflected perfectly in the grading	Reflected perfectly in the grading	Reflected perfectly in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	read the notes online, experimenting with the ...
2	3	3	I think they reflect my understanding well	Reflected moderately in the grading	Reflected moderately in the grading	Reflected moderately in the grading	Should be less reflected in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	read the notes online, download and run the no...
3	4	3	I think they reflect my understanding well	Reflected moderately in the grading	Reflected moderately in the grading	Reflected moderately in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	read the notes online, reading the documentati...
4	5	2	I think they reflect my understanding well	Reflected moderately in the grading	Reflected strongly in the grading	Reflected a little in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	Should be reflected more in the grading	read the notes online, experimenting with the ...

```
feedback_df_raw.columns
```

```
Index(['How much do you think you've learned so far this semester?',
       'How much of the material that's been taught do you feel you understand?',
       'How do you think the achievements you've earned so far align with your
understanding?',
       'Rank the following as what you feel the grading (when you do/not earn
achievements) so far actually reflects about your performance in the class [How
well I follow instructions]',
       'Rank the following as what you feel the grading (when you do/not earn
achievements) so far actually reflects about your performance in the class [What I
understand about the material]',
       'Rank the following as what you feel the grading (when you do/not earn
achievements) so far actually reflects about your performance in the class [How
much effort I put into assignments]',
       'How fair do you think the amount each of the following is reflected in the
grading [How well I follow instructions]',
       'How fair do you think the amount each of the following is reflected in the
grading [What I understand about the material]',
       'How fair do you think the amount each of the following is reflected in the
grading [How much effort I put into assignments]',
       'Which of the following have you done to support your learning outside of
class time? '],
      dtype='object')
```

```
short_names = {"How much do you think you've learned so far this
semester?":'learned',
    "How much of the material that's been taught do you feel you
understand?":'understand',
    "How do you think the achievements you've earned so far align with your
understanding?":'achievements',
    'Rank the following as what you feel the grading (when you do/not earn
achievements) so far actually reflects about your performance in the class [How
well I follow instructions]':'grading_instructions',
    'Rank the following as what you feel the grading (when you do/not earn
achievements) so far actually reflects about your performance in the class [What I
understand about the material]':'grading_understanding',
    'Rank the following as what you feel the grading (when you do/not earn
achievements) so far actually reflects about your performance in the class [How
much effort I put into assignments]':'grading_effort',
    'How fair do you think the amount each of the following is reflected in the
grading [How well I follow instructions]':'fairness_instructions',
    'How fair do you think the amount each of the following is reflected in the
grading [What I understand about the material]':'fairness_understanding',
    'How fair do you think the amount each of the following is reflected in the
grading [How much effort I put into assignments]':'fairness_effort',
    'Which of the following have you done to support your learning outside of
class time? ':'learning_activities'}
```

```
feedback_df_cols = feedback_df_raw.rename(columns=short_names)
feedback_df_cols.head(2)
```

	learned	understand	achievements	grading_instructions	grading_understanding	grading_effort	fairness_instructions	fairness_understanding	fairness_effort	learning
0	4	4	I think they reflect my understanding well	Reflected perfectly in the grading	Reflected perfectly in the grading	Reflected perfectly in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	rea experin
1	5	4	I think they reflect my understanding well	Reflected perfectly in the grading	Reflected perfectly in the grading	Reflected perfectly in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	Is fairly reflected in the grading	rea experin

```
learning_lists = feedback_df_cols['learning_activities'].str.split(',')
learning_stacked = learning_lists.apply(pd.Series).stack()
learning_df = pd.get_dummies(learning_stacked).sum(level=0)
learning_df.head()
```

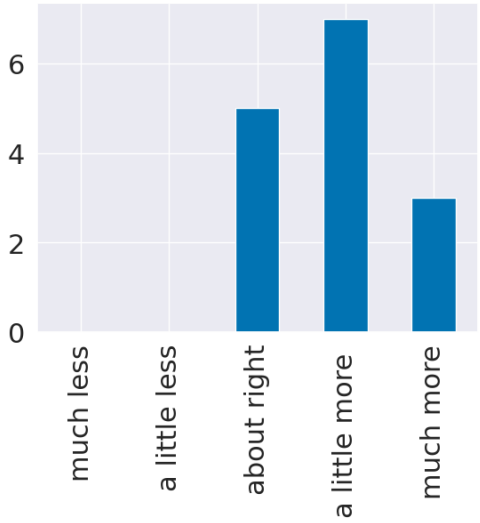
```
/tmp/ipykernel_2189/3116911913.py:3: FutureWarning: Using the level keyword in
DataFrame and Series aggregations is deprecated and will be removed in a future
version. Use groupby instead. df.sum(level=1) should use
df.groupby(level=1).sum().
  learning_df = pd.get_dummies(learning_stacked).sum(level=0)
```

	attended Aiden's office hours	attended Dr. Brown's office hours	download and run the notes	experimenting with the code from class	reading blogs or tutorials I find on my own	reading the documentation or course text	tinkering with code to answer other aspects of the material that I'm curious about	update the notes I took during class time	watching videos that I find on my own	download and run the notes	read the notes online	update the notes I took during class time
0	0	0	0	1	1	0	0	0	1	0	1	0
1	0	0	0	1	0	1	1	0	0	0	1	0
2	0	1	1	1	0	0	0	1	0	0	1	0
3	1	1	0	0	0	1	1	0	0	0	1	0
4	1	0	0	1	1	1	0	0	1	0	1	0

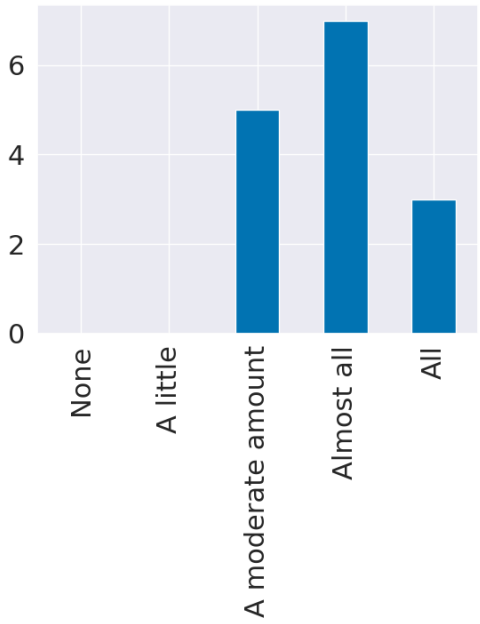
```
learning_df.sum()
```

```
 attended Aiden's office hours
7
 attended Dr. Brown's office hours
4
 download and run the notes
4
 experimenting with the code from class
11
 reading blogs or tutorials I find on my own
9
 reading the documentation or course text
11
 tinkering with code to answer other aspects of the material that I'm curious
about     5
 update the notes I took during class time
4
 watching videos that I find on my own
7
download and run the notes
1
read the notes online
13
update the notes I took during class time
1
dtype: int64
```

```
el_meaning = {1: 'much less',
 2: 'a little less',
 3: 'about right',
 4: 'a little more ',
 5: 'much more'}

question_text = list(short_names.keys())
[list(short_names.values()).index('learned')]
el_counts,_ = np.histogram(feedback_df_cols['learned'],bins = [i+.5 for i in
range(6)])
el_df = pd.DataFrame(data = el_counts,index = el_meaning.values(),columns=
[question_text],)

# el_df.rename_axis(index='amount relative to expectations',inplace=True)
el_df.plot.bar(legend=False);
# sns.displot(feedback_df['learned'].replace(el_meaning))
```




```
u_meaning = {0:'None', 1:'A little',3:'A moderate amount', 4:'Almost all',5:'All'}


question_text = list(short_names.keys())
[list(short_names.values()).index('understand')]
u_counts,_ = np.histogram(feedback_df_cols['understand'],bins = [i+.5 for i in
range(6)])
u_df = pd.DataFrame(data = el_counts,index = u_meaning.values(),columns=
[question_text],)


u_df.plot.bar(legend=False);
```



20. Responses to Qualitative Comments

- Going forward, the assignments are more tied together so you do not need to find as many datasets. I've made myself a note to go back and update A1-A5 for next year.
- help by contirbuting recommended datasets
- help by contributing do not use list to course website
- I will try to make sure that most code is prepared so I can send on prismia
- I recommend aligning your prismia & notebook side by side. (I cannot model this because of font size limitations)
- I've requested for longer blocks for next fall… but classroom availability (this is a good idea!)

20.1. Regression

We're going to predict **tip** from **total bill** using 80% of the data for training. This is a regression problem because the target, *tip* is a continuous value, the problems we've seen so far were all classification, species of iris and the character in that corners data were both categorical.

Using linear regression is also a good choice because it makes sense that the tip would be approximately linearly related to the total bill, most people pick some percentage of the total bill. If we our prior knowledge was that people typically tipped with some more complicated function, this would not be a good model.

```
tips_df = sns.load_dataset("tips")
```

```
tips_df.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Split the data so that we use 80% of the data to train a modle to predict the tip from the total bill

```
tips_X = tips_df['total_bill'].values
tips_X = tips_X[:,np.newaxis] # add an axis
tips_y = tips_df['tip']

tips_X_train,tips_X_test, tips_y_train, tips_y_test =
train_test_split(tips_X,tips_y,train_size=.8)
```

To see what that new bit of code did, we can examine the shapes:

```
tips_X.shape
```

```
(244, 1)
```

what we ended up is 2 dimensions (there are two numbers) even though the second one is 1.

```
tips_df['total_bill'].values.shape
```

```
(244,)
```

this, without the `newaxis` is one dimension, we can see that because there is no number after the comma.

Now that our data is ready, we create the linear regression estimator object

```
regr = linear_model.LinearRegression()
```

```
type(regr)
```

```
sklearn.linear_model._base.LinearRegression
```

Now we fit the model.

```
regr.fit(tips_X_train,tips_y_train)
```

```
▼ LinearRegression
LinearRegression()
```

We can examine the coefficients and intercept.

```
regr.coef_, regr.intercept_
```

```
(array([0.10095314]), 0.992802323813295)
```

These define a line (y = mx+b) coef is the slope.

> **Important**
>
> This is what our model *predicts* the tip will be based on the past data. It is important to note that this is not what the tip *should* be by any sort of virtues. For example, a typical normative rule for tipping is to tip 15% or 20%. the model we learned, from this data, however is ~%10 + $1.(it's actually 9.681.028)$

To interpret this, we can apply it for a single value. We trained this to predict the tip from the total bill. So, we can put in any value that's a plausible total bill and get the predicted tip.

We can predict the data

```
tips_y_pred = regr.predict(tips_X_test)
```

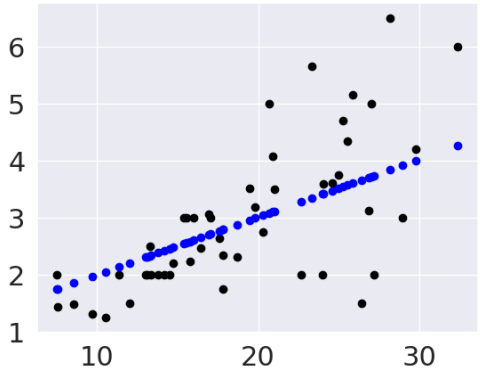
```
type(tips_y_pred)
```

```
numpy.ndarray
```

To visualize in more detail, we'll plot the data as black points and the predictions as blue points. To highlight that this is a perfectly linear prediction, we'll also add a line for the prediction.

```
plt.scatter(tips_X_test,tips_y_test, color='black')
plt.scatter(tips_X_test,tips_y_pred, color='blue')
```

```
<matplotlib.collections.PathCollection at 0x7fb77b49e700>
```



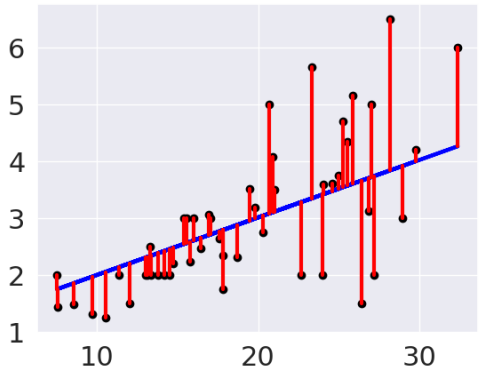
``

20.2. Evaluating Regression - Mean Squared Error

From the plot, we can see that there is some error for each point, so accuracy that we've been using, won't work. One idea is to look at how much error there is in each prediction, we can look at that visually first.

These red lines are the residuals.

```
plt.scatter(tips_X_test, tips_y_test, color='black')
plt.plot(tips_X_test, tips_y_pred, color='blue', linewidth=3)

# draw vertical lines frome each data point to its predict value
[plt.plot([x,x],[yp,yt], color='red', linewidth=3)
        for x, yp, yt in zip(tips_X_test, tips_y_pred,tips_y_test)];
```



We can use the average length of these red lines to capture the error. To get the length, we can take the difference between the prediction and the data for each point. Some would be positive and others negative, so we will square each one then take the average.

```
mean_squared_error(tips_y_test,tips_y_pred)
```

```
0.91136641197245
```

We can get back to the units being dollars, by taking the square root.

```
np.sqrt(mean_squared_error(tips_y_test, tips_y_pred))
```

```
0.9546551272435769
```

This is equivalent to using absolute value instead

```
np.mean(np.abs(tips_y_test - tips_y_pred))
```

```
0.7184144016211816
```

20.3. Evaluating Regression - R2

We can also use the R^2 score, the coefficient of determination.

If we have the following:

- n `=len(y_test)``
- y =`y_test`
- y_i =`y_test[i]`
- $\hat{y}$ = `y_pred`
- $\bar{y} = \frac{1}{n}\sum_{i=0}^{n} y_i$ = `sum(y_test)/len(y_test)`

$$R^2(y, \hat{y}) = 1 = \frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n}(y_i - \bar{y}_i)^2}$$

```
r2_score(tips_y_test, tips_y_pred)
```

```
0.445722425385784
```

This is a bit harder to interpret, but we can use some additional plots to visualize. This code simulates data by randomly picking 20 points, spreading them out and makes the “predicted” y values by picking a slope of 3. Then I simulated various levels of noise, by sampling noise and multiplying the same noise vector by different scales and adding all of those to a data frame with the column name the r score for if that column of target values was the truth.

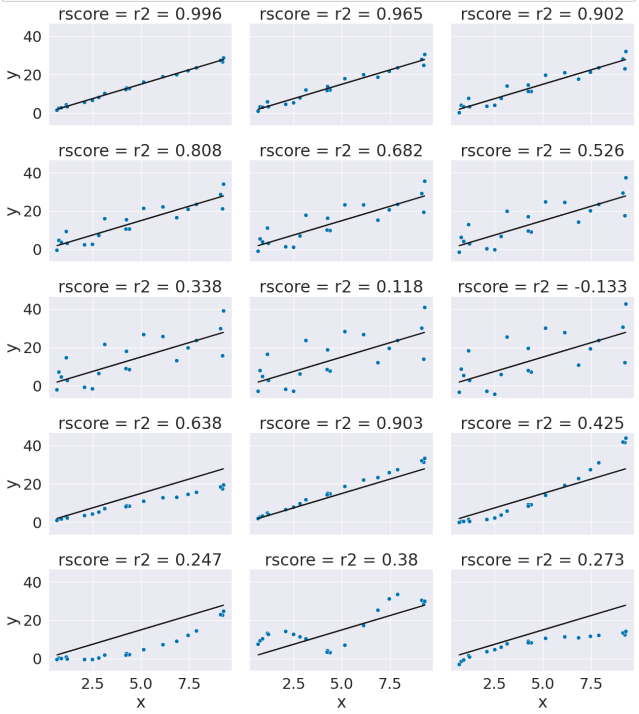
Then I added some columns of y values that were with different slopes and different functions of x. These all have the small amount of noise.

```
x = 10*np.random.random(20)
y_pred = 3*x
ex_df = pd.DataFrame(data = x,columns = ['x'])
ex_df['y_pred'] = y_pred
n_levels = range(1,18,2)
noise = (np.random.random(20)-.5)*2
for n in n_levels:
  y_true = y_pred + n* noise
  ex_df['r2 = '+ str(np.round(r2_score(y_pred,y_true),3))] = y_true

f_x_list = [2*x,3.5*x,.5*x**2, .03*x**3, 10*np.sin(x)+x*3,3*np.log(x**2)]
for fx in f_x_list:
  y_true = fx + noise
  ex_df['r2 = '+ str(np.round(r2_score(y_pred,y_true),3))] = y_true

xy_df = ex_df.melt(id_vars=['x','y_pred'],var_name='rscore',value_name='y')
# sns.lmplot(x='x',y='y', data = xy_df,col='rscore',col_wrap=3,)
g = sns.FacetGrid(data = xy_df,col='rscore',col_wrap=3,aspect=1.5,height=3)
g.map(plt.plot, 'x','y_pred',color='k')
g.map(sns.scatterplot, "x", "y",)
```

Tip

Facet Grids allow more customization than the figure level plotting functions we have used otherwise, but each of those combines a FacetGrid with a particular type of plot.

```
<seaborn.axisgrid.FacetGrid at 0x7fb7b4fd2d30>
```



By default, the regression estimator uses the R2 score.

```
regr.score(tips_X_test,tips_y_test)
```

```
0.445722425385784
```

21. Interpretting Regression

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
sns.set_theme(font_scale=2,palette='colorblind')
```

21.1. Multivariate Regression

We can also load data from Scikit learn.

This dataset includes 10 features measured on a given date and an measure of diabetes disease progression measured one year later. The predictor we can train with this data might be someting a doctor uses to calculate a patient's risk.

```
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
X_train,X_test, y_train,y_test = train_test_split(diabetes_X, diabetes_y ,
                         test_size=20,random_state=0)
```

```
X_train.shape
```

```
(422, 10)
```

```
regr_db = linear_model.LinearRegression()
```

```
regr_db.__dict__
```

```
{'fit_intercept': True,
 'normalize': 'deprecated',
 'copy_X': True,
 'n_jobs': None,
 'positive': False}
```

We have an empty estimator object at this point and then we can fit it as usual.

```
regr_db.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

This model predicts what lab measure a patient will have one year in the future based on lab measures in a given day. Since we see that this is not a very high r2, we can say that this is not a perfect predictor, but a Doctor, who better understands the score would have to help interpret the core.

```
regr_db.__dict__
```

```
{'fit_intercept': True,
 'normalize': 'deprecated',
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'n_features_in_': 10,
 'coef_': array([ -32.30360126, -257.44019182,  513.32582416,  338.46035389,
        -766.84661714,  455.83564292,   92.5514199 ,  184.75080624,
         734.91009007,   82.72479583]),
 'rank_': 10,
 'singular_': array([1.95678408, 1.19959858, 1.08212757, 0.95268243, 0.79449825,
        0.76416914, 0.71267072, 0.64259342, 0.27343748, 0.0914504 ]),
 'intercept_': 152.391853606725}
```

We can look at the estimator again and see what it learned. It describes the model like a line:

$$\hat{y} = mx + b$$

except in this case it's multivariate, so we can write it like:

$$\hat{y} = \beta^T x + \beta_0$$

where β is the `regr_db.coef_` and β_0 is `regr_db.intercept_` and that's a vector multiplication and $\hat{y}$ is `y_pred` and y is `y_test`.

In scalar form it can be written like

$$\hat{y} = \sum_{k=0}^{d} (x_k * \beta_k) + \beta_0$$

where there are d features, that is d= `len(X_test[k])` and k indexed into it. For example in the below $k = 0$

```
X_test[0]
```

```
array([ 0.01991321,  0.05068012,  0.10480869,  0.0700723 , -0.03596778,
       -0.0266789 , -0.02499266, -0.00259226,  0.00370906,  0.04034337])
```

We can compute the prediction manually:

```
sum(regr_db.coef_*X_test[0])  + regr_db.intercept_
```

```
234.91095893227003
```

and see that it matches

```
regr_db.predict(X_test[:2])
```

```
array([234.91095893, 246.81176987])
```

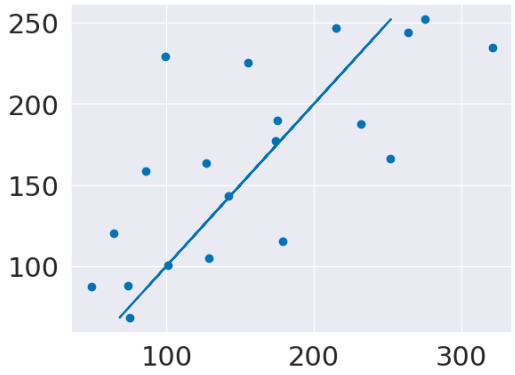
```
regr_db.score(X_test,y_test)
```

```
0.5195333332288746
```

We can also look at the residuals and see that they are fairly randomly spread out.

```
y_pred = regr_db.predict(X_test)
plt.scatter(y_test,y_pred)
plt.plot(y_pred,y_pred)
```

```
[<matplotlib.lines.Line2D at 0x7f4c80d01d00>]
```



21.2. Questions After Class

21.2.1. Is there a way to calculate p value in addition to r^2?

The R^2 is a measure of the quality of the regression fit, but a p-value is not. A p-value implies a broader experimental framework, so it is not included in this object. `sklearn` does have tools that will calculate a p-value when appropriate, but it not built into this estimator object. The **`scipy.stats.linregress`** estimator does compute a p-value, and allows you to specify under which statistical test you want that p-value computed (two-sized, one-sided greater, or one-sided lesser).

Broadly, p-values are very frequently misinterpretted and misused. Their use is not common in ML (because they're usually not the right thing) and being actively discouraged in other fields (eg psychology) in favor of more reliable and consistently well interpretted statistics.

22. Sparse and Polynomial Regression

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
sns.set_theme(font_scale=2,palette='colorblind')
```

22.1. Exmaining Residuals

```
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
X_train,X_test, y_train,y_test = train_test_split(diabetes_X, diabetes_y ,
                        test_size=20,random_state=0)
regr_db = linear_model.LinearRegression()
regr_db.fit(X_train, y_train)
regr_db.score(X_test,y_test)
```

```
0.5195333332288746
```

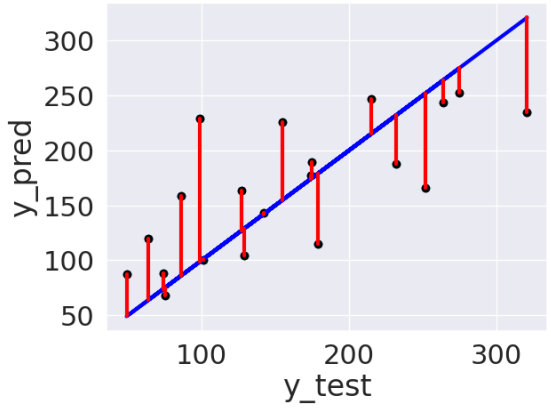
```
regr_db.__dict__
```

```
{'fit_intercept': True,
 'normalize': 'deprecated',
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'n_features_in_': 10,
 'coef_': array([ -32.30360126, -257.44019182,  513.32582416,  338.46035389,
        -766.84661714,  455.83564292,   92.5514199 ,  184.75080624,
         734.91009007,   82.72479583]),
 'rank_': 10,
 'singular_': array([1.95678408, 1.19959858, 1.08212757, 0.95268243, 0.79449825,
        0.76416914, 0.71267072, 0.64259342, 0.27343748, 0.0914504 ]),
 'intercept_': 152.391853606725}
```

```
y_pred = regr_db.predict(X_test)
```

```
plt.scatter(y_test,y_pred)

plt.scatter(y_test, y_pred, color='black')
plt.plot(y_test, y_test, color='blue', linewidth=3)

# draw vertical lines frome each data point to its predict value
[plt.plot([x,x],[yp,yt], color='red', linewidth=3)
        for x, yp, yt in zip(y_test, y_pred,y_test)];

plt.xlabel('y_test')
plt.ylabel('y_pred')
```

```
Text(0, 0.5, 'y_pred')
```



Important

plotting the resiudals this way is okay for seeing how it did, but is not intepretted the same way, I've changed it for the correct interpretation. This says that there might not be enough information for high values to predict them, *or* maybe a better model could do it.

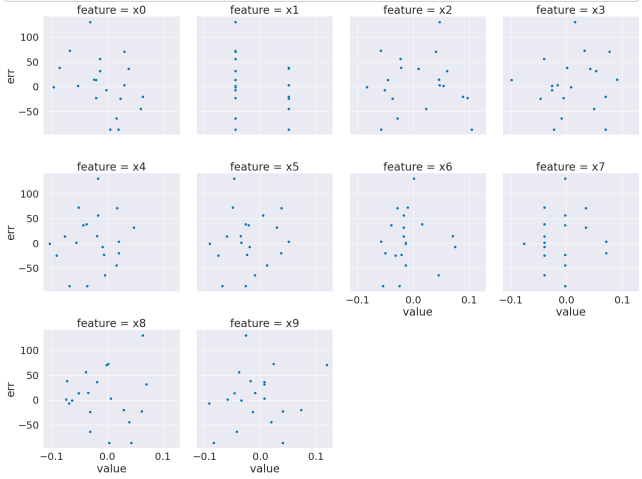
We can plot the residuals, more directly too:

```
test_df = pd.DataFrame(X_test,columns = ['x'+str(i) for i in
range(len(X_test[0]))])
test_df['err'] = y_pred-y_test
x_df = test_df.melt(var_name = 'feature',id_vars=['err'])
x_df.head()
```

	err	feature	value
0	-86.089041	x0	0.019913
1	31.811770	x0	-0.012780
2	36.464326	x0	0.038076
3	56.062302	x0	-0.012780
4	14.540509	x0	-0.023677

```
sns.relplot(data= x_df,x='value',col='feature', col_wrap=4,y='err')
```

```
<seaborn.axisgrid.FacetGrid at 0x7efc2cc63160>
```



Here, we see that the residuals for most features are pretty uniform, but a few (eg the first one) are correlated to the feature value.

22.2. Polynomial regression

Polynomial regression is still a linear problem. Linear regression solves for the β_i for a d dimensional problem.

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d = \sum_i^d \beta_i x_i$$

Quadratic regression solves for

$$y = \beta_0 + \sum_i^d \beta_i x_i \$ + \sum_j^d \sum_i^d \beta_{d+i} x_i x_j + \sum_i^d x_i^2 \$$$

This is still a linear problem, we can create a new X matrix that has the polynomial values of each feature and solve for more β values.

We use a transformer object, which works similarly to the estimators, but does not use targets. First, we instantiate.

```
poly = PolynomialFeatures(include_bias=False,)
```

Then we can transform the data

```
X2_train = poly.fit_transform(X_train)
```

```
X2_train.shape
```

```
(422, 65)
```

```
X_train.shape
```

```
(422, 10)
```

```
regr_db2 = linear_model.LinearRegression()
regr_db2.fit(X2_train,y_train)
regr_db2.score(X2_test,y_test)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [12], line 3
      1 regr_db2 = linear_model.LinearRegression()
      2 regr_db2.fit(X2_train,y_train)
----> 3 regr_db2.score(X2_test,y_test)

NameError: name 'X2_test' is not defined
```

We see the performance is a little better, but let's look at the residuals again.

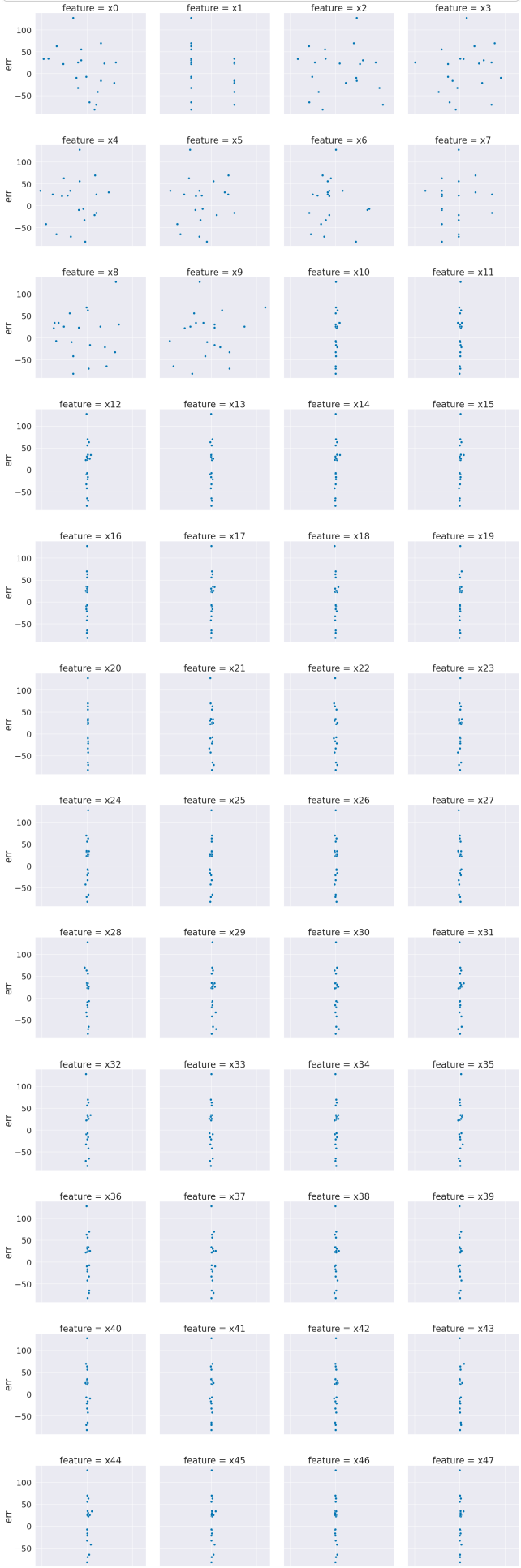
```
X2_test = poly.transform(X_test)
y_pred2 = regr_db2.predict(X2_test)
```

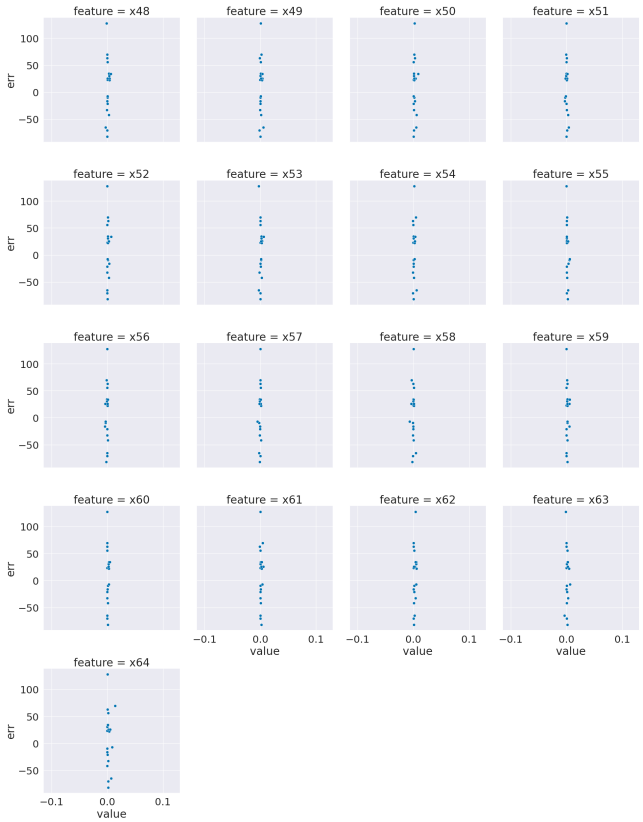
```
regr_db2.coef_
```

```
array([ 2.26718871e+01, -2.84321941e+02,  4.77972133e+02,  3.55751429e+02,
       -1.05551594e+03,  7.64836015e+02,  1.92998441e+02,  1.29510126e+02,
        9.95077095e+02,  6.97824322e+01,  1.35199820e+03,  3.45185945e+03,
        1.47333609e+02, -6.34976797e+01, -3.44685551e+03, -1.39445637e+03,
        5.68431155e+03,  5.24944560e+03,  1.93320706e+03,  1.44078438e+03,
       -1.71687299e+00,  1.07459375e+03,  1.71895547e+03,  1.01185087e+04,
       -8.18260464e+03, -2.98856060e+03, -2.75793035e+03, -2.63196845e+03,
        4.87268727e+02,  3.31912403e+02,  2.57451083e+03, -5.38594465e+03,
        3.88190888e+03,  2.88432660e+03,  4.95180080e+02,  2.98317779e+03,
       -5.23610008e+02, -6.08039517e+01,  8.85704665e+03, -5.89220694e+03,
       -3.89995076e+03, -1.62903703e+03, -2.50355665e+03, -2.07339059e+03,
        9.56513316e+04, -1.35146919e+05, -8.45241707e+04, -4.18163194e+04,
       -8.84048804e+04, -6.63039722e+03,  5.00187584e+04,  5.43127248e+04,
        2.05139512e+04,  6.78912952e+04,  4.41920280e+03,  2.10042065e+04,
        2.61797171e+04,  3.74988567e+04,  5.16395623e+03,  1.12705277e+04,
        1.13915240e+04,  4.08084006e+03,  2.05240548e+04,  2.20132713e+03,
        1.91176349e+03])
```

```
test_df2 = pd.DataFrame(X2_test,columns = ['x'+str(i) for i in
range(len(X2_test[0]))])
test_df2['err'] = y_pred2-y_test
x_df2 = test_df2.melt(var_name = 'feature',id_vars=['err'])
sns.relplot(data= x_df2,x='value',col='feature', col_wrap=4,y='err')
```

```
<seaborn.axisgrid.FacetGrid at 0x7efc68563e20>
```





From this we can see that most of these features do not vary much at all.

```
X2_test.shape
```

```
(20, 65)
```

22.3. Sparse Regression

```
lasso = linear_model.Lasso()
```

```
lasso.fit(X_train,y_train)
```

▼ Lasso
Lasso()

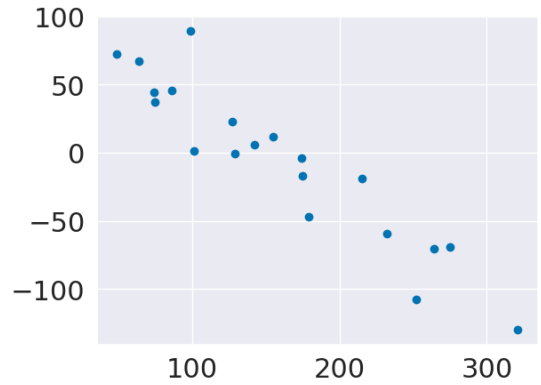
```
y_pred_lasso = lasso.predict(X_test)
```

```
lasso.coef_
```

```
array([  0.        ,  -0.        , 358.27705585,   9.71139556,
         0.        ,   0.        ,  -0.        ,   0.        ,
       309.50698469,   0.        ])
```

```
plt.scatter(y_test,y_pred_lasso-y_test)
```

```
<matplotlib.collections.PathCollection at 0x7efc1c5cfd90>
```




```
lasso.score(X_test,y_test)
```

```
0.42249260665177746
```

```
regr_db.score(X_test,y_test)
```

```
0.5195333332288746
```

```
lassoa2 = linear_model.Lasso(.3)
```

```
lassoa2.fit(X_train,y_train)
```

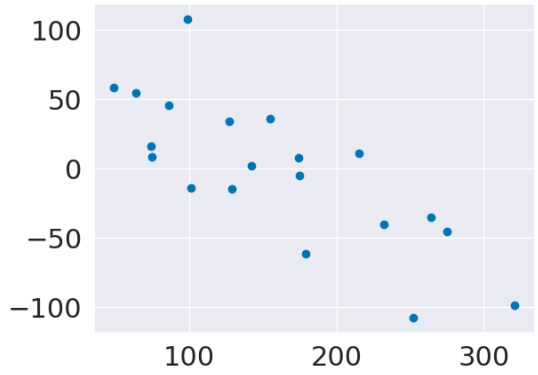
```
▾ Lasso
Lasso(alpha=0.3)
```

```
lassoa2.coef_
```

```
array([   0.        ,  -11.95340191,  500.51749129,  196.16550764,
         -0.        ,   -0.        , -118.56561325,    0.        ,
        435.84950435,    0.        ])
```

```
y_pred_lasso2 = lassoa2.predict(X_test)
plt.scatter(y_test,y_pred_lasso2-y_test)
```

```
<matplotlib.collections.PathCollection at 0x7efc1cd51d00>
```




```
lassoa2.score(X_test,y_test)
```

```
0.549634807596075
```

```
lassoa2.score(X_train,y_train)
```

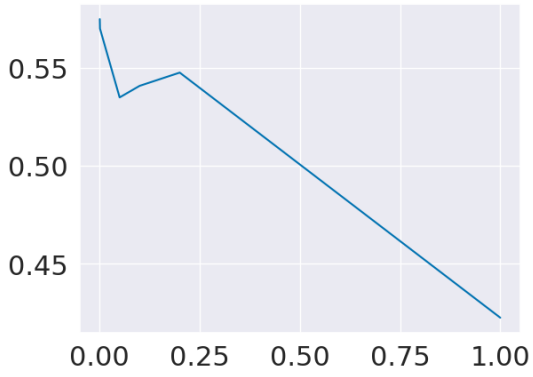
```
0.47656951646350165
```

22.4. Combining them

```
scores = []
alpha_vals = [1,.2,.1,.05,.001,.0005]
for alpha in alpha_vals:
    lasso2 = linear_model.Lasso(alpha)
    lasso2.fit(X2_train,y_train)
    scores.append(lasso2.score(X2_test,y_test))

plt.plot(alpha_vals,scores,)
```

```
/opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 2.855e+05, tolerance: 2.501e+02
  model = cd_fast.enet_coordinate_descent(
/opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 3.623e+05, tolerance: 2.501e+02
  model = cd_fast.enet_coordinate_descent(
```

```
[<matplotlib.lines.Line2D at 0x7efc1cc193d0>]
```



```
lasso2 = linear_model.Lasso(.001)
lasso2.fit(X2_train,y_train)
lasso2.score(X2_test,y_test)
```

```
/opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/sklearn/linear_model/_coordinate_descent.py:648: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 2.855e+05, tolerance: 2.501e+02
  model = cd_fast.enet_coordinate_descent(
```

```
0.5700228082080702
```

```
lasso2.coef_
```

```
array([    7.2847557 ,  -265.03975714,   493.61070188,   349.64975297,
       -1615.37347603,  1235.50131986,   395.29739908,   180.43060963,
        1044.09619028,    85.9484476 ,  1459.71954915,  3939.7666462 ,
           0.        ,    62.32177997,    -0.        , -1303.73035699,
        1552.1397576 ,   104.05324926,  1360.85779052,   846.76212154,
          -0.        ,   981.75629416,  1279.46510906,     0.        ,
        -304.4411084 ,  1774.14494353,  -462.17132174,     0.        ,
           0.        ,   257.09550983,  2689.80253642,  -273.88698246,
          -0.        ,    -0.        ,     0.        ,   194.70180318,
          -0.        ,   -19.31963832,   499.65744841,     0.        ,
         409.75290934,    -0.        ,   206.38881023, -1512.15365558,
           0.        ,   964.06977666,     0.        , -3929.82319169,
          -0.        ,     0.        ,     0.        ,  -742.73744294,
           0.        ,  2350.36850741,     0.        ,     0.        ,
          -0.        ,   586.77368375,   538.26423584,   984.9462285 ,
       -1134.51637764,  1454.88328498,  2559.00186391,     0.        ,
        1590.6765506 ])
```

22.5. Tips example

```
tips = sns.load_dataset("tips").dropna()
tips_X = tips['total_bill'].values
tips_X = tips_X[:,np.newaxis] # add an axis
tips_y = tips['tip']

tips_X_train,tips_X_test, tips_y_train, tips_y_test = train_test_split(
                    tips_X,
                    tips_y,
                    train_size=.8,
                    random_state=0)
```

In this example, we have one feature

```
regr_tips = linear_model.LinearRegression()
regr_tips.fit(tips_X_train,tips_y_train)
regr_tips.score(tips_X_test,tips_y_test)
```

```
0.5906895098589039
```

```
regr_tips.coef_, regr_tips.intercept_
```

```
(array([0.0968534]), 1.0285439454607272)
```

We can interpret this as that the expected tips is $1.02 + 9.7%.

```
tip_poly = PolynomialFeatures(include_bias=False,)
tips_X2_train = tip_poly.fit_transform(tips_X_train)
tips_X2_test = tip_poly.transform(tips_X_test)
```

These people were not good tippers

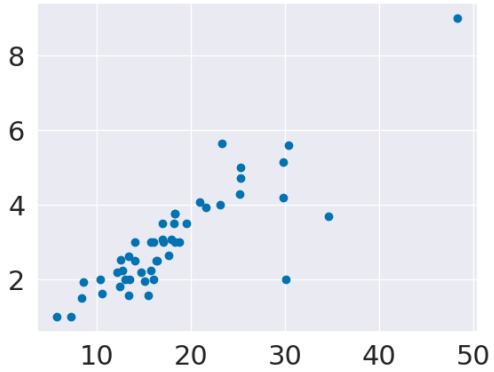
```
regr2_tips = linear_model.LinearRegression()
regr2_tips.fit(tips_X2_train,tips_y_train)
regr2_tips.score(tips_X2_test,tips_y_test)
```

```
0.5907071399659565
```

In this case, it doesn't do any better, so the relationship is really linear.

```
plt.scatter(tips_X_test,tips_y_test)
```

```
<matplotlib.collections.PathCollection at 0x7efc1cc265e0>
```



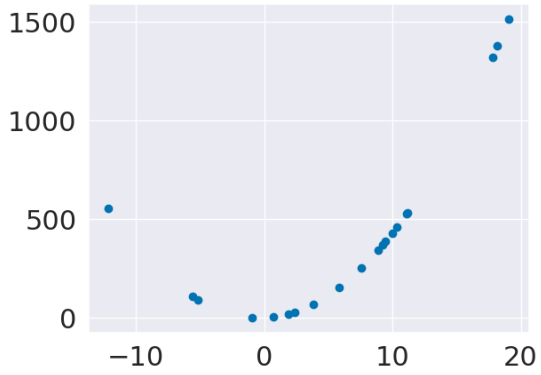
22.6. Toy example for visualization

we'll sample some random values for x and then add a coefficient to the value and the square

```
N = 20
slope = 3
sq = 4
sample_x = lambda num_smaples: np.random.normal(5,scale=10,size = num_smaples)
compute_y = lambda cx: slope*cx + sq*cx*cx
x_train = sample_x(N)[:,np.newaxis]
y_train = compute_y(x_train)

x_test = sample_x(N)[:,np.newaxis]
y_test = compute_y(x_test)
```

```
plt.scatter(x_train,y_train)
```

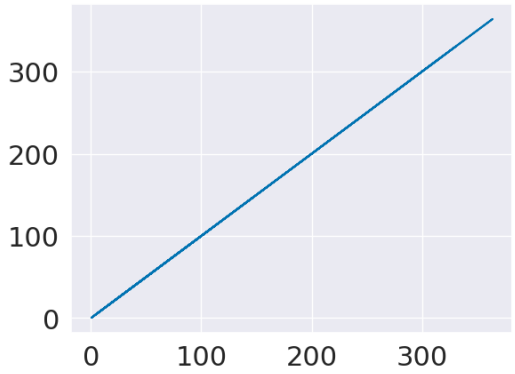
```
<matplotlib.collections.PathCollection at 0x7efc18544400>
```



Now, we'll transform, since this is one value

```
pf = PolynomialFeatures(include_bias=False,)
x2_train = pf.fit_transform(x_train,)
x2_test = pf.transform(x_test,)

plt.plot(x_train*x_train, x2_train[:,1])
```

```
[<matplotlib.lines.Line2D at 0x7efc184a89a0>]
```




```
regr = linear_model.LinearRegression()
regr.fit(x2_train,y_train)

regr.score(x2_test,y_test)
```

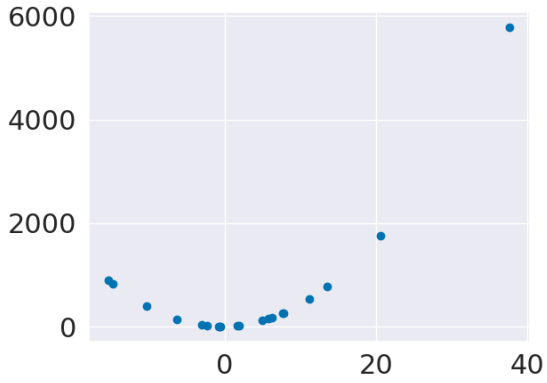
```
1.0
```

```
regr.coef_
```

```
array([[3., 4.]])
```

```
y_pred = regr.predict(x2_test)
plt.scatter(x_test,y_pred)
```

```
<matplotlib.collections.PathCollection at 0x7efc1848a760>
```



23. Clustering

Clustering is unsupervised learning. That means we do not have the labels to learn from. We aim to learn both the labels for each point and some way of characterizing the classes at the same time.

Computationally, this is a harder problem. Mathematically, we can typically solve problems when we have a number of equations equal to or greater than the number of unknowns. For N data points ind d dimensions and K clusters, we have N equations and $N + K * d$ unknowns. This means we have a harder problem to solve.

For today, we'll see K-means clustering which is defined by K a number of clusters and a mean (center) for each one. There are other K-centers algorithms for other types of centers.

Clustering is a stochastic (random) algorithm, so it can be a little harder to debug the models and measure performance. For this reason, we are going to lootk a little more closely at what it actuall *does* than we have with classification or regression.

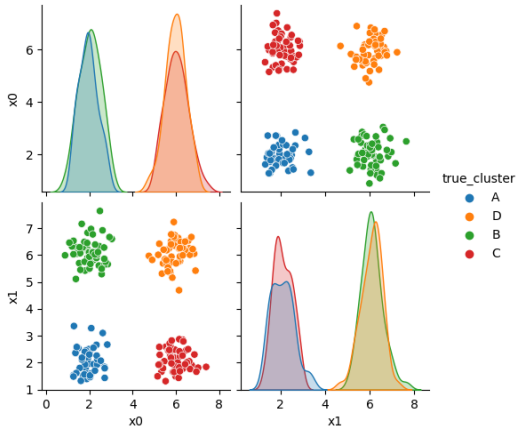
```
import matplotlib.pyplot as plt
import numpy as np
import itertools as it
import seaborn as sns
from sklearn import datasets
from sklearn.cluster import KMeans
import string
import pandas as pd
```

23.1. How does Kmeans work?

We will start with some synthetics data and then see how the clustering works.

```
C = 4
N = 200
offset = 2
spacing = 2

# choose the first C uppcase letters
classes = list(string.ascii_uppercase[:C])
# get the number of grid locations needed
G = int(np.ceil(np.sqrt(C)))
# get the locations for each axis
grid_locs = a = np.linspace(offset,offset+G*spacing,G)
# compute grid (i,j) for each combination of values above & keep C values
means = [(i,j) for i, j in it.product(grid_locs,grid_locs)][:C]
# store in dictionary with class labels
mu = {c: i for c, i in zip(classes,means)}
# random variances
sigma = {c: i*.5 for c, i in zip(classes,np.random.random(4))}

#randomly choose a class for each point, with equal probability
clusters_true = np.random.choice(classes,N)
# draw a randome point according to the means from above for each point
data = [np.random.multivariate_normal(mu[c],.25*np.eye(2)) for c in clusters_true]
# rounding to make display neater later
df = pd.DataFrame(data = data,columns = ['x' + str(i) for i in range(2)]).round(2)

# store in dataFram
df['true_cluster'] = clusters_true

sns.pairplot(data =df, hue='true_cluster')
```

```
<seaborn.axisgrid.PairGrid at 0x7f531836d880>
```




```
df.head()
```

	x0	x1	true_cluster
0	1.95	2.24	A
1	6.10	6.28	D
2	6.26	6.27	D
3	1.96	5.47	B
4	1.54	6.07	B

23.2. Kmeans

Next, we'll pick 4 random points to be the starting points as the means.

```
K = 4
mu0 = df.sample(n=K).values
mu0
```

Note

This code should be fairly readable, but is not strictly required. One tricky part is `zip` which is a builtin fuction for iteratinv over groups of things together.

And `itertools` is a core library for related more iterating.

```
array([[5.18, 6.01, 'D'],
       [5.43, 6.2, 'D'],
       [2.69, 5.86, 'B'],
       [2.01, 2.38, 'A']], dtype=object)
```

Now, we will compute, fo each sample which of those four points it is closest to first by taking the difference, squaring it, then summing along each row.

```
[((df-mu_i)**2).sum(axis=1) for mu_i in mu0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:165, in _na_arithmetic_op(left, right, op,
is_cmp)
    164 try:
--> 165     result = func(left, right)
    166 except TypeError:

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/computation/expressions.py:241, in evaluate(op, a, b,
use_numexpr)
    239     if use_numexpr:
    240         # error: "None" not callable
--> 241         return _evaluate(op, op_str, a, b)  # type: ignore[misc]
    242 return _evaluate_standard(op, op_str, a, b)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/computation/expressions.py:70, in _evaluate_standard(op,
op_str, a, b)
     69     _store_test_result(False)
---> 70 return op(a, b)

TypeError: unsupported operand type(s) for -: 'str' and 'str'

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
Cell In [5], line 1
----> 1 [((df-mu_i)**2).sum(axis=1) for mu_i in mu0]

Cell In [5], line 1, in <listcomp>(.0)
----> 1 [((df-mu_i)**2).sum(axis=1) for mu_i in mu0]

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/common.py:72, in _unpack_zerodim_and_defer.
<locals>.new_method(self, other)
     68             return NotImplemented
     70 other = item_from_zerodim(other)
---> 72 return method(self, other)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/arraylike.py:110, in OpsMixin.__sub__(self, other)
    108 @unpack_zerodim_and_defer("__sub__")
    109 def __sub__(self, other):
--> 110     return self._arith_method(other, operator.sub)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:7583, in DataFrame._arith_method(self, other, op)
   7579 other = ops.maybe_prepare_scalar_for_op(other, (self.shape[axis],))
   7581 self, other = ops.align_method_FRAME(self, other, axis, flex=True,
level=None)
-> 7583 new_data = self._dispatch_frame_op(other, op, axis=axis)
   7584 return self._construct_result(new_data)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:7622, in DataFrame._dispatch_frame_op(self, right,
func, axis)
   7616     # TODO: The previous assertion `assert right._indexed_same(self)`
   7617     #  fails in cases with empty columns reached via
   7618     #  _frame_arith_method_with_reindex
   7619
   7620     # TODO operate_blockwise expects a manager of the same type
   7621     with np.errstate(all="ignore"):
-> 7622         bm = self._mgr.operate_blockwise(
   7623             # error: Argument 1 to "operate_blockwise" of "ArrayManager"
has
   7624             # incompatible type "Union[ArrayManager, BlockManager]";
expected
   7625             # "ArrayManager"
   7626             # error: Argument 1 to "operate_blockwise" of "BlockManager"
has
   7627             # incompatible type "Union[ArrayManager, BlockManager]";
expected
   7628             # "BlockManager"
   7629             right._mgr,  # type: ignore[arg-type]
   7630             array_op,
   7631         )
   7632     return self._constructor(bm)
   7634 elif isinstance(right, Series) and axis == 1:
   7635     # axis=1 means we want to operate row-by-row

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/internals/managers.py:1565, in
BlockManager.operate_blockwise(self, other, array_op)
   1561 def operate_blockwise(self, other: BlockManager, array_op) ->
BlockManager:
   1562     """
   1563     Apply array_op blockwise with another (aligned) BlockManager.
   1564     """
-> 1565     return operate_blockwise(self, other, array_op)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/internals/ops.py:63, in operate_blockwise(left, right,
array_op)
     61 res_blks: list[Block] = []
     62 for lvals, rvals, locs, left_ea, right_ea, rblk in _iter_block_pairs(left,
right):
---> 63     res_values = array_op(lvals, rvals)
     64     if left_ea and not right_ea and hasattr(res_values, "reshape"):
     65         res_values = res_values.reshape(1, -1)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:226, in arithmetic_op(left, right, op)
    222     _bool_arith_check(op, left, right)
    224     # error: Argument 1 to "_na_arithmetic_op" has incompatible type
    225     # "Union[ExtensionArray, ndarray[Any, Any]]"; expected "ndarray[Any,
Any]"
--> 226     res_values = _na_arithmetic_op(left, right, op)  # type: ignore[arg-
type]
    228 return res_values

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:172, in _na_arithmetic_op(left, right, op,
is_cmp)
    166 except TypeError:
    167     if not is_cmp and (is_object_dtype(left.dtype) or
is_object_dtype(right)):
```

```
    168         # For object dtype, fallback to a masked operation (only operating
    169         #  on the non-missing values)
    170         # Don't do this for comparisons, as that will handle complex
numbers
    171         #  incorrectly, see GH#32047
--> 172         result = _masked_arith_op(left, right, op)
    173     else:
    174         raise

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:110, in _masked_arith_op(x, y, op)
    108     # See GH#5284, GH#5035, GH#19448 for historical reference
    109     if mask.any():
--> 110         result[mask] = op(xrav[mask], yrav[mask])
    112 else:
    113     if not is_scalar(y):

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

This gives us a list of 4 data DataFrames, one for each mean (mu), with one row for each point in the dataset with the distance from that point to the corresponding mean. We can stack these into one DataFrame.

```
pd.concat([((df-mu_i)**2).sum(axis=1) for mu_i in mu0],axis=1).head()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:165, in _na_arithmetic_op(left, right, op,
is_cmp)
    164 try:
--> 165     result = func(left, right)
    166 except TypeError:

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/computation/expressions.py:241, in evaluate(op, a, b,
use_numexpr)
    239     if use_numexpr:
    240         # error: "None" not callable
--> 241         return _evaluate(op, op_str, a, b)  # type: ignore[misc]
    242 return _evaluate_standard(op, op_str, a, b)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/computation/expressions.py:70, in _evaluate_standard(op,
op_str, a, b)
     69     _store_test_result(False)
---> 70 return op(a, b)

TypeError: unsupported operand type(s) for -: 'str' and 'str'

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
Cell In [6], line 1
----> 1 pd.concat([((df-mu_i)**2).sum(axis=1) for mu_i in mu0],axis=1).head()

Cell In [6], line 1, in <listcomp>(.0)
----> 1 pd.concat([((df-mu_i)**2).sum(axis=1) for mu_i in mu0],axis=1).head()

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/common.py:72, in _unpack_zerodim_and_defer.
<locals>.new_method(self, other)
     68             return NotImplemented
     70 other = item_from_zerodim(other)
---> 72 return method(self, other)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/arraylike.py:110, in OpsMixin.__sub__(self, other)
    108 @unpack_zerodim_and_defer("__sub__")
    109 def __sub__(self, other):
--> 110     return self._arith_method(other, operator.sub)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:7583, in DataFrame._arith_method(self, other, op)
   7579 other = ops.maybe_prepare_scalar_for_op(other, (self.shape[axis],))
   7581 self, other = ops.align_method_FRAME(self, other, axis, flex=True,
level=None)
-> 7583 new_data = self._dispatch_frame_op(other, op, axis=axis)
   7584 return self._construct_result(new_data)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:7622, in DataFrame._dispatch_frame_op(self, right,
func, axis)
   7616     # TODO: The previous assertion `assert right._indexed_same(self)`
   7617     #  fails in cases with empty columns reached via
   7618     #  _frame_arith_method_with_reindex
   7619
   7620     # TODO operate_blockwise expects a manager of the same type
   7621     with np.errstate(all="ignore"):
-> 7622         bm = self._mgr.operate_blockwise(
   7623             # error: Argument 1 to "operate_blockwise" of "ArrayManager"
has
   7624             # incompatible type "Union[ArrayManager, BlockManager]";
expected
   7625             # "ArrayManager"
   7626             # error: Argument 1 to "operate_blockwise" of "BlockManager"
has
   7627             # incompatible type "Union[ArrayManager, BlockManager]";
expected
   7628             # "BlockManager"
   7629             right._mgr,  # type: ignore[arg-type]
   7630             array_op,
   7631         )
   7632     return self._constructor(bm)
   7634 elif isinstance(right, Series) and axis == 1:
   7635     # axis=1 means we want to operate row-by-row

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/internals/managers.py:1565, in
BlockManager.operate_blockwise(self, other, array_op)
   1561 def operate_blockwise(self, other: BlockManager, array_op) ->
BlockManager:
   1562     """
   1563     Apply array_op blockwise with another (aligned) BlockManager.
   1564     """
-> 1565     return operate_blockwise(self, other, array_op)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/internals/ops.py:63, in operate_blockwise(left, right,
array_op)
     61 res_blks: list[Block] = []
     62 for lvals, rvals, locs, left_ea, right_ea, rblk in _iter_block_pairs(left,
right):
---> 63     res_values = array_op(lvals, rvals)
     64     if left_ea and not right_ea and hasattr(res_values, "reshape"):
     65         res_values = res_values.reshape(1, -1)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
```

```
packages/pandas/core/ops/array_ops.py:226, in arithmetic_op(left, right, op)
    222     _bool_arith_check(op, left, right)
    224     # error: Argument 1 to "_na_arithmetic_op" has incompatible type
    225     # "Union[ExtensionArray, ndarray[Any, Any]]"; expected "ndarray[Any,
Any]"
--> 226     res_values = _na_arithmetic_op(left, right, op)  # type: ignore[arg-
type]
    228 return res_values

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:172, in _na_arithmetic_op(left, right, op,
is_cmp)
    166 except TypeError:
    167     if not is_cmp and (is_object_dtype(left.dtype) or
is_object_dtype(right)):
    168         # For object dtype, fallback to a masked operation (only operating
    169         #  on the non-missing values)
    170         # Don't do this for comparisons, as that will handle complex
numbers
    171         #  incorrectly, see GH#32047
--> 172         result = _masked_arith_op(left, right, op)
    173     else:
    174         raise

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:110, in _masked_arith_op(x, y, op)
    108     # See GH#5284, GH#5035, GH#19448 for historical reference
    109     if mask.any():
--> 110         result[mask] = op(xrav[mask], yrav[mask])
    112 else:
    113     if not is_scalar(y):

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

Now we have one row per sample and one column per mean, with with the distance from that point to the mean. What we want is to calculate the assignment, which mean is closest, for each point. Using `idxmin` with `axis=1` we take the minimum across each row and returns the index (location) of that minimum.

```
pd.concat([((df-mu_i)**2).sum(axis=1) for mu_i in
mu0],axis=1).idxmin(axis=1).head()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:165, in _na_arithmetic_op(left, right, op,
is_cmp)
    164 try:
--> 165     result = func(left, right)
    166 except TypeError:

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/computation/expressions.py:241, in evaluate(op, a, b,
use_numexpr)
    239     if use_numexpr:
    240         # error: "None" not callable
--> 241         return _evaluate(op, op_str, a, b)  # type: ignore[misc]
    242 return _evaluate_standard(op, op_str, a, b)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/computation/expressions.py:70, in _evaluate_standard(op,
op_str, a, b)
     69     _store_test_result(False)
---> 70 return op(a, b)

TypeError: unsupported operand type(s) for -: 'str' and 'str'

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
Cell In [7], line 1
----> 1 pd.concat([((df-mu_i)**2).sum(axis=1) for mu_i in
mu0],axis=1).idxmin(axis=1).head()

Cell In [7], line 1, in <listcomp>(.0)
----> 1 pd.concat([((df-mu_i)**2).sum(axis=1) for mu_i in
mu0],axis=1).idxmin(axis=1).head()

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/common.py:72, in _unpack_zerodim_and_defer.
<locals>.new_method(self, other)
     68             return NotImplemented
     70 other = item_from_zerodim(other)
---> 72 return method(self, other)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/arraylike.py:110, in OpsMixin.__sub__(self, other)
    108 @unpack_zerodim_and_defer("__sub__")
    109 def __sub__(self, other):
--> 110     return self._arith_method(other, operator.sub)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:7583, in DataFrame._arith_method(self, other, op)
   7579 other = ops.maybe_prepare_scalar_for_op(other, (self.shape[axis],))
   7581 self, other = ops.align_method_FRAME(self, other, axis, flex=True,
level=None)
-> 7583 new_data = self._dispatch_frame_op(other, op, axis=axis)
   7584 return self._construct_result(new_data)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:7622, in DataFrame._dispatch_frame_op(self, right,
func, axis)
   7616     # TODO: The previous assertion `assert right._indexed_same(self)`
   7617     #  fails in cases with empty columns reached via
   7618     #  _frame_arith_method_with_reindex
   7619
   7620     # TODO operate_blockwise expects a manager of the same type
   7621     with np.errstate(all="ignore"):
-> 7622         bm = self._mgr.operate_blockwise(
   7623             # error: Argument 1 to "operate_blockwise" of "ArrayManager"
has
   7624             # incompatible type "Union[ArrayManager, BlockManager]";
expected
   7625             # "ArrayManager"
   7626             # error: Argument 1 to "operate_blockwise" of "BlockManager"
has
   7627             # incompatible type "Union[ArrayManager, BlockManager]";
expected
   7628             # "BlockManager"
   7629             right._mgr,  # type: ignore[arg-type]
   7630             array_op,
   7631         )
   7632     return self._constructor(bm)
   7634 elif isinstance(right, Series) and axis == 1:
   7635     # axis=1 means we want to operate row-by-row

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/internals/managers.py:1565, in
```

```
BlockManager.operate_blockwise(self, other, array_op)
   1561 def operate_blockwise(self, other: BlockManager, array_op) ->
BlockManager:
   1562     """
   1563     Apply array_op blockwise with another (aligned) BlockManager.
   1564     """
-> 1565     return operate_blockwise(self, other, array_op)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/internals/ops.py:63, in operate_blockwise(left, right,
array_op)
     61 res_blks: list[Block] = []
     62 for lvals, rvals, locs, left_ea, right_ea, rblk in _iter_block_pairs(left,
right):
---> 63     res_values = array_op(lvals, rvals)
     64     if left_ea and not right_ea and hasattr(res_values, "reshape"):
     65         res_values = res_values.reshape(1, -1)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:226, in arithmetic_op(left, right, op)
    222     _bool_arith_check(op, left, right)
    224     # error: Argument 1 to "_na_arithmetic_op" has incompatible type
    225     # "Union[ExtensionArray, ndarray[Any, Any]]"; expected "ndarray[Any,
Any]"
--> 226     res_values = _na_arithmetic_op(left, right, op)  # type: ignore[arg-
type]
    228 return res_values

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:172, in _na_arithmetic_op(left, right, op,
is_cmp)
    166 except TypeError:
    167     if not is_cmp and (is_object_dtype(left.dtype) or
is_object_dtype(right)):
    168         # For object dtype, fallback to a masked operation (only operating
    169         #  on the non-missing values)
    170         # Don't do this for comparisons, as that will handle complex
numbers
    171         #  incorrectly, see GH#32047
--> 172         result = _masked_arith_op(left, right, op)
    173     else:
    174         raise

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:110, in _masked_arith_op(x, y, op)
    108     # See GH#5284, GH#5035, GH#19448 for historical reference
    109     if mask.any():
--> 110         result[mask] = op(xrav[mask], yrav[mask])
    112 else:
    113     if not is_scalar(y):

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

We'll save all of this in a column named `'0'`. Since it is our 0th iteration.

```
df['0'] = pd.concat([((df-mu_i)**2).sum(axis=1) for mu_i in
mu0],axis=1).idxmin(axis=1)

df.head()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:165, in _na_arithmetic_op(left, right, op,
is_cmp)
    164 try:
--> 165     result = func(left, right)
    166 except TypeError:

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/computation/expressions.py:241, in evaluate(op, a, b,
use_numexpr)
    239     if use_numexpr:
    240         # error: "None" not callable
--> 241         return _evaluate(op, op_str, a, b)  # type: ignore[misc]
    242 return _evaluate_standard(op, op_str, a, b)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/computation/expressions.py:70, in _evaluate_standard(op,
op_str, a, b)
     69     _store_test_result(False)
---> 70 return op(a, b)

TypeError: unsupported operand type(s) for -: 'str' and 'str'

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
Cell In [8], line 1
----> 1 df['0'] = pd.concat([((df-mu_i)**2).sum(axis=1) for mu_i in
mu0],axis=1).idxmin(axis=1)
      3 df.head()

Cell In [8], line 1, in <listcomp>(.0)
----> 1 df['0'] = pd.concat([((df-mu_i)**2).sum(axis=1) for mu_i in
mu0],axis=1).idxmin(axis=1)
      3 df.head()

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/common.py:72, in _unpack_zerodim_and_defer.
<locals>.new_method(self, other)
     68             return NotImplemented
     70 other = item_from_zerodim(other)
---> 72 return method(self, other)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/arraylike.py:110, in OpsMixin.__sub__(self, other)
    108 @unpack_zerodim_and_defer("__sub__")
    109 def __sub__(self, other):
--> 110     return self._arith_method(other, operator.sub)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:7583, in DataFrame._arith_method(self, other, op)
   7579 other = ops.maybe_prepare_scalar_for_op(other, (self.shape[axis],))
   7581 self, other = ops.align_method_FRAME(self, other, axis, flex=True,
level=None)
-> 7583 new_data = self._dispatch_frame_op(other, op, axis=axis)
   7584 return self._construct_result(new_data)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:7622, in DataFrame._dispatch_frame_op(self, right,
func, axis)
   7616     # TODO: The previous assertion `assert right._indexed_same(self)`
   7617     #  fails in cases with empty columns reached via
   7618     #  _frame_arith_method_with_reindex
   7619
   7620     # TODO operate_blockwise expects a manager of the same type
   7621     with np.errstate(all="ignore"):
```

```
-> 7622         bm = self._mgr.operate_blockwise(
   7623             # error: Argument 1 to "operate_blockwise" of "ArrayManager"
has
   7624             # incompatible type "Union[ArrayManager, BlockManager]";
expected
   7625             # "ArrayManager"
   7626             # error: Argument 1 to "operate_blockwise" of "BlockManager"
has
   7627             # incompatible type "Union[ArrayManager, BlockManager]";
expected
   7628             # "BlockManager"
   7629             right._mgr,  # type: ignore[arg-type]
   7630             array_op,
   7631         )
   7632     return self._constructor(bm)
   7634 elif isinstance(right, Series) and axis == 1:
   7635     # axis=1 means we want to operate row-by-row

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/internals/managers.py:1565, in
BlockManager.operate_blockwise(self, other, array_op)
   1561 def operate_blockwise(self, other: BlockManager, array_op) ->
BlockManager:
   1562     """
   1563     Apply array_op blockwise with another (aligned) BlockManager.
   1564     """
-> 1565     return operate_blockwise(self, other, array_op)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/internals/ops.py:63, in operate_blockwise(left, right,
array_op)
     61 res_blks: list[Block] = []
     62 for lvals, rvals, locs, left_ea, right_ea, rblk in _iter_block_pairs(left,
right):
---> 63     res_values = array_op(lvals, rvals)
     64     if left_ea and not right_ea and hasattr(res_values, "reshape"):
     65         res_values = res_values.reshape(1, -1)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:226, in arithmetic_op(left, right, op)
    222     _bool_arith_check(op, left, right)
    224     # error: Argument 1 to "_na_arithmetic_op" has incompatible type
    225     # "Union[ExtensionArray, ndarray[Any, Any]]"; expected "ndarray[Any,
Any]"
--> 226     res_values = _na_arithmetic_op(left, right, op)  # type: ignore[arg-
type]
    228 return res_values

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:172, in _na_arithmetic_op(left, right, op,
is_cmp)
    166 except TypeError:
    167     if not is_cmp and (is_object_dtype(left.dtype) or
is_object_dtype(right)):
    168         # For object dtype, fallback to a masked operation (only operating
    169         #  on the non-missing values)
    170         # Don't do this for comparisons, as that will handle complex
numbers
    171         #  incorrectly, see GH#32047
--> 172         result = _masked_arith_op(left, right, op)
    173     else:
    174         raise

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/ops/array_ops.py:110, in _masked_arith_op(x, y, op)
    108     # See GH#5284, GH#5035, GH#19448 for historical reference
    109     if mask.any():
--> 110         result[mask] = op(xrav[mask], yrav[mask])
    112 else:
    113     if not is_scalar(y):

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

Here, we'll set up some helper code.

```
data_cols = ['x0','x1']
def mu_to_df(mu,i):
    mu_df = pd.DataFrame(mu,columns=data_cols)
    mu_df['iteration'] = str(i)
    mu_df['class'] = ['M'+str(i) for i in range(K)]
    mu_df['type'] = 'mu'
    return mu_df

# color maps, we select every other value from this palette that has 8 values & is
paird
cmap_pt = sns.color_palette('tab20',8)[1::2] # starting from 1
cmap_mu = sns.color_palette('tab20',8)[0::2] # starting form 0
```

Further Reading

For more on color palettes see the seaborn docs

Now we can plot the data, save the axis, and plot the means on top of that. Seaborn plotting functions return an axis, by saving that to a variable, we can pass it to the `ax` parameter of another plotting function so that both plotting functions go on the same figure.

```
sfig = sns.scatterplot(data=df,x='x0',y='x1', hue='0',
                       palette =cmap_pt,legend=False)
mu_df = mu_to_df(mu0,0)
sns.scatterplot(data=mu_df, x='x0',y='x1',hue='class',
                palette=cmap_mu,legend=False, ax=sfig)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [10], line 1
----> 1 sfig = sns.scatterplot(data=df,x='x0',y='x1', hue='0',
      2                        palette =cmap_pt,legend=False)
      3 mu_df = mu_to_df(mu0,0)
      4 sns.scatterplot(data=mu_df, x='x0',y='x1',hue='class',
      5                 palette=cmap_mu,legend=False, ax=sfig)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/relational.py:742, in scatterplot(data, x, y, hue, size, style,
palette, hue_order, hue_norm, sizes, size_order, size_norm, markers, style_order,
legend, ax, **kwargs)
    732 def scatterplot(
    733     data=None, *,
    734     x=None, y=None, hue=None, size=None, style=None,
   (...)
    738     **kwargs
    739 ):
    741     variables = _ScatterPlotter.get_semantics(locals())
--> 742     p = _ScatterPlotter(data=data, variables=variables, legend=legend)
    744     p.map_hue(palette=palette, order=hue_order, norm=hue_norm)
    745     p.map_size(sizes=sizes, order=size_order, norm=size_norm)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/relational.py:538, in _ScatterPlotter.__init__(self, data,
variables, legend)
    529 def __init__(self, *, data=None, variables={}, legend=None):
    530
    531     # TODO this is messy, we want the mapping to be agnostic about
    532     # the kind of plot to draw, but for the time being we need to set
    533     # this information so the SizeMapping can use it
    534     self._default_size_range = (
    535         np.r_[.5, 2] * np.square(mpl.rcParams["lines.markersize"])
    536     )
--> 538     super().__init__(data=data, variables=variables)
    540     self.legend = legend

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:640, in VectorPlotter.__init__(self, data, variables)
    635 # var_ordered is relevant only for categorical axis variables, and may
    636 # be better handled by an internal axis information object that tracks
    637 # such information and is set up by the scale_* methods. The analogous
    638 # information for numeric axes would be information about log scales.
    639 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 640 self.assign_variables(data, variables)
    642 for var, cls in self._semantic_mappings.items():
    643
    644     # Create the mapping function
    645     map_func = partial(cls.map, plotter=self)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:701, in VectorPlotter.assign_variables(self, data,
variables)
    699 else:
    700     self.input_format = "long"
--> 701     plot_data, variables = self._assign_variables_longform(
    702         data, **variables,
    703     )
    705 self.plot_data = plot_data
    706 self.variables = variables

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:938, in
VectorPlotter._assign_variables_longform(self, data, **kwargs)
    933 elif isinstance(val, (str, bytes)):
    934
    935     # This looks like a column name but we don't know what it means!
    937     err = f"Could not interpret value `{val}` for parameter `{key}`"
--> 938     raise ValueError(err)
    940 else:
    941
    942     # Otherwise, assume the value is itself data
    943
    944     # Raise when data object is present and a vector can't matched
    945     if isinstance(data, pd.DataFrame) and not isinstance(val, pd.Series):

ValueError: Could not interpret value `0` for parameter `hue`
```

We see that each point is assigned to the lighter shade of its matching mean. These points are the one that is closest to each point, but they're not the centers of the point clouds. Now, we can compute new means of the points assigned to each cluster, using groupby.

```
mu1 = df.groupby('0')[data_cols].mean().values
mu1
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In [11], line 1
----> 1 mu1 = df.groupby('0')[data_cols].mean().values
      2 mu1

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:8389, in DataFrame.groupby(self, by, axis, level,
as_index, sort, group_keys, squeeze, observed, dropna)
   8386     raise TypeError("You have to supply one of 'by' and 'level'")
   8387 axis = self._get_axis_number(axis)
-> 8389 return DataFrameGroupBy(
   8390     obj=self,
   8391     keys=by,
   8392     axis=axis,
   8393     level=level,
   8394     as_index=as_index,
   8395     sort=sort,
   8396     group_keys=group_keys,
   8397     squeeze=squeeze,
   8398     observed=observed,
   8399     dropna=dropna,
   8400 )

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/groupby/groupby.py:959, in GroupBy.__init__(self, obj, keys,
axis, level, grouper, exclusions, selection, as_index, sort, group_keys, squeeze,
observed, mutated, dropna)
    956 if grouper is None:
    957     from pandas.core.groupby.grouper import get_grouper
--> 959     grouper, exclusions, obj = get_grouper(
    960         obj,
    961         keys,
    962         axis=axis,
    963         level=level,
    964         sort=sort,
    965         observed=observed,
    966         mutated=self.mutated,
    967         dropna=self.dropna,
    968     )
    970 self.obj = obj
    971 self.axis = obj._get_axis_number(axis)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/groupby/grouper.py:888, in get_grouper(obj, key, axis, level,
sort, observed, mutated, validate, dropna)
    886         in_axis, level, gpr = False, gpr, None
    887     else:
--> 888         raise KeyError(gpr)
    889 elif isinstance(gpr, Grouper) and gpr.key is not None:
    890     # Add key to exclusions
    891     exclusions.add(gpr.key)

KeyError: '0'
```

We can plot these again, the same data, but with the new means.

```
sfig = sns.scatterplot(data=df,x='x0',y='x1', hue='0',
                       palette =cmap_pt,legend=False)
mu_df = mu_to_df(mu1,0)
sns.scatterplot(data=mu_df, x='x0',y='x1',hue='class',
                palette=cmap_mu,legend=False, ax=sfig)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [12], line 1
----> 1 sfig = sns.scatterplot(data=df,x='x0',y='x1', hue='0',
      2                        palette =cmap_pt,legend=False)
      3 mu_df = mu_to_df(mu1,0)
      4 sns.scatterplot(data=mu_df, x='x0',y='x1',hue='class',
      5                 palette=cmap_mu,legend=False, ax=sfig)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/relational.py:742, in scatterplot(data, x, y, hue, size, style,
palette, hue_order, hue_norm, sizes, size_order, size_norm, markers, style_order,
legend, ax, **kwargs)
    732 def scatterplot(
    733     data=None, *,
    734     x=None, y=None, hue=None, size=None, style=None,
   (...)
    738     **kwargs
    739 ):
    741     variables = _ScatterPlotter.get_semantics(locals())
--> 742     p = _ScatterPlotter(data=data, variables=variables, legend=legend)
    744     p.map_hue(palette=palette, order=hue_order, norm=hue_norm)
    745     p.map_size(sizes=sizes, order=size_order, norm=size_norm)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/relational.py:538, in _ScatterPlotter.__init__(self, data,
variables, legend)
    529 def __init__(self, *, data=None, variables={}, legend=None):
    530
    531     # TODO this is messy, we want the mapping to be agnostic about
    532     # the kind of plot to draw, but for the time being we need to set
    533     # this information so the SizeMapping can use it
    534     self._default_size_range = (
    535         np.r_[.5, 2] * np.square(mpl.rcParams["lines.markersize"])
    536     )
--> 538     super().__init__(data=data, variables=variables)
    540     self.legend = legend

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:640, in VectorPlotter.__init__(self, data, variables)
    635 # var_ordered is relevant only for categorical axis variables, and may
    636 # be better handled by an internal axis information object that tracks
    637 # such information and is set up by the scale_* methods. The analogous
    638 # information for numeric axes would be information about log scales.
    639 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 640 self.assign_variables(data, variables)
    642 for var, cls in self._semantic_mappings.items():
    643
    644     # Create the mapping function
    645     map_func = partial(cls.map, plotter=self)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:701, in VectorPlotter.assign_variables(self, data,
variables)
    699 else:
    700     self.input_format = "long"
--> 701     plot_data, variables = self._assign_variables_longform(
    702         data, **variables,
    703     )
    705 self.plot_data = plot_data
    706 self.variables = variables

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:938, in
VectorPlotter._assign_variables_longform(self, data, **kwargs)
    933 elif isinstance(val, (str, bytes)):
    934
    935     # This looks like a column name but we don't know what it means!
    937     err = f"Could not interpret value `{val}` for parameter `{key}`"
--> 938     raise ValueError(err)
    940 else:
    941
    942     # Otherwise, assume the value is itself data
    943
    944     # Raise when data object is present and a vector can't matched
    945     if isinstance(data, pd.DataFrame) and not isinstance(val, pd.Series):

ValueError: Could not interpret value `0` for parameter `hue`
```

We see that now the means are in the center of each cluster, but that there are now points in one color that are assigned to other clusters.

So, again we can update the assignments.

```
df['1'] = pd.concat([((df[data_cols]-mu_i)**2).sum(axis=1) for mu_i in
mu1],axis=1).idxmin(axis=1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [13], line 1
----> 1 df['1'] = pd.concat([((df[data_cols]-mu_i)**2).sum(axis=1) for mu_i in
mu1],axis=1).idxmin(axis=1)

NameError: name 'mu1' is not defined
```

And plot again.

```
sfig = sns.scatterplot(data=df,x='x0',y='x1', hue='0',
                       palette =cmap_pt,legend=False)
mu_df = mu_to_df(mu,0)
sns.scatterplot(data=mu_df, x='x0',y='x1',hue='class',
                palette=cmap_mu,legend=False, ax=sfig)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [14], line 1
----> 1 sfig = sns.scatterplot(data=df,x='x0',y='x1', hue='0',
      2                        palette =cmap_pt,legend=False)
      3 mu_df = mu_to_df(mu,0)
      4 sns.scatterplot(data=mu_df, x='x0',y='x1',hue='class',
      5                 palette=cmap_mu,legend=False, ax=sfig)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/relational.py:742, in scatterplot(data, x, y, hue, size, style,
palette, hue_order, hue_norm, sizes, size_order, size_norm, markers, style_order,
legend, ax, **kwargs)
    732 def scatterplot(
    733     data=None, *,
    734     x=None, y=None, hue=None, size=None, style=None,
   (...)
    738     **kwargs
    739 ):
    741     variables = _ScatterPlotter.get_semantics(locals())
--> 742     p = _ScatterPlotter(data=data, variables=variables, legend=legend)
    744     p.map_hue(palette=palette, order=hue_order, norm=hue_norm)
    745     p.map_size(sizes=sizes, order=size_order, norm=size_norm)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/relational.py:538, in _ScatterPlotter.__init__(self, data,
variables, legend)
    529 def __init__(self, *, data=None, variables={}, legend=None):
    530
    531     # TODO this is messy, we want the mapping to be agnostic about
    532     # the kind of plot to draw, but for the time being we need to set
    533     # this information so the SizeMapping can use it
    534     self._default_size_range = (
    535         np.r_[.5, 2] * np.square(mpl.rcParams["lines.markersize"])
    536     )
--> 538     super().__init__(data=data, variables=variables)
    540     self.legend = legend

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:640, in VectorPlotter.__init__(self, data, variables)
    635 # var_ordered is relevant only for categorical axis variables, and may
    636 # be better handled by an internal axis information object that tracks
    637 # such information and is set up by the scale_* methods. The analogous
    638 # information for numeric axes would be information about log scales.
    639 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 640 self.assign_variables(data, variables)
    642 for var, cls in self._semantic_mappings.items():
    643
    644     # Create the mapping function
    645     map_func = partial(cls.map, plotter=self)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:701, in VectorPlotter.assign_variables(self, data,
variables)
    699 else:
    700     self.input_format = "long"
--> 701     plot_data, variables = self._assign_variables_longform(
    702         data, **variables,
    703     )
    705 self.plot_data = plot_data
    706 self.variables = variables

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/seaborn/_oldcore.py:938, in
VectorPlotter._assign_variables_longform(self, data, **kwargs)
    933 elif isinstance(val, (str, bytes)):
    934
    935     # This looks like a column name but we don't know what it means!
    937     err = f"Could not interpret value `{val}` for parameter `{key}`"
--> 938     raise ValueError(err)
    940 else:
    941
    942     # Otherwise, assume the value is itself data
    943
    944     # Raise when data object is present and a vector can't matched
    945     if isinstance(data, pd.DataFrame) and not isinstance(val, pd.Series):

ValueError: Could not interpret value `0` for parameter `hue`
```

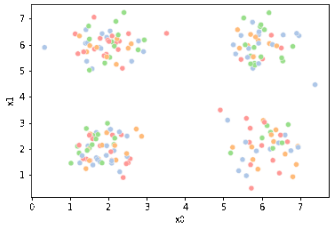
If we keep going back and forth like this, eventually, the assignment step will not change any assignments. We call this condition convergence. We can implement the algorithm with a while loop.

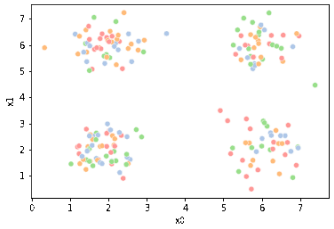
> **Correction**
>
> In the following I swapped the order of the mean update and assignment steps.
> My previous version had a different *initialization* (the above part) so it was okay for the steps to be in the other order.

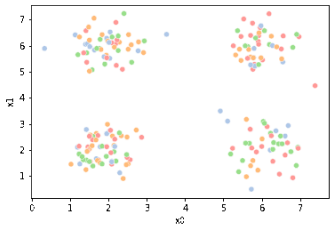
```
i =1
mu_list = [mu_to_df(mu0,i), mu_to_df(mu1,i)]
cur_old = str(i-1)
cur_new = str(i)
while sum(df[cur_old] !=df[cur_new]) >0:
    cur_old = cur_new
    i +=1
    cur_new = str(i)
        #     update the means and plot with current generating assignments
    mu = df.groupby(cur_old)[data_cols].mean().values
    mu_df = mu_to_df(mu,i)
    mu_list.append(mu_df)

    fig = plt.figure()
    sfig = sns.scatterplot(data
=df,x='x0',y='x1',hue=cur_old,palette=cmap_pt,legend=False)
    sns.scatterplot(data
=mu_df,x='x0',y='x1',hue='class',palette=cmap_mu,ax=sfig,legend=False)


    #     update the assigments and plot with the associated means
    df[cur_new] = pd.concat([((df[data_cols]-mu_i)**2).sum(axis=1) for mu_i in
mu],axis=1).idxmin(axis=1)
    fig = plt.figure()
    sfig = sns.scatterplot(data
=df,x='x0',y='x1',hue=cur_new,palette=cmap_pt,legend=False)
    sns.scatterplot(data
=mu_df,x='x0',y='x1',hue='class',palette=cmap_mu,ax=sfig,legend=False)


n_iter = i
```

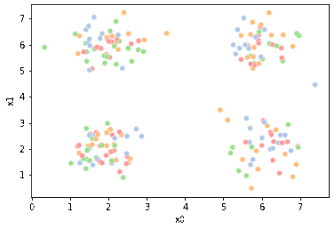
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [15], line 2
      1 i =1
----> 2 mu_list = [mu_to_df(mu0,i), mu_to_df(mu1,i)]
      3 cur_old = str(i-1)
      4 cur_new = str(i)

Cell In [9], line 3, in mu_to_df(mu, i)
      2 def mu_to_df(mu,i):
----> 3     mu_df = pd.DataFrame(mu,columns=data_cols)
      4     mu_df['iteration'] = str(i)
      5     mu_df['class'] = ['M'+str(i) for i in range(K)]

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/frame.py:720, in DataFrame.__init__(self, data, index,
columns, dtype, copy)
    710         mgr = dict_to_mgr(
    711             # error: Item "ndarray" of "Union[ndarray, Series, Index]" has
no
    712             # attribute "name"
   (...)
    717             typ=manager,
    718         )
    719     else:
--> 720         mgr = ndarray_to_mgr(
    721             data,
    722             index,
    723             columns,
    724             dtype=dtype,
    725             copy=copy,
    726             typ=manager,
    727         )
    729 # For data is list-like, or Iterable (will consume into list)
    730 elif is_list_like(data):

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/internals/construction.py:349, in ndarray_to_mgr(values,
index, columns, dtype, copy, typ)
    344 # _prep_ndarraylike ensures that values.ndim == 2 at this point
    345 index, columns = _get_axes(
    346     values.shape[0], values.shape[1], index=index, columns=columns
    347 )
--> 349 _check_values_indices_shape_match(values, index, columns)
    351 if typ == "array":
    353     if issubclass(values.dtype.type, str):

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/pandas/core/internals/construction.py:420, in
_check_values_indices_shape_match(values, index, columns)
    418 passed = values.shape
    419 implied = (len(index), len(columns))
--> 420 raise ValueError(f"Shape of passed values is {passed}, indices imply
{implied}")

ValueError: Shape of passed values is (4, 3), indices imply (4, 2)
```

This algorithm is random. So each time we run it, it looks a little different.









23.3. Questions After class

23.3.1. Are there any better way to optimize than run multiple times?

Not totally, the Kmeans ++ algorithm for initialization will help, but sometimes just multiple times is all we have because there is so much more unknown than known.

23.3.2. How do we use sklearn to cluster data?

We will see that Wednesday, but the short answer is that there is a an estimator object for the clustering model and then we use the `fit` method

24. Clustering with Sci-kit Learn

```
import seaborn as sns
import numpy as np
from sklearn import datasets
from sklearn.cluster import KMeans
import pandas as pd
sns.set_theme(palette='colorblind')

# set global random seed so that the notes are the same each time the site builds
np.random.seed(1103)
```

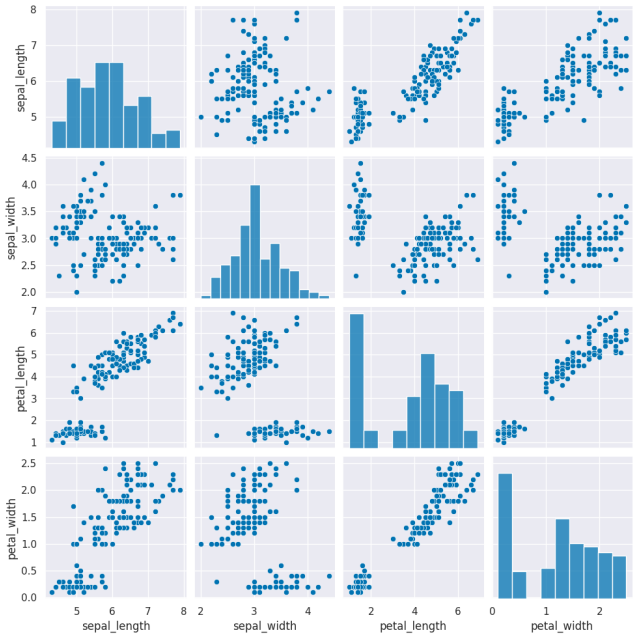
First we will load the iris data from Seaborn

```
iris_df = sns.load_dataset('iris')
```

To consider what our clustering algorithm sees, we will plot the grid of subplots without the species labeling the point.s.

```
sns.pairplot(data=iris_df)
```

```
<seaborn.axisgrid.PairGrid at 0x7fab14807fd0>
```



We need a copy of the data that's appropriate for clustering. Remember that clustering is *unsupervised* so it doesn't have a target variable. We also can do clustering on the data with or without splitting into test/train splits, since it doesn't use a target variable, we can evaluate how good the clusters it finds are on the actual data that it learned from.

```
iris_df.head(2)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa

> **Hint**
>
> We can either pick the measurements out or drop the species column. remember most data frame operations return a copy of the dataframe.

We'll do this by dropping the species for now, but we could have also selected the measurement columns.

```
iris_X = iris_df.drop(columns=['species'])
iris_X.head(1)
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2

25. KMeans Estimator

Create a Kmeans estimator object with 3 clusters, since we know that the iris data has 3 species of flowers. We refer to these three groups as classes in classification (the goal is to label the classes…) and in clustering we typically borrow that word. Sometimes, clustering literature will be more abstract and refer to partitions, this is especially common in more mathematical/statistical work as opposed to algorithmic work on clustering.

```
km = KMeans(n_clusters=3)
```

we use this to instantiate the object with the right number since we know that is correct.

> **Question**
>
> How do we know there are three classes? didnt' we just drop them?

We dropped the *column* that tells us which of the three classes that each sample(row) belongs to. We still have data from three species of flows.

> **Hint**
>
> use shift+tab or another jupyter help to figure out what the parameter names are for any function or class you're working with.

26. Fit and Predict

Since we don't have separate test and train data, we can use the `fit_predict` method. This is what the kmeans algorithm always does anyway, it both learns the means and the assignment (or prediction) for each sample at the same time. On Monday, that would be the last column of the dataframe, the one in the highest.

```
km.fit_predict(iris_X)
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2,
       2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1], dtype=int32)
```

```
iris_df['species'].values
```

```
array(['setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'virginica', 'virginica', 'virginica'], dtype=object)
```

This gives the labeled cluster by index, or the assignment, of each point.

These are similar to the outputs in classification, except that in classification, it's able to tell us a specific species for each. Here it can only say clust 0, 1, or 2. It can't match those groups to the species of flower.

Now that we know what these are, we can save them to a variable.

```
cluster_assignments = km.fit_predict(iris_X)
cluster_assignments
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2,
       2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1], dtype=int32)
```

Use the `get_params` method to look at the parameters. Read the documentation to see what they mean.

```
km.get_params(deep=True)
```

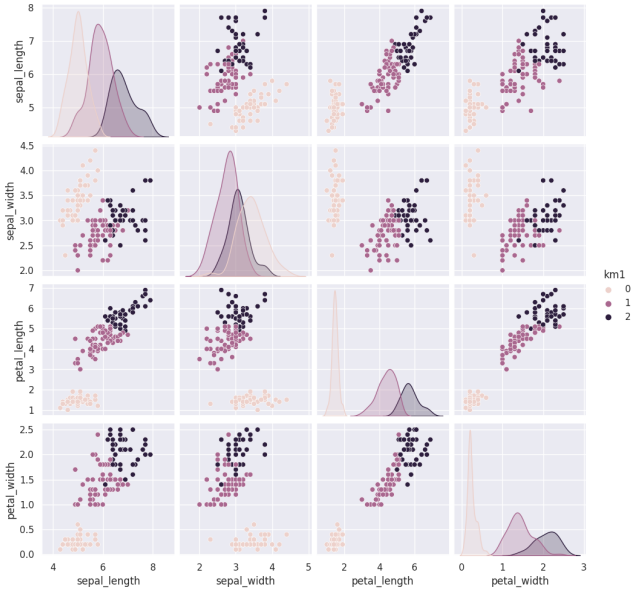
```
{'algorithm': 'lloyd',
 'copy_x': True,
 'init': 'k-means++',
 'max_iter': 300,
 'n_clusters': 3,
 'n_init': 10,
 'random_state': None,
 'tol': 0.0001,
 'verbose': 0}
```

27. Visualizing the outputs

Add the predictions as a new column to the original `iris_df` and make a `pairplot` with the points colored by what the clustering learned.

```
iris_df['km1'] = cluster_assignments
sns.pairplot(data=iris_df,hue='km1')
```

```
<seaborn.axisgrid.PairGrid at 0x7faad7514670>
```




```
iris_df['km2'] = km.fit_predict(iris_X)
```

```
iris_df.head(1)
```

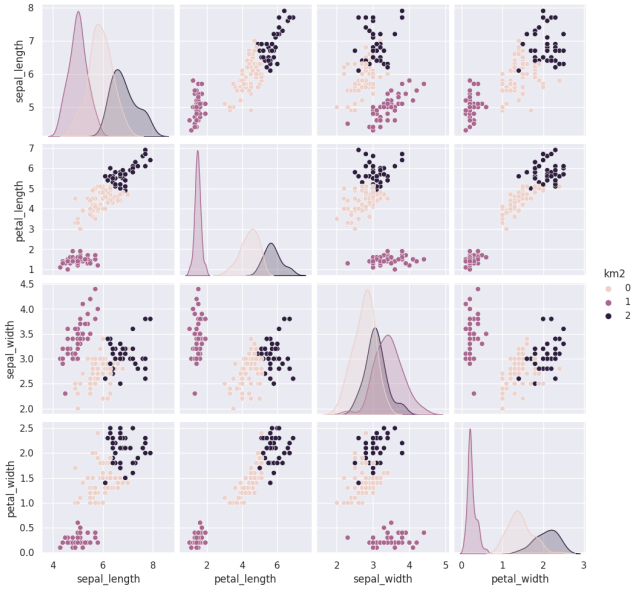
	sepal_length	sepal_width	petal_length	petal_width	species	km1	km2
0	5.1	3.5	1.4	0.2	setosa	0	1

```
measurement_cols = ['sepal_length','petal_length','sepal_width','petal_width']
```

We need to pick out only the measurement columns for plotting. The way have done this in the past is to subset the dataFrame

```
sns.pairplot(iris_df[measurement_cols+['km2']],hue='km2')
```

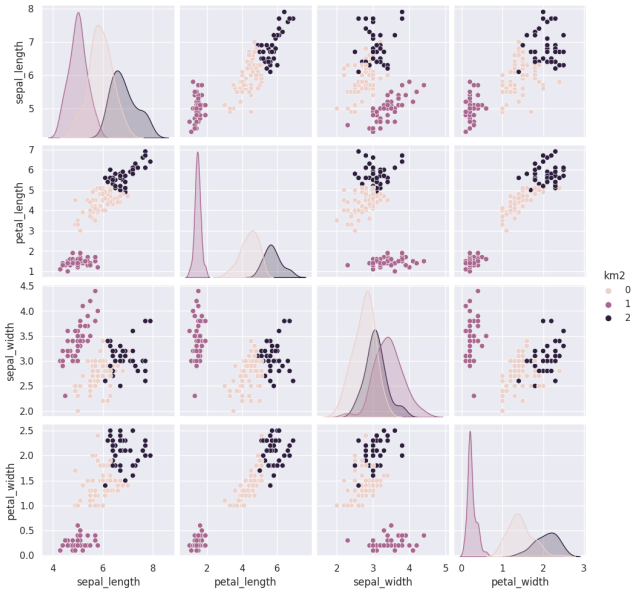
```
<seaborn.axisgrid.PairGrid at 0x7faad7dd38e0>
```



We can use the `vars` parameter to plot only the measurement columns and not the cluster labels. We didn't have to do this before, because `species` is strings, but the cluster predictions are also numerical, so by default seaborn plots them.

```
sns.pairplot(data=iris_df, hue='km2', vars=measurement_cols)
```

```
<seaborn.axisgrid.PairGrid at 0x7faaccc2b550>
```



28. Clustering Persistence

We can run kmeans a few more times and plot each time and/or compare with a neighbor/ another group.

```
for i in [3,4,5,6,7]:
    iris_df['km' + str(i)] = km.fit_predict(iris_X)
```

> **Tip**
>
> using the i as a loop variable here makes sense since we're actually just repeating for the sake of repeating

```
iris_df.sample(10)
```

	sepal_length	sepal_width	petal_length	petal_width	species	km1	km2	km3	km4	km5	km6	km7
116	6.5	3.0	5.5	1.8	virginica	2	2	2	2	2	2	2
92	5.8	2.6	4.0	1.2	versicolor	1	0	1	1	0	1	1
34	4.9	3.1	1.5	0.2	setosa	0	1	0	0	1	0	0
7	5.0	3.4	1.5	0.2	setosa	0	1	0	0	1	0	0
132	6.4	2.8	5.6	2.2	virginica	2	2	2	2	2	2	2
55	5.7	2.8	4.5	1.3	versicolor	1	0	1	1	0	1	1
45	4.8	3.0	1.4	0.3	setosa	0	1	0	0	1	0	0
96	5.7	2.9	4.2	1.3	versicolor	1	0	1	1	0	1	1
99	5.7	2.8	4.1	1.3	versicolor	1	0	1	1	0	1	1
141	6.9	3.1	5.1	2.3	virginica	2	2	2	2	2	2	2

The *grouping* of the points stay the same across different runs, but which color each group gets assigned to changes. Look at the 5th time compared to the ones before and 6 compared to that. Which blob is which color changes.

Today, we saw that the clustering solution was pretty similar each time in terms of which points were grouped together, but the labeling of the groups (which one was each number) was different each time. We also saw that clustering can only number the clusters, it can't match them with certainty to the species. This makes evaluating clustering somewhat different, so we need new metrics.

What might be our goal for evaluating clustering? We'll start from evaluating clustering on Friday.

29. Question After Class

29.1. How can we examine if the groups are the same each time?

One way using what we have already seen is to do it visually.

We could also groupby one of the `km` columns and then look at the `std` of the others.

```
iris_df.groupby('km1').std()
```

```
/tmp/ipykernel_2302/2783782698.py:1: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.std is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  iris_df.groupby('km1').std()
```

km1	sepal_length	sepal_width	petal_length	petal_width	km2	km3	km4	km5	km6	km7
0	0.352490	0.379064	0.173664	0.105386	0.0	0.0	0.0	0.0	0.0	0.0
1	0.466410	0.296284	0.508895	0.297500	0.0	0.0	0.0	0.0	0.0	0.0
2	0.494155	0.290092	0.488590	0.279872	0.0	0.0	0.0	0.0	0.0	0.0

30. Clutering Evalutation

Today, we will learn how to evaluate if a clustering model was effective or not. In the process, we will also learn how to determine if we have the right number of clusters.

```
import seaborn as sns
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn import metrics
import pandas as pd
```

Extract out the features to `iris_X`

```
iris_df = sns.load_dataset('iris')
iris_X = iris_df.drop(columns=['species'])
```

First we will fit the model with the correct number of clusters.

```
km3 = KMeans(n_clusters=3)
km3.fit(iris_X)
```

```
▾        KMeans
KMeans(n_clusters=3)
```

When we use the fit method, it does not return the predictions, but it does store them in the estimator object.

```
km3.labels_
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2,
       2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2,
       2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1], dtype=int32)
```

```
km2 = KMeans(n_clusters=2)
km2.fit(iris_X)
km4 = KMeans(n_clusters=4)
km4.fit(iris_X)
```

```
▾        KMeans
KMeans(n_clusters=4)
```

How do we tell if this is good?

One way to intuitively think about a good clustering solution is that every point should be close to points in the same cluster and far from points in other clusters. By definition with Kmeans, they will always be clos*er* to points in the same cluster, but we also what that the clusters aren't just touching, but actually spaced apart, if the clustering actually captures meaningfully different groups.

30.1. Silhouette Score

The Silhouette score computes a ratio of how close points are to points in the same cluster vs other clusters.

$$s = \frac{b - a}{max(a, b)}$$

a: The mean distance between a sample and all other points in the same class.

b: The mean distance between a sample and all other points in the next nearest cluster.

We can calculate this score for each point and get the average.

If the cluster is really tight, all of the points are close, then a will be small.
If the clusters are really far apart, b will be large. If both of these are true then b-a will be close to the value of b and the denominator will be b, so the score will be 1.

If the clusters are spread out and close together, then a and be will be close in value, and the s will be close 0. These are overlapping clusters, or possibly too may clusters for this data.

Let's check our clustering solution:

```
metrics.silhouette_score(iris_X, km3.labels_)
```

```
0.5528190123564102
```

This looks pretty good. But we can also compare it for the other two models that we fit.

```
metrics.silhouette_score(iris_X, km2.labels_)
```

```
0.6810461692117465
```

We see in this case this actually fits better.

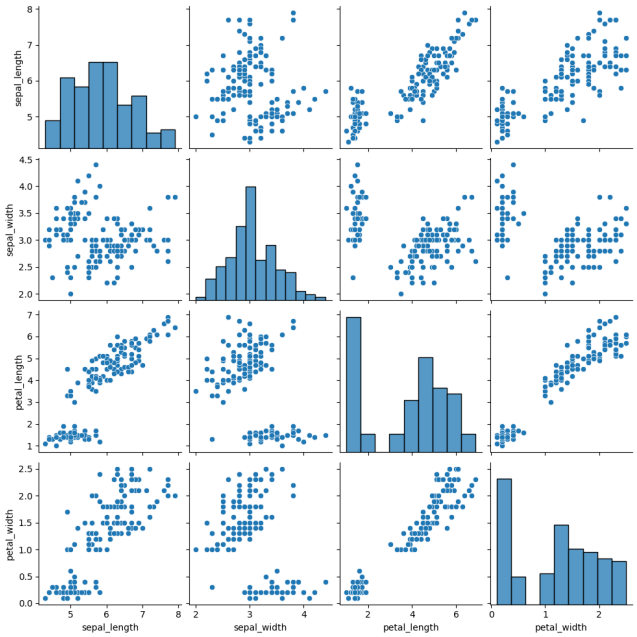
```
metrics.silhouette_score(iris_X, km4.labels_)
```

```
0.49805050499728815
```

and 4 fits worse.

```
sns.pairplot(iris_X)
```

```
<seaborn.axisgrid.PairGrid at 0x7f97994205e0>
```



If we look at the data, we see that it does look somewhat more like two clusters than 4.

30.2. What's a good Silhouette Score?

To think through what a good silhouette score is, we can apply the score to data that represents different scenarios.

First, I'll re-sample some data like we used on Monday, but instead of applying K-means and checking the score of the actual algorithm, we'll add a few different scenarios and add that score.

```
K = 4
N = 200
classes = list(string.ascii_uppercase[:K])
mu = {c: i for c, i in zip(classes,[[2,2], [6,6], [2,6],[6,2]])}


# sample random cluster assignments
target = np.random.choice(classes,N)

# sample points with different means according to those assignments
data = [np.random.multivariate_normal(mu[c],.25*np.eye(2)) for c in target]
feature_names = ['x' + str(i) for i in range(2)]
df = pd.DataFrame(data = data,columns = feature_names)

# save the true assignments
df['true'] = target

# random assignments, right number of clusters
df['random'] = np.random.choice(classes,N)

# random assignments, way too many clusters
charsK4 = list(string.ascii_uppercase[:K*4])
df['random10'] = np.random.choice(charsK4,N)

# Kmeans with 2x number of clusters
kmrK2 = KMeans(K*2)
kmK2.fit(df[['x0','x1']])
df['km' + str(K*2)] = kmK2.labels_

# assign every point to its own cluster
df['id'] = list(range(N))

df.head()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [10], line 3
      1 K = 4
      2 N = 200
----> 3 classes = list(string.ascii_uppercase[:K])
      4 mu = {c: i for c, i in zip(classes,[[2,2], [6,6], [2,6],[6,2]])}
      7 # sample random cluster assignments

NameError: name 'string' is not defined
```

```
assignment_cols = ['true','random', 'random10','km8','id']

s = [metrics.silhouette_score(iris_X, df[col]) for col in assignment_cols]

pd.Series(data = s,index=assignment_cols)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [11], line 3
      1 assignment_cols = ['true','random', 'random10','km8','id']
----> 3 s = [metrics.silhouette_score(iris_X, df[col]) for col in assignment_cols]
      5 pd.Series(data = s,index=assignment_cols)

Cell In [11], line 3, in <listcomp>(.0)
      1 assignment_cols = ['true','random', 'random10','km8','id']
----> 3 s = [metrics.silhouette_score(iris_X, df[col]) for col in assignment_cols]
      5 pd.Series(data = s,index=assignment_cols)

NameError: name 'df' is not defined
```

```
[sns.pairplot(data =df, hue=col, vars= feature_names) for col in assignment_cols]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [12], line 1
----> 1 [sns.pairplot(data =df, hue=col, vars= feature_names) for col in
assignment_cols]

Cell In [12], line 1, in <listcomp>(.0)
----> 1 [sns.pairplot(data =df, hue=col, vars= feature_names) for col in
assignment_cols]

NameError: name 'df' is not defined
```

30.3. Mutual Information

When we know the truth, we can see if the learned clusters are related to the true groups, we can't compare them like accuracy but we can use a metric that is intuitively like a correlation for categorical variables, the mutual information.

Formally mutual information uses the joint distribution between the true labels and the cluster assignments to see how much they co-occur. If they're the exact same, then they have maximal mutual information. If they're completely and independent, then they have 0 mutual information. Mutual information is related to entropy in physics.

The `mutual_info_score` method in the `metrics` module computes mutual information.

```
metrics.mutual_info_score(iris_df['species'],km2.labels_)
```

```
0.5738298777087831
```

```
metrics.mutual_info_score(iris_df['species'],km3.labels_)
```

```
0.8255910976103357
```

```
metrics.mutual_info_score(iris_df['species'],km4.labels_)
```

```
0.8880235203213085
```

When we know the truth, we can see if the learned clusters are related to the true groups, we can't compare them like accuracy but we can use a metric that is intuitively like a correlation for categorical variables, the mutual information.

The `adjusted_mutual_info_score` methos in the `metrics` module computes a version of mutual information that is normalized to have good properties. Apply that to the two different clustering solutions and to a solution for K=4.

```
metrics.adjusted_mutual_info_score(iris_df['species'],km3.labels_)
```

```
0.7551191675800486
```

```
metrics.adjusted_mutual_info_score(iris_df['species'],km2.labels_)
```

```
0.653838071376278
```

```
metrics.adjusted_mutual_info_score(iris_df['species'],km4.labels_)
```

```
0.7172081944051023
```

Further Reading

Sklearn proides many mutual information based scores. See the user guide for definitions of each, pros and cons and examples.

30.4. Questions After Class

30.4.1. If the silhouette score is 0 does it mean that both clusters are the almost the same?

Yes

30.4.2. Are there any other scores to look at to determine best number of clusters?

Any of the scores can be used in that way. And, next week, we will learn how to optimize models.

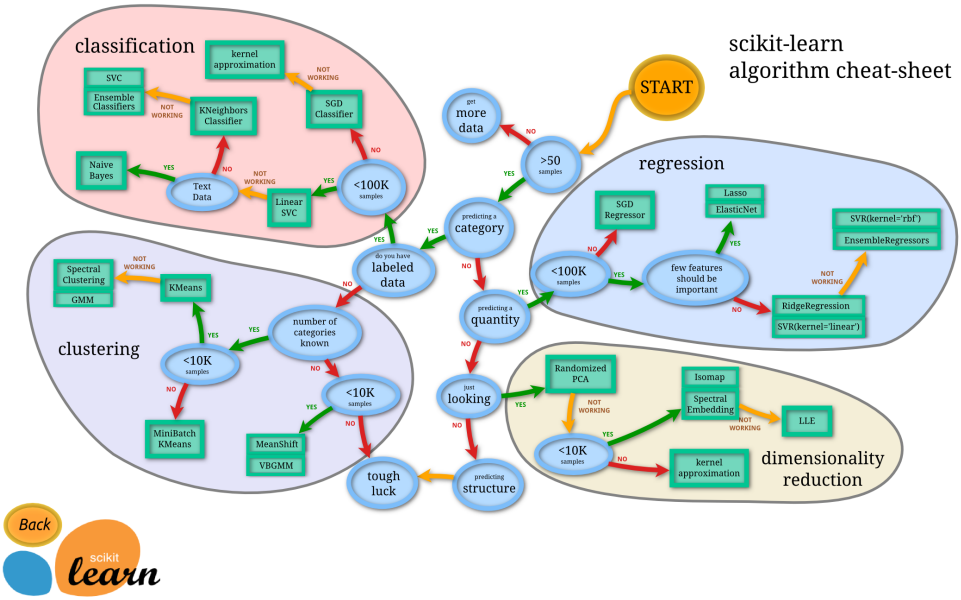
31. ML Task Review Cross Validation

31.1. Relationship between Tasks

We learned classification first, because it shares similarities with each regression and clustering, while regression and clustering have less in common.

Classification is supervised learning for a categorical target.
Regression is supervised learning for a continuous target. Clustering is unsupervised learning for a categorical target.

Sklearn provides a nice flow chart for thinking through this.



Predicting a category is another way of saying categorical target. Predicting a quantitiy is another way of saying continuous target. Having lables or not is the difference between

The flowchart assumes you know what you want to do with data and that is the ideal scenario. You have a dataset and you have a goal. For the purpose of getting to practice with a variety of things, in this course we ask you to start with a task and then find a dataset. Assignment 9 is the last time that's true however. Starting with Assignment 10 and the last portflios, you can choose and focus on a specific application domain and then choose the right task from there.

Thinking about this, however, you use this information to move between the tasks within a given type of data. For example, you can use the same data for clustering as you did for classification. Switching the task changes the questions though: classification evaluation tells us how separable the classes are given that classifiers decision rule. Clustering can find other subgroups or the same ones, so the evaluation we choose allows us to explore this in more ways.

Regression requires a continuous target, so we need a dataset to be suitable for that, we can't transform from the classification dataset to a regression one. However, we can go the other way and that's how some classification datasets are created.

The UCI adult Dataset is a popular ML dataset that was dervied from census data. The goal is to use a variety of features to predict if a person makes more than $50k per year or not. While income is a continuous value, they applied a threshold$(50k) to it to make a binary variable. The dataset does not include income in dollars, only the binary indicator.

> **Further Reading**
>
> Recent work reconsturcted the dataset with the continuous valued income. Their repository contains the data as well as links to their paper and a video of their talk on it.

31.2. Cross Validation

This week our goal is to learn how to optmize models. The first step in that is to get a good estimate of its performance.

We have seen that the test train splits, which are random, influence the performance.

```
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn import metrics
```

We'll use the Iris data with a decision tree.

```
iris_df = sns.load_dataset('iris')

iris_X = iris_df.drop(columns=['species'])
iris_y = iris_df['species']
```

```
dt =tree.DecisionTreeClassifier()
```

We can split the data, fit the model, then compute a score, but since the splitting is a randomized step, the score is a random variable.

For example, if we have a coin that we want to see if it's fair or not. We would flip it to test. One flip doesn't tell us, but if we flip it a few times, we can estimate the probability it is heads by counting how many of the flips are heads and dividing by how many flips.

We can do something similar with our model performance. We can split the data a bunch of times and compute the score each time.

`cross_val_score` does this all for us.

It takes an estimator object and the data.

By default it uses 5-fold cross validation. It splits the data into 5 sections, then uses 4 of them to train and one to test. It then iterates through so that each section gets used for testing.

```
cross_val_score(dt,iris_X,iris_y)
```

```
array([0.96666667, 0.96666667, 0.9       , 0.96666667, 1.        ])
```

We get back a score for each section or “fold” of the data. We can average those to get a single estimate.

```
np.mean(cross_val_score(dt,iris_X,iris_y))
```

```
0.9600000000000002
```

We can use more folds.

```
np.mean(cross_val_score(dt,iris_X,iris_y,cv=10))
```

```
0.96
```

We can peak inside what this actually does to see it more clearly.

31.3. What Does Cross validation really do?

It uses StratifiedKfold for classification, but since we’re using regression it will use `KFold`. `test_train_split` uses `ShuffleSplit` by default, let’s load that too to see what it does.

> **Warning**
>
> The key in the following is to get the *concepts* not all of the details in how I evaluate and visualize. I could have made figures separately to explain the concept, but I like to show that Python is self contained.

```
from sklearn.model_selection import KFold, ShuffleSplit
```

```
kf = KFold(n_splits = 10)
```

When we use the `split` method it gives us a generator.

```
kf.split(diabetes_X, diabetes_y)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [9], line 1
----> 1 kf.split(diabetes_X, diabetes_y)

NameError: name 'diabetes_X' is not defined
```

We can use this in a loop to get the list of indices that will be used to get the test and train data for each fold. To visualize what this is doing, see below.

```
N_samples = len(diabetes_y)
kf_tt_df = pd.DataFrame(index=list(range(N_samples)))
i = 1
for train_idx, test_idx in kf.split(diabetes_X, diabetes_y):
    kf_tt_df['split ' + str(i)] = ['unused']*N_samples
    kf_tt_df['split ' + str(i)][train_idx] = 'Train'
    kf_tt_df['split ' + str(i)][test_idx] = 'Test'
    i +=1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [10], line 1
----> 1 N_samples = len(diabetes_y)
      2 kf_tt_df = pd.DataFrame(index=list(range(N_samples)))
      3 i = 1

NameError: name 'diabetes_y' is not defined
```

We can count how many times ‘Test’ and ‘Train’ appear

How would you use those indices to get a out actual test and train data?

```
count_test = lambda part: len([v for v in part if v=='Test'])
count_train = lambda part: len([v for v in part if v=='Train'])
```

When we apply this along `axis=1` we to check that each sample is used in exactly 1 test set how may times each sample is used

```
sum(kf_tt_df.apply(count_test,axis = 1) ==1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [12], line 1
----> 1 sum(kf_tt_df.apply(count_test,axis = 1) ==1)

NameError: name 'kf_tt_df' is not defined
```

and exactly 9 training sets

```
sum(kf_tt_df.apply(count_test,axis = 1) ==9)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [13], line 1
----> 1 sum(kf_tt_df.apply(count_test,axis = 1) ==9)

NameError: name 'kf_tt_df' is not defined
```

the describe helps ensure that all fo the values are exa

We can also visualize:

```
cmap = sns.color_palette("tab10",10)
g = sns.heatmap(kf_tt_df.replace({'Test':1,'Train':0}),cmap=cmap[7:9],cbar_kws=
{'ticks':[.25,.75]},linewidths=0,
    linecolor='gray')
colorbar = g.collections[0].colorbar
colorbar.set_ticklabels(['Train','Test'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [14], line 2
      1 cmap = sns.color_palette("tab10",10)
----> 2 g =
sns.heatmap(kf_tt_df.replace({'Test':1,'Train':0}),cmap=cmap[7:9],cbar_kws=
{'ticks':[.25,.75]},linewidths=0,
      3     linecolor='gray')
      4 colorbar = g.collections[0].colorbar
      5 colorbar.set_ticklabels(['Train','Test'])

NameError: name 'kf_tt_df' is not defined
```

Note that unlike `test_train_split` this does not always randomize and shuffle the data before splitting.

If we apply those `lambda` functions along `axis=0`, we can see the size of each test set

```
kf_tt_df.apply(count_test,axis = 0)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [15], line 1
----> 1 kf_tt_df.apply(count_test,axis = 0)

NameError: name 'kf_tt_df' is not defined
```

and training set:

```
kf_tt_df.apply(count_train,axis = 0)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [16], line 1
----> 1 kf_tt_df.apply(count_train,axis = 0)

NameError: name 'kf_tt_df' is not defined
```

We can verify that these splits are the same size as what `test_train_split` does using the right settings. 10-fold splits the data into 10 parts and tests on 1, so that makes a test size of 1/10=.1, so we can use the `train_test_split` and check the length.

```
X_train2,X_test2, y_train2,y_test2 = train_test_split(diabetes_X, diabetes_y ,
                                     test_size=.1,random_state=0)

[len(split) for split in [X_train2,X_test2,]]
```

Under the hood `train_test_split` uses `ShuffleSplit` We can do a similar experiment as above to see what `ShuffleSplit` does.

```
skf = ShuffleSplit(10)
N_samples = len(diabetes_y)
ss_tt_df = pd.DataFrame(index=list(range(N_samples)))
i = 1
for train_idx, test_idx in skf.split(diabetes_X, diabetes_y):
    ss_tt_df['split ' + str(i)] = ['unused']*N_samples
    ss_tt_df['split ' + str(i)][train_idx] = 'Train'
    ss_tt_df['split ' + str(i)][test_idx] = 'Test'
    i +=1

ss_tt_df
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [17], line 2
      1 skf = ShuffleSplit(10)
----> 2 N_samples = len(diabetes_y)
      3 ss_tt_df = pd.DataFrame(index=list(range(N_samples)))
      4 i = 1

NameError: name 'diabetes_y' is not defined
```

And plot

```
cmap = sns.color_palette("tab10",10)
g = sns.heatmap(ss_tt_df.replace({'Test':1,'Train':0}),cmap=cmap[7:9],cbar_kws=
{'ticks':[.25,.75]},linewidths=0,
    linecolor='gray')
colorbar = g.collections[0].colorbar
colorbar.set_ticklabels(['Train','Test'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [18], line 2
      1 cmap = sns.color_palette("tab10",10)
----> 2 g =
sns.heatmap(ss_tt_df.replace({'Test':1,'Train':0}),cmap=cmap[7:9],cbar_kws=
{'ticks':[.25,.75]},linewidths=0,
      3     linecolor='gray')
      4 colorbar = g.collections[0].colorbar
      5 colorbar.set_ticklabels(['Train','Test'])

NameError: name 'ss_tt_df' is not defined
```

31.4. Cross validation with clustering

We can use *any* estimator object here.

```
km = KMeans(n_clusters=3)
```

```
cross_val_score(km,iris_X)
```

Tip

`sns.heatmap` doesn't work on strings, so we can replace them for the plotting

```
array([ -9.062     , -14.93195873, -18.93234207, -23.70894258,
       -19.55457726])
```

```
km.score()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [21], line 1
----> 1 km.score()

TypeError: score() missing 1 required positional argument: 'X'
```

31.5. Grid Search Optimization

We can optimize, however to determing the different parameter settings.

A simple way to do this is to fit the model for different parameters and score for each and compare.

The

```
from sklearn.model_selection import GridSearchCV
```

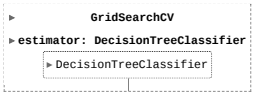
The `GridSearchCV` object is constructed first and requires an estimator object and a dictionary that describes the parameter grid to search over. The dictionary has the parameter names as the keys and the values are the values for that parameter to test.

The `fit` method on the Grid Search object fits all of the separate models.

In this case, we will optimize the depth of this Decision Tree.

```
param_grid = {'max_depth':[2,3,4,5]}
dt_opt = GridSearchCV(dt,param_grid)
```

```
dt_opt.fit(iris_X,iris_y)
```



Then we can look at the output.

```
dt_opt.cv_results_
```

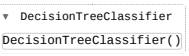
```
{'mean_fit_time': array([0.00220914, 0.00174689, 0.00178728, 0.0017498 ]),
 'std_fit_time': array([6.82693563e-04, 3.28284289e-05, 3.63944536e-05,
6.06964069e-05]),
 'mean_score_time': array([0.00139256, 0.0012569 , 0.00127978, 0.00126529]),
 'std_score_time': array([1.78804804e-04, 3.43492916e-05, 7.84166499e-05,
2.41615792e-05]),
 'param_max_depth': masked_array(data=[2, 3, 4, 5],
              mask=[False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'max_depth': 2},
  {'max_depth': 3},
  {'max_depth': 4},
  {'max_depth': 5}],
 'split0_test_score': array([0.93333333, 0.96666667, 0.96666667, 0.96666667]),
 'split1_test_score': array([0.96666667, 0.96666667, 0.96666667, 0.96666667]),
 'split2_test_score': array([0.9       , 0.93333333, 0.9       , 0.9       ]),
 'split3_test_score': array([0.86666667, 1.        , 1.        , 0.96666667]),
 'split4_test_score': array([1., 1., 1., 1.]),
 'mean_test_score': array([0.93333333, 0.97333333, 0.96666667, 0.96      ]),
 'std_test_score': array([0.04714045, 0.02494438, 0.03651484, 0.03265986]),
 'rank_test_score': array([4, 1, 2, 3], dtype=int32)}
```

We note that this is a dictionary, so to make it more readable, we can make it a DataFrame.

```
pd.DataFrame(dt_opt.cv_results_)
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score
0	0.002209	0.000683	0.001393	0.000179	2	{'max_depth': 2}	0.933333	0.966667	0.900000	0.866667
1	0.001747	0.000033	0.001257	0.000034	3	{'max_depth': 3}	0.966667	0.966667	0.933333	1.000000
2	0.001787	0.000036	0.001280	0.000078	4	{'max_depth': 4}	0.966667	0.966667	0.900000	1.000000
3	0.001750	0.000061	0.001265	0.000024	5	{'max_depth': 5}	0.966667	0.966667	0.900000	0.966667

```
dt
```



31.6. Questions After Class

31.6.1. Do we have to do anything to pick the highest ranking model from the GridSearchCV function?

No, we can use it directly. For example:

```
plt.figure(figsize=(15,20))
tree.plot_tree(dt_opt.best_estimator_, rounded =True, class_names = ['A','B'],
      proportion=True, filled =True, impurity=False,fontsize=10);
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [28], line 1
----> 1 plt.figure(figsize=(15,20))
      2 tree.plot_tree(dt_opt.best_estimator_, rounded =True, class_names =
['A','B'],
      3       proportion=True, filled =True, impurity=False,fontsize=10);

NameError: name 'plt' is not defined
```

31.6.2. Is there anything similar to a gridsearch object for different estimators, where it can try different methods of estimation and rank them?

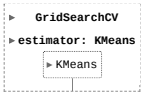
No, you would use multiple gridsearch (or similar model optimizer with a different search strategy) one for each model. Each model class/ estimator object

31.6.3. I would like to learn how to apply cross validation and especially program optimization to unsupervised clustering models.

It would look a lot like what we did with the decision tree, but we use the right parameter name, for example:

```
km = KMeans()
param_grid = {'n_clusters': list(range(2,8))}
km_opt = GridSearchCV(km,param_grid)
```

```
km_opt.fit(iris_X)
```




```
pd.DataFrame(km_opt.cv_results_)
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_clusters	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score
0	0.008184	0.001285	0.001270	0.000066	2	{'n_clusters': 2}	-13.417292	-19.334754	-58.409800	-56.347656
1	0.010378	0.000420	0.001224	0.000046	3	{'n_clusters': 3}	-9.062000	-14.931959	-18.932342	-23.708943
2	0.012476	0.001296	0.001224	0.000032	4	{'n_clusters': 4}	-9.062000	-11.395150	-13.714817	-17.596427
3	0.013572	0.000262	0.001239	0.000035	5	{'n_clusters': 5}	-9.062000	-10.723786	-10.840587	-17.596427
4	0.015833	0.000262	0.001341	0.000068	6	{'n_clusters': 6}	-9.062000	-8.144084	-10.996277	-12.698270
5	0.017909	0.000186	0.001330	0.000046	7	{'n_clusters': 7}	-9.062000	-6.682665	-9.066973	-12.183027

31.6.4. Is it better to split the data in more folds when using the cross-validation?

this is a tricky question, we'll revisit it in class on Wednesday.

31.6.5. "What is this "model" we are training? What are the scores, scoring?

In this example, the model was a decision tree at the beginning and later K-means. The score describes how well the fit model works on the held out data; accuracy or a general fit statistic.

The model vs algorithm section in the introduction of the Model Based ML book (free) is a good thing to read to clarify these relationships.

sklearn provides a flowchart for choosing their different estimator objects. In sklearn, they implement each model as an estimator object; more specifically, they have a Base Estimator class that the other estimators inherit. For example the decision tree source shows that it inherits the `ClassifierMixin` and **`BaseDecisionTree`** which inherits `BaseEstimator`

Term	Definition	Example
task	the type of algorithm that we will use machine learning to write	classification, regression, clustering
model	the specific form and set of assumptions that will be used in the algorithm	decision tree (classification) Gaussian Naive Bayes (classification), linear regression, sparse regression/LASSO, K-means, spectral clustering, etc.
score	a measure of how well the model completes the task	accuracy (for classification), mean squared error (for regression), silhouette score (for clustering)

also review the intro to models in machine learning class notes

32. Model Optimization

> **Important**
>
> Remember that best is context dependent and relative. The best accuracy might not be the best overall. Automatic optimization can only find the best thing in terms of a single score.

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn import datasets
from sklearn import cluster
from sklearn import svm, datasets
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn import tree
```

32.1. Model validation

We will work with the iris data again.

```
iris_df = sns.load_dataset('iris')


iris_X = iris_df.drop(columns=['species'])

iris_y = iris_df['species']
```

We will still use the test train split to keep our test data separate from the data that we use to find our preferred parameters.

```
iris_X_train, iris_X_test, iris_y_train, iris_y_test =
train_test_split(iris_X,iris_y,

random_state=0)
```

Today we will optimize a decision tree over three parameters. One is the criterion, which is how it decides where to create thresholds in parameters. Gini is the default and it computes how concentrated each class is at that node, another is entropy, entropy is, generally how random something is. Intuitively these do similar things, which makes sense because they are two ways to make the same choice, but they have slightly different calculations.

The other two parameters we have seen some before. Max depth is the height of the tree and min smaples per leaf makes it keeps the leaf sizes small.

Further Reading

the criteria are discussed in the mathematical formulation of the sklearn documentation

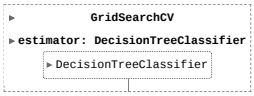
```
dt = tree.DecisionTreeClassifier()
params_dt = {'criterion':['gini','entropy'],'max_depth':[2,3,4],
       'min_samples_leaf':list(range(2,20,2))}
```

We will fit it with default CV settings.

```
dt_opt = GridSearchCV(dt,params_dt)
```

then we can fit

```
dt_opt.fit(iris_X_train,iris_y_train)
```



we can use ti to get predictions

```
y_pred = dt_opt.predict(iris_X_test)
```

we can also score it as regular.

```
dt_opt.score(iris_X_test,iris_y_test)
```

```
0.9473684210526315
```

we can also see the best parameters.

```
dt_opt.best_params_
```

```
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2}
```

we can look at the overall results this way:

```
dt_5cv_df = pd.DataFrame(dt_opt.cv_results_)
dt_5cv_df.head()
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_criterion	param_max_depth	param_min_samples_leaf	params	split0_test_score	split1_test
0	0.001898	0.000284	0.001269	0.000060	gini	2	2	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.956522	0.9
1	0.001652	0.000072	0.001205	0.000045	gini	2	4	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.956522	0.9
2	0.001712	0.000026	0.001191	0.000089	gini	2	6	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.956522	0.9
3	0.001699	0.000026	0.001199	0.000068	gini	2	8	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.956522	0.9
4	0.001672	0.000050	0.001220	0.000033	gini	2	10	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.956522	0.9

To see more carefully what it does, we can look at its shape.

```
dt_5cv_df.shape
```

```
(54, 16)
```

we can see that the total number of rows matched the product of the length of each list of parameters to try.

```
np.product([len(param_list) for param_list in params_dt.values()])
```

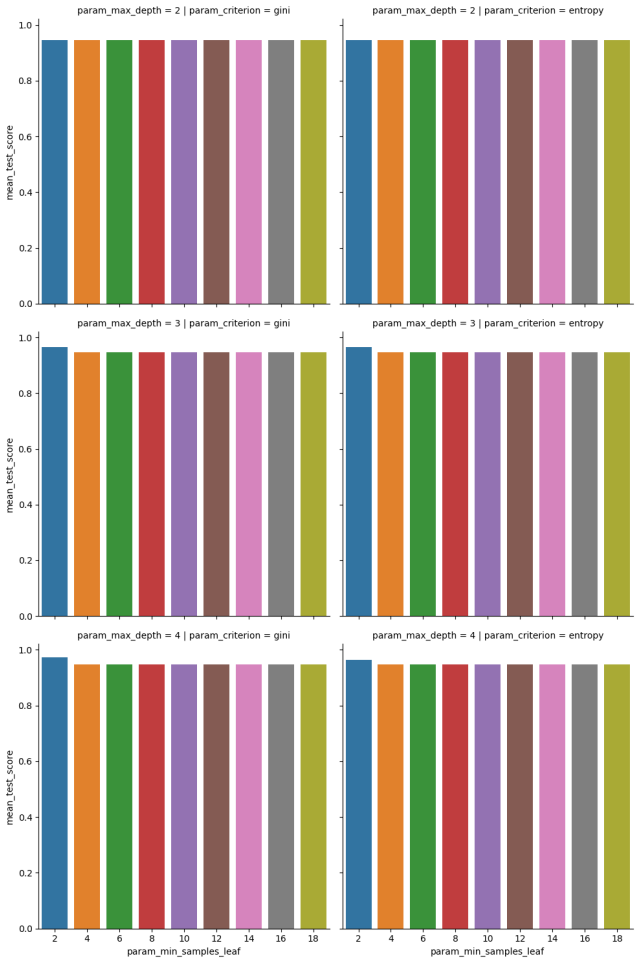
```
54
```

This means that it tests every combination of features– without us writing a bunch of nested loops.

We can also plot the dta and look at the performance.

```
sns.catplot(data=dt_5cv_df,x='param_min_samples_leaf',y='mean_test_score',
            col='param_criterion', row= 'param_max_depth', kind='bar',)
```

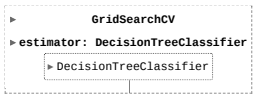
```
<seaborn.axisgrid.FacetGrid at 0x7fad349a0370>
```



this makes it clear that none of these stick out much in terms of performance.

32.2. Impact of CV parameters

```
dt_opt10 = GridSearchCV(dt,params_dt,cv=10)
dt_opt10.fit(iris_X_train,iris_y_train)
```




```
dt_10cv_df = pd.DataFrame(dt_opt10.cv_results_)
```

We can stack the columns we want from the two results together with a new indicator column `cv`

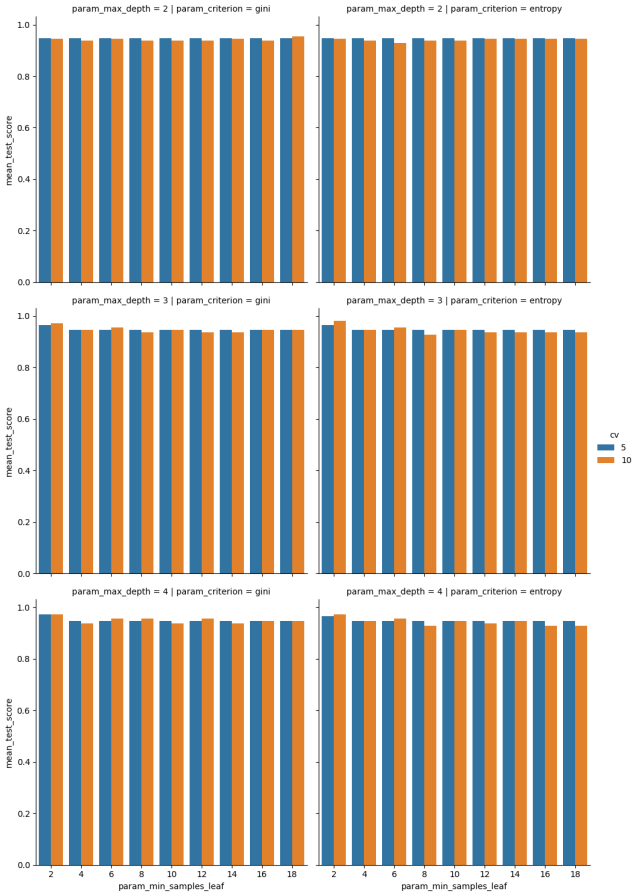
```
plot_cols = ['param_min_samples_leaf','std_test_score','mean_test_score',
             'param_criterion','param_max_depth','cv']
dt_10cv_df['cv'] = 10
dt_5cv_df['cv'] = 5

dt_cv_df = pd.concat([dt_5cv_df[plot_cols],dt_10cv_df[plot_cols]])
dt_cv_df.head()
```

	param_min_samples_leaf	std_test_score	mean_test_score	param_criterion	param_max_depth	cv
0	2	0.033305	0.94664	gini	2	5
1	4	0.033305	0.94664	gini	2	5
2	6	0.033305	0.94664	gini	2	5
3	8	0.033305	0.94664	gini	2	5
4	10	0.033305	0.94664	gini	2	5

this can be used to plot.

```
sns.catplot(data=dt_cv_df,x='param_min_samples_leaf',y='mean_test_score',
            col='param_criterion', row= 'param_max_depth', kind='bar',
            hue = 'cv')
```

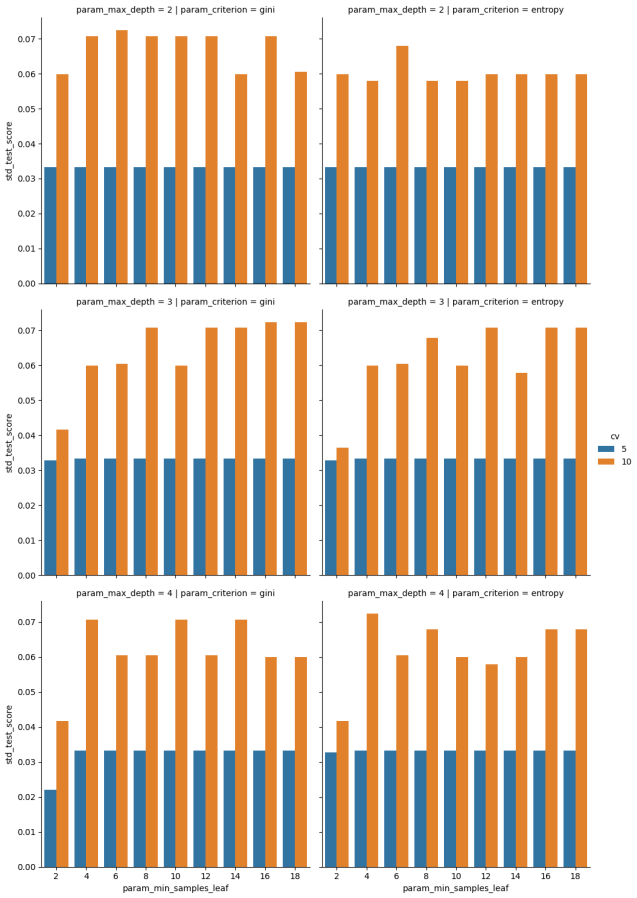
```
<seaborn.axisgrid.FacetGrid at 0x7fad313c9c70>
```



we see that the mean scores are not very different, but that 10 is a little higher in some cases. This makes sense, it has more data to learn from, so it found something that applied better, on average, to the test set.

```
sns.catplot(data=dt_cv_df,x='param_min_samples_leaf',y='std_test_score',
            col='param_criterion', row= 'param_max_depth', kind='bar',
            hue = 'cv')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fad320901c0>
```



However here we see that the variabilty in those scores is much higher, so maybe the 5 is better.

We can compare to see if it finds the same model as best:

```
dt_opt.best_params_
```

```
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2}
```

```
dt_opt10.best_params_
```

```
{'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 2}
```

In some cases they will and others they will not.

```
dt_opt.score(iris_X_test,iris_y_test)
```

```
0.9473684210526315
```

```
dt_opt10.score(iris_X_test,iris_y_test)
```

```
0.9736842105263158
```

In some cases they will find the same model and score the same, but it other time they will not.

The takeaway is that the cross validation parameters impact our ability to measure the score and possibly how close that cross validation mean score will match the true test score. Mostly it will change the variability in the estimate of the score. It does not change necessarily which model is best, that is up to the data iteself (the original test/train split would impact this).

> **Try it yourself**
>
> Does this vary if you repeat this with different test sets? How much does it depend on that? Does repeating it produce the same scores? Does one take longer to fit or score?

32.3. Other searches

```
from sklearn import model_selection
from sklearn.model_selection import LeaveOneOut
```

```
rand_opt = model_selection.RandomizedSearchCV(dt,params_dt).fit(iris_X_train,
                                                iris_y_train)
```

```
rand_opt.score(iris_X_test,iris_y_test)
```

```
0.9736842105263158
```

```
rand_opt.best_params_
```

```
{'min_samples_leaf': 2, 'max_depth': 3, 'criterion': 'gini'}
```

It might find the same solution, but it also might not. If you do some and see that the parameters overall do not impact the scores much, then you can trust whichever one, or consider other criteria to choose the best model to use.

32.4. Questions after class

32.4.1. What does cross-validation do that clustering doesn't?

Cross validation can be used *with* clustering if you want. Cross validation is a technique for evaluating how well a model performs; in this case we are combining with a search to find the best parameters.

32.4.2. Is grid search commonly used or is it too expensive for more realistically sized datasets?

Grid search is realistic for simple models. It becomes more expensive with models that have more options for parameters. For example a parameter that can be continuously valued or take a large range, there become too many to test, so we use the random search or something else. The cost does go up some with larger datasets, but this scales primarily with the nubmer of parameters and values for each.

For example, bayesian optimization is a more complex type of random search.

32.4.3. What is the difference between gini and entropy?

the criteria are discussed in the mathematical formulation of the sklearn documentation

32.4.4. Is it more accurate to do cross vaididation or test_train_split?

Ideally both: use train test split to set aside a test set for the value that you will report when you say this is how good my final model is and cross validation to choose which model you call the final model.

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn import datasets
from sklearn import cluster
from sklearn import svm
from sklearn import tree
from sklearn import model_selection
```

```
from sklearn import model_selection as ms
```

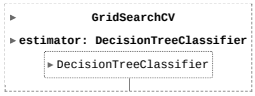
```
200*.8/5
```

```
32.0
```

```
iris_df = sns.load_dataset('iris')
iris_X = iris_df.drop(columns='species')
iris_y = iris_df['species']

iris_X_train, iris_X_test, iris_y_train, iris_y_test =
model_selection.train_test_split(iris_X,iris_y, test_size =.2)
```

```
dt = tree.DecisionTreeClassifier()
params_dt = {'criterion':['gini','entropy'],'max_depth':[2,3,4],
       'min_samples_leaf':list(range(2,20,2))}
```

```
dt_opt = model_selection.GridSearchCV(dt, params_dt)
dt_opt.fit(iris_X_train,iris_y_train)
```




```
dt_cv_df = pd.DataFrame(dt_opt.cv_results_)
```

```
dt_cv_df
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_criterion	param_max_depth	param_min_samples_leaf	params	split0_test_score	split1_tes
0	0.002108	0.000336	0.001432	0.000217	gini	2	2	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.958333	
1	0.001797	0.000048	0.001366	0.000165	gini	2	4	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.958333	
2	0.001886	0.000230	0.001293	0.000045	gini	2	6	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.958333	
3	0.001988	0.000262	0.001409	0.000179	gini	2	8	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.958333	
4	0.001736	0.000017	0.001285	0.000019	gini	2	10	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.958333	
5	0.001736	0.000017	0.001268	0.000026	gini	2	12	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.958333	
6	0.001746	0.000028	0.001276	0.000024	gini	2	14	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.958333	
7	0.001731	0.000012	0.001273	0.000016	gini	2	16	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.958333	
8	0.001742	0.000021	0.001284	0.000032	gini	2	18	{'criterion': 'gini', 'max_depth': 2, 'min_sam...	0.958333	
9	0.001776	0.000053	0.001257	0.000014	gini	3	2	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.958333	
10	0.001718	0.000015	0.001266	0.000023	gini	3	4	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.958333	
11	0.001742	0.000030	0.001258	0.000011	gini	3	6	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.958333	
12	0.001722	0.000019	0.001243	0.000015	gini	3	8	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.958333	
13	0.001725	0.000017	0.001231	0.000008	gini	3	10	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.958333	
14	0.001709	0.000017	0.001253	0.000044	gini	3	12	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.958333	
15	0.002065	0.000710	0.001308	0.000119	gini	3	14	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.958333	
16	0.001963	0.000165	0.001441	0.000213	gini	3	16	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.958333	
17	0.002113	0.000511	0.001459	0.000314	gini	3	18	{'criterion': 'gini', 'max_depth': 3, 'min_sam...	0.958333	
18	0.001787	0.000131	0.001287	0.000056	gini	4	2	{'criterion': 'gini', 'max_depth': 4, 'min_sam...	0.958333	
19	0.001791	0.000107	0.001321	0.000058	gini	4	4	{'criterion': 'gini', 'max_depth': 4, 'min_sam...	0.958333	

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_criterion	param_max_depth	param_min_samples_leaf	params	split0_test_score	split1_tes
20	0.001693	0.000139	0.001232	0.000036	gini	4	6	{'criterion': 'gini', 'max_depth': 4, 'min_sam...	0.958333	
21	0.001855	0.000214	0.001284	0.000052	gini	4	8	{'criterion': 'gini', 'max_depth': 4, 'min_sam...	0.958333	
22	0.001814	0.000157	0.001246	0.000007	gini	4	10	{'criterion': 'gini', 'max_depth': 4, 'min_sam...	0.958333	
23	0.001757	0.000061	0.001261	0.000033	gini	4	12	{'criterion': 'gini', 'max_depth': 4, 'min_sam...	0.958333	
24	0.001779	0.000053	0.001240	0.000017	gini	4	14	{'criterion': 'gini', 'max_depth': 4, 'min_sam...	0.958333	
25	0.001791	0.000080	0.001359	0.000183	gini	4	16	{'criterion': 'gini', 'max_depth': 4, 'min_sam...	0.958333	
26	0.001693	0.000042	0.001286	0.000058	gini	4	18	{'criterion': 'gini', 'max_depth': 4, 'min_sam...	0.958333	
27	0.001753	0.000125	0.001248	0.000049	entropy	2	2	{'criterion': 'entropy', 'max_depth': 2, 'min_...	0.958333	
28	0.001765	0.000069	0.001289	0.000089	entropy	2	4	{'criterion': 'entropy', 'max_depth': 2, 'min_...	0.958333	
29	0.002215	0.000601	0.001298	0.000049	entropy	2	6	{'criterion': 'entropy', 'max_depth': 2, 'min_...	0.958333	
30	0.002169	0.000080	0.001374	0.000237	entropy	2	8	{'criterion': 'entropy', 'max_depth': 2, 'min_...	0.958333	
31	0.001758	0.000115	0.001355	0.000090	entropy	2	10	{'criterion': 'entropy', 'max_depth': 2, 'min_...	0.958333	
32	0.002117	0.000257	0.001553	0.000274	entropy	2	12	{'criterion': 'entropy', 'max_depth': 2, 'min_...	0.958333	
33	0.002077	0.000152	0.001349	0.000141	entropy	2	14	{'criterion': 'entropy', 'max_depth': 2, 'min_...	0.958333	
34	0.001811	0.000138	0.001499	0.000207	entropy	2	16	{'criterion': 'entropy', 'max_depth': 2, 'min_...	0.958333	
35	0.001806	0.000198	0.001527	0.000153	entropy	2	18	{'criterion': 'entropy', 'max_depth': 2, 'min_...	0.958333	
36	0.002052	0.000165	0.001287	0.000091	entropy	3	2	{'criterion': 'entropy', 'max_depth': 3, 'min_...	0.958333	
37	0.001945	0.000147	0.001427	0.000187	entropy	3	4	{'criterion': 'entropy', 'max_depth': 3, 'min_...	0.958333	
38	0.001987	0.000199	0.001359	0.000151	entropy	3	6	{'criterion': 'entropy', 'max_depth': 3, 'min_...	0.958333	
39	0.002044	0.000096	0.001322	0.000121	entropy	3	8	{'criterion': 'entropy', 'max_depth': 3, 'min_...	0.958333	
40	0.002099	0.000086	0.001368	0.000156	entropy	3	10	{'criterion': 'entropy', 'max_depth': 3, 'min_...	0.958333	
41	0.001875	0.000149	0.001394	0.000181	entropy	3	12	{'criterion': 'entropy', 'max_depth': 3, 'min_...	0.958333	
42	0.001862	0.000163	0.001489	0.000178	entropy	3	14	{'criterion': 'entropy', 'max_depth': 3, 'min_...	0.958333	

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_criterion	param_max_depth	param_min_samples_leaf	params	split0_test_score	split1_tes
43	0.001766	0.000113	0.001334	0.000144	entropy	3	16	{'criterion': 'entropy', 'max_depth': 3, 'min_...	0.958333	
44	0.001898	0.000181	0.001243	0.000027	entropy	3	18	{'criterion': 'entropy', 'max_depth': 3, 'min_...	0.958333	
45	0.001866	0.000110	0.001262	0.000143	entropy	4	2	{'criterion': 'entropy', 'max_depth': 4, 'min_...	0.958333	
46	0.001911	0.000125	0.001275	0.000034	entropy	4	4	{'criterion': 'entropy', 'max_depth': 4, 'min_...	0.958333	
47	0.001851	0.000113	0.001312	0.000139	entropy	4	6	{'criterion': 'entropy', 'max_depth': 4, 'min_...	0.958333	
48	0.001847	0.000114	0.001324	0.000117	entropy	4	8	{'criterion': 'entropy', 'max_depth': 4, 'min_...	0.958333	
49	0.001819	0.000082	0.001310	0.000134	entropy	4	10	{'criterion': 'entropy', 'max_depth': 4, 'min_...	0.958333	
50	0.001898	0.000184	0.001199	0.000111	entropy	4	12	{'criterion': 'entropy', 'max_depth': 4, 'min_...	0.958333	
51	0.001990	0.000189	0.001183	0.000041	entropy	4	14	{'criterion': 'entropy', 'max_depth': 4, 'min_...	0.958333	
52	0.001760	0.000152	0.001284	0.000075	entropy	4	16	{'criterion': 'entropy', 'max_depth': 4, 'min_...	0.958333	
53	0.001675	0.000047	0.001245	0.000077	entropy	4	18	{'criterion': 'entropy', 'max_depth': 4, 'min_...	0.958333	

```
type(dt_opt.best_estimator_)
```

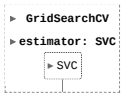
```
sklearn.tree._classes.DecisionTreeClassifier
```

```
dt_opt.best_params_
```

```
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 2}
```

```
svmclf = svm.SVC()
param_grid_svm = {'kernel':['linear','rbf'], 'C':[.5, 1, 10]}
svm_opt = model_selection.GridSearchCV(svmclf,param_grid_svm)
```

```
svm_opt.fit(iris_X_train,iris_y_train)
```




```
svm_opt.best_params_
```

```
{'C': 1, 'kernel': 'linear'}
```

```
svm_opt.score(iris_X_test,iris_y_test), dt_opt.score(iris_X_test,iris_y_test)
```

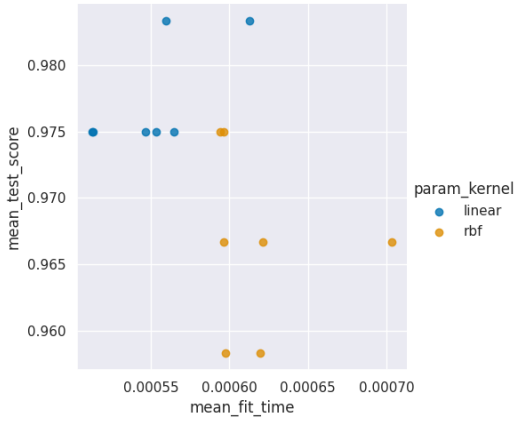
```
(0.9666666666666667, 0.9333333333333333)
```

33. Model Comparison: when do differences matter?

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn import datasets
from sklearn import cluster
from sklearn import svm
from sklearn import tree
# import the whole model selection module
from sklearn import model_selection
sns.set_theme(palette='colorblind')

# load data, 20% test
iris_X, iris_y = datasets.load_iris(return_X_y=True)
iris_X_train, iris_X_test, iris_y_train, iris_y_test =
model_selection.train_test_split(
  iris_X,iris_y, test_size =.2)


# setup DT & params
dt = tree.DecisionTreeClassifier()

params_dt = {'criterion':['gini','entropy'],
       'max_depth':[2,3,4,5,6],
    'min_samples_leaf':list(range(2,20,2))}

dt_opt = model_selection.GridSearchCV(dt,params_dt)


# optmize DT
dt_opt.fit(iris_X_train,iris_y_train)

# store DT results in dataframe
dt_df = pd.DataFrame(dt_opt.cv_results_)


# setup svm and params
svm_clf = svm.SVC()
param_grid = {'kernel':['linear','rbf'], 'C':[.5, .75,1,2,5,7, 10]}
svm_opt = model_selection.GridSearchCV(svm_clf,param_grid)

# optmize and save svm results
svm_opt.fit(iris_X_train,iris_y_train)
sv_df = pd.DataFrame(svm_opt.cv_results_)
```

```
description_vars = ['param_C', 'param_kernel', 'params',]
vars_to_plot = ['mean_fit_time', 'std_fit_time', 'mean_score_time',
'std_score_time']
svm_time = sv_df.melt(id_vars= description_vars,
          value_vars=vars_to_plot)
sns.lmplot(data=sv_df, x='mean_fit_time',y='mean_test_score',
     hue='param_kernel',fit_reg=False)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f88b0646340>
```




```
len(iris_X_train)/5
```

```
24.0
```

```
23/24
```

```
0.9583333333333334
```

```
svm_opt.predict_proba
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In [5], line 1
----> 1 svm_opt.predict_proba

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/sklearn/utils/metaestimators.py:127, in
_AvailableIfDescriptor.__get__(self, obj, owner)
    121 attr_err = AttributeError(
    122     f"This {repr(owner.__name__)} has no attribute
{repr(self.attribute_name)}"
    123 )
    124 if obj is not None:
    125     # delegate only on instances, not the classes.
    126     # this is to allow access to the docstrings.
--> 127     if not self.check(obj):
    128         raise attr_err
    129     out = MethodType(self.fn, obj)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/sklearn/model_selection/_search.py:363, in _estimator_has.
<locals>.check(self)
    360 _check_refit(self, attr)
    361 if hasattr(self, "best_estimator_"):
    362     # raise an AttributeError if `attr` does not exist
--> 363     getattr(self.best_estimator_, attr)
    364     return True
    365 # raise an AttributeError if `attr` does not exist

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/sklearn/utils/metaestimators.py:127, in
_AvailableIfDescriptor.__get__(self, obj, owner)
    121 attr_err = AttributeError(
    122     f"This {repr(owner.__name__)} has no attribute
{repr(self.attribute_name)}"
    123 )
    124 if obj is not None:
    125     # delegate only on instances, not the classes.
    126     # this is to allow access to the docstrings.
--> 127     if not self.check(obj):
    128         raise attr_err
    129     out = MethodType(self.fn, obj)

File /opt/hostedtoolcache/Python/3.9.15/x64/lib/python3.9/site-
packages/sklearn/svm/_base.py:819, in BaseSVC._check_proba(self)
    817 def _check_proba(self):
    818     if not self.probability:
--> 819         raise AttributeError(
    820             "predict_proba is not available when  probability=False"
    821         )
    822     if self._impl not in ("c_svc", "nu_svc"):
    823         raise AttributeError("predict_proba only implemented for SVC and
NuSVC")

AttributeError: predict_proba is not available when  probability=False
```

```
def classification_confint(acc, n):
    '''
    Compute the 95% confidence interval for a classification problem.
    acc -- classification accuracy
    n -- number of observations used to compute the accuracy
    Returns a tuple (lb,ub)
    '''
    interval = 1.96*np.sqrt(acc*(1-acc)/n)
    lb = max(0, acc - interval)
    ub = min(1.0, acc + interval)
    return (lb,ub)
```

```
type(dt_opt)
```

```
sklearn.model_selection._search.GridSearchCV
```

```
dt_opt.best_score_, svm_opt.best_score_
```

```
(0.9416666666666668, 0.9833333333333334)
```

```
classification_confint(dt_opt.best_score_, len(iris_X_train))
```

```
(0.8997320725326444, 0.9836012608006891)
```

```
classification_confint(dt_opt.best_score_, len(iris_X_train)*10)
```

```
(0.928405783644842, 0.9549275496884915)
```

1. Portfolio Setup, Data Science, and Python

Due: 2020-09-11

1.1. Objective & Evaluation

This assignment is an opportunity to earn level 1 achievements for the `process` and `python` and confirm that you have all of your tools setup, including your portfolio.

> **Note**
>
> Typically, assignments are for level 2, but this week we did setup stuff in class instead of a comprehension checks. Also, the data science process takes a bit more time to really familiar with, it's hard to explain the modeling step for example, without knowing how to build models.
>
> You will be able to earn python level 2 starting in assignment 2 and process level 2 starting with assignment 6.

1.2. To Do

Your task is to:

1. Install required software from the Tools & Resource page
2. Create your portfolio, by accepting the assignment

3. Learn about your portfolio from the README file on your repository.
4. edit `_config.yml` to set your name as author and (optionally) change the logo image
5. Fill in `about/index.md` with information about yourself(not evaluated, but useful) and your own definition of data science (graded for **level 1 process**)
6. Add a Jupyter notebook called `grading.ipynb` to the `about` folder and write a function that computes a grade for this course, with the following docstring. Include:
    - a Markdown cell with a heading
    - your function called `compute_grade`
    - three calls to your function that verify it returns the correct value for different number of badges that produce at three different letter grades.
    - a basic function that uses conditionals in python will earn **level 1 python**
7. Add the line `- file: about/grading` in your `_toc.yml` file.

> **Important**
>
> remember to add, commit, and push your changes so we can see them

```
    '''
    Computes a grade for CSC/DSP310 from numbers of achievements at each level

    Parameters:
    -----------
    num_level1 : int
      number of level 1 achievements earned
    num_level2 : int
      number of level 2 achievements earned
    num_level3 : int
      number of level 3 achievements earned

    Returns:
    --------
    letter_grade : string
      letter grade with possible modifier (+/-)
    '''
```

1.3. Tips

Here are some sample tests you could run to confirm that your function works correctly:

```
assert compute_grade(15,15,15) == 'A'

assert compute_grade(15,15,13) == 'A-'

assert compute_grade(15,14,14) == 'B-'

assert compute_grade(14,14,14) == 'C-'

assert compute_grade(4,3,1) == 'D'

assert compute_grade(15,15,6) =='B+'
```

1.4. Submission Instructions

1.4.1. If using GitHub in the browser only

Create a Jupyter Notebook with your function in a portfolio folder on your computer where you will save all of your work for the portfolio. Then upload it to GitHub, following GitHub instructions for adding a file

1.4.2. If using git offline

Create a Jupyter Notebook with your function in your portfolio folder, then add, commit, and push the changes.

In your browser, view the `gh-pages` branch to see your compiled submission, as `portfolio.pdf` or by viewing your website.

1.5. Thinking Ahead

> **Note**
>
> Thinking Ahead is an optional section you can do to get a head start on the next level achievements

Add a markdown file to your portfolio called `ideas.md` and start answering some of the following questions:

1. Given what you know about the Data Science Process, which steps do you think you will like most? least?
2. What steps will use the most domain knowledge?
3. What applications of data science do you have domain knowledge for?

2. Assignment 2: Practicing Python and Accessing Data

due : 2022-09-21

2.1. Setting

Next week, we are going to learn about summarizing data. In this assignment, you are going to build a small dataset about datasets. In class next week, we will combine all of your datasets about datasets together in order to be able to answer questions like:

- how much total data did you all load
- how many students picked the same dataset?
- how many total rows of data did each student load?

2.2. Objective & Evaluation

This assignment is an opportunity to earn level 1 and 2 achievements in `python` and `access` and begin working toward level 1 for `summarize`. You can also earn level 1 for `process`.

In this assignment, you'll practice/ review python skills by manipulating datasets and extracting basic information about them.

> **Note**
>
> If you get stuck on any of this after accepting the assignment and creating a repository, you can create an issue on your repository, describing what you're stuck on and tag us: @rhodypro4dg/fall22instructors
>
> To do this click Issues at the top, the green "New Issue" button and then type away.

> **Important**
>
> If you have trouble, check the GitHub FAQ on the left before e-mailing

> **Warning**
>
> your function can have a different name than `compute_grade`, but make sure it's your function name, with those parameter values in your tests.

> **Note**
>
> when the value of the expression after `assert` is `True`, it will look like nothing happened. `assert` is used for testing

> **Note**
>
> to earn **level 2 python** use pythonic code to write a loop that tests your function's correctness, by iterating over a list or dictionary. You will have many chances to earn level 2 achievement in python so this step is optional.

Task	Skills (max level)
identify possible uses for data in a data science pipeline	[process (1)]
load data from one file format	[access (1)]
load data from at least two of (.csv, .tsv, .dat, database, .json)	[access (2)]
compare the data formats	[access (2)]
complete the assignment in python	[python (1)]
use python data types (eg dictionaries) to prepare information about datasets	[python (2)]
use informative variable names, pythonic iteration, and other common PEP 8 conventions	[python (2)]
display DataFrame properties	[summarize (1)]

Table 2.1 practice python by manipulating data files, load datasets of different types

First, accept the assignment . It contains a notebook with some template structure (and will set you up for grading).

2.3. Find Datasets

Find 3 datasets of interest to you that are provided in at least two different file formats. Choose datasets that are not too big, so that they do not take more than a few second to load. At least one dataset, must have non numerical (eg string or boolean) data in at least 1 column.

In your notebook, create a markdown cell for each dataset that includes:

- heading of the dataset's name
- a 1-2 sentence summary of what the dataset contains and why it was collected
- a "more info" link to where someone can learn about the dataset
- 1-2 questions you would like to answer with that dataset.

Important

After finding datasets, how to do the rest of these steps can be found within the course site (notes and glossary) or the pandas documentation.

Learning to use the documentation effectively is important; libraries will change over time and random pages on the internet will not be updated accordingly, but in a well maintained library the documentation will get upated with changes.

2.4. Store them for loading

Create a list of dictionaries in `datasets.py`, so that there is one dictionary for each dataset. Each dictionary should have the following keys:

`url`	with the url
`short_name`	a short name
`load_function`	(the actual function handle) what function should be used to load the data into a `pandas.DataFrame`.

Table 2.2 Meta Data Description of the dictionary to create

2.5. Make a dataset about your datasets

Import the list from the `datasets` module you created in the step above. Then iterate over the list of dictionaries, and:

1. load each dataset from the url
2. save the dataset to a local csv using the short name you provided for the dataset as the file name, without writing the index column to the file.
3. record attributes about the dataset as in the table below in a list of lists or dictionary
4. Use that to create a DataFrame with columns that match the rows of the following table.

name	a short name for the dataset
source	a url to where you found the data
num_rows	number of rows in the dataset
num_columns	number of columns in the dataset
num_numerical	number of numerical variables in the dataset

Table 2.3 Meta Data Description of the DataFrame to build

Note

The term iterate is defined in the site glossary.

Hint

DataFrame objects have many input/output methods beyond the read methods that we have seen so far, including methods to save a DataFrame to your computer's hard drive. Saving it to your computer makes a local (not on the internet) copy.

2.6. Explore Your Datasets

Hint

Notice that I refer to loading the datasets in two different ways, once from a URL, once from a relative path. What does that mean about the way that you can store it for use while you iterate?

For one dataset that includes nonnumerical data:

- load it in from your local csv using a relative path
- display the heading and the last 4 rows
- make a numpy array of only the numerical data and save it to a new variable (select these programmatically)
- was the format that the data was provided in a good format? why or why not?

For any other dataset:

- load it in from your local csv using a relative path
- display the heading with the first three rows
- display the datatype for each column
- Are there any variables where pandas may have read in the data as a datatype that's not what you expect (eg a numerical column mistaken for strings)? If so, investigate and try to figure out why.

For the third dataset:

- load it in from your local csv using a relative path

- display the first 3 multiples of 3 rows (eg 3,6,9) of the data for two columns of your choice

2.7. Exploring data files

There are two files in the data folder, both can be read in with `read_csv` but need some options or fixing.

- try to read in the `german.data` file, what happens with the default settings? What option do you need to use to make it look right?
- try to read in the `.csv` file that's included in the template repository (), use the error messages you get to try to fix the file manually (any text editor, including jupyter can edit a `.csv`), making notes about what changes you made in a markdown cell.

2.8. Submission

This time you have to separately submit from posting your code to make grading easier. Go to the actions tab and run the action called "Submit".

2.9. Thinking ahead

> **Important**
>
> his section is not required, but is intended to help you get started thinking about ideas for your portfolio. If you complete it, we'll give your feedback to help shape your ideas to get to level 3 achievements. If you want to focus only on level 2 at this moment in time, feel free to skip this part. You could also think about this after submitting the assignment, since you do not have to get a grade for it. If you want, you could discuss these ideas in office hours.

1. When might you prefer one datatype over another?
2. How does PEP 8 standard code help you be collaborative?
3. Learn about Datasheets for Datasets and find some examples, (eg this google scholar result) How could something like this impact your work as a datascientist?

3. Assignment 3: Exploratory Data Analysis

__Due:2022-09-28 11:59pm __

Template repo for submission

3.1. Objective & Evaluation

This week your goal is to do a small exploratory data analysis for two datasets of your choice.

Eligible Skills:

- summarize
- visualize
- access

3.2. Choose Datasets

Each Dataset must have at least three variables, but can have more. Both datasets must have multiple types of variables. These can be datasets you used last week, if they meet the criteria below.

3.2.1. Dataset 1 (d1)

must include at least:

- two continuous valued variables and
- one categorical variable.

3.2.2. Dataset 2 (d2)

must include at least:

- two categorical variables and
- one continuous valued variable

3.3. EDA

Use a separate notebook for each dataset, name them `dataset_01.ipynb` and `dataset_02.ipynb`.

For **each** dataset, in a dedicated notebook, complete the following:

1. Load the data to a notebook as a `DataFrame` from url or local path, if local, include the data in your repository.
2. Explore the dataset in a notebook enough to describe its structure use the heading `## Description`
    - shape
    - columns
    - variable types
    - overall summary statisics
3. Write a short description of what the data contains and what it could be used for
4. Ask and answer 4 questions by using and interpreting statistics and visualizations as appropriate. Include a heading for each question using a markdown cell and H2:`##`. Make sure your analyses meet the criteria in the check lists below.
5. Describe what, if anything might need to be done to clean or prepare this data for further analysis in a finale `## Future analysis` markdown cell in your notebook.

3.3.1. Question checklist

be sure that every question (all eight, 4 per dataset) has:

- a heading
- at least 1 statistic or plot
- interpretation that answers the question

3.3.2. Dataset 1 Checklist

make sure that your `dataset_01.ipynb` has:

- Overall summary statistics grouped by a categorical variable
- A single statistic grouped by a categorical variable
- at least one plot that uses 3 total variables
- a plot and summary table that convey the same information. This can be one statistic or many.

3.3.3. Dataset 2 Checklist

- two individual summary statistics for one variable
- one summary statistic grouped by two categorical variables
- a figure with a grid of subplots that correspond to two categorical variables
- a plot and summary table that convey the same information. This can be one statistic or many.

> **Tip**
>
> Be sure to start early and use help hours to make sure you have a plan for all of these.

3.4. Peer Review

> **Note**
>
> This is optional, but if you do a review, you only need to do one analysis each.

> **Warning**
>
> Be familiar with the collaboration policy before you choose to go this route

> **Important**
>
> get group permissions in advance

With a partner (or group of 3 where person 1 review's 2 work, 2 reviews 3, and 3 reviews 1) read your partner's notebook and complete a peer review on their pull request. You can do peer review when you have done most of your analysis, and explanation, even if some parts of the code do not work. After you each do your reviews, update your own analysis.

3.4.1. Review

In your review:

- Use inline comments to denote places that are confusing or if you see solutions to problems your classmate could not solve
- keep the questions below in mind
- Use the template below for your summary review

3.4.1.1. Review Questions

1. How was the analysis overall to read? easy? hard? cohesive? jumpy?
2. Did the data summaries tell you enough about the data to understand the analysis and anticipate what kinds of questions could be answered? If not, what questions do you still have about the data?
3. Do the questions make sense based on the data? Are they interesting questions? What could improve the questions
4. Are the statistics and plots appropriate for the questions?
5. Are the interpretations complete, clear, and consistent with the statistics and plots?
6. What could be done to make the explanations more clear and complete?
7. What additional analysis might make the analysis more compelling and clear?

3.4.1.2. Review Template

```
<!-- delete sections that are not needed -->
## Overall

This analysis was ...

## Data Summaries

- [ ] complete

To understand this analysis I still need to know ...

## Checklist

- [ ] questions fit the data
- [ ] questions are in natural language
- [ ] chosen statistics and plots match questions
- [ ] all statistics and plots have an interpretation in English

## Areas of improvement
```

3.4.2. Response

Respond to your review either inline comments, replies, and by updating your analysis accordingly.

> **Think Ahead**
>
> 1. How could you make more customized summary tables?
> 2. Could you use any of the variables in this dataset to add more variables that would make interesting ways to apply split-apply-combine? (eg thresholding a continuous value to make a categorical value)

> **Warning**
>
> This section is not required, but is intended to help you get started thinking about ideas for your portfolio. If you complete it, we'll give your feedback to help shape your ideas to get to level 3 achievements. If you want to focus only on level 2 at this moment in time, feel free to skip this part.

4. Assignment 4: Cleaning Data

Due: 2022-10-06 6:59pm

accept the assignment

Eligible skills:

- prepare
- summarize
- visualize
- access

4.1. Check the Datasets you have worked with already

In the datasets you have used in your past assignments identify at least one thing you could not do because the data was not in an appropriate format.

Apply one fix and show one summary statistic or plot to show that it works.

Some examples:

- a column that was a list
- missing values
- a column that was continuous, but more interesting as a categorical
- too many header rows

> **Think Ahead**
>
> *this box is not required, but ideas for portfolio* cleaning a dataset to make it able to answer questions that were not possible could satisfy the level 3 prepare requirements.

4.2. Clean example datasets

There are notebooks in the template that have instructions for how to work with each dataset, including how to load it and what high level cleaning should be done. Your job is to execute.

To earn prepare level 2, clean one dataset and do just enough exploratory data analysis to show that the data is usable (eg 1 stat and/or plot).

To *also* earn python level 2: clean the CS degrees dataset (use a function or lambda AND loop or comprehension)

To *also* earn access level 2: clean the airline data (to get data in a second file type).

To *also* earn summarize and/or visualize level 2: add extra exploratory data analyses of your cleaned dataset meeting the criteria from the checklist (eg follow a3 checklists).

This means that if you want to earn prepare, python, and access, you will need to clean two datasets.

> **Hint**
>
> renaming things is often done well with a dictionary comprehension or lambda.

4.3. Study Cleaned Datasets

Read example data cleaning notes or scripts. To do this find at least one dataset for which the messy version, clean version, and a script or notes about how it was cleaned are available, answer the following questions in a markdown file or additional notebook in your repository. (some example datasets and one is in the notes are added to the course website)

1. What are 3 common problems to look for in a dataset?
2. For a specific of your choice give an example of a question that would require making different choices than were made. Include a bit about the data, what was done, the question, what would need to be done instead and justification.
3. Explain in your own words, with a concrete example, how domain expertise can help you when cleaning data. Use either a made up example or one that you read about.

> **Important**
>
> Remember to run the "Submit" Workflow from the actions tab of your repository. see how on the How tos page

5. Assignment 5: Constructing Datasets

accept the assignment

Due: 2020-10-12

Eligible skills: (links to checklists)

- **first chance** construct 1 and 2
- (last assignment*) access 1 and 2
- (last assignment*) python 1 and 2
- (last assignment*) prepare 1 and 2
- summarize 1 and 2
- visualize 1 and 2

these skills will be eligible in future portfolio checks, but not future assignments

let me know if listing the skills this way is better for you

5.1. Constructing Datasets

> **Hint**
>
> there is a section of datasets that are provided in multiple parts

Your goal is to programmatically construct three (3) ready to analyze datasets from multiple sources.

- Each dataset must combine at least 2 source tables(minimum 4 total source tables).
- At least one source table must come from a database or from web scraping.
- You should use at least two different joins(types of merges, or concat).

The notebook you submit should include:

- a motivating question for why you're combining the datasets in an introduction section
- code and description of how you built and prepared each dataset. For each step, describe what you're about to do, the code with output, interpretation that leads into the next step.
- exploratory data analysis that shows why you built the data and confirms that is prepared enough to analyze.
- For one pair of tables, show how a different merge could answer a different question.

For construct only you can include very minimal EDA.

If you get really stuck finding data

You could also demonstrate understanding of how merges work by converting a dataset that is provided as a single table with redundant information into a number of smaller tables and putting them *into* a database instead of taking data *from* a database.

5.2. Additional achievements

Important

I want you to learn and I need evidence that you have learned to give you credit for it (first achievements then a semester grade). It is not important to me exactly **how* you provide me that evidence. We grade by looking at the checklists for the corresponding skills.

To earn additional achievements, you must do more cleaning and/or exploratory data analysis.

5.2.1. Prepare level 2

To earn level 2 for prepare, you must, either on component table(s) or the final dataset apply and explain transformations that meet whatever components are unchecked on your prepare level 2 issue.

5.2.2. Summarize and Visualize level 2

To earn level 2 for summarize and/or visualize, include additional analyses after building the datasets.

Check your issues for what components we have not seen from you.

5.2.3. Python Level 2

Use pythonic naming conventions throughout, AND:

- Use pythonic loops and a list or dictionary OR
- use a list or dictionary comprehension

Thinking Ahead

Compare the level 2 skill definitions to level 3, how could you extend and adapt what you've done to meet level 3?

6. Assignment 6: Auditing Algorithms

accept the assignment

Due: 2020-10-20

Eligible skills: (links to checklists)

- **first chance** evaluate 1 and 2
- construct 1 and 2
- summarize 1 and 2
- visualize 1 and 2
- python 1 and 2

Important

the definition for evaluate is going to be updated in your gradetrackers soon, the version on the site is correct

6.1. About the data

We have provided a version of the [Adult] Dataset, which is a popular benchmark dataset for training machine learning models that comes from a recent paper about the risks of that dataset. The classic Adult dataset tries to predict if a person makes more or less than 50k.

Researchers reconstructed the Adult dataset with the actual value of the income. We trained models to predict `income>=$10k`, `income>=$20k` , etc. We used three different learning algorithms, nicknamed 'LR', 'GPR', and 'RPR' for each target.

`adult_models_only.csv` has the model's predictions and `adult_reconstruction_bin.csv` has the data. Both have a unique identifier column included.

Think Ahead

Why might the dataset have more samples in it than the model predictions one?

6.2. Complete an audit

Thoroughly audit any one model. In your audit, use three different performance metrics. Compare and contrast performance in those metrics across both racial and gender groups.

Include easy to read tables with your performance metrics and interpretations of the model's overall performance and any disparities that could be understood by a general audience.

If the model you chose was used for some real world decision what might the risks be?

6.3. Extend your Audit

Note

optional (for more Achievements or deeper understanding/more practice)

Use functions and loops to build a dataset about the performance of the different models so that you can answer the following questions:

1. Which model (target and learning algorithm) has the best accuracy?
2. Which target value has the least average disparity by race? by gender?
3. Which learning algorithm has the least average disparity by race? by gender?
4. Which model (target and learning algorithm) do you think is overall the best?

y	model	score	value	subset
`>=10k`	LR	accuracy	.873	overall
`>=20k`	RPR	false_pos_rate	.873	men

Table 6.1 Example table format

This table is not real data, just headers with one example value to help illustrate what the column name means.

> **Hint**
>
> This step you should make separate data frames and then merge them together for construct. If you don't need construct you can build it as one, for visualize you should use appropriate groupings

7. Assignment 7: Classification

accept the assignment

Due: 2020-10-26

Eligible skills: (links to checklists)

- **first chance** classification 1 and 2
- evaluate 1 and 2
- summarize 1 and 2
- visualize 1 and 2
- python 1 and 2

7.1. Dataset and EDA

Choose a datasets that is well suited for classification and that has all numerical features. If you want to use a dataset with nonnumerical features you will have to convert the categorical features to one hot encoding.

> **Hint**
>
> Use the UCI ML repository

1. Include a basic description of the data(what the features are)
2. Write your own description of what the classification task is
3. Use EDA to determine if you expect the classification to get a high accuracy or not.
4. Explain why or why not Gaussian Naive Bayes and Decision Trees are a reasonable model to try for this data.
5. Hypothesize which will do better and why you think that.

7.2. Basic Classification

1. Fit a your chosen classifier with the default parameters on 50% of the data
2. Test it on 50% held out data and generate a classification report
3. Inspect the model to answer the questions appropriate to your model.
    - Does this model make sense?
    - (DT) Are there any leaves that are very small?
    - (DT) Is this an interpretable number of levels?
    - (GNB) do the parameters fit the data well?
    - (GNB) do the parameters generate similar synthetic data
4. Repeat the split, train, and test steps 5 times to use 5 different random splits of the data, save the scores into a dataframe. Compute the mean and std of the scores.
    - Is the performance consistent enough you trust it?
5. Interpret the model and its performance in terms of the application. Example questions to consider in your response include

- do you think this model is good enough to use for real?
- is this a model you would trust?
- do you think that a more complex model should be used?
- do you think that maybe this task cannot be done with machine learning?

7.3. Exploring Problem Setups

> **Important**
>
> Understanding the impact of test/train size is a part of classification. This exercise is *also* a chance at python level 2.

Do an experiment to compare test set size vs performance:

1. Train a model (if decision tree set the max depth 2 less than the depth it found above) on 10%, 30%, … , 90% of the data. Compute the training accuracy and test accuracy for each size training data in a DataFrame with columns ['train_pct','n_train_samples','n_test_samples','train_acc','test_acc']
2. Plot the accuracies vs training percentage in a line graph.
3. Interpret these results. How does training vs test size impact the model?

use a loop for this part, possibly also a function

> **Tip**
>
> The summary statistics and visualization we used before are useful for helping to investigate the performance of our model. We can try fitting a model with different settings to create a new "dataset" for our experiments. The same skills apply.

> **Hint**
>
> The most important thing about the max depth here is that it's the same across all of the models. If you get an error, try making it smaller.

Thinking Ahead

ideas for level 3

Repeat the problem setup experiment with cross validation and plot with error bars.

- What is the tradeoff to be made in choosing a test/train size?
- What is the best test/train size for this dataset?

or with variations:

- allowing it to figure out the model depth for each training size, and recording the depth in the loop as well.
- repeating each size 10 items, then using summary statistics on that data

Use the extensions above to experiment further with other model parameters.

some of this we'll learn how to automate in a few weeks, but getting the ideas by doing it yourself can help

8. Assignment 8: Linear Regression

8.1. Quick Facts

- accept the assignment
- **Due: 2022-11-04**

8.2. Assessment

Eligible skills: (links to checklists)

- **first chance** regression 1 and 2
- evaluate 1 and 2
- summarize 1 and 2
- visualize 1 and 2

8.3. Related notes

- 2022-10-24
- 2022-10-26
- 2022-10-28

8.4. Instructions

Find a dataset suitable for regression. We recommend a dataset from the UCI repository.

8.4.1. Linear Regression Basics

Fit a linear regression model, measure the fit with two metrics, and make a plot that helps visualize the result.

1. Include a basic description of the data(what the features are)
2. Write your own description of what the prediction task it, why regression is appropriate.
3. Fit a linear model with 75% training data.
4. Test it on 25% held out test data and measure the fit with two metrics and one plot
5. Inspect the model to answer:
    - Does this model make sense?
    - What to the coefficients tell you?
    - What to the residuals tell you?
6. Repeat the split, train, and test steps 5 times.
    - Is the performance consistent enough you trust it?
7. Interpret the model and its performance in terms of the application. Some questions you might want to answer in order to do this include:

- do you think this model is good enough to use for real?
- is this a model you would trust?
- do you think that a more complex model should be used?
- do you think that maybe this task cannot be done with machine learning?

1. Try fitting the model only on one feature. Justify your choice of feature based on the results above. Plot this result.

8.4.2. Part 2: Exploring Evaluation

Do an experiment to compare test set size vs performance:

1. Train a regression model on 10%, 30%, … , 90% of the data. Save the results of both test and train performance for each size training data in a DataFrame with columns ['train_pct','n_train_samples','n_test_samples','train_r2','test_r2']
2. Plot the accuracies vs training percentage in a line graph.
3. Interpret these results. How does training vs test size impact the model?

Tip

The summary statistics and visualization we used before are useful for helping to investigate the performance of our model. We can try fitting a model with different settings to create a new "dataset" for our experiments. The same skills apply.

Thinking Ahead

Try these experiments with a different type of regression.

9. Assignment 9

accept the assignment

Due: 2022-11-11

9.1.

Eligible skills: (links to checklists)

- **first chance** clustering 1 and 2
- evaluate 1 and 2
- python 1 and 2
- summarize 1 and 2
- visualize 1 and 2

9.2. Related notes

- 2022-10-31
- 2022-11-02
- 2022-11-04

9.3. Instructions

Use the same dataset you used for assignment 7, unless there was a problem, or pick one of the recommended ones for that assignment if you did not complete assignment 7.

1. Describe what question you'd be asking in applying clustering to this dataset.
2. Apply Kmeans using the known, correct number of clusters, K.
3. Evaluate how well clustering worked on the data:
    - using a true clustering metric and
    - using visualization and
    - using a clustering metric that uses the ground truth labels
4. Include a discussion of your results that addresses the following:
    - describes what the clustering means
    - what the metrics show
    - Does this clustering work better or worse than expected based on the classification performance (if you didn't complete assignment 7, also apply a classifier)
5. Repeat your analysis using a 2 different numbers (1 higher, one lower) of clusters:
    - can you interpret the new clusters?
    - how to they relate to the original clusters? are they completely different, did one split?
    - is there a reasonable explanation for more clusters than there are classes in this dataset?

10. Assignment 10: Tuning Model Parameters

10.1. Quick Facts

- accept the assignment
- **Due: 2020-11-16**

10.2. Related notes

- 2022-11-07
- 2022-11-09

10.3. Assessment

Eligible skills: (links to checklists)

- **first chance** optimization 1 and 2
- clustering 1 and 2
- regression 1 and 2
- classification 1 and 2
- evaluate (must use extra metrics to earn this here) 1 and 2
- summarize 1 and 2
- visualize 1 and 2

10.4. Instructions

summary Extend the work you did in assignment 7,8, or 9, by optimizing the model parameters.

1. Choose your dataset, task, and a model. It can be any model we have used in class so far. Include a breif description of the task.
2. Fit the model with default parameters and check the score. Interpret the score in context.
3. Choose reasonable model parameter values for your parameter grid by assessing how well the model fit. Examine the model.
4. Use grid search to find the best performing model parameters.
5. Examine the best fit model, how different is it from the default? Score the best fit model on a held out test set.
6. Examine and interpret the cross validation results. How do they vary in terms of time? Is the performance meaningfully different or just a little?
7. Try varying the cross validation parameters (eg the number of folds and/or type of cross validation). Does this change your conclusions?

> **Tip**
>
> this is best for regression or classification, but if you use clustering use the `scoring` parameter to pass better metrics than the default of the score method.

> **Hint**
>
> Assignment 11 will be to optimize two models and then compare two models on the same task

> **Thinking Ahead**
>
> What other tradeoffs might you want to make in choosing a model? How could you present these results using your EDA skills?

Portfolio

```
/tmp/ipykernel_2485/1159722460.py:26: FutureWarning: Using the level keyword in
DataFrame and Series aggregations is deprecated and will be removed in a future
version. Use groupby instead. df.sum(level=1) should use
df.groupby(level=1).sum().
  assignment_dummies  =
pd.get_dummies(rubric_df['assignments'].apply(pd.Series).stack()).sum(level=0)
/tmp/ipykernel_2485/1159722460.py:31: FutureWarning: Using the level keyword in
DataFrame and Series aggregations is deprecated and will be removed in a future
version. Use groupby instead. df.sum(level=1) should use
df.groupby(level=1).sum().
  portfolio_dummies  =
pd.get_dummies(rubric_df['portfolios'].apply(pd.Series).stack()).sum(level=0)
```

This section of the site has a set of portfolio prompts and this page has instructions for portfolio submissions.

Starting in week 3 it is recommended that you spend some time each week working on items for your portfolio, that way when it's time to submit you only have a little bit to add before submission.

The portfolio is your only chance to earn Level 3 achievements, however, if you have not earned a level 2 for any of the skills in a given check, you could earn level 2 then instead. The prompts provide a starting point, but remember that to earn achievements, you'll be evaluated by the rubric. You can see the full rubric for all portfolios in the syllabus. Your portfolio is also an opportunity to be creative, explore things, and answer your own questions that we haven't answered in class to dig deeper on the topics we're covering. Use the feedback you get on assignments to inspire your portfolio.

Each submission should include an introduction and a number of 'chapters'. The grade will be based on both that you demonstrate skills through your chapters that are inspired by the prompts and that your summary demonstrates that you *know* you learned the skills. See the formatting tips for advice on how to structure files.

On each chapter(for a file) of your portfolio, you should identify which skills by their keyword, you are applying.

You can view a (fake) example in this repository as a pdf or as a rendered website

Upcoming Checks

- Portfolio Check 2 is due November 14
- Portfolio check 3 is due December 6
- Portfolio check 4 is due December 19

> **Important**
>
> start early, assignment 9 and 10 will be due on Wednesdays like regular the week before and of this deadline. You will get feedback on Assignment 9 by Friday so that you can use that to update your portfolio on the construct achievements.

Portfolio check 2 will assess the following *new* achievements in addition to an a chance to make up any that you have missed:

	Level 3
keyword	
python	reliable, efficient, pythonic code that consistently adheres to pep8
process	Compare different ways that data science can facilitate decision making
access	access data from both common and uncommon formats and identify best practices for formats in different contexts
construct	merge data that is not automatically aligned
summarize	Compute and interpret various summary statistics of subsets of data
visualize	generate complex plots with pandas and plotting libraries and customize with matplotlib or additional parameters
prepare	apply data reshaping, cleaning, and filtering manipulations reliably and correctly by assessing data as received
evaluate	Evaluate a model with multiple metrics and cross validation
classification	fit and apply classification models and select appropriate classification models for different contexts
regression	fit and explain regrularized or nonlinear regression
clustering	apply multiple clustering techniques, and interpret results

Formatting Tips

> **Warning**
>
> This is all based on you having accepted the portfolio assignment on github and having a cloned copy of the template. If you are not enrolled or the initial assignment has not been issued, you can view the template on GitHub

Your portfolio is a jupyter book. This means a few things:

- it uses myst markdown
- it will run and compile Jupyter notebooks

This page will cover a few basic tips.

Managing Files and version

You can either convert your ipynb files to earier to read locally or on GitHub.

The GitHub version means installing less locally, but means that after you push changes, you'll need to pull the changes that GitHub makes.

To manage with a precommit hook jupytext conversion

change your `.pre-commit-config.yaml` file to match the following:

```
repos:
-   repo: https://github.com/mwouts/jupytext
    rev: v1.10.0  # CURRENT_TAG/COMMIT_HASH
    hooks:
    - id: jupytext
      args: [--from, ipynb, --to, myst]
```

Run Precommit over all the files to actually apply that script to your repo.

```
pre-commit install
pre-commit run --all-files
```

If you do `git status` now, you should have a `.md` file for each `ipynb` file that was in your repository, now add and commit those.

Now, each time you commit, it will run jupytext first.

To manage with a gh action jupytext conversion

create a file at `.github/workflows/jupytext.yml` and paste the following:

```
name: jupytext

# Only run this when the master branch changes
on:
  push:
    branches:
    - main
    # If your git repository has the Jupyter Book within some-subfolder next to
    # unrelated files, you can make this run only if a file within that specific
    # folder has been modified.
    #
    # paths:
    # - some-subfolder/**

# This job installs dependencies, build the book, and pushes it to `gh-pages`
jobs:
  jupytext:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    # Install dependencies
    - name: Set up Python 3.7
      uses: actions/setup-python@v1
      with:
        python-version: 3.7

    - name: Install dependencies
      run: |
        pip install jupytext
    - name: convert
      run: |
          jupytext */*.ipynb --to myst
          jupytext *.ipynb --to myst
    - uses: EndBug/add-and-commit@v4 # You can change this to use a specific version
      with:
        # The arguments for the `git add` command (see the paragraph below for more info)
        # Default: '.'
        add: '.'

        # The name of the user that will be displayed as the author of the commit
        # Default: author of the commit that triggered the run
        author_name: Your Name

        # The email of the user that will be displayed as the author of the commit
        # Default: author of the commit that triggered the run
        author_email: you@uri.edu

        # The local path to the directory where your repository is located. You should use actions/checkout first to set it up
        # Default: '.'
        cwd: '.'

        # Whether to use the --force option on `git add`, in order to bypass eventual gitignores
        # Default: false
        force: true

        # Whether to use the --signoff option on `git commit`
        # Default: false
        signoff: true

        # The message for the commit
        # Default: 'Commit from GitHub Actions'
        message: 'convert notebooks to md'

        # Name of the branch to use, if different from the one that triggered the workflow
        # Default: the branch that triggered the workflow (from GITHUB_REF)
        ref: 'main'

        # Name of the tag to add to the new commit (see the paragraph below for more info)
        # Default: ''
        tag: "v1.0.0"

      env:
        # This is necessary in order to push a commit to the repo
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }} # Leave this line unchanged
```

Organization

The summary of for the `part` or whole submission, should match the skills to the chapters. Which prompt you're addressing is not important, the prompts are a *starting point* not the end goal of your portfolio.

Data Files

Also note that for your portfolio to build, you will have to:

- include the data files in the repository and use a relative path OR
- load via url

using a full local path(eg that starts with `///file:`) **will not work** and will render your portfolio unreadable.

Structure of plain markdown

Use a heading like this:

```
# Heading of page
## Heading 2
### Heading 3
```

in the file and it will appear in the sidebar.

You can also make text *italic* or **bold** with either `*asterics*` or `__underscores__` with `_one for italic_` or `**two for bold**` in either case

File Naming

It is best practice to name files without spaces. Each `chapter` or file should have a descriptive file name (`with_no_spaces`) and descriptive title for it.

Syncing markdown and ipynb files

If you have the precommit hook working, git will call a script and convert your notebook files from the ipynb format (which is json like) to Myst Markdown, which is more plain text with some header information. The markdown format works better with version control, largely because it doesn't contain the outputs.

If you don't get the precommit hook working, but you do get jupytext installed, you can set each file to sync.

Adding annotations with formatting or margin notes

You can either install jupytext and convert locally or upload /push a notebook to your repository and let GitHub convert.
Then edit the .md file with a text editor of your choice. You can run by uploading if you don't have jupytext installed, or locally if you have installed jupytext or jupyterbook.

In your .md file use backticks to mark special content blocks

````
```{note}
Here is a note!
```
````

````
```{warning}
Here is a warning!
```
````

````
```{tip}
Here is a tip!
```
````

````
```{margin}
Here is a margin note!
```
````

For a complete list of options, see the **`sphinx-book-theme`** documentation.

Links

Markdown syntax for links

```
[text to show](path/or/url)
```

Configurations

Things like the menus and links at the top are controlled as settings, in `_config.yml`. The following are some things that you might change in your configuration file.

Show errors and continue

To show errors and continue running the rest, add the following to your configuration file:

```
# Execution settings
execute:
  allow_errors             : true
```

Using additional packages

You'll have to add any additional packages you use (beyond pandas and seaborn) to the `requirements.txt` file in your portfolio.

Portfolio Check 1 Ideas

Remember you'll be graded against the [rubric] and the [achievement checklists], but these are ideas for the structure.

You can mix and match different formats to cover the skills collectively.

If you goal is, for example, a B+ (you need 5 level 3s) you could only do 1-2 skills per portfolio check (there are 4).

Long single analysis

Collect data from multiple sources, prepare each for analysis, and merge them together then do some exploratory data analysis. Describe each step, interpret all outputs, and put the analysis in context of the Data Science Process.

This would be one long notebook that covers all of the skills.

Several shorter reflections/analyses

You could also submit a few shorter pieces that in total cover all of the skills. Some example formats:

Tutorial

Write a notebook that explains a concept related to a skill with examples in a real dataset and with visuals or a toy dataset (minimal number of columns rows)

Cheatsheet

Make a detailed reference with code outputs on a topic or a few topics.

Blog post

Write a blog post styled Notebook that compares or analyzes something, for example:

- how do different ways of loading data compare
- describe best practices you've learned and show why they're good with examples

Correction & Reflection

If you had trouble with an assignment so far, you can revise what you submitted and resubmit it, with reflections and explanation of what you were confused about, what you tried initially, how you eventually figured it out, and explains the correct answer. Then go a little deeper in exploring the topic in that context to also earn level 3.

Practice Problems and Solutions

Based on the level 3 rubric descriptions, write practice problems that build off of the lecture notes. Include solutions and descriptions for each. These can be open ended or multiple choice questions with plausible distractors. A plausible distractor is an incorrect answer that represents a way that you think someone could misunderstand.

For example if the question is 37 + 15 = ?, MCQ with plausible distractors might be:

- 52 (correct)
- 412 (didn't carry the one, correctly: 7+5 = 12, 3+1 = 4)
- 42 (dropped the one 7+5 = 12, ones place is 2, 3+1 = 4)
- 43 (carried one into wrong column, 7 + 5 = 12, 1+2 = 3, 3+1 = 3)

Check 2 Ideas

For Check 2, all of the prompts from check 1 apply, plus the following additional prompts, since there are new skills.

If you have other ideas, you can also ask and those are likely posible.

Level 1 Achievement Catchup

To make up level 1 achievements, include a detailed introduction file to your portfolio and one of the following (per skill):

- minor extensions to what we did in class
- answers to problems from the notes
- additional glossary terms
- psuedocode for one of the other prompts

Extend Assignment 7, 8, or 9

Assignments 7-9 help you think through what machine learning tasks are. Extend those ideas by adding additional experiments based on your own questions or the questions in your feedback.

Build a data set for Prediction

Build a dataset that works for prediction (classification, regression, or clustering) from other sources.

> **Tip**
>
> If you are short on time these achievements are easiest to combine with the optimize, compare, representation and workflow in the next portfolio

Learn a new model

Repeat what you did in 7, 8, or 9, with a different model.

Create datasets that fail

Create datasets that violate assumptions of a model we have learned. The sklearn data generators are a good place to start.

Process level 3

Process level 3 is a little different than most of the others. You may be able to work it into an analysis notebook, but likely, you'll need to do one of the following.

Data Science Pipeline Comparisons

Find two different sources that describe the data science pipeline or lifecycle. Write a blog style post that discusses their differences and hypothesizes about why they may be different? Are they for different audiences? Is one domain specific? How do they emphasize different modeling tasks? Include a Recommendation for when you think each one is better

Write a short story

Write a short story that explains the concepts of data science to demonstrate your understanding of process.

Media Review

Watch/listen/read to an episode of a high quality[1] podcast or other type of media and write a blog style summary and review. Highlight what you learned and how it relates to topics covered in class.

Approved Media:

- Pod of Asclepius, Fall Series: The Philosophy of Data Science
- Chapter 1 & 2 of Think like a Data Scientist in particular, if you think these would be helpful to assign as reading or teach from at the beginning of the semester next year.
- Algorithms of Oppression (book)
- Weapons of Math Destruction (book)
- Coded Bias (film, available on netflix & PBS)

[1] approved Dr. Brown by creating a pull request to add it to the list on this page that is successfully merged. To create a PR, use the suggest an edit button at the top of this page.

FAQ

This section will grow as questions are asked and new content is introduced to the site. You can submit questions:

- via e-mail to Dr. Brown (brownsarahm) or Beibhinn (beibhinn)
- via Prismia.chat during class
- by creating an issue

Syllabus and Grading FAQ

How much does assignment x, class participation, or a portfolio check weigh in my grade?

There is no specific weight for any activities, because your grade is based on earning achievements for the skills listed in the skills rubric.

However, if you do not submit (or earn no achievements from) assignments or portfolios, the maximum grade you can earn is a C. If you do not submit (or earn no achievements from) your portfolio, the maximum grade you can earn is a B.

Can I submit this assignment late if …?

Late assignments are not accepted, however, your grade is based on the skills, not the assignments. All skills are assessed in at least two assignments, so missing any one will not hurt your grade. If you need an accommodation because you cannot submit multiple assignments, contact Dr. Brown.

I don't understand my grade on this assignment

If you have questions about your grade, the best place to get feedback is to reply on the Feedback PR. Either reply directly to one of the inline comments, or the summary.

Be specific about what you think you should have earned and why.

Git and GitHub

I can't push to my repository, I get an error that updates were rejected

```
! [rejected] main -> main (fetch first)
error: failed to push some refs to <repository name>
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Your local version and github version are out of sync, you need to pull the changes from github to your local computer before you can push new changes there.

After you run

```
git pull
```

You'll probably have to resolve a merge conflict

The content I added to my portfolio isn't in the pdf

There was an error in the original `_toc.yml` file, change yours to match the following:

```
format: jb-book
root: intro
parts:
  - caption: About
    chapters:
    - file: about/index
    - file: about/grading
#  - caption: Check 1
#    chapters:
#    - file: submission_1_intro
```

uncomment the later lines and add any new files you add.

My command line says I cannot use a password

GitHub has strong rules about authentication You need to use SSH with a public/private key; HTTPS with a Personal Access Token or use the GitHub CLI auth

My .ipynb file isn't showing in the staging area or didn't push

.ipynb files are json that include all of the output, including tables as html and plots as svg, so, unlike plain code files, they don't play well with version control.

Your portfolio has `*/*.ipynb` in the `.gitignore` file, so that these files do not end up in your repository. Instead, you'll convert your notebooks to Myst Markdown with jupytext via a precommit hook.

Your portfolio has the code to do this already, what you should do is make sure that `pre-commit` is installed and then run `pre-commit install`
(see your portfolio's README.md file for more detail)

If this doesn't work, you can follow the alterntive in the porfolio readme.

If that doesn't work, and you have time before the deadline, create an issue to get help.

As a last resort, use the jupyter interface to download (File > Download as > ...)your notebook as `.md` if avialable or `.py` if not and then move that file from your Downloads folder to your repository. We'll set up another workflow for future work

My portfolio won't compile

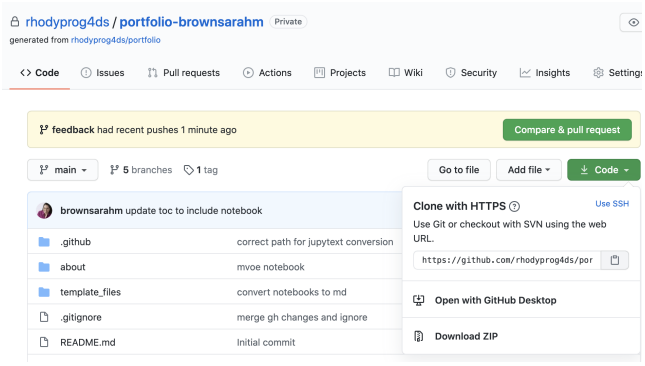
If there's an error your notebook it can't complete running. You can allow it to run if the error is on purpose by changing settings as mentioned on the formatting page.

Help! I accidentally merged the Feedback Pull Request before my assignment was graded

That's ok. You can fix it.

You'll have to work offline and use GitHub in your browser together for this fix. The following instuctions will work in terminal on Mac or Linux or in GitBash for Windows. (see Programming Environment section on the tools page).

First get the url to clone your repository (unless you already have it cloned then skip ahead): on the main page for your repository, click the green "Code" button, then copy the url that's show



Next open a terminal or GitBash and type the following.

```
git clone
```

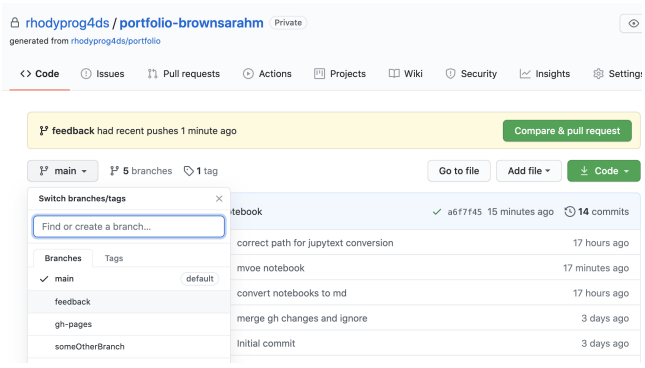
then past your url that you copied. It will look something like this, but the last part will be the current assignment repo and your username.

```
git clone https://github.com/rhodyprog4ds/portfolio-brownsarahm.git
```

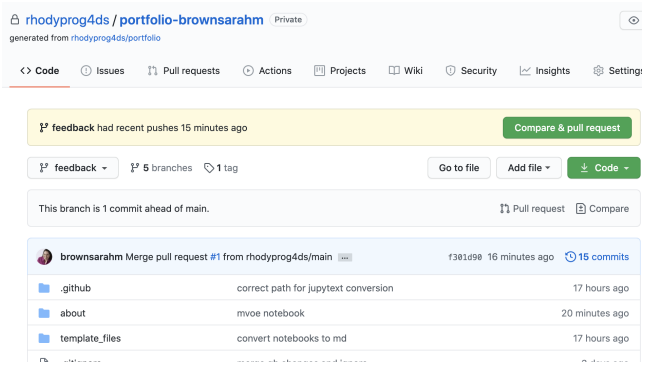
When you merged the Feedback pull request you advanced the `feedback` branch, so we need to hard reset it back to before you did any work. To do this, first check it out, by navigating into the folder for your repository (created when you cloned above) and then checking it out, and making sure it's up to date with the `remote` (the copy on GitHub)

```
cd portfolio-brownsarahm
git checkout feedback
git pull
```

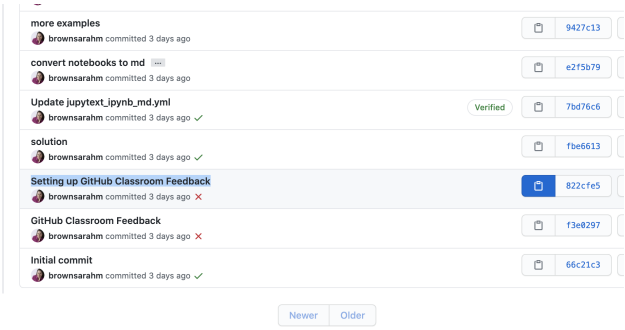
Now, you have to figure out what commit to revert to, so go back to GitHub in your browser, and swithc to the feedback branch there. Click on where it says `main` on the top right next to the branch icon and choose feedback from the list.



Now view the list of all of the commits to this branch, by clicking on the clock icon with a number of commits



On the commits page scroll down and find the commit titled “Setting up GitHub Classroom Feedback” and copy its hash, by clicking on the clipboard icon next to the short version.



Now, back on your terminal, type the following

```
git reset --hard
```

then paste the commit hash you copied, it will look something like the following, but your hash will be different.

```
git reset --hard 822cfe51a70d356d448bcaede5b15282838a5028
```

If it works, your terminal will say something like

```
HEAD is now at 822cfe5 Setting up GitHub Classroom Feedback
```

but the number on yours will be different.

Now your local copy of the `feedback` branch is reverted back as if you had not merged the pull request and what’s left to do is to push those changes to GitHub. By default, GitHub won’t let you push changes unless you have all of the changes that have been made on their side, so we have to tell Git to force GitHub to do this.

Since we’re about to do something with forcing, we should first check that we’re doing the right thing.

```
git status
```

and it should show something like

```
On branch feedback
Your branch is behind 'origin/feedback' by 12 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```

Your number of commits will probably be different but the important things to see here is that it says `On branch feedback` so that you know you’re not deleting the `main` copy of your work and `Your branch is behind origin/feedback` to know that reverting worked.

Now to make GitHub match your reverted local copy.

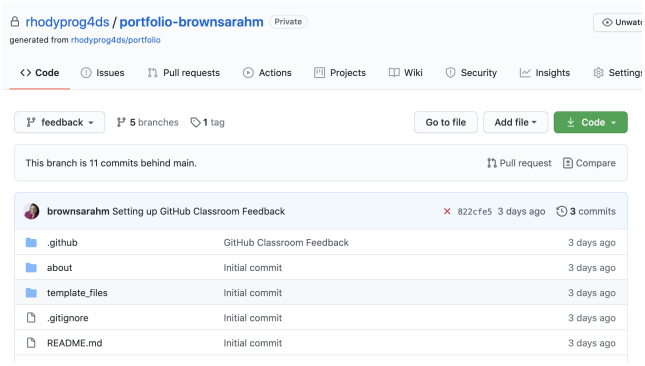
```
git push origin -f
```

and you’ll get something like this to know that it worked

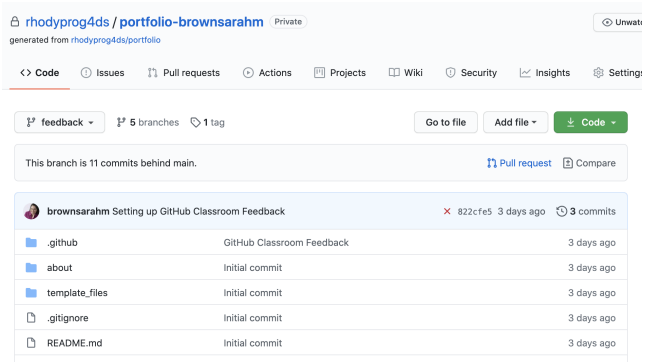
```
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/rhodyprog4ds/portfolio-brownsarahm.git
 + f301d90...822cfe5 feedback -> feedback (forced update)
```

Again, the numbers will be different and it will be your url, not mine.

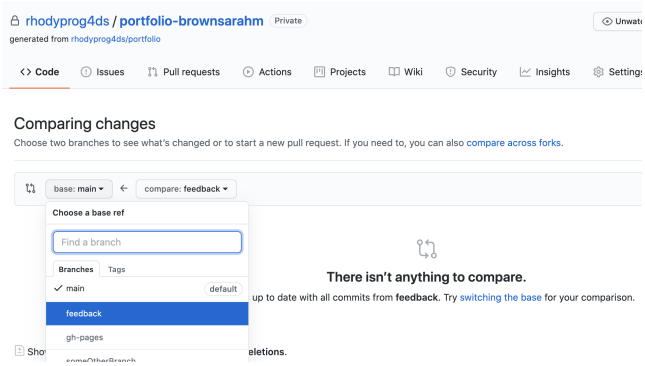
Now back on GitHub, in your browser, click on the code tab. It should look something like this now. Notice that it says, “This branch is 11 commits behind main” your number will be different but it should be 1 less than the number you had when you checked `git status`. This is because we reverted the changes you made to main (11 for me) and the 1 commit for merging main into feedback. Also the last commit (at the top, should say “Setting up GitHub Classroom Feedback”).



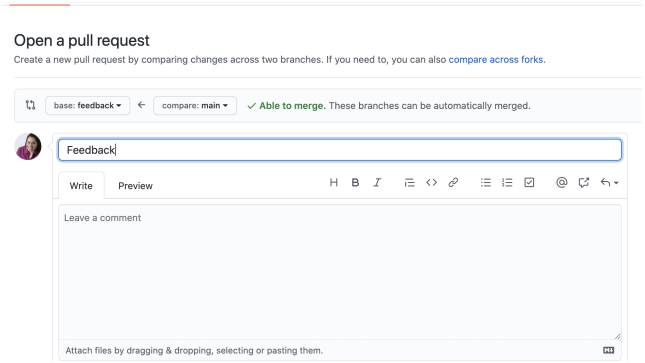
Now, you need to recreate your Pull Request, click where it says pull request.



It will say there isn't anything to compare, but this is because it's trying to use `feedback` to update `main`. We want to use `main` to update `feedback` for this PR. So we have to swap them. Change base from `main` to `feedback` by clicking on it and choosing `feedback` from the list.



Then the change the compare `feedback` on the right to `main`. Once you do that the page will change to the "Open a Pull Request" interface.



Make the title "Feedback" put a note in the body and then click the green "Create Pull Request" button.

Now you're done!

If you have trouble, create an issue and tag `@@rhodyprog4ds/fall22instructors` for help.

Code Errors

Key Error

If you get a key error for a pandas operation, it means that the column name as you typed it is not in the DataFrame. Check the spelling, leading or trailing whitespace can be especially troubling.

<bound method

You're probably missing `()` on a method, so Python returned the method itself as an object instead of calling it and returning the output.

Glossary

> **Ram Token Opportunity**
>
> Contribute glossary items and links for further reading using the suggest an edit button behind the GitHub menu at the top of the page.

aggregate

to combine data in some way, a function that can produce a customized summary table

anonymous function

a function that's defined on the fly, typically to lighten syntax or return a function within a function. In python, they're defined with the lambda keyword.

BeautifulSoup

a python library used to assist in web scraping, it pulls data from html and xml files that can be parsed in a variety of different ways using different methods.

corpus

(NLP) a set of documents for analysis

DataFrame

a data structure provided by pandas for tabular data in python.

dictionary

(data type) a mapping array that matches keys to values. (in NLP) all of the possible tokens a model knows

document

unit of text for analysis (one sample). Could be one sentence, one paragraph, or an article, depending on the goal

git

a version control tool; it's a fully open source and always free tool, that can be hosted by anyone or used without a host, locally only.

GitHub

a hosting service for git repositories

index

(verb) to index into a data structure means to pick out specified items, for example index into a list or a index into a data frame. Indexing usually invovlees square brackets `[]` (noun) the index of a dataframe is like a column, but it can be used to refer to the rows. It's the list of names for the rows.

interpreter

the translator from human readable python code to something the computer can run. An interpreted language means you can work with python interactively

iterate

To do the same thing to each item in an iterable data structure, typically, an iterable type. Iterating is usually described as iterate over some data structure and typically uses the `for` keyword

iterable

any object in python that can return its members one at a time. The most common example is a list, but there are others.

kernel

in the jupyter environment, the kernel is a language specific computational engine

lambda

they keyword used to define an anonymous function; lambda functions are defined with a compact syntax `<name> = lambda <parameters>: <body>`

PEP 8

Python Enhancement Proposal 8, the Style Guide for Python Code.

repository

a project folder with tracking information in it in the form of a .git file

residual

the errors in a regression problem; calculated by taking the difference between the true and predicted values

suffix

additional part of the name that gets added to end of a name in a merge operation

Series

a data structure provided by pandas for single columnar data with an index. Subsetting a Dataframe or applying a function to one will often produce a Series

Split Apply Combine

a paradigm for splitting data into groups using a column, applying some function(aggregation, transformation, or filtration) to each piece and combinging in the individual pieces back together to a single table

stop words

Words that do not convey important meaning, we don't need them (like a, the, an,). Note that this is context dependent. These words are removed when transforming text to numerical representation

test accuracy

percentage of predictions that the model predict correctly, based on held-out (previously unseen) test data

Tidy Data Format

Tidy data is a database format that ensures data is easy to manipulate, model and visualize. The specific rules of Tidy Data are as follows: Each variable is a column, each row is an observation, and each observable unit is a table.

token

a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing (typically a word, but more gneeral)

TraceBack

an error message in python that traces back from the line of code that had caused the exception back through all of the functions that called other functions to reach that line. This is sometimes call tracing back through the stack

training accuracy

percentage of predictions that the model predict correctly, based on the training data

Web Scraping

the process of extracting data from a website. In the context of this class, this is usually done using the python library beautiful soup and a html parser to retrieve specific data.

References on Python

Official Documentation

- Python
- Pandas
- Matplotlib
- Seaborn

Key Resources

- Course Text this book roughly covers things that we cover in the course, but since things change quickly, we don't rely on it too closely
- Real Python this site includes high quality tutorials
- Towards Data Science this blog has some good tutorials, but old ones are not always updated, so always check the date and don't rely too much on posts more than 2 years old.

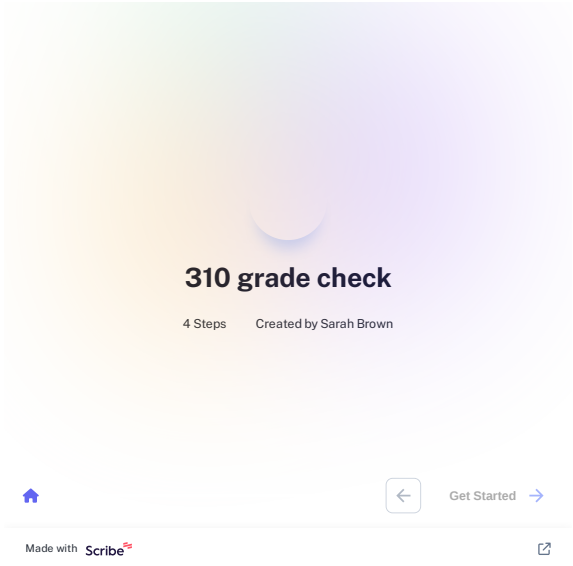
> **Ram Token Opportunity**
>
> If you find other high quality, reliable sources that you want to share, you can earn ram tokens.

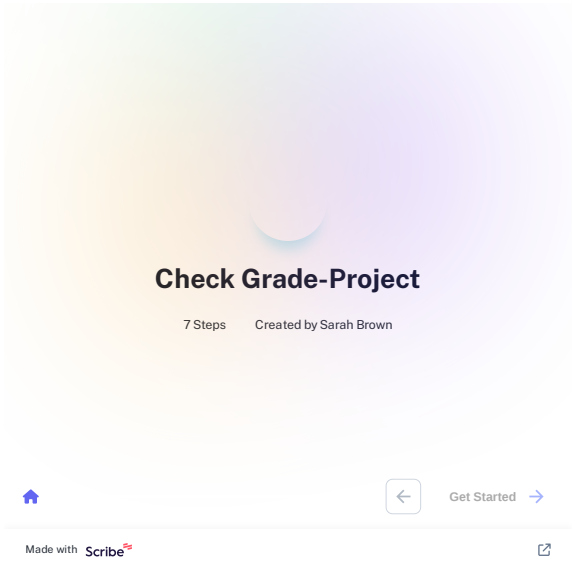
How Tos

Important Info

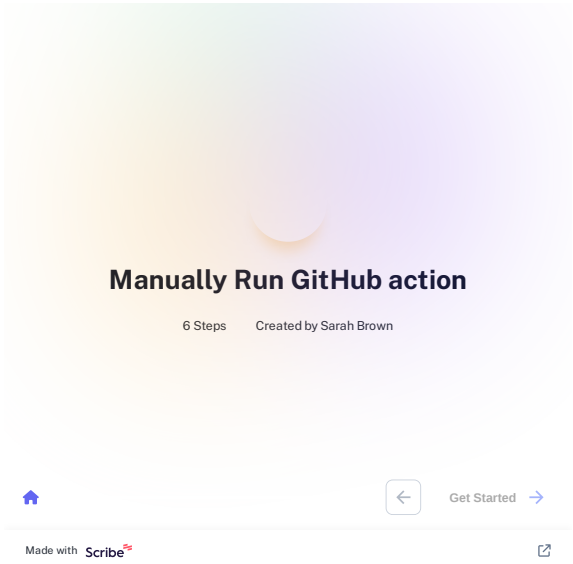
GitHub Org page when you are logged in you see a private version.

Track achievements





Submit Action



Cheatsheet

Patterns and examples of how to use common tips in class

How to use brackets

symbol	use
`[val]`	indexing item val from an object; `val` is int for iterables, or any for mapping
`[val : val2]`	slicing elemtns val to val2-1 from a listlike object
`[ item1,item2 ]`	creating a list consisting of `item1` and `item2`
`(param)`	function calls
`(item1,item2)`	defining a tuple of `item1` and `item2`
`{item1,item2}`	defining a set of `item1` and `item2`
`{key:val1, key2: val2}`	defining a dictionary where key1 indexes to val2

Axes

First build a small dataset that's just enough to display

```
data = [[1,0],[5,4],[1,4]]
df = pd.DataFrame(data = data,
  columns = ['A','B'])

df
```

	A	B
0	1	0
1	5	4
2	1	4

This data frame is originally 3 rows, 2 columns. So summing across rows will give us a Series of length 3 (one per row) and long columns will give length 2, (one per column). Setting up our toy dataset to not be a square was important so that we can use it to check which way is which.

```
df.sum(axis=0)
```

```
A    7
B    8
dtype: int64
```

```
df.sum(axis=1)
```

```
0    1
1    9
2    5
dtype: int64
```

```
df.apply(sum,axis=0)
```

```
A    7
B    8
dtype: int64
```

```
df.apply(sum,axis=1)
```

```
0    1
1    9
2    5
dtype: int64
```

Indexing

```
df['A'][1]
```

```
5
```

```
df.iloc[0][1]
```

```
0
```

Data Sources

This page is a semi-curated source of datasets for use in assignments. The different sections have datasets that are good for different assignments.

Best for loading directly into a notebook

- Tidy Tuesday inside the folder for each year there is a README file with list of the datasets. These are .csv files
- Json Datasets **warning some of these require API calls and that is not recommended until at least A4**
- National Center for Education Statistics Digest 2019 These data tables are available for download as excel and visible on the page.
- Lots of wikipedia pages have tables in them.
- Messy Artists .csv file, that needs to be cleaned, containing data on artists
- Messy Wheels .csv file, that needs to be cleaned, containing data on various wheel attractions around the globe
- Clean Artists .csv file, already cleaned, containing data on artists
- Clean Wheels, .csv file, already cleaned, containing data on various wheel attractions around the globe

General Sources

These may require some more work

- Stackoverflow Developer Survey This data comes with readme info all packaged together in a .zip. You'll need to unzip it first.
- Google Dataset Search
- Kaggle most Kaggle datasets will require you to download and unzip them first and then you can copy them into your repo folder.
- UCI Data Repository Machine Learning focused datasets, can filter by task

- A curated list of datasets by task It includes datasets for cleaning, visualization, machine learning, and “data analysis” which would align with EDA in this course.
- Hugging Face NLP Datasets lots of text datasets

Datasets in many parts

- Makeup Shades
- Kenya Census
- Wealth and Income over time
- UN Votes
- Deforestation
- Survivor
- Billboard
- Caribou Tracking
- Video games from steam 2021 and from 2019
- BBC Rap Artists

Datasets with time

- Superbowl commercials

Databases

- SQLite Databases

If you have others please share by creating a pull request or issue on this repo (from the GitHub logo at the top right, `suggest edit`).

General Tips and Resources

This section is for materials that are not specific to this course, but are likely useful. They are not generally required readings or installs, but are options or advice I provide frequently.

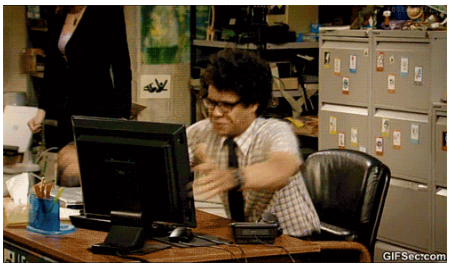
on email

- how to e-mail professors

How to Study in this class

This is a programming intensive course and it’s about data science. This course is designed to help you learn how to program for data science and in the process build general skills in both programming and using data to understand the world. Learning two things at once is more complex. In this page, I break down how I expect learning to work for this class.

Remember the goal is to avoid this:



Why this way?

Learning to program requires iterative practice. It does not require memorizing all of the specific commands, but instead learning the basic patterns. Using reference materials frequently is a built in part of programming, most languages have built in help as a part of the language for this reason. This course is designed to have you not only learn the material, but also to build skill in learning to program. Following these guidelines will help you build habits to not only be successful in this class, but also in future programming.

A new book that might be of interest if you find programming classes hard is the Programmers Brain As of 2021-09-07, it is available for free by clicking on chapters at that linked table of contents section.

> **Where are your help tools?**
>
> In Python and Jupyter notebooks, what help tools do you have?

Learning in class

> **Important**
>
> My goal is to use class time so that you can be successful with *minimal frustration* while working outside of class time.

Programming requires both practical skills and abstract concepts. During class time, we will cover the practical aspects and introduce the basic concepts. You will get to see the basic practical details and real examples of debugging during class sessions. Learning to debug something you’ve never encountered before and setting up your programming envrionment, for example, are *high frustration* activities, when you’re learning, because you don’t know what you don’t know. On the other hand, diving deeper into options and more complex applications of what you have already seen in class, while challenging, is something I’m confident that you can all be successful at with minimal frustration once you’ve seen basic ideas in class. My goal is that you can repeat the patterns and processes we use in class outside of class to complete assignments, while acknowledging that you will definitely have to look things up and read documentation outside of class.

Each class will open with some time to review what was covered in the last session before adding new material.

To get the most out of class sessions, you should have a laptop with you. During class you should be following along with Dr. Brown, typing and running the same code. You’ll answer questions on Prismia chat, when you do so, you should try running necessary code to answer those questions. If you encounter errors, share them via prismia chat so that we can see and help you.

After class

After class, you should practice with the concepts introduced.
This means reviewing the notes: both yours from class and the annotated notes posted to the course website.
When you review the notes, you should be adding comments on tricky aspects of the code and narrative text between code blocks in markdown cells. While you review your notes and the annotated course notes, you should also read the documentation for new modules, libraries, or functions introduced that day.

In the annotated notes, there will often be extra questions or ideas on how to extend and practice the concepts. Try these out.

If you find anything hard to understand or unclear, write it down to bring to class the next day.

Assignments

In assignments, you will be asked to practice with specific concepts at an intermediate level. Assignments will apply the concepts from class with minimal extensions. You will probably need to use help funcitons and read documentation to complete assignments, but mostly to look up things you saw in class and make minor variations. Most of what you need for assignments will be in the class notes, which is another reason to read them after class.

Portfolios

In portfolios, your goal is to extend and apply the concepts taught in class and practiced in assignments to solve more realistic problms. You may also reflect on your learning in order to demonstrate deep understanding. These will require significant reading beyond what we cover in class.

Getting Help with Programming

Asking Questions

One of my favorite resources that describes how to ask good questions is this blog post by Julia Evans, a developer who writes comics about the things she learns in the course of her work and publisher of wizard zines.

Describing what you have so far

Stackoverflow is a common place for programmers to post and answer questions.
As such, they have written a good guide on creating a minimal, reproducible example.

Creating a minimal reproducible example may even help you debug your own code, but if it does not, it will definitely make it easier for another person to understand what you have, what your goal is, and what's working.

> **Note**
>
> A fun version of this is rubber duck debugging

Understanding Errors

Error messages from the compiler are not always straight forward.

The TraceBack can be a really long list of errors that seem like they are not even from your code. It will trace back to all of the places that the error occurred. It is often about how you called the functions from a library, but the compiler cannot tell that.

To understand what the traceback is, how to read one, and common examples, see this post on Real Python.

One thing to try, is friendly traceback a python package that is designed to make that error message text more clear and help you figure out what to do next.

> **Ram Token Opportunity**
>
> If you try out friendly traceback and find it helpful, add a testimonial here. using
>
> ```
> ```{epigraph}
> ```
> ```

Terminals and Environments

Why all this work?

Managing environments is **one of the hardest parts of programming** so, as instructors, we often design our courses around not having to do it. In this class, however, I'm choosing to take the risk and help you all through beginning to manage your own environments.

These issues will be the most painful in the course, I promise.

I think it's worth this type of pain though, because all fo the code you ever run must run in *some* sort of environment. By giving you control, I'm hoping to increase your indepence as a programmer. This also means responsibility and some messy debugging, but I think this is a good tradeoff. This is an upper level (300+) level course, so increasing some complexity is expected and I want as much as possible to keep you close to realisitc programming environments; so that what you see in this course is **directly, and immediately,** applicable in real world contexts. You should be able to pick up data science side projects or an internship with ease after this course.

I *know* some of these things will be frustrating at times, but I want you to feel supported in that and know that your grade will not be blocked by you having environment issues, as long as you ask for help in a timely matter.

Windows

Windows has a sort of multiverse of terminal environments.

The least setup required involves using anaconda prompt and `conda` to manage you python environment and GitBash to work with git (and it can also do other bash related things).

Instead of managing two terminals, you may configure your path in GitBash to make Anaconda work

MacOS

MacOS has one terminal app, but it can run different shells.

On MacOS You may want to switch to bash (using the `bash` command or make it your default and update bash.

Getting Organized for class

The only **required** things are in the Tools section of the syllabus, but this organizational structure will help keep you on top of what is going on.

Your username will be appended to the end of of the repository name for each of your assignments in class.

File structure

I recommend the following organization structure for the course:

```
CSC310
  |- notes
  |- portfolio-username
  |- 02-accessing-data-username
  |- ...
```

This is one top level folder will all materials in it. A folder inside that for in class notes, and one folder per repository.

Please **do not** include all of your notes or your other assignments all inside your portflio, it will make it harder to grade.

Finding repositories on github

Each assignment repository will be created on GitHub with the `rhodyprog4ds` organization as the owner, not your personal acount. Since your account is not the owner, they do not show on your profile.

Your assignment repositories are all private during the semester. At the end, you may take ownership of your portfolio[^pttrans] if you would like.

If you go to the main page of the organization you can search by your username (or the first few characters of it) and see only your repositories.

> **Warning**
>
> Don't try to work on a repository that does not end in your username; those are the template repositories for the course and you don't have edit permission on them.

Letters to Future students

This section is a place for students enrolled in Fall 2020 to write letters to future students taking this class with Professor Brown. The websites for future sections will link back here for them to read.

Contributed Notes

Reviewing notes

Especially when it comes to the difficult topics make sure you go back through the notes and try and add to them yourself. Actively using the new topics will help you learn a lot better than just copying the notes.

Attend Office Hours

Attending office hours will help you better understand course material, complete homework assignments, and learn new things that might not normally come up in class.

- David

To contribute

Via GitHub directly:

1. Use the edit button above to add a note to this file following the example that's commented out
2. create a pull request

> **Note**
>
> We know that we don't currently teach a lot of this in our department, so in Spring 22 I'm teaching a brand new course on Computer Systems, that will help you understand the underlying concepts that make all of this stuff make sense, instead of just following recipes and debugging here and there.

If, for example, you come to me in week 5 and have never got an any environment working and you're trying for the first time, your grade will be hurt because you will be very far behind at that point. Ask for help early and often.