

About this Book

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

This class meets MWF 3-3:50pm in Chafee Social Sci Center 235.

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.



Notes will have exercises marked like this



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



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.



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.

Ram Token Opportunity

Chances to earn ram tokens are highlighted this way.

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

Basic Facts

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 syllabus

This syllabus is a *living* document and accessible from BrightSpace, as a pdf for download directly online at rhodyprog4ds.github.io/BrownFall21/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 You can view the date of changes and exactly what changes were made on the Github [commit history](#) page.

Creating an [issue](#) 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 M Brown Office hours: TBA via zoom, link on BrightSpace

Dr. Sarah M Brown is a second year 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. You can learn more about me at my [website](#) or my research on my [lab site](#).

You can call me Professor Brown or Dr. Brown, I use she/her pronouns.

The best way to contact me is e-mail or an issue on an assignment repo. For more details, see the [Communication Section](#)

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.

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.

Course Manual

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/BrownFall21/](#). Links to the course reference text and code documentation will also be included here in the assignments and class notes.

GitHub Classroom

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)

Important

TL;DR [\[1\]](#)

- check Brightspace
- Log in to Prismia Chat
- Make a GitHub Account
- Install Python
- Install Git

Note

Seeing the BrightSpace site requires logging in with your URI SSO and being enrolled in the course

- [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.

ℹ️ 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

Textbook

The text for this class is a reference book and will not be a source of assignments. It will be a helpful reference and you may be directed there for answers to questions or alternate explanations of 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.

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 multiple 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 3-3:50pm in Chafee Social Sci Center 235. 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.

Programming assignments that will be due each week Tuesday by 11:59pm.

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, Naive Bayes, classification performance metrics	[classification, evaluate]
7	decision trees, cross validation	[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]

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.

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.

	skill	Level 1	Level 2	Level 3
keyword				
python	pythonic code writing	python code that mostly runs, occasional pep8 adherence	python code that reliably runs, frequent pep8 adherence	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 contrast 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 standard 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
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 regularized or nonlinear regression
clustering	Clustering	describe what clustering is	apply basic clustering	apply multiple clustering techniques, and interpret results
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

	skill	Level 1	Level 2	Level 3
keyword				
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 quantitateve 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-structured, open-ended problems, apply common structure to learn new features of standard tools	Independently scope and solve realistic data science problems OR independently learn reletaed 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.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	# Assignm
keyword														
python	1	1	0	1	1	0	0	0	0	0	0	0	0	0
process	1	1	0	0	0	0	1	1	1	1	1	0	0	0
access	0	1	1	1	1	0	0	0	0	0	0	0	0	0
construct	0	0	0	0	1	0	1	1	0	0	0	0	0	0
summarize	0	0	1	1	1	1	1	1	1	1	1	1	1	1
visualize	0	0	1	1	0	1	1	1	1	1	1	1	1	1
prepare	0	0	0	1	1	0	0	0	0	0	0	0	0	0
classification	0	0	0	0	0	0	1	1	0	0	1	0	0	0
regression	0	0	0	0	0	0	0	0	1	0	0	1	0	0
clustering	0	0	0	0	0	0	0	0	0	1	0	1	0	0
evaluate	0	0	0	0	0	0	0	1	1	0	1	1	0	0
optimize	0	0	0	0	0	0	0	0	0	0	1	1	0	0
compare	0	0	0	0	0	0	0	0	0	0	0	1	0	1
representation	0	0	0	0	0	0	0	0	0	0	0	1	1	1
workflow	0	0	0	0	0	0	0	0	0	1	1	1	1	1

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.

	Level 3	P1	P2	P3	P4
keyword					
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
classification	fit and apply classification models and select appropriate classification models for different contexts	0	1	1	1
regression	fit and explain regularized or nonlinear regression	0	1	1	1
clustering	apply multiple clustering techniques, and interpret results	0	1	1	1
evaluate	Evaluate a model with multiple metrics and cross validation	0	1	1	1
optimize	Select optimal parameters based of mutiple quantitative 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 releted tools and describe strengths and weaknesses of common tools	0	0	1	1

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 discuss 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. You can also earn level 1 achievements from adding annotation to a section of the class notes.

Assignments

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

Assignments will each evaluate certain skills. After your assignment is reviewed, you will get qualitative feedback on your work, and an assessment of your demonstration of the targeted skills.

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 achieved mastery of all of the skills 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

On Brightspace there are 45 Grade items that you will get a 0 or a 1 grade for. These will be revealed, so that you can view them as you have an opportunity to demonstrate each one. The table below shows the minimum number of skills at each level to earn each letter grade.

Level 3 Level 2 Level 1

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.

The letter grade can be computed as follows

⚠ 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.

ℹ Note

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

```

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

```

For example you can run the code like this in a cell to see the output

```
compute_grade(15,15,15)
```

```
'A'
```

```
compute_grade(14,14,14)
```

```
'C-'
```

Or use `assert` to test it formally

```
assert compute_grade(14,14,14) == 'C-'
```

```
assert compute_grade(15,15,15) == 'A'
```

```
assert compute_grade(15,15,11) == 'A-'
```

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.

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.

Grading Examples

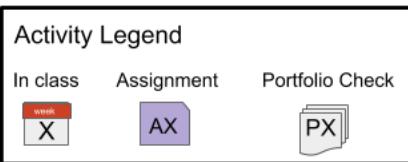
If you always attend and get everything correct, you will earn an A and you won't need to submit the 4th portfolio check or assignment 13.

Getting an A Without Perfection

Map to an A

How Achievements were earned

	Level 1	Level 2	Level 3
python	A1	A3	P1
process	A1	P1	P2
access	2 week	A2	P1
construct	5 week	A5	P1
summarize	3 week	A3	P1
visualize	3 week	A3	P2
prepare	4 week	A5	P2
classification	A10	P2	P3
regression	8 week	A11	P2
clustering	9 week	A9	P3
evaluate	7 week	A11	P3
optimize	10 week	A11	P4
compare	11 week	A13	P3
unstructured	12 week	A13	P4
tools	11 week	A13	P3



Other Activities

	Attended, but did not understand
	Submitted, but incorrect
	Missed class
	Not submitted
	Submitted, but incorrect
	Not submitted
	Not submitted
	Attended, but all level 1 complete
	Attended, but all level 1 complete

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

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.

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

Map to a B easily

	Level 1	Level 2	Level 3
python	week 1	A3	
process	week 1	A1	
access	2	A2	
construct	week 5	A5	
summarize	week 3	A3	
visualize	3	A3	
prepare	week 4	A4	
classification	week 10	A6	
regression	week 8	A11	
clustering	week 9	A9	
evaluate	week 7	A10	
optimize	week 10	A10	
compare	week 11	A11	
unstructured	week 12	A12	
tools	week 11	A12	



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

Map to a B, having trouble

	Level 1	Level 2	Level 3
python	A1	P1	
process	A1	P2	P1
access	A2	P1	
construct	A5	P1	
summarize	A3	P1	
visualize	A3	P2	
prepare	A5	P2	
classification	A10	P3	
regression	A11	P2	
clustering	A9	P3	
evaluate	A11	P3	
optimize	A11	P4	
compare	A13	P3	
unstructured	A13	P4	
tools	A13	P3	



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.

Ram Tokens

Ram Tokens in this course will be used as a currency for extra effort. You can earn Ram Tokens by doing work that supports your learning or class activities, but do not directly demonstrate achievements. You can spend Ram Tokens to get extra grading. This will be mostly applicable to Portfolio Checks. In Checks 3 & 4, some achievements will not be eligible for grading as per the [table](#). However, you can exchange Ram Tokens to make more achievements eligible for assessment. This system rewards you for putting in consistent effort, even if it takes you many tries to understand a concept.

To accumulate Ram Tokens, you submit a 'Deposit' to the [Ram Token Bank: http://drsmb.co/ramtoken](http://drsmb.co/ramtoken) with a link to what you did to earn a token. To apply Ram tokens for extra grading, submit the same form, with a link to the assignment and add the Ramtoken label to the Feedback PR.

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.

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 specific comment.

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.

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 requesting and tag @brownsarahm

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

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.

Course Communications

Help Hours

Day	Time	Location	Host
Monday	9:30:00 AM-10:30 AM	inperson Tyler Hall 140	Chamudi
Monday	12:30:00 PM-2:00 PM	inperson Tyler Hall 139	Chamudi
Tuesday	2:00 PM-3:00 PM	gather.town	Sarah
Wednesday	4:00:00 PM-5:00 PM	inperson Tyler Hall 139	Chamudi
Wednesday	1:30:00 PM-3:00 PM	inperson Tyler Hall 140	Chamudi
Wednesday	7:00:00 PM-8:30	gather.town	Sarah
By appointment	scheduling link on Brightspace	in person Tyler 134	Sarah

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

To reach out, By usage

usage	platform	area	note
in class	prismia	chat	outside of class time this is not monitored closely
any time	prismia	message board	for discussion with peers
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	github	pull request on website	remember to request ram tokens if applicable
matters that don't fit into another category	e-mail	to brownsarahn@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 (Professor + TA) to collaborate on who responds to things more easily.

By Platform

Use e-mail for

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

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	pull request on website	remember to request ram tokens if applicable

Use prismia for

usage	area	note
in class	chat	outside of class time this is not monitored closely
any time	message board	for discussion with peers
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-mail

- 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

Today's goals:

1. Operate tools for in-class participation
 2. Understand what Data Science is, in broad terms
 3. Understand the syllabus (grading, topics covered, schedule, etc)
 4. Understand how to learn in this course
-

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. What is Data Science

In general:



statistics is the type of math we use to make sense of data. Formally, a statistic is just a function of data.

computer science is so that we can manipulate visualize and automate the inferences we make.

domain expertise helps us have the intuition to know if what we did worked right. A statistic must be interpreted in context; the relevant context determines what they mean and which are valid. The context will say whether automating something is safe or not, it can help us tell whether our code actually worked right or not.



We'll focus on the programming as our main means of studying data science, but we will use bits of the other parts. In particular, you're encouraged to choose datasets that you have domain expertise about, or that you want to learn about.

But there are many definitions. We'll use this one, but you may come across others.

1.2.1. How does data science happen?



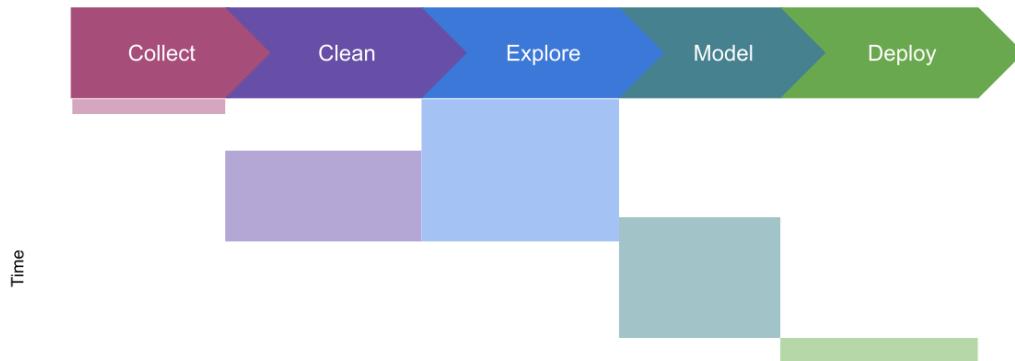
1.2.2. how we'll cover it, in depth



- *collect*: Discuss only a little; Minimal programming involved
- *clean*: Cover the main programming techniques; Some requires domain knowledge beyond scope of course
- *explore*: Cover the main programming techniques; Some requires domain knowledge beyond scope of course

- *model*: Cover the main programming, basic idea of models; How to use models, not how learning algorithms work
- *deploy*: A little bit at the end, but a lot of preparation for decision making around deployment

1.2.2.1. how we'll cover it in time



We'll cover exploratory data analysis before cleaning because those tools will help us check how we've cleaned the data.

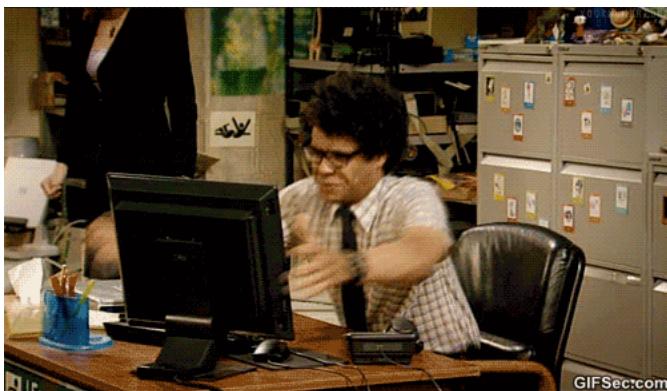
1.3. How this class will work

- today is an exception
- in general we'll be live coding

Let's look at the [syllabus](#)

Read carefully to make sure you understand the grading; it's not typical points and an average.

Class is designed to avoid this:



1.4.

1.5. Learning Cycle

 **Julia Evans**  
@b0rk

we think about debugging as a technical skill (and it absolutely is!!) but a huge amount of it is managing your feelings so you don't get discouraged and being self-aware so you can recognize your incorrect assumptions

5:35 PM · Jun 11, 2021 

 4.3K  Reply  Copy link to Tweet

[Read 95 replies](#)

Read more about how I'm designing this course to help you learn on the [how to learn](#) page.

1.6. Check your understanding of the syllabus

It's easy when reading something long to lose track of it. Your eyes can go over each word, without actually retaining the information, but it's important to understand the syllabus for the course.

You can find the answers to the following questions on the syllabus. If you've already read it, try answering them to check your understanding. If you haven't read it yet, use these to guide you to get familiar with finding key facts about the course on the syllabus.

1. What do you need to bring to class each day?
2. What is the basis of grading for this course?
3. How do you reference the course text?
4. What is the penalty for missing an assignment?

More information about the course is available throughout the site, the next few questions will help you self-check that you've found the important things. Remember, the goal is not necessarily to memorize all of this, but to be able to find it.

1. When & what are you expected to read for this class?

- [] read the text book before class
- [] review notes & documentation after class
- [] preview the notes & documentation before class
- [] read documentation and text book after class

1. Your assignment says to find a dataset that has variables of a specific type, which website can you use?
2. Your assignment says to find a dataset of any type about something you're interested in, which resource would you use?

2. Jupyter Notebook Tour & Python Review

2.1. A jupyter notebook tour

Launch a [jupyter notebook](#):

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

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

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

2.1.1. 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'
```

Note

For a little more python review, see my [2020 CSC310 notes](#) this is just enough for this assignment.

2.2. Just enough Git for Assignment 1

2.2.1. Assignment 1:

Goals for this assignment

- setup your portfolio
- check that you understand the grading
- review Python basics
- practice with git and GitHub

2.2.2. Why Version control

We often want to keep track of the different versions in case we want to go back, but this can be painful:

"FINAL".doc



FINAL.doc!



FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.CORRECTIONS.doc



FINAL_rev.18.comments7.corrections9.MORE.30.doc

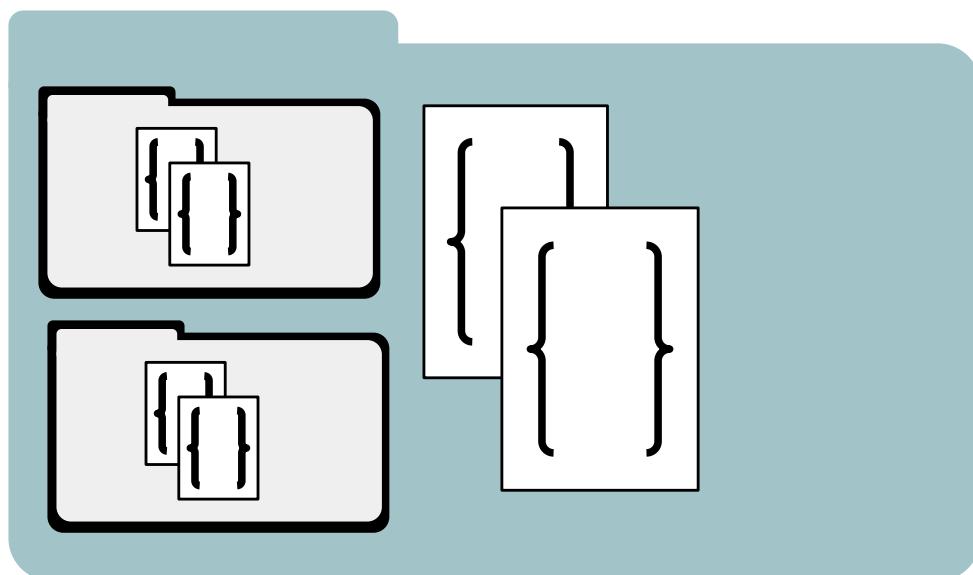


FINAL_rev.22.comments49.corrections.10.#@\$%WHYDID
ICOMETOGRAD SCHOOL????.doc



WWW.PHDCOMICS.COM

We typically organize projects in folder



A [repository](#) is a folder with a hidden directory named `.git`



The `git` application manages that hidden directory, we don't write to it directly, which is why we keep it hidden.

Git is a distributed system, you have a local version and a remote version.



Once a repository exists on GitHub, we get a local copy by cloning it after we get its address from the GitHub interface, by clicking on the green code button that is below the menu area to the right. It's at the top right corner of the list of files in the repository.

[rhodyprog4ds / portfolio-brownsarahm](#) Private
generated from [rhodyprog4ds/portfolio](#)

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

feedback had recent pushes 1 minute ago Compare & pull request

main 5 branches 1 tag Go to file Add file Code

brownsarahm update toc to include notebook

.github	correct path for jupytext conversion
about	mvoe notebook
template_files	convert notebooks to md
.gitignore	merge gh changes and ignore
README.md	Initial commit

Clone with HTTPS Use SSH
Use Git or checkout with SVN using the web URL.
<https://github.com/rhodyprog4ds/por> Open with GitHub Desktop Download ZIP

For this part, use GitBash on windows or terminal otherwise: If you set up a Personal Access Token you can use the https version

After `cd/to/where/you/want/your/repo/locally:`

If you set up ssh keys you use that instead

Once it's cloned, then you can navigate into the new folder:

Then you can change files, for example adding to the intro.

Some common actions in Git, you'll want.

Check on the status of your repository:

Add files to the staging area:

Add all changes to the staging area:

Commit your changes to the repository:

git commit -m 'a message that will help your future self know what this part is'

Note

These notes can be downloaded as an actual notebook, click the GitHub logo at the top of the page and choose .ipynb. The following is not runnabel in the notebook as is.

Push your changes to GitHub

```
git push
```

Pull changes from GitHub

```
git pull
```

You can also go through these same basic steps: add, commit, push

2.3. More on git

- [GitHub Hello World](#)
- [Software Carpentry Git Novice Lesson](#)

Also, in Spring 2022, I'm teaching a section of CSC392: Topics in Computing, Introduction to Computer Systems, that will cover tools of the trade (git, bash, etc) and how they all work in great detail.

2.4. More on Python

Read [Pep 8](#) to see what good style in Python is.

3. Getting help, object inspection, loading data

3.1. First, Don't Worry members

Class Response Summary:

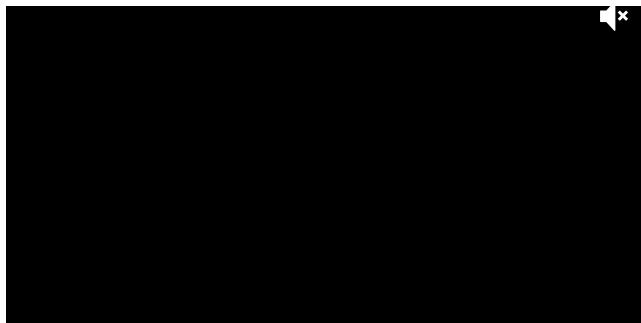


EVERYBODY CALM DOWN

I GOT THIS

quickmeme.com

[funny CS memes](#)





USES CLASSIC MEME FORMAT



PEOPLE LIKE IT



3.2. Getting Help in Jupyter

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 use jupyter popup help with shift +tab or ?

```
print?
```

Or the base python `help` function

```
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.
```

Notice that function can take multiple arguments and has a keyword argument (must be used like `argument=value`) described as `sep=' '`. This means that by default it adds a space

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

```
Sarah Brown
```

But we can change the separator.

```
Sarah_Brown
```

Note that it also defaults to end to use \n

```
print(first,last)
print('hello')
```

```
Sarah Brown
hello
```

Where does this help information come from?

```
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
```

 Note

You can copy code from the notes, try hovering over this

We can apply `help` on the function we wrote

```
help(compute_grade)
```

```

Help on function compute_grade in module __main__:

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 (+/-)

```

It gets the docstring

3.3. Everything is an Object in Python

we can use the builtin function `type` to inspect them, and get attributes with `.`

```
type(compute_grade)
```

```
function
```

```
compute_grade.__name__
```

```
'compute_grade'
```

```
c = 4.5
```

```
type(c)
```

```
float
```

```
c= 'hello'
```

```
type(c)
```

```
str
```

When do we use single vs double quotes?

- You can use either, unless you need to put one inside the string then use the other.

```
my_sentence = "The professor's name is Dr. Brown"
```

```
my_sentence = 'The professor's name is Dr. Brown'
```

```

File "/tmp/ipykernel_1812/607286316.py", line 1
    my_sentence = 'The professor's name is Dr. Brown'
                                         ^
SyntaxError: invalid syntax

```

Yes we can escape special characters:

```
my_sentence = 'The professor\'s name is Dr. Brown'
```

but, it's less readable and not recommended.

3.4. Good Code is always relative

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](#)

3.5. Coffee Data

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.

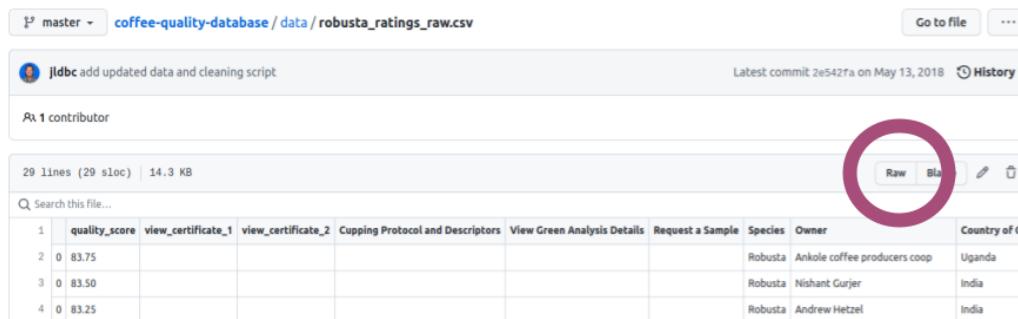
what format is it provided in?

- csv (Comma Separated Values)

what other information is in this repository?

- the code to scrape

Get raw url for the dataset click on the raw button on the [csv page](#), then copy the url.



1	quality_score	view_certificate_1	view_certificate_2	Cupping Protocol and Descriptors	View Green Analysis Details	Request a Sample	Species	Owner	Country of Origin
2	83.75						Robusta	Ankole coffee producers coop	Uganda
3	83.50						Robusta	Nishant Gurjer	India
4	83.25						Robusta	Andrew Hetzel	India

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

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

We will use a library called Pandas

```
import pandas as pd  
# import library and give it an alias (nickname) pd
```

```
pd.read_csv(data_url)
```

Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number		
0	1 Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	prc	
1	2 Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethi	
2	3 Robusta	andrew hetzel	India	sethuraman estate	NaN		
3	4 Robusta	ugacof	Uganda	ugacof project area	NaN		
4	5 Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	devel	
5	6 Robusta	andrew hetzel	India	NaN	NaN		
6	7 Robusta	andrew hetzel	India	sethuraman estates	NaN		
7	8 Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethi	
8	9 Robusta	nishant gurjer	India	sethuraman estate	RKR	sethi	
9	10 Robusta	ugacof	Uganda	ishaka	NaN	n	
10	11 Robusta	ugacof	Uganda	ugacof project area	NaN		
11	12 Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethi	
12	13 Robusta	andrew hetzel	India	sethuraman estates	NaN		
13	14 Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN		
14	15 Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	prc coo	
15	16 Robusta	andrew hetzel	India	sethuraman estate	NaN		
16	17 Robusta	andrew hetzel	India	sethuraman estates	NaN	sethi	
17	18 Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	ka	
18	19 Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nit	
19	20 Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	r	
20	21 Robusta	andrew hetzel	India	sethuraman estates	NaN		

Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number		
21	22 Robusta	andrew hetzel	India	sethuraman estates	NaN	sethi	
22	23 Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethi	
23	24 Robusta	luis robles	Ecuador	robustasa	Lavado 1	our c	
24	25 Robusta	luis robles	Ecuador	robustasa	Lavado 3	lab	
25	26 Robusta	james moore	United States	fazenda cazengo	NaN	c	
26	27 Robusta	cafe politico	India		NaN		NaN
27	28 Robusta	cafe politico	Vietnam		NaN		NaN

28 rows × 44 columns

💡 Try it yourself

Read the data in again, but with the index correct and save it to a variable.

Once we read it in, we can view the first 5 rows with the `head` method.

```
coffee_df.head()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	IC
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/11
3	Robusta	andrew hetzel	India	sethuraman estate	NaN		NaN
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	

5 rows × 43 columns

❗ Important

Remember to comment & annotate your code

3.6. Follow Up questions

3.6.1. General Questions

How do you create code to scrape data from a website and compile it into a csv file?



Will we be using pandas a lot during the semester?



3.6.2. Clarifying

How do you auto finish your directories



How do you properly shut down Jupyter Notebook



Is pd some sort of variable we set or was it built in?



How should I be organized for this class? Keep it all in a single folder? Keep it on GitHub?



I'm still not sure how to keep everything together in a portfolio for the semester?



I am still wondering if I am using anaconda or just normal terminal



Can I push this code into my portfolio using the anaconda terminal



3.6.3. Grading Questions

How do we keep track of which achievements we've earned?



I don't really have many questions from today, but I was wondering if office hours were posted.



Will we always submit homework through the portfolio folder in github?



I'm just confused as how to view my feedback from the assignment



3.6.4. Questions we'll answer later this week

- does each column have a number assigned to it in data frames?
- Can other data types be imported into a notebook and edited the same way as .csv files?

3.7. More Practice

- How could you check if `pd` is built in or if we defined it?
- If we wanted to see more than 5 rows when printing the head of the dataset how would we do so?

Ram Token Opportunity

Contribute possible practice questions to the notes using the suggest an edit button behind the GitHub menu at the top of the page.

4. Pandas DataFrames

Today, we're going to explore [DataFrames](#)s in greater detail. We'll continue using that same coffee dataset.

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

4.1. More about loading libraries

We can import pandas without the alias `pd` if we want, but then we have to use the full name everywhere

```
import pandas
```

```
pandas.read_csv(coffee_data_url)
```

Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number		
0	1 Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	prc	
1	2 Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethi	
2	3 Robusta	andrew hetzel	India	sethuraman estate	NaN		
3	4 Robusta	ugacof	Uganda	ugacof project area	NaN		
4	5 Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	devel	
5	6 Robusta	andrew hetzel	India	NaN	NaN		
6	7 Robusta	andrew hetzel	India	sethuraman estates	NaN		
7	8 Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethi	
8	9 Robusta	nishant gurjer	India	sethuraman estate	RKR	sethi	
9	10 Robusta	ugacof	Uganda	ishaka	NaN	n	
10	11 Robusta	ugacof	Uganda	ugacof project area	NaN		
11	12 Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethi	
12	13 Robusta	andrew hetzel	India	sethuraman estates	NaN		
13	14 Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN		
14	15 Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	prc coo	
15	16 Robusta	andrew hetzel	India	sethuraman estate	NaN		
16	17 Robusta	andrew hetzel	India	sethuraman estates	NaN	sethi	
17	18 Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	ka	
18	19 Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nit	
19	20 Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	r	
20	21 Robusta	andrew hetzel	India	sethuraman estates	NaN		

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number		
21	22 Robusta	andrew hetzel	India	sethuraman estates	NaN	sethi	
22	23 Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethi	
23	24 Robusta	luis robles	Ecuador	robustasa	Lavado 1	our c	
24	25 Robusta	luis robles	Ecuador	robustasa	Lavado 3	lab	
25	26 Robusta	james moore	United States	fazenda cazengo	NaN	c	
26	27 Robusta	cafe politico	India		NaN	NaN	
27	28 Robusta	cafe politico	Vietnam		NaN	NaN	

28 rows × 44 columns

We'll use `pd` because that's the more common convention and so that we can type fewer characters throughout our code

```
import pandas as pd
```

4.2. Examining DataFrames

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

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

```
df.head()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	IC
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/11
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN	
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof	
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust	

5 rows × 43 columns

Using `help`, we can see that that `head` takes one parameter and has a default value of 5, which is why we got 5 rows, but we can get 2 instead

```
df.head(2)
```

	Species	Owner	Country.of-Origin	Farm.Name	Lot.Number	Mill	ICO.I
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers	
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate	14/1148/

2 rows × 43 columns

We can look at the last rows with `tail`

```
df.tail(3)
```

	Species	Owner	Country.of-Origin	Farm.Name	Lot.Number	Mill	ICO.Number
26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo	NaN
27	Robusta	cafe politico	India	NaN	NaN	NaN	14-1118-2014-0087
28	Robusta	cafe politico	Vietnam	NaN	NaN	NaN	NaN

3 rows × 43 columns

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

```
type(df)
```

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

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

```
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')
```

These are a special type called Index

```
type(df.columns)
```

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

It also has an index

```
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')
```

and values

```
df.values
```

```
array([['Robusta', 'ankole coffee producers coop', 'Uganda', ..., 1488.0,
       1488.0, 1488.0],
      ['Robusta', 'nishant gurjer', 'India', ..., 3170.0, 3170.0,
       3170.0],
      ['Robusta', 'andrew hetzel', 'India', ..., 1000.0, 1000.0, 1000.0],
      ...,
      ['Robusta', 'james moore', 'United States', ..., 795.0, 795.0,
       795.0],
      ['Robusta', 'cafe politico', 'India', ..., nan, nan, nan],
      ['Robusta', 'cafe politico', 'Vietnam', ..., nan, nan, nan]],
      dtype=object)
```

it also knows its own shape

```
df.shape
```

```
(28, 43)
```

we can use builtin functions on our DataFrame too not just its own methods and attributes.

```
len(df)
```

```
28
```

Why does `len` turn green? it's a python reserve word

4.3. Building a Data Frame programmatically

One way to build a data frame is from a dictionary:

```
people = {'names': ['Sarah', 'Connor', 'Kenza'],
          'username': ['brownsarahm', 'sudoPsych', 'kbdlh']}
```

```
people
```

```
{'names': ['Sarah', 'Connor', 'Kenza'],
 'username': ['brownsarahm', 'sudoPsych', 'kbdlh']}
```

```
type(people)
```

```
dict
```

```
people_df = pd.DataFrame(people)
people_df
```

	names	username
0	Sarah	brownsarahm
1	Connor	sudoPsych
2	Kenza	kbdlh

```
type(people['names'])
```

```
list
```

```
type(people)
```

```
dict
```

```
type({4,5,5})
```

```
set
```

```
{4,5,5}
```

```
{4, 5}
```

```
people['names']
```

```
['Sarah', 'Connor', 'Kenza']
```

```
type(set(people['names']))
```

```
set
```

```
unique_people = set(people['names'])  
type(unique_people)
```

```
set
```

```
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')
```

```
for col in df.columns:  
    print(col.split('.'))
```

```
['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']
```

```
for key,value in people.items():
    print(key,':',value)
```

```
names : ['Sarah', 'Connor', 'Kenza']
username : ['brownsarahm', 'sudoPsych', 'kbdlh']
```

```
df['Owner']
```

```
1      ankole coffee producers coop
2          nishant gurjer
3          andrew hetzel
4          ugacof
5      katuka development trust ltd
6          andrew hetzel
7          andrew hetzel
8          nishant gurjer
9          nishant gurjer
10         ugacof
11         ugacof
12         nishant gurjer
13         andrew hetzel
14  kasozi coffee farmers association
15      ankole coffee producers coop
16          andrew hetzel
17          andrew hetzel
18          kawacom uganda ltd
19          nitubaasa ltd
20  mannya coffee project
21          andrew hetzel
22          andrew hetzel
23          andrew hetzel
24          luis robles
25          luis robles
26          james moore
27          cafe politico
28          cafe politico
Name: Owner, dtype: object
```

```
df.Owner
```

```
1      ankole coffee producers coop
2          nishant gurjer
3          andrew hetzel
4          ugacof
5      katuka development trust ltd
6          andrew hetzel
7          andrew hetzel
8          nishant gurjer
9          nishant gurjer
10         ugacof
11         ugacof
12         nishant gurjer
13         andrew hetzel
14  kasozi coffee farmers association
15  ankole coffee producers coop
16          andrew hetzel
17          andrew hetzel
18          kawacom uganda ltd
19          nitubaasa ltd
20  mannya coffee project
21          andrew hetzel
22          andrew hetzel
23          andrew hetzel
24          luis robles
25          luis robles
26          james moore
27          cafe politico
28          cafe politico
```

```
Name: Owner, dtype: object
```

```
df
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill
1	Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	ankole coffee producers
2	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethuraman estate 14/1
3	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN
4	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof
5	Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	katuka development trust
6	Robusta	andrew hetzel	India	NaN	NaN	(self)
7	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN
8	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	7	sethuraman estate 14/1
9	Robusta	nishant gurjer	India	sethuraman estate	RKR	sethuraman estate 14/1
10	Robusta	ugacof	Uganda	ishaka	NaN	nsubuga umar
11	Robusta	ugacof	Uganda	ugacof project area	NaN	ugacof
12	Robusta	nishant gurjer	India	sethuraman estate kaapi royale	RC AB	sethuraman estate 14/1
13	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN
14	Robusta	kasozi coffee farmers association	Uganda	kasozi coffee farmers	NaN	NaN
15	Robusta	ankole coffee producers coop	Uganda	kyangundu coop society	NaN	ankole coffee producers coop union ltd
16	Robusta	andrew hetzel	India	sethuraman estate	NaN	NaN
17	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates
18	Robusta	kawacom uganda ltd	Uganda	bushenyi	NaN	kawacom
19	Robusta	nitubaasa ltd	Uganda	kigezi coffee farmers association	NaN	nitubaasa
20	Robusta	mannya coffee project	Uganda	mannya coffee project	NaN	mannya coffee project
21	Robusta	andrew hetzel	India	sethuraman estates	NaN	NaN

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill
22	Robusta	andrew hetzel	India	sethuraman estates	NaN	sethuraman estates
23	Robusta	andrew hetzel	United States	sethuraman estates	NaN	sethuraman estates
24	Robusta	luis robles	Ecuador	robustasa	Lavado 1	our own lab
25	Robusta	luis robles	Ecuador	robustasa	Lavado 3	own laboratory
26	Robusta	james moore	United States	fazenda cazengo	NaN	cafe cazengo
27	Robusta	cafe politico	India		NaN	NaN
28	Robusta	cafe politico	Vietnam		NaN	NaN

28 rows × 43 columns

Key points:

write three things to remember from today's class

4.4. Questions After Classroom

many overlapping questions today

4.5. General

How to know which function to use in certain problems or situations



4.6. Clarifying

Is there a way to have a set show the duplicates that get discarded?



being able to access the code somewhere without asking to scroll would be nice



4.7. Course Admin

When will homeworks be posted/due typically?



4.8. Questions we'll answer later

can you use cast a pandas dataframe into a set?



4.9. Try it yourself

- Create variables of three different types with facts about yourself. Use descriptive variable names relative to the contents, not their types.

- Create a list, again with a descriptive name, and print out the types

```
<class 'str'>
<class 'int'>
<class 'list'>
```

- Write a function, `type_extractor` that takes a list and a type and returns the item of that type from the list
- Test your function on all three items from your dictionary.
- Use one type of jupyter help on your function, what does it display? If it doesn't display anything modify your function so that help will work.
- Make yourself notes in the most memorable way for you about what a DataFrame is.

Ram Token Opportunity

Contribute possible practice questions to the notes using the suggest an edit button behind the GitHub menu at the top of the page.

5. More Loading Data, Indexing, and Iterables

As always, we'll start with loading pandas.

```
import pandas as pd
```

5.1. Checking in on help hours

if you missed class, check over the office hours schedule and e-mail if you can or cannot attend at least one time

5.2. Portfolio Preparation and Maintaintance

We'll spend a little time today getting your portfolio ready for the first check.

5.2.1. Access your portfolio

Go to your portflolio

- from the [course organization](#)
- from the list of your recent repositories on the left hand side of the [GitHub home page](#)

optionally, open it locally as well (we're going to update content and)

5.2.2. Start your Know, Want to Know, Learned Table

In each portfolio submission introduction, you'll reflect on what you've learned. To get ready for that, we'll first make note of what you already know and what you want to know.

1. edit `submission_1_intro` in your portfolio locally or on GitHub:
2. In the KWL section in the first two bullets after each skill with what you know and want to know. You can edit these in more detail later.

This will render as a table in your built portfolio, for reference on the syntax, [refer to the Tables section of the Jupyterbook Myst Markdown cheatsheet](#).

Warning

If you work on this in the GitHub website, be sure to pull these chances locally before you start working offline next

5.2.3. Merge the setup work

Once you're done, Go to your pull request tab, and select the feedback Pull Reques. Commit any suggestions if you'd like and then merge the PR.

⚠ Warning

only do this after grading

ℹ Note

To view the feedback, after merging the PR, remove `is:open` from the search bar on the PR page

5.3. Indexing

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

What will `topics[-1]` return?

```
topics[-1]
```

```
'pandas'
```

Using negative indices starts from the right. The last element is -1. The first is 0.

5.4. Reading DataFrames from Websites

We'll first read from the course website.

```
course_comms_url =
'https://rhodyprog4ds.github.io/BrownFall21/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, so lets see what it does:

ℹ 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.

```
pd.read_html(course_comms_url)
```

```
[  Day Time Location \
0 Monday 9:30:00 AM-10:30 AM inperson Tyler Hall 140
1 Monday 12:30:00 PM-2:00 PM inperson Tyler Hall 139
2 Tuesday 2:00 PM-3:00 PM gather.town
3 Wednesday 4:00:00 PM-5:00 PM inperson Tyler Hall 139
4 Wednesday 1:30:00 PM-3:00 PM inperson Tyler Hall 140
5 Wednesday 7:00:00 PM-8:30 PM gather.town
6 By appointment scheduling link on Brightspace in person Tyler 134

Host
0 Chamudi
1 Chamudi
2 Sarah
3 Chamudi
4 Chamudi
5 Sarah
6 Sarah ,
           usage platform \
0           in class prismia
1           any time prismia
2           any time prismia
3   private questions to your assignment github
4   for general questions that can help others github
5           to share resources github
6 matters that don't fit into another category e-mail

area                               note
0   chat outside of class time this is not monitored cl...
1   message board                   for discussion with peers
2   download transcript             use after class to get preliminary notes eg if...
3 issue on assignment repo        eg bugs in your code"
4 issue on course website         eg what the instructions of an assignment mean...
5 pull request on website         remember to request ram tokens if applicable
6 to brownsarahm@uri.edu         remember to include `[CSC310]` or `[DSP310]` (... ,
           usage area \
0 matters that don't fit into another category to brownsarahm@uri.edu

note
0 remember to include `[CSC310]` or `[DSP310]` (... ,
           usage area \
0   private questions to your assignment issue on assignment repo
1   for general questions that can help others issue on course website
2           to share resources pull request on website

note
0           eg bugs in your code"
1 eg what the instructions of an assignment mean...
2   remember to request ram tokens if applicable ,
     usage area \
0 in class chat
1 any time message board
2 any time download transcript

note
0 outside of class time this is not monitored cl...
1           for discussion with peers
2 use after class to get preliminary notes eg if... ]
```

It appears to have read all of them, lets check the type:

```
type(pd.read_html(course_comms_url))
```

```
list
```

Since we know it's a list, we'll save it to a variable that indicates that.

```
comms_list = pd.read_html(course_comms_url)
```

If we get just the first element,

```
type(comms_list[0])
```

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

it's a DataFrame and prints accordingly.

```
comms_list[0]
```

	Day	Time	Location	Host
0	Monday	9:30:00 AM-10:30 AM	inperson Tyler Hall 140	Chamudi
1	Monday	12:30:00 PM-2:00 PM	inperson Tyler Hall 139	Chamudi
2	Tuesday	2:00 PM-3:00 PM	gather.town	Sarah
3	Wednesday	4:00:00 PM-5:00 PM	inperson Tyler Hall 139	Chamudi
4	Wednesday	1:30:00 PM-3:00 PM	inperson Tyler Hall 140	Chamudi
5	Wednesday	7:00:00 PM-8:30	gather.town	Sarah
6	By appointment	scheduling link on Brightspace	in person Tyler 134	Sarah

Since it's a list, we can use base python's `len` function to check how many tables there are

```
len(comms_list)
```

```
5
```

We've seen the first table and know it's the help hours, so we can save that to a separate variable and use it

```
help_df = comms_list[0]
```

We've inspected the dataframe some before, but we can also check the type of each column.

```
help_df.dtypes
```

```
Day      object
Time     object
Location  object
Host      object
dtype: object
```

ⓘ Further Reading

You can read more about the [details of data types](#) in Pandas in the documentation

5.5. How are objects printed in jupyter?

ⓘ Question from class

Q: Why does it have `dtype:object` after the type for each row? A: the last line is information about the object that is being printed out.

To understand this, let's save the thing we're curious to a variable so we can examine it multiple ways more easily.

Note

this section is added, it didn't happen this way in class. This section, describes **how** I figured out the answer to the question about why that extra line is displayed.

```
help_df_types = help_df.dtypes
```

Next we'll check the type of this object and its shape

```
type(help_df_types)
```

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

a [Series](#) is like a DataFrame, but just one row with headings, and then rotated.

```
help_df_types.shape
```

```
(4,)
```

This means that it's length is 4 and it's a 1 dimensional object; the column headers have converted to an index and are treated as metadata, but not a part of the actual data.

So, the line we're interested in is not a part of the object, because it's length 4 and the thing we're curious about is the fifth line.

We'll pick one variable from the DataFrame and check its type

```
type(help_df['Day'])
```

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

This is also a Series, so let's check its output

```
help_df['Day']
```

```
0      Monday
1      Monday
2    Tuesday
3  Wednesday
4  Wednesday
5  Wednesday
6  By appointment
Name: Day, dtype: object
```

The last line of this one is information about the Series, its name, and its dtype.

Let's make another series, and see how it prints

```
pd.Series([5,4,5])
```

```
0    5
1    4
2    5
dtype: int64
```

The last line is the dtype of the Series; so in our original object, that last line is because the list of dtypes is the of type object.

```
help_df_types
```

```
Day      object
Time     object
Location    object
Host      object
dtype: object
```

5.6. How do we know what to check?

we examined the DataFrame so far by (me) knowing what to look for.

In python objects you can programmatically find what to look for with the `__dict__` attribute or we can rely on the [online documentation](#) or use it via help.

In ipython (what we use in jupyter, by default) we can use the `?` for help

```
pd.DataFrame?
```

```
help(pd.DataFrame)
```

💡 Everything is Data

writing good documentation lets people who use your code get help for free Not only do help tools use the docs, but that website is generated programmatically using a tool called Sphinx from the documentation inside the code. You can access the docstring of a python using the `.__doc__` attribute

⚠️ Caution

[ipython help](#) read about how it works, if it doesn't work for you to try to figure out why

```
Help on class DataFrame in module pandas.core.frame:  
  
class DataFrame(pandas.core.generic.NDFrame, pandas.core.arraylike.OpsMixin)  
| DataFrame(data=None, index: 'Axes | None' = None, columns: 'Axes | None' = None,  
dtype: 'Dtype | None' = None, copy: 'bool | None' = None)  
|  
| Two-dimensional, size-mutable, potentially heterogeneous tabular data.  
|  
| Data structure also contains labeled axes (rows and columns).  
| Arithmetic operations align on both row and column labels. Can be  
| thought of as a dict-like container for Series objects. The primary  
| pandas data structure.  
|  
| Parameters  
|-----  
| data : ndarray (structured or homogeneous), Iterable, dict, or DataFrame  
|     Dict can contain Series, arrays, constants, dataclass or list-like objects. If  
|     data is a dict, column order follows insertion-order.  
|  
| .. versionchanged:: 0.25.0  
|     If data is a list of dicts, column order follows insertion-order.
```

Try it yourself!

Make a list of the shape of all of the tables on the syllabus Achievements page.

```
achievements_url =  
'https://rhodyprog4ds.github.io/BrownFall21/syllabus/achievements.html'
```

```
[(14, 3), (15, 5), (15, 15), (15, 6)]
```

This solution uses a list comprehension which allows us to compress a loop. It's equivalent to the following with a for loop

```
[(14, 3), (15, 5), (15, 15), (15, 6)]
```

5.7. Lambdas and Dictionaries for switching

What if we want to print out the first column for the DataFrame if it has more than 3 columns and the whole thing if it has 3 or less columns?

Two ways of writing a function

Question from class

Python does have [ternary operators](#) but the dictionary is a more common way to achieve this and goal and more common patterns are better for readability

Further Reading

You can refer to the [official documentation](#) on [lambda](#) functions for a brief syntax description or this [tutorial on Real Python](#) for more context, history, and examples.

Important

We'll see lambdas again and again. Starting with summarizing data and again when cleaning. Understanding how to use these will help.

```
# with the def key
def first_col_f(d):
    return d[d.columns[0]]

# lambda (anonymous function)
first_col_l = lambda d: d[d.columns[0]]

first_col_f(help_df) == first_col_l(help_df)
```

```
0    True
1    True
2    True
3    True
4    True
5    True
6    True
Name: Day, dtype: bool
```

Try it yourself

read the code excerpt above carefully and try to match up the parts of a function: its name, the parameter list, and the body. Try writing your own lambda function.

Lambdas are an example of an [anonymous function](#)

We can put functions in dictionaries, or even define a lambda right in the dictionary.

```
df_display = {True: lambda d: d[d.columns[0]],
              False: lambda d: d}

for df in pd.read_html(achievements_url):
    _, n_cols = df.shape
    print(df_display[n_cols>3](df))
```

```
    Unnamed: 0_level_0                                topics \
    week
0          1                               [admin, python review]
1          2                               Loading data, Python review
2          3                           Exploratory Data Analysis
3          4                           Data Cleaning
4          5           Databases, Merging DataFrames
5          6 Modeling, Naive Bayes, classification performa...
6          7           decision trees, cross validation
7          8                           Regression
8          9                           Clustering
9         10           SVM, parameter tuning
10        11           KNN, Model comparison
11        12           Text Analysis
12        13           Images Analysis
13        14           Deep Learning

    skills
    Unnamed: 2_level_1
0      process
1  [access, prepare, summarize]
2  [summarize, visualize]
3 [prepare, summarize, visualize]
4  [access, construct, summarize]
5  [classification, evaluate]
6  [classification, evaluate]
7  [regression, evaluate]
8  [clustering, evaluate]
9  [optimize, tools]
10  [compare, tools]
11  [unstructured]
12  [unstructured, tools]
13  [tools, compare]
0     python
1     process
2     access
3     construct
4     summarize
5     visualize
6     prepare
7     classification
8     regression
9     clustering
10    evaluate
11    optimize
12    compare
13    representation
14    workflow
Name: (Unnamed: 0_level_0, keyword), dtype: object
0     python
1     process
2     access
3     construct
4     summarize
5     visualize
6     prepare
7     classification
8     regression
9     clustering
10    evaluate
11    optimize
12    compare
13    representation
14    workflow
Name: (Unnamed: 0_level_0, keyword), dtype: object
0     python
1     process
2     access
3     construct
4     summarize
5     visualize
6     prepare
7     classification
8     regression
9     clustering
10    evaluate
11    optimize
12    compare
13    representation
14    workflow
Name: (Unnamed: 0_level_0, keyword), dtype: object
```

In that excerpt `df_display` has a key that is defined to be true or false. The value for each item in the dictionary (separated by commas ,) is a lambda function, both of which take a parameter `d`, and one of which returns the first column `d[d.columns[0]]` and the other row which returns the whole data frame `d`.

In the loop, we set index into the dictionary with the key `n_cols > 3` with `df_display[n_cols>3]` sometimes it will be true and other times it will be false, which matches the two keys defined in the dictionary. Then the parameter it takes is `d`, which we pass the whole data frame `df` with the (`df`) at the end of the line.

Try it Yourself!

What does the `_` do?

Try using that dictionary outside of the loop. What is the value for a given key if you print it out like `df_display[a_key_value]?` What if you use `True` or `False` directly? What if you try a number? What is the type of that? What if you pass something that's not a DataFrame as the parameter? Make notes about these outputs

5.8. Questions after class

Ram Token Opportunity

add a question with a pull request; earn 1-2 ram tokens for submitting a question with the answer (with sources)

5.9. More Practice

- What `type` is the shape of a `pandas.DataFrame`?
- use a list comprehension to create a list that you could use as column names for data that consists of `N` measurements. Set `N=5` for now, but you suspect that the number might change.
- create a list of items with different types, then Create a dictionary with the types as keys using a dictionary comprehension. Dictionary comprehensions are similar to list comprehensions, in their form.
- Create a `lambda` function to print return the first 2 rows of a data frame

Ram Token Opportunity

Contribute possible practice questions to the notes using the suggest an edit button behind the GitHub menu at the top of the page.

6. Exploratory Data Analysis

```
import pandas as pd
```

6.1. 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

6.2. Data Frame from lists of Lists

On Friday, we collectively made a list of strings

```
# %load http://drsmb.co/310read
# share a sentence or a few words about how class is going or
# one takeaway you have from class so far.
# and attribute with a name after a hyphen(-)
# You can remain anonymous (this page & the notes will be fully public)
# by attributing it to a celebrity or pseudonym, but include *some* sort of attribution
sentence_list = [
    'Programming is a Practice. - Dr. Sarah Brown',
    "So far it's going pretty good. - Matt",
    'Pretty good pace, Github is still a little confusing is all - A',
    "Class is going well, I'm really excited for the new skills that I will attain through
    this course. - Diondra",
    'Good straight forward - Adam',
    'Good pace and engaging - Jacob',
    'I like cheese - Anon',
    'Going well, I enjoy python - Aiden',
    "Class is going very well, I'm excited to learn more about manipulating data sets -
    Greg",
    'Really enjoying the class so far, Clear instructions and outcomes - Michael',
    'So far this class is going well, very engaging lectures. - Anon',
    'Great pace and notes have been really useful - Brandon',
    'Good pace and easy to follow - Muhammad',
    'Class is going well and engaging, but also a little difficult getting into the swing of
    things - Isaiah',
    'Well paced, informative, and helpful - Vinnie',
    'Spectacular -Michael Jackson',
    'Very interesting! I am enjoying it a lot so far. Getting to experience pandas as well
    as using github/jupyter has been cool. One thing I would change though is slowing down
    the pace a bit. - Max'
]
```

We can check that by using type, first on the whole thing

```
type(sentence_list)
```

```
list
```

And then we can index into the list. Lists can be indexed by integers:

```
sentence_list[4]
```

```
'Good straight forward - Adam'
```

is one item from the list and then check its type:

```
type(sentence_list[4])
```

```
str
```

First, we'll convert our list of strings, to a list of lists with a list comprehension. This will [iterate](#) over each string in that list and apply the string method [split](#). Since we passed the parameter `' - '`, it will split at that character.

```
[sent_attr.split(' - ') for sent_attr in sentence_list]
```

ⓘ Question From Class

The split method will split at every occurrence of the character passed.

ⓘ Try it Yourself

Try splitting based on a different character (eg `' '`). What happens?

```

[['Programming is a Practice. ', ' Dr. Sarah Brown'],
 ['So far it's going pretty good. ', ' Matt'],
 ['Pretty good pace, Github is still a little confusing is all ', ' A'],
 ['Class is going well, I'm really excited for the new skills that I will attain through
this course. ',
 ' Diondra'],
 ['Good straight forward ', ' Adam'],
 ['Good pace and engaging ', ' Jacob'],
 ['I like cheese ', ' Anon'],
 ['Going well, I enjoy python ', ' Aiden'],
 ['Class is going very well, I'm excited to learn more about manipulating data sets ',
 ' Greg'],
 ['Really enjoying the class so far, Clear instructions and outcomes ',
 ' Michael'],
 ['So far this class is going well, very engaging lectures. ', ' Anon'],
 ['Great pace and notes have been really useful ', ' Brandon'],
 ['Good pace and easy to follow ', ' Muhammad'],
 ['Class is going well and engaging, but also a little difficult getting into the swing
of things ',
 ' Isaiah'],
 ['Well paced, informative, and helpful ', ' Vinnie'],
 ['Spectacular ', ' Michael Jackson'],
 ['Very interesting! I am enjoying it a lot so far. Getting to experience pandas as well
as using github/jupyter has been cool. One thing I would change though is slowing down
the pace a bit. ',
 ' Max']]
```

If we save it to a variable, we can analyze it better, for example indexing it

```
list_of_lists = [sent_attr.split('-') for sent_attr in sentence_list]
list_of_lists[0]
```

```
['Programming is a Practice. ', ' Dr. Sarah Brown']
```

This is a list, which we can check with:

```
type(list_of_lists[0])
```

```
list
```

If we take one item from that, it's a string

```
list_of_lists[0][1]
```

```
' Dr. Sarah Brown'
```

The list of lists is the same length as our original sentence_list, because it was made from that.

```
len(sentence_list)
```

```
17
```

```
len(list_of_lists)
```

Question From Class

While the list comprehension `iterate` over each string in that list, because it's a list, in order to `index` into it, we have to use an integer.

Iterating is taking each member in an object in turn, indexing is picking out a specified item. Iterating typically is done with the `for` keyword (either in a loop or a comprehension), indexing is done with `[]`

Then we can pass our list of lists to the DataFrame constructor and set the column headings with the `columns` parameter, which accepts a list.

```
pd.DataFrame([sent_attr.split('-') for sent_attr in sentence_list],
             columns=['sentence','attribution'])
```

	sentence	attribution
0	Programming is a Practice.	Dr. Sarah Brown
1	So far it's going pretty good.	Matt
2	Pretty good pace, Github is still a little con...	A
3	Class is going well, I'm really excited for th...	Diondra
4	Good straight forward	Adam
5	Good pace and engaging	Jacob
6	I like cheese	Anon
7	Going well, I enjoy python	Aiden
8	Class is going very well, I'm excited to learn...	Greg
9	Really enjoying the class so far, Clear instru...	Michael
10	So far this class is going well, very engaging...	Anon
11	Great pace and notes have been really useful	Brandon
12	Good pace and easy to follow	Muhammad
13	Class is going well and engaging, but also a l...	Isaiah
14	Well paced, informative, and helpful	Vinnie
15	Spectacular	Michael Jackson
16	Very interesting! I am enjoying it a lot so fa...	Max

💡 Hint

We built the list of lists here with a list comprehension because it's only a few items, but for a list of longer lists, you might use a for loop and `append`

6.3. Summarizing Data

```
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)
```

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()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number		
0	1 Robusta	ankole coffee producers coop	Uganda	kyangundu cooperative society	NaN	proc	€
1	2 Robusta	nishant gurjer	India	sethuraman estate kaapi royale	25	sethui	
2	3 Robusta	andrew hetzel	India	sethuraman estate	NaN		
3	4 Robusta	ugacof	Uganda	ugacof project area	NaN	l	
4	5 Robusta	katuka development trust ltd	Uganda	katikamu capca farmers association	NaN	develo	€

5 rows × 44 columns

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
count	28.000000	28.000000	28.000000	28.000000	28.000000
mean	14.500000	168.000000	2013.964286	7.702500	7.630714
std	8.225975	143.226317	1.346660	0.296156	0.303656
min	1.000000	1.000000	2012.000000	6.750000	6.670000
25%	7.750000	1.000000	2013.000000	7.580000	7.560000
50%	14.500000	170.000000	2014.000000	7.670000	7.710000
75%	21.250000	320.000000	2015.000000	7.920000	7.830000
max	28.000000	320.000000	2017.000000	8.330000	8.080000

8 rows × 21 columns

We can also select one variable at a time

```
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 there are 28 values.

```
balance_sorted = coffee_df['Balance'].sort_values().values
balance_sorted[6]
```

We can also extract each of the statistics that the `describe` method calculates individually, by name. The quantiles are tricky, we can't use `.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(.25)
```

```
7.560000000000005
```

We can also pass other values

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

```
7.83
```

Calculate the mean of the `Aftertaste` column:

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

```
7.559642857142856
```

6.4. What about the nonnumerical variables?

For example the color

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

```
0    Green
1      NaN
2    Green
3    Green
4    Green
Name: Color, 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
```

➊ further reading

On the [documentation page for describe](#) the “➊ 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 that's similar to what you want and then look at the related functions.

➊ 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?

What country is most prevalent in this dataset?

```
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 `idxmax`

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

```
'India'
```

ⓘ Question From Class

Q: Can we calculate the mode to find the most prevalent? A: Yes. We can also use the mode function, which works on both numerical or nonnumerical values.

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

```
0    India
dtype: object
```

6.5. Questions After Class

6.5.1. General Questions

6.5.1.1. How to know what functions are compatible with other functions?

6.5.1.2. Best place to find all the individual functions based on the `.describe()` function

6.5.1.3. Are there panda functions to read files other than CSV?

6.5.2. Clarifying

6.5.2.1. Is `sent_attr` in the DataFrame loop its own variable that we assign, or is it a base Python function?

6.5.2.2. Difference between `%load` and how we've been importing links to datasets in previous classes?

6.5.2.3. When I ran `[sent_attr.split('-') for sent_attr in sentence_list]` in Jupyter, I got a nameerror

Why do we use the value quantile instead of using quartile since we can use mean to find mean?

6.5.3. Course Admin and Assignment Questions

6.5.3.1. When are the achievements we earn in class going to be inputted in brightspace?

6.5.3.2. Is there anything we have to do for the assignment after committing the changes to the files?

6.5.3.3. When we push the homework are we pushing it to the portfolio?

6.5.3.4. Still a little unsure about what the num_numerical specifically is, is it just the number of numerical columns?

6.6. More Practice

1. Produce a table with the mean for each score.
2. Which variables have the most missing data?
3. What's the total number of bags of coffee produced?
4. Which ratings have similar ranges? (max, min)
5. Which rating are most consistent across coffees?
6. What score cutoff could you apply in order to select the top 10 best for Balance?

7. Visualization

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

7.1. Jupyter FAQ

Question from class

Why doesn't my jupyter print things out that are on the last line?

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 uses html to render the object in a more visually appealing way.

7.2. Review of describe

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'  
coffee_df = pd.read_csv(arabica_data_url)
```

We can describe it again, to see it has mostly the same variables we saw before, but some different as well.

```
coffee_df.describe()
```

	Unnamed: 0	Number.of.Bags	Aroma	Flavor	Aftertaste	Acid
count	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000	1311.000000
mean	656.000763	153.887872	7.563806	7.518070	7.397696	7.5331
std	378.598733	129.733734	0.378666	0.399979	0.405119	0.3815
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	328.500000	14.500000	7.420000	7.330000	7.250000	7.3300
50%	656.000000	175.000000	7.580000	7.580000	7.420000	7.5000
75%	983.500000	275.000000	7.750000	7.750000	7.580000	7.7500
max	1312.000000	1062.000000	8.750000	8.830000	8.670000	8.7500

Question from class

Why do we need the `()` on the describe but not on just the data

As is often the case, again this comes back to the type.

```
type(coffee_df)
```

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

is a data frame which has the `_repr_html_` method

Note

notice that this one has the ... in the columns and rows directions.

```
coffee_df
```

Unnamed: 0	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number
0	1	Arabica	metad plc	Ethiopia	metad plc
1	2	Arabica	metad plc	Ethiopia	metad plc
2	3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch
3	4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation
4	5	Arabica	metad plc	Ethiopia	metad plc
...
1306	1307	Arabica	juan carlos garcia lopez	Mexico	el centenario
1307	1308	Arabica	myriam kaplan-pasternak	Haiti	200 farms
1308	1309	Arabica	exportadora atlantic, s.a.	Nicaragua	finca las marías 017-053-0211/ 017-053-0212
1309	1310	Arabica	juan luis alvarado romero	Guatemala	finca el limon
1310	1312	Arabica	bismarck castro	Honduras	los hicaques

1311 rows × 44 columns

so it prints nicely as `tdid the coffee_df.describe()`

If we leave the `()` off we don't get nice formatting

Try it Yourself

what can you check in a `dataFrame` to predict if it will display with or without the `...`?

Hint

when objects do not have the `_repr_html_` method their output includes the type

```
coffee_df.describe
```

		Species	Owner
Country.of.Origin \			
0	1	Arabica	metad plc
1	2	Arabica	metad plc
2	3	Arabica	grounds for health admin
3	4	Arabica	yidnekachew dabessa
4	5	Arabica	metad plc
...
1306	1307	Arabica	juan carlos garcia lopez
1307	1308	Arabica	myriam kaplan-pasternak
1308	1309	Arabica	exportadora atlantic, s.a.
1309	1310	Arabica	juan luis alvarado romero
1310	1312	Arabica	bismarck castro
		Farm.Name	Lot.Number \
0		metad plc	NaN
1		metad plc	NaN
2	san marcos barrancas "san cristobal cuch		NaN
3	yidnekachew dabessa coffee plantation		NaN
4		metad plc	NaN
...
1306		el centenario	NaN
1307		200 farms	NaN
1308		finca las marías	017-053-0211/ 017-053-0212
1309		finca el limon	NaN
1310		los hicaques	103
		Mill \	
0		metad plc	
1		metad plc	
2			NaN
3		wolensu	
4		metad plc	
...
1306	la esperanza, municipio juchique de ferrer, ve...		
1307	coeb koperativ ekselsyo basen (350 members)		
1308		beneficio atlantic condega	
1309		beneficio serben	
1310		cigrah s.a de c.v.	
	ICO.Number		Company \
0	2014/2015	metad agricultural developmet plc	
1	2014/2015	metad agricultural developmet plc	
2	NaN		NaN
3	NaN	yidnekachew debessa coffee plantation	
4	2014/2015	metad agricultural developmet plc	
...
1306	1104328663		terra mia
1307	NaN		haiti coffee
1308	017-053-0211/ 017-053-0212		exportadora atlantic s.a
1309	11/853/165		unicafe
1310	13-111-053		cigrah s.a de c.v
	Altitude	...	Color Category.Two.Defects \
0	1950-2200	...	Green 0
1	1950-2200	...	Green 1
2	1600 - 1800 m	...	NaN 0
3	1800-2200	...	Green 2
4	1950-2200	...	Green 2
...
1306	900	...	None 20
1307	~350m	...	Blue-Green 16
1308	1100	...	Green 5
1309	4650	...	Green 4
1310	1400	...	Green 2
	Expiration		Certification.Body \
0	April 3rd, 2016	METAD Agricultural Development plc	
1	April 3rd, 2016	METAD Agricultural Development plc	
2	May 31st, 2011	Specialty Coffee Association	
3	March 25th, 2016	METAD Agricultural Development plc	
4	April 3rd, 2016	METAD Agricultural Development plc	
...
1306	September 17th, 2013		AMECAFE
1307	May 24th, 2013		Specialty Coffee Association
1308	June 6th, 2018		Instituto Hondureno del Café
1309	May 24th, 2013		Asociacion Nacional Del Café
1310	April 28th, 2018		Instituto Hondureno del Café
	Certification.Address	\	
0	309fcf77415a3661ae83e027f7e5f05dad786e44		
1	309fcf77415a3661ae83e027f7e5f05dad786e44		
2	36d0d00a3724338ba7937c52a378d085f2172daa		

```

3    309fcf77415a3661ae83e027f7e5f05dad786e44
4    309fcf77415a3661ae83e027f7e5f05dad786e44
...
1306 59e396ad6e22a1c22b248f958e1da2bd8af85272
1307 36d0d00a3724338ba7937c52a378d085f2172daa
1308 b4660a57e9f8cc613ae5b8f02bfce8634c763ab4
1309 b1f20fe3a819fd6b2ee0eb8fdc3da256604f1e53
1310 b4660a57e9f8cc613ae5b8f02bfce8634c763ab4

        Certification.Contact unit_of_measurement \
0    19fef5a731de2db57d16da10287413f5f99bc2dd      m
1    19fef5a731de2db57d16da10287413f5f99bc2dd      m
2    0878a7d4b9d35ddb0fe2ce69a2062cccb45a660      m
3    19fef5a731de2db57d16da10287413f5f99bc2dd      m
4    19fef5a731de2db57d16da10287413f5f99bc2dd      m
...
1306 0eb4ee5b3f47b20b049548a2fd1e7d4a2b70d0a7      m
1307 0878a7d4b9d35ddb0fe2ce69a2062cccb45a660      m
1308 7f521ca403540f81ec99daec7da19c2788393880      m
1309 724f04ad10ed31dbb9d260f0fd221ba48be8a95      ft
1310 7f521ca403540f81ec99daec7da19c2788393880      m

    altitude_low_meters altitude_high_meters altitude_mean_meters
0            1950.00           2200.00           2075.00
1            1950.00           2200.00           2075.00
2            1600.00           1800.00           1700.00
3            1800.00           2200.00           2000.00
4            1950.00           2200.00           2075.00
...
1306          900.00           900.00           900.00
1307          350.00           350.00           350.00
1308          1100.00          1100.00          1100.00
1309          1417.32          1417.32          1417.32
1310          1400.00          1400.00          1400.00

```

[1311 rows x 44 columns]>

so lets check the type of that.

```
type(coffee_df.describe)
```

```
method
```

it's a **bound method** or a function that *will* be applied to the DataFrame, but we didn't actually run the method. To see that it hasn't run, we can use an ipython [1] [magic %timeit](#)

```
%%timeit
coffee_df.describe
```

85.6 ns ± 1.71 ns per loop (mean ± std. dev. of 7 runs, 10000000 loops each)

```
%%timeit
coffee_df.describe()
```

42.1 ms ± 532 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)

Further reading

The [magic functions](#) can be defined for a cell or a line. [Timeit](#) is actually a standard python library that you can call on the command line or in a python script as well, but jupyter, by way of ipython gives us easy access to it with nice defaults.

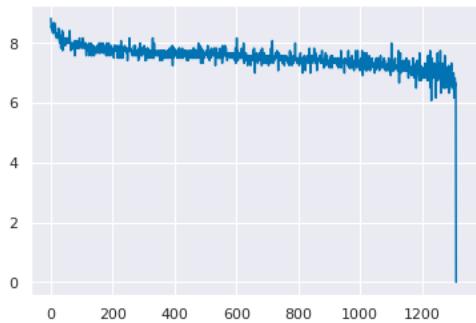
Note that without the `()` it runs much much faster, signaling that it did less finding the method, is less calculation than computing statistics on the data

7.3. Basic plots in pandas

Pandas gives us basic plots.

```
coffee_df['Flavor'].plot()
```

```
<AxesSubplot:>
```



Since we chose a series, it plotted that data as line vs the index.

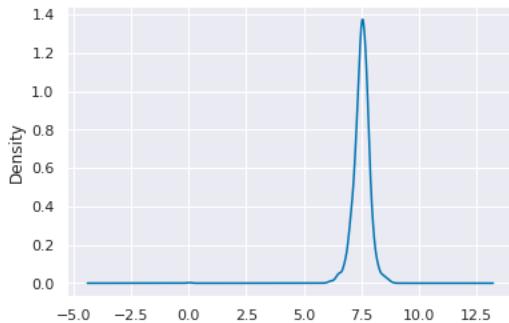
```
coffee_df.index
```

```
RangeIndex(start=0, stop=1311, step=1)
```

We can change the kind, for example to a [Kernel Density Estimate](#). This approximates the distribution of the data, you can think of it roughly like a smoothed out histogram.

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

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

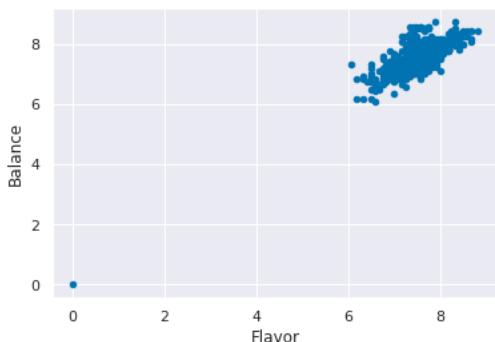


We can also plot two variables as a scatter plot, by specifying the `x`, `y` and `kind`

```
coffee_df.plot(x='Flavor',y='Balance', kind='scatter')
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```

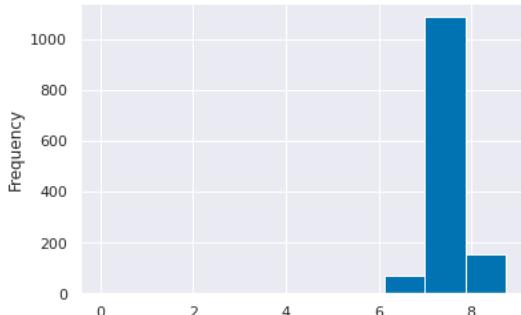
```
<AxesSubplot:xlabel='Flavor', ylabel='Balance'>
```



Let's Make a histogram plot of the Balance variable

```
coffee_df['Balance'].plot(kind='hist')
```

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



💡 Question from class

Can we plot two histograms with `coffee_df['Balance']['Flavor'].plot(kind='hist')`

```
:tags: ["raises-exception"]
coffee_df['Balance']['Flavor'].plot(kind='hist')
```

```
File "/tmp/ipykernel_1901/3818478052.py", line 1
  :tags: ["raises-exception"]
 ^
SyntaxError: invalid syntax
```

Let's break down why that errors. When we append things to the left, python interprets them by passing the output of one step to the input of the next one.

So `coffee_df['Balance'].plot(kind='hist')` first made a series, then plotted it. In the above, we again got the series, which works

```
coffee_df['Balance'].head(2)
```

💡 Note

Since I know these will be DataFrames, I'm appending the `head()` method to reduce the output for presentation.

```
0    8.42
1    8.42
Name: Balance, dtype: float64
```

But then, we tried to index it with 'Flavor', but we don't have that any more

```
coffee_df['Balance']['Flavor']

-----
KeyError                                 Traceback (most recent call last)
/tmp/ipykernel_1901/1214282069.py in <module>
----> 1 coffee_df['Balance']['Flavor']

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/pandas/core/series.py
in __getitem__(self, key)
  940
  941      elif key_is_scalar:
--> 942          return self._get_value(key)
  943
  944      if is_hashable(key):

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/pandas/core/series.py
in _get_value(self, label, takeable)
 1049
 1050     # Similar to Index.get_value, but we do not fall back to positional
--> 1051     loc = self.index.get_loc(label)
 1052     return self.index._get_values_for_loc(self, loc, label)
 1053

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/indexes/range.py in get_loc(self, key, method, tolerance)
  386         except ValueError as err:
  387             raise KeyError(key) from err
--> 388     raise KeyError(key)
  389     return super().get_loc(key, method=method, tolerance=tolerance)
  390

KeyError: 'Flavor'
```

So we get a key error and we know this is the part of the line we have to change.

We need to index into the DataFrame and pick two columns at once. When we index, we can use the name of a variable as a string or a list. We can build this list on the fly and python executes from the inside out.

The outer [] index and the inner [] make a list

```
coffee_df[['Balance', 'Flavor']].head(2)
```

	Balance	Flavor
0	8.42	8.83
1	8.42	8.67

we could also build the list first, then index for readability

```
hist_vars = ['Balance', 'Flavor'].head(2)
coffee_df[hist_vars]
```

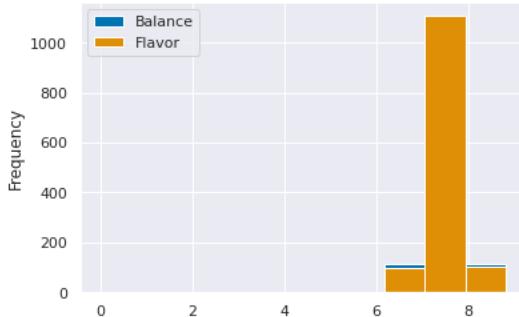
```
-----
AttributeError                                Traceback (most recent call last)
/tmp/ipykernel_1901/2943951791.py in <module>
----> 1 hist_vars = ['Balance', 'Flavor'].head(2)
  2 coffee_df[hist_vars]

AttributeError: 'list' object has no attribute 'head'
```

This gives us a data frame, which we can plot.

```
coffee_df[['Balance', 'Flavor']].plot(kind='hist')
```

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



We'll see ways to improve this on Friday.

7.4. 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.

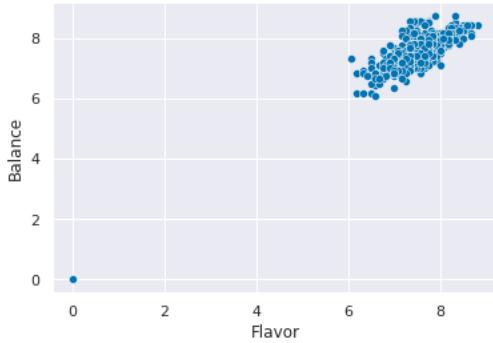
We can get that basic plot back.

Think Ahead

Learning ggplot is a way to earn level 3 for visualize

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

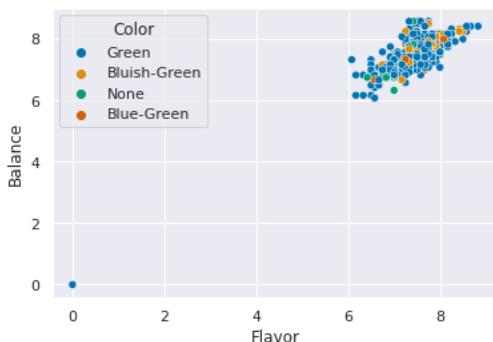
```
<AxesSubplot:xlabel='Flavor', ylabel='Balance'>
```



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

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

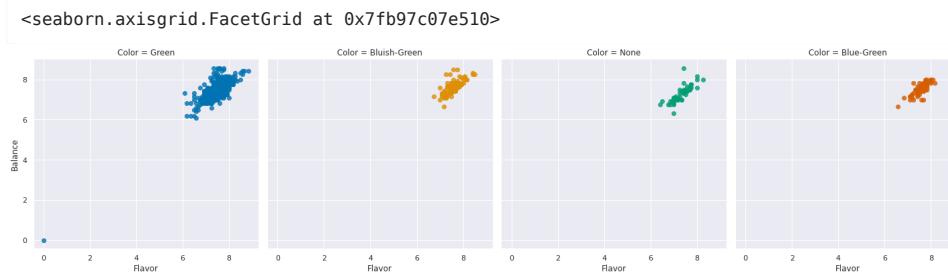
```
<AxesSubplot:xlabel='Flavor', ylabel='Balance'>
```



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 balance are related.

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. We'll turn that off for now, with `,fit_reg=False`.

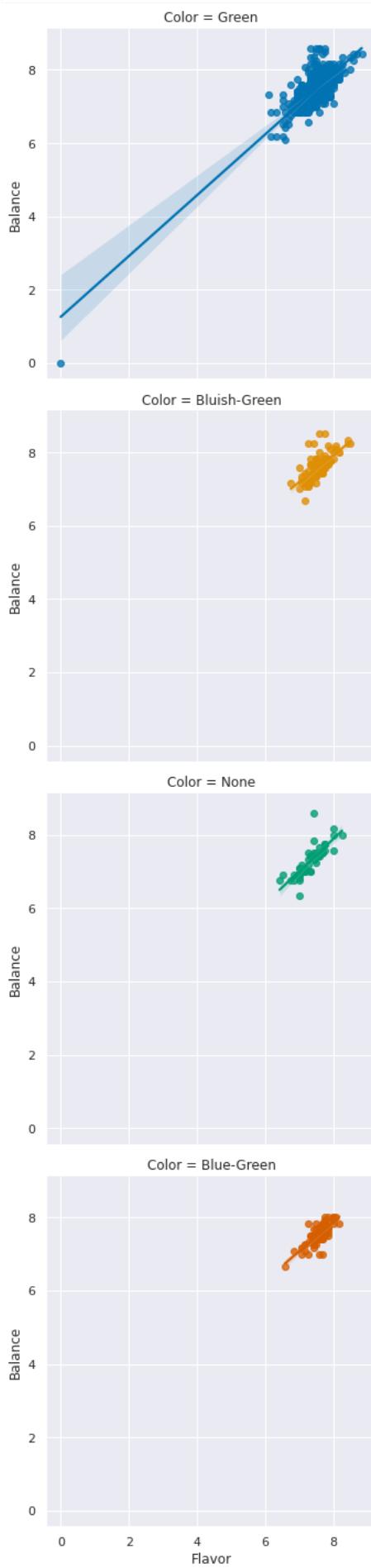
```
sns.lmplot(data=coffee_df,x='Flavor',y='Balance',hue='Color',
            col='Color',fit_reg=False)
```



`col` stands for column. We can also use `row`

```
sns.lmplot(data=coffee_df,x='Flavor',y='Balance',hue='Color',
            row='Color')
```

<seaborn.axisgrid.FacetGrid at 0x7fb977f8b510>



We can also use both together:

```
sns.lmplot(data=coffee_df,x='Flavor',y='Balance',hue='Color',
            row='Color',col='Variety')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb97c019c50>
```

```
Error in callback <function flush_figures at 0x7fb97f298320> (for post_execute):
```

```
-----  
KeyboardInterrupt                                     Traceback (most recent call last)  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/matplotlib_inline/backend_inline.py in flush_figures()  
    119         # ignore the tracking, just draw and close all figures  
    120         try:  
--> 121             return show(True)  
    122         except Exception as e:  
    123             # safely show traceback if in IPython, else raise  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/matplotlib_inline/backend_inline.py in show(close, block)  
    41             display(  
    42                 figure_manager.canvas.figure,  
--> 43                 metadata=_fetch_figure_metadata(figure_manager.canvas.figure)  
    44             )  
    45     finally:  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/display.py in display(include, exclude, metadata, transient,  
display_id, *objs, **kwargs)  
    318         publish_display_data(data=obj, metadata=metadata, **kwargs)  
    319     else:  
--> 320         format_dict, md_dict = format(obj, include=include, exclude=exclude)  
    321         if not format_dict:  
    322             # nothing to display (e.g. _ipython_display_ took over)  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/formatters.py in format(self, obj, include, exclude)  
    178         md = None  
    179         try:  
--> 180             data = formatter(obj)  
    181         except:  
    182             # FIXME: log the exception  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/decorator.py in  
fun(*args, **kw)  
    230         if not kwsyntax:  
    231             args, kw = fix(args, kw, sig)  
--> 232         return caller(func, *(extras + args), **kw)  
    233     fun.__name__ = func.__name__  
    234     fun.__doc__ = func.__doc__  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/formatters.py in catch_format_error(method, self, *args, **kwargs)  
    222     """show traceback on failed format call"""  
    223     try:  
--> 224         r = method(self, *args, **kwargs)  
    225     except NotImplementedError:  
    226         # don't warn on NotImplementedError  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/formatters.py in __call__(self, obj)  
    339         pass  
    340     else:  
--> 341         return printer(obj)  
    342     # Finally look for special method names  
    343     method = get_real_method(obj, self.print_method)  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/IPython/core/pylabtools.py in print_figure(fig, fmt, bbox_inches, base64,  
**kwargs)  
    149     FigureCanvasBase(fig)  
    150  
--> 151     fig.canvas.print_figure(bytes_io, **kw)  
    152     data = bytes_io.getvalue()  
    153     if fmt == 'svg':  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/matplotlib/backend_bases.py in print_figure(self, filename, dpi, facecolor,  
edgecolor, orientation, format, bbox_inches, pad_inches, bbox_extra_artists, backend,  
**kwargs)  
    2288         )  
    2289         with getattr(renderer, "_draw_disabled", nullcontext()):  
-> 2290             self.figure.draw(renderer)  
    2291  
    2292         if bbox_inches:  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/artist.py  
in draw_wrapper(artist, renderer, *args, **kwargs)  
    71     @wraps(draw)  
    72     def draw_wrapper(artist, renderer, *args, **kwargs):  
--> 73         result = draw(artist, renderer, *args, **kwargs)  
    74         if renderer._rasterizing:  
    75             renderer.stop_rasterizing()  
  
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/artist.py
```

```
in draw_wrapper(artist, renderer)
    48         renderer.start_filter()
    49
---> 50     return draw(artist, renderer)
    51 finally:
    52     if artist.get_agg_filter() is not None:

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/figure.py
in draw(self, renderer)
    2802     self.patch.draw(renderer)
    2803     mimage._draw_list_compositing_images(
-> 2804         renderer, self, artists, self.suppressComposite)
    2805
    2806     for sfig in self.subfigs:

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/image.py in
_draw_list_compositing_images(renderer, parent, artists, suppress_composite)
    130     if not_composite or not has_images:
    131         for a in artists:
--> 132             a.draw(renderer)
    133     else:
    134         # Compose any adjacent images together

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/artist.py
in draw_wrapper(artist, renderer)
    48         renderer.start_filter()
    49
---> 50     return draw(artist, renderer)
    51 finally:
    52     if artist.get_agg_filter() is not None:

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/axes/_base.py in draw(self, renderer)
    3081     mimage._draw_list_compositing_images(
-> 3083         renderer, self, artists, self.figure.suppressComposite)
    3084
    3085     renderer.close_group('axes')

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/image.py in
_draw_list_compositing_images(renderer, parent, artists, suppress_composite)
    130     if not_composite or not has_images:
    131         for a in artists:
--> 132             a.draw(renderer)
    133     else:
    134         # Compose any adjacent images together

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/artist.py
in draw_wrapper(artist, renderer)
    48         renderer.start_filter()
    49
---> 50     return draw(artist, renderer)
    51 finally:
    52     if artist.get_agg_filter() is not None:

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/axis.py in
draw(self, renderer, *args, **kwargs)
    1161
    1162     for tick in ticks_to_draw:
-> 1163         tick.draw(renderer)
    1164
    1165     # scale up the axis label box to also find the neighbors, not

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/artist.py
in draw_wrapper(artist, renderer)
    48         renderer.start_filter()
    49
---> 50     return draw(artist, renderer)
    51 finally:
    52     if artist.get_agg_filter() is not None:

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/axis.py in
draw(self, renderer)
    297     for artist in [self.gridline, self.tick1line, self.tick2line,
    298                   self.label1, self.label2]:
--> 299         artist.draw(renderer)
    300     renderer.close_group(self.__name__)
    301     self.stale = False

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/artist.py
in draw_wrapper(artist, renderer)
    48         renderer.start_filter()
    49
---> 50     return draw(artist, renderer)
    51 finally:
    52     if artist.get_agg_filter() is not None:
```

```

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/lines.py in
draw(self, renderer)
    746     renderer.open_group('line2d', self.get_gid())
    747     if self._lineStyles[self._linestyle] != '_draw_nothing':
--> 748         tpath, affine = (self._get_transformed_path()
    749             .get_transformed_path_and_affine())
    750     if len(tpath.vertices):

```

```

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/lines.py in
_get_transformed_path(self)
    706     """Return this line's `~matplotlib.transforms.TransformedPath`."""
    707     if self._transformed_path is None:
--> 708         self._transform_path()
    709     return self._transformed_path
    710

```

```

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/matplotlib/lines.py in
_transform_path(self, subslice)
    701     else:
    702         _path = self._path
--> 703     self._transformed_path = TransformedPath(_path, self.get_transform())
    704
    705     def _get_transformed_path(self):

```

```

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/matplotlib/transforms.py in __init__(self, path, transform)
    2750     self._path = path
    2751     self._transform = transform
-> 2752     self.set_children(transform)
    2753     self._transformed_path = None
    2754     self._transformed_points = None

```

```

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/matplotlib/transforms.py in set_children(self, *children)
    215     # parents are destroyed, references from the children won't
    216     # keep them alive.
--> 217     for child in children:
    218         # Use weak references so this dictionary won't keep obsolete nodes
    219         # alive; the callback deletes the dictionary entry. This is a

```

KeyboardInterrupt:

How could we choose which countries to select to make this not show the ones with very few points?

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

Or we can focus on the countries, but wrap them.

```
sns.lmplot(data=coffee_df,x='Flavor',y='Balance',hue='Color',
            col='Country.of.Origin',col_wrap=5)
```

7.5. Questions after class

Ram Token Opportunity

add a question with a pull request; earn 1-2 ram tokens for submitting a question with the answer (with sources)

7.6. More practice

1. Plot the kde for the `Aftertaste`
2. How does `Total.Cup.Points` vary by `Certification.Body`
3. Are moisture and sweetness related? Does that relationship vary by Color?

`[[1]]` the kernel of python we're using

8. Exploratory Data Analysis

```
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'
coffee_df = pd.read_csv(arabica_data_url)
```

Which of the following scores is distributed most similarly to Sweetness?

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

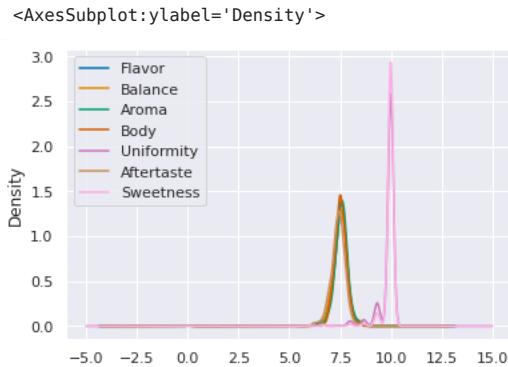
First step is to subset the data:

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

	Flavor	Balance	Aroma	Body	Uniformity	Aftertaste	Sweetness
0	8.83	8.42	8.67	8.50	10.0	8.67	10.0
1	8.67	8.42	8.75	8.42	10.0	8.50	10.0

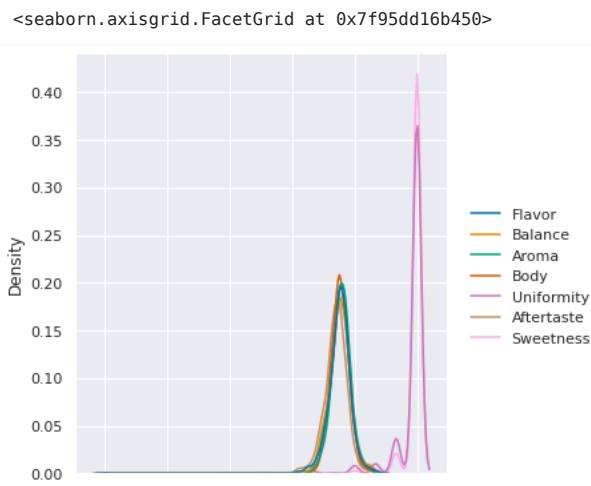
Then we produce a kde plot

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



We could also do it with seaborn

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



If we forget the parameter `kind`, we get its default value, which is histogram

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.

```
``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)
```

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

```
<seaborn.axisgrid.FacetGrid at 0x7f95dbf39cd0>
```



8.1. Summarizing with two 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 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 `value_counts()` on the `Country.of.Origin` column.

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

```
Mexico      236  
Colombia    183  
Name: Country.of.Origin, dtype: int64
```

The data also has `Number.of.Bags` however. How can we check which has the most bags?

We can do this with groupby.

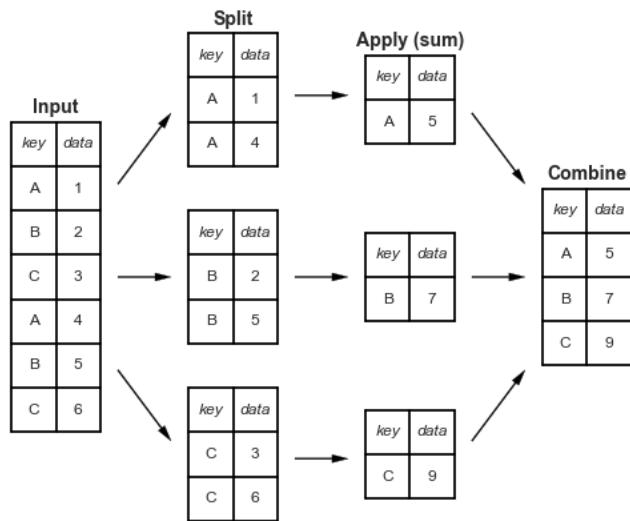
```
coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()
```

```

Country.of.Origin
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
Name: Number.of.Bags, dtype: int64

```

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` because we picked one variable out)

8.2. How doe Groupby Work?

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 0x7f95d9b4bbd0>
```

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'>
```

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

Question from class

How can we sort it?

First, we'll make it a variable to keep the code legible, then we'll use `sort_values()`

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

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

Which, by default uses ascending order, the method has an `ascending` parameter and its default value is `True`, so we can switch it to `False` to get descending

```
bag_total_df.sort_values(ascending=False,)
```

```

Country.of.Origin      Number.of.Bags
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

```

8.3. Customizing Data Summaries

We've looked at an overall summary with `describe` on all the variables, describe on one variable, individual statistics on all variables, and individual statistics on one variable. We can also build summaries of multiple variables with a custom subset of summary statistics, with `aggregate` or using its alias `agg`

```
country_grouped.agg({'Number.of.Bags':'sum',
                     'Balance':['mean','count'],})
```

Learning Tip

When a person reads a line of code, they have to use their working memory to hold the whole thing in order to make sense of it. Human working memory only holds 5-9 things at a time, that's why a phone number is 7 digits without the area code, (when people used to have to actually dial the digits, they also didn't need the area codes for short-distance calls, which were most of the calls).

How many concepts does a person have to hold in working memory at once to parse the second version?

If you are writing the code, and building it up, you hold the previous pieces in your memory differently than a person coming to it for the first time.

Country.of.Origin	Number.of.Bags	Balance	
	sum	mean	count
Brazil	30534	7.531515	132
Burundi	520	7.415000	2
China	55	7.548125	16
Colombia	41204	7.708415	183
Costa Rica	10354	7.637255	51
Cote d'Ivoire	2	7.080000	1
Ecuador	1	7.830000	1
El Salvador	4449	7.711429	21
Ethiopia	11761	7.972273	44
Guatemala	36868	7.469890	181
Haiti	390	7.056667	6
Honduras	13167	7.163962	53
India	20	7.420000	1
Indonesia	1658	7.520000	20
Japan	20	7.830000	1
Kenya	3971	7.800400	25
Laos	81	7.416667	3
Malawi	557	7.371818	11
Mauritius	1	7.170000	1
Mexico	24140	7.328686	236
Myanmar	10	7.133750	8
Nicaragua	6406	7.278462	26
Panama	537	7.875000	4
Papua New Guinea	7	8.250000	1
Peru	2336	7.666000	10
Philippines	259	7.400000	5
Rwanda	150	7.750000	1
Taiwan	1914	7.426000	75
Tanzania, United Republic Of	3760	7.469750	40
Thailand	1310	7.524063	32
Uganda	3868	7.660769	26
United States	361	7.947500	8
United States (Hawaii)	833	7.644110	73
United States (Puerto Rico)	71	7.647500	4
Vietnam	10	7.547143	7
Zambia	13	7.420000	1

We could also string this together, but splitting into interim variables makes code more readable. Shorter lines are easier to read (and sometimes auto-enforced on projects). Also, by giving a good variable name to the interim states we get more description, which can help a human better read it.

```
coffee_df.groupby('Country.of.Origin').agg({'Number.of.Bags':'sum', 'Balance': ['mean', 'count'],})
```

Country.of.Origin	Number.of.Bags	Balance	
	sum	mean	count
Brazil	30534	7.531515	132
Burundi	520	7.415000	2
China	55	7.548125	16
Colombia	41204	7.708415	183
Costa Rica	10354	7.637255	51
Cote d'Ivoire	2	7.080000	1
Ecuador	1	7.830000	1
El Salvador	4449	7.711429	21
Ethiopia	11761	7.972273	44
Guatemala	36868	7.469890	181
Haiti	390	7.056667	6
Honduras	13167	7.163962	53
India	20	7.420000	1
Indonesia	1658	7.520000	20
Japan	20	7.830000	1
Kenya	3971	7.800400	25
Laos	81	7.416667	3
Malawi	557	7.371818	11
Mauritius	1	7.170000	1
Mexico	24140	7.328686	236
Myanmar	10	7.133750	8
Nicaragua	6406	7.278462	26
Panama	537	7.875000	4
Papua New Guinea	7	8.250000	1
Peru	2336	7.666000	10
Philippines	259	7.400000	5
Rwanda	150	7.750000	1
Taiwan	1914	7.426000	75
Tanzania, United Republic Of	3760	7.469750	40
Thailand	1310	7.524063	32
Uganda	3868	7.660769	26
United States	361	7.947500	8
United States (Hawaii)	833	7.644110	73
United States (Puerto Rico)	71	7.647500	4
Vietnam	10	7.547143	7
Zambia	13	7.420000	1

💡 Question from Class

What does the `count` do? or how can we figure it out?

In this case, let's compare `count` to `shape` we know that `shape` returns the number of rows and columns. Also `shape` is an attribute, not a method (so no `()` for `shape`). However, there's a more important difference.

```
coffee_df[['Balance', 'Farm.Name', 'Lot.Number']].shape
```

```
-----
KeyError Traceback (most recent call last)
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
    3360         try:
-> 3361             return self._engine.get_loc(casted_key)
    3362         except KeyError as err:
/pandas/_libs/index/IndexEngine.get_loc()

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/pandas/_libs/index.pyx
in pandas._libs.index.IndexEngine.get_loc()
/pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()
/pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('Balance', 'Farm.Name', 'Lot.Number')

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

KeyError Traceback (most recent call last)
/tmp/ipykernel_1950/4170084615.py in <module>
----> 1 coffee_df[['Balance', 'Farm.Name', 'Lot.Number']].shape

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/frame.py in __getitem__(self, key)
    3456         if self.columns.nlevels > 1:
    3457             return self._getitem_multilevel(key)
-> 3458         indexer = self.columns.get_loc(key)
    3459         if is_integer(indexer):
    3460             indexer = [indexer]

/pandas/_libs/indexes/base.py in get_loc(self, key, method, tolerance)
    3361             return self._engine.get_loc(casted_key)
    3362         except KeyError as err:
-> 3363             raise KeyError(key) from err
    3364
    3365     if is_scalar(key) and isna(key) and not self.hasnans:
KeyError: ('Balance', 'Farm.Name', 'Lot.Number')
```

```
coffee_df[['Balance', 'Farm.Name', 'Lot.Number']].count()
```

```
KeyError Traceback (most recent call last)
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
    3360         try:
    3361             return self._engine.get_loc(casted_key)
-> 3361     except KeyError as err:
    3362         raise KeyError(key) from err

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/pandas/_libs/index.pyx
in pandas._libs.index.IndexEngine.get_loc()

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/pandas/_libs/index.pyx
in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('Balance', 'Farm.Name', 'Lot.Number')

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

KeyError Traceback (most recent call last)
/tmp/ipykernel_1950/48982150.py in <module>
----> 1 coffee_df['Balance','Farm.Name','Lot.Number'].count()

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/pandas/core/frame.py
in __getitem__(self, key)
    3456         if self.columns.nlevels > 1:
    3457             return self._getitem_multilevel(key)
-> 3458         indexer = self.columns.get_loc(key)
    3459         if is_integer(indexer):
    3460             indexer = [indexer]

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
    3361         return self._engine.get_loc(casted_key)
    3362     except KeyError as err:
-> 3363         raise KeyError(key) from err
    3364
    3365     if is_scalar(key) and isna(key) and not self.hasnans:
```

Here we get different number for **Balance** and **Farm Name**. The shape tells us how many rows there are, while count tells us how many are not null.

We can verify visually that some are null with:

```
coffee_df.head()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	M
0	1 Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc
1	2 Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc
2	3 Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN
3	4 Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolens
4	5 Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc

5 rows × 44 columns

Correction

This response is slightly corrected from what I said in class, because in the balance column, it does match, but in general it doesn't. `count` on grouped will only be the same as `value_count` when `count` is applied to a column that does not have any missing values where the grouping variable has a value.

On the balance column alone in class, we checked that the grouped count matched the country value counts.

```
country_grouped[['Balance', 'Farm.Name', 'Lot.Number']].count().sort_values(by='Balance'
ascending=False)
```

```
File "/tmp/ipykernel_1950/3556086896.py", line 2
    ascending=False)
    ^
SyntaxError: invalid syntax
```

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

```
Mexico          236
Colombia        183
Guatemala      181
Brazil          132
Taiwan          75
United States (Hawaii)    73
Honduras         53
Costa Rica       51
Ethiopia         44
Tanzania, United Republic Of 40
Thailand         32
Uganda           26
Nicaragua        26
Kenya            25
El Salvador      21
Indonesia        20
China             16
Malawi           11
Peru              10
United States     8
Myanmar          8
Vietnam          7
Haiti             6
Philippines       5
Panama           4
United States (Puerto Rico) 4
Laos              3
Burundi          2
Ecuador          1
Rwanda           1
Japan             1
Zambia           1
Papua New Guinea 1
Mauritius        1
Cote d'Ivoire     1
India             1
Name: Country.of.Origin, dtype: int64
```

In class, they were the same, because `Balance` doesn't have missing values. `Farm.Name` and `Lot.Number` have a lot of missing values, so they're different numbers.

Another function we could use when we first examine a dataset is `info` this tells us about the NaN values up front.

```
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
```

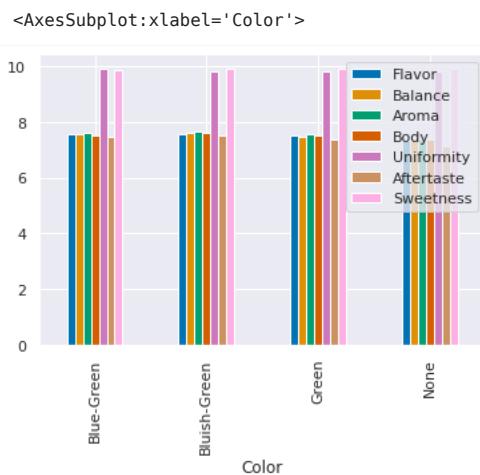
8.4. Using summaries for visualization

! Important

This section is an extension, that we didn't get to in class, but might help in your assignment

For example, we can group by color and take the mean of each of the scores from the beginning of class and then use a bar chart.

```
color_grouped = coffee_df.groupby('Color')[scores_of_interest].mean()
color_grouped.plot(kind='bar')
```



8.5. Questions after class

Ram Token Opportunity

add a question with a pull request; earn 1-2 ram tokens for submitting a question with the answer (with sources)

8.6. More Practice

- Make a table that's 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.

9. Reshaping Data

Today, we'll begin reshaping data. We'll cover:

- filtering
- applying a function to all rows
- what is tidy data
- reshaping data into tidy data

First some setup:

```
import pandas as pd
import seaborn as sns

sns.set_theme(font_scale=2, palette='colorblind')
arabica_data_url = 'https://raw.githubusercontent.com/jldbc/coffee-quality-database/master/data/arabica_data_cleaned.csv'
```

Tip

I used `set_theme` to change both the font 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.

9.1. Cleaning Data

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
- changing the representation of information

9.2. 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 + file, na_values= '') for file in datasets]
```

```
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()
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/ipykernel_launcher.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.
    """Entry point for launching an IPython kernel.
```

```
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)
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/ipykernel_launcher.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.
    """Entry point for launching an IPython kernel.
```

```
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()
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/ipykernel_launcher.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.
    """Entry point for launching an IPython kernel.
```

```
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)
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/ipykernel_launcher.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.
    """Entry point for launching an IPython kernel.
```

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

For the third one, however, we can use groupby

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

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3457: FutureWarning: Dropping invalid columns
in DataFrameGroupBy.mean is deprecated. In a future version, a TypeError will be raised.
Before calling .mean, select only columns which should be valid for the function.
exec(code_obj, self.user_global_ns, self.user_ns)
```

result

person

```
Jane Doe  805.5
John Smith -1.0
Mary Johnson 15.5
```

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

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3457: FutureWarning: Dropping invalid columns
in DataFrameGroupBy.mean is deprecated. In a future version, a TypeError will be raised.
Before calling .mean, select only columns which should be valid for the function.
exec(code_obj, self.user_global_ns, self.user_ns)
```

result

treatment

```
a -54.333333
b 703.666667
```

The original [Tidy Data](#) paper is worth reading to build a deeper understanding of these ideas.

9.3. 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

Correction

I fixed the three data files so the spaces can be removed. You will need to

```
treat_df.melt(value_vars = ['treatmenta','treatmentb'],
               id_vars = ['name'],
               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

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

```
tidy_treat_df.groupby('name').mean()
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3457: FutureWarning: Dropping invalid columns
in DataFrameGroupBy.mean is deprecated. In a future version, a TypeError will be raised.
Before calling .mean, select only columns which should be valid for the function.
exec(code_obj, self.user_global_ns, self.user_ns)
```

name
Jane Doe
John Smith
Mary Johnson

9.4. Filtering Data by a column

Let's go back to the coffee dataset

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

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Num
1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2
2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2
3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	
4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	
5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2

5 rows × 43 columns

Recall on Friday we computed the total number of bags per country.

```
# compute total bags per country
bag_total_df = coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()
```

We can subset this to get only the countries with over 15000 using a boolean mask:

```
bag_total_df[bag_total_df>15000]
```

```
Country.of.Origin
Brazil      30534
Colombia    41204
Guatemala   36868
Mexico      24140
Name: Number.of.Bags, dtype: int64
```

what we put in the `[]` has to be the same length and each element has to be boolean

```
len(bag_total_df>15000)
```

```
36
```

```
mask = bag_total_df>15000
type(mask[0])
```

```
numpy.bool_
```

9.5. Augmenting a dataset

We want the names of the countries as a list, so we extract the index of that series and then cast it to a list.

```
high_prod_countries = list(bag_total_df[bag_total_df>15000].index)
```

Next we want to be able to check if a country is in this list, so we'll make a lambda that can do that

```
high_prod = lambda c: c in high_prod_countries
```

Recall, the `lambda` keyword makes a function

```
type(high_prod)
```

```
function
```

We can test it

```
high_prod('Mexico'), high_prod('Ethiopia')
```

```
(True, False)
```

Now, we can apply that lambda function to each country in our whole coffee data frame. and save that to a new DataFrame.

```
coffee_df['high_production'] = coffee_df['Country.of.Origin'].apply(high_prod)
```

```
coffee_df.head()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	Mill	ICO.Nun
1	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2
2	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2
3	Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN	
4	Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolensu	
5	Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc	2014/2

5 rows × 44 columns

Finally, we can filter the whole data frame using that new column.

```
high_prod_coffee_df = coffee_df[coffee_df['high_production']]
```

Question from class

How can we get the ones not on that list?

```
low_prod_coffee_df = coffee_df[coffee_df['high_production']==False]
```

Try it Yourself

Replace the FIXMEds in the excerpt below to reshape the data to have a value column with the value of the score and a Score column that indicates which score is in that . Keep the color and country as values

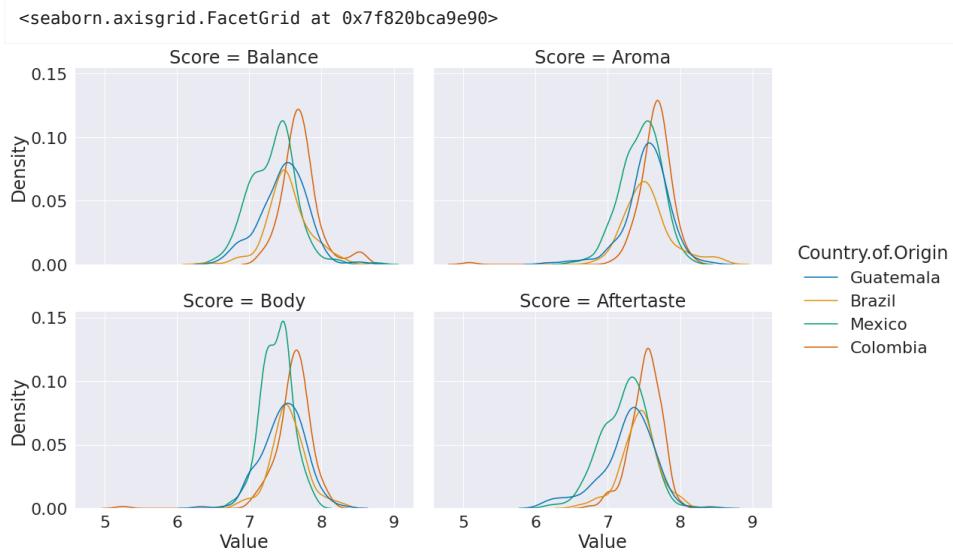
```
scores_of_interest = ['Balance', 'Aroma', 'Body', 'Aftertaste']
attrs_of_interest = ['Country.of.Origin', 'Color']
high_prod_coffee_df_melted = high_prod_coffee_df.melt(
    id_vars = FIXME,
    value_vars = FIXME,
    value_name = 'Value',
    var_name = 'Score')
```

so that it looks like the following

	Country.of.Origin	Color	Score	Value
0	Guatemala	NaN	Balance	8.42
1	Brazil	Bluish-Green	Balance	8.33
2	Mexico	Green	Balance	8.17
3	Brazil	Green	Balance	8.00
4	Brazil	Green	Balance	8.00

Try it Yourself

Plot the distribution of each score on a separate subplot and use a different color for each country. Use a kde for the distributions.



10. More Reshaping

Continuing from Friday.

```
import pandas as pd
import seaborn as sns

# make plots look nicer, increase font size, and use colorblind compatible colors
sns.set_theme(font_scale=2, palette='colorblind')
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)

# compute ____ per ____
bag_total_df = coffee_df.groupby('Country.of.Origin')['Number.of.Bags'].sum()

# subset the summary Series for countries with over 15000 total and store as a list
high_prod_countries = list(bag_total_df[bag_total_df>15000].index)

# a lambda function that checks if a string c is one of the
# countries in high_prod_countries
high_prod = lambda c: c in high_prod_countries

# add a column that indicates that the country is a high producer
coffee_df['high_production'] = coffee_df['Country.of.Origin'].apply(high_prod)

# filter based on production level threshold
high_prod_coffee_df = coffee_df[coffee_df['high_production']]
```

What happened when we filtered the data?

```
coffee_df.shape, high_prod_coffee_df.shape
```

```
((1311, 45), (732, 45))
```

We have many fewer rows.

Now that we've filtered the data. Let's practice reshaping data to be Tidy again.

```
# replace the FIXMES
scores_of_interest = ['Balance', 'Aroma', 'Body', 'Aftertaste']
attrs_of_interest = ['Country.of.Origin', 'Color']
high_prod_coffee_df_melted = high_prod_coffee_df.melt(
    id_vars = attrs_of_interest,
    value_vars = scores_of_interest,
    var_name = 'Score')
```

What happened?

```
high_prod_coffee_df_melted.shape
```

```
(2928, 4)
```

Now the shape is 4 times as long (because the length of the list we passed to value_vars is 4). And it has 4 columns: the length of the list we passed to `id_vars` + 2 (variable, value)

```
len(scores_of_interest)
```

```
4
```

```
len(scores_of_interest)*len(high_prod_coffee_df)
```

```
2928
```

We can see the column names and what they have in them here:

```
high_prod_coffee_df_melted.head()
```

	Country.of.Origin	Color	Score	value
0	Guatemala	NaN	Balance	8.42
1	Brazil	Bluish-Green	Balance	8.33
2	Mexico	Green	Balance	8.17
3	Brazil	Green	Balance	8.00
4	Brazil	Green	Balance	8.00

Note that we passed a value to `var_name` to make that column named "Score". We could also not pass that

```
high_prod_coffee_df.melt(
    id_vars = attrs_of_interest,
    value_vars = scores_of_interest)
```

	Country.of.Origin	Color	variable	value
0	Guatemala	NaN	Balance	8.42
1	Brazil	Bluish-Green	Balance	8.33
2	Mexico	Green	Balance	8.17
3	Brazil	Green	Balance	8.00
4	Brazil	Green	Balance	8.00
...
2923	Mexico	Green	Aftertaste	6.42
2924	Mexico	Green	Aftertaste	6.83
2925	Brazil	Green	Aftertaste	6.83
2926	Mexico	None	Aftertaste	6.25
2927	Guatemala	Green	Aftertaste	6.67

2928 rows × 4 columns

then we have `variable` and `value` as column names.

Try it yourself

How could you rename the `value` column?

The head has only 'Balance' in the 'Score' column, we could use `sample` to pick a random subset of the rows instead to see different values.

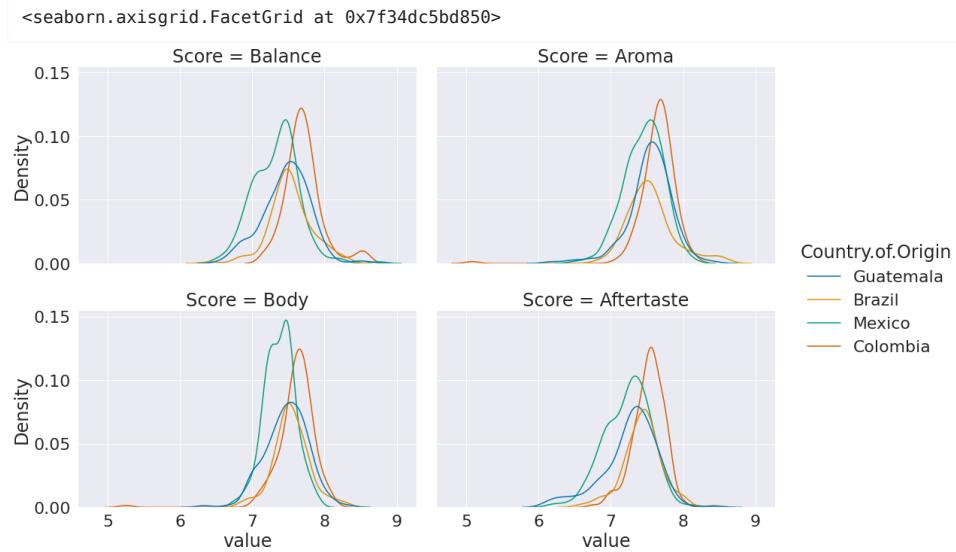
```
high_prod_coffee_df_melted.sample(5)
```

	Country.of.Origin	Color	Score	value
2263	Guatemala	Green	Aftertaste	7.58
2049	Brazil	Green	Body	7.08
1917	Guatemala	Green	Body	7.75
2259	Brazil	Nan	Aftertaste	7.50
263	Colombia	Green	Balance	7.17

What does this let us do?

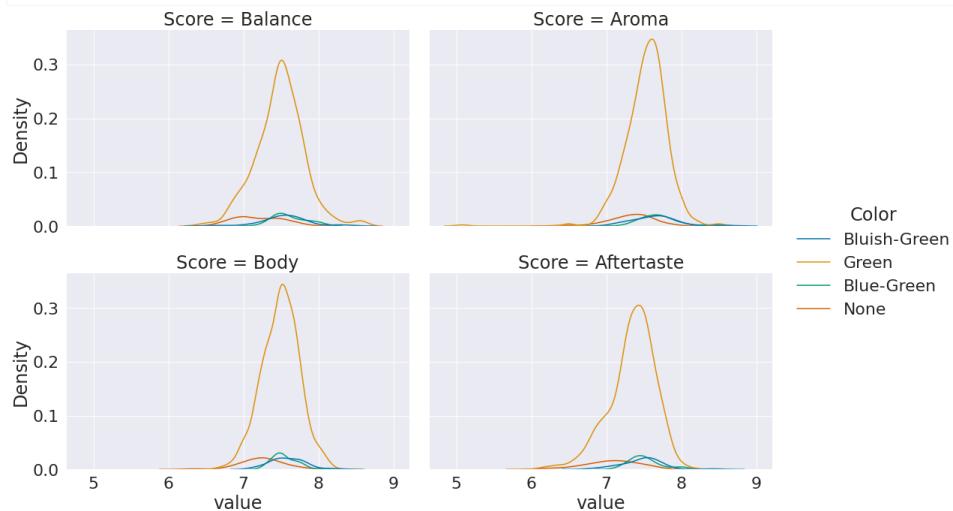
One thing is it makes plots easier, because seaborn is organized around tidy data.

```
sns.displot(data= high_prod_coffee_df_melted,
            x='value',hue='Country.of.Origin',
            col = 'Score', col_wrap=2, kind='kde',aspect =1.5)
```



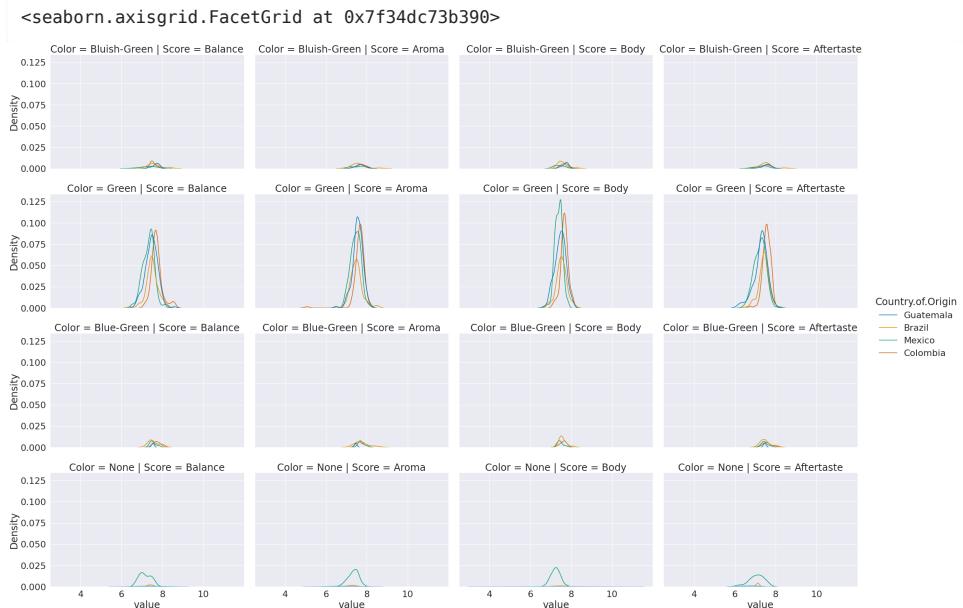
```
sns.displot(data= high_prod_coffee_df_melted,
            x='value',hue='Color',
            col = 'Score', col_wrap=2, kind='kde',aspect =1.5)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f34dca698d0>
```



```
sns.displot(data= high_prod_coffee_df_melted,  
            x='value',hue='Country.of-Origin',  
            col = 'Score', row='Color', kind='kde',aspect =1.5)
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/seaborn/distributions.py:316: UserWarning: Dataset has 0 variance; skipping  
density estimate. Pass `warn_singular=False` to disable this warning.  
    warnings.warn(msg, UserWarning)
```



10.1. 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
0	19285751557	PushEvent	{"id": 10656079, "login": "brownsarahm", "disp...	{"id": 400283911, "name": "rhodyprog4ds/BrownF...	{"push_id": 8593961606, "size": 3, "distinct_s...	True
1	19261725757	PushEvent	{"id": 41898282, "login": "github-actions[bot]..."}	{"id": 400283911, "name": "rhodyprog4ds/BrownF...	{"push_id": 8581966785, "size": 1, "distinct_s...	True
2	19261588410	PushEvent	{"id": 10656079, "login": "brownsarahm", "disp...	{"id": 400283911, "name": "rhodyprog4ds/BrownF...	{"push_id": 8581900859, "size": 1, "distinct_s...	True
3	19227505408	IssuesEvent	{"id": 10656079, "login": "brownsarahm", "disp...	{"id": 400283911, "name": "rhodyprog4ds/BrownF...	{"action": "opened", "issue": {"url": "https://...}}	True
4	19209647509	PushEvent	{"id": 41898282, "login": "github-actions[bot]..."}	{"id": 400283911, "name": "rhodyprog4ds/BrownF...	{"push_id": 8555694300, "size": 1, "distinct_s...	True

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_qb_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	
0	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
1	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
2	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
3	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
4	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
5	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
6	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
7	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
8	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
9	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
10	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
11	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
12	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
13	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
14	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
15	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
16	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
17	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
18	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
19	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
20	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
21	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
22	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
23	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
24	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
25	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
26	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
27	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
28	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
29	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions

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	
0	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
1	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
2	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
3	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
4	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
5	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
6	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
7	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
8	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
9	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
10	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
11	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
12	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
13	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
14	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
15	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
16	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
17	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
18	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
19	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
20	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
21	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
22	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
23	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
24	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
25	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm
26	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions
27	10656079	brownsarahm	brownsarahm		https://api.github.com/users/brownsarahm

	id	login	display_login	gravatar_id	
28	10656079	brownsarahm	brownsarahm		https://api.github.com/users/browns
29	41898282	github-actions[bot]	github-actions		https://api.github.com/users/github-actions

30 rows × 24 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, because 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, 24)

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, 17)

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.

Try it yourself

How could you anticipate how many are shared?

```
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	
0	10656079	brownsarahm	brownsarahm		https://api.github.com/
1	41898282	github-actions[bot]	github-actions		https://api.github.com/
2	10656079	brownsarahm	brownsarahm		https://api.github.com/
3	10656079	brownsarahm	brownsarahm		https://api.github.com/
4	41898282	github-actions[bot]	github-actions		https://api.github.com/
5	10656079	brownsarahm	brownsarahm		https://api.github.com/
6	10656079	brownsarahm	brownsarahm		https://api.github.com/
7	41898282	github-actions[bot]	github-actions		https://api.github.com/
8	10656079	brownsarahm	brownsarahm		https://api.github.com/
9	41898282	github-actions[bot]	github-actions		https://api.github.com/
10	10656079	brownsarahm	brownsarahm		https://api.github.com/
11	41898282	github-actions[bot]	github-actions		https://api.github.com/
12	10656079	brownsarahm	brownsarahm		https://api.github.com/
13	41898282	github-actions[bot]	github-actions		https://api.github.com/
14	10656079	brownsarahm	brownsarahm		https://api.github.com/
15	41898282	github-actions[bot]	github-actions		https://api.github.com/
16	10656079	brownsarahm	brownsarahm		https://api.github.com/
17	41898282	github-actions[bot]	github-actions		https://api.github.com/
18	10656079	brownsarahm	brownsarahm		https://api.github.com/
19	41898282	github-actions[bot]	github-actions		https://api.github.com/
20	10656079	brownsarahm	brownsarahm		https://api.github.com/
21	41898282	github-actions[bot]	github-actions		https://api.github.com/
22	10656079	brownsarahm	brownsarahm		https://api.github.com/
23	41898282	github-actions[bot]	github-actions		https://api.github.com/
24	10656079	brownsarahm	brownsarahm		https://api.github.com/
25	10656079	brownsarahm	brownsarahm		https://api.github.com/
26	41898282	github-actions[bot]	github-actions		https://api.github.com/
27	10656079	brownsarahm	brownsarahm		https://api.github.com/

```
actor_id  actor_login  actor_display_login  actor_gravatar_id  
28 10656079 brownsarahm      brownsarahm      https://api.github.com/  
29 41898282 github-  
actions[bot]      github-actions      https://api.github.com/  
30 rows × 24 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 column 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	
0	10656079	brownsarahm	brownsarahm		https://api.github.com/
1	41898282	github-actions[bot]	github-actions		https://api.github.com/
2	10656079	brownsarahm	brownsarahm		https://api.github.com/
3	10656079	brownsarahm	brownsarahm		https://api.github.com/
4	41898282	github-actions[bot]	github-actions		https://api.github.com/
5	10656079	brownsarahm	brownsarahm		https://api.github.com/
6	10656079	brownsarahm	brownsarahm		https://api.github.com/
7	41898282	github-actions[bot]	github-actions		https://api.github.com/
8	10656079	brownsarahm	brownsarahm		https://api.github.com/
9	41898282	github-actions[bot]	github-actions		https://api.github.com/
10	10656079	brownsarahm	brownsarahm		https://api.github.com/
11	41898282	github-actions[bot]	github-actions		https://api.github.com/
12	10656079	brownsarahm	brownsarahm		https://api.github.com/
13	41898282	github-actions[bot]	github-actions		https://api.github.com/
14	10656079	brownsarahm	brownsarahm		https://api.github.com/
15	41898282	github-actions[bot]	github-actions		https://api.github.com/
16	10656079	brownsarahm	brownsarahm		https://api.github.com/
17	41898282	github-actions[bot]	github-actions		https://api.github.com/
18	10656079	brownsarahm	brownsarahm		https://api.github.com/
19	41898282	github-actions[bot]	github-actions		https://api.github.com/
20	10656079	brownsarahm	brownsarahm		https://api.github.com/
21	41898282	github-actions[bot]	github-actions		https://api.github.com/
22	10656079	brownsarahm	brownsarahm		https://api.github.com/
23	41898282	github-actions[bot]	github-actions		https://api.github.com/
24	10656079	brownsarahm	brownsarahm		https://api.github.com/
25	10656079	brownsarahm	brownsarahm		https://api.github.com/
26	41898282	github-actions[bot]	github-actions		https://api.github.com/
27	10656079	brownsarahm	brownsarahm		https://api.github.com/

```

actor_id actor_login actor_display_login actor_gravatar_id

28 10656079 brownsarahm brownsarahm https://api.github.com/
29 41898282 github-actions[bot] github-actions https://api.github.com/

```

30 rows × 24 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 going 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	19285751557	PushEvent	True	2021-12-11 00:31:34+00:00
1	19261725757	PushEvent	True	2021-12-09 16:45:41+00:00
2	19261588410	PushEvent	True	2021-12-09 16:37:48+00:00
3	19227505408	IssuesEvent	True	2021-12-07 21:35:00+00:00
4	19209647509	PushEvent	True	2021-12-06 23:37:27+00:00
5	19209546719	PushEvent	True	2021-12-06 23:27:05+00:00
6	19207343390	IssuesEvent	True	2021-12-06 20:35:08+00:00
7	19178059387	PushEvent	True	2021-12-04 00:27:05+00:00
8	19178008248	PushEvent	True	2021-12-04 00:18:40+00:00
9	19176704023	PushEvent	True	2021-12-03 21:40:21+00:00
10	19176620043	PushEvent	True	2021-12-03 21:32:10+00:00
11	19176556056	PushEvent	True	2021-12-03 21:26:04+00:00
12	19176447579	PushEvent	True	2021-12-03 21:16:03+00:00
13	19160523235	PushEvent	True	2021-12-02 23:15:29+00:00
14	19160438759	PushEvent	True	2021-12-02 23:07:04+00:00
15	19142623934	PushEvent	True	2021-12-02 01:46:29+00:00
16	19142550433	PushEvent	True	2021-12-02 01:38:05+00:00
17	19103725812	PushEvent	True	2021-11-30 03:13:28+00:00
18	19103655728	PushEvent	True	2021-11-30 03:06:11+00:00
19	19043375185	PushEvent	True	2021-11-25 02:41:21+00:00
20	19043305902	PushEvent	True	2021-11-25 02:31:56+00:00
21	19006020128	PushEvent	True	2021-11-23 00:03:38+00:00
22	19005952720	PushEvent	True	2021-11-22 23:55:37+00:00
23	19004164679	PushEvent	True	2021-11-22 21:09:44+00:00
24	19004066700	PushEvent	True	2021-11-22 21:02:26+00:00
25	18999176284	IssuesEvent	True	2021-11-22 15:37:23+00:00
26	18976050422	PushEvent	True	2021-11-20 02:03:54+00:00
27	18976013436	PushEvent	True	2021-11-20 01:55:35+00:00
28	18953662477	IssueCommentEvent	True	2021-11-18 17:37:35+00:00
29	18940534872	PushEvent	True	2021-11-18 01:57:44+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	ε
0	19285751557	PushEvent	True	2021-12-11 00:31:34+00:00	10656079	brownsarahm	
1	19261725757	PushEvent	True	2021-12-09 16:45:41+00:00	41898282	github-actions[bot]	
2	19261588410	PushEvent	True	2021-12-09 16:37:48+00:00	10656079	brownsarahm	
3	19227505408	IssuesEvent	True	2021-12-07 21:35:00+00:00	10656079	brownsarahm	
4	19209647509	PushEvent	True	2021-12-06 23:37:27+00:00	41898282	github-actions[bot]	
5	19209546719	PushEvent	True	2021-12-06 23:27:05+00:00	10656079	brownsarahm	
6	19207343390	IssuesEvent	True	2021-12-06 20:35:08+00:00	10656079	brownsarahm	
7	19178059387	PushEvent	True	2021-12-04 00:27:05+00:00	41898282	github-actions[bot]	
8	19178008248	PushEvent	True	2021-12-04 00:18:40+00:00	10656079	brownsarahm	
9	19176704023	PushEvent	True	2021-12-03 21:40:21+00:00	41898282	github-actions[bot]	
10	19176620043	PushEvent	True	2021-12-03 21:32:10+00:00	10656079	brownsarahm	
11	19176556056	PushEvent	True	2021-12-03 21:26:04+00:00	41898282	github-actions[bot]	
12	19176447579	PushEvent	True	2021-12-03 21:16:03+00:00	10656079	brownsarahm	
13	19160523235	PushEvent	True	2021-12-02 23:15:29+00:00	41898282	github-actions[bot]	
14	19160438759	PushEvent	True	2021-12-02 23:07:04+00:00	10656079	brownsarahm	
15	19142623934	PushEvent	True	2021-12-02 01:46:29+00:00	41898282	github-actions[bot]	
16	19142550433	PushEvent	True	2021-12-02 01:38:05+00:00	10656079	brownsarahm	
17	19103725812	PushEvent	True	2021-11-30 03:13:28+00:00	41898282	github-actions[bot]	
18	19103655728	PushEvent	True	2021-11-30 03:06:11+00:00	10656079	brownsarahm	
19	19043375185	PushEvent	True	2021-11-25 02:41:21+00:00	41898282	github-actions[bot]	
20	19043305902	PushEvent	True	2021-11-25 02:31:56+00:00	10656079	brownsarahm	
21	19006020128	PushEvent	True	2021-11-23 00:03:38+00:00	41898282	github-actions[bot]	
22	19005952720	PushEvent	True	2021-11-22 23:55:37+00:00	10656079	brownsarahm	
23	19004164679	PushEvent	True	2021-11-22 21:09:44+00:00	41898282	github-actions[bot]	
24	19004066700	PushEvent	True	2021-11-22 21:02:26+00:00	10656079	brownsarahm	
25	18999176284	IssuesEvent	True	2021-11-22 15:37:23+00:00	10656079	brownsarahm	
26	18976050422	PushEvent	True	2021-11-20 02:03:54+00:00	41898282	github-actions[bot]	
27	18976013436	PushEvent	True	2021-11-20 01:55:35+00:00	10656079	brownsarahm	

	id	type	public	created_at	actor_id	actor_login	...
28	18953662477	IssueCommentEvent	True	2021-11-18 17:37:35+00:00	10656079	brownsarahm	
29	18940534872	PushEvent	True	2021-11-18 01:57:44+00:00	41898282	github-actions[bot]	

30 rows × 28 columns

To see how the list math works

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

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

results in one list

but without the `[]` we get a type error

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

```
-----  
TypeError: can only concatenate str (not "list") to str  
/tmp/ipykernel_1997/858457300.py in <module>  
----> 1 'a' + ['b', 'c', 'd']
```

```
TypeError: can only concatenate str (not "list") to str
```

List operations return `None` and mutate the list in place so

```
orig_list = ['a']
new_items = ['b', 'c', 'd']
orig_list.extend(new_items)
```

outputs nothing because `None` was returned and it changes the original variable.

```
orig_list
```

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

```
type(orig_list.extend(new_items))
```

```
NoneType
```

is none.

10.2. Questions After Class

All clarifying questions today

10.2.1. How does Axis work?

10.2.2. How does melt work?

10.2.3. What about the NaNs that are still left?

11. Missing Data and Inconsistent coding

```
import pandas as pd
import seaborn as sns
import numpy as np

sns.set_theme(palette= "colorblind")
na_toy_df = pd.DataFrame(data = [[1,3,4,5],[2 ,6, np.nan]])

# make plots look nicer and increase font size
sns.set_theme(font_scale=2,palette='colorblind')
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)
```

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. To see the types of things:

[Stanford Policy Lab Open Policing Project data readme](#) [Propublica Machine Bias](#) the “How we acquired data” section

11.1. Missing Values

Dealing with missing data is a whole research area. There isn't one solution.

[in 2020 there was a workshop on it](#)

There are also many classic approaches both when training and when [applying models](#).

[example application in breast cancer detection](#)

In pandas, even representing [missing values](#) is under [experimentation](#). Currently, it uses `numpy.NaN`, but the experiment is with `pd.NA`.

Missing values even causes the [datatypes to change](#)

Pandas gives a few basic tools:

- drop with (`dropna`)
- fill with `fillna`

```
coffee_df.head()
```

	Species	Owner	Country.of.Origin	Farm.Name	Lot.Number	M
0	1 Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc
1	2 Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc
2	3 Arabica	grounds for health admin	Guatemala	san marcos barrancas "san cristobal cuch	NaN	NaN
3	4 Arabica	yidnekachew dabessa	Ethiopia	yidnekachew dabessa coffee plantation	NaN	wolens
4	5 Arabica	metad plc	Ethiopia	metad plc	NaN	metad plc

5 rows × 44 columns

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	

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

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()
```

0	1
1	1
2	1
3	1
4	1
Name: Lot.Number, dtype: object	

```
coffee_df['Lot.Number'].head()
```

```
0    NaN  
1    NaN  
2    NaN  
3    NaN  
4    NaN  
Name: Lot.Number, dtype: object
```

💡 Tip

Note that even after we called `fillna` we display it again and the original data is unchanged.

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 reassigned the pointer. It's 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)
```

💡 Question in Class

When I use value counts it treats the filled ones as different. Why?

```
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
```

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

```
1                  1041  
1                   18  
020/17              6  
019/17              5  
102                 3  
..  
3-59-2318            1  
8885                1  
5055                1  
MCCFWXA15/16          1  
017-053-0211/ 017-053-0212  1  
Name: lot_number_clean, Length: 222, dtype: int64
```

If we switch to `1` as a string, then we'd see all of the one values as the same thing.

```
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
```

This was our goal, so in this case, it's the right thing to do to overwrite the value.

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.

To illustrate how `dropna` works, we'll use the `shape` method:

```
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.

```
coffee_df.dropna(subset=['altitude_low_meters']).shape
```

```
(1084, 45)
```

Note

subset operates along columns by default, because axis is set to 0, by default.

whatever you do, document it

Try it Yourself

use the `na_toy_df` DataFrame that's defined in the first cell, to experiment with subset and axis parameters to understand them better.

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.

```
n_rows, n_cols = coffee_df.shape
coffee_df.dropna(thresh=.7*n_rows, 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.

```
coffee_df.dropna(thresh=.85*n_rows, axis=1).shape
```

```
(1311, 34)
```

! Important

Everththing after this is new material that we did not have time for in class, but is important and helpful in your assignment (and for your portflio).

11.2. 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()
```

```

AMECAFE
205
Africa Fine Coffee Association
49
Almacafé
178
Asociacion Nacional Del Café
155
Asociación Mexicana De Cafés y Cafeterías De Especialidad A.C.
6
Asociación de Cafés Especiales de Nicaragua
8
Blossom Valley International
58
Blossom Valley International\n
1
Brazil Specialty Coffee Association
67
Central De Organizaciones Productoras De Café y Cacao Del Perú - Central Café & Cacao
1
Centro Agroecológico del Café A.C.
8
Coffee Quality Institute
7
Ethiopia Commodity Exchange
18
Instituto Hondureño del Café
60
Kenya Coffee Traders Association
22
METAD Agricultural Development plc
15
NUCOFFEE
36
Salvadoran Coffee Council
11
Specialty Coffee Ass
1
Specialty Coffee Association
295
Specialty Coffee Association of Costa Rica
42
Specialty Coffee Association of Indonesia
10
Specialty Coffee Institute of Asia
16
Tanzanian Coffee Board
6
Torch Coffee Lab Yunnan
2
Uganda Coffee Development Authority
22
Yunnan Coffee Exchange
12
Name: In.Country.Partner, dtype: int64

```

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()

```

```

AMECAFE
205
Africa Fine Coffee Association
49
Almacafé
178
Asociacion Nacional Del Café
155
Asociación Mexicana De Cafés y Cafeterías De Especialidad A.C.
6
Asociación de Cafés Especiales de Nicaragua
8
Blossom Valley International
59
Brazil Specialty Coffee Association
67
Central De Organizaciones Productoras De Café y Cacao Del Perú - Central Café & Cacao
1
Centro Agroecológico del Café A.C.
8
Coffee Quality Institute
7
Ethiopia Commodity Exchange
18
Instituto Hondureño del Café
60
Kenya Coffee Traders Association
22
METAD Agricultural Development plc
15
NUCOFFEE
36
Salvadoran Coffee Council
11
Specialty Coffee Association
296
Specialty Coffee Association of Costa Rica
42
Specialty Coffee Association of Indonesia
10
Specialty Coffee Institute of Asia
16
Tanzanian Coffee Board
6
Torch Coffee Lab Yunnan
2
Uganda Coffee Development Authority
22
Yunnan Coffee Exchange
12
Name: in_country_partner_clean, dtype: int64

```

and now we see the corrected values. We can also pass lambdas or put lambdas in the dictionary if there are systemic patterns.

11.3. Fixing data at load time

Explore some of the different parameters in [read_csv](#)

How can we read in data that looks like this:

Ethnicity	Asian				Black				Hispanic			
	Female		Male		Female		Male		Female		Male	
	count	mean	count	mean	count	mean	count	mean	count	mean	count	mean
Gender												
Age Cohort												
0 - 5	6	1544.33333							18	1431.33333	26	1366
51 +	-	-	-	-	-	-	-	-	-	-	-	-

```
pd.read_csv('fancy_formatting.xlsx', header = list(range(4)))
```

Many problems can be repaired with parameters in [read_csv](#).

11.4. 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 structured 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?

Things to keep in mind:

- always save new copies of data when you mutate it
- add new columns rather than overwriting columns
- long variable names are better than ambiguous naming

11.5. Your observations from Monday:

I promised we'd come back to your observations on what problems could occur in data. Here they are, organized by rough categories of when/how to fix them.

We can fix while reading in data:

- decimal was indicated with ',' instead of '.' so pandas saw value as a string rather than a float
- missing header
- reading the index as a column
- large datasets might be too slow or not fit in memory
- missing data represented with a value or special character

We can fix by reshaping data:

- Data can get read into tables in bizarre ways depending on how the data was entered originally.
- every value in one column, instead of separated

We can repair by changing values or filtering:

- information represented inconsistently eg "Value" and " Value " or twenty-two instead of 22
- blank rows or blank columns or data that is N/A
- date/time information can be represented lots of different ways
- representing categorical with numbers that are ambiguous
- spaces or other symbols in column names
- some numbers as strings, others as ints within a column
- symbols being mis interpreted

Real problems, but beyond our scope:

- corrupt data files

11.6. More Practice

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](#).

12. Building Datasets From multiple Sources

```
1. remember, that  
1. with markdown  
1. you can make a numbered list  
1. using all `1.`
```

renders as:

1. remember, that
2. with markdown
3. you can make a numbered list
4. using all 1.

```
import pandas as pd  
  
course_data_url = 'https://raw.githubusercontent.com/rhodyprog4ds/rhodyds/main/data/'
```

Today we're going to look at some data on NBA (National Basketball Association) players.

12.1. Combining Multiple Tables

```
p18 = pd.read_csv(course_data_url + '2018-players.csv')  
p19 = pd.read_csv(course_data_url + '2019-players.csv')
```

```
p18.head()
```

	TEAM_ID	PLAYER_ID	SEASON
0	1610612761	202695	2018
1	1610612761	1627783	2018
2	1610612761	201188	2018
3	1610612761	201980	2018
4	1610612761	200768	2018

12.1.1. Stacking with Concat

We've seen one way of putting dataframes together with `concat` we used it when [Unpacking Jsons](#). In that case, we stacked them side by side, now we want to stack them vertically, which is the default of concat.

```
players_df = pd.concat([p18,p19])  
players_df.head()
```

	TEAM_ID	PLAYER_ID	SEASON	PLAYER_NAME
0	1610612761	202695	2018	NaN
1	1610612761	1627783	2018	NaN
2	1610612761	201188	2018	NaN
3	1610612761	201980	2018	NaN
4	1610612761	200768	2018	NaN

This has the same columns, and we can see what happened more by looking at the shape of each.

```
players_df.shape
```

```
(1374, 4)
```

```
p18.shape, p19.shape
```

```
((748, 3), (626, 4))
```

We can verify that the length of the new data frame is the sum of the original two DataFrames.

```
assert len(p18) + len(p19) == len(players_df)
```

12.1.2. Combining Data with Merge

What is we want to see which players changed teams from 2018 to 2019? We can do this with `merge`. For `merge` we have to tell it a left DataFrame and a right DataFrame and on what column to match them up.

The left and right DataFrames will be used different ways, but any DataFrame can be put in either position.

For this case, we will use 2018 data as left ,and 2019 as right and then merge `on='PLAYER_ID'`.

```
pd.merge(p18,p19,on='PLAYER_ID').head()
```

	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
2	1610612761	201188	2018	Marc Gasol	1610612761	2019
3	1610612763	201188	2018	Marc Gasol	1610612761	2019
4	1610612761	201980	2018	Danny Green	1610612747	2019

Now, we get what we expect, but the column names have `_x` and `_y` on the end (as a suffix, appended to the original). We'll add 2018 and 2019 respectively, separated with a `_`.

```
year_over_year = pd.merge(p18,p19,on='PLAYER_ID',suffixes=('_2018','_2019'))
year_over_year.head()
```

	TEAM_ID_2018	PLAYER_ID	SEASON_2018	PLAYER_NAME	TEAM_ID_2019	SEASO
0	1610612761	202695	2018	Kawhi Leonard	1610612746	
1	1610612761	1627783	2018	Pascal Siakam	1610612761	
2	1610612761	201188	2018	Marc Gasol	1610612761	
3	1610612763	201188	2018	Marc Gasol	1610612761	
4	1610612761	201980	2018	Danny Green	1610612747	

Now that it's a little bit cleaner, we will examine how it works by looking at the shape.

```
year_over_year.shape, p18.shape, p19.shape
```

```
((538, 6), (748, 3), (626, 4))
```

This kept only the players that played both years, with repetitions for each team they played on for each year.

We can check the calculation with set math (python `set` type has operations from math sets like intersect and set difference)

```

# player IDs for each year, no repeats
p19_u = set(p19['PLAYER_ID'])
p18_u = set(p18['PLAYER_ID'])

# player IDs that played both years
p1819 = p18_u.intersection(p19_u)

# teams per player per year
teams_per_p18 = p18['PLAYER_ID'].value_counts()
teams_per_p19 = p19['PLAYER_ID'].value_counts()

# total number of team-player combinations
# multiply number of teams each player played for in 18 by number of teams in 19
# then sum. (most of these are 1*1)
sum(teams_per_p19[p1819]* teams_per_p18[p1819])

```

538

We can also merge so that we keep all players on either team using `how='outer'` the default value for `how` is inner, which takes the intersection (but with duplicates, does some extra things as we saw). With `outer` it takes the union, but with extra handling for the duplicates.

```
pd.merge(p18,p19,on='PLAYER_ID',suffixes=('_2018','_2019'),how='outer').shape
```

(927, 6)

It's the total of the rows we had before, plus the total number of player-teams for players that only played in one of the two years.

```

#players tha tonly played in one year
o18 = p18_u.difference(p19_u)
o19 = p19_u.difference(p18_u)
# teams those players played for + the above 538
teams_per_p19[o19].sum() + teams_per_p18[o18].sum() + sum(teams_per_p19[p1819]*
teams_per_p18[p1819])

```

927

We can save this to a variable

```
year_over_year_outer = pd.merge(p18,p19,on='PLAYER_ID',suffixes=('_2018','_2019'),how='outer')
```

then look at a few rows to see that it has indeed filled in with NaN in the places where there wasn't a value.

```
year_over_year_outer.sample(10)
```

	TEAM_ID_2018	PLAYER_ID	SEASON_2018	PLAYER_NAME	TEAM_ID_2019	SEAS
717	1.610613e+09	202730	2018.0	NaN	NaN	
263	1.610613e+09	1628989	2018.0	Kevin Huerter	1.610613e+09	
82	1.610613e+09	202710	2018.0	Jimmy Butler	1.610613e+09	
328	1.610613e+09	1627885	2018.0	Shaquille Harrison	1.610613e+09	
384	1.610613e+09	203967	2018.0	Dario Saric	1.610613e+09	
32	1.610613e+09	1628449	2018.0	Chris Boucher	1.610613e+09	
383	1.610613e+09	203952	2018.0	Andrew Wiggins	1.610613e+09	
70	1.610613e+09	203999	2018.0	Nikola Jokic	1.610613e+09	
76	1.610613e+09	1628420	2018.0	Monte Morris	1.610613e+09	
851	NaN	1629967	NaN	Skyler Flatten	1.610613e+09	

Note

We can also tell that there are NaN because it cast the year to float from int.

12.2. Merge types, in detail

We can examine how these things work more visually with smaller DataFrames:

```
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"], how='inner')
```

	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' )
```

Note

inner is default, but we can state it to be more explicit

	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

12.3. Duplicate Keys

```
left = pd.DataFrame({"A": [1, 2], "B": [2, 2]})

right = pd.DataFrame({"A": [4, 5, 6], "B": [2, 2, 2]})

result = pd.merge(left, right, on="B", how="outer")
```

left

	A	B
0	1	2
1	2	2

right

	A	B
0	4	2
1	5	2
2	6	2

result

	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

If we ask pandas to validate the merge with `validate` and a specific type of merge, it will throw an error if that type of merge is not possible.

```
pd.merge(left, right, on='B', validate='one_to_one')
```

```

-----  

MergeError                                     Traceback (most recent call last)  

/tmp/ipykernel_2055/2916808787.py in <module>  
----> 1 pd.merge(left, right, on='B', validate='one_to_one')  
  

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/pandas/core/reshape/merge.py in merge(left, right, how, on, left_on, right_on,  
left_index, right_index, sort, suffixes, copy, indicator, validate)  
    117         copy=copy,  
    118         indicator=indicator,  
--> 119         validate=validate,  
    120     )  
    121     return op.get_result()  
  

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/pandas/core/reshape/merge.py in __init__(self, left, right, how, on, left_on,  
right_on, axis, left_index, right_index, sort, suffixes, copy, indicator, validate)  
    707         # are in fact unique.  
    708         if validate is not None:  
--> 709             self._validate(validate)  
    710  
    711     def get_result(self) -> DataFrame:  
  

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-  
packages/pandas/core/reshape/merge.py in _validate(self, validate)  
    1419         if not left_unique and not right_unique:  
    1420             raise MergeError(  
-> 1421                 "Merge keys are not unique in either left "  
    1422                 "or right dataset; not a one-to-one merge"  
    1423             )  
  
MergeError: Merge keys are not unique in either left or right dataset; not a one-to-one  
merge

```

Further Reading

The [pandas documentation](#) is a good place to read through their examples on how validate works. It's important to note the types of possible joins from the [beginning of the section](#)

12.4. Questions at the End of Class

12.4.1. How to remove certain columns in merges (i.e. the year columns in the basketball dataset since they are redundant)

12.4.2. Can we merge as many dataframes as we would want?

12.4.3. Is there no way to merge the left and right for the A and B data?

12.4.4. Can you use merging to check correlation between two different sets?

12.4.5. Is there any way to compare the team ids in both datasets so that it outputs the players that changed teams between those times?

```

change_check = lambda r: not(r['TEAM_ID_2018']==r['TEAM_ID_2019'])
year_over_year_clean['CHANGE_TEAM'] = year_over_year_clean.apply(change_check, axis=1)
year_over_year_clean.head()

```

```

NameError Traceback (most recent call last)
/tmp/ipykernel_2055/805718336.py in <module>
     1 change_check = lambda r: not(r['TEAM_ID_2018']==r['TEAM_ID_2019'])
----> 2 year_over_year_clean['CHANGE_TEAM'] =
year_over_year_clean.apply(change_check, axis=1)
     3 year_over_year_clean.head()

NameError: name 'year_over_year_clean' is not defined

```

Then we can filter the data by the new column, then take out only the player information and drop any duplicates for players that played in multiple teams in one year or the other.

```

y_o_y_changes = year_over_year_clean[year_over_year_clean['CHANGE_TEAM']]
y_o_y_change_players = y_o_y_changes[['PLAYER_ID', 'PLAYER_NAME']].drop_duplicates()
y_o_y_change_players.head()

```

```

NameError Traceback (most recent call last)
/tmp/ipykernel_2055/1142329717.py in <module>
----> 1 y_o_y_changes = year_over_year_clean[year_over_year_clean['CHANGE_TEAM']]
     2 y_o_y_change_players =
y_o_y_changes[['PLAYER_ID', 'PLAYER_NAME']].drop_duplicates()
     3 y_o_y_change_players.head()

NameError: name 'year_over_year_clean' is not defined

```

and we can check how many players that is

```
y_o_y_change_players.shape
```

```

NameError Traceback (most recent call last)
/tmp/ipykernel_2055/2092251871.py in <module>
----> 1 y_o_y_change_players.shape

NameError: name 'y_o_y_change_players' is not defined

```

12.4.6. in `year_over_year`, where `suffixes=('_2018','_2019')`, why is there an underscore?

12.4.7. If we merge two dataframes which each have a column (or columns) that are the same, but each have different data types in those columns, what happens?

12.4.8. how does suffix to know to change the date?

12.5. More Practice

1. Use a merge to figure out how many players did not return for the 2019 season
2. Also load the `conferences.csv` data. Use this to figure out how many players moved conferences from 2018 to 2019.

13. Reviewing Merges & Databases

```
import pandas as pd
```

```
pd.merge?
```

a review problem

If `weather_df` has a row for every date in the past 100 years, a column named `date` and columns for the weather that day and `birthdays_df` has a row for each member of the CS department with a column `birthdate` and a second column name, how would you merge these two datasets to find out the weather on the day each person was born?

```
bday_weather = pd.merge(left= weather_df, right = birthdays_df,  
    left_on = 'date', right_on ='birthdate', how= 'inner')
```

13.1. Working with Databases

Off the shelf, pandas cannot read databases by default. We'll use the [sqlite3](#) library.

```
import sqlite3
```

Note

[SQLite](#) is a type of database.

First, we set up a connection, that links the the notebook to the database.

```
conn = sqlite3.connect('data/nba1819.db')
```

```
conn
```

```
<sqlite3.Connection at 0x7f1647398d50>
```

To use it, we add a cursor.

```
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 0x7f164728ab90>
```

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',)]
```

We can also pass queries to the database through pandas and then get it returned as a DataFrame.

```
pd.read_sql('SELECT PLAYER_ID, PLAYER_NAME FROM playerNames', conn).head(1)
```

```
PLAYER_ID  PLAYER_NAME
```

```
0      202695  Kawhi Leonard
```

We can use `*` to get all of the columns and `LIMIT` to reduce the number of rows.

```
pd.read_sql('SELECT * FROM playerNames LIMIT 5',conn)
```

	index	PLAYER_NAME	PLAYER_ID
0	0	Kawhi Leonard	202695
1	1	Pascal Siakam	1627783
2	2	Marc Gasol	201188
3	3	Danny Green	201980
4	4	Kyle Lowry	200768

How can get all of the data from the teams table?

```
teams_df = pd.read_sql('SELECT * FROM teams',conn)
teams_df.head()
```

	index	LEAGUE_ID	TEAM_ID	MIN_YEAR	MAX_YEAR	ABBREVIATION	NICKNAME
0	0	0	1610612737	1949	2019	ATL	Hawks
1	1	0	1610612738	1946	2019	BOS	Celtics
2	2	0	1610612740	2002	2019	NOP	Pelicans
3	3	0	1610612741	1966	2019	CHI	Bulls
4	4	0	1610612742	1980	2019	DAL	Mavericks

13.2. More practice

For each of the following, think about what to query from the database and how to merge the tables after. Also, consider if these questions are all specific enough or require more decisions to be made.

1. How many players changed from the `East` conference to the `West` conference during the 2018 season?
2. How many from the `East` conference to the `West` conference from the 2018 season to the 2019 season?
3. Did teams that were founded earlier have better records in the 2018 season?

💡 Think ahead

For a portfolio, you could ask questions like do events in a city impact if the home team wins later that week?

14. Web Scraping

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

Tip

To update a package while running jupyter in a notebook:

```
!pip install packagename update
```

then restart the kernel

14.1. Figuring out what to scrape

We're going to create a DataFrame about URI CS & Statistics Faculty.

from the [people page](#) of the department website.

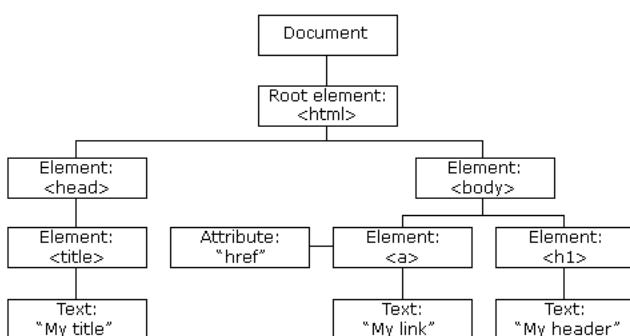
With great power comes great responsibility.

- always check [robots.txt](#)
- do not do things that the owner says not to do
- government websites are typically safe, because of open data rules

We can inspect the page to check that it's well structured by right clicking in a browser tab and looking at the same code that our browser sees in order to render the page.

We can basically think of web scraping as loading data that's *not* tabular but instead is formatted as html code. HTML code *can* be well strucutred and hierarchical within a single page, or you could collect information about a broad topic by using a little bit from many many pages. We're going to work from one page here.

HTML code consists of tags and the text of the page. The tags label the content and define different structure of the page.



Think Ahead

You can use this same basic logic, that anything can be data to consider other questions like, for example:

- How do the sizes of datasets for Tidy Tuesday vary? (you don't need to download and load all of the data, only traverse the readmes)
- On average, how many code cells are there in each class? How much does the amount of code in the posted notes vary from your notes you take in class?

Once we've decided that it will work, we can begin working.

```
cs_people_url = 'https://web.uri.edu/cs/people/'
```

14.2. Loading with requests

First, we [get](#) the data using the python [requests](#) library.

```
requests.get(cs_people_url)
```

```
<Response [200]>
```

this returns an object, but we want the content from it, so we'll save that to a variable

```
cs_people_html = requests.get(cs_people_url).content  
cs_people_html
```


[Sarah Brown](https://web.uri.edu/cs/files/Sarah-Brown.png)

[Mike Conti](https://web.uri.edu/cs/files/mike-conti-scaled.jpg)

[Noah Daniels](https://web.uri.edu/cs/files/noah-daniels-web.png)

[Lisa DiPippo](https://web.uri.edu/cs/files/lisa-dipippo-web.jpg)

[Victor Fay-Wolfe](https://web.uri.edu/cs/files/victor-fay-wolfe-web.jpg)

[Abdelatab Hendawi](https://web.uri.edu/cs/files/abdelatab-hendawi/)

style="clear:both;"></div>\n<t></div>\n<div class="peopleitem h-card has-thumbnail">\n<t><header>\n<t><div class="header">\n<t><t><t><figure>\n<t><t><t><t><t><t>\n<t><t><t><t><figure>\n<t><t><n><t><t><t><h3 class="p-name">Jean-Yves Hervé<x3>xa9</h3>\n<t><t><n><t><t><t><div>\n<t><header>\n<t><div class="inside">\n<n><t><div class="people-item p-job-title">Associate Professor</p>\n<t><t><n><t><t><p class="people-department">Computer Science</p>\n<t><t><n><t><t><t><p class="people-misc">jyh@cs.uri.edu</p>\n<t><t><n><t><div style="clear:both;"></div>\n<t><div>\n<div class="peopleitem h-card has-thumbnail">\n<t><header>\n<t><div class="header">\n<t><t><t><figure>\n<t><t><t><t><t><t>\n<t><t><t><t><figure>\n<t><t><n><t><t><t><h3 class="p-name">Natalia Katenka</h3>\n<t><t><n><t><t><t><div>\n<t><header>\n<t><div class="inside">\n<n><t><t><p class="people-item 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><t><t><t><p class="people-misc">nkatenka@uri.edu</p>\n<t><t><n><t><div style="clear:both;"></div>\n<t><div class="inside">\n<n><t><t><div class="header">\n<t><t><t><t><t><h3 class="p-name">Soheyb Kouider</h3>\n<t><t><t><t><n><t><div>\n<t><header>\n<t><div class="inside">\n<n><t><t><p class="people-item p-job-title">Lecturer</p>\n<t><t><n><t><t><t><p class="people-department">Statistics</p>\n<t><t><n><t><t><t><t><p class="people-misc">401.874.2562 soheyb@uri.edu</p>\n<t><t><n><t><div style="clear:both;"></div>\n<t><div>\n<div class="peopleitem h-card has-thumbnail">\n<t><header>\n<t><div class="header">\n<t><t><t><figure>\n<t><t><t><t><t><t>\n<t><t><t><t><figure>\n<t><t><t><t><t><t><h3 class="p-name">Edmund Lamagna</h3>\n<t><t><t><t><n><t><div>\n<t><header>\n<t><div class="inside">\n<n><t><t><p class="people-item 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><t><t><p class="people-misc">eal@cs.uri.edu</p>\n<t><t><n><t><div style="clear:both;"></div>\n<t><div>\n<div class="peopleitem h-card has-thumbnail">\n<t><header>\n<t><div class="header">\n<t><t><t><figure>\n<t><t><t><t><t><t>\n<t><t><t><t><figure>\n<t><t><t><t><t><t><h3 class="p-name">Indrani Mandal</h3>\n<t><t><t><t><n><t><div>\n<t><header>\n<t><div class="inside">\n<n><t><t><p class="people-item p-job-title">Lecturer</p>\n<t><t><n><t><t><t><p class="people-department">Computer Science</p>\n<t><t><n><t><t><t><t><p class="people-misc">indrani_mandal@uri.edu</p>\n<t><t><n><t><div style="clear:both;"></div>\n<t><div>\n<div class="peopleitem h-card has-thumbnail">\n<t><header>\n<t><div class="header">\n<t><t><t><figure>\n<t><t><t><t><t><t>\n<t><t><t><t><figure>\n<t><t><n><t><t><t><h3 class="p-name">Gavino Puggioni</h3>\n<t><t><t><t><n><t><div>\n<t><header>\n<t><div class="inside">\n<n><t><t><p class="people-item p-job-title">Associate Professor | Statistics Section Head | Director of Graduate Studies</p>\n<t><t><n><t><t><t><t><p class="people-misc">401.874.4388 gppuggioni@uri.edu</p>\n<t><t><n><t><div style="clear:both;"></div>\n<t><div>\n<div class="peopleitem h-card has-thumbnail">\n<t><header>\n<t><div class="header">\n<t><t><t><figure>\n<t><t><t><t><t><t><a

[Joseph Squillace](https://web.uri.edu/cs/meet/joseph-squillace/)

A portrait of Joseph Squillace, a man with short dark hair, wearing a light-colored shirt and a dark tie. He is smiling at the camera.

[Krishna Venkatasubramanian](https://web.uri.edu/cs/meet/krishna-venkatasubramanian/)

A portrait of Krishna Venkatasubramanian, a man with dark hair, wearing a light-colored shirt and a dark tie. He is smiling at the camera.

[Jing Wu](https://web.uri.edu/cs/meet/jing-wu/)

A portrait of Jing Wu, a woman with dark hair, wearing a light-colored shirt and a dark tie. She is smiling at the camera.

[Yichi Zhang](https://web.uri.edu/cs/meet/yichi-zhang/)

A portrait of Yichi Zhang, a woman with dark hair, wearing a light-colored shirt and a dark tie. She is smiling at the camera.

[Guangyu Zhu](https://web.uri.edu/cs/meet/guangyu-zhu/)

A portrait of Guangyu Zhu, a woman with dark hair, wearing a light-colored shirt and a dark tie. She is smiling at the camera.

[Ashley Buchanan](https://web.uri.edu/cs/meet/ashley-buchanan/)

A portrait of Ashley Buchanan, a woman with dark hair, wearing a light-colored shirt and a dark tie. She is smiling at the camera.

This is literally just the html as a browser would see, but we pulled it into python.

14.3. Parsing with BeautifulSoup

Next, we'll use BeautifulSoup to parse the text. Parsing means to make sense of it. In this case, it transforms from a string or characters, to a datastructure that we can work with.

```
cs_people = BeautifulSoup(cs_people_html, 'html.parser')
```

First we note that it now formats the new lines.

cs_people

```
<!DOCTYPE html>

<html lang="en-US">
<head>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<link href="http://gmpg.org/xfn/11" rel="profile"/>
<title>People – Department of Computer Science and Statistics</title>
<meta content="max-image-preview:large" name="robots">
<link href="//s.w.org" rel="dns-prefetch">
<link href="https://web.uri.cs/feed/" rel="alternate" title="Department of Computer Science and Statistics » Feed" type="application/rss+xml"/>
<link href="https://web.uri.cs/comments/feed/" rel="alternate" title="Department of Computer Science and Statistics » Comments Feed" type="application/rss+xml"/>
<script type="text/javascript">
    window._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.cs/wp-includes/js/wp-emoji-release.min.js?ver=5.7.1"}};
    !function(e,a,t){var
n,r,o,i=a.createElement("canvas"),p=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()}funct
ion c(e){var
t=a.createElement("script");t.src=e,t.defer=t.type="text/javascript",a.getElementsByTagName
me("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]]=function(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:[55356,56826,55356,56819],
[55356,56826,8203,55356,56819])&&!s([55356,57332,56128,56423,56128,56418,56128,56421,5612
```

```
8,56430,56128,56423,56128,56447],  
[55356,57332,8203,56128,56423,8203,56418,8203,56128,56421,8203,56128,56430,8203,561  
28,56423,8203,56128,56447]);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.sup  
ports.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("DOMContentLoadered",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);  
        </script>  
<style type="text/css">  
img.wp-smiley,  
img.emoji {  
    display: inline !important;  
    border: none !important;  
    box-shadow: none !important;  
    height: 1em !important;  
    width: 1em !important;  
    margin: 0 .07em !important;  
    vertical-align: -0.1em !important;  
    background: none !important;  
    padding: 0 !important;  
}  
</style>  
<link href="https://web.uri.edu/cs/wp-includes/css/dist/block-library/style.min.css?  
ver=5.7.1" id="wp-block-library-css" media="all" rel="stylesheet" type="text/css"/>  
<link href="https://web.uri.edu/cs/wp-content/plugins/uri-component-  
library/css/cl.built.css?ver=5.0.2" id="uricl-css-css" media="all" rel="stylesheet"  
type="text/css"/>  
<link href="https://web.uri.edu/cs/wp-content/plugins/uri-courses/assets/courses.css?  
ver=5.7.1" id="uri-courses-styles-css" media="all" rel="stylesheet" type="text/css"/>  
<link href="https://web.uri.edu/cs/wp-content/plugins/uri-people-tool/assets/people.css?  
ver=5.7.1" id="uri-people-styles-css" media="all" rel="stylesheet" type="text/css"/>  
<link href="https://web.uri.edu/cs/wp-content/plugins/wp-lightbox-  
2/styles/lightbox.min.css?ver=1.3.4" id="wp-lightbox-2.min.css-css" media="all"  
rel="stylesheet" type="text/css"/>  
<link href="https://web.uri.edu/cs/wp-content/themes/uri-modern/style.css?ver=2.4.0"  
id="uri-modern-style-css" media="all" rel="stylesheet" type="text/css"/>  
<link href="https://web.uri.edu/cs/wp-content/plugins/tablepress/css/default.min.css?  
ver=1.13" id="tablepress-default-css" media="all" rel="stylesheet" type="text/css"/>  
<script id="jquery-core-js" src="https://web.uri.edu/cs/wp-  
includes/js/jquery/jquery.min.js?ver=3.5.1" type="text/javascript"></script>  
<script id="jquery-migrate-js" src="https://web.uri.edu/cs/wp-includes/js/jquery/jquery-  
migrate.min.js?ver=3.3.2" type="text/javascript"></script>  
<link href="https://web.uri.edu/cs/wp-json/" rel="https://api.w.org/"><link  
href="https://web.uri.edu/cs/wp-json/v2/pages/629" rel="alternate"  
type="application/json"/><link href="https://web.uri.edu/cs/xmlrpc.php?rsd" rel="EditURI"  
title="RSD" type="application/rsd+xml"/>  
<link href="https://web.uri.edu/cs/wp-includes/wlwmanifest.xml" rel="wlwmanifest"  
type="application/wlwmanifest+xml"/>  
<meta content="WordPress 5.7.1" name="generator">  
<link href="https://web.uri.edu/cs/people/" rel="canonical"/>  
<link href="https://web.uri.edu/cs/?p=629" rel="shortlink"/>  
<link href="https://web.uri.edu/cs/wp-json/oembed/1.0/embed?  
url=https%3A%2F%2Fweb.uri.edu%2Fpeople%2F" rel="alternate"  
type="application/json+oembed"/>  
<link href="https://web.uri.edu/cs/wp-json/oembed/1.0/embed?  
url=https%3A%2F%2Fweb.uri.edu%2Fpeople%2F&format=xml" rel="alternate"  
type="text/xml+oembed"/>  
<meta content="People Full-time Faculty Adjunct Faculty and Limited Join Appointments"  
name="description"/>  
<meta content="summary_large_image" name="twitter:card"/>  
<meta content="@universityofrri" name="twitter:site"/>  
<meta content="@universityofrri" name="twitter:creator"/>  
<meta content="https://web.uri.edu/cs/people/" property="og:url"/>  
<meta content="People" property="og:title"/>  
<meta content="People Full-time Faculty Adjunct Faculty and Limited Join Appointments"  
property="og:description"/>  
<meta content="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/logo-  
wordmark.png" property="og:image"/>  
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':  
new Date().getTime(),event:'gtm.js'});var f=d.createElementByTagName(s)[0],  
j=d.createElement(s),dl=l?'&l='+l:'';j.async=true;j.src=  
'https://www.googletagmanager.com/gtm.js?id='+i+dl+f.parentNode.insertBefore(j,f),  
})(window,document,'script','dataLayer','GTM-K5GL9W');</script>  
<style id="wp-custom-css" type="text/css">  
    .button-list .cl-button {  
        margin-bottom: 1rem;  
    }  
  
#calendar .date,#calendar-wrap header {  
    text-align:center
```

```
}

#calendar {
    width:100%
}
#calendar a {
    color:#005eff;
    text-decoration:none
}
#calendar ul {
    list-style:none;
    padding:0;
    margin:0;
    width:100%
}
#calendar li {
    display:block;
    float:left;
    width:14.342%;
    padding:5px;
    box-sizing:border-box;
    border:1px solid #ccc;
    margin-right:-1px;
    margin-bottom:-1px
}
#calendar ul.weekdays {
    height:40px;
    background:#002147
}
#calendar ul.weekdays li {
    text-align:center;
    text-transform:uppercase;
    line-height:1.2;
    border:none!important;
    padding:.75rem .5rem;
    color:#fff;
    font-size:.9rem
}
#calendar ul.days.mobile {
    display:none
}
#calendar .days li {
    height:15rem
}
#calendar .days.events-zero li {
    height:10rem
}
#calendar .days.events-one li {
    height:10.75rem
}
#calendar .days.events-two li {
    height:11.5rem
}
#calendar .days.events-three li {
    height:15rem
}
#calendar .date {
    margin-bottom:5px;
    padding:4px;
    color:#000;
    padding-right: 1rem;
    float:right;
}
#calendar .event {
    clear:both;
    display:block;
    font-size: 1rem;
    font-family: hind,arial,sans-serif;
    border-radius:30px;
    padding:.4rem .25rem .25rem 5rem;
    margin-top:5px;
    margin-bottom:5px;
    line-height:1.75;
    background:#e4f2f2;
    text-decoration:none;
    float:left;
    position:relative;
    width:calc(100% * 4.68)
}
#calendar .event.four-day{
    width:calc(76% * 4.68)
}
#calendar .event.mobile {
    display:none
}
#calendar .event.scratch {
    background:#E2D58B;
}
```

```

.calendar .event.lego {
    background:#E1EDE8;
}
.calendar .event.datacence {
    background:#FCB97D;
}
.calendar .event.girlstech {
    background:#C2CFB2;
}
.calendar .event.games {
    background:#F79365;
}
.calendar .event.programming {
    background:#BCDCE0;
}
.calendar .event.webdesign {
    background:#E09FA2;
}
.calendar .event-desc {
    color:#666;
    margin:3px 0 7px;
    text-decoration:none;
    display:inline-block
}
.calendar .event-time {
    margin:3px 0 7px 5px;
    text-decoration:none;
    display:inline-block
}
.calendar .other-month {
    background:#f5f5f5;
    color:#666
}
.desc {
    display:none;
    background:#eee;
    box-shadow: 0 0 4px #999;
    padding:10px;
    color: #000;
}
.calendar .event-desc:hover+desc {
    display:block;
    position:absolute;
    z-index:2
}
.days+header h1 {
    padding-top:2rem;
    clear: left
}
@media(max-width:768px) {
    .calendar img,#calendar .other-month,#calendar .weekdays {
        display:none
    }
    .calendar .event {
        width:calc(100% - 2em);
        padding-left:20px;
        padding-right:10px
    }
    .calendar .event.mobile,#calendar ul.days.mobile {
        display:block
    }
    .calendar li {
        height:auto!important;
        border:1px solid #eddede;
        width:100%;
        padding:10px;
        margin-bottom:-1px
    }
    .calendar .date {
        float:none
    }
}

.peopleitem.has-thumbnail img {
    width: 200px;
}

```

</style>

<!-- Favicons -->

<link color="#005eff" href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/safari-pinned-tab.svg" rel="mask-icon"/>

<link href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/favicon.png" rel="icon" type="image/png"/>

<link href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/apple-touch-icon.png" rel="apple-touch-icon"/>

<link href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/apple-touch-icon-180x180.png" rel="apple-touch-icon" sizes="180x180"/>

```
</meta></link></meta></head>
<body class="page-template-default page page-id-629 page-parent group-blog ln-people">
<noscript><iframe height="0" src="https://www.googletagmanager.com/ns.html?id=GTM-K5GL9W">
<div style="display:none;visibility:hidden" width="0"></iframe></noscript>
<div class="site" id="page">
<a class="skip-link screen-reader-text" href="#content">Skip to content</a>
<div id="masthead">
<header class="site-header" id="brandbar" role="banner">
<div id="identity-print"></div>
<div id="globalsearch" role="search">
<input aria-label="Toggle visibility of the search box." id="gsform-toggle" role="presentation" type="checkbox"/>
<label for="gsform-toggle" id="gsform"><span>Search</span></label>
<form action="https://www.uri.edu/search" id="gs" method="get" name="global_general_search_form">
<input name="cx" type="hidden" value="016863979916529535900:17qai8akniu">
<input name="cof" type="hidden" value="FORID:11"/>
<label for="gs-query" id="gs-query-label">Searchbox</label>
<input id="gs-query" name="q" placeholder="Search" role="searchbox" type="text" value="" />
<input class="searchsubmit" id="gs-submit" name="searchsubmit" type="submit" value="Search"/>
</input></form>
</div>
<div id="globalbanner-wrapper">
<div id="globalbanner">
<a href="https://www.uri.edu/" title="University of Rhode Island"><div id="identity">University of Rhode Island</div></a>
<div id="gateways">
<input aria-label="Open the audience gateways menu when browsing on mobile" id="gateways-toggle" role="presentation" type="checkbox"/>
<label for="gateways-toggle" id="gateways-label"><span>You</span></label>
<ul id="gateways-menu" role="menu">
<li><a href="https://www.uri.edu/gateway/future-students" role="menuitem">Future Students</a></li>
<li><a href="https://www.uri.edu/gateway/students" role="menuitem">Students</a></li>
<li><a href="https://www.uri.edu/gateway/faculty" role="menuitem">Faculty</a></li>
<li><a href="https://www.uri.edu/gateway/staff" role="menuitem">Staff</a></li>
<li><a href="https://www.uri.edu/gateway/families" role="menuitem">Parents and Families</a></li>
<li><a href="https://www.uri.edu/gateway/alumni" role="menuitem">Alumni</a></li>
<li><a href="https://www.uri.edu/gateway/community" role="menuitem">Community</a></li>
</ul>
</div>
</div>
</div>
</header><!-- #brandbar -->
<header id="siteheader">
<div class="light" id="sitebanner">
<div id="sb-backdrop">
<div id="sb-background-image" style="background-image:url(&#47;https://web.uri.edu/cs/files/cropped-Big-Data.jpg)"></div>
<div id="sb-screen"></div>
</div>
<div id="sitebranding">
<div id="siteidentity">
<h1 class="site-title">
<a href="https://web.uri.edu/cs/" rel="home">
                    Department of Computer Science and Statistics </a>
</h1>
<h2 class="site-description">College of Arts and Sciences</h2>
</div>
<div id="sitesocial">
</div>
</div><!-- #sitebranding -->
</div><!-- #sitebanner -->
<div class="content-width" id="navigation">
<nav aria-label="Breadcrumb" id="breadcrumbs">
<ol><li><a href="https://www.uri.edu/">URI</a></li><li><a href="https://web.uri.edu/artsci">Arts and Sciences</a></li><li><a href="https://web.uri.edu/cs">Department of Computer Science and Statistics</a></li><li aria-current="page">People</li></ol></nav>
<div id="localnav">
<section class="cl-wrapper cl-menu-wrapper"><div class="cl-menu" data-name="Site Menu" data-show-title="0" id="cl-localnav"><ul class="cl-menu-list cl-menu-list-no-js" id="menu-navigation"><li class="menu-item menu-item-type-custom menu-item-object-custom menu-item-home menu-item-8243" id="menu-item-8243"><a href="https://web.uri.edu/cs" title="Home">Home</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8255" id="menu-item-8255"><a href="https://web.uri.edu/cs/about/" title="About">About</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8806" id="menu-item-8806"><a href="https://web.uri.edu/cs/academics/">Academics</a></li>
<li 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" id="menu-item-8257"><a aria-
```

```
current="page" href="https://web.uri.edu/cs/people/" title=""
">>People</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-9661"
id="menu-item-9661"><a href="https://web.uri.edu/cs/research/">Research</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-10871"
id="menu-item-10871"><a href="https://web.uri.edu/cs/news-and-events/">News and
Events</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8267"
id="menu-item-8267"><a href="https://web.uri.edu/cs/contact/" title="
">Contact</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-15823"
id="menu-item-15823"><a href="https://web.uri.edu/cs/cs-colloquium-series/">Colloquia</a>
</li>
</ul></div></section></div>
</div>
</header>
</div>
<div class="site-content" id="content">
<main class="site-main" id="main" role="main">
<article class="post-629 page type-page status-publish hentry" id="post-629">
<div class="entry-content">
<h1>People</h1>
<section class="cl-wrapper cl-menu-wrapper"><div class="cl-menu" data-name="people" data-
show-title="0" id=""><ul class="cl-menu-list cl-menu-list-no-js" id="menu-people"><li
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" id="menu-item-8737"><a aria-
current="page" href="https://web.uri.edu/cs/people/">Faculty</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8746"
id="menu-item-8746"><a href="https://web.uri.edu/cs/people/staff/">Staff</a></li>
<li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-11844"
id="menu-item-11844"><a href="https://web.uri.edu/cs/people/faculty-emeriti/">Faculty
Emeriti</a></li>
</ul></div></section>
<h2>Full-time Faculty</h2>

<div class="uri-people-tool cl-tiles halves"><div class="peopleitem h-card has-
thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/prince-allotey/"></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/prince-allotey/">Prince
Allotey</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Lecturer </p>
<p class="people-department">Statistics</p>
<p class="people-misc"><span class="p-tel">401.874.2216</span> – <a class="u-email"
href="mailto:prince_allotey@uri.edu">prince_allotey@uri.edu</a></p>
<div style="clear:both;"></div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/marco-alvarez/"></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">Assistant 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 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">Lecturer</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/"></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/sarah-brown/">Sarah Brown</a>
</h3>
</div>
</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/"></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/michael-conti/">Michael Conti</a>
</h3>
</div>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Lecturer</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/"></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/noah-daniels/">Noah Daniels</a>
</h3>
</div>
</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/"></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/"></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:wolfe@cs.uri.edu">wolfe@cs.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/abdeljawad-hendawi/"></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/abdeljawad-hendawi/">Abdeljawad 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/"></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">jyh@cs.uri.edu</p>

<div style="clear:both;"></div>

</div>

<div class="peopleitem h-card has-thumbnail">

<header>

<div class="header">

<figure>

</figure>

<h3 class="p-name">Natallia Katenka</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">nkatenka@uri.edu</p>

<div style="clear:both;"></div>

</div>

</div>

<div class="peopleitem h-card">

<header>

<div class="header">

<h3 class="p-name">Soheyb Kouider</h3>

</div>

</header>

<div class="inside">

<p class="people-title p-job-title">Lecturer</p>

<p class="people-department">Statistics</p>

<p class="people-misc">401.874.2562 - soheyb@uri.edu</p>

<div style="clear:both;"></div>

</div>

</div>

<div class="peopleitem h-card has-thumbnail">

<header>

<div class="header">

<figure>

</figure>

<h3 class="p-name">Edmund Lamagna</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">eal@cs.uri.edu</p>

<div style="clear:both;"></div>

</div>

</div>

<div class="peopleitem h-card has-thumbnail">

<header>

<div class="header">

<figure>

</figure>

<h3 class="p-name">Indrani Mandal</h3>

</div>

</header>

<div class="inside">

<p class="people-title p-job-title">Lecturer</p>

<p class="people-department">Computer Science</p>

<p class="people-misc">indrani_mandal@uri.edu</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/"></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:gppuggioni@uri.edu">gppuggioni@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/joseph-squillace/"></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/joseph-squillace/">Joseph Squillace</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Post-Doctoral Fellow</p>
<p class="people-department">Statistics</p>
<p class="people-misc"><a class="u-email" href="mailto:josephps@uri.edu">josephps@uri.edu</a></p>
<div style="clear:both;"></div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/krishna-venkatasubramanian/"></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 class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/jing-wu/"></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">Assistant 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/"></a>  
</figure>  
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/yichi-zhang/">Yichi Zhang</a></h3>  
</div>  
</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/"></a>  
</figure>  
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/guangyu-zhu/">Guangyu Zhu</a></h3>  
</div>  
</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>  
<p>  
<h2>Adjunct Faculty and Limited Joint Appointments</h2>  
</p>  
<div class="uri-people-tool cl-tiles halves"><div class="peopleitem h-card has-thumbnail">  
<header>  
<div class="header">  
<figure>  
<a href="https://web.uri.edu/cs/meet/ashley-buchanan/"></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:ashley.buchanan@uri.edu">ashley.buchanan@uri.edu</a></p>

```
href="mailto:buchanan@uri.edu">buchanan@uri.edu</p>
<div style="clear:both;"></div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Nina Kajiji
</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">nina@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Rachel Schwartz</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">401.874.5404 – rsschwartz@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Ying Zhang</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">401.874.4915 – yingzhang@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>
</div>
<p>
</p>
</p></div><!-- .entry-content -->
</article><!-- #post-## -->
</main><!-- #main -->
</div><!-- #content -->
<div id="actionbar-wrapper">
<div id="actionbar" role="menu">
ConnectApplyTourGive </div>
</div><!-- #actionbar-wrapper -->
<footer id="globalfooter">
<div id="basement">
<div id="storagebins">
<div id="sb-university">
<input checked="" id="sb-university-toggle" name="storagebin" role="presentation" type="radio" value="university"/>
<label aria-label="Open the University footer menu when browsing on mobile." for="sb-
```

```
university-toggle">University</label>
<ul aria-label="The University footer menu." role="menu">
Leadership
Diversity and Inclusion

<div id="sb-campus-life">
<input id="sb-campus-life-toggle" name="storagebin" role="presentation" type="radio" value="campus-life"/>
<label aria-label="Open the Campus Life footer menu when browsing on mobile." for="sb-campus-life-toggle">Campus Life</label>
<ul aria-label="The Campus Life footer menu." role="menu">
Housing
Dining
Athletics and Recreation
Health and Wellness
Events

</div>
<div id="sb-academics">
<input id="sb-academics-toggle" name="storagebin" role="presentation" type="radio" value="academics"/>
<label aria-label="Open the Academics footer menu when browsing on mobile." for="sb-academics-toggle">Academics</label>
<ul aria-label="The Academics footer menu." role="menu">
Undergraduate
Graduate
Advising
Libraries
Internships

</div>
</div>
<div id="gimmicks">
<!-- Tides Widget -->
<div "" class="uri-tides-widget darkmode" data-height="22" data-station="8454049"></div>
<hr/>
<!-- Social Media Component -->
<aside class="cl-wrapper cl-social-wrapper"><ul class="cl-social light">FacebookInstagramTwitterYouTube
</aside> </div>
</div>
<div id="tagline"></div>
<div id="legal">
<p>Copyright © University of Rhode Island | University of Rhode Island, Kingston, RI 02881, USA | 1.401.874.1000</p>
<p>URI is an equal opportunity employer committed to the principles of affirmative action. Work at URI</p>
</div>
</footer><!-- #globalfooter -->
</div><!-- #page -->
<link href="https://web.uri.edu/cs/wp-content/plugins/uri-tides-1.2/css/tides.css?ver=5.7.1" id="uri-tides-css-css" media="all" rel="stylesheet" type="text/css">
<script id="uricl-js-js" src="https://web.uri.edu/cs/wp-content/plugins/uri-component-library/js/cl.built.js?ver=20211211" type="text/javascript"></script>
<script id="wp-jquery-lightbox-js-extra" type="text/javascript">
/* <![CDATA[*/
var JQLBSettings =
{"fitToScreen": "0", "resizeSpeed": "400", "displayDownloadLink": "0", "navbarOnTop": "0", "loopImages": "", "resizeCenter": "", "marginSize": "", "linkTarget": "", "help": "", "prevLinkTitle": "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_image_info_bg_color": "#ffff", "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"
</pre>
```



```
</link></body>
</html>
```

We can use `prettify` method to format it more nicely. This would add tabs even if there were none in the source code.

```
print(cs_people.prettify())
```

```

<!DOCTYPE html>
<html lang="en-US">
<head>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<link href="http://gmpg.org/xfn/11" rel="profile"/>
<title>
People – Department of Computer Science and Statistics
</title>
<meta content="max-image-preview:large" name="robots">
<link href="/s.w.org" rel="dns-prefetch">
<link href="https://web.uri.edu/cs/feed/" rel="alternate" title="Department of Computer Science and Statistics » Feed" type="application/rss+xml"/>
<link href="https://web.uri.edu/cs/comments/feed/" rel="alternate" title="Department of Computer Science and Statistics » Comments Feed" type="application/rss+xml"/>
<script type="text/javascript">
window._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"}, "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.getElementsByTagName("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]]=function(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([55356,57332,56128,56423,56128,56418,56128,56421,56128,56430,56128,56423,8203,56423,56128,56447],[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(){if("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);
</script>
<style type="text/css">
img.wp-smiley,
img.emoji {
 display: inline !important;
 border: none !important;
 box-shadow: none !important;
 height: 1em !important;
 width: 1em !important;
 margin: 0 .07em !important;
 vertical-align: -0.1em !important;
 background: none !important;
 padding: 0 !important;
}
</style>
<link href="https://web.uri.edu/cs/wp-includes/css/dist/block-library/style.min.css?ver=5.7.1" id="wp-block-library-css" media="all" rel="stylesheet" type="text/css"/>
<link href="https://web.uri.edu/cs/wp-content/plugins/uri-component-library/css/cl.built.css?ver=5.0.2" id="uricl-css-css" media="all" rel="stylesheet" type="text/css"/>
<link href="https://web.uri.edu/cs/wp-content/plugins/uri-courses/assets/courses.css?ver=5.7.1" id="uri-courses-styles-css" media="all" rel="stylesheet" type="text/css"/>
<link href="https://web.uri.edu/cs/wp-content/plugins/uri-people-tool/assets/people.css?ver=5.7.1" id="uri-people-styles-css" media="all" rel="stylesheet" type="text/css"/>
<link href="https://web.uri.edu/cs/wp-content/plugins/wp-lightbox-2/styles/lightbox.min.css?ver=1.3.4" id="wp-lightbox-2.min.css-css" media="all" rel="stylesheet" type="text/css"/>
```

```
<link href="https://web.uri.edu/cs/wp-content/themes/uri-modern/style.css?ver=2.4.0" id="uri-modern-style-css" media="all" rel="stylesheet" type="text/css"/>
<link href="https://web.uri.edu/cs/wp-content/plugins/tablepress/css/default.min.css?ver=1.13" id="tablepress-default-css" media="all" rel="stylesheet" type="text/css"/>
<script id="jquery-core-js" src="https://web.uri.edu/cs/wp-includes/js/jquery/jquery.min.js?ver=3.5.1" type="text/javascript">
</script>
<script id="jquery-migrate-js" src="https://web.uri.edu/cs/wp-includes/js/jquery/jquery-migrate.min.js?ver=3.3.2" type="text/javascript">
</script>
<link href="https://web.uri.edu/cs/wp-json/" rel="https://api.w.org/">
<link href="https://web.uri.edu/cs/wp-json/wp/v2/pages/629" rel="alternate" type="application/json">
<link href="https://web.uri.edu/cs/xmlrpc.php?rsd" rel="EditURI" title="RSD" type="application/rsd+xml"/>
<link href="https://web.uri.edu/cs/wp-includes/wlwmanifest.xml" rel="wlwmanifest" type="application/wlwmanifest+xml"/>
<meta content="WordPress 5.7.1" name="generator">
<link href="https://web.uri.edu/cs/people/" rel="canonical"/>
<link href="https://web.uri.edu/cs/?p=629" rel="shortlink"/>
<link href="https://web.uri.edu/cs/wp-json/oembed/1.0/embed?url=https%3A%2F%2Fweb.uri.edu%2Fc%2Fpeople%2F" rel="alternate" type="application/json+oembed"/>
<link href="https://web.uri.edu/cs/wp-json/oembed/1.0/embed?url=https%3A%2F%2Fweb.uri.edu%2Fc%2Fpeople%2F&format=xml" rel="alternate" type="text/xml+oembed"/>
<meta content="People Full-time Faculty Adjunct Faculty and Limited Join Appointments" name="description"/>
<meta content="summary_large_image" name="twitter:card"/>
<meta content="@universityofri" name="twitter:site"/>
<meta content="@universityofri" name="twitter:creator"/>
<meta content="https://web.uri.edu/cs/people/" property="og:url"/>
<meta content="People" property="og:title"/>
<meta content="People Full-time Faculty Adjunct Faculty and Limited Join Appointments" property="og:description"/>
<meta content="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/logo-wordmark.png" property="og:image"/>
<script>
(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.createElement(s)[0],
j=d.createElement(s),dl=l?'dataLayer':'';j.async=true;j.src=
'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-K5GL9W');
</script>
<style id="wp-custom-css" type="text/css">
.button-list .cl-button {
margin-bottom: 1rem;
}
#calendar .date,#calendar-wrap header {
text-align:center
}
#calendar {
width:100%
}
#calendar a {
color:#005eff;
text-decoration:none
}
#calendar ul {
list-style:none;
padding:0;
margin:0;
width:100%
}
#calendar li {
display:block;
float:left;
width:14.342%;
padding:5px;
box-sizing:border-box;
border:1px solid #ccc;
margin-right:-1px;
margin-bottom:-1px
}
#calendar ul.weekdays {
height:40px;
background:#002147
}
#calendar ul.weekdays li {
text-align:center;
text-transform:uppercase;
line-height:1.2;
border:none!important;
padding:.75rem .5rem;
color:#fff;
font-size:.9rem
```

```
}

#calendar ul.days.mobile {
 display:none
}
#calendar .days li {
 height:15rem
}
#calendar .days.events-zero li {
 height:10rem
}
#calendar .days.events-one li {
 height:10.75rem
}
#calendar .days.events-two li {
 height:11.5rem
}
#calendar .days.events-three li {
 height:15rem
}
#calendar .date {
 margin-bottom:5px;
 padding:4px;
 color:#000;
 padding-right: 1rem;
 float:right;
}
#calendar .event {
 clear:both;
 display:block;
 font-size: 1rem;
 font-family: hind,arial,sans-serif;
 border-radius:30px;
 padding:.4rem .25rem .25rem 5rem;
 margin-top:5px;
 margin-bottom:5px;
 line-height:1.75;
 background:#e4f2f2;
 text-decoration:none;
 float:left;
 position:relative;
 width:calc(100% * 4.68)
}
#calendar .event.four-day{
 width:calc(76% * 4.68)
}
#calendar .event.mobile {
 display:none
}
#calendar .event.scratch {
 background:#E2D58B;
}
#calendar .event.lego {
 background:#E1EDE8;
}
#calendar .event.datacience {
 background:#FCB97D;
}
#calendar .event.girlstech {
 background:#C2CFB2;
}
#calendar .event.games {
 background:#F79365;
}
#calendar .event.programming {
 background:#BCDCE0;
}
#calendar .event.webdesign {
 background:#E09FA2;
}
#calendar .event-desc {
 color:#666;
 margin:3px 0 7px;
 text-decoration:none;
 display:inline-block
}
#calendar .event-time {
 margin:3px 0 7px 5px;
 text-decoration:none;
 display:inline-block
}
#calendar .other-month {
 background:#f5f5f5;
 color:#666
}
#desc {
 display:none;
 background:#eee;
```

```

 box-shadow: 0 0 4px #999;
 padding: 10px;
 color: #000;
 }
 #calendar .event-desc:hover + #desc {
 display: block;
 position: absolute;
 z-index: 2
 }
 .days + header h1 {
 padding-top: 2rem;
 clear: left
 }
 @media (max-width: 768px) {
 #calendar .event img, #calendar .other-month, #calendar .weekdays {
 display: none
 }
 #calendar .event {
 width: calc(100% - 2em);
 padding-left: 20px;
 padding-right: 10px
 }
 #calendar .event.mobile, #calendar ul.days.mobile {
 display: block
 }
 #calendar li {
 height: auto !important;
 border: 1px solid #eddede;
 width: 100%;
 padding: 10px;
 margin-bottom: -1px
 }
 #calendar .date {
 float: none
 }
 }
}

.peopleitem.has-thumbnail img {
 width: 200px;
}

```

</style>

<!-- Favicons -->

<link color="#005eff" href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/safari-pinned-tab.svg" rel="mask-icon"/>

<link href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/favicon.png" rel="icon" type="image/png"/>

<link href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/apple-touch-icon.png" rel="apple-touch-icon"/>

<link href="https://web.uri.edu/cs/wp-content/themes/uri-modern/images/apple-touch-icon-180x180.png" rel="apple-touch-icon" sizes="180x180"/>

</meta>

</link>

</meta>

</head>

<body class="page-template-default page-id-629 page-parent group-blog ln-people">

<noscript>

<iframe height="0" src="https://www.googletagmanager.com/ns.html?id=GTM-K5GL9W" width="0"></iframe>

</noscript>

<div class="site" id="page">

<a class="skip-link screen-reader-text" href="#content">

Skip to content

</a>

<div id="masthead">

<header class="site-header" id="brandbar" role="banner">

<div id="identity-print">



</div>

<div id="globalsearch" role="search">

<input aria-label="Toggle visibility of the search box." id="gsform-toggle" role="presentation" type="checkbox"/>

<label for="gsform-toggle" id="gsform">

<span>

Search

</span>

</label>

<form action="https://www.uri.edu/search" id="gs" method="get" name="global\_general\_search\_form">

<input name="cx" type="hidden" value="016863979916529535900:17qai8akniu">

<input name="cof" type="hidden" value="FORID:11"/>

<label for="gs-query" id="gs-query-label">

Searchbox

</label>

<input id="gs-query" name="q" placeholder="Search" role="searchbox" type="text">

```
value="" />
 <input class="searchsubmit" id="gs-submit" name="searchsubmit" type="submit"
value="Search"/>
 </input>
</form>
</div>
<div id="globalbanner-wrapper">
 <div id="globalbanner">

 <div id="identity">
 University of Rhode Island
 </div>

 <div id="gateways">
 <input aria-label="Open the audience gateways menu when browsing on mobile"
id="gateways-toggle" role="presentation" type="checkbox"/>
 <label for="gateways-toggle" id="gateways-label">

 You

 </label>
 <ul id="gateways-menu" role="menu">

 Future Students

 Students

 Faculty

 Staff

 Parents and Families

 Alumni

 Community

 </div>
 </div>
</div>
</div>
<!-- #brandbar -->
<header id="siteheader">
 <div class="light" id="sitebanner">
 <div id="sb-backdrop">
 <div id="sb-background-image" style="background-
image:url(<https://web.uri.edu/cs/files/cropped-Big-Data.jpg>)">
 </div>
 <div id="sb-screen">
 </div>
 </div>
 <div id="sitebranding">
 <div id="siteidentity">
 <h1 class="site-title">

 Department of Computer Science and Statistics

 </h1>
 <h2 class="site-description">
 College of Arts and Sciences
 </h2>
 </div>
 <div id="sitesocial">
 </div>
 </div>
 <!-- #sitebranding -->
```

```
</div>
<!-- #sitebanner -->

- URI
- Arts and Sciences
- Department of Computer Science and Statistics
- People

- Home
- About
- Academics
- People
- Research
- News and Events
- Contact
- Colloquia


```

```
</div>
<div class="site-content" id="content">
<main class="site-main" id="main" role="main">
<article class="post-629 page type-page status-publish hentry" id="post-629">
<div class="entry-content">
<h1>
 People
</h1>
<section class="cl-wrapper cl-menu-wrapper">
 <div class="cl-menu" data-name="people" data-show-title="0" id="">
 <ul class="cl-menu-list cl-menu-list-no-js" id="menu-people">
 <li 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" id="menu-item-8737">
 <a aria-current="page" href="https://web.uri.edu/cs/people/">
 Faculty

 <li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-8746" id="menu-item-8746">

 Staff

 <li class="menu-item menu-item-type-post_type menu-item-object-page menu-item-11844" id="menu-item-11844">

 Faculty Emeriti

 </div>
</section>
<h2>
 Full-time Faculty
</h2>
<div class="uri-people-tool cl-tiles halves">
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Prince Allotey

 </h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">
 Lecturer
 </p>
 <p class="people-department">
 Statistics
 </p>
 <p class="people-misc">

 401.874.2216

 -

 prince_allotey@uri.edu

 </p>
 <div style="clear:both;">
 </div>
 </div>
 </div>
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Marco Alvarez

 </h3>
 </div>
 </header>
 </div>

```

```
</h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">
 Assistant Professor | Director of Graduate Studies
</p>
<p class="people-department">
 Computer Science
</p>
<p class="people-misc">

 401.874.5009

-

 malvarez@uri.edu

</p>
<div style="clear:both;">
</div>
</div>
</div>
<div class="peopleitem h-card">
<header>
<div class="header">
<h3 class="p-name">

 Samantha Armenti

</h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">
 Lecturer
</p>
<p class="people-department">
 Computer Science
</p>
<p class="people-misc">

 sarmenti@uri.edu

</p>
<div style="clear:both;">
</div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">

 Sarah Brown

</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">

 brownsarahm@uri.edu

</p>
<div style="clear:both;">
</div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
```

```


</figure>
<h3 class="p-name">

 Michael Conti

</h3>
</div>
</header>
<div class="inside">
 <p class="people-title p-job-title">
 Lecturer
 </p>
 <p class="people-department">
 Computer Science
 </p>
 <p class="people-misc">

 michaelconti@uri.edu

 </p>
 <div style="clear:both;">
 </div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Noah Daniels

 </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">

 noah_daniels@uri.edu

 </p>
 <div style="clear:both;">
 </div>
 </div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Lisa DiPippo

 </h3>
 </div>
 </header>
```

```
</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">

 ldipippo@uri.edu

 </p>
 <div style="clear:both;">
 </div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Victor Fay-Wolfe

 </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">

 wolfe@cs.uri.edu

 </p>
 <div style="clear:both;">
 </div>
 </div>
</div>
<div class="peopleitem h-card">
 <header>
 <div class="header">
 <h3 class="p-name">

 Lutz Hamel

 </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">

 lutzhamel@uri.edu

 </p>
 <div style="clear:both;">
 </div>
 </div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 </div>
 </header>
```

```
<h3 class="p-name">

 Abdeljawad Hendawi

</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">

 401.874.5738

-

 hendawi@uri.edu

</p>
<div style="clear:both;">
</div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">

 Jean-Yves Hervé

</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">

 jyh@cs.uri.edu

</p>
<div style="clear:both;">
</div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">

 Natalia Katenka

</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">
```

```

nkatenka@uri.edu

</p>
<div style="clear:both;">
</div>
</div>
</div>
<div class="peopleitem h-card">
<header>
<div class="header">
<h3 class="p-name">

Soheyb Kouider

</h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">
Lecturer
</p>
<p class="people-department">
Statistics
</p>
<p class="people-misc">

401.874.2562

-

soheyb@uri.edu

</p>
<div style="clear:both;">
</div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">

Edmund Lamagna

</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">

eal@cs.uri.edu

</p>
<div style="clear:both;">
</div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">

Indrani Mandal

</h3>
</div>
</header>
```

```
</header>
<div class="inside">
 <p class="people-title p-job-title">
 Lecturer
 </p>
 <p class="people-department">
 Computer Science
 </p>
 <p class="people-misc">

 indrani_mandal@uri.edu

 </p>
 <div style="clear:both;">
 </div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Gavino Puggioni

 </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">

 401.874.4388

 -

 gppuggioni@uri.edu

 </p>
 <div style="clear:both;">
 </div>
 </div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Joseph Squillace

 </h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">
 Post-Doctoral Fellow
 </p>
 </div>
</div>
```

```
</p>
<p class="people-department">
 Statistics
</p>
<p class="people-misc">

 josephps@uri.edu

</p>
<div style="clear:both;">
</div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Krishna Venkatasubramanian

 </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">

 krish@uri.edu

 </p>
 <div style="clear:both;">
 </div>
 </div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Jing Wu

 </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">

 401.874.4504

 -

 jing_wu@uri.edu

 </p>
 </div>
</div>
```

```

 </p>
 <div style="clear:both;">
 </div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Yichi Zhang

 </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">

 yichizhang@uri.edu

 </p>
 <div style="clear:both;">
 </div>
 </div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Guangyu Zhu

 </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">

 guangyuzhu@uri.edu

 </p>
 <div style="clear:both;">
 </div>
 </div>
</div>
<p>
 <h2>
 Adjunct Faculty and Limited Joint Appointments
 </h2>
</p>
<div class="uri-people-tool cl-tiles halves">
 <div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Ashley Buchanan

 </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">

 401.874.4739

 -

 buchanan@uri.edu

 </p>
 <div style="clear:both;">
 </div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Nina Kajiji

 </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">

 nina@uri.edu

 </p>
 <div style="clear:both;">
 </div>
 </div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Rachel Schwartz

 </h3>
 </div>
 </header>
 <div class="inside">
 <p class="people-title p-job-title">
 Assistant Professor – Limited Joint Appointment
 </p>
```

```
</p>
<p class="people-department">
 Biological Sciences
</p>
<p class="people-misc">

 401.874.5404

 -

 rsschwartz@uri.edu

</p>
<div style="clear:both;">
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
 <header>
 <div class="header">
 <figure>

 </figure>
 <h3 class="p-name">

 Ying Zhang

 </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">

 401.874.4915

 -

 yingzhang@uri.edu

 </p>
 <div style="clear:both;">
 </div>
 </div>
 </div>
 <p>
 </p>
 </div>
 <!-- .entry-content -->
</article>
<!-- #post-## -->
</main>
<!-- #main -->
</div>
<!-- #content -->
<div id="actionbar-wrapper">
 <div id="actionbar" role="menu">

 Connect

 Apply

 Tour

 Give

 </div>
</div>
```

```

</div>
</div>
<!-- #actionbar-wrapper -->
<footer id="globalfooter">
<div id="basement">
<div id="storagebins">
<div id="sb-university">
<input checked="" id="sb-university-toggle" name="storagebin" role="presentation" type="radio" value="university"/>
<label aria-label="Open the University footer menu when browsing on mobile." for="sb-university-toggle">

 University

</label>
<ul aria-label="The University footer menu." role="menu">

 Leadership

 Diversity and Inclusion

 Global

 Campuses

 Safety

</div>
<div id="sb-campus-life">
<input id="sb-campus-life-toggle" name="storagebin" role="presentation" type="radio" value="campus-life"/>
<label aria-label="Open the Campus Life footer menu when browsing on mobile." for="sb-campus-life-toggle">

 Campus Life

</label>
<ul aria-label="The Campus Life footer menu." role="menu">

 Housing

 Dining

 Athletics and Recreation

 Health and Wellness

 Events

</div>
<div id="sb-academics">
<input id="sb-academics-toggle" name="storagebin" role="presentation" type="radio" value="academics"/>
<label aria-label="Open the Academics footer menu when browsing on mobile." for="sb-academics-toggle">

```

```
 Academics

</label>
<ul aria-label="The Academics footer menu." role="menu">

 Undergraduate

 Graduate

 Advising

 Libraries

 Internships

</div>
</div>
<div id="gimmicks">
 <!-- Tides Widget -->
 <div "" class="uri-tides-widget darkmode" data-height="22" data-
station="8454049">

 </div>
 <hr/>
 <!-- Social Media Component -->
 <aside class="cl-wrapper cl-social-wrapper">
 <ul class="cl-social light">

 <a class="cl-social-facebook" href="https://www.facebook.com/universityofri"
title="Facebook">
 Facebook

 <a class="cl-social-instagram" href="https://www.instagram.com/universityofri/"
title="Instagram">
 Instagram

 <a class="cl-social-twitter" href="https://twitter.com/universityofri"
title="Twitter">
 Twitter

 <a class="cl-social-youtube" href="https://www.youtube.com/user/UniversityOfRI"
title="YouTube">
 YouTube

 </aside>
</div>
</div>
<div id="tagline">
</div>
<div id="legal">
 <p>
 Copyright ©

 University of Rhode Island

 | University of Rhode Island, Kingston, RI 02881, USA | 1.401.874.1000
 </p>
 <p>
 URI is an equal opportunity employer committed to the principles of affirmative
action.

 Work at URI

 </p>
</div>
```

```

</p>
</div>
</footer>
<!-- #globalfooter -->
</div>
<!-- #page -->
<link href="https://web.uri.edu/cs/wp-content/plugins/uri-tides-1.2/css/tides.css?ver=5.7.1" id="uri-tides-css-css" media="all" rel="stylesheet" type="text/css">
<script id="uricl-js-js" src="https://web.uri.edu/cs/wp-content/plugins/uri-component-library/js/cl.built.js?ver=20211211" type="text/javascript">
</script>
<script id="wp-jquery-lightbox-js-extra" type="text/javascript">
/* <![CDATA[*/
var JQLBSettings =
{"fitToScreen": "0", "resizeSpeed": "400", "displayDownloadLink": "0", "navbarOnTop": "0", "loopImages": "", "resizeCenter": "", "marginSize": "", "linkTarget": "", "help": "", "prevLinkTitle": "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_image_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_size_fix": "0"};
/*]]> */
</script>
<script id="wp-jquery-lightbox-js" src="https://web.uri.edu/cs/wp-content/plugins/wp-lightbox-2/js/dist/wp-lightbox-2.min.js?ver=1.3.4.1" type="text/javascript">
</script>
<script id="uri-modern-navigation-js" src="https://web.uri.edu/cs/wp-content/themes/uri-modern/js/navigation.js?ver=2.4.0" type="text/javascript">
</script>
<script id="uri-modern-smoothscroll-js" src="https://web.uri.edu/cs/wp-content/themes/uri-modern/js/smoothscroll.min.js?ver=2.4.0" type="text/javascript">
</script>
<script id="uri-modern-skip-link-focus-fix-js" src="https://web.uri.edu/cs/wp-content/themes/uri-modern/js/skip-link-focus-fix.js?ver=2.4.0" type="text/javascript">
</script>
<script id="uri-modern-scripts-js-extra" type="text/javascript">
/* <![CDATA[*/
var URIMODERN = {"base": "https://web.uri.edu/cs", "path": {"page": "https://web.uri.edu/cs/people/", "theme": "https://web.uri.edu/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": "2.4.0", "textDomain": "uri"}, "is": {""404": false, "childTheme": false, "admin": false}, "features": []};
/*]]> */
</script>
<script id="uri-modern-scripts-js" src="https://web.uri.edu/cs/wp-content/themes/uri-modern/js/script.min.js?ver=2.4.0" type="text/javascript">
</script>
<script id="page-links-to-js" src="https://web.uri.edu/cs/wp-content/plugins/page-links-to/dist/new-tab.js?ver=3.3.5" type="text/javascript">
</script>
<script id="wp-embed-js" src="https://web.uri.edu/cs/wp-includes/js/wp-embed.min.js?ver=5.7.1" type="text/javascript">
</script>
<script id="uri-tides-js-extra" type="text/javascript">
/* <![CDATA[*/
var tides = {"temperature": {"metadata": {"id": "8454049", "name": "Quonset Point", "lat": "41.5868", "lon": "-71.4110"}, "data": [{"t": "2021-12-11 00:24", "v": "44.8", "f": "0,0,0"}, {"t": "2021-12-10 05:32", "v": "3.653", "type": "H"}, {"t": "2021-12-10 10:39", "v": "0.427", "type": "L"}, {"t": "2021-12-10 17:57", "v": "3.760", "type": "H"}, {"t": "2021-12-11 00:26", "v": "0.358", "type": "L"}, {"t": "2021-12-11 06:29", "v": "3.589", "type": "H"}, {"t": "2021-12-11 12:35", "v": "0.624", "type": "L"}, {"t": "2021-12-11 18:53", "v": "3.486", "type": "H"}, {"t": "2021-12-12 01:36", "v": "0.395", "type": "L"}, {"t": "2021-12-12 07:27", "v": "3.562", "type": "H"}, {"t": "2021-12-12 14:23", "v": "0.580", "type": "L"}, {"t": "2021-12-12 19:50", "v": "3.270", "type": "H"}, {"t": "2021-12-13 02:03", "v": "0.394", "type": "L"}, {"t": "2021-12-13 08:26", "v": "3.579", "type": "H"}, {"t": "2021-12-13 15:08", "v": "0.505", "type": "L"}, {"t": "2021-12-13 20:47", "v": "3.141", "type": "H"}], "date": "1639182721", "expires_on": "1639183021"};
/*]]> */
</script>
<script id="uri-tides-js" src="https://web.uri.edu/cs/wp-content/plugins/uri-tides-</pre>

```

It also makes the tags (structure in HTML) attributes.

cs\_people.title

<title>People – Department of Computer Science and Statistics</title>

For tags that have multiple instances like the `<a>` tag that defines a link, it returns the first instance.

cs\_people.a

[Skip to content](#content)

## 14.4. Finding all instances of a tag

We can use `find_all` to make a list of all occurrences of a tag. For example, we could get all of the links from a page:

```
cs_people.find_all('a')
```

```
[Skip to content,
 <div
id="identity">University of Rhode Island</div>,
 Future
Students,
 Students,
 Faculty,
 Staff,
 Parents and Families,
 Alumni,
 Community,

 Department of Computer Science and Statistics
 ,
 URI,
 Arts and Sciences,
```

<a href="https://web.uri.edu/cs">Department of Computer Science and Statistics</a>,  
<a href="https://web.uri.edu/cs" title="">Home</a>,  
<a href="https://web.uri.edu/cs/about/" title="">About</a>,  
<a href="https://web.uri.edu/cs/academics/">Academics</a>,  
<a aria-current="page" href="https://web.uri.edu/cs/people/" title="">People</a>,  
<a href="https://web.uri.edu/cs/research/">Research</a>,  
<a href="https://web.uri.edu/cs/news-and-events/">News and Events</a>,  
<a href="https://web.uri.edu/cs/contact/" title="">Contact</a>,  
<a href="https://web.uri.edu/cs/cs-colloquium-series/">Colloquia</a>,  
<a aria-current="page" href="https://web.uri.edu/cs/people/">Faculty</a>,  
<a href="https://web.uri.edu/cs/people/staff/">Staff</a>,  
<a href="https://web.uri.edu/cs/people/faculty-emeriti/">Faculty Emeriti</a>,  
<a href="https://web.uri.edu/cs/meet/prince-allotey/"></a>,  
<a href="https://web.uri.edu/cs/meet/prince-allotey/">Prince Allotey</a>,  
<a class="u-email" href="mailto:prince\_allotey@uri.edu">prince\_allotey@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/marco-alvarez/"></a>,  
<a href="https://web.uri.edu/cs/meet/marco-alvarez/">Marco Alvarez</a>,  
<a class="u-email" href="mailto:malvarez@uri.edu">malvarez@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/samantha-armenti/">Samantha Armenti</a>,  
<a class="u-email" href="mailto:sarmenti@uri.edu">sarmenti@uri.edu </a>,  
<a href="https://web.uri.edu/cs/meet/sarah-brown/"></a>,  
<a href="https://web.uri.edu/cs/meet/sarah-brown/">Sarah Brown</a>,  
<a class="u-email" href="mailto:brownsarahm@uri.edu">brownsarahm@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/michael-conti/"></a>,  
<a href="https://web.uri.edu/cs/meet/michael-conti/">Michael Conti</a>,  
<a class="u-email" href="mailto:michaelconti@uri.edu ">michaelconti@uri.edu </a>,  
<a href="https://web.uri.edu/cs/meet/noah-daniels/"></a>,  
<a href="https://web.uri.edu/cs/meet/noah-daniels/">Noah Daniels</a>,  
<a class="u-email" href="mailto:noah\_daniels@uri.edu">noah\_daniels@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/lisa-dipippo/"></a>,  
<a href="https://web.uri.edu/cs/meet/lisa-dipippo/">Lisa DiPippo</a>,  
<a class="u-email" href="mailto:ldipippo@uri.edu">ldipippo@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/victor-fay-wolfe/"></a>,  
<a href="https://web.uri.edu/cs/meet/victor-fay-wolfe/">Victor Fay-Wolfe</a>,  
<a class="u-email" href="mailto:wolfe@cs.uri.edu">wolfe@cs.uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/lutz-hamel/">Lutz Hamel</a>,  
<a class="u-email" href="mailto:lutzhamel@uri.edu">lutzhamel@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/abdeljawab-hendawi/"></a>,  
<a href="https://web.uri.edu/cs/meet/abdeljawab-hendawi/">Abdeljawab Hendawi</a>,  
<a class="u-email" href="mailto:hendawi@uri.edu">hendawi@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/jean-yves-herve/"></a>,  
<a href="https://web.uri.edu/cs/meet/jean-yves-herve/">Jean-Yves Hervé</a>,  
<a class="u-email" href="mailto:jyh@cs.uri.edu">jyh@cs.uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/natallia-katzenka/"></a>,  
<a href="https://web.uri.edu/cs/meet/natallia-katenka/">Natallia Katenka</a>,  
<a class="u-email" href="mailto:nkatenka@uri.edu">nkatenka@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/soheyb-kouider/">Soheyb Kouider</a>,  
<a class="u-email" href="mailto:soheyb@uri.edu">soheyb@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/edmund-lamagna/"></a>,  
<a href="https://web.uri.edu/cs/meet/edmund-lamagna/">Edmund Lamagna</a>,  
<a class="u-email" href="mailto:eal@cs.uri.edu">eal@cs.uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/indrani-mandal/"></a>,  
<a href="https://web.uri.edu/cs/meet/indrani-mandal/">Indrani Mandal</a>,  
<a class="u-email" href="mailto:indrani\_mandal@uri.edu">indrani\_mandal@uri.edu </a>,  
<a href="https://web.uri.edu/cs/meet/gavino-puggioni/"></a>,  
<a href="https://web.uri.edu/cs/meet/gavino-puggioni/">Gavino Puggioni</a>,  
<a class="u-email" href="mailto:gpuggioni@uri.edu">gpuggioni@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/joseph-squillace/"></a>,  
<a href="https://web.uri.edu/cs/meet/joseph-squillace/">Joseph Squillace</a>,  
<a class="u-email" href="mailto:josephps@uri.edu">josephps@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/krishna-venkatasubramanian/"></a>,  
<a href="https://web.uri.edu/cs/meet/krishna-venkatasubramanian/">Krishna  
Venkatasubramanian</a>,  
<a class="u-email" href="mailto:krish@uri.edu">krish@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/jing-wu/"></a>,  
<a href="https://web.uri.edu/cs/meet/jing-wu/">Jing Wu</a>,  
<a class="u-email" href="mailto:jing\_wu@uri.edu">jing\_wu@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/yichi-zhang/"></a>,  
<a href="https://web.uri.edu/cs/meet/yichi-zhang/">Yichi Zhang</a>,  
<a class="u-email" href="mailto:yichizhang@uri.edu">yichizhang@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/guangyu-zhu/"></a>,  
<a href="https://web.uri.edu/cs/meet/guangyu-zhu/">Guangyu Zhu</a>,  
<a class="u-email" href="mailto:guangyuzhu@uri.edu">guangyuzhu@uri.edu</a>,  
<a href="https://web.uri.edu/cs/meet/ashley-buchanan/"></a>,
<a href="https://web.uri.edu/cs/meet/ashley-buchanan/">Ashley Buchanan</a>,
<a class="u-email" href="mailto:buchanan@uri.edu">buchanan@uri.edu</a>,
<a href="https://web.uri.edu/cs/meet/nina-kajiji/"></a>,
<a href="https://web.uri.edu/cs/meet/nina-kajiji/">Nina Kajiji</a>,
<a class="u-email" href="mailto:nina@uri.edu">nina@uri.edu</a>,
<a href="https://web.uri.edu/cs/meet/rachel-schwartz/"></a>,
<a href="https://web.uri.edu/cs/meet/rachel-schwartz/">Rachel Schwartz</a>,
<a class="u-email" href="mailto:rsschwartz@uri.edu">rsschwartz@uri.edu</a>,
<a href="https://web.uri.edu/cs/meet/ying-zhang/"></a>,
<a href="https://web.uri.edu/cs/meet/ying-zhang/">Ying Zhang</a>,
<a class="u-email" href="mailto:yingzhang@uri.edu">yingzhang@uri.edu</a>,
<a href="https://www.uri.edu/connect" 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>,
<a href="https://www.uri.edu/about/leadership/" role="menuitem">Leadership</a>,
<a href="https://web.uri.edu/diversity/" role="menuitem">Diversity and Inclusion</a>,
<a href="https://web.uri.edu/global/" role="menuitem">Global</a>,
<a href="https://web.uri.edu/about/campuses/" role="menuitem">Campuses</a>,
<a href="https://www.uri.edu/safety/" role="menuitem">Safety</a>,
<a href="https://web.uri.edu/housing/" role="menuitem">Housing</a>,
<a href="https://web.uri.edu/dining/" role="menuitem">Dining</a>,
<a href="https://www.uri.edu/athletics/" role="menuitem">Athletics and Recreation</a>,
<a href="https://www.uri.edu/campus-life/health-and-wellness/" role="menuitem">Health and Wellness</a>,
<a href="https://events.uri.edu" role="menuitem">Events</a>,
<a href="https://www.uri.edu/academics/" role="menuitem">Undergraduate</a>,
<a href="https://web.uri.edu/graduate-school/" role="menuitem">Graduate</a>,
<a href="https://web.uri.edu/advising/" role="menuitem">Advising</a>,
<a href="https://web.uri.edu/library/" role="menuitem">Libraries</a>,
<a href="https://web.uri.edu/career/students/" role="menuitem">Internships</a>,
<a class="cl-social-facebook" href="https://www.facebook.com/universityofri" title="Facebook">Facebook</a>,
<a class="cl-social-instagram" href="https://www.instagram.com/universityofri/" title="Instagram">Instagram</a>,
<a class="cl-social-twitter" href="https://twitter.com/universityofri" title="Twitter">Twitter</a>,
<a class="cl-social-youtube" href="https://www.youtube.com/user/University0fRI" title="YouTube">YouTube</a>,
<a class="subtle" href="http://www.uri.edu/">University of Rhode Island</a>,
<a class="jobs" href="https://jobs.uri.edu/">Work at URI</a>]

```

We noted before that each person's information is contained in a `div` tag with the `class = peopleitem`.

`find_all` can also take values for attributes of a tag. Attributes are the modifiers of a tag, in this case, a class is a label that defines formatting, and in this case, acts as metadata about what is in the div. Scraping relies on the HTML code being well organized.

```
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/prince-allotey/"></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/prince-allotey/">Prince
Allotey</a></h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Lecturer </p>
<p class="people-department">Statistics</p>
<p class="people-misc"><span class="p-tel">401.874.2216</span> – <a class="u-email"
href="mailto:prince_allotey@uri.edu">prince_allotey@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/marco-alvarez/"></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">Assistant 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">Lecturer</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/"></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/sarah-brown/">Sarah Brown</a></h3>
</div>
</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/"></a>
</figure>
<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/michael-conti/">Michael Conti</a></h3>
</div>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Lecturer</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 class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
<a href="https://web.uri.edu/cs/meet/noah-daniels/"></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/"></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/"></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:wolfe@cs.uri.edu">wolfe@cs.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/abdeljawad-hendawi/"></a>](https://web.uri.edu/cs/files/unnamed-150x150.jpg)

</figure>

<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/abdeljawad-hendawi/">Abdeljawad  
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/"></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/natalia-katenka/"></a>

</figure>

<h3 class="p-name"><a href="https://web.uri.edu/cs/meet/natalia-katenka/">Natalia  
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">Lecturer</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/"></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">eal@cs.uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>,
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Indrani Mandal</h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Lecturer</p>
<p class="people-department">Computer Science</p>
<p class="people-misc">indrani_mandal@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>,
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Gavino Puggioni</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">401.874.4388 - gppuggioni@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>,
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Joseph Squillace</h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Post-Doctoral Fellow</p>
<p class="people-department">Statistics</p>
<p class="people-misc">josephps@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>,
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>
```

```

</figure>
<h3 class="p-name">Krishna Venkatasubramanian</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">krish@uri.edu</p>
</p>
<div style="clear:both;"></div>
</div>
</div>,
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Jing Wu</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">401.874.4504 - jing_wu@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>,
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Yichi Zhang</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">yichizhang@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>,
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Guangyu Zhu</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">guangyuzhu@uri.edu</p>
```

```
<div style="clear:both;"></div>
</div>
</div>
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Ashley Buchanan</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">401.874.4739 – buchanan@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>,
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Nina Kajiji</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">nina@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>,
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Rachel Schwartz</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">401.874.5404 – rsschwartz@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>,
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Ying Zhang</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">401.874.4915 – yingzhang@uri.edu</p>
```

```
<div style="clear:both;"></div>
</div>
</div>]
```

We could use this to see how many people there are

```
len(cs_people.find_all("div","peopleitem"))
```

```
25
```

or use multiple to see how many people have thumbnail

```
len(cs_people.find_all("div",{"has-thumbnail"}))
```

```
22
```

## 14.5. Finding data we can make tabular

We can look at the first one in detail to determine what to extract for each of the column.

```
cs_people.find_all("div","peopleitem")[0]
```

```
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Prince
Allotey</h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Lecturer </p>
<p class="people-department">Statistics</p>
<p class="people-misc">401.874.2216 – <a class="u-email"
href="mailto:prince_allotey@uri.edu">prince_allotey@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>
```

We can see that the name is an `<h3>` tag with `class = "p-name"`

```
first_name = cs_people.find_all("h3","p-name")[0]
```

We can examine this using our typical tools:

```
type(first_name)
```

```
bs4.element.Tag
```

It has attributes, since it's a tag object:

```
first_name.contents
```

```
[Prince Allotey]
```

What we want is the string:

```
first_name.string
```

```
'Prince Allotey'
```

### Question in class

How can we extract the link?

That's a child, because the `<a>` tag is inside of the the `<h3>` tag, so let's explore the children.

```
[type(c) for c in first_name.children]
```

```
[bs4.element.Tag]
```

Alternatively, we can pick the `a` tag out by name.

```
first_name.a
```

```
Prince Allotey
```

the url or `href` is an attribute of the `a` tag.

```
first_name.a.attrs
```

```
{'href': 'https://web.uri.edu/cs/meet/prince-allotey/'}
```

```
first_name.a.attrs.href
```

```

AttributeError Traceback (most recent call last)
/tmp/ipykernel_2098/12535113.py in <module>
----> 1 first_name.a.attrs.href

AttributeError: 'dict' object has no attribute 'href'
```

we can get it out using the `[]` to index into the `attrs` dictionary

```
first_name.a.attrs['href']
```

```
'https://web.uri.edu/cs/meet/prince-allotey/'
```

## 14.6. Building a DataFrame

Now that we know what to look for, we can start building. First, we'll find all of the names and extract the string from each using a list comprehension.

```
names = [name.string for name in cs_people.find_all("h3", "p-name")]
names
```

```
['Prince Allotey',
'Marco Alvarez',
'Samantha Armenti',
'Sarah Brown',
'Michael Conti',
'Noah Daniels',
'Lisa DiPippo',
'Victor Fay-Wolfe',
'Lutz Hamel',
'Abdeltawab Hendawi',
'Jean-Yves Hervé',
'Natalia Katenka',
'Soheyb Kouider',
'Edmund Lamagna',
'Indrani Mandal',
'Gavino Puggioni',
'Joseph Squillace',
'Krishna Venkatasubramanian',
'Jing Wu',
'Yichi Zhang',
'Guangyu Zhu',
'Ashley Buchanan',
'Nina Kajiji',
'Rachel Schwartz',
'Ying Zhang']
```

We can use the same process for each other attribut we want. First, we'll look at the whole peopleitem again , and then decide what we want.

```
cs_people.find_all("div","peopleitem")[0]
```

```
<div class="peopleitem h-card has-thumbnail">
<header>
<div class="header">
<figure>

</figure>
<h3 class="p-name">Prince
Allotey</h3>
</div>
</header>
<div class="inside">
<p class="people-title p-job-title">Lecturer </p>
<p class="people-department">Statistics</p>
<p class="people-misc">401.874.2216 - <a class="u-email"
href="mailto:prince_allotey@uri.edu">prince_allotey@uri.edu</p>
<div style="clear:both;"></div>
</div>
</div>
```

We'll extract the department, title, and e-mail.

```
disciplines = [d.string for d in cs_people.find_all("p",
 'people-department')]
titles = [t.string for t in cs_people.find_all("p", "people-title")]
emails = [e.string for e in cs_people.find_all("a", 'u-email')]
pd.DataFrame({'name':names, 'title':titles,
 'e-mails':emails, 'discipline':disciplines})
```

	<b>name</b>	<b>title</b>	<b>e-mails</b>	<b>discipline</b>
0	Prince Allotey	Lecturer	prince_allotey@uri.edu	Statistics
1	Marco Alvarez	Assistant Professor   Director of Graduate Stu...	malvarez@uri.edu	Computer Science
2	Samantha Armenti	Lecturer	sarmenti@uri.edu	Computer Science
3	Sarah Brown	Assistant Professor	brownsarahm@uri.edu	Computer Science
4	Michael Conti	Lecturer	michaelconti@uri.edu	Computer Science
5	Noah Daniels	Assistant Professor	noah_daniels@uri.edu	Computer Science
6	Lisa DiPippo	Professor   Chair	ldipippo@uri.edu	Computer Science
7	Victor Fay-Wolfe	Professor	wolfe@cs.uri.edu	Computer Science
8	Lutz Hamel	Associate Professor	lutzhamel@uri.edu	Computer Science
9	Abdeltawab Hendawi	Assistant Professor	hendawi@uri.edu	Data Science   Computer Science
10	Jean-Yves Hervé	Associate Professor	jyh@cs.uri.edu	Computer Science
11	Natallia Katenka	Associate Professor   Director of Undergraduat...	nkatenka@uri.edu	Statistics
12	Soheyb Kouider	Lecturer	soheyb@uri.edu	Statistics
13	Edmund Lamagna	Professor	eal@cs.uri.edu	Computer Science
14	Indrani Mandal	Lecturer	indrani_mandal@uri.edu	Computer Science
15	Gavino Puggioni	Associate Professor   Statistics Section Head...	gpuggioni@uri.edu	Statistics
16	Joseph Squillace	Post-Doctoral Fellow	josephps@uri.edu	Statistics
17	Krishna Venkatasubramanian	Assistant Professor	krish@uri.edu	Computer Science
18	Jing Wu	Assistant Professor	jing_wu@uri.edu	Statistics
19	Yichi Zhang	Assistant Professor	yichizhang@uri.edu	Statistics
20	Guangyu Zhu	Assistant Professor	guangyuzhu@uri.edu	Statistics
21	Ashley Buchanan	Limited Joint Appointment	buchanan@uri.edu	Biostatistics
22	Nina Kajiji	Adjunct Associate Professor	nina@uri.edu	Computer Science
23	Rachel Schwartz	Assistant Professor – Limited Joint Appointment	rsschwartz@uri.edu	Biological Sciences
24	Ying Zhang	Assistant Professor – Limited Joint Appointment	yingzhang@uri.edu	Computer Science

```
sp22_csc_sta_url =
'https://raw.githubusercontent.com/rhodyprog4ds/rhodyds/main/data/reg_CSCSTA_courses.csv'
```

```
courses_df = pd.read_csv(sp22_csc_sta_url)
courses_df.head()
```

	Subject	Cat#	Component	Section	GenEd		Title	Max Size	Campus
0	CSC	101	LEC	1	GE	Computing Concepts	35	UF	
1	CSC	101	LEC	2	GE	Computing Concepts	35	ONLII	
2	CSC	101	LEC	3	GE	Computing Concepts	35	ONLII	
3	CSC	101	LEC	L01	GE	Computing Concepts	35	ONLII	
4	CSC	104	LEC	1	GE	Puzzles+Games=Analytical Think	40	UF	

We saw the `attrs` for a link above, where there was only one attribute on the tag, but for example, the images on each person's card have many attributes.

```
cs_people.find_all("div","peopleitem")[0].img.attrs
```

```
{'width': '200',
'height': '200',
'src': 'https://web.uri.edu/cs/files/prince-allotey-web.jpg',
'class': ['u-photo', 'wp-post-image'],
'alt': '',
'loading': 'lazy',
'srcset': 'https://web.uri.edu/cs/files/prince-allotey-web.jpg 200w,
https://web.uri.edu/cs/files/prince-allotey-web-150x150.jpg 150w',
'sizes': '(max-width: 200px) 100vw, 200px'}
```

```
cs_people.find_all("p")
```

```
[<p class="people-title p-job-title">Lecturer </p>,
<p class="people-department">Statistics</p>,
<p class="people-misc">401.874.2216 – <a class="u-email"
href="mailto:prince_allotey@uri.edu">prince_allotey@uri.edu</p>,
<p class="people-title p-job-title">Assistant Professor | Director of Graduate
Studies</p>,
<p class="people-department">Computer Science</p>,
<p class="people-misc">401.874.5009 – <a class="u-email"
href="mailto:malvarez@uri.edu">malvarez@uri.edu</p>,
<p class="people-title p-job-title">Lecturer</p>,
<p class="people-department">Computer Science</p>,
<p class="people-misc"><a class="u-email" href="mailto:sarmenti@uri.edu"
">sarmenti@uri.edu </p>,
<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</p>,
<p class="people-title p-job-title">Lecturer</p>,
<p class="people-department">Computer Science</p>,
<p class="people-misc"><a class="u-email" href="mailto:michaelconti@uri.edu"
">michaelconti@uri.edu </p>,
<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</p>,
<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</p>,
<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:wolfe@cs.uri.edu">wolfe@cs.uri.edu</p>,
<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</p>,
<p class="people-title p-job-title">Assistant Professor</p>,
<p class="people-department">Data Science | Computer Science</p>,
<p class="people-misc">401.874.5738 – <a class="u-email"
href="mailto:hendawi@uri.edu">hendawi@uri.edu</p>,
<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</p>,
```

```

<p class="people-title p-job-title">Associate Professor | Director of Undergraduate Studies</p>
<p class="people-department">Statistics</p>
<p class="people-misc">nkatenka@uri.edu</p>
<p class="people-title p-job-title">Lecturer</p>
<p class="people-department">Statistics</p>
<p class="people-misc">401.874.2562 – soheyb@uri.edu</p>
<p class="people-title p-job-title">Professor</p>
<p class="people-department">Computer Science</p>
<p class="people-misc">eal@cs.uri.edu</p>
<p class="people-title p-job-title">Lecturer</p>
<p class="people-department">Computer Science</p>
<p class="people-misc">indrani_mandal@uri.edu</p>
<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">401.874.4388 – gppuggioni@uri.edu</p>
<p class="people-title p-job-title">Post-Doctoral Fellow</p>
<p class="people-department">Statistics</p>
<p class="people-misc">josephps@uri.edu</p>
<p class="people-title p-job-title">Assistant Professor</p>
<p class="people-department">Computer Science</p>
<p class="people-misc">krish@uri.edu</p>
</p>
<p class="people-title p-job-title">Assistant Professor</p>
<p class="people-department">Statistics</p>
<p class="people-misc">401.874.4504 – jing_wu@uri.edu</p>
<p class="people-title p-job-title">Assistant Professor </p>
<p class="people-department">Statistics</p>
<p class="people-misc">yichizhang@uri.edu</p>
<p class="people-title p-job-title">Assistant Professor</p>
<p class="people-department">Statistics</p>
<p class="people-misc">guangyuzhu@uri.edu</p>
<p>
<h2>Adjunct Faculty and Limited Joint Appointments</h2>
</p>
<p class="people-title p-job-title">Limited Joint Appointment</p>
<p class="people-department">Biostatistics</p>
<p class="people-misc">401.874.4739 – buchanan@uri.edu</p>
<p class="people-title p-job-title">Adjunct Associate Professor</p>
<p class="people-department">Computer Science</p>
<p class="people-misc">nina@uri.edu</p>
</p>
<p class="people-title p-job-title">Assistant Professor – Limited Joint Appointment</p>
<p class="people-department">Biological Sciences</p>
<p class="people-misc">401.874.5404 – rsschwartz@uri.edu</p>
<p class="people-title p-job-title">Assistant Professor – Limited Joint Appointment</p>
<p class="people-department">Computer Science</p>
<p class="people-misc">401.874.4915 – yingzhang@uri.edu</p>
<p>
</p>
<p>
 <!--Copyright © University of Rhode Island | University of Rhode Island, Kingston, RI 02881, USA | 1.401.874.1000</p>
 <!--URI is an equal opportunity employer committed to the principles of affirmative action. Work at URI</p>

```

URI websites are probably formatted consistently, so we could build information about more departments.

- [csc/sta emeriti](#)
- [a&s dean's office](#)
- [math](#)
- [philosophy](#)
- [business](#)

## 14.7. Thinking Ahead

The spreadsheet of spring classes in the department is posted:

```
sp22_csc_sta_url =
'https://raw.githubusercontent.com/rhodyprog4ds/rhodyds/main/data/reg_CSCSTA_courses.csv'
```

this is a minimal copy where I removed enrollments and locations in case those change. this is derived from the last version the Dean asked us to make corrections to, so things will definitely be different before registration opens (eg I'm teaching the CSC392 and it's listed as "Staff")

sections have definitely been added/removed and teaching assignments changed, so don't use this for making plans.

How could you merge this with the DataFrame we just scraped?

## 14.8. More Practice

1. Add a phone number column
2. On the page linked from each person's name, their office number; add a column for office number.
3. Make the code we wrote in class into a function so that you can pass a page and get back a DataFrame.
4. Parse the [course descriptions](#) page and make a DataFrame with columns for subject code, course number, course title, and course description.
5. Merge the descriptions with the table about enrollments and instructors.

## 15. Intro to Machine learning

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.

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

### Further Reading

this [paper](#) by Christopher M. Bishop, a leading ML researcher who also wrote one of 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.

## 15.1. 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]

### Hint

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.

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](#)

### Further Reading

read more in the [Model Based Machine Learning Book](#)

## 15.2. 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  $i$ th sample

$$y_i = f(\mathbf{x}_i)$$

## 15.3. 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](#)

## 15.4. 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**

## 15.5. 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.

## 15.6. Machine Learning with Scikit Learn

```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
sns.set_theme(palette= "colorblind")
```

```
type(GaussianNB)
```

```
abc.ABCMeta
```

```
type(train_test_split)
```

```
function
```

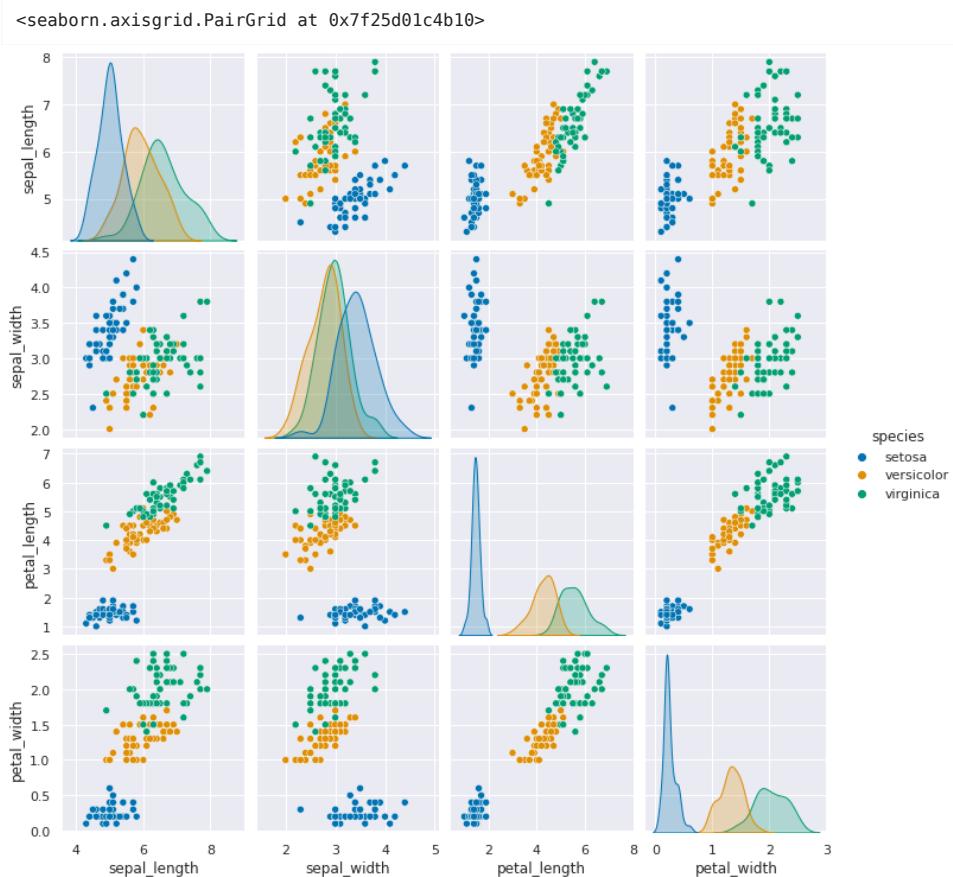
We're trying to build an automatic flower classifier that, for measurements of a new flower returns the predicted species. To do this, we have a DataFrame with columns for species, petal width, petal length, sepal length, and sepal width. The species is what type of flower it is the petal and sepal are parts of the flower.

We'll load the data from Seaborn

```
iris_df = sns.load_dataset('iris')
```

First, we'll look at the data.

```
sns.pairplot(data=iris_df, hue='species')
```



We can see that this data is well suited for classification not only because in the world it makes sense, that we can use the measurements of flower petals to differentiate species, but also because the different species do not overlap too much. We call this **separable** data.

We'll pick out the target and features from this data frame

```
target = iris_df['species'].values
type(target)
```

```
numpy.ndarray
```

```
features = iris_df.values[:, :4]
features.shape
```

```
(150, 4)
```

Next, we use the sklearn function to create train and test data.

```
X_train, X_test, y_train, y_test = train_test_split(features, target, random_state=0)
```

### Try it Yourself

Try using a different random seed (value for `random_state`) or a different split size. Also try splitting by taking the first 80% of the data vs the last 20%. What happens?

We'll check what the split does by looking at the shape.

```
X_train.shape
```

```
(112, 4)
```

```
X_test.shape
```

```
(38, 4)
```

Now we instantiate our classifier with the constructor method

```
gnb = GaussianNB()
```

```
gnb
```

```
GaussianNB()
```

### Try it Yourself

1. what other constructors have we used?
2. What happens if you skip this step?
3. Can you cascade this step?

This created an empty object, we can look at its contents:

```
gnb.__dict__
```

```
{'priors': None, 'var_smoothing': 1e-09}
```

We can fit our model, or learn the model parameters, with the `fit` method. The fit method implements the actual learning algorithm.

```
gnb.fit(X_train, y_train)
```

```
GaussianNB()
```

Once we fit we can see it knows about our dataset now

```
gnb.__dict__
```

```
{'priors': None,
 'var_smoothing': 1e-09,
 'classes_': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 'n_features_in_': 4,
 'epsilon_': 3.2135586734693877e-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])}
```

We can apply our classifier with the predict method, which generates a prediction based on what it learned from the training data for each sample in the test data.

```
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'll see better ways to evaluate on Friday, but as a first check, we can check if it matches

```
sum(y_pred == y_test)
```

```
38
```

and compare to how many.

```
len(y_test)
```

```
38
```

## 15.7. Questions at the End of Class

### 15.7.1. Clarifying Questions

15.7.1.1. Does this method work if the dataframe has more than one categorical variable

15.7.1.2. Were the train and test variables randomly generated from the library based off of the dataset we used?

15.7.1.3. Is this method we learned today using linear regression?

### 15.7.2. Foreshadowing Questions

15.7.2.1. If we were to use a larger sample size for the test, would this improve results?

15.7.2.2. How robust/used is this specific machine learning algorithm? Is there more powerful open source ones we can play with

### 15.7.3. Logistical Questions

15.7.3.1. Will we be scraping data from sites and try to implement machine learning?

15.7.3.2. Will sklearn be used in assignments

### 15.7.4. Context Questions

15.7.4.1. When did Google change to machine learning for their search queries?

## 15.8. More Practice

See the try it yourself boxes above

# 16. Interpreting and Evaluating Naive Bayes

We'll pick up where we left off on Wednesday and we'll cover what the Naive Bayes model assumes and some more evaluation. Next week, we'll see a bit more about its predictions and see a new classifier.

```
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
iris_df = sns.load_dataset('iris')
sns.set_theme(palette= "colorblind")
```

Again we'll load the data and split it to test and train and indicate which variables to use as feature and the target.

```
feature_vars = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width',]
target_var = 'species'
X_train, X_test, y_train, y_test = train_test_split(iris_df[feature_vars],
 iris_df[target_var],
 test_size=.5, random_state=0)
```

This time we also made the test size larger (50% instead of default 25%)

## 16.1. What does Naive Bayes do?

[docs](#)

Gaussian = features distributed according to the Gaussian Distribution (normal curve) Naive = independent features

Bayes = most probable

## [Bayes Estimator](#)

To see, first we'll fit the model, then examine it's structure.

```
gnb= GaussianNB()
gnb.fit(X_train,y_train)

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'],
 [5.935 , 2.71 , 4.185 , 1.3],
 [6.77692308, 3.09230769, 5.73461538, 2.10769231]),
 'theta_': array([[4.97586207, 3.35862069, 1.44827586, 0.23448276],
 [5.935 , 2.71 , 4.185 , 1.3],
 [6.77692308, 3.09230769, 5.73461538, 2.10769231]]),
 'var_': array([[0.10321047, 0.13208086, 0.01629013, 0.00846612],
 [0.256275 , 0.0829 , 0.255275 , 0.046],
 [0.38869823, 0.10147929, 0.31303255, 0.04763314]]),
 'class_count_': array([29., 20., 26.]),
 'class_prior_': array([0.38666667, 0.26666667, 0.34666667])}
```

The attributes of the [estimator object](#) (`gnb`) describe the data (eg the class list) and the model's parameters. The `theta_` ( $(\theta)$ ) represents the mean and the `sigma_` ( $(\sigma)$ ) 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?

When we look at the data with a pair plot we see the distribution. This data is not perfectly Gaussian, but it's pretty close to unimodal (one peak) for each feature and they're relatively [normal curve](#) like (as opposed to a very sharp spike like a [laplacian](#) )

```
sns.pairplot(data=iris_df, hue='species')
```

### Further Reading

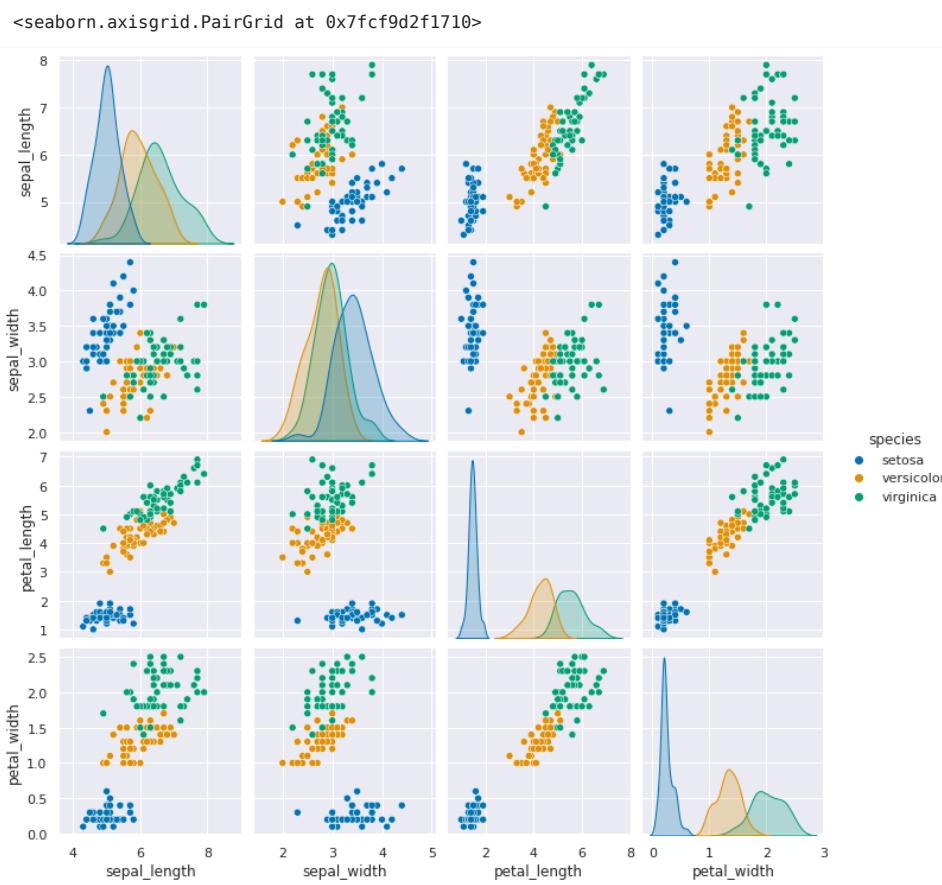
All of this is beyond the scope of this course, but may be of interest. The Scikit Learn [User Guide](#) is actually a really good place to learn the details of machine learning. It is high quality documentation from both a statistical and computer science perspective of every element of the library.

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>). Scikit Learn documentation also includes a [related projects](#) page.

### Tip

If you're interested in fair machine learning, let me know, this is what my research is



Because the GaussianNB classifier calculates parameters that describe the assumed distribution of the data it is called a generative classifier. From a generative classifier, we can generate synthetic data that is from the distribution the classifier 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.

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.

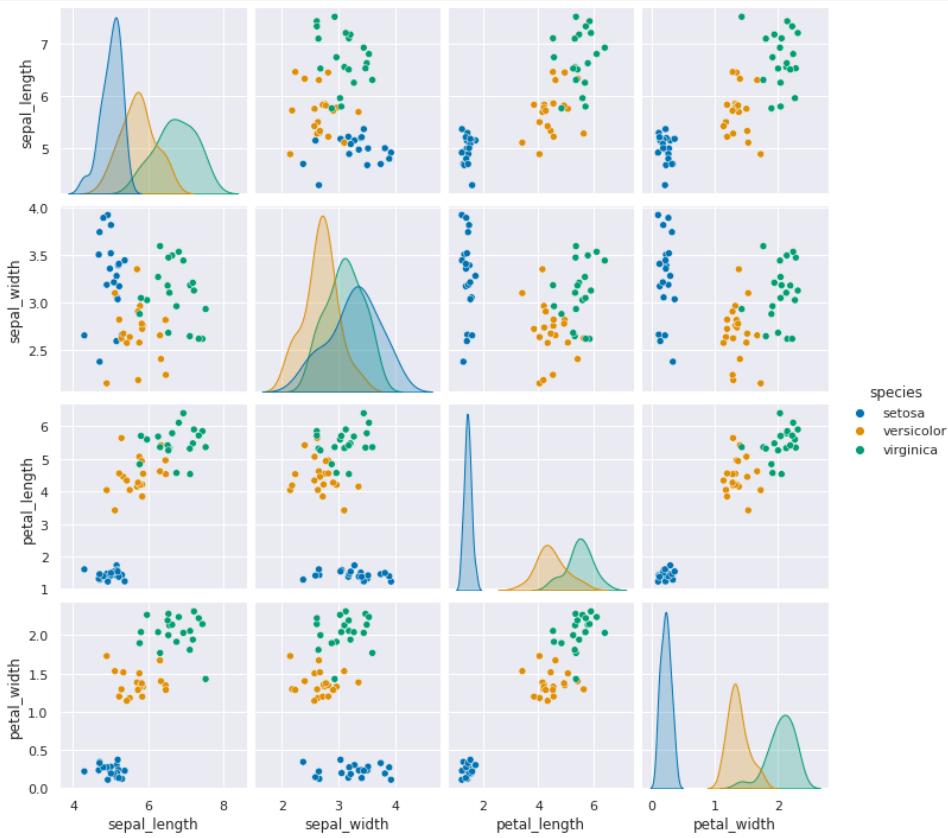
```
N = 20
gnb_df = pd.DataFrame(np.concatenate([np.random.multivariate_normal(th, sig*np.eye(4), N)
 for th, sig in zip(gnb.theta_, gnb.sigma_)]),
 columns = gnb.feature_names_in_)
gnb_df['species'] = [ci for cl in [[c]*N for c in gnb.classes_] for ci in cl]
sns.pairplot(data = gnb_df, hue='species')
```

### 💡 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 and then wrapped them all together.

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/sklearn/utils/deprecation.py:103: FutureWarning: Attribute `sigma_` was
deprecated in 1.0 and will be removed in 1.2. Use `var_` instead.
warnings.warn(msg, category=FutureWarning)
```

```
<seaborn.axisgrid.PairGrid at 0x7fcf98239150>
```



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.

## 16.2. Evaluating

On Wednesday, we used predict and checked the results

```
y_pred = gnb.predict(X_test)
y_pred
```

```
array(['virginica', 'versicolor', 'setosa', 'virginica', 'setosa',
 'virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
 'versicolor', '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', 'versicolor',
 'versicolor', 'virginica', 'setosa', 'virginica', 'setosa',
 'setosa', 'versicolor', 'virginica', 'virginica', 'versicolor',
 'virginica', 'versicolor', 'virginica', 'versicolor', 'versicolor',
 'virginica', 'versicolor', 'setosa', 'virginica', 'versicolor',
 'versicolor', 'versicolor', 'versicolor', 'virginica', 'setosa',
 'setosa', 'virginica', 'versicolor', 'setosa', 'setosa',
 'versicolor'], dtype='<U10')
```

Then compared to see how many matched the ground truth.

### 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](#)

```
sum(y_pred == y_test)
```

71

out of the total number

```
len(y_test)
```

75

Scikit Learn also provides a `score` method for all of the estimator object (different models). For a classifier, it provides the accuracy (percent correct).

```
gnb.score(X_test,y_test)
```

0.9466666666666667

which we could calculate with:

```
sum(y_pred == y_test)/len(y_test)
```

0.9466666666666667

The [confusion matrix](#) counts the number for each class (in this case the species) that truly have that value and that were predicted to have that value. The wikipedia article on [confusion matrices](#) also summarized all of the different metrics. Its written in terms of two outcomes, but we have 3.

```
confusion_matrix(y_test, y_pred,labels = gnb.classes_)
```

```
array([[21, 0, 0],
 [0, 30, 0],
 [0, 4, 20]])
```

### Try it yourself

Could you create a Data Frame with columns for predicted and true class, then get the count of how many samples were in each combination? Does it match the table.

We can also get a report with a few metrics.

- Recall is the percent of each species that were predicted correctly.
- Precision is the percent of the ones predicted to be in a species that are truly that species.
- the F1 score is combination of the two

```
print(classification_report(y_test,y_pred))
```

### Tip

Note, we used the same random seed, but with a different test set size and got a different result because we have a different number of samples.

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	21
versicolor	0.88	1.00	0.94	30
virginica	1.00	0.83	0.91	24
accuracy			0.95	75
macro avg	0.96	0.94	0.95	75
weighted avg	0.95	0.95	0.95	75

## 16.3. Questions

### 💡 Ram token Opportunity

Contribute a question as an issue or contribute a solution to one of the try it yourself above.

## 17. Making Predictions in Generative Model

```
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'll load the data again

```
iris_df.head(1)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa

Next we indicate the feature variables and target and split into test and train sets. We'll use 80% of the data for training.

```
feature_vars = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width',]
target_var = 'species'

X_train, X_test, y_train, y_test = train_test_split(iris_df[feature_vars],
 iris_df[target_var], train_size=.8, random_state=0)
```

We can confirm the shape is as expected

```
X_train.shape
```

```
(120, 4)
```

```
iris_df.shape
```

```
(150, 5)
```

```
.8*150
```

```
120.0
```

Next we again initialize and fit the classifier

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

```
GaussianNB()
```

We can compute predictions

```
y_pred = gnb.predict(X_test)
y_pred
```

```
array(['virginica', 'versicolor', 'setosa', 'virginica', 'setosa',
 'virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
 'versicolor', 'versicolor', 'versicolor', 'versicolor',
 'versicolor', 'setosa', 'versicolor', 'versicolor', 'setosa',
 'setosa', 'virginica', 'versicolor', 'setosa', 'setosa',
 'virginica', 'setosa', 'versicolor', 'versicolor', 'setosa'])
'setosa'], dtype='<U10')
```

And score the results

```
gnb.score(X_test,y_test)
```

```
0.9666666666666667
```

We saw last week that when we fit the Gaussian Naive Bayes, it computes a mean  $\langle\theta\rangle$  and variance  $\langle\sigma^2\rangle$  and adds them to model parameters in attributes `gnb.theta_`, `gnb.sigma_`.

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.

```
gnb.predict_proba(X_test)
```

```
array([[1.63380783e-232, 2.18878438e-006, 9.99997811e-001],
[1.82640391e-082, 9.99998304e-001, 1.69618390e-006],
[1.00000000e+000, 7.10250510e-019, 3.65449801e-028],
[1.58508262e-305, 1.04649020e-006, 9.99998954e-001],
[1.00000000e+000, 8.59168655e-017, 4.22159374e-027],
[6.39815011e-321, 1.56450314e-010, 1.00000000e+000],
[1.00000000e+000, 1.09797313e-016, 5.30276557e-027],
[1.25122812e-146, 7.74052109e-001, 2.25947891e-001],
[5.34357526e-150, 9.07564955e-001, 9.24350453e-002],
[5.67261712e-093, 9.99882109e-001, 1.17891111e-004],
[2.38651144e-210, 5.29609631e-001, 4.70390369e-001],
[8.12047631e-132, 9.43762575e-001, 5.62374248e-002],
[5.25177109e-132, 9.98864361e-001, 1.13563851e-003],
[1.24498038e-139, 9.49838641e-001, 5.01613586e-002],
[4.08232760e-140, 9.88043864e-001, 1.19561365e-002],
[1.00000000e+000, 7.12837229e-019, 4.10162749e-029],
[4.19553996e-131, 9.87944980e-001, 1.20550201e-002],
[4.13286716e-111, 9.99942383e-001, 5.76167389e-005],
[1.00000000e+000, 2.24933112e-015, 3.63624519e-026],
[1.00000000e+000, 9.86750131e-016, 2.42355087e-025],
[1.85930865e-186, 1.66966805e-002, 9.83303319e-001],
[8.83060167e-130, 9.92757232e-001, 7.24276827e-003],
[1.00000000e+000, 4.26380344e-013, 4.34222344e-023],
[1.00000000e+000, 1.28045851e-016, 1.26708019e-027],
[2.43739221e-168, 1.83516225e-001, 8.16483775e-001],
[1.00000000e+000, 2.62431469e-018, 6.72573168e-029],
[1.00000000e+000, 3.20605389e-011, 1.52433420e-020],
[2.20964201e-110, 9.99291229e-001, 7.08771072e-004],
[1.39297338e-046, 9.99999972e-001, 2.81392389e-008],
[1.00000000e+000, 1.85943966e-013, 1.58833385e-023]])
```

These are hard to interpret as is, one option is to plot them

```

make the probabilities 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']
add 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
0	0	virginica	virginica	0,virginica	0,virginica	True	setosa
1	1	versicolor	versicolor	1,versicolor	1,versicolor	True	setosa 1.
2	2	setosa	setosa	2,setosa	2,setosa	True	setosa 1.0
3	3	virginica	virginica	3,virginica	3,virginica	True	setosa
4	4	setosa	setosa	4,setosa	4,setosa	True	setosa 1.0

Now we have a data frame where each row 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
```

```
(90, 8)
```

```
len(y_pred)*len(gnb.classes_)
```

```
90
```

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'`

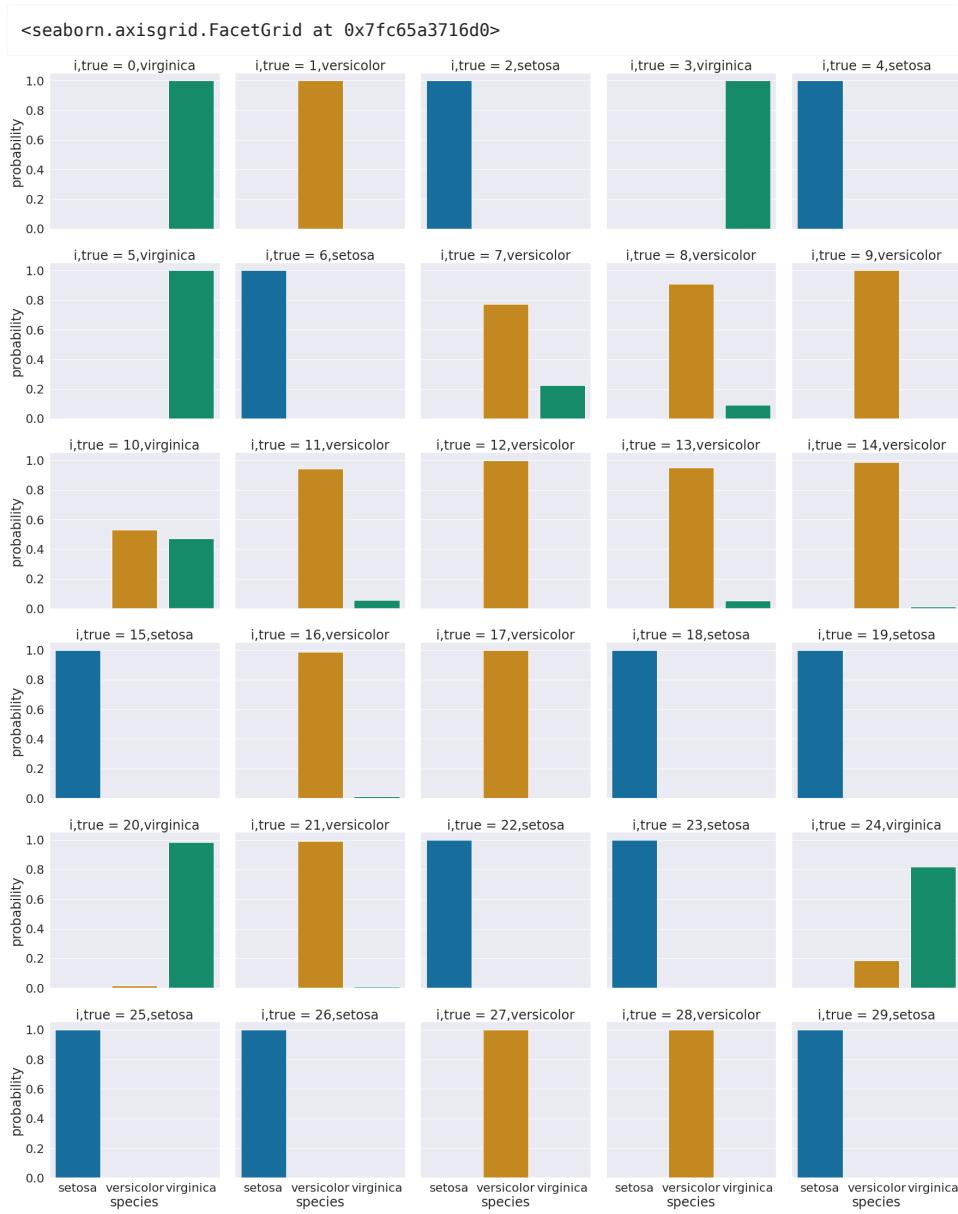
```

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 font 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.



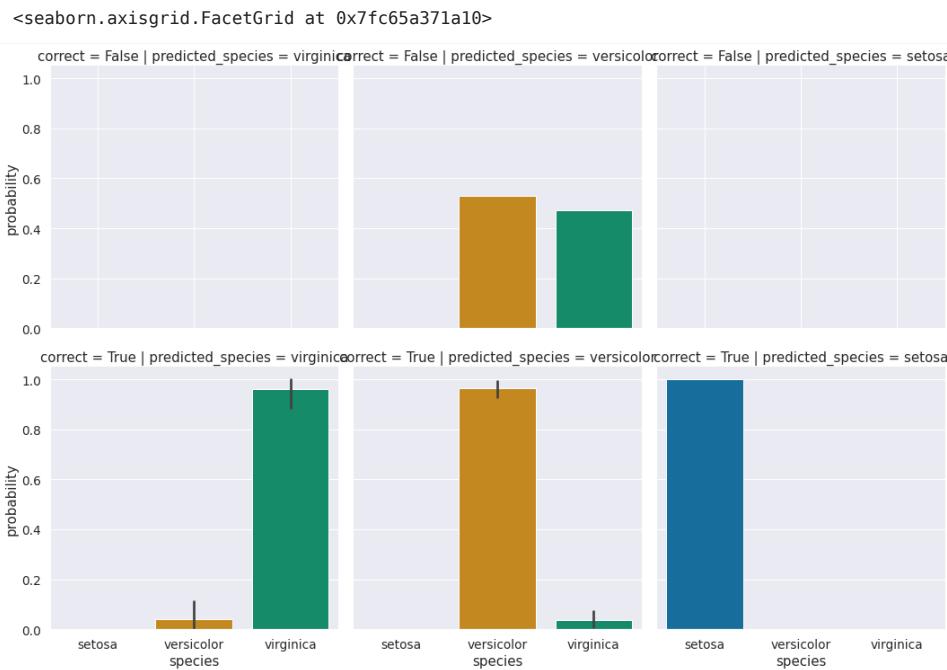
We see that most samples have nearly all of their probability mass (all probabilities in a distribution sum (or integrate if continuous) to 1, but a few samples are not.

### Try it yourself

Try adding a column that could change the headings to include an indicator of which are correct or not.

For now, we'll group and look at on average, what the distributions are for correct vs incorrect based on predictions.

```
sns.set(font_scale=1.25, palette= "colorblind")
sns.catplot(data =prob_df_melted, x = 'species', y='probability' ,
 col ='predicted_species',row = 'correct', kind='bar')
```



We see that the errors were all for versicolor, and on average the distribution is very uncertain for those samples. Those samples are probably hard to distinguish. We could check by creating a data frame with the data and the information about predictions and correct values.

```
prob_data_df = pd.concat([prob_df,X_test.reset_index()],axis=1).drop(columns=['index'])
prob_data_df.head(2)
```

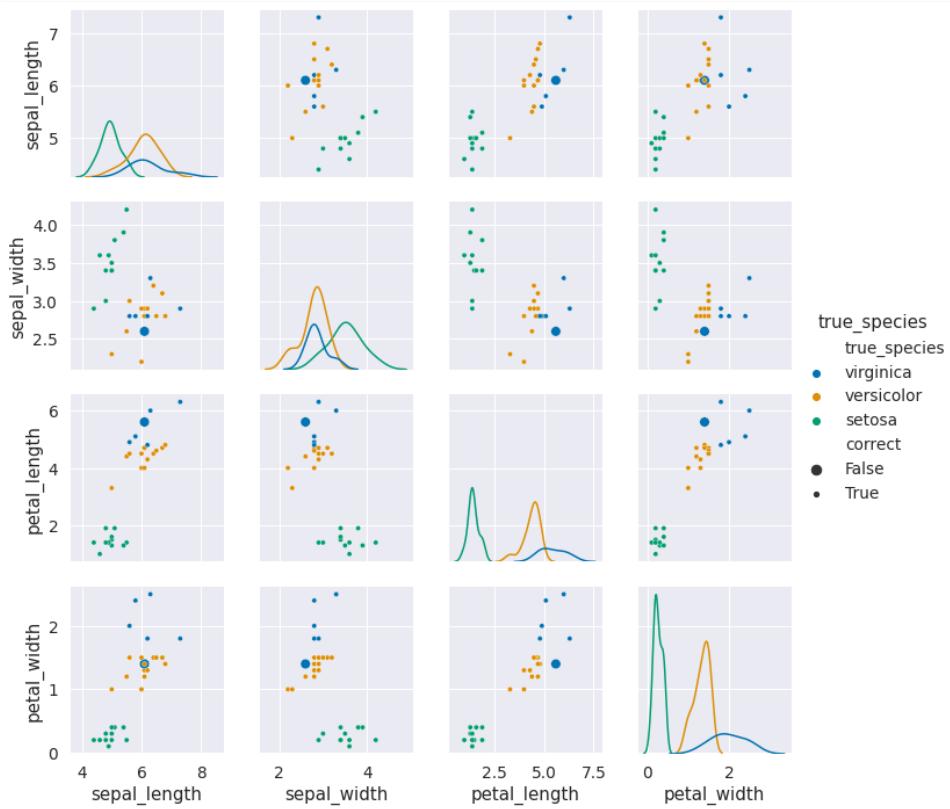
	setosa	versicolor	virginica	predicted_species	true_species	i,pred	correc
0	1.633808e-232	0.000002	0.999998	virginica	virginica	0,virginica	True
1	1.826404e-82	0.999998	0.000002	versicolor	versicolor	1,versicolor	True

```
feature_vars
```

```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
```

```
g = sns.PairGrid(prob_data_df,x_vars=feature_vars,y_vars=feature_vars,hue='true_species')
g.map_diag(sns.kdeplot)
g.map_offdiag(sns.scatterplot, size=prob_data_df["correct"])
g.add_legend()
```

```
<seaborn.axisgrid.PairGrid at 0x7fc655303390>
```



Here we see that the large dots (the incorrect ones) are all nearby to points of a different color. They were in fact samples that are similar to the other species. So again, this result makes sense and helps us see when classifiers that are a good fit for the data will still make mistakes.

We can also look at the probabilities of the predicted sample using max

```
p_predicted = np.max(gnb.predict_proba(X_test), axis=1)
p_predicted
```

```
array([0.99999781, 0.9999983 , 1. , 0.99999895, 1.
 , 1. , 0.77405211, 0.90756495, 0.99988211,
 0.52960963, 0.94376258, 0.99886436, 0.94983864, 0.98804386,
 1. , 0.98794498, 0.99994238, 1. , 1. ,
 0.98330332, 0.99275723, 1. , 1. , 0.81648378,
 1. , 1. , 0.99929123, 0.99999997, 1.])
```

We see here that most of the predictions are pretty confident. We can also use the probabilities to then compute predictions and compare these to what the `predict` method gave, to confirm that this is how the predict method works.

```
pd.DataFrame(data = gnb.predict_proba(X_test), columns = gnb.classes_).idxmax(axis=1)
==y_pred
```

```
0 True
1 True
2 True
3 True
4 True
5 True
6 True
7 True
8 True
9 True
10 True
11 True
12 True
13 True
14 True
15 True
16 True
17 True
18 True
19 True
20 True
21 True
22 True
23 True
24 True
25 True
26 True
27 True
28 True
29 True
dtype: bool
```

## 18. Midsemester feedback and Decision Trees

```
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
sns.set(palette='colorblind')
```

### 18.1. Feedback

#### Note

Analysis of the structured questions will be added

#### 18.1.1. Key takeaways

- You're learning and happy with how much you're learning
- You've noticed and appreciate that the assignments build on what we cover in class
- You're reading the notes & like them a lot
- Assignment instructions are sometimes hard to understand

```
feedback_df_raw = pd.read_csv('data/mid_sem_feedback_struct.csv')
feedback_df_raw.head()
```

			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 following what you the grac (when do/not e achieveme so far actu reflects ab y performa in the cl [How m effort I assignme
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?			
0	4	4	I think the achievements underestimate what I ...	Reflected strongly in the grading	Reflected moderately in the grading
1	5	4	I think they reflect my understanding well	Reflected strongly in the grading	Reflected perfectly in the grading
2	4	4	I think they reflect my understanding well	Reflected strongly in the grading	Reflected strongly in the grading
3	3	3	I think they reflect my understanding well	Reflected moderately in the grading	Reflected moderately in the grading
4	3	3	I think the achievements overestimate what I k...	Reflected moderately in the grading	Reflected moderately in the grading

First, we'll make the dataframe column names easier to work with and save a key of them as a dictionary that we then use to rename. First we look at them, then copy & paste them to make the dictionary.

```
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()

```

	learned	understand	achievements	grading_instructions	grading_understanding	gr
0	4	4	I think the achievements underestimate what I ...	Reflected strongly in the grading	Reflected moderately in the grading	pe
1	5	4	I think they reflect my understanding well	Reflected strongly in the grading	Reflected perfectly in the grading	pe
2	4	4	I think they reflect my understanding well	Reflected strongly in the grading	Reflected strongly in the grading	s
3	3	3	I think they reflect my understanding well	Reflected moderately in the grading	Reflected moderately in the grading	r
4	3	3	I think the achievements overestimate what I k...	Reflected moderately in the grading	Reflected moderately in the grading	r

```

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()

```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/ipykernel_launcher.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().
 This is separate from the ipykernel package so we can avoid doing imports until
```

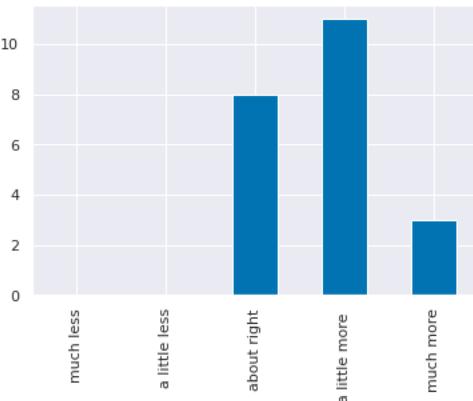
	attended Chamudi's office hours	attended Dr. Brown's office hours	attended Dr. Brown's office hours	download and run the notes	experimenting with the code from class	reading blogs or tutorials I find on my own	reading the documentation or course text
0	0	0	0	0	0	1	1
1	0	0	0	0	1	0	1
2	1	0	0	0	0	0	0
3	0	0	0	0	0	1	0
4	0	0	0	0	1	0	1

```
feedback_df = pd.concat([feedback_df_cols,learning_df])
feedback_df.head(2)
```

learned	understand	achievements	grading_instructions	grading_understanding	gr	
0	4.0	4.0	I think the achievements underestimate what I ...	Reflected strongly in the grading	Reflected moderately in the grading	pe
1	5.0	4.0	I think they reflect my understanding well	Reflected strongly in the grading	Reflected perfectly in the grading	pe

2 rows × 23 columns

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['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))
----------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------	------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------	---------------------------------------------------------------------------	-------------------------------	-----------------------------------------------------------



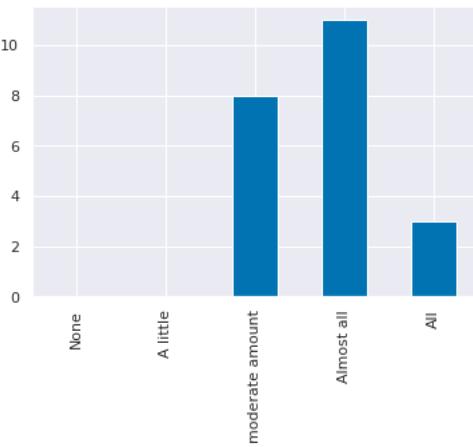
```

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['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);

```



### 18.1.2. Reminders based on requests

- all deadlines are on the Brightspace calendar, you can sync that with your own calendar/scheduling tool

### 18.1.3. Changes going forward

#### Assignments:

- clarified instructions on assignments 7+ & a note to myself to fix assignments 1-5 for next year
- a reminder when I post assignments to read before class Friday; time on Friday for clarifying questions
- Chamudi & I will meet, re: grading consistency
- [regrade request policy](#) posted on website
- more advice on choosing a dataset and some possible good ones
- will add "related notes" section
- Some skill changes are coming, will be posted soon, to make more chances on a few skills

#### In class:

- will continue working on remembering to send all of the code on prismia; feel free to post there to ask for it or raise your hand
- will denote key things that will relate to assignments as much as possible

- to make sure there's space for questions at the end of class, I'll use a google form exit ticket instead of prismia. I'll still answer all of those questions in the notes, but it makes it easier to ask both pace & follow up. It will always be at: <http://drsmb.co/310exit>

#### Office hours:

- I can change *when* but not how many hours per week, in particular no one has attended Tuesday afternoon, so I may move that.
- :

#### 18.1.4. Requests I will not fulfill

- regular in person office hours; by appointment only to ensure 1 person at a time
- more total office hours
- link directly where in the notes for assignments, part of the goal is for you to filter out the relevant parts from what we learn in a week in order to apply it. You *can* always ask questions in office hours though.

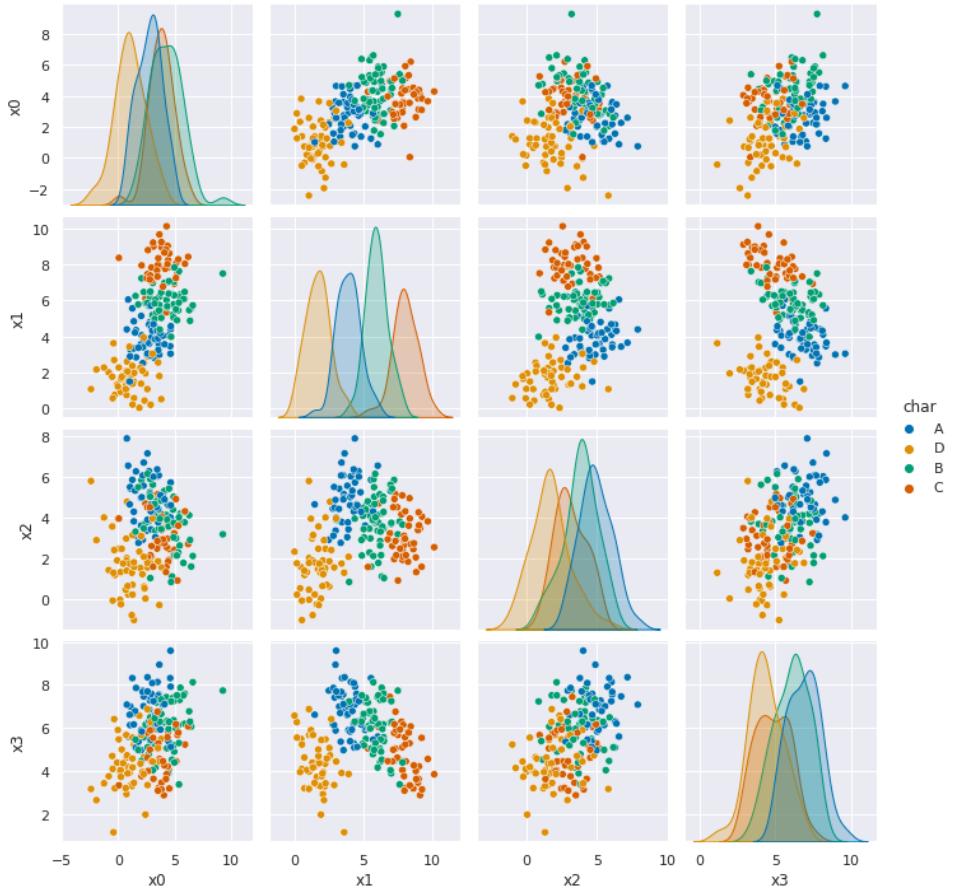
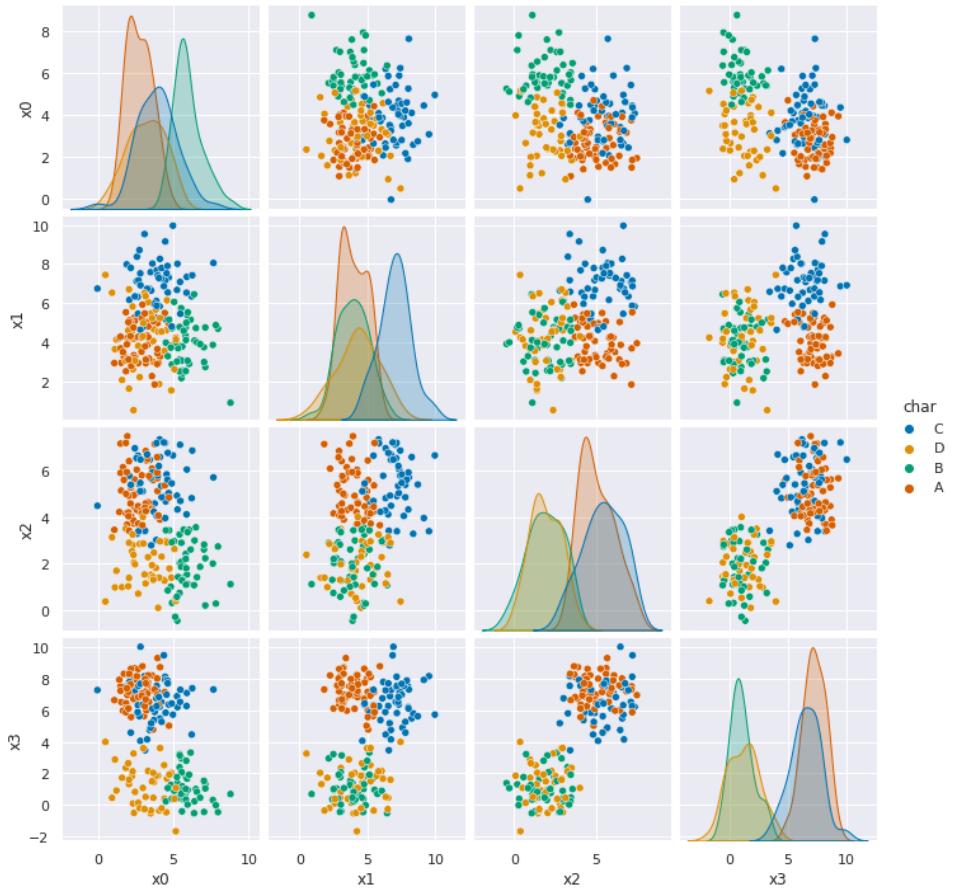
#### 18.1.5. Notes

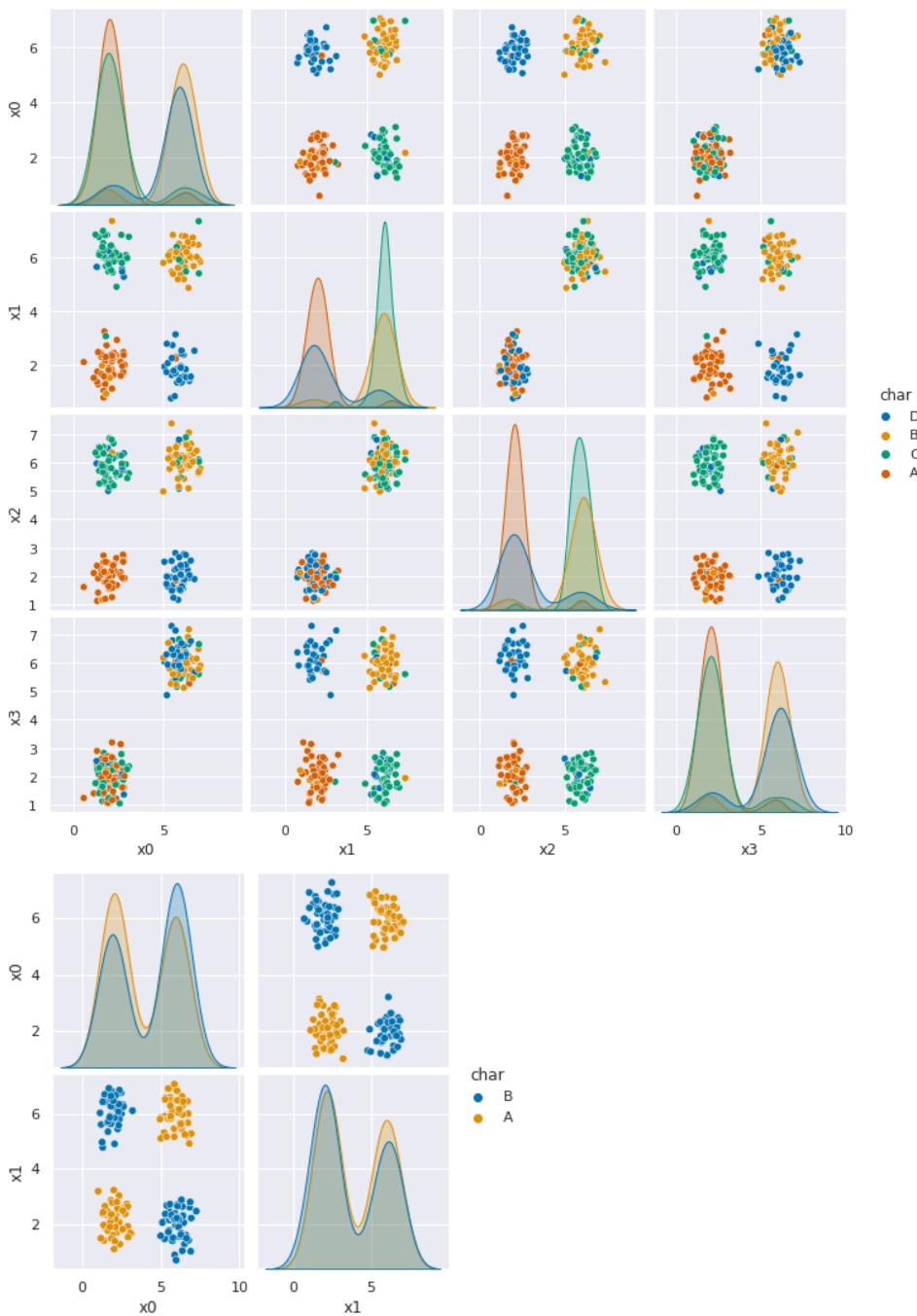
- this was anonymous, comments/updates/questions about grading specifically I cannot reply to; e-mail me if applicable
- use [this form](#) to list issues with assignment text/ portfolio requirements to help me improve them in exchange for Ram Tokens

### 18.2. A6 data review

```
a6_data = 'https://raw.githubusercontent.com/rhodyprog4ds/06-naive-
bayes/f425ba121cc0c4dd8bcaa7ebb2ff0b40b0b03bff/data/dataset'
req_datasets = [1,2,5,6]
data_urls = [a6_data + str(i) +'.csv' for i in req_datasets]
[sns.pairplot(data =pd.read_csv(url,index_col=0), hue='char') for url in data_urls]
```

```
[<seaborn.axisgrid.PairGrid at 0x7fe49fda6390>,
 <seaborn.axisgrid.PairGrid at 0x7fe497a5d310>,
 <seaborn.axisgrid.PairGrid at 0x7fe49712dc90>,
 <seaborn.axisgrid.PairGrid at 0x7fe4963cf490>]
```





Dataset 1 & 2 it should perform reasonably well. They're both Gaussian data, with some skew, but not too much and some overlap (more in 2).

Dataset 5, doesn't do so well, but its performance is clearly because there's some points in each group of points that are labeled differently than the others. This is to simulate the model for *label bias* which is one way that our traditional performance metrics can fool us. Our GNB classifier finds the four groups very reliably, but is "incorrect" on the points that are labeled differently. In real data this can occur in systematic ways, people from disadvantaged groups who otherwise may for example, qualify for a loan, have been denied historically. So, in training data they would be like the points in each group that are labeled differently.

Dataset 6, seems separable, but Guassian Naive Bayes gets about 50% accuracy.

### 18.3. Let's explore that last one...

#### Further reading

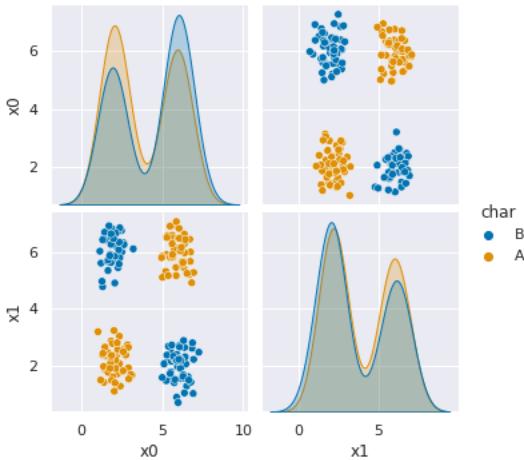
For more on label bias [a paper I co-authored with an undergrad from Brown](#) after we worked together for 2 years. It also has citations to prior work on label bias in it.

### Tip

You can earn CSC499/491 by doing research with a faculty member. If fairness is of interest to you, send me an e-mail and we can talk! I am almost always accepting new students and sometimes have funding. We can also arrange to use workstudy or other funds if you qualify. I will have 1 paid position in the spring semester and 2 in summer + you could apply for the Arts & Sciences Fellow funding.

```
df6= pd.read_csv(data_urls[-1],usecols=[1,2,3])
sns.pairplot(data=df6, hue='char')
```

```
<seaborn.axisgrid.PairGrid at 0x7fe497882b10>
```



This data is *separable* even though the classifier we saw last week doesn't work well for it, because not all of the points for each class are in a single blob. We could imagine a rule that would succeed: if 'x0' and 'x1' are both less than 4 or both more than 4, predict **A** otherwise predict **B**.

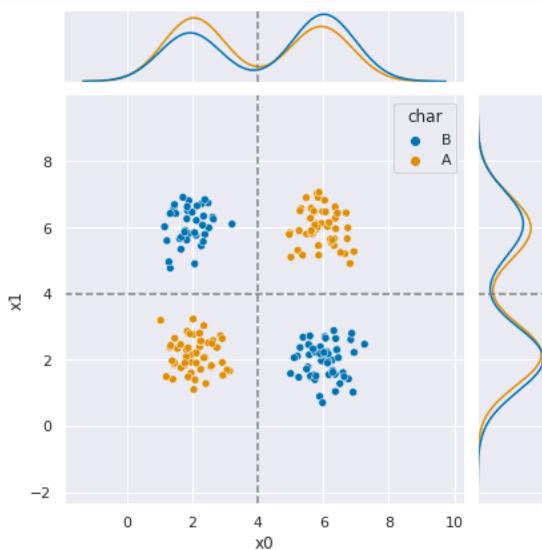
```
g = sns.JointGrid(data=df6, x='x0', y ='x1', hue='char')
g.plot_joint(sns.scatterplot)
g.plot_marginals(sns.kdeplot)
g.refline(x=4, y=4)
```

### Further Reading

I customized this plot to show additional ideas on top of the data you can read about the [JointGrid](#) in the docs.

Using it will require the most up to date version of seaborn.

```
<seaborn.axisgrid.JointGrid at 0x7fe496aa0a10>
```



The dashed line here shows that those boundaries would separate the classes. I stated that rule in a single if, but we could also imagine it like a tree, a set of binary decisions.

#### • Ram Token Opportunity

Contribute a diagram for this part of the notes

## 18.4. Learning a Decision Tree

We can learn a rule like this one from the data using a `DecisionTreeClassifier`. We'll instantiate one from the `tree` module we imported from `sklearn`.

```
dt = tree.DecisionTreeClassifier()
```

As usual, we split the data into test and train and features and labels:

```
X_train, X_test, y_train, y_test = train_test_split(df6[['x0','x1']],
 df6['char'],
 random_state = 3094)
```

We fit it just like we fit the Gaussian Naive Bayes, using the `fit` method.

```
dt.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

#### • Further Reading

- [Decision Tree Docs](#)
- [Decision Tree on Iris](#)

## 18.5. Examining a Decision Tree

Since decision trees have common functions whether they're for regression or classification, the `tree` module provides some functions that we can use.

```
plt.figure(figsize=(15,20))
tree.plot_tree(dt, rounded =True, class_names = ['A','B'],
 proportion=True, filled =True, impurity=False, fontsize=10)
```

#### • Correction

I was able to make it slightly easier to read, but this was something that I had to look up on [stackoverflow](#)

```
[Text(553.0178571428571, 978.48, 'X[0] <= 5.88\nsamples = 100.0%\nvalue = [0.493, 0.507]\nclass = B'),
Text(388.60714285714283, 761.0400000000001, 'X[1] <= 5.325\nsamples = 70.7%\nvalue = [0.557, 0.443]\nclass = A'),
Text(239.14285714285714, 543.6, 'X[0] <= 4.07\nsamples = 40.0%\nvalue = [0.733, 0.267]\nclass = A'),
Text(119.57142857142857, 326.1600000000001, 'X[1] <= 3.98\nsamples = 28.7%\nvalue = [0.953, 0.047]\nclass = A'),
Text(59.785714285714285, 108.7200000000003, 'samples = 27.3%\nvalue = [1.0, 0.0]\nclass = A'),
Text(179.35714285714286, 108.7200000000003, 'samples = 1.3%\nvalue = [0.0, 1.0]\nclass = B'),
Text(358.7142857142857, 326.1600000000001, 'X[1] <= 3.89\nsamples = 11.3%\nvalue = [0.176, 0.824]\nclass = B'),
Text(298.92857142857144, 108.7200000000003, 'samples = 9.3%\nvalue = [0.0, 1.0]\nclass = B'),
Text(418.5, 108.7200000000003, 'samples = 2.0%\nvalue = [1.0, 0.0]\nclass = A'),
Text(538.0714285714286, 543.6, 'X[0] <= 4.085\nsamples = 30.7%\nvalue = [0.326, 0.674]\nclass = B'),
Text(478.2857142857143, 326.1600000000001, 'samples = 20.7%\nvalue = [0.0, 1.0]\nclass = B'),
Text(597.8571428571429, 326.1600000000001, 'samples = 10.0%\nvalue = [1.0, 0.0]\nclass = A'),
Text(717.4285714285714, 761.0400000000001, 'X[1] <= 3.805\nsamples = 29.3%\nvalue = [0.341, 0.659]\nclass = B'),
Text(657.6428571428571, 543.6, 'samples = 19.3%\nvalue = [0.0, 1.0]\nclass = B'),
Text(777.2142857142857, 543.6, 'samples = 10.0%\nvalue = [1.0, 0.0]\nclass = A')]
```



We can also get it out as text; since it returns a plain string, we'll pass it through the print function.

```
print(tree.export_text(dt))
```

```
|--- feature_0 <= 5.88
| |--- feature_1 <= 5.33
| | |--- feature_0 <= 4.07
| | | |--- feature_1 <= 3.98
| | | | |--- class: A
| | | | |--- class: B
| | |--- feature_0 > 4.07
| | | |--- feature_1 <= 3.89
| | | | |--- class: B
| | | | |--- feature_1 > 3.89
| | | | |--- class: A
| |--- feature_1 > 5.33
| | |--- feature_0 <= 4.09
| | | |--- class: B
| | | |--- feature_0 > 4.09
| | | | |--- class: A
|--- feature_0 > 5.88
| |--- feature_1 <= 3.80
| | |--- class: B
| | |--- feature_1 > 3.80
| | | |--- class: A
```

this is the same tree as above.

### Try it yourself

Take the time to match this representation to the one above. Read through the parts of the above and see how much it tells you

This tree is many more levels (it's deeper) than the one we figured out by looking at the plotted data above. Before we experiment with changing that, let's see how well it works.

```
dt.score(X_test,y_test)
```

```
1.0
```

It does well, but let's see if we can change the depth and still do well?

## 18.6. Training parameters

For the Gaussian Naive Bayes classifier, we didn't change how it trained at all. It's a simple classifier, so it's not as impactful. Since Decision Trees are more complex, the model has hyper parameters and the training algorithm (`fit` method) has parameters.

With `sklearn` these parameters are set when the object is instantiated. We'll see more on Friday, but for now, we'll set the `max_depth`.

```
dt2 = tree.DecisionTreeClassifier(max_depth=2)
dt2.fit(X_train,y_train)

dt2.score(X_test,y_test)
```

```
0.74
```

```
print(tree.export_text(dt2))
```

```
|--- feature_0 <= 5.88
| |--- feature_1 <= 5.33
| | |--- class: A
| | |--- feature_1 > 5.33
| | | |--- class: B
|--- feature_0 > 5.88
| |--- feature_1 <= 3.80
| | |--- class: B
| | |--- feature_1 > 3.80
| | | |--- class: A
```

## 18.7. Questions After Class

18.7.1. What is it best used on?

18.7.2. Is there a maximum amount of branches for a tree

18.7.3. Why do the decision trees work for some data but not others.

18.7.4. What could we really use these trees for?

18.7.5. How is the tree created?

18.7.6. Do I use a gnb when I know the distribution is gaussian?

## 19. Decision Tree Setting and more Evaluation

```
import pandas as pd
import seaborn as sns
from sklearn import tree
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
sns.set(palette = 'colorblind') # this improves contrast

from sklearn.metrics import confusion_matrix, classification_report
```

## 20. Review

```
corner_data = 'https://raw.githubusercontent.com/rhodyprog4ds/06-naive-
bayes/f425ba121cc0c4dd8bcaa7ebb2ff0b40b0b03bff/data/dataset6.csv'
df6= pd.read_csv(corner_data,usecols=[1,2,3])
iris_df = sns.load_dataset('iris')
```

```
df6.columns
```

```
Index(['x0', 'x1', 'char'], dtype='object')
```

set up the same splits again

```
X_train, X_test, y_train, y_test = train_test_split(df6[['x0','x1']],
 df6['char'],
 random_state=34)
```

Fit a baseline model with default settings

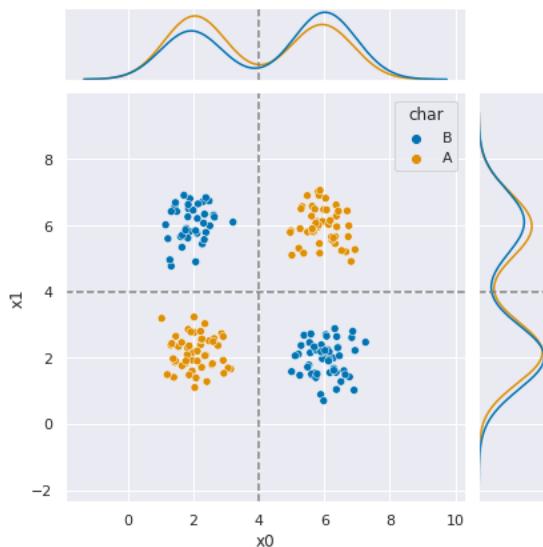
```
dt = tree.DecisionTreeClassifier()
dt.fit(X_train,y_train)
dt.score(X_test,y_test)
```

```
1.0
```

it does well, but let's examine it more closely. First, we'll look back at the data.

```
g = sns.JointGrid(data=df6, x='x0', y ='x1', hue='char')
g.plot_joint(sns.scatterplot)
g.plot_marginals(sns.kdeplot)
g.refline(x=4, y=4)
```

```
<seaborn.axisgrid.JointGrid at 0x7f485ba3ebd0>
```

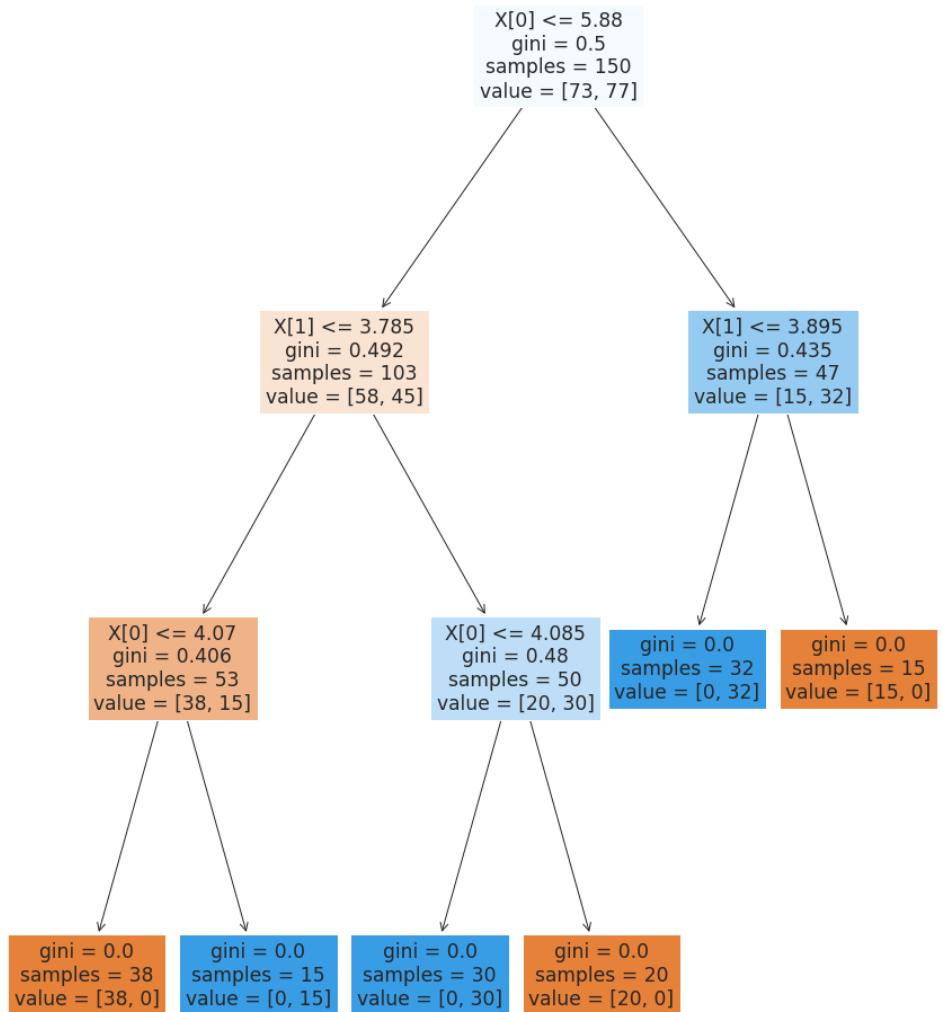


In this, the dashed lines represent boundaries that separate the two classes. Next, we can look at what it learned.

Using `filled=True` makes the nodes shaded, so we can examine it better.

```
plt.figure(figsize=(15,20))
tree.plot_tree(dt,filled=True)
```

```
[Text(494.5909090909091, 951.3000000000001, 'X[0] <= 5.88\ngini = 0.5\nsamples =
150\nvalue = [73, 77']),
Text(304.36363636364, 679.5, 'X[1] <= 3.785\ngini = 0.492\nsamples = 103\nvalue = [58,
45'],
Text(152.18181818182, 407.7000000000005, 'X[0] <= 4.07\ngini = 0.406\nsamples =
53\nvalue = [38, 15'],
Text(76.0909090909091, 135.8999999999998, 'gini = 0.0\nsamples = 38\nvalue = [38, 0']),
Text(228.272727272728, 135.8999999999998, 'gini = 0.0\nsamples = 15\nvalue = [0,
15'],
Text(456.54545454545456, 407.7000000000005, 'X[0] <= 4.085\ngini = 0.48\nsamples =
50\nvalue = [20, 30'],
Text(380.4545454545455, 135.8999999999998, 'gini = 0.0\nsamples = 30\nvalue = [0,
30'],
Text(532.6363636363636, 135.8999999999998, 'gini = 0.0\nsamples = 20\nvalue = [20,
0']),
Text(684.81818181819, 679.5, 'X[1] <= 3.895\ngini = 0.435\nsamples = 47\nvalue = [15,
32'],
Text(608.7272727272727, 407.7000000000005, 'gini = 0.0\nsamples = 32\nvalue = [0,
32'],
Text(760.909090909091, 407.7000000000005, 'gini = 0.0\nsamples = 15\nvalue = [15, 0'])]
```



Each node in the tree shows the threshold that was compared, the gini score and the number of samples from each class from the training data that pass through that node.

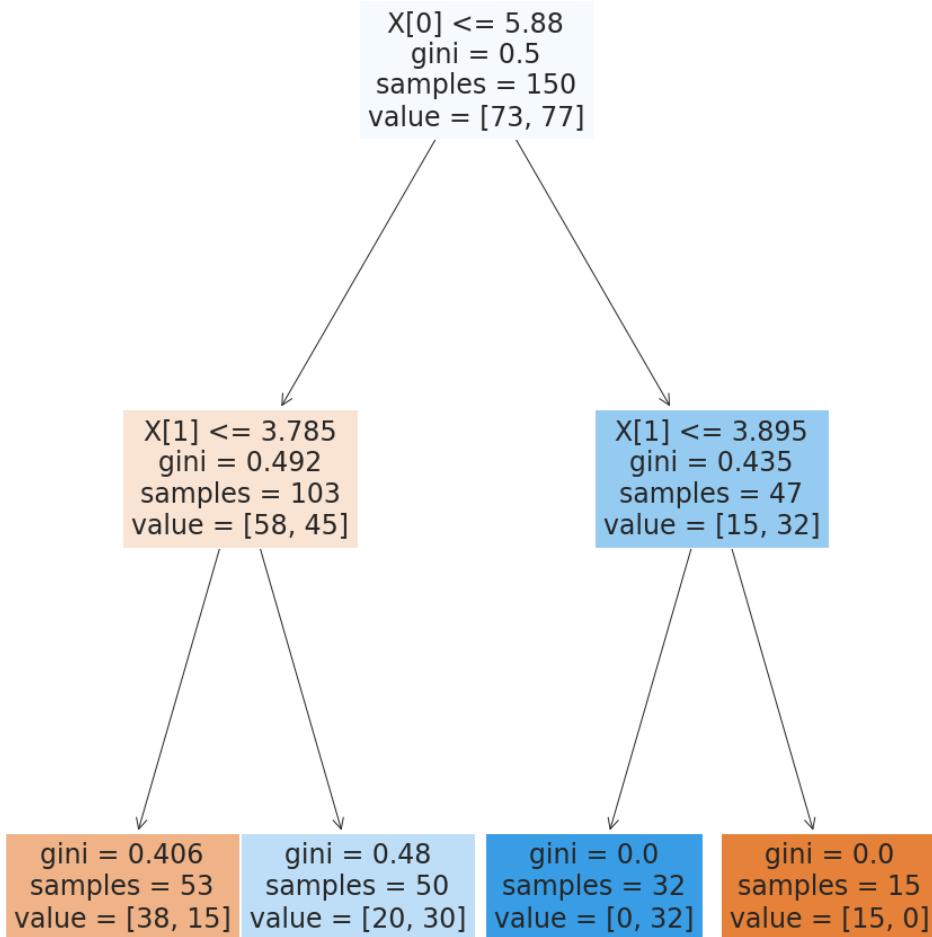
we can see from the graph what happened, but this is still not finding what we know would be the best performing decision tree.

Let's try limiting when it can split based on the share of the data.

```
dt_large_split = tree.DecisionTreeClassifier(min_samples_split=.2,
 max_depth=2)
dt_large_split.fit(X_train,y_train)
dt_large_split.score(X_test,y_test)
```

0.8

```
plt.figure(figsize=(15,20))
tree.plot_tree(dt_large_split,filled=True);
```



```
dt_large_split = tree.DecisionTreeClassifier(min_samples_split=.4,
 max_depth=2)
dt_large_split.fit(X_train,y_train)
dt_large_split.score(X_test,y_test)
```

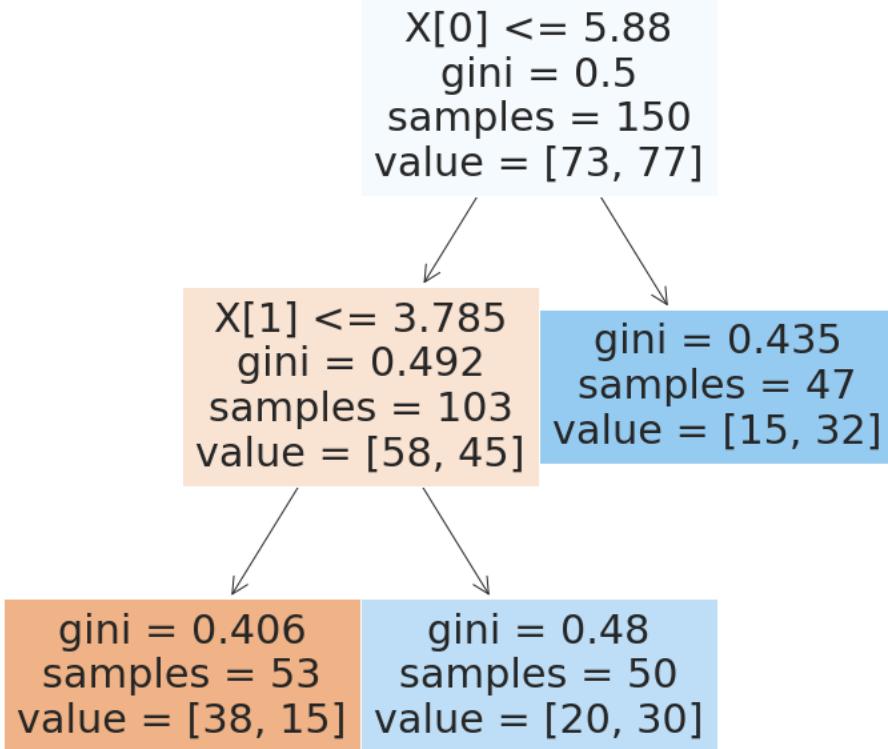
0.66

We can also limit based on how much data has to be in each leaf.

```
dt_large_leaf = tree.DecisionTreeClassifier(min_samples_leaf=.2)
dt_large_leaf.fit(X_train,y_train)
dt_large_leaf.score(X_test,y_test)
```

```
0.66
```

```
plt.figure(figsize=(12,12))
tree.plot_tree(dt_large_leaf,filled=True);
```



This one gets the right number of levels, but still does not have good performance.

## 21. More evaluation

To do more detailed evaluation than the main score, we have to get the predictions.

```
y_pred_ll = dt_large_leaf.predict(X_test)
print(classification_report(y_test,y_pred_ll))
```

	precision	recall	f1-score	support
A	0.76	0.57	0.65	28
B	0.59	0.77	0.67	22
accuracy			0.66	50
macro avg	0.67	0.67	0.66	50
weighted avg	0.68	0.66	0.66	50

We can also get the confusion matrix out

```
confusion_matrix(y_test,y_pred_ll)
```

```
array([[16, 12],
 [5, 17]])
```

We can unpack those values into individual elements, that match the [true negative](#), [false positive](#), [false negative](#), [true positive labels](#) using [ravel](#) flattens a 2d numpy array; default is 'C' row major order; can change with order param

```
tn, fp, fn, tp = confusion_matrix(y_test,y_pred_ll).ravel()
```

We can compute other metrics from the confusion matrix:

Accuracy is the true ones/ total number of samples

```
(tp + tn)/(tp+fp+fn+tn)
```

```
0.66
```

### Further Reading

More on confusion matrices, use both to match up and figure out how you can calculate other metrics from these.

- [sklearn](#)
- [wiki](#)

## 22. Parsing Assignment 7

What does data good for classification look like?

- there has to be a categorical variable to be the target
- there has to be other variables as the features where it makes sense to predict the target from those variables

Datasets for Machine Learning: the [UCI repository](#).

The new [beta](#) has a nicer interface for finding data.

You can filter by task and type of data. So far, we've only worked with tabular data and studied classification.

The screenshot shows a user interface for filtering datasets. At the top left is a blue "Options" button. To its right is a "Clear" button with a circular "X" icon. Below these are two dropdown menus: "Dataset" and "Characteristics\*". The "Dataset" menu is currently set to "Tabular", indicated by a blue circle with a white dot. The "Characteristics\*" menu is currently set to "Time-Series", indicated by a blue circle with a white dot. Below these dropdowns is a list of categories with counts: Tabular (261), Sequential (44), Time-Series (87), Text (50), and Image (3). The counts for Sequential, Time-Series, Text, and Image are displayed in red.

Type	Count
Tabular	261
Sequential	44
Time-Series	87
Text	50
Image	3

Other

111

Subject Area ▼

Associated Tasks\* ▲

Classification

261

Regression

83

Clustering

70

Other

14

# Attributes ▼

# Instances ▼

## 23. Questions after class

23.1. Is there some kind of rule of thumb to make it go faster?

## 24. More Practice

1. Write a function that uses if, else to implement the predict function of a decision tree
2. Compute the metrics from the confusion matrix (accuracy, precision, recall)
3. Apply Decision tree to the iris data

## 25. Linear 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')
```

## 25.1. Setting up a linear regression

```
tips = sns.load_dataset("tips")
```

```
tips.head(1)
```

```
total_bill tip sex smoker day time size
0 16.99 1.01 Female No Sun Dinner 2
```

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.

```
sklearn requires 2D object of features even for 1 feature
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)
```

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['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()
```

Now we fit the model.

```
regr.fit(tips_X_train,tips_y_train)
```

```
LinearRegression()
```

We can examine the coefficients and intercept.

```
regr.coef_, regr.intercept_
```

```
(array([0.0968534]), 1.0285439454607272)
```

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  $\sim 10 + \frac{1}{1.028}$ . (it's actually 9.68% +  $\frac{1}{1.028}$ )

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.

```
my_bill = np.asarray([17.78]).reshape(1,-1)
regr.predict(my_bill)
```

```
array([2.75059744])
```

We can also apply the function, as usual.

```
tips_y_pred = regr.predict(tips_X_test)
```

This gives a vector of values.

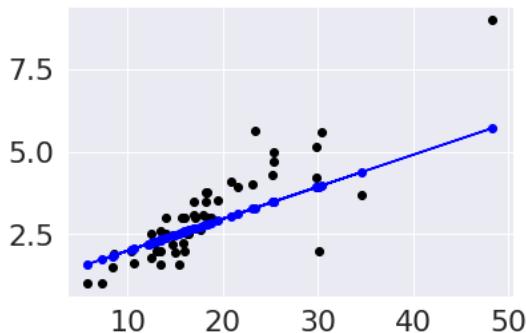
```
tips_y_pred
```

```
array([2.7321953 , 2.79999268, 2.91621676, 1.73073111, 2.60434881,
 1.58545101, 2.76415692, 3.28813383, 2.7864332 , 4.38451435,
 3.47699796, 3.47021823, 2.39127132, 2.28763818, 2.32831661,
 3.97288739, 1.83726986, 2.38449158, 2.84745085, 3.26585755,
 3.93995723, 3.05471713, 2.57819839, 2.48521912, 2.33703342,
 2.61693975, 2.20628132, 3.91477534, 3.4779665 , 2.55592211,
 2.45519457, 2.23727441, 2.52202341, 2.05422148, 2.79999268,
 2.32541101, 2.66827205, 2.02903959, 5.7094689 , 2.57626132,
 1.85954614, 2.23243174, 2.54817383, 3.91961801, 2.26439336,
 2.67214619, 2.79515001, 3.11864037, 2.68183153])
```

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.plot(tips_X_test,tips_y_pred, color='blue')
plt.scatter(tips_X_test,tips_y_pred, color='blue')
```

```
<matplotlib.collections.PathCollection at 0x7fb02e1d450>
```



## 25.2. Evaluating Regression - Mean Squared Error

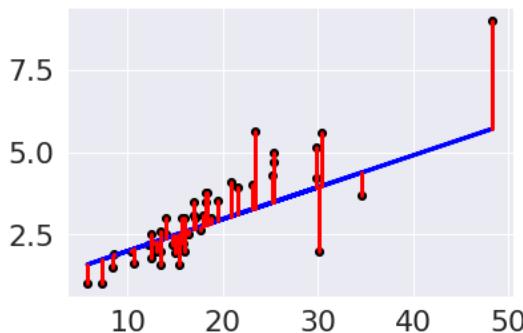
From the plot, we can see that there is some error for each point, so accuracy that we've been using, won't work. One idea is to look at how much error there is in each prediction, we can look at that visually first.

```

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 from 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)
```

```
0.821309064276629
```

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))
```

```
0.9062610353957787
```

This is equivalent to using absolute value instead

```
np.mean(np.abs(tips_y_test - tips_y_pred))
```

```
0.6564074900962107
```

## 25.3. Evaluating Regression - R2

We can also use the  $(R^2)$  regression coefficient.

```
r2_score(tips_y_test,tips_y_pred)
```

```
0.5906895098589039
```

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.

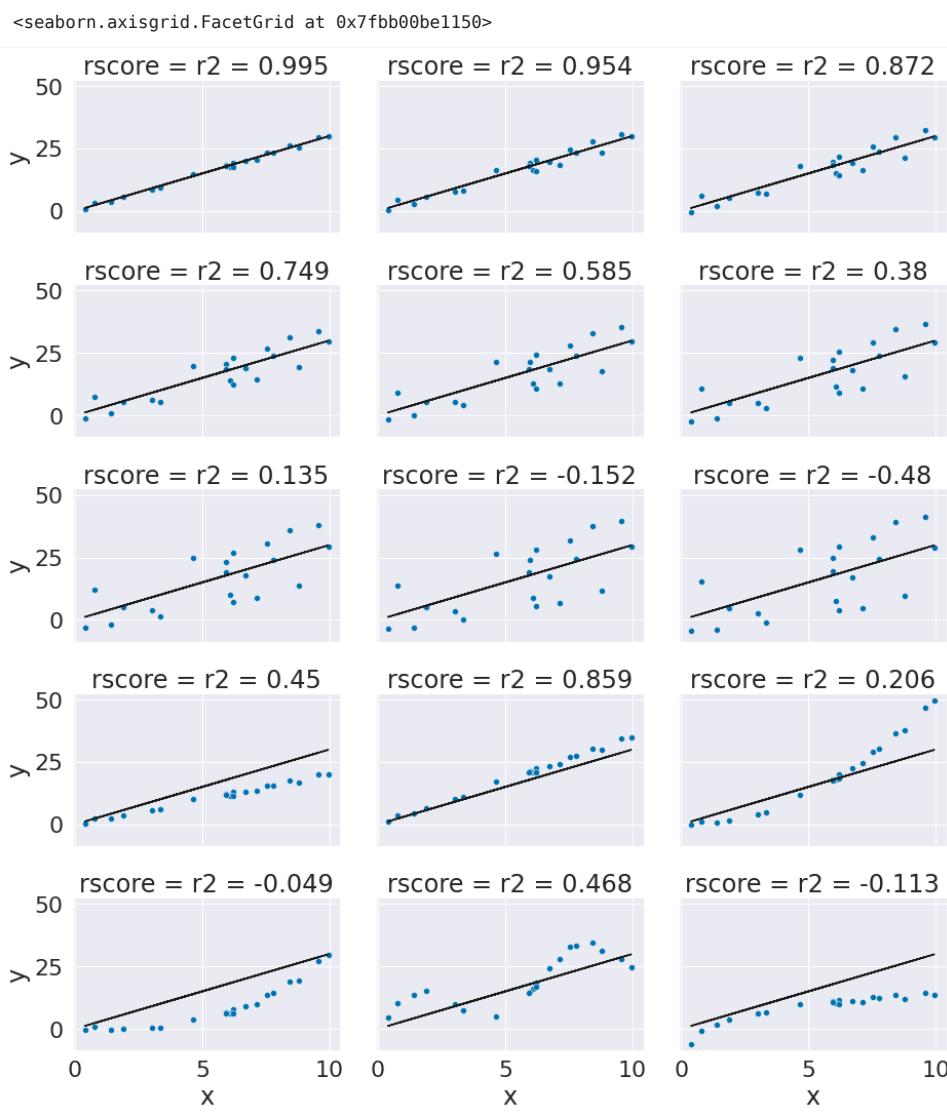
### Tip

[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.

```
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)
sample 0 mean noise
noise = (np.random.random(20)-.5)*2
add varying noise levels
for n in n_levels:
 # add noise, scaled
 y_true = y_pred + n* noise
 # compute the r2 in the column name, assign the "true" (data) here
 ex_df['r2 = '+ str(np.round(r2_score(y_pred,y_true),3))] = y_true

add functions
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
 # compute the r2 in the column name, assign the "true" (data) here
 ex_df['r2 = '+ str(np.round(r2_score(y_pred,y_true),3))] = y_true

melt the data frame for plotting
xy_df = ex_df.melt(id_vars=['x','y_pred'],var_name='rscore',value_name='y')
create a FacetGrid so that we can add two types of plots per subplot
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",)
```



## 25.4. Multivariate Regression

We can also load data from Scikit learn.

This dataset includes 10 features measured on a given date and a measure of diabetes disease progression measured one year later. The predictor we can train with this data might be something a doctor uses to calculate a patient's risk.

```
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y = True)
```

```
diabetes_X.shape
```

```
(442, 10)
```

```
diabetes_X_train, diabetes_X_test, diabetes_y_train, diabetes_y_test = train_test_split(
 diabetes_X, diabetes_y)
regr_diabetes = linear_model.LinearRegression()
```

```
regr_diabetes.fit(diabetes_X_train,diabetes_y_train)
```

```
LinearRegression()
```

## 25.5. What score does linear regression use?

```
regr_diabetes.score(diabetes_X_test,diabetes_y_test)
```

```
0.47624258453892865
```

```
diabetes_y_pred = regr_diabetes.predict(diabetes_X_test)
```

```
r2_score(diabetes_y_test,diabetes_y_pred)
```

```
0.47624258453892865
```

```
mean_squared_error(diabetes_y_test,diabetes_y_pred)
```

```
2872.3359974314435
```

It uses the R2 score.

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.

## 25.6. Questions After class

25.6.1. How I should use these with data most effectively? What is the proper use of these methods?

25.6.2. Why is that even when random state is set to 0 numbers are still a little different compared to yours and my neighbor even

## 26. Interpreting Regression

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import itertools as itr
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')
```

we'll return to the same data we used on Monday, first.

```
tips = sns.load_dataset("tips").dropna()
```

```
tips.shape
```

```
(244, 7)
```

```
tips.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

Again, we'll prepare the data.

```
sklearn requires 2D object of features even for 1 feature
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)
```

Next, we'll fit the model

```
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

This doesn't perform all that well, but let's investigate it further. We'll start by looking at the residuals

```
tips_y_pred = regr_tips.predict(tips_X_test)
```

## 26.1. Examining Residuals

The error, the difference between the predictions and the truth is called the residual.

```
tips_y_pred - tips_y_test
```

```

64 0.092195
63 -0.960007
55 -0.593783
111 0.730731
225 0.104349
92 0.585451
76 -0.315843
181 -2.361866
188 -0.713567
180 0.704514
73 -1.523002
107 -0.819782
150 -0.108729
198 0.287638
224 0.748317
44 -1.627113
145 0.337270
110 -0.615508
243 -0.152549
189 -0.734142
210 1.939957
104 -1.025283
138 0.578198
8 0.525219
199 0.337033
203 0.116940
220 0.006281
125 -0.285225
5 -1.232034
22 0.325922
74 0.255195
124 -0.282726
12 0.952023
168 0.444221
45 -0.200007
158 -0.284589
37 -0.401728
136 0.029040
212 -3.290531
223 -0.423739
222 -0.060454
118 0.432432
231 -0.451826
155 -1.220382
209 0.034393
18 -0.827854
108 -0.964850
15 -0.801360
71 -0.318168
Name: tip, dtype: float64

```

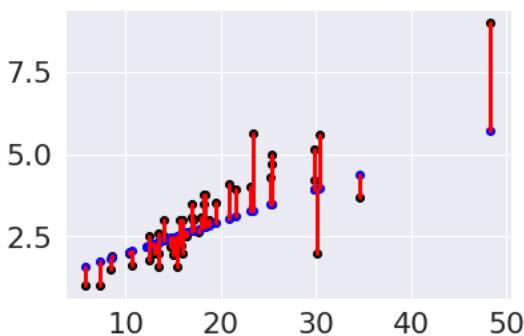
To examine these, we can plot them on the data:

```

plt.scatter(tips_X_test,tips_y_test, color='black')
plt.scatter(tips_X_test,tips_y_pred, color='blue')

[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)];

```



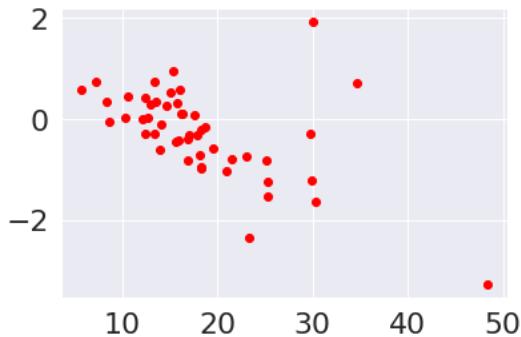
We can plot them as a scatter plot as well.

```

tips_residuals = tips_y_pred - tips_y_test
plt.scatter(tips_X_test,tips_residuals, color='red')

```

```
<matplotlib.collections.PathCollection at 0x7f9890b89d10>
```



One thing we notice is that the residuals are smaller for some values of the `total_bill` and larger for others. This suggests that there is more information left. A good fit, would have residuals that are evenly distributed, not correlated with the feature(s) in this case, the total bill.

## 26.2. Polynomial regression

Polynomial regression is still a linear problem. Linear regression solves for the  $(\beta_i)$  for a  $(d)$  dimensional problem.

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d = \sum_i \beta_i x_i$$

Quadratic regression solves for

$$y = \beta_0 + \sum_i \beta_i x_i + \sum_j \beta_{d+j} x_i x_j + \sum_i \beta_{d+2i} x_i^2$$

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  $(\beta)$  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 apply it

```
tips_X2_train = poly.fit_transform(tips_X_train)
tips_X2_test = poly.fit_transform(tips_X_test)
```

We can see what it did by looking at the shape.

```
tips_X_train.shape, tips_X2_train.shape
```

```
((195, 1), (195, 2))
```

```
tips_X2_train[:5,1]
```

```
array([722.5344, 1067.9824, 320.0521, 419.8401, 2320.3489])
```

```
tips_X_train[:5]
```

```
array([[26.88],
 [32.68],
 [17.89],
 [20.49],
 [48.17]])
```

```
tips_X2_train[:5,0]
```

```
array([26.88, 32.68, 17.89, 20.49, 48.17])
```

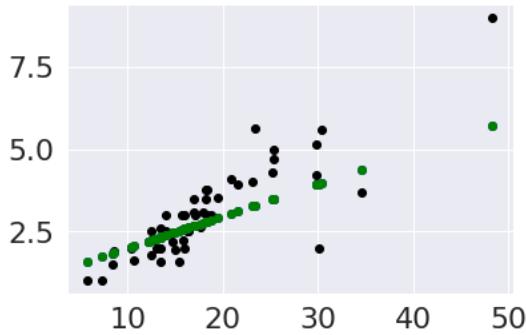
Now, we can fit a linear model on this data, which learns a weight for the data and it's squared value.

```
regr2_tips = linear_model.LinearRegression()
regr2_tips.fit(tips_X2_train,tips_y_train)
tips2_y_pred = regr2_tips.predict(tips_X2_test)
```

Then we can plot it.

```
plt.scatter(tips_X_test,tips_y_test, color='black')
plt.scatter(tips_X_test,tips_y_pred, color='blue')
plt.scatter(tips_X_test,tips2_y_pred, color='green')
```

```
<matplotlib.collections.PathCollection at 0x7f9890ada6d0>
```



We can see that this is somewhat better, the residuals are more uniformly distributed, but it doesn't look very nonlinear.

We will examine this further in the next step, but first we will drop the linear column to see the quadratic more clearly.

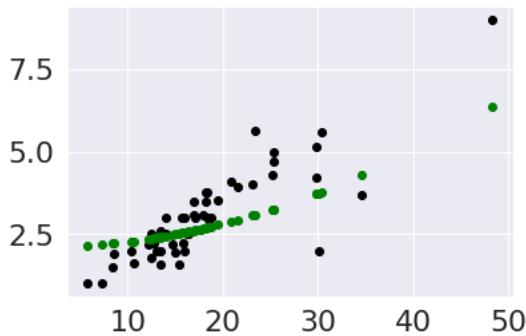
```
poly = PolynomialFeatures()

tips_Xq_train = poly.fit_transform(tips_X_train)[:, ::2]
tips_Xq_test = poly.fit_transform(tips_X_test)[:, ::2]

regr_qu_tips = linear_model.LinearRegression(fit_intercept=False)
regr_qu_tips.fit(tips_Xq_train,tips_y_train)
tips2_q_pred = regr_qu_tips.predict(tips_Xq_test)
```

```
plt.scatter(tips_X_test,tips_y_test, color='black')
plt.scatter(tips_X_test,tips2_q_pred, color='green')
```

```
<matplotlib.collections.PathCollection at 0x7f9890a87090>
```



### Try it Yourself

How would you make it cubic? what about 4th dimension?

## 26.3. Examining Coefficients

Now we can compare the coefficients. We saw above that the quadratic didn't help much, so let's look at those.

```
regr2_tips.coef_
```

```
array([9.70620903e-02, -4.18198822e-06])
```

The second parameter is very very small, so that explains why it didn't change the fit much. We can use the features to figure out how important each feature is to the prediction. Large numbers strongly influence the prediction smaller ones influence it less.

```
regr_tips.coef_
```

```
array([0.0968534])
```

## 26.4. Sparse Regression

An extreme is for some coefficients to be zero. The LASSO model, constrains some of the coefficients to be 0, so it learns simultaneously how to combine the features to predict the target and which subset of the features to use.

### Further Reading

For the mathematical formulation see the sklearn [User Guide Section on LASSO](#) and the code in [LASSO docs](#)

### Thinking Ahead

LASSO is not required for assignment 8, but is one way you could earn level 3. Here is a preview, but you can investigate it further on your own.

```
tips_lasso = linear_model.Lasso(alpha=.0025)
tips_lasso.fit(tips_all_X_train,tips_all_y_train)
tips_lasso_y_pred = tips_lasso.predict(tips_all_X_test,)
tips_lasso.score(tips_all_X_test,tips_all_y_test)
```

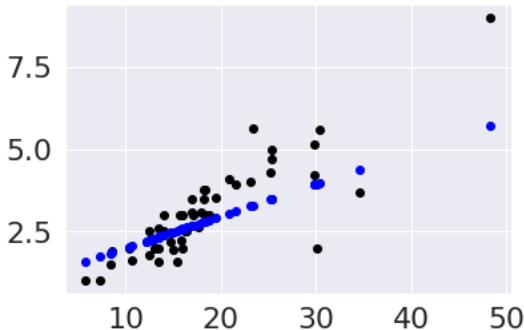
```
NameError: name 'tips_all_X_train' is not defined
```

Traceback (most recent call last)  
/tmp/ipykernel\_2296/3624267071.py in <module>  
 1 tips\_lasso = linear\_model.Lasso(alpha=.0025)  
--> 2 tips\_lasso.fit(tips\_all\_X\_train,tips\_all\_y\_train)  
 3 tips\_lasso\_y\_pred = tips\_lasso.predict(tips\_all\_X\_test,  
 4 tips\_lasso.score(tips\_all\_X\_test,tips\_all\_y\_test)

```
plt.scatter(tips_X_test,tips_y_test, color='black')
plt.scatter(tips_X_test,tips_y_pred, color='blue')
plt.scatter(tips_X_test,tips_lasso_y_pred, color='green')
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2296/1282835438.py in <module>
 1 plt.scatter(tips_X_test,tips_y_test, color='black')
 2 plt.scatter(tips_X_test,tips_y_pred, color='blue')
----> 3 plt.scatter(tips_X_test,tips_lasso_y_pred, color='green')

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



```
sum(tips_lasso.coef_ ==0)/len(tips_lasso.coef_)
```

```
AttributeError Traceback (most recent call last)
/tmp/ipykernel_2296/1981322900.py in <module>
----> 1 sum(tips_lasso.coef_ ==0)/len(tips_lasso.coef_)

AttributeError: 'Lasso' object has no attribute 'coef_'
```

```
tips_onehot.shape, tips_interacion.shape
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2296/3228345519.py in <module>
----> 1 tips_onehot.shape, tips_interacion.shape

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

The transform changed our data from 10 columns to 55.

```
tips_interacion.head
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2296/2901223600.py in <module>
----> 1 tips_interacion.head

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

## 26.5. Questions After Class

### 26.5.1. When do we do regression?

### 26.5.2. What should I look for in datasets to know whether a linear model or non-linear model is best?

### 26.5.3. How can we tell if a dataset is going to be useful through tweaking or is just not worth it?

## 27. Clustering

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn import datasets
from sklearn.cluster import KMeans
import string
import pandas as pd
```

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 in  $\{d\}$  dimensions and  $\{K\}$  clusters, we have  $\{N\}$  equations and  $\{N + K \cdot d\}$  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.

### 27.1. How does Kmeans work?

We will start with some synthetics data and then see how the clustering works.

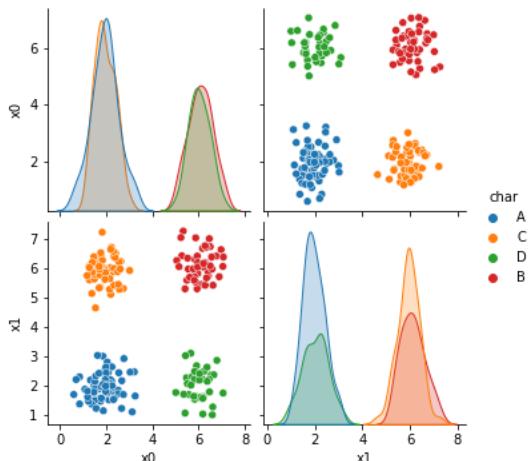
```
C = 4
N = 200
classes = list(string.ascii_uppercase[:C])
mu = {c: i for c, i in zip(classes, [[2,2], [6,6], [2,6],[6,2]])}
sigma = {c: i*.5 for c, i in zip(classes,np.random.random(4))}
sigma

target5 = np.random.choice(classes,N)
data5 = [np.random.multivariate_normal(mu[c],.25*np.eye(2)) for c in target5]
df5 = pd.DataFrame(data = data5,columns = ['x' + str(i) for i in range(2)]).round(2)

df5['char'] = target5

sns.pairplot(data =df5, hue='char')
```

```
<seaborn.axisgrid.PairGrid at 0x7fe2296e1210>
```



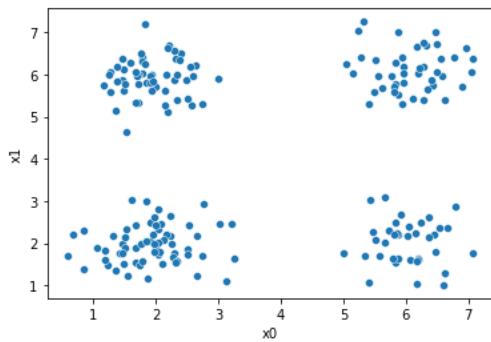
First, we'll pick just the data columns.

```
df = df5[['x0','x1']]
df.head()
```

```
x0 x1
0 1.68 1.55
1 1.48 6.12
2 2.11 1.52
3 5.83 1.54
4 5.47 2.27
```

```
sns.scatterplot(data=df,x='x0',y='x1')
```

```
<AxesSubplot:xlabel='x0', ylabel='x1'>
```



Next, we'll pick 4 random points to be the starting points as the means.

```
K = 4
mu0 = df.sample(n=K).values
mu0
```

```
array([[1.56, 1.23],
 [1.69, 6.05],
 [5.51, 2.07],
 [1.27, 5.59]])
```

Now, we will compute, for 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]
```

```
[0 0.1168
1 23.9185
2 0.3866
3 18.3290
4 16.3697
...
195 3.0325
196 30.5204
197 18.6457
198 24.4217
199 45.8289
Length: 200, dtype: float64,
0 20.2501
1 0.0490
2 20.6973
3 37.4797
4 28.5768
...
195 21.7936
196 47.3269
197 1.0610
198 40.9960
199 20.4404
Length: 200, dtype: float64,
0 14.9393
1 32.6434
2 11.8625
3 0.3833
4 0.0416
...
195 5.2840
196 2.5049
197 20.2970
198 1.0000
199 17.1364
Length: 200, dtype: float64,
0 16.4897
1 0.3250
2 17.2705
3 37.1961
4 28.6624
...
195 19.4440
196 48.2697
197 1.5850
198 41.4800
199 24.7172
Length: 200, dtype: float64]
```

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()
```

	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>
<b>0</b>	0.1168	20.2501	14.9393	16.4897
<b>1</b>	23.9185	0.0490	32.6434	0.3250
<b>2</b>	0.3866	20.6973	11.8625	17.2705
<b>3</b>	18.3290	37.4797	0.3833	37.1961
<b>4</b>	16.3697	28.5768	0.0416	28.6624

Now we have one row per sample and one column per mean, 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()
```

```
0 0
1 1
2 0
3 2
4 2
dtype: int64
```

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()
```

	x0	x1	0
0	1.68	1.55	0
1	1.48	6.12	1
2	2.11	1.52	0
3	5.83	1.54	2
4	5.47	2.27	2

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 paired
cmap_pt = sns.color_palette('tab20',8)[1::2] # starting from 1
cmap_mu = sns.color_palette('tab20',8)[0::2] # starting from 0
```

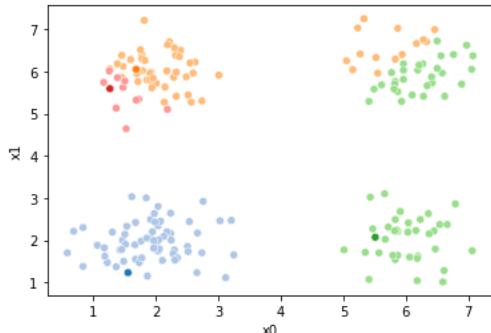
### Further Reading

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)
```

```
<AxesSubplot:xlabel='x0', ylabel='x1'>
```



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.

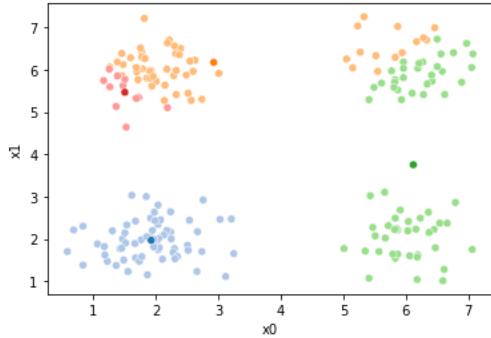
```
mu1 = df.groupby('0')[data_cols].mean().values
```

```
array([[1.9303125 , 1.96625],
 [2.92913793, 6.17068966],
 [6.11746269, 3.75268657],
 [1.50818182, 5.46454545]])
```

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)
```

<AxesSubplot:xlabel='x0', ylabel='x1'>



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.

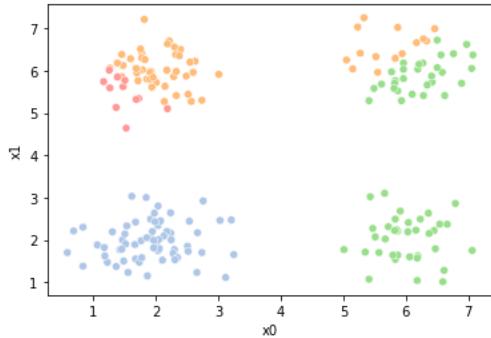
So, again we can update the assignments.

```
df['1'] = pd.concat([(df[data_cols]-mu_i)**2).sum(axis=1) for mu_i in
mu1],axis=1).idxmin(axis=1)
```

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)
```

<AxesSubplot:xlabel='x0', ylabel='x1'>



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.

```

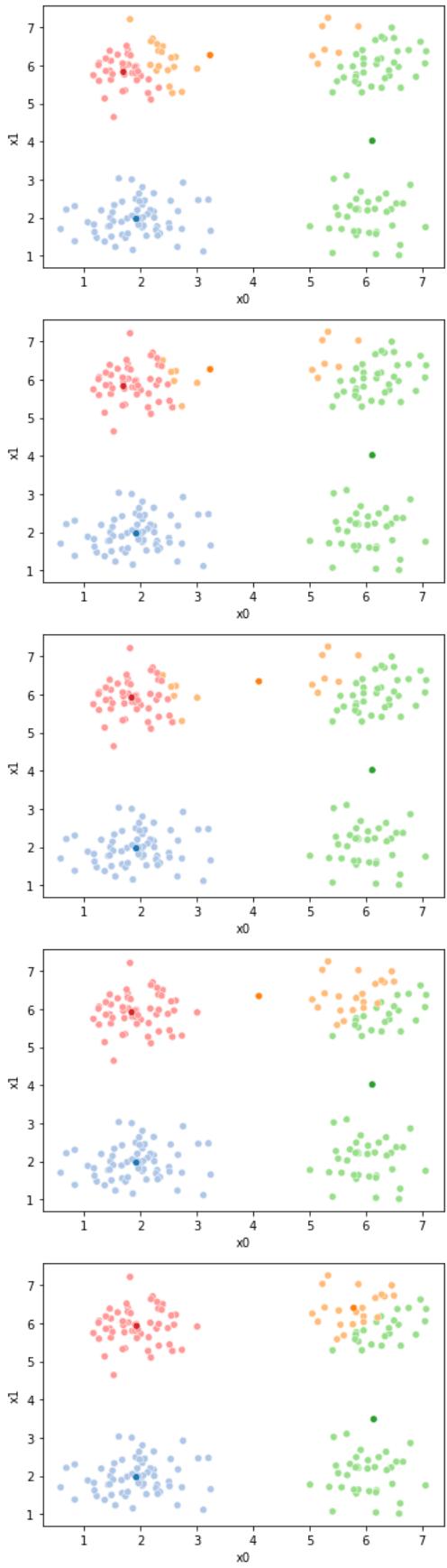
i =1
mu_list = [mu_to_df(mu0,i), mu_to_df(mu1,i)]
cur_old = str(i-1)
cur_new = str(i)
while sum(df[cur_old] !=df[cur_new]) >0:
 cur_old = cur_new
 i +=1
 cur_new = str(i)
 # update the means and plot with current generating assignments
 mu = df.groupby(cur_old)[data_cols].mean().values
 mu_df = mu_to_df(mu,i)
 mu_list.append(mu_df)

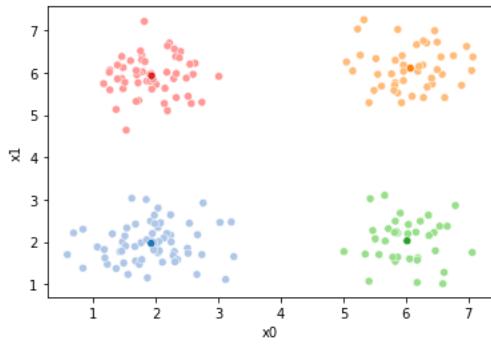
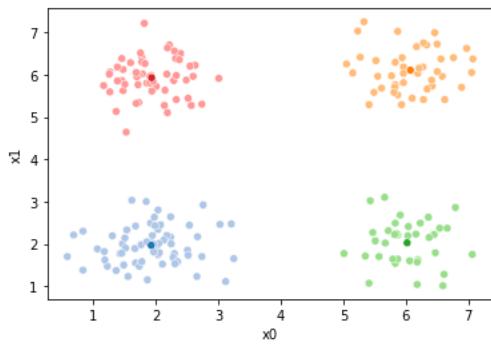
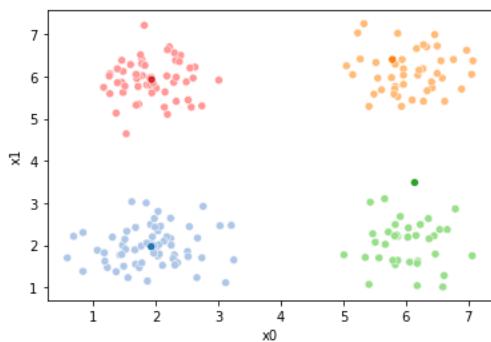
 fig = plt.figure()
 sfig = sns.scatterplot(data
=df,x='x0',y='x1',hue=cur_old,palette=cmap_pt,legend=False)
 sns.scatterplot(data
=mu_df,x='x0',y='x1',hue='class',palette=cmap_mu,ax=sfig,legend=False)

 # update the assignments 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)

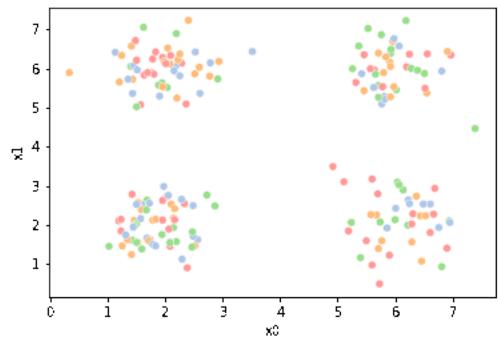
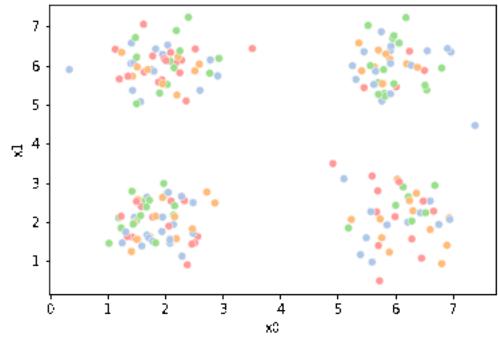
n_iter = i

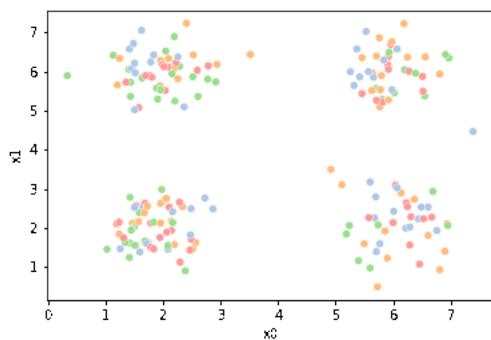
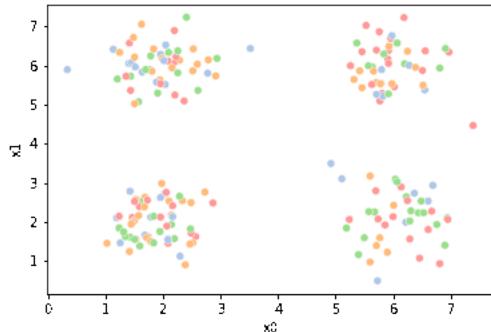
```





This algorithm is random. So each time we run it, it looks a little different.





## 27.2. Questions After Class

### **i Ram token Opportunity**

Contribute a question as an issue or contribute a solution to one of the try it yourself above.

## 28. Clustering

### 28.1. KMeans review

Review the notes from [Monday](#), including the gifs at the bottom for more illustration of what kmeans does under the hood. Ask any open questions on Prismia

### 28.2. Clustering with Sci-kit Learn

Read through this notebook and answer questions either by adding code or typing in markdown cells.

For hints about code, switch the markdown cells with the prompt to edit mode. There are hints in them stored as html(markdown) comments.

```
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')

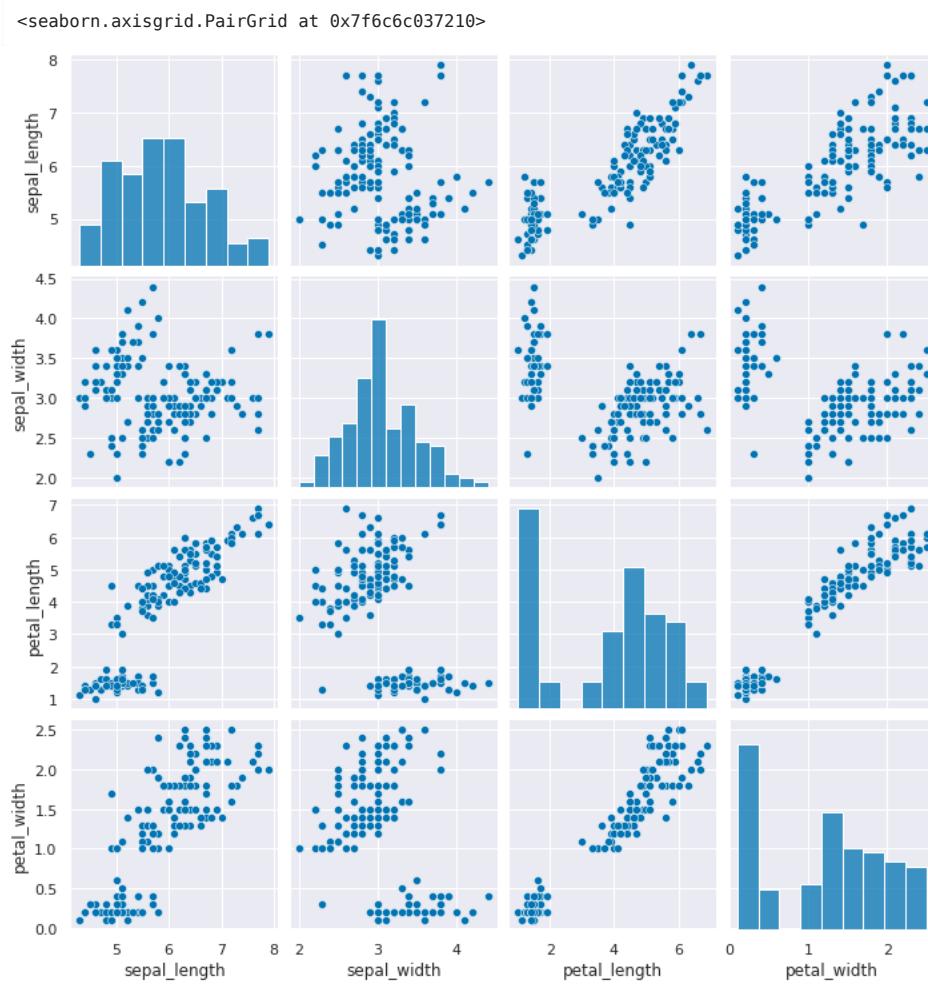
set global random seed so that the notes are the same each time the site builds
np.random.seed(1103)
```

Load the iris data from seaborn

```
iris_df = sns.load_dataset('iris')
```

Create a grid of scatterplots of the data, without coloring the points differently

```
sns.pairplot(iris_df)
```



Create 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.

### 💡 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 picking out the measurement columns, but we could also drop the species for now.

```
measurement_cols = ['sepal_length', 'petal_length', 'sepal_width', 'petal_width']
iris_X = iris_df[measurement_cols]
iris_X = iris_df.drop(columns=['species']) # equivalent to above
```

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.

### ℹ️ Question

How do we know there are three classes? didn't' 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.

A yellow lightbulb icon with a glowing yellow bulb and a black outline.

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.

```
km = KMeans(n_clusters=3)
```

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.

Use the `fit_predict` method and look at what it outputs.

```
km.fit_predict(iris_X)
```

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)
```

Use the `get_params` method to look at the parameters. Read the documentation to see what they mean.

```
km.get_params(deep=True)
```

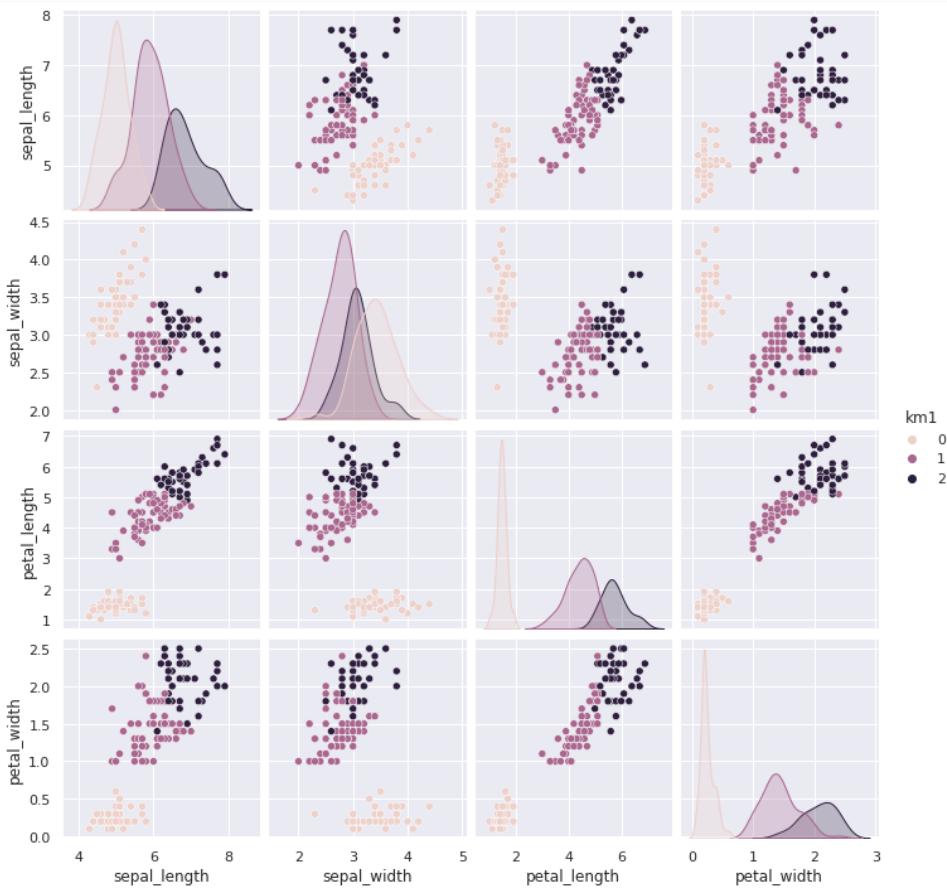
```
{'algorithm': 'auto',
 'copy_x': True,
 'init': 'k-means++',
 'max_iter': 300,
 'n_clusters': 3,
 'n_init': 10,
 'random_state': None,
 'tol': 0.0001,
 'verbose': 0}
```

### 28.3. 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')
```

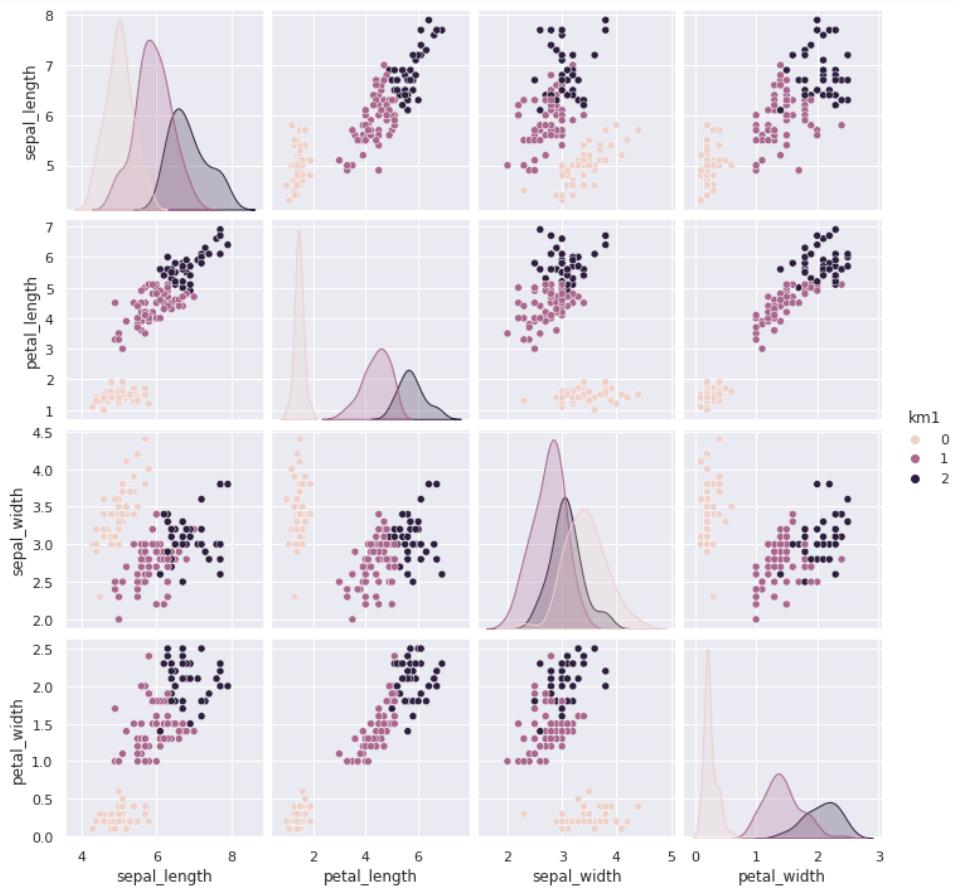
```
<seaborn.axisgrid.PairGrid at 0x7f6c6c0376d0>
```



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.

```
iris_df['km1'] = cluster_assignments
sns.pairplot(data=iris_df, hue='km1', vars=measurement_cols)
```

```
<seaborn.axisgrid.PairGrid at 0x7f6c2952d950>
```

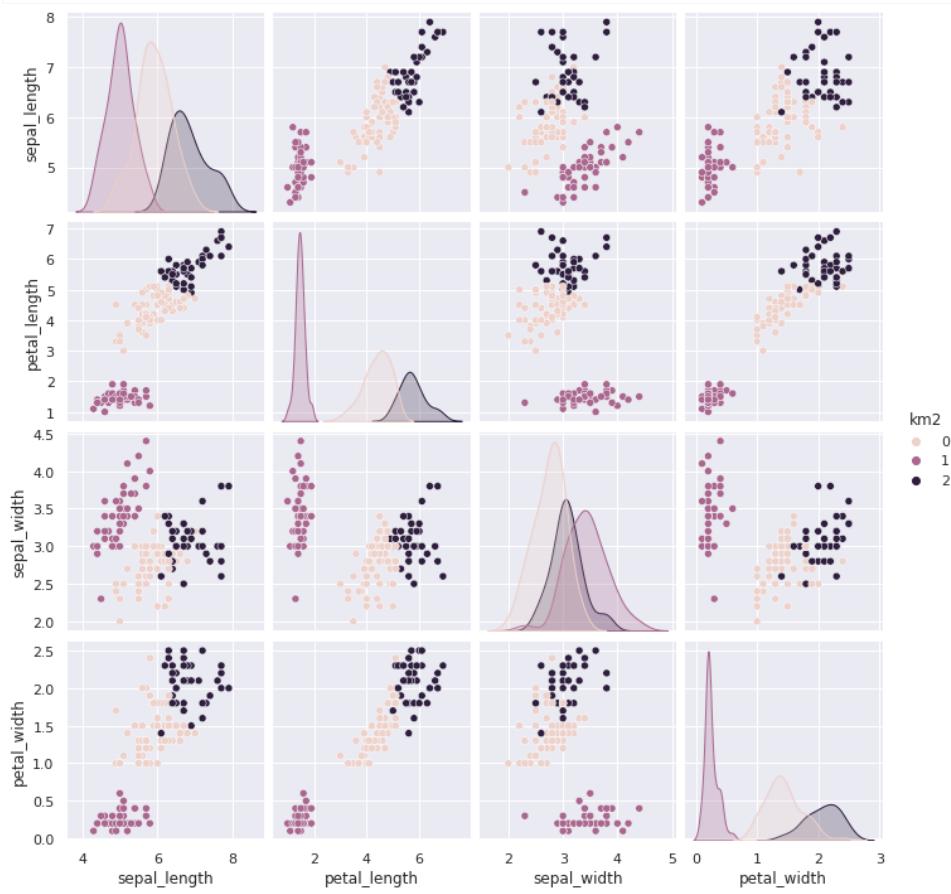


## 28.4. Clustering Persistence

We can run kmeans a few more times and plot each time and/or compare with a neighbor/ another group.

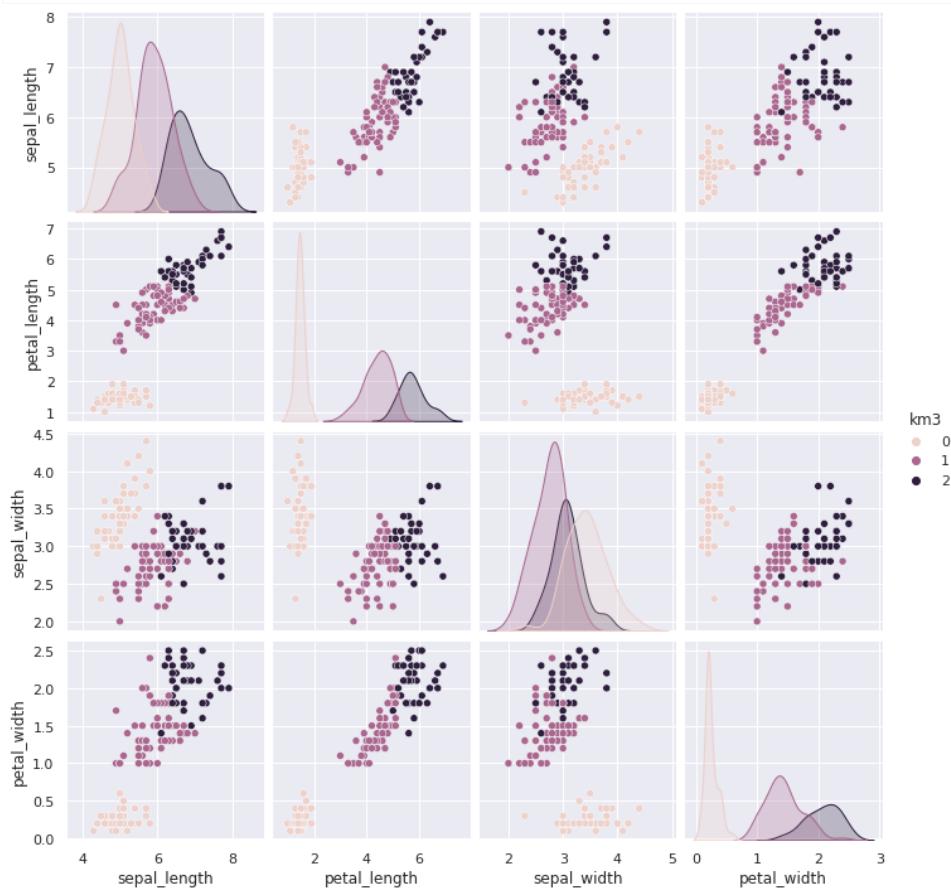
```
iris_df['km2'] = km.fit_predict(iris_X)
sns.pairplot(data=iris_df, hue='km2', vars=measurement_cols)
```

```
<seaborn.axisgrid.PairGrid at 0x7f6c1b7aea90>
```



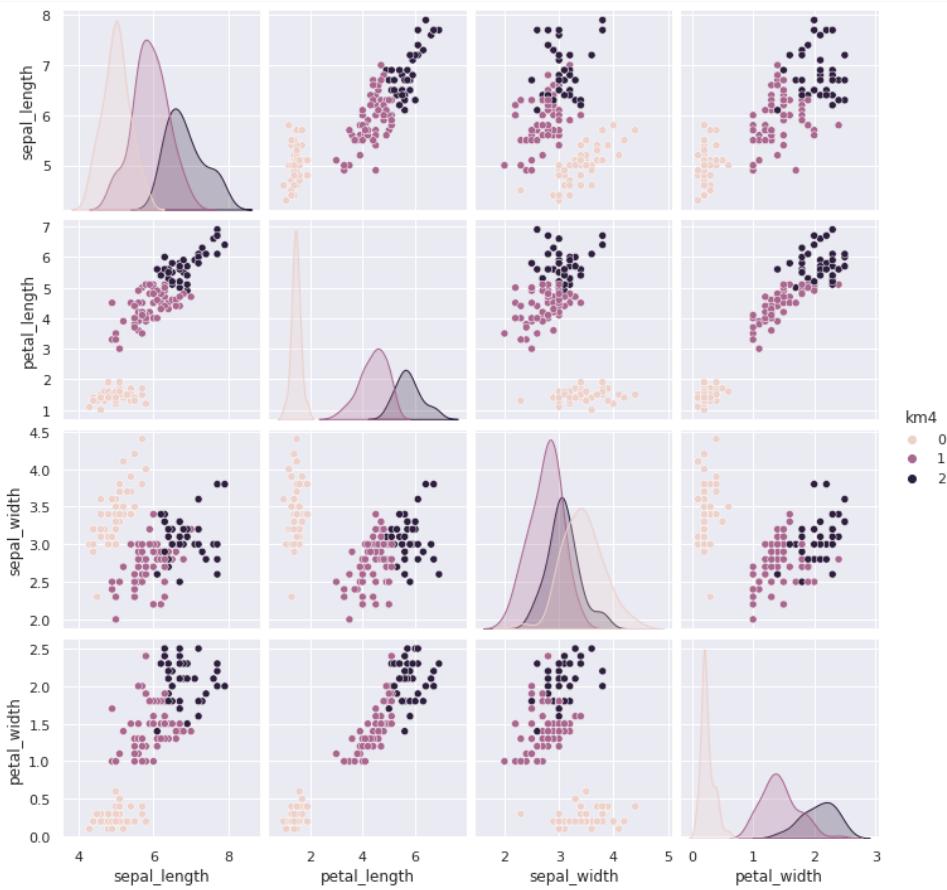
```
iris_df['km3'] = km.fit_predict(iris_X)
sns.pairplot(data=iris_df, hue='km3', vars=measurement_cols)
```

```
<seaborn.axisgrid.PairGrid at 0x7f6c1a72ee90>
```



```
iris_df['km4'] = km.fit_predict(iris_X)
sns.pairplot(data=iris_df, hue='km4', vars=measurement_cols)
```

```
<seaborn.axisgrid.PairGrid at 0x7f6c1bd66bd0>
```



We could also use a loop (or list comprehension) to repeat kmeans multiple times.

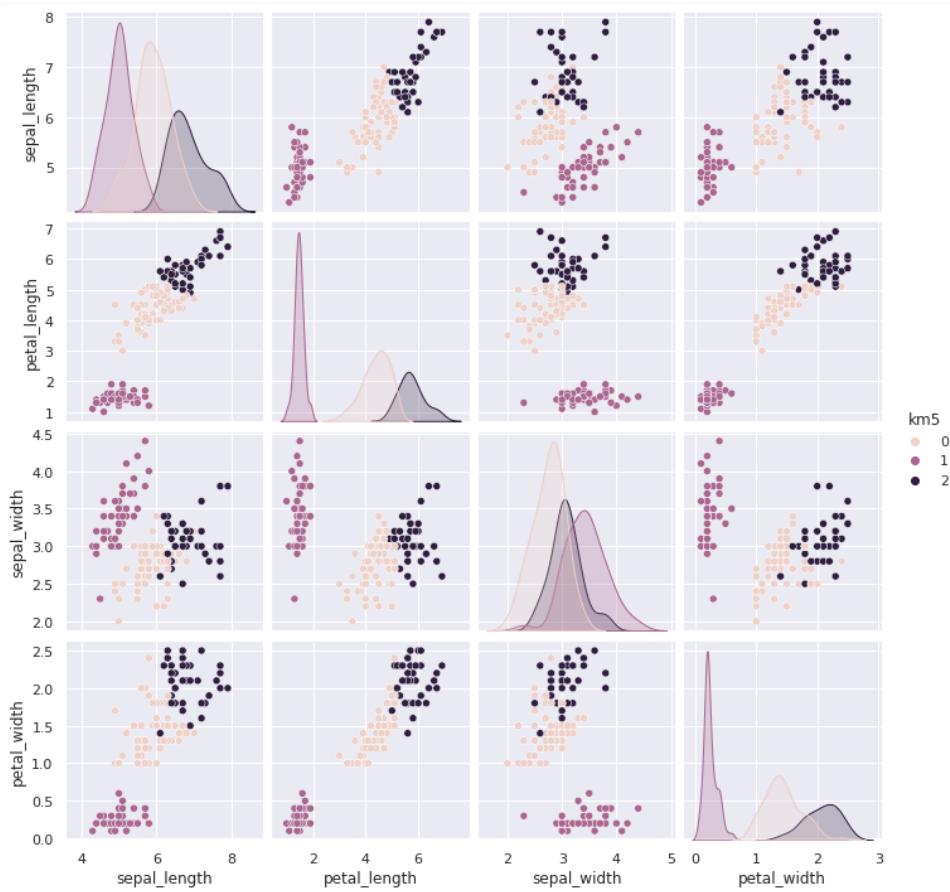
```
for i in [5,6,7]:
 iris_df['km' + str(i)] = km.fit_predict(iris_X)

sns.pairplot(data=iris_df, hue='km5', vars=measurement_cols)
```

### Tip

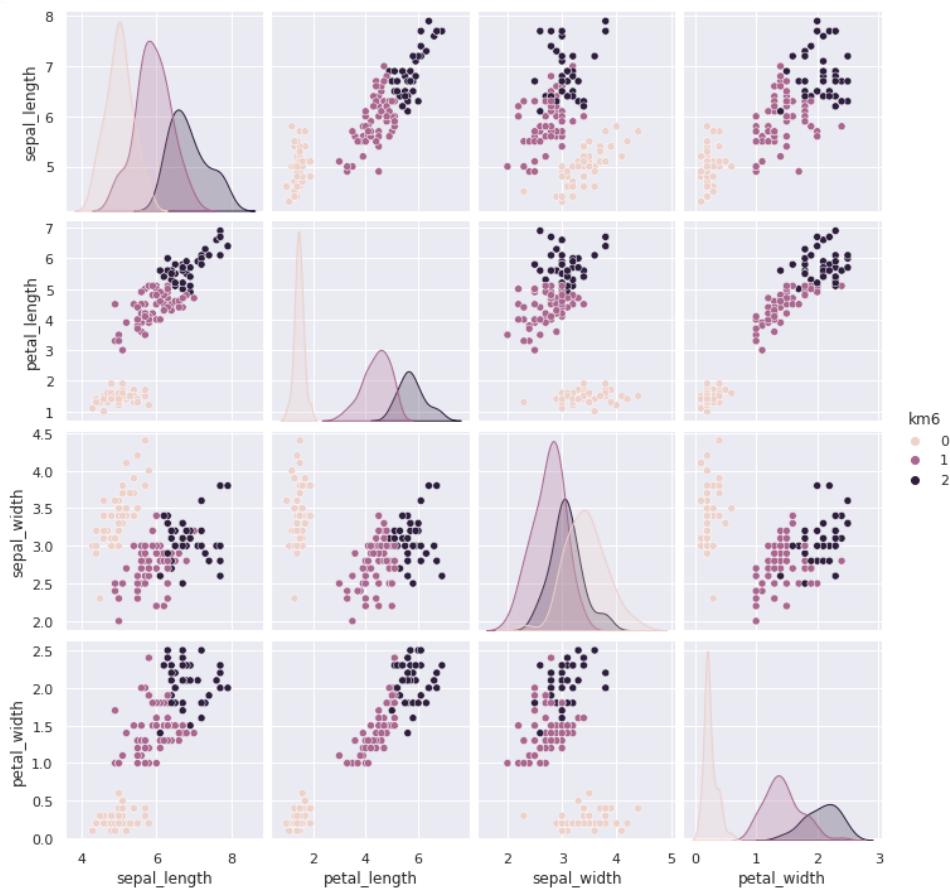
using the `i` as a loop variable here makes sense since we're actually just repeating for the sake of repeating

```
<seaborn.axisgrid.PairGrid at 0x7f6c1ac8c250>
```

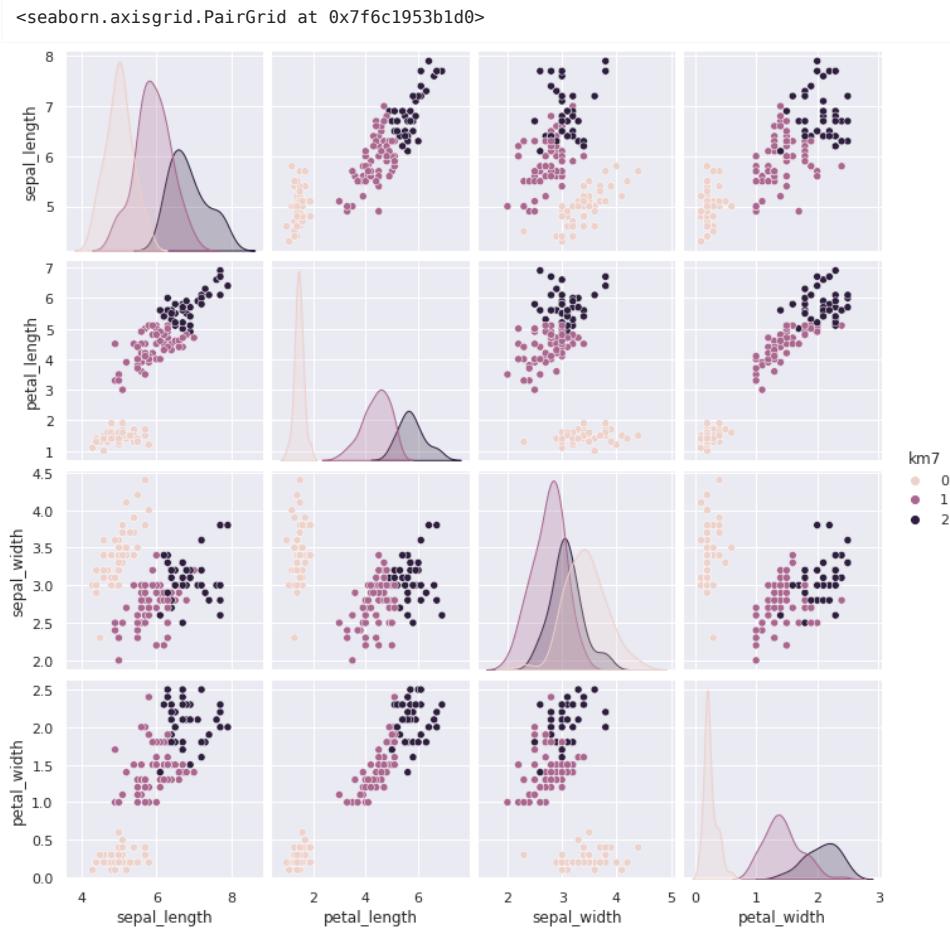


```
sns.pairplot(data=iris_df, hue='km6', vars=measurement_cols)
```

```
<seaborn.axisgrid.PairGrid at 0x7f6c19268c50>
```



```
sns.pairplot(data=iris_df, hue='km7', vars=measurement_cols)
```



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.

## 28.5. Questions Before and After Class

### 28.5.1. How can I do linear regression with multiple features

### 28.5.2. How can we do multiple iterations of KMeans easily?

### 28.5.3. If my results do not form good distinct clusters what does it mean?

## 28.5.4. Are there more ways to tell how accurate the model is

# 29. Evaluating Clustering

```
import seaborn as sns
import string
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn import metrics
import pandas as pd
```

We'll continue with the iris dataset because it's visually clear.

```
measurement_cols = ['sepal_length', 'petal_length', 'sepal_width', 'petal_width']

iris_df = sns.load_dataset('iris')
iris_X = iris_df[measurement_cols]
```

## 29.1. Clustering with KMeans

```
km3 = KMeans(n_clusters=3)
```

```
km3.fit(iris_X)
```

```
KMeans(n_clusters=3)
```

```
km3.labels_
```

```
array([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,
 2, 2, 2, 1, 1, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 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, 2, 1], dtype=int32)
```

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 closer to points in the same cluster, but we also want that the clusters aren't just touching, but actually spaced apart, if the clustering actually captures meaningfully different groups.

## 29.2. Silhouette Score

The [Silhouette score](#) computes a ratio of how close points are to points in the same cluster vs other clusters.

$$\text{if } s = \frac{b-a}{\max(a,b)}$$

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 b will be close in value, and the s will be close 0. These are overlapping clusters, or possibly too many clusters for this data.

### Further Reading

See the [user guide](#) for more detail on this score including citations to source materials and advantages and disadvantages.

Let's check our clustering solution:

```
metrics.silhouette_score(iris_X,km3.labels_)
```

```
0.5528190123564098
```

```
km2 = KMeans(n_clusters=2)
km4 = KMeans(n_clusters=4)
km2.fit(iris_X)
km4.fit(iris_X)
```

```
KMeans(n_clusters=4)
```

```
metrics.silhouette_score(iris_X,km2.labels_)
```

```
0.6810461692117464
```

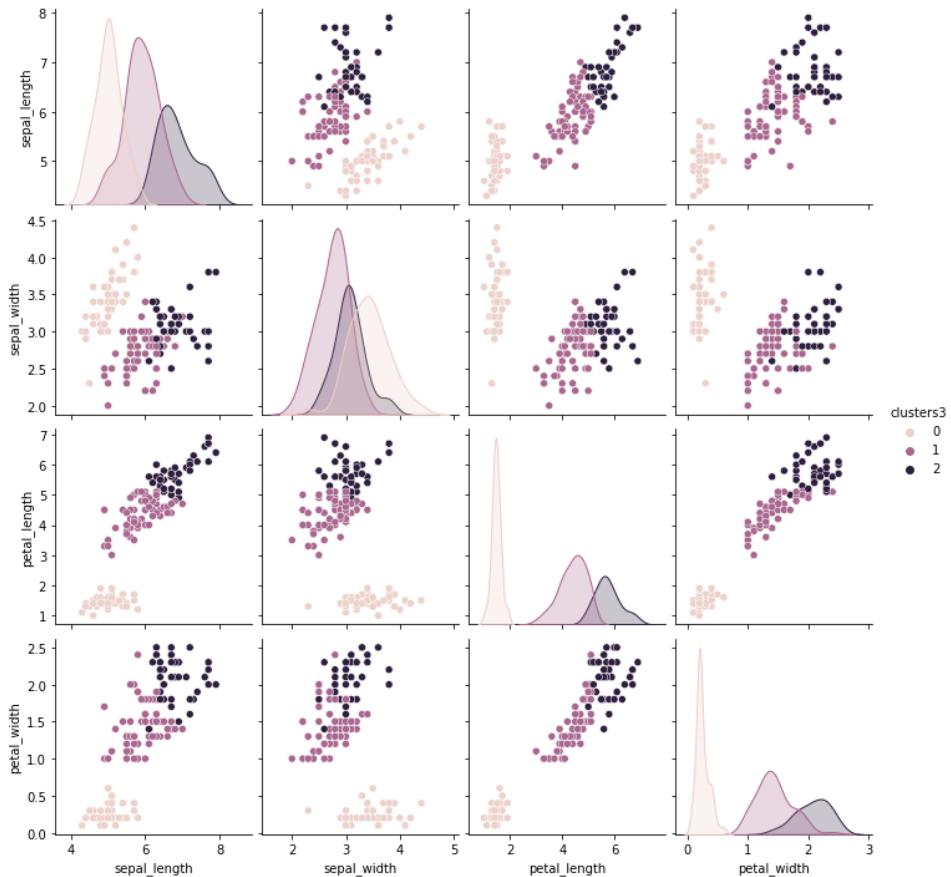
```
metrics.silhouette_score(iris_X,km4.labels_)
```

```
0.49805050499728815
```

```
iris_df['clusters3'] = km3.labels_
```

```
sns.pairplot(data= iris_df,hue= 'clusters3',)
read docs to figure out hwy it didnt' plot clusters3
```

```
<seaborn.axisgrid.PairGrid at 0x7fc57f162a50>
```



## 29.3. 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]
df = pd.DataFrame(data = data,columns = ['x' + str(i) for i in range(2)])

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)
kmrK2.fit(df[['x0','x1']])
df['km' + str(K*2)] = kmrK2.labels_

assign every point to its own cluster
df['id'] = list(range(N))

df.head()
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2395/700309636.py in <module>
 6
 7 # sample random cluster assignments
----> 8 target = np.random.choice(classes,N)
 9
 10 # sample points with different means according to those assignments

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

```
sns.pairplot(data =df, hue='char')
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2395/3639306760.py in <module>
----> 1 sns.pairplot(data =df, hue='char')

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

### Try it yourself

Compute the score for each and make plots.

## 29.4. 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

### Further Reading

Other types of clustering:

[sklearn overview](#)

[classifier comparison](#)

### Further Reading

Sklearn provides many

[mutual information based](#)

[scores](#).

See the user guide for definitions of each, pros and cons and examples.

physics.

The `mutual_info_score` method in the `metrics` module computes mutual information.

```
metrics.mutual_info_score(iris_df['species'],km2.labels_)
```

```
metrics.mutual_info_score(iris_df['species'],km3.labels_)
```

```
metrics.mutual_info_score(iris_df['species'],km4.labels_)
```

There's some random chance of getting things correct, so the adjusted mutual information corrects for that.

#### Try it yourself

See how adjusted mutual information and mutual information compare on the iris data, the synthetic data above, or even use it in assignment 9.

## 29.5. Questions After Class

29.5.1. How I would use all of those different clustering algorithms

29.5.2. How exactly do I find numbers of clusters

29.5.3. what do you mean by the data is represented in the 4th dimension?

29.5.4. How do we utilize these clusters once we have them?

29.5.5. How do we interpret the cluster labels?

29.5.6. How does Kmeans work in the simplest terms?

#### Important

if you don't see your question and the above don't answer it, then I misunderstood your question (the last two are my significant rephrasing of questions I had trouble with). If that's the case, **please** reach out to help me understand what you're confused about.

## 30. ML Task Review and Cross Validation

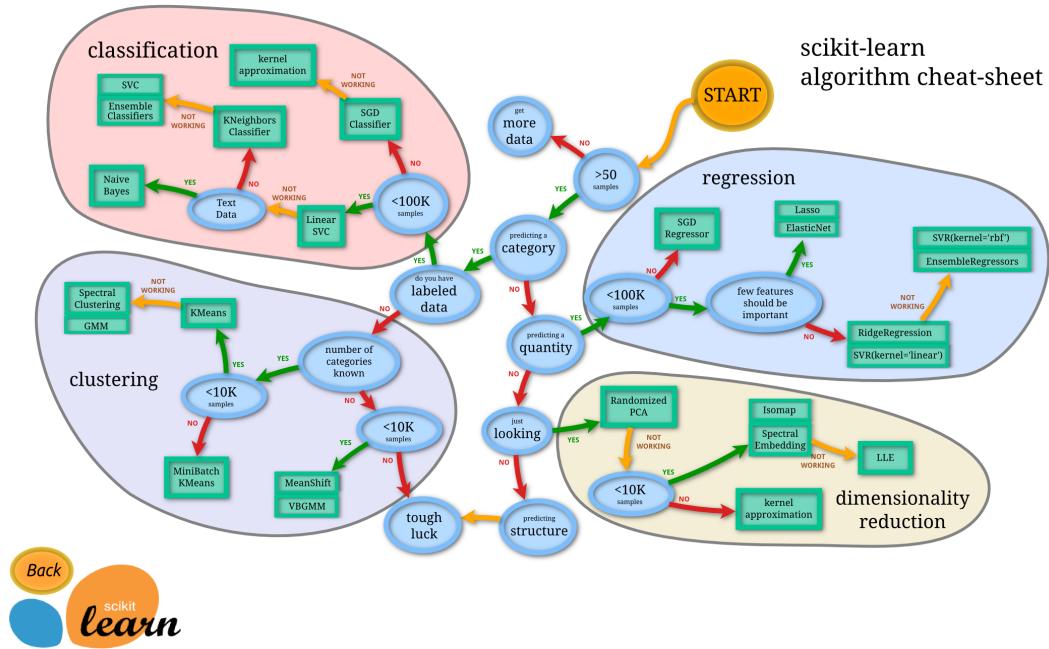
## 30.1. Relationship between Tasks

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 quantity is another way of saying continuous target. Having labels 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 portfolios, 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 derived from census data. The goal is to use a variety of features to predict if a person makes more than <math>\\$50k</math> per year or not. While income is a continuous value, they applied a threshold (<math>\\$50k</math>) to it to make a binary variable. The dataset does not include income in dollars, only the binary indicator.

### Further Reading

Recent work reconstructed 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.

## 30.2. Cross Validation

This week our goal is to learn how to optimize 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, 0.96666667, 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.

```
cross_val_score(dt,iris_X,iris_y,cv=10)

array([1.0, 0.93333333, 1.0, 0.93333333, 0.93333333,
 0.86666667, 0.93333333, 0.93333333, 1.0, 1.0])
```

### Try it yourself

What is the equivalent `train_size` for 5 fold? what about 10-fold?

```
np.mean(cross_val_score(dt,iris_X,iris_y, cv=10))
```

```
0.96
```

We can use *any* estimator object here.

```
km = KMeans(n_clusters=3)
```

```
cross_val_score(km,iris_X,)
```

```
array([-9.062, -14.93195873, -18.93234207, -23.70894258, -19.55457726])
```

### 30.3. Notes

1. Assignments 9 assesses up to level 2 for classification
2. Heads up: Assignment 10 and 11 ask you to explore your work from 2 of (7,8,9) by optimizing the parameter and comparing different models for the same task. So the dataset selection problem is going away, little by little.

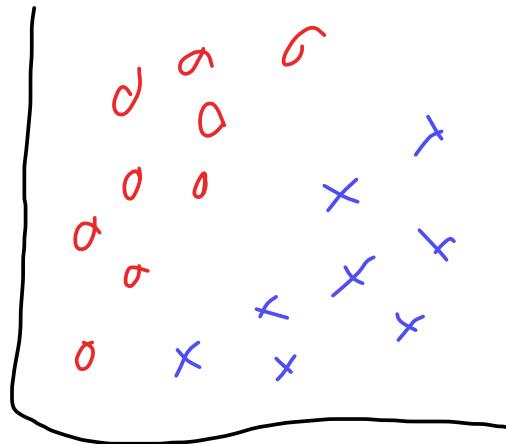
## 31. SVM and Parameter Optimizing

```
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
```

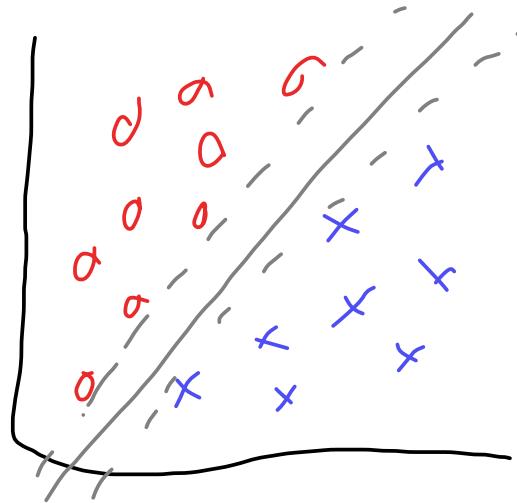
### 31.1. Support Vectors

#### 31.1.1. Basic Idea

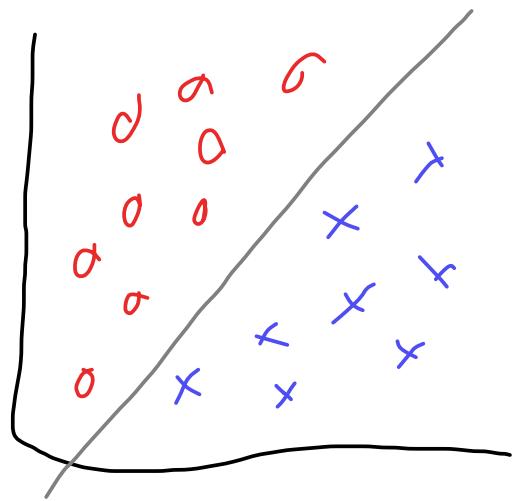
Imagine we have data that is like this



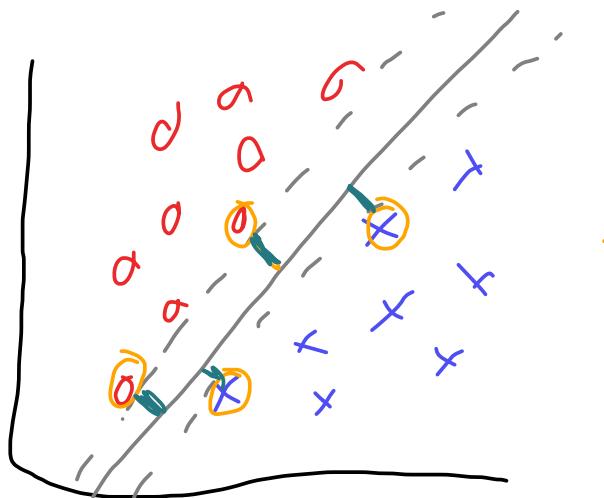
We might want to choose a decision boundary to separate it. We could choose any one of these three gray lines and get 100% training accuracy.



We could say that the best one is the solid one because it best separates the data.

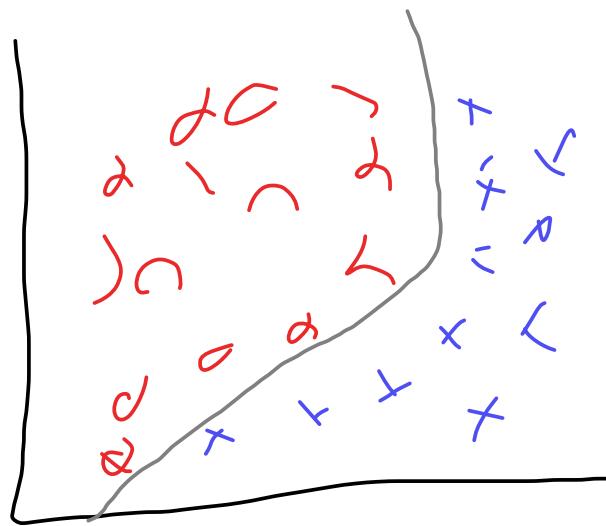


SVM does this, it finds the 'support vectors' which are the points of each class closes to the others and then finds the decision boundary that has the maximum margin, where the margin is the space between the boundary and each class.



When SVM is looking only for straight lines, it's called linear SVM, but SVM can look for different type of boundaries. We do this by changing the kernel function. A popular one is called the radial basis function or `rbf` it allows smooth curvy lines.

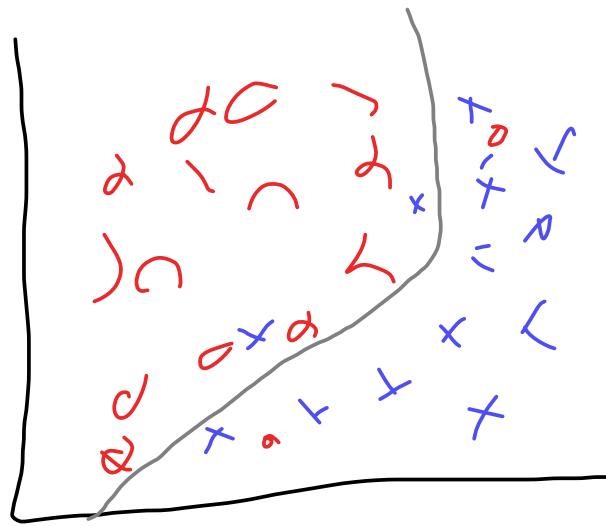
So that the SVM can work on data like this:



#### Note

Additional parameters control how smooth or wavy that line can be.

It can also handle data that is not perfectly separable like the following by minimizing the number of errors and maximizing the margin.



### 31.1.2. SVM in Sklearn

First we'll load the data and separate the features and target (`(X)` and `(y)`)

```
iris_df = sns.load_dataset('iris')
iris_X = iris_df.drop(columns='species')
iris_y = iris_df['species']
```

Next, we will split the data into test and train.

```
iris_X_train, iris_X_test, iris_y_train, iris_y_test = train_test_split(iris_X,iris_y)
```

Fitting the model is just like other models we have seen:

1. instantiate the object
2. fit the model
3. score the model on the test dat

```
svm_clf = svm.SVC()
svm_clf.fit(iris_X_train, iris_y_train)
svm_clf.score(iris_X_test, iris_y_test)
```

```
0.9473684210526315
```

We see that this fits pretty well with the default parameters.

## 31.2. Grid Search Optimization

We can optimize, however to determine the different parameter settings.

A simple way to do this is to fit the model for different parameters and score for each and compare.

We'll focus on the kernel, which controls the type of line, and  $\backslash(C\backslash)$  which controls the regularization.

```
param_grid = {'kernel': ['linear', 'rbf'], 'C': [.5, 1, 10]}
svm_opt = GridSearchCV(svm_clf, param_grid,)
```

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.

```
svm_opt.fit(iris_X_train, iris_y_train)
```

```
GridSearchCV(estimator=SVC(),
 param_grid={'C': [0.5, 1, 10], 'kernel': ['linear', 'rbf']})
```

Then we can look at the output.

```
svm_opt.cv_results_
```

```
{'mean_fit_time': array([0.0025116 , 0.00252013, 0.00229897, 0.0027132 , 0.00235915,
 0.00242515]),
 'std_fit_time': array([3.12782954e-04, 8.81972682e-05, 6.89796661e-05, 6.47515488e-04,
 6.45365228e-05, 9.05806623e-05]),
 'mean_score_time': array([0.00172973, 0.00177789, 0.00176764, 0.00174327, 0.00184431,
 0.0017303]),
 'std_score_time': array([8.90227007e-05, 2.35458114e-05, 9.09113521e-05, 1.44249348e-05,
 1.88978970e-04, 2.00785251e-05]),
 'param_C': masked_array(data=[0.5, 0.5, 1, 1, 10, 10],
 mask=[False, False, False, False, False, False],
 fill_value='?',
 dtype=object),
 'param_kernel': masked_array(data=['linear', 'rbf', 'linear', 'rbf', 'linear', 'rbf'],
 mask=[False, False, False, False, False, False],
 fill_value='?',
 dtype=object),
 'params': [{ 'C': 0.5, 'kernel': 'linear'},
 { 'C': 0.5, 'kernel': 'rbf'},
 { 'C': 1, 'kernel': 'linear'},
 { 'C': 1, 'kernel': 'rbf'},
 { 'C': 10, 'kernel': 'linear'},
 { 'C': 10, 'kernel': 'rbf'}],
 'split0_test_score': array([0.91304348, 0.86956522, 0.91304348, 0.86956522, 0.91304348,
 0.91304348]),
 'split1_test_score': array([1. , 0.86956522, 1. , 0.95652174, 1. ,
 1.]),
 'split2_test_score': array([1., 1., 1., 1., 1.]),
 'split3_test_score': array([1., 1., 1., 1., 1.]),
 'split4_test_score': array([1. , 0.95454545, 1. , 1. , 1. ,
 1.]),
 'mean_test_score': array([0.9826087 , 0.93873518, 0.9826087 , 0.96521739, 0.9826087 ,
 0.9826087]),
 'std_test_score': array([0.03478261, 0.05886542, 0.03478261, 0.05070393, 0.03478261,
 0.03478261]),
 'rank_test_score': array([1, 6, 1, 5, 1, 1], dtype=int32)}
```

We note that this is a dictionary, so to make it more readable, we can make it a DataFrame.

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	param_kerr
0	0.002512	0.000313	0.001730	0.000089	0.5	line
1	0.002520	0.000088	0.001778	0.000024	0.5	
2	0.002299	0.000069	0.001768	0.000091	1	line
3	0.002713	0.000648	0.001743	0.000014	1	
4	0.002359	0.000065	0.001844	0.000189	10	line
5	0.002425	0.000091	0.001730	0.000020	10	

It also has a `best_estimator_` attribute, which is an estimator object.

```
type(svm_opt.best_estimator_)
```

```
sklearn.svm._classes.SVC
```

This is the model that had the best cross validated score among all of the parameter settings tested.

```
svm_opt.best_estimator_.score(iris_X_test,iris_y_test)
```

```
0.9736842105263158
```

We can then use this model on the test data.

### Try it Yourself

Find the best criterion, max depth, and minimum number of samples per leaf

```
dt = tree.DecisionTreeClassifier()
params_dt = {'criterion':['gini','entropy'],'max_depth':[2,3,4],
'min_samples_leaf':list(range(2,20,2))}
```

To do this, we do just as we did above, instantiate and fit the model.

```
dt_opt = GridSearchCV(dt,params_dt)
dt_opt.fit(iris_X_train,iris_y_train)
```

```
GridSearchCV(estimator=DecisionTreeClassifier(),
param_grid={'criterion': ['gini', 'entropy'],
'max_depth': [2, 3, 4],
'min_samples_leaf': [2, 4, 6, 8, 10, 12, 14, 16, 18]})
```

Then we can use the `best_params_` attribute to see the best parameter settings.

```
dt_opt.best_params_
```

```
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 2}
```

### 31.3. Questions after class

#### 31.3.1. Can this be used on more types of machine learning than just decision trees and svm?

## 32. Model Comparison

To compare models, we will first optimize the parameters of two different models and look at how the different parameters settings impact the model comparison. Later, we'll see how to compare across models of different classes.

```
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')
```

We'll use the iris data again.

```
iris_X, iris_y = datasets.load_iris(return_X_y=True)
```

Remember, we need to split the data into training and test. The cross validation step will help us optimize the parameters, but we don't want *data leakage* where the model has seen the test data multiple times. So, we split the data here for train and test and the cross validation splits the training data into train and "test" again, but this test is better termed validation.

```
iris_X_train, iris_X_test, iris_y_train, iris_y_test = model_selection.train_test_split(
 iris_X, iris_y, test_size=.2)
```

Then we can make the object, the parameter grid dictionary and the Grid Search object. We split these into separate cells, so that we can use the built in help to see more detail.

```
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)
```

Then we fit the Grid search using the training data, and remember this actually resets the parameters and then cross validates multiple times.

```
dt_opt.fit(iris_X_train,iris_y_train)
```

```
GridSearchCV(estimator=DecisionTreeClassifier(),
 param_grid={'criterion': ['gini', 'entropy'],
 'max_depth': [2, 3, 4],
 'min_samples_leaf': [2, 4, 6, 8, 10, 12, 14, 16, 18]})
```

and look at the results

dt\_opt.cv\_results\_





```

0.875 , 0.875 , 0.875 , 0.875 , 0.875 ,
0.875 , 0.875 , 0.875 , 0.875 , 0.875 ,
0.875 , 0.875 , 0.875 , 0.875 , 0.875],
'split2_test_score': array([0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.95833333,
0.95833333, 0.95833333, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.95833333,
0.95833333, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667],
'split3_test_score': array([0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667,
0.91666667, 0.91666667, 0.91666667, 0.91666667],
'split4_test_score': array([0.95833333, 0.875 , 0.95833333, 0.95833333, 0.95833333,
0.95833333, 0.95833333, 0.95833333, 0.95833333,
0.95833333, 0.95833333, 0.875 , 0.95833333, 0.95833333,
0.875 , 0.875 , 0.95833333, 0.95833333, 0.95833333,
0.95833333, 0.95833333, 0.95833333, 0.95833333, 0.95833333,
0.95833333, 0.95833333, 0.95833333, 0.875 , 0.875 ,
0.875 , 0.95833333, 0.95833333, 0.95833333, 0.95833333,
0.95833333, 0.95833333, 0.95833333, 0.95833333, 0.875 ,
0.95833333, 0.95833333, 0.95833333, 0.95833333, 0.875 ,
0.95833333, 0.875 , 0.875 , 0.95833333, 0.875],
'mean_test_score': array([0.93333333, 0.91666667, 0.93333333, 0.93333333, 0.93333333,
0.93333333, 0.93333333, 0.93333333, 0.93333333, 0.95 ,
0.94166667, 0.94166667, 0.91666667, 0.93333333, 0.93333333,
0.91666667, 0.91666667, 0.93333333, 0.93333333, 0.94166667,
0.94166667, 0.93333333, 0.93333333, 0.93333333, 0.93333333,
0.93333333, 0.93333333, 0.925 , 0.90833333, 0.90833333,
0.90833333, 0.925 , 0.925 , 0.90833333, 0.925 ,
0.925 , 0.94166667, 0.94166667, 0.94166667, 0.93333333,
0.90833333, 0.90833333, 0.925 , 0.925 , 0.90833333,
0.93333333, 0.94166667, 0.94166667, 0.93333333, 0.90833333,
0.925 , 0.90833333, 0.90833333, 0.925]),
'std_test_score': array([0.04249183, 0.04564355, 0.04249183, 0.04249183, 0.04249183,
0.04249183, 0.04249183, 0.04249183, 0.04249183, 0.03118048,
0.04249183, 0.04249183, 0.04564355, 0.04249183, 0.04249183,
0.04564355, 0.04564355, 0.04249183, 0.04249183, 0.04249183,
0.04249183, 0.04249183, 0.04249183, 0.04249183, 0.04249183,
0.04249183, 0.04249183, 0.03118048, 0.03118048, 0.03118048,
0.03118048, 0.03118048, 0.03118048, 0.03118048, 0.03118048,
0.03118048, 0.03118048, 0.03118048, 0.03118048, 0.03118048,
0.03118048, 0.03118048, 0.03118048, 0.03118048, 0.03118048,
0.03118048, 0.03118048, 0.03118048, 0.03118048, 0.03118048]),
'rank_test_score': array([11, 41, 11, 11, 11, 11, 11, 11, 11, 11, 1, 2, 2, 41, 11, 11,
41, 41,
11, 11, 2, 2, 11, 11, 11, 11, 32, 45, 45, 45, 45, 32, 32, 45,
32, 32, 2, 2, 11, 45, 45, 32, 32, 45, 11, 2, 2, 11, 45, 32,
45, 45, 32], dtype=int32)}

```

We can reformat it into a dataframe for further analysis.

```

dt_df = pd.DataFrame(dt_opt.cv_results_)
dt_df.head(2)

```

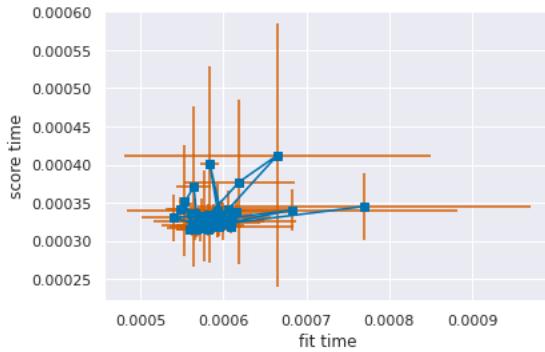
	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_criterion	para
0	0.000770	0.000200	0.000345	0.000044		gini
1	0.000584	0.000041	0.000320	0.000005		gini

## Correction

The parameters in this function were in the wrong order in this function in class

I changed the markers and the color of the error bars for readability.

```
plt.errorbar(x=dt_df['mean_fit_time'],y=dt_df['mean_score_time'],
 xerr=dt_df['std_fit_time'],yerr=dt_df['std_score_time'],
 marker='s',ecolor='r')
plt.xlabel('fit time')
plt.ylabel('score time')
save the limits so we can reuse them
xmin, xmax, ymin, ymax = plt.axis()
```

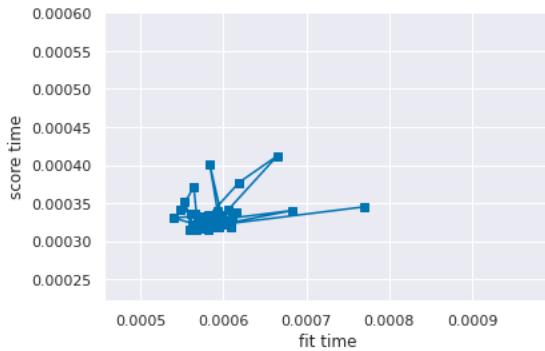


The “points” are at the mean fit and score times. The lines are the “standard deviation” or how much we expect that number to vary, since means are an estimate. Because the data shows an upward trend, this plot tells us that mostly, the models that are slower to fit are also slower to apply. This makes sense for decision trees, deeper trees take longer to learn and longer to traverse when predicting. Because the error bars mostly overlap the other points, this tells us that mostly the variation in time is not a reliable difference. If we re-ran the GridSearch, we could get them in different orders.

To interpret the error bar plot, let's look at a line plot of just the means, with the same limits so that it's easier to compare to the plot above.

```
plt.plot(dt_df['mean_fit_time'],
 dt_df['mean_score_time'], marker='s')
plt.xlabel('fit time')
plt.ylabel('score time')
match the axis limits to above
plt.ylim(ymin, ymax)
plt.xlim(xmin,xmax)
```

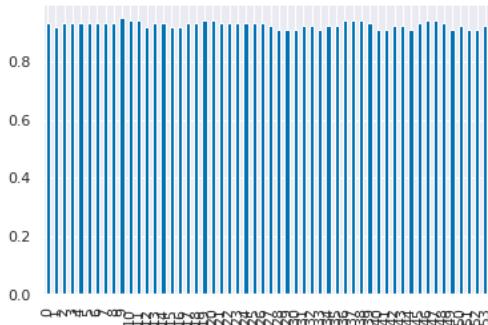
(0.0004567914792290151, 0.000994380938439441)



this plot shows the mean times, without the error bars.

```
dt_df['mean_test_score'].plot(kind='bar')
```

```
<AxesSubplot:>
```



```
dt_df['mean_test_score']
```

```
0 0.933333
1 0.916667
2 0.933333
3 0.933333
4 0.933333
5 0.933333
6 0.933333
7 0.933333
8 0.933333
9 0.950000
10 0.941667
11 0.941667
12 0.916667
13 0.933333
14 0.933333
15 0.916667
16 0.916667
17 0.933333
18 0.933333
19 0.941667
20 0.941667
21 0.933333
22 0.933333
23 0.933333
24 0.933333
25 0.933333
26 0.933333
27 0.925000
28 0.908333
29 0.908333
30 0.908333
31 0.925000
32 0.925000
33 0.908333
34 0.925000
35 0.925000
36 0.941667
37 0.941667
38 0.941667
39 0.933333
40 0.908333
41 0.908333
42 0.925000
43 0.925000
44 0.908333
45 0.933333
46 0.941667
47 0.941667
48 0.933333
49 0.908333
50 0.925000
51 0.908333
52 0.908333
53 0.925000
Name: mean_test_score, dtype: float64
```

Now let's compare with a different model, we'll use the parameter optimized version for that model.

```
svm_clf = svm.SVC()
param_grid = {'kernel':['linear','rbf'], 'C':[.5, 1, 10]}
svm_opt = GridSearchCV(svm_clf,param_grid,)
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2470/492760704.py in <module>
 1 svm_clf = svm.SVC()
 2 param_grid = {'kernel':['linear','rbf'], 'C':[.5, 1, 10]}
----> 3 svm_opt = GridSearchCV(svm_clf,param_grid,)

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

The error above is because we didn't import `GridSearchCV` directly today, we imported the whole `model_selection` module, so we have to use that in order to access the class.

```
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,cv=10)
```

```
type(model_selection)
```

```
module
```

```
dt_opt.__dict__
```

```

{'scoring': None,
'estimator': DecisionTreeClassifier(),
'n_jobs': None,
'refit': True,
'cv': None,
'verbose': 0,
'pre_dispatch': '2*n_jobs',
'error_score': nan,
'return_train_score': False,
'param_grid': {'criterion': ['gini', 'entropy'],
'max_depth': [2, 3, 4],
'min_samples_leaf': [2, 4, 6, 8, 10, 12, 14, 16, 18]},
'multimetric_': False,
'best_index_': 9,
'best_score_': 0.95,
'best_params_': {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 2},
'best_estimator_': DecisionTreeClassifier(max_depth=3, min_samples_leaf=2),
'refit_time_': 0.00041365623474121094,
'scorer_': <function sklearn.metrics._scorer._passthrough_scorer(estimator, *args,
**kwargs)>,
'cv_results_': {'mean_fit_time': array([0.00076952, 0.00058403, 0.00055919, 0.00056267,
0.00055308,
0.00056534, 0.00056882, 0.00056458, 0.00054846, 0.0005825 ,
0.00059304, 0.00056095, 0.0005662 , 0.00059791, 0.00053997,
0.0005723 , 0.00057631, 0.00057125, 0.00058732, 0.00061116,
0.00058732, 0.00058374, 0.00059891, 0.00058069, 0.00068378,
0.00057058, 0.00056515, 0.00058784, 0.00057545, 0.00057435,
0.0005825 , 0.0005734 , 0.00056725, 0.00061908, 0.00066533,
0.00060592, 0.00058761, 0.00060296, 0.00059533, 0.0005827 ,
0.00060987, 0.00058422, 0.00057592, 0.00060148, 0.0005619 ,
0.00059862, 0.00057874, 0.00057321, 0.00057049, 0.00061488,
0.00061121, 0.00059958, 0.00058393, 0.00059314]),
'std_fit_time': array([2.00425248e-04, 4.06030690e-05, 1.53461578e-05, 1.41098685e-05,
6.09978920e-06, 3.26643547e-05, 1.40026217e-05, 2.05720836e-05,
1.79727762e-05, 1.86493933e-05, 5.62493249e-05, 8.38393094e-06,
1.63544718e-05, 3.70626723e-05, 3.79786353e-05, 4.69955860e-05,
2.33593759e-05, 2.37189942e-05, 1.68416022e-05, 4.57732875e-05,
1.99256322e-05, 1.89296544e-05, 4.98378099e-05, 1.51350548e-05,
1.99026255e-04, 1.13126036e-05, 1.61791909e-05, 2.60815542e-05,
1.54296429e-05, 3.01208639e-05, 2.20582416e-05, 1.47688715e-05,
2.37665896e-05, 6.68291317e-05, 1.84103523e-04, 7.25254426e-05,
1.39865369e-05, 4.08020504e-05, 2.68688324e-05, 9.99881018e-06,
7.40560499e-05, 1.61812988e-05, 1.91606675e-05, 8.56882344e-05,
7.43376742e-06, 5.38509411e-06, 9.95505350e-06, 5.18504227e-06,
3.74977174e-06, 6.79570746e-05, 2.85583662e-05, 2.40877820e-05,
1.16189864e-05, 3.52702879e-05]),
'mean_score_time': array([0.00034513, 0.00032001, 0.00031433, 0.00031595, 0.00035257,
0.00031633, 0.00032482, 0.00037112, 0.00034099, 0.00033436,
0.00032525, 0.0003366 , 0.00033584, 0.00033517, 0.00033073,
0.00032034, 0.00033278, 0.00032811, 0.00033503, 0.00033107,
0.00031881, 0.00032682, 0.00032568, 0.0003264 , 0.00034046,
0.00031633, 0.0003211 , 0.00031886, 0.00031743, 0.00031791,
0.00031576, 0.0003294 , 0.00031486, 0.00037718, 0.00041227,
0.00034118, 0.00032902, 0.00032415, 0.00031948, 0.00032005,
0.00031948, 0.00031896, 0.00032449, 0.00032558, 0.00032234,
0.00032201, 0.00032377, 0.00033326, 0.0003264 , 0.00033822,
0.0003315 , 0.0003263 , 0.0004004 , 0.00033975]),}

```

```

'std_score_time': array([4.37388874e-05, 5.05852087e-06, 6.08523432e-06, 9.81335042e-
06,
 7.21479045e-05, 1.56395319e-05, 1.83254420e-05, 1.04452222e-04,
 1.63537767e-05, 1.43484413e-05, 2.07220723e-05, 1.35537555e-05,
 1.89002033e-05, 1.93371431e-05, 3.05396983e-05, 2.15879301e-05,
 5.98301247e-05, 1.43300475e-05, 8.54244154e-06, 1.80333988e-05,
 3.02105600e-06, 9.81914117e-06, 2.39896959e-05, 1.39860492e-05,
 2.74017534e-05, 1.36878015e-06, 5.22653550e-06, 8.67607797e-06,
 6.91923414e-06, 7.79829233e-06, 9.30577982e-06, 8.32432588e-06,
 1.01406123e-04, 1.07947234e-04, 1.73126590e-04, 2.53235999e-05,
 1.92567839e-05, 7.47829053e-06, 1.27779903e-05, 4.55523014e-06,
 9.25481840e-06, 9.26635823e-06, 1.12588110e-05, 1.24568631e-05,
 8.83500893e-06, 4.65887229e-06, 1.13703418e-05, 2.28714882e-05,
 1.29862641e-05, 1.63114555e-05, 1.23946479e-05, 1.10946389e-05,
 1.28451523e-04, 3.51332246e-05]),

'param_criterion': masked_array(data=['gini', 'gini', 'gini', 'gini', 'gini', 'gini',
'gini',
 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
 'gini', 'gini', 'gini', 'gini', 'gini', 'gini',
 'entropy', 'entropy', 'entropy', 'entropy', 'entropy',
 'entropy', 'entropy', 'entropy', 'entropy', 'entropy'],
mask=[False, False, False, False, False, False, False, False,
 False, False, False, False, False, False, False, False],
fill_value='?',
dtype=object),
'param_max_depth': masked_array(data=[2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3,
3, 3,
 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4],
mask=[False, False, False, False, False, False, False, False, False,
 False, False, False, False, False, False, False, False],
fill_value='?',
dtype=object),
'param_min_samples_leaf': masked_array(data=[2, 4, 6, 8, 10, 12, 14, 16, 18, 2, 4, 6,
8, 10, 12, 14,
 16, 18, 2, 4, 6, 8, 10, 12, 14, 16, 18, 2, 4, 6,
 12, 14, 16, 18, 2, 4, 6, 8, 10, 12, 14, 16, 18, 2, 4,
 6, 8, 10, 12, 14, 16, 18],
mask=[False, False, False, False, False, False, False, False, False,
 False, False, False, False, False, False, False, False],
fill_value='?',
dtype=object),
'params': [{criterion: 'gini', 'max_depth': 2, 'min_samples_leaf': 2},
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 4},
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 6},
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 8},
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 10},
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 12},
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 14},
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 16},
{'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 18},
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 2},
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 4},
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 6},
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 8},
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 10},
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 12},
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 14},
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 16},
{'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 18},
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2},
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 4},
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 6},
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 8},
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 10},
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 12},
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 14}]}

```



```

0.916666667, 0.916666667, 0.933333333, 0.933333333, 0.941666667,
0.941666667, 0.933333333, 0.933333333, 0.933333333, 0.933333333,
0.933333333, 0.933333333, 0.925 , 0.908333333, 0.908333333,
0.908333333, 0.925 , 0.908333333, 0.925 ,
0.925 , 0.941666667, 0.941666667, 0.941666667, 0.933333333,
0.908333333, 0.908333333, 0.925 , 0.925 , 0.908333333,
0.933333333, 0.941666667, 0.941666667, 0.933333333, 0.908333333,
0.925 , 0.908333333, 0.908333333, 0.925],
'std_test_score': array([0.04249183, 0.04564355, 0.04249183, 0.04249183, 0.04249183,
0.04249183, 0.04249183, 0.04249183, 0.03118048,
0.04249183, 0.04249183, 0.04564355, 0.04249183, 0.04249183,
0.04564355, 0.04564355, 0.04249183, 0.04249183, 0.04249183,
0.04249183, 0.04249183, 0.04249183, 0.04249183, 0.04249183,
0.04249183, 0.04249183, 0.03118048, 0.03118048, 0.03118048,
0.03118048, 0.03118048, 0.03118048, 0.03118048, 0.03118048,
0.03118048, 0.03118048, 0.03118048, 0.03118048, 0.03118048,
0.03118048, 0.03118048, 0.03118048, 0.03118048, 0.03118048],
'rank_test_score': array([11, 41, 11, 11, 11, 11, 11, 11, 11, 11, 1, 2, 2, 2, 41, 11, 11,
41, 41,
11, 11, 2, 2, 11, 11, 11, 11, 11, 32, 45, 45, 45, 45, 32, 32, 45,
32, 32, 2, 2, 2, 11, 45, 45, 32, 32, 45, 11, 2, 2, 11, 45, 32,
45, 45, 32], dtype=int32)},
'n_splits': 5}

```

This doesn't have attributes yet, even though they are the same type, because we have not fit it to data yet.

```
type(svm_opt), type(dt_opt)
```

```
(sklearn.model_selection._search.GridSearchCV,
 sklearn.model_selection._search.GridSearchCV)
```

Now we can fit the model to the training data of this second model.

```
fit the model and put the CV results in a dataframe
svm_opt.fit(iris_X_train,iris_y_train)
sv_df = pd.DataFrame(svm_opt.cv_results_)
```

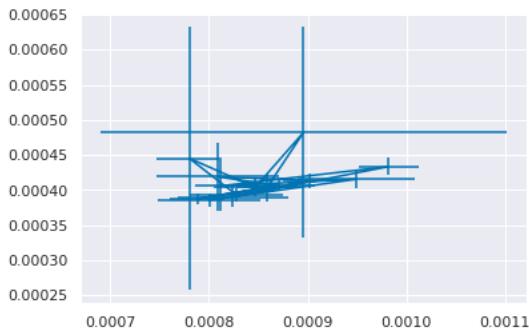
```
sv_df.head(2)
```

**mean fit time std fit time mean score time std score time param C param kerr**

<b>0</b>	0.000824	0.000057	0.000390	0.000014	0.5	line
<b>1</b>	0.000982	0.000030	0.000433	0.000012	0.5	

```
plt.errorbar(x=sv_df['mean_fit_time'],xerr=sv_df['std_fit_time'],
 y=sv_df['mean_score_time'],yerr=sv_df['std_score_time'])
```

```
<ErrorbarContainer object of 3 artists>
```



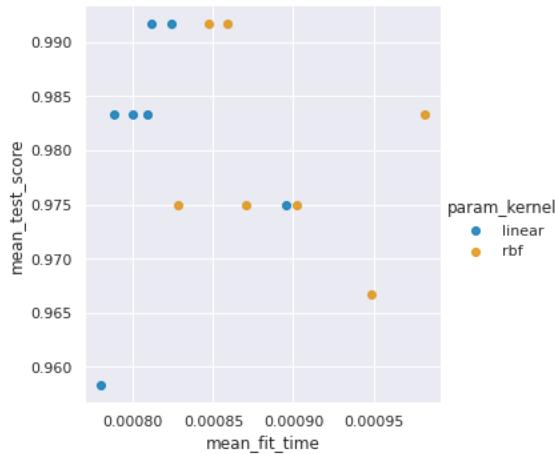
sv\_df.columns

```
Index(['mean_fit_time', 'std_fit_time', 'mean_score_time', 'std_score_time',
 'param_C', 'param_kernel', 'params', 'split0_test_score',
 'split1_test_score', 'split2_test_score', 'split3_test_score',
 'split4_test_score', 'split5_test_score', 'split6_test_score',
 'split7_test_score', 'split8_test_score', 'split9_test_score',
 'mean_test_score', 'std_test_score', 'rank_test_score'],
 dtype='object')
```

We can see if the models that take longer to fit or score perform better.

```
svm_time = sv_df.melt(id_vars=['param_C', 'param_kernel', 'params'],
 value_vars=['mean_fit_time', 'std_fit_time', 'mean_score_time',
 'std_score_time'])
sns.lmplot(data=sv_df, x='mean_fit_time', y='mean_test_score',
 hue='param_kernel', fit_reg=False)
```

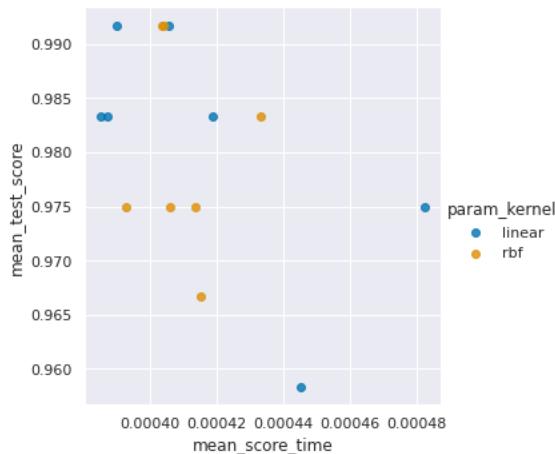
<seaborn.axisgrid.FacetGrid at 0x7f12f12fa990>



This looks like mostly no.

```
sns.lmplot(data=sv_df, x='mean_score_time', y='mean_test_score',
 hue='param_kernel', fit_reg=False)
```

<seaborn.axisgrid.FacetGrid at 0x7f12f1769210>



Again, for score time, the slower models don't appear to be better. Remember though the time differences weren't that different.

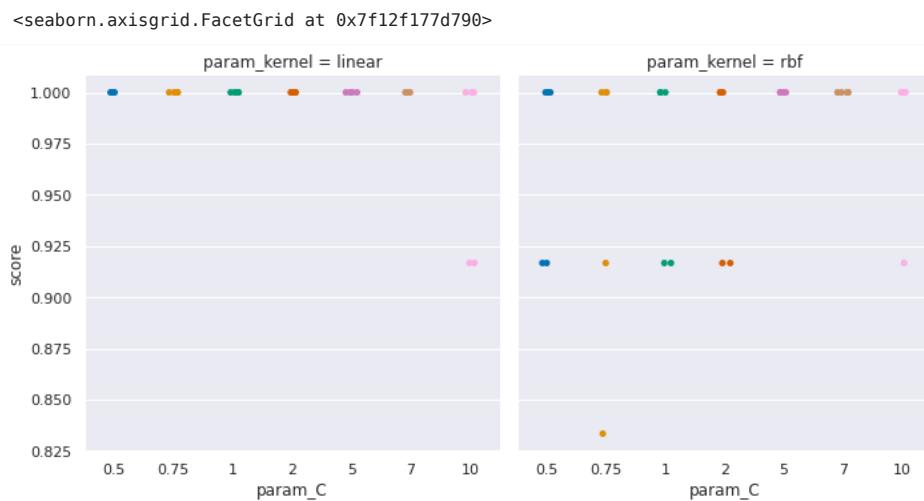
### Try it yourself

Try this same analysis for the decision tree, does it matter there?

```
sv_df_scores = sv_df.melt(id_vars=['param_C', 'param_kernel', 'params'],
 value_vars=['split0_test_score',
 'split1_test_score', 'split2_test_score', 'split3_test_score',
 'split4_test_score'], value_name='score')
sv_df_scores.head()
```

	param_C	param_kernel	params	variable	score
0	0.5	linear	{'C': 0.5, 'kernel': 'linear'}	split0_test_score	1.0
1	0.5	rbf	{'C': 0.5, 'kernel': 'rbf'}	split0_test_score	1.0
2	0.75	linear	{'C': 0.75, 'kernel': 'linear'}	split0_test_score	1.0
3	0.75	rbf	{'C': 0.75, 'kernel': 'rbf'}	split0_test_score	1.0
4	1	linear	{'C': 1, 'kernel': 'linear'}	split0_test_score	1.0

```
sns.catplot(data=sv_df_scores,x='param_C',y='score',
 col='param_kernel')
```

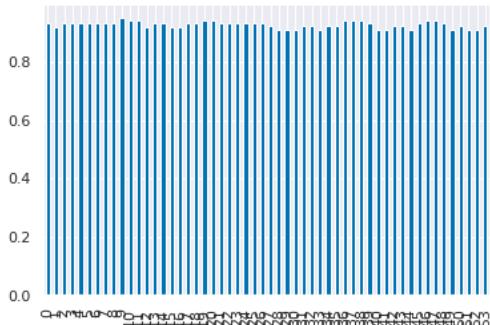


### Try it yourself

Try interpreting the plot above, what does it say? what can you conclude from it.

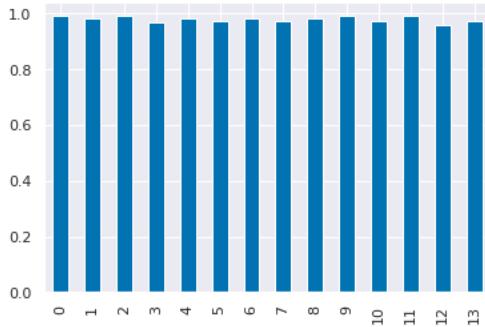
```
dt_df['mean_test_score'].plot(kind='bar')
```

```
<AxesSubplot:>
```



```
sv_df['mean_test_score'].plot(kind='bar')
```

```
<AxesSubplot:>
```



From these last two plots we see that the SVM performance is more sensitive to its parameters, where for the parameters tested, the decision tree is not impacted.

What can we say based on this? We'll pick up from here on Wednesday.

## 33. Model Selection

We'll pick up from where we left off on Monday.

### ⓘ Further Reading

If you struggled to understand this code excerpt to fill in the comments, some generic strategies to understand code may help, beyond applying what we have covered in class.

The Programmer's brain is an overview of how brains work, as applied to programming, written for working developers. This means that it assumes you know most CS concepts and at least two programming languages. If you don't there may be some parts that do not make sense to you, but the general ideas should still make sense. The author is a professor who researchers how people learn programming and how to effectively teach it.

```

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 and split the data
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)

create dt, set param grid & create optimizer
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, cv=10)

fit the model and optimize
dt_opt.fit(iris_X_train,iris_y_train)

store the results in a dataframe
dt_df = pd.DataFrame(dt_opt.cv_results_)

create svm, its parameter grid and optimizer
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, cv=10)

fit the model and put the CV results in a dataframe
svm_opt.fit(iris_X_train,iris_y_train)
sv_df = pd.DataFrame(svm_opt.cv_results_)

```

We can compare how the best models fit during validation:

```
svm_opt.best_score_, dt_opt.best_score_
```

```
(0.975, 0.933333333333332)
```

We see that the SVM does a little bit better.

We can also apply the best estimator from each model class (that is the best parameter settings for each SVM and Decision Tree) to the test data to get our final score.

```
svm_opt.best_estimator_.score(iris_X_test,iris_y_test)
```

```
0.9666666666666667
```

```
dt_opt.best_estimator_.score(iris_X_test,iris_y_test)
```

```
0.9666666666666667
```

We can also examine the results to think through this choice more clearly.

```
sv_df.head(2)
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	param_kerr
0	0.000813	0.000074	0.000383	0.000009	0.5	line
1	0.000929	0.000035	0.000414	0.000009	0.5	

We can use EDA to understand how the score varied across all of the parameter settings we tried.

```
sv_df['mean_test_score'].describe()
```

```
count 14.000000
mean 0.961310
std 0.007016
min 0.950000
25% 0.958333
50% 0.958333
75% 0.964583
max 0.975000
Name: mean_test_score, dtype: float64
```

```
dt_df['mean_test_score'].describe()
```

```
count 90.000000
mean 0.926019
std 0.002745
min 0.925000
25% 0.925000
50% 0.925000
75% 0.925000
max 0.933333
Name: mean_test_score, dtype: float64
```

From this we see that in both cases the standard deviation (std) is really low. This tells us that the parameter changes didn't impact the performance much. Combined with the overall high accuracy this tells us that the data is probably really easy to classify. If the performance had been uniformly bad, it might have instead told us that we did not try a wide enough range of parameters.

To confirm how many parameter settings we have used we can check a couple different ways. First, above in the count of the describe.

We can also calculate directly from the parameter grids before we even do the fit.

```
n_combinations = 1
for param, vals in param_grid.items():
 n_combinations *= len(vals)
```

### 33.1. When do differences matter?

We can calculate a confidence interval to determine more precisely when the performance of two models is meaningfully different.

This function calculates the 95% confidence interval. The range within which we are 95% confident the quantity we have estimated is truly within in. When we have more samples in the test set used to calculate the score, we are more confident in the estimate, so the interval is narrower.

```
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)
```

```
svm_opt.best_score_, dt_opt.best_score_
```

```
(0.975, 0.9333333333333332)
```

We can calculate the number of observations used to compute the accuracy using the size of the training data and the fact that we set it to 10-fold cross validation. That means that 10% (100/10) of the data was used for each fold and each validation set.

```
len(iris_X_train)*.1
```

```
12.0
```

```
split0_score = dt_df['split0_test_score'][dt_opt.best_index_]
classification_confint(split0_score ,len(iris_X_train)*.1)
```

```
(1.0, 1.0)
```

### ⓘ Correction

since the best score is cross validated it actually uses the whole training data (by averaging 10 scores together).

```
classification_confint(dt_opt.best_score_,len(iris_X_train))
```

```
(0.8887021699971536, 0.9779644966695129)
```

```
len(iris_y_test)
```

```
30
```

```
classification_confint(dt_opt.best_estimator_.score(iris_X_test,iris_y_test),len(iris_y_test))
```

```
(0.9024314507569886, 1.0)
```

If we take the exact value we can see how more samples narrows the interval.

```
classification_confint(.95,12)
```

```
(0.8266860375572443, 1.0)
```

```
classification_confint(.95,30)
```

```
(0.8720094022760863, 1.0)
```

We can further explore this by plotting directly from the calculation out of the function

```
n_list = list(range(5,200))
intervals = [1.96*np.sqrt(acc*(1-acc)/n) for n in n_list]
plt.plot(n_list, intervals)
```

```
File "/tmp/ipykernel_2492/2188489612.py", line 1
 n_list = list(range(5,200))
 ^
SyntaxError: invalid syntax
```

## 33.2. Questions After Class

### 33.2.1. When would I choose to use an svm?

### 33.2.2. How can we tell how many parameter settings we tried?

## 34. Learning Curves

```
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 naive_bayes
from sklearn import svm
from sklearn import tree
import the whole model selection module
from sklearn import model_selection
sns.set_theme(palette='colorblind')
```

Today, we'll load a new dataset and use the default sklearn data structure for datasets. We get back the default data structure when we use a `load_` function without any parameters at all.

```
digits = datasets.load_digits()
```

This shows us that the type is defined by sklearn and they called it `bunch`:

```
type(digits)
```

```
sklearn.utils.Bunch
```

We can print it out to begin exploring it.

```
digits
```

```
{'data': array([[0., 0., 5., ..., 0., 0., 0.],
 [0., 0., 0., ..., 10., 0., 0.],
 [0., 0., 0., ..., 16., 9., 0.],
 ...,
 [0., 0., 1., ..., 6., 0., 0.],
 [0., 0., 2., ..., 12., 0., 0.],
 [0., 0., 10., ..., 12., 1., 0.]]),
'target': array([0, 1, 2, ..., 8, 9, 8]),
'frame': None,
'feature_names': ['pixel_0_0',
 'pixel_0_1',
 'pixel_0_2',
 'pixel_0_3',
 'pixel_0_4',
 'pixel_0_5',
 'pixel_0_6',
```

```

'pixel_0_7',
'pixel_1_0',
'pixel_1_1',
'pixel_1_2',
'pixel_1_3',
'pixel_1_4',
'pixel_1_5',
'pixel_1_6',
'pixel_1_7',
'pixel_2_0',
'pixel_2_1',
'pixel_2_2',
'pixel_2_3',
'pixel_2_4',
'pixel_2_5',
'pixel_2_6',
'pixel_2_7',
'pixel_3_0',
'pixel_3_1',
'pixel_3_2',
'pixel_3_3',
'pixel_3_4',
'pixel_3_5',
'pixel_3_6',
'pixel_3_7',
'pixel_4_0',
'pixel_4_1',
'pixel_4_2',
'pixel_4_3',
'pixel_4_4',
'pixel_4_5',
'pixel_4_6',
'pixel_4_7',
'pixel_5_0',
'pixel_5_1',
'pixel_5_2',
'pixel_5_3',
'pixel_5_4',
'pixel_5_5',
'pixel_5_6',
'pixel_5_7',
'pixel_6_0',
'pixel_6_1',
'pixel_6_2',
'pixel_6_3',
'pixel_6_4',
'pixel_6_5',
'pixel_6_6',
'pixel_6_7',
'pixel_7_0',
'pixel_7_1',
'pixel_7_2',
'pixel_7_3',
'pixel_7_4',
'pixel_7_5',
'pixel_7_6',
'pixel_7_7'],
'target_names': array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
'images': array([[[0., 0., 5., ..., 1., 0., 0.],
 [0., 0., 13., ..., 15., 5., 0.],
 [0., 3., 15., ..., 11., 8., 0.],
 ...,
 [0., 4., 11., ..., 12., 7., 0.],
 [0., 2., 14., ..., 12., 0., 0.],
 [0., 0., 6., ..., 0., 0., 0.]],

[[0., 0., 0., ..., 5., 0., 0.],
 [0., 0., 0., ..., 9., 0., 0.],
 [0., 0., 3., ..., 6., 0., 0.],
 ...,
 [0., 0., 1., ..., 6., 0., 0.],
 [0., 0., 1., ..., 6., 0., 0.],
 [0., 0., 0., ..., 10., 0., 0.]],

[[0., 0., 0., ..., 12., 0., 0.],
 [0., 0., 3., ..., 14., 0., 0.],
 [0., 0., 8., ..., 16., 0., 0.],
 ...,
 [0., 9., 16., ..., 0., 0., 0.],
 [0., 3., 13., ..., 11., 5., 0.],
 [0., 0., 0., ..., 16., 9., 0.]],

...,
[[0., 0., 1., ..., 1., 0., 0.],
 [0., 0., 13., ..., 2., 1., 0.],
 [0., 0., 16., ..., 16., 5., 0.],
```

```

[...,
 [0., 0., 16., ..., 15., 0., 0.],
 [0., 0., 15., ..., 16., 0., 0.],
 [0., 0., 2., ..., 6., 0., 0.]],

[[0., 0., 2., ..., 0., 0., 0.],
 [0., 0., 14., ..., 15., 1., 0.],
 [0., 4., 16., ..., 16., 7., 0.],
 ...,
 [0., 0., 0., ..., 16., 2., 0.],
 [0., 0., 4., ..., 16., 2., 0.],
 [0., 0., 5., ..., 12., 0., 0.]],

[[0., 0., 10., ..., 1., 0., 0.],
 [0., 2., 16., ..., 1., 0., 0.],
 [0., 0., 15., ..., 15., 0., 0.],
 ...,
 [0., 4., 16., ..., 16., 6., 0.],
 [0., 8., 16., ..., 16., 8., 0.],
 [0., 1., 8., ..., 12., 1., 0.]]],
```

'DESCR': "... \_digits\_dataset:\n\nOptical recognition of handwritten digits dataset\n---\n-----\n\*\*Data Set Characteristics:\*\*\n:Number of Instances: 1797\n :Number of Attributes: 64\n :Attribute Information:\n8x8 image of integer pixels in the range 0..16.\n :Missing Attribute Values: None\n:Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)\n :Date: July; 1998\n\nThis is a copy\nof the test set of the UCI ML hand-written digits\ndatasets\nhttps://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits\n\nThe data set contains images of hand-written digits: 10 classes where each class\nrefers to a digit.\n\nPreprocessing programs made available by NIST were used to\nextract\nnormalized bitmaps of handwritten digits from a preprinted form. From a\nof 43 people, 30 contributed to the training set and different 13\nthe test set. 32x32\nbitmaps are divided into nonoverlapping blocks of\n4x4 and the number of on pixels are\ncounted in each block. This generates\nan input matrix of 8x8 where each element is an\ninteger in the range\n0..16. This reduces dimensionality and gives invariance to\nsmall\ndistortions.\nFor info on NIST preprocessing routines, see M. D. Garris, J. L.\nBlue, G.\nT. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.\nWilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,\n1994.\n.. topic::\nReferences\n - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their\n Applications to Handwritten Digit Recognition, MSc Thesis, Institute of\n Graduate\nStudies in Science and Engineering, Bogazici University.\n - E. Alpaydin, C. Kaynak\n(1998) Cascading Classifiers, Kybernetika.\n - Ken Tang and Ponnuthurai N. Suganthan and\nXi Yao and A. Kai Qin.\n - Linear dimensionality reduction using relevance weighted LDA.\nSchool of\n Electrical and Electronic Engineering Nanyang Technological University.\n2005.\n - Claudio Gentile. A New Approximate Maximal Margin Classification\nAlgorithm. NIPS. 2000.\n"}

We note that it has key value pairs, and that the last one is called `DESCR` and is text that describes the data. If we send that to the print function it will be formatted more readably.

```
print(digits['DESCR'])
```

```
.. _digits_dataset:

Optical recognition of handwritten digits dataset

Data Set Characteristics:

:Number of Instances: 1797
:Number of Attributes: 64
:Attribute Information: 8x8 image of integer pixels in the range 0..16.
:Missing Attribute Values: None
:Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
:Date: July; 1998
```

This is a copy of the test set of the UCI ML hand-written digits datasets  
<https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits>

The data set contains images of hand-written digits: 10 classes where each class refers to a digit.

Preprocessing programs made available by NIST were used to extract normalized bitmaps of handwritten digits from a preprinted form. From a total of 43 people, 30 contributed to the training set and different 13 to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0..16. This reduces dimensionality and gives invariance to small distortions.

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G. T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C. L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469, 1994.

.. topic:: References

- C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their Applications to Handwritten Digit Recognition, MSc Thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University.
- E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
- Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin. Linear dimensionality reduction using relevance weighted LDA. School of Electrical and Electronic Engineering Nanyang Technological University. 2005.
- Claudio Gentile. A New Approximate Maximal Margin Classification Algorithm. NIPS. 2000.

This tells us that we are going to be predicting what digit (0,1,2,3,4,5,6,7,8, or 9) is in the image.

To get an idea of what the images look like, we can use `matshow` which is short for `matrix show`. It takes a 2D matrix and plots it as a grayscale image. To get the actual color bar, we use the `matplotlib.pyplot.gray()`.

```
plt.gray()
plt.matshow(digits.images[0])
```

### ➊ Try it yourself

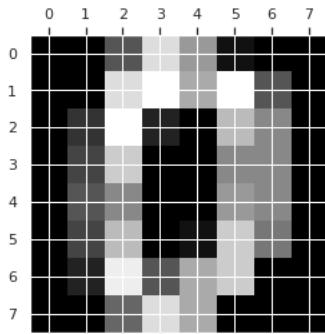
Try using `matshow` without `plt.gray()`. How is it different? What might alternatives do? What other code in this notebook influences how plots look?

### 💡 Tip

Removing a line from an excerpt of code can help you see what that line did and learn more about how each piece work.

```
<matplotlib.image.AxesImage at 0x7fd6a6f465d0>
```

```
<Figure size 432x288 with 0 Axes>
```



`bunch` objects are designed for machine learning, so they have the features as "data" and target explicitly identified.

```
digits_X = digits.data
digits_y = digits.target
```

We can further check the type and shape of these

```
type(digits_y)
```

```
numpy.ndarray
```

### Note

because this is the target, it's okay that this is one dimensional.

```
digits_y.shape
```

```
(1797,)
```

```
digits_X.shape
```

```
(1797, 64)
```

This has one row for each sample and has reshaped the 8x8 image into a 64 length vector. So we have one 'feature' for each pixel in the images.

We are going to do some model comparison, so we will instantiate estimator objects for two different classifiers.

```
svm_clf = svm.SVC(gamma=0.001)
gnb_clf = naive_bayes.GaussianNB()
```

We're going to use a [ShuffleSplit](#) object to do Cross validation with 100 iterations to get smoother mean test and train score curves, each time with 20% data randomly selected as a validation set.

### Further Reading

You can see visualization of different [cross validation](#) types in the sklearn documentation.

```
cv = model_selection.ShuffleSplit(n_splits=100, test_size=0.2, random_state=0)
```

### Note

This object has a `random_state` object, the `GridSearchCV` that we were using didn't have a way to control the random state directly, but it accepts not only integers, but also cross validation objects to the `cv` parameter. The `KFold` cross validation object also has that parameter, so we could repeat what we did in previous classes by creating a `KFold` object with a fixed random state.

We'll also create a linearly spaced list of training percentages and we'll also divide it into more jobs to be more computationally efficient.

```
train_sizes=np.linspace(0.1, 1.0, 5)
n_jobs=4
```

### Try it yourself

Try varying the `n_jobs` parameter and timing the execution using the [time magic](#)

Now we can create the learning curve.

```
train_sizes_svm, train_scores_svm, test_scores_svm, fit_times_svm, score_times_svm =
model_selection.learning_curve(
 svm_clf,
 digits_X,
 digits_y,
 cv=cv,
 n_jobs=n_jobs,
 train_sizes=train_sizes,
 return_times=True,)
```

It returns the list of the counts for each training size (we input percentages and it returns counts)

```
train_sizes_svm
```

```
array([143, 467, 790, 1113, 1437])
```

The other parameters, it returns a list for each length that's 100 long because our cross validation was 100 iterations.

```
fit_times_svm.shape
```

```
(5, 100)
```

We can save it in a DataFrame after averaging over the 100 trials.

```
svm_learning_df = pd.DataFrame(data = train_sizes_svm, columns = ['train_size'])
svm_learning_df['train_size'] = train_sizes_svm
svm_learning_df['train_score'] = np.mean(train_scores_svm, axis=1)
svm_learning_df['test_score'] = np.mean(test_scores_svm, axis=1)
svm_learning_df['fit_time'] = np.mean(fit_times_svm, axis=1)
svm_learning_df['score_time'] = np.mean(score_times_svm, axis=1)
```

then we can look at the DataFrame

```
svm_learning_df.head()
```

	train_size	train_score	test_score	fit_time	score_times
0	143	0.999510	0.923611	0.005824	0.010911
1	467	0.999101	0.978278	0.034431	0.023905
2	790	0.998848	0.986278	0.067925	0.034513
3	1113	0.998913	0.989500	0.107535	0.041753
4	1437	0.998859	0.991306	0.164037	0.053989

We can use our skills in transforming data to make it easier to examine just a subset of the scores.

```
svm_learning_df_scores = svm_learning_df.melt(id_vars=['train_size'], value_vars=['train_score', 'test_score'])

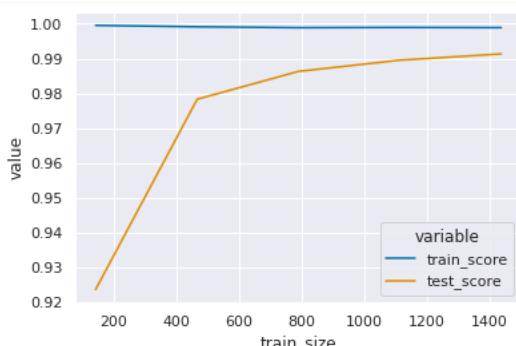
svm_learning_df_scores.head(2)
```

	train_size	variable	value
0	143	train_score	0.999510
1	467	train_score	0.999101

This new DataFrame allows us to make convenient plots.

```
sns.lineplot(data = svm_learning_df_scores, x ='train_size', y='value',hue='variable')
```

```
<AxesSubplot:xlabel='train_size', ylabel='value'>
```



### 💡 Hint

This is one thing we can analyze, but there are others. To earn prepare on assignment 11, manipulate your results a different way.

### 💡 Tip

Getting used to thinking through these sorts of manipulations can take time, but is valuable. Investing time to learn these things will help you both write shorter, more readable, easy to examine, code (which is nicer to your co-workers) and help you develop flexible mental representation. The flexibility of your mental model of material is the way learning scientists distinguish competent practitioners from experts.

```

train_sizes_gnb, train_scores_gnb, test_scores_gnb, fit_times_gnb, score_times_gnb =
model_selection.learning_curve(
 gnb_clf,
 digits_X,
 digits_y,
 cv=cv,
 n_jobs=n_jobs,
 train_sizes=train_sizes,
 return_times=True,
)

```

We can do the same for Gaussian Naive Bayes

```

gnb_learning_df = pd.DataFrame(data = train_sizes_gnb, columns = ['train_size'])
gnb_learning_df['train_size'] = train_sizes_gnb
gnb_learning_df['train_score'] = np.mean(train_scores_gnb, axis=1)
gnb_learning_df['test_score'] = np.mean(test_scores_gnb, axis=1)
gnb_learning_df['fit_time'] = np.mean(fit_times_gnb, axis=1)
gnb_learning_df['score_time_gnb'] = np.mean(score_times_gnb, axis=1)

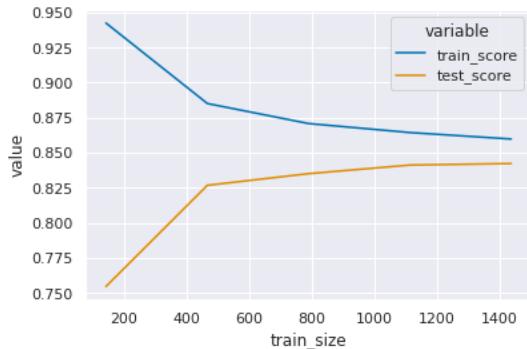
```

```

gnb_learning_scores = gnb_learning_df.melt(id_vars=['train_size'], value_vars=
['train_score', 'test_score'])
sns.lineplot(data = gnb_learning_scores, x ='train_size', y='value', hue='variable')

```

<AxesSubplot:xlabel='train\_size', ylabel='value'>



These scores are overall not as high as the SVM (note the values on the y axis. )

#### ⓘ Question in Class

This was a question after class, but the answer makes more sense inline here.

## 34.1. Questions After Class

### 34.1.1. When would I use an SVM?

## 35. Intro to NLP- representing text data

```

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import euclidean_distances
import pandas as pd

```

```

%load http://drsmb.co/310read

add an entry to the following dictionary with a name as the key and a sentence as the
value
share a sentence about how you're doing this week
remember this will be python code, don't use
You can remain anonymous (this page & the notes will be fully public)
by attributing it to a celebrity or pseudonym, but include *some* sort of attribution
sentence_dict = {
 'Professor Brown': "I'm excited for Thanksgiving.",
 'Matt Langton': "I'm doing pretty good, I'll be taking the days off to catch up on
various classwork.",
 'Evan': "I'm just here so my grade doesn't get fined",
 'Greg Bassett': "I'm doing well, my birthday is today. I'm looking forward to seeing my
family this Thursday, I haven't seen a lot of them in a long time.",
 'Noah N': "I'm doing well! I can't wait to take opportunity of this long weekend to catch
up on various HW's, projects, etc.",
 'Tuyetlinh': "I'm struggling to get all my work done before break, but I'm excited to
have that time off when I'm all done.",
 'Kenza Bouabdallah': "I am doing good. How are you ?",
 'Chris Kerfoot': "I'm doing pretty good. I'm happy to have some days off this week
because of Thanksgiving!",
 'Kang Liu': "New week, new start",
 'Aiden Hill': "I am very much enjoying this class.",
 'Muhammad S': "I am doing pretty well. I am looking forward to taking a few days off.",
 'Max Mastrorocco': "Cannot wait for a break.",
 'Daniel': "I am doing well. I am ready and excited for break!",
 'Nate': "I'm just vibing right now, ready for break",
 'Jacob': "I am going to eat Turkey.",
 'Anon': "nom nom nom"
}

```

How can we analyze these? All of the machine learning models we have seen only use numerical features organized into a table with one row per sample and one column per feature.

That's actually generally true. All ML models require numerical features, at some point. The process of taking data that is not numerical and tabular, which is called unstructured, into structured (tabular) format we require is called feature extraction. There are many, many ways to do that. We'll see a few over the course of the rest of the semester. Some more advanced models hide the feature extraction, by putting it in the same function, but it's always there.

## 35.1. Terms

- document: unit of text we're analyzing (one sample)
- token: sequence of characters in some particular document that are grouped together as a useful semantic unit for processing (basically a word)
- stop words: no meaning, we don't need them (like a, the, an.). Note that this is context dependent
- dictionary: all of the possible words that a given system knows how to process

We'll start by taking out one sentence and analyzing that:

```

s1 = sentence_dict['Professor Brown']

s1

```

```
"I'm excited for Thanksgiving."
```

## 35.2. Bag of Words Representation

We're going to learn a representation called the bag of words. It ignores the order of the words within a document. To do this, we'll first extract all of the tokens (tokenize) the documents and then count how many times each word appears. This will be our numerical representation of the data.

Then we initialize our transformer

### Further Reading

[Transformers](#) are another broad class of sklearn objects. We've seen Estimators mostly so far. We're focusing on the [text feature extraction](#) for now.

```
counts = CountVectorizer()
```

We can use the fit transform method to fit the vectorizer model and apply it to this sentence.

```
counts.fit_transform([s1])
```

```
<1x3 sparse matrix of type '<class 'numpy.int64'>'
with 3 stored elements in Compressed Sparse Row format>
```

To see the output better, we use the toarray method.

```
counts.fit_transform([s1]).toarray()
```

```
array([[1, 1, 1]])
```

We can also examine attributes of the object.

```
counts.vocabulary_
```

```
{'excited': 0, 'for': 1, 'thanksgiving': 2}
```

We see that what it does is creates an ordered (the values are the order) list of words as the parameters of this model (ending in `_` is an attribute of the object or parameter of the model)

### Tip

Notice that we keep using the same tools over and over to explore how things work. You can do this on your own, when you're learning new things. Example code is readily available online but not all of it is well documented or clearly explained.

Also, in a job, much, much, more of your time will be spent reading code than writing code from scratch. These strategies will help you get familiar with a new code base and get up to speed faster.

### Try it yourself

What other model parameters have we seen? How have we used model parameters in the past?

To see what happens a bit more, let's add a second sentence.

```
s2 = sentence_dict['Kang Liu']
s2
```

```
'New week, new start'
```

```
counts.fit_transform([s1,s2])
counts.vocabulary_
```

```
{'excited': 0, 'for': 1, 'thanksgiving': 4, 'new': 2, 'week': 5, 'start': 3}
```

Now we can see that it puts the words in the [vocabulary\\_](#) attribute (aka the [dictionary](#)) in alphabetical order.

```
counts.fit_transform([s1,s2]).toarray()
```

```
array([[1, 1, 0, 0, 1, 0],
 [0, 0, 2, 1, 0, 1]])
```

From this we can see that the representation is the count of how many times each word appears.

Now we can apply it to all of the sentences, or our whole [corpus](#)

```
mat = counts.fit_transform(sentence_dict.values()).toarray()
mat
```

```
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, 1, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0]])
```

We can get the dictionary out in order using the [get\\_feature\\_names](#) method. This method has a generic name, not specific to text, because it's a property of transformers in general.

```
counts.get_feature_names()
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is
deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use
get_feature_names_out instead.
 warnings.warn(msg, category=FutureWarning)
```

```
['all',
 'am',
 'and',
 'are',
 'be',
 'because',
 'before',
 'birthday',
 'break',
 'but',
 'can',
 'cannot',
 'catch',
 'class',
 'classwork',
 'days',
 'doesn',
 'doing',
 'done',
 'eat',
 'enjoying',
 'etc',
 'excited',
 'family',
 'few',
 'fined',
 'for',
 'forward',
 'get',
 'going',
 'good',
 'grade',
 'happy',
 'have',
 'haven',
 'here',
 'how',
 'hw',
 'in',
 'is',
 'just',
 'll',
 'long',
 'looking',
 'lot',
 'much',
 'my',
 'new',
 'nom',
 'now',
 'of',
 'off',
 'on',
 'opportuity',
 'pretty',
 'projects',
 'ready',
 'right',
 'seeing',
 'seen',
 'so',
 'some',
 'start',
 'struggling',
 'take',
 'taking',
 'thanksgiving',
 'that',
 'the',
 'them',
 'this',
 'thursday',
 'time',
 'to',
 'today',
 'turkey',
 'up',
 'various',
 'very',
 'vibing',
 'wait',
 'week',
 'weekend',
 'well',
 'when',
 'work',
 'you']
```

We can use a dataframe again to see this more easily. We can put labels on both the index and the column headings.

```
sentence_df = pd.DataFrame(data = mat, columns =counts.get_feature_names(),
 index=sentence_dict.keys())
sentence_df
```

	all	am	and	are	be	because	before	birthday	break	but	...	variou
<b>Professor Brown</b>	0	0	0	0	0	0	0	0	0	0	0	...
<b>Matt Langton</b>	0	0	0	0	1	0	0	0	0	0	0	...
<b>Evan</b>	0	0	0	0	0	0	0	0	0	0	0	...
<b>Greg Bassett</b>	0	0	0	0	0	0	0	1	0	0	0	...
<b>Noah N</b>	0	0	0	0	0	0	0	0	0	0	0	...
<b>Tuyetlinh</b>	2	0	0	0	0	0	1	0	1	1	1	...
<b>Kenza Bouabdallah</b>	0	1	0	1	0	0	0	0	0	0	0	...
<b>Chris Kerfoot</b>	0	0	0	0	0	1	0	0	0	0	0	...
<b>Kang Liu</b>	0	0	0	0	0	0	0	0	0	0	0	...
<b>Aiden Hill</b>	0	1	0	0	0	0	0	0	0	0	0	...
<b>Muhammad S</b>	0	2	0	0	0	0	0	0	0	0	0	...
<b>Max Mastrorocco</b>	0	0	0	0	0	0	0	0	1	0	0	...
<b>Daniel</b>	0	2	1	0	0	0	0	0	0	1	0	...
<b>Nate</b>	0	0	0	0	0	0	0	0	0	1	0	...
<b>Jacob</b>	0	1	0	0	0	0	0	0	0	0	0	...
<b>Anon</b>	0	0	0	0	0	0	0	0	0	0	0	...

16 rows × 87 columns

### 35.3. How can we find the most commonly used word?

One guess

```
sentence_df.max()
```

```
all 2
am 2
and 1
are 1
be 1
...
weekend 1
well 1
when 1
work 1
you 1
Length: 87, dtype: int64
```

This is the maximum number of times each word appears in single “document”, but it's also not sorted, it's alphabetical.

This shows the word that appears the most times.:

```
sentence_df.max().sort_values()
```

```
looking 1
start 1
some 1
so 1
seen 1
..
my 2
done 2
am 2
all 2
nom 3
Length: 87, dtype: int64
```

To get what we want we need to sum, which by default is along the columns, or per word.

```
sentence_df.sum()
```

```
all 2
am 7
and 1
are 1
be 1
..
weekend 1
well 4
when 1
work 1
you 1
Length: 87, dtype: int64
```

Agaain it's unsorted, but we can apply max

```
sentence_df.sum().max
```

```
<bound method NDFrame._add_numeric_operations.<locals>.max of all 2
am 7
and 1
are 1
be 1
..
weekend 1
well 4
when 1
work 1
you 1
Length: 87, dtype: int64>
```

this gives only t the value though, we want the word. When we summed we got back a Series with the words in the index:

```
sentence_df.sum().index
```

```
Index(['all', 'am', 'and', 'are', 'be', 'because', 'before', 'birthday',
 'break', 'but', 'can', 'cannot', 'catch', 'class', 'classwork', 'days',
 'doesn', 'doing', 'done', 'eat', 'enjoying', 'etc', 'excited', 'family',
 'few', 'fined', 'for', 'forward', 'get', 'going', 'good', 'grade',
 'happy', 'have', 'haven', 'here', 'how', 'hw', 'in', 'is', 'just', 'll',
 'long', 'looking', 'lot', 'much', 'my', 'new', 'nom', 'now', 'of',
 'off', 'on', 'opportunity', 'pretty', 'projects', 'ready', 'right',
 'seeing', 'seen', 'so', 'some', 'start', 'struggling', 'take', 'taking',
 'thanksgiving', 'that', 'the', 'them', 'this', 'thursday', 'time', 'to',
 'today', 'turkey', 'up', 'various', 'very', 'vibing', 'wait', 'week',
 'weekend', 'well', 'when', 'work', 'you'],
 dtype='object')
```

So we can use idxmax

```
sentence_df.sum().idxmax()
```

```
'to'
```

### 35.4. Distances in text

We can now use a distance function to calculate how far apart the different sentences are.

```
euclidean_distances(sentence_df)
```

```
array([[0. , 4.24264069, 3.31662479, 5.19615242, 4.89897949,
 5.09901951, 3. , 3.87298335, 3. , 3. ,
 4.12310563, 2.23606798, 3.16227766, 2.82842712, 2.82842712,
 3.46410162],
 [4.24264069, 0. , 4.79583152, 5.91607978, 4.69041576,
 5.83095189, 4.12310563, 4.12310563, 4.58257569, 4.58257569,
 4.12310563, 4.35889894, 4.89897949, 4.69041576, 4.24264069,
 4.89897949],
 [3.31662479, 4.79583152, 0. , 5.29150262, 5.38516481,
 5.38516481, 3.74165739, 4.69041576, 3.74165739, 3.74165739,
 4.69041576, 3.46410162, 4.35889894, 3.60555128, 3.60555128,
 4.12310563],
 [5.19615242, 5.91607978, 5.29150262, 0. , 5.56776436,
 6.244998 , 5.29150262, 5.47722558, 5.47722558, 5.29150262,
 5.29150262, 5.29150262, 5.56776436, 5.56776436, 5.19615242,
 5.74456265],
 [4.89897949, 4.69041576, 5.38516481, 5.56776436, 0. ,
 6.164414 , 5. , 5. , 5.19615242, 5. ,
 5.19615242, 4.79583152, 5.29150262, 5.29150262, 4.69041576,
 5.47722558],
 [5.09901951, 5.83095189, 5.38516481, 6.244998 , 6.164414 ,
 0. , 5.56776436, 5.56776436, 5.56776436,
 5.74456265, 5.19615242, 5.65685425, 5.47722558, 5.09901951,
 5.83095189],
 [3. , 4.12310563, 3.74165739, 5.29150262, 5. ,
 5.56776436, 0. , 4. , 3.46410162, 3.16227766,
 3.74165739, 3.16227766, 3.31662479, 3.60555128, 3. ,
 3.87298335],
 [3.87298335, 4.12310563, 4.69041576, 5.47722558, 5. ,
 5.56776436, 4. , 0. , 4.24264069, 4.24264069,
 4.24264069, 4.24264069, 4.79583152, 4.58257569, 4.12310563,
 4.79583152],
 [3. , 4.58257569, 3.74165739, 5.47722558, 5.19615242,
 5.56776436, 3.46410162, 4.24264069, 0. , 3.46410162,
 4.47213595, 3.16227766, 4.12310563, 3.60555128, 3.31662479,
 3.87298335],
 [3. , 4.58257569, 3.74165739, 5.29150262, 5. ,
 5.56776436, 3.16227766, 4.24264069, 3.46410162, 0. ,
 4. , 3.16227766, 3.60555128, 3.60555128, 3. ,
 3.87298335],
 [4.12310563, 4.12310563, 4.69041576, 5.29150262, 5.19615242,
 5.74456265, 3.74165739, 4.24264069, 4.47213595, 4. ,
 0. , 4.24264069, 3.60555128, 4.58257569, 3.60555128,
 4.79583152],
 [2.23606798, 4.35889894, 3.46410162, 5.29150262, 4.79583152,
 5.19615242, 3.16227766, 4.24264069, 3.16227766, 3.16227766,
 4.24264069, 0. , 3.31662479, 2.64575131, 3. ,
 3.60555128],
 [3.16227766, 4.89897949, 4.35889894, 5.56776436, 5.29150262,
 5.65685425, 3.31662479, 4.79583152, 4.12310563, 3.60555128,
 3.60555128, 3.31662479, 0. , 3.46410162, 3.46410162,
 4.47213595],
 [2.82842712, 4.69041576, 3.60555128, 5.56776436, 5.29150262,
 5.47722558, 3.60555128, 4.58257569, 3.60555128, 3.60555128,
 4.58257569, 2.64575131, 3.46410162, 0. , 3.46410162,
 4.],
 [2.82842712, 4.24264069, 3.60555128, 5.19615242, 4.69041576,
 5.09901951, 3. , 4.12310563, 3.31662479, 3. ,
 3.60555128, 3. , 3.46410162, 3.46410162, 0. ,
 3.74165739],
 [3.46410162, 4.89897949, 4.12310563, 5.74456265, 5.47722558,
 5.83095189, 3.87298335, 4.79583152, 3.87298335, 3.87298335,
 4.79583152, 3.60555128, 4.47213595, 4. , 3.74165739,
 0.]])
```

We can make this easier to read by making it a Data Frame.

```
dist_df = pd.DataFrame(data = euclidean_distances(sentence_df),
 index= sentence_dict.keys(), columns= sentence_dict.keys())
dist_df
```

	<b>Professor Brown</b>	<b>Matt Langton</b>	<b>Evan</b>	<b>Greg Bassett</b>	<b>Noah N</b>	<b>Tuyetlinh</b>	<b>Kenza Bouabdallah</b>
<b>Professor Brown</b>	0.000000	4.242641	3.316625	5.196152	4.898979	5.099020	3.000000
<b>Matt Langton</b>	4.242641	0.000000	4.795832	5.916080	4.690416	5.830952	4.123100
<b>Evan</b>	3.316625	4.795832	0.000000	5.291503	5.385165	5.385165	3.741650
<b>Greg Bassett</b>	5.196152	5.916080	5.291503	0.000000	5.567764	6.244998	5.291500
<b>Noah N</b>	4.898979	4.690416	5.385165	5.567764	0.000000	6.164414	5.000000
<b>Tuyetlinh</b>	5.099020	5.830952	5.385165	6.244998	6.164414	0.000000	5.567760
<b>Kenza Bouabdallah</b>	3.000000	4.123106	3.741657	5.291503	5.000000	5.567764	0.000000
<b>Chris Kerfoot</b>	3.872983	4.123106	4.690416	5.477226	5.000000	5.567764	4.000000
<b>Kang Liu</b>	3.000000	4.582576	3.741657	5.477226	5.196152	5.567764	3.464100
<b>Aiden Hill</b>	3.000000	4.582576	3.741657	5.291503	5.000000	5.567764	3.162270
<b>Muhammad S</b>	4.123106	4.123106	4.690416	5.291503	5.196152	5.744563	3.741650
<b>Max Mastrorocco</b>	2.236068	4.358899	3.464102	5.291503	4.795832	5.196152	3.162270
<b>Daniel</b>	3.162278	4.898979	4.358899	5.567764	5.291503	5.656854	3.316620
<b>Nate</b>	2.828427	4.690416	3.605551	5.567764	5.291503	5.477226	3.605550
<b>Jacob</b>	2.828427	4.242641	3.605551	5.196152	4.690416	5.099020	3.000000
<b>Anon</b>	3.464102	4.898979	4.123106	5.744563	5.477226	5.830952	3.872980

Who wrote the most similar question to me?

```
dist_df['Professor Brown'].drop('Professor Brown').idxmin()
```

```
'Max Mastrorocco'
```

## 35.5. Questions After Classroom

### 35.5.1. How can this be used for training a classifier?

### 35.5.2. How are ram tokens tracked?

## 35.6. More Practice

1. Which two people wrote the most similar sentences?
2. Do you think this representation captures all cases of similiary? Can you generate a case where it doesn't do well?
3. Try

## 36. More NLP & Solving problems with ML

Imagine you work for a news agency, current articles are stored in a database in a table where they are indexed by a unique identifier. A separate table indicates the primary category for most articles, but some are missing. Your assignment is to create an automatic category generator to save editors time. Your manager suggests that an SVM or decision tree is probably best, but is unsure what text representations will perform best. You are expected to produce an accessible report with tables and plots that visualize the performance and impact of various modeling choices.

Write the import statements you would need to solve this problem. Include whole libraries or modules as is most appropriate.

```
import pandas as pd
import sqlite3
from sklearn.feature_extraction import text
from sklearn import tree
from sklearn import svm
from sklearn import model_selection
import seaborn as sns
```

Given the following vocabulary,

['and', 'are', 'cat', 'cats', 'dogs', 'pets', 'popular', 'videos']

represent "Cats and dogs are pets" in vector format.

[1,1,0,1,1,1,0,0]

```
from sklearn.feature_extraction import text
from sklearn.metrics.pairwise import euclidean_distances
from sklearn import datasets
import pandas as pd
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split

import a python utility for examining objects for the notes
from sys import getsizeof
import numpy as np
import seaborn as sns
ng_X, ng_y = datasets.fetch_20newsgroups(categories=['comp.graphics', 'sci.crypt'],
 return_X_y=True)
```

type(ng\_y)

numpy.ndarray

ng\_y[:5]

array([0, 0, 1, 0, 0])

ng\_X[0]

"From: robert@cpuserver.acsc.com (Robert Grant)\nSubject: Virtual Reality for X on the CHEAP!\nOrganization: USCACSC, Los Angeles\nLines: 18\nDistribution: world\nReply-To: robert@cpuserver.acsc.com (Robert Grant)\nNNTP-Posting-Host: cpuserver.acsc.com\nHi everyone,\nI thought that some people may be interested in my VR\software on these groups:\n\*\*\*\*\*Announcing the release of Multiverse-1.0.\*\*\*\*\*\nMultiverse is a multi-user, non-immersive, X-Windows based Virtual Reality\software, primarily focused on entertainment/research.\n\nFeatures:\nClient-Server based model, using Berkeley Sockets.\nNo limit to the number of users (apart from performance).\nGeneric clients.\nCustomizable servers.\nHierarchical Objects (allowing attachment of cameras and light sources).\nMultiple light sources (ambient, point and spot).\nObjects can have extension code, to handle unique functionality, easily\nattached.\n\nFunctionality:\nClient:\nThe client is built around a 'fast' render loop. Basically it changes things\nwhen told to by the server and then renders an image from the user's\nviewpoint. It also provides the server with information about what the user's\nactions - which can then be communicated to other clients and therefore to\nother users.\n\nThe client is designed to be generic - in other words you don't need to\ndevelop a new client when you want to enter a new world. This means that\nresources can be spent on enhancing the client software rather than adapting\nit. The adaptations, as will be explained in a moment, occur in the servers.\n\nThis release of the client software supports the following functionality:\n\no Hierarchical Objects (with associated addressing)\no Multiple Light Sources and Types (Ambient, Point and Spot)\no User Interface Panels\no Colour Polygonal Rendering with Phong Shading (optional wireframe for\nfaster frame rates)\no Mouse and Keyboard Input\n\n(Some people may be disappointed that this software doesn't support the\nPowerGlove as an input device - this is not because it can't, but because I don't have one! This will, however, be one of the first enhancements!)\n\nServer(s):\n\nThis is where customization can take place. The following basic support is\nprovided in this release for potential world server developers:\n\no Transparent Client Management\no Client Message Handling\n\nThis may not sound like much, but it takes away the headache of accepting and\nterminating clients and receiving messages from them - the application writer\ncan work with the assumption that things are happening locally.\n\nThings get more interesting in the object extension functionality. This is\nwhat is provided to allow you to animate your objects:\n\no Server Selectable Extension Installation:\nWhat this means is that you can decide which objects have extended\nfunctionality in your world. Basically you call the extension\ninitialisers you want.\n\no Event Handler Registration:\nWhen you develop extensions for an object you basically write callback\nfunctions for the events that you want the object to respond to.\n\n(Current events supported: INIT, MOVE, CHANGE, COLLIDE & TERMINATE)\n\no Collision Detection Registration:\nIf you want your object to respond to collision events just provide\nsome basic information to the collision detection management software.\nYour callback will be activated when a collision occurs.\n\nThis software is kept separate from the worldServer applications because\nthe application developer wants to build a library of extended objects\nfrom which to choose.\n\nThe following is all you need to make a World Server application:\n\no Provide an initWorld function:\nThis is where you choose what object extensions will be supported, plus\nany initialization you want to do.\n\no Provide a positionObject function:\nThis is where you determine where to place a new client.\n\no Provide an installWorldObjects function:\nThis is where you load the world (.wld) file for a new client.\n\no Provide a getWorldType function:\nThis is where you tell a new client what persona they should have.\n\no Provide an animateWorld function:\nThis is where you can go wild! At a minimum you should let the objects\nmove (by calling a move function) and let the server sleep for a bit\n(to avoid outrunning the clients).\n\nThat's all there is to it! And to prove it here are the line counts for the\nthree world servers I've provided:\n\n- generic - 81 lines\n- dactyl - 270 lines (more complicated collision detection due to the\nstairs! Will probably be improved with future\nversions)\n\n- dogfight - 72 lines\n\nLocation:\n\nThis software is located at the following site:\nftp.u.washington.edu\n\nDirectory:\n\npub/virtual-worlds\n\nFile:\n\nmultiverse-1.0.2.tar.Z\n\nFutures:\n\nClient:\n\no Texture mapping.\n\no More realistic rendering: i.e. Z-Buffering (or similar), Gouraud shading\n\no HMD support.\n\no Etc, etc...\n\nServer:\n\no Physical Modelling (gravity, friction etc).\n\no Enhanced Object Management/Interaction\n\no Etc, etc...\n\nBoth:\n\no Improved Comms!!!\n\nI hope this provides people with a good understanding of the Multiverse\software, unfortunately it comes with practically zero documentation, and I'm not sure\nwhether that will ever be able to be rectified! :-(\n\nI hope people enjoy this software and that it is useful in our explorations of\nthe Virtual Universe - I've certainly found fascinating developing it, and I would \*LOVE\*\nto add support for the PowerGlove...and an HMD :-(\n\nFinally one major disclaimer:\n\nThis is totally amateur code. By that I mean there is no support for this code\\nother than what I, out the kindness of my heart, or you, out of pure\\ndesperation, provide. I cannot be held responsible for anything good or bad\\nthat may happen through the use of this code - USE IT AT YOUR OWN RISK!\n\nDisclaimer over!\n\nOf course if you love it, I would like to hear from you. And anyone with\\nPOSITIVE contributions/criticisms is also encouraged to contact me. Anyone who\\nhates it: >\n\n-----\n\nAnd if anyone wants to let me do this for a living: you know where to\\nwright :-)\n\n-----\n\nThanks,\n\nRobert.\n\nrobert@acsc.com\n\n\*\*\*\*\*\n

We're going to instantiate the object and fit it to the whole dataset.

```
count_vec = text.CountVectorizer()
count_vec.fit(ng_X)
```

```
CountVectorizer()
```

### ! Important

I changed the following a little bit from class so that we can use the same test/train split for the two different types of transformation so that we can compare them more easily.

This also helps illustrate when using the fit and transform separately is helpful.

```
ng_X_train, ng_X_test, ng_y_train, ng_y_test = train_test_split(
 ng_X, ng_y, random_state=0)
```

Now, we can use the transformation that we fit to the whole dataset to transform the train and test portions of the data separately.

The transform method also returns the sparse matrix directly so we no longer need the `toarray` method.

```
ng_vec_train = count_vec.transform(ng_X_train)
ng_vec_test = count_vec.transform(ng_X_test)
```

```
ng_vec_train[0]
```

```
<1x24257 sparse matrix of type '<class 'numpy.int64'>'
with 84 stored elements in Compressed Sparse Row format>
```

```
clf = MultinomialNB()
```

```
clf.fit(ng_vec_train, ng_y_train).score(ng_vec_test, ng_y_test)
```

```
0.9830508474576272
```

```
tfidf = text.TfidfTransformer()
tfidf.fit_transform(ng_X)
```

```

 ValueError Traceback (most recent call last)
/tmp/ipykernel_2594/2043128607.py in <module>
 1 tfidf = text.TfidfTransformer()
--> 2 tfidf.fit_transform(ng_X)

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/sklearn/base.py in
fit_transform(self, X, y, **fit_params)
 845 if y is None:
 846 # fit method of arity 1 (unsupervised transformation)
--> 847 return self.fit(X, **fit_params).transform(X)
 848 else:
 849 # fit method of arity 2 (supervised transformation)

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/sklearn/feature_extraction/text.py in fit(self, X, y)
 1613 # dtype NPY_INTP which is int32 for 32bit platforms. See #20923
 1614 X = self._validate_data(
--> 1615 X, accept_sparse=("CSR", "CSC"), accept_large_sparse=not _IS_32BIT
 1616)
 1617 if not sp.issparse(X):

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/sklearn/base.py in
_validate_data(self, X, y, reset, validate_separately, **check_params)
 559 raise ValueError("Validation should be done on X, y or both.")
 560 elif not no_val_X and no_val_y:
--> 561 X = check_array(X, **check_params)
 562 out = X
 563 elif no_val_X and not no_val_y:
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/sklearn/utils/validation.py in check_array(array, accept_sparse,
accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd,
ensure_min_samples, ensure_min_features, estimator)
 "Reshape your data either using array.reshape(-1, 1) if "
 "your data has a single feature or array.reshape(1, -1) "
 "'if it contains a single sample.'".format(array)
)
)

ValueError: Expected 2D array, got 1D array instead:
array=[From: robert@cpuserver.acsc.com (Robert Grant)\nSubject: Virtual Reality for X on
the CHEAP!\nOrganization: USCACSC, Los Angeles\nLines: 187]\nDistribution: world\nReply-
To: robert@cpuserver.acsc.com (Robert Grant)\nNNTP-Posting-Host: cpuserver.acsc.com\n\nHi
everyone,\nI thought that some people may be interested in my VR\nsoftware on these
groups:\n*****Announcing the release of Multiverse-1.0.2*****\nMultiverse is a
multi-user, non-immersive, X-Windows based Virtual Reality\nsystem, primarily focused on
entertainment/research.\n\nFeatures:\n\n Client-Server based model, using Berkeley
Sockets.\n No limit to the number of users (apart from performance).\n Generic
clients.\n Customizable servers.\n Hierarchical Objects (allowing attachment of
cameras and light sources).\n Multiple light sources (ambient, point and spot).\n
Objects can have extension code, to handle unique functionality, easily\n
attached.\n\nFunctionality:\n\n Client:\n The client is built around a 'fast' render
loop. Basically it changes things\n when told to by the server and then renders an
image from the user's\n viewpoint. It also provides the server with information about
the user's\n actions - which can then be communicated to other clients and therefore
to\n other users.\n\n The client is designed to be generic - in other words you don't
need to\n develop a new client when you want to enter a new world. This means that\n
resources can be spent on enhancing the client software rather than adapting\n it. The
adaptations, as will be explained in a moment, occur in the servers.\n\n This release
of the client software supports the following functionality:\n\n o Hierarchical
Objects (with associated addressing)\n\n o Multiple Light Sources and Types (Ambient,
Point and Spot)\n\n o User Interface Panels\n\n o Colour Polygonal Rendering with
Phong Shading (optional wireframe for\nfaster frame rates)\n\n o Mouse and Keyboard
Input\n\n (Some people may be disappointed that this software doesn't support the\n
PowerGlove as an input device - this is not because it can't, but because\n I don't
have one! This will, however, be one of the first enhancements!)\n\n Server(s):\n This
is where customization can take place. The following basic support is\n provided in
this release for potential world server developers:\n\n o Transparent Client
Management\n\n o Client Message Handling\n\n This may not sound like much, but it
takes away the headache of\naccepting and\n terminating clients and receiving messages
from them - the\napplication writer\n can work with the assumption that things are
happening locally.\n\n Things get more interesting in the object extension
functionality. This is\n what is provided to allow you to animate your objects:\n\n
o Server Selectable Extension Installation:\n\n What this means is that you can
decide which objects have extended\n functionality in your world. Basically you
call the extension\n initialisers you want.\n\n o Event Handler Registration:\n
When you develop extensions for an object you basically write callback\n functions
for the events that you want the object to respond to.\n\n (Current events
supported: INIT, MOVE, CHANGE, COLLIDE & TERMINATE)\n\n o Collision Detection
Registration:\n\n If you want your object to respond to collision events just
provide\n some basic information to the collision detection management software.\n
Your callback will be activated when a collision occurs.\n\n This software is kept
separate from the worldServer applications because\n the application developer wants
to build a library of extended objects\n from which to choose.\n\n The following is
all you need to make a World Server application:\n\n o Provide an initWorld
function:\n\n This is where you choose what object extensions will be supported,
plus\n any initialization you want to do.\n\n o Provide a positionObject
function:\n\n This is where you determine where to place a new client.\n\n
o Provide an installWorldObjects function:\n\n This is where you load the world (.wld)
file for a new client.\n\n o Provide a getWorldType function:\n\n This is where
you tell a new client what persona they should have.\n\n o Provide an animateWorld
function:\n\n This is where you can go wild! At a minimum you should let the
objects\n move (by calling a move function) and let the server sleep for a bit\n
(to avoid outrunning the clients).\n\n That's all there is to it! And to prove it here
are the line counts for the\n three world servers I've provided:\n\n generic -
81 lines\n dactyl - 270 lines (more complicated collision detection due to the\n
stairs! Will probably be improved with future\n versions)\n dogfight - 72 lines\n\nLocation:\n\n This software is located at the following site:\n
ftp.u.washington.edu/\n\n Directory:\n pub/virtual-worlds\n\n File:\n multiverse-
1.0.2.tar.Z\n\nFutures:\n\n Client:\n\n o Texture mapping.\n\n o More realistic
rendering: i.e. Z-Buffering (or similar), Gouraud shading\n\n o HMD support.\n\n o
Etc, etc....\n\n Server:\n\n o Physical Modelling (gravity, friction etc).\n\n o
Enhanced Object Management/Interaction\n\n o Etc, etc....\n\n Both:\n\n o
Improved Comms!!!\n\nI hope this provides people with a good understanding of the
Multiverse\nsoftware, \nunfortunately it comes with practically zero documentation, and
I'm not sure\nwhether that will ever be able to be rectified! :-(\n\nI hope people enjoy
this software and that it is useful in our explorations of\nthe Virtual Universe - I've
certainly found fascinating developing it, and I\nwould *LOVE* to add support for the
PowerGlove...and an HMD :-)!)\n\nFinally one major disclaimer:\n\nThis is totally amateur
code. By that I mean there is no support for this code\nother than what I, out the
kindness of my heart, or you, out of pure\ndesperation, provide. I cannot be held
responsible for anything good or bad\nthat may happen through the use of this code - USE
IT AT YOUR OWN RISK!\n\nDisclaimer over!\n\nOf course if you love it, I would like to
here from you. And anyone with\nPOSITIVE contributions/criticisms is also encouraged to
contact me. Anyone who\nhates it: >
/dev/null!\n*****
```

We can figure out why, we can compare what that method takes as input to what the count vectorizer takes as input. While working, the easiest way to do this is using the jupyter help (shift+tab, or ?) but for the notes, I'll print them out differently.

```
print('\n'.join(tfidf.fit_.doc_.split('\n')[:7]))
```

Learn the idf vector (global term weights).

### Parameters

-----

X : sparse matrix of shape n\_samples, n\_features)  
A matrix of term/token counts.

```
print('\n'.join(count_vec.fit.__doc__.split('\n')[:7]))
```

```
Learn a vocabulary dictionary of all tokens in the raw documents.
```

```
Parameters
```

```

```

```
raw_documents : iterable
```

```
 An iterable which generates either str, unicode or file objects.
```

We wanted the TfidfVectorizer, not the transformer, so that it accepts documents not features. We will again, instantiate the object and then fit on the whole dataset.

```
tfidf = text.TfidfVectorizer()
tfidf.fit(ng_X)
```

```
TfidfVectorizer()
```

We can see this works, because the code runs, but for completeness , we can also check the input again to compare with the above.

```
print('\n'.join(tfidf.fit.__doc__.split('\n')[:6]))
```

```
Learn vocabulary and idf from training set.
```

```
Parameters
```

```

```

```
raw_documents : iterable
```

```
 An iterable which generates either str, unicode or file objects.
```

This now takes documents as we wanted. Since we split the data before transforming, we can then apply the new fit using the transform on the train/test splits.

```
ng_tfidf_train = tfidf.transform(ng_X_train)
ng_tfidf_test = tfidf.transform(ng_X_test)
```

Since these splits were made before we can use the same targets we used above.

```
clf.fit(ng_tfidf_train,ng_y_train).score(ng_tfidf_test,ng_y_test)
```

```
0.9288135593220339
```

## 36.1. Comparing representations

To start, we will look at one element from each in order to compare them.

```
ng_tfidf_train[0]
```

```
<1x24257 sparse matrix of type '<class 'numpy.float64'>'
with 84 stored elements in Compressed Sparse Row format>
```

```
ng_vec_train[0]
```

```
<1x24257 sparse matrix of type '<class 'numpy.int64'>'
with 84 stored elements in Compressed Sparse Row format>
```

To start they both have 84 elements, since it is two different representations of the same document, that makes sense. We can check a few others as well

```
ng_tfidf_train[1]
```

```
<1x24257 sparse matrix of type '<class 'numpy.float64'>'
with 202 stored elements in Compressed Sparse Row format>
```

```
ng_vec_train[1]
```

```
<1x24257 sparse matrix of type '<class 'numpy.int64'>'
with 202 stored elements in Compressed Sparse Row format>
```

```
(ng_vec_train[4]>0).sum() == (ng_tfidf_train[4]>0).sum()
```

```
True
```

Let's pick out a common word so that the calculation is meaningful and do the tfidf calcuation. To find a common word in the dictionary, we'll first filter the vocabulary to keep only the words that occur at least 300 times in the training set. We sum along the columns of the matrix, transform it to an array, then iterate over the sum, enumerated (assigning the number to each element of the sum) and use that to get the word out, if its total is over 300. I saw that this is actually a sort of long list, so I chose to only print out the first 25. We print them out with the index so we can use it for the one we choose.

```
[(count_vec.get_feature_names()[i], i) for i, n in
 enumerate(np.asarray(ng_vec_train.sum(axis=0))[0])
 if n>300][:25]
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is
deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use
get_feature_names_out instead.
 warnings.warn(msg, category=FutureWarning)
```

```
[('about', 3326),
 ('all', 3819),
 ('also', 3874),
 ('an', 3964),
 ('and', 4003),
 ('any', 4115),
 ('are', 4263),
 ('article', 4347),
 ('as', 4363),
 ('at', 4456),
 ('available', 4612),
 ('be', 4851),
 ('been', 4887),
 ('bit', 5096),
 ('but', 5582),
 ('by', 5605),
 ('can', 5779),
 ('chip', 6191),
 ('clipper', 6393),
 ('com', 6568),
 ('computer', 6786),
 ('could', 7180),
 ('cs', 7426),
 ('data', 7687),
 ('db', 7730)]
```

Let's use computer.

```
computer_idx = 6786
count_vec.get_feature_names()[computer_idx]
```

```
'computer'
```

```
ng_vec_train[:,computer_idx].toarray()[:10].T
```

```
array([[0, 0, 0, 5, 0, 0, 0, 0, 0, 0]])
```

```
ng_tfidf_train[:,computer_idx].toarray()[:10].T
```

```
array([[0. , 0. , 0. , 0.06907742, 0. , 0. , 0. , 0. , 0. , 0.]])
```

So, we can see they have non zero elements in the same places, meaning that in both representations the column refers to the same thing.

We can compare the untransformed to the count vectorizer:

```
len(ng_X_train[0].split())
```

```
100
```

```
ng_vec_train[0].sum()
```

```
100
```

We see that it is just a count of the number of total words, not unique words, but total

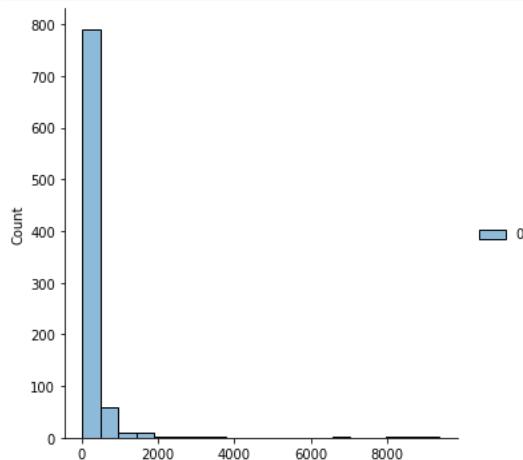
```
ng_tfidf_train[0].sum()
```

```
7.558255873996122
```

The tf-idf matrix, however is normalized to make the sums smaller. Each row is not the same, but it is more similar.

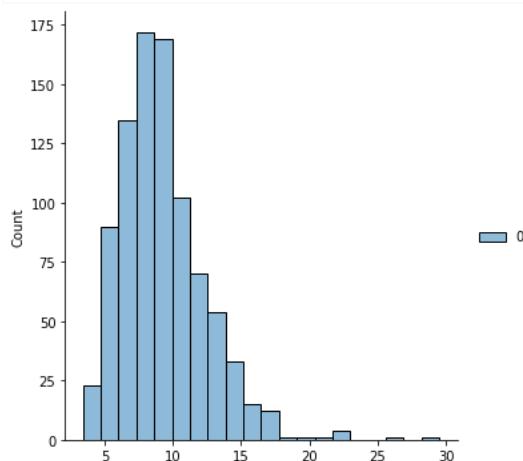
```
sns.displot(ng_vec_train.sum(axis=1), bins=20)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f3dc4f01a90>
```



```
sns.displot(ng_tfidf_train.sum(axis=1), bins=20)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f3dc4f41650>
```



We can see that the **tf-idf** makes the totals across documents more spread out.

When we sum across words, we then get to see how

From the documentation we see that the idf is not exactly the inverse of the number of documents, it's also rescaled some.

$$\text{idf} = \log \frac{1 + n}{1 + df} + 1$$

In this implementation, each row is the normalized as well, to keep them small, so that documents of different sizes are more comparable. For example, if we return to what we did last week.

```
%load http://drsmb.co/310read

add an entry to the following dictionary with a name as the key and a sentence as the
value
share a sentence about how you're doing this week
remember this will be python code, don't use
You can remain anonymous (this page & the notes will be fully public)
by attributing it to a celebrity or pseudonym, but include *some* sort of attribution
sentence_dict = {
 'Professor Brown': "I'm excited for Thanksgiving.",
 'Matt Langton': "I'm doing pretty good, I'll be taking the days off to catch up on
various classwork.",
 'Evan': "I'm just here so my grade doesn't get fined",
 'Greg Bassett': "I'm doing well, my birthday is today. I'm looking forward to seeing my
family this Thursday, I haven't seen a lot of them in a long time.",
 'Noah N': "I'm doing well! I can't wait to take opportunity of this long weekend to catch
up on various HW's, projects, etc.",
 'Tuyetlinh': "I'm struggling to get all my work done before break, but I'm excited to
have that time off when I'm all done.",
 'Kenza Bouabdallah': "I am doing good. How are you ?",
 'Chris Kerfoot': "I'm doing pretty good. I'm happy to have some days off this week
because of Thanksgiving!",
 'Kang Liu': "New week, new start",
 'Aiden Hill': "I am very much enjoying this class.",
 'Muhammad S': "I am doing pretty well. I am looking forward to taking a few days off.",
 'Max Mastrorocco': "Cannot wait for a break.",
 'Daniel': "I am doing well. I am ready and excited for break!",
 'Nate': "I'm just vibing right now, ready for break",
 'Jacob': "I am going to eat Turkey.",
 'Anon': "nom nom nom"
}
create a new count vectorizer so that we can return to the news analysis
count_vec_ex = text.CountVectorizer()
tfidf_ex = text.TfidfVectorizer()
mat_c = count_vec_ex.fit_transform(sentence_dict.values()).toarray()
mat_t = tfidf_ex.fit_transform(sentence_dict.values()).toarray()
sentence_df = pd.DataFrame(data = mat_c,
columns = counts.get_feature_names_out(),
index=sentence_dict.keys())

dist_df_count = pd.DataFrame(data = euclidean_distances(mat_c),
 index= sentence_dict.keys(), columns= sentence_dict.keys())
dist_df_tfidf = pd.DataFrame(data = euclidean_distances(mat_t),
 index= sentence_dict.keys(), columns= sentence_dict.keys())

ref = 'Professor Brown'
```

```
count_closest = dist_df_count[ref].drop(ref).idxmin()
count_closest
```

```
'Max Mastrorocco'
```

```
tfidf_closest = dist_df_tfidf[ref].drop(ref).idxmin()
tfidf_closest
```

```
'Daniel'
```

```
sentence_dict[ref]
```

```
"I'm excited for Thanksgiving."
```

```
sentence_dict[count_closest]
```

```
'Cannot wait for a break.'
```

```
sentence_dict[tfidf_closest]
```

```
'I am doing well. I am ready and excited for break!'
```

Now, the closest sentence actually shares a word that's similar.

```
dist_df_count[ref].drop(ref).sort_values()
```

```
Max Mastrorocco 2.236068
Nate 2.828427
Jacob 2.828427
Kenza Bouabdallah 3.000000
Kang Liu 3.000000
Aiden Hill 3.000000
Daniel 3.162278
Evan 3.316625
Anon 3.464102
Chris Kerfoot 3.872983
Muhammad S 4.123106
Matt Langton 4.242641
Noah N 4.898979
Tuyetlinh 5.099020
Greg Bassett 5.196152
Name: Professor Brown, dtype: float64
```

```
dist_df_tfidf[ref].drop(ref).sort_values()
```

```
Daniel 1.153404
Max Mastrorocco 1.248303
Chris Kerfoot 1.279299
Nate 1.299060
Tuyetlinh 1.341626
Noah N 1.414214
Kenza Bouabdallah 1.414214
Aiden Hill 1.414214
Muhammad S 1.414214
Jacob 1.414214
Anon 1.414214
Matt Langton 1.414214
Evan 1.414214
Greg Bassett 1.414214
Kang Liu 1.414214
Name: Professor Brown, dtype: float64
```

Now, the distances are all much smaller and Chris who also talked about Thanksgiving is much higher on the list too.

```
mat_c.sum(axis=1)
```

```
array([3, 15, 8, 22, 19, 19, 6, 14, 4, 6, 12, 4, 9, 7, 5, 3])
```

```
mat_t.sum(axis=1)
```

```
array([1.72586983, 3.81967092, 2.81780765, 4.4761216 , 4.1953457 ,
 3.75971487, 2.39375957, 3.68712845, 1.61308516, 2.41468677,
 3.22271608, 1.97816206, 2.73725829, 2.62109557, 2.17780895,
 1.])
```

The sums are again much closer so the total length isn't as much of the signal. Since it's smaller, we can also sum along words.

```
sent_word_df = pd.DataFrame(data = count_vec_ex.get_feature_names(),
 columns = ['word'])
sent_word_df['count_tot'] = mat_c.sum(axis=0)
sent_word_df['tfidf_tot'] = mat_t.sum(axis=0)
sent_word_df['doc_count'] = (mat_c>0).sum(axis=0)
```

```
/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is
deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use
get_feature_names_out instead.
 warnings.warn(msg, category=FutureWarning)
```

```
sent_word_df.sort_values(by='doc_count', ascending=False)
```

	word	count_tot	tfidf_tot	doc_count
73	to	9	1.513881	7
17	doing	7	1.326396	7
1	am	7	1.943291	5
51	off	4	0.860791	4
70	this	4	0.899345	4
...	...	...	...	...
32	happy	1	0.327365	1
31	grade	1	0.374658	1
29	going	1	0.517461	1
25	fined	1	0.374658	1
86	you	1	0.479913	1

87 rows × 4 columns

## 36.2. Sparse Matrices

To understand the sparse matrices, we can examine the size of the objects as a sparse matrix and after exporting to array.

```
getsizeof(ng_vec_train[0])
```

```
64
```

```
getsizeof(ng_vec_train[0].toarray())
```

```
194176
```

This is another reason it is better to use the sparse matrices (as returned by `transform`, but not by `fit_transform`)

```
ng_tfidf_train[0].toarray()
```

```
array([[0., 0., 0., ..., 0., 0., 0.]])
```

## 36.3. More Practice

1. On the sentence dataset, try implementing the tf-idf calculation.

## 37. Neural Networks

We started thinking about machine learning with the idea that the basic idea is that we assume that our target variable ( $y_i$ ) is related to the features ( $x_i$ ) by some function (for sample  $i$ ):  
$$y_i = f(x_i)$$

But we don't know that function exactly, so we assume a type of  $f$  (a decision tree, a boundary for SVM, a probability distribution) that has some parameters  $\theta$  and then use a machine learning algorithm  $\mathcal{A}$  to estimate the parameters for  $f$ . In the decision tree the parameters are the thresholds to compare to, in the GaussianNB the parameters are the mean and variance, in SVM it's the support vectors that define the margin.

$\theta = \mathcal{A}(X, y)$

That we can use to test on our test data:

$\hat{y}_i = f(x_i; \theta)$

A neural net allows us to not assume a specific form for  $f$  first, it does universal function approximation. For one hidden layer and a binary classification problem:

$f(x) = W_2 g(W_1^T x + b_1) + b_2$

where the function  $g$  is called the activation function. so we approximate some unknown, complicated function  $f$  by taking a weighted sum of all of the inputs, and passing those through another, known function.

```
from sklearn.neural_network import MLPClassifier
from sklearn import svm
import pandas as pd
import sklearn

from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn import model_selection
import numpy as np
```

We're going to use the digits dataset again.

```
digits = datasets.load_digits()
digits_X = digits.data
digits_y = digits.target
X_train, X_test, y_train, y_test = model_selection.train_test_split(digits_X, digits_y)
```

```
digits.images[0]
```

```
array([[0., 0., 5., 13., 9., 1., 0., 0.],
 [0., 0., 13., 15., 10., 15., 5., 0.],
 [0., 3., 15., 2., 0., 11., 8., 0.],
 [0., 4., 12., 0., 0., 8., 8., 0.],
 [0., 5., 8., 0., 0., 9., 8., 0.],
 [0., 4., 11., 0., 1., 12., 7., 0.],
 [0., 2., 14., 5., 10., 12., 0., 0.],
 [0., 0., 6., 13., 10., 0., 0., 0.]])
```

Sklearn provides an estimator for the MLP. We can see one with one layer to start.

```
mlp = MLPClassifier(
 hidden_layer_sizes=(16),
 max_iter=500,
 alpha=1e-4,
 solver="lbfgs",
 verbose=10,
 random_state=1,
 learning_rate_init=0.1,
)
```

```
mlp.fit(X_train,y_train)
mlp.score(X_test,y_test)
```

RUNNING THE L-BFGS-B CODE

\* \* \*

```
Machine precision = 2.220D-16
N = 1210 M = 10
At X0 0 variables are exactly at the bounds
At iterate 0 f= 9.47554D+00 |proj g|= 7.33519D+00
At iterate 1 f= 7.92931D+00 |proj g|= 7.33017D+00
At iterate 2 f= 3.20247D+00 |proj g|= 1.98460D+00
```

At iterate	3	f= 2.36848D+00	proj g = 3.94993D-01
At iterate	4	f= 2.26928D+00	proj g = 2.31186D-01
At iterate	5	f= 2.11917D+00	proj g = 3.15914D-01
At iterate	6	f= 1.98921D+00	proj g = 3.21863D-01
At iterate	7	f= 1.72179D+00	proj g = 3.47733D-01
At iterate	8	f= 1.59843D+00	proj g = 4.09306D-01
At iterate	9	f= 1.51384D+00	proj g = 3.66170D-01
At iterate	10	f= 1.43619D+00	proj g = 6.53666D-01
At iterate	11	f= 1.28192D+00	proj g = 3.57314D-01
At iterate	12	f= 1.16492D+00	proj g = 2.80573D-01
At iterate	13	f= 1.09468D+00	proj g = 3.94633D-01
At iterate	14	f= 1.04140D+00	proj g = 1.76279D-01
At iterate	15	f= 9.83582D-01	proj g = 2.47615D-01
At iterate	16	f= 9.29940D-01	proj g = 3.11779D-01
At iterate	17	f= 8.41472D-01	proj g = 8.31291D-01
At iterate	18	f= 8.06873D-01	proj g = 4.51863D-01
At iterate	19	f= 7.54939D-01	proj g = 3.95567D-01
At iterate	20	f= 7.20614D-01	proj g = 1.87061D-01
At iterate	21	f= 6.82834D-01	proj g = 5.90170D-01
At iterate	22	f= 6.55423D-01	proj g = 1.79157D-01
At iterate	23	f= 6.41193D-01	proj g = 1.44769D-01
At iterate	24	f= 5.97437D-01	proj g = 4.12334D-01
At iterate	25	f= 5.72861D-01	proj g = 3.63440D-01
At iterate	26	f= 5.52706D-01	proj g = 1.49874D-01
At iterate	27	f= 5.23011D-01	proj g = 2.87436D-01
At iterate	28	f= 5.01585D-01	proj g = 2.52775D-01
At iterate	29	f= 4.76538D-01	proj g = 3.66908D-01
At iterate	30	f= 4.14723D-01	proj g = 2.95507D-01
At iterate	31	f= 3.65200D-01	proj g = 5.21213D-01
At iterate	32	f= 3.36424D-01	proj g = 1.76871D-01
At iterate	33	f= 3.20011D-01	proj g = 1.04968D-01
At iterate	34	f= 2.89998D-01	proj g = 4.37189D-01
At iterate	35	f= 2.76997D-01	proj g = 1.48235D-01
At iterate	36	f= 2.65409D-01	proj g = 1.59619D-01
At iterate	37	f= 2.52231D-01	proj g = 1.92672D-01
At iterate	38	f= 2.45755D-01	proj g = 3.26925D-01
At iterate	39	f= 2.28780D-01	proj g = 2.37536D-01
At iterate	40	f= 2.20868D-01	proj g = 1.77740D-01
At iterate	41	f= 2.13254D-01	proj g = 1.71852D-01
At iterate	42	f= 2.01624D-01	proj g = 1.86204D-01
At iterate	43	f= 1.92510D-01	proj g = 1.60988D-01
At iterate	44	f= 1.79026D-01	proj g = 1.02444D-01
At iterate	45	f= 1.70874D-01	proj g = 8.99446D-02

At iterate	46	f= 1.64194D-01	proj g = 9.72189D-02
At iterate	47	f= 1.50036D-01	proj g = 9.19580D-02
At iterate	48	f= 1.43597D-01	proj g = 1.09162D-01
At iterate	49	f= 1.33135D-01	proj g = 6.74089D-02
At iterate	50	f= 1.29634D-01	proj g = 6.00602D-02
At iterate	51	f= 1.24045D-01	proj g = 8.14586D-02
At iterate	52	f= 1.19464D-01	proj g = 9.33045D-02
At iterate	53	f= 1.14092D-01	proj g = 1.07134D-01
At iterate	54	f= 1.12061D-01	proj g = 1.68373D-01
At iterate	55	f= 1.07810D-01	proj g = 4.56517D-02
At iterate	56	f= 1.05812D-01	proj g = 3.84778D-02
At iterate	57	f= 1.01699D-01	proj g = 5.38138D-02
At iterate	58	f= 9.68784D-02	proj g = 7.48051D-02
At iterate	59	f= 9.13231D-02	proj g = 5.88384D-02
At iterate	60	f= 8.90751D-02	proj g = 6.45715D-02
At iterate	61	f= 8.60047D-02	proj g = 3.00412D-02
At iterate	62	f= 8.38610D-02	proj g = 3.08844D-02
At iterate	63	f= 8.01552D-02	proj g = 4.65718D-02
At iterate	64	f= 7.78359D-02	proj g = 6.56319D-02
At iterate	65	f= 7.27556D-02	proj g = 3.53741D-02
At iterate	66	f= 7.05975D-02	proj g = 2.23545D-02
At iterate	67	f= 6.89105D-02	proj g = 2.83771D-02
At iterate	68	f= 6.70830D-02	proj g = 2.89134D-02
At iterate	69	f= 6.46165D-02	proj g = 2.30697D-02
At iterate	70	f= 6.17405D-02	proj g = 3.82387D-02
At iterate	71	f= 5.99824D-02	proj g = 4.43660D-02
At iterate	72	f= 5.72498D-02	proj g = 5.94260D-02
At iterate	73	f= 5.47125D-02	proj g = 2.29015D-02
At iterate	74	f= 5.22934D-02	proj g = 2.03032D-02
At iterate	75	f= 5.02647D-02	proj g = 2.71974D-02
At iterate	76	f= 4.88819D-02	proj g = 2.21185D-02
At iterate	77	f= 4.77417D-02	proj g = 2.77729D-02
At iterate	78	f= 4.57025D-02	proj g = 4.62318D-02
At iterate	79	f= 4.52140D-02	proj g = 2.94958D-02
At iterate	80	f= 4.37762D-02	proj g = 1.74819D-02
At iterate	81	f= 4.28996D-02	proj g = 1.96283D-02

This problem is unconstrained.

At iterate	82	f= 4.19210D-02	proj g = 1.36587D-02
At iterate	83	f= 3.97302D-02	proj g = 2.41358D-02
At iterate	84	f= 3.93714D-02	proj g = 1.05456D-01
At iterate	85	f= 3.74295D-02	proj g = 2.77955D-02
At iterate	86	f= 3.63239D-02	proj g = 1.13597D-02
At iterate	87	f= 3.51322D-02	proj g = 2.95212D-02
At iterate	88	f= 3.37561D-02	proj g = 2.03094D-02
At iterate	89	f= 3.30676D-02	proj g = 2.54903D-02
At iterate	90	f= 3.22344D-02	proj g = 1.12178D-02
At iterate	91	f= 3.15172D-02	proj g = 1.56846D-02
At iterate	92	f= 3.06170D-02	proj g = 2.64344D-02
At iterate	93	f= 2.93450D-02	proj g = 2.51925D-02
At iterate	94	f= 2.83450D-02	proj g = 1.14953D-02
At iterate	95	f= 2.77793D-02	proj g = 1.14373D-02
At iterate	96	f= 2.65315D-02	proj g = 2.16130D-02
At iterate	97	f= 2.52443D-02	proj g = 2.00634D-02
At iterate	98	f= 2.41697D-02	proj g = 4.23867D-02
At iterate	99	f= 2.30569D-02	proj g = 2.68174D-02
At iterate	100	f= 2.18404D-02	proj g = 1.38388D-02
At iterate	101	f= 2.09592D-02	proj g = 1.57725D-02
At iterate	102	f= 2.00352D-02	proj g = 1.24923D-02
At iterate	103	f= 1.84950D-02	proj g = 1.12403D-02
At iterate	104	f= 1.73341D-02	proj g = 1.79188D-02
At iterate	105	f= 1.63665D-02	proj g = 1.82136D-02
At iterate	106	f= 1.56600D-02	proj g = 1.30069D-02
At iterate	107	f= 1.49936D-02	proj g = 1.37772D-02
At iterate	108	f= 1.43278D-02	proj g = 9.84300D-03
At iterate	109	f= 1.34371D-02	proj g = 9.96785D-03
At iterate	110	f= 1.26814D-02	proj g = 1.14418D-02
At iterate	111	f= 1.21720D-02	proj g = 8.93171D-03
At iterate	112	f= 1.17177D-02	proj g = 7.77785D-03
At iterate	113	f= 1.12636D-02	proj g = 9.62429D-03
At iterate	114	f= 1.08662D-02	proj g = 2.22119D-02
At iterate	115	f= 1.04985D-02	proj g = 3.41123D-02
At iterate	116	f= 9.71373D-03	proj g = 1.50002D-02
At iterate	117	f= 8.81026D-03	proj g = 1.68044D-02
At iterate	118	f= 8.08467D-03	proj g = 1.88993D-02
At iterate	119	f= 7.69817D-03	proj g = 7.13404D-03
At iterate	120	f= 7.51001D-03	proj g = 4.48318D-03
At iterate	121	f= 7.34166D-03	proj g = 4.65579D-03
At iterate	122	f= 7.08709D-03	proj g = 5.73124D-03
At iterate	123	f= 6.78207D-03	proj g = 4.94755D-03
At iterate	124	f= 6.46682D-03	proj g = 8.52138D-03
At iterate	125	f= 6.16375D-03	proj g = 3.87181D-03

At iterate 126	f= 5.91699D-03	proj g = 4.14600D-03
At iterate 127	f= 5.81678D-03	proj g = 7.05007D-03
At iterate 128	f= 5.70983D-03	proj g = 5.06206D-03
At iterate 129	f= 5.49156D-03	proj g = 6.16759D-03
At iterate 130	f= 5.35610D-03	proj g = 4.59600D-03
At iterate 131	f= 5.19132D-03	proj g = 7.42770D-03
At iterate 132	f= 4.95709D-03	proj g = 3.97121D-03
At iterate 133	f= 4.70936D-03	proj g = 3.17004D-03
At iterate 134	f= 4.56731D-03	proj g = 7.31589D-03
At iterate 135	f= 4.37064D-03	proj g = 6.68310D-03
At iterate 136	f= 4.04359D-03	proj g = 7.98850D-03
At iterate 137	f= 3.73281D-03	proj g = 6.72001D-03
At iterate 138	f= 3.53165D-03	proj g = 6.15807D-03
At iterate 139	f= 3.32106D-03	proj g = 7.75939D-03
At iterate 140	f= 2.90008D-03	proj g = 7.85534D-03
At iterate 141	f= 2.68699D-03	proj g = 9.24405D-03
At iterate 142	f= 2.22540D-03	proj g = 9.61545D-03
At iterate 143	f= 1.96770D-03	proj g = 5.83884D-03
At iterate 144	f= 1.82554D-03	proj g = 4.88459D-03
At iterate 145	f= 1.68438D-03	proj g = 7.21354D-03
At iterate 146	f= 1.50726D-03	proj g = 9.34121D-03
At iterate 147	f= 1.38335D-03	proj g = 2.68490D-03
At iterate 148	f= 1.29452D-03	proj g = 2.69141D-03
At iterate 149	f= 1.13358D-03	proj g = 2.44580D-03
At iterate 150	f= 1.03196D-03	proj g = 3.45458D-03
At iterate 151	f= 9.60751D-04	proj g = 4.12150D-03
At iterate 152	f= 8.77723D-04	proj g = 5.90000D-03
At iterate 153	f= 8.41470D-04	proj g = 2.31527D-03
At iterate 154	f= 8.08360D-04	proj g = 3.37485D-03
At iterate 155	f= 7.67414D-04	proj g = 2.03398D-03
At iterate 156	f= 7.34300D-04	proj g = 5.33148D-03
At iterate 157	f= 6.38833D-04	proj g = 1.70412D-03
At iterate 158	f= 6.12364D-04	proj g = 1.20828D-03
At iterate 159	f= 5.84667D-04	proj g = 2.24354D-03
At iterate 160	f= 5.48353D-04	proj g = 8.68824D-04
At iterate 161	f= 5.23253D-04	proj g = 8.20376D-04
At iterate 162	f= 4.76407D-04	proj g = 8.46503D-04
At iterate 163	f= 4.45754D-04	proj g = 2.67693D-03
At iterate 164	f= 4.01891D-04	proj g = 8.88978D-04
At iterate 165	f= 3.68106D-04	proj g = 6.91941D-04
At iterate 166	f= 3.43798D-04	proj g = 8.29751D-04
At iterate 167	f= 3.26771D-04	proj g = 1.10573D-03
At iterate 168	f= 3.07787D-04	proj g = 5.13844D-04

```

At iterate 169 f= 2.90516D-04 |proj g|= 4.17116D-04
At iterate 170 f= 2.69100D-04 |proj g|= 7.10983D-04
At iterate 171 f= 2.40278D-04 |proj g|= 8.70179D-04
At iterate 172 f= 2.24578D-04 |proj g|= 6.42199D-04
At iterate 173 f= 2.15504D-04 |proj g|= 3.43801D-04
At iterate 174 f= 1.98889D-04 |proj g|= 4.12353D-04
At iterate 175 f= 1.74294D-04 |proj g|= 5.04688D-04
At iterate 176 f= 1.54679D-04 |proj g|= 3.22614D-04
At iterate 177 f= 1.39023D-04 |proj g|= 2.36646D-04
At iterate 178 f= 1.33202D-04 |proj g|= 2.20948D-03
At iterate 179 f= 1.10663D-04 |proj g|= 3.62416D-04
At iterate 180 f= 1.07292D-04 |proj g|= 1.96128D-04
At iterate 181 f= 9.89917D-05 |proj g|= 1.25479D-04
At iterate 182 f= 9.12923D-05 |proj g|= 1.98654D-04
At iterate 183 f= 8.25346D-05 |proj g|= 4.98350D-04
At iterate 184 f= 7.77186D-05 |proj g|= 3.30357D-04
At iterate 185 f= 7.21364D-05 |proj g|= 1.10668D-04
At iterate 186 f= 7.02495D-05 |proj g|= 8.08394D-05

* * *

Tit = total number of iterations
Tnf = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip = number of BFGS updates skipped
Nact = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F = final function value

* * *

 N Tit Tnf Tnint Skip Nact Projg F
1210 186 198 1 0 0 8.084D-05 7.025D-05
 F = 7.024949673231059E-005

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL

```

0.92

We can also see what happens if we increase the size of the hidden layer.

```

mlp = MLPClassifier(
 hidden_layer_sizes=(64),
 max_iter=500,
 alpha=1e-4,
 solver="lbfgs",
 verbose=10,
 random_state=1,
 learning_rate_init=0.1,
)

```

```

mlp.fit(X_train,y_train)
mlp.score(X_test,y_test)

```

## RUNNING THE L-BFGS-B CODE

\* \* \*

```
Machine precision = 2.220D-16
N = 4810 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 1.05395D+01 |proj g|= 8.40448D+00
At iterate 1 f= 9.55652D+00 |proj g|= 4.22276D+00
At iterate 2 f= 8.37333D+00 |proj g|= 4.72378D+00
At iterate 3 f= 6.76492D+00 |proj g|= 2.80587D+00
At iterate 4 f= 4.90315D+00 |proj g|= 2.52016D+00
At iterate 5 f= 3.54400D+00 |proj g|= 2.26026D+00
At iterate 6 f= 2.35942D+00 |proj g|= 1.12600D+00
At iterate 7 f= 1.78162D+00 |proj g|= 9.59480D-01
At iterate 8 f= 1.33318D+00 |proj g|= 1.91270D+00
At iterate 9 f= 9.57028D-01 |proj g|= 5.19014D-01
At iterate 10 f= 7.33001D-01 |proj g|= 2.85575D-01
At iterate 11 f= 5.78415D-01 |proj g|= 2.27551D-01
At iterate 12 f= 4.38653D-01 |proj g|= 2.11513D-01
At iterate 13 f= 2.99840D-01 |proj g|= 1.59860D-01
At iterate 14 f= 2.41473D-01 |proj g|= 9.44723D-02
At iterate 15 f= 1.95764D-01 |proj g|= 6.24662D-02
At iterate 16 f= 1.76333D-01 |proj g|= 3.36412D-01
At iterate 17 f= 1.44582D-01 |proj g|= 7.16637D-02
At iterate 18 f= 1.32827D-01 |proj g|= 5.17305D-02
At iterate 19 f= 1.16554D-01 |proj g|= 6.67419D-02
At iterate 20 f= 9.52489D-02 |proj g|= 5.77022D-02
At iterate 21 f= 8.51024D-02 |proj g|= 8.33721D-02
At iterate 22 f= 7.11660D-02 |proj g|= 2.77687D-02
At iterate 23 f= 6.06905D-02 |proj g|= 2.31052D-02
At iterate 24 f= 5.43634D-02 |proj g|= 6.89654D-02
At iterate 25 f= 4.65455D-02 |proj g|= 3.51934D-02
At iterate 26 f= 4.16877D-02 |proj g|= 2.52962D-02
At iterate 27 f= 3.52003D-02 |proj g|= 2.36364D-02
At iterate 28 f= 2.92648D-02 |proj g|= 2.18708D-02
```

```

At iterate 29 f= 2.52708D-02 |proj g|= 7.40319D-02
At iterate 30 f= 2.03338D-02 |proj g|= 1.61263D-02
At iterate 31 f= 1.87875D-02 |proj g|= 1.50281D-02
At iterate 32 f= 1.39342D-02 |proj g|= 2.48641D-02
At iterate 33 f= 1.24512D-02 |proj g|= 2.41981D-02
At iterate 34 f= 1.13949D-02 |proj g|= 1.65862D-02
At iterate 35 f= 1.01692D-02 |proj g|= 9.82169D-03
At iterate 36 f= 9.05706D-03 |proj g|= 1.24135D-02
At iterate 37 f= 7.29985D-03 |proj g|= 1.62075D-02
At iterate 38 f= 5.55858D-03 |proj g|= 1.05728D-02
At iterate 39 f= 4.68727D-03 |proj g|= 4.54450D-03
At iterate 40 f= 3.69357D-03 |proj g|= 5.88379D-03
At iterate 41 f= 2.88362D-03 |proj g|= 6.26167D-03
At iterate 42 f= 2.05455D-03 |proj g|= 6.16726D-03
At iterate 43 f= 1.69602D-03 |proj g|= 2.82139D-03

```

This problem is unconstrained.

```

At iterate 44 f= 1.26304D-03 |proj g|= 3.08991D-03
At iterate 45 f= 9.62334D-04 |proj g|= 3.07127D-03
At iterate 46 f= 7.26593D-04 |proj g|= 1.79304D-03
At iterate 47 f= 5.43571D-04 |proj g|= 1.31426D-03
At iterate 48 f= 3.80379D-04 |proj g|= 1.37833D-03
At iterate 49 f= 2.64290D-04 |proj g|= 6.37364D-04
At iterate 50 f= 2.04633D-04 |proj g|= 4.18778D-04
At iterate 51 f= 1.72472D-04 |proj g|= 1.96083D-03
At iterate 52 f= 1.09313D-04 |proj g|= 6.70507D-04
At iterate 53 f= 9.14439D-05 |proj g|= 4.30659D-04
At iterate 54 f= 6.70092D-05 |proj g|= 1.29468D-04
At iterate 55 f= 4.79595D-05 |proj g|= 1.12761D-04
At iterate 56 f= 4.02545D-05 |proj g|= 3.21691D-04
At iterate 57 f= 2.82140D-05 |proj g|= 1.52558D-04
At iterate 58 f= 2.15135D-05 |proj g|= 7.78492D-05

```

\* \* \*

```

Tit = total number of iterations
Tnf = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip = number of BFGS updates skipped
Nact = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F = final function value

```

\* \* \*

N	Tit	Tnf	Tnint	Skip	Nact	Projg	F
4810	58	63	1	0	0	7.785D-05	2.151D-05
F = 2.1513514351945298E-005							

CONVERGENCE: NORM\_OF\_PROJECTED\_GRADIENT\_<= PGtol

0.9844444444444445

We can compare it to SVM:

```
svm_clf = svm.SVC(gamma=0.001)
svm_clf.fit(X_train, y_train)
svm_clf.score(X_test,y_test)
```

```
0.9955555555555555
```

We can also have multiple hidden layers:

```
mlp = MLPClassifier(
 hidden_layer_sizes=(64,64),
 max_iter=500,
 alpha=1e-4,
 solver="lbfgs",
 verbose=10,
 random_state=1,
 learning_rate_init=0.1,
)
mlp.fit(X_train,y_train)
mlp.score(X_test,y_test)
```

## RUNNING THE L-BFGS-B CODE

\* \* \*

```
Machine precision = 2.220D-16
N = 8970 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.73848D+00 |proj g|= 5.37953D+00
At iterate 1 f= 6.53825D+00 |proj g|= 5.36787D+00
At iterate 2 f= 4.47669D+00 |proj g|= 2.11448D+00
At iterate 3 f= 3.26563D+00 |proj g|= 1.30852D+00
At iterate 4 f= 2.53774D+00 |proj g|= 1.06961D+00
At iterate 5 f= 1.84001D+00 |proj g|= 7.59070D-01
At iterate 6 f= 1.41014D+00 |proj g|= 4.83064D-01
At iterate 7 f= 1.03517D+00 |proj g|= 3.41293D-01
At iterate 8 f= 7.68562D-01 |proj g|= 2.84265D-01
At iterate 9 f= 5.57535D-01 |proj g|= 3.19787D-01
At iterate 10 f= 4.22471D-01 |proj g|= 2.25469D-01
At iterate 11 f= 3.48625D-01 |proj g|= 2.04539D-01
At iterate 12 f= 2.89147D-01 |proj g|= 1.88931D-01
At iterate 13 f= 2.56977D-01 |proj g|= 2.22067D-01
At iterate 14 f= 2.23300D-01 |proj g|= 9.89751D-02
At iterate 15 f= 1.87220D-01 |proj g|= 1.11342D-01
At iterate 16 f= 1.61657D-01 |proj g|= 2.46659D-01
At iterate 17 f= 1.36384D-01 |proj g|= 5.35463D-02
At iterate 18 f= 1.23409D-01 |proj g|= 4.65261D-02
At iterate 19 f= 9.95937D-02 |proj g|= 6.38887D-02
At iterate 20 f= 8.47673D-02 |proj g|= 7.59234D-02
At iterate 21 f= 7.50660D-02 |proj g|= 4.71787D-02
At iterate 22 f= 6.81760D-02 |proj g|= 4.23555D-02
At iterate 23 f= 5.34497D-02 |proj g|= 6.61903D-02
At iterate 24 f= 4.63148D-02 |proj g|= 7.34516D-02
At iterate 25 f= 3.97646D-02 |proj g|= 3.55219D-02
At iterate 26 f= 3.41041D-02 |proj g|= 2.51034D-02
At iterate 27 f= 2.94909D-02 |proj g|= 2.36769D-02
At iterate 28 f= 2.35213D-02 |proj g|= 2.64504D-02
```

This problem is unconstrained.

```

At iterate 29 f= 2.09838D-02 |proj g|= 5.21924D-02
At iterate 30 f= 1.67480D-02 |proj g|= 2.39733D-02
At iterate 31 f= 1.38224D-02 |proj g|= 2.21329D-02
At iterate 32 f= 1.20977D-02 |proj g|= 2.22472D-02
At iterate 33 f= 1.09785D-02 |proj g|= 1.68300D-02
At iterate 34 f= 9.92350D-03 |proj g|= 1.67730D-02
At iterate 35 f= 7.74261D-03 |proj g|= 1.85545D-02
At iterate 36 f= 6.73298D-03 |proj g|= 3.20351D-02
At iterate 37 f= 5.69798D-03 |proj g|= 9.68910D-03
At iterate 38 f= 4.74800D-03 |proj g|= 5.57582D-03
At iterate 39 f= 4.01092D-03 |proj g|= 1.33248D-02
At iterate 40 f= 3.58887D-03 |proj g|= 5.84670D-03
At iterate 41 f= 3.23361D-03 |proj g|= 3.61209D-03
At iterate 42 f= 2.51771D-03 |proj g|= 5.49044D-03
At iterate 43 f= 1.99784D-03 |proj g|= 1.27486D-02
At iterate 44 f= 1.46561D-03 |proj g|= 3.99404D-03
At iterate 45 f= 1.15025D-03 |proj g|= 2.31925D-03
At iterate 46 f= 8.94320D-04 |proj g|= 1.79783D-03
At iterate 47 f= 6.16356D-04 |proj g|= 4.92093D-03
At iterate 48 f= 4.15271D-04 |proj g|= 1.87738D-03
At iterate 49 f= 3.59546D-04 |proj g|= 1.10006D-03
At iterate 50 f= 2.47733D-04 |proj g|= 6.82931D-04
At iterate 51 f= 1.66948D-04 |proj g|= 1.16699D-03
At iterate 52 f= 1.54703D-04 |proj g|= 4.54335D-03
At iterate 53 f= 7.79137D-05 |proj g|= 8.88699D-04
At iterate 54 f= 6.90817D-05 |proj g|= 5.24030D-04
At iterate 55 f= 5.57325D-05 |proj g|= 2.01207D-04
At iterate 56 f= 4.50666D-05 |proj g|= 2.11565D-04
At iterate 57 f= 3.37098D-05 |proj g|= 1.92841D-04
At iterate 58 f= 2.38456D-05 |proj g|= 1.63885D-04

```

```

At iterate 59 f= 1.80926D-05 |proj g|= 1.08797D-04
At iterate 60 f= 1.43565D-05 |proj g|= 5.69150D-05

```

\* \* \*

```

Tit = total number of iterations
Tnf = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip = number of BFGS updates skipped
Nact = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F = final function value

```

\* \* \*

N	Tit	Tnf	Tnint	Skip	Nact	Projg	F
8970	60	64	1	0	0	5.691D-05	1.436D-05
F =	1.4356519896978652E-005						

CONVERGENCE: NORM\_OF\_PROJECTED\_GRADIENT\_<=\_PGTOL

We saw that the SVM performed a bit better, but this is a simple problem. We can also compare these based on much they store, the number of parameters is related to the complexity.

```
svm_clf.support_vectors_.shape
```

```
(680, 64)
```

```
[c.shape for c in mlp.coefs_]
```

```
[(64, 64), (64, 64), (64, 10)]
```

```
np.prod(list(svm_clf.support_vectors_.shape))
```

```
43520
```

```
np.sum([np.prod(list(c.shape)) for c in mlp.coefs_])
```

```
8832
```

We see this is much smaller.

## 37.1. Questions after class

### 37.1.1. How can we use this in our assignment?

You do not have to, but you could try an MLP in your assignment, but all that is required is any classifier and a text representation.

### 37.1.2. How do we know how to change the parameters?

If it doesn't work well, trying more layers or bigger layers is a good idea.

Neural nets work like a black box, they're hard to interpret, so while there are good heuristics, there isn't as solid theory for how to know what to do with them.

They work well, but because they're hard to understand, that's a risk of using them.

## 38. Predicting with Neural Networks

```
from scipy.special import expit
from sklearn.datasets import make_classification
from sklearn.neural_network import MLPClassifier

from sklearn import svm
import pandas as pd
import numpy as np
import sklearn

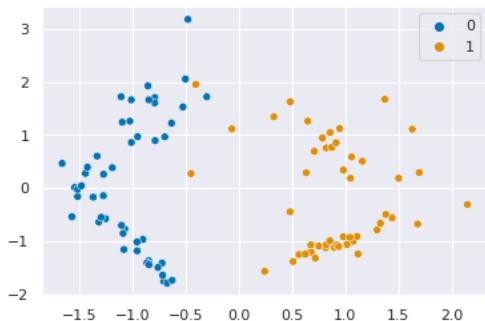
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.model_selection import train_test_split

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

Today, we're going to use very simple data in order to examine how a neural network works.

```
X, y = make_classification(n_samples=100, random_state=1, n_features=2, n_redundant=0)
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, stratify=y,
random_state=1)
sns.scatterplot(x=X[:,0], y=X[:,1], hue=y)
```

<AxesSubplot:>



First, we'll train and score a tiny neural net: with 1 hidden layer of 1 neuron.

```
clf = MLPClassifier(
 hidden_layer_sizes=(1), # 1 hidden layer, 1 artificial neuron
 max_iter=100, # maximum 100 iterations in optimization
 alpha=1e-4, # regularization
 solver="lbfgs", #optimization algorithm
 verbose=10, # how much detail to print
 activation= 'identity' # how to transform the hidden layer before passing it to the
 next layer
)
clf.fit(X_train, y_train)

clf.score(X_test, y_test)
```

### 💡 Tip

This is actually equivalent to another classifier, called logistic Regression

### ⓘ Correction

I've removed the parameter `learning_rate_init=0.1` and `random_state=1` because sklearn actually only uses the learning rate and randomizaion for some of the optimization algorithms: adam and sgd, which we are not using.

These algorithms are good in high dimensional problems, our problem is simple, so we don't need the advanced parameters or algorithms.

## RUNNING THE L-BFGS-B CODE

```
* * *

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 4.23332D-01 |proj g|= 3.14200D-01
At iterate 1 f= 1.46977D-01 |proj g|= 8.71647D-02
At iterate 2 f= 8.77685D-02 |proj g|= 4.22158D-02
At iterate 3 f= 6.67777D-02 |proj g|= 2.09278D-02
At iterate 4 f= 5.77858D-02 |proj g|= 1.76798D-02
At iterate 5 f= 5.23822D-02 |proj g|= 1.18999D-02
At iterate 6 f= 4.97929D-02 |proj g|= 8.35703D-03
At iterate 7 f= 4.88585D-02 |proj g|= 8.74979D-03
At iterate 8 f= 4.79681D-02 |proj g|= 5.86741D-03
At iterate 9 f= 4.79108D-02 |proj g|= 1.52605D-03
```

This problem is unconstrained.

1.0

```
At iterate 10 f= 4.79036D-02 |proj g|= 1.38911D-04
At iterate 11 f= 4.79035D-02 |proj g|= 1.43740D-05
```

\* \* \*

```
Tit = total number of iterations
Tnf = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip = number of BFGS updates skipped
Nact = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F = final function value
```

\* \* \*

N	Tit	Tnf	Tnint	Skip	Nact	Projg	F
5	11	12	1	0	0	1.437D-05	4.790D-02
F =	4.7903482810364120E-002						

CONVERGENCE: NORM\_OF\_PROJECTED\_GRADIENT\_<=\_PGTOL

Now we can see that it actually has another activation, that we didn't change the output layer still has a logistic activation layer, which we want. If we didn't then the output layer wouldn't be able to be interpreted as a probability, because probability always needs to be between 0 and 1.

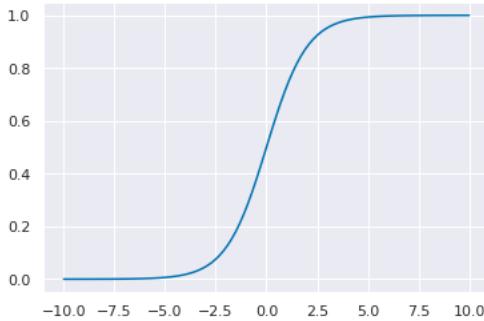
clf.out\_activation\_

'logistic'

The logistic function looks like this:

```
x_logistic = np.linspace(-10,10,100)
y_logistic = expit(x_logistic)
plt.plot(x_logistic,y_logistic)
```

```
[<matplotlib.lines.Line2D at 0x7f51e670b790>]
```



The fit method learned the following weights:

```
clf.coefs_
```

```
[array([[-3.84094322],
 [0.1105853]]),
 array([[[-3.42073886]]])
```

and biases

```
clf.intercepts_
```

```
[array([-1.43859861]), array([0.77370254])]
```

These are called coefficients and intercepts because the weights are multiplied by the inputs and the biases you can interpret as geometrically as shifting things, like a line intercept (recall  $y=mx+b$ )

### 38.1. Reconstructing the Predict method

we'll use an acutally new point, we can make one up

```
type([[-1, 2]])
```

```
list
```

we want a numpy array so we will cast it

```
pt = np.array([[-1, 2]])
```

```
type(pt)
```

```
numpy.ndarray
```

numpy's `matmul` does matrix multiplicaion (multiply columns by rows element wise and sum)

$\text{f}(x) = W_2g(W_1^T x + b_1) + b_2$

the  $g()$  is the activation function, which we set to identity  $(g(x) = x)$  so we don't have to do more

```
np.matmul(pt,clf.coefs_[0]) + clf.intercepts_[0]*clf.coefs_[1] + clf.intercepts_[1])
```

```
File "/tmp/ipykernel_2647/81862389.py", line 1
 np.matmul(pt,clf.coefs_[0]) + clf.intercepts_[0])*clf.coefs_[1] + clf.intercepts_[1])
 ^
```

```
SyntaxError: invalid syntax
```

but we're not quite done, the output layer still transforms using the logistic function, which is also known as [expit](#) and we have imported from scipy.

```
expit((np.matmul(pt,clf.coefs_[0]) + clf.intercepts_[0])*clf.coefs_[1] +
clf.intercepts_[1])
```

```
array([[0.0002744]])
```

We can compare this to the classifier's output. It outputs a probability for each class, we only computed the probability of the 1 class.

```
clf.predict_proba(pt)
```

```
array([[9.99725602e-01, 2.74397622e-04]])
```

and we can see how it predicts on that point.

```
clf.predict(pt)
```

```
array([0])
```

A single artificial neuron like the function below. where it has parameters that have to be determined before we can use it on an input vector.

```
def aritificial_neuron_template(activation,weights,bias,inputs):
 """
 simple artificial neuron

 Parameters

 activation : function
 activation function of the neuron
 weights : numpy array
 weights for summing inputs
 bias: numpy array
 bias term added to the weighted sum
 inputs : numpy array
 input to the neuron

 """
 return activation(np.matmul(inputs,weights) +bias)

two common activation functions
identity_activation = lambda x: x
logistic_activation = lambda x: expit(x)
```

When we instantiate the multilayer perceptron object, `MLPClassifier`, we pick the activation function and when we give data to the `fit` method, we get the weights and biases.

A neural network passes the data to the hidden layer, and the output of the hidden layer to the output layer. In our neural network, we have just one neuron at each layer.

So the `predict_proba` method is the same as the following:

```
aritificial_neuron_template(logistic_activation,clf.coefs_[1],clf.intercepts_[1],
aritificial_neuron_template(identity_activation,clf.coefs_[0],clf.intercepts_[0],
clf.intercepts_[0],pt))
```

```
array([[0.0002744]])
```

To make this easier to read, we can make the intermediate neurons their own lambda functions.

```
hidden_neuron = lambda x:
aritificial_neuron_template(identity_activation,clf.coefs_[0],clf.intercepts_[0],x)
output_neuron = lambda x:
aritificial_neuron_template(expit,clf.coefs_[1],clf.intercepts_[1],x)
output_neuron(hidden_neuron(pt))
```

```
array([[0.0002744]])
```

We can confirm that this works the same as the predict probability method:

```
clf.predict_proba(pt)
```

```
array([[9.99725602e-01, 2.74397622e-04]])
```

## 38.2. More Features and More Hidden Neurons

First, we'll sample more features and then train a new classifier

```
X, y = make_classification(n_samples=100, random_state=1, n_features=4, n_redundant=0)
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
 random_state=5)
pt_4d = np.asarray([[-1, -2, 2, -1], [1.5, 0, .5, 1]])
clf_4d = MLPClassifier(
 hidden_layer_sizes=(1),
 max_iter=5000,
 alpha=1e-4,
 solver="lbfgs",
 verbose=10,
 activation='identity'
)
clf_4d.fit(X_train, y_train)

clf_4d.score(X_test, y_test)
```

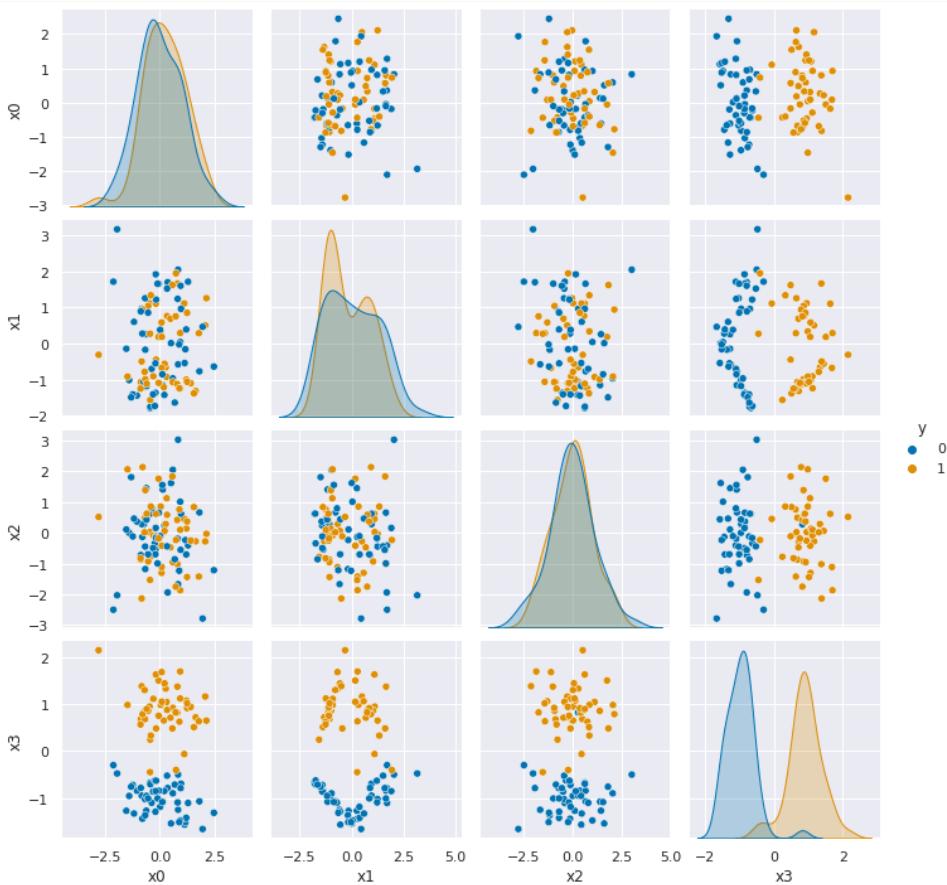
```
NameError Traceback (most recent call last)
/tmp/ipykernel_2647/3721524382.py in <module>
 1 X, y = make_classification(n_samples=100,
 2 random_state=1, n_features=4, n_redundant=0)
----> 3 X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
 4 random_state=5)
 5 pt_4d = np.asarray([[-1, -2, 2, -1], [1.5, 0, .5, 1]])
 6 clf_4d = MLPClassifier()

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

We can look at this data

```
df = pd.DataFrame(X, columns=['x0', 'x1', 'x2', 'x3'])
df['y'] = y
sns.pairplot(df, hue='y')
```

```
<seaborn.axisgrid.PairGrid at 0x7f51e91710d0>
```



and based on this, we'll pick a new pair of points to test on:

```
pt_4d = np.asarray([[-2, 2, 2, -2], [1.5, 0, -1, 3]])
```

This neural network is just like the one before:

```
hidden_neuron_4d = lambda x: artificial_neuron_template(identity_activation,
clf_4d.coefs_[0],clf_4d.intercepts_[0],x)
output_neuron_4d = lambda x: artificial_neuron_template(logistic_activation,
clf_4d.coefs_[1],clf_4d.intercepts_[1],x)

output_neuron_4d(hidden_neuron_4d(pt_4d))
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2647/2138475403.py in <module>
 5
 6
----> 7 output_neuron_4d(hidden_neuron_4d(pt_4d))

/tmp/ipykernel_2647/2138475403.py in <lambda>(x)
 1 hidden_neuron_4d = lambda x: artificial_neuron_template(identity_activation,
----> 2 clf_4d.coefs_[0],clf_4d.intercepts_[0],x)
 3 output_neuron_4d = lambda x: artificial_neuron_template(logistic_activation,
 4
clf_4d.coefs_[1],clf_4d.intercepts_[1],x)
 5

NameError: name 'clf_4d' is not defined
```

```
clf_4d.predict_proba(pt_4d)
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2647/898774351.py in <module>
----> 1 clf_4d.predict_proba(pt_4d)

NameError: name 'clf_4d' is not defined
```

However, remember this one was not as accurate:

```
clf_4d.score(X_test, y_test)
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2647/1039820928.py in <module>
----> 1 clf_4d.score(X_test, y_test)

NameError: name 'clf_4d' is not defined
```

To try importing it, we will add more layers and a different activation function:

```
clf_4d_4h = MLPClassifier(
 hidden_layer_sizes=(4),
 max_iter=500,
 alpha=1e-4,
 solver="lbfgs",
 verbose=10,
 activation='logistic'
)
clf_4d_4h.fit(X_train, y_train)

clf_4d_4h.score(X_test, y_test)
```

## RUNNING THE L-BFGS-B CODE

\* \* \*

```
Machine precision = 2.220D-16
N = 17 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.06231D-01 |proj g|= 6.70642D-02
At iterate 1 f= 6.36052D-01 |proj g|= 1.36767D-01
At iterate 2 f= 3.08528D-01 |proj g|= 9.74622D-02
At iterate 3 f= 1.08324D-01 |proj g|= 3.44220D-02
At iterate 4 f= 7.90743D-02 |proj g|= 3.20636D-02
At iterate 5 f= 7.09305D-02 |proj g|= 9.59120D-03
At iterate 6 f= 6.82623D-02 |proj g|= 8.73319D-03
At iterate 7 f= 5.63789D-02 |proj g|= 4.17658D-03
At iterate 8 f= 5.26999D-02 |proj g|= 1.01417D-02
At iterate 9 f= 5.01675D-02 |proj g|= 5.84425D-03
At iterate 10 f= 4.88128D-02 |proj g|= 3.54815D-03
At iterate 11 f= 4.75515D-02 |proj g|= 5.12976D-03
At iterate 12 f= 4.69894D-02 |proj g|= 2.09150D-03
At iterate 13 f= 4.66031D-02 |proj g|= 2.54255D-03
At iterate 14 f= 3.75869D-02 |proj g|= 1.02502D-02
At iterate 15 f= 3.26356D-02 |proj g|= 1.21006D-02
At iterate 16 f= 2.73451D-02 |proj g|= 2.70078D-03
At iterate 17 f= 2.71765D-02 |proj g|= 2.13841D-03
At iterate 18 f= 2.68613D-02 |proj g|= 1.75365D-03
At iterate 19 f= 2.65689D-02 |proj g|= 6.16861D-04
At iterate 20 f= 2.63870D-02 |proj g|= 6.64209D-04
At iterate 21 f= 2.61667D-02 |proj g|= 1.90172D-04
At iterate 22 f= 2.61453D-02 |proj g|= 1.69874D-04
At iterate 23 f= 2.60935D-02 |proj g|= 1.79784D-04
At iterate 24 f= 2.59629D-02 |proj g|= 1.12488D-03
At iterate 25 f= 2.55827D-02 |proj g|= 1.39081D-03
At iterate 26 f= 2.43911D-02 |proj g|= 1.63124D-03
At iterate 27 f= 2.36313D-02 |proj g|= 1.91681D-03
At iterate 28 f= 2.35883D-02 |proj g|= 1.87747D-03
At iterate 29 f= 2.33350D-02 |proj g|= 1.60467D-03
At iterate 30 f= 2.29051D-02 |proj g|= 1.00197D-03
At iterate 31 f= 2.27255D-02 |proj g|= 9.77243D-04
At iterate 32 f= 2.24862D-02 |proj g|= 2.76035D-03
At iterate 33 f= 2.22322D-02 |proj g|= 1.54221D-03
At iterate 34 f= 2.20493D-02 |proj g|= 5.86099D-04
At iterate 35 f= 2.20145D-02 |proj g|= 3.90243D-04
At iterate 36 f= 2.19711D-02 |proj g|= 3.56517D-04
```

This problem is unconstrained.

0.96

```

At iterate 37 f= 2.19611D-02 |proj g|= 1.64629D-04
At iterate 38 f= 2.19592D-02 |proj g|= 1.74599D-04
At iterate 39 f= 2.19524D-02 |proj g|= 6.23788D-05

* * *

Tit = total number of iterations
Tnf = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip = number of BFGS updates skipped
Nact = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F = final function value

* * *

 N Tit Tnf Tnint Skip Nact Projg F
 17 39 55 1 0 0 6.238D-05 2.195D-02
 F = 2.1952412525950062E-002

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL

```

we see some improvement.

This network is more complicated. It has 5 total neurons:

```
hidden_neuron_4d_h0 = lambda x: aritificial_neuron_template(logistic_activation,
 clf_4d_4h.coefs_[0]
[::,0],clf_4d_4h.intercepts_[0][0],x)
hidden_neuron_4d_h1 = lambda x: aritificial_neuron_template(logistic_activation,
 clf_4d_4h.coefs_[0]
[::,1],clf_4d_4h.intercepts_[0][1],x)
hidden_neuron_4d_h2 = lambda x: aritificial_neuron_template(logistic_activation,
 clf_4d_4h.coefs_[0]
[::,2],clf_4d_4h.intercepts_[0][2],x)
hidden_neuron_4d_h3 = lambda x: aritificial_neuron_template(logistic_activation,
 clf_4d_4h.coefs_[0]
[::,3],clf_4d_4h.intercepts_[0][3],x)
output_neuron_4d_4h = lambda x: aritificial_neuron_template(logistic_activation,
 clf_4d_4h.coefs_[1],clf_4d_4h.intercepts_[1],x)
```

And we have to take the output of all 4 hidden neurons into the output neuron, because they are a single layer, not in sequence.

```

ValueError Traceback (most recent call last)
/tmp/ipykernel_2647/4049144657.py in <module>
----> 1 output_neuron_4d_4h(np.asarray([hidden_neuron_4d_h0(pt_4d),
 2 hidden_neuron_4d_h1(pt_4d),
 3 hidden_neuron_4d_h2(pt_4d),
 4 hidden_neuron_4d_h3(pt_4d)]).T)

/tmp/ipykernel_2647/2777926706.py in <lambda>(x)
 1 hidden_neuron_4d_h0 = lambda x: aritificial_neuron_template(logistic_activation,
----> 2 clf_4d_4h.coefs_[0]
[::,0],clf_4d_4h.intercepts_[0][0],x)
 3 hidden_neuron_4d_h1 = lambda x: aritificial_neuron_template(logistic_activation,
----> 4 clf_4d_4h.coefs_[0]
[::,1],clf_4d_4h.intercepts_[0][1],x)
 5 hidden_neuron_4d_h2 = lambda x: aritificial_neuron_template(logistic_activation,

/tmp/ipykernel_2647/1765182718.py in aritificial_neuron_template(activation, weights,
bias, inputs)
 15
 16 ''
---> 17 return activation(np.matmul(inputs,weights) +bias)
 18
 19 # two common activation functions

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc
signature (n?,k),(k,m?)->(n?,m?) (size 2 is different from 4)

```

And again, we see this is the probability of predicting 1:

```
clf_4d_4h.predict_proba(pt_4d)
```

```

ValueError Traceback (most recent call last)
/tmp/ipykernel_2647/3474441194.py in <module>
----> 1 clf_4d_4h.predict_proba(pt_4d)

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/sklearn/neural_network/_multilayer_perceptron.py in predict_proba(self, X)
 1241 """
 1242 check_is_fitted(self)
-> 1243 y_pred = self._forward_pass_fast(X)
 1244
 1245 if self.n_outputs_ == 1:

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-
packages/sklearn/neural_network/_multilayer_perceptron.py in _forward_pass_fast(self, X)
 157 """The decision function of the samples for each class in the model.
 158 """
--> 159 X = self._validate_data(X, accept_sparse=["csr", "csc"], reset=False)
 160
 161 # Initialize first layer

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/sklearn/base.py in
_validate_data(self, X, y, reset, validate_separately, **check_params)
 578
 579 if not no_val_X and check_params.get("ensure_2d", True):
--> 580 self._check_n_features(X, reset=reset)
 581
 582 return out

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/sklearn/base.py in
_check_n_features(self, X, reset)
 394 if n_features != self.n_features_in_:
 395 raise ValueError(
--> 396 f"X has {n_features} features, but {self.__class__.__name__} "
 397 f"is expecting {self.n_features_in_} features as input."
 398)

ValueError: X has 4 features, but MLPClassifier is expecting 2 features as input.

```

### 38.3. (optional) What is a numerical optimization algorithm?

Numerical Optimization algorithms are at the core of many of the fit methods.

One way we can optimize a function is to take the derivative, set it equal to zero and solve for the parameter. If we know the function is convex (like a bowl or valley shape) then the place where the derivative (slope) is 0 is the bottom or lowest point of the valley.

Numerical optimization is for when we can't analytically solve that problem once we set it equal to zero.

Optimization algorithms are sort of like search algorithms but can work in high dimensions and use strategy based on calculus.

The basic idea in many numerical optimization algorithms is to start at a point (initial setting of the coefficients in this case) and then compute the value of the function then change the coefficients a little and compute again. We can use those two points to see if the direction we "moved" or the way we changed the parameters made it better or worse. If it was better, we change them more in the same direction, (if we made both smaller then we make them both smaller again) if it got worse, we change in a different direction.

You can think of this like trying to find the bottom of a valley, without being able to see, just check your altitude. You take a step left, right, forward or back and then see if your altitude went up or down.

LBFGS actually uses the derivative, so it's like you can see the direction of the hill you're on, but you have to keep taking steps and then if you reach a point where you can't go down anymore you know you are done. When the algorithm finds it can't get better, that's called convergence.

Stochastic gradient descent works in high dimensions where it's too hard to do the derivative, but you can randomly move in different directions (or take the partial derivative in a small number of definitions). Adam is a special version of that with better strategy.

Numerical optimization is a whole research area. In graduate school, I took a whole semester long course just learning different algorithms for this.

## 39. Review, IDEA, & Preparing for Deep Learning

### 39.1. Review for Level 1

Remember that to earn a C or better (or for your level 2 achievements to influence) your grade at all you have to earn all of the level 1s.

Review the [Achievement Definitions](#) to be sure you know what we are looking for.

### 39.2. Representation

We change the representation of data when it is not tabular. As we've seen with text and images, we have to transform the data into a tabular form for the algorithms we have seen so far. Deep learning combines learning a representation with learning a prediction rule.

### 39.3. Matching Tools to Steps in the ML Pipeline

pandas	loading data, cleaning data, summarizing
seaborn	visualization
BeautifulSoup	webscraping
sklearn	fitting models, evaluating models, transforming data for modeling

### 39.4. Deep Learning Ecosystem

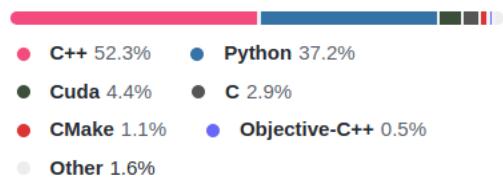
While `sklearn` can do basic neural networks, it doesn't have many types of neurons or fancy layers.

What makes deep learning work is its ability to transform data in new ways. We will see them in more detail.

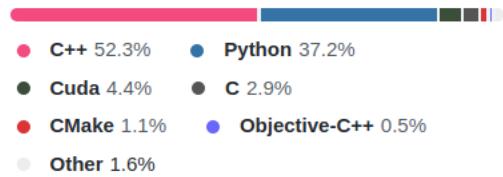
Two popular Deep Learning Libraries are [Pytorch \(code\)](#) and [TensorFlow \(code\)](#), both are open source. These are both complex, high performance libraries with optimized code. We have used Python in class because it is a user-friendly programming language, but it is not optimal, performance-wise, so much of the code in these

libraries is in C++.

## Languages



## Languages



[Keras](#) is a high level library written on top of tensorflow in python. The goal of this library is to enable fast experimentation. You can think of the relationship between [tensorflow](#) and [keras](#) like the relationship between [matplotlib](#) and [seaborn](#).

In [seaborn](#) we got high level plot figures with high level parameters like using variables in our dataset to create rows and columns of subplots. In [matplotlib](#), we would have to separately create the subplot grid, then break the data apart, and assign a plot using a subset of the data for each subplot. However, under the hood, [seaborn](#) uses [matplotlib](#). Just as we have mostly used seaborn, but occasionally use matplotlib to tweak things, we will use [keras](#).

Deep learning is computationally expensive, which is why the code is so optimized. However this means installing deep learning libraries is harder. They require more dependencies than we have worked with to date. Also, the code is optimized in order to work on high performance hardware: GPUs and other parallel computing systems, which most laptops do not have. We will use [google Colab](#) to do deep learning in class.

### ⚠ Warning

There is a whole semester long course on just Deep Learning (CSC541). We will not cover everything; the goal is just enough that you know where to start.

If you want to use deep learning after this class alone, you'll need to study substantially on your own, but past students have taken this foundation and made it work.

### ℹ Note

Note however that the relationship here is somewhat different, as keras is actually a submodule of tensorflow. Technically, the relationship is probably closer to [matplotlib](#) as a whole to [matplotlib.pyplot](#), but we have not used base [matplotlib](#) at all.

## 39.5. IDEA

The IDEA feedback is important feedback for helping Faculty, Departments, and URI at large monitor how we are doing at teaching. Filling out this feedback is important, even though it does not impact your time in this class. You fill it out to help future students in this class and help instructors improve our overall teaching, and you could see any of us again.

### [IDEA Feedback](#)

### ❗ Important

If the class gets to 70% response rate, you'll all be allowed to reply to portfolio 4 feedback and change earn achievements you were close on but didn't quite get. (whatever number is appropriate; still a deadline on 12/22, so you'll get ~24 hours to reply & change your grade )

## 39.6. More Practice

### ! Important

these questions will help you be prepared for questions to earn the last few achievements in class time.

1. What questions would you like to ask a Data scientist about how they do their work? Bring questions to the last two classes for our speakers (see bios from Prismia on 12/3).
2. How do different machine learning tasks (classification, regression, clustering) compare?
3. How do different models within a task compare?
4. What can your model's performance tell you?
5. Pick an ML application and think about what data, and tools you would need to use in order to replicate the system.
6. Review the import cells in different classes and be sure you know why we import each package, module, class, and function
7. What does each part of the output of a `GridSearchCV` experiment tell you?

## 40. Neural Networks with Keras

```
import numpy as np # advanced math library
import matplotlib.pyplot as plt
import random # for generating random numbers

from keras.datasets import mnist # MNIST dataset is included in Keras
from keras.models import Sequential # Model type to be used

from keras.layers.core import Dense, Dropout, Activation # Types of layers to be used in
our model
from keras.utils import np_utils # NumPy related tools

from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D,
Flatten
from keras.layers import BatchNormalization
```

```

ModuleNotFoundError Traceback (most recent call last)
/tmp/ipykernel_2675/353079506.py in <module>
 3 import random # for generating random numbers
 4
----> 5 from keras.datasets import mnist # MNIST dataset is included in Keras
 6 from keras.models import Sequential # Model type to be used
 7

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/keras/__init__.py in
<module>
 19 """
 20 # pylint: disable=unused-import
---> 21 from tensorflow.python import tf2
 22 from keras import distribute
 23

ModuleNotFoundError: No module named 'tensorflow'
```

First, we'll use `keras` to build similar networks to the ones we saw with `sklearn` it will be more complex, and we're using a different version of the digits data (larger images) but this will still be just learning the predictions, not the representation for now. On Friday, we'll learn how to add new layers that transform the data and learn the representation at the same time.

### 40.1. Preparing data for deep learning

The MNIST data is split between 60,000 28 x 28 pixel training images and 10,000 28 x 28 pixel images. It's realated to the digits data that we have seen before.

We will load the data and look at a random sample.

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()

plt.rcParams['figure.figsize'] = (9,9) # Make the figures a bit bigger

for i in range(9):
 plt.subplot(3,3,i+1)
 num = random.randint(0, len(X_train))
 plt.imshow(X_train[num], cmap='gray', interpolation='none')
 plt.title("Class {}".format(y_train[num]))

plt.tight_layout()
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/114129937.py in <module>
----> 1 (X_train, y_train), (X_test, y_test) = mnist.load_data()
 2
 3
 4 plt.rcParams['figure.figsize'] = (9,9) # Make the figures a bit bigger
 5

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

Next, we need to transform the data to be compatible by reshaping 28 x 28 matrices into vectors, then converting to floats and normalizing.

28\*28

784

Now that we know the length, we can transform.

```
X_train = X_train.reshape(60000, 784) # 60,000 training samples
X_test = X_test.reshape(10000, 784) # 10,000 test samples

X_train = X_train.astype('float32') # change integers to 32-bit floating point numbers
X_test = X_test.astype('float32')

X_train /= 255 # normalize each value for each pixel for the entire vector
for each input
X_test /= 255

print("Training matrix shape", X_train.shape)
print("Testing matrix shape", X_test.shape)
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/3863644356.py in <module>
----> 1 X_train = X_train.reshape(60000, 784) # 60,000 training samples
 2 X_test = X_test.reshape(10000, 784) # 10,000 test samples
 3
 4 X_train = X_train.astype('float32') # change integers to 32-bit floating point
numbers
 5 X_test = X_test.astype('float32')

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

We will use one-hot encoding for the target variable, since our neural net's output layer is actually the probability distribution over the 10 digits, not a value from 0 to 9.

```
nb_classes = 10 # number of unique digits

Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/757403537.py in <module>
 1 nb_classes = 10 # number of unique digits
 2
----> 3 Y_train = np_utils.to_categorical(y_train, nb_classes)
 4 Y_test = np_utils.to_categorical(y_test, nb_classes)

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

```
y_train[0]
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/1880631569.py in <module>
----> 1 y_train[0]

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

```
Y_train[0]
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/1686574290.py in <module>
----> 1 Y_train[0]

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

## 40.2. Building a neural network in Keras

We will start with a network similar to what we have seen in `sklearn`. It will have a number of layers and, when predicting pass data from one layer to the next sequentially.

```
model = Sequential()
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/2043716270.py in <module>
----> 1 model = Sequential()

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

Next we add layers. In `keras` what we saw as one neuron (connections + activation) before is treated as two separate layers, where the size of the layer is the number of neurons.

```
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/1306613557.py in <module>
----> 1 model.add(Dense(512, input_shape=(784,)))
 2 model.add(Activation('relu'))

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

```
model.add(Dense(512))
model.add(Activation('relu'))
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/1864917906.py in <module>
----> 1 model.add(Dense(512))
 2 model.add(Activation('relu'))

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

```
model.add(Dense(10))
model.add(Activation('softmax'))
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/1394869408.py in <module>
----> 1 model.add(Dense(10))
 2 model.add(Activation('softmax'))

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

```
model.summary()
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/3470139634.py in <module>
----> 1 model.summary()

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

Keras also has parameters about the optimization, like we saw before, but we set those with the `compile` method.

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/2998010697.py in <module>
----> 1 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=
 ['accuracy'])

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

```
model.fit(X_train, Y_train,
 batch_size=128, epochs=5,
 verbose=1)
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/3754089279.py in <module>
----> 1 model.fit(X_train, Y_train,
 2 batch_size=128, epochs=5,
 3 verbose=1)

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

```
score = model.evaluate(X_test, Y_test)
score
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2675/649540647.py in <module>
----> 1 score = model.evaluate(X_test, Y_test)
 2 score

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

#### 40.3. What kind of mistakes did it make?

```

predicted_prob = model.predict(X_test)
predicted_classes = np.argmax(predicted_prob, axis=1)

Check which items we got right / wrong
correct_indices = np.nonzero(predicted_classes == y_test)[0]
incorrect_indices = np.nonzero(predicted_classes != y_test)[0]

plt.figure()
for i, correct in enumerate(correct_indices[:9]):
 plt.subplot(3,3,i+1)
 plt.imshow(X_test[correct].reshape(28,28), cmap='gray', interpolation='none')
 plt.title("Predicted {}, Class {}".format(predicted_classes[correct],
y_test[correct]))

plt.tight_layout()

plt.figure()
for i, incorrect in enumerate(incorrect_indices[:9]):
 plt.subplot(3,3,i+1)
 plt.imshow(X_test[incorrect].reshape(28,28), cmap='gray', interpolation='none')
 plt.title("Predicted {}, Class {}".format(predicted_classes[incorrect],
y_test[incorrect]))

plt.tight_layout()

```

```

NameError Traceback (most recent call last)

/tmp/ipykernel_2675/798327639.py in <module>

----> 1 predicted_prob = model.predict(X_test)

 2 predicted_classes = np.argmax(predicted_prob, axis=1)

 3

 4

 5 # Check which items we got right / wrong

NameError: name 'model' is not defined

```

## 40.4. Dropout for less overfitting

Dropout makes some of the neurons zero.

```

model_dropout = Sequential()

model_dropout.add(Dense(512, input_shape=(784,)))

model_dropout.add(Activation('relu'))

model_dropout.add(Dropout(0.2))

model_dropout.add(Dense(512))

model_dropout.add(Activation('relu'))

model_dropout.add(Dropout(0.2))

model_dropout.add(Dense(10))

model_dropout.add(Activation('softmax'))

model_dropout.summary()

```

```

NameError Traceback (most recent call last)

/tmp/ipykernel_2675/1322532985.py in <module>

----> 1 model_dropout = Sequential()

 2

 3 model_dropout.add(Dense(512, input_shape=(784,)))

 4 model_dropout.add(Activation('relu'))

 5 model_dropout.add(Dropout(0.2))

NameError: name 'Sequential' is not defined

```

Again, we set the optimization parameters and then we can fit.

```

model_dropout.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model_dropout.fit(X_train, Y_train,
 batch_size=128, epochs=5,
 verbose=1)

```

```

NameError Traceback (most recent call last)
/tmp/ipykernel_2675/105453219.py in <module>
----> 1 model_dropout.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
 2 model_dropout.fit(X_train, Y_train,
 3 batch_size=128, epochs=5,
 4 verbose=1)

NameError: name 'model_dropout' is not defined

```

```

score = model_dropout.evaluate(X_test, Y_test)
score

```

```

NameError Traceback (most recent call last)
/tmp/ipykernel_2675/439353392.py in <module>
----> 1 score = model_dropout.evaluate(X_test, Y_test)
 2 score

NameError: name 'model_dropout' is not defined

```

Now we see the gap between the train and test performance is smaller and the test performance is higher.

## 41. Convolutional Neural Networks

```

import numpy as np # advanced math library
import matplotlib.pyplot as plt
import random # for generating random numbers

from keras.datasets import cifar10
from keras.models import Sequential # Model type to be used

from keras.layers.core import Dense, Dropout, Activation # Types of layers to be used in
our model
from keras.utils import np_utils # NumPy related tools

from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling2D,
Flatten
from keras.layers import BatchNormalization

```

```

ModuleNotFoundError Traceback (most recent call last)
/tmp/ipykernel_2695/1206398682.py in <module>
 3 import random # for generating random numbers
 4
----> 5 from keras.datasets import cifar10
 6 from keras.models import Sequential # Model type to be used
 7

/opt/hostedtoolcache/Python/3.7.12/x64/lib/python3.7/site-packages/keras/__init__.py in
<module>
 19 """
 20 # pylint: disable=unused-import
--> 21 from tensorflow.python import tf2
 22 from keras import distribute
 23

ModuleNotFoundError: No module named 'tensorflow'

```

```

(c_X_train, c_y_train), (c_X_test, c_y_test) = cifar10.load_data()

```

```

NameError Traceback (most recent call last)
/tmp/ipykernel_2695/1907254862.py in <module>
----> 1 (c_X_train, c_y_train), (c_X_test, c_y_test) = cifar10.load_data()

NameError: name 'cifar10' is not defined

```

```

c_X_test.shape

```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2695/607418696.py in <module>
----> 1 c_X_test.shape

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

```
nb_classes = 10

c_X_train_flat = c_X_train.reshape(50000, 3072) #add an additional dimension to
represent the single-channel
c_X_test_flat = c_X_test.reshape(10000, 3072)

c_X_train_flat = c_X_train_flat.astype('float32') # change integers to 32-bit
floating point numbers
c_X_test = c_X_test.astype('float32')

c_X_train_flat /= 255 # normalize each value for each pixel
for the entire vector for each input
c_X_test /= 255

c_Y_train = np_utils.to_categorical(c_y_train, nb_classes)
c_Y_test = np_utils.to_categorical(c_y_test, nb_classes)
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2695/3266050630.py in <module>
 1 nb_classes = 10
 2
----> 3 c_X_train_flat = c_X_train.reshape(50000, 3072) #add an additional dimension to
represent the single-channel
 4 c_X_test_flat = c_X_test.reshape(10000, 3072)
 5

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

```
model_dropout = Sequential()

model_dropout.add(Dense(512, input_shape=(3072,)))
model_dropout.add(Activation('sigmoid'))
model_dropout.add(Dropout(0.2))

model_dropout.add(Dense(512))
model_dropout.add(Activation('relu'))
model_dropout.add(Dropout(0.2))

model_dropout.add(Dense(10))
model_dropout.add(Activation('softmax'))
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2695/2866788784.py in <module>
----> 1 model_dropout = Sequential()
 2
 3 model_dropout.add(Dense(512, input_shape=(3072,)))
 4 model_dropout.add(Activation('sigmoid'))
 5 model_dropout.add(Dropout(0.2))

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

```
model_dropout.summary()
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2695/1531633143.py in <module>
----> 1 model_dropout.summary()

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

```
model_dropout.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model_dropout.fit(c_X_train_flat, c_Y_train,
 batch_size=128, epochs=5,
 verbose=1)

score = model_dropout.evaluate(c_X_test_flat, c_Y_test)
score
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2695/856665321.py in <module>
----> 1 model_dropout.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
 2 model_dropout.fit(c_X_train_flat, c_Y_train,
 3 batch_size=128, epochs=5,
 4 verbose=1)
 5

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

```
c_X_train = c_X_train.astype('float32') # change integers to 32-bit floating
point numbers
c_X_train = c_X_train.astype('float32')

c_X_train /= 255 # normalize each value for each pixel for
the entire vector for each input
c_X_train /= 255
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2695/2259824588.py in <module>
----> 1 c_X_train = c_X_train.astype('float32') # change integers to 32-bit
floating point numbers
 2 c_X_train = c_X_train.astype('float32')
 3
 4 c_X_train /= 255 # normalize each value for each
pixel for the entire vector for each input
 5 c_X_train /= 255

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

```

model_cnn = Sequential() # Linear stacking of layers

Convolution Layer 1
model_cnn.add(Conv2D(32, (3, 3), input_shape=(32,32,3))) # 32 different 3x3 kernels -- so 32 feature maps
model_cnn.add(BatchNormalization(axis=-1)) # normalize each feature map before activation
convLayer01 = Activation('relu') # activation
model_cnn.add(convLayer01)

Convolution Layer 2
model_cnn.add(Conv2D(32, (3, 3))) # 32 different 3x3 kernels -- so 32 feature maps
model_cnn.add(BatchNormalization(axis=-1)) # normalize each feature map before activation
model_cnn.add(Activation('relu')) # activation
convLayer02 = MaxPooling2D(pool_size=(2,2)) # Pool the max values over a 2x2 kernel
model_cnn.add(convLayer02)

Convolution Layer 3
model_cnn.add(Conv2D(64,(3, 3))) # 64 different 3x3 kernels -- so 64 feature maps
model_cnn.add(BatchNormalization(axis=-1)) # normalize each feature map before activation
convLayer03 = Activation('relu') # activation
model_cnn.add(convLayer03)

Convolution Layer 4
model_cnn.add(Conv2D(64, (3, 3))) # 64 different 3x3 kernels -- so 64 feature maps
model_cnn.add(BatchNormalization(axis=-1)) # normalize each feature map before activation
model_cnn.add(Activation('relu')) # activation
convLayer04 = MaxPooling2D(pool_size=(2,2)) # Pool the max values over a 2x2 kernel
model_cnn.add(convLayer04)
model_cnn.add(Flatten()) # Flatten final 4x4x64 output matrix into a 1024-length vector

Fully Connected Layer 5
model_cnn.add(Dense(512)) # 512 FCN nodes
model_cnn.add(BatchNormalization()) # normalization
model_cnn.add(Activation('relu')) # activation

Fully Connected Layer 6
model_cnn.add(Dropout(0.2)) # 20% dropout of randomly selected nodes
model_cnn.add(Dense(10)) # final 10 FCN nodes
model_cnn.add(Activation('softmax')) # softmax activation

```

```

NameError Traceback (most recent call last)

/tmp/ipykernel_2695/587939668.py in <module>

----> 1 model_cnn = Sequential() # Linear stacking of layers

 2

 3 # Convolution Layer 1

 4 model_cnn.add(Conv2D(32, (3, 3), input_shape=(32,32,3))) # 32 different 3x3 kernels -- so 32 feature maps

 5 model_cnn.add(BatchNormalization(axis=-1)) # normalize each feature map before activation

```

NameError: name 'Sequential' is not defined

```
model_cnn.summary()
```

```

NameError Traceback (most recent call last)

/tmp/ipykernel_2695/53097851.py in <module>

----> 1 model_cnn.summary()


```

NameError: name 'model\_cnn' is not defined

```

model_cnn.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model_cnn.fit(c_X_train, c_Y_train,
 batch_size=128, epochs=5,
 verbose=1)

score = model_cnn.evaluate(c_X_test, c_Y_test)

```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2695/3457640511.py in <module>
----> 1 model_cnn.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
 2 model_cnn.fit(c_X_train, c_Y_train,
 3 batch_size=128, epochs=5,
 4 verbose=1)
 5

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

```
score_cnn = model_cnn.evaluate(c_X_test, c_Y_test)
score_cnn
```

```
NameError Traceback (most recent call last)
/tmp/ipykernel_2695/1771262991.py in <module>
----> 1 score_cnn = model_cnn.evaluate(c_X_test, c_Y_test)
 2 score_cnn

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

# 1. Portfolio Setup, Data Science, and Python

Due: 2020-09-12

## 1.1. Objective & Evaluation

This assignment is an opportunity to earn level 2 achievements for the `process` and `python` and confirm that you have all of your tools setup, including your portfolio.

## 1.2. To Do

### Important

If you have trouble, check the GitHub FAQ on the left before e-mailing

### 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/fall21instructors

To do this click Issues at the top, the green "New Issue" button and then type away.

```
```{warning}
If you have trouble with the (*)d steps, don't worry, we can help work around these later. To help
us out, document the errors as bugs on your repository.
````
```

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 change the logo if you wish
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. (\*) Install some additional python packages with: `pip install pip install -r requirements.txt` (this is a python operation, so use anaconda prompt on Windows, if the pip version doesn't work, try it with conda: `conda install --file requirements.txt`) form inside the portfolio folder

7. (\*) Configure precommit to help keep your repo clean with `pre-commit install`. If this step doesn't work, see the portfolio README under "Using your Jupyter Book Portfolio"
8. 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**
  - 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. Remember you will have many chances to earn level 2 achievement in python
9. 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 (+/-)
...
```

Here are some sample tests you could run to confirm that your function works correctly:

### ! 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.

### i Note

when the value of the expression after `assert` is `True`, it will look like nothing happened. `assert` is used for testing

```
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+'
```

## 1.3. Submission Instructions

Create a Jupyter Notebook with your function in your portfolio folder 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.

There will be a pull request on your repository that is made by GitHub classroom, [request a review](#) from @rhodypro4dg/fall21instructors.

## 2. Practicing Python and Accessing Data

due : 2020-09-21

### 2.1. 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. The following table summarizes the grading. It supplements the skill definitions from the [Achievement Definitions](#).

| 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 rubric for grading

First, [accept the assignment](#). It contains a notebook with some template structure (and will set you up for grading).

### 2.2. 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 notebook that includes:

- heading of the dataset's name
- a link to where someone can learn about the dataset
- a 1-2 sentence summary of what the dataset contains and why it was collected
- 1-2 questions you would like to answer with that dataset.

#### 💡 Hint

The [Datasets page](#) has information about data for any assignment. For this assignment, the [Best for loading directly into a notebook](#) section is probably the best place to start.

### 2.3. Store them for loading

Create a list of dictionaries in [datasets.py](#), so that there is one dictionary for each dataset with the url, a name, and what function should be used to load the data into a [pandas.DataFrame](#).

### 💡 Hint

The goal here is to set up this list of dictionaries so that you can load data using different functions in each pass through the loop, without an `if` statement. You'll be iterating over the list of dictionaries, so the loop variable be a different dictionary each time.

see the where we used a lambda in a dictionary in [the class notes](#) for more information.

### 💡 Hint

Any `.py` file can become a [module](#)

### 💡 Tip

Urls are strings. The `string` class in python has a lot of helpful methods for manipulating strings, like `split`.

### 💡 Tip

The [pandas IO](#) page has information about how to read data in and save data out of pandas.

## 2.4. 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. save it 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.
2. record attributes about the dataset as in the table below in a list of lists:
3. Use that to create a DataFrame with the following columns:

|               |                                              |
|---------------|----------------------------------------------|
| 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 2.2 Meta Data Description of the DataFrame to build

## 2.5. Manipulate your datasets

For one dataset that includes nonnumerical data:

- display the heading and the last 4 rows
- make and display a new data frame with only the numerical columns (select these programmatically)

For any other dataset:

- display the heading and 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:

- display the first 3 odd rows (eg 1,3,5) of the data for two columns of your choice

## 2.6. Exploring data files

For each dataset, in a separate section of your notebook titled `When things go wrong`:

- try reading in data with the wrong `read_` function and make notes about what happens.
- was the format that the data was provided in a good format? why or why not?
- 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.

### 💡 Think Ahead

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](#) eg this [google scholar result](#) How could something like this impact your work as a data scientist?

### ⚠️ 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.

## 3. Assignment 3: Exploratory Data Analysis

—Due:2020-09-28 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.

| task                                                                 | skill (max level)            |
|----------------------------------------------------------------------|------------------------------|
| compute and display overall statistics                               | summarize (2)                |
| compute and display individual statistics of a datasets              | summarize (2)                |
| group a data set by a variable and compute summary statics           | summarize (2)                |
| plot two different pairwise relationships                            | visualize (2)                |
| interpret the statistics and plots                                   | summarize (2), visualize (2) |
| load data from at least two different file types                     | access (2)                   |
| compare and contrast the file types and/or sources                   | access (2)                   |
| match questions appropriate to the dataset and match plots and stats | process (1)                  |

Table 3.1 plot basic views of data and generate descriptive statistics and basic plots

### 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 statistics
3. Write a short description of what the data contains and what it could be used for
4. Ask and answer 4 questions using statistics, split-apply-combine, and visualizations. Make a heading for each question using a markdown cell and `H2:##`. Make sure your analyses meet the criteria in the check lists below. Interpret the answer from the analysis/plot.
5. Describe what, if anything might need to be done to clean or prepare this data for further analysis

### 3.3.1. Dataset 1 Checklist

1. Overall summary statistics grouped by a categorical variable
2. A single statistic grouped by a categorical variable
3. a scatter plot with the points colored by a categorical variable
4. a plot and summary table that convey the same information. This can be one statistic or many.

### 3.3.2. Dataset 2 Analysis

1. two individual summary statistics for one variable
2. one summary statistic grouped by two categorical variables
3. a figure with a grid of subplots that correspond to two categorical variables
4. 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.

 **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:

Due: 2020-10-05 11:59pm

[accept the assignment](#)

| task                                                      | skill (max level)    |
|-----------------------------------------------------------|----------------------|
| drop nan rows from a dataset                              | prepare (2)          |
| display parts of a dataframe                              | summarize (1)        |
| impute a value to fill missing values                     | prepare (2)          |
| filter data based on extreme values or other outliers     | prepare (2)          |
| convert a variable to one hot encoding                    | prepare (2)          |
| add a new column computed from one or more other columns  | prepare (2)          |
| transform a dataset to tidy format                        | prepare (2)          |
| compute overall and individual summary statistics         | summarize (2)        |
| use split-apply-combine paradigm                          | summarize (2)        |
| generate at least two types of plots                      | visualize (2)        |
| interpret statistics and plots                            | summarize, visualize |
| use list comprehensions or loops and pythonic conventions | python (2)           |
| load data from at least two types                         | access (2)           |
| compare data storage formats                              | access (2)           |
| match EDA techniques to questions appropriately           | process (1)          |

Table 4.1 practice basic pandas by reshaping and organizing data

For this assignment, prepare the provided datasets. Your preparation needs to include the following steps and narrative description of how you're making decisions about your data cleaning.

The notebooks in the template have instructions for how to work with each dataset.

To earn prepare level 2, clean the data and do just enough exploratory data analysis to show that the data is usable (eg 1 stat and/or plot). For prepare level 2:

- travel\_times AND one of:
- cs\_degrees, airlines, and coffee

To earn summarize and visualize level 2, add extra exploratory data analyses meeting the criteria above.

To earn python level 2, make sure that you use a function or lambda and comprehension or pythonic loops somewhere. The CS degrees data will have that, but it's harder. The coffee data will be the easiest one to get all python level 2.

For access level 2 you must clean the airline data (to get data in a second file type).

#### 💡 Hint

renaming thing is often done well with a dictionary comprehension or lambda.

## 5. Assignment 5: Constructing Datasets and Using Databases

[accept the assignment](#)

Due: 2020-10-12 11:59pm

| task                                                      | skill                |
|-----------------------------------------------------------|----------------------|
| drop nan rows from a dataset                              | prepare (2)          |
| impute a value to fill missing values                     | prepare (2)          |
| filter data based on extreme values or other outliers     | prepare (2)          |
| convert a variable to one hot encoding                    | prepare (2)          |
| add a new column computed from one or more other columns  | prepare (2)          |
| transform a dataset to tidy format                        | prepare (2)          |
| append a dataset provided in pieces                       | construct (2)        |
| merge data with a shared column                           | construct (2)        |
| compute overall and individual summary statistics         | summarize (2)        |
| use split-apply-combine paradigm                          | summarize (2)        |
| generate at least two types of plots                      | visualize (2)        |
| interpret statistics and plots                            | summarize, visualize |
| use list comprehensions or loops and pythonic conventions | python (2)           |

Table 5.1 access data from a database and merge multiple tables from a dataset

## 5.1. Constructing Datasets

Your goal is to programmatically construct two ready to analyze dataset from multiple sources.

- Each dataset must combine at least 2 source tables(4 total).
- At least one source table(of the 4) must come from an sqlite 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 construct, this can be very minimal EDA.

**You may build one dataset from three tables instead of two from two each if you'd like**

## 5.2. Earning additional achievements

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:

- transform into a tidy format
- add a new column by computing from others
- handle NaN values by dropping or filling
- drop a column, row, or duplicates in another way

### 5.2.2. Summarize and Visualize level 2

To earn level 2 for summarize and/or visualize, include additional analyses after building the datasets. Include:

- compute overall summary statistics
- compute individual summary statistics
- use split-apply-combine with two categorical variables
- at least two types of plots for visualize
- use a categorical variable to modify the plot (color points or create subplots)

### 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: Understanding Classification

[accept the assignment](#)

**Due: 2020-10-19 11:59pm**

| task                                                                 | skill              |
|----------------------------------------------------------------------|--------------------|
| fit a naive bayes classifier                                         | classification (1) |
| explain when the model works/does not and what to use to investigate | classification (2) |
| evaluate the performance of a classifier                             | evaluate (1)       |

Table 6.1 demonstrate understanding of classification as a tasks and how to evaluate them

#### Important

You only need to do datasets 1,2, 5, and 6.

For each dataset, answer the following:

1. Do you expect Gaussian Naive Bayes to work well on this dataset, why or why not?
  - think about the assumptions of naive bayes and classification in general)
  - *explanation is essential here, because you can actually use the classifier to check*
2. How well does a Gaussian Naive Bayes classifier work on this dataset?
  - check the overall performance
3. How does the actual performance compare to your prediction? If it performs much better or much worse than you expected, what might you use to figure out why?

#### Tip

*you do not have to figure out why your predictions were not correct, just list tools you've learned in class that might help you figure that out*

### 💡 Think ahead

Do you think a different classifier might work better or do you think this data cannot be predicted any better than this?

- check the type of errors
- are the errors random or are some errors more common than others?

## 7. Assignment 7: Decision Trees

### 7.1. Quick Facts

- [accept the assignment](#)
- Recommended completion:
- Due: 2021-10-27 11:59pm

### 7.2. Related notes

- [2021-10-18](#)
- [2021-10-20](#)
- [2021-10-22](#)

### 7.3. Assessment

| task                                                                                 | skill                        |
|--------------------------------------------------------------------------------------|------------------------------|
| fit a decision tree                                                                  | classification (2)           |
| apply a decision tree to get predictions                                             | classification (2)           |
| interpret the model assumed by a decision tree                                       | classification (2)           |
| use multiple metrics evaluate performance                                            | evaluate (2)                 |
| interpret how decisions (test/train size, model parameters) impact model performance | evaluate (2)                 |
| interpret the classifier performance in the context of the dataset                   | process (2)                  |
| analyze the impact of model parameters on model performance                          | process (2)                  |
| use loops and lists effectively                                                      | python (2)                   |
| use EDA techniques to examine the experimental results                               | summarize (2), visualize (2) |
| create a dataset by combining data from multiple sources                             | construct (2)                |

Table 7.1 fit a decision tree

### 7.4. Instructions

Choose a datasets that is well suited for classification and that has only numerical features.

### 💡 Tip

A file can be a “comma separated file” and read in with `pd.read_csv` even if the file name does not end in “.csv”. The part after the ‘.’ in a file name is called the file *extension* and its a sort of metadata built into a file. CSV is a specification for how to write data to a file, or a file *format*. It’s best practice to make the file extension match the file format, but it’s very much not required. Especially the older files on the UCI repository, the extension is something else (eg dat, or data, or names), but the actual contents of the files are comma separated and compatible with `read_csv`

Practice using decision trees and exploring how classification works, and what evaluations mean in the following exercises.

### 💡 Hint

the Wisconsin Breast Cancer data from UCI is a good option as is the wine data set, which has the red & white wines separated, so you can earn prepare with this. You could also use the NSA data and try to predict who will makes the NBA 75 based on their game stats, this would require some manipulation and so would be a way to earn construct.

### ℹ Note

If you want to use a dataset with nonnumerical features you will have to convert the categorical features to one hot encoding or drop them if there are enough continuous values in addition to the categorical.

#### 7.4.1. Part 1: DT Basics

1. Include a basic description of the data(what the features are)
2. Write your own description of what the classification task is and why a decision tree is a reasonable model to try for this data.
3. Fit a decision tree with the default parameters on 50% of the data
4. Test it on 50% held out data and generate a classification report
5. Inspect the model to answer:
  - Does this model make sense?
  - Are there any leaves that are very small?
  - Is this an interpretable number of levels?
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?

#### 7.4.2. Part 2: Exploring Evaluation

Do an experiment to compare test set size vs performance:

1. Train decision tree with max depth 2 less than the depth it found above on 10%, 30%, … , 90% of the data. Save the results of both test accuracy and training 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?

### 💡 Hint

use a loop for this part, possibly also a function

### 💡 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.

#### 7.4.3. Part 3: DT parameters

## Experiment with DT Parameters:

1. Choose one parameter to change in the training that you think might improve the model and say why, then train a second decision tree
2. Check the performance of the new decision tree with at least two performance metrics
3. Did changing the parameter do what you expected?
4. Choose a second parameter to change in the training that you think might improve the model and say why, then train a third decision tree
5. Validate your third decision tree with at least two performance metrics.
6. Did changing the parameter do what you expected?

### 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

Repeat your experiment from Part 2 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?

Repeat the experiment in part 2 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: Regression

## 8.1. Quick Facts

- [accept the assignment](#)
- Due: 2020-11-03 11:59pm

## 8.2. Related notes

- [2021-10-25](#)
- [2021-10-29](#)

## 8.3. Assessment

| task                                                                                 | skill                        |
|--------------------------------------------------------------------------------------|------------------------------|
| fit a linear regression model                                                        | regression (2)               |
| evaluate fit of linear regression                                                    | evaluate (2)                 |
| use multiple metrics evaluate performance                                            | evaluate (2)                 |
| interpret how decisions (test/train size, model parameters) impact model performance | evaluate (2)                 |
| interpret the model performance in the context of the dataset                        | process (2)                  |
| analyze the impact of model parameters on model performance                          | process (2)                  |
| use loops and lists effectively                                                      | python (2)                   |
| use EDA techniques to examine the experimental results                               | summarize (2), visualize (2) |
| create a dataset by combining data from multiple sources                             | construct (2)                |

## 8.4. Instructions

Find a dataset suitable for regression. We recommend a dataset from the UCI repository. Complete the following in a single notebook.

### 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 regression task is and why a linear model is a reasonable model to try for this data.
  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:
    - o Does this model make sense?
    - o What do the coefficients tell you?
    - o What do the residuals tell you?
  6. Repeat the split, train, and test steps 5 times.
    - o 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

#### Note

If you have the relevant level 2 achievements (evaluation, summarize, visualize) you can skip this part, but it might still be interesting.

Do an experiment to compare test set size vs performance:

1. Re-fit your regression model using 10%, 30%, ... , 90% of the data for training. Save the results of both test and train r2 and MSE for each size training data in a DataFrame with columns  
['train\_pct','n\_train\_samples','n\_test\_samples','train\_r2','test\_r2','train\_mse','test\_mse']
2. Plot the metrics 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

1. Try these experiments with a different type of regression.
2. How do your evaluation experiment results compare in regression vs classification?

## 9. Assignment 9: Clustering

### 9.1. Quick Facts

- [accept the assignment](#)
- Due: 2020-11-10 11:59pm

### 9.2. Related notes

- [2021-11-01](#)
- [2021-11-03](#)
- [2021-11-05](#)

### 9.3. Assessment

| task                                                               | skill                        |
|--------------------------------------------------------------------|------------------------------|
| apply and interpret kmeans clustering                              | clustering (2)               |
| use multiple metrics evaluate performance                          | evaluate (2)                 |
| interpret how decisions impact model performance                   | evaluate (2)                 |
| interpret the classifier performance in the context of the dataset | process (2)                  |
| analyze the impact of model parameters on model performance        | process (2)                  |
| use EDA techniques to interpret the experimental results           | summarize (2), visualize (2) |

### 9.4. 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,  $\backslash(K\}$ .
3. Evaluate how well clustering worked on the data:
  - using a true clustering metric
  - using visual inspection
  - 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 different number of clusters:

- can you interpret the new clusters?
- how do they relate to the original clusters? are they completely different, did one split? did some merge?
- is there a reasonable explanation for more clusters than there are classes in this dataset?

#### Think Ahead

How can clustering be used to ask many different questions? What can you do with clustering results?

#### Hint

a true clustering metric works in an unsupervised way, it does not need to use the true labels, unlike the metrics we used for classification and regression. However, when we have the labels, we can see if the clustering algorithm recovers the same groups we know exist in the data.

See the notes on [Evaluating Clustering](#) for an example of each. There's also a margin note, with a link to sklearn docs with more. If you're curious you can try different metrics from the notes.

## 10. Assignment 10: Tuning Model Parameters

### 10.1. Quick Facts

- [accept the assignment](#)
- Due: 2020-11-17 11:59pm

### 10.2. Related notes

- [2021-11-08](#)
- [2021-11-12](#)

### 10.3. Assessment

| task                                                                                  | skill                                                         |
|---------------------------------------------------------------------------------------|---------------------------------------------------------------|
| test the model on test data and interpret in context                                  | classification, clustering, OR regression (2)                 |
| choose and justify appropriate parameters parameter grid for the context              | classification, clustering, OR regression (2) AND process (2) |
| evaluate fit of the model while varying the cross validation parameters and interpret | evaluate (2)                                                  |
| optimize model parameter (s) and interpret                                            | optimize (2)                                                  |
| ask relevant questions of the data domain and interpret results in context            | process (2)                                                   |
| use EDA techniques to examine the experimental results                                | summarize (2), visualize (2)                                  |

Table 10.1 Optimize model parameters

Note you can only earn one of classification, clustering, OR regression, but to earn one of those you must both interpret the model and the parameters.

For process you must situate your overall analysis in context (which you should do even if you already have the process achievement; you should always do this) and explain how you're picking parameters to evaluate. Your reasons only have to be reasonable, they don't have to be correct. It's okay if what you try doesn't improve the model, but then you have to interpret that.

### 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.
2. Fit the model and show some exploration to choose reasonable model parameter values for your parameter grid
3. Use grid search to find the best performing model parameters

4. Examine and interpret the cv results. How do they vary in terms of time? Is the performance meaningfully different or just a little?
5. Try varying the cross validation parameters. 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?

## 11. Assignment 11: Model Comparison

### 11.1. Quick Facts

[accept the assignment](#)

**Due:** 2020-11-24 11:59pm

### 11.2. Related notes

- [2021-11-15](#)
- [2021-11-17](#)
- 2021-11-19

### 11.3. Assessment

| task                                                                                  | skill                                                         |
|---------------------------------------------------------------------------------------|---------------------------------------------------------------|
| determine the best model for a given dataset                                          | compare (2)                                                   |
| test the model on test data and interpret in context                                  | classification, clustering, OR regression (2)                 |
| choose and justify appropriate parameters parameter grid for the context              | classification, clustering, OR regression (2) AND process (2) |
| evaluate fit of the model while varying the cross validation parameters and interpret | evaluate (2)                                                  |
| interpret the classifier performance in the context of the dataset                    | process (2)                                                   |
| analyze the impact of model parameters on model performance                           | process (2)                                                   |
| use EDA techniques to interpret the experimental results                              | summarize (2), visualize (2)                                  |

*Table 11.1 compare models and make a recommendation*

Choose a dataset, it can be appropriate for classification, regression, or clustering. Fit at least two models for the same task and choose the appropriate metrics to compare the fit. Decide which model you would recommend based on a realistic setting for that dataset and include evidence justifying that choice. Summarize your findings with plots and tables as appropriate.

This will be easiest if you use a dataset you've used on for one of the previous assignments or choose another.

### Think Ahead

How would this decision making compare for a more complex model or in more realistic setting.

## 12. Assignment 12: Fake News

### 12.1. Quick Facts

- [accept the assignment](#)
- First feedback: 2021-12-02 6:00pm **Final due date: 2021-12-08 11:59pm**

### 12.2. Evaluation

| task                                                          | skill                        |
|---------------------------------------------------------------|------------------------------|
| transform text data to a format compatible with ML            | representation (2)           |
| plan to solve a real world problem using the tools from class | workflow (2)                 |
| use EDA techniques to interpret the experimental results      | summarize (2), visualize (2) |
| determine the best model for this task                        | compare (2)                  |
| determine the best parameter settings for this task           | optimize (2)                 |

Table 12.1 use text to predict fake from real news

### 12.3. Instructions

Use the dataset in the assignment template repo to answer the following questions. The data includes variables:

- 'text': contents of an article
- 'label': whether it is real or fake news
- 'title': title of the article

1. Is the text or the title of an article more predictive of whether it is real or fake?
2. Are titles of real or fake news more similar to one another?

Consider what difference you can have in how you represent the data and how that might impact your model performance in order. Use summary statistics and visualizations appropriately in order to explain your results.

### Hint

The data set contains a large number of articles (takes a long time to train), you can downsample this to something like a 1,000 articles or so in order to speed up training and evaluation (hint: use shuffle).

## Portfolio Dates and Key Facts

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, you can also earn level 1 or 2. 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.

### **Important**

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.

In 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](#)

## Current: Check 3

The third submission will be graded on the following criteria and due on December 5:

### **Level 3**

| <b>keyword</b>        |                                                                                                                                                           |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>process</b>        | Compare different ways that data science can facilitate decision making                                                                                   |
| <b>classification</b> | fit and apply classification models and select appropriate classification models for different contexts                                                   |
| <b>regression</b>     | fit and explain regularized or nonlinear regression                                                                                                       |
| <b>clustering</b>     | apply multiple clustering techniques, and interpret results                                                                                               |
| <b>evaluate</b>       | Evaluate a model with multiple metrics and cross validation                                                                                               |
| <b>optimize</b>       | Select optimal parameters based of mutiple quanttiateve criteria and automate parameter tuning                                                            |
| <b>compare</b>        | Evaluate tradeoffs between different model comparison types                                                                                               |
| <b>representation</b> | apply transformations in different contexts OR compare and contrast multiple representations a single type of data in terms of model performance          |
| <b>workflow</b>       | Independenlty scope and solve realistic data science problems OR independently learn releted tools and describe strengths and weakenesses of common tools |

Submission Checklist:

- [ ] update your gh action or precommit hook
- [ ] complete your KWL chart
- [ ] add notebooks

## Upcoming Checks

- Check 4: December 20

## Submission Introductions

Your portfolio will be assessed both on your demonstration of skills through your chapters that are inspired by the prompts and that your summary demonstrates that you *know* you learned the skills.

Each portfolio submission, you will edit the corresponding [submission\\_x\\_intro.md](#) file. This is where you write your summary and reflect on your learning. This reflection does not need to be long, it shouldn't take a very long time. It's okay for it to be brief, mostly bullet points.

## KWL Chart

One part of your introduction is a **KWL** or **Know, Want to know, Learned**, chart.

In your file it will look like this:(but longer)

```
```{list-table} Portfolio 1 KWL Chart
:header-rows: 1
:name: kwl1

* - Skill
- Know
- Want to Know
- Learned
* - python
- basics
- more efficient types
-
* - access
- that datasets are collated on kaggle
- how to load data for analysis
- how to load data from 3 different types and compare them
* - ...
- ...
- ...
- ...
````
```

and when you build your portfolio it will render like this:

| Skill  | Know                                 | Want to Know                  | Learned                                                  |
|--------|--------------------------------------|-------------------------------|----------------------------------------------------------|
| python | basics                               | more efficient patterns       | pep8 patterns and common conventions,                    |
| access | that datasets are collated on kaggle | how to load data for analysis | how to load data from 3 different types and compare them |
| ...    | ...                                  | ...                           | ...                                                      |

Table 1 Portfolio 1 KWL Chart

## Overview

In the overview, you summarize the contents of your portfolio. Think of it as the the introduction to the overall submission. Your goal is to help us know what to expect when grading your portfolio and to know that you know what you've learned.

Writing this out will help give you a space to confirm that you're on track by checking your own work against the [Achievement Definitions](#) table. If your work does not earn level 3 achievements, your summary will help us identify if you are on track or if you're not on track. If you're not on track it will help us distinguish between if it's because of a misunderstanding in the expectations or the material.

This summary helps us help you achieve your own goals and lets us help you accordingly. We want you succeed in the course and the best way to do that is to check in frequently.

### Learning Tip

This reflection process also help you learn better, in addition to being an accountability check. What you draw your attention to gets reinforced in your memory, so reflecting on what you've learned helps you learn better.

## 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 earlier 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, but these are ideas for the structure.

### 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 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 possible.

## 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.

## 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<sup>[1]</sup> 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](#))

---

<sup>[1]</sup> 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?

Can I submit this assignment late if ...?

I don't understand my grade on this assignment

## Git and GitHub

I can't push to my repository, I get an error that updates were rejected

The content I added to my portfolio isn't in the pdf

My command line says I cannot use a password

My .ipynb file isn't showing in the staging area or didn't push

My portfolio won't compile

Help! I accidentally merged the Feedback Pull Request before my assignment was graded

## Code Errors

Key Error

<bound method

## 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

### 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 combining 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 general)

### **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 called 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.

## 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 elements 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 summing across 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](#)
- [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.

### 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.

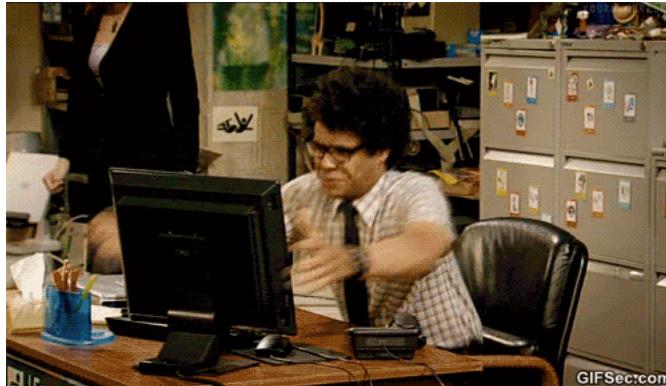
### on email

- [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 environment, 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.

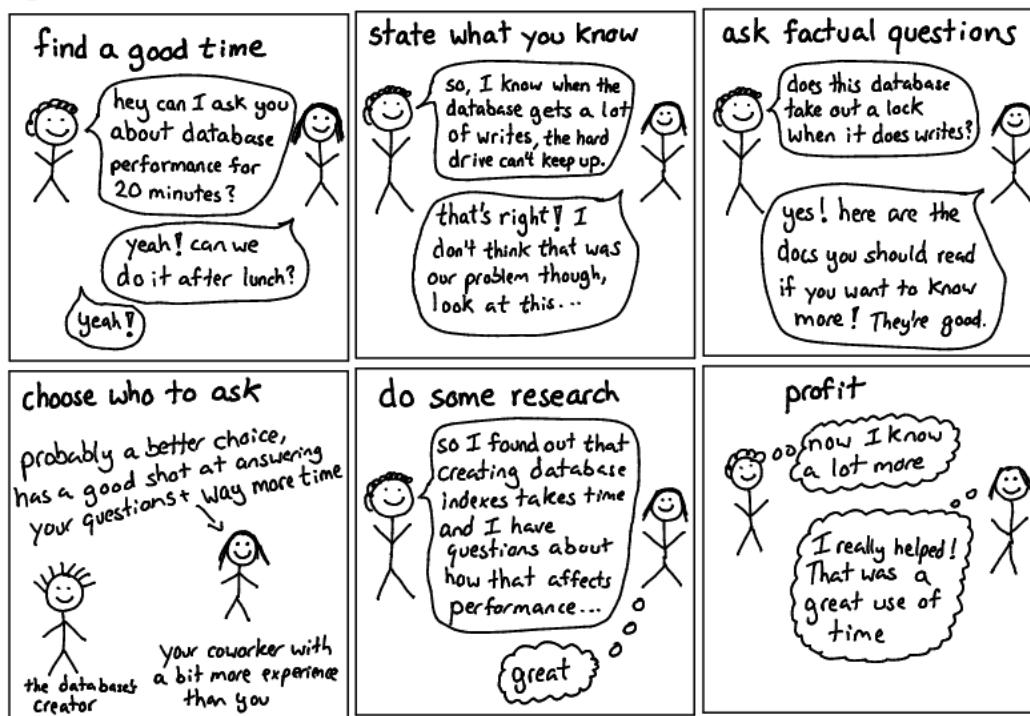
## Portfolios

In portfolios, your goal is to extend and apply the concepts taught in class and practiced in assignments to solve more realistic problems. 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

## asking good 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 of the code you ever run must run in *some* sort of environment. By giving you control, I'm hoping to increase your independence 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 realistic 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 manner.

### 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.

## Windows

Windows has a sort of multiverse of terminal environments.

The least setup required involves using anaconda prompt and **conda** to manage your 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](#)

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.

## 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 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 with 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 portfolio, 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 account. 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.

---

By Professor Sarah M Brown

© Copyright 2021.