Welcome!

261 - Machine Learning at Scale

Summer 2023

Week 1
Live Session

Overview

1. Introductions
2. Housekeeping
    - Things to know
    - Action Items
    - Expectations
3. Goals of the course
4. HW 1 preview
5. Week 1 material
    - Bias-variance
    - Word Count
    - Sort command in Linux
6. Appendix: setup map-reduce cluster in the cloud

Introductions

Teaching Team

Name	Day Job	w261 Role	Time Zone	Email/Slack
Jimi Shanahan	deep learning | computer vision | edge-based computing | large scale data science consultant	Lecturer | Course Creator | Course Coordinator	PST (SF)	jimi@ischool.berkeley.edu
Ramki Gummadi	CEO Argonaut AI	Lecturer	PST (SV)	rkgmd1729@gmail.com
Vinicio De Sola	Senior Data Scientist Aspen Capital (W205 Lecturer)	Lecturer	PST+3 (NY)	vinicio.desola@ischool.berkeley.edu
Luis Villarreal	Cloud Enterprise Architect, Accenture	Lecturer	PST+2 (Texas)	luis.villarreal@ischool.berkeley.edu
Zach Gallante	W261 Alum	TA	EST (Bost)	
Toby Petty	W261 Alum	TA	PST+3 (NY)	
Samuel Gomes	W261 Alum	TA	PST	

Office Hours: weekly

(access details will announced via Slack/DigitalCampus just prior to the start of each OH)

Name	OH period
Jimi Shanahan, and Samuel	Wednesday 3 PM PST
Toby	Thursday 5 PM PST
Vinicio De Sola	Sunday 10 AM PST
Ramki Gummadi	Sunday 3PM PST
Luis (on infrastructure)	Wednesday 6 PM PST for weeks 1 -3

Housekeeping

Register now (if you have not done so already)!!

Welcome to the Summer 2023 edition of W261,

Re: Slack registration [2 Actions Required]

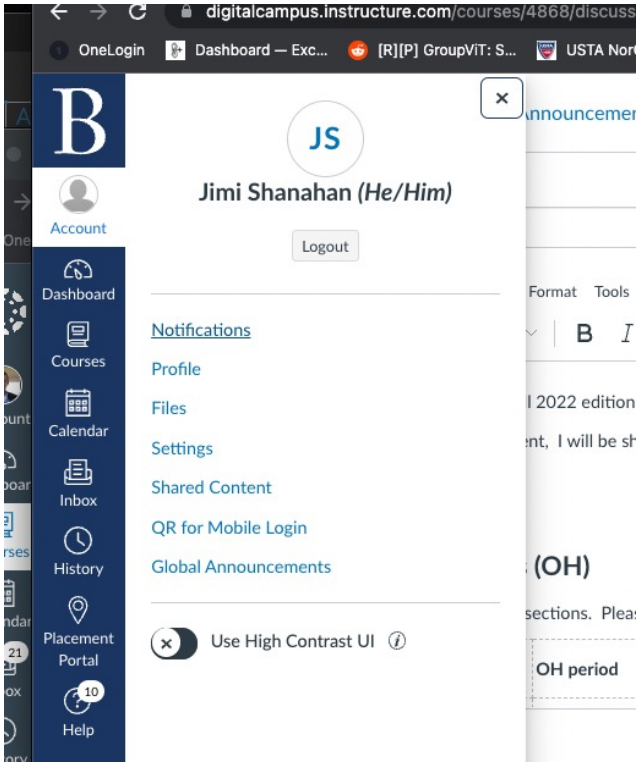
We'll be using Slack for important course communications in addition to the Announcements function from Digital Campus. We an action item here:

Action Item: register yourself on the course slack channels

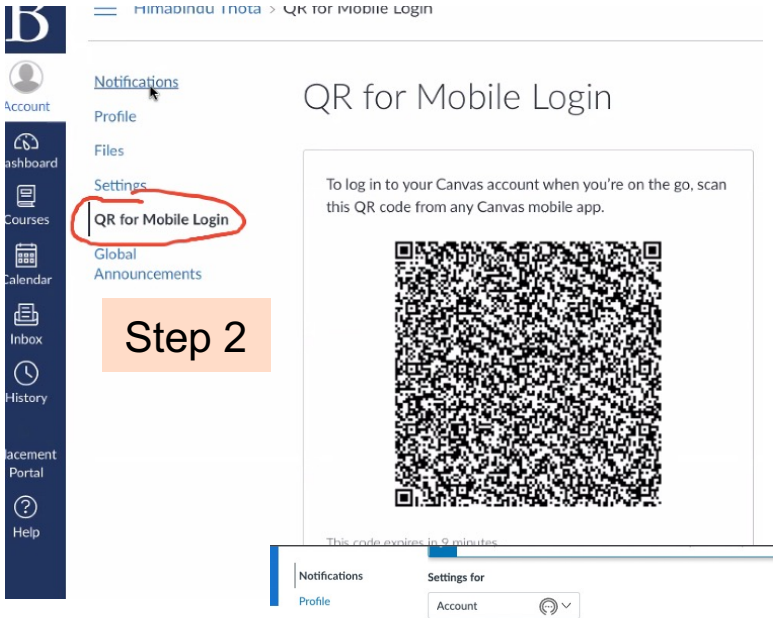
- Please register on the following Slack channels and add yourself using your slack ID (note: the channels will remain open for the next 10 days)
    - datasci-261-2023-summer-announcements
    - datasci-261-2023-summer-main
    - datasci-261-2023-summer-infrastructure

See you in class shortly,
Your W261 Instructional Team:
Jimi, Ramki, Vinicio

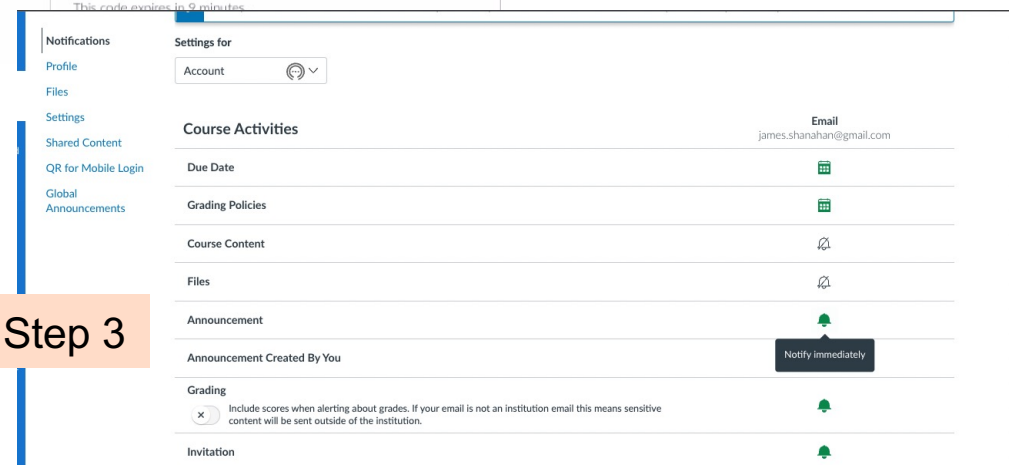
Download the Canvas App: set notifications to *notify immediately*



Step 1



Step 2



Step 3

Live session expectations

Come prepared: do the reading, watch the async, ask questions

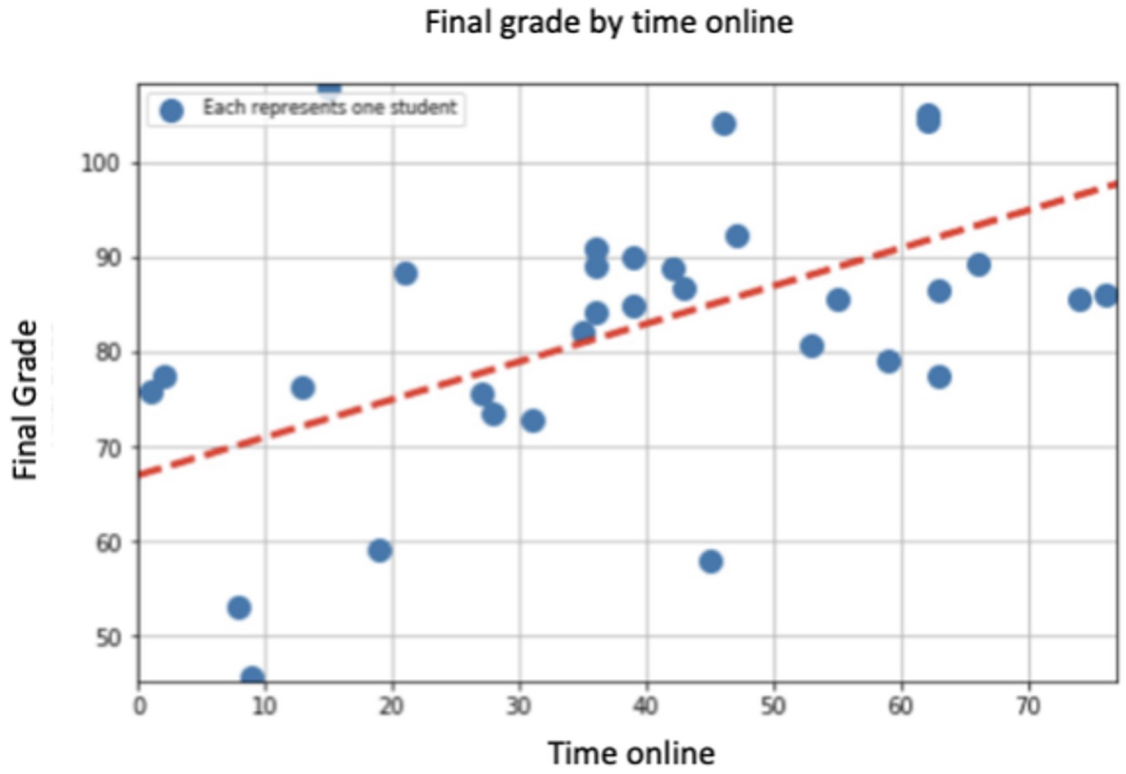
Be prepared for breakout sessions using in-class demo notebooks - have your environment and in-class demo notebooks up and running before class starts (you do not have to complete them).

Learn from each other: Ask and answer questions on Slack, but do not share code/answers

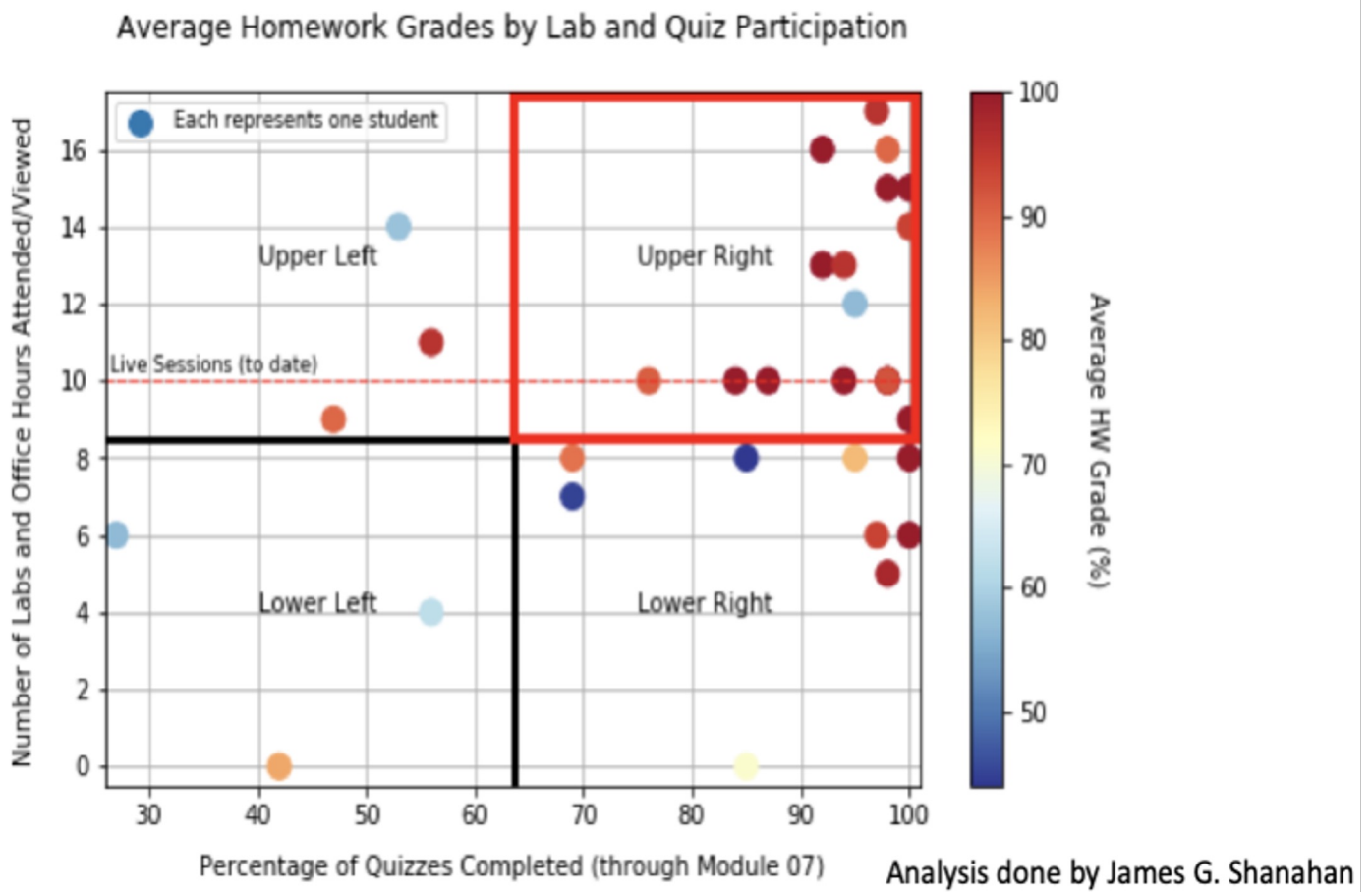
Come to office hours

Complete Homework on time

Online masters students final grades analysis



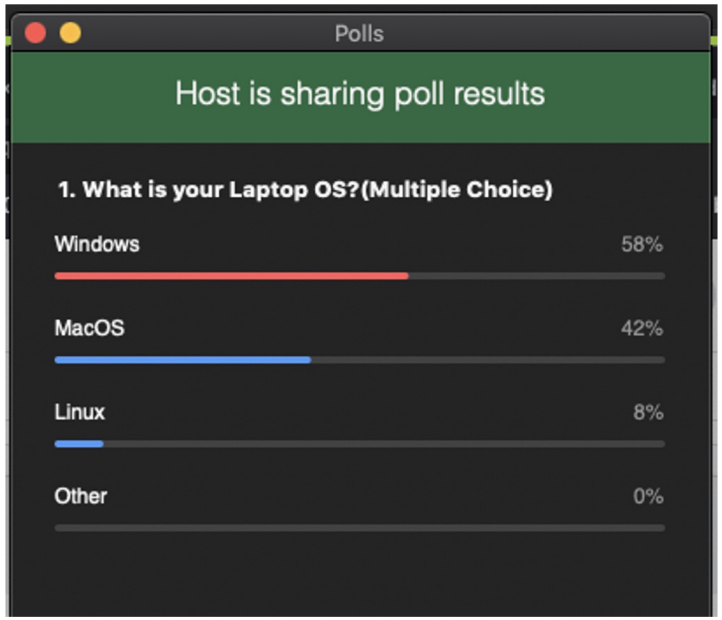
Analysis done by James G. Shanahan


Average Homework Grades by Lab and Quiz Participation
Each represents one student
Upper Left
Upper Right
Lower Left
Lower Right
Live Sessions (to date)
Number of Labs and Office Hours Attended/Viewed
Percentage of Quizzes Completed (through Module 07)
Average HW Grade (%)
Analysis done by James G. Shanahan

Things to Know

- DigitalCampus (Canvas) is the source of all truth

- As for infrastructure, we will be using
  - Google Cloud (DataProc) until week 10
  - Google Colab (for Spark)
  - Databricks (running on Azure)
    - Access Instructions will be provided around week 10
    - Required the final project

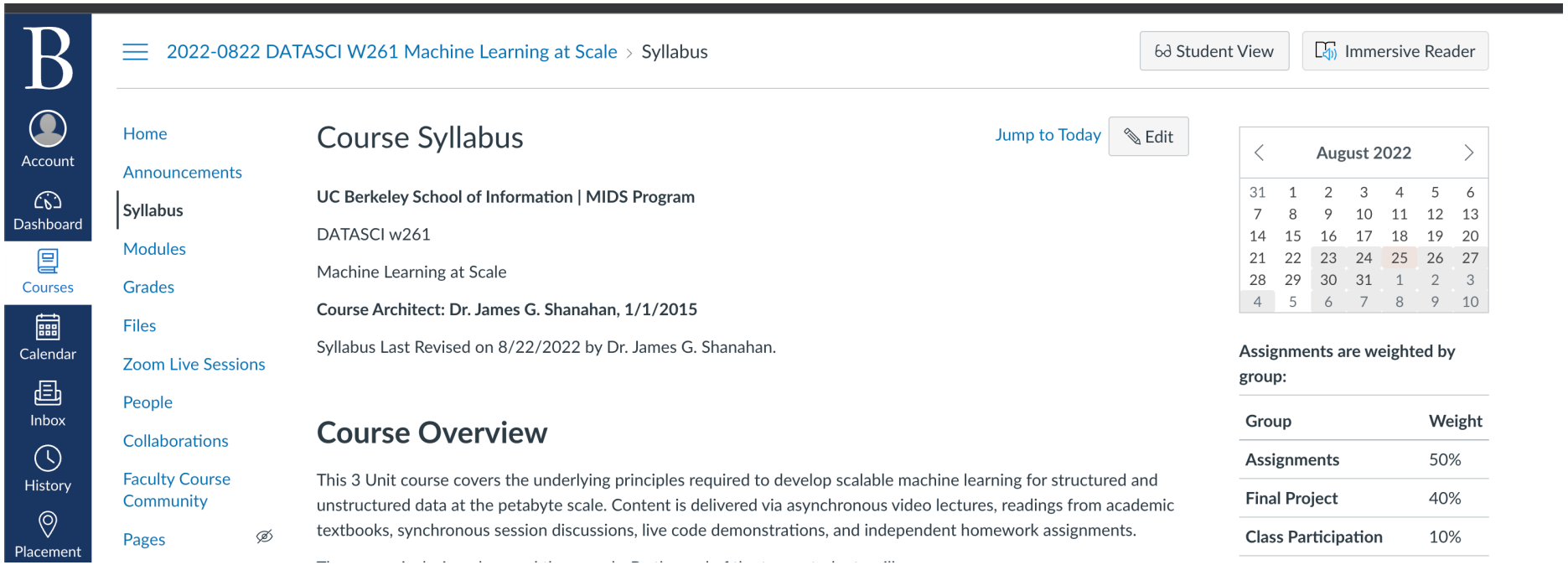


Important Dates

See Canvas

Syllabus

- https://digitalcampus.instructure.com/courses/4868/assignments/syllabus



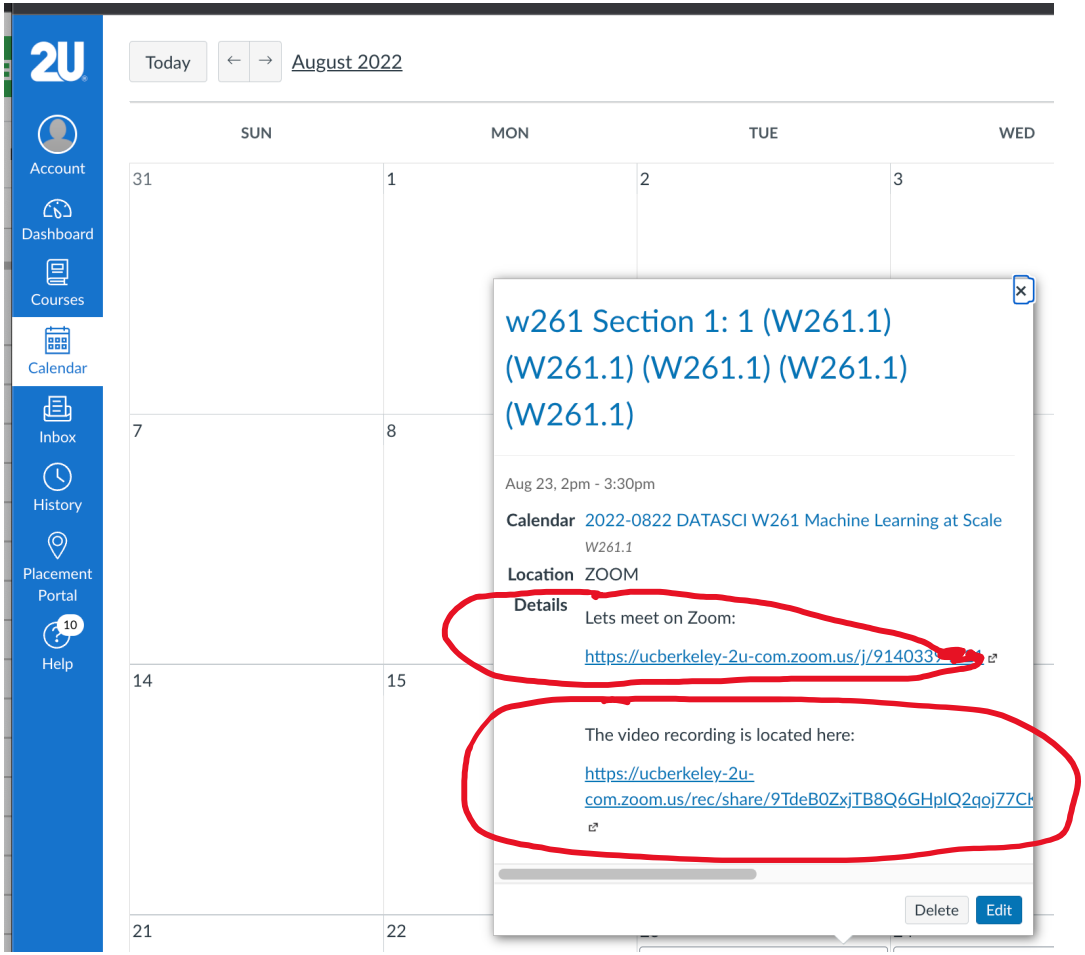
Section 1: intro to ML, Map Reduce, Naïve Bayes

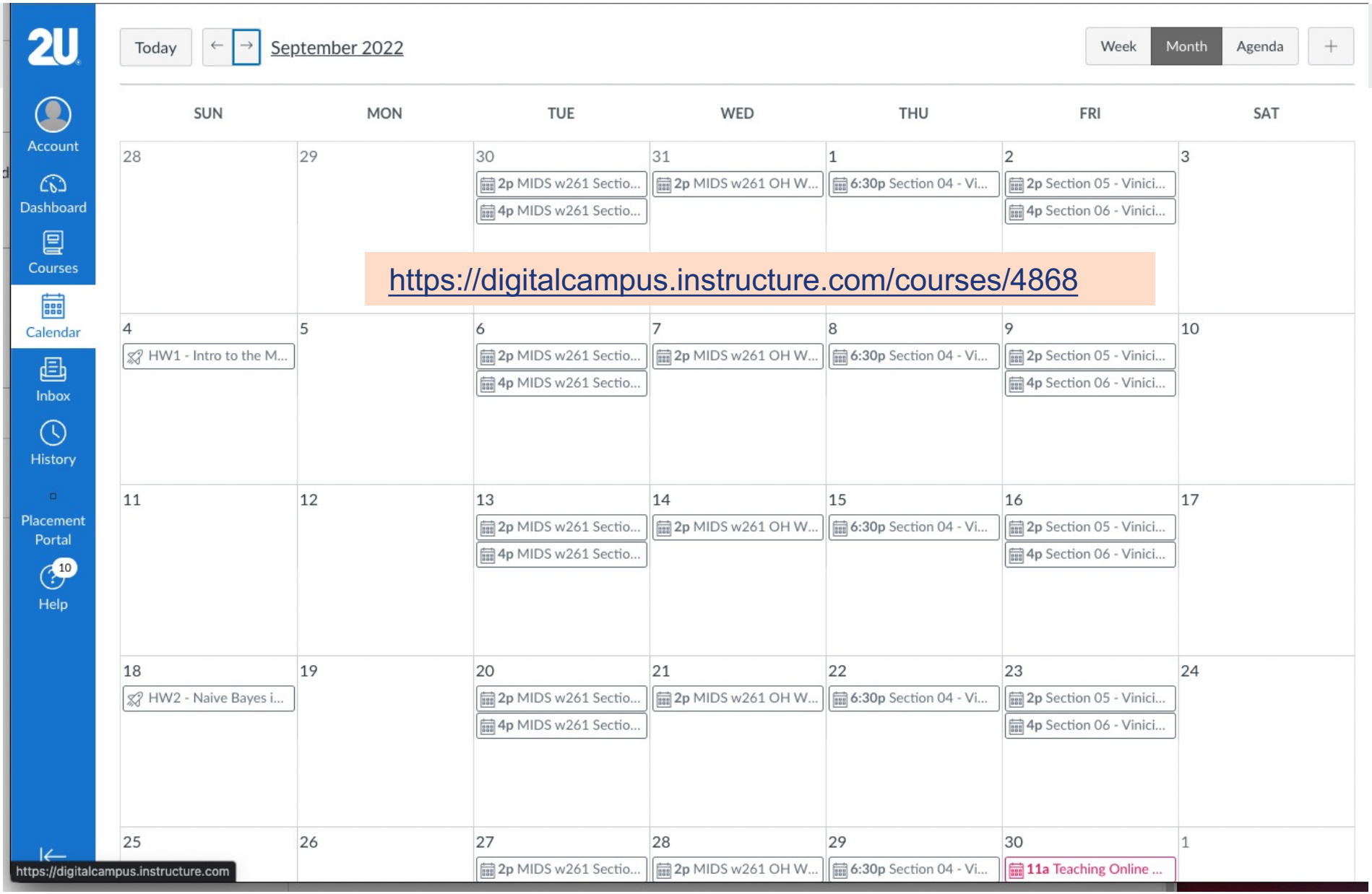
Module	Section	Async Lecture	Sync Lab + Review quiz	Assignments	Reading Materials
1	**Intro and review**	Machine Learning at Scale introduction and course overview	Command-line-based map-reduce, Google Cloud		Read: ISL chapter 1 and sections 2.1 & 2.2 Skim: Adam Drake Blog Post (Links to an external Optional: Fortmann-Roe Essay Clever Machine Bl Live Lab 1 SLIDES: See Module TAB
2		Introduction to Hadoop Streaming	WordCount, secondary sorts, Naive Bayes	**HW01 MapReduce via CMDLine due Midnight Sunday (of week 2)**	Read: DITP Chapter 1 & Chapter 2 Read: IIR CH.13 Optional: Michael Noll Hadoop MR Tutorial
3		Map-Reduce Algorithm Design patterns and map-shuffle-reduce	Relative frequencies, custom partitioning, order inversion, inverted index		Read: DITP sections 2.4 - 2.7 and 3.1-3.4 (Links t Read: IIR sections 13.1 and 13.2 Read: HDG - Part II Chapter 7 - How MapReduce Skim: Total Order Sort Guide, EECS Map Reduce I OPTIONAL: http://blog.ditullio.fr/categor Slides: Lab Notebook is self-contained

Section 2: intro to Spark

4	**Spark**	Intro to Spark/Map-Reduce with RDDs (part 1)	Transformations, actions, RDDs, dataframes, datasets, broadcasting, closures	**HW02 (Hadoop NB) due Midnight Sunday.**	Read: HP Spark chapter 2 Read: Spark RDD Programming Guide Skim: Learning Spark ch 3 & 4 Skim: DISCO Paper Skim: DocSim Paper Additional Spark resources for weeks 4, and 5 • Holden Karau - Spark summit 2017https://ww v=4xsBQYdHgn8&feature=youtu.be [40 minu • Debugging Spark Holden Karau -Dec 2017htt v=s5p15QT0Zj8&list=WL&index=7&t=0s [45
5		Intro to Spark/Map-Reduce with RDDs (part 2)	KMeans Clustering, pairs and stripes		• Watch: Debugging Spark - Holden Karau - Dec 2017 [45 • Skim Chapter 16 in IIR Book

Live Sessions and their corresponding recordings




2U
Account
Dashboard
Courses
Calendar
Inbox
History
Placement Portal
Help
10
Today
September 2022
Week
Month
Agenda
SUN
MON
TUE
WED
THU
FRI
SAT
https://digitalcampus.instructure.com/courses/4868
https://digitalcampus.instructure.com

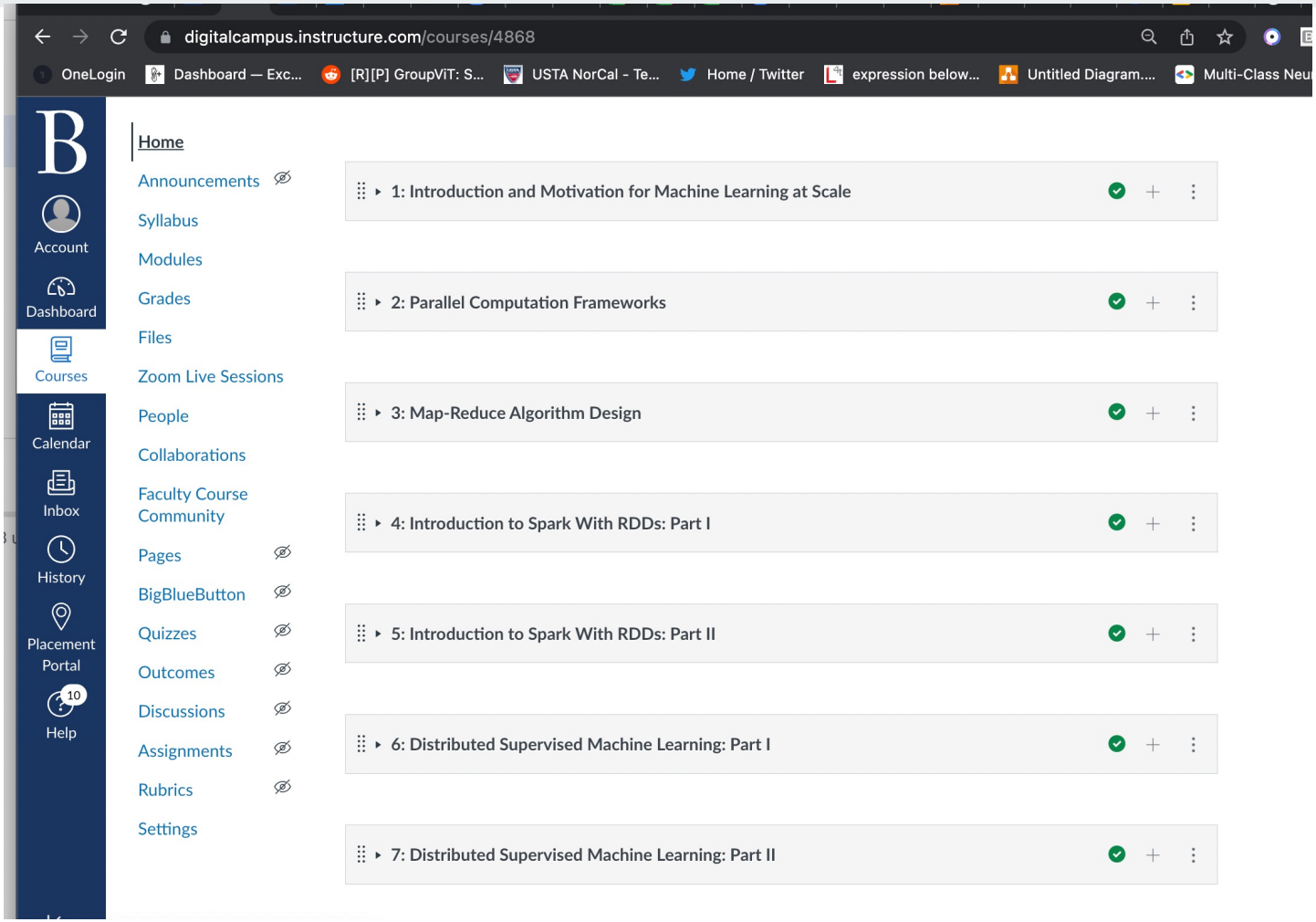
HW Schedule

5 HWs

And a Final project with multiple phases

https://www.ischool.berkeley.edu/intranet/students/mids/calendar

Modules



https://digitalcampus.instructure.com/courses/4868

Mastery based grading

We follow an approach called mastery based grading which is a little different if you've never encountered it before. Basically for each question the score you get is not an indication of a "percent of the sub questions you got right" instead you should think of it as a categorical indicator. **Each question gets an overall score regardless of number of subquestions/parts.**

Each question has some one or two core concepts plus a lot of details. The goal is to avoid docking students multiple times for little details but still ensure that overall you get a clear message if you miss an important concept - so anything that shows confusion/error around the core learning concept gets a 50 while any other error or combination of errors gets a 90.

100 - illustrates understanding of core concepts, no errors
90 - illustrates understanding of core concepts, some errors in calculations, typos, etc
50 - core concepts missed
0 - not attempted

Grades breakdown

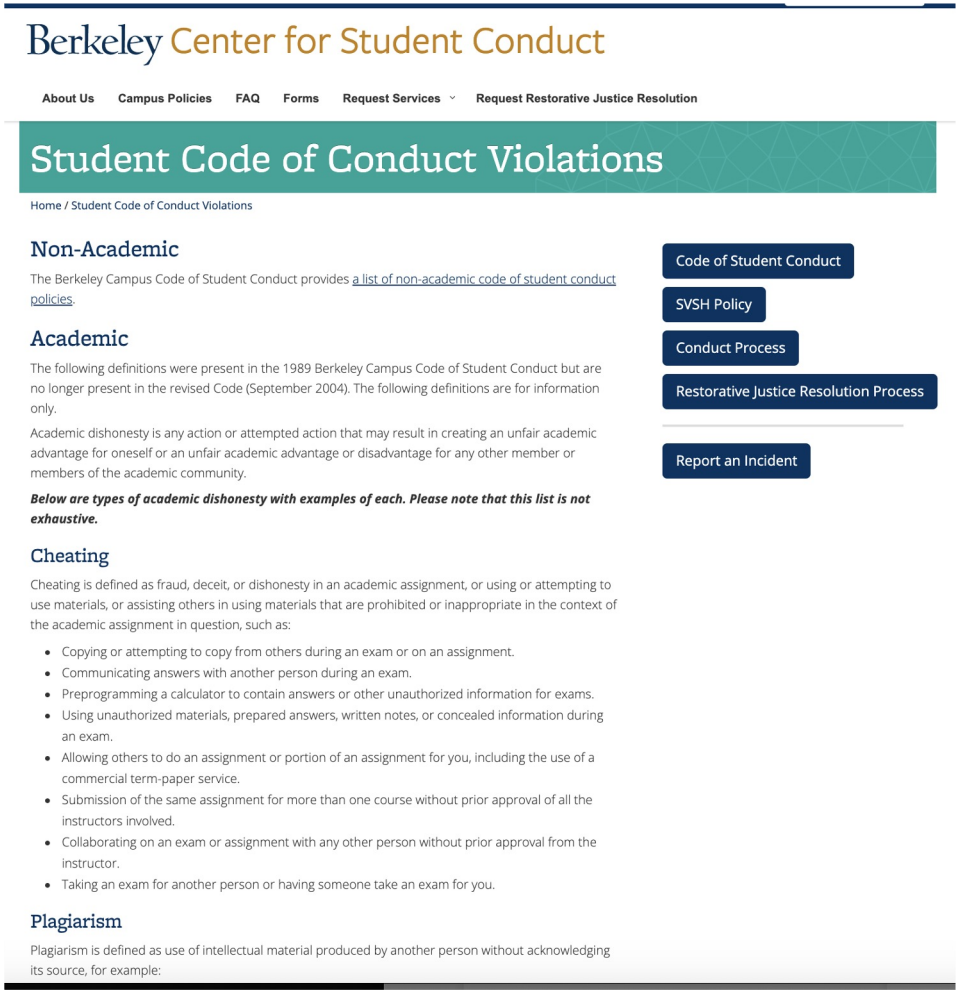
% of Final Grade	Component
50%	Homework Assignments (5 assignments, 10% each)
40%	Final Project
10%	Live session attendance and participation

Final Letter Grades breakdown for w261

Letter grades	Range		Interpretation
A+	<= 100%	to 98 %	Excellent
A	< 98%	to 92.5%	Excellent
A-	< 92.5%	to 90%	Very Good
B+	< 90%	to 82.5%	Good
B	< 82.5%	to 75%	Good
B-	< 75%	to 67.5%	Fair
C+	< 67.5%	to 57.5%	Below Expectations
C	< 57.5%	to 50%	Below Expectations
C-	< 50%	to 42.5%	Below Expectations
D+	< 42.5%	to 32.5%	Below Expectations
D	< 32.5%	to 25%	Below Expectations
F	< 25%	to 0%	Below Expectations
I	Incomplete		Work incomplete due to circumstances beyond the student's control, but of passing quality; not included in grade-point computation after fall 1973.

Student Code of Conduct

https://conduct.berkeley.edu/violations/



Berkeley Center for Student Conduct

About Us | Campus Policies | FAQ | Forms | Request Services | Request Restorative Justice Resolution

Student Code of Conduct Violations

Home / Student Code of Conduct Violations

Non-Academic

The Berkeley Campus Code of Student Conduct provides a list of non-academic code of student conduct policies.

Academic

The following definitions were present in the 1989 Berkeley Campus Code of Student Conduct but are no longer present in the revised Code (September 2004). The following definitions are for information only.

Academic dishonesty is any action or attempted action that may result in creating an unfair academic advantage for oneself or an unfair academic advantage or disadvantage for any other member or members of the academic community.

Below are types of academic dishonesty with examples of each. Please note that this list is not exhaustive.

Cheating

Cheating is defined as fraud, deceit, or dishonesty in an academic assignment, or using or attempting to use materials, or assisting others in using materials that are prohibited or inappropriate in the context of the academic assignment in question, such as:

- Copying or attempting to copy from others during an exam or on an assignment.
- Communicating answers with another person during an exam.
- Preprogramming a calculator to contain answers or other unauthorized information for exams.
- Using unauthorized materials, prepared answers, written notes, or concealed information during an exam.
- Allowing others to do an assignment or portion of an assignment for you, including the use of a commercial term-paper service.
- Submission of the same assignment for more than one course without prior approval of all the instructors involved.
- Collaborating on an exam or assignment with any other person without prior approval from the instructor.
- Taking an exam for another person or having someone take an exam for you.

Plagiarism

Plagiarism is defined as use of intellectual material produced by another person without acknowledging its source, for example:

Code of Student Conduct

SVSH Policy

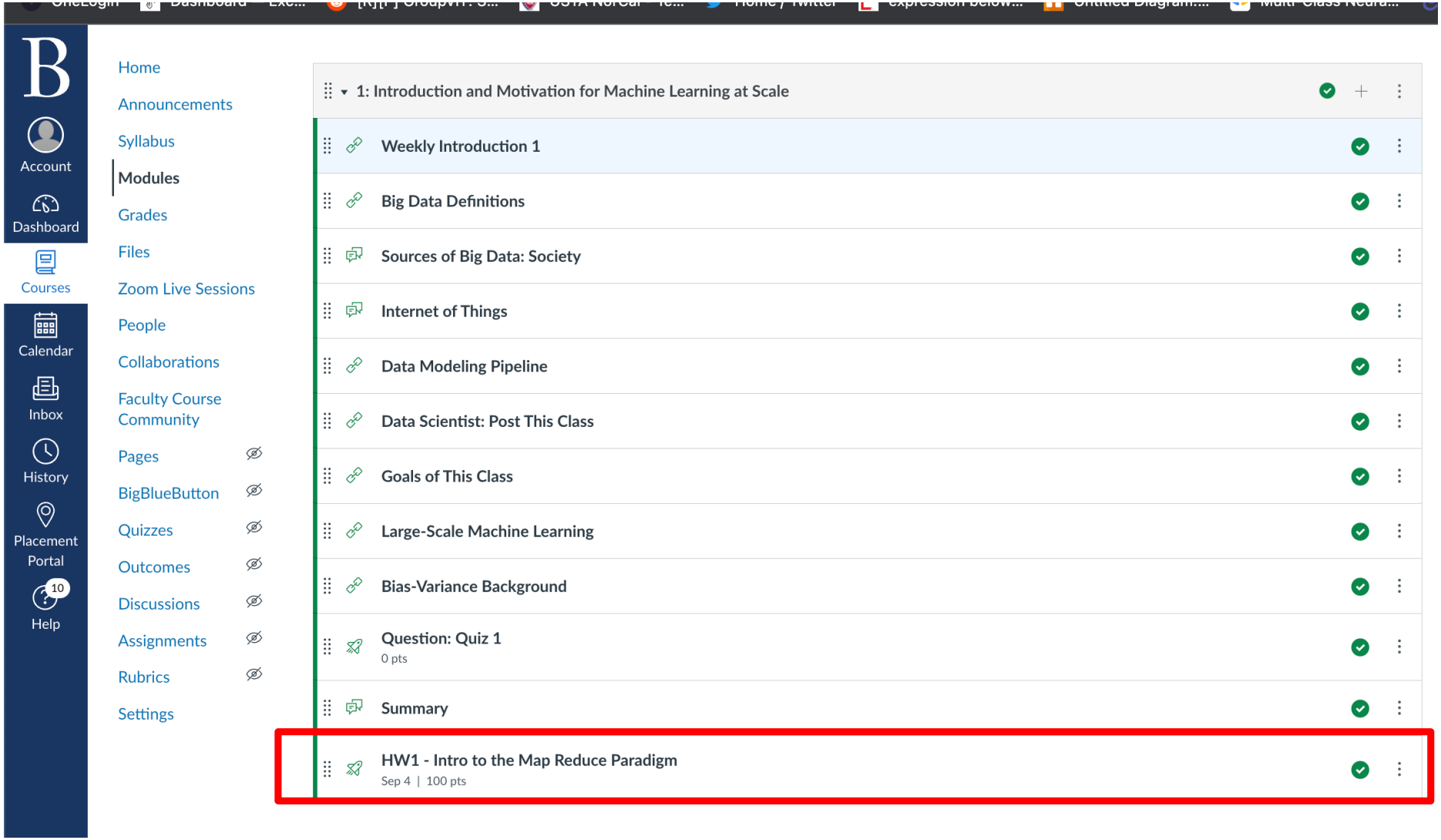
Conduct Process

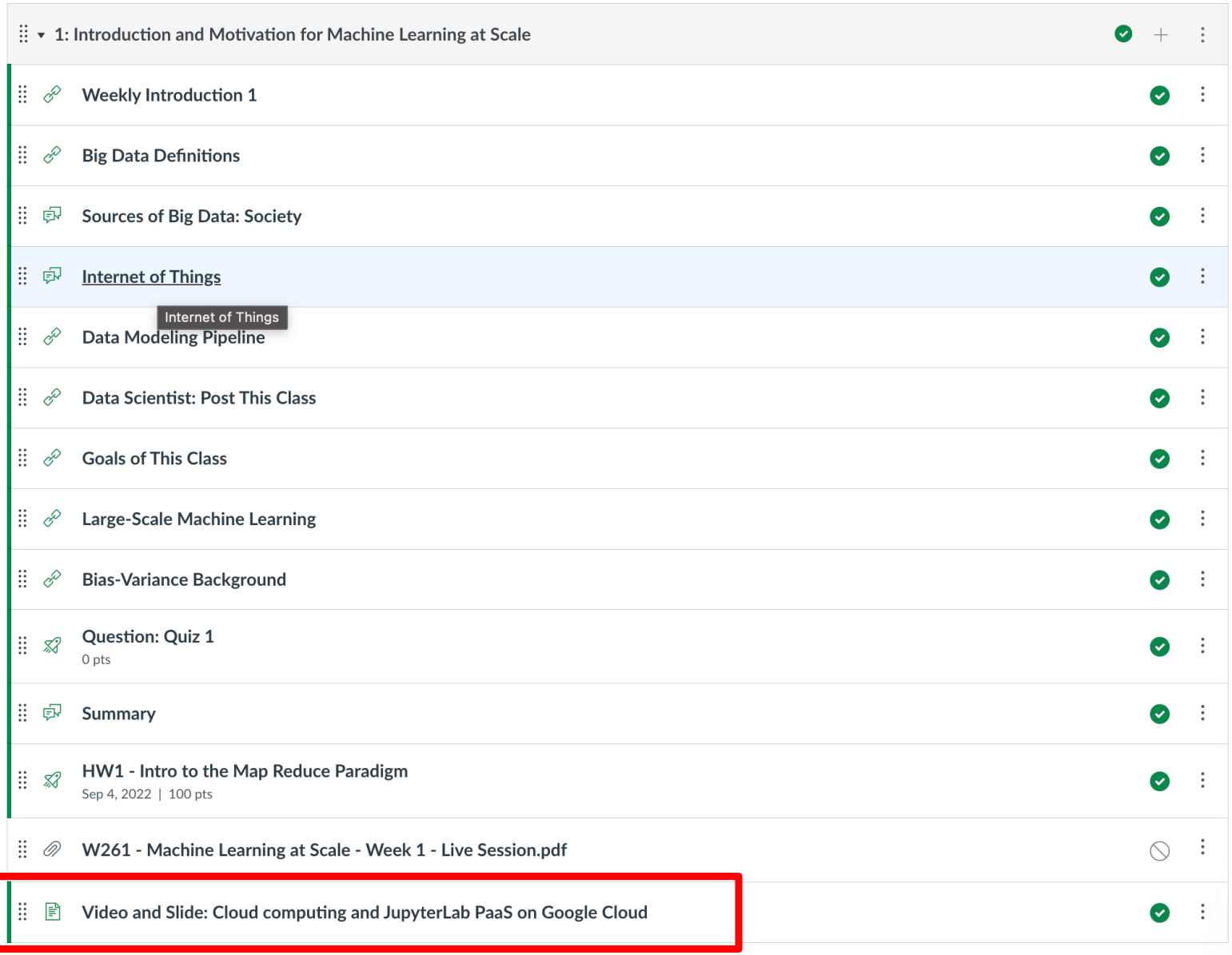
Restorative Justice Resolution Process

Report an Incident

Preview of HW 1

https://github.com/UCB-w261/main/blob/master/Assignments/HW1/hw1_Workbook.ipynb


Account
Dashboard
Courses
Calendar
Inbox
History
Placement Portal
Help
10
Home
Announcements
Syllabus
Modules
Grades
Files
Zoom Live Sessions
People
Collaborations
Faculty Course Community
Pages
BigBlueButton
Quizzes
Outcomes
Discussions
Assignments
Rubrics
Settings
1: Introduction and Motivation for Machine Learning at Scale
Weekly Introduction 1
Big Data Definitions
Sources of Big Data: Society
Internet of Things
Data Modeling Pipeline
Data Scientist: Post This Class
Goals of This Class
Large-Scale Machine Learning
Bias-Variance Background
Question: Quiz 1
0 pts
Summary
HW1 - Intro to the Map Reduce Paradigm
Sep 4 | 100 pts


1: Introduction and Motivation for Machine Learning at Scale
Weekly Introduction 1
Big Data Definitions
Sources of Big Data: Society
Internet of Things
Internet of Things
Data Modeling Pipeline
Data Scientist: Post This Class
Goals of This Class
Large-Scale Machine Learning
Bias-Variance Background
Question: Quiz 1
0 pts
Summary
HW1 - Intro to the Map Reduce Paradigm
Sep 4, 2022 | 100 pts
W261 - Machine Learning at Scale - Week 1 - Live Session.pdf
Video and Slide: Cloud computing and JupyterLab PaaS on Google Cloud

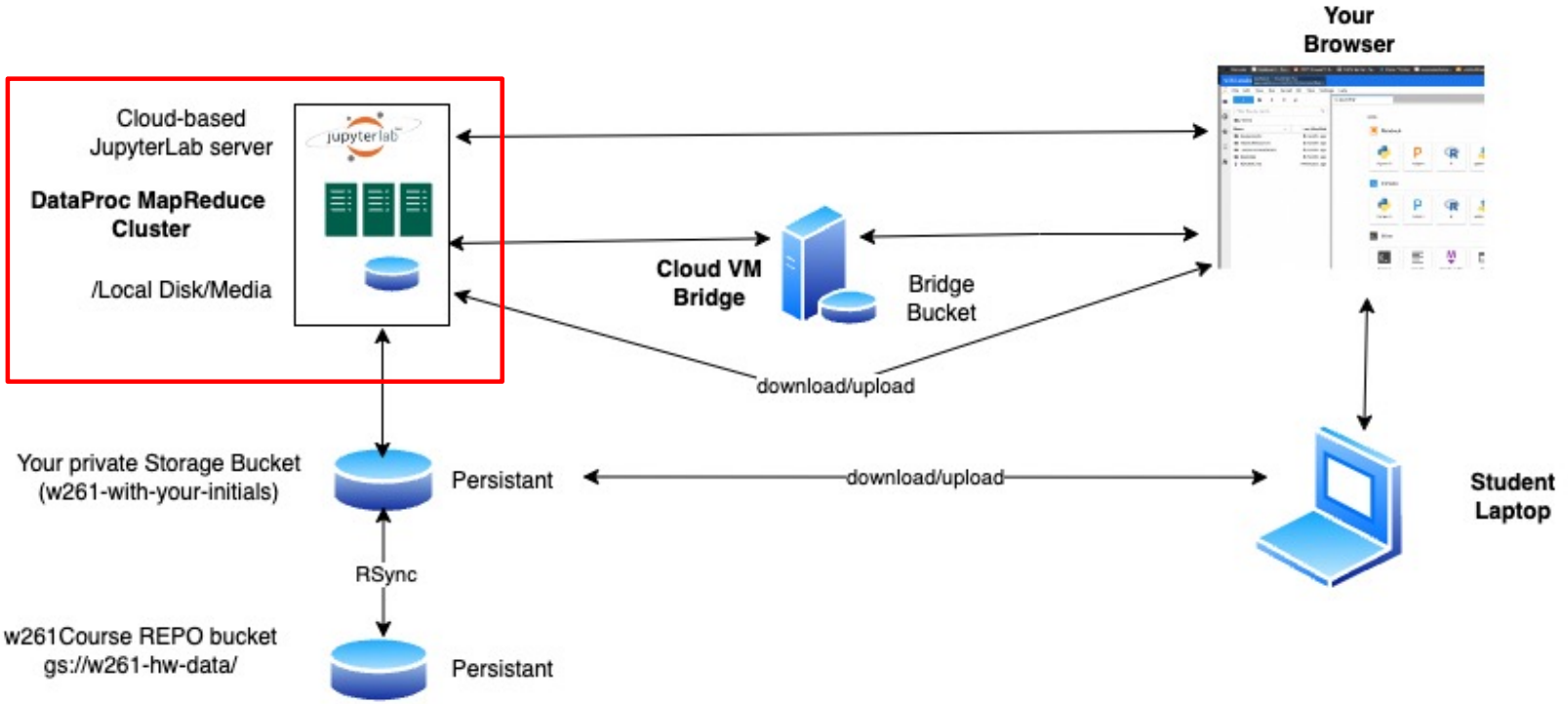
DataProc

JupterLab PaaS at $0.20 per hour

Dataproc is a fully managed and highly scalable service for running Hadoop, Apache Spark, Apache Flink, Presto, and 30+ open source tools and frameworks.

Setup a hadoop/Spark (aka DataProc) cluster on Google cloud with a **single node** (4 CPUs)
The cluster costs $0.20 per hour and has 100 Gig of local Disk

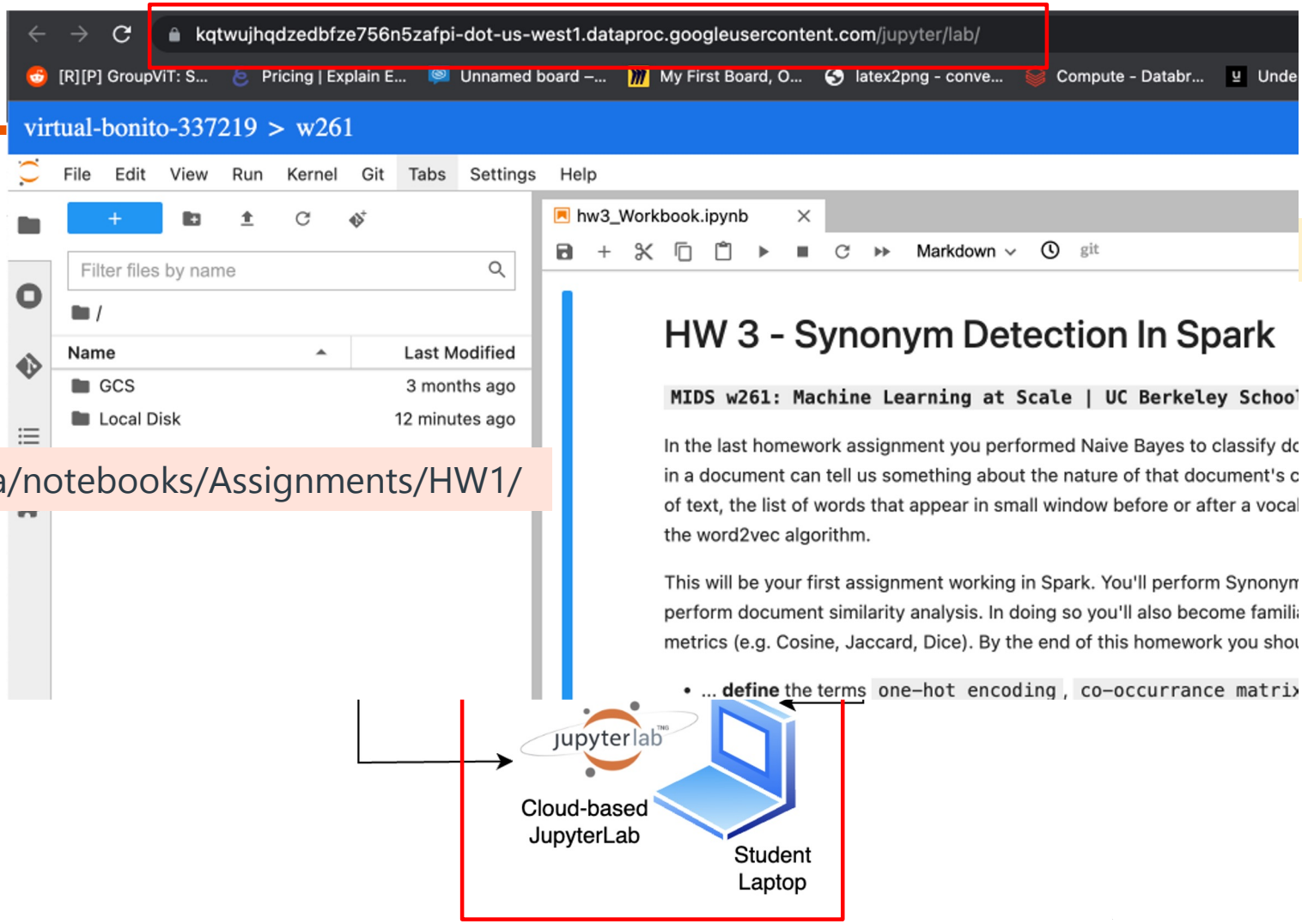
~$35 per week
~ $300 for 9 weeks



Click here to access CloudShell/Bridge: https://console.cloud.google.com/

Orchestration shell script cmd: gsutil cat gs://w261-hw-data/w261_env.sh | bash -euo pipefail

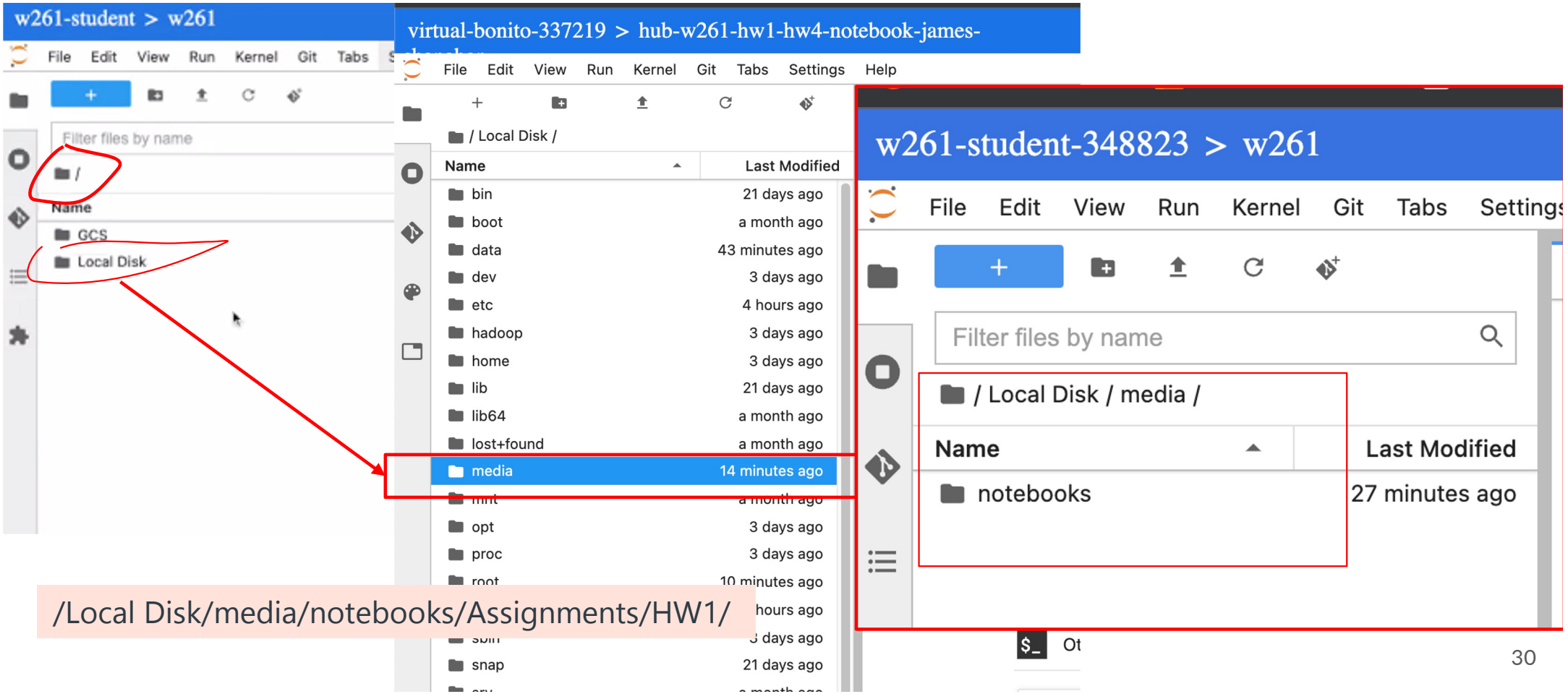
Cloud-based JupterLab (backed by a DataProc Cluster)



/Local Disk/media/notebooks/Assignments/HW1/

```
bash -euo pipefail
```

Click on / folder and select Local Disk

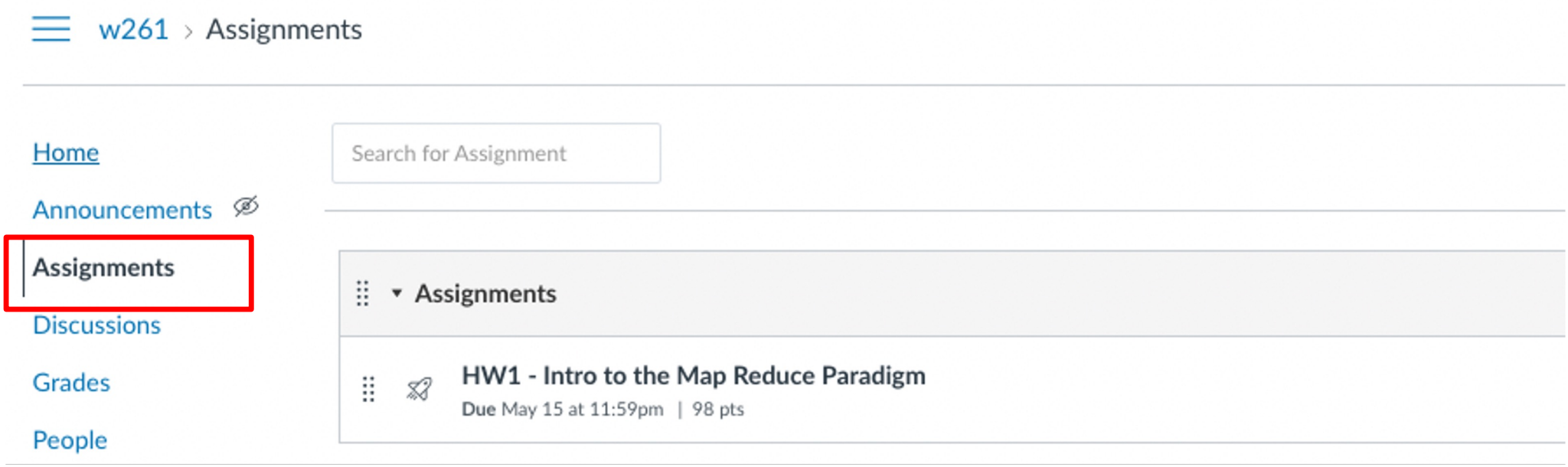


/Local Disk/media/notebooks/Assignments/HW1/

Directory structure for HW

```
W261-f21-jimi  # course-semester-mygitid
    --Assignments
        --HW1
            |__hw1_Workbook.ipynb ← change the name of the notebook to avoid collisions
            |__collateCounts.py
            |__pWordCount.sh
            |__wordCount.py
            |__README.md
            |__ <directory for scratchwork>
        --HW2
            |__ etc ...
     --HelpfulResources
     etc..
```

HW submission via Canvas (LMS)



HW1 notebook

Question 1: Introductions

```
The Caterpillar and Alice looked at each other for some time in silence: at last the Caterpillar took the hookah out of its mouth, and addressed her in a languid, sleepy voice. "Who are you?" said the Caterpillar.
```

-- Lewis Carroll, _Alice's Adventures in Wonderland_, Chapter 4

Tell us about yourself! Briefly describe where you live, how far along you are in MIDS, what other classes you are taking and what you want to get out of w261.

> Type your response here!

Question 2: Bias - Variance

In 1-2 paragraphs (~200 and *absolutely no more* than 300 words!), explain the bias-variance trade off. Describe what it means to "decompose" sources of error. How is this used in machine learning? Please use mathematical equation(s) to support your explanation. (Use `$` signs to take advantage of $\LaTeX$ formatting, eg. `$f(x)$` will look like: $f(x)$). Please also cite any sources that informed your answer.

> Type your response here!

Question 3: Tokenizing

A number of our assignments this term will involve extracting information from text. A common preprocessing step when working with raw files is to 'tokenize' (i.e. extract words from) the text. Within the field of Natural Language Processing a lot of thought goes into what specific tokenizing makes most sense for a given task. For example, you might choose to remove punctuation or to consider punctuation symbols 'tokens' in their own right. **In this question you'll use the Python `re` module to create a tokenizer to use when you perform WordCount on the *Alice In Wonderland* text.**

Q3 Tasks:

- **a) short response:** In the Naive Bayes algorithm (which we'll implement next week), we'll estimate the *likelihood* of a word by counting the number of times it appears and dividing by the size of the vocabulary (total number of unique words). Using the text: *"Alice had an adventure that took alice to wonderland"*, give a concrete example of how two different tokenizers could cause us to get two different results on this calculation. [`HINT` : *you should not need to read up on Naive Bayes to answer this question*]
- **b) short response:** When tokenizing in this assignment we'll remove punctuation and discard numerical digits by making everything lowercase and then capturing only consecutive letters a to z. Suppose **`tokenize(x)`** is a Python function that performs the desired tokenization. What would **`tokenize("By-the-bye, what became of Alice's 12 hats?!")`** output? Type the answer in the space provided below.
- **c) code:** Fill in the regular expression pattern in the cell labeled `part c` so that the subsequent call to `re.findall(RE_PATTERN, ...)` returns the

Question 1: Introductions

```
The Caterpillar and Alice looked at each other for some time in silence: at last the Caterpillar took the hookah out of its mouth, and addressed her in a languid, sleepy voice. "Who are you?" said the Caterpillar.
```

-- Lewis Carroll, *Alice's Adventures in Wonderland*, Chapter 4

Tell us about yourself! Briefly describe where you live, how far along you are in MIDS, what other classes you are taking and what you want to get out of w261.

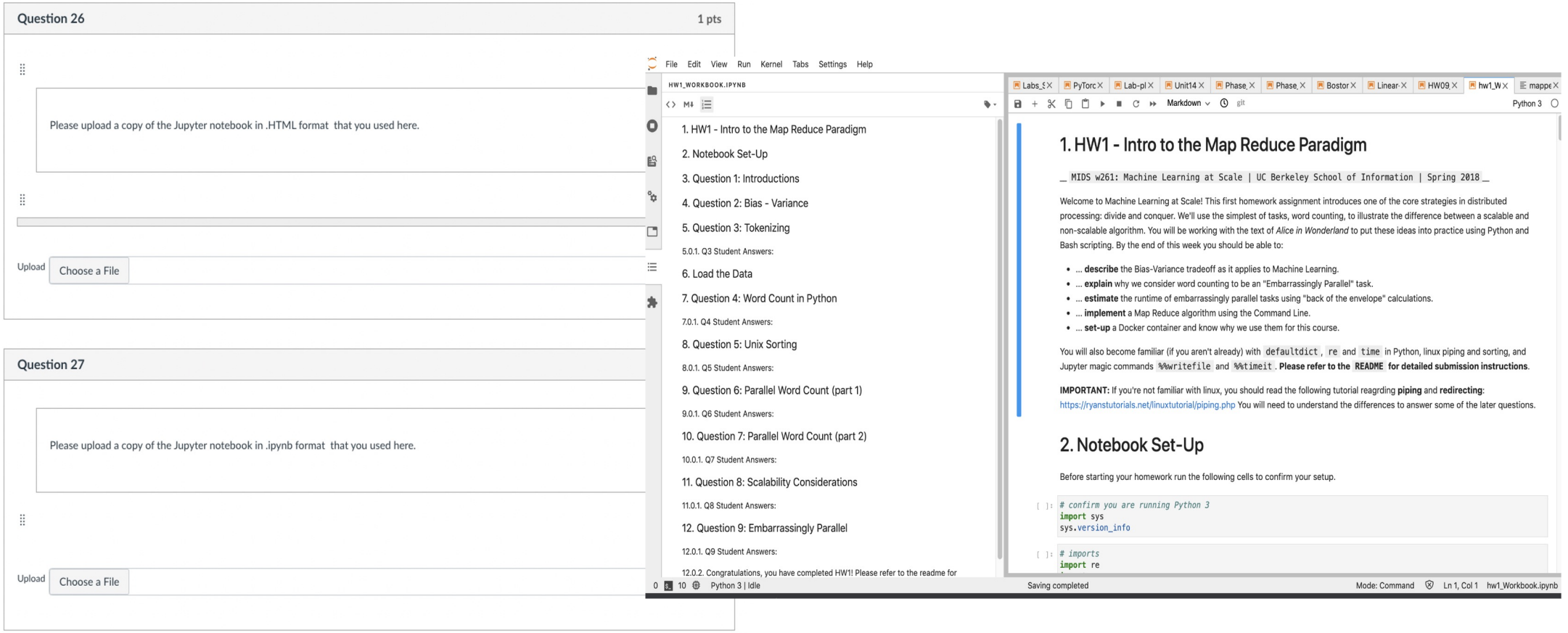
Question 2: Bias - Variance

In 1-2 paragraphs (~200 and *absolutely no more* than 300 words!), explain the bias-variance trade off. Describe what it means to "decompose" sources of error. How is this used in machine learning? Please use mathematical equation(s) to support your explanation. (Use `$` signs to take advantage of $LaTeX$ formatting, eg. `$f(x)$` will look like: $f(x)$. Please also cite any sources that informed your answer.

Question 3: Tokenizing Pick 3 questions, 5 pts per question

3.a

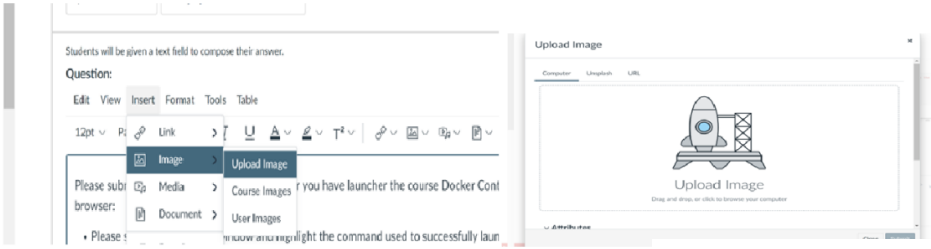
Canvas form + Jupyter Notebook



HomeWork Assignments: Code Submission Guidelines

For HW questions that require for code snippets, please submit a screenshot of the code cell from your Jupyter notebook using the guidelines provided.

- Please make sure to **submit screenshots of code cells with line numbers on.**
  - To enable line numbers within a Jupyter notebook:
    - Click **View --> Show Line Numbers** in your Jupyter Lab.
    - Line numbers are mandatory and **zero points** can be be awarded if line numbers are missing.
- **To upload an image as part of a HW answer please follow the steps outlined here:**
  - Click on Insert -> Image -> Upload Image -> Drag and drop or browse your computer:



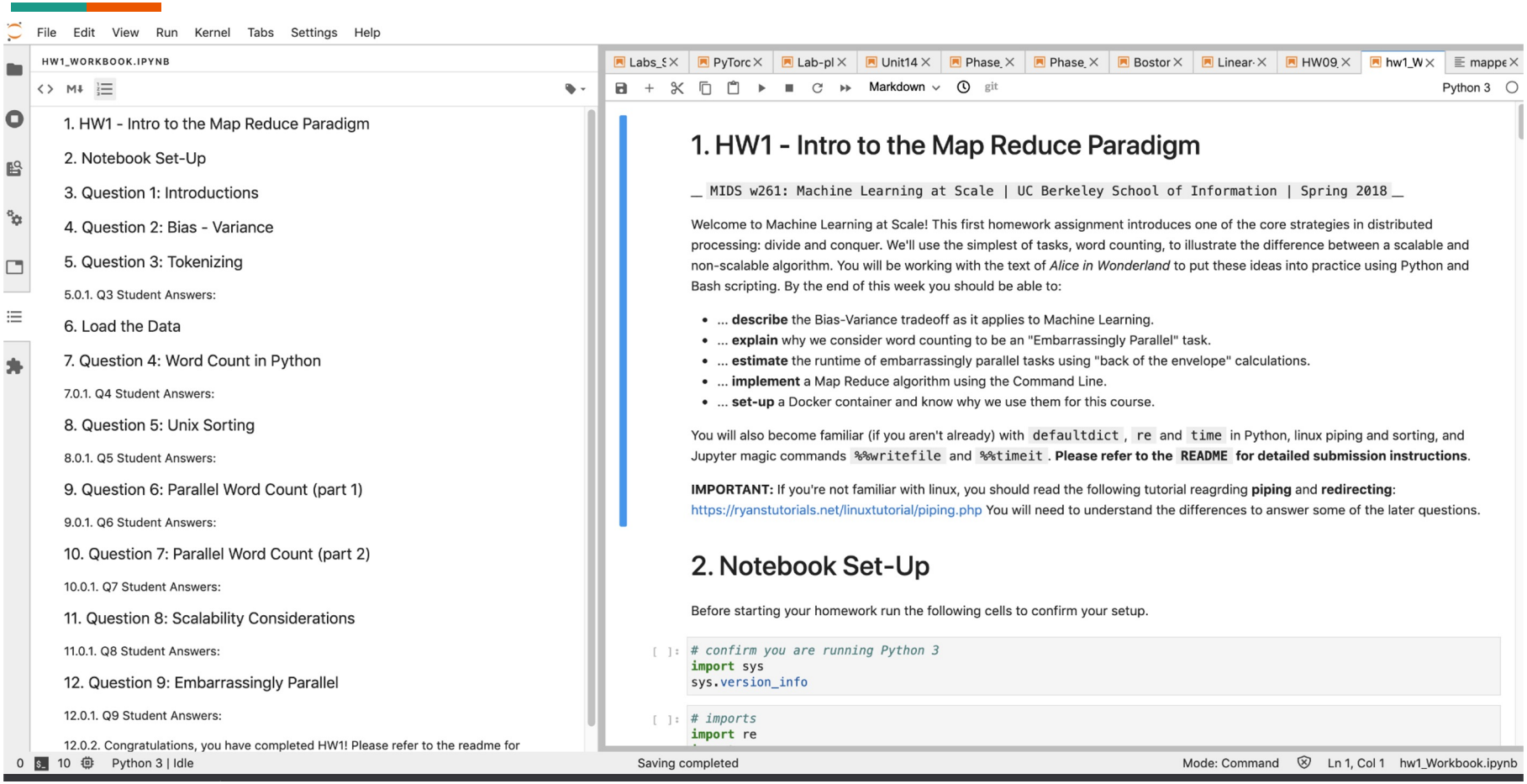
Guidelines for creating screenshots

- Please make sure that the image is properly uploaded and you can VIEW t the screenshots are missing.
- To enable line numbers within Jupyter:
  - Click **View --> Show Line Numbers** in your Jupyter Lab Notebook.
  - Line numbers are mandatory and **zero points** can be be awarded if line

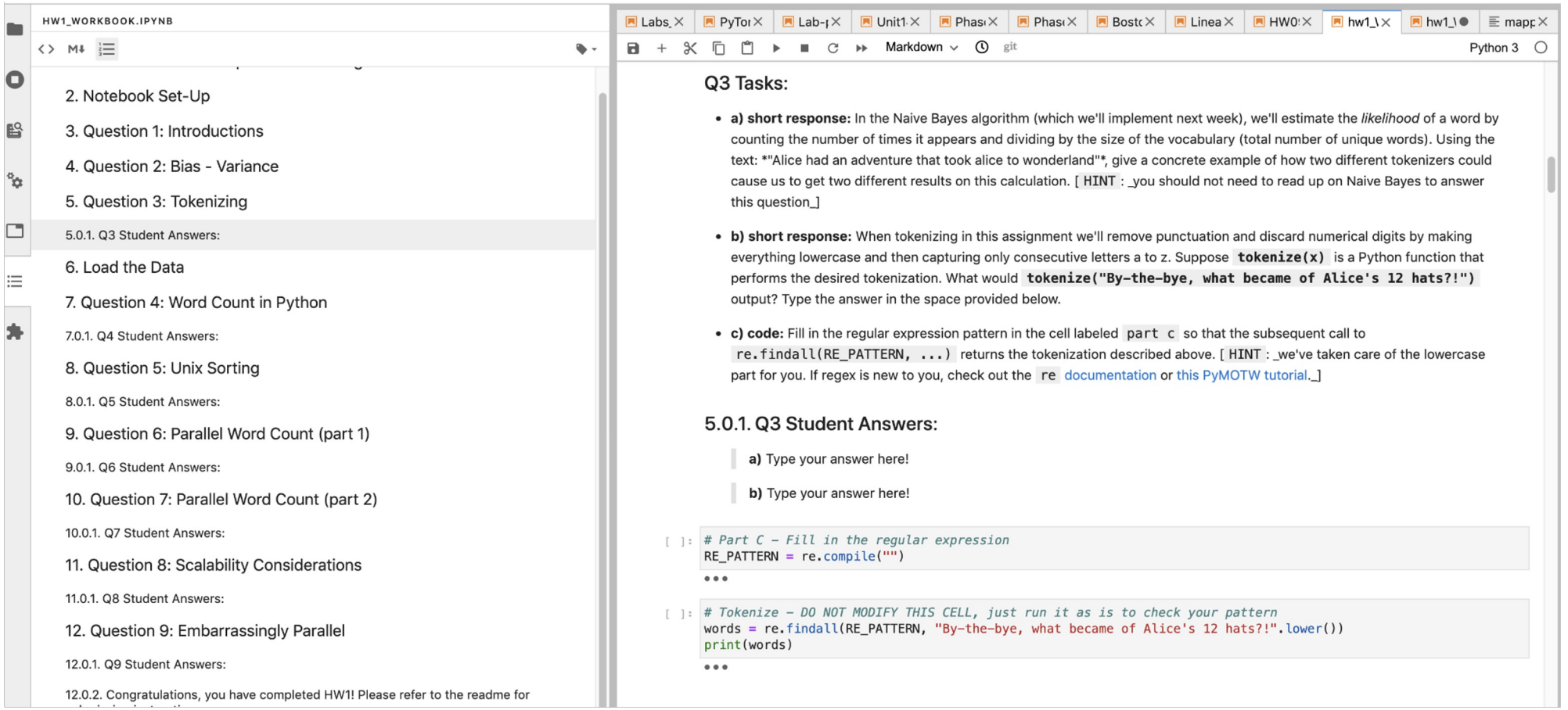
```
1
2  #=====================================================#
3  #                Your code starts here                #
4  #=====================================================#
5
6  # TODO:
7  def step(x, eta):
8      update = eta * .... #Insert f prime here
9      return x - update
10
11 step(x=..., eta=..)
12
13 #=====================================================#
14 #                Your code ends here                  #
15 #=====================================================#
```

```
1
2  #=====================================================#
3  #                Your code starts here                #
4  #=====================================================#
5
6  # TODO:
7  def step(x, eta):
8      update = eta * .... #Insert f prime here
9      return x - update
10
11 step(x=..., eta=..)
12
13 #=====================================================#
14 #                Your code ends here                  #
15 #=====================================================#
```

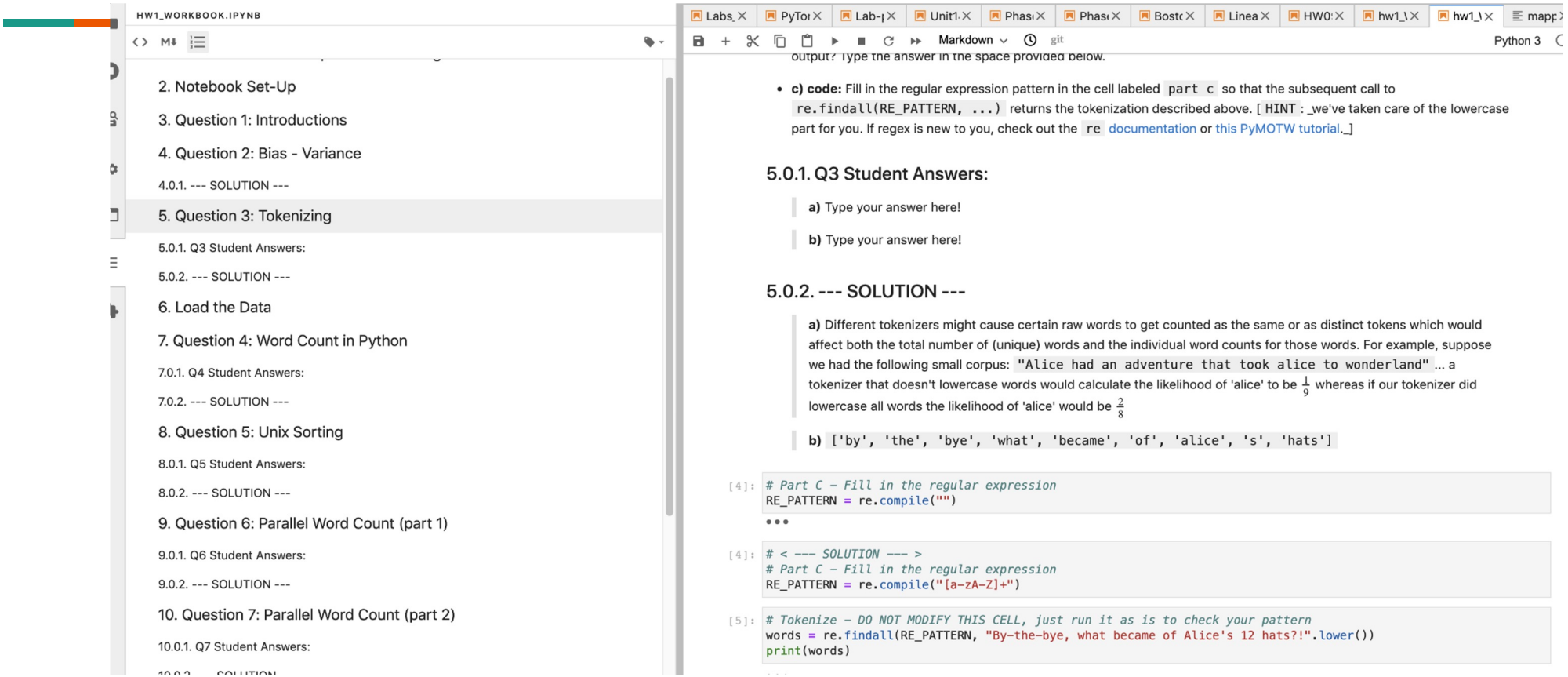
HW1 preview (9 questions)



Short response versus code questions: Q3



Sample Solution to Q3



What this course is and isn't

What the course is about

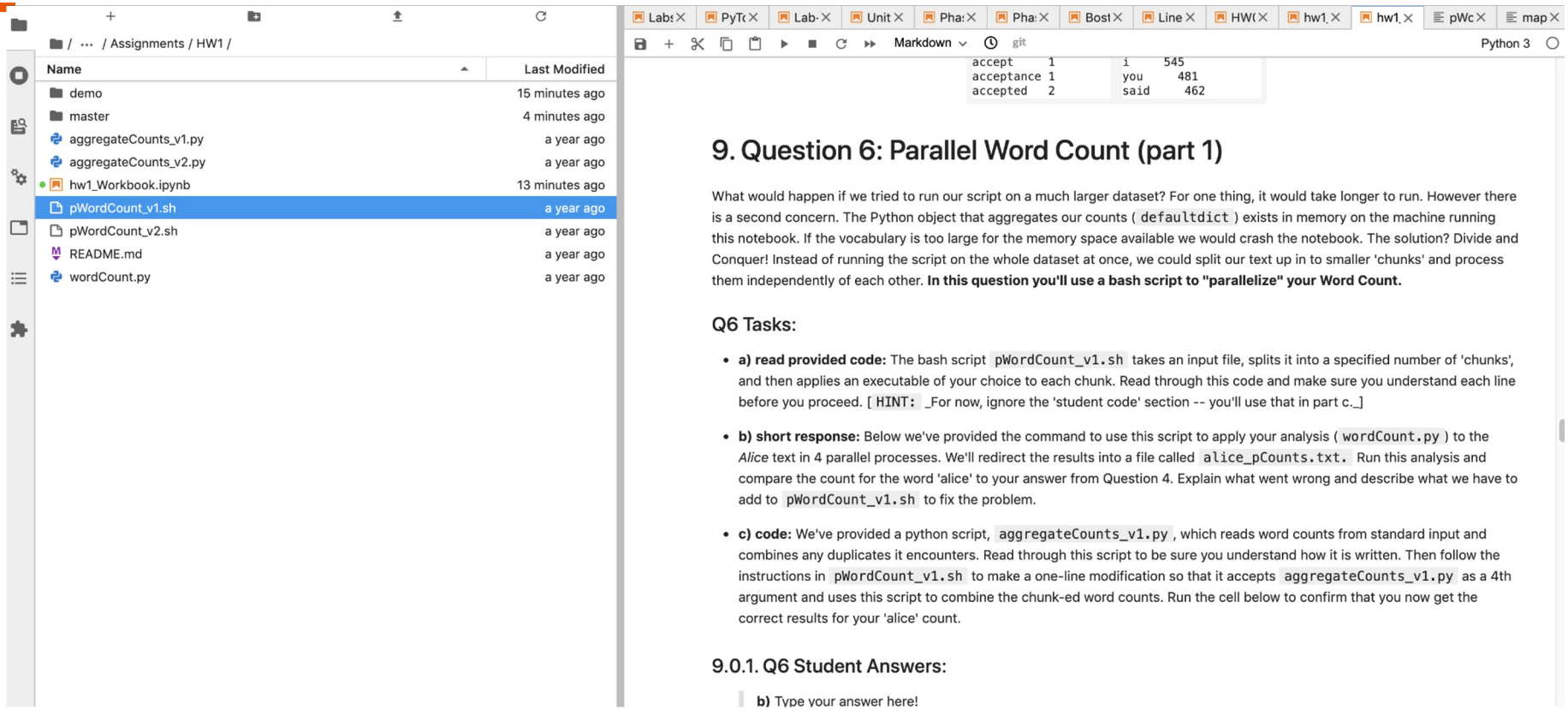
This class is focused on understanding the how and why of implementing large scale machine learning.

- Dive in to the theory of machine learning, unpack equations
- Code up the equations and heuristics on a single CPU
  - and then code them on a cluster of machines
  - Working on a realworld big data machine learning project

What it isn't about

- Spark in production
- MLOPs
- Computational learning theory
- Hard core data engineering

Starter code is provided

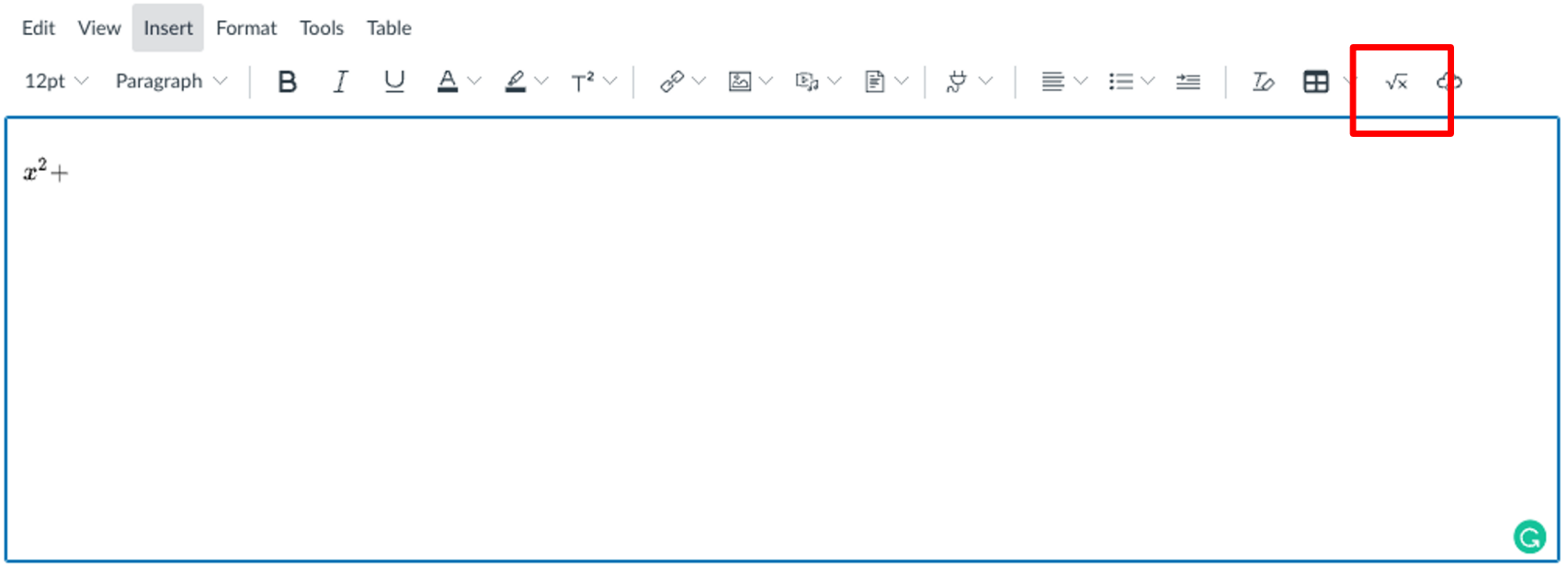


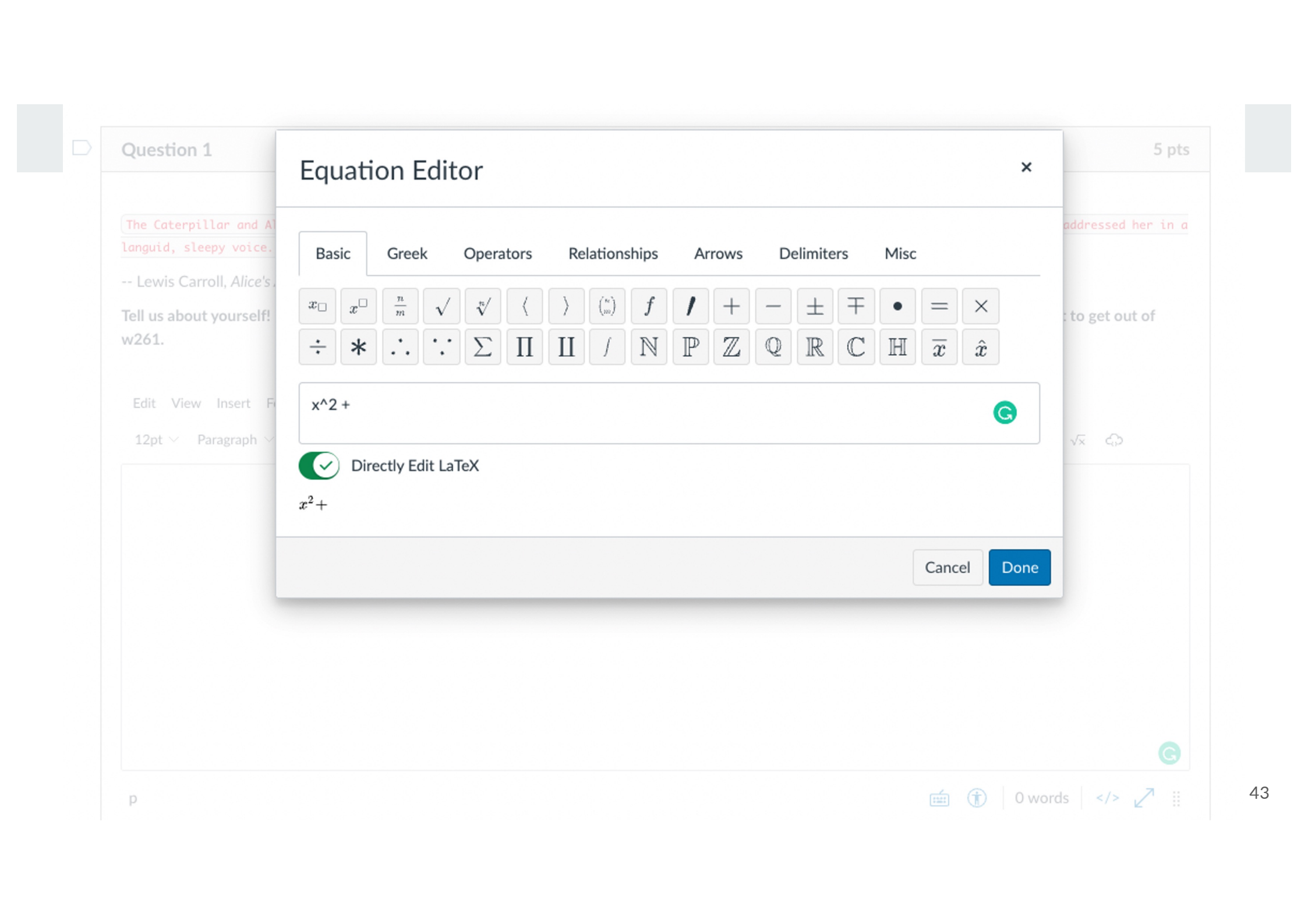
Question 1 5 pts

```
The Caterpillar and Alice looked at each other for some time in silence: at last the Caterpillar took the hookah out of its mouth, and addressed her in a languid, sleepy voice. "Who are you?" said the Caterpillar.
```

-- Lewis Carroll, *Alice's Adventures in Wonderland*, Chapter 4

Tell us about yourself! Briefly describe where you live, how far along you are in MIDS, what other classes you are taking and what you want to get out of w261.



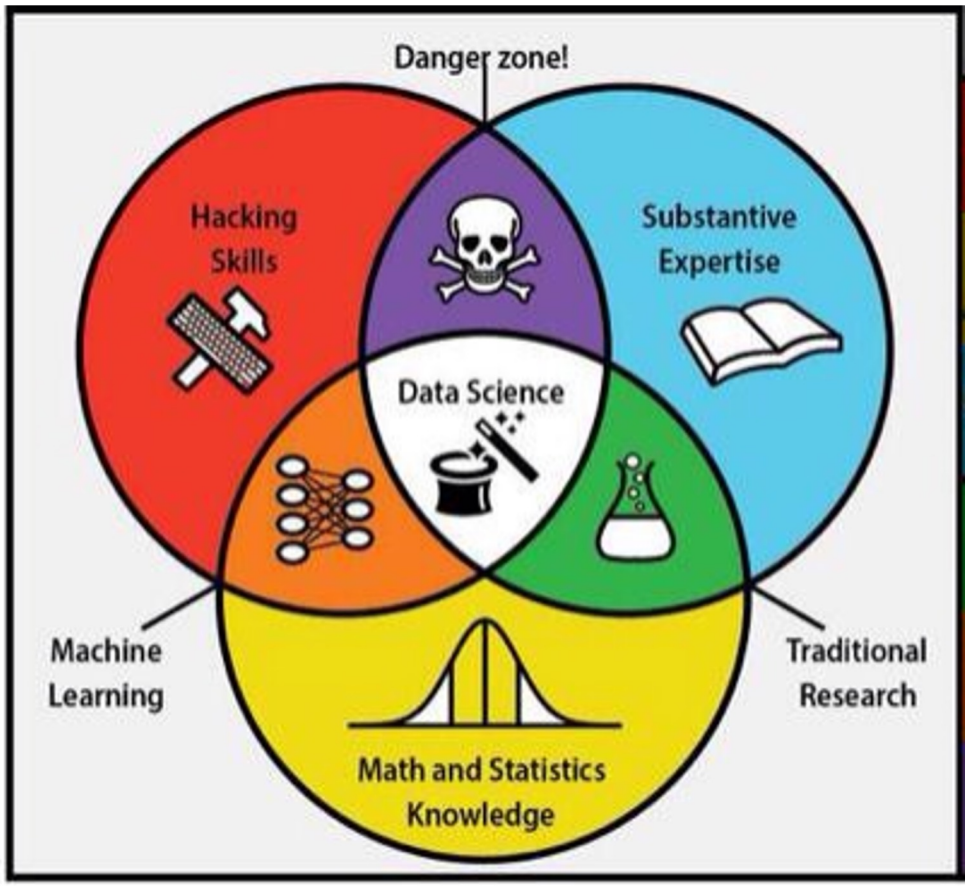

Question 1
5 pts
The Caterpillar and Al
addressed her in a
languid, sleepy voice.
-- Lewis Carroll, Alice's
Tell us about yourself!
to get out of
w261.
Edit View Insert F
12pt Paragraph
p
0 words
Equation Editor
Basic Greek Operators Relationships Arrows Delimiters Misc
x^2 +
Directly Edit LaTeX
x^2+
Cancel
Done

Goals of the Course

Tooling

- ## Hadoop, MapReduce, Spark
  - Where did Hadoop come from?
  - Understanding the MapReduce paradigm
  - What are the limitations of Hadoop MapReduce?
  - Spark as a next generation tool (still MapReduce paradigm)
  - **How to build machine learning algorithms in Spark** (using the Spark operators)

Data Scientist



Implementing ML algorithms from scratch

- Focus on theory, implementation and practice
- **Avoid the "Danger Zone"** (lots of great libraries but….)
- There are several different reasons why implementing algorithms from scratch can be useful:
  - it can help us to understand the inner workings of an algorithm
  - Let's us figure out how to parallelize
  - we can add new features to an algorithm or experiment with different variations of the core idea
  - we want to invent new algorithms or implement algorithms no one has implemented/shared yet
  - we are not satisfied with the API and/or we want to integrate it more "naturally" into an existing software library
  - we could try to implement an algorithm more efficiently
  - we circumvent licensing issues (e.g., Linux vs. Unix) or platform restrictions

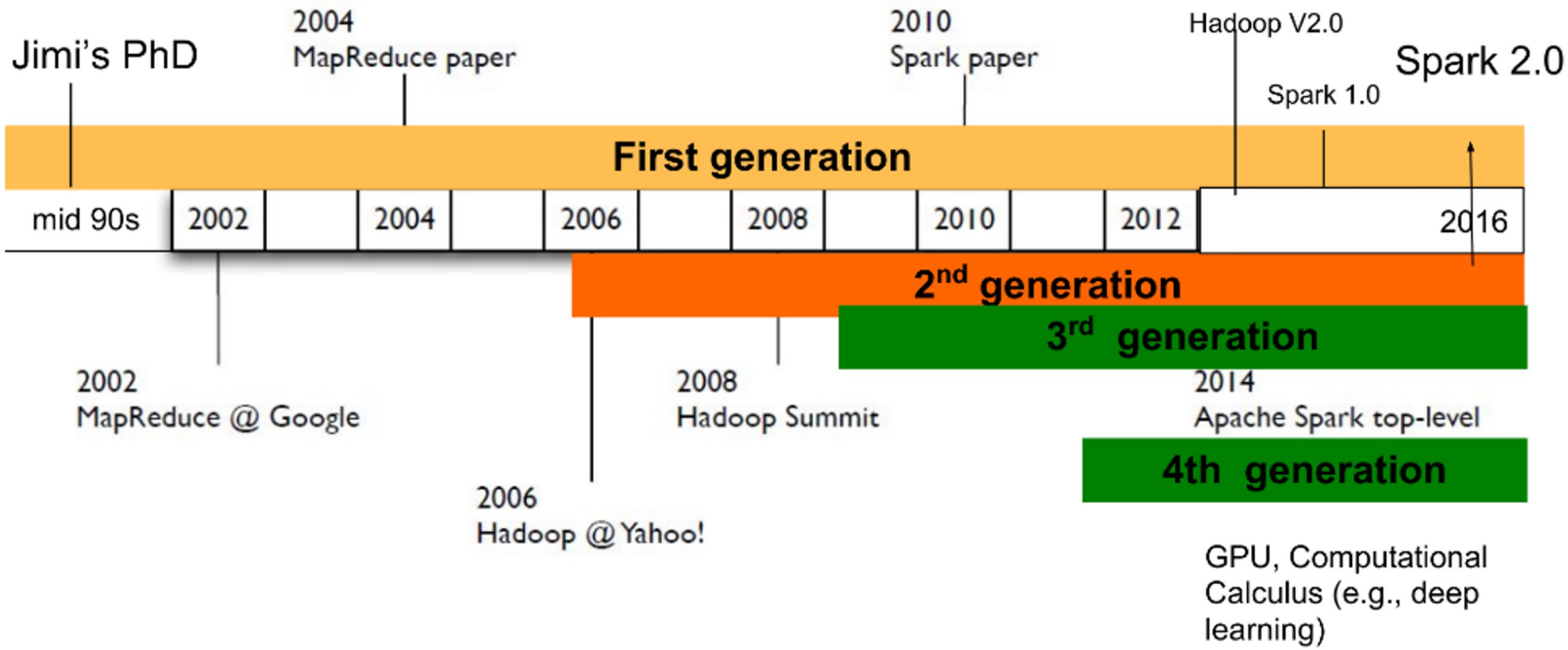
Preview of things to come

- Parallel frameworks for big data
  - Unix, Hadoop, Spark
- Supervised Machine Learning
  - Convex optimization, gradient descent, linear regression, decision trees, ensembles of models, support vector machines
- Unsupervised
  - Expectation maximization, matrix multiplication, alternating least squares
- Graphs
  - Random walks, PageRank, graph search algorithms such as BFS, shortest path
- Hybrid algos
  - Supervised ML + Random walks
- Applications
  - Digital advertising, social media, healthcare, e-commerce, entertainment

Data sets

- Enron SPAM Data (100k articles)
- Google ngram corpus (2.2 GB)
- Wikipedia graph (2 GB)
- $100 Billion problem CTR prediction (20-30 GB)
- Airlines and Weather - predicting delays (1 TB uncompressed)

Evolution of MapReduce frameworks



Other big data frameworks - “High-performance computing”

https://en.wikipedia.org/wiki/Supercomputer

HPC - High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business.

DryadLINQ - DryadLINQ combines two important pieces of Microsoft technology: the Dryad distributed execution engine and the .NET Language Integrated Query (LINQ).

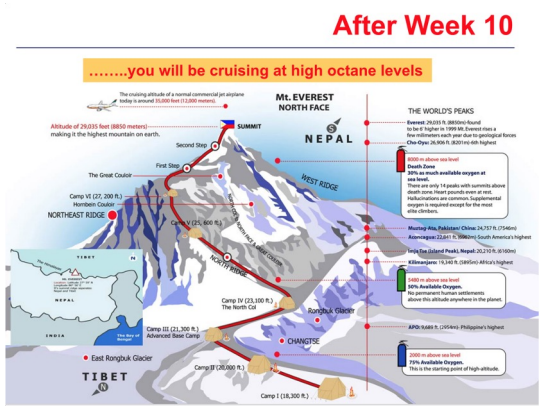
MPI - Message Passing Interface (MPI) is a standardized and portable message-passing standard designed by a group of researchers from academia and industry to function on a wide variety of parallel computing architectures. The standard defines the syntax and semantics of a core of library routines useful to a wide range of users writing portable message-passing programs in C, C++, and Fortran.

CUDA - The CUDA platform, created by NVIDIA, is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels.

DASK

Machine Learning @ Scale: "Type II" Fun

- Getting lost, getting cold, getting hungry, getting wet, getting scared, and coming out on top; that's the stuff you remember.
- That's "Type II" fun.
- Type I is the easy, fun-while-it's-happening stuff—mellow powder skiing, lazy cragging, afternoon hiking. You're bummed when it's over, but you'd be hard-pressed to remember more than a few specific examples.
- Type II Tough going, character building and
- Type III fun resides at the other end of the scale—miserable while it's happening, still miserable when it's over and just as miserable to think about later. NOT TO BE CONFUSED WITH "DIFFICULT".



See more at: http://www.backcountry.com/explore/type-ii-fun#sthash.CZankeqe.dpuf

“Fun” Challenge

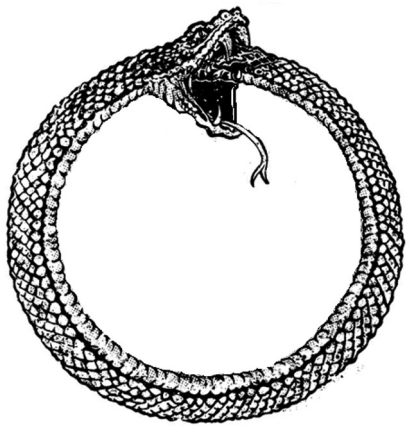
Jane came up with a brilliant new distributed algorithm for solving world hunger. She has a 1TB dataset upon which to train her model. She has just finished writing the last line of her python code! She has provisioned an auto-scaling Spark cluster on Data Proc and she is about to run her algorithm on 1TB of data.

What could possibly go wrong?

(HINT: see Type III fun)

What can Jane do to preserve her sanity?

(*HINT: perform sanity checks!)*



Problem solving at Scale

1. Local development plus debugging (unit test, systems test)
    1. Benchmark again publicly available libraries
2. Test on the cloud (unit test, systems test)
3. Deploy on the cloud (full scale experiment)

TEST, TEST, AND TEST AGAIN!!!

Week 1 - “Hello world”

Week 1 Highlights

- Bias Variance Trade-off
- Word count in MapReduce
  - From Word Counts to Multinomial Naïve Bayes
- Embarrassingly Parallel
- Linear Scaling in MapReduce

bias-variance tradeoff

In supervised machine learning, the bias–variance tradeoff is the property of a model that the variance of the parameter estimates across samples can be reduced by increasing the bias in the estimated parameters.

The bias–variance dilemma or bias–variance problem is the conflict in trying to simultaneously minimize these two sources of error that prevent supervised learning algorithms from generalizing beyond their training set:

- Underfitting: The bias error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
- Overfitting: The variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

Finding an $\hat{f}$ that generalizes to points outside of the training set can be done with any of the countless algorithms used for supervised learning. It turns out that whichever function $\hat{f}$ we select, we can decompose its expected error on an unseen sample x as follows:[6]:34[7]:223

$$\mathrm{E}_{D,\varepsilon}\left[\left(y-\hat{f}\left(x;D\right)\right)^2\right]=\left(\mathrm{Bias}_D\left[\hat{f}\left(x;D\right)\right]\right)^2+\mathrm{Var}_D\left[\hat{f}\left(x;D\right)\right]+\sigma^2$$

where

$$\mathrm{Bias}_D\left[\hat{f}\left(x;D\right)\right]=\mathrm{E}_D\left[\hat{f}\left(x;D\right)-f(x)\right]=\mathrm{E}_D\left[\hat{f}\left(x;D\right)\right]-\mathrm{E}\left[y(x)\right]$$

and

$$\mathrm{Var}_D\left[\hat{f}\left(x;D\right)\right]=\mathrm{E}_D[\left(\mathrm{E}_D[\hat{f}\left(x;D\right)]-\hat{f}\left(x;D\right)\right)^2].$$

The expectation ranges over different choices of the training set $D=\{(x_1,y_1)\ldots,(x_n,y_n)\}$, all sampled from the same joint distribution $P(x,y)$ which can for example be done via bootstrapping. The three terms represent:

- the square of the *bias* of the learning method, which can be thought of as the error caused by the simplifying assumptions built into the method. E.g., when approximating a non-linear function $f(x)$ using a learning method for linear models, there will be error in the estimates $\hat{f}\left(x\right)$ due to this assumption;
- the *variance* of the learning method, or, intuitively, how much the learning method $\hat{f}\left(x\right)$ will move around its mean;
- the irreducible error σ^2.

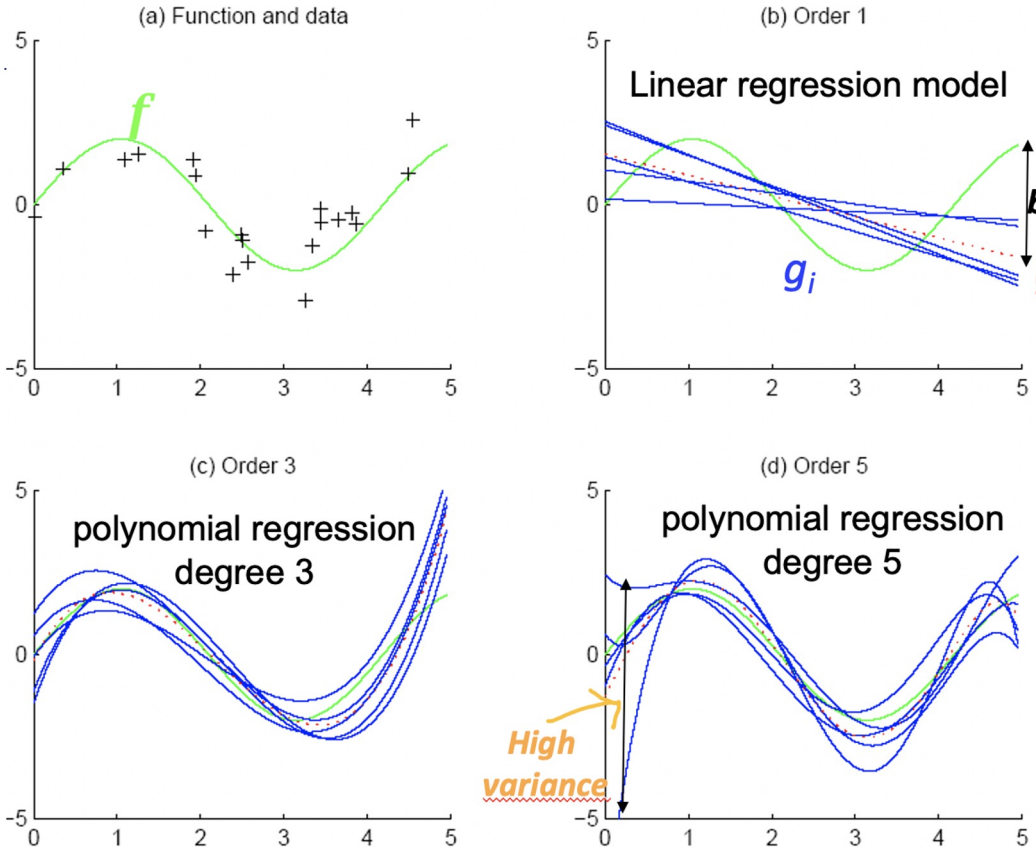
Estimating Bias-Variance: 2 Cases

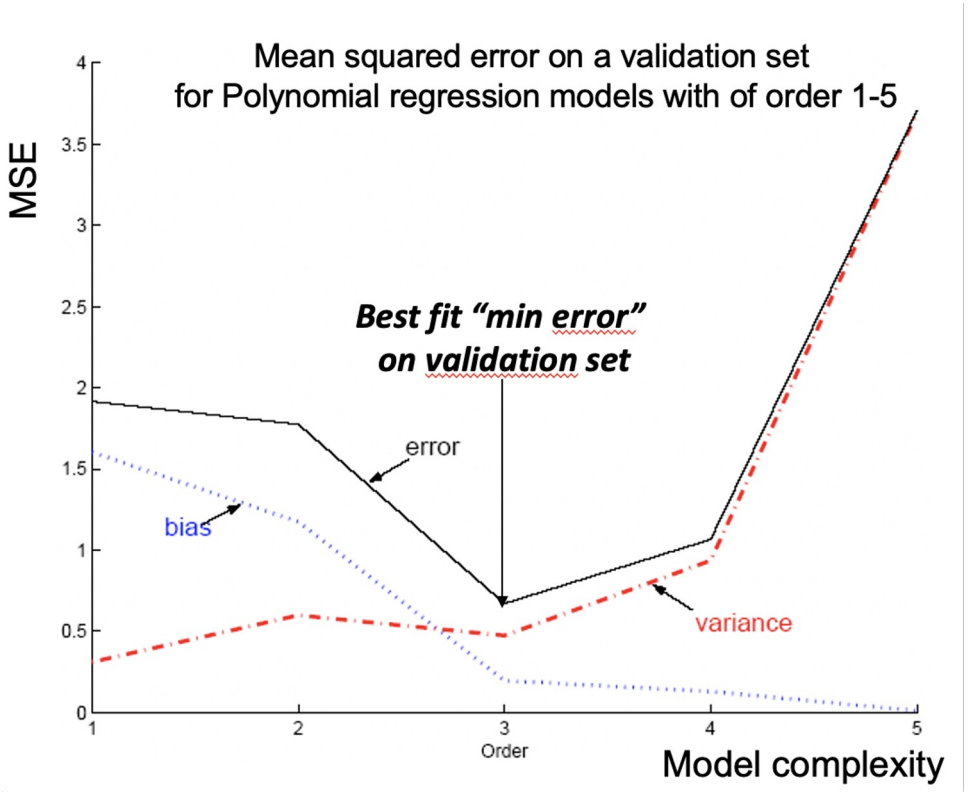
- **Case 1: Simulated world: we know the true target function; decompose error (e.g., MSE) into bias, variance, and irreducible error via bootstrap sampling type of analysis**
  - Estimating Bias and Variance in a simulated world where we know the target function
  - We sample training data and train multiple models
  - Then using a validation dataset (a test set is not used in this analysis)
    - Estimate variance of different model predictions
    - Estimate bias (mean prediction of all trained model versus true value)
    - Estimate irreducible error (as we know the true value and the observed noisy value)
- **Case 2: Real world: do NOT know the true target function; decompose error (e.g., MSE) into bias and variance (but not the irreducible error) via bootstrap sampling type of analysis**
  - Estimating Bias and Variance in the real world where we do NOT know the target function
    - Can assume zero noise and estimate bias and variance where the observed target value serves also as the true value

Reference

–https://blog.insightdatascience.com/bias-variance-tradeoff-explained-fa2bc28174c4

Bias-variance decomp for a simulated dataset





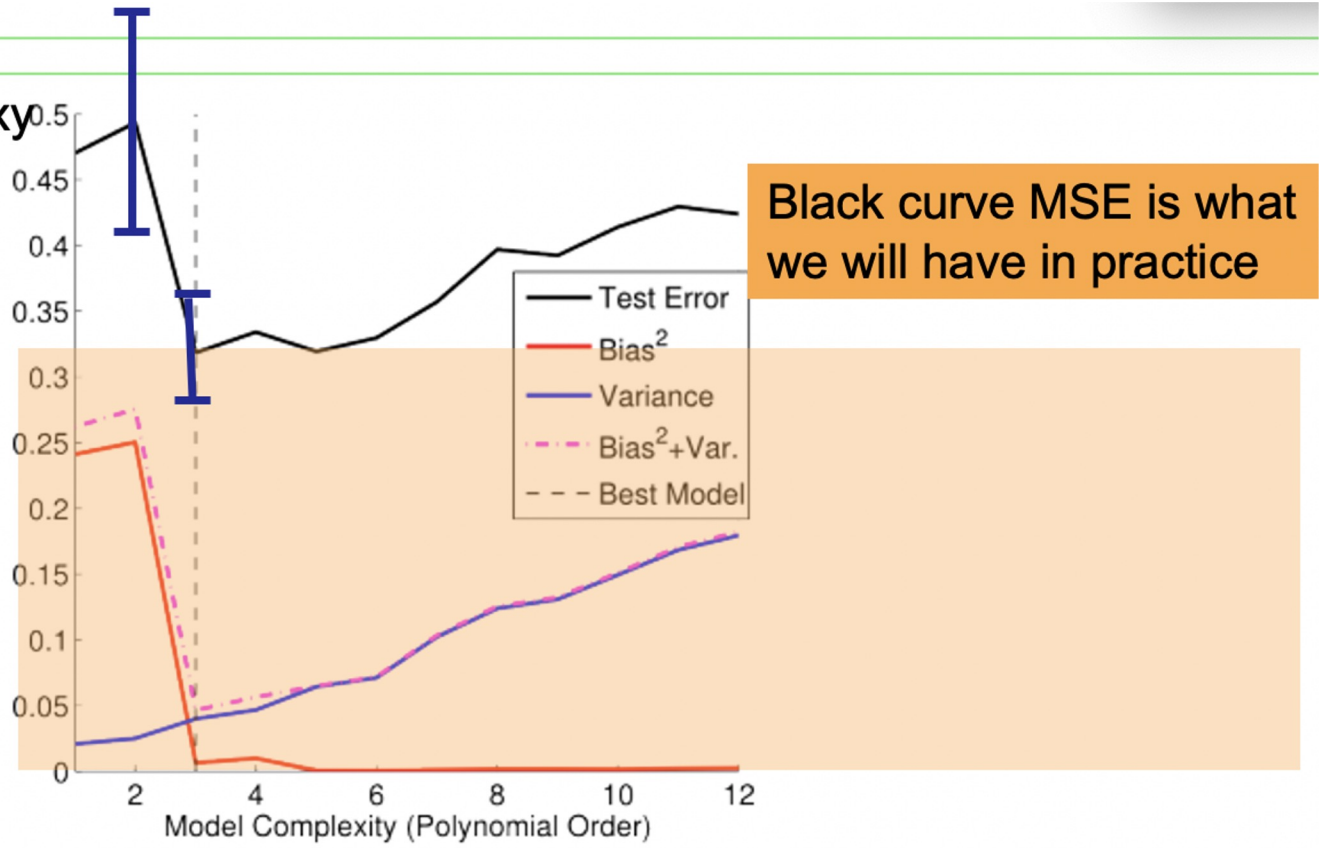
Bias-Variance Decomposition vs MSE

+ expand source

Black curve is a good proxy
Bias-variance decomp.

Cross fold validation
Gives us mean and std

To do a bias-variance decomposition assume TRUE function is available



Black curve MSE is what we will have in practice

http://scott.fortmann-roe.com/docs/BiasVariance.html

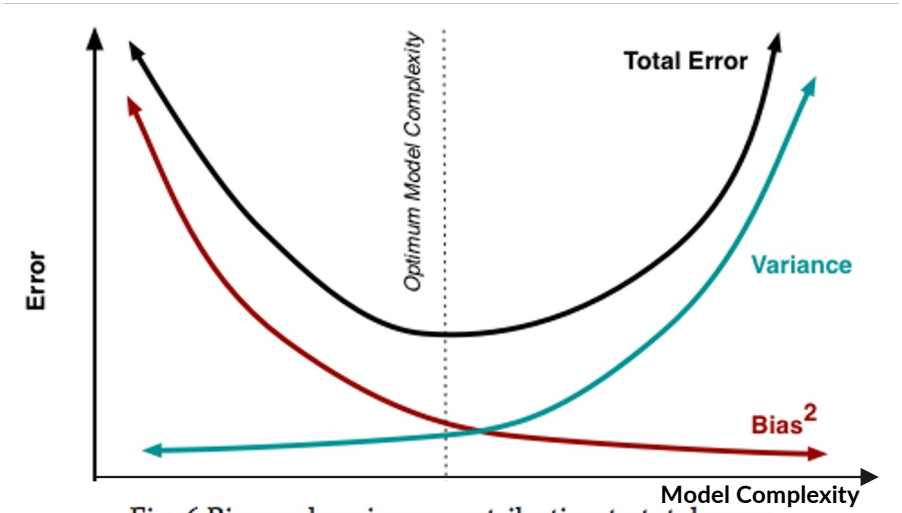


Fig. 6 Bias and variance contributing to total error.

Understanding bias and variance is critical for understanding the behavior of prediction models, but in general what you really care about is overall error, not the specific decomposition. The sweet spot for any model is the level of complexity at which the increase in bias is equivalent to the reduction in variance. Mathematically:

$$\frac{dBias}{dComplexity} = -\frac{dVariance}{dComplexity}$$

Bias & Variance

- Trade-off between bias and variance

What is Bias? (underfitting)
The bias error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

What is Variance? (overfitting)
The variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

Bias-Variance in lay persons language

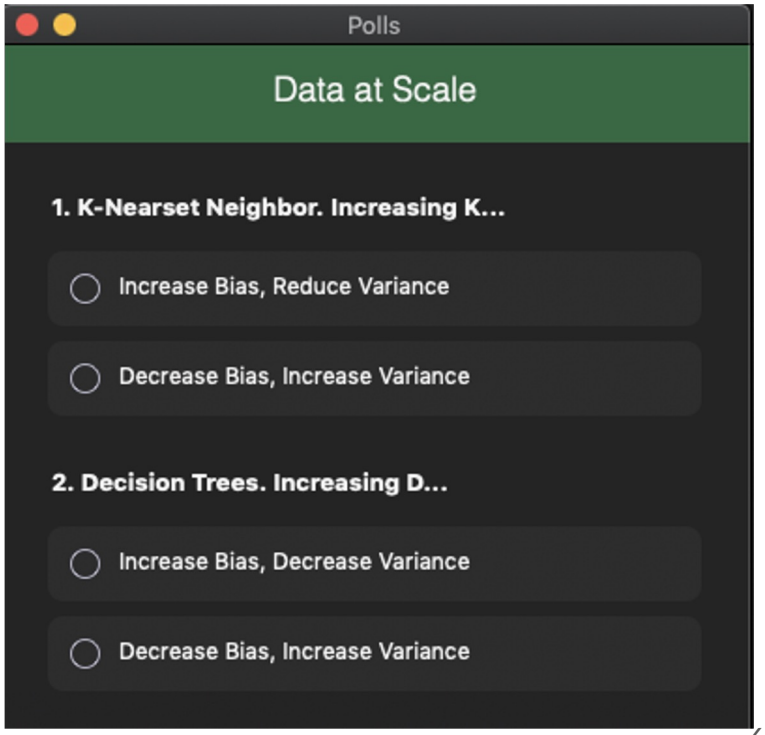
- BIAS: Assume we have a model that is too simple to explain the data we have (e.g., a linear model when the problem is non-linear).
  - In that case, known as *high bias*, adding more data will not help.
  - high bias models will not benefit from more training examples, but they might very well benefit from more features
- VARIANCE: Assume a model that is too complicated for the amount of data we have. This situation, known as *high variance*, leads to model overfitting.
  - We know that we are facing a high variance issue when the training error is much lower than the test error.
  - High variance problems can be addressed by reducing the number of features, and… yes, by increasing the number of data points.
  - So, no, **more data does not always help**. More features to the rescue!
- Be careful, even Malcolm Gladwell or Chris Anderson get it wrong:
  - **The End of the Scientific Method?**
    - Of course, whenever there is a heated debate about a possible paradigm change, there are people like Malcolm Gladwell or Chris Anderson that make a living out of heating it even more (don't get me wrong, I am a fan of both, and have read most of their books). In this case, Anderson picked on some of Norvig's comments, and misquoted them in an article entitled: The End of Theory: The Data Deluge Makes the Scientific Method Obsolete.

https://www.kdnuggets.com/2015/06/machine-learning-more-data-better-algorithms.html

Quiz

How do the following model parameters affect bias and variance?

- K-Nearest Neighbor
  a. Typically, how does increasing k affect bias and variance?
- Decision trees of depth D
  a. Typically, how does increasing D affect bias and variance?



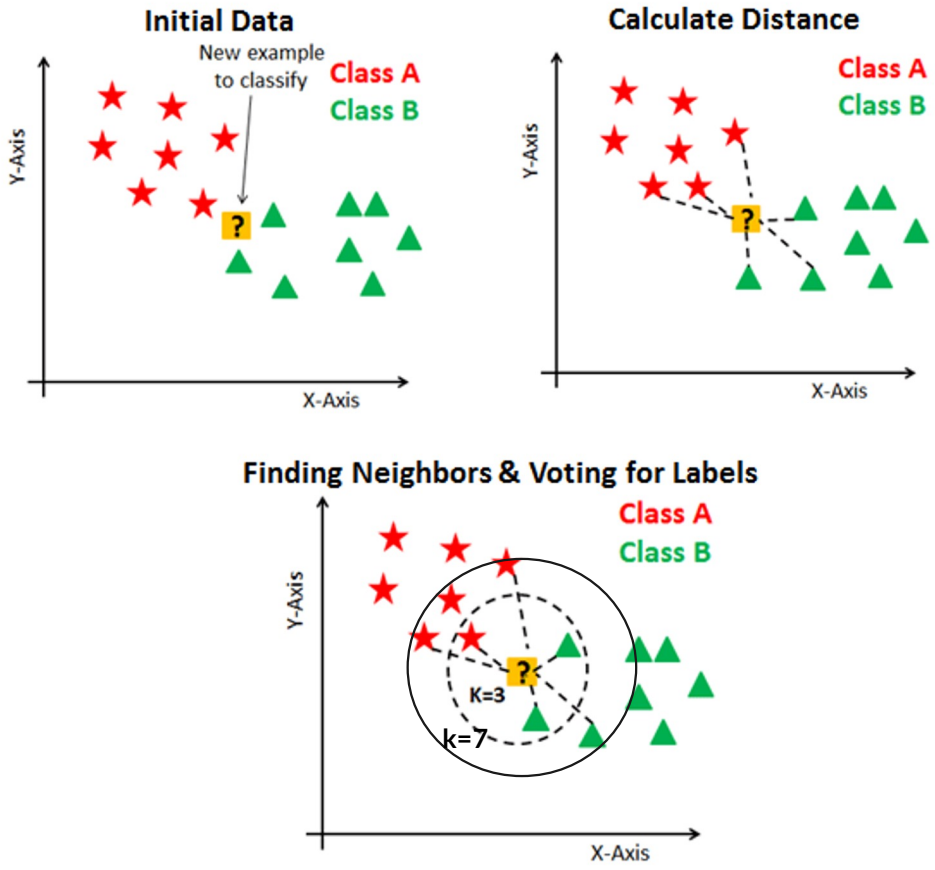
KNN Classifier with two classes: star versus triangle

large **k** => simple model => underfit => low variance & high bias (less focus, blurred response)

small **k** => complex model => overfit => high variance & low bias

When **k** increases to inf, the model is simplest. All test data points will belong to the same class: the majority class. This is under-fit, that is, high bias and low variance.

When **k** decreases, let's say k=1, the granularity or resolution is too fine, which is overfit. Overfit => high variance



WORD COUNT
The Hello World of Map reduce

Word count on a single core machine

How would you go about doing a word count on a single core machine?

What data structure in python would you use?

INPUT

```
doc = """
Chinese Beijing Chinese
Chinese Chinese Shanghai
Chinese Macao
Tokyo Japan Chinese
Chinese Chinese Chinese Tokyo Japan.
"""
```

OUTPUT

```
('beijing', 1)
('chinese', 9)
('japan', 2)
('macao', 1)
('shanghai', 1)
('tokyo', 2)
```

Word count on a single core machine

How would you go about doing a word count on a single core machine?

What data structure in python would you use?

```
#version 1
word_counts = {}
for word in doc.lower().split():
    If word in word_counts:
        word_counts [word] +=1
    else:
        word_counts [word] = 1
```

INPUT

```
doc = """
Chinese Beijing Chinese
Chinese Chinese Shanghai
Chinese Macao
Tokyo Japan Chinese
Chinese Chinese Chinese Tokyo Japan.
"""
```

OUTPUT

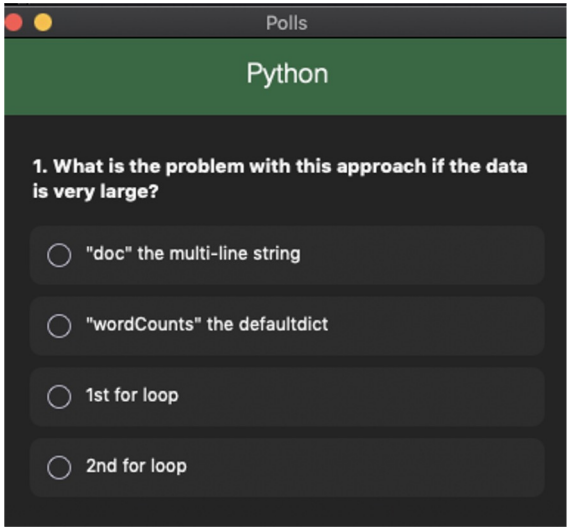
```
('beijing', 1)
('chinese', 9)
('japan', 2)
('macao', 1)
('shanghai', 1)
('tokyo', 2)
```

What is the problem with this approach if the data is very large?

Word count on a single machine using a key-value dictionary

back to top

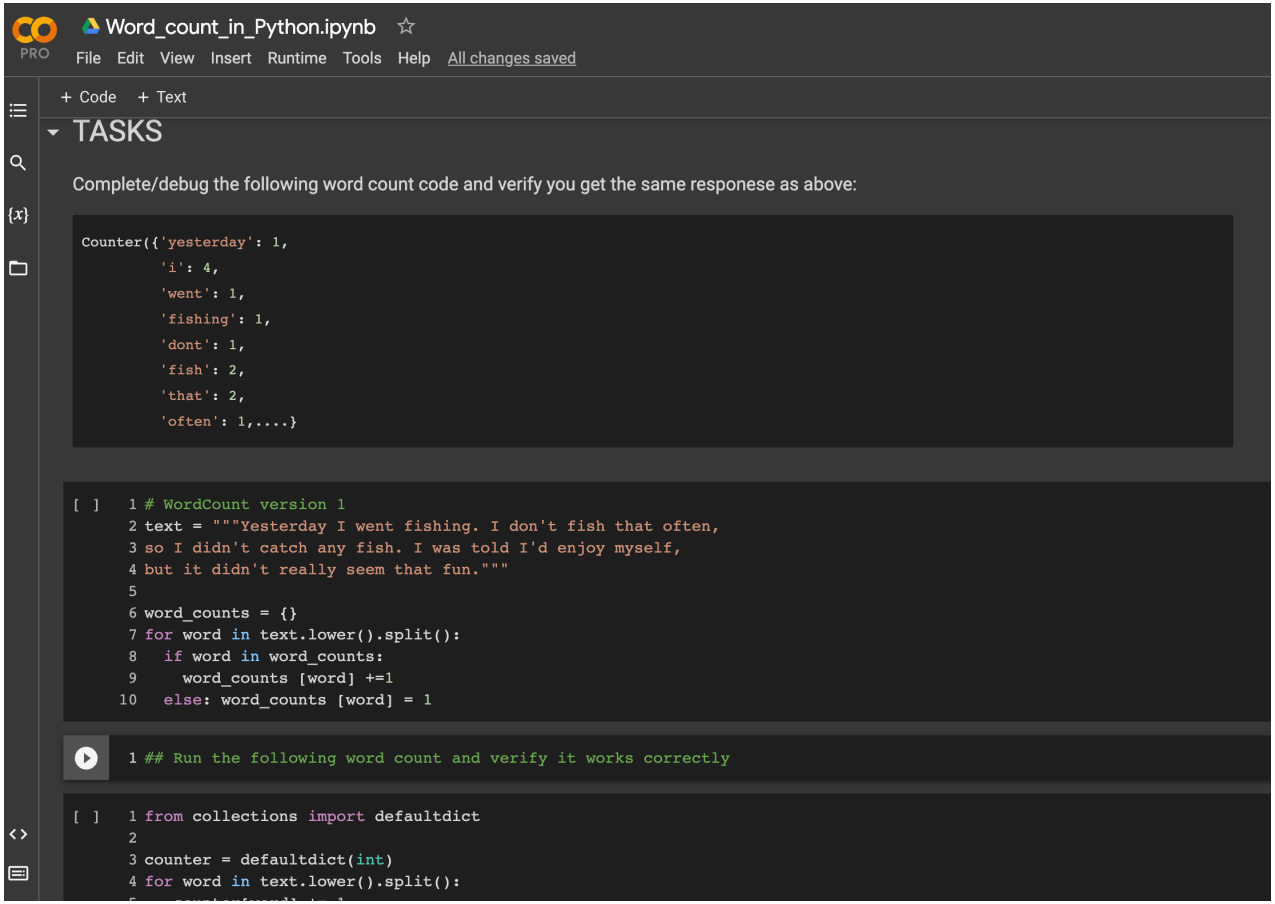
```
# Here is an example of wordcounting with a defaultdict (dictionary structure with a nice
# default behaviours when a key does not exist in the dictionary
import re
from collections import defaultdict

doc = """
Chinese Beijing Chinese
Chinese Chinese Shanghai
Chinese Macao
Tokyo Japan Chinese
Chinese Chinese Chinese Tokyo Japan.
"""

# Initialize a defaultdict, where the dafault value is an int
wordCounts=defaultdict(int)

# loop over all the words in the doc, and increment the count as you see words
for word in re.findall(r'[a-z]+', doc.lower()):
    wordCounts[word] += 1

# Sort by word, and print the top 10 words with their counts
for key in sorted(wordCounts)[0:10]:
    print (key, wordCounts[key])
```




```
('beijing', 1)
('chinese', 9)
('japan', 2)
('macao', 1)
('shanghai', 1)
('tokyo', 2)
```

WordCount in a Colab Notebook

https://colab.research.google.com/drive/1mPmTWRQ8f4y8QSYgBYd-0T2Qw53hQkf3?usp=sharing



pdb in a jupyter notebook cell

The built-in Python debugger `pdb` works just fine in a Jupyter notebook. With `import pdb; pdb.set_trace()` we can enter the debugger and get a little interactive prompt.

```
1 def add_to_life_universe_everything(x):
2     answer = 42
3     import pdb; pdb.set_trace()
4     answer += x
5     return answer

add_to_life_universe_everything(12)
```

For more background on pdb see
https://davidhamann.de/2017/04/22/debugging-jupyter-notebooks/

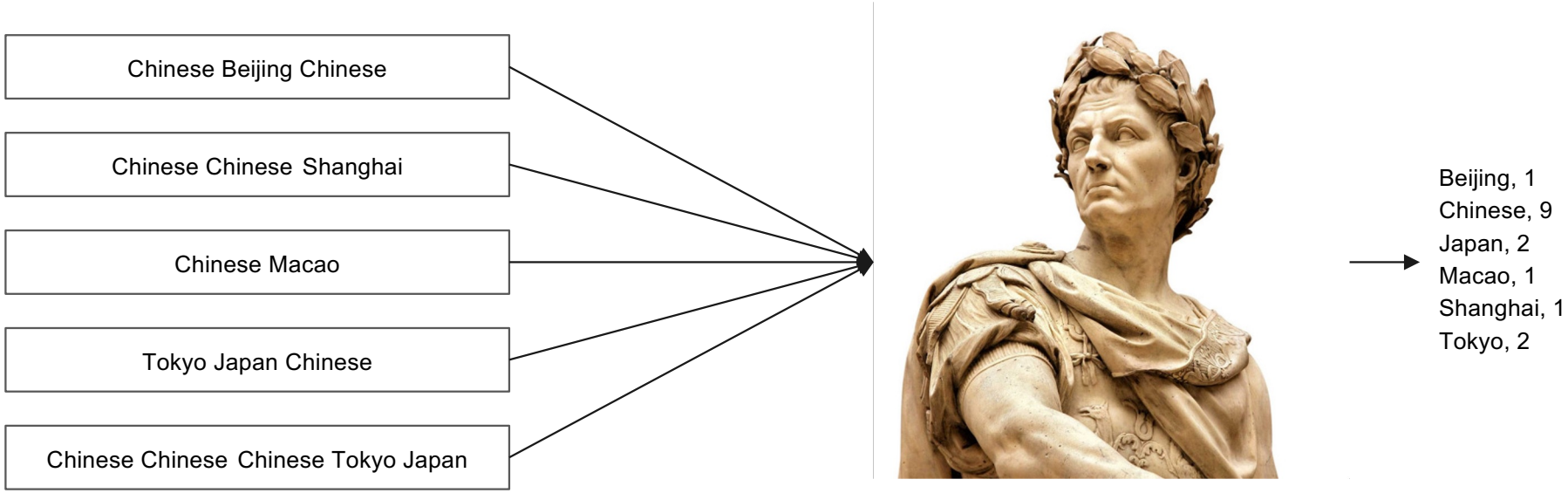
In the debugger we can then print our variables, evaluate code to inspect the current stack, etc.

```
> <ipython-input>(4)add_to_life_universe_everything(12)
-> answer += x
(Pdb) answer + 12 – “dummy”
42
(Pdb) n
> <ipython-input>(6)add_to_life_universe_everything(6)
-> return answer
(Pdb) answer
54
(Pdb) c
```

Here, we print `answer`, then execute and go to the next line with `n`, print `answer` again and then continue the program with `c`. **You can get an overview of the commands with `h`, or specifically with `help <command>`.**

Divide and Conquer (a.k.a. merge-sort) to the rescue!

Can we take our big task, and split it up into a bunch of smaller tasks that we can perform independently?



https://en.wikipedia.org/wiki/Merge_sort

In computer science, **merge sort** (also commonly spelled **mergesort**) is an efficient, general-purpose, comparison-based sorting algorithm. Most implementations produce a stable sort, which means that the order of equal elements is the same in the input and output. Merge sort is a divide and conquer algorithm that was invented by John von Neumann in 1945.

An example of merge sort. First divide the list into the smallest unit (1 element), then compare each element with the adjacent list to sort and merge the two adjacent lists. Finally all the elements are sorted and merged.

Worst-case performance: O(*n* log *n*)

Best-case performance O(*n* log *n*) typical,

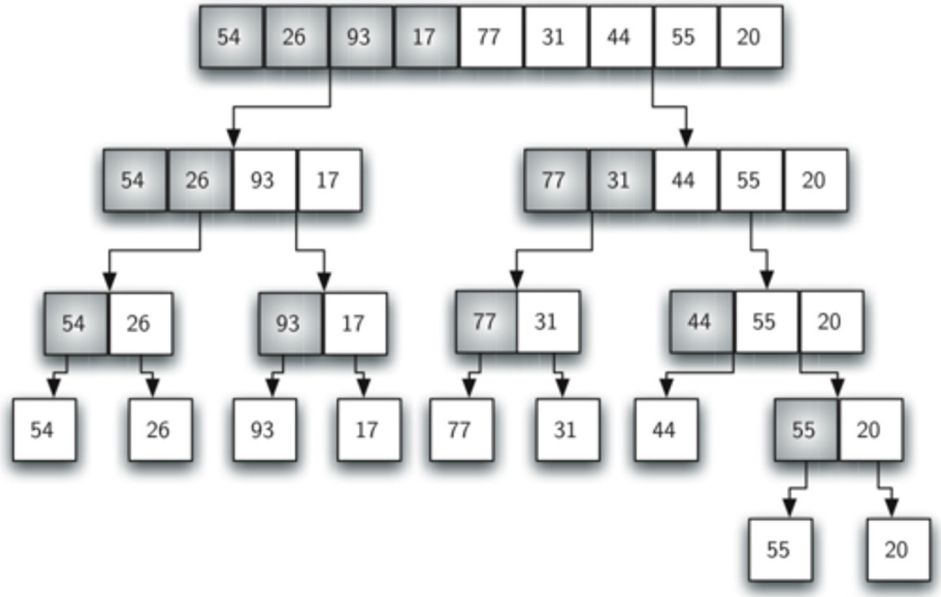
O(*n*) natural variant

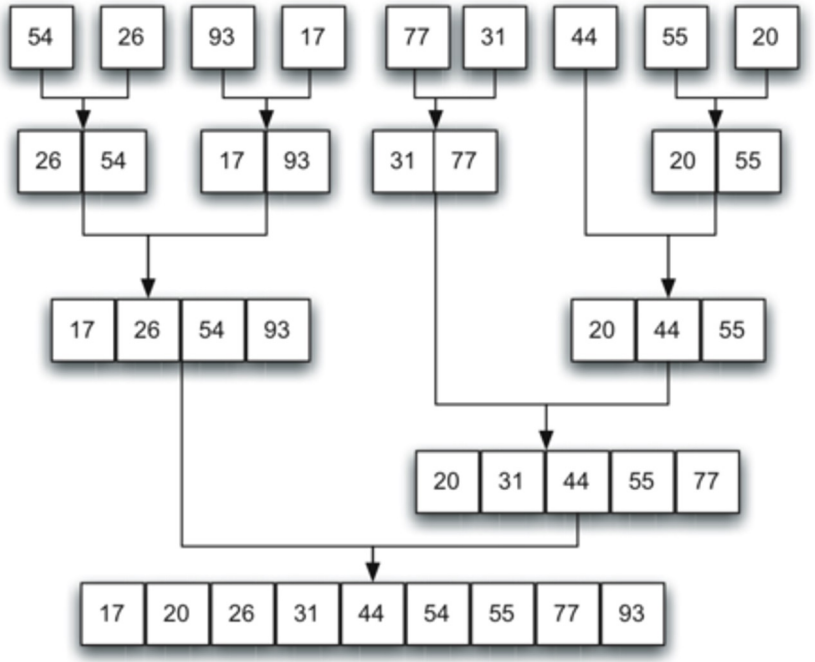
Average performance: O(*n* log *n*)

Worst-case space complexity

O(*n*) total with O(*n*) auxiliary, O(*1*) auxiliary with linked lists[1]

Merge Sort





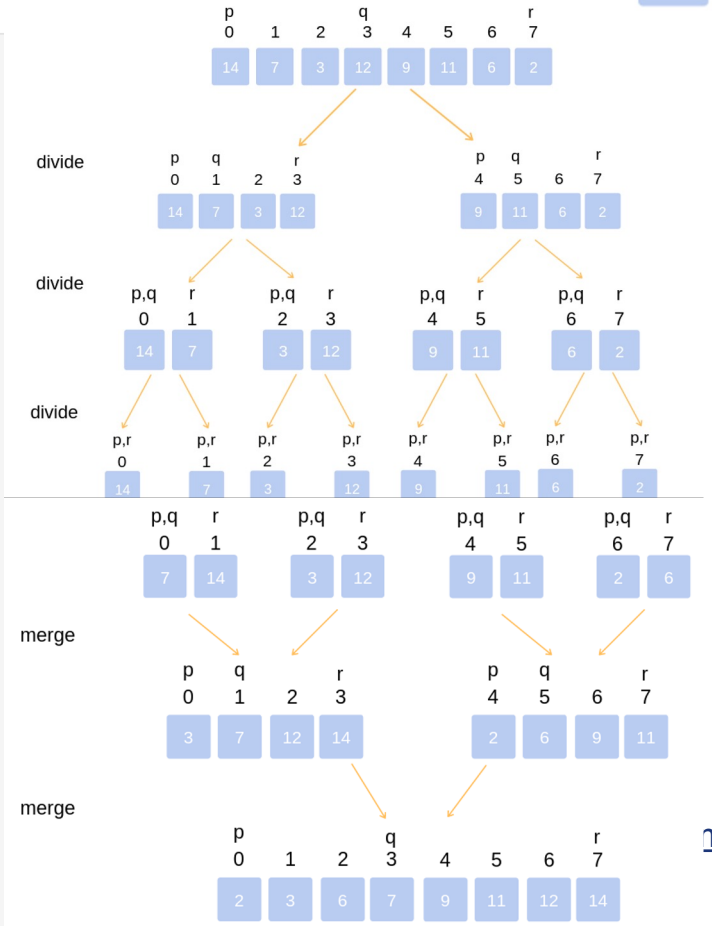
/HelpfulResources/merge-sort.ipynb

https://www.interviewbit.com/tutorial/merge-sort-algorithm/

RECURSIVE merge sort in python

Top-down implementation

```python
# example of merge sort in Python (RECURSIVE)
# merge function take two intervals
# one from start to mid
# second from mid+1, to end
# and merge them in sorted order
def merge(Arr, start, mid, end) :


    # create a temp array
    temp[] = [0] * (end - start + 1)

    # crawlers for both intervals and for temp
    i, j, k = start, mid+1, 0

    # traverse both lists and in each iteration add smaller of both elements in temp
    while(i <= mid and j <= end) :
        if(Arr[i] <= Arr[j]) :
            temp[k] = Arr[i]
            k += 1; i += 1
        else :
            temp[k] = Arr[j]
            k += 1; j += 1

    # add elements left in the first interval
    while(i <= mid)
        temp[k] = Arr[i]
        k += 1; i += 1

    # add elements left in the second interval
    while(j <= end)
        temp[k] = Arr[j]
        k += 1; j += 1

    # copy temp to original interval
    for(i = start; i <= end; i += 1)
        Arr[i] = temp[i - start]


# Arr is an array of integer type
# start and end are the starting and ending index of current interval of Arr

def mergeSort(Arr, start, end) {

    if(start < end) :
        mid = (start + end) / 2
        mergeSort(Arr, start, mid)
        mergeSort(Arr, mid+1, end)
        merge(Arr, start, mid, end)
```



Divide the unsorted list into n sublists, each comprising 1 element (a list of 1 element is sorted)

Merge two lists

m/tutorial/merge-sort-algorithm/

TimSort used in Spark (instead of mergesort)

TimSort was born in practice, a hybrid of Merge Sort and Insertion (decrease and conquer) Sort

Tim Sort was first implemented in 2002 by Tim Peters for use in Python. It allegedly came from the understanding that most sorting algorithms are born in schoolrooms, and not designed for practical use on real world data. Tim Sort takes advantage of common patterns in data, and utilizes a combination of:

- Chunk, Runs, and insertion sort
- Merge from mergesort

Example: Merge Sort in Unix

Consider files `a.txt` and `b.txt` with following sorted content.

Then one can merge the two files and generate a sorted output using the `-m` switch

Challenge:

- Tweak the merge-sort command to merge sort on the second field.
- Does the output change?

```
cat a.txt   #show contents of sorted file
KEY   PAYLOAD
Deep     1
Jimi      2
Learning  2
```

```
cat b.txt   #show contents of sorted file
Amil      1
Jimi      3
```

```
$sort -m a.txt b.txt  #merge sort the two files
Amil     1
Deep     1
Jimi      2
Jimi      3
Learning 2
```

Sort command in linux with examples (sorting is key in map-reduce)

Sorting by a Single Column

Sorting by single column requires the use of the `-k` option. You must also specify the start column and end column to sort by. When sorting by a single column, these numbers will be the same. Here is an example of sorting a CSV (comma delimited) file by the second column.

```
#sorting a CSV by the values in column 2
sort -k2,2 -t ',' sample.txt
```

Sorting by Multiple Columns: contiguous versus non-contiguous

Sorting by multiple columns is similar to sorting by a single column. To sort on a range of columns, simply specify the start and end columns in the columr range to use for sorting. Here is an example of sorting a CSV on columns 2 through 4.

```
#sorting a CSV by the values in columns 2-4
sort -k2,4 -t ',' sample.txt
```

To sort on a **non-contiguous set of columns**, you must use the `-k` option multiple times. Here is an example of sorting by column 2 then column 6:

```
#sorting a CSV by the values in column 2 then column 6
sort -k2,2 -k6,6 -t ',' sample.txt
```

For details please see: https://phpfog.com/linux-sort-command-examples/

Unix sort examples

%%writefile unix-sort-example.txt
Unix,30
Solaris,10
Linux,25
Linux,20
HPUX,100
AIX,25

sort -t"," -k1,1 sort -t"," -k2,2nr sort -t"," -k1,1 -k2,2nr

Sort by field 1 (default alphabetically), deliminator ","

cat unix-sort-example.txt sort -t"," -k1,1 unix-sort-example.txt

```
cat unix-sort-example.txt  sort -t"," -k1,1 unix-sort-example.txt
```

Unix,30	AIX,25
Solaris,10	HPUX,100
Linux,25	Linux,20
Linux,20	Linux,25
HPUX,100	Solaris,10
AIX,25	Unix,30

Sort by field 2 numerically reverse, deliminator ","

```
cat unix-sort-example.txt  sort -t"," -k2,2nr  unix-sort-example.txt
```

Unix,30	HPUX,100
Solaris,10	Unix,30
Linux,25	AIX,25
Linux,20	Linux,25
HPUX,100	Linux,20
AIX,25	Solaris,10

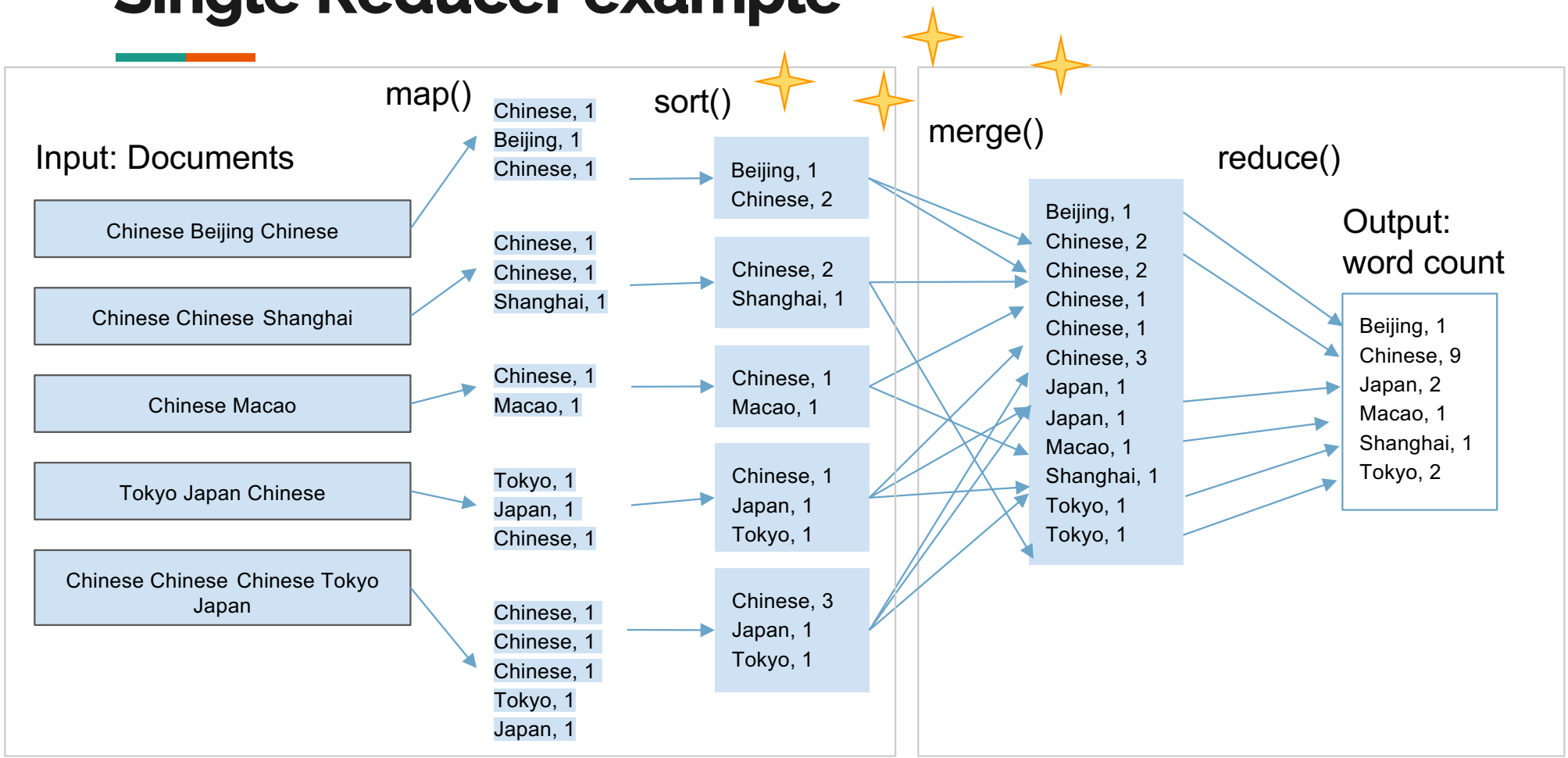
Sort by field 1 alphabetically first, then by field 2 numeric reverse

```
cat unix-sort-example.txt  sort -t"," -k1,1 -k2,2nr  unix-sort-example.txt
```

Unix,30	AIX,25
Solaris,10	HPUX,100
Linux,25	Linux,25
Linux,20	Linux,20
HPUX,100	Solaris,10
AIX,25	Unix,30

GCS/HelpfulResources/TotalSortGuide/total-sort-guide-hadoop-streaming.ipynb

Single Reducer example



MapReduce - Key Concepts

Phases of MapReduce

- Map phase
  - Each mapper takes a line as input and breaks it into words. It then emits a key/value pair of the word and 1.
- Shuffle/sort phase (*partition and sort the mapper output*)
  - The output from the mappers is sorted by key and shuffled into the reducers such that key/value pairs with the same key end up in the same reducer.
  - Sort is a simple and very useful command found in Unix systems. It rearranges lines of text numerically and/or alphabetically. (*Hadoop Streaming KeyBasedComparator is modeled after Unix sort, and utilizes command line options which are the same as Unix sort command line options*).
  - http://www.theunixschool.com/2012/08/linux-sort-command-examples.html
- Reduce phase
  - Each reducer sums the counts for each word and emits a single key/value with the word and sum.

Embarrassingly parallel

In parallel computing, an **embarrassingly parallel** workload or problem (also called **perfectly parallel**, **delightfully parallel** or **pleasingly parallel**) is one where little or no effort is needed to separate the problem into a number of parallel tasks.[1] This is often the case where there is little or no dependency or need for communication between those parallel tasks, or for results between them.[

No shared state

- a `map()` task does not need to know about any other map task
- a `reduce()` task has all the context it needs to aggregate for any particular key, it does not share any state with other reduce tasks.

Taken as a whole, this design means that the stages of the pipeline can be easily distributed to an arbitrary number of machines.

Workflows requiring massive datasets can be easily distributed across hundreds of machines because there are no inherent dependencies between the tasks requiring them to be on the same machine (ie, sub tasks never need to communicate).

Test your hacking skills

- Linux command line
  - e.g., a sample task could be count the number lines in a file (wc); sort a file based on the second field in reverse numerical order; extract the second field from a TSV file
- Python dictionaries
  - e.g., implement word count by themselves
- Regular expressions
  - e.g., find all the phone numbers in a chunk of text, 415-630-0893, (415)-6300893, 4156300893
- Debugging in Jupyter notebooks using pdb
- STRETCH TASK: Numpy basics for vector multiplication/addition, matrix initialization, matrix multiplication
  - e.g., randomly define a 4x3 matrix, and a 3x4 matrix and multiply them.
- STRETCH TASK: review sorting algos
  - e.g., mergesort and possibly implement it

From Word Counts to Multinomial Naïve Bayes

Next Week:

How can we use what we learned from word count to implement a distributed Naive Bayes?

End of lecture

Feel free to contact me at James.Shanahan __AT__ gmail.com