

# ISA 414 – Managing Big Data

## Lecture 1 – Course Introduction

Dr. Arthur Carvalho  
[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Reminders

---

- Masks are required indoors
  - What about instructors?
    - “*Faculty may remove masks while teaching if others in the class are masked and if they are able to maintain some distance, e.g., at the whiteboard, podium, or generally six feet of distance.*” (Provost Message to Academic Affairs - August 16, 2021)
- No food or drinks (other than water) in the lab

# Course Introduction

- The power and perils of big data
  - Example: social media
    - Very rich data
      - Likes, emotions, comments, replies, photos, location, groups, ...
    - Highly **unstructured** data
      - Textual data, images, geolocation, ...
    - **Massive amounts** of data
      - E.g., Facebook has over 2 billion users
      - Each one with her/his own photos, comments, likes, ...



# Course Introduction

---

- The power and perils of big data
  - Example: social media
    - What can be done with such data?
      - Personalized advertising



# Course Introduction

---

- The power and perils of big data
  - Example: social media
    - What can be done with such data?
      - Trend detection

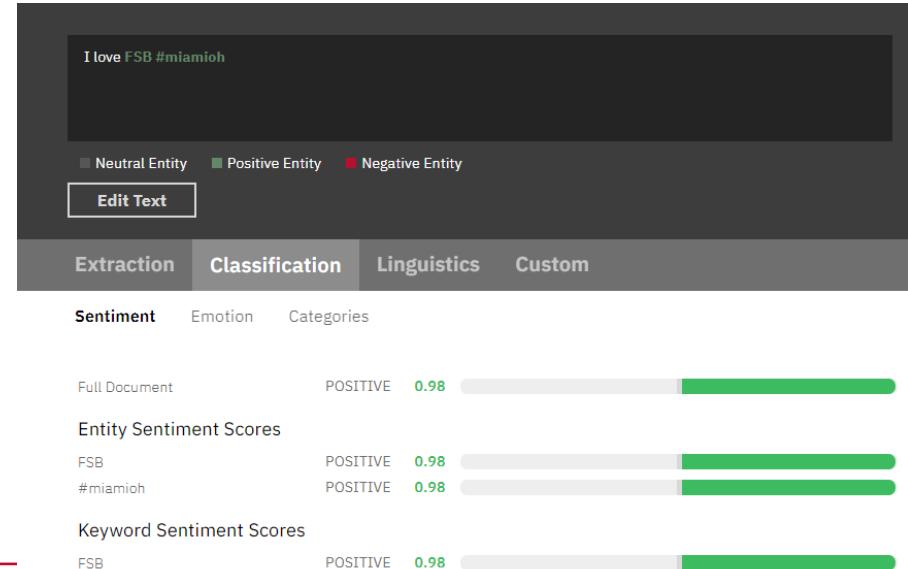
The screenshot shows the Facebook Analytics interface. The left sidebar has a navigation menu with items like Overview, Insights, Dashboards, Create Dashboard, Activity, People, Demographics, **Interests** (which is selected and highlighted in blue), Technology, and User Properties. The main content area is titled 'Interests' and displays a table titled 'Top Page Categories'. The table lists categories from 1 to 6, each with a relevance score and a list of pages associated with it.

Category ↑	Relevance ↑	Pages
Health/Beauty	1	L'Oréal Paris • Elle 18 • Maybelline New York • I Love Lakme • Dove •
Business & Economy Website	2	Nykaa
Entertainment Website	3	iDiva • Desi Humor
Fashion Designer	4	SHYAMAL & BHUMIKA
Broadcasting & Media Production Company	5	Washington Post • NPR
E-commerce Website	6	Myntra • Hopsocatch.in • Peachmode • FirstCry.com

# Course Introduction

---

- The power and perils of big data
  - Example: social media
    - What can be done with such data?
      - **Sentiment Analysis**



Source:

<https://www.ibm.com/demos/live/natural-language-understanding/self-service/home>

# Course Introduction

---

- The power and perils of big data
  - Example: social media
    - The previous services have tremendous value to companies and (arguably) consumers
      - *Personalized advertising*: consumers do not see irrelevant content
      - *Trend detection*: companies can focus on what consumers want
      - *Sentiment analysis*: companies can quickly gather consumers' opinions and react accordingly in real-time
    - What about the “perils”?
      - *E.g.*, Cambridge Analytica

# Course Introduction

---

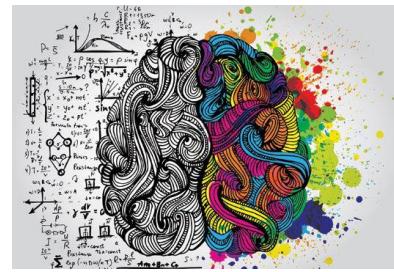
- The power and perils of big data
  - Example: social media
    - Cambridge Analytica and Project Alamo

Source: The Great Hack (Netflix)

Personal data



Psychological Profile



Targeted Ads



# Course Introduction

- The power and perils of big data
  - Example: social media
    - Cambridge Analytica built a propaganda machine during the Brexit and US 2016 presidential election
      - Goal: drive behavior by suppressing/increasing turnout
      - Consequences: democracies might have been hacked

Personal data



Psychological Profile

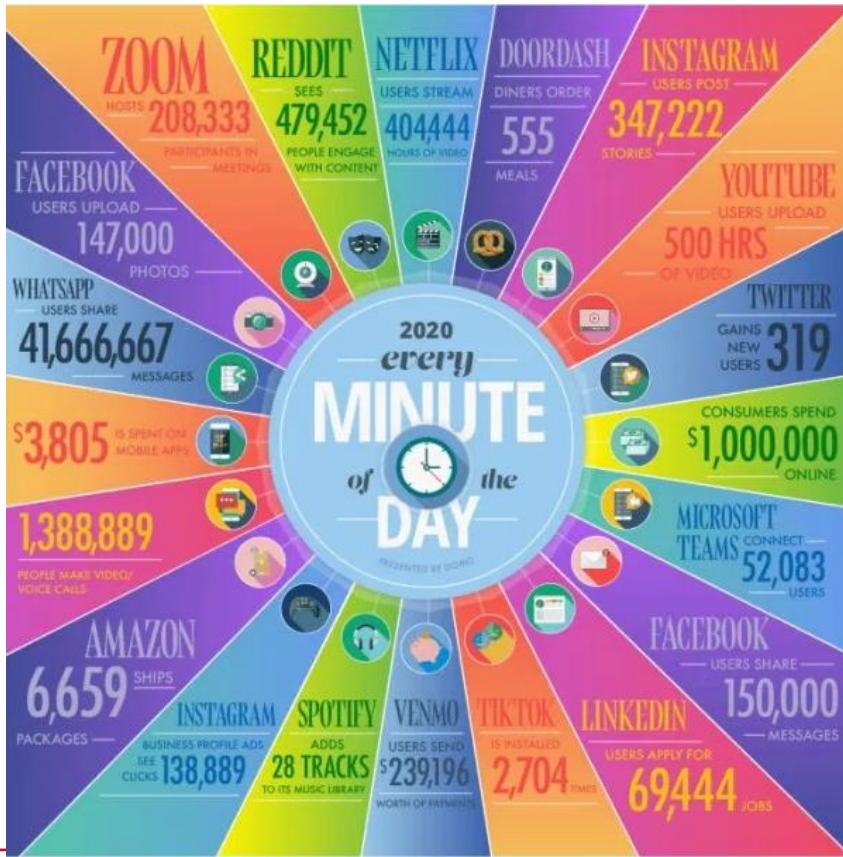


Targeted Ads



# Course Introduction

- Why is this happening?
  - We live in the “Big Data” era
    - Every 60 seconds



# Course Introduction

---

- Why is this happening?
  - Current data
    - Easy to collect
    - Cheap to store
    - Abundant (GPS, WWW, Smartphones, ...)
- Where are we moving to?
  - Data rich → data smart
    - Data-centric business processes
    - Data-centric public policies
    - Data-centric education (e.g., Canvas)

# Course Introduction

- Good news for you: \$\$\$
  - FSB salary and placement data

	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Information Systems	\$49.0	\$51.0	\$52.0	\$57.0		\$58.6	\$57.2	\$57.2	\$61.3	\$63.5	\$62.3	\$65.4
Supply Chain/Operations Management	\$47.0		\$51.0	\$53.0		\$53.7	\$55.0	\$53.7	\$55.2	\$59.2	\$54.8	\$61.4
Finance	\$47.0	\$47.0	\$49.0	\$56.0		\$54.5	\$61.8	\$59.5	\$58.3	\$60.3	\$59.5	\$63.5
Accountancy	\$43.0	\$43.0	\$48.0	\$55.0		\$51.9	\$53.7	\$54.0	\$54.0	\$56.0	\$57.6	\$60.4
Business Economics	\$42.0	\$51.0	\$47.0	\$55.0		\$49.5	\$47.1	\$58.3	\$53.6	\$59.9	\$61.1	\$61.6
Marketing	\$39.0	\$34.0	\$42.0	\$46.0		\$50.4	\$50.3	\$50.3	\$49.1	\$51.2	\$50.8	\$56.2
Management	\$37.0	\$36.0	\$42.0	\$39.0		\$47.9	\$43.9	\$52.0	\$49.0	\$52.4	\$52.2	\$58.0
Analytics (co-major)								\$58.9	\$59.5	\$63.8	\$63.5	\$64.3
Entrepreneurship	Highest			2 <sup>nd</sup> Highest							\$64.5	

Company	# ISA 2020 Grads
Deloitte	15
EY	14
PwC	9
KPMG	5
RSM	5
West Monroe	5
cap Gemini	4
FIS	4
JP Morgan	4
Key Bank	4
Nielsen	4
Protiviti	4
Accenture	3
Crowe LLP	3
DHL	3
Textron	3

# Course Introduction

---

- This course is about how to manage big data
  - *I.e.,* how to help you get those analytics jobs
  - We will learn about all the buzzwords like *big data, analytics, Hadoop, unstructured data, cloud computing, ...*
- But first: what is “Big data”?

# Big Data

---

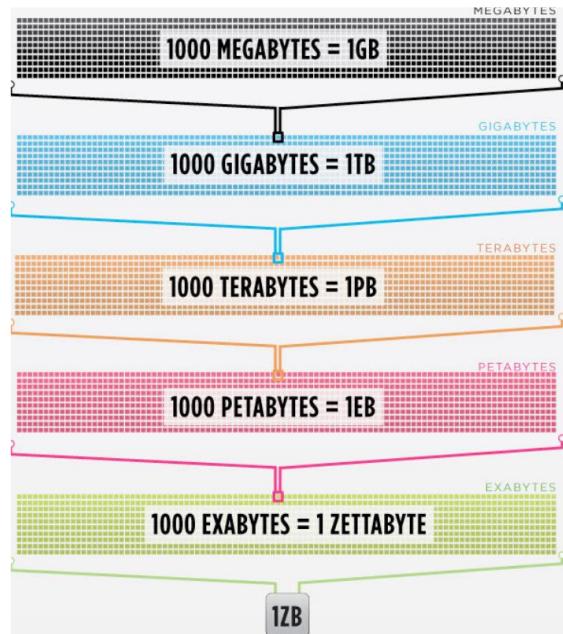
- Traditionally defined by a number of “Vs”
  - Volume: vast amount of data
    - *E.g.*, the Cincinnati Reds deals with 5 TB of new data annually
  - Variety: different forms of data
    - *E.g.*, text, images, voice, geospatial, *etc.*
  - Velocity: speed at which data are generated/analyzed
    - *E.g.*, real-time analysis of tweets
  - Veracity: biases, noise, abnormalities, uncertainties, truthfulness, trustworthiness of the data sources
    - *E.g.*, fake news = fake data

# Big Data

---

## ➤ Volume

- Volume = size



100 MBs ≈ couple of volumes of encyclopedias

A DVD ≈ 5 GBs

1 TB ≈ 300 hours of high quality video

Large Hadron Collider ≈ 15 PBs a year

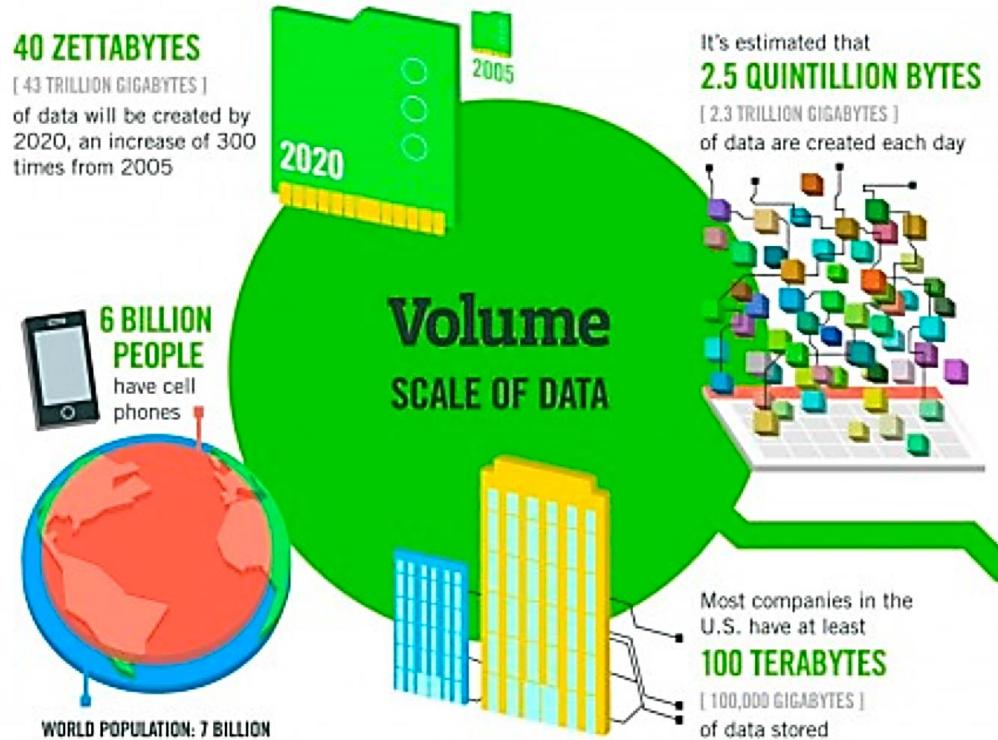
Information created and stored in 2011: 1.8 zettabytes

# Big Data

Source:  
<https://www.ibmbigdatahub.com/infographic/four-vs-big-data>

## ➤ Volume

- More data → More information
  - Safer flights
  - Personalized healthcare treatments
  - Better understanding of consumers' behavior



# Big Data

---

## ➤ Volume

- Challenges associated with volume
  - Time to analyze the data
    - Sampling matters!!!
  - Costs associated with storage
    - Hardware
    - Energy
    - Manpower
    - Physical location
  - Data retrieval
    - Bandwidth

# Big Data

---

## ➤ Variety

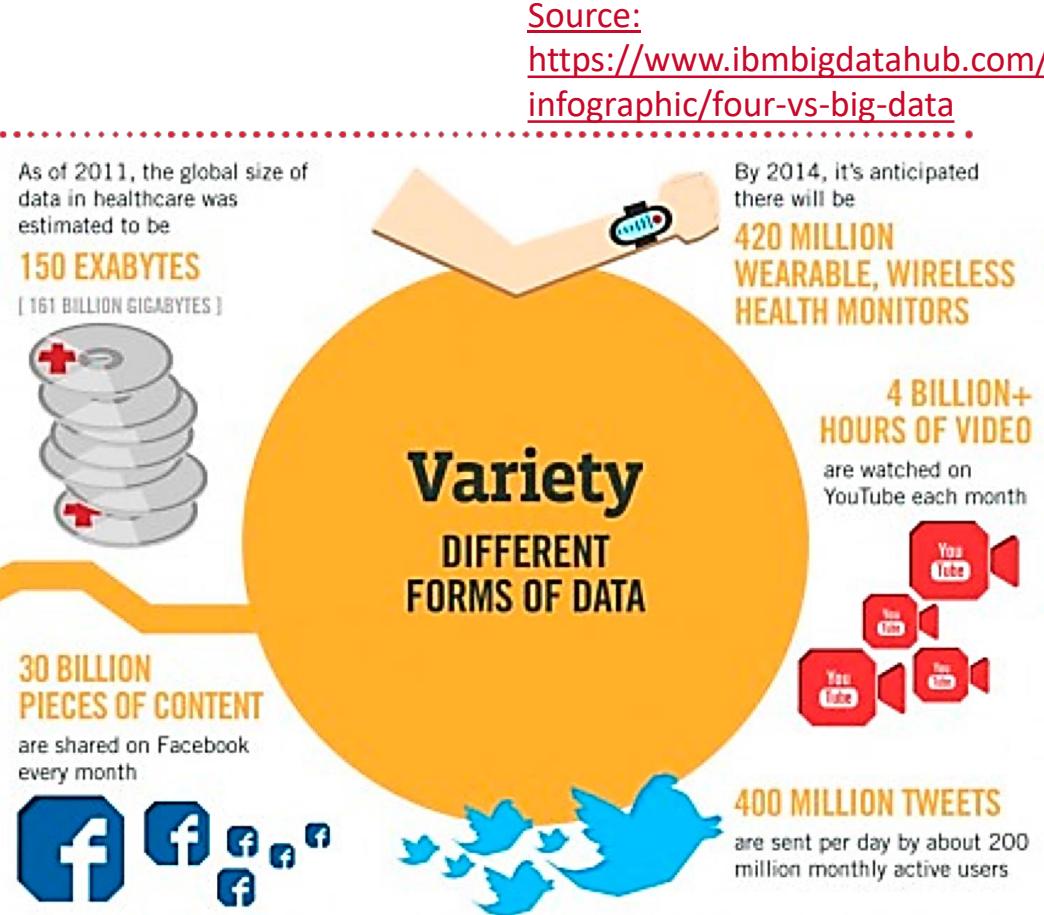
- Variety = structure of the data
- Traditional view of data
  - Well-defined tabular structure
  - Typically stored in relational databases (recall ISA 245)

	Return	SAT	MBA	Age	Tenure
1	-21.10	1211.0	.0	47.0	4.0
2	-6.38	1163.0	.0	43.0	5.0
3	-3.32	1217.0	1.0	39.0	3.0
4	-2.27	1185.0	1.0	35.0	2.0
5	-9.46	1030.0	.0	36.0	5.0
6	-1.28	1141.0	1.0	38.0	4.0
7	4.66	1019.0	1.0	38.0	2.0
8	7.06	1012.0	.0	38.0	4.0
9	-12.39	925.0	1.0	36.0	4.0
10	-10.27	1195.0	.0	46.0	5.0
11	-3.31	845.0	.0	33.0	3.0
12	-7.74	902.0	1.0	46.0	5.0
13	7.49	1076.0	1.0	43.0	6.0

# Big Data

## Variety

- Data are becoming more unstructured and heterogeneous
  - Text
  - Images
  - Videos
  - Maps



# Big Data

---

## ➤ Variety

- Challenges associated with variety
  - Difficult to define a database
    - Relational databases are not always a good solution
  - Difficult to analyze the data
    - Most statistical techniques assume a tabular format

# Big Data

## ➤ Velocity

- Speed of creating, storing, and analyzing data
- Many scenarios require real-time actions
  - *E.g.*, a search on google can immediately trigger a new ad placed on Facebook

The New York Stock Exchange captures  
**1 TB OF TRADE INFORMATION**  
during each trading session

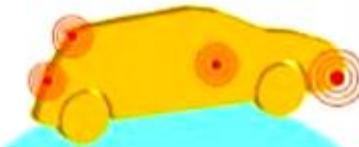


By 2016, it is projected there will be  
**18.9 BILLION NETWORK CONNECTIONS**  
– almost 2.5 connections per person on earth



## Velocity

ANALYSIS OF STREAMING DATA



Source:

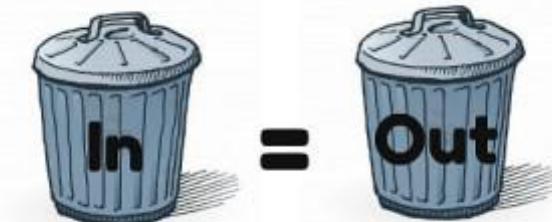
<https://www.ibmbigdatahub.com/infographic/four-vs-big-data>

# Big Data

---

## ➤ Veracity

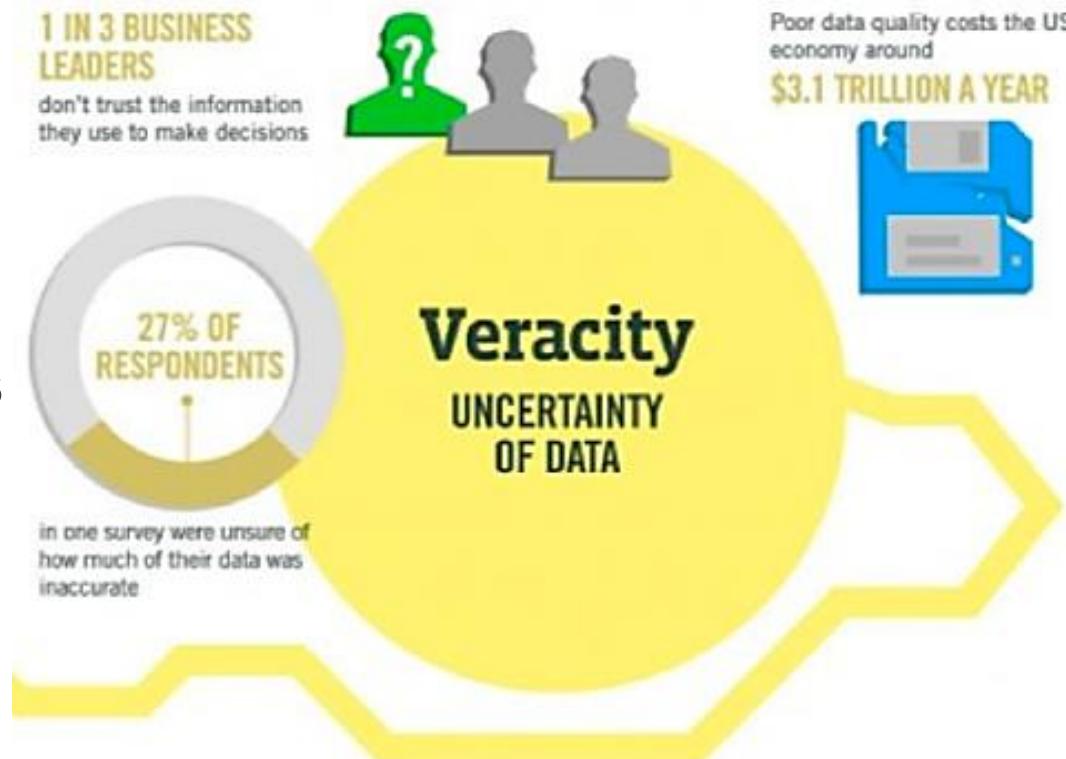
- Veracity = Quality
  - Accuracy of data
  - Reliability of the data source
- Poor-quality data results in poor-quality information and decision
  - *E.g.*, would you spend money advertising chocolate to a person who hates chocolate?



# Big Data

## ➤ Veracity

- Particularly important when data reflect subjective opinions
  - Opinion polls
  - Surveys
  - Reviews



# Big Data

---

- To summarize, big data is equal to:

*Huge amounts of data in different formats and varying quality which often must be processed quickly*

# Big Data Management

---

- Since we now know what big data is about, we can then talk about what this course is about
  - Introduction to big data management
    1. How to effectively **collect, store, and analyze** potentially **unstructured** and **large data sets**
    2. Modern technologies to address the challenges brought by big data
      - Cloud computing and storage (*e.g.*, IBM Cloud)
      - Computing clusters
        - Distributed storage (*e.g.*, HDFS)
        - Parallel, distributed computing paradigms (*e.g.*, Spark)

# Agenda

---

- How are we going to do this?
  - The course is divided into two parts
  - 1<sup>st</sup> part of the course
    - **Preliminaries: Introduction to Programming in Python**
      - Tailored to big-data business problems
      - Question: “*Do I need to (learn how to) program in this course?*”
        - Answer: ABSOLUTELY

# Agenda

---

- How are we going to do this?
  - The course is divided into two parts
  - 1<sup>st</sup> part of the course
    - **Data collection**
      - Web scrapping
      - APIs
        - REST
        - XML
        - JSON
      - NoSQL database (MongoDB)

# Agenda

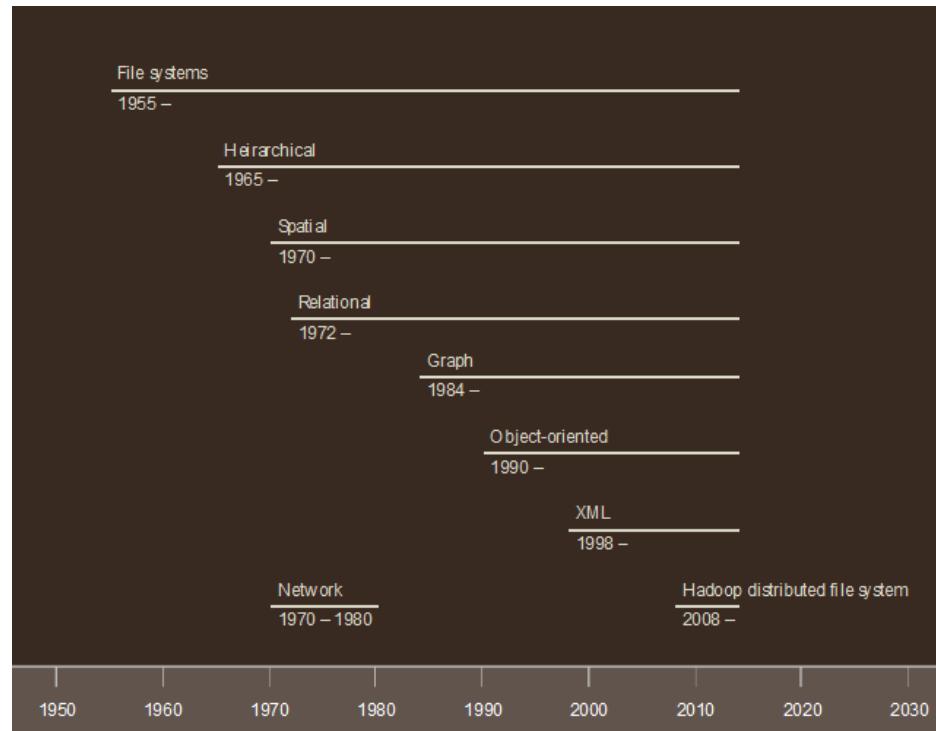
---

- How are we going to do this?
  - The course is divided into two parts
  - 1<sup>st</sup> part of the course
    - **Data analysis**
      - Focus on analysis of textual, unstructured data
        - Supervised learning, sentiment analysis, topic modeling
      - Not as in-depth as ISA 291 and ISA 491

# Agenda

---

- How are we going to do this?
  - The course is divided into two parts
  - 2<sup>nd</sup> part of the course
    - Cloud technologies
    - Hadoop environment
    - No lectures on design and implementation of relational databases



# Agenda

---

- Full schedule on Canvas
  - Class schedule might change depending on our progress
- All homework, assignments, and project deadlines are already available on Canvas
  - Please, prepare your schedule (interviews, trips, ...) accordingly

# Changelog (What is New in Fall 2021)

---

- Course is now entirely in Python
- New development environment (VS Code + Jupyter)
  - Brand new updates (August 2021)
- No more MapReduce classes
  - More emphasis on Spark
- New topic modeling class

# Outcomes of the Course

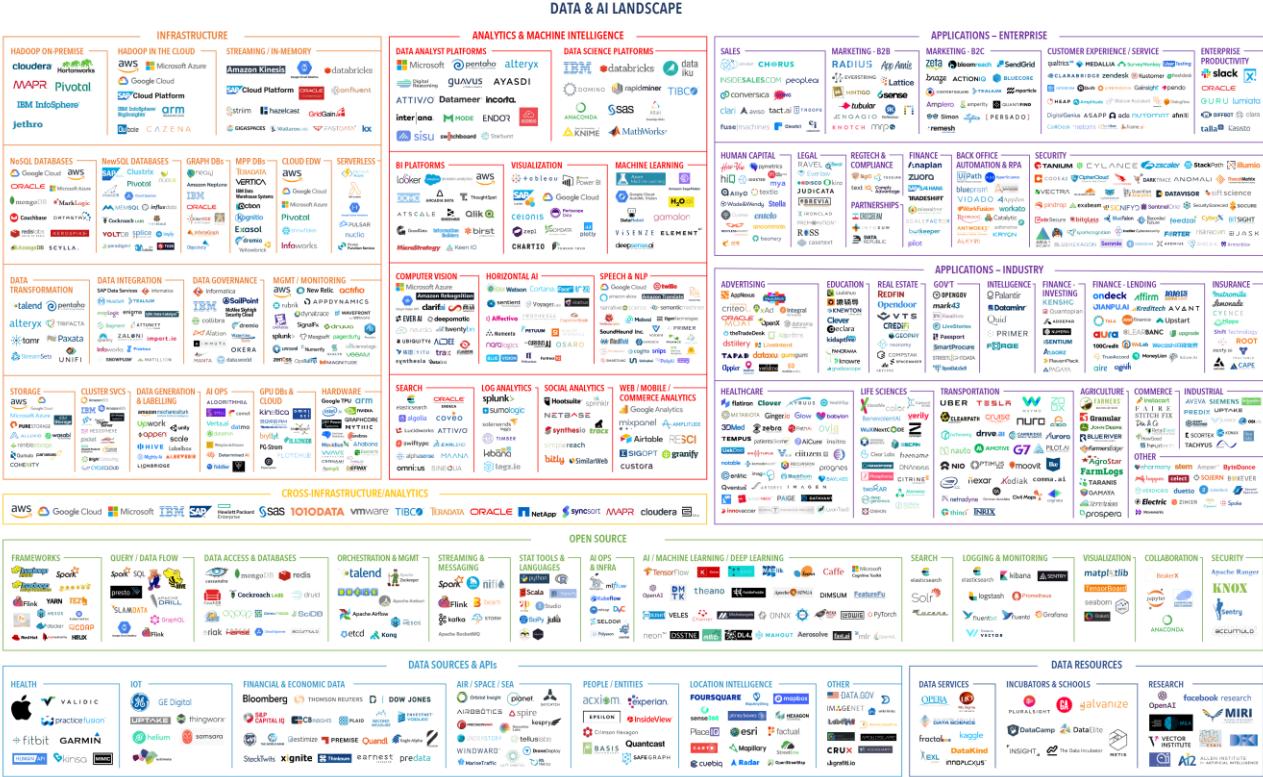
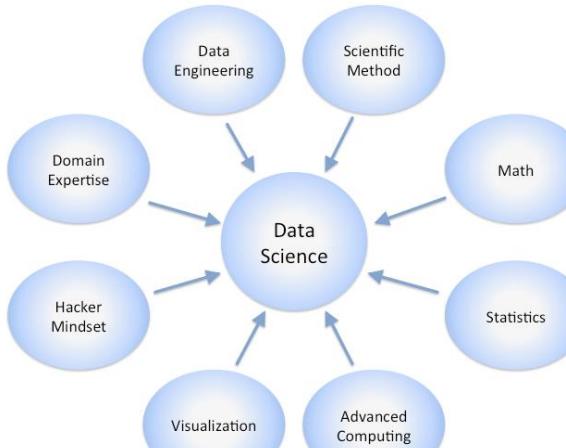
---

- To summarize, this course teaches core analytics skills
  - Data collection, data analysis, distributed storage and computing, cloud storage and computing
  - Highly relevant skills in great demand in today's job market
    - Previous students are working at Disney Streaming Services, EY, IBM, Progressive, Textron, Zillow, ...

# Managing Expectations

- The diagram illustrates the Data & AI Landscape across three main categories:

  - INFRASTRUCTURE** (Left):
    - Hadoop On-Premises: cloudera, Hortonworks, MAPR, Pivotal
    - Hadoop in the Cloud: AWS, Microsoft Azure, Google Cloud
    - Streaming / In-Memory: Amazon Kinesis, datarailics, StreamCloud Platform, Streamly
  - DATA & AI LANDSCAPE** (Center):
    - DATA ANALYST PLATFORMS**: Microsoft, pentaho, alteryx, guavus, AYASDI
    - DATA SCIENCE PLATFORMS**: IBM, datarailics, iku, insidesales.com, people.ai
  - APPLICATIONS – ENTERPRISE** (Right):
    - SALES**: salesforce, CHIRUS, Zoho, Radius, AppAnnie, eMarketer, Bluecore, Lattice
    - MARKETING – B2B**: Marketo, Pardot, MarketCloud, Zendesk, Zoho, Radius, AppAnnie, eMarketer, Bluecore, Lattice
    - CUSTOMER EXPERIENCE**: Zendesk, MarketCloud, Marketo, Pardot, MarketCloud, Zendesk, Zoho, Radius, AppAnnie, eMarketer, Bluecore, Lattice



# Managing Expectations

- The perfect data scientist is like a unicorn or a sheep of five legs
  - Teamwork is crucial!!!
- Arguably the most important skill
  - Curiosity, aka, the **hacker mindset**

## MODERN DATA SCIENTIST

Data Scientist, the sexiest job of the 21th century, requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.

### MATH & STATISTICS

- ★ Machine learning
- ★ Statistical modeling
- ★ Experiment design
- ★ Bayesian inference
- ★ Supervised learning: decision trees, random forests, logistic regression
- ★ Unsupervised learning: clustering, dimensionality reduction
- ★ Optimization: gradient descent and variants



### PROGRAMMING & DATABASE

- ★ Computer science fundamentals
- ★ Scripting language e.g. Python
- ★ Statistical computing packages, e.g., R
- ★ Databases: SQL and NoSQL
- ★ Relational algebra
- ★ Parallel databases and parallel query processing
- ★ MapReduce concepts
- ★ Hadoop and Hive/Pig
- ★ Custom reducers
- ★ Experience with xaaS like AWS

### DOMAIN KNOWLEDGE & SOFT SKILLS

- ★ Passionate about the business
- ★ Curious about data
- ★ Influence without authority
- ★ Hacker mindset
- ★ Problem solver
- ★ Strategic, proactive, creative, innovative and collaborative

### COMMUNICATION & VISUALIZATION

- ★ Able to engage with senior management
- ★ Story telling skills
- ★ Translate data-driven insights into decisions and actions
- ★ Visual art design
- ★ R packages like ggplot or lattice
- ★ Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau

# Managing Expectations

---

## ➤ The hacker mindset

- You are expected to actively research and look for more information outside the classroom
  - **Most of your learning will take place outside the classroom**
    - So, you must learn how to learn
  - From an INFORMS (a prestigious analytics professional association) panel:

*The panelists indicated that the ability to learn a new tool is more important than learning a particular tool/software, as software selection will continuously change and evolve over time. Upon graduation from an analytics program, the students are expected to acquire some popular tools, such as R, to build a “toolbox” that they can consistently build upon. However, students need to realize that the existing “toolbox” is always going to grow in the future, and new knowledge has to be timely updated based on their working experience and requirements to ensure their success beyond college.*

# Managing Expectations

---

- Is this course the right course for me?
  - The answer is **no** if one of the following is true:
    - You dislike programming and/or are not willing to learn about it
    - You do not like learning about new technologies
    - You are expecting an easy, not demanding course
- My expectations of you are very high. In previous years:
  - We made the news
    - <http://miamioh.edu/fsb/fsbnews/?up=/news/170526142334%20IBM-Watson-Carvalho-class>
  - We wrote academic papers
    - “*Off-The-Shelf Artificial Intelligence Technologies for Sentiment and Emotion Analysis: A Tutorial on Using IBM Natural Language Processing*”
    - “*Understanding What Drives Bitcoin Trading Activities*”

# Lectures

---

- All the material used in each lecture (slides, code, data) are posted on Canvas before the lecture
- Class attendance will not be taken
  - You are still expected to attend every class

# Evaluation

Item	Weight
Assignments	20%
Homework	25%
Midterm project	20%
Final project	25%
Final exam	10%

- 10 individual, equally-weighted **homework**
  - Released after a few classes
    - Deadline: about 36 hours after being released
  - First one is already on Canvas
  - Goal: make sure you are on top of the game

# Evaluation

---

- 4 individual, equally-weighted **assignments**
  - More complex than homework
    - Cover more than one topic
  - Deadline: about 3 to 7 days after being released
- 1 individual, conceptual **final exam**
  - You can take the exam anytime during the finals week

# Evaluation

---

## ➤ Midterm Project

- Guided project, *i.e.*, everybody works on the same topic
  - Collect, store, and analyze data in real-time from a social network
- Groups of 2

## ➤ Final Project

- The most important evaluation component in this course
- You will define a complex business/research problem and propose and implement an analytics solution
  - Very high expectations
- Groups of 4

# Grades

---

- Final letter grades will be assigned according to the following scale →
  - Not the standard FSB scale
  - **You should not expect grades to be rounded up ...**
    - ... I might nonetheless raise the grades of committed students
      - For example, if you are actively asking and answering questions, attending classes, *etc.*
      - This is a unilateral decision; So, please do not email me

Percentage Grade	Letter Grade
[97, 100]	A+
[94, 97)	A
[90, 94)	A-
[87, 90)	B+
[84, 87)	B
[80, 84)	B-
[77, 80)	C+
[74, 77)	C
[70, 74)	C-
[67, 70)	D+
[64, 67)	D
[60, 64)	D-
[0, 60)	F

# Textbook

---

- This course has no official textbook
  - All the materials will be shared on Canvas
    - Slides from lectures
    - Source codes
- Copyright notice (from the syllabus)

“The course materials provided to you, including presentations, tests, outlines, and similar materials, are copyright protected by the faculty member teaching this course. You may make copies of course materials solely for your own use. **You may not copy, reproduce or electronically transmit any course materials to any person or company for commercial or other purposes without the faculty member’s express permission.** Violation of this prohibition may subject the student to discipline/suspension/dismissal under the Miami’s Code of Student Conduct or Academic Integrity Policy. **By taking this course, you also explicitly acknowledge that you will not share your own solutions and deliverables with others beyond the course instructor, including uploading the same to third-party websites.”**

# Instructor – Dr. Arthur Carvalho

---

- I am not into titles
  - Feel free to call me Arthur
  
- Office hours (**FSB 3024**)
  - Tuesday and Thursday 3:00 pm to 4:30 pm



# End of Lecture 1

---

- Do it before next class
  - Complete Homework 1
  
- Next Class
  - Introduction to Python and VS Code

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 2 – Preliminaries (Part I)

*VS Code, Jupyter Notebooks, Markdown*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Lecture Objectives

---

- Learn the basics of VS Code
- Learn about Jupyter Notebooks
- Learn basic Markdown notation

# Introduction to Python

---

## ➤ What is Python?

- General-purpose programming language
- Contrast the above to R
  - An environment for statistical computing, data visualization, and data analytics



## ➤ Python is (arguably) the most popular programming language for data analytics

- The programming language we use in this course

# Introduction to Python

---

- But what is a programming language?
  - It is a **formal language** used to program computers
    - **Syntax:** set of rules that define the combinations of symbols that are considered to be correctly structured in a language
    - **Semantics:** meaning of constructs in a language
- Unlike **natural languages**, formal languages are very rigid
  - Any syntax violation will produce errors

# Introduction to Python

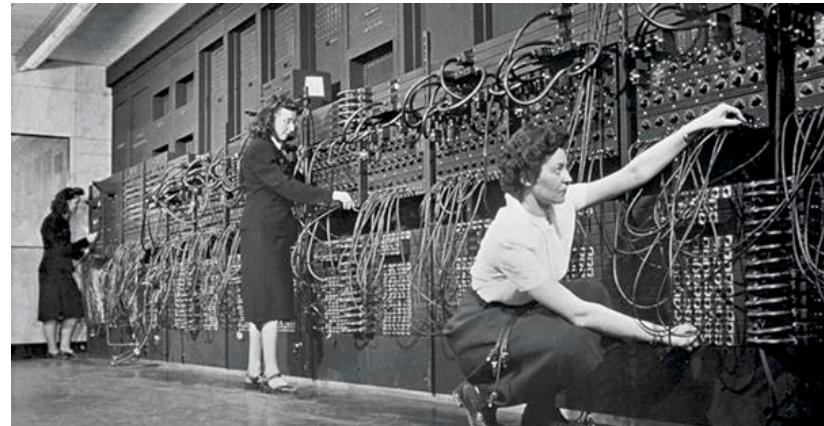
---

- How can one use a language to program a computer? (CS 101)
  - Computers only understand 0s and 1s
    - Absence/presence of electricity
  - Sequences of 0s and 1s define numbers
    - *E.g.*, 0000000000000001 = 1, 00000000000000010 = 2
  - The central processing unit (CPU) of a computer contains arithmetic logic unit (ALU), capable of performing arithmetic operations
    - Arithmetic: basis of modern mathematics

# Introduction to Python

---

- How can one use a language to program a computer?
  - Turn electricity on/off
    - Produce sequences of 0s and 1s
    - Not very productive



# Introduction to Python

---

- How can one use a language to program a computer?
  - Low-level programming language
    - Easier to understand than 0s and 1s, but still incredibly hard to work with
    - Very strong correspondence between the language and machine code instructions
    - Very efficient
    - *E.g., Assembly X86*

pushl	%ebp	// push ebp onto stack
movl	\$1, %eax	// eax=1 (return value if n==0)
movl	%esp, %ebp	// ebp now points to top of stack
subl	\$8, %esp	// allocate 8 bytes on top of stack
movl	%ebx, -4(%ebp)	// store ebx on stack (see .L1 for restoring)
movl	8(%ebp), %ebx	// ebx = n (1 <sup>st</sup> argument of function)
testl	%ebx, %ebx	// if n==0, set zero flag (0 flag means equal, so no jump if n==0)
ine	.L5	

# Introduction to Python

---

➤ How can one use a language to program a computer?

- High-level programming language
  - Easier to understand than machine code and low-level languages
  - Low correspondence between the language and machine code instructions
  - Not as efficient as low-level languages, but easy to program
  - *E.g.*, Python:

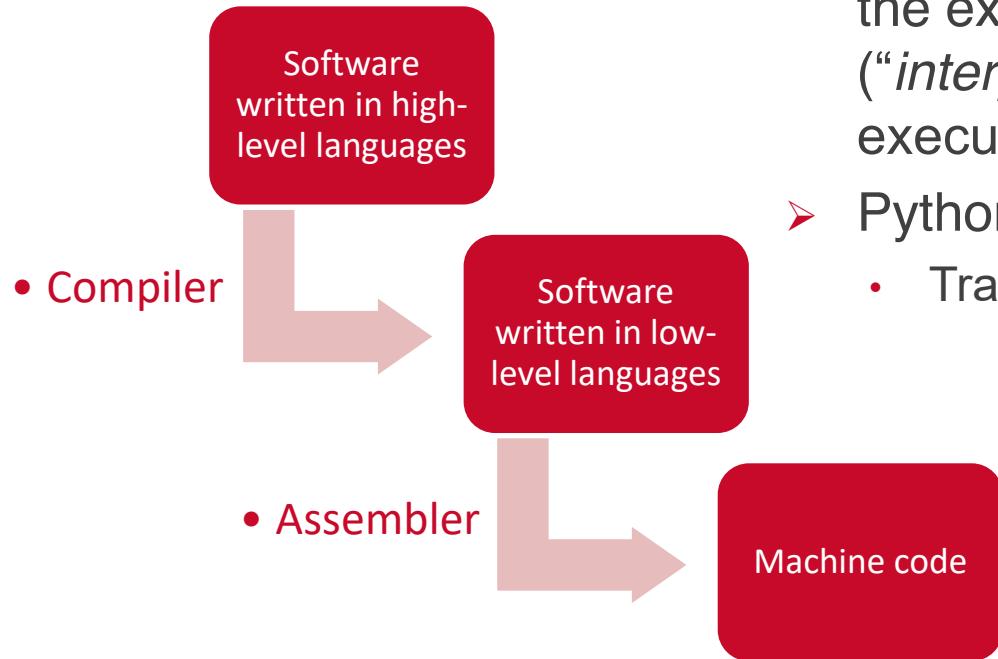
```
x = 5
if x > 0:
    print("Non-negative number")
else:
    print("Negative number")
```

# Introduction to Python

---

- How can one use a language to program a computer?
  - Software written in low-level programming languages are more efficient, but more difficult to program
    - Closer to machine code
  - Software written in high-level programming languages must be translated to low-level languages by specialized software
    - *Interpreters, Compilers, and Assemblers*

# Introduction to Python



- This translation process can happen during the execution of the software (“*interpretation*”) or only once before execution (“*ahead-of-time compilation*”)
- Python is a high-level interpreted language
  - Translation “on-the-fly”



# Introduction to Python

---

- It is commonplace to use specialized software known as **IDE** (Integrated Development Environment) to help with programming tasks
  - The *de facto* IDE for R is called **RStudio**
  - We will use the IDE called **Visual Studio (VS) Code** for Python
- Python and VS Code are already installed on the lab computers and remote desktop
- To run Python and VS Code on your own computer:
  1. Download and install Python (version 3.8+)
    - <https://www.python.org/downloads/>
  2. Download and install VS Code
    - <https://code.visualstudio.com/download>

---

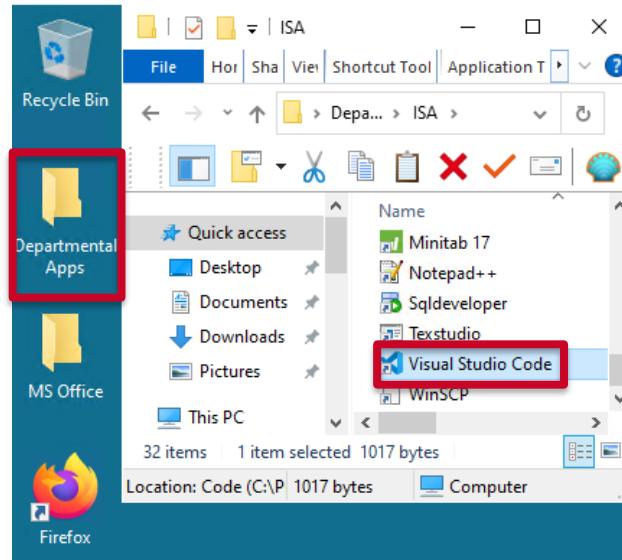
# INTRO TO VS CODE

# Introduction to VS Code

---

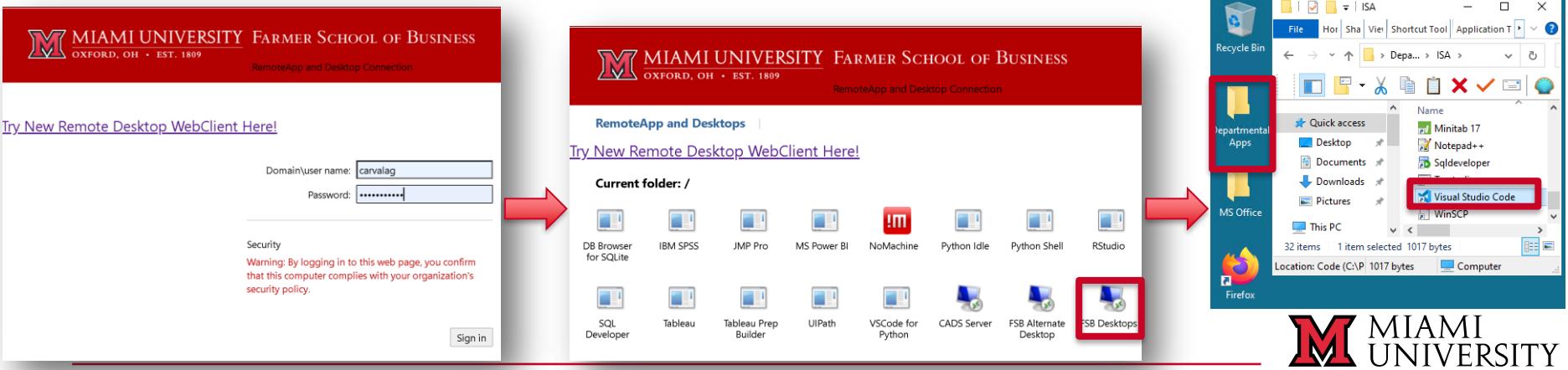
➤ Open VS Code now

- Open the file at “*Departmental Apps*” -> *ISA* -> “*Visual Studio Code*”



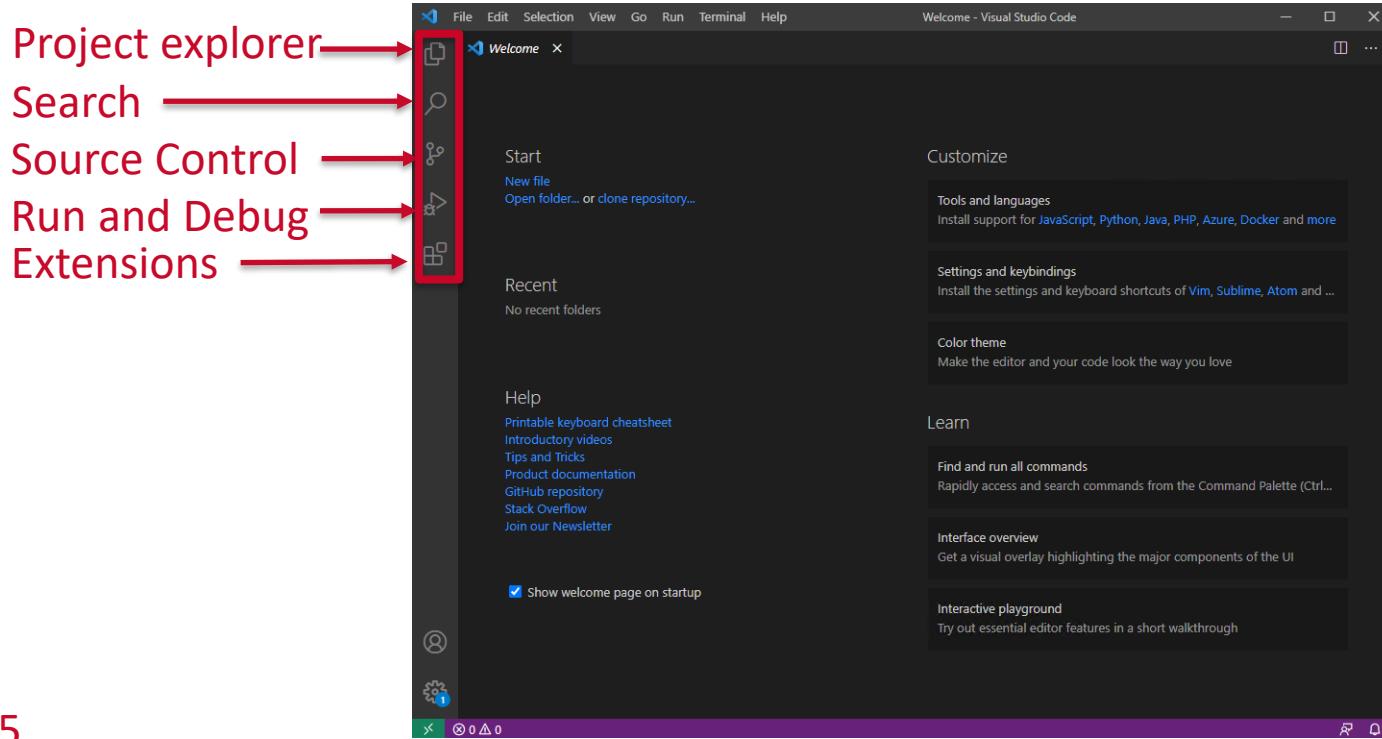
# Introduction to VS Code

- You can also access VS Code remotely
  - Go to <https://virtualpc.fsb.miamioh.edu/> and sign in
  - Select “FSB Desktops”
  - Open the downloaded file and sign in again
  - Open VS Code at “Departmental Apps” -> ISA -> “Visual Studio Code”



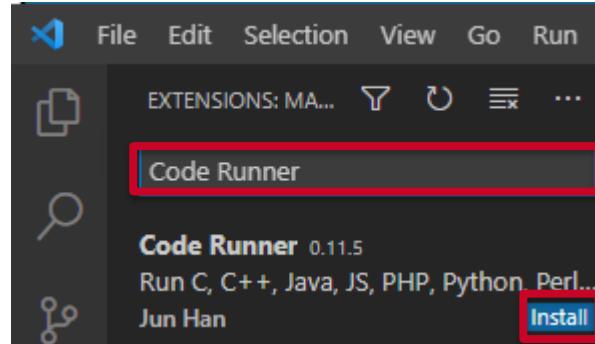
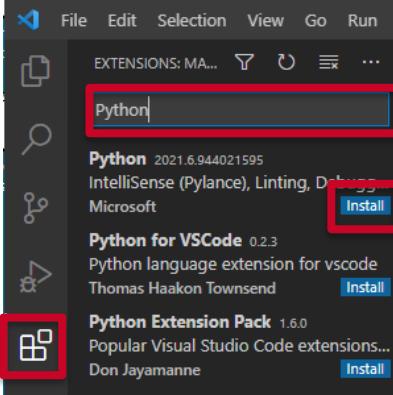
# Introduction to VS Code

- This is the initial screen at the time of writing



# Introduction to VS Code

- VS Code can be used to program in virtually any programming language
  - The appropriate extensions must be installed first
  - Let's install three extensions: *Python*, *Jupyter*, and *Code Runner*
    - Click on the *Extensions* icon, type *Python*, and click on *Install*
    - Do the same for *Code Runner* and *Jupyter*



# Introduction to VS Code

---

- You can always change the appearance of VS Code
  - For example, go to *File* -> *Preferences* -> “*Color Theme*” to change the background color
  
- You can always search for commands using the *search bar*
  - Press F1 for the search bar to appear

.....

# INTRO TO PYTHON IN VS CODE

# Introduction to Python

---

- We will learn how to run Python programs in three different ways
  - Non-interactively
  - Interactively
  - Jupyter notebooks

# 1) Non-Interactive Python

---

- Let's create our first python file
  - Go to *File* -> “*New File*”
  - Type the following:

```
x = "Hello ISA 414"  
print(x)
```
  - Go to *File* -> *Save*
    - Save the file as *hello.py*
    - Because of the *.py extension*, VS Code now recognizes this is a Python file

# 1) Non-Interactive Python

➤ Let's run our first Python code now

- Click on the “play” button (top right) and select “Run Code”

```
File Edit Selection View Go Run Terminal Help hello.py - Visual Studio Code
hello.py  x
C: > Users > carvalag > hello.py > ...
1  x = "Hello ISA 414"
2  print(x)
3

Run Code
Run Python File in Terminal
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Code [Running] python -u "c:\Users\carvalag\hello.py"  
Hello ISA 414  
[Done] exited with code=0 in 0.19 seconds

output from our code

# 1) Non-Interactive Python

---

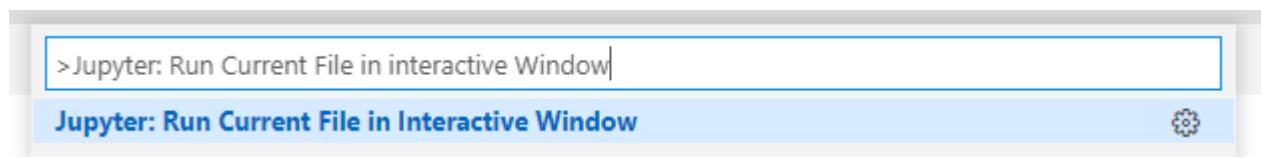
- The previous procedure will run the entire code
  - One can run only a few lines by selecting these lines and running the code
  - Highlight the first line and run the code again
    - Nothing is printed because the print function was not selected
  - What happens if one subsequently highlights the second line and run the code again?
    - An error occurs
    - **x** is not defined

```
[Running] python -u "c:\Users\carvalag\tempCodeRunnerFile.py"
Traceback (most recent call last):
  File "c:\Users\carvalag\tempCodeRunnerFile.py", line 1, in <module>
    print(x)
NameError: name 'x' is not defined
```

## 2) Interactive Python

---

- The previous error happened because we ran Python non-interactively
  - The environment forgets everything in between runs
  
- Let's now run Python interactively
  - Press F1 and search for “*Jupyter: Run Current File in Interactive Window*”



## 2) Interactive Python

---

- You can not type commands at the bottom of the environment
  - For those familiar with R
    - This is similar to how the console in RStudio works

The screenshot shows the Jupyter Notebook interface with a single tab titled "Interactive-1". The toolbar includes buttons for "Clear All", "Restart", "Interrupt", and "Python 3.9.5 64-bit". The main area displays the startup message for the Python 3.9.5 64-bit kernel, followed by a code cell containing the Python command "x = "isa"" and its continuation "... isa". At the bottom of the code cell, there is a red rectangular box highlighting the placeholder text "Type code here and press Shift+Enter to run".

## 2) Interactive Python

---

### ➤ Let's test the previous environment

- Type `y = x + " and ISA 515"` at the bottom of the environment
  - Press the keys *shift + enter* to run the code
  - See how the code now knows what `x` is
- How do we know the current values of `x` and `y`?



# 3) Jupyter Notebooks

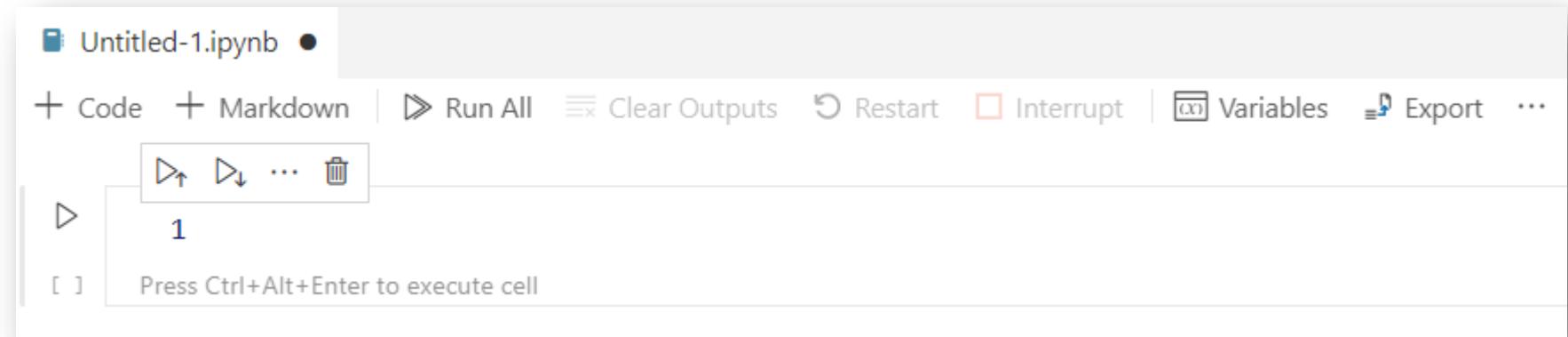
---

- The third way of running Python applications will be through **Jupyter notebooks**
- What is a notebook?
  - Notebooks are incredibly popular in data science
  - Among many other things, they allow analysts to write well-formatted text, code, and results (graphs, tables, ...) all in one place
    - This is great to enhance *readability* and *reproducibility*
- There are some notebooks in Python
  - We shall use Jupyter notebooks embedded in VS Code

# 3) Jupyter Notebooks

---

- Let's create our first Jupyter notebook
  - Press F1 and search for *Jupyter: Create New Blank Notebook*



# 3) Jupyter Notebooks

---

## ➤ Notebooks have **cells**

- For our purposes in this course, each cell will be either text written in a language called **Markdown** or code written in **Python**
- Our first cell is a Python cell
  - Let's add the following code to it

```
x = 1  
print(x)  
x = x + 1  
print(x)
```

A screenshot of a Jupyter Notebook interface. At the top, there are tabs for 'Code' (selected), 'Markdown', 'Run All', and 'Python 3.9.5 64-bit'. Below the tabs is a toolbar with icons for running cells, inserting cells, and deleting cells. A code cell contains the following Python code:

```
1 x = 1  
2 print(x)  
3 x = x + 1  
4 print(x)
```

At the bottom of the cell, it says 'Press Ctrl+Alt+Enter to execute c'. To the right of the cell, a red box highlights the 'Python' tab.

### 3) Jupyter Notebooks

---

- Click on the play button on the top-left of the cell to run the entire code (or press ctrl + enter)

A screenshot of a Jupyter Notebook cell. The cell contains the following Python code:

```
1 x = 1
2 print(x)
3 x = x + 1
4 print(x)
```

The cell has a status bar at the bottom indicating [2] and 0.7s. Above the cell are buttons for '+ Code', '+ Markdown', and 'Run All'. A red bracket labeled 'Code' points to the first four lines of the code. A red bracket labeled 'Output' points to the two lines of output: '1' and '2'.

- See how the code and output are close together

# 3) Jupyter Notebooks

- Click on the “+ Code” button to add another cell to your notebook
- Add the following code to and run the second cell

```
y = x + 1  
print(y)
```

- “Run All” cells
  - See how notebooks are interactive
  - A variable in one cell can be called from another

A screenshot of a Jupyter Notebook interface. At the top, there are tabs for "Code" (highlighted with a red box), "Markdown", "Run All", and "Python 3.9.5 64-bit". Below the tabs, there is a toolbar with icons for running cells, saving, and deleting. The main area shows two code cells. The first cell contains the following Python code:

```
1 x = 1  
2 print(x)  
3 x = x + 1  
4 print(x)
```

Below the code, it says "Press Ctrl+Alt+Enter to execute cell". The "+" Code button is also highlighted with a red box.

A screenshot of the Jupyter Notebook interface after running all cells. The "Run All" button is highlighted with a red box. The notebook now shows the output of the second cell, which is the value 2.

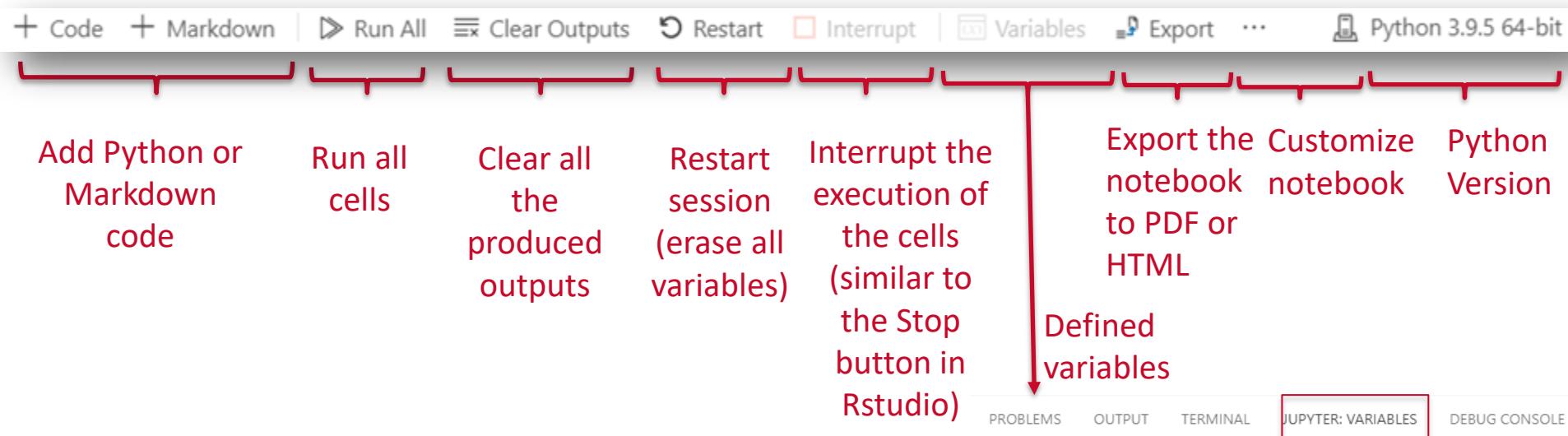
A screenshot of the Jupyter Notebook interface showing the results of running all cells. The output of the second cell is displayed below the first cell's output. The notebook then shows a new cell with the following code:

```
1 y = x + 1  
2 print(y)
```

The "+" Code button is highlighted with a red box. The output of this new cell is shown below, with a green checkmark and the time "0.2s".

# 3) Jupyter Notebooks

## ➤ Understanding the top buttons



Name	Type	Size	Value
x	int		2
y	int		3

### 3) Jupyter Notebooks

---

- Saving your Jupyter Notebook to a `.ipynb` file
  - Go to the menu item *File* -> *Save*
- Opening a Jupyter Notebook
  - Go to the menu item *File* -> *Open* and select the file
- Moving forward
  - I will share a Jupyter notebook file with you every single class
  - Available on Canvas at least 12h before class

---

# INTRO TO MARKDOWN

# Markdown

---

- Technically speaking, notebook cells can be of different types
  - We focus on two: Python and Markdown
- Markdown: markup language to format texts
  - We barely scratch the surface of what can be done with markdown
    - Enough to produce neat reports/deliverables
  - Quick reference: <https://www.markdownguide.org/cheat-sheet/>

# Markdown

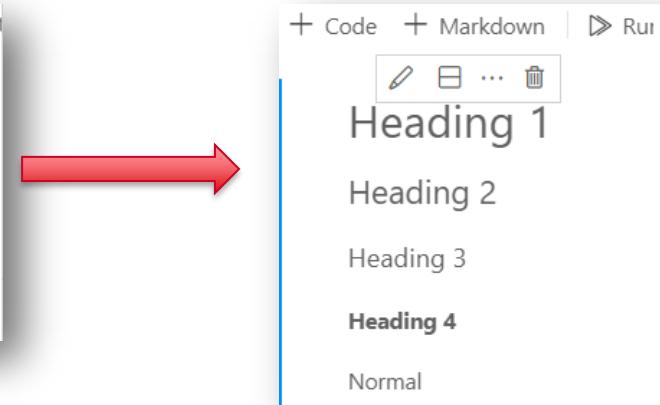
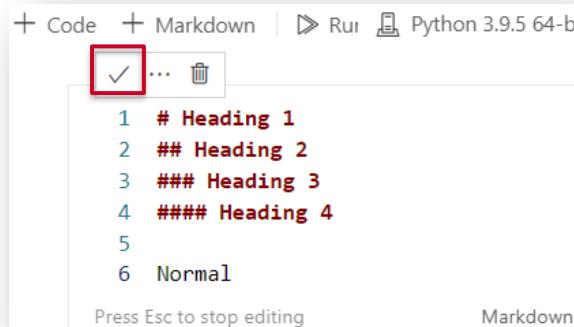
---

## ➤ Create a markdown cell

- Add some code to your cell (look at the '#' symbol)
  - Run the code (ctrl + enter or check button)

```
# Heading 1  
## Heading 2  
### Heading 3  
#### Heading 4
```

Normal text



# Markdown

---

- Let's try to mix together Python and Markdown
  - Create a markdown, followed by a Python, and finally another Markdown cell
  - Add the following code to and run all the cells:

Cell #1 ⌂ # This is my \*first\* \*\*notebook\*\*

Cell #2 ⌂ x = 1

The code `x=1` assigns the value 1 to `x`

And here is an old image of ISA Faculty:

Link: [ISA

Cell #3 Professors]([https://www.miamioh.edu/fsb/\\_files/images/isa/isameet-800x400.jpg](https://www.miamioh.edu/fsb/_files/images/isa/isameet-800x400.jpg))

![ISA Professors]([https://www.miamioh.edu/fsb/\\_files/images/isa/isameet-800x400.jpg](https://www.miamioh.edu/fsb/_files/images/isa/isameet-800x400.jpg))

# Markdown

---

## ➤ Why use Markdown?

- *De facto* language to report data analysis and for technical documentation
  - For example, it is used on GitHub
- Markdown is portable
  - Unlike Microsoft Word or similar software that lock content into a proprietary file format
- Markdown is platform independent
- You will see the power of Python + Markdown inside Jupyter notebooks as we progress

# Summary

---

- We have learned the basics of VS Code and Markdown
  - More about Markdown: <https://www.markdownguide.org/>
- Next lecture: Preliminaries (part II)
  - How to use built-in functions
  - How to declare variables
  - How to control the flow of a Python code

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 3 – Preliminaries (Part II)

*Built-in Functions, Variables, and Control Flow*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Lecture Objectives

---

- Learn basic commands in Python
  - How to declare variables
  - How to use built-in functions
  - How to control the flow of Python scripts

# Lecture Instructions

---

1. Download the notebook “*Lecture 3.ipynb*” available on Canvas
2. Open the file “*Lecture 3.ipynb*” with VS Code

# Introduction to Python

---

- Python was released in 1991
- We will focus on Python as used in data science
  - For example, using Jupyter notebooks
  - There is much more to it
- Currently, Pythons can be used for:
  - Data analytics
  - Scientific calculations
  - System/software development
  - Web development
  - ...

# Introduction to Python

---

- Jupyter notebooks will abstract away many difficulties when working with Python
  - *E.g.*, the need to work with command lines
  - Consequently, we do not experience the full power of Python in this course
- As we progress, keep in mind that Python uses new lines to complete a command
  - Other programming languages often use semicolons or parentheses

---

# VARIABLES

# Introduction to Python

---

## ➤ Variables

- Can be used to store previously calculated values
- *E.g.*, saving the result of  $1+1+1$  to a variable called `x`

$$x = 1 + 1 + 1$$

Name	Type	Size	Value
x	int		3

- Note that the user-defined variable `x` can now be used in future computations
  - *E.g.*, try the following command: `print(x + 2)`

# Introduction to Python

---

- We saw the traditional way of assigning **values** to **variables**
  - One value to one variable
- Python allows one the declare variables in different ways
  - Many values to many variables
    - *E.g., x, y, z = 1, 2, 3*
  - One value to many variables
    - *E.g., a = b = c = 10*
  - Many values to one variable
    - *E.g., lists* (see future classes)

# Introduction to Python

---

- Python has several built-in variable/data structure types

Text Type:	<code>str</code>
Numeric Types:	<code>int, float, complex</code>
Sequence Types:	<code>list, tuple, range</code>
Mapping Type:	<code>dict</code>
Set Types:	<code>set, frozenset</code>
Boolean Type:	<code>bool</code>
Binary Types:	<code>bytes, bytearray, memoryview</code>

Covered today

Covered in the future

Not covered in this course

- So, where is “data frame”?
  - Not a built-in data structure (see future lectures)
- One can use the `type` function to determine the type of a variable
  - Examples in the following slides

# Introduction to Python

---

## ➤ Numeric variables

- Store numbers
- Try the following:
  1. Assign the value **10.5** to a variable called **x**
  2. Apply the function **type** to **x**
  3. What is the result?
- Next, try the following:
  1. Assign the value **10** to a variable called **x**
  2. Apply the function **type** to **x**
  3. What is the result?

# Introduction to Python

---

## ➤ Boolean variables

- Store Boolean values (`True` or `False`) often resulting from logical comparisons
  - Internally, `True` = 1, `False` = 0
- Try the following:
  - Assign the value `10 > 5` to a variable called `x`
  - Assign the value `-20 > 0` to a variable called `y`
  - What is the value of `x`? What is the value of `y`?
  - What is the type of `x`? What is the type of `y`?

# Introduction to Python

---

## ➤ Boolean variables

- Standard logical operators: `==` (equal), `!=` (not equal) `and`, `or`, `not` (negation)
- Try the following:
  - Set the variables `x = True` and `y = False`
  - What is the value of `x == y` ?
  - What is the value of `x != y` ?
  - What is the value of `x and y` ?
  - What is the value of `x or y` ?
  - What is the value of `not x` ?
  - What is the value of `x + 5`?
  - What is the value of `y - 4`?

# Introduction to Python

---

## ➤ String variables

- Used to represent string values (texts) in Python
  - Delimited by quotes (either single or double)
- Crucial when working with textual data
  - More details in the future
- Example: `my_first_string = "ISA 414/515"`

Name	▲ Type	Size	Value
<code>my_first_string</code>	str	11	ISA 414/515

# Introduction to Python

---

- Why do attribute types matter?
  - Certain operations require specific attribute types
  - Without running any code, answer the questions below:
    - How much is "Arthur" + "Carvalho" ?
    - How much is `pow("Arthur", 2)`, i.e.,  $\text{Arthur}^2$ ?
    - How much is `(True + False)*True` ?

# Introduction to Python

---

- One can always try to coerce an attribute type into another one by using the functions `int`, `float`, `bool`, and `str`
  - Example:

```
w = "1"  
x = int(w)  
y = float(x)  
z = bool(x)
```

Name	▲ Type	Size	Value
w	str	1	1
x	int		1
y	float		1.0
z	bool		True

---

# BUILT-IN FUNCTIONS

# Introduction to Python

---

- Python comes loaded with built-in functions
  - A function is a block of code that only runs when it is **called**
  - One can **pass** data, known as **arguments** or **parameters**, into a function
  - A function can **return** data as a **result**
- We have already seen and used a few functions
  - `type()`, `str()`, `int()`, `float()`, and `bool()`
- Another crucial function is **print()**
  - Print the value of variables
  - Example: `print(x)`

# Introduction to Python

---

## ➤ Documentation

- Typing the question mark (?) character after a built-in function name returns the documentation associated with that function
- Example: try `pow?`

**Signature:** `pow(base, exp, mod=None)`

**Docstring:**

Equivalent to `base**exp` with 2 arguments or `base**exp % mod` with 3 arguments

Some types, such as `ints`, are able to use a more efficient algorithm when invoked using the three argument form.

**Type:** `builtin_function_or_method`

# Introduction to Python

---

## ➤ Applying predefined functions

- Most functions we will be using in this course have more than one argument
- For example, `pow`
  - First argument = base
  - Second argument = exponent
  - The notation `mod=None` means that if the third argument is not provided, then the **default value** is *none*

Signature: `pow(base, exp, mod=None)`

Docstring:

Equivalent to `base**exp` with 2 arguments or `base**exp % mod` with 3 arguments

# Introduction to Python

---

- When one calls a function, the order of the arguments matter, otherwise unexpected behavior might occur
  - For example: `pow(10, 2)` is not the same as `pow(2, 10)`
- Make sure the order of the arguments is the same as the order listed in the documentation of the function
- Another option is to explicitly use the name of the arguments when calling a function
  - The order of the arguments no longer matters
  - Example: `pow(base = 10, exp = 2)`      *Signature: pow(base, exp, mod=None)*  
`pow(exp = 2, base = 10)`



# CONTROL FLOW

# Introduction to Python

---

- Our code has been very structured thus far
  - One command after the other
- We can specify conditions that must be satisfied before our code is executed
  - Control the flow of the code
- **CRUCIAL POINT TO REMEBER**
  - Python relies on **indentation** to define “scope” (blocks of code)
    - *E.g.*, the scope of loops and functions
    - Other languages, such as R, use curly brackets
  - **Indentation** = spaces at the beginning of a code line

# Introduction to Python

---

- The best way to define indentation is by using the **tab** (tabulator) key
  - By default, it creates 4 spaces in VS Code
    - This can be changed
  - By using tab, you do not have to memorize how many spaces you are using
- Technically speaking, any number of spaces is fine
  - As long as that number is the same inside each block of code

# Introduction to Python

---

## ➤ Control flow

- There are three major ways of controlling the order in which individual commands are executed in Python

### 1. IF-ELSE statement:

```
if <logical_expression>:
```

```
    ...
```

Syntax:      `elif <logical_expression> :`

```
    ...
```

```
else:
```

```
    ...
```

# Introduction to Python

---

## 1. IF-ELSE statement:

- The “`elif`” and “`else`” statements are optional
- Example:

```
x = 5
if x > 0 :
    print("Non-negative number")
elif x < 0:
    print("Negative number")
else:
    print("the number is 0")
```

# Introduction to Python

---

## 1. IF-ELSE statement:

- Why do you get an error with the code below?

Code #1

```
x = 5
if x > 0 :
    print("Non-negative number")
```

Code #2

```
x = 5
if x > 0 :
    print("Non-negative number")
    print("Non-negative number - print 2")
```

# Introduction to Python

---

## 2. FOR statement (loop):

- Repeats a group of commands for each possible value in a sequence/list of values

Syntax:

```
for value in sequence:
```

...

Example:

```
for x in range(10):  
    if x > 5:  
        print(x)
```

- Note: by default, the range function returns a sequence of numbers from 0 to the argument value minus 1

`range(10) = 0, 1, ..., 9`

# Introduction to Python

---

## 2. FOR statement (loop):

- One can specify different values inside `range()`
  - Syntax: `range(start, end, by)`
  - Example:

```
for x in range(-1, 10, 2):  
    print(x)
```

# Introduction to Python

---

## 2. FOR statement (loop):

- Python allows one to easily iterate over a string

- Example:

```
for x in "Arthur":  
    print(x)
```

- Python allows one to iterate over values in a **list**

- More on that in the future

```
for x in ["John", "Paul", "George", "Ringo"]:  
    print(x)
```

# Introduction to Python

---

## 3. WHILE statement (loop)

- Repeats a group of commands until the Boolean condition ceases to apply

Syntax:

```
while <logical_expression>:
```

```
    ...
```

Example:

```
x = 5
while x > 0:
    print(x)
    x = x - 1
```

---

# Homework #2

# Homework #2

---

- You have been recently hired to work in the analytics team at Goldman Sachs
  - Your team is responsible for building models that predict stock prices
  - Instead of using a single model, each member of your team is responsible for creating one individual model
    - Individual models are later aggregated to form an *ensemble model*
  - You are responsible for developing a model based on Brownian motion

# Homework #2

---

- The simplest version of Brownian motion has the form:  $price_t = price_{t-1} + \epsilon$  where  $\epsilon \sim N(0, \sigma)$ 
  - The standard deviation  $\sigma$  represents the uncertainty in the price movement. The subscript  $t$  represents an end-of-day stock price
- **Task:** write a script that simulates a single Brownian motion over 100 days
  - Start by importing the module `random` in the first line of your code
    - More on this in future classes
    - `import random`
  - In the second line, assume the initial stock price is equal to 50
    - `price = 50`
  - Hint: write a `for` loop that iterates 100 times. Assume  $\sigma = 0.01$ 
    - In each iteration, draw a value from a normal distribution using the function `random.gauss(mu = 0, sigma=0.01)` and update the value stored in `price`
  - Upload your code on Canvas (Homework 2)

# Summary

---

- We have learned the basics of Python
  - How to declare variables
  - How to use built-in functions
  - How to control the flow of Python scripts
  
- Next lecture
  - Advanced variable types and data structures

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 4 – Preliminaries (Part III)

### *Advanced Variable Types*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Lecture Objectives

---

- Review Homework #2
- Learn about multivalued variable types
  - List
  - Data Frames
- “Case study”
  - Learn about web logs
  - Learn about the CSV file format
    - Learn how to load/save CSV files to/from Python

# Lecture Instructions

---

1. Download the notebook “*Lecture 4.ipynb*” available on Canvas
2. Download the data file “*web\_log.csv*” available on Canvas
  - **Make sure both files are inside the same folder**
3. Open the notebook “*Lecture 4.ipynb*” with VS Code

# Introduction to Python

---

- In the previous lecture, we learned how to define single-valued variables
  - Types: int, float, bool, str
- More complex variable types can be defined by allowing multivalued variables

- List
- Data Frame

Text Type:	<code>str</code>
Numeric Types:	<code>int, float, complex</code>
Sequence Types:	<code>list, tuple, range</code>
Mapping Type:	<code>dict</code>
Set Types:	<code>set, frozenset</code>
Boolean Type:	<code>bool</code>
Binary Types:	<code>bytes, bytearray, memoryview</code>

Not covered in this course

# List

---

- Used to store multiple values in a single variable
- One of 4 built-in data types in Python used to store collections of data
  - The other three are *Tuple*, *Set*, and *Dictionary*
- Important points to remember
  - List items are ordered, changeable, and allow for duplicate values
  - List items are indexed
    - The first item has index [0]

# List

---

- Lists are created using square brackets
  - Example: `my_first_list = ["Paul", "John", "Ringo", "George"]`
- A list can contain different data types
  - Example: `my_second_list = ["Paul", 1, "John", True, "Ringo", 2.0, "George"]`
- One can obtain the length (*i.e.*, number of elements) of a list by using the `len()` function
  - Example: `len(my_second_list)`

# List

---

## ➤ Accessing elements

- List elements are indexed
  - One can access an element by referring to its index number
  - **The first item has index 0**
  - Example: `my_second_list[2]`
- Negative indexing
  - Negative indexing means start from the end
  - Example: accessing the last element: `my_second_list[-1]`

# List

---

## ➤ Accessing elements

- One can create a range of elements to be retrieved
  - Syntax: `start:end:step`
    - Important: the end-value is not included, and step is optional
    - Example: access all the elements from position 0 to 6 by 2  
`my_second_list[0:6:2]`
  - By leaving out the start (end) value, the range will start at the first (end at the last) item
    - Example: `my_second_list[0:]`  
`my_second_list[:5]`

# List

---

## ➤ Adding elements

- The `append()` function adds an element to the end of the list
  - Example: `my_second_list.append("ISA 414")`
- The `insert()` function adds an element to a specific locations of the list
  - Example: `my_second_list.insert(0, "first element")`
- One can concatenate two lists by summing them
  - Example: `my_third_list = my_first_list + my_second_list`

# List

---

- Removing elements
  - The `remove()` function removes elements based on values
    - Example: `my_third_list.remove("ISA 414")`
  - The `pop()` function removes elements based on indexes
    - Example: `my_third_list.pop(0)`

# List

---

## ➤ Lists and loops

- We will often loop through a list of items

index values

0	Paul
1	John
2	Ringo
3	George
4	first el...
5	Paul
6	1
7	John
8	true
9	Ringo
10	2
11	George
12	ISA 414

- Looping through item values:

```
for x in my_third_list:  
    print(x)
```

- Looping through index numbers:

```
for x in range(len(my_third_list)):  
    print(my_third_list[x])
```

- Note that `len()` gives the number of elements

- Then, `range(len( ))` creates a range from position 0 to the total number of elements in a list minus one

# List

---

## ➤ Lists and loops

- We will often create loops to add elements to a list
  - Example: `my_fourth_list = []  
for i in my_third_list:  
 if type(i) == str:  
 my_fourth_list.append(i)`
- Overall, loops are very slow in interpreted languages, such as Python and R
  - They work on each element at a time, instead of a whole list

# List

---

## ➤ Lists and loops

- **List comprehension** offers a shorter and more efficient way of creating a new list
  - Syntax: `newlist = [value for item in list if <condition>]`
    - The if-statement is optional; it filters values from `list`
    - Example: `my_fourth_list = [i for i in my_third_list if type(i) == str]`
  - Note that the syntax changes if there is an else statement
    - Syntax: `newlist = [value_1 if <condition> else value_2 for item in list]`
    - Example: `another_list = [i if type(i) == str else None for i in my_third_list]`
  - List comprehension will be incredibly useful when manipulating textual data

# Data Frames

---

- Data frames are two dimensional structures that follow a tabular format
  - There is no native data frame structure in Python
- We need to install the **Pandas** “package” that offers such a functionality
  - Packages are called **modules** or **libraries** in Python
  - More about modules in our next class
- Note that columns in a Pandas data frame are called **Series**
  - Special type of lists

# Data Frames

- Go to the terminal type: `pip install pandas`
  - Menu item *Terminal* -> “*New Terminal*”



- Let's import Pandas and create a data frame with two columns

```
import pandas
```

```
df = pandas.DataFrame({"Col_1":another_list, "Col_2":my_third_list})
```

# Data Frames

---

- Note how each row has a “name” (index)

- Unique identifier of the row

	Col_1	Col_2
0	Paul	Paul
1	John	John
2	Ringo	Ringo
3	George	George

- Accessing columns by name
    - Example: `df["Col_1"]`  
`df[["Col_1", "Col_2"]]`

# Data Frames

---

- Accessing rows and/or columns by location
  - Example: selecting all rows (:) for the second column  
`df.iloc[:,1]`
  - Example: selecting all the columns (:) for the first and third rows  
`df.iloc[ [0,2], :]`

# Data Frames

---

- Adding rows to a data frame
  - The `append()` function adds rows similar to how we created data frames
    - Example: `df = df.append({"Col_1":1, "Col_2":2}, ignore_index=True)`
  
- Removing rows from a data frame
  - The `drop()` function removes elements based on indexes
    - Example: `df = df.drop(10)`

.....

# Case Study: Analyzing Web Logs

# Case Study: Analyzing Web Logs

---

- We are now in a position to start talking about and practicing the management of big data
- Keep in mind that:
  - Big data management ≠ data storage
    - Narrow perspective
  - Big data management is about how to:
    - Effectively collect, process, store, and analyze potentially large and unstructured data sets

# Case Study: Analyzing Web Logs

---

- Throughout this course, we will learn about different data types and formats
  - Today we learn about (web) logs and the CSV format
- Background story
  - You work for a company that offers very expensive butler services
    - Target market: European billionaires
  - Due to the recent economic recession, your company is considering to expand to emerging markets

# Case Study: Analyzing Web Logs

---

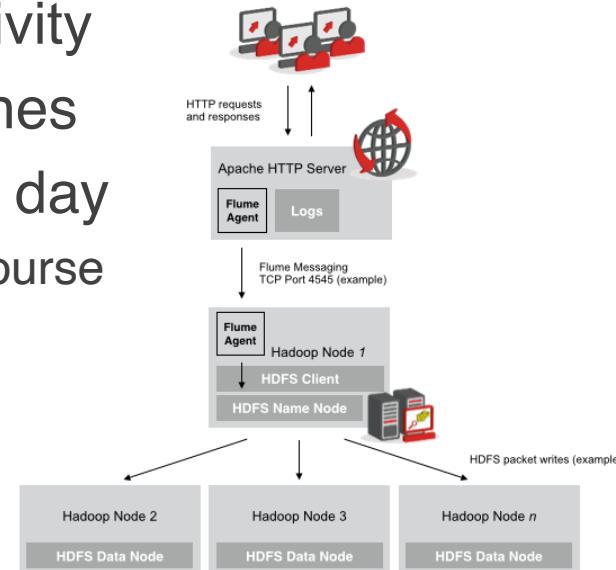
## ➤ Background story

- Question: which countries are expected to have high demand for the offered butler services?
  - The answer will guide marketing efforts
- Don Draper, the Chief Marketing Officer (CMO), asks you to find an answer



# Case Study: Analyzing Web Logs

- Talking to the IT staff, you learn the following about the company's data infrastructure
  - Whenever a user accesses the company's website, an Apache web server logs the user's activity
  - These logs are eventually sent in batches to a Hadoop cluster at the end of each day
    - We will learn more about that later in this course



# Case Study: Analyzing Web Logs

---

- Web logs
  - Whenever a user access a company's web page (e.g., [www.butlerforyou.com](http://www.butlerforyou.com)), that is how Apache web server stores the user's activity

```
64.246.220.203 - - [14/Jun/2014:10:30:20 -0400] "GET /home HTTP/1.1" 200 760 "-"  
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/35.0.1916.153 Safari/537.36"
```

```
64.246.220.203 - - [14/Jun/2014:10:30:30 -0400] "GET /about/butlers HTTP/1.1" 200 1671 "-"  
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/35.0.1916.153"
```

# Case Study: Analyzing Web Logs

---

## ➤ Web logs

- (Web) logs are often (semi-) structured data
  - Not ready for statistical analysis (must be pre-processed)
  - There are patterns (structures) in the data

```
1 [79.133.215.123] - - [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"  
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/35.0.1916.153 Safari/537.36"
```

IP_Address	Access_Date	HTML_Method	Requested_Page	HTTP_Version	Code_1	Code_2	User_Agent
79.133.215.123	14/Jun/2014:10:30:13	GET	/home	1.1	200	1671	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916. 153 Safari/537.36

# Case Study: Analyzing Web Logs

---

- Knowing that, you ask the IT staff to do the following for you
  1. Extract the IP addresses from the web logs
  2. From each IP address, obtain the user's latitude and longitude
  3. From the user's latitude and longitude, obtain the user's city and country
  - Your ingenious idea is to count the number of accesses per country, and use such a number as a **proxy** for the butler service demand
- The IT staff complies with your request and gives you a CSV file called “*web\_logs.csv*” (available on Canvas)
  - You will learn how to do all of the above steps in the upcoming lectures

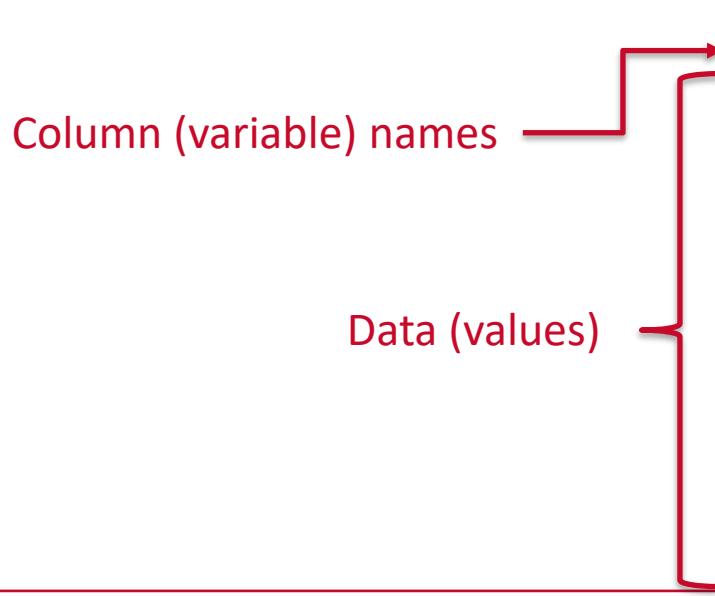
# Case Study: Analyzing Web Logs

---

- Comma-Separated Values (CSV) files
  - A CSV-file is a straightforward text file
    - Each line is an observation
    - The values or columns are separated by a semicolon (;) or a comma (,)
  - Standard format used in data analysis
  - Usually, column names are in the first line of the file
  - We will analyze the data in the CSV file “*web\_log.csv*”, which is available on Canvas

# Case Study: Analyzing Web Logs

- Comma-Separated Values (CSV) files
  - If you open the file “*web\_log.csv*” with VS Code



Column (variable) names	Data (values)
1 City,Client_IP,Country_Name,Latitude,Longitude	
2 Singapore,128.199.234.236,Singapore,1.293099999,103.8557968	
3 Singapore,128.199.234.236,Singapore,1.293099999,103.8557968	
4 Singapore,128.199.234.236,Singapore,1.293099999,103.8557968	
5 Singapore,128.199.234.236,Singapore,1.293099999,103.8557968	
6 Singapore,128.199.234.236,Singapore,1.293099999,103.8557968	
7 Mountain View,66.249.76.236,United States,37.38600159,-122.0838013	
8 Guiyang,222.85.131.87,China,26.58329964,106.7166977	
9 Guiyang,222.85.131.87,China,26.58329964,106.7166977	
10 Shanghai,101.226.168.225,China,31.04560089,121.3996964	
11 Mountain View,66.249.76.225,United States,37.38600159,-122.0838013	
12 Shanghai,101.226.166.230,China,31.04560089,121.3996964	
13 Shanghai,101.226.166.231,China,31.04560089,121.3996964	
14 Zhengzhou,182.118.25.227,China,34.68360138,113.5325012	
15 Zhengzhou,182.118.25.227,China,34.68360138,113.5325012	
16 Zhengzhou,182.118.25.228,China,34.68360138,113.5325012	
17 Zhengzhou,182.118.25.230,China,34.68360138,113.5325012	
18 Zhengzhou,182.118.25.229,China,34.68360138,113.5325012	
19 Hyderabad,122.175.18.128,India,17.37529945,78.47440338	
20 Shanghai,101.226.166.230,China,31.04560089,121.3996964	
21 Shanghai,101.226.166.234,China,31.04560089,121.3996964	
22 Shanghai,101.226.166.233,China,31.04560089,121.3996964	
23 Shanghai,101.226.166.231,China,31.04560089,121.3996964	

# Case Study: Analyzing Web Logs

---

## ➤ Comma-Separated Values (CSV) files

- If you open the file “*web\_log.csv*” with spreadsheet software, like Excel, that is how it will look like

1	City	Client_IP	Country_Name	Latitude	Longitude
2	Singapore	128.199.234.236	Singapore	1.293099999	103.8557968
3	Singapore	128.199.234.236	Singapore	1.293099999	103.8557968
4	Singapore	128.199.234.236	Singapore	1.293099999	103.8557968
5	Singapore	128.199.234.236	Singapore	1.293099999	103.8557968
6	Singapore	128.199.234.236	Singapore	1.293099999	103.8557968
7	Mountain View	66.249.76.236	United States	37.38600159	-122.0838013
8	Guiyang	222.85.131.87	China	26.58329964	106.7166977
9	Guiyang	222.85.131.87	China	26.58329964	106.7166977
10	Shanghai	101.226.168.225	China	31.04560089	121.3996964

# Case Study: Analyzing Web Logs

---

## ➤ Let's analyze the data

- Importing CSV files into Python

- In order to perform our analysis, we need to import the data from the CSV file into Python
  - We can do so by using the function `read_csv` in the Pandas module

```
web_data = pandas.read_csv("web_log.csv")
```

- The first argument of the above function is the location of the CSV file
    - In the above example, the file “`web_log.csv`” must be located in same the same directory as the notebook

# Case Study: Analyzing Web Logs

---

- We can now count the number of web pages requests per country
  - In statistical terms, we want to calculate the frequency table for the variable “*Country\_Name*”
  - To do so, we will use the function *value\_counts*
    - This function can be applied to data frames or series

```
pandas.value_counts(web_data["Country_Name"])
```

India	357
Thailand	232
United States	190

# Case Study: Analyzing Web Logs

---

- One can also save a data frame or a series in Python to a CSV file using the function `to_csv`
  - Example: suppose one wants to save the frequency table to a csv file

```
results = pandas.value_counts(web_data["Country_Name"])
results.to_csv("ftable.csv")
```

# Homework #3

---

Suppose your company now wants to determine the demand for butlers at the city level, as opposed to country level

- Write a script that solves the following tasks and report you answers on Canvas
- Task #1: save the frequency table per city to a variable called `city_freq`
- Task #2: visualize the created series
- Task #3: which city should your company focus its marketing efforts on?

# Summary

---

- We have learned about more advanced data types/structures in Python
  - List and Data Frames
- We learned about web logs and the CSV format
- Next lecture
  - Introduction to Python: user-defined functions and modules

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 5 – Introduction to Python

*User-Defined Functions and Modules*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Lecture Objectives

---

- Review Homework #3
- Learn how to create functions in Python
- Learn how to install and use modules in Python

# Lecture Instructions

---

1. Download the notebook “*Lecture 5.ipynb*” available on Canvas
2. Download the data file *web\_log.csv* available on Canvas
  - **Make sure both files are inside the same folder**
3. Open the file “*Lecture 5.ipynb*” with VS Code

# Introduction to Python

---

- Our code will start to become more and more complex
  - It is essential to document your code/analysis
    - Reproducibility
    - Accountability
  - We learned before about markdown
  - We learn today about code comments

# Introduction to Python

---

## ➤ Comments

- One can add explanatory comments in Python by adding the character '#' before the comment
- Example:

```
#The command below calculates the sum of 1 + 1
```

```
1+1
```

- Comments are just for the sake of improving code readability
  - They are ignored when one runs a code

# Introduction to Python

---

- Comments can be placed at the end of a line
  - Python interpreter will ignore the rest of the line
  - Example: `1 + 1 # another comment`
- One must use three quotation marks before and three after a comment when using multiple lines
  - Example: `"""  
A multi-line  
comment  
in Python  
"""`



# User-Defined Functions

# Introduction to Python

---

- The basic installation of python comes with many built-in functions
  - E.g., `range`, `len`, `int`, `str`, etc.
  - One can further extend Python's core by defining new functions
  - Syntax: the `return` statement is optional

```
def function_name(argument_1, argument_2, ...):  
    ... commands ...  
    return something
```

# Introduction to Python

---

## ➤ Functions

- Example: let's create a very basic function called *RatioMaxMin*
  - Receives two numeric arguments
  - Returns the ratio between the maximum and the minimum argument

```
def RatioMaxMin(value_1, value_2):  
    ratio = max(value_1, value_2) / min(value_1, value_2)  
    return ratio
```

# Introduction to Python

---

## ➤ Functions

- Let's apply the previously defined function

RatioMaxMin(10,2)

RatioMaxMin(2, 10)

RatioMaxMin(1,2)

RatioMaxMin(10,0)

RatioMaxMin(10,'2')



Why do we get an error here?

# Introduction to Python

---

## ➤ Functions

- One can predefine the values of the arguments of a function
  - Example:

```
def RatioMaxMin2(value_1, value_2 = 10):  
    ratio = max(value_1, value_2) / min(value_1, value_2)  
    return ratio
```

- If the second argument is not provided when calling the function `RatioMaxMin2`, then the function assumes that such a value is equal to 10

# Introduction to Python

---

## ➤ Functions

- Let's apply the previously defined function

RatioMaxMin2(10)

RatioMaxMin2(5)

RatioMaxMin2(5, 2)

# Introduction to Python

---

## ➤ Functions

- It is good practice to check the type of the arguments inside the functions to prevent unexpected behavior
  - Example:

```
def RatioMaxMin3(value_1, value_2 = 10):  
    if type(value_1) == int and type(value_2) == int:  
        ratio = max(value_1, value_2) / min(value_1, value_2)  
        return ratio  
    else:  
        return "One of the arguments is not of type integer"
```

# Introduction to Python

---

## ➤ Functions

- A more elegant way of handling errors (exceptions) is by using try/except/finally statements
  - Handling exceptions is beyond the scope of this course
  - Example:

```
def RatioMaxMin3(value_1, value_2 = 10):  
    try:  
        ratio = max(value_1, value_2) / min(value_1, value_2)  
        return ratio  
    except:  
        return "One of the arguments is not a number"
```

# Introduction to Python

---

- There is much more to functions than what we learned today
  - Example:
    - Arbitrary number of arguments/values
      - Add a \* (tuple) or \*\* (dictionary) in front of a parameter
    - Lambda notation
    - ...
- But what we learned is enough for our purposes



# Modules

# Introduction to Python

---

- Python modules (packages/libraries)
  - Code libraries
  - Contain useful and reusable Python functions
  - Hypothetical example
    - Suppose you develop a new statistical technique
    - You code that technique in Python by means of creating functions
    - You then create a module with all the relevant functions and documentation, and share the module with the community

# Introduction to Python

---

- Let's create and import our own module

- Create a blank file called *mymodule.py*
  - Inside that file add the following code:

```
my_var = ["ISA", 414]
```

```
def increment_by_two (value):  
    return value + 2
```

- Save the file

# Introduction to Python

---

- Let's import that module now inside our notebook
  - Make sure the file *mymodule.py* is in the same folder as “*Lecture 5.ipynb*”

```
import mymodule
mymodule.increment_by_two(10)
```
  - It is common practice to use aliases when importing modules  

```
import mymodule as mm
mm.increment_by_two(10)
```

# Introduction to Python

---

- One can always import only parts from a module
  - Syntax: `from <module> import <part>`
  - Example: 

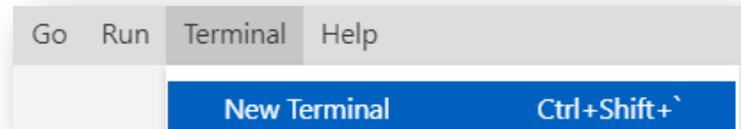
```
from mymodule import my_var  
print(my_var[1])
```

    - Note that the module's name is not required when using only parts

# Introduction to Python

---

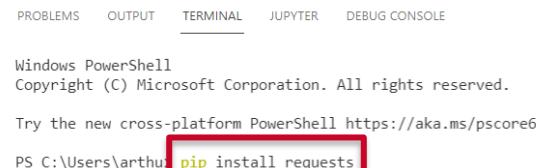
- How can one download packages/modules from the web?
  - Similar to how `install.packages()` works in R
  - Answer: **PIP** (Preferred Installer Program)
    - Downloads modules from repositories such as pypi.org
  - One must execute PIP commands from a **terminal**
    - Interface in which you can type and execute text-based commands
  - Let's try PIP
    - Go to *Terminal* -> “*New Terminal*”



# Introduction to Python

---

- PIP has several purposes
  - We shall only focus on installing packages
  - Syntax: `pip install <package name>`
  
- Let's install the module `requests`
  - Go to the terminal and type:  
`pip install requests`
  - You can now import the `requests` module



PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\arthur> `pip install requests`

.....

# Case

# Introduction to Python

---



## ➤ Let's revisit last lecture's example

- You investigated the question: *which countries are expected to have high demand for the offered butler services?*
- You answered the above questions by looking at the number of web access to the company's web page per country
- You have now to present your results to your boss, the great Don Draper

# Introduction to Python

---

## ➤ Case

- When preparing your presentation, you have the great idea of showing Don Draper a map with the location of visitors who accessed the company's web page
- You search on Google “*how to plot maps with Python*”, and come across a reliable package called *ipyleaflet*
  - <https://ipyleaflet.readthedocs.io/en/latest/>
  - No need to reinvent the wheel

# Introduction to Python

---

## ➤ Case

- Let's install the [ipyleaflet](#) module
  - Go to the terminal and type:  
`pip install ipyleaflet`
- We will also need Pandas to load .CSV files
  - If you haven't installed Pandas before, then run the following command:  
`pip install pandas`

# Introduction to Python

---

- Let's start by loading the modules
  - From the `ipyleaflet` module, we only need the `Map()` and `Marker()` functions
    - How do I know? I read the documentation at  
<https://ipyleaflet.readthedocs.io/en/latest/>

```
from ipyleaflet import Map, Marker  
import pandas
```

# Introduction to Python

---

- Let's now load the data and perform some preprocessing
  - Time to learn some extra functions

```
df = pandas.read_csv("web_log.csv")
```

```
df = df.drop_duplicates(["Latitude","Longitude"])
```

```
df["City"] = df["City"].fillna("NA")
```

# Introduction to Python

---

- Let's plot our map
  - For each row in our cleaned data frame, we create one specific marker

```
map = Map(zoom = 1)
```

```
for i in range(len(df)):  
    marker = Marker(location = (df.iloc[i,3], df.iloc[i,4]),  
                    draggable = False,  
                    title = df.iloc[i,0])  
    map.add_layer(marker)
```

# Introduction to Python

---

## ➤ Python modules

1. DO NOT REINVENT THE WHEEL
  - Always look for well-established modules that do what you want
  - Easy to find packages: just Google it
    - *E.g.*, “plotting maps with Python”
2. BE CAREFUL WITH THE MODULES YOU USE
  - There are many great modules out there, but many badly/wrongly implemented ones as well
    - Be extra careful with statistical modeling packages
  - Always look for info about the authors, (white) papers, etc.
  - For security reasons, many organizations do not allow employees to install external Python modules

# Introduction to Python

---

## ➤ Some famous data-science Python modules

- **Scikit-Learn**: statistical modeling
- **Numpy**: used internally by several other modules when dealing with multi-dimensional arrays and matrices
- **Keras**: deep-learning models
- **PyTorch**: primarily deep-learning models
- **SciPy**: scientific computing (e.g., optimization solvers)
- **Pandas**: provides high-level data structures

# Homework #4

---

- Recall the problem from the previous lecture
  - Your solution:
    1. Extract IP addresses from log files (**Assignment 1**)
    2. Derive locations from IP addresses
      - The IT staff gave you a CSV file with the required data
      - See the file *web\_log.csv*
    3. Count the number of times each location appears in your data (**Lecture 3**)
      - Proxy for demand
  - You will now put yourself in the shoes of the IT staff
    - Complete step 2

# Homework #4

---

- You will use the `ip2address` function in the *homework4.py* module to derive location from IP addresses

- Don't try to understand the function now
- Simply import the *homework4.py* module

```
def ip2address(ips):  
    import requests  
    import pandas  
  
    df = pandas.DataFrame()  
  
    for ip in ips:  
        url = "https://api.ipgeolocation.io/ipgeo?  
            apiKey=addfff7fcd22470aa78fd9a66cbdf500&  
            ip=" + ip  
        response = requests.get(url).json()  
        response_df = pandas.DataFrame(  
            from_dict(response, orient='index').  
            transpose())  
        if df.empty:  
            df = response_df  
        else:  
            df = df.append(response_df)  
    df.index = pandas.RangeIndex(len(df.index))  
    return df
```

# Homework #4

---

## ➤ Let's now prepare our data

- Goal: from a list of IP addresses, derive the data that we used in Lecture 4

- *E.g.,*

ip	country_name	city	latitude	longitude
165.70.141.165	United States	Dallas	32.86200	-96.80980
93.215.167.191	Germany	Darmstadt	49.86470	8.62546

- For this homework, consider the following list of IP address

```
ip_addresses = ["165.70.141.165", "93.215.167.191", "197.177.61.197", "206.236.177.182",
"5.77.63.24", "158.227.31.151", "82.233.65.66", "63.123.244.194"]
```

# Homework #4

---

➤ Now, it is with you:

- Step 1: apply the function `ip2address` to `ip_addresses` and store the results in a variable called `web_data`
  - **Please, do not run this step multiple times**
  - Collectively, we can only query 1000 IP addresses per day
- Step 2: remove the unnecessary columns from `web_data`
  - That is, keep only the variables `ip`, `country_name`, `city`, `latitude`, `longitude`

# Homework #4

---

- Step 3: save `web_data` to a CSV file
  - Do not include indexes (*i.e.*, row numbers) in the CSV file
    - Google will tell you the answer
  - This is how your results should look like:

ip	country_name	city	latitude	longitude
165.70.141.165	United States	Dallas	32.86197	-96.80983
93.215.167.191	Germany	Darmstadt	49.86571	8.62604
197.177.61.197	Kenya	Nairobi	-1.28222	36.87931
206.236.177.182	United States	Washington	38.90701	-77.0527
5.77.63.24	United Kingdom	Leeds	53.74717	-1.60168
158.227.31.151	Spain		42.84227	-2.68359
82.233.65.66	France	Bordeaux	44.83779	-0.57918
63.123.244.194	United States	Princeton	40.36076	-74.66446

- Upload your solution (code) on Canvas

# Summary

---

- We have learned how to:
  - Define functions
  - Install and load modules
  
- Next lecture
  - String Manipulation

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 6 – Introduction to Python

*String Manipulation and Regular Expressions*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Announcements

---

- Assignment 1 is now available on Canvas
  - Deadline: Wednesday, September 15<sup>th</sup>, before 11:59 pm

# Lecture Objectives

---

- Quick review of Homework 4
- Learn how to manipulate strings in Python
- Learn about regular expressions

# Lecture Instructions

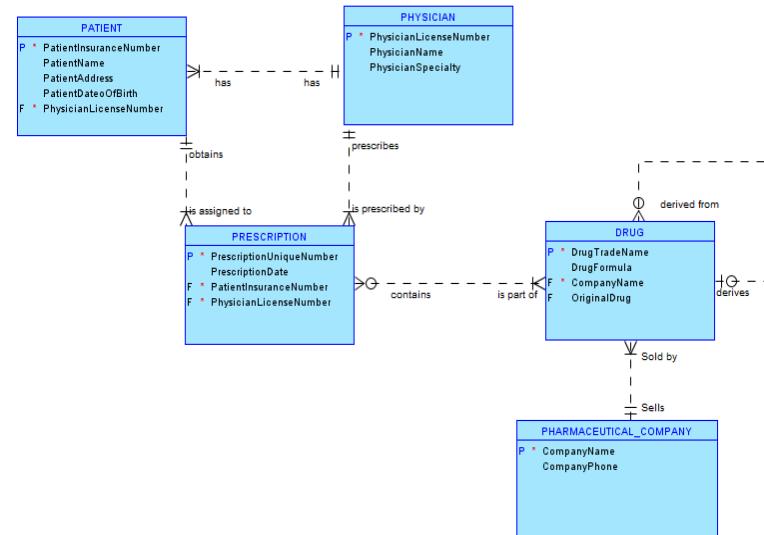
---

- Download from Canvas:
  - The notebook “*Lecture 6.ipynb*”
  - The log file “*access.log.txt*”
- Make sure all the files are inside the same folder
- Open the file “*Lecture 6.ipynb*” with VS Code

# Introduction to Python

- Traditional view of data (table, data frames, ...)
  - Well-defined structure

	Return	SAT	MBA	Age	Tenure	var	var
1	-21.10	1211.0	.0	47.0	4.0		
2	-6.38	1163.0	.0	43.0	5.0		
3	-3.32	1217.0	1.0	39.0	3.0		
4	-2.27	1185.0	1.0	35.0	2.0		
5	-9.46	1030.0	.0	36.0	5.0		
6	-1.28	1141.0	1.0	38.0	4.0		
7	4.66	1019.0	1.0	38.0	2.0		
8	7.06	1012.0	.0	38.0	4.0		
9	-12.39	925.0	1.0	36.0	4.0		
10	-10.27	1195.0	.0	46.0	5.0		
11	-3.31	845.0	.0	33.0	3.0		
12	-7.74	902.0	1.0	46.0	5.0		
13	7.49	1076.0	1.0	43.0	6.0		
14	-13.03	1320.0	.0	50.0	6.0		
15	8.92	1154.0	1.0	36.0	2.0		
16	2.04	1213.0	1.0	40.0	6.0		
17	-.50	1220.0	.0	33.0	4.0		
18	-.30	943.0	.0	38.0	4.0		
19	11.16	1064.0	1.0	34.0	1.0		
20	-7.61	789.0	1.0	46.0	2.0		
21	-1.21	1094.0	1.0	49.0	4.0		
22	.60	1212.0	1.0	40.0	4.0		



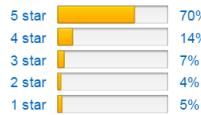
# Introduction to Python

## ➤ Unstructured data: *text*

### Customer Reviews

★★★★★ 8.597

4.4 out of 5 stars ▾



[See all 8,597 customer reviews ▾](#)

### Top Customer Reviews

★★★★☆ Flawed But Ultimately Thought-Provoking and Worthwhile

By Kenny O. on March 1, 2003

Format: Paperback

Yes, much of what negative reviewers of this book have to say is true: the writing is blunt and simple, the characters lack depth and complexity, it is quite male-focused in its subject matter and language, it has a bunch of quasi-religious mumbo-jumbo, and so on. This book should not be put on the list of great literature for the ages. There are doubtless many novels that cover subject matter from this book far more artfully. As I read the book, I was aware of its hokeyness and lack of redeeming literary qualities. I am, in fact, usually the first person to criticize books that read like this.

And yet, I have to say - and I feel a bit sheepish about this - that I found it meaningful, even profound at times. How can I say this, given my criticisms? First of all, unlike many reviewers, I did not approach this book with great expectations. No one told me that this was Shakespeare or Tolstoy; I had never even heard of it until it was recommended to me recently. And by the end of page 2, I had adjusted my expectations further. This clearly was not going to be winning the Booker prize.

But I found the book moving in its simple way. The characters deliver their statements without subtlety, but subtlety is more a literary virtue than a philosophical one. In fact, I essentially came to view this work as a life philosophy expressed as a fable, so I didn't particularly mind that its messages were not buried far beneath the surface.

Are those messages novel? No, but what of it? Novelists have been recycling themes for centuries, because many themes are of enduring interest and relevance. The point is, the messages are worthwhile and deserving of consideration. [Read more ▾](#)



Don't fly [@BritishAirways](#). Their customer service is horrendous.

Promoted by

9/2/13, 7:57 PM

# Introduction to Python

---

- (Semi) structured data: *web logs*
  - Try opening the file *access.log.txt* with VS Code

```
1 79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"  
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/35.0.1916.153 Safari/537.36"  
2 162.235.161.200 - - [14/Jun/2014:10:30:13 -0400] "GET  
/department/apparel/category/featured%20shops/product/adidas%20Kids'%20RG%20III%20M  
id%20Football%20Cleat HTTP/1.1" 200 1175 "-" "Mozilla/5.0 (Macintosh; Intel Mac  
OS X 10_9_3) AppleWebKit/537.76.4 (KHTML, like Gecko) Version/7.0.4  
Safari/537.76.4"  
3 39.244.91.133 - - [14/Jun/2014:10:30:14 -0400] "GET /department/fitness HTTP/1.1"  
200 1435 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36  
(KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"  
4 150.47.54.136 - - [14/Jun/2014:10:30:14 -0400] "GET  
/department/fan%20shop/category/water%20sports/product/Pelican%20Sunstream%20100%20  
Kayak/add_to_cart HTTP/1.1" 200 1932 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X  
10_9_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36"
```

# Introduction to Python

---

- (Semi) structured data: *web logs*
  - Not in a standard tabular format, but the data nonetheless contain precious information
    - One can explore certain patterns in order to impose a tabular structure

```
1 79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"  
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/35.0.1916.153 Safari/537.36"
```

# Introduction to Python

---

- (Semi) structured data: *web logs*
  - Let's impose a tabular structure on the data
    - Assignment 1

```
1 79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.153 Safari/537.36"
```

IPs	Date	Request	Page	HTTP_Version	First_Code	Second_Code	User_Agent
79.133.215.123	14/Jun/2014:10:30:13	GET	/home	1.1	200	1671	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.3 6 (KHTML, like Gecko) Chrome/35.0.1916. 153 Safari/537.36

# Introduction to Python

---

- One needs to manipulate characters (strings) in order to impose some structure on textual data
  - We will next learn a series of commands to manipulate and process textual data in Python
  - Highly relevant when collecting and pre-processing textual data
    - Web scrapping, text mining

# Introduction to Python

---

- Let's start by loading the data

```
file = open('access.log.txt', 'r')  
web_logs = file.readlines()
```

- The variable `web_logs` is a list containing 360,000 elements (log lines)

Name	Type	Size	Value
file	TextIOWrapper		<_io.TextIOWrapper name='access.log.txt' mode='r' encoding='utf-8' errors='strict' closefd=True>
web_logs	list	360000	['79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET / HTTP/1.1" 200 1234', ...]

# Introduction to Python

---

- Let's look at the first element in `web_logs`  
`web_logs[0]`
- Let's calculate the number of characters in the  
first log entry  
`len(web_logs[0])`

# Introduction to Python

---

- Now let's start imposing a tabular structure on our data
  - First step: obtain the variable IP
    - Question 1 a), Assignment 1
    - Which pattern can we explore?
      - How can one retrieve IP addresses without hardcoding positions?

```
1 79.133.215.123 -- [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"  
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/35.0.1916.153 Safari/537.36"  
2 162.235.161.200 -- [14/Jun/2014:10:30:13 -0400] "GET  
/department/apparel/category/featured%20shops/product/adidas%20Kids'%20RG%20III%20M  
id%20Football%20Cleat HTTP/1.1" 200 1175 "-" "Mozilla/5.0 (Macintosh; Intel Mac  
OS X 10_9_3) AppleWebKit/537.76.4 (KHTML, like Gecko) Version/7.0.4  
Safari/537.76.4"
```

# Introduction to Python

---

- First approach to obtain IP addresses: split the input log data into several parts based on another string
  - In our case, split each log entry based on the presence of blank spaces
    - Function: `split()`
    - Example: splitting the first log entry  
`words = web_logs[0].split(" ")`
    - Let's look at the content of `words`



```
79.133.215.123
-
-
[14/Jun/2014:10:30:13
-0400]
"GET
/home
HTTP/1.1"
200
1671
"_
Mozilla/5.0
(Windows
NT
...
...
```

# Introduction to Python

---

- Let's now retrieve all the IP addresses
  - Using a for-loop

```
IP = []
for log in web_logs:
    words = log.split()
    IP.append(words[0])
```

# Introduction to Python

---

- Let's now retrieve all the IP addresses
  - Using list comprehension

```
IP = [web_logs[i].split(" ")[0] for i in range(len(web_logs))]
```

# Introduction to Python

---

## ➤ Second approach to obtain IP addresses

- Procedure

1. Locate the index (position) of the first blank space
2. Obtain all characters from a log entry starting at position 0 and ending at the position right before the position of the first blank space

IP address:

position 0 to 13

first space: position 14

```
1 79.133.215.123 -- [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-"
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/35.0.1916.153 Safari/537.36"
```

# Introduction to Python

---

- Locating the index (position) of the first blank space
  - One can do this using **regular expressions**
    - Very powerful language with its own syntax and semantics
    - Concise and powerful method for *searching for a specific string pattern*
    - Example: how can you automatically search for names of video games in a given text?
      - Common pattern: The <something> of <something>
        - The <World> of <Warcraft>
        - The <Lord> of <the Rings>
        - The <Legend> of <Zelda>
        - The <House> of <the Dead>
        - ....

# Introduction to Python

---

- Locating the index (position) of the first blank space
  - The pattern we are looking for is very simple
    - The first occurrence of blank space
  - The function `search` from the `re` module returns the first position of a searched pattern
    - Let's search for the first occurrence of the pattern containing only the blank space in the first log entry

```
import re  
hits = re.search(" ", web_logs[0])  
pos_first_space = hits.start(0)
```

# Introduction to Python

---

- Locating the index (position) of the first blank space
  - If we wanted all the positions containing single blank spaces, we would then use the **finditer** function

```
hits= re.finditer(" ", web_logs[0])
```

```
all_space_positions = [m.start(0) for m in hits]
```

# Introduction to Python

---

- Now, we can obtain all characters from position 0 to *pos\_first\_space* by substring a string
  - Notation: `string[start:end:step]`
    - Remember that the element in the end position is not included

```
string = web_logs[0]
first_ip = string[0:pos_first_space:1]
```

# Introduction to Python

---

- Even though we have focused on the first log line, it is possible to extend what we did to all logs

```
Get the result of the search  
for the first space           ← For each log (line) in web_logs  
  
all_pos_first_space = [re.search(" ", log).start(0) for log in web_logs]  
  
Access one log (str)  Substring the log      For each value from 1 to 360,000  
  
IPs = [web_logs[i][0:all_pos_first_space[i]:1] for i in range(len(web_logs))]
```

# Introduction to Python

---

- More on regular expressions
  - These may (or may not) be useful in Assignment 1
  - Finding the first position of the string HTTP

```
all_pos_HTTP = [re.search("HTTP", log).start(0) for log in web_logs]
```
  - Finding the first position of a double quotation mark (")

```
all_pos_first_quote = [re.search('"', log).start(0) for log in web_logs]
```
  - Finding the first position of a left square bracket ( [ )
    - The need for escaping will be explained in Lecture 8

```
all_pos_first_bracket = [re.search("[", log).start(0) for log in web_logs]
```

# Introduction to Python

---

- What about finding the position of the second blank space
  - Can you do the same using list comprehension

```
all_pos_second_space = []
```

```
for logs in web_logs:
```

```
    hits = re.finditer(" ", logs)
```

```
    positions = [m.start(0) for m in hits]
```

```
    all_pos_second_space.append(positions[1])
```

- We will learn about more advanced regular expressions in upcoming lectures

# Introduction to Python

---

- Potentially useful “trick” for Assignment 1
  - You can always remove an obtained variable from the original data set after collecting that variable
  - Example: suppose you obtained the variable **IP** by following the previous steps. You can then remove **IP** from the original data set:

```
modified_logs = [web_logs[i][all_pos_first_space[i]+1:] for i in range(len(web_logs))]
```

  
i-th log   Substring i-th log from after first space to the end

The above might make your life easier

- *E.g.*, you will never have to search for the second blank space

.....

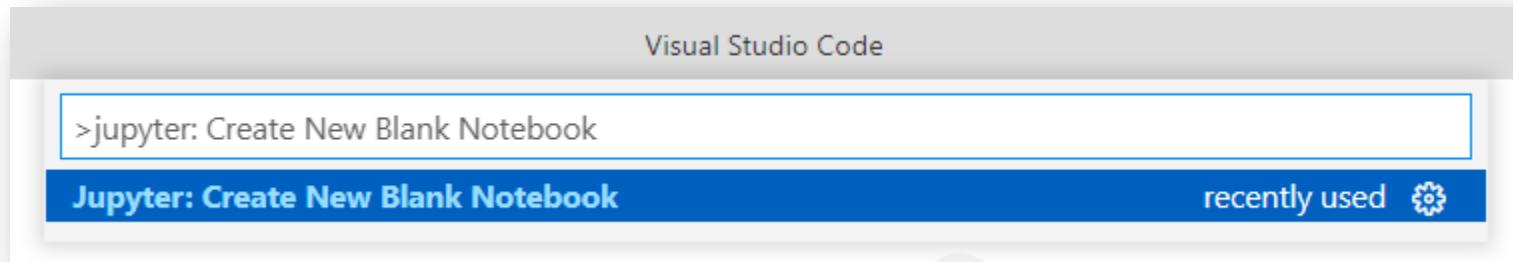
# Assignment 1: Head Start

# Assignment 1

---

➤ Let's put things together to get a head start in Assignment 1

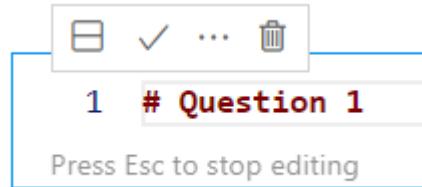
- Remember: you must submit a `.ipynb` file
- Start by creating a new Jupyter notebook
  - Press F1 and type “*Jupyter: Create New Blank Notebook*”



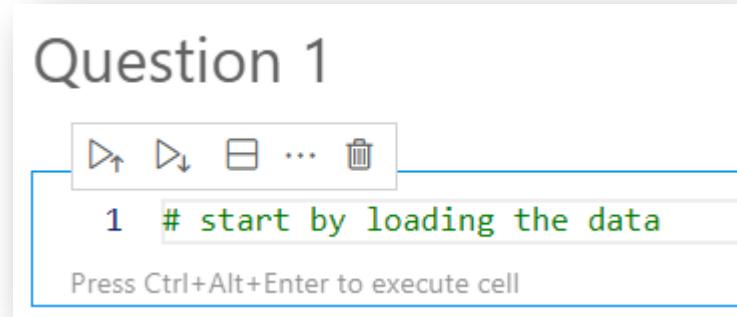
# Assignment 1

---

- Add question-related comments using Markdown



- Add Python code with explanatory notes (optional)

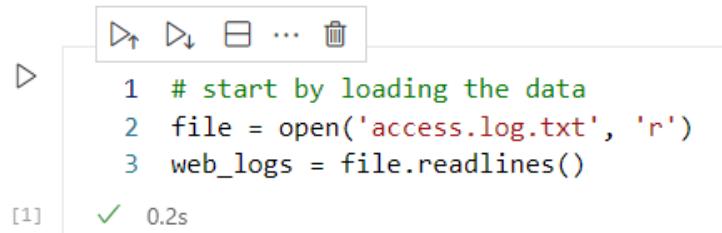


# Assignment 1

---

- Write the Python code to load the required data set

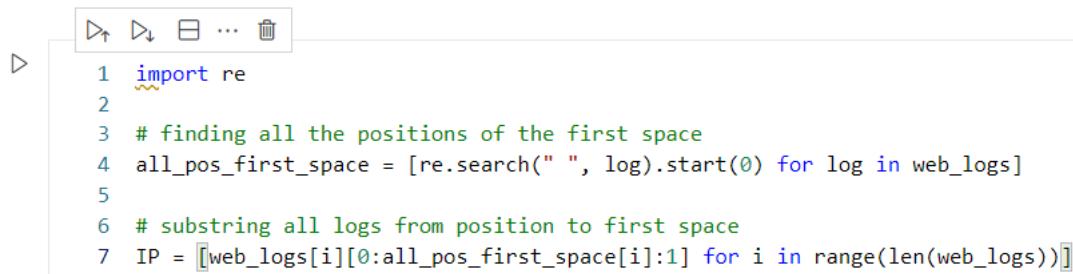
## Question 1



```
1 # start by loading the data
2 file = open('access.log.txt', 'r')
3 web_logs = file.readlines()
[1] ✓ 0.2s
```

- Let's add the solution to Q1 - a)

Q1-a)



```
1 import re
2
3 # finding all the positions of the first space
4 all_pos_first_space = [re.search(" ", log).start(0) for log in web_logs]
5
6 # substring all logs from position to first space
7 IP = [web_logs[i][0:all_pos_first_space[i]:1] for i in range(len(web_logs))]
```

# Assignment 1

---

- Run the previous code chunk for the sake of testing

PROBLEMS	OUTPUT	TERMINAL	JUPYTER: VARIABLES	DEBUG CONSOLE
Name	▲ Type		Size	Value
all_pos_first_spac	list		360000	[14, 15, 13, 13, 13, 12, 13, 13, 14, 13, 14, 11, 14]
file	TextIOWrapper			<_io.TextIOWrapper name='access.log.txt' mode='r' encoding='utf-8' errors='strict' closefd=True>
IP	list		360000	['79.133.215.123', '162.235.161.200', '39.244.91.111']
web_logs	list		360000	['79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET / HTTP/1.1" 200 1234']

# Assignment 1

---

- Save your source code
  - *File -> Save*
  - *E.g., “Assignment\_1.ipynb”*
  - This is what you will submit on Canvas after you are done with all the questions

# Assignment 1

---

- Remember that, unless otherwise stated, you can solve any question however you want to
  - Use any modules, techniques, functions, ...
  - Obviously, whatever you do must be accurate

# Summary

---

- We learned how to manipulate textual data
- This was the last lecture on the basics of Python
  - We will continue to learn about Python in the next lectures, but in a much more applied context
    - Mostly, using modules
- Next lecture: collecting data via web scrapping

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 7 – Data Collection

*Web Scraping (Part I)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

NETWORK WITH COMPANIES ON THE  
ISA ADVISORY BOARD BEFORE  
CAREER FAIR!

# Announcements

- ISA Recruiting Event
  - September 17
  - 1 – 3 pm
  - FSB West Commons

## ISA RECRUITING EVENT

September 17, 1-3 pm  
FSB West Commons

Deloitte

EY

FIS/Worldpay

GE Aviation

KPMG

Kroger

Nationwide

PwC

TransImpact

VNDLY

West Monroe



SCAN THE QR CODE TO LEARN  
MORE ABOUT THE COMPANIES  
AND READ JOB DESCRIPTIONS!

OR VISIT:  
[HTTP://WWW.FSB.MIAMIOH.EDU/  
ISAEVENTS](http://www.fsb.miamioh.edu/isaevents)

# Reminders

---

- We have in-person office hours and classes
  - The virtual option is only in case you have a serious reason
    - *E.g.*, health, religious, serious accidents, etc.
  - You should let me know of that at least 24h before class
    - Be prepared to provide documentation
- Please, avoid “Hail Marys”
  - Last minute requests for help, extending deadlines, etc.
  - It is very unlikely I will reply to those emails

# Lecture Objectives

---

- Review of Homework #4
- Learn about the CRISP-DM methodology
- Learn how to scrap data from web pages

# Lecture Instructions

---

1. Download the script “*Lecture 7.ipynb*” available on Canvas
2. Open the file “*Lecture 7.ipynb*” with VS Code
3. [Optional] Download the file *webpage.html* available on Canvas

# CRISP-DM

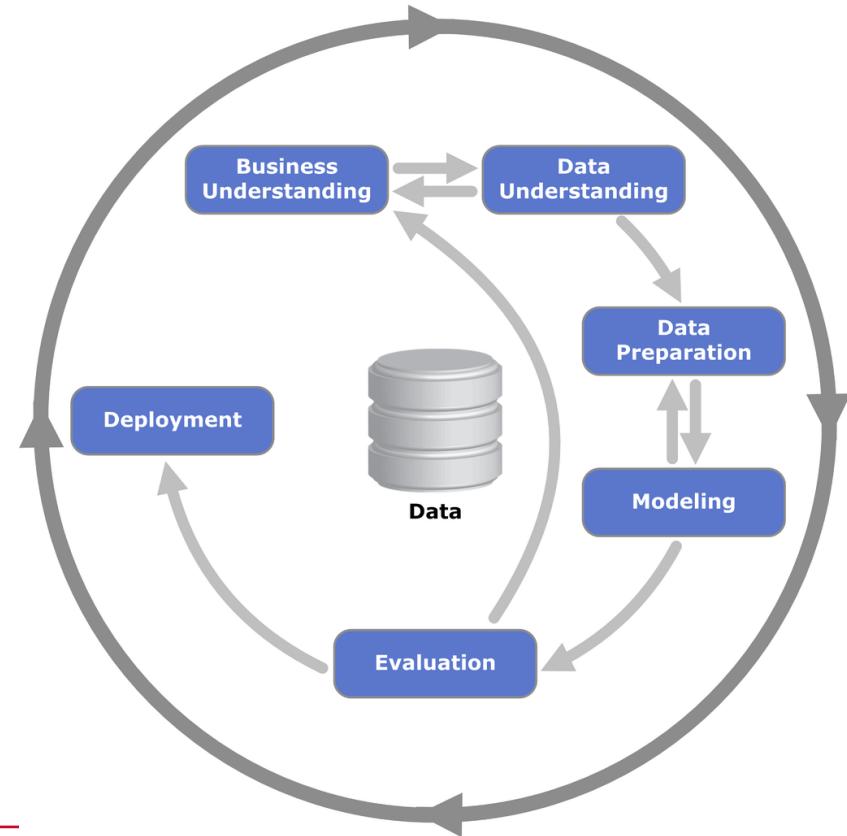
---

- We are reasonably proficient in Python now
  - It is time to start working on more complex data analytics tasks
- Starting and successfully completing an analytics project is no easy endeavor
  - Requires serious project management
  - Standard process model: CRISP-DM methodology
    - Cross Industry Standard Process for Data Mining
    - Series of steps to follow in order to solve an analytics-related business problem

# CRISP-DM

---

- Process model: 6 phases
  - Goal: extract information from data
  - Your final course project will follow the first five phases of this model

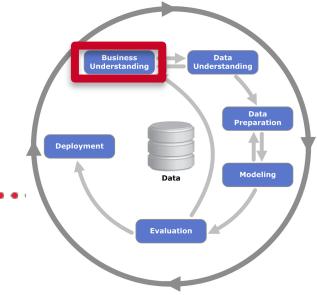


# CRISP-DM

---

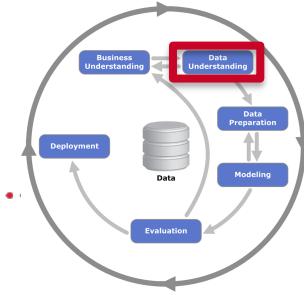
## ➤ Business understanding

- What is exactly the business problem to be solved?
  - Not at all trivial
  - Requires a number of meetings with key stakeholders/sponsors
- Can we formulate a data science solution to solve the business problem?
  - If we can, then is the problem a supervised or unsupervised problem?
    - Is there a clear target variable? (more on this in future lectures)
- What decisions should the solution support?
- Who will sponsor the project?



# CRISP-DM

- Data understanding (and collection)
  - How does the data generating process look like?
    - Surveys, questionnaires, GPS, social media, experiments,...
  - Data come at a price (but it is an asset)
    - Can previously collected data sets or data stored in internal databases (relational, Hadoop, *etc.*) be used?
    - Otherwise, how will the required data be collected?
      - How costly will that be? It might require complex IT operations
  - Potential issues
    - *E.g.*, privacy (see the ING issue  
<https://www.bloomberg.com/news/articles/2014-03-10/ing-plan-to-share-customer-payment-data-spurs-privacy-concerns>)

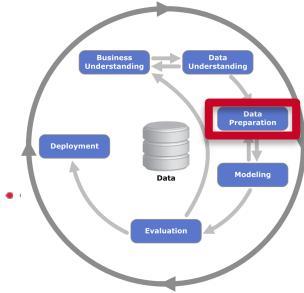


# CRISP-DM

---

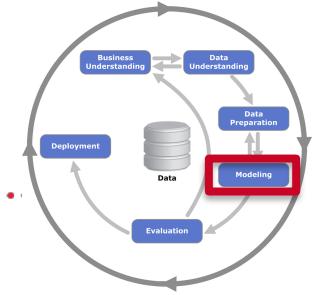
## ➤ Data preparation

- Data sets are rarely in the required form for analysis
  - *E.g.*, unstructured data such as tweets, Facebook posts
- Missing values, wrong values, mixed data types, transformations
- Merging data from different sources



# CRISP-DM

---



## ➤ Modeling

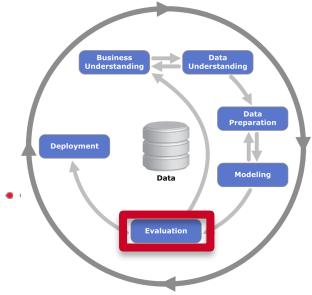
- Extract information using various techniques
  - *E.g.*, face recognition, voice analytics, sentiment analysis, ...
- Often, the most exciting and easiest part of the data-mining process
- The most appropriate technique to be used depends on a series of factors
  - Interpretability, predictive power, speed of learning, speed of application, ...
  - *E.g.*, Netflix paid one million dollars for a model that is not being used (<http://www.forbes.com/sites/ryanholiday/2012/04/16/what-the-failed-1m-netflix-prize-tells-us-about-business-advice/#617cb97757c7>)

# CRISP-DM

---

## ➤ Evaluation

- Internal metrics, domain experts, comparison against baselines
- Acceptance of the solution may be subject to socio-cultural factors
- Feedback to business problem understanding

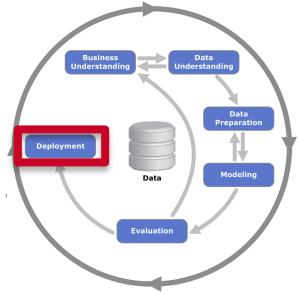


# CRISP-DM

---

## ➤ Deployment

- Put things to use
  - Model will be part of something bigger, *e.g.*, a web page, software, decision support systems
- Evaluation of the acceptance and usage
- Feedback to subsequent business problems
- Reports and presentations

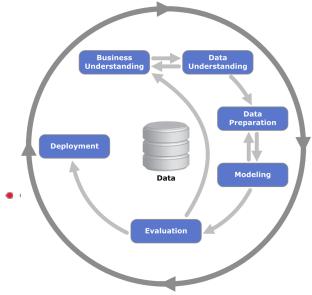


# CRISP-DM

---

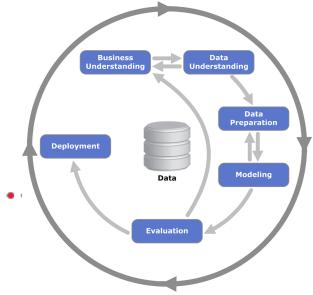
## ➤ Example: Butler company (Lectures 3 – 5)

- Business problem: estimate service demand
- Data understanding: logs from in-house web server
- Data preprocessing
  - Extract IP addresses from logs
  - Find latitude/longitude from IP addresses
- Modeling: simple descriptive analysis (frequencies)
- Evaluation: solution is not satisfactory
  - Strong assumption (service demand = web access)



# CRISP-DM

---

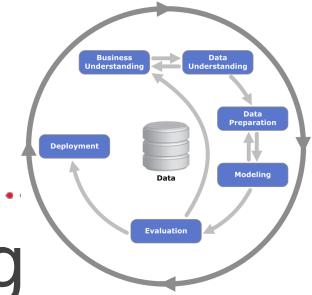


- CRISP-DM can be, and often is, used in conjunction with other project management frameworks (*e.g.*, agile methodologies)
  - SCRUM
  - Kanban
  - Extreme programming
- Again, you will experience CRISP-DM during the course project

# CRISP-DM

---

- In the next five lectures, we will be focusing on the data understanding (collection) step
- This is followed by five lectures on data modeling/evaluation (supervised learning) and data preparation (text mining)
- Focus on unstructured data



# Data Collection

---

## ➤ Where do data come from?

- Internal data

- Relational and NoSQL databases
- Spreadsheets/pivot tables
- Files such as pdfs, slides, doc files
- Logs, e.g., web logs, transaction logs

- External data

- Data freely available on the web
- Private data accessed through APIs, often for a price
- Questionnaires, opinion polls, surveys

# Data Collection

---

- This first lecture is about collecting data “freely” available on the web
  - Textual data embedded in HTML code
  - Other data types/formats (e.g., pdf files)
- Before going any further, let’s review important definitions related to HTML
  - HTML (HyperText Markup Language) is a markup language for formatting the content of web pages
  - Main characteristic: links texts via hyperlinks, which creates a *web*
  - Often used in conjunction with other markup and programming languages such as CSS, JavaScript, and PHP

# HTML

---

- HTML uses "markups" (tags) to annotate text, images, and other content for display in a Web browser
  - *E.g., <head>, <title>, <body>, <header>, <section>, <p>, <div>, <span>, <img>, and many others*
- Let's play with HTML
  - Open the file *webpage.html*

# Web Scraping

---

➤ Let's see the source code of *webpage.html*

- For Chrome browsers, right click on the page and select “View page source” (or press CTRL + U)
- For Internet Explorer/MS Edge, press CTRL + U
- For other browsers see  
<http://www.computerhope.com/issues/ch000746.htm>

# HTML

---

## ➤ Understanding the code



```
<title> My web page </title>
Bla bla bla bla <br>
ble ble ble ble <b> bla bla bla bla </b> <i> asd asd asd
asd</i><br>
<a href="http://www.users.miamioh.edu/carvalag/">
Arthur's webpage</a>
```

## ➤ So, why do we care about HTML?

- Web pages sometimes contain valuable textual data (e.g., tweets)
- Web scraping
  - We will import HTML files into Python and preprocess the HTML code via string manipulations to obtain the desired data

# Case Study

---

- You work for a sports analytics company, which recently signed a contract with ESPN
  - Project: design a predictive model that predicts the final NFL regular season standings per conference
  - The project manager of the analytics team you belong to follows the CRISP-DM methodology
  - Your group is now in the data understanding phase, brainstorming about the relevant variables/features one should include in a predictive model
    - You suggest the number of NFL titles each team won before might be a relevant feature
    - Good news: this info is freely available online

# Background Story

---

- <http://www.espn.com/nfl/superbowl/history/winners>

Super Bowl Winners and Results			
NO.	DATE	SITE	RESULT
I	Jan. 15, 1967	Los Angeles Memorial Coliseum	Green Bay 35, Kansas City 10
II	Jan. 14, 1968	Orange Bowl (Miami)	Green Bay 33, Oakland 14
III	Jan. 12, 1969	Orange Bowl (Miami)	New York Jets 16, Baltimore 7
IV	Jan. 11, 1970	Tulane Stadium (New Orleans)	Kansas City 23, Minnesota 7
V	Jan. 17, 1971	Orange Bowl (Miami)	Baltimore 16, Dallas 13
VI	Jan. 16, 1972	Tulane Stadium (New Orleans)	Dallas 24, Miami 3
VII	Jan. 14, 1973	Los Angeles Memorial Coliseum	Miami 14, Washington 7
VIII	Jan. 13, 1974	Rice Stadium (Houston)	Miami 24, Minnesota 7
IX	Jan. 12, 1975	Tulane Stadium (New Orleans)	Pittsburgh 16, Minnesota 6
X	Jan. 18, 1976	Orange Bowl (Miami)	Pittsburgh 21, Dallas 17
XI	Jan. 9, 1977	Rose Bowl (Pasadena, Calif.)	Oakland 32, Minnesota 14
XII	Jan. 15, 1978	Superdome (New Orleans)	Dallas 27, Denver 10
XIII	Jan. 21, 1979	Orange Bowl (Miami)	Pittsburgh 35, Dallas 31

- How can we collect the above data?

# Web Scraping

---

## ➤ First approach: manual data collection

- Just copy the data from the website and paste it on a spreadsheet
- Works well for our problem
- Can be tedious and very time-consuming for massive data sets
  - You will see this in Assignment 2

# Web Scraping

---

- Second approach: automated data collection
  - We will write scripts that automatically collect data for us
  - Works well when the layout of the web pages do not change often
  - Common approach:
    1. Look for relevant HTML tags/markups
    2. Obtain the data between the open markups (<>) and the close markups (</>)

# Web Scraping

---

- Let's begin
  - Install required package
    - Go to the Terminal and run `pip install requests`

# Web Scraping

---

- Let's load the web page into Python
  - Function from the `requests` module: `get`
    - More on this function during the API classes

```
import requests  
r = requests.get("http://www.espn.com/nfl/superbowl/history/winners")  
html_code = r.text
```

- The whole HTML code is now inside Python
  - One string variable `r.text`

# Web Scraping

- Let's look at the HTML code of the ESPN webpage
- Table tags
  - Row tags `<tr class="...">` and `</tr>`
  - Cell tags `<td>` and `</td>`

NO.	DATE	SITE	RESULT
I	Jan. 15, 1967	Los Angeles Memorial Coliseum	Green Bay 35, Kansas City 10
II	Jan. 14, 1968	Orange Bowl (Miami)	Green Bay 33, Oakland 14
III	Jan. 12, 1969	Orange Bowl (Miami)	New York Jets 16, Baltimore 7
IV	Jan. 11, 1970	Tulane Stadium (New Orleans)	Kansas City 23, Minnesota 7

```
<td colspan="4">Super Bowl Winners and Results</td>
</tr>
<tr class="colhead">
<td>NO.</td>
<td>DATE</td>
<td>SITE</td>
<td>RESULT</td>
</tr>
<tr class="oddrow">
<td>I</td>
<td>Jan. 15, 1967</td>
<td>Los Angeles Memorial Coliseum</td>
<td>Green Bay 35, Kansas City 10</td>
</tr>
<tr class="evenrow">
<td>II</td>
<td>Jan. 14, 1968</td>
<td>Orange Bowl (Miami)</td>
<td>Green Bay 33, Oakland 14</td>
</tr>
<tr class="oddrow">
<td>III</td>
<td>Jan. 12, 1969</td>
<td>Orange Bowl (Miami)</td>
<td>New York Jets 16, Baltimore 7</td>
</tr>
<tr class="evenrow">
<td>IV</td>
<td>Jan. 11, 1970</td>
<td>Tulane Stadium (New Orleans)</td>
<td>Kansas City 23, Minnesota 7</td>
</tr>
```

# Web Scraping

---

- Our approach
  1. Find the positions of the pattern <td> (open tag)
  2. Find the positions of the pattern </td> (close tag)
  3. Obtain the data between the first and second positions
    - String functions
  4. Build a matrix (optional)

# Web Scraping

---

- Finding the positions of the pattern <td>
  - Recall the function `finditer`
    - Obtains **all** the positions where a given pattern occurs

```
import re
```

```
matches = re.finditer("<td>", r.text)  
open_tags = [m.start(0) for m in matches]
```

# Web Scraping

---

- Finding the positions of the pattern </td>

```
matches = re.finditer("</td>", r.text)
```

```
close_tags = [m.start(0) for m in matches]
```

- There is one more element in `close_tags` than in `open_tags`

- The first element in `close_tags` appears first than the first element in `open_tags`
- Let's remove that element from `close_tags`
  - Pay attention to the index 0

```
close_tags.pop(0)
```



```
83 <td colspan="4">Super Bowl Winners and Results</td>
84 </tr>
85 <tr class="colhead">
86 <td>NO.</td>
87 <td>DATE</td>
88 <td>SITE</td>
89 <td>RESULT</td>
90 </tr>
```

# Web Scraping

---

## ➤ Obtaining the data between the first and second positions

- The code below is not the most efficient one (remember: loops are bad in Python), but it is relatively easy to understand
- Try to rewrite the same code using list comprehension

```
web_data = []
for i in range(len(open_tags)):
```

```
    web_data.append(html_code[open_tags[i] + 4:close_tags[i]])
```

```
web_data
```

- Why `close_tags[i]` instead of `close_tags[i] – 1?`
- Why `open_tags[i] + 4` instead of `open_tags[i]`?

# Web Scraping

---

## ➤ Building a data frame

```
import pandas as pd
```

```
rows = []
```

```
#creating a list of lists, each one having 4 elements
```

```
for i in range(4, len(web_data), 4):  
    rows.append(web_data[i:i+4])
```

```
#creating a data frame
```

```
df = pd.DataFrame(rows, columns = web_data[0:4])  
df
```

# Web Scraping

---

- There are easier ways of scraping data from the web with language-dependent solutions
  - E.g., using Python modules such as **beautiful soap**
    - Not always 100% accurate
- We can write more effective code by using more complex regular expressions
  - Example: find all patterns `<td>something</td>`
    - We will learn how to do so in the next lecture

# Web Scraping

---

- Web scraping is not only about HTML
  - One can scrape data from any files on the web
    - PDF, Word/text, Excel/spreadsheet, ...

# Nongraded Homework

---

- Can you spot anything wrong with the resulting `web_data` matrix?
  - Look at the last 13 rows
- Can you further preprocess the data to fix that problem?
  - Hint: for the problematic rows only, you might consider finding what is in between `>` and `</a>`

# Summary

---

- We learned how to scrap data from the web
  - Finding appropriate patterns (e.g., HTML tags)
  - Retrieving content based on patterns (e.g., inside open and close tags)
- Next lecture: web scraping part II (tough lecture)
  - Scraping web data via complex regular expressions

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 8 – Data Collection

*Web Scraping (Part II)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Announcements

---

- Assignment 2
  - Now available on Canvas
  - **New deadline:** Friday, September 24<sup>th</sup>, at 11:59 pm
- Homework 5
  - **New deadline:** Monday, September 27<sup>th</sup>, at 11:59 pm
- Assignment 1
  - I will review the solutions on Tuesday

# Lecture Objectives

---

- Learn how to scrap data from web pages
  - Using regular expressions

# Lecture Instructions

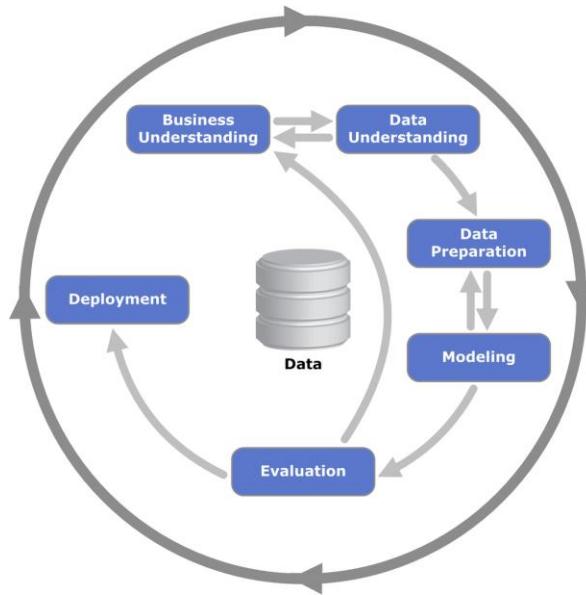
---

- Download the script “*Lecture 8.ipynb*” available on Canvas
- Open the file “*Lecture 8.ipynb*” with VS Code

# CRISP-DM

---

- From previous lecture
  - Big data (data analytics) project management



# CRISP-DM

---

- We are currently focusing on data understanding and collection
  - To a less degree on data preparation
  - Topic of this and the next three lectures
- One very important source of data is the web
  - Big picture: any file freely available on the web can be downloaded and used in an analytics process
    - One must be very careful with copyright violations
- Eventually, we will move on to data analysis
  - Modeling and evaluation

# Web Scraping

---

- Let's continue scraping data from the web
  - Recall the background story
    - You work for a sports analytics company, which recently signed a contract with ESPN
      - Project: design a predictive model that predicts the final NFL regular season standings per conference
      - The project manager of the analytics team you belong to follows the CRISP-DM methodology
      - Your group is now in the data understanding phase, brainstorming about the relevant variables/features one should include in the predictive model
      - You suggest the number of NFL titles each team won before might be a relevant feature

# Background Story

---

- <http://www.espn.com/nfl/superbowl/history/winners>

Super Bowl Winners and Results			
NO.	DATE	SITE	RESULT
I	Jan. 15, 1967	Los Angeles Memorial Coliseum	Green Bay 35, Kansas City 10
II	Jan. 14, 1968	Orange Bowl (Miami)	Green Bay 33, Oakland 14
III	Jan. 12, 1969	Orange Bowl (Miami)	New York Jets 16, Baltimore 7
IV	Jan. 11, 1970	Tulane Stadium (New Orleans)	Kansas City 23, Minnesota 7
V	Jan. 17, 1971	Orange Bowl (Miami)	Baltimore 16, Dallas 13
VI	Jan. 16, 1972	Tulane Stadium (New Orleans)	Dallas 24, Miami 3
VII	Jan. 14, 1973	Los Angeles Memorial Coliseum	Miami 14, Washington 7
VIII	Jan. 13, 1974	Rice Stadium (Houston)	Miami 24, Minnesota 7
IX	Jan. 12, 1975	Tulane Stadium (New Orleans)	Pittsburgh 16, Minnesota 6
X	Jan. 18, 1976	Orange Bowl (Miami)	Pittsburgh 21, Dallas 17
XI	Jan. 9, 1977	Rose Bowl (Pasadena, Calif.)	Oakland 32, Minnesota 14
XII	Jan. 15, 1978	Superdome (New Orleans)	Dallas 27, Denver 10
XIII	Jan. 21, 1979	Orange Bowl (Miami)	Pittsburgh 35, Dallas 31

- How can we collect the above data?

# Web Scraping

---

- Our first solution
  - Find the index (position) of all the HTML tags <td> and </td>
  - Extract what is in between the tags
- This solution is independent of the underlying programming language
  - Relies on very basic **regular expressions** to find patterns
    - *E.g.*, "<td>", "</td>"

# Web Scraping

---

## ➤ Regular expressions

- Very powerful language with its own syntax and semantics
  - Can be used in conjunction with virtually any programming and querying languages
    - JAVA, C++, SQL, R, ...
- Concise and powerful method for searching for a specific string pattern
  - We will learn how to write more complex regular expressions today
    - Example: <td>some text</td>



vague specification

# Web Scraping

---

- Regular expressions can be used in conjunction with most string functions
- In our classes, we shall focus on the functions `search`, `finditer`, and `sub`
  - `search`: finds the first occurrence of a pattern
  - `finditer`: can be used to find positions and matches of a pattern
  - `sub`: replace the matches of a pattern with another string
- All the above functions are from the `re` module
  - Documentation: <https://docs.python.org/3/library/re.html>

# Regular Expressions

---

- Up to now, we used regular expressions to search for patterns containing entire strings (no vagueness)
  - *E.g.*, “HTTP” or “<td>”
- But we can be more vague in our expressions
  - This vagueness is often captured by the fundamental building blocks that match a single character
  - *E.g.*, the expression “San Francisco [1-9]” will look for the string “San Francisco ” followed by one character: a digit between 1 and 9
    - **The brackets [ ] in a regular expression indicate that we are looking for one particular character**

# Regular Expressions

---

- Let's try the previous expression

```
import requests  
import re
```

```
r = requests.get("http://www.espn.com/nfl/superbowl/history/winners")
```

```
html_code = r.text
```

```
hits = re.finditer("San Francisco [0-9]", html_code)
```

```
positions = [m.start(0) for m in hits]
```

```
positions
```

- Result: positions where the matches start

# Regular Expressions

---

Syntax:  
some of the  
most important  
expressions to  
find individual  
characters

Expression	Description
.	Any character
[0-9]	Any numeric character in the range 0, 1, 2, . . . , 9
[A-Z]	Any uppercase alphabetic character in the range A, B, . . . , Z
[a-z]	Any lowercase alphabetic character in the range a, b, . . . , z
\s	Space characters: tab, newline, vertical tab, form feed, carriage return, space

More expressions at: <https://docs.python.org/3/library/re.html>

# Regular Expressions

---

- Example #1: has San Francisco ever been to a Super Bowl?
  - Pattern: search for the first occurrence of “San Francisco”
  - Two possible ways:
    1. Search for the whole string “San Francisco”
    2. Search for “San” followed by a space character \s, followed by “Francisco”

```
hits = re.finditer("San Francisco", html_code)
```

```
hits = re.finditer("San\sFrancisco", html_code)
```

# Regular Expressions

---

- Example #2: retrieve the date of each Super Bowl

- Pattern: [A-Z][a-z][a-z]. [0-9][0-9], [0-9][0-9][0-9][0-9]

Example: J a n . 1 5 , 1 9 6 7

- Not a perfect pattern (why?)

```
hits = re.finditer("[A-Z][a-z][a-z]. [0-9][0-9], [0-9][0-9][0-9][0-9]", html_code)  
positions = [m.start(0) for m in hits]
```

- Note that the above code only retrieves the position of the patterns
  - Not the matched values

# Regular Expressions

---

- Example #2: retrieve the date of each Super Bowl
  - Retrieving matched values
    - Previously, we looked for two separate regular expressions to get data in between the patterns' locations
    - We are now looking for a single regular expression

```
hits = re.finditer("[A-Z][a-z][a-z]. [0-9][0-9], [0-9][0-9][0-9][0-9]", html_code)  
matches = [m.group(0) for m in hits]
```

# Regular Expressions

---

- Difference between `start()` and `group()` in `finditer`
  - `start`: returns the positions where the hits start
  - `group`: returns the precise matches
- So, what's with the argument '0' in `start()` and `group()`?
  - One can define “groups” in a pattern using parenthesis
    - The value 0 returns the positions/matches for the entire hit
    - The values 1, 2, 3, ... return the positions/matches for the first, second, third, ... groups
  - Example: creating two groups for the previous pattern

```
hits = re.finditer("(?P<first>[A-Z][a-z][a-z]). (?P<second>[0-9][0-9]), (?P<third>[0-9][0-9][0-9][0-9])", html_code)
```

```
matches_group_1 = [m.group(1) for m in hits]
```

# Regular Expressions

---

- Up to now, we coded repetitions of a character by explicitly writing the same pattern multiple times
  - Example: [A-Z][a-z][a-z]. [0-9][0-9], [0-9][0-9][0-9][0-9]
    - *E.g.*, the last [0-9] pattern is written multiple times
  - There are more general ways of defining repetitions:

Quantifier	Meaning
*	The preceding term will be matched zero or more times
?	The preceding term is optional and will be matched at most once
+	The preceding term will be matched one or more times
{n}	The preceding term is matched exactly n times
{n, }	The preceding term is matched n or more times
{n, m}	The preceding term is matched at least $n$ times, but no more than $m$ times

# Regular Expressions

---

## ➤ Example #2: retrieve the date of each Super Bowl

- Previous expression: [A-Z][a-z][a-z]. [0-9][0-9], [0-9][0-9][0-9][0-9]
- Alternative: [A-Z][a-z]{2}. [0-9]{2}, [0-9]{4}
- Note that there are dates that are missing in the above result  
(wrong pattern)
  - *E.g.*, Feb. 1, 2004

# Regular Expressions

---

- Example #2: retrieve the date of each Super Bowl
  - Fixing the previous expression
    - Expression: [A-Z][a-z][a-z]. [0-9]{1,2}, [0-9]{4}
      - Explanation: the day of the game can have at least 1 and at most 2 digits

# Regular Expressions

## Example #3:

- Obtaining all the relevant data from <http://www.espn.com/nfl/superbowl/history/winners>
- Cell tags
  - <td> and </td>

NO.	DATE	SITE	RESULT
I	Jan. 15, 1967	Los Angeles Memorial Coliseum	Green Bay 35, Kansas City 10
II	Jan. 14, 1968	Orange Bowl (Miami)	Green Bay 33, Oakland 14
III	Jan. 12, 1969	Orange Bowl (Miami)	New York Jets 16, Baltimore 7
IV	Jan. 11, 1970	Tulane Stadium (New Orleans)	Kansas City 23, Minnesota 7

```
<td colspan="4">Super Bowl Winners and Results</td>
</tr>
<tr class="colhead">
<td>NO.</td>
<td>DATE</td>
<td>SITE</td>
<td>RESULT</td>
</tr>
<tr class="oddrow">
<td>I</td>
<td>Jan. 15, 1967</td>
<td>Los Angeles Memorial Coliseum</td>
<td>Green Bay 35, Kansas City 10</td>
</tr>
<tr class="evenrow">
<td>II</td>
<td>Jan. 14, 1968</td>
<td>Orange Bowl (Miami)</td>
<td>Green Bay 33, Oakland 14</td>
</tr>
<tr class="oddrow">
<td>III</td>
<td>Jan. 12, 1969</td>
<td>Orange Bowl (Miami)</td>
<td>New York Jets 16, Baltimore 7</td>
</tr>
<tr class="evenrow">
<td>IV</td>
<td>Jan. 11, 1970</td>
<td>Tulane Stadium (New Orleans)</td>
<td>Kansas City 23, Minnesota 7</td>
</tr>
```

# Regular Expressions

---

## ➤ Example #3: (last class)

- Obtaining all the relevant data from  
<http://www.espn.com/nfl/superbowl/history/winners>
  - Pattern: <td>.+?</td>
    - Rationale: Find a pattern that begins with <td>, ends with </td>, and has one or more characters in between
      - . = any character
      - + = the preceding item will be matched one or more times
      - ? = the preceding item will be matched at most once
        - This is to avoid “greediness,” i.e., the pattern stops as soon as it finds </td>

# Regular Expressions

---

- Understanding the role of ‘?’
  - The \* and + qualifiers are all greedy
    - They match as much text as possible
    - Adding ? after the qualifier makes it perform the match in non-greedy or minimal fashion
  - Example
    - Consider the string <a> b c>
      - The pattern <.+> returns the whole string <a> b c>
      - The pattern <.+?> returns the string <a>

# Regular Expressions

---

## ➤ Escape characters

- How does one search for a pattern containing a dot, a parenthesis, a bracket, *etc.*?
  - For example, the character . cannot be used because it means “all characters”
  - We have to use “escaping”
    - Escaping is done with a single backslash \ in regular expressions in Python
- Example: look at the difference between the codes below

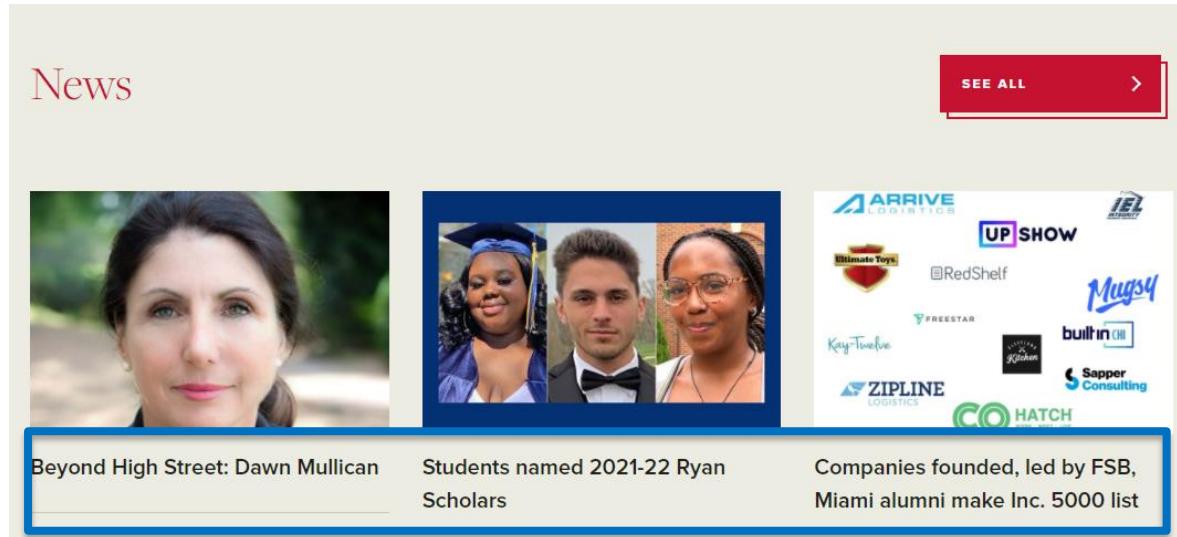
```
hits = re.findall(".", html_code)  
hits = re.findall("\.", html_code)
```

.....

# Case Study: FSB Website

# Regular Expressions

- Obtaining Farmer School news
  - Headlines only
  
- Open the source code of  
<http://miamioh.edu/fsb/>



# Regular Expressions

---

- Step #1: find an appropriate regular expression
  - How would you retrieve the headline “*Beyond High Street: Dawn Mullican*”

```
</div>
<article class="cards-news__card">
    <div class="cards-news__image-container">
        
    <div class="cards-news__card-inner">
        <p class="cards-news__headline">Beyond High Street: Dawn Mullican</p>
    </div>
    </a>
</article>
<article class="cards-news__card">
    <div class="cards-news__image-container">
        
```

# Regular Expressions

---

## ➤ Obtaining headlines from FSB website

- Let's use the pattern "**cards-news\_\_headline.+</p>**", but first let's remove unnecessary characters from the HTML page

```
r = requests.get("https://miamioh.edu/fsb/")
html_code = r.text
hits = re.finditer("cards-news__headline.+?</p>", html_code)
headlines = [m.group(0) for m in hits]
```

```
</div>
<article class="cards-news__card">
    <div class="cards-news__image-container">
        <a class="cards-news__card-link" href="https://miamioh.edu/fsb/news/2021/09/Beyond-High-Street-Dawn-Mullican.html" target="_parent">
            
        <div class="cards-news__card-inner">
            <p class="cards-news__headline">Beyond High Street: Dawn Mullican</p>
        </div>
    </a>
</article>
<article class="cards-news__card">
    <div class="cards-news__image-container">
        <a class="cards-news__card-link" href="https://miamioh.edu/fsb/news/2021/09/Students-named-2021-22-Ryan-Scholars.html" target="_parent">
            

```

# Regular Expressions

---

- Obtaining headlines from FSB website
  - Let's look inside the **headlines** variable
    - The data of interest is there alongside lots of garbage (undesirable HTML tags)
    - Nongraded homework: clean our data with regular expressions

# Web Scraping

---

➤ Can one scrap all the data on the web? No!!!

- Legal issues
  - Copyright
- Website owners can:
  - Block an IP address who is requesting too many webpages in a short period of time
  - Use captchas
  - Make small/random changes to HTML code
- Controlled access to data is often done via API
  - Next lecture



# Nongraded Homework

---

- Let's clean our headlines using regular expressions
  - Task #1: remove the start of the pattern immediately before the headline

For example, after this step, the entry: "*cards-news\_\_headline*">*Beyond High Street: Dawn Mullican*</p>" becomes: "*Beyond High Street: Dawn Mullican*</p>"
  - Task #2: remove the end of the pattern

For example, after this step, the entry: "*Beyond High Street: Dawn Mullican*</p>" becomes: "*Beyond High Street: Dawn Mullican*"
  - Hint: Use the function `re.sub` to replace a pattern with `""` (i.e., nothing)
    - Syntax: `re.sub(pattern, replacement, text_data)`

# Summary

---

- We learned how to scrap data from the web
  - Using more complex regular expressions
- Further readings
  - [https://en.wikipedia.org/wiki/Web\\_scraping](https://en.wikipedia.org/wiki/Web_scraping)
  - [https://en.wikipedia.org/wiki/Regular\\_expression](https://en.wikipedia.org/wiki/Regular_expression)
  - <https://docs.python.org/3/library/re.html>
- Next Lecture
  - Collecting data via APIs

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 9 – Data Collection

*APIs (Part I)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Lecture Objectives

---

- Quick review of Assignment 1
- Learn about APIs and how they are used during data collection
- Understand the data exchange format called JSON

# Lecture Instructions

---

- Download the notebook “*Lecture 9.ipynb*” available on Canvas
  - Open the above file with VS Code
- Download the file NYTimes.json (optional)

# Lecture Instructions

---

- Go to <https://developer.nytimes.com/>
  - Click on “*Get Started*” -> “*Create account*”
  - Fill in the text fields
  - Check your email and verify your identity
  - Click on “*Sign in,*” and use your email and password to sign in

# Lecture Instructions

---

## ➤ Continuing ...

- Click on your email address (top right) -> “Apps”
- Click on “+ NEW APP”
  - Pick an app name (e.g., ISA414)
  - Enable the “Article Search API” option
- Click on Save

Name	Description	Status	Actions
Archive API	Get all NYT article metadata for a given month.	—	<input type="button" value="Enable"/>
Article Search API	Search for New York Times articles.	—	<input checked="" type="button" value="Save to enable"/> <input type="button" value="Cancel"/>
Books API	Get NYT Best Sellers Lists and lookup book reviews.	—	<input type="button" value="Enable"/>
Community API	Get user comments. (BETA)	—	<input type="button" value="Enable"/>
Most Popular API	Popular articles on NYTimes.com.	—	<input type="button" value="Enable"/>
Movie Reviews API	Search for movie reviews.	—	<input type="button" value="Enable"/>
RSS Feeds	NYT RSS section feeds.	—	<input type="button" value="Enable"/>
Semantic API	Get semantic terms (people, places, organizations, and locations).	—	<input type="button" value="Enable"/>
Times Tags API	NYT controlled vocabulary.	—	<input type="button" value="Enable"/>
Times Wire API	Real-time feed of NYT article publishes.	—	<input type="button" value="Enable"/>
Top Stories API	Get articles currently on a section front or the home page.	—	<input type="button" value="Enable"/>

# Lecture Instructions

---

- Copy your API Key and save it somewhere

API Keys

Key	Copy to clipboard	Status	Created	Expires
jz0FpxsVi6O0p4jwIe3YRsU... 	mX7B1nVtgbuC...	Active	2019-08-10	never 

# Data Collection

---

➤ Can one scrap all the data on the web? No!!!

- Legal issues
  - Copyright
- Website owners can:
  - Block an IP address that is requesting too many webpages in a short period of time
  - Use captchas
  - Make small/random changes to HTML code
- Controlled access to data is often done via API
  - Today's lecture



# Data Collection

---

## ➤ Data are assets

- Not all relevant data are freely available on the web
- Many companies make money by directly or indirectly selling data, e.g., Twitter, Google, Facebook
  - How can we access these companies' data?
    - Which protocol should we follow?
  - What is the format of the provided data?
    - Unlikely to be CSV
  - We will address the above questions in this lecture

# API

---

## ➤ API: Application Programming Interface

- Set of protocols that allow two+ software to communicate with each other
  - *E.g.*, to exchange data
- Example:
  - When you write a script that formally requests data from the NY Times, that script must follow a certain protocol
    - Set of rules

# API

---

- There are different types of APIs
  - Operating Systems (OS) APIs
    - Software/apps can access functionalities of the underlying OS
    - *E.g.*, an app can request Android to access a phone's camera
      - The request must follow a certain pre-defined protocol
      - The OS can approve or deny the request
  - Web APIs
    - Sometimes referred to as *web services*
    - Requests are often made via a protocol called REST
    - Responses are typically expressed in JSON or XML

# API

---

- When using or developing web APIs, one must consider both:
  1. The protocol required when requesting data
  2. The format of the resulting responses
- We will learn how to use web APIs to request data, but not how to program/create APIs
  - Beyond the scope of this course

## ➤ Protocols

- There are different ways of specifying an API protocol
- Our focus in this course: **REST** or **RESTful** systems
  - Currently, the most popular method to create web APIs
  - Every resource (e.g., data) is represented uniquely
    - For example, *via* URLs
  - Standard HTTP methods are very often used for retrieving, replacing, uploading, or deleting data
    - For example, one can request data similarly to requesting a web page via URLs

# API

---

## ➤ HTTP-based REST operations

Uniform Resource Locator (URL)	GET	PUT	POST	DELETE
Collection of files, such as <code>http://api.example.com/resources/</code>	List the URLs	Replace the entire collection of files with another collection	Create a new entry in the collection of files	Delete the entire collection
Element, such as <code>http://api.example.com/resources/item10</code>	Retrieve an element of the collection	Replace an element of the collection	Create a new resource to an existent element	Delete an element of the collection

# API

---

- We will focus primarily on request (GET) operations
  - We shall work with POST operations in the second half of the course
- Let's make some GET requests
  - Background story
    - Suppose you work for Microsoft
    - Your job: monitor social media and news outlets
      - Goal: quickly react to bad/good news
    - Let's find out what the NY Times is writing about Microsoft

# API

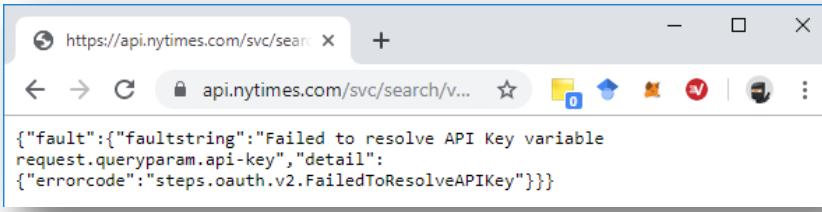
---

## ➤ Let's make some GET requests

- First, via a web browser to build intuition, then via Python
- Base URL of NY Times API:

<https://api.nytimes.com/svc/search/v2/articlesearch.json>

- After requesting data via the above URL, we get the following response
  - Some sort of an error



A screenshot of a web browser window. The address bar shows the URL <https://api.nytimes.com/svc/search/v2/articlesearch.json>. The main content area displays a JSON error response:

```
{"fault": {"faultstring": "Failed to resolve API Key variable request.queryparam.api-key", "detail": {"errorcode": "steps.oauth.v2.FailedToResolveAPIKey"}}}
```

# API

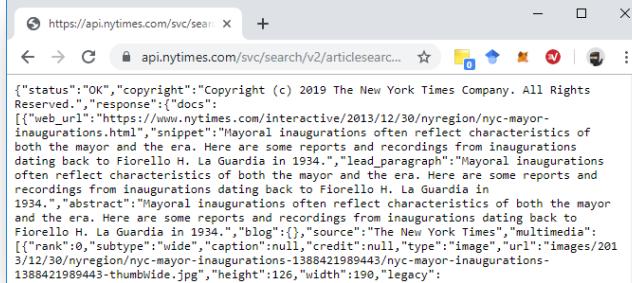
---

- The previous message says that the API Key is missing
  - That is, we are not authorized to request data
- API authorization is often tackled via login/pass and/or a token
  - Data access control
  - NY Times requires a token you obtained in the beginning of the lecture
    - Not always free
  - Let's try another URL:  
[https://api.nytimes.com/svc/search/v2/articlesearch.json?api-key=add –your-token-here](https://api.nytimes.com/svc/search/v2/articlesearch.json?api-key=add-your-token-here)
    - The question mark ‘?’ in the URL means that what comes next is a series of key-value pairs that will be used by the API to provide the right answer to the request

# API

---

- It seems we get something
  - Data in a weird format (more on that soon)



A screenshot of a web browser window displaying a JSON response from the New York Times API. The URL in the address bar is <https://api.nytimes.com/svc/search/v2/articlesearch.json>. The page content shows a single article document:

```
{"status": "OK", "copyright": "Copyright (c) 2019 The New York Times Company. All Rights Reserved.", "response": {"docs": [{"web_url": "https://www.nytimes.com/interactive/2013/12/30/nyregion/nyc-mayor-inaugurations.html", "snippet": "Mayoral inaugurations often reflect characteristics of both the mayor and the era. Here are some reports and recordings from inaugurations dating back to Fiorello H. La Guardia in 1934.", "lead_paragraph": "Mayoral inaugurations often reflect characteristics of both the mayor and the era. Here are some reports and recordings from inaugurations dating back to Fiorello H. La Guardia in 1934.", "abstract": "Mayoral inaugurations often reflect characteristics of both the mayor and the era. Here are some reports and recordings from inaugurations dating back to Fiorello H. La Guardia in 1934.", "blog": {}, "source": "The New York Times", "multimedia": [{"rank": 0, "subtype": "wide", "caption": null, "credit": null, "type": "image", "url": "images/2013/12/30/nyregion/nyc-mayor-inaugurations-1388421989443/nyc-mayor-inaugurations-1388421989443-thumbWide.jpg", "height": 126, "width": 190, "legacy": true}]}]}
```

- Let's get data (articles) about Microsoft
  - Key =  $q$ , value = 'Microsoft'

<https://api.nytimes.com/svc/search/v2/articlesearch.json?api-key=add-your-token-here&q='Microsoft'>

# API

---

- Once again, we get some data in a weird format, but this time about Microsoft

```
{"status": "OK", "copyright": "Copyright (c) 2018 The New York Times Company. All Rights Reserved.", "response": {"docs": [{"web_url": "https://topics.nytimes.com/top/news/business/companies/microsoft_corporation/index.html", "snippet": "News about Microsoft Corporation including commentary and archival articles published in The New York Times.", "blog": {}, "multimedia": [], "headline": {"main": "Microsoft Corporation", "kicker": null, "content_kicker": null, "print_headline": null, "name": null, "seo": null, "sub": null}, "keywords": []}, {"document_type": "topic", "type_of_material": "timestopic", "id": "527ab6b8138f0d86606631f57", "word_count": 15, "score": 13.6317077}, {"web_url": "https://topics.nytimes.com/topic/company/microsoft-corporation", "snippet": "News about Microsoft Corporation, including commentary and archival articles published in The New York Times.", "blog": {}, "multimedia": [], "headline": {"main": "Microsoft Corporation", "kicker": null, "content_kicker": null, "print_headline": null, "name": null, "seo": null, "sub": null}, "keywords": []}, {"document_type": "topic", "type_of_material": "timestopic", "id": "56e0233c38f0d836dc036a59", "word_count": 15, "score": 13.6236665}, {"web_url": "https://topics.nytimes.com/top/news/business/companies/yahoo_inc/yahoo-microsoft-deal/index.html", "snippet": "News about Yahoo-Microsoft Deal, including commentary and archival articles published in The New York Times.", "blog": {}, "multimedia": [], "headline": {"main": "Yahoo-Microsoft Deal", "kicker": null, "content_kicker": null, "print_headline": null, "name": null, "seo": null, "sub": null}, "keywords": []}, {"document_type": "topic", "type_of_material": "timestopic", "id": "527ab88238f0d866066321c", "word_count": 16, "score": 2.8483667}, {"web_url": "https://www.nytimes.com/2018/02/15/technology/personaltech/microsoft-office-chromebook.html", "snippet": "If you have a device running the Chrome OS but prefer Microsoft Office to Google Docs, you don't have to change your ways much.", "blog": {}, "source": "The New York Times", "multimedia": [{"rank": 0, "subType": "xLarge", "caption": null, "type": "Image", "url": "images/2018/02/16/technology/personaltech/16techtipwebART/16techtipwebART-articleLarge.jpg", "height": 497, "width": 600, "legacy": {"xLargeWidth": 600, "xLarge": "images/2018/02/16/technology/personaltech/16techtipwebART/16techtipwebART-articleLarge.jpg", "xLargeHeight": 497}, "subType": "xLarge", "crop_name": "articleLarge"}, {"rank": 0, "subType": "wide", "caption": null, "credit": null, "type": "Image", "url": "images/2018/02/16/technology/personaltech/16techtipwebART/16techtipwebART-thumbWide.jpg", "height": 126, "width": 190, "legacy": {"wide": "images/2018/02/16/technology/personaltech/16techtipwebART/16techtipwebART-thumbWide.jpg", "wideWidth": 190, "wideHeight": 126}}, "subType": "wide", "crop_name": "thumbWide"}]}]
```

- Note the use of ‘&’ in the previous URL
  - Implies that another key-value term is coming
- How does one know the key-value pairs to use in a request?
  - E.g., *api-key* and *q*
  - Answer: read the API documentation:  
<https://developer.nytimes.com/docs/articlesearch-product/1/overview>

# API

---

## ➤ To summarize

- REST protocol to request data
  - GET, PUT, POST, DELETE operations
    - GET operations is similar to requesting a webpage, *i.e.*, one needs a URL
  - Common URL format:  
[www.baseurl.com/somepage?key1=value1&key2=value2...](http://www.baseurl.com/somepage?key1=value1&key2=value2...)
    - Example:  
<https://api.nytimes.com/svc/search/v2/articlesearch.json?api-key=jz0FpxsVi6O0p4jwle3YRsUcQHPFdDhx&q='Microsoft'>

# JSON

---

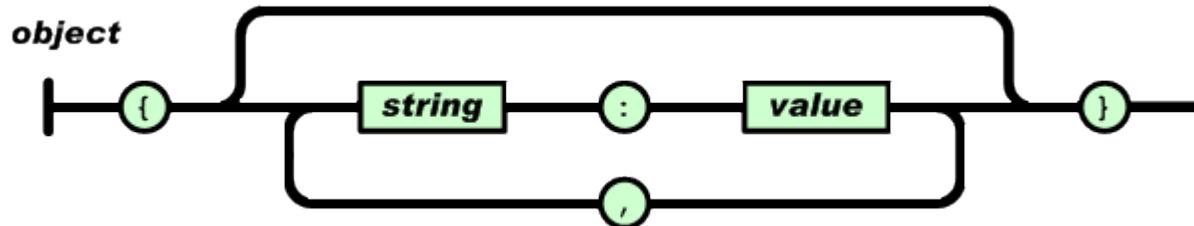
- Let's understand the retrieved data now
  - It is in a format called JSON (JavaScript Object Notation)
    - Data-interchange language
      - Widely used to communicate and store unstructured data
    - Textual (like CSV)
      - Format is completely language independent
    - Two key structures
      - A collection of key/value pairs, also known as *objects*
      - An ordered list of values, also known as *vector/array*

# JSON

---

## ➤ Object

- An unordered set of key-value pairs
- Begins with { and ends with }
- Each key is followed by : and the key-value pairs are separated by ,



# JSON

---

## ➤ Object

- Example: Professor (3 key-value pairs)

```
{"name":"Arthur",  
 "dept":"ISA",  
 "position":"Paliwal Innov. Chair",  
 "position 2":"Assistant Professor"}
```

```
{ "name":"Skip",  
 "dept":"ISA",  
 "position":"Dept. Chair" }
```

name	dept	position	position 2
Arthur	ISA	Assistant Professor	Paliwal Innov. Chair
Skip	ISA	Dept. Chair	

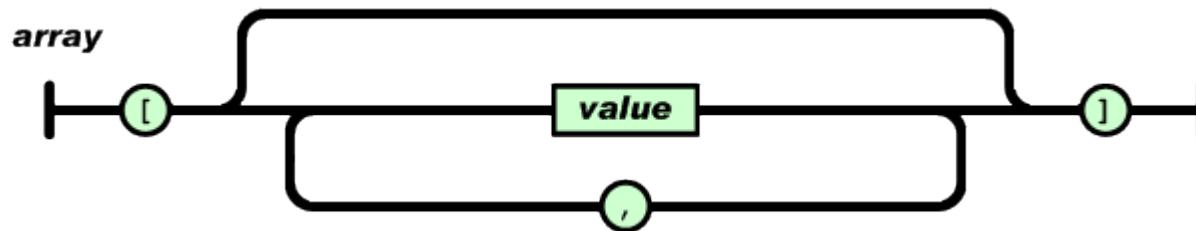
The above data could also be stored as a table. So why using JSON?  
Answer: flexibility

# JSON

---

## ➤ Array/vector

- Ordered collection of values
- Begins with [ and ends with ] (*i.e.*, square brackets)
- Values are separated by comma



# JSON

---

- Array/vector

- Example:

```
{ "TWD characters" : ["Rick", "Michonne", "Negan"] }
```

# JSON

---

- We can always combine objects and vectors

```
{  
  "menu": {  
    "id": "night",  
    "menuitem": [  
      {"Item": "Rice", "Price": 3},  
      {"Item": "Beef", "Price": 10},  
      {"Item": "Chicken", "Price": 9}  
    ]  
  }  
}
```

- Main object: **menu**
- Value associated with **menu**: an object containing 2 key-value pairs
  - Key-value pair #1: "id":"night"
  - Key-value pair #2: "menuitem":array containing 3 objects
    - Item: Rice, Price: 3
    - Item: Beef, Price: 10
    - Item: Chicken, Price:9

# JSON

---

- Let's look at the JSON file you got from the NY Times
  - Open the file *NYTimes.json* (File -> Open File ...) with VS Code

```
{  
  "response": {  
    "meta": {  
      "hits": [29997],  
      "time": [9],  
      "offset": [0]  
    }  
    "docs": [...],  
  }  
  "status": ["OK"],  
  "copyright": ["Copyright (c) ..."]  
}
```

**Line 1:** the '{' indicates that the whole response is an object

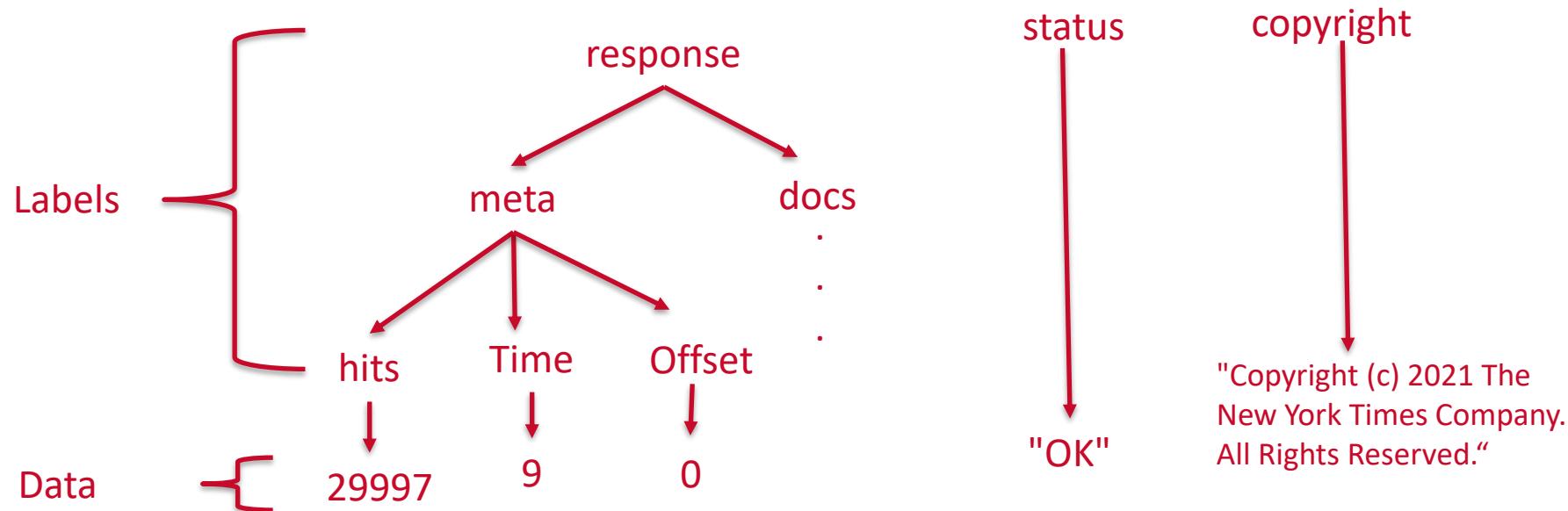
**Black lines:** the key *response* is associated with an object. That object in turn has two key-value pairs: *meta* and *docs*

**Green line:** 2<sup>nd</sup> key-value pair

**Blue line:** 3<sup>rd</sup> key-value pair

# JSON

- The best way of interpreting JSON files is by means of a hierarchical tree



# JSON

---

## ➤ Let's process JSON files with Python

- Install the required modules: `pip install requests`
- Request the JSON file

```
import requests
```

```
response = requests.get("https://api.nytimes.com/svc/search/v2/articlesearch.json?api-key=ADD YOUR TOKE HERE&q='Microsoft'")
```

```
json_data = response.json()
```

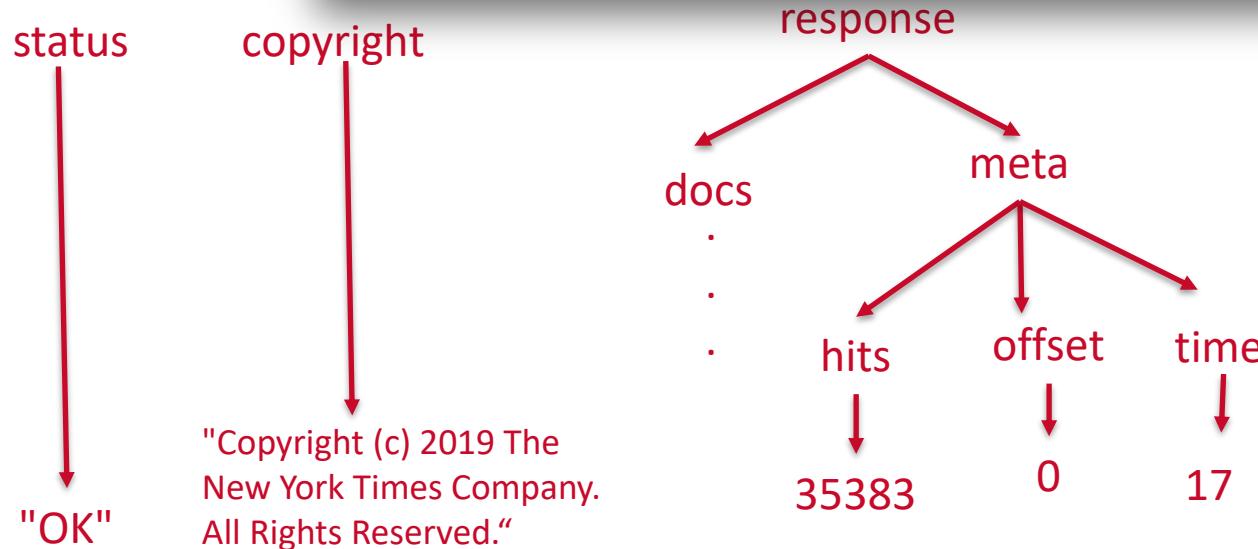
- Python has a data structure called **dictionary** that handles the JSON format
  - We used that format before when creating data frames

# JSON

- Let's navigate through the JSON file

- With the help of VS Code, we can see that `json_data` is a **dictionary**

Name	Type	Size	Value
<code>json_data</code>	dict	3	<code>{'status': 'OK', 'copyright': 'Copyright (c) 2019 The New York Times Company. All Rights Reserved.'}</code>



# JSON

---

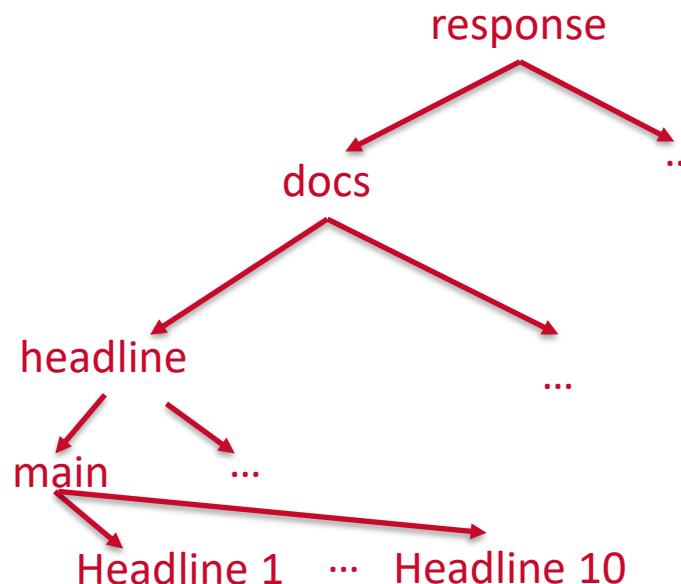
- The data we are looking for are inside/below *response* and *docs*
  - Let's see what is inside `response`

```
response_data = json_data['response']  
response_data.keys()
```

# JSON

---

## ➤ Obtaining the headlines



```
docs = response_data['docs']
```

```
headlines = []
for i in range(len(docs)):
    headlines.append(docs[i]['headline']['main'])
```

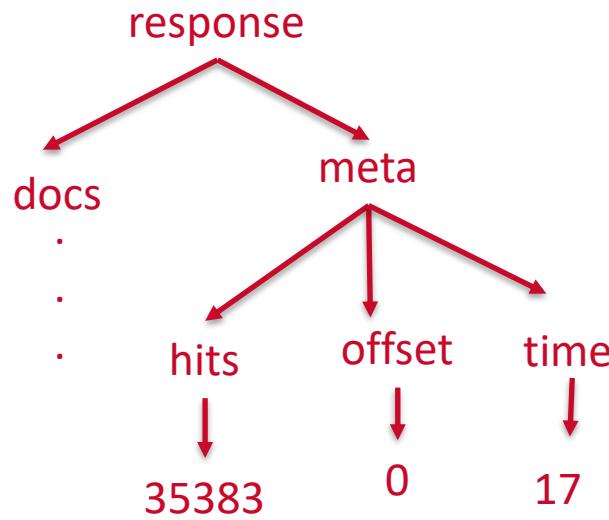
# JSON

---

- Let's obtain the number of hits (articles about Microsoft)

```
meta = response_data['meta']
```

```
meta['hits']
```



# API

---

## ➤ Final note

- APIs make data collection very easy
- Think about how cumbersome it would be to collect Microsoft-related articles from the NY Times via web scrapping (assuming this is allowed)
  - Download all articles
  - Filter the articles related to Microsoft
- Can one get the content of a NYT article?
  - No (at the time of writing)

# API

---

## ➤ Final note

- There are a number of APIs out there
  - YouTube: <https://developers.google.com/youtube/>
  - Google Maps: <https://developers.google.com/maps/>
  - Flickr: <https://www.flickr.com/services/api/>
  - Facebook: <https://developers.facebook.com/docs/graph-api>
  - Amazon Product Advertising: <https://affiliate-program.amazon.com/gp/advertising/api/detail/main.html>
  - Fitbit: <https://dev.fitbit.com/>
  - Twitter: <https://developer.twitter.com/en/docs.html>
  - KDnuggets 52 useful APIs: <http://www.kdnuggets.com/2017/02/machine-learning-data-science-apis-updated.html>

# Summary

---

- We learned about the concept of APIs
  - Request: REST protocol
  - Response: JSON file
- Next lecture
  - Introduction to APIs – Part II
    - Response: XML

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 10 – Data Collection

*APIs (Part II)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Lecture Objectives

---

- Continue to learn about APIs and how they are used during data collection
  - Understand the data exchange format called XML

# Lecture Instructions (Part I)

---

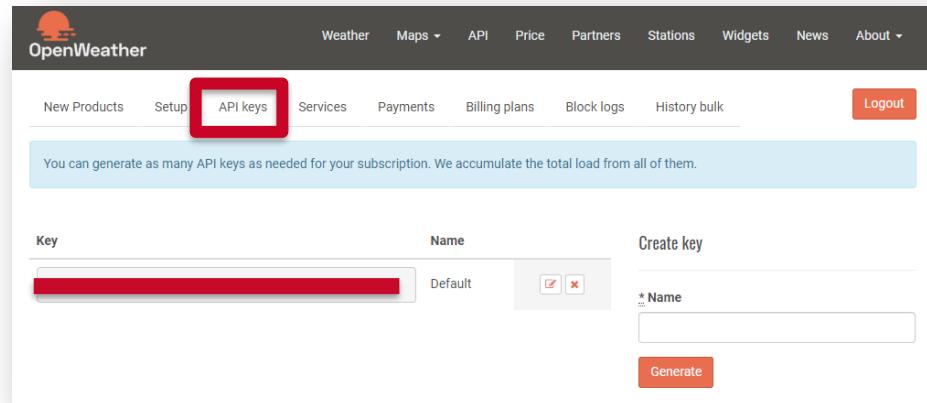
- Download the notebook “*Lecture 10.ipynb*” available on Canvas
  - Open the above file with VS Code
  
- Download the file *cdlib.xml* available on Canvas (optional)

# Lecture Instructions (Part II)

---

## ➤ Obtain an API key from OpenWeather

1. Go to [https://home.openweathermap.org/users/sign\\_up](https://home.openweathermap.org/users/sign_up)
  - Fill in the text fields
  - Click on “Create Account”
2. Go to “API keys” or  
“My API Keys” under your  
username
  - Save your key



# Lecture Instructions (Part III)

---

## ➤ Obtain an API key from News API

- Go to <https://newsapi.org>
- Click on “Get API key”
- Fill in the forms and click on “Submit”
- Save your API key

### Register for API key

First name

Arthur

Email address

carvalag@miamioh.edu

Choose a password

.....



You are...

I am an individual

I am a business, or am working on behalf of a business

I'm not a robot



Privacy • Terms

I agree to the [terms](#).

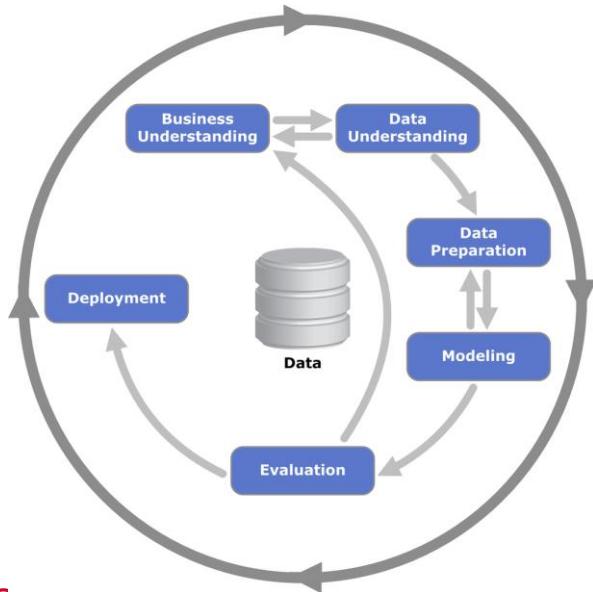
I promise to add an [attribution link](#) on my website or app to NewsAPI.org.

Submit

# CRISP-DM

---

- From previous lecture
  - Big data (data analytics) project management



- We are currently focusing on data understanding/collection
  - To a less degree, on data preparation

# Data Collection

---

- What have we learned up to know?
  - Scrap (unstructured) data from the web
    - Retrieve relevant data from HTML files
      - Regular expressions
  - Collect data via APIs
    - Request: REST + HTTP
    - Response: JSON

# Data Collection

---

- What does come next?
  - Today: collect data via APIs
    - Request: REST + HTTP
    - Response: XML
  - Next lectures: collect data from databases
    - Querying a document-oriented database (MongoDB)
      - Unstructured data

# Data Collection

---

## ➤ Data are assets

- Not all relevant data are freely available on the web
- Many companies make money by directly or indirectly selling data, e.g., Twitter, Google, Facebook
  - How can we access these companies' data?
    - Via APIs
  - Which protocol must we follow?
    - REST (most popular), SOAP, ...
  - What is the format of the provided data?
    - JSON, XML, ...

# XML

---

## ➤ eXtensible Markup Language

- Markup language for data exchange
  - Similar to HTML, there are tags that indicate the beginning and end of each field in an XML file
  - But different than HTML, XML is not used for formatting texts
    - Instead, it is used to store data

Text  
formatting

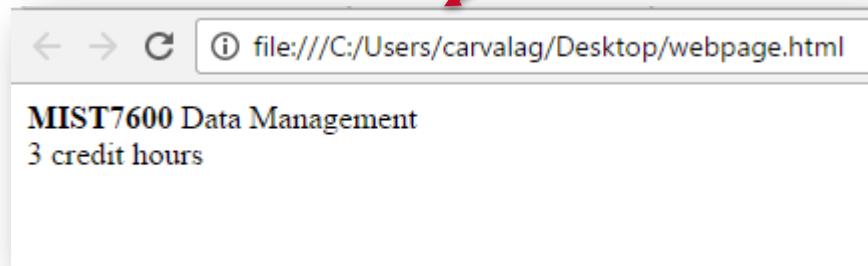
HTML	XML
<pre>&lt;p&gt;&lt;b&gt;MIST7600&lt;/b&gt; Data Management&lt;br&gt; 3 credit hours&lt;/p&gt;</pre>	<pre>&lt;course&gt; &lt;code&gt;MIST7600&lt;/code&gt; &lt;title&gt;Data Management&lt;/title&gt; &lt;credit&gt;3&lt;/credit&gt; &lt;/course&gt;</pre>

Data about  
a course

# XML

HTML	XML
<pre>&lt;p&gt;&lt;b&gt;MIST7600&lt;/b&gt; Data Management&lt;br&gt; 3 credit hours&lt;/p&gt;</pre>	<pre>&lt;course&gt; &lt;code&gt;MIST7600&lt;/code&gt; &lt;title&gt;Data Management&lt;/title&gt; &lt;credit&gt;3&lt;/credit&gt; &lt;/course&gt;</pre>

equivalent to



equivalent to

Table COURSE

Code	Title	Credit
MIST7600	Data Management	3

# XML

---

- Major rules
  - Elements must have both an opening and closing tag
    - Not always true with HTML (e.g., <br>)
  - Elements must follow a strict tree-like hierarchy with only one root element
  - XML is case sensitive

# XML

---

➤ Example: describing a library of CDs with XML

- A library contains many CDs
- Each CD might contain:
  - An ID
  - Title
  - Release year
  - Multiple tracks
- Each track contains:
  - Track number
  - Track title
  - Track length

# XML

---

## ➤ Example: describing a library of CDs with XML

Line 1: XML declaration

Line 2: root element called  
*cdlibrary*

Line 3: details of each CD  
in the library are  
in between the tags  
*<cd>* and *</cd>*

Line 8: details of each track  
are in between the  
tags *<track>* and  
*</track>*

```
<?xml version = "1.0" encoding= "UTF-8"?>
<cdlibrary>
  <cd>
    <cdid>A2 1325</cdid>
    <cdlabel>Atlantic</cdlabel>
    <cdtitle>Pyramid</cdtitle>
    <cdyear>1960</cdyear>
    <track>
      <trknum>1</trknum>
      <trkttitle>Vendome</trkttitle>
      <trklen>00:02:30</trklen>
    </track>
    ...
  </cd>
  ...
</cdlibrary>
```

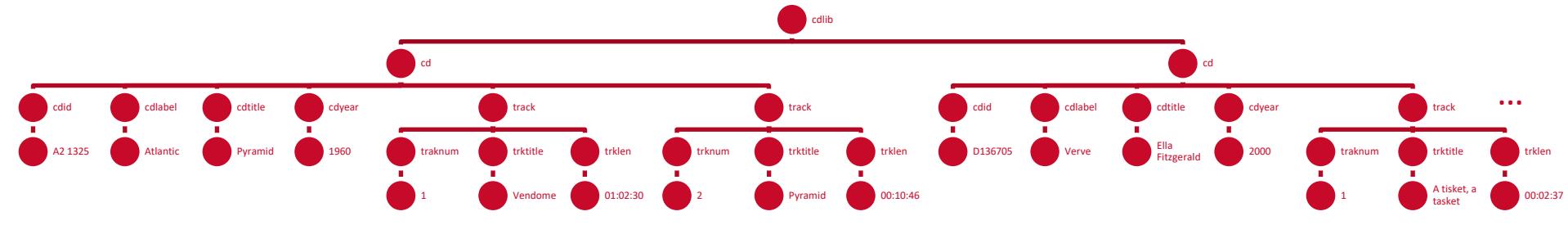
# XML

## ➤ Example: describing a library of CDs with XML

- Let's take a closer look at the previous example
  - Open the file *cdlib.xml* (available on Canvas) with VS Code
- This example clearly shows that XML files have a hierarchical structure

```
<?xml version="1.0" encoding="UTF-8"?>
- <cdlibrary>
  - <cd>
    <cdid>A2 1325</cdid>
    <cdlabel>Atlantic</cdlabel>
    <cdtitle>Pyramid</cdtitle>
    <cdyear>1960</cdyear>
    - <track>
      <trknum>1</trknum>
      <trkttitle>Vendome</trkttitle>
      <trklen>01:02:30</trklen>
    </track>
    - <track>
      <trknum>2</trknum>
      <trkttitle>Pyramid</trkttitle>
      <trklen>00:10:46</trklen>
    </track>
  </cd>
  - <cd>
    <cdid>D136705</cdid>
    <cdlabel>Verve</cdlabel>
    <cdtitle>Ella Fitzgerald</cdtitle>
    <cdyear>2000</cdyear>
    - <track>
      <trknum>1</trknum>
      <trkttitle>A tisket, a tasket</trkttitle>
      <trklen>00:02:37</trklen>
    </track>
    - <track>
      <trknum>2</trknum>
      <trkttitle>Vote for Mr. Rhythm</trkttitle>
      <trklen>00:02:25</trklen>
    </track>
    - <track>
      <trknum>3</trknum>
      <trkttitle>Betcha nickel</trkttitle>
      <trklen>00:02:52</trklen>
    </track>
  </cd>
</cdlibrary>
```

# XML



# XML

---

## ➤ Let's analyze XML files inside Python

- Background story

- You work for a company that predicts the attendance of public events
  - Event organizers pay you for that information, and use it for adjusting ticket prices
- A key predictor of event attendance is the weather
  - Your company decide to collect weather data from a reliable source and incorporate that in a predictive model
    - Open Weather: <https://openweathermap.org/>

# XML

---

- Example: current weather in Cincinnati
  - Let's request an XML file with data about the current weather condition in Cincinnati and build an XML tree
    - API documentation: <https://openweathermap.org/current>
    - Request: REST HTTP; Response: XML

```
import requests
import xml.etree.ElementTree as ET

url = "http://api.openweathermap.org/data/2.5/weather?q=Cincinnati&mode=xml&appid=YOUR_API_KEY"
response = requests.get(url)

with open('weather.xml', 'wb') as file:
    file.write(response.content)
```

# XML

---

- Example: current weather in Cincinnati

- Let's get the root node and all its children from the tree

```
tree = ET.parse('weather.xml')
root = tree.getroot()
```

```
<?xml version="1.0" encoding="UTF-8"?>
<current>
  <><city id="4508722" name="Cincinnati">
    <coord lon="-84.51" lat="39.1"/>
    <country>US</country>
    <timezone>-14400</timezone>
    <sun rise="2019-09-23T11:26:23" set="2019-09-23T23:34:36"/>
  </city>
  <temperature value="295.36" min="294.15" max="296.48" unit="kelvin"/>
  <humidity value="94" unit="%" />
  <pressure value="1016" unit="hPa"/>
  <><wind>
    <speed value="3.1" unit="m/s" name="Light breeze"/>
    <gusts/>
    <direction value="230" code="SW" name="Southwest"/>
  </wind>
  <clouds value="90" name="overcast clouds"/>
  <visibility value="16093"/>
  <precipitation value="0.25" mode="rain" unit="1h"/>
  <weather number="500" value="light rain" icon="10d"/>
  <lastupdate value="2019-09-23T14:57:14"/>
</current>
```

# XML

---

- Example: current weather in Cincinnati
    - Let's navigate through the XML file
      - Looking at the name and number of children of the root node
- print(root.tag)
- print(len(root))
- for child in root:
- print(child.tag)

```
<current>
  <city id="4508722" name="Cincinnati">
    <coord lon="-84.51" lat="39.1"/>
    <country>US</country>
    <timezone>14400</timezone>
    <sun rise="2019-09-23T11:26:23" set="2019-09-23T23:34:36"/>
  </city>
  <temperature value="295.36" min="294.15" max="296.48" unit="kelvin"/>
  <humidity value="94" unit="%" />
  <pressure value="1016" unit="hPa"/>
  <wind>
    <speed value="3.1" unit="m/s" name="Light breeze"/>
    <gusts/>
    <direction value="230" code="SW" name="Southwest"/>
  </wind>
  <clouds value="90" name="overcast clouds"/>
  <visibility value="16093"/>
  <precipitation value="0.25" mode="rain" unit="1h"/>
  <weather number="500" value="light rain" icon="10d"/>
  <lastupdate value="2019-09-23T14:57:14"/>
</current>
```

# XML

---

## ➤ Example: current weather in Cincinnati

- Let's navigate through the XML file
  - Obtaining the 1<sup>st</sup> child of the root node

city = root[0]

- Retrieving the time zone value

print(city[2].text)

```
▼<current>
  <city id="4508722" name="Cincinnati">
    <coord lon="-84.51" lat="39.1"/>
    <country>US</country>
    <timezone>14400</timezone>
    <sun rise="2019-09-23T11:26:23" set="2019-09-23T23:34:36"/>
  </city>
  <temperature value="295.36" min="294.15" max="296.48" unit="kelvin"/>
  <humidity value="94" unit="%" />
  <pressure value="1016" unit="hPa"/>
  ▼<wind>
    <speed value="3.1" unit="m/s" name="Light breeze"/>
    <gusts/>
    <direction value="230" code="SW" name="Southwest"/>
  </wind>
  <clouds value="90" name="overcast clouds"/>
  <visibility value="16093"/>
  <precipitation value="0.25" mode="rain" unit="1h"/>
  <weather number="500" value="light rain" icon="10d"/>
  <lastupdate value="2019-09-23T14:57:14"/>
</current>
```

# XML

---

- Example: current weather in Cincinnati

- Let's navigate through the XML file
  - Retrieving the value associated with the key "value" for the temperature node

```
temperature = root[1]
```

```
print(temperature.get("value"))
```

```
<current>
  <city id="4508722" name="Cincinnati">
    <coord lon="-84.51" lat="39.1"/>
    <country>US</country>
    <timezone>-14400</timezone>
    <sun rise="2019-09-23T11:26:23" set="2019-09-23T23:34:36"/>
  </city>
  <temperature value="295.36" min="294.15" max="296.48" unit="kelvin"/>
  <humidity value="94" unit="%"/>
  <pressure value="1016" unit="hPa"/>
  <wind>
    <speed value="3.1" unit="m/s" name="Light breeze"/>
    <gusts/>
    <direction value="230" code="SW" name="Southwest"/>
  </wind>
  <clouds value="90" name="overcast clouds"/>
  <visibility value="16093"/>
  <precipitation value="0.25" mode="rain" unit="1h"/>
  <weather number="500" value="light rain" icon="10d"/>
  <lastupdate value="2019-09-23T14:57:14"/>
</current>
```

# XML

---

- Note that sometimes it is difficult to navigate through JSON and XML files
  - One must know *ex-ante* the relevant fields and tree structure to navigate through the hierarchical trees
  - There are modules that help with this for very popular APIs, such as Twitter

# API

- Many APIs/web services offer free, but limited services
  - Frequent queries and more complex data often require payments
  - Example:  
<https://openweathermap.org/> (as of 07/19/2021)

Free	Startup 40 USD / month	Developer 180 USD / month	Professional 470 USD / month	Enterprise 2,000 USD / month
<a href="#">Get API key</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>
60 calls/minute <b>1,000,000 calls/month</b>	600 calls/minute 10,000,000 calls/month	3,000 calls/minute 100,000,000 calls/month	30,000 calls/minute 1,000,000,000 calls/month	200,000 calls/minute 5,000,000,000 calls/month
<a href="#">Current Weather</a>	<a href="#">Current Weather</a>	<a href="#">Current Weather</a>	<a href="#">Current Weather</a>	<a href="#">Current Weather</a>
<a href="#">Minute Forecast 1 hour*</a>	<a href="#">Minute Forecast 1 hour**</a>	<a href="#">Minute Forecast 1 hour</a>	<a href="#">Minute Forecast 1 hour</a>	<a href="#">Minute forecast 1 hour</a>
<a href="#">Hourly Forecast 2 days*</a>	<a href="#">Hourly Forecast 2 days**</a>	<a href="#">Hourly Forecast 4 days</a>	<a href="#">Hourly Forecast 4 days</a>	<a href="#">Hourly Forecast 4 days</a>
<a href="#">Daily Forecast 7 days*</a>	<a href="#">Daily Forecast 16 days</a>	<a href="#">Daily Forecast 16 days</a>	<a href="#">Daily Forecast 16 days</a>	<a href="#">Daily Forecast 16 days</a>
<a href="#">National Weather Alerts*</a>	<a href="#">National Weather Alerts**</a>	<a href="#">National Weather Alerts</a>	<a href="#">National Weather Alerts</a>	<a href="#">National Weather Alerts</a>
<a href="#">Historical weather 5 days*</a>	<a href="#">Historical weather 5 days**</a>	<a href="#">Historical weather 5 days</a>	<a href="#">Historical weather 5 days</a>	<a href="#">Historical weather 5 days</a>
<a href="#">Climatic Forecast 30 days</a>	<a href="#">Climatic Forecast 30 days</a>	<a href="#">Climatic Forecast 30 days</a>	<a href="#">Climatic Forecast 30 days</a>	<a href="#">Climatic Forecast 30 days</a>
<a href="#">Bulk Download</a>	<a href="#">Bulk Download</a>	<a href="#">Bulk Download</a>	<a href="#">Bulk Download</a>	<a href="#">Bulk Download</a>
<a href="#">Basic weather maps</a>	<a href="#">Basic weather maps</a>	<a href="#">Advanced weather maps</a>	<a href="#">Advanced weather maps</a>	<a href="#">Advanced weather maps</a>
<a href="#">Historical maps</a>	<a href="#">Historical maps</a>	<a href="#">Historical maps</a>	<a href="#">Historical maps</a>	<a href="#">Historical maps</a>
<a href="#">Global Precipitation Map</a>	<a href="#">Global Precipitation Map</a>	<a href="#">Global Precipitation Map</a>	<a href="#">Global Precipitation Map</a>	<a href="#">Global Precipitation Map</a>
<a href="#">Road Risk API</a>	<a href="#">Road Risk API</a>	<a href="#">Road Risk API</a>	<a href="#">Road Risk API</a>	<a href="#">Road Risk API</a>
<a href="#">Air Pollution API</a>	<a href="#">Air Pollution API</a>	<a href="#">Air Pollution API</a>	<a href="#">Air Pollution API</a>	<a href="#">Air Pollution API</a>
<a href="#">Geocoding API</a>	<a href="#">Geocoding API</a>	<a href="#">Geocoding API</a>	<a href="#">Geocoding API</a>	<a href="#">Geocoding API</a>
<a href="#">Weather widgets</a>	<a href="#">Weather widgets</a>	<a href="#">Weather widgets</a>	<a href="#">Weather widgets</a>	<a href="#">Weather widgets</a>
Uptime 95%	Uptime 95%	Uptime 99.5%	Uptime 99.5%	Uptime 99.9%

# API

---

- How do companies charge an API user?
  - How do they know the usage by a certain user?
    - Via mandatory API keys
    - Now, you know why I did not share my API keys with you ☺
- How to circumvent API limitations?
  - Request multiple free API keys
    - Ethical? Legal?
      - Unlikely for business applications
      - Likely for noncommercial applications (e.g., academic studies)
  - Put the “code to sleep” to avoid asking for data too often

## ➤ Business implications

- Selling access to databases is becoming an incredibly common revenue stream
  - Think about all the “free-to-use” social networks out there
- Two major steps an organization must follow:
  - Implement (code) the interface (API)
    - Tools: RESTify, Swagger, Python Flask, ...
  - Define access policies (price, constraints, etc.)
    - Tools: IBM API Management Service, ...

.....

## CASE STUDY: NEWS API

# News API

---

- Monitoring news media
  - What is the media saying about a certain person and/or organization?
  - Why/when can this be important?
    - Election?
    - Marketing campaigns?
    - Proxy to evaluate managerial actions?
    - Crisis?

# News API

---

- We will collect news articles using News API
  - Some of the API limitations (as of 07/19/2021)

Developer	Business	Enterprise
 \$0 totally free, start now	 \$449 per month, billed monthly	 \$849 per month, billed monthly
For all development and testing of personal or commercial projects.	For production and published commercial projects.	For enterprise projects that require exceptional resources and support.
Search all articles and get live top headlines <small>②</small>	Search all articles and get live top headlines <small>②</small>	Search all articles and get live top headlines <small>②</small>
New articles available with 1 hour delay <small>②</small>	New articles available in <b>real-time</b> <small>②</small>	New articles available in <b>real-time</b> <small>②</small>
Search articles up to a month old <small>②</small>	Search articles up to <b>3 years</b> old <small>②</small>	Search articles up to <b>3 years</b> old <small>②</small>
CORS enabled for localhost <small>②</small>	CORS enabled for <b>all origins</b> <small>②</small>	CORS enabled for <b>all origins</b> <small>②</small>
100 requests per day <small>②</small>	250,000 requests per month included	250,000 requests per month included
No extra requests available <small>②</small>	\$44.90 per 25,000 extra requests <small>②</small>	\$44.90 per 25,000 extra requests <small>②</small>
No uptime SLA	No uptime SLA	<b>99.95% uptime SLA</b> <small>②</small>
Basic support <small>②</small>	Email support <small>②</small>	Premium email and phone support <small>②</small>
<a href="#">Start with Developer</a>	<a href="#">Start with Business</a>	<a href="#">Start with Enterprise</a>

**Source:**  
<https://newsapi.org/pricing>

# News API

---

- News API returns data in JSON
  - Documentation: <https://newsapi.org/docs>
- Example: returning latest news about Bitcoin

```
import requests  
response = requests.get("https://newsapi.org/v2/everything?q=Bitcoin&apiKey= add-your_key_here")  
json_data = response.json()
```

# Homework #5

---

- News API allows one to collect the latest 100 online articles on any topic
  - But our query currently returns only 20 articles
  - Goal: write a script that collects *only* the ***description*** of the latest 100 articles about Bitcoin
    - The result after running your script should be a list of strings whose length is 100
    - Hint: read the API documentation and find a key that allows you to collect more than 20 articles <https://newsapi.org/docs/endpoints/everything>
  - Report your code on Canvas before the deadline

# Summary

---

- We learned more about APIs
  - Responses in XML
- We have not learned how to create APIs
  - Beyond the scope of this course
- Next lecture: collecting data from document-oriented databases

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 11 – Data Collection

*Querying Document-Oriented Databases*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Lecture Objectives

---

- Quick review of Assignment 2
- Learn about document-oriented databases
  - MongoDB
    - How a database can store data in JSON format

# Lecture Instructions (Part I)

---

- Download the notebook “*Lecture 11.ipynb*” available on Canvas
  
- Open the file “*Lecture 11.ipynb*” with VS Code
  
- Download the file *cars.csv*
  - Add this file to same folder as “*Lecture 11.ipynb*”

# Lecture Instructions (Part II)

---

- Let's create a MongoDB account on the cloud
  - A MongoDB database in the cloud
    - Go to <https://www.mongodb.com/cloud/atlas>
      - Click on "Get started now"
      - Fill in the fields
        - Memorize your password and username
      - Verify your email
      - Answer the survey questions
      - Select the Shared (free) option
        - Click on "Create"

# MongoDB

- You should see a screen like this ->
  - Select “AWS” as the cloud provider
  - Select “North America / N. Virginia” as region
  - Click on “Create Cluster”
- You will understand what we are doing here in the future

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our [documentation](#).

The screenshot shows the MongoDB Atlas 'Create Cluster' interface. At the top, there are three tabs: 'PREVIEW' (highlighted), 'Serverless', 'Dedicated', and 'FREE Shared' (highlighted with a green border). Below the tabs, a message states: 'For learning and exploring MongoDB in a sandbox environment. Basic configuration controls. No credit card required to start. Upgrade to dedicated clusters for full functionality. Explore with sample datasets. Limit of one free cluster per project.' A close button 'X' is in the top right corner.

**Cloud Provider & Region**

AWS, N. Virginia (us-east-1) (highlighted with a green border)

Cloud Providers: AWS (highlighted with a red border), Google Cloud, Azure

Regions:

- NORTH AMERICA**: Oregon (us-west-2) ★ (highlighted with a red border)
- EUROPE: Frankfurt (eu-central-1) ★
- ASIA: Mumbai (ap-south-1)
- ASIA**: Singapore (ap-southeast-1) ★
- AUSTRALIA**: Sydney (ap-southeast-2) ★

**Cluster Tier**: M0 Sandbox (Shared RAM, 512 MB Storage) > Encrypted

**Additional Settings**: MongoDB 4.4, No Backup >

**Cluster Name**: Cluster0 >

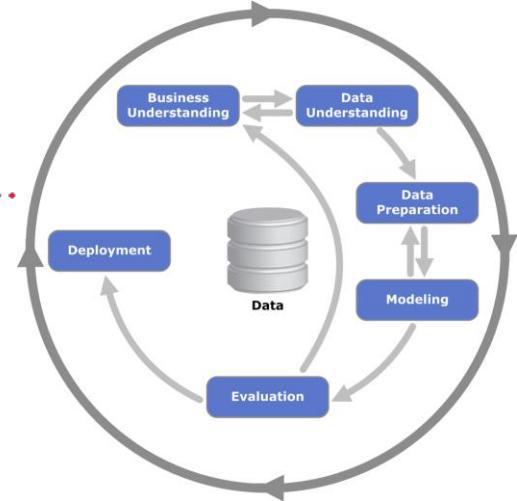
**FREE**: Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

**Create Cluster** (highlighted with a green border)

# Data Collection

## ➤ What have we learned up to know?

- Scrap (unstructured) data from the web
  - Retrieve relevant data from HTML files
    - Regular expressions
- Collect data via APIs
  - Request: REST + HTTP
  - Response: JSON, XML
- From previous courses: collect data by querying relational databases (SQL)



# API

---

➤ Bird's-eye perspective of APIs

- As we use in this course



1) user requests service (data) over  
the web (web service)

3) the server returns some data  
(usually in JSON or XML)



2) the API processes the  
request and usually  
accesses a database to  
retrieve the underlying  
data

# API

---

- When coding an API, an organization must define:
  - How to interpret a request (protocol)
  - How to gather data related to the request
  - How to return the collected data (format)
- If the underlying database is relational and the API response is in JSON, then the API must convert tables to the JSON format
  - Question: can we avoid this extra conversion step by having a database that already stores data using the JSON format, as opposed to tables?
    - Yes: document-oriented databases

# Document-Oriented Databases

---

- Recall that **relational databases** store tables that can be in relationships (see ISA 245)
  - One category of databases
- Today, we focus on **document-oriented databases**
  - Another category of databases
  - Store documents (texts)
  - Highly efficient when storing unstructured (textual) data

# Document-Oriented Databases

---

- There are several relational-database implementations
  - Oracle Express, Microsoft SQL Server, MySQL, ...
- Likewise, there are several document-oriented databases
  - MongoDB, CouchDB, Solr, ...
- We shall focus on MongoDB
  - A NoSQL database

# MongoDB

---

- Uses JSON-like flexible schemas (“designs”)
- Supports NoSQL-based queries
  - Often, a query looks for documents containing certain keys and values
- Powerful features
  - Replication, load balancing, quick data retrieval
- Cofounded by **Dwight Merriman**, Miami '89

# MongoDB

---

## ➤ Let's play with MongoDB

- When setting up a database, an organization/person has roughly two options
  - Install a database “locally” (on premise)
    - Requires powerful hardware and IT expertise (database administration)
  - Install a database on “someone else’s computer” (cloud)
    - Easy process, but one might not know where the data are
- We will follow the second option so as to have a first contact with cloud storage
  - We further explore this topic in the second part of this course
  - You will see how easy it is to have your own database in the cloud

# MongoDB

---

- We have already created a MongoDB account (Lecture Instructions)
- Each one of us now has one entire database at our disposal
  - We are now database administrators
  - Stored and running on computers owned by Amazon somewhere in the East Coast and managed by MongoDB (the company)
    - No need to deal with complex issues such as replication, load balance, *etc.*
  - Clearly, you could have installed MongoDB on your own machine
    - Choosing between cloud services (*e.g.*, running MongoDB on AWS) and in-house services (*e.g.*, running MongoDB on a company's computer) is a common question IT/IS managers often face

# MongoDB

- Next step: let's create a database user
  - Click on “Database Access” -> “Add New Database User”
  - Select a username and password
    - For the sake of illustration, my username is arthur my password is abcd1234
      - See next slide
    - Keep in mind that you have two usernames and passwords now
      - One is the administrator of the database system
      - The other is a user of the database

The screenshot shows the MongoDB Atlas web interface. At the top, it says "ARTHUR'S ORG - 2019-08-17 > PROJECT 0". On the left, there's a sidebar with "CONTEXT" set to "Project 0", "ATLAS" (selected), "Clusters" (highlighted with a red box), "Data Lake BETA", "SECURITY", "PROJECT", and "AI". Under "Clusters", it shows "Cluster0" (Version 4.0.12) with "CONNECT", "METRICS", "COLLECTIONS", and "...". Below the cluster list, it shows "CLUSTER TIER M0 Sandbox (General)", "REGION AWS / N. Virginia (us-east-1)", and "TYPE Replica Set - 3 nodes". A modal window titled "Connect to Atlas" is open, with a progress bar at 20% and a checklist:

- Build your first cluster (checkmark)
- Create your first database user (radio button highlighted with a red box)
- Whitelist your IP address
- Load Sample Data (Optional)
- Connect to your cluster

A "No thanks" button is at the bottom right of the modal.

# MongoDB

- Creating a database user called *arthur*

## Authentication Method

Password

Certificate

AWS IAM  
(MongoDB 4.4 and up)

MongoDB uses **SCRAM** as its default authentication method.

### Password Authentication

arthur

abcd1234

HIDE

Autogenerate Secure Password

Copy

### Database User Privileges

Select a [built-in role or privileges](#) for this user.

Read and write to any database



### Restrict Access to Specific Clusters/Data Lakes

Enable to specify the resources this user can access. By default, all resources in this project are accessible.



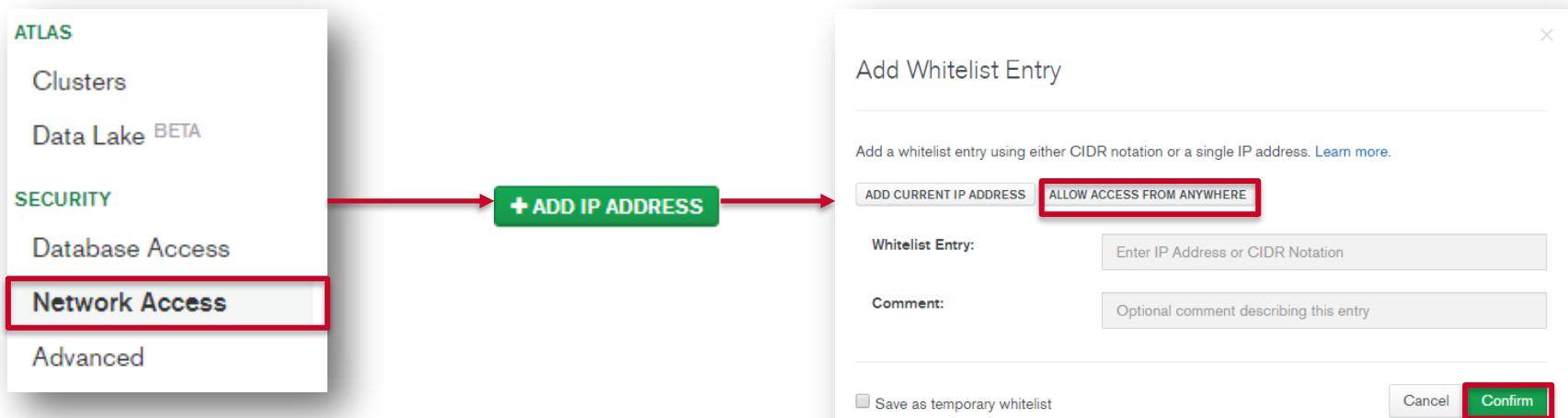
### Temporary User

This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week.



# MongoDB

- As a security feature, MongoDB (company) requires the database administrator to specify which IP addresses can access the database
  - Click on “Network Access” -> “Add IP Address” -> “ALLOW ACCESS FROM ANYWHERE” -> “Confirm”
  - This make take a few seconds



# MongoDB

---

- We now have a database in the cloud and a database user
  - Let's access and play with the created database from Python
  - Install the required packages (pymongo)
    - Go to the Terminal and type `pip install 'pymongo[snappy,gssapi,srv,tls]'`
  - We shall use the cars data set  
`import pandas as pd`

```
pd.read_csv("cars.csv")
```

# MongoDB

- Let's create a connection to our MongoDB database
    - On [cloud.mongodb.com](https://cloud.mongodb.com), go to *Clusters* → *Connect* → “*Connect your application*”
    - Select *Driver* = “*Python*” and *Version* = “*3.4 or later*”

**DEPLOYMENT**

- Databases
- Triggers
- Data Lake

**SECURITY**

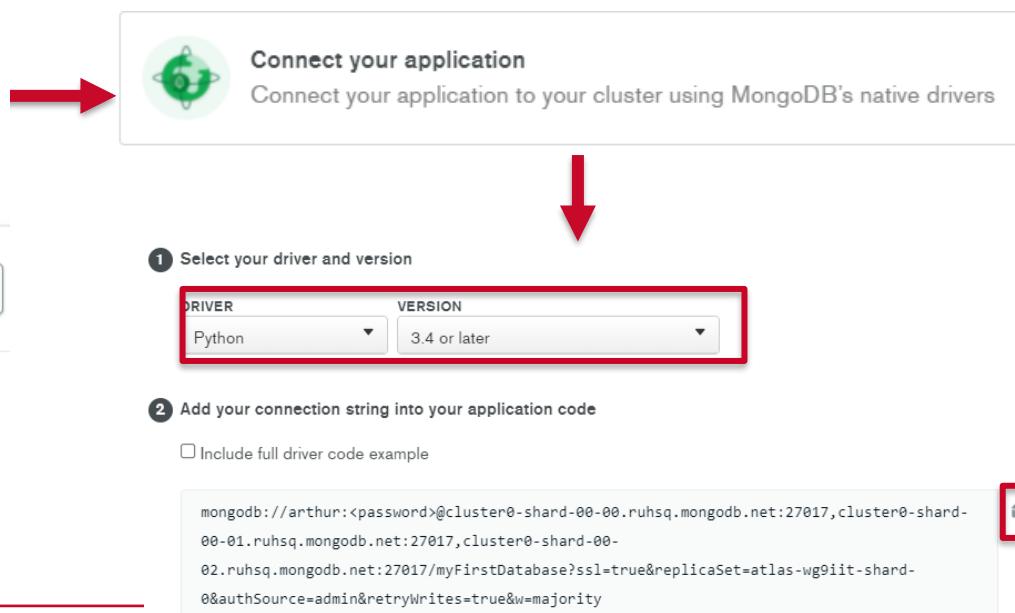
- Database Access
- Network Access

TEST > PROJECT 0

# Database Deployments

Find a database deployment... 

● Cluster0  View Monitoring



# MongoDB

---

- Let's create a connection to our MongoDB database
  - Now, paste that URL into your code

```
import pymongo  
client = pymongo.MongoClient("PASTE THE COPIED STRING HERE")
```
  - Remember to replace <password> with the database password you created before
    - For example, my password is *abcd1234*
    - Remember to remove the <> symbols

```
abcd1234  
mongodb://arthur:[REDACTED]@cluster0-shard-00-00.ruhsq.mongodb.net:27017,cluster0-shard-00-01.ruhsq.mongodb.net:27017,cluster0-shard-00-02.ruhsq.mongodb.net:27017/myFirstDatabase?ssl=true&replicaSet=atlas-wg9it-shard-0&authSource=admin&retryWrites=true&w=majority
```

# MongoDB

---

- Let's create a database called *ISA414*

`db = client.ISA414`

- Within that database, let's create a collection *cars*

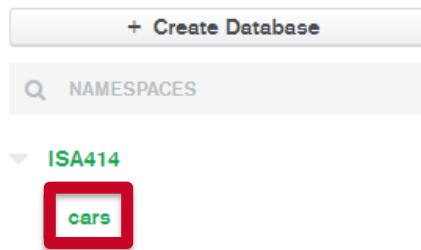
`collection = db.cars`

# MongoDB

---

- Let's send our cars data (`df`) to our database
  - Recall that `df` is a data frame
  - Let's transform it into a dictionary (JSON)  
`data_json = df.to_dict("records")`
  - Inserting the data into the collection  
`collection.insert_many(data_json)`

- Now, go to *Databases* -> “*Browse Collections*”



Yay! We were able to send the “cars” data set from Python to our MongoDB database in the cloud

# MongoDB

---

- Let's take a look at our database

- Click on the **collection** named “cars”
- Note that:
  - Each observation in our data frame becomes a **document**
  - The database is a **collection of documents**

```
_id: ObjectId("60f09819adfb83b8cf2c684b")
name: "Mazda RX4"
mpg: 21
cyl: 6
disp: 160
hp: 110
drat: 3.9
wt: 2.62
qsec: 16.46
vs: 0
am: 1
gear: 4
carb: 4
```

```
_id: ObjectId("60f09819adfb83b8cf2c684c")
name: "Mazda RX4 Wag"
mpg: 21
cyl: 6
disp: 160
hp: 110
drat: 3.9
wt: 2.875
qsec: 17.02
vs: 0
am: 1
gear: 4
carb: 4
```

# MongoDB

- Let's take a look at our database

name	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4

- Each document has an “id”
  - In our case, it was automatically created
- Each table’s attribute-value becomes a key-value pair

```
{  
  "_id": ObjectId(  
    "5d574bd121590000470023e2"),  
  "name": "Mazda RX4",  
  "mpg": 21,  
  "cyl": 6,  
  "disp": 160,  
  "hp": 110,  
  "drat": 3.9,  
  "wt": 2.62,  
  "qsec": 16.46,  
  "vs": 0,  
  "am": 1,  
  "gear": 4,  
  "carb": 4  
}
```

# MongoDB

---

- We are now in a position to understand important facts about MongoDB
  - Based on the concept of *key-value pairs*, crucial NoSQL database concept
  - Document-oriented database
    - Document is often represented as a JSON file
  - Important distinction
    - Relational databases store data in separate *tables* that are defined *ex ante* by the database designer
    - Document-oriented databases store all the data in documents, and one document might have different fields (no need to be predefined)

# MongoDB

---

- Example: relational vs document-oriented database
  - Skip is a Full Professor of ISA
  - Arthur is an Assistant Professor of ISA

ID	Position	Name	Department
1	Full Professor	Skip	ISA
2	Assistant Professor	Arthur	ISA

```
{  
  "ID": 1,  
  "Position": "Full Professor",  
  "Name": "Skip",  
  "Department": "ISA"  
}  
  
{  
  "ID": 2,  
  "Position": "Assistant Professor",  
  "Name": "Arthur",  
  "Department": "ISA"  
}
```

# MongoDB

---

- Example: relational vs document-oriented database
  - What if we want to add a second “position” to a table
    - *E.g.*, Skip is also the ISA chair
  - Relational database: redesign the original table
    - Side effect: extra missing values

ID	Position	Position 2	Name	Department
1	Full Professor	Chair	Skip	ISA
2	Assistant Professor		Arthur	ISA

# MongoDB

---

- Example: relational vs document-oriented database
  - Missing values often consume storage space in relational databases
  - For example, suppose the tables are stored as CSV files
    - The addition of a new column to a table means that we need one extra comma per row (extra storage cost = 1 byte)  
*2, Assistant Professor,,Arthur,ISA*
  - Suppose the underlying table has 1 billion rows
    - Total storage space required by commas: 1 billion bytes = 1 GB
    - That is, adding a new column with no data to a table might drastically increase the required storage space
      - Not much flexibility here

# MongoDB

---

- Example: relational database vs MongoDB
  - What if we want to add a second “position” to a document?
    - *E.g.*, Skip is also the ISA chair
  - MongoDB: simply add a new key-value pair to the first document

```
{  
  "ID": 1,  
  "Position": "Full Professor",  
  "Position 2": "Chair",  
  "Name": "Skip",  
  "Department": "ISA"  
}  
{  
  "ID": 2,  
  "Position": "Assistant Professor",  
  "Name": "Arthur",  
  "Department": "ISA"  
}
```

# MongoDB

---

- Illustrating the previous point
  - Let's add a key-value pair to the first car

The screenshot shows a MongoDB document in a graphical interface. The document contains the following fields and their values:

- `_id: ObjectId("5d574bd121590000470023e2")`
- `mpg: 21`
- `cyl: 6`
- `disp: 160`
- `hp: 110`
- `drat: 3.9`
- `wt: 2.62`
- `qsec: 16.46`
- `vs: 0`
- `am: 1`
- `gear: 4`
- `carb: 4`
- `_row: "Mazda RX4"`

On the right side of the document, there is a toolbar with four icons: a pencil (highlighted with a red box), a copy icon, a paste icon, and a delete icon.

# MongoDB

## ➤ Illustrating the previous point

The image shows a screenshot of a MongoDB document editor interface. On the left, a document is displayed with the following fields:

```
1 _id: ObjectId("60f09819adfb83b8cf2c684b")
2 name : "Mazda RX4 //"
3 mpg : 21
4 cyl : 6
5 disp : 160
6 hp : 110
7 drat : 3.9
8 wt : 2.62
9 qsec : 16.46
10 vs : 0
11 am : 1
12 gear : 4
13 carb : 4
```

A red box highlights the "Add Field After carb" button at the bottom of the list.

A red arrow points from the left screen to the right screen, indicating a transition or comparison.

On the right, the document has been modified to include a new field:

```
1 _id: ObjectId("60f09819adfb83b8cf2c684b")
2 name : "Mazda RX4 //"
3 mpg : 21
4 cyl : 6
5 disp : 160
6 hp : 110
7 drat : 3.9
8 wt : 2.62
9 qsec : 16.46
10 vs : 0
11 am : 1
12 gear : 4
13 carb : 4
14 color : "blue //"
```

The new field "color" is highlighted with a red box. To the right of the document, a vertical stack of boxes shows the data types for each field: ObjectId, String, Double, Int32, Double, Int32, Double, Double, Double, Int32, Int32, Int32, Int32, and String.

At the bottom of the right screen, there are "CANCEL" and "UPDATE" buttons.

Document Modified.

# MongoDB

---

- The previously-discussed flexibility makes document-oriented databases very attractive to store textual/unstructured data
  - No need to design a database
  - For example: one can have Facebook posts and tweets inside the same collection of documents
    - Tweet-like documents can have fields such as the number of likes, shares, retweets, *etc.*
    - Facebook-like documents can have fields such as the number of likes, smiley faces, sad faces, hearts, replies, shares, *etc.*

# MongoDB

---

## ➤ Querying

- A database is rather useless if one cannot easily retrieve data from it
- How to query document-oriented databases?
  - SQL cannot really be used (NoSQL)
- Idea: search for key-value pairs
  - Our query is actually a JSON “file”
  - Result: JSON “file”

# MongoDB

---

## ➤ Querying examples

- Finding all the cars (documents) containing the key (variable) *mpg* associated with the value 21

```
result = collection.find({"mpg":21})    {  
    for document in result:                "mpg":21  
        print(document)                  }  
    }
```

- Finding all the cars (documents) containing the *key:value* pairs *mpg:21* and *color:blue*

```
result = collection.find({"mpg":21, "color":"blue"}) {  
    for document in result:  
        print(document)  
    }
```

# MongoDB

- MongoDB has several operators that can be used in queries
  - Finding all the cars (documents) where the variable *mpg* (key) is greater than (\$gt) 20 (value)
    - The value associated with *mpg* is "\$gt":20

```
result = collection.find({"mpg": {"$gt": 20}})  
for document in result:  
    print(document)
```

```
{  
    "mpg": {  
        "$gt": 20  
    }  
}
```

- Finding all the cars (documents) where the variable *mpg* (key) is greater than (\$gt) 20, but less than (\$lt) 25

```
result = collection.find({"mpg": {"$gt": 20, "$lt": 25}})  
for document in result:  
    print(document)
```

```
{  
    "mpg": {  
        "$gt": 20,  
        "$lt": 25  
    }  
}
```

# MongoDB

---

- One can also perform update and delete operations with MongoDB
  - Examples:
    - Updating the mpg to 100 for cars named Mazda RX4

```
myquery = {"name": "Mazda RX4"}  
newvalues = {"$set": {"mpg": 100}}  
  
collection.update_many(myquery, newvalues)
```

# MongoDB

---

- One can also perform update and delete operations with MongoDB

- Examples:

- Deleting all 6-cylinder cars

```
collection.delete_many({"cyl":6})
```

- Deleting all documents

```
collection.delete_many({})
```

# MongoDB

---

- We have barely scratched the surface of what can be done with MongoDB and document-oriented databases
  - When should one use MongoDB?
    - Textual data (documents)
    - No predefined database model
    - When the database is used primarily to support APIs
  - MongoDB vs Relational Databases
    - Detailed technical analysis  
<https://www.mongodb.com/compare/mongodb-mysql>
- “Using MongoDB removes the complex object-relational mapping (ORM) layer that translates objects in code to relational tables. MongoDB’s flexible data model also means that your database schema can evolve with business requirements.”*

# Homework #6

---

- Suppose you work for FAA - Federal Aviation Administration
  - Job: database administrator
  - Current tasks:
    1. Migrate FAA's table-based database to a document-oriented database
    2. Provide data to FAA decision makers

# Homework #6

---

## ➤ Preliminaries

- Task 1: load the original data
  - *flights.csv*: available on Canvas (Assignment -> Homework 6)
- Task 2: create a new collection called *flights* inside the *ISA414* database
- Task 3: populate the flights collection

# Homework #6

---

- Queries: submit the solutions on Canvas by the end of tomorrow
1. Add the new key-value pair "*delayed*":"*true*" to the documents containing delayed flights (*dep\_delay > 0*)
    - Hint #1: this is an update statement
    - Hint #2: the “query” part concerns finding the documents where *dep\_delay > 0*
    - Hint #3: the “update” part is where you set "*delayed*":"*true*"
  2. Return all the data about delayed flights (i.e., *delayed:true* )
  3. Return only the carrier names for the delayed flights (i.e., *delayed:true* )
    - Hint: look at the section “*Specify the Fields to Return*” in the document at <https://docs.mongodb.com/manual/reference/method/db.collection.find/>

# Summary

---

- We learned about document-oriented databases (MongoDB)
- References
  - List of query commands
    - <https://docs.mongodb.com/manual/reference/operator/query/>
  - List of aggregate commands
    - <https://docs.mongodb.com/manual/reference/operator/aggregation/>
  - List of update commands
    - <https://docs.mongodb.com/manual/reference/operator/update/>
- Next lecture: supervised learning

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 12 – Data Analysis

*Supervised Learning (Part I)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Announcements

---

## ➤ Assignment 3

- Now available on Canvas
- Deadline: Wednesday, 10/06, before 11:59 pm

# Lecture Objectives

---

- Quick review of Homework 5 and 6
- Learn about predictive analytics
  - Classification and regression problems
    - Decision trees
    - Random forests
  - How to evaluate models
    - Training and test sets

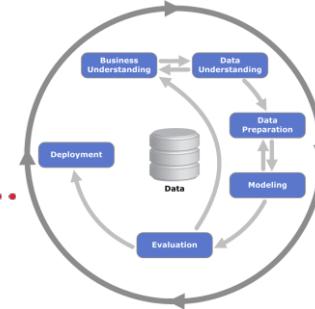
# Lecture Instructions

---

- Download the following files available on Canvas
  - *Lecture 12.ipynb*
  - *energy\_data.csv*
  - *Description of the variables.xlsx*
- Place the above files inside the same folder
- Open the file *Lecture 12.ipynb* with VS Code

# CRISP-DM

---



- Business Understanding phase
  - Important question: can a certain business problem be formulated as a data-analytics problem?
- Different types of problems/answers
  - Descriptive analytics: provides answers on “*what happened*”
    - Descriptive statistics, SQL queries, OLAP (business intelligence), ...
  - Prescriptive analytics: provide answers on “*what could have happened*”
    - Optimization, simulation, ...
  - Predictive analytics: provide answers on “*what will happen*”
    - Focus of this lecture

# Business Understanding

---

- The solutions to the category of problems we will be working on are based on a paradigm called supervised learning
  - Assumption: there is a clear target variable whose values we are trying to predict

Problem	Supervised	Unsupervised
Classification	X	
Regression	X	
Clustering		X
Co-occurrence grouping		X
Profiling		X
...	...	...

Our focus today →

Next class →

# Supervised Learning

---

## ➤ Classification (or class probability estimation)

- Goal: determine which class a new observation (likely) belongs to
  - The target variable is qualitative (categorical/factor)
- Example (Assignment 3): given a data set containing data about bank clients, can we build a statistical model that classifies future clients as “good” or “bad” in terms of paying back a loan?
  - Clear categorical target (two values: “good” and “bad”)

# Supervised Learning

---

## ➤ Regression problems

- Goal: predict the value of a target variable using a model structure that relates the target with informative attributes
  - The target variable is quantitative (numeric)
- Example: what is the market price of a house in Oxford with 4 bedrooms, 3 bathrooms, built in 2016, ...
  - Clear numeric target (price)

# Supervised Learning

---

## ➤ Illustrative problem

- An energy (electricity) provider has historical data with information about current clients
  - *E.g.*, Age, salary, marital status, etc.
- Each client (observation) in the data set is associated with a category
  - Electricity tariff (*e.g.*, time-of-use tariff, peak tariff, flat tariff)
- Business question:
  - Which electricity tariff should this company suggest to a new client?
    - Is this a classification or a regression problem?

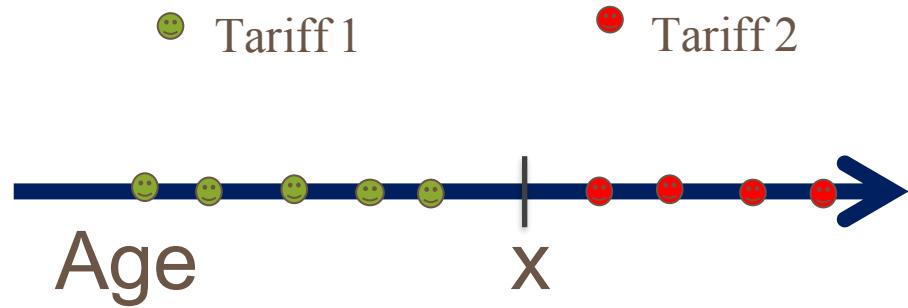
# Classification

---

- Potential solution:
  - Translate this problem into a classification problem
  - Collect the data
    - *E.g.*, query internal databases
  - Build a *classifier*
    - A machine learning model that classifies clients based on tariffs
  - Classify the new client
    - Using the classifier to estimate which category the new client likely belongs to

# Classification

- Consider the following hypothetical situation:
  - All clients younger than  $x$  in your data set prefer Tariff 1
  - All clients older than  $x$  in your data set prefer Tariff 2
- What would be a good classifier?
  - Which tariff would you suggest to a new client?

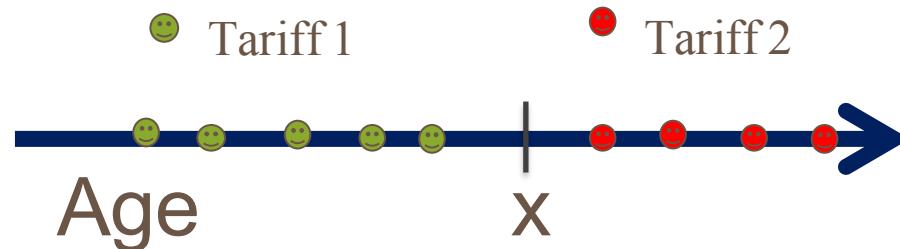


# Classification

---

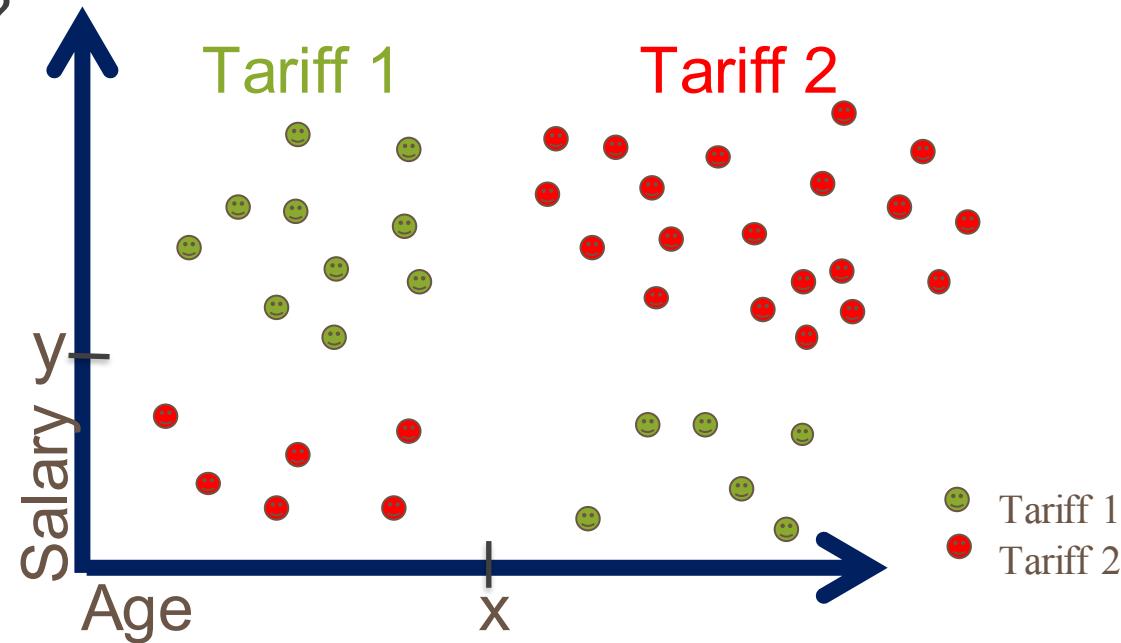
## ➤ Classifier:

- New client's age >  $x$ 
  - Suggest Tariff 2
- Otherwise
  - Suggest Tariff 1



# Classification

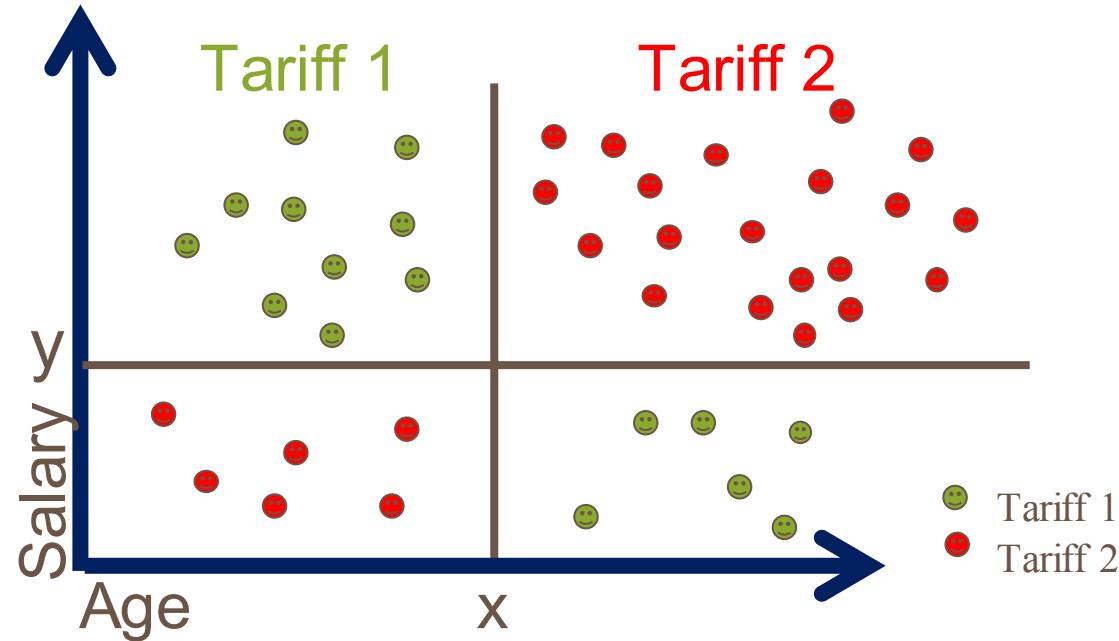
- What about this more realistic case?
  - Two variables



# Classification

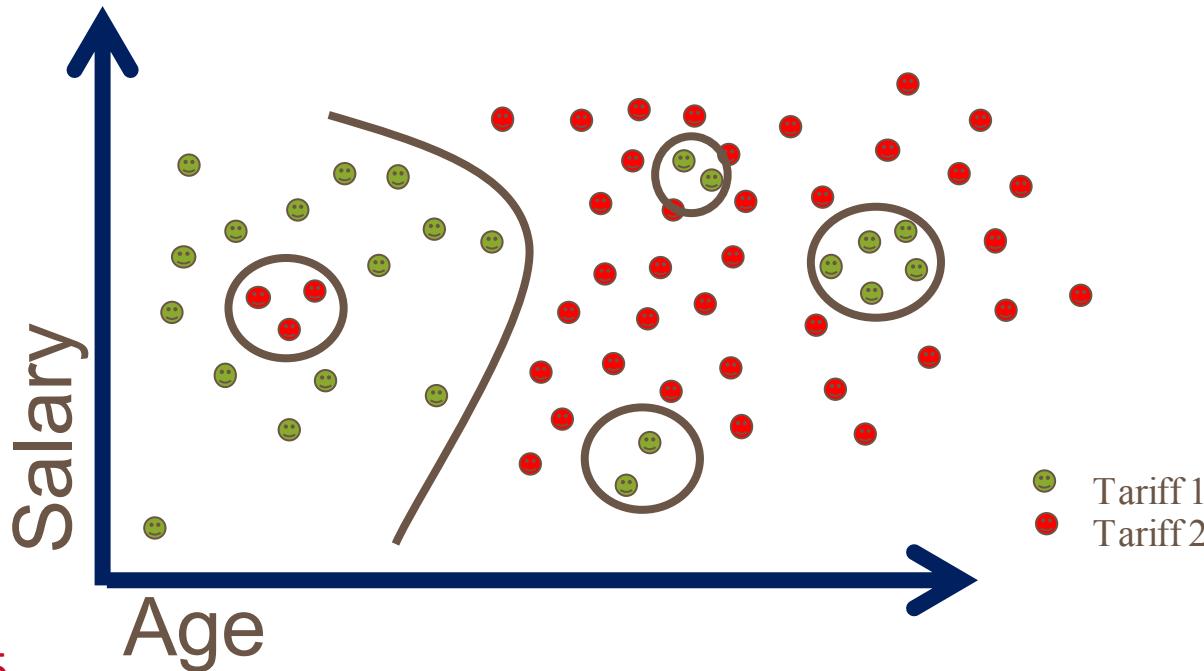
## Classifier:

- New client's age > x AND new client's salary > y
  - Suggest Tariff 2
- New client's age > x AND new client's salary < y
  - Suggest Tariff 1
- New client's age < x AND new client's salary > y
  - Suggest Tariff 1
- New client's age < x AND new client's salary < y
  - Suggest Tariff 2



# Classification

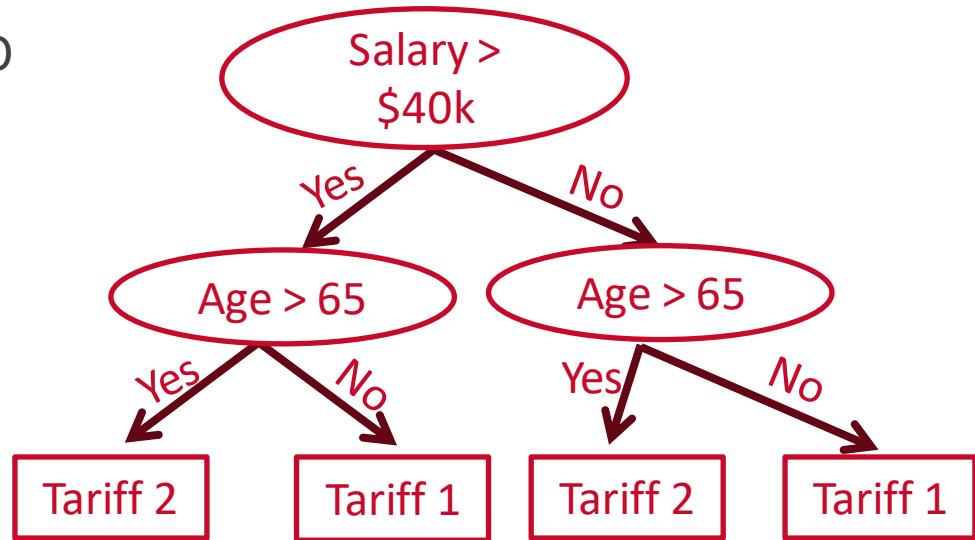
- Keep in mind that real-life is a mess



# Decision Trees

- The previous approach of segmenting the attribute space is precisely what a technique called *Decision Trees* does

- Create “if-then” rules
  - E.g., IF ‘Salary > \$40K’ AND ‘Age > 65’  
THEN suggest Tariff 2



# Decision Trees

---

- Formally, decision trees create hyperrectangles
- Categories are the “leaves”
  - Bottom of the tree
- There are many algorithms for building decision trees from a data set
  - Beyond the scope of this course
  - General idea: choose a variable at each level that best splits the data set
    - *E.g.*, reduce entropy (information gain)

# Decision Trees

---

- Henceforth, we shall heavily use the `sklearn` and `pandas` modules in our modeling efforts
  - Note that `sklearn` has several submodules that serve different purposes
    - We shall import them as we progress
  - Install the required modules

```
pip install sklearn  
pip install pandas
```
  - Load pandas and the data set we use today

```
import pandas as pd  
energy_data = pd.read_csv("energy_data.csv")
```

# Decision Trees

---

- Before analyzing the data, always check whether each attribute is of the right type
  - If it is not, then change attribute types
  - Note that pandas store strings as objects  
`energy_data.dtypes`
- **sklearn** does not allow for non-numeric variables when building models
  - Its `preprocessing` submodule has several functions to transform categorical into continuous variables
    - Example: `OrdinalEncoder()`, `OneHotEncoder()`, `LabelEncoder()`

# Decision Trees

---

- Oftentimes (not always) one should encode categorical variables as **dummies**
  - Except for the **target** variable, which should have its values replaced by numbers
- Let's derive dummies for the predictors
  - **pandas** offer a simpler way to derive dummies than **sklearn**

```
energy_data = pd.get_dummies(energy_data,  
                             columns = ["MaritalStatus", "IncomeLevel", "DwellingArea",  
                                         "HasChildren", "SolarRoof", "ShiftableLoad",  
                                         "AttitudeSustainability" ],  
                             drop_first = True)
```

# Decision Trees

---

- Let's derive recode the target variable
  - (Sub)module preprocessing in `sklearn`

```
from sklearn import preprocessing
```

```
enc = preprocessing.LabelEncoder()
```

```
energy_data["Tariff"] = enc.fit_transform(energy_data["Tariff"])
```

# Decision Trees

---

- Building the tree model
  - One must create two sets of columns
    - The feature (independent) variables

```
x = energy_data.drop(columns=["Tariff"])
```
    - The target (dependent) variable

```
y = energy_data["Tariff"]
```
  - Next, it is time to create and fit a model

```
model = DecisionTreeClassifier()  
model = model.fit(x,y)
```

# Model Evaluation

---

- How do we know our model is any good?
  - One possible way: split the original data set into two parts
    - Train the model using one data set (training set)
    - Test the model using the complementary data set (test set)
      - Use the trained model to predict the target values in the test set, and compare the predictions against the true values
      - The higher the number of times the predictions agree with the true values, the more accurate the classifier is

# Model Evaluation

---

- Analogy: exam
  - A professor gives you a study guide
    - Set of problems with answers
  - What if the professor asks you the same questions in the exam?
    - The professor is not really testing your knowledge
    - The professor is testing whether you are capable of memorizing answers
  - What if the professor asks you slightly different questions about the same material in the exam?
    - The professor is now testing your knowledge (generalization power)
  - Study guide = training; Exam = testing

# Model Evaluation

---

- Let's redo what we did before and evaluate our model
  - Randomly split the original data into two data frames
    - **Training set** (66% of the observations)
    - **Test set** (34% of the observations)
      - The 66/34 division is just one common way of doing it
  - Train the model using the training set
  - Evaluate the model using the test set

# Model Evaluation

---

- Let's use the submodule `model_selection` in `sklearn` to split a data frame

```
from sklearn.model_selection import train_test_split  
  
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.34)
```

- Building a decision tree using the training set

```
model = DecisionTreeClassifier()  
  
model = model.fit(x_train,y_train)
```

# Model Evaluation

---

- Evaluating a decision tree using the test set
  - The `metrics` submodule contains several metrics to evaluate statistical models
  - We shall use the overall accuracy metric
    - Percentage of correctly classified instances
  - Step 1: use the model to predict the class of each observation in the test set  
`y_pred = model.predict(x_test)`
  - Step 2: calculate how often the predictions in the model agree with the true class in the test set  
`metrics.accuracy_score(y_test, y_pred)`
  - An accuracy of, say, 0.60 means that the model is expected to correctly classify 60% of future instances

# Decision Trees

---

- Strengths
  - Simple to understand and interpret
  - Performs reasonably well for big data sets
- Drawbacks
  - Learning the optimal tree is not always computationally feasible
  - Might result in a high bias towards the training set
    - Tendency to **overfit**

# Classification

---

- There are many models for classification
  - SVM, random forests, GBM, logistic regression...
  - Which one is the best?
    - In theory, all algorithms are equally good in expectation !!!
      - No Free-Lunch Theorem
    - Common approach when predictive accuracy is the only important factor
      - Build and evaluate multiple classifiers
      - Perform statistical analysis on the obtained results to determine the most accurate model

# Classification

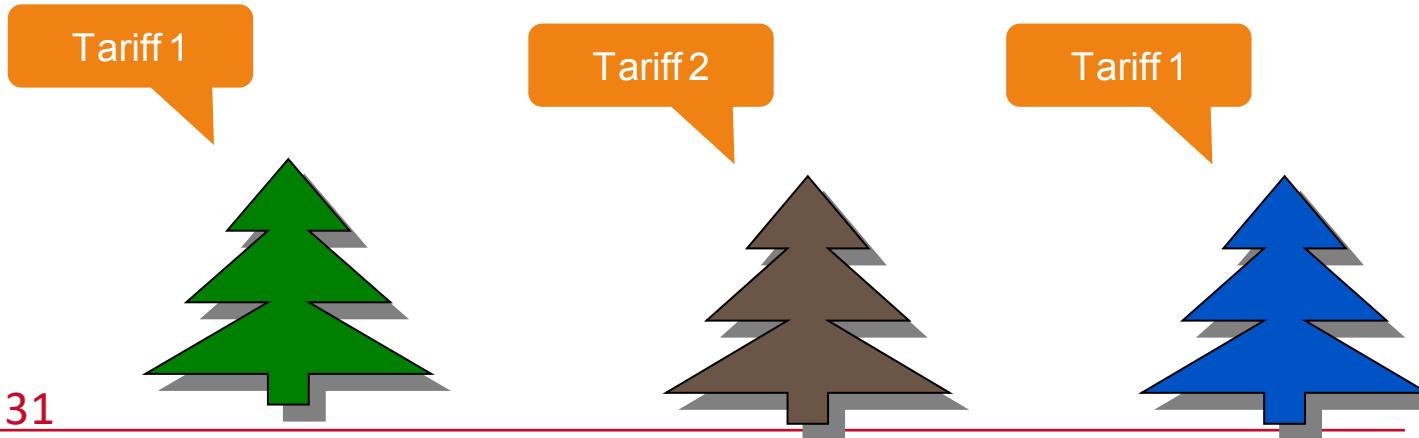
---

- Some models perform well for certain problems, and poorly for other problems
- Another common approach: combine several models
  - Ensemble learning
  - Compensate poor individual performance
    - (Almost) free lunch
  - Diversity matters
    - Formally, one wants the errors produced by the individual models to be as uncorrelated as possible

# Random Forests

---

- Ensemble model
- Idea: build several decision trees semi-randomly
  - Each tree individually classifies a new observation
  - The most popular predicted category is chosen as the outcome of the model



# Random Forests

---

- Building a random forest (200 trees) using the training set
  - Submodule `sklearn.ensemble`

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=200)
model = model.fit(x_train,y_train)
```
- Evaluating a random forest model using the test set

```
y_pred = model.predict(x_test)
metrics.accuracy_score(y_test,y_pred)
```

# Random Forests

---

- One of the most popular models in forecasting competitions (alongside GBM and Neural Networks)
  - Knowledge Discovery in Databases (KDD)
  - Kaggle.com
- Strengths
  - Tackles the bias problem with single decision trees
- Drawbacks
  - No longer easy to interpret and explain the results

# Classification

---

- This lecture summarized what is often taught across many data mining classes
- Keep in mind that:
  - There are many different statistical models for different types of problems
  - There are many different evaluation metrics other than using the percentage of correctly classified observations
    - *E.g.*, specificity, sensitivity, ROC area
  - There are many different ways of estimating model errors
    - K-fold cross validation, nested cross validation

# Summary

---

- Summary
  - Data-analytics problems: classification problem
  - Decision trees and random forests
  - Evaluation: training and test sets
- Useful references
  - <http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>
  - <https://docs.google.com/presentation/d/1kSuQyW5DTnkVaZEjGYCkfOxzCqGEFzWBy4e9Uedd9k/edit>
- Next class: regression problems

# ISA 414 – Managing Big Data

## Lecture 13 – Data Analysis

*Supervised Learning (Part II)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Announcements

---

- **Information Systems & Analytics Seminar Series**
  - **Applying Machine Learning & Artificial Intelligence to Ecommerce/Digital Business Operations across Marketing, Merchandising, and Customer Support**
    - Tuesday, October 12, 2021, 7:00 pm
    - <https://miamioh.zoom.us/j/81146372341?pwd=SzJldk1PTWdhVDhBcXZOSTZFVVV3UT09>
  - **By Matt Fritz**
    - Senior Director, Machine Learning & Artificial Intelligence @ Samsung Electronics

# Announcements

---

- Python cheat sheet available on Canvas
- Assignment 3
  - I messed up with the server and connection string last Thursday
  - Please redownload the assignment description

# Lecture Objectives

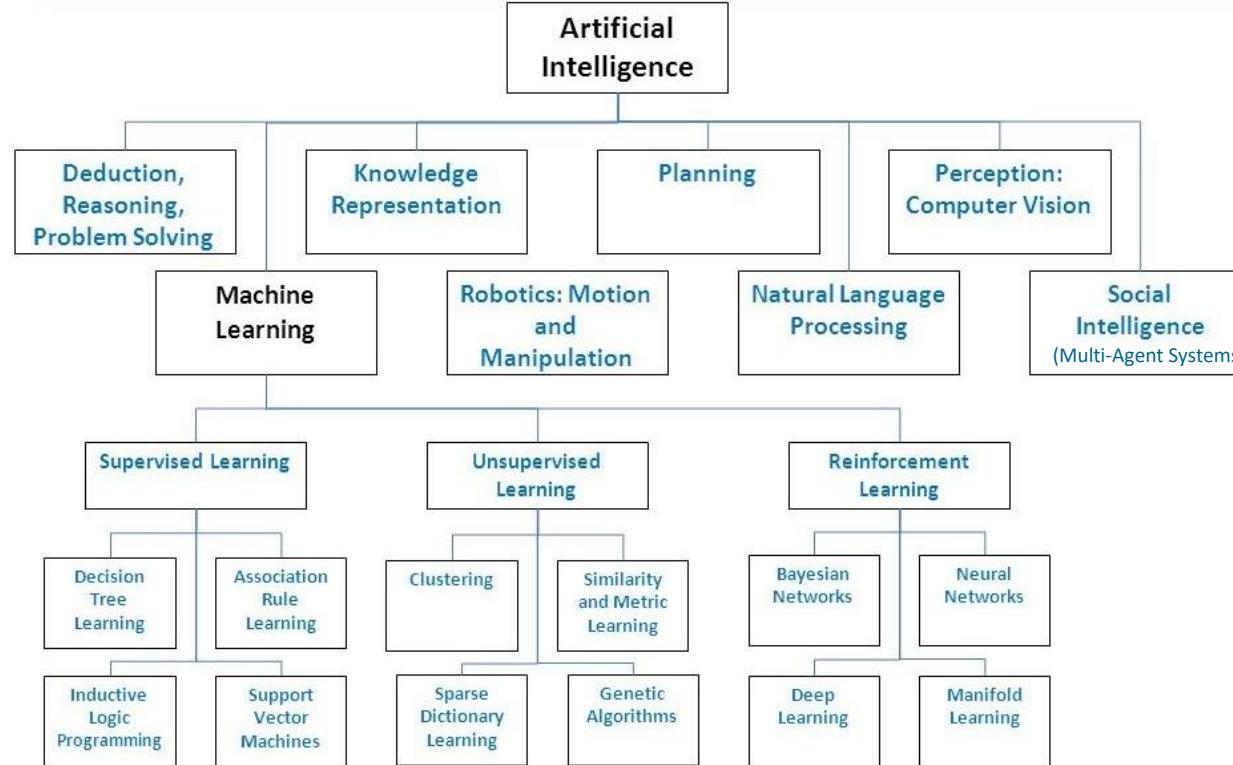
---

- Learn about predictive models for regression problems

# Lecture Instructions

---

- Download the following files available on Canvas
  - *Lecture 13.ipynb*
  - *energy\_data.csv*
  - *Description of the variables.xlsx*
- Place the above files inside the same folder
- Open the file *Lecture 13.ipynb* with VS Code



# Machine Learning

---

- The solutions to the category of problems we will be working on are based on a paradigm called supervised learning
  - Assumption: there is a clear target variable whose values we are trying to predict

	Problem	Supervised	Unsupervised
Previous class →	Classification	X	
Our focus today →	Regression	X	
	Clustering		X
	Co-occurrence grouping		X
	Profiling		X
	...	...	...

# Supervised Learning

---

- Classification (or class probability estimation)
  - Goal: determine which class a new observation (likely) belongs to
    - The target variable is qualitative (categorical/factor)
  - Example: given a data set containing data about previous and current clients, can we build a statistical model to suggest an electricity tariff to a new client?
    - Categorical target

# Supervised Learning

---

## ➤ Regression problems

- Goal: predict the value of a target variable using a model structure that relates the target with informative attributes
  - The target variable is quantitative (numeric)
- Example: what will be the annual electricity consumption for a 4-bedroom house in Oxford having 2 adults and 2 kids, an electric vehicle, ...?
  - Clear numeric target (consumption)

# Supervised Learning

---

- Illustrative problem
  - An energy (electricity) provider has historical data with information about current clients
    - *E.g., Age, salary, marital status, etc.*
  - Each client (observation) in the data set is associated with an annual-consumption value
  - Business question:
    - Can we build a model to predict the annual consumptions of future and current clients?
      - Why? To determine whether new power plants are required

# Regression Problems

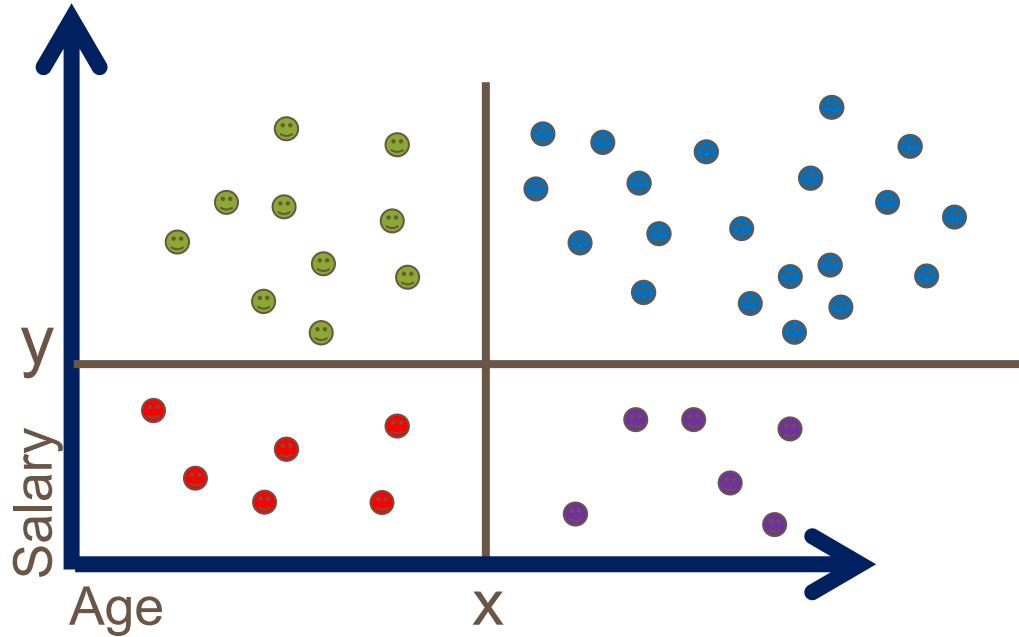
---

➤ Potential solution:

- Translate this problem into a regression problem
- Collect the data
  - Query internal databases
- Build a statistical model to predict future consumption
- Use the model to predict the consumption for all new and current clients

# Regression Problems

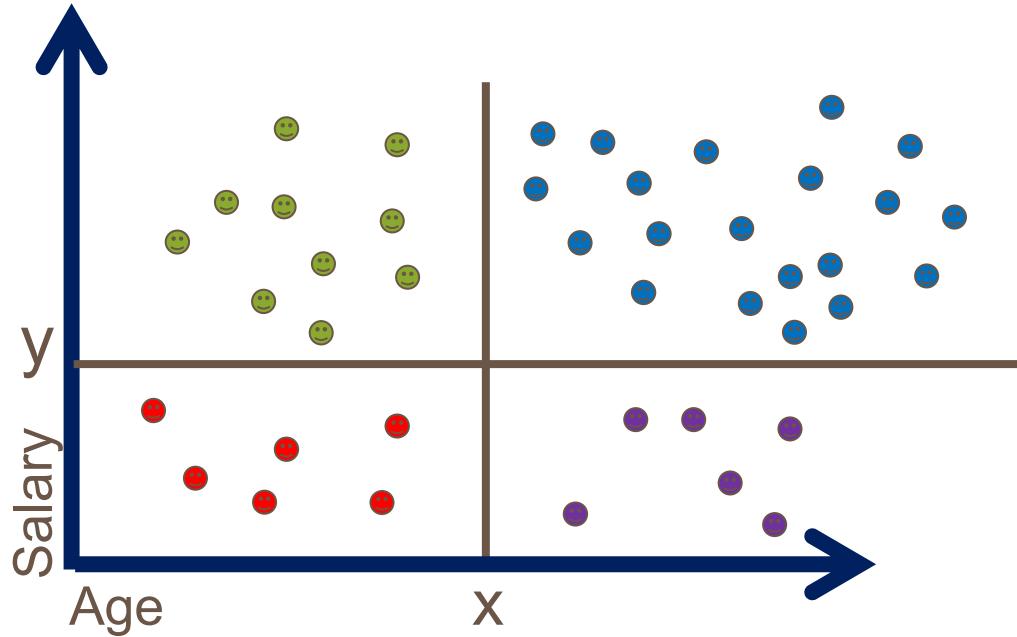
- Decision trees and random forests can also be applied to regression problems
  - For example, a predicted value can simply be the average consumption for all the clients in a partition created by a decision tree



# Regression Problems

## Example:

- New client's age  $> x$  AND new client's salary  $> y$ 
  - Prediction = average consumption of the blue clients
- New client's age  $> x$  AND new client's salary  $< y$ 
  - Prediction = average consumption of the purple clients
- New client's age  $< x$  AND new client's salary  $> y$ 
  - Prediction = average consumption of the green clients
- New client's age  $< x$  AND new client's salary  $< y$ 
  - Prediction = average consumption of the red clients

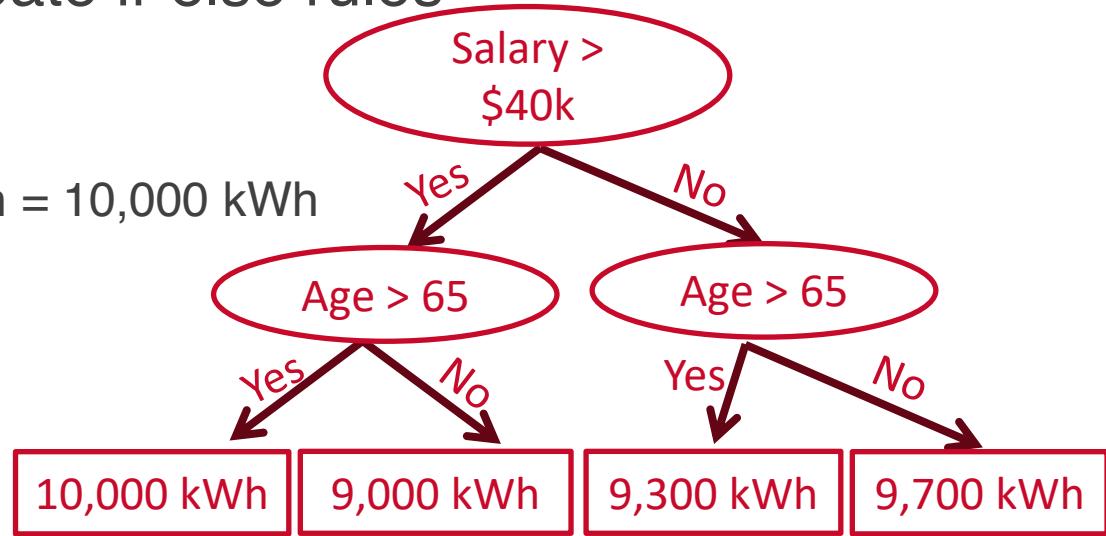


# Regression Problems

- The resulting model is often called *regression tree*, i.e., a decision tree for regression problems
- Regression trees also create if-else rules

E.g., IF 'Salary > \$40K' AND  
'Age > 65'

THEN PredictedConsumption = 10,000 kWh



# Regression Problems

---

- There are many algorithms for building regression trees from a data set
  - These are part of a field called *machine learning*
    - Beyond the scope of this course
- We take a more applied approach
  - Let's build a regression tree to predict annual consumption

# Regression Trees

---

- Regression Trees in Python
  - Let's build a regression tree using the module `sklearn`
  - Load pandas and the data set we use today

```
import pandas as pd
energy_data = pd.read_csv("energy_data.csv")
```
- Before analyzing the data, always check whether each attribute is of the right type (class)
  - If it is not, then change attribute types

```
energy_data.dtypes
```

# Regression Trees

---

- Let's create dummies for the qualitative predictors
  - Remember that we do this because `sklearn` does not accept qualitative predictors, regardless of the model

```
energy_data = pd.get_dummies(energy_data,  
                             columns = ["MaritalStatus", "IncomeLevel", "DwellingArea",  
                                         "HasChildren", "SolarRoof", "ShiftableLoad",  
                                         "AttitudeSustainability", "Tariff" ],  
                             drop_first = True)
```

- Note that our target is now *AnnualConsumption*
  - Hence, we create dummies for *Tariff*

# Regression Trees

---

- Let's split the data now into training and test sets

```
from sklearn.model_selection import train_test_split  
  
x = energy_data.drop(columns=["AnnualConsumption"])  
y = energy_data["AnnualConsumption"]  
  
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.34)
```

# Regression Trees

---

- Building a decision tree using the training set

```
from sklearn.tree import DecisionTreeRegressor
```

```
tree_model = DecisionTreeRegressor()
```

```
tree_model = tree_model.fit(x_train, y_train)
```

- Pay attention to the function `DecisionTreeRegressor` instead of `DecisionTreeClassifier`

# Regression Trees

---

- Let's now use our model to predict the consumption of the clients left out in the test set

```
y_pred = tree_model.predict(x_test)
```

- Now, we have two lists of interest
  - `y_pred` contains the predictions made by our model
  - `y_test` contains the true consumption values
  - How do we compare them?
    - Can we simply check how often these values agree with other?

# Model Evaluation

- Example: Consider the values in the table
  - What is the accuracy of the model if we just compare whether the predicted values are equal to the true value?
  - We must use a **distance metric** to measure how accurate models are in regression problems
    - Different metrics, e.g., MSE, RMSE, MAD, ...

Predictions on the test data ( $\hat{y}_t$ )	True values in the test data ( $y_t$ )
10,009 kWh	9,993 kWh
11,057 kWh	10,990 kWh
12,781 kWh	13,163 kWh
11,800 kWh	11,789 kWh
9,783 kWh	9,999 kWh

# Model Evaluation

---

## Approach #1: Mean Squared Error (MSE)

$$MSE = \frac{1}{n} \times \sum_t (y_t - \hat{y}_t)^2$$

- MSE penalizes large errors because the errors are squared
- It is not in the same units as the data

	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Client 9	Client 10
Consumption in 1,000 kWh	2.444	2.801	2.993	2.963	3.046	3.033	2.637	2.319	2.287	2.38
Forecast	<b>2.3565</b>	<b>2.399</b>	<b>2.6225</b>	<b>2.897</b>	<b>2.978</b>	<b>3.0045</b>	<b>3.0395</b>	<b>2.835</b>	<b>2.478</b>	<b>2.303</b>
Error	<b>0.0875</b>	<b>0.402</b>	<b>0.3705</b>	<b>0.066</b>	<b>0.068</b>	<b>0.0285</b>	<b>-0.4025</b>	<b>-0.516</b>	<b>-0.191</b>	<b>0.077</b>
Error <sup>2</sup>	<b>0.0077</b>	<b>0.1616</b>	<b>0.1373</b>	<b>0.0044</b>	<b>0.0046</b>	<b>0.0008</b>	<b>0.1620</b>	<b>0.2663</b>	<b>0.0365</b>	<b>0.0059</b>

$$\begin{aligned} MSE &\approx (0.0077 + 0.1616 + 0.1373 + 0.0044 + 0.0046 + 0.0008 + 0.1620 + 0.2663 + 0.0365 + 0.0059)/10 \\ &\approx 0.07871 \end{aligned}$$

# Model Evaluation

---

## ➤ Approach #2: Root Mean Square Error (RMSE)

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \times \sum_t (y_t - \hat{y}_t)^2}$$

- RMSE is in the same units as the data
- It assigns more weight to large deviations such as outliers (same as MSE)
  - Large squared differences become larger
  - Small squared differences become smaller

	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Client 9	Client 10
Consumption in 1,000 kWh	2.444	2.801	2.993	2.963	3.046	3.033	2.637	2.319	2.287	2.38
Forecast	2.3565	2.399	2.6225	2.897	2.978	3.0045	3.0395	2.835	2.478	2.303
Error	0.0875	0.402	0.3705	0.066	0.068	0.0285	-0.4025	-0.516	-0.191	0.077
Error <sup>2</sup>	0.0077	0.1616	0.1373	0.0044	0.0046	0.0008	0.1620	0.2663	0.0365	0.0059

$$RMSE \approx \sqrt{0.07871} \approx 0.28$$

# Model Evaluation

---

- Approach #3: Mean Absolute Deviation (MAD) or Mean Absolute Error (MAE)

$$MAD = \frac{1}{n} \times \sum_t |y_t - \hat{y}_t|$$

- MAD is in the same units as the data and it is easy to interpret
- Assigns less weight to outliers

	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Client 9	Client 10
Consumption in 1,000 kWh	2.444	2.801	2.993	2.963	3.046	3.033	2.637	2.319	2.287	2.38
Forecast	2.3565	2.399	2.6225	2.897	2.978	3.0045	3.0395	2.835	2.478	2.303
Error	0.0875	0.402	0.3705	0.066	0.068	0.0285	-0.4025	-0.516	-0.191	0.077
Error	0.0875	0.402	0.3705	0.066	0.068	0.0285	0.4025	0.516	0.191	0.077

$$\begin{aligned} MAD &\approx (0.0875 + 0.402 + 0.3705 + 0.066 + 0.068 + 0.0285 + 0.4025 + 0.5160 + 0.191 + 0.077)/10 \\ &\approx 0.22 \end{aligned}$$

# Model Evaluation

---

## Approach #4: Mean Absolute Percentage Error (MAPE)

$$MAPE = 100 \times \frac{1}{n} \times \sum \frac{|y_t - \hat{y}_t|}{|y_t|}$$

- MAPE compares the absolute errors to the magnitude of the estimated quantity
- It is easy to interpret because it is a percentage, independent of the unit of the target variable

	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Client 9	Client 10
Consumption in 1,000 kWh	2.444	2.801	2.993	2.963	3.046	3.033	2.637	2.319	2.287	2.38
Forecast	2.3565	2.399	2.6225	2.897	2.978	3.0045	3.0395	2.835	2.478	2.303
Error	0.0875	0.402	0.3705	0.066	0.068	0.0285	-0.4025	-0.516	-0.191	0.077
Error  /  y <sub>t</sub>	0.0358	0.1435	0.1238	0.0223	0.0223	0.0094	0.1526	0.2225	0.0835	0.0324

$MAPE \approx (0.0358 + 0.1435 + 0.1238 + 0.0223 + 0.0223 + 0.0094 + 0.1526 + 0.2225 + 0.0835 + 0.0324) * 100 / 10$   
 $\approx 8.48\%$

# Model Evaluation

---

- Which error measure should one use?
  - It depends on a few factors
    - Penalize outliers, interpretability, *etc.*
  - It is common practice to report forecast errors using more than one measure when evaluating models
    - For example, RMSE plus MAD

# Model Evaluation

---

- Let's evaluate our model using the MAE metric
  - `mean_absolute_error` function from the `metrics` submodule

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, y_pred)
```
- Interpretation
  - A *MAE* equal to, say, 887 means that the predictions from the model are expected to be 887 units away from the true predictions
    - *I.e.*, our model is on average 887 kWh off

# Random Forests

---

- Let's build a second model using the same data
  - Random forests: 200 trees

```
from sklearn.ensemble import RandomForestRegressor
```

```
model = RandomForestRegressor(n_estimators=200)
```

```
model = model.fit(x_train,y_train)
```

- Pay attention to the function `RandomForestRegressor` instead of `RandomForestClassifier`

# Random Forests

---

- Let's build a second model using the same data
  - Predictions and evaluation

```
#predictions
```

```
y_pred = model.predict(x_test)
```

```
#evaluation
```

```
mean_absolute_error(y_test, y_pred)
```

# Model Evaluation

---

- Recall that we are evaluating models based only on accuracy
  - Interpretation is another possible evaluation aspect
  - Predictions by regression/decision trees are explainable
    - For example, look at the function `plot_tree` in the `sklearn.tree module`
    - It might take a long time to run that function

# Summary

---

- Summary
  - Data-analytics problems: regression problems
  - Regression trees and random forests
  - Evaluation: training and test sets
- Useful references
  - <http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>
  - <https://docs.google.com/presentation/d/1kSuQyW5DTnkVaZEjGYCkfOxvzCqGEFzWBy4e9Uedd9k/edit>
- Next class: text mining

# ISA 414 – Managing Big Data

## Lecture 14 – Text Mining (Part I)

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Lecture Objectives

---

- Learn how to prepare textual data for analysis
  - Bag of words approach
    - Key concepts
      - Term Frequency (TF)
      - Inverse Document Frequency (IDF)
      - Term Frequency Inverse Document Frequency (TFIDF)
    - Text cleaning
      - Stemming
      - Stop words

# Lecture Instructions

---

- Download the notebook “*Lecture 14.ipynb*” available on Canvas
  - Open the above file with VS Code

# Data Analysis

---

- Previous lectures
  - Data analytics problem: classification and regression
  - Modeling: decision trees, random forests
- Assumption in our analysis
  - Data are in tabular format
- What if the data are unstructured (e.g., textual)?
  - Can we still build models to make predictions?

# Data Analysis

---

- Text is different
  - No predefined, uniformly applicable format
    - *E.g.*, contrast a tweet against an article on the NY Times
  - Context dependent
    - *E.g.*, 'ring' can be a verb or a noun; 'incredible' can have positive or negative qualification; kkk = laughing in Portuguese
  - Spelling mistakes and grammatical errors may contaminate texts
    - Homonyms and synonyms increase the complexity of an analysis

# Data Analysis

---

- Text mining (analytics): deriving information from textual data
  - Many different tasks
    - Text categorization (e.g., spam filter – next lecture)
    - Sentiment analysis (next lecture)
    - Text clustering (e.g., plagiarism checker)
    - Automatic summarization (e.g., “consensus” of several reviews)
    - Topic modeling (future classes)
  - Involves extensive preprocessing
    - Oftentimes, a tabular structure is imposed
      - Topic of this lecture
  - Let's build intuition first

# Human Classifier: SPAM or NOT SPAM?

---

Greetings to you my friend,

I know this will come to you as a surprise because you do not know me.

I am John Alison and I work at Central Bank of Nigeria, packaging and courier department.

I got your contact among others from a search on the internet and I was inspired to seek your co-operation. I want you to help me clear this consignment that is already in Europe which I shipped through our CBN accredited courier agent. The content of the package is \$20,000,000 all in \$100 bills, but the courier company does not know that the consignment contains money.

All I want you to do for me now is to give me your mailing address, your private phone number, and credit card information so that I can deposit some money to cover your upfront costs.

Please, let me know your response as soon as possible. WE CANNOT WASTE THIS OPPORTUNITY.

With Love,

john\_alison444@yahoo.com

# Human Classifier: SPAM or NOT SPAM?

---

Hi Professor Carvalho,

I first wanted to thank you for all the kind feedback we received on our project. We spent a lot of time on it, as I'm sure you did grading them, and I really appreciate your comments.

My grade is currently lingering around a B+ due to the difficulties I had in the first exam. I was wondering if the 90% cut off for an A- is a hard cut off, or if I receive say an 89.9, would that be an A-? I understand every professor has their own grading policies, but I just wanted to know for sure what I needed to score on the final to get an A or an A-.

Thanks again for a great semester, I really appreciate all the time you spend grading our assignments and working with us individually.

# Text Mining

---

- How do you know that the first email is SPAM, but the second is not?
  1. Context
    - Requires deep understanding of the language
  2. Presence of certain keywords
    - \$20,000,000, credit, card, love
    - Other common keywords
      - Viagra, sex, chat, money, currency, bitcoin, ...
  - Bottom line: simply checking for the presence/absence of certain words can help with the classification task

# Text Mining

---

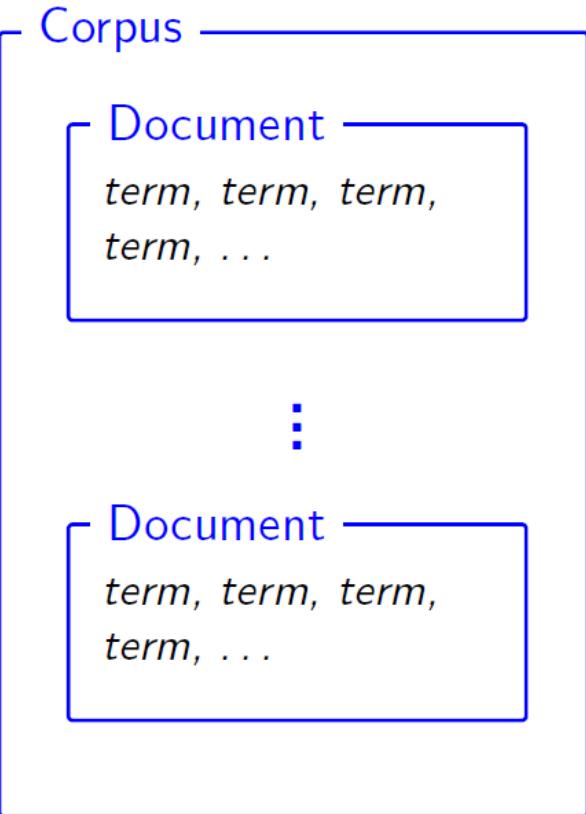
- Based on the previous observation, we have one way of imposing a tabular structure
  - Rows = observations (textual documents)
  - Columns = individual words
  - Cells = some sort frequency measure (counting)
  - Example:

Doc	the	hotel	has	one	bad	room	of	bathroom	is	other	good
1	1	1	1	1	1	1	0	0	0	0	0
2	1	1	0	0	1	1	1	0	1	0	0
10	3	1	0	0	1	1	0	0	1	1	1

# Text Mining

---

- Terminology taken from the Information Retrieval (IR) domain
  - A piece of text is referred to as a **document**
  - Documents consist of items called **terms**, **tokens** or, plainly, **words**
  - Documents that belong together form a **corpus**
    - In document-oriented databases, like MongoDB, this is called a **collection**

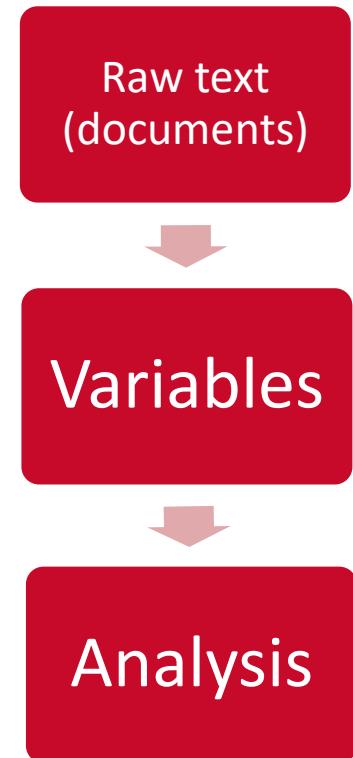


# Text Mining

---

## ➤ Our approach in this course

- Raw documents form the basis of the analysis
- Preprocessing is done to transform the raw texts into a tabular format
- The constructed data sets are used in different analyses (next lecture)
  - Descriptive
    - Sentiment analysis
    - Word association
  - Statistical modeling
    - Classification/regression problems



# Text Mining

---

- There are different ways of representing a document
- A well-known representation format is the “**bag of words**”
  - Each term, token, or word in a document is considered individually
    - Only counts of individual words matter
  - The syntax of the language is ignored
    - That is, grammar, word order, sentence structure
  - Simplistic, but still powerful approach

# Bag of Words

---

- Counts of individual words will lead to the familiar tabular format
  - **Document-Term Matrix (DTM)**
  - Different ways of counting words
    - Within documents: **term frequency** (TF)
    - Across documents: **inverse document frequency** (IDF)
    - Within and across documents: **term frequency + inverse document frequency** (TFIDF)

# Term Frequency (TF)

# Term Frequency

---

- Term frequency (TF) refers to the occurrence of words in a document
  - Binary TF
  - Frequency-based TF
    - Absolute
    - Normalized

# Term Frequency

---

- Binary term frequencies: indicates whether a term is present
  - All the occurrences of a word in a document count as one
  - Ideal for short documents like tweets
  - Example
    - Doc 1: the hotel has one bad room
    - Doc 2: the room of the hotel is bad
    - Doc 3: one bathroom is bad, the other bathroom is good

Doc	the	hotel	has	one	bad	room	of	bathroom	is	other	good
1	1	1	1	1	1	1	0	0	0	0	0
2	1	1	0	0	1	1	1	0	1	0	0
3	1	0	0	1	1	0	0	1	1	1	1

# Term Frequency

---

## ➤ Frequency-based term frequencies

- Ideal for long documents like product reviews
- Can be either absolute or normalized
  - **Absolute:** counts the number of occurrences of a word in a document
  - **Normalized:** counts the number of occurrences of a word in a document divided by the number of words in the document
    - Deals with documents of varying length

# Term Frequency

---

## ➤ Absolute term frequency

- Example

- Doc 1: jazz music has a swing rhythm
- Doc 2: swing is hard to explain
- Doc 3: swing rhythm is a natural rhythm

Doc	a	explain	hard	has	is	jazz	music	natural	rhythm	swing	to
1	1	0	0	1	0	1	1	0	1	1	0
2	0	1	1	0	1	0	0	0	0	1	1
3	1	0	0	0	1	0	0	1	2	1	0

# Term Frequency

---

- Normalized text frequency
  - Example
    - Doc 1 (length 6): jazz music has a swing rhythm
    - Doc 2 (length 5): swing is hard to explain
    - Doc 3 (length 6): swing rhythm is a natural rhythm

Doc	a	explain	hard	has	is	jazz	music	natural	rhythm	swing	to
1	0.166	0	0	0.166	0	0.166	0.166	0	0.166	0.166	0
2	0	0.2	0.2	0	0.2	0	0	0	0	0.2	0.2
3	0.166	0	0	0	0.166	0	0	0.166	0.332	0.166	0

.....

# Inverse Document Frequency (IDF)

# Inverse Document Frequency

---

- Term frequencies measure how popular words are inside documents
  - But not across documents
  - Incomplete metric
    - Example: the word “the” is likely to occur many times in many documents
      - Can you classify an email as SPAM based on the presence/absence of the word “the”?
    - Example: the word “Viagra” is likely to occur a few times in only a few documents
      - Is it an important term when classifying emails?

# Inverse Document Frequency

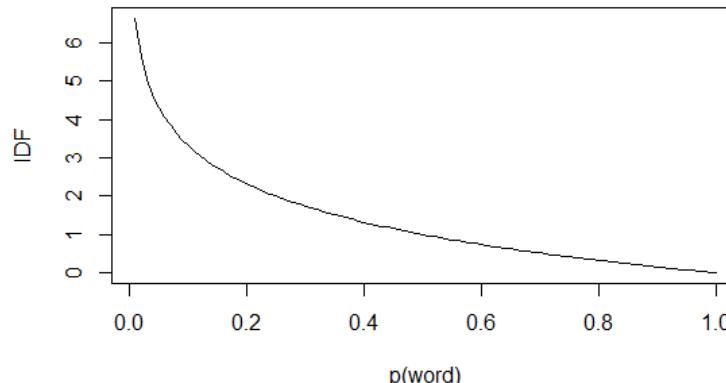
---

- The inverse document frequency (IDF) measures how rare words are in a corpus
  - A term is likely to represent a single document well if the term is rare in the corpus
- Let  $p(\text{word}) = \frac{\text{Number of documents containing the term "word"}}{\text{total number of documents}}$ 
  - Fraction of documents that contains the term 'word'
- $IDF(\text{word}) = \log_2 \frac{1}{p(\text{word})}$

# Inverse Document Frequency

---

- One can interpret IDF as a boost a term gets for being rare
  - Plot
    - Left region: a word receives high IDF score when it is very rare
    - Right region: IDF of very common words (less discriminatory)



# Inverse Document Frequency

---

## ➤ Example

- Doc 1: jazz music has a swing rhythm
- Doc 2: swing is hard to explain
- Doc 3: swing rhythm is a natural rhythm

Doc	a	explain	hard	has	is	jazz	music	natural	rhythm	swing	to
# of docs	2	1	1	1	2	1	1	1	2	3	1
P(t)	0.66	0.33	0.33	0.33	0.66	0.33	0.33	0.33	0.66	1	0.33
IDF	0.58	1.58	1.58	1.58	0.58	1.58	1.58	1.58	0.58	0	1.58

.....

# Term Frequency Inverse Document Frequency (TFIDF)

# TFIDF

---

- TF measures the frequency of word occurrences in a document without reference to the corpus
- IDF indicates how rare words are in the corpus
- What about combining these two metrics?
  - TFIDF

# TFIDF

---

- $\text{TFIDF}(\text{word}; \text{document}) = \text{TF}(\text{word}, \text{doc}) * \text{IDF}(\text{word})$ 
  - IDF reduces the weight of common terms and inflates the weight of rare terms
  - Example: Absolute TFIDF
    - Doc 1: jazz music has a swing rhythm
    - Doc 2: swing is hard to explain
    - Doc 3: swing rhythm is a natural rhythm

	Doc	a	explain	hard	has	is	jazz	music	natural	rhythm	swing	to
TF	1	1	0	0	1	0	1	1	0	1	1	0
TF	2	0	1	1	0	1	0	0	0	0	1	1
TF	3	1	0	0	0	1	0	0	1	2	1	0
IDF		0.58	1.58	1.58	1.58	0.58	1.58	1.58	1.58	0.58	0	1.58

# TFIDF

---

- $\text{TFIDF}(\text{word}; \text{document}) = \text{TF}(\text{word}, \text{doc}) * \text{IDF}(\text{word})$ 
  - IDF reduces the weight of common terms and inflates the weight of rare terms
  - Example: Absolute TFIDF
    - Doc 1: jazz music has a swing rhythm
    - Doc 2: swing is hard to explain
    - Doc 3: swing rhythm is a natural rhythm

	Doc	a	explain	hard	has	is	jazz	music	natural	rhythm	swing	to
TFIDF	1	0.58	0	0	1.58	0	1.58	1.58	0	0.58	0	0
	2	0	1.58	1.58	0	0.58	0	0	0	0	0	1.58
	3	0.58	0	0	0	0.58	0	0	1.58	1.16	0	0

# TFIDF

---

- Which counting (frequency metric) should one use?
  - Hard to say without a context
    - TFIDF is one of the most popular metrics
      - Short documents: Binary TF
      - Documents with similar lengths: Absolute TF
      - Documents with drastically different lengths: Normalized TF
  - One can always try all metrics when making predictions!
    - Build one predictive model for each different DTM
    - Evaluate each model
    - Pick the most accurate one

.....

# TFIDF in Python

# TFIDF

---

- We shall use the (sub)module `feature_extraction` inside `sklearn`

```
from sklearn.feature_extraction.text import TfidfVectorizer  
  
corpus = [  
    'jazz music has a swing rhythm',  
    'swing is hard to explain',  
    'swing rhythm is a natural rhythm',  
]  
  
vectorizer = TfidfVectorizer()  
  
tfidf_result = vectorizer.fit_transform(corpus)
```

# TFIDF

---

## ➤ Important technical points

- The IDF formula used by `sklearn` is not the “textbook version”
  - They use smoothing techniques to avoid division by 0
- The resulting TFIDF values are normalized
  - Technical details available at [https://scikit-learn.org/stable/modules/feature\\_extraction.html#text-feature-extraction](https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction)
- The data structure that stores the TFIDF values is called “sparse matrix”
  - It is a concise way of storing data since most TFIDF values are equal to 0 in practice
  - Look at today’s notebook for a sample code on how to convert a sparse matrix to a data frame

# Text Mining

---

- Note that not every single word in your corpus will be part of the DTM
  - By default, sklearn removes words of length one and punctuation marks
    - E.g., "a", "[!@#\$%^&\*(){};:<>,.?/~`"
  - Moreover, all words are automatically converted to lowercase
  - One can further remove words that are not informative (too common)
    - These are called **stop words**
    - sklearn already comes with a stop-word list; use the argument `stop_words = "english"` in the function `TfidfVectorizer` (see today's notebook)
- The above steps help decreasing the number of variables
  - Otherwise, a simple corpus can result in thousands of variables

# Text Mining

---

- It is also commonplace to perform **stemming**
  - Stemming: reduce words to their roots
    - *E.g.*, playing -> play
  - Not covered in this course (search for CountVectorizer)
- The 'bag of words' disregards word combinations, like 'bed and breakfast' or 'New York'
  - "BUS" = vehicle; "420" = code for drugs; "BUS 420" = FSB course
  - **N-grams** consider combinations of  $n$  words
    - Uni-grams, bi-grams, tri-grams, ...
    - Advantage: more information is brought to the analysis
    - Disadvantage: number of variables may become very large
    - Look at today's notebook for a sample code

# Text Mining

---

- How can we use document-term matrices for predictive analytics?
  - From unstructured to structured data
    - Doc 1: you won \$1000000 dollars (target: SPAM)
    - Doc 2: I love ISA 414 (target: NOT SPAM)
    - Doc 3: improve your life now: buy Viagra (target: SPAM)

Doc	1000000	414	buy	dollars	i	improve	isa	life	love	now	viagra	won	you	your	Target
1	0.4	0	0	0.4	0	0	0	0	0	0	0	0.4	0.4	0	SPAM
2	0	0.4	0	0	0.4	0	0.4	0	0.4	0	0	0	0	0	NOT SPAM
3	0	0	0.26	0	0	0.26	0	0.26	0	0.26	0.26	0	0	0.26	SPAM

# Summary

---

- We learned techniques to preprocess textual data
  - Bag of words (tabular structure)
  - There are other ways of imposing a well-defined structure that take grammar into account
    - Word2Vec, FastText, ...
    - Beyond the scope of this course
- Next lecture: text mining (part II)

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 15 – Text Mining (Part II)

*Sentiment Analysis*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Announcements

---

- Midterm project
  - To be released on Tuesday, October 19<sup>th</sup>
  - Deadline: 11:59 pm on October 22<sup>nd</sup>
  - Individual or in groups of two
    - Send me an email in case you already have a defined group
    - Send me an email in case you want to be paired up with a random person
      - No guarantee there will be another person available

# Lecture Objectives

---

- Quick review of Assignment 3
- Learn descriptive-analysis techniques involving textual data
  - Sentiment analysis
- Learn how to build classifiers using textual data

# Lecture Instructions

---

- Download the notebook “*Lecture 15.ipynb*” available on Canvas
  - Open the above file with VS Code
- Download the following files to the same folder our notebook is:
  - starbucks.csv
  - IMDB.csv
  - negative-words.txt
  - positive-words.txt

# Text Mining

---

- Previous lecture
  - We learned one approach to preprocess textual data
    - Bag of words
      - DTM
    - Counting metrics
      - TF
      - IDF
      - TFIDF
    - Other preprocessing techniques
      - N-grams
      - Stop words

# Text Mining

---

- Today, we focus on data analysis
  - Textual data
  - Techniques:
    - Sentiment analysis
    - Statistical modeling (classification)

# Text Mining

---

## ➤ Background story #1: Starbucks and refugees

- Starbucks pledged to hire 10,000 refugees on January 29<sup>th</sup>, 2017

*"There are more than 65 million citizens of the world recognized as refugees by the United Nations, and we are developing plans to hire 10,000 of them over five years in the 75 countries around the world where Starbucks does business. And we will start this effort here in the U.S. by making the initial focus of our hiring efforts on those individuals who have served with U.S. troops as interpreters and support personnel in the various countries where our military has asked for such support."*

Howard Schultz (CEO)

- Several people voiced their opinions on social media
  - From support to boycott calls
  - But we are not going to the same in this course
    - Your opinions on this matter are yours
    - Here, we are unbiased/unopinionated data scientists

# Text Mining

---

- Background story #1: **Starbucks and refugees**
  - Previous students taking the ISA 414 course decided to analyze how the sentiment towards Starbucks changed from before to after the announcement
    - Case study part of the paper “*Off-The-Shelf Artificial Intelligence Technologies for Sentiment and Emotion Analysis: A Tutorial on Using IBM Natural Language Processing*” available on Canvas
  - They collected posts on Starbucks Facebook page 14 days before to 14 days after the announcement
    - Let’s work with that data, but with a shorter time window (01/29 to 01/31)
    - Data set “starbucks.csv”



# Text Mining

---

## ➤ Preliminaries

- Load the required packages

```
import pandas
```

- Load the data set

```
raw_data = pandas.read_csv("starbucks.csv")
```

- Let's now preprocess our data

- We shall build a TF-based DTM, *i.e.*, no IDF

- Why? In our first analysis, we only need to know whether a word is present in a text

# Text Mining

---

## ➤ Data preprocessing

- Vectorizing data and creating a corpus

```
from sklearn.feature_extraction.text import TfidfVectorizer  
vectorizer = TfidfVectorizer(use_idf = False)  
tf_values = vectorizer.fit_transform(raw_data["message"])
```

- Recall that tf\_values is a sparse matrix

- Notation  $(x, y) z$  means the score  $z$  the word indexed by column  $y$  in document  $x$  receives
  - In our analysis, we are interested in knowing the precise words represented by column index  $y$  in each document  $x$  that have score  $z > 0$

# Text Mining

---

## ➤ Data preprocessing

- From our previous class, we learned about the function `get_feature_names()` that gives us all the words in our corpus
  - To get the words in a document, we have to subset the result from `get_feature_names()`
- Example: let's retrieve all the words in the first document of our preprocess corpus

```
all_word_names = vectorizer.get_feature_names()  
doc0_word_index = tf_values[0,:].nonzero()[1]  
doc0_word_names = [all_word_names[w] for w in doc0_word_index]
```

.....

# SENTIMENT ANALYSIS

# Text Mining

---

## ➤ Sentiment analysis

- Goal: obtain the overall sentiment (positive/negative) behind texts
  - Many different approaches that take linguistic aspects into account
  - In spirit, the approach we learn in this course is similar to the bag-of-words approach
    - Consider the difference between the number of positive and the number of negative words in a text

# Text Mining

---

- Sentiment analysis
  - Big picture
    - Break each text into individual words
    - Count the number of positive and negative words
    - Calculate the difference between the above counts
      - Score > 0: the overall sentiment behind a document is positive
      - Score < 0: the overall sentiment behind a document is negative
      - Score = 0: the overall sentiment behind a document is neutral

# Text Mining

---

- Sentiment analysis
  - List of positive and negative words available on Canvas
    - Officially called a **lexicon**
    - Thanks to Dr. Bing Liu
  - Let's start by loading these lists

```
file = open('positive-words.txt', 'r')
positive_words = file.read().splitlines()
```

```
file = open('negative-words.txt', 'r')
negative_words = file.read().splitlines()
```

# Text Mining

---

## ➤ Sentiment analysis

- Example: estimating sentiment behind the 80<sup>th</sup> text
  - “*To the Starbucks Corporation, as the wife of a uniformed service member, I want to **applaud** your pledge to employ refugees. **Thank** you for the ray of hope in a **frightening** time, and for demonstrating humanity in the realm of business.*”

```
doc80_word_index = tf_values[79,:].nonzero()[1]
doc80_word_names = [all_word_names[w] for w in doc80_word_index]
count_positive_words = [positive_words.count(word) for word in doc80_word_names]
count_negative_words = [negative_words.count(word) for word in doc80_word_names]
score = sum(count_positive_words) - sum(count_negative_words)
```

# Text Mining

---

## ➤ Sentiment analysis

- Example: estimating sentiment behind the 248<sup>th</sup> text
  - “I have been a *loyal* consumer of Starbucks products for over 10 years. But *unfortunately* recent events and comments by upper management have *disgusted* me to the point that I will no longer use your products.”

```
doc248_word_index = tf_values[247,:].nonzero()[1]
doc248_word_names = [all_word_names[w] for w in doc248_word_index]
count_positive_words = [positive_words.count(word) for word in doc248_word_names]
count_negative_words = [negative_words.count(word) for word in doc248_word_names]
score = sum(count_positive_words) - sum(count_negative_words)
```

# Text Mining

---

## ➤ Sentiment analysis

- The previous approach to estimate sentiment is rather naïve
  - *E.g.*, it does not take into account sarcasm
- Nonetheless, it tends to work quite well with short texts like tweets (less so for Facebook posts)
- There are many lexicons out there
  - SenticNet: <http://sentic.net/senticnet-4.0.zip>
  - MPQA: <http://mpqa.cs.pitt.edu>

# Text Mining

---

- A different approach to performing sentiment analysis is to build classifiers
  1. Manually label all the documents in a corpus
    - *i.e.*, to manually set the target variable as either positive or negative
  2. Build a model to predict the label (target)
  3. Use the trained model to determine the sentiment of new, unlabeled documents
- We explore this approach next

.....

# STATISTICAL MODELING

# Text Mining

---

- Background story #2
  - Suppose that Rotten Tomatoes wants to include movie reviews from reviewers who do not explicitly report numerical ratings
  - Goal: develop a system that predicts whether a movie is “fresh” (positive review) or “rotten” (negative review) based on a textual review
    - Data: 1800 reviews manually labeled (positive/negative) by an expert

# Text Mining

---

- Let's start by loading and processing the data
  - Note that we now build a TFIDF DTM, and we remove stop words to make our DTM smaller

```
import pandas  
from sklearn.feature_extraction.text import TfidfVectorizer  
  
raw_data = pandas.read_csv("IMDB.csv")  
  
vectorizer = TfidfVectorizer(stop_words = "english")  
tfidf_values = vectorizer.fit_transform(raw_data['text'])
```

# Text Mining

---

- Since we will build a model (recall we didn't do it before), we need now to explicitly create a data frame

```
dtm = pandas.DataFrame(tfidf_values.toarray())
dtm.columns = vectorizer.get_feature_names()
```

- How many columns does `dtm` have?
  - Recall that there are only 1800 rows
  - Can you see how quickly a `dtm` can grow in size?
    - Look at today's notebook for the code that calculates the actual size

# Text Mining

---

- Thus far, we transformed the `text` column in the original data into a TF-IDF matrix, *i.e.*, we created a DTM
- Let's add a target (`class`) to our DTM
  - Recall that `class` was created by humans when manually labeling the data

```
dtm['target_var'] = raw_data['class']
```
- Our data frame now contains close to 37,500+ variables, including one target
  - Let's build a classifier (random forest)
  - No need to create dummies (why?)

# Text Mining

---

- Splitting data into training and test sets

```
from sklearn.model_selection import train_test_split
```

```
x = dtm.drop(columns=["target_var"])
```

```
y = dtm["target_var"]
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.34)
```

# Text Mining

---

- Training a random forest (this might take a while)

```
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier(n_estimators=200)
```

```
model = model.fit(x_train,y_train)
```

- Evaluating our model

```
from sklearn import metrics
```

```
y_pred = model.predict(x_test)
```

```
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

# Text Mining

---

- The model's accuracy is around 73%-75%
  - Better than guessing
  - A lot of room for improvements
    - More powerful models (e.g., more trees in the forest)
    - More meaningful variables
      - Manually remove uninformative words
    - Better preprocessing
      - *E.g.*, n-grams, stemming, ...

# Text Mining

---

- Assumption in our previous analysis: target values are available
  - Manually created by an expert
    - Expensive and time consuming
  - Alternative: crowdsourcing
    - Non-experts
    - Platforms: Amazon Mechanical Turk and CrowdFlower
    - Quick, cheap, but data quality is often an issue

# Text Mining

## ➤ Example

- Amazon Mechanical Turk

Requester: Arthur      Reward: \$0.05 per HIT      HITs available: 0      Duration: 1 Hours

Qualifications Required: Masters has been granted

HIT Preview

**Tweet Sentiment Analysis Instructions (Click to collapse)**

Pick the sentiment based on the following criterion:

Sentiment	Guidance
Positive	Select this if the item embodies emotion that was generally happy or satisfied. For example, "Sure I'll shop there again."
Neutral	Select this if the item does not embody positive or negative emotion toward the topic. For example, "Yeah, I guess it's ok." or "Is their customer service open 24x7?"
Negative	Select this if the item embodies emotion that is perceived to be angry or upset toward the topic. For example, "I don't know if I'll shop there again because I don't trust them."

**Tweet:** Amazing talk about leadership by Drew Dudley  
@miamiuniversity #MUEngLdrInst @DayOneDrew

**Sentiment expressed by the Tweet:**

Positive
Neutral
Negative

**Submit**

# Analysis of Unstructured Data

---

- We are focusing on texts, but there are many other forms of unstructured data
  - Image analysis
  - Video analysis
  - Speech analysis

# Homework #7

---

- Let's go back to the Starbucks problem
  - How many comments are positive?
    - Score > 0
  - How many comments are negative?
    - Score < 0
- Write a script that answer the above questions and report it on Canvas
  - Remember: do **NOT** use IDF when creating a DTM
  - Hint: write a FOR-loop that iterates over the length of `raw_data`

# Summary

---

- We learned a few techniques to analyze textual data
  - Sentiment analysis
  - Modeling
- Next class
  - Topic modeling

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 16 – Text Mining (Part III)

*Topic Modeling*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Announcements

---

- Midterm project
  - To be released on Tuesday, October 19<sup>th</sup>
  - Deadline: 11:59 pm on October 22<sup>nd</sup>
  - Individual or in groups of two
    - Send me an email in case you already have a defined group
- No office hours
  - Virtual office hours on Friday from 10:30 to 11:30
  - Email me for a WebEx link

# Lecture Objectives

---

- Quick review of Homework 7
- Learn how to automatically extract topics from textual data

# Lecture Instructions

---

- Download the notebook “*Lecture 16.ipynb*” available on Canvas
  - Open the above file with VS Code
- Download the file “*abcnews-date-text.csv*” to the same folder our notebook is

# Text Mining

---

- Text mining (analytics): deriving information from textual data
  - Tasks
    - Text categorization (Lecture 15)
    - Sentiment analysis (Lecture 15)
    - Text clustering (e.g., plagiarism checker)
    - Automatic summarization (e.g., “consensus” of several reviews)
    - Topic modeling (today)

# Text Mining

---

- Running example: can we automatically find the topic of news headlines?
  - Data: over 1 million news headlines
    - Source: [www.kaggle.com/therohk/million-headlines](http://www.kaggle.com/therohk/million-headlines)
    - Two columns: `published_date` and `headline_text`
    - We only focus on the latter
  - Traditional problems faced in ISA 414 projects:
    - Which topics are users discussing when mentioning company/product X on social media?

# Topic Modeling

---

- The mathematics behind the techniques we learn today are beyond the scope of this course
  - Traditionally covered in PhD courses
- These techniques can be seen as **unsupervised learning** methods
  - We **do not** have to define a target variable
    - We “cluster” the documents into groups
  - If we had a target, we could train models to classify (find the topics) of future texts
    - See our previous class

# Topic Modeling

---

- Traditional topic modeling techniques
  - Latent Semantic Analysis (LSA or LSI)
    - Technically, it is a dimensionality reduction model
      - Main idea: words occur in similar pieces of text if they have similar meaning
  - Latent Dirichlet Allocation (LDA)
    - Probabilistic model for identifying abstract topics from a corpus
      - Main idea: each document has a mix of topics, and every topic has a mix of words
    - Our focus in this course
    - For the brave and bold:
      - Search for the paper “Latent Dirichlet Allocation” published in the Journal of Machine Learning Research in 2003

# LDA

---

- LDA is based upon two general assumptions:
  - Documents that have similar words usually have the same topic
  - Documents that have groups of words frequently occurring together usually have the same topic
  
- Mathematically, the above two assumptions can be represented as:
  - Documents are probability distributions over latent topics
  - Topics are probability distributions over words

# LDA

---

- Think about LDA as an algorithm that learns two “layers” from a corpus of documents
  - The first layer is the distribution of categories/topics
    - For example, financial, weather, and political news
  - The second layer is distribution of words within each topic
    - For example, “sunny” and “cloud” are often in “weather” news while “money” and “stock” are in “financial” news

Finance	Weather	Arts	Sport
Money	Sunny	Music	World Cup
Stock	Cloud	Piano	Soccer
Trade	Humanity	Calligraphy	Tennis
Market	Temperature	Photography	Sailing

# LDA

---

## ➤ Important points

- Some words (e.g., “a”, “with”, “can”) do not contribute to topic modeling
  - These words exist in almost all documents
  - Consequently, they will likely be in all categories with the same probability
  - Therefore, data preprocessing is crucial
    - For example, stop words removal
- LDA works with counts of words
  - That is, with a DTM having absolute TF values
  - So far, we have only learned how to create DTMs that have TFIDF and normalized TF scores

# Topic Modeling

---

- Let's return to our example
  - Can we automatically find the topic of news headlines?
- Let's start by loading the data
  - There are over 1.2 million data points
    - It would take a very long time to work with that much data
    - Let's work with a sample of 10,000 headlines only

```
import pandas  
raw_data = pandas.read_csv("abcnews-date-text.csv")  
corpus = raw_data["headline_text"]  
corpus = corpus[0:10000]
```

# Topic Modeling

---

## ➤ Next, let's create a DTM

- Remember that LDA works with absolute frequencies
- We use the `CountVectorizer` in `sklearn` instead of `TfidfVectorizer` we used in previous classes

```
from sklearn.feature_extraction.text import CountVectorizer  
count_vect = CountVectorizer(stop_words='english')  
dtm = count_vect.fit_transform(corpus)
```

- There is a lot more we could do when creating the DTM
  - *E.g.*, put an upper (`max_df`) and lower (`min_df`) threshold on the proportion of documents having a term
    - See [http://scikit-learn.org/stable/modules/generated/sklearn.feature\\_extraction.text.CountVectorizer.html](http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html)

# Topic Modeling

---

- Finally, let's run LDA on the DTM
  - More important points
    - One **must** define (guess) the number of topics when running LDA
      - It is common practice to run LDA with different numbers
    - One may also define (guess) initial probabilities of topics to help the algorithm
      - See <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html>
  - LDA with 5 topics

```
from sklearn.decomposition import LatentDirichletAllocation  
LDA = LatentDirichletAllocation(n_components=5)  
LDA.fit(dtm)
```

# Topic Modeling

---

- LDA: understanding the results
  - We now have 5 topics, each one defined by words

```
first_topic      = LDA.components_[0]
second_topic    = LDA.components_[1]
third_topic     = LDA.components_[2]
fourth_topic    = LDA.components_[3]
fifth_topic     = LDA.components_[4]
```
  - How popular is each word inside a topic?
    - Look inside any of the above lists
    - You will see “scores” for each index column of our DTM

# Topic Modeling

---

- LDA: understanding the results
  - Let's retrieve the top 10 words in each topic
    - 2-step process in Python:
      - Step 1: sort the indexes of the words in each topic by score and retrieve the top 10

```
top10_index_1st_topic = first_topic.argsort()[-10:]
```

```
top10_index_2nd_topic = second_topic.argsort()[-10:]
```

```
top10_index_3rd_topic = third_topic.argsort()[-10:]
```

```
top10_index_4th_topic = fourth_topic.argsort()[-10:]
```

```
top10_index_5th_topic = fifth_topic.argsort()[-10:]
```

# Topic Modeling

---

- LDA: understanding the results
    - Let's retrieve the top 10 words in each topic
      - 2-step process in Python:
        - Step 2: retrieve the top 10 words based on their indexes
- ```
all_words = count_vect.get_feature_names()
```

```
top10_words_1st_topic = [all_words[i] for i in top10_index_1st_topic]
top10_words_2nd_topic = [all_words[i] for i in top10_index_2nd_topic]
top10_words_3rd_topic = [all_words[i] for i in top10_index_3rd_topic]
top10_words_4th_topic = [all_words[i] for i in top10_index_4th_topic]
top10_words_5th_topic = [all_words[i] for i in top10_index_5th_topic]
```

# Topic Modeling

---

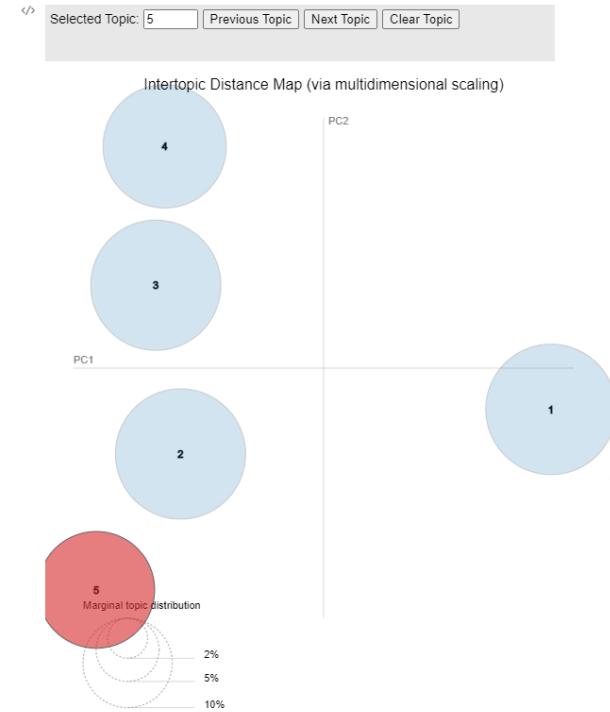
- LDA: understanding the results
  - Let's look at other metrics to understand the composition of topics
    - Install the `pyLDAvis` module  
`pip install pyldavis`
    - Run the LDA visualization module for the top 10 words

```
import pyLDAvis
import pyLDAvis.sklearn
pyLDAvis.enable_notebook()
pyLDAvis.sklearn.prepare(LDA, dtm, count_vect, R = 10)
```

# Topic Modeling

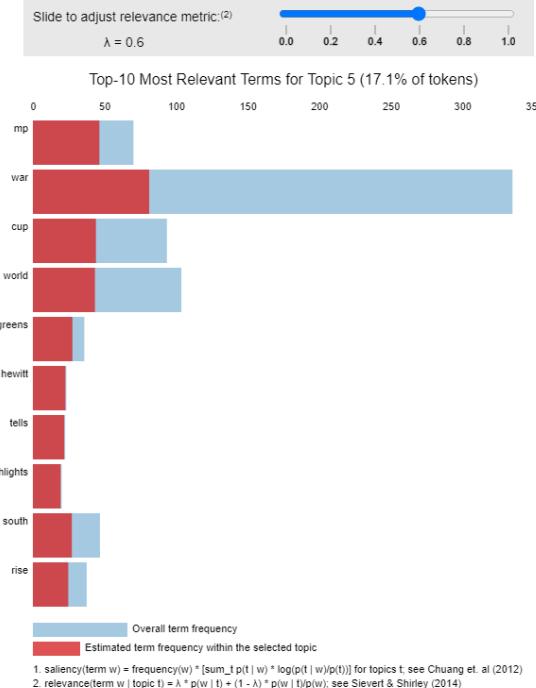
## ➤ LDA: understanding the results

- The left panel shows the predominance of the topics and how related (overlapping) they are
  - Ideally, one wants to avoid overlapping as much as possible
  - There are different ways of calculating the distance between topics
    - Technically, topics are mapped through dimensionality reduction (PCA/t-sne) on distances between each topic's probability distributions into 2D space
    - For more information, search for the paper “*LDAvis: A method for visualizing and interpreting topics*”



# Topic Modeling

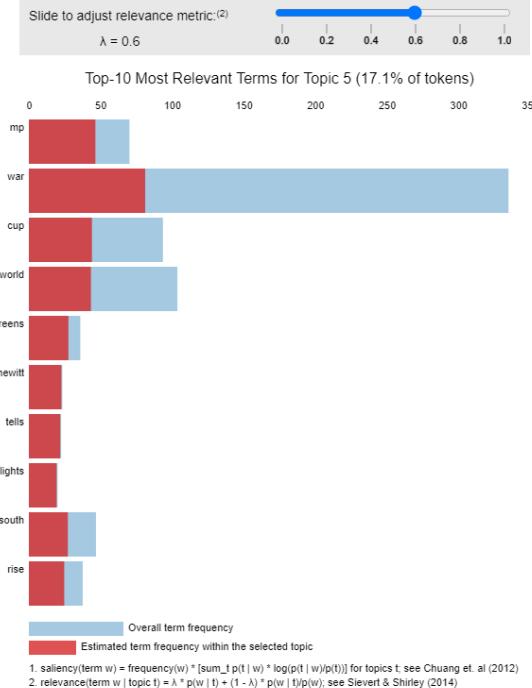
- LDA: understanding the results
  - The right panel shows two metrics
    - **Salience**
      - Thresholding method for selecting which terms are included in the visualization
      - We are selecting the 10 most salient words



# Topic Modeling

## ➤ LDA: understanding the results

- The right panel shows the **relevance** metric
  - Defined by a parameter lambda  $\lambda$ 
    - Value close to 0 selects potentially rare but more exclusive terms for the selected topic
    - Values close to 1 selects more frequently occurring terms in the document that might not be exclusive to the topic
  - Technically  $\lambda$  weighs the probability of a term within a topic relative to its lift
    - For more information search for the paper “*LDAvis: A method for visualizing and interpreting topics*”



# Topic Modeling

---

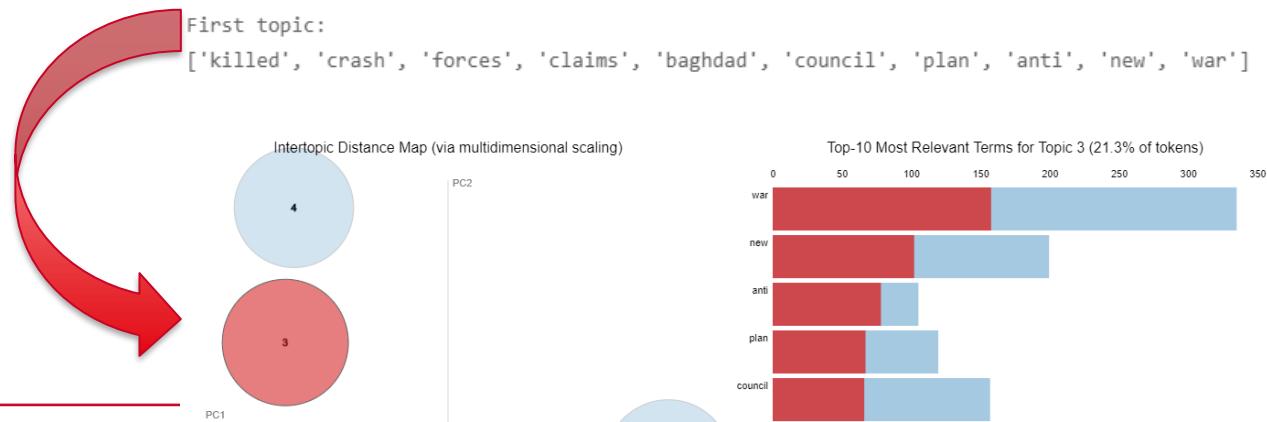
## ➤ LDA: understanding the results

- The optimal value of  $\lambda$  is context-dependent
  - The authors of this visualization tool got good results with  $\lambda = 0.6$
- So, what are the topics in our corpus?
  - It depends upon the data analyst to find similar characteristics between the documents of one cluster and assign it an appropriate label or topic
  - Consequently, it is extremely difficult to evaluate the performance of topic modeling since there are no right answers

# Topic Modeling

- LDA: understanding the results
  - The optimal value of  $\lambda$  is context-dependent
    - The authors of this visualization tool got good results with  $\lambda = 0.6$
  - When  $\lambda = 1$ , we get the same results we obtained before in Slide 18
    - **VERY IMPORTANT POINT:** the topic numbers in **LDAVis** are not the same as in the LDA model
  - Example:

The first topic of the LDA model is the 3<sup>rd</sup> in pyLDAvis



# Topic Modeling

---

- LDA: understanding the results
  - Before moving on, it is important to map each topic from our LDA model onto the topics in [pyLDAvis](#)
    - Set  $\lambda = 1$  and determine matches between the top 10 words
    - Example always build a table like this:

| LDA     | pyLDAvis |
|---------|----------|
| Topic 1 | Topic 3  |
| Topic 2 | Topic 5  |
| Topic 3 | Topic 2  |
| Topic 4 | Topic 1  |
| Topic 5 | Topic 4  |

# Topic Modeling

---

- LDA: understanding the results
  - Let's assume we agree on the following topics in our corpus

| LDA     | pyLDAvis | Topic Label   |
|---------|----------|---------------|
| Topic 1 | Topic 3  | Rally         |
| Topic 2 | Topic 5  | ?             |
| Topic 3 | Topic 2  | International |
| Topic 4 | Topic 1  | Trial         |
| Topic 5 | Topic 4  | Security      |

# Topic Modeling

---

- LDA: understanding the results
  - One can always evaluate the appropriateness of the topic labels and numbers by going back to the original data
    - Let's apply our LDA model to our DTM to assign a topic to each document

```
topic_values = LDA.transform(dtm)
```
  - The above command results in a distribution over topics for each documents
    - Example for the first document: [0.03 0.04 0.53 0.03 0.37]

# Topic Modeling

---

- LDA: understanding the results
  - Which label (topic) should we assign to the documents?
    - Traditional approach: the topic associated with the highest probability
  - Technical note: all the arrays we are working with today are of a special type from the **numpy** module
    - N-dimensional array (**ndarray**)
    - **ndarray** comes with several handy functions
      - **argmax**: returns the index of the maximum value in a row
      - **astype**: transforms all the elements of the array to a single type

```
max_topic_values = topic_values.argmax(axis=1)
```

```
max_topic_values = max_topic_values.astype("str")
```

# Topic Modeling

---

## ➤ LDA: understanding the results

- It is time now to replace topic labels
  - That is, instead of number, let's use meaningful words (see slide 25)
    - Recall that 0 is the first index in Python

```
max_topic_values[max_topic_values == "0"] = "Rally"
```

```
max_topic_values[max_topic_values == "1"] = "?"
```

```
max_topic_values[max_topic_values == "2"] = "International"
```

```
max_topic_values[max_topic_values == "3"] = "Trial"
```

```
max_topic_values[max_topic_values == "4"] = "Security"
```

| LDA     | Topic Label   |
|---------|---------------|
| Topic 1 | Rally         |
| Topic 2 | ?             |
| Topic 3 | International |
| Topic 4 | Trial         |
| Topic 5 | Security      |

# Topic Modeling

---

## ➤ LDA: understanding the results

- Now, we have topics, and they are appropriately labeled
- Let's merge these labels with our original, raw data
  - We will then be able to observe whether the labels/topics are good

```
data_w_topic = pandas.DataFrame()  
data_w_topic["Text"] = raw_data.iloc[0:10000,1]  
data_w_topic["Topic"] = max_topic_values
```

# Topic Modeling

---

- A trained LDA model can be used to extract topics from never-before-seen documents
  - Same procedure as before
    - Create a DTM with the new data
    - Apply the model to the DTM
    - Assign the topic with the highest probability to each document
  - Look at today's notebook for an example involving two documents that do not belong to the training data

# Topic Modeling

---

- Keep in mind that topic modeling is not a one-shot approach
  - Our results are nowhere close to be perfect
  - We could try the following:
    - Remove some more words (stop words)
    - Perform n-gram (2-gram)
    - Plot more than 10 documents per document
    - Increase the amount of data used for training
      - We only used 10,000
    - Increase the number of topics
      - We assumed only 5
    - Play with some arguments of LDA
    - ...

# Homework #8

---

- Let's retrain our LDA model by using a different corpus
  - This will not necessarily improve your model, but it will help you understand how to better prepare a corpus for LDA training
  - Task 1: create vector having 10000 headlines (same data we used today)
  - Task 2: add the word “man” to the stop list
    - Hint: `import sklearn.feature_extraction.text.ENGLISH_STOP_WORDS` and create a new stop list as follows: `new_stop_list = ENGLISH_STOP_WORDS.union(["man"])`
  - Task 3: create a `CountVectorizer` transformation using:
    1. The new stop list;
    2. 1- and 2-grams (parameter `ngram_range`),
    3. Using only words present in at most 80% of the corpus (parameter `max_df`)
    4. Using words present in at least 2 documents (parameter `min_df`)
    - Hint: read the documentation at [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_extraction.text.CountVectorizer.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html)

# Homework #8

---

- Let's retrain our LDA model by using a different corpus
  - Task 4: create a DTM
  - Task 5: fit an LDA model to the DTM having 5 topics
  - Task 6: use `pyLDAvis` to help you determine the topics inside each text
  - Task 7: answer the following questions on Canvas:
    1. Based on your analysis, what are the 5 resulting topics?
    2. You will likely have overlapping circles in the left panel produced by `pyLDAvis`. What does that mean?
    3. Why do you have terms that are made up of two words in the right panel of `pyLDAvis`?

# Summary

---

- We learned another technique to analyze textual data
  - Topic modeling
- Next class
  - Social media analytics and the midterm project

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 17 – Midterm Project

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Agenda

---

- Today: introduction to the midterm project problem
  - APIs, libraries, etc.
- Thursday, 10/21: Project consulting (optional)
  - No lecture. Why?
    - API keys will be shared among all
      - You might need extra time to complete the project because of limited API quota
      - I need extra time to monitor API usage and react to problems
    - I want to give you time to play and explore the Twitter API
      - It might (or might not) be useful in your final project
- Friday, 10/22 at 11:59 pm: Project deadline

# Lecture Objectives

---

- Quick review of Homework 8
- Learn about social media analytics
  - Focus on Twitter API
- Introduction to the midterm project

# Lecture Instructions

---

- Download the notebook “*Lecture 17.ipynb*” available on Canvas
  - Open the above file with VS Code

# Social Media Analytics

## ➤ Recall our first class

- Social media
  - Very rich data
    - Likes, emotions, comments, replies, photos, location, groups, ...
  - Highly **unstructured** data
    - Textual data, images, geolocation, ...
  - **Massive amounts** of data
    - *E.g.*, Facebook has over 2 billion users
    - Each one with her/his own photos, comments, likes, ...



# Social Media Analytics

---

- Social media analytics is the process of monitoring, collecting, and analyzing social media data
  - Usually driven by specific requirements from an application
- Goals:
  - Facilitate conversations and interactions between online communities
  - Extract useful patterns and insights to serve stakeholders
    - *E.g.*, active users, the underlying platform, external organizations, ...

# Social Media Analytics

---

- Our focus today will be primarily on monitoring social media
  - Because of social media companies have:
    - Strengthened their ability to engage in company-customer dialog
    - Created mechanisms for customer-customer dialog
    - Monitored and mediated the above dialogues
  - As a consequence, recent years have seen several new platforms focused on social media monitoring
    - E.g., Radian6 (now part of Sales Force), Pulsar, ...

# Social Media Analytics

---

- Those dialogues help companies in virtually all phases of a product or service life cycle

## 1. Product design

- Capture and understand conversations involving loyal customers, average customers, or (potential) new customers
- “Co-creation” of new products and/or services



- “I wish this product would do ...”
- Example: Del Monte Foods, Inc. created a new dog-food product in just six weeks by listening to dog owners
  - [https://www.youtube.com/watch?v=yP\\_3bpCPZaQ](https://www.youtube.com/watch?v=yP_3bpCPZaQ) (3 min)

# Social Media Analytics

---

- Those dialogues help companies in virtually all phases of a product or service life cycle

## 2. Product production (supply responsiveness)

- Businesses can anticipate significant changes in demand based on changing customer tastes and behavior
  - Adjust accordingly by ramping production up or down
- Example: Popeyes' Chicken Sandwich in 2019
  - Went viral on Twitter
  - Drove sales up 16%, rose profits 13%, and ...
  - ... Popeyes runs out of chicken sandwich



# Social Media Analytics

---

- Those dialogues help companies in virtually all phases of a product or service life cycle

## 3. Product utilization

- Who are the consumers buying the product?
  - Info from their social media profile
- What do the consumers think of the product/brand?
- Consumer support
- Our focus in the midterm project
  - What do “clients” think of #ISA414?

# Social Media Analytics

---

- Those dialogues help companies in virtually all phases of a product or service life cycle

## 4. Product disposal

- Build a “green” brand
  - Companies can engage in online conversations about disposal
  - Track consumer concerns
  - Suggest disposal places...
  - ... and replacement items

# Social Media Analytics

---

- Traditional techniques
  - Sentiment/Emotion analysis
    - Automatically extract user sentiment or emotion
  - Topic modeling
    - Automatically detect dominant themes or topics
  - Social network analysis
    - Used on a social network graph to understand its underlying structure and properties as well as to identify the relative importance of different nodes within the network

# Social Media Analytics

---

- Social media platforms: data collection via API
  - Facebook (Graph API): <https://developers.facebook.com/>
    - One must create an App and request permissions
      - Most are only granted to businesses
      - Process is incredibly cumbersome (since 2016/2017)
    - My request to access my list of friends via an API 😞

## `user_friends`

The `user_friends` permission allows your app to get a list of a person's friends using that app. The allowed usage for this permission is to provide Facebook-related content to personalize a person's experience. You may also use this permission to request analytics insights to improve your app and for marketing or advertising ...

● Request Rejected

## `/user/friends`

`/user/taggable_friends`

Requires Business or Individual Verification

Request

Here is what I can currently access:

## `public_profile`

The `public_profile` permission allows apps to read the Default Public Profile Fields on a User node. The allowed usage for this permission is to authenticate app users and provide them with a personalized in-app experience. This permission is automatically granted to all apps. You may also use this permission to request ...

● Live

## `email`

The `email` permission allows your app to read a person's primary email address. The allowed usage for this permission is to let end users log into your app with the email address associated with their Facebook profile. This permission is automatically granted to all apps. You may also use this permission to request ...

● Live

# Social Media Analytics

---

- Social media platforms: data collection via API
  - Reddit: <https://www.reddit.com/dev/api/>
    - Add .json or .xml to any Reddit page to get its content in JSON or XML

```
# posts from https://www.reddit.com/r/miamioh/
import requests
import pandas

response = requests.get("https://www.reddit.com/r/miamioh.json")
response = response.json()

df = pandas.DataFrame()

for post in response["data"]["children"]:
    json_to_row = pandas.json_normalize(post["data"])
    df = df.append(json_to_row)
```

# Social Media Analytics

---

- Social media platforms: data collection via API
  - Reddit: <https://www.reddit.com/dev/api/>
    - Add .json or .xml to any Reddit page to get its content in JSON or XML

```
# posts from https://www.reddit.com/r/miamioh/comments/oygda9/email_from_president_crawford_to_all_faculty
import requests
import pandas

response = requests.get("https://www.reddit.com/r/miamioh/comments/oygda9/email_from_president_crawford_to_all_faculty.json")
response = response.json()

df_post = pandas.json_normalize(response[0]["data"]["children"][0]["data"])
df_comments = pandas.DataFrame()

for post in response[1]["data"]["children"]:
    json_to_row = pandas.json_normalize(post["data"])
    df_comments = df_comments.append(json_to_row)
```

# Social Media Analytics

---

- Social media platforms: data collection via API
  - Twitter: <https://developer.twitter.com/>
    - One must create an App and answer several questions to receive an API key
      - “The core use case, intent, or business purpose for your use of the Twitter APIs.”
      - “If you intend to analyze Tweets, Twitter users, or their content, share details about the analyses you plan to conduct, and the methods or techniques.”
      - “If your use involves Tweeting, Retweeting, or liking content, share how you’ll interact with Twitter accounts, or their content.”
      - “If you’ll display Twitter content off of Twitter, explain how, and where, Tweets and Twitter content will be displayed with your product or service, including whether Tweets and Twitter content will be displayed at row level, or aggregated.”
      - ...
    - You are highly encouraged to get your own API keys
      - But I will share mine with all you (fingers crossed)
      - <https://developer.twitter.com/en/apply-for-access>

# Twitter API

## ➤ API limitations

### GET endpoints

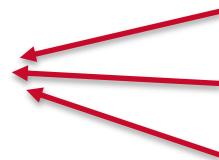
The standard API rate limits described in this table refer to GET (read) endpoints. Note that endpoints not listed in the chart default to 15 requests per allotted user. All request windows are 15 minutes in length. These rate limits apply to the standard API endpoints only, does not apply to premium APIs.

| Endpoint                          | Requests / window per user | Requests / window per app |
|-----------------------------------|----------------------------|---------------------------|
| GET account/verify_credentials    | 75                         | 0                         |
| GET application/rate_limit_status | 180                        | 180                       |
| GET favorites/list                | 75                         | 75                        |
| GET followers/ids                 | 15                         | 15                        |
| GET followers/list<br>...         | 15                         | 15                        |
| GET search/tweets                 | 180                        | 450                       |
| GET statuses/lookup               | 900                        | 300                       |
| GET statuses/mentions_timeline    | 75                         | 0                         |
| GET statuses/retweeters/ids       | 75                         | 300                       |
| GET statuses/retweets_of_me       | 75                         | 0                         |
| GET statuses/retweets/:id         | 75                         | 300                       |

# Twitter API

- Twitter authentication is slightly more complex than what we have seen so far
  - You have to sign each API request by passing several generated keys and tokens in an authorization header
  - You can generate the keys and tokens after your API access request is accepted

App using  
Twitter API  
consumer\_key  
consumer\_secret



Twitter users accessing Twitter  
data through the API/App  
access\_token  
access\_token\_secret } One set per user

## API key and secret:

1. consumer\_key
2. consumer\_secret

Think of these as the username and password that represents the App when making API requests.

## Access token and secret:

1. access\_token
2. access\_token\_secret

An access token and access token secret specific to the Twitter account the request is made on behalf of.

# Twitter API

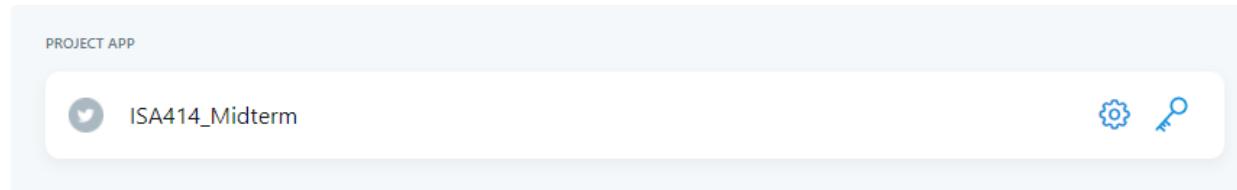
---

- How are we going to do in this course?
  - I have already created an app

Project 1 >

V1.1 ACCESS

V2 ACCESS

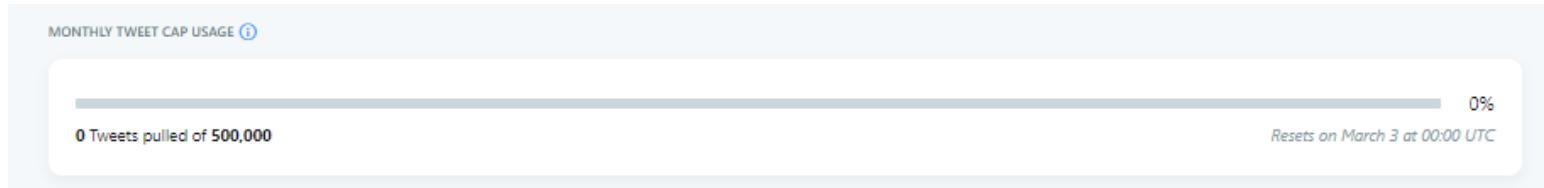


- We will all act as if we are a single Twitter user
  - @arthurc50170909
    - No, this is not my official account
  - This is easier than asking each of you to create a Twitter account

# Twitter API

---

- I will be monitoring your usage
  - Individual (via IP addresses) and collective



- We will use the module [tweepy](#) to help accessing the API

[pip install tweepy](#)

# Twitter API

---

- Now, it is time to define our API keys
  - **YOU ARE NOT ALLOWED TO SHARE THESE KEYS WITH ANYONE OUTSIDE THE ISA 414/514 COURSE**

```
import tweepy
```

```
consumer_key = "zs0yFmnhHnfjnIS3eOXU9AtJU"
```

```
consumer_secret = "dbsGwYqw4jxgpiuY1kQH5Yf4dECdk9JCsJVAKaYH1ExUAoVz9B"
```

```
access_token = "1357013845647101955-t5k1nAOzGIKb6oohfw99I7F2aOVIJp"
```

```
access_token_secret = "DkgJun3wp8zdVSmUDr6rEYF82deUyr9ElipjcnmFPzDzl"
```

- Next, authenticate

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
```

```
auth.set_access_token(access_token, access_token_secret)
```

```
api = tweepy.API(auth)
```

# Twitter API

---

- Checking the API limitations
  - We are primarily concerned with searches

```
import time
response = api.rate_limit_status()
remaining_searches = response['resources']['search'][ '/search/tweets' ]
```

## Twitter API v2 rate limits

The following table lists the rate limits for the Twitter API v2 with Standard and Academic Research product tracks at the Basic access level. These rate limits are also documented on each endpoint's API Reference page and also displayed on the developer dashboard.

| Resource      | Endpoint                                           | Requests per 15-minute window unless otherwise stated |          |
|---------------|----------------------------------------------------|-------------------------------------------------------|----------|
|               |                                                    | Per App                                               | Per user |
| Tweets        | Tweet lookup                                       | 300                                                   | 900      |
|               | Timelines                                          |                                                       |          |
|               | - User Tweet timeline                              | 1500                                                  | 900      |
|               | - User mention timeline                            | 450                                                   | 180      |
| Search Tweets | Recent search                                      | 450                                                   |          |
|               | Full-archive search                                | 300                                                   |          |
|               | Full-archive also has a 1 request / 1 second limit |                                                       | 180      |

<https://developer.twitter.com/en/docs/twitter-api/rate-limits>

# Twitter API

---

## ➤ Searching tweets

- It returns a number of tweets based on a specified term
- **It only returns data from the past 7 days**
  - One must have a premium (*i.e.*, paid) account to go beyond that
- Example: searching for the latest 100 tweets having Microsoft  
`search_results = api. search_tweets(q="Microsoft", lang="en", count=100)`
- Note that the results are in JSON
  - The lecture's notebook explains how to convert JSON to data frame

# Twitter API

---

## ➤ What else can we do with the API?

- Really a lot
  - See <https://docs.tweepy.org/en/stable/api.html>
- Example: retrieving the IDs of 'miamiuniversity' followers

```
search_results = api.get_followers(screen_name = "miamiuniversity", count = 100)
```
- Note that the results are in JSON
  - The lecture's notebook explains how to convert JSON to data frame

# Twitter API

---

- What else can we do with the API?
  - Trending topics
    - One can collect the trending topics per region
      - Only certain regions are available
    - Obtaining the locations  
`api. available_trends()`
    - Obtaining the trend topics in Cincinnati  
`api. get_place_trends(2380358)`

# Twitter API

---

- What else can we do with the API?
  - Obtaining tweets from a user's timeline
    - *E.g.*, Microsoft

```
search_results = api.user_timeline("microsoft", count = 1000, exclude_replies = True)
```

- Note that the results are in JSON
  - The lecture's notebook explains how to convert JSON to data frame

# Twitter API

---

## ➤ What else can you do with Twitter API?

- Twitter ads API

- Not covered in this course
- Programmatically create, schedule, and manage ad campaigns to engage people on Twitter
- Create and manage tailored audiences using Twitter, web or mobile data
- Granular insights of ad campaigns by a full range of metrics

### Twitter API

The Twitter API enables programmatic access to Twitter in unique and advanced ways. Use it to analyze, learn from, and interact with Tweets, Direct Messages, users, and other key Twitter resources.

### Twitter Ads API

The Twitter Ads API connects developers to Twitter's advertising platform. Build solutions to meet the needs of advertisers around the world.

# Twitter API

---

- One last word about Twitter API
  - The [tweepy](#) package is incredibly useful
    - Manages crazy OAuth authentication
    - Handles API requests and parses API responses
  - ... but can you see the information security issue?
    - Behind the scenes, what is [tweepy](#) doing with our keys and data?
    - Is it theoretically possible for the package to be collecting and sending that information to a third-party?
      - Yes, but you can always check the package's source code

# Project Description

---

- Goal: monitor social media
  - What are the students saying about ISA 414?
    1. Continually search for tweets having term #ISA414
    2. Calculate the sentiment behind new tweets in real-time
- For more details, see the project description on Canvas
- Groups are now available on Canvas

# Summary

---

- We learned about social media analytics
  - Focus on Twitter API
- Next lecture: in-class project work (optional)

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent.

# ISA 414 – Managing Big Data

## Lecture 19 – Cloud Computing and Storage

(Part I)

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Announcements

---

- Final course project
  - Description is available on Canvas
  - Groups of 4
    - I will form the groups later
    - Your preferences will be taken into account
  - Sample reports are available on canvas
    - Keep in mind that the requirements have changed over the years

# Announcements

---

- Final course project
  - Phase 1: presentation of initial ideas
    - November 16<sup>th</sup> and November 18<sup>th</sup>
    - Each presentation should last for 15 (ISA 414)/20 (ISA 514) minutes, leaving about 5 minutes for questions
    - Each presentation should cover the following:
      - Business background
      - Problem definition
      - Proposed solution: a bird's-eye view of the proposed solution
      - Potential pros and cons (e.g., anticipated problems)

# Announcements

---

- Final course project
  - Phase 2: work on project + report
    - In-class group work on November 23<sup>rd</sup> and November 30<sup>th</sup>
    - There is no final presentation
    - Main deliverable: a report that identifies the relevant problem and describes the proposed solution
      - Deadline: December 5th
    - Several constraints and requirements
      - See the project description on Canvas

# Lecture Objectives

---

- Review the midterm project
- Learn some basic concepts related to cloud computing and storage
  - Different cloud service models
    - IaaS – Infrastructure as a Service
    - PaaS – Platform as a Service
    - SaaS – Software as a Service

# Lecture Instructions

---

- Download the notebook “*Lecture 19.ipynb*” available on Canvas
  - Open that file with VS Code

# Big Data

---

## ➤ Managing big data

- This course:
  1. Past: **techniques** to effectively collect and analyze potentially unstructured data sets
  2. Present: big-data enablers, *i.e.*, modern **technologies** to address the challenges brought by big data
    - Cloud computing and storage
    - Distributed file systems (HDFS)
    - Parallel, distributed computing paradigms (Spark)

# Big Data

---

- The volume and velocity aspects of big data bring new, complex challenges
  - How to store the data?
  - How to quickly capture and analyze the data?
- Several companies are embracing the challenges
  - Recent years have seen the rise of several powerful technologies and paradigms
    - *E.g.*, cloud computing/storage and Hadoop

# Cloud Technologies

---

- For our purposes:
  - Cloud = IT infrastructure and applications on rent over the internet (*i.e.*, “*someone else's computer*”)
- Let's consider a few things first before talking about cloud service models
  - To a certain degree, IT infrastructure is a commodity
    - Setting up such an infrastructure in-house requires:
      - (Disposable) Hardware
        - Computing devices
        - Storage devices
        - Network
        - ...

# Cloud Technologies

---

- For our purposes:
  - Cloud = IT infrastructure and applications on rent over the internet (*i.e.*, “*someone else's computer*”)
- Let's consider a few things first before talking about cloud service models
  - To a certain degree, IT infrastructure is a commodity
    - Setting up such an infrastructure in-house requires:
      - People (experts)
        - IT/management team
        - Security team
      - High up-front capital investment

# Cloud Technologies

---

- Maintaining an in-house IT infrastructure is neither cheap nor easy
  - Disposable hardware
    - A new computer today will be a mediocre computer in 3 years and an old computer in 6 years
  - People move
    - Jobs in high demand = many new offers
    - Costly training process

# Cloud Technologies

---

- Maintaining an in-house IT infrastructure is neither cheap nor easy
  - Many startups or companies growing too fast often struggle keeping up with a modern infrastructure
- Alternative: cloud computing and storage
  - On-demand infrastructure
    - Computation anytime anywhere – whenever there is a demand
  - Pay to use someone else's IT infrastructure
    - Rental service: rent what you want, and return upon usage

# Cloud Technologies

---

## ➤ Analogy

- Would you buy (or build) a truck every time you have to move a piece of furniture?
  - Do you have the resources and/or skills?
  - Cost/benefit tradeoff
- Would it make more sense to rent a truck, and return it when the moving is done?



## ➤ Similarly, why building an IT infrastructure when you can rent one?

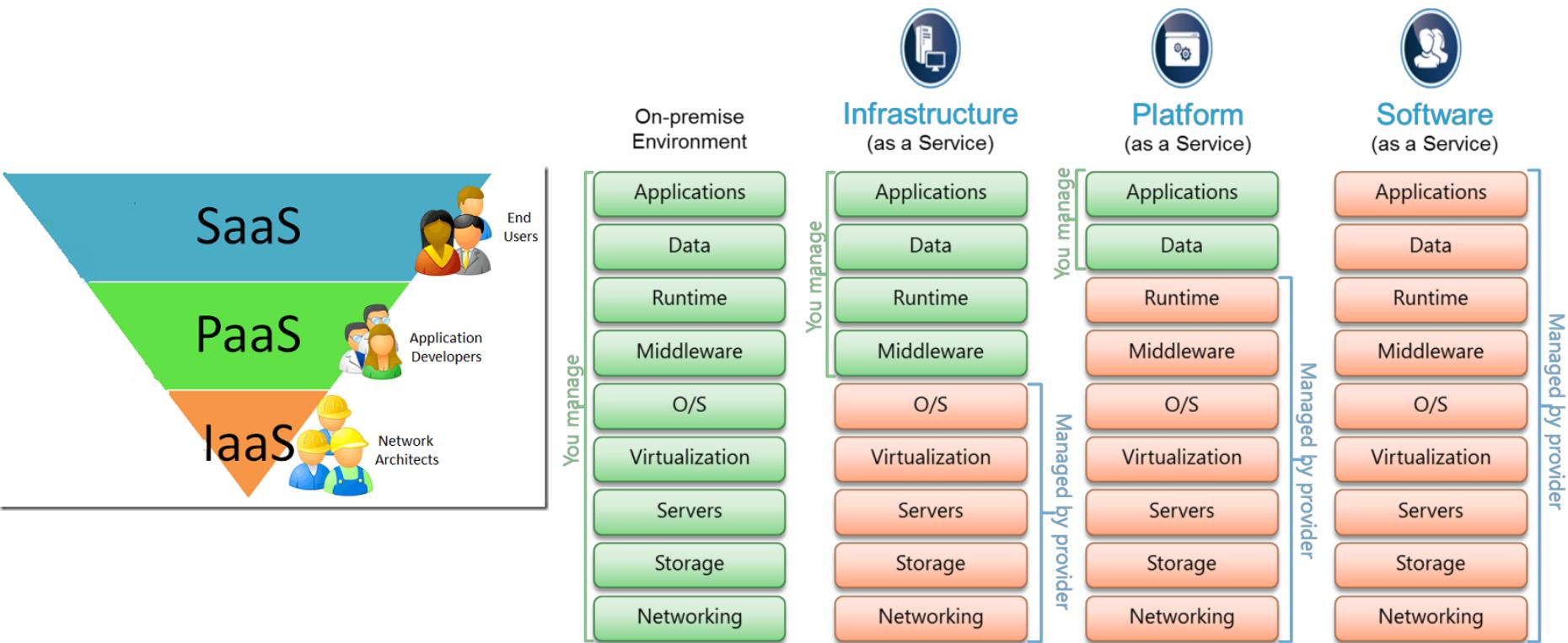


# Cloud Technologies

---

- So far, we have focused our discussion on IT infrastructure
  - Infrastructure as a Service(IaaS)
  - Other cloud service models include renting:
    - Platform (Platform as a Service)
    - Application (Software as a Service)
  - Bottom line: there are different business models around the concept of cloud with different levels of engagement and servicing
    - Similar to rental agreements

# Cloud Service Models

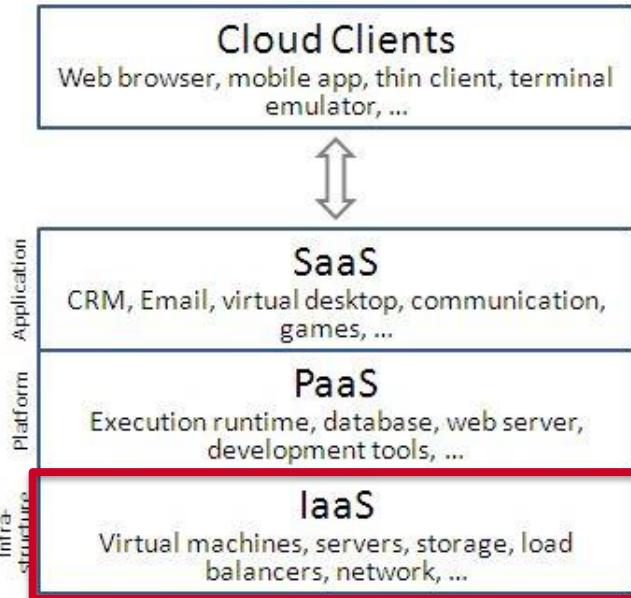


# IaaS

---

## ➤ IaaS: Infrastructure as a Service

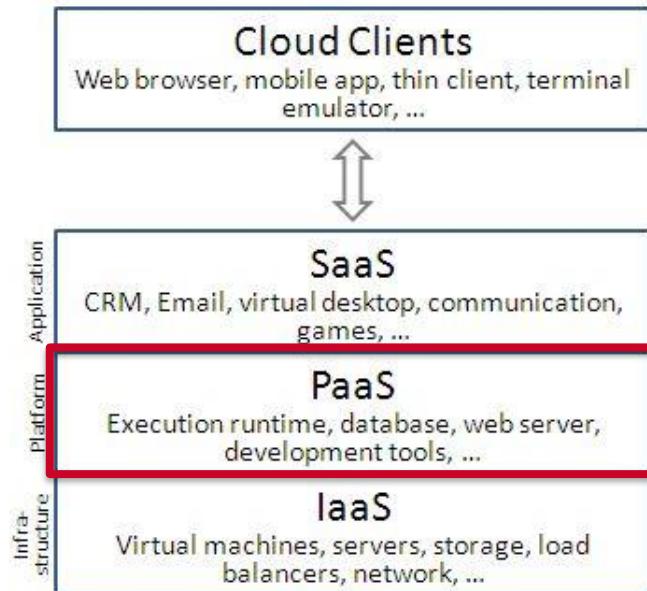
- Service: hardware only
  - Bare minimum rental service
- The client still has to install and maintain any required software or service



# PaaS

## ➤ PaaS: Platform as a Service

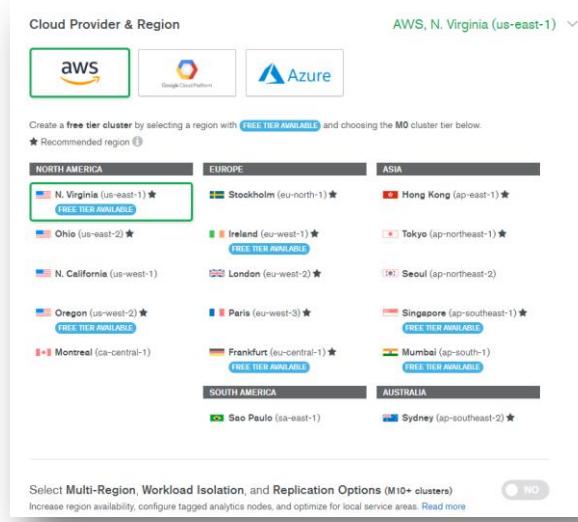
- Service: computing environment
  - Might include:
    - Programming languages and development tools
    - Web/database servers



# PaaS

## ➤ Example: cloud-based MongoDB

- Recall Lecture 11
  - MongoDB database provided by MongoDB company

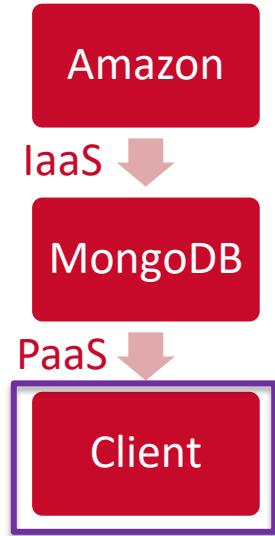


# PaaS

---

## ➤ Example: cloud-based MongoDB

- A client pays MongoDB for a database
  - PaaS (no need to think about OS, servers, *etc.*)
- The client must still manage the database
  - *E.g.*, create users, set privacy constraints, *etc.*
- The client can build apps/software that use the database (*e.g.*, Assignment 3)
  - No need to worry about the hardware infrastructure
  - Buy more storage space from MongoDB (company) as required

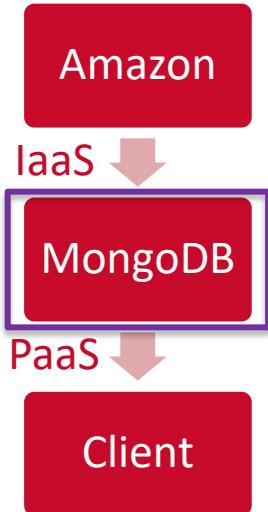


# PaaS

---

## ➤ Example: cloud-based MongoDB

- MongoDB (company) pays Amazon for using their computers, storage device, network
  - IaaS
- MongoDB (company) manages the database infrastructure
  - Monitor storage capacity, database speed, security
  - Buy more storage and computational power from Amazon when needed (e.g., too many new clients)

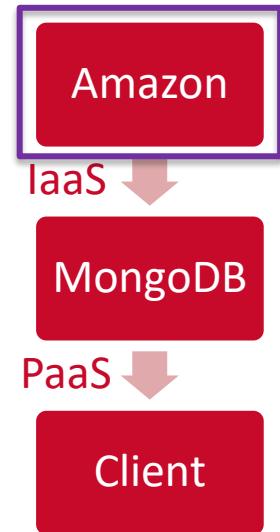


# PaaS

---

## ➤ Example: cloud-based MongoDB

- Amazon
  - Manages the hardware
    - Computers, storage devices, network, ...
- Business model: monthly payment
  - MongoDB (company) is the “man in the middle”
    - Easy way of making money



# PaaS

Available in most AWS regions with prices (in USD) varying across regions. Prices below are for the US East (N. Virginia) region.

## ➤ Example

- Pricing scheme:  
(as of 07/14/2021)

### DEDICATED CLUSTER

#### Standard

This line offers the most economical plans for production applications running on AWS.

Each Replica Set comes standard with 2 data-bearing nodes and high-availability via auto-failover.

|    | RAM    | vCPUs | Storage | Monthly price |
|----|--------|-------|---------|---------------|
| M1 | 2 GB   | 1     | 40 GB   | \$180         |
| M2 | 4 GB   | 2     | 60 GB   | \$360         |
| M3 | 8 GB   | 2     | 120 GB  | \$720         |
| M4 | 16 GB  | 4     | 240 GB  | \$1,430       |
| M5 | 32 GB  | 8     | 480 GB  | \$2,260       |
| M6 | 64 GB  | 16    | 700 GB  | \$3,520       |
| M7 | 122 GB | 16    | 700 GB  | \$4,540       |

### DEDICATED CLUSTER

#### High Storage

This line offers a higher storage-to-RAM ratio than our Standard line and is geared towards applications that need to store large amounts of data but have more modest performance requirements.

Each Replica Set comes standard with 2 data-bearing nodes and high-availability via auto-failover.

|    | RAM    | vCPUs | Storage | Monthly price |
|----|--------|-------|---------|---------------|
| M1 | 2 GB   | 1     | 75 GB   | \$220         |
| M2 | 4 GB   | 2     | 150 GB  | \$440         |
| M3 | 8 GB   | 2     | 300 GB  | \$880         |
| M4 | 16 GB  | 4     | 600 GB  | \$1,760       |
| M5 | 32 GB  | 8     | 1 TB    | \$2,730       |
| M6 | 64 GB  | 16    | 1 TB    | \$3,790       |
| M7 | 122 GB | 16    | 1 TB    | \$4,810       |

### DEDICATED CLUSTER

#### High Performance

This line is designed for applications with the most demanding workloads, featuring local SSD storage with very high throughput and extremely low latency.

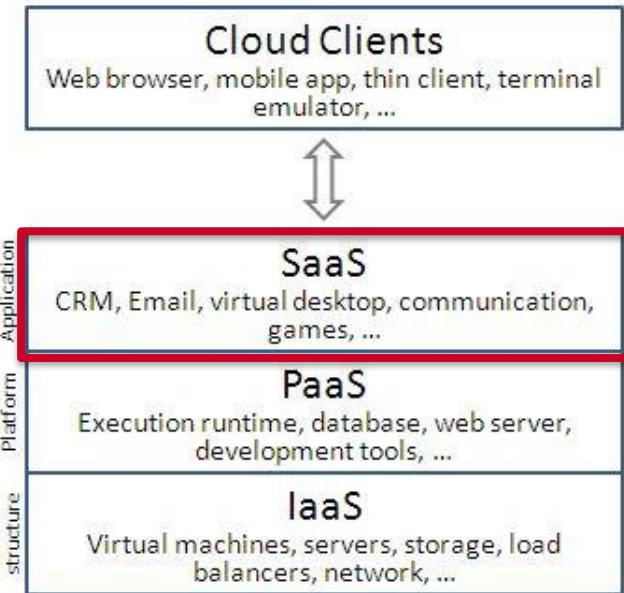
Each Replica Set comes standard with 2 data-bearing nodes and high-availability via auto-failover. A hidden 3rd node is included for increased durability and backups.

|    | RAM    | vCPUs | Storage | Monthly price |
|----|--------|-------|---------|---------------|
| M5 | 30 GB  | 4     | 850 GB  | \$3,750       |
| M6 | 60 GB  | 8     | 1 TB    | \$5,890       |
| M7 | 122 GB | 16    | 1 TB    | \$10,130      |

# SaaS

---

- **SaaS: Software as a Service**
  - Service: full software/service on demand
  - Cloud service provider takes care of the whole environment
    - Hardware + software
  - Example:
    - Dropbox, Google Docs, Microsoft Office online



# XaaS

---

- XaaS: anything as a service
  - Marketing aaS
    - *E.g.*, IBM Watson Marketing Insights
  - Analytics aaS
    - *E.g.*, IBM Watson Analytics
  - ...

---

## Case Study

Sentiment Analysis with Natural Language  
Understanding: A SaaS Provided by IBM Cloud

# IBM Cloud

---

- There are many cloud platforms that offer different SaaS, PaaS, IaaS
  - Microsoft Azure
  - IBM Cloud (formerly, IBM Bluemix)
  - Google Cloud
  - Amazon Web Services (AWS)
- We will be focusing on IBM Cloud
  - Until a few weeks before the beginning of the course, IBM Cloud was the only provider that did not require credit card info when creating free accounts
  - The above has changed ☹

# IBM Cloud

- All the services provided by IBM Cloud
  - We will explore them in our next class

The screenshot shows the IBM Cloud catalog interface. On the left, there's a sidebar with a dark background containing a navigation menu. The 'Software' section is currently selected. Below it, under 'Category', 'AI / Machine Learning' is checked. Other categories like 'Compute', 'Networking', 'Storage', 'Analytics', 'Databases', 'Developer Tools', 'Logging and Monitoring', 'Integration', 'Security', and 'Mobile' are also listed. At the bottom of the sidebar, there are sections for 'Cloud Paks' and 'Helm charts'. The main content area has a light gray background. It features a heading 'Software' with a sub-instruction 'Explore our expanding catalog of software solutions and take advantage of a simplified installation process.' Below this, there's a filter bar with 'AI / Machine Learning' selected and a 'Clear all' button. A section title 'AI / Machine Learning 6 items' is followed by a grid of six service cards. Each card includes a small icon, the service name, a brief description, and deployment options like 'Helm charts', 'IBM Kubernetes Service', and 'Free'. The services listed are Apache MXNet (Incubating), Natural Language Understanding Node.js App, PyTorch, Spark, TensorFlow Serving, and OVA Images.

| Service                                    | Description                         | Deployment Options                          | Status     |
|--------------------------------------------|-------------------------------------|---------------------------------------------|------------|
| Apache MXNet (Incubating)                  | Third party • AI / Machine Learning | Helm charts • IBM Kubernetes Service • Free | Incubating |
| Natural Language Understanding Node.js App | IBM • AI / Machine Learning         |                                             |            |
| PyTorch                                    | Third party • AI / Machine Learning | Helm charts                                 |            |
| Spark                                      | Third party • AI / Machine Learning | Helm charts • IBM Kubernetes Service • Free |            |
| TensorFlow Serving                         | Third party • AI / Machine Learning | OVA Images                                  |            |
| OVA Images                                 | vCenter Server                      | Free                                        |            |

# Natural Language Understanding

---

- Let's experience one IBM Cloud's service
  - Natural Language Understanding
    - Sentiment analysis
  - Go to <https://www.ibm.com/demos/live/natural-language-understanding/self-service> for a demo
    - We will soon request this service using Python
      - REST + JSON

# Natural Language Understanding

---

## ➤ Business model:

- Payment per call (+ tailored model)

(as of 07/14/2021)

### Natural Language Understanding

Prices displayed for region: Dallas

#### Lite

30,000 NLU Items Per Month

1 Custom Model

Fixed API Rate Limit. See Standard plan for higher API Rate Limit

NOTE: A NLU item is based on the number of data units enriched and the number of enrichment features applied. A data unit is 10,000 characters or less. For example: extracting Entities and Sentiment from 15,000 characters of text is (2 Data Units \* 2 Enrichment Features) = 4 NLU Items. A custom model refers to an annotation model developed with Watson Knowledge Studio.

The Lite plan gets you started with 30,000 NLU Items per month at no cost. This plan also enables use of one custom model published through Watson Knowledge Studio.

#### Standard

Unlimited NLU Items Per Month

You will be charged per NLU Item & per Custom Model

Increased API Rate Limit can be configured upon request

NOTE: A NLU item is based on the number of data units enriched and the number of enrichment features applied. A data unit is 10,000 characters or less. For example: extracting Entities and Sentiment from 15,000 characters of text is (2 Data Units \* 2 Enrichment Features) = 4 NLU Items. A custom model refers to an annotation model developed with Watson Knowledge Studio.

This plan allows unlimited NLU Items and adds the ability to use multiple custom models published through Watson Knowledge Studio.

| TIERS               | PRICING                                      |
|---------------------|----------------------------------------------|
| 1 - 1               | \$800.00 USD/Custom Model Instance per Month |
| 2 - 250,000         | \$0.003 USD/NLU Item                         |
| 250,001 - 5,000,000 | \$0.001 USD/NLU Item                         |
| 5,000,000+          | \$0.0002 USD/NLU Item                        |

#### Premium Tier1

Usage and Training Data is Private + Stored in an Isolated Single Tenant Environment

Transaction logging for service improvement is disabled by default

High Availability and Service Level Agreements on Uptime

IBM Cloud Service Endpoints

For more info or to purchase a premium plan, contact us at <https://ibm.biz/contact-wdc-premium>

# Natural Language Understanding

---

- There are 9 features one can extract from a text using NLU
  - Entities
  - Keywords
  - Concepts
  - Relations
  - Sentiment
  - Emotion
  - Categories
  - Semantic Roles
  - Syntax

The screenshot shows a user interface for NLU. At the top, there is a dark gray header with white text. Below it is a main content area with a dark background and white text. A red rectangular box highlights the navigation bar at the bottom of the content area. The navigation bar has five tabs: Extraction (which is selected), Classification, Linguistics, and Custom. Underneath the tabs, there is a row of buttons labeled Entities, Keywords, Concepts, and Relations. The Entities button is also highlighted with a red box.

Under the IBM Board Corporate Governance Guidelines, the Directors and Corporate Governance Committee and the full Board annually review the financial and other relationships between the independent directors and IBM as part of the assessment of director independence. The Directors and Corporate Governance Committee makes recommendations to the Board about the independence of non-management directors, and the Board determines whether those directors are independent. In addition to this annual assessment of director independence

■ Entities (Out of the Box Model)

# Natural Language Understanding

---

- API (<https://cloud.ibm.com/apidocs/natural-language-understanding>)
  - Request: REST protocol
    - GET operation
  - Response: JSON file
  - The API's documentation shows examples using *curl* and a specific Python module
    - We shall instead use the [requests](#) module we are already familiar with

# Natural Language Understanding

---

- We will be using the same API keys today
  - You will learn how to get your own API keys in our next class
  - Please, do not overuse the provided API keys

```
base_url = "https://api.us-south.natural-language-
understanding.watson.cloud.ibm.com/instances/d6058b89-d39d-464c-a756-50658dd3124b"
```

```
api_key = "LeNnOkIAOfA5VBWG6B7luAFzEJn4Q-z24AqSrzHaAGuG"
```

# Natural Language Understanding

---

- Let's use the following text as example

content = "She's got a smile that it seems to me  
reminds me of childhood memories  
where everything was as fresh as the bright blue sky"

- We are going to send this text to IBM Cloud  
using a REST request
  - Note how there is no preprocessing

# Natural Language Understanding

---

- Retrieving sentiment
  - REST request: GET operation with authentication
    - How do I know this? From IBM Cloud's documentation

```
key_values = {'version': '2021-08-01', 'text': content, 'features':'sentiment'}  
response = requests.get(base_url+"/v1/analyze", key_values, auth = ('apikey', api_key))  
  
response.json()
```

# Natural Language Understanding

---

- Retrieving sentiment, keywords, emotion, and concepts
  - Make sure that there is no blank space in between the “features”

```
key_values = {'version': '2021-08-01', 'text': content, 'features':'sentiment,emotion,keywords'}  
response = requests.get(base_url+"/v1/analyze", key_values, auth = ('apikey', api_key))  
  
response.json()
```

# Natural Language Understanding

---

- It is relatively easy to use cloud services to perform sentiment analysis
  - Arguably easier than what we did before using the bag-of-words approach
- Google, Microsoft, and Amazon, also offer similar services
  - How good are these services?
    - Search for the paper “*Off-the-Shelf Technologies for Sentiment Analysis of Social Media Data: Two Empirical Studies*” written by a previous ISA 414 student
  - How do they work?
    - Who knows ... (proprietary technologies)

# Summary

---

- We learned about different cloud services
  - IaaS, PaaS, SaaS, XaaS
- We learned about IBM Cloud
  - Can you think of interesting ways of applying this in your project?
- Interesting story: Netflix and the cloud
  - <https://media.netflix.com/en/company-blog/completing-the-netflix-cloud-migration>
- Next lecture: cloud computing and storage (part II)
  - Business considerations and hands-on time with some SaaS

# ISA 414 – Managing Big Data

## Lecture 20 – Cloud Computing and Storage

*(Part II)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Lecture Objectives

---

- Understand some business issues related to the use of cloud technologies
- Hands-on experience with some cloud services
  - Software as a Service
    - IBM Natural Language Understanding
    - Microsoft Azure Computer Vision

# Lecture Instructions

---

- Download the notebook “*Lecture 20.ipynb*” available on Canvas
  - Open that file with VS Code
- Download the file *iamaperson.jpg* from Canvas
- Make sure to add all the files to the same folder

# Lecture Instructions

---

- (Optional) Go to <https://cloud.ibm.com/login>
  - Click on “Create an account”
  - Fill in the application and create an account
    - To avoid fake accounts, you will be asked for your credit card info
      - That is why creating an account is optional
  - Check your email and confirm the registration
  - Sign in
  - After login, click on *Catalog* (top right)



# Cloud Service Models

---

- Cloud computing and storage
  - Crucial big-data analytics enabler
- Cloud = rental agreement (on demand)
  - Different business models
    - Different levels of engagement
    - Different levels of servicing
- Traditional cloud service models include:
  - Application (SaaS)
  - Platform (PaaS)
  - Infrastructure (IaaS)

# Cloud Service Models

---

- How do cloud technologies enable big-data analytics?
  - Infrastructure (IaaS)
    - Mitigate the issue of purchasing/maintaining/updating hardware
      - This is “outsourced” to the cloud provider
    - An organization can easily expand/reduce the IT infrastructure
      - Resources are rented as needed
      - No waste of computational resources
  - Platform (PaaS)
    - Focus on managing and using data-analytics platforms/environments
    - No time wasted on maintaining the physical devices running the platform
  - Application (SaaS)
    - Has the potential to speed up the development of data-analytics solutions
    - Focus on using an application

# Cloud Service Models

---

- We will not have hands-on experience with IaaS in this course
  - Too many hardware/IT details that are beyond the scope of this course
- We experienced PaaS before with MongoDB cloud
- We shall now focus on SaaS tailored to (big) data analytics
  - AI services on IBM Cloud and Microsoft Azure

# Cloud Service Models

---

- Important business discussion
  - In-house vs cloud infrastructure
    - Considerations
      - Costs
      - Demand
      - Skills
      - Security

# Cloud Service Models

---

## ➤ Cost: in-house (on-premise) infrastructure

- Initial purchase and future upgrades (3-, 5-, 10-year plan)
  - Tough question: how to estimate the hardware needs?
    - Common approach: peak analysis
    - Example: previous ISA 414 cluster
      - 35 students working on 10 GB data sets simultaneously = 350 GB of RAM
      - 2 cores per student = 70 cores
- Real-state costs (machine room)
- Electricity bill (keep the machines and the cooling system on)
- Staff costs

# Cloud Service Models

---

## ➤ Cost: cloud

- Usually, cost per usage time
  - Cloud-service providers often have great cost calculators
    - Remove most of the cost-related uncertainty
- How to estimate the hardware needs?
  - One does not need to when using the cloud
  - Infrastructure is easily (and often automatically) adjustable
    - *E.g.*, one can often increase storage and computational power in a few clicks (or even have this automatically adjusted as needed)

# Cloud Service Models

---

## ➤ Demand

- Key question: will the infrastructure be heavily utilized?
  - Example: does it make sense to buy powerful machines to be used by ISA 414/514 students?
    - ISA 414/514 is taught twice a year (8 months)
- Underutilization
  - Often (not always) means unjustified investments
  - Utilization is not so easy to estimate
- Cloud infrastructure is likely a better option if the infrastructure will be underutilized now or in the future
  - Remember that cloud = pay as you need

# Cloud Service Models

---

## ➤ Skills

- Is there an in-house team capable of setting up and maintaining the infrastructure?
  - IT, security, database, network professionals
- If not, how much will hiring these professionals cost?
  - See the cost discussion
- Benefits of having an in-house, talented team:
  - Exclusivity
- Cloud also means “outsourcing” the maintenance team
  - Do you trust the cloud provider?

# Cloud Service Models

---

## ➤ Security

- How will the cloud provider secure the infrastructure (including stored data)?
  - Not only about protection against internal/external hackers
  - Where is the server located?
    - Where is the *cloud*?
      - Would you be comfortable with your bank storing data about you in a 3<sup>rd</sup> party computer?
    - Geopolitics matter
      - Some of the cloud platforms allow users to select servers' locations

# Cloud Service Models

---

- In-house vs cloud infrastructure
  - It really is context dependent
    - There is no bullet-proof solution
  - Good news: one does not have to choose one over another
    - Hybrid of cloud and in-house infrastructure is possible
      - Keep the crucial, sensitive data in an in-house database; send the rest to the cloud
      - Example: Best Buy buys extra storage capacity and processing power during Black Friday and Cyber Monday

.....

# SaaS Case Study #1: Text Analytics

# Cloud Services

---

- Question: can we automate the analysis of sentiment and emotions in textual data?
  - Yes, we did it before!
    - We learned how to preprocess textual data using the bag-of-words approach
    - We learned how to calculate sentiment scores based on counts of positive and negative words
      - Very technical, yet naïve approach
  - Let's do it differently now
    - SaaS: IBM Cloud's Natural Language Understanding
    - <https://www.ibm.com/demos/live/natural-language-understanding/self-service/home>

# Lecture Instructions

---

- Let's experience some SaaS now
  - Search for the term “Natural Language Understanding”

The screenshot shows the IBM Cloud search interface. On the left, there is a sidebar with navigation links: Catalog, IBM Cloud catalog, Featured, Services, Software, and Consulting. The 'Featured' link is currently selected, highlighted in blue. The main area has a search bar at the top containing the text "Natural language Understanding". Below the search bar, there are two sections: "Resource Results" and "Catalog Results". In the "Resource Results" section, there is one item: "Natural Language Understanding-fy Service". In the "Catalog Results" section, there are three items: "Knowledge Studio", "Natural Language Understanding", and "Natural Language Understanding Node.js App". The "Natural Language Understanding" item in the catalog results is also highlighted with a red box, matching the one around the search term in the search bar.

IBM Cloud

Natural language Understanding

Resource Results

View all resource results

Natural Language Understanding-fy  
Service

Catalog Results

View all catalog results

Knowledge Studio

Natural Language Understanding

Natural Language Understanding Node.js App

# Lecture Instructions

---

- Click on *Natural Language Understanding*
  - Select the region closest to where you are
  - Select Lite and click on “Create”

The screenshot shows the "Create" page for the Watson Natural Language Understanding service. At the top, there are two tabs: "Create" (which is selected) and "About". Below the tabs, a section titled "Select a location" contains a dropdown menu with "Dallas (us-south)" selected. This dropdown is highlighted with a red rectangle. Below this, another section titled "Select a pricing plan" is shown, with a note that prices do not include tax and are for the United States. A table compares the "Lite" plan with other options. The "Lite" row is also highlighted with a red rectangle. The table has columns for "Plan", "Features", and "Pricing". The "Lite" plan includes 30,000 NLU Items Per Month, 1 Custom Model, and a Fixed API Rate Limit. It is described as "Free". The "Standard" plan is listed as having a higher API Rate Limit. The "Watson" plan is listed as being based on Watson Knowledge Studio. At the bottom of the page, a note states that the Lite plan gets you started with 30,000 NLU Items per month at no cost, and that services are deleted after 30 days of inactivity.

| Plan     | Features                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Pricing |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| Lite     | <p>30,000 NLU Items Per Month<br/>1 Custom Model<br/>Fixed API Rate Limit. See Standard plan for higher API Rate Limit</p> <p>NOTE: A NLU item is based on the number of data units enriched and the number of enrichment features applied. A data unit is 10,000 characters or less. For example: extracting Entities and Sentiment from 15,000 characters of text is (2 Data Units * 2 Enrichment Features) = 4 NLU Items. A custom model refers to an annotation model developed with Watson Knowledge Studio.</p> | Free    |
| Standard | Higher API Rate Limit                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |         |
| Watson   | Based on Watson Knowledge Studio                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |         |

The Lite plan gets you started with 30,000 NLU Items per month at no cost. This plan also enables use of one custom model published through Watson Knowledge Studio.  
Lite plan services are deleted after 30 days of inactivity.

# Lecture Instructions

---

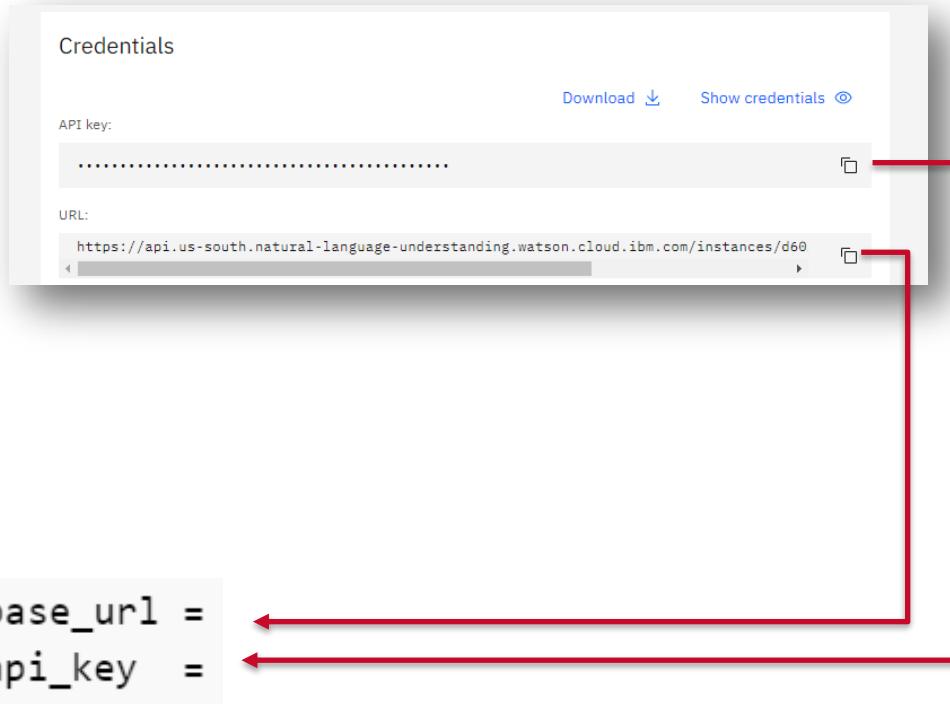
- Click on *Manage*
- Copy the API Key and URL to a safe place

The screenshot shows the IBM Watson Natural Language Understanding service management interface. At the top, it displays the service name "Natural Language Understanding-fy" with status "Active" and an "Add tags" button. A sidebar on the left has a "Manage" tab selected, showing links for "Getting started", "Service credentials", "Plan", and "Connections". The main content area includes a message about deprecated endpoint URLs and a link to update them. It also features a "Start by viewing the tutorial" section with a "Getting started tutorial" button and an "API reference" link. Below this is a "Credentials" section where the "API key" is listed as a redacted string, with "Download" and "Show credentials" options. The "URL" is listed as <https://api.us-south.natural-language-understanding.watson.cloud.ibm.com/instances/d60>, with copy and download icons.

# Natural Language Understanding

---

- Copy your credentials from IBM Cloud to the notebook “*Lecture 20.ipynb*”

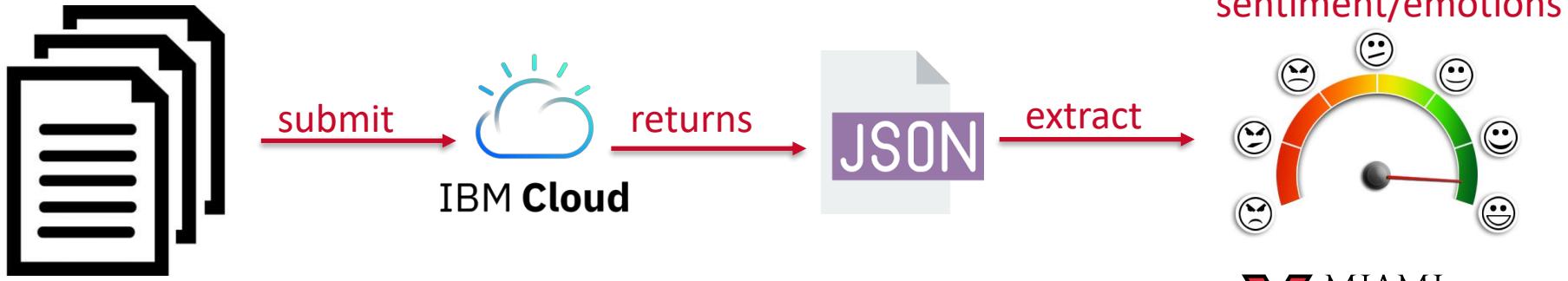


# Cloud Services

---

- Let's experience IBM's Natural Language Understanding from inside Python
  - Illustrative problem: how did consumers' sentiment towards Starbucks after its pledge to hire refugees compare to consumers' sentiment before the pledge?

Facebook posts



# Cloud Services

---

- Natural Language Understanding API
  - Request: REST protocol (GET operation)
    - The text to be analyzed goes together with the base URL
  - Response: JSON
  - Documentation:  
<https://cloud.ibm.com/apidocs/natural-language-understanding>

# Cloud Services

---

## ➤ Natural Language Understanding API

- We have just signed for the IBM Natural Language service to get an API key
- Pricing scheme

| Plan          | Features                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Pricing                                                                                                                         |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Lite          | <p>30,000 NLU Items Per Month<br/>1 Custom Model<br/>Fixed API Rate Limit. See Standard plan for higher API Rate Limit<br/><br/>NOTE: A NLU item is based on the number of data units enriched and the number of enrichment features applied. A data unit is 10,000 characters or less. For example: extracting Entities and Sentiment from 15,000 characters of text is (2 Data Units * 2 Enrichment Features) = 4 NLU Items. A custom model refers to an annotation model developed with Watson Knowledge Studio.</p>                                   | Free                                                                                                                            |
| Standard      | <p>Unlimited NLU Items Per Month<br/>You will be charged per NLU Item &amp; per Custom Model<br/>Increased API Rate Limit can be configured upon request<br/><br/>NOTE: A NLU item is based on the number of data units enriched and the number of enrichment features applied. A data unit is 10,000 characters or less. For example: extracting Entities and Sentiment from 15,000 characters of text is (2 Data Units * 2 Enrichment Features) = 4 NLU Items. A custom model refers to an annotation model developed with Watson Knowledge Studio.</p> | <p>\$800.00 USD/Custom Model Instance per Month<br/>\$0.003 USD/NLU Item<br/>\$0.001 USD/NLU Item<br/>\$0.0002 USD/NLU Item</p> |
| Premium Tier1 | <p>Everything in Standard plus...<br/>Usage and Training Data is Private + Stored in an Isolated Single Tenant Environment<br/>Transaction logging for service improvement is disabled by default<br/>High Availability and Service Level Agreements on Uptime<br/>IBM Cloud Service Endpoints<br/>HIPAA - Washington DC Only</p>                                                                                                                                                                                                                         |                                                                                                                                 |

# Cloud Services

---

## ➤ Request & Response

```
import requests
```

```
content = "NEVER STARBUCKS EVER AGAIN"
```

```
key_values = {'version': '2021-03-25', 'text': content, 'features':'sentiment'}
```

```
response = requests.get(base_url+"/v1/analyze",
                        key_values,
                        auth = ('apikey', api_key))
```

```
response.json()
```

# Cloud Services

---

## ➤ Reflection

- IBM Cloud's Natural Language Understanding allows one to easily analyze textual data
  - No need to understand/perform cumbersome preprocessing techniques
  - We can program the whole script using less than 20 lines of code
- Do you trust the obtained results?
  - Can we do better?
    - Training costs: OpenAI GPT-3 (175 billion parameters) = \$4.6 M
- How does NLU work?
  - Black-box model (proprietary technology)
  - How much do you trust IBM?

# Cloud Services

## ➤ Reflection: accuracy study

- Off-the-Shelf Technologies for Sentiment Analysis of Social Media Data: Two Empirical Studies (By Arthur Carvalho and Lucas Harris)

Tweets

| Technique/<br>Technology    | Accuracy<br>(Negative) | Accuracy<br>(Neutral) | Accuracy<br>(Positive) | Accuracy<br>(Overall) |
|-----------------------------|------------------------|-----------------------|------------------------|-----------------------|
| IBM                         | 6,696/7,341            | 796/1,531             | 1,369/1,508            | 8,861/10,380          |
| NLU                         | [91.2%]                | [52.0%]               | [90.8%]                | [85.4%]               |
| Amazon                      | 4,902/7,341            | 1,251/1,531           | 1,391/1,508            | 7,544/10,380          |
| Comprehend                  | [66.8%]                | [81.7%]               | [92.2%]                | [72.7%]               |
| Microsoft<br>Text Analytics | 5,036/7,341            | 479/1,531             | 1,361/1,508            | 6,876/10,380          |
|                             | [68.6%]                | [31.3%]               | [90.3%]                | [66.2%]               |
| Google<br>Natural Language  | 5,705/7,341            | 603/1,531             | 1,384/1,508            | 7,692/10,380          |
|                             | [77.7%]                | [39.4%]               | [91.8%]                | [74.1%]               |
| Bag of Words                | 3,413/7,341            | 1,073/1,531           | 1,128/1,508            | 5,614/10,380          |
|                             | [46.5%]                | [70.1%]               | [74.8%]                | [54.1%]               |

Facebook Posts

| Technique/<br>Technology    | Accuracy<br>(Negative) | Accuracy<br>(Neutral) | Accuracy<br>(Positive) | Accuracy<br>(Overall) |
|-----------------------------|------------------------|-----------------------|------------------------|-----------------------|
| IBM NLU                     | 1174/1770              | 39/133                | 1128/1264              | 2341/3167             |
|                             | [66.3%]                | [29.3%]               | [89.2%]                | [73.9%]               |
| Amazon<br>Comprehend        | 1258/1770              | 63/133                | 1096/1264              | 2417/3167             |
|                             | [71.1%]                | [47.4%]               | [86.7%]                | [76.3%]               |
| Microsoft<br>Text Analytics | 994/1770               | 32/133                | 1058/1264              | 2084/3167             |
|                             | [56.2%]                | [24.1%]               | [83.7%]                | [65.8%]               |
| Google<br>Natural Language  | 1188/1770              | 38/133                | 894/1264               | 2120/3167             |
|                             | [67.1%]                | [28.6%]               | [70.7%]                | [66.9%]               |
| Bag of Words                | 612/1770               | 57/133                | 1005/1264              | 1674/3167             |
|                             | [34.6%]                | [42.9%]               | [79.5%]                | [52.9%]               |

.....

## SaaS Case Study #2: Image Analytics

# Cloud Services

---

- Question: can we automatically recognize objects (e.g., human beings) in an image?
- (Business) Applications
  - Surveillance
  - Autonomous cars
  - Aiding visually impaired people
    - Toy electric car + high resolution cams + speakers + Raspberry PI running an image-recognition software = a toy that can see and talk
- SaaS: Microsoft Azure's Computer Vision
  - <https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/>

# Cloud Services

---

- Let's experience Microsoft Azure's Computer Vision
- Illustrative story: you created a new dating website [iwannabeamiamimerger.com](http://iwannabeamiamimerger.com)
  - Constraint: each profile picture must have a person in it



# Cloud Services

---

- We shall focus on the last two steps
  - Request: REST protocol (POST operation)
    - We need to send (“post”) an image to Azure
    - Illustrative file: iamaperson.jpeg (available on Canvas)
  - Response: JSON
- I have already signed for the Computer Vision service to get an API key
  - You can use my API key today
  - The service is free, but unfortunately Azure requires credit card information to create an account

# Cloud Services

---

## ➤ Microsoft Azure's Computer Vision

- Documentation: <https://westcentralus.dev.cognitive.microsoft.com/docs/services/computer-vision-v3-1-ga>

- Pricing scheme:

| INSTANCE             | TRANSACTIONS PER SECOND (TPS)** | FEATURES                                                                | PRICE                                                                                                                                                                                                                |
|----------------------|---------------------------------|-------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Free - Web/Container | 20 per minute                   |                                                                         | 5,000 transactions free per month                                                                                                                                                                                    |
| S1 - Web/Container   | 10 TPS                          | Tag<br>Face<br>GetThumbnail<br>Color<br>Image Type<br>GetAreaOfInterest | 0-1M transactions — \$1 per 1,000 transactions<br>1M-10M transactions — \$0.65 per 1,000 transactions<br>10M-100M transactions — \$0.60 per 1,000 transactions<br>100M+ transactions — \$0.40 per 1,000 transactions |
|                      |                                 | OCR<br>Adult<br>Celebrity<br>Landmark<br>Detect, Objects<br>Brand       | 0-1M transactions — \$1 per 1,000 transactions<br>1M-10M transactions — \$0.65 per 1,000 transactions<br>10M-100M transactions — \$0.60 per 1,000 transactions<br>100M+ transactions — \$0.40 per 1,000 transactions |
|                      |                                 | Describe*<br>Read                                                       | 0-1M transactions — \$1.50 per 1,000 transactions<br>1M+ transactions — \$0.60 per 1,000 transactions                                                                                                                |

# Cloud Services

---

## ➤ Request & Response

- We must “post” the image to get a response

import requests

```
base_url = "https://arthur-carvalho-computer-vision.cognitiveservices.azure.com/vision/v3.0/analyze?visualFeatures=Tags"  
api_key = "5b6645a4c53c44498c66f2f4004e7e48"
```

```
image = {'media': open('iamaperson.jpg', 'rb')}  
authentication = {"Ocp-Apim-Subscription-Key":api_key}
```

```
response = requests.post(base_url, files= image, headers=authentication)
```

```
response.json()
```

# Cloud Services

---

## ➤ Reflection

- Microsoft Azure's Computer Vision allows one to analyze images
  - No need to understand cumbersome preprocessing techniques used in image analytics
  - One can build entire businesses/products around this service
  - Other possible services:
    - Detect gory or sexually suggestive content
    - Detect brands, celebrities, landmarks, handwritten text...
- Would you trust the obtained results?
  - Can we do better? (do we have the expertise?)

# Homework #9

---

- Time to implement the `iwannabeamiamimerger.com` feature
  - Code to answer the question: is there a person in a picture?
    - Use the code in Slide 32 as a starting point
    - Save the JSON response to a variable (say, `response`)
    - Create a for-loop that iterates over the objects in `response["tags"]`
      - For each object, check whether there is a key-value pair `name:person`
        - Hint: `object["name"] == "person"`
      - If there is a `name:person` key-value pair, then check whether the confidence associated with the found key-value value is greater than 0.9
        - Hint: `object["confidence"] > 0.9`
      - If both conditions are true, then print “*Account Approved*”
    - Test your code using different images

# Summary

---

- There are several cloud technologies out there that can seriously improve big-data analytics process
  - From infrastructure to off-the-shelf software
- Many SaaS services are “black boxes”
  - We do not really know what is going on behind the scenes
  - This leads to the questions:
    - How much do you trust the service provider?
    - Do you feel comfortable using a technology without knowing what/how it analyzes data?
      - To a certain degree you do: have you ever checked the implementation of any R/SAS/Python function/package?
- Next lecture: Hadoop

# ISA 414 – Managing Big Data

## Lecture 21 – Introduction to Hadoop

*(Part I)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Lecture Objectives

---

- Quick review of Homework 9
- Introduction to the Hadoop ecosystem

# Lecture Instructions

---

- There are several new concepts today
  - Suggestion: actively take notes
  - Important keywords are highlighted in the slides
  
- Recall that several of such concepts will be in the final exam

# Agenda

---

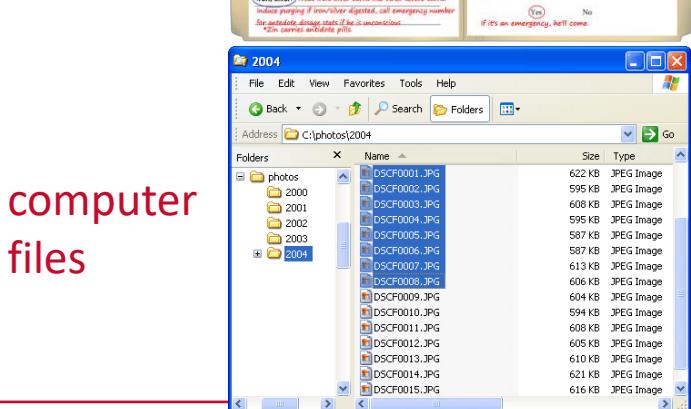
- First part of the course: CRISP-DM
  - Managing big data projects
- Second part of the course: technologies/big-data enablers
  - Cloud computing and storage (previous 2 lectures)
    - IaaS and PaaS help with the required infrastructure
    - SaaS might help with the analysis
  - Hadoop framework (rest of the course)
- Before learning about Hadoop, we must first learn about two relevant concepts
  - Distributed storage
  - Distributed computations

# Distributed Storage

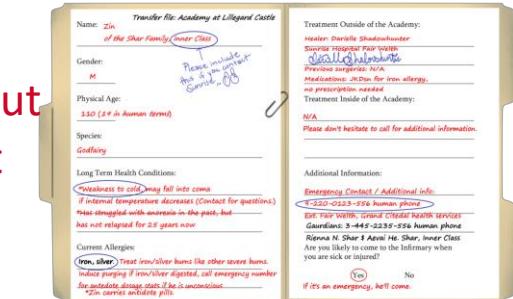
## ➤ Data are stored inside **files**

- For our purposes, a file provides a way of storing and retrieving data/information
- Different technologies
  - *E.g.*, paper records, computer files
- Different formats
  - CSV, JSON, XML, XLSX, ...

data about  
a patient



computer  
files



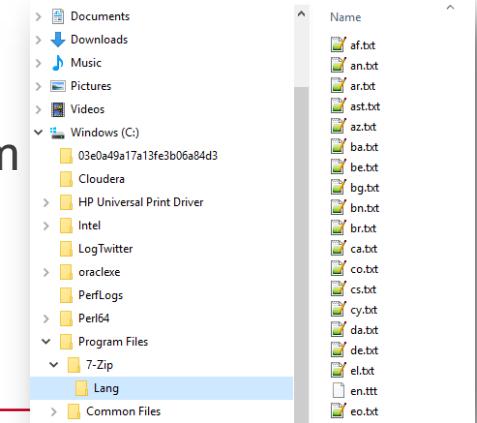
# Distributed Storage

- How can we organize files?
    - Example: file cabinets organize paper-based files
      - File/folder organization/sorting is subjective



# Distributed Storage

- How to organize computer files?
  - **(Computer) file system**
    - Managed by the operating system
      - Linux, Windows, Android,...
    - Often use file directories (like a file cabinet) and hierarchies (hierarchical trees)
      - Folders might contain subfolders
      - Files have exact addresses in the file system
        - **Paths** = branch of the hierarchical tree
        - *E.g.*, C:\carvalag\ISA414\Lecture19.pdf



# Distributed Storage

---

- What about computer (digital) files? How can we store and organize them?
  - Series of bytes
  - Stored inside **storage devices** (e.g., hard-drive disks, flash-based solid-state drives, ...)
    - Non-volatile memory
    - More on this in future lectures



# Distributed Storage

---

- Different computers have different storage capacities



- What happens when one runs out of storage space?
  - Remember that big data is often defined in terms of volume
- Should one just replace an old storage device with a new one?
  - Big hassle: transfer all the data to the new device
    - Think about an organization: potentially, hundreds of terabytes or even petabytes of data

# Distributed Storage

---

- One way of tackling the previous problem is by storing data across multiple machines/storage devices
  - One can simply add a new machine or a storage device to a collection of machines when running out of storage
    - **Distributed storage**
    - No need to transfer data or replace old computers
  - How does one know where a certain file is?
    - Each machine has its own file system
    - The collection of machines has a **Distributed File System (DFS)**
      - Helps to store and index files across multiple machines

# Distributed Storage

---

- Distributed storage tackles one issue related to big data
  - Namely the increasing need for storage space due to the volume aspect of big data
  - Summary of the main idea (we will elaborate on this later):
    - One can use the storage devices (e.g., SSD devices) of many **commodity computers** to store data in a distributed fashion
      - “Many computers” = a **cluster** of computers
    - A *distributed file system* helps to organize and determine where each file is stored in a cluster (computer + file path)

# Distributed Storage

---

- (Over) simplified example of a file system
  - Every file in the system is associated with a path

| File          | Path                                     |
|---------------|------------------------------------------|
| Picture1.jpeg | C:/users/carvalag/pictures/Picture1.jpeg |
| data1.csv     | C:/users/carvalag/data/data1.csv         |

- (Over) simplified example of a distributed file system
  - Every file in the system is associated with a path and storage device

| File       | Device ID    | (Local) Path              |
|------------|--------------|---------------------------|
| noshow.csv | 173.16.157.4 | /user/carvalag/noshow.csv |
| data.csv   | 173.16.157.2 | /user/smith2/data.csv     |

# Distributed Computing

---

- Distributed storage does not tackle another problem associated with data volume
  - The increasing need for computational power
- Complex data-analytics tasks can often benefit from **parallel computation**

# Distributed Computing

---

- Different ways of performing parallel computations

1. Single **nodes** (computers)

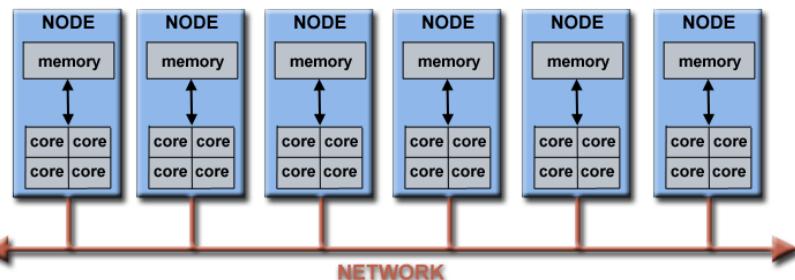
- Multi-core processors: single computing component (CPU) with two or more independent units (“cores”)
- Relatively cheap and easy to program (threads)
  - For example, see the Python module [threading](#)

# Distributed Computing

## ➤ Different ways of performing parallel computations

### 2. Parallel computers (or “super computers”)

- Multiple CPUs:
  - Very large number of single computing nodes
  - Connected via some network (part of the machine)
  - Very expensive
    - From a few to hundreds of millions of dollars



Sunway TaihuLight

RAM: 1,310,000 GB

Storage: 20,000 TB

CPUs: 40,960

Cores: 10,649,600

Cost: \$273 million



# Distributed Computing

---

- Different ways of performing parallel computations

## 3. Commodity cluster

- Distributed computations across many relatively cheap (commodity) individual computers, each one having potentially many cores
- Example: ISA 414 cluster (first request)
  - 5 Computer Nodes
    - 2 Nodes with 48 cores, 256 GB RAM each
    - 3 Nodes with 72 cores, 768 GB RAM each
    - 500 TB of shared storage capacity
  - Price tag: \$116,351.64
    - Including service, racks, and other hardware



# Distributed Computing

---

- Different ways of performing parallel computations

## 3. Commodity cluster

- Distributed computations across many relatively cheap (commodity) individual computers, each one having potentially many cores
- Yahoo! Cluster (2010)
  - 3500 nodes. A typical cluster node has:
    - 2 quad core Xeon processors @ 2.5ghz
    - 4 hard disks (one terabyte each)
    - 16GB RAM

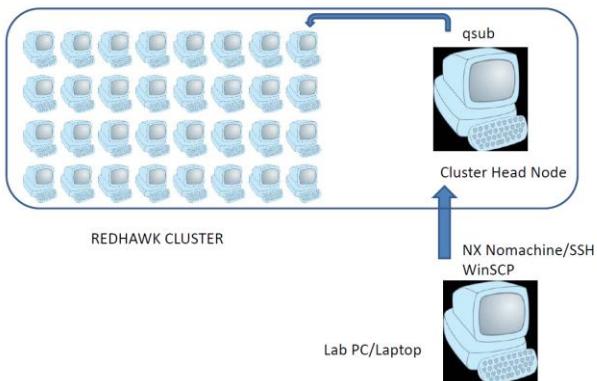
Source: paper “*The Hadoop Distributed File System*”

# Distributed Computing

- Different ways of performing parallel computations

## 3. Commodity cluster

- RedHawks Cluster  
(<https://www.miamioh.edu/research/research-computing-support/services/hpc-cluster/index.html>)



Miami's current HPC cluster consists of:

- 2 login nodes – 24 cores, 384 GB of memory each. Machine names:
  - mualhplp01
  - mualhplp02
- 26 compute nodes – 24 cores, Intel Xeon Gold 6126 2.6 GHz processors, 96 GB of memory each. Machine names:
  - mualhpcp10.mpi-mualhpcp26.mpi
  - mualhpcp28.mpi-mualhpcp35.mpi
  - mualhpcp37.mpi
- 5 compute nodes - 24 cores, Intel Xeon Gold 6226 2.7 GHz processors, 96 GB memory each. Machine names:
  - mualhpcp42.mpi-mualhpcp45.mpi
  - mualhpcp47.mpi
- 2 large memory nodes – 24 cores, Intel Xeon Gold 6126 2.6 GHz processors , 1.5 TB of memory each. Machine names:
  - mualhpcp27.mpi
  - mualhpcp36.mpi
- 4 GPU nodes – 96 GB of RAM, 24 cores, Intel Xeon Gold 6126 2.6 GHz processors and each with 2 Nvidia Tesla V100-PCIE-16GB GPUs. Machine names:
  - mualhpcp38.mpi-mualhpcp41.mpi
- Shared storage system with approximately 30 TB of storage, expandable.

# Distributed Computing

---

- Example of a top-of-the-line “commodity” computer
  - Cisco UCS C240 M4 Rack Server
    - 128 GB RAM
    - Dual Intel E5-2680v3 12-Core 2.50 GHz CPU
    - 2 disks, each having 1TB HDD
    - NvidiaTesla K80 GPU
    - Price tag: \$6,500
- Individual computers are stacked one on top of another in racks



# Distributed Computing

---

## ➤ Commodity cluster

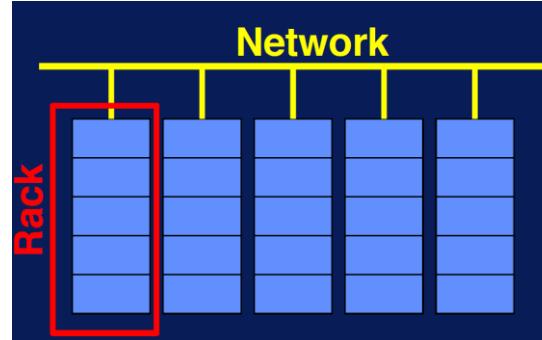
- Much cheaper than supercomputers
- Less powerful
- One can also have a cluster of old, very cheap computers



# Distributed Computing

---

- Computer clusters
  - Many jobs and/or applications can run in parallel
    - Different machines
  - This tackle the problem with growing computational demand due to big data
    - Main idea: one can break down a demanding computation into pieces, which will be executed in parallel in different nodes



# Distributed Storage and Computing

---

➤ Let's put things together now

- A cluster of computers allows for:
  - Distributed storage
    - Requires a distributed file system
  - Distributed computing
    - Different computers can work on different (sub)tasks in parallel
- Hence, a cluster of computers can solve some storage and computational challenges brought by big data
  - Big-data enabler
- Commodity cluster means that the above can be done cheaply

# Distributed Storage and Computing

---

- Let's put things together now
  - New computation paradigm: **move computation to data**
    - Different computers store different pieces of data
    - A task/job that needs access to a piece of data will be executed in the computer where the data are stored
  - Benefit: moving task/jobs require less bandwidth than moving data
    - *I.e.*, it does not mess up the network
  - We did the opposite in class
    - Move data (from a database server) to computation (Python code)
    - Assignment 3: we downloaded data from a MongoDB database

# Distributed Storage and Computing

---

- How to manage a commodity cluster?
  - How to distribute data and computations across nodes?
  - Ideal storage operations:
    - Split volumes of data across nodes
    - Quickly retrieve distributed data
    - Enable the addition of more racks (nodes) without losing performance
    - Fault-tolerant
      - Replicate data partitions across nodes

# Distributed Storage and Computing

---

- How to manage a commodity cluster?
  - How to distribute data and computations across nodes?
  - Ideal computational operations:
    - Scheduling many tasks at the same time running in different nodes
    - Automatic job restart when a node fail:
      - A rack (or individual computer) stop working
      - Network connection is lost

# Distributed Storage and Computing

---

## ➤ Hadoop

- Framework used for distributed storage and computing
  - *I.e.*, a tool that manages commodity clusters
  - Accomplishes all the ideal operations listed before
- Distributed storage
  - Hadoop Distributed File System (HDFS)
- Distributed computation
  - MapReduce
  - Spark
  - ...

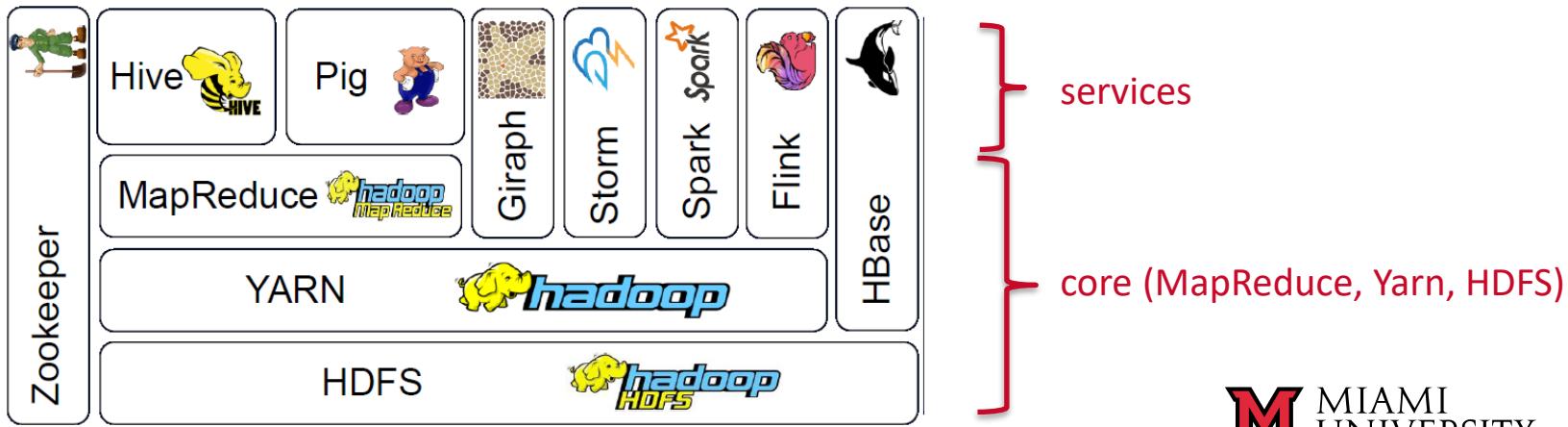
# Distributed Storage and Computing

---

- Hadoop: timeline
  - 2004: MapReduce paper released by Google
    - Title: "*The Google File System*"
    - *Google File System* as distributed file system
    - MapReduce as distributed computing model
  - 2005: Yahoo! releases an open-source implementation of Google's framework called *Hadoop*
  - 2006: Apache continues to develop Hadoop
  - 2006 – present: many services built on top of core Hadoop
    - The *zoo*: *Hive*, *Pig*, *Giraph* ... over 100+ services and counting

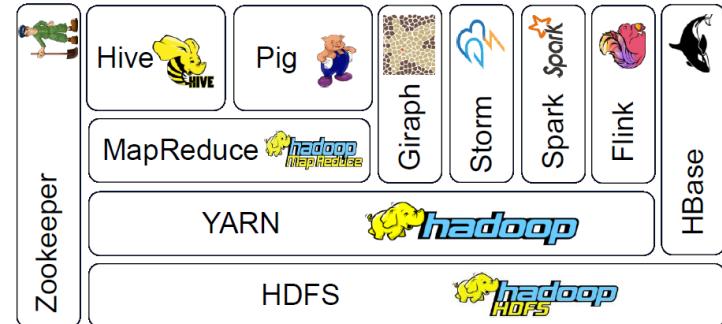
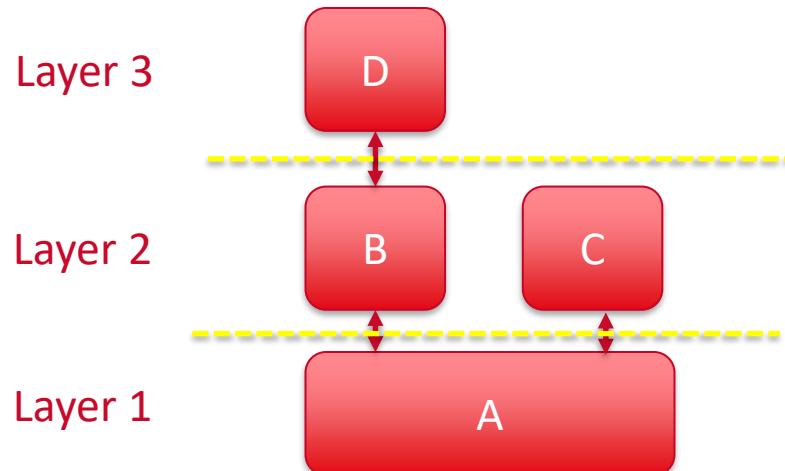
# The Hadoop Ecosystem

- Hadoop version 2
  - Simplified version
    - Many more services: Flume (log collector) Ssqoop (data exchange), ...



# The Hadoop Ecosystem

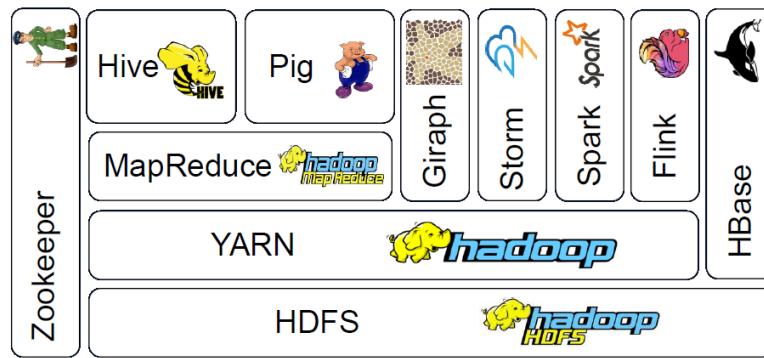
- Hadoop version 2
  - Layer diagram (or stack)
    - A component uses the functionalities/capabilities of the layer below it



# The Hadoop Ecosystem

## ➤ Hadoop version 2

- Layer diagram (or stack)
  - A component uses the functionalities/capabilities of the layer below it



higher levels: interactivity

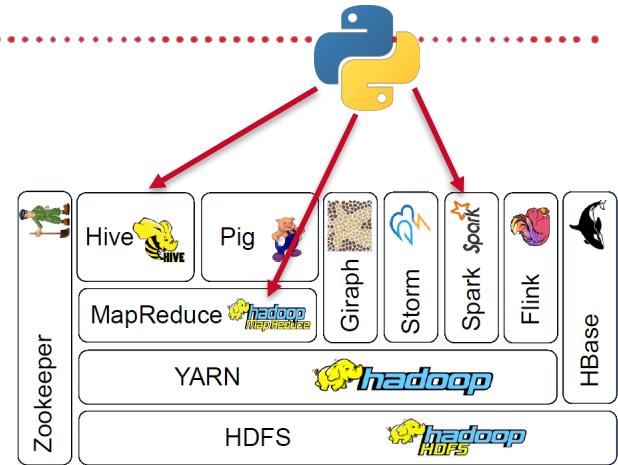


lower levels:  
storage and task scheduling

# The Hadoop Ecosystem

- Overview and agenda

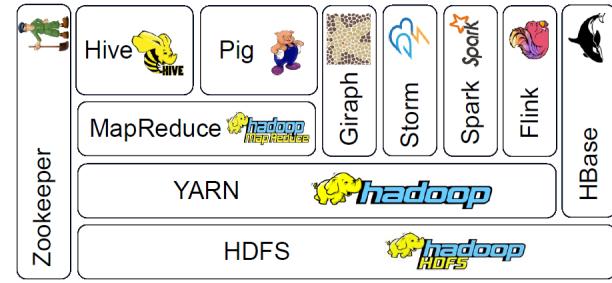
- HDFS (Lecture 22)
  - Hadoop Distributed File System
  - Scalable and reliable storage
- Yarn (Lecture 22 - brief discussion)
  - Schedule jobs/task over HDFS storage
- Spark (Lecture 23 and 24)
  - Built for real-time, in memory processing of data



# The Hadoop Ecosystem

## ➤ Other services

- Pig (created by Yahoo!)
  - Dataflow scripting
- Giraph (created by Facebook)
  - Processing large graphs (social networks)
- Storm/Flink (created by Twitter/Data Artisans)
  - Built for real-time, in memory processing of data
- HBase (created by Facebook)
  - NoSQL database
  - Used by Facebook's messaging platform
- Zookeeper (created by Yahoo!)
  - Manage services named after animals



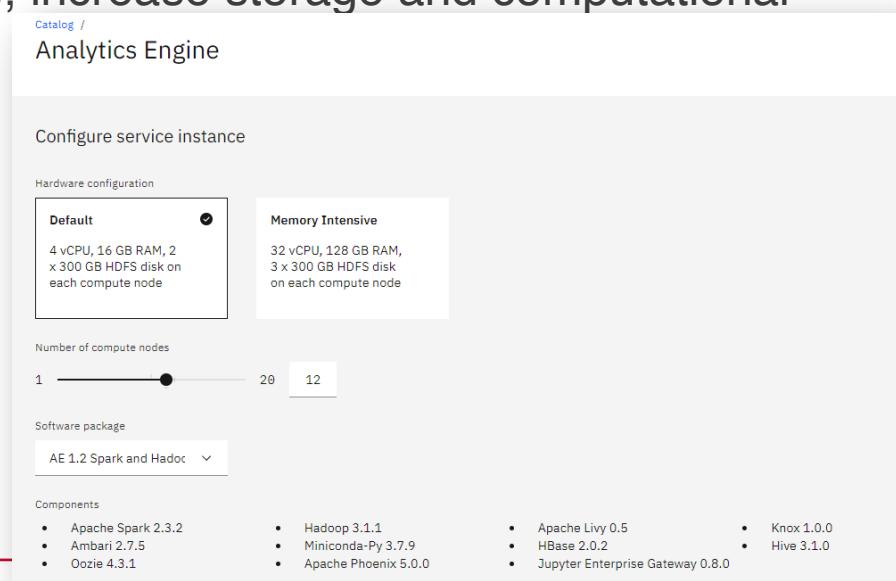
# The Hadoop Ecosystem

---

- How to install Hadoop
  - All the previous tools are free and open-source (why?)
    - Large community for support
  - One can download and install each service/tool separately
    - Obviously, one must install lower-level services first (*e.g.*, HDFS, YARN) before installing higher-level services
    - Requires technical expertise (*e.g.*, advanced Linux/Unix skills)
  - Alternative #1: install a pre-built system (*i.e.*, stacks of these tools)
    - Cloudera, MAPR, Hortonworks
    - Offer commercial support for production environments

# The Hadoop Ecosystem

- Alternative #2: cloud service (PaaS)
  - *E.g.*, you can have your own cluster of computers on IBM Cloud
    - Service name: *Analytics Engine*
    - Few clicks to add extra nodes (*i.e.*, increase storage and computational power) and Hadoop services



---

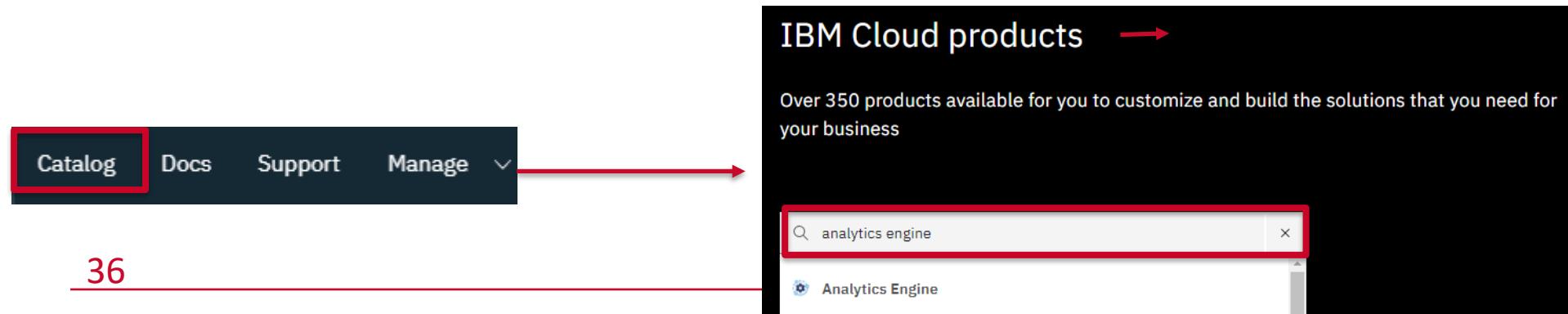
## Demonstration

### Creating a Hadoop Cluster on IBM Cloud

# Hadoop on IBM Cloud

---

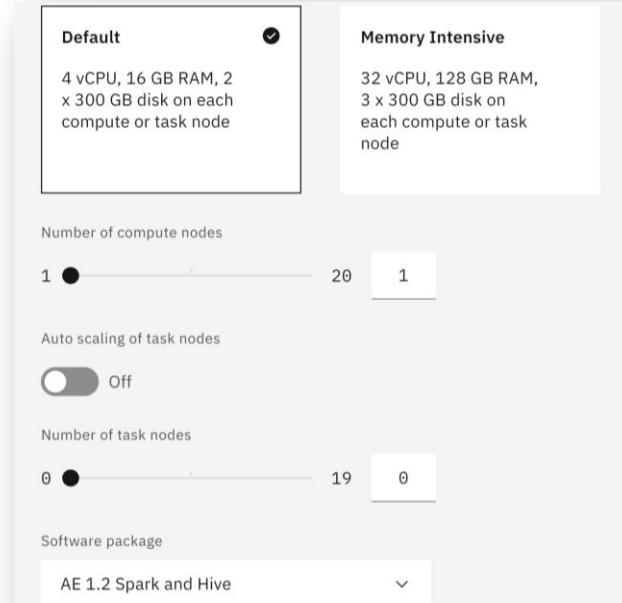
- After creating an account on IBM Cloud
  - Go to <https://www.ibm.com/cloud>
  - Log in
  - Select *Catalog -> Analytics*
  - Search for “*Analytics Engine*”



# Hadoop on IBM Cloud

---

- Select the cloud's region
- Select the pricing plan, configure the cluster, and click on *Create*
  - It might take a few minutes for the cluster to be created



# Hadoop on IBM Cloud

---

- Creating a cluster is incredibly easy
  - Difficult part: integrate a cluster with current business processes and in-house infrastructure
  
- We learn a few more details about Hadoop, HDFS, and Hadoop (PaaS) in our next class
  - Real-time demo with a cluster

# Summary

---

- We learned how distributed storage and computing can tackle volume-related problems associated with big data
  - Hadoop = framework that manages distributed storage and computing
- Next lecture: we study the core of Hadoop
  - HDFS
  - Yarn (brief discussion)

# ISA 414 – Managing Big Data

## Lecture 22 – Introduction to Hadoop

*(Part II)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



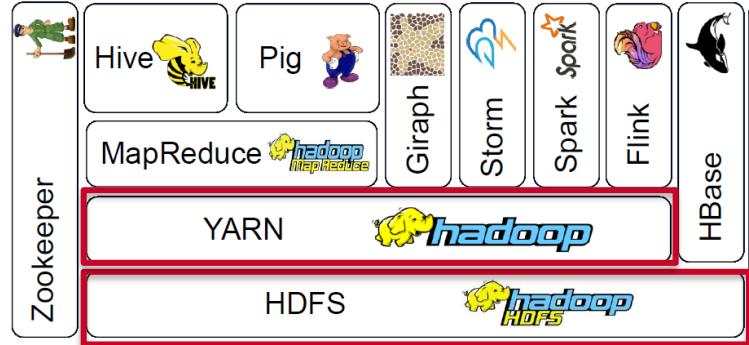
MIAMI UNIVERSITY

Copyright © Arthur Carvalho

# Lecture Objectives

---

- Understand the inner workings of Hadoop
  - HDFS



# Lecture Instructions

---

- There are several new concepts today
  - Suggestion: actively take notes
  - Important keywords are highlighted in the slides

# Hadoop

---

- Summary of previous lecture
  - Handling massive amounts of data is one of the biggest challenges brought by big data
    - How to store massive amounts of data?
      - Cheap solution: **commodity clusters** and **distributed storage**
        - Individual files are stored in different computers in a cluster
        - Massive files are broken down into chunks of data, which are stored in different computers (nodes) in a cluster
        - A **distributed file system** tracks where each piece of data is located

# Hadoop

---

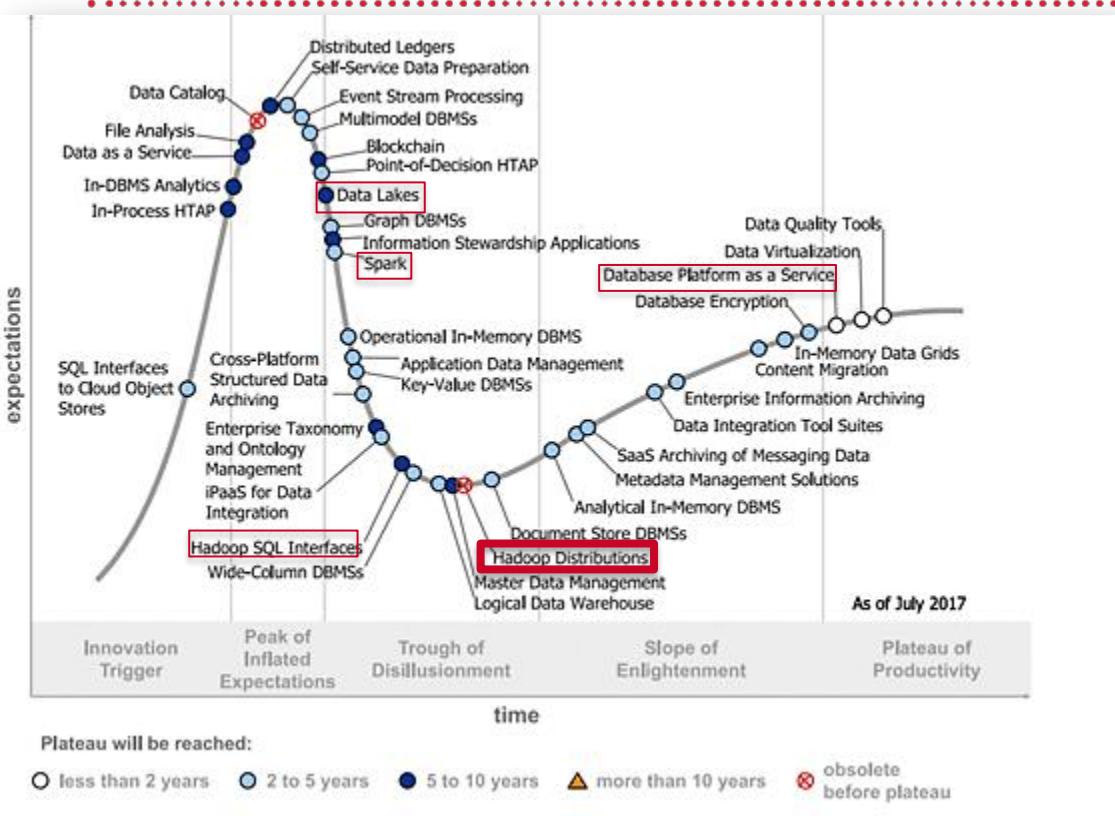
- Summary of previous lecture
  - Handling massive amounts of data is one of the biggest challenges brought by big data
    - How to analyze (perform computations using) big data?
      - Cheap solution: **commodity clusters** and **distributed computation**
        - Paradigm: “**move computation to data**”

# Hadoop

---

- Summary of previous lecture
  - **Hadoop**: framework used for distributed storage and computing
    - *I.e.*, a tool that manages commodity clusters
  - A collection of technologies
    - From data storage to data collection and analysis, these technologies might have drastically different roles
  - Many of the Hadoop technologies are still part of Gartner's Hype Cycle
    - [https://blogs.gartner.com/andrew\\_white/2020/08/07/data-and-analytics-hype-cycles-for-2020-just-published/](https://blogs.gartner.com/andrew_white/2020/08/07/data-and-analytics-hype-cycles-for-2020-just-published/)

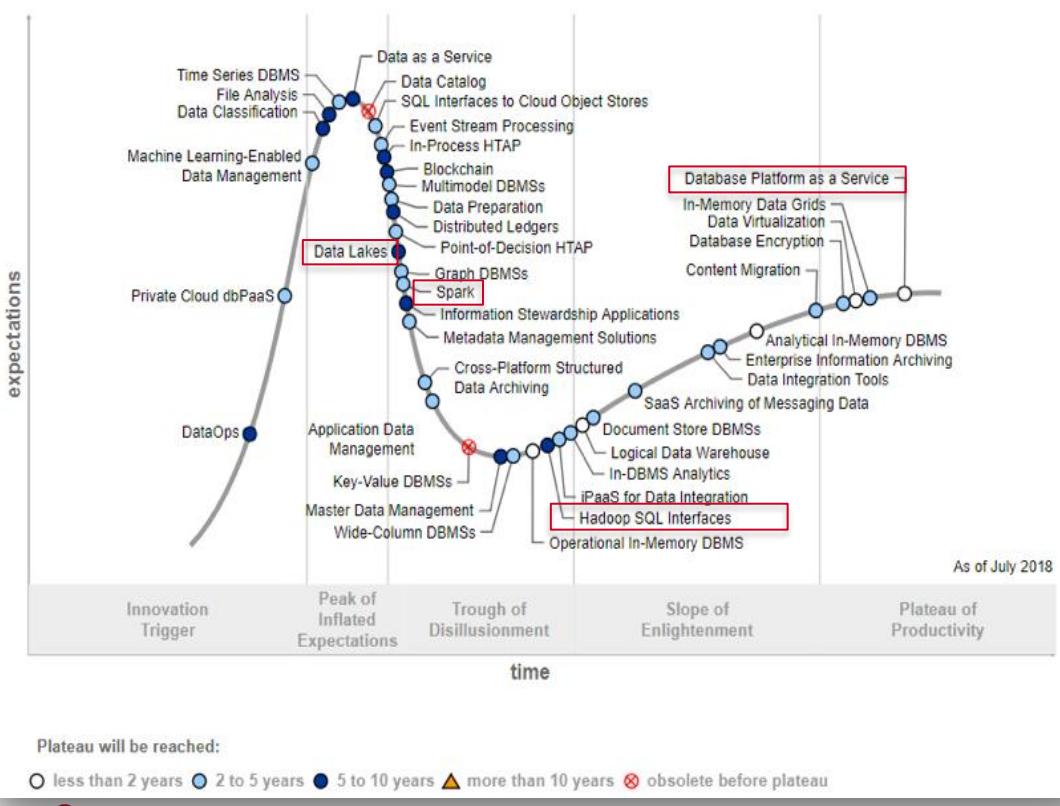
# Hadoop



## Gartner 2017: Hype Cycle for Data Management

(source: <https://www.gartner.com/newsroom/id/3809163>)

# Hadoop

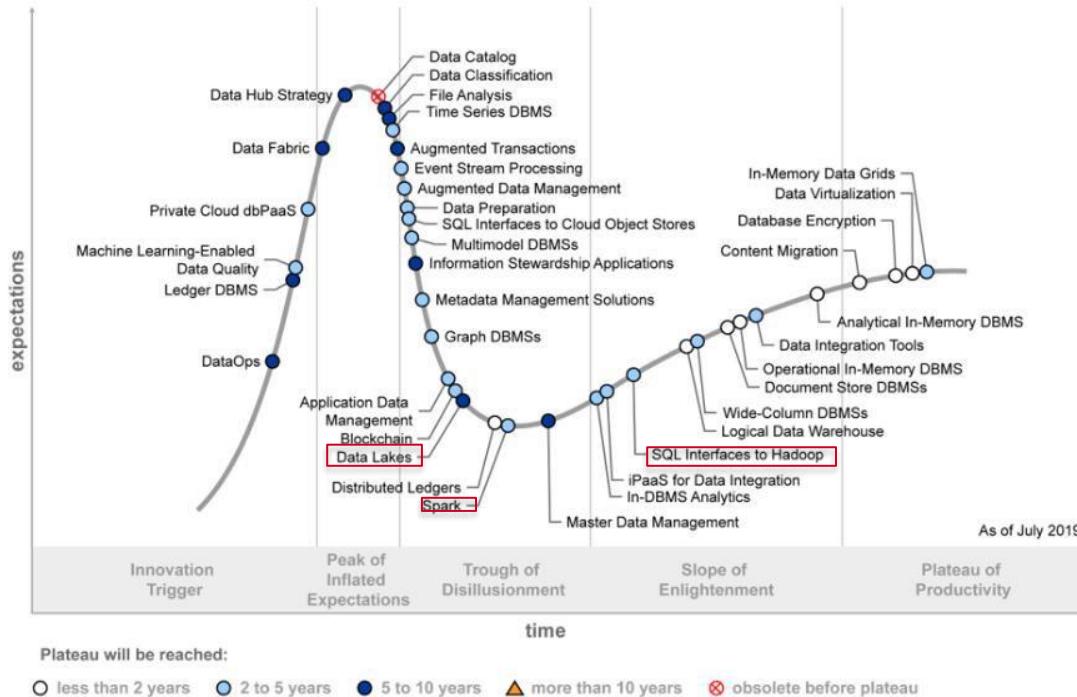


## Gartner 2018: Hype Cycle for Data Management

(source: <https://www.gartner.com/en/newsroom/press-releases/2018-09-11-gartner-hype-cycle-for-data-management-positions-three-technologies-in-the-innovation-trigger-phase-in-2018>)

# Hadoop

## Hype Cycle for Data Management, 2019



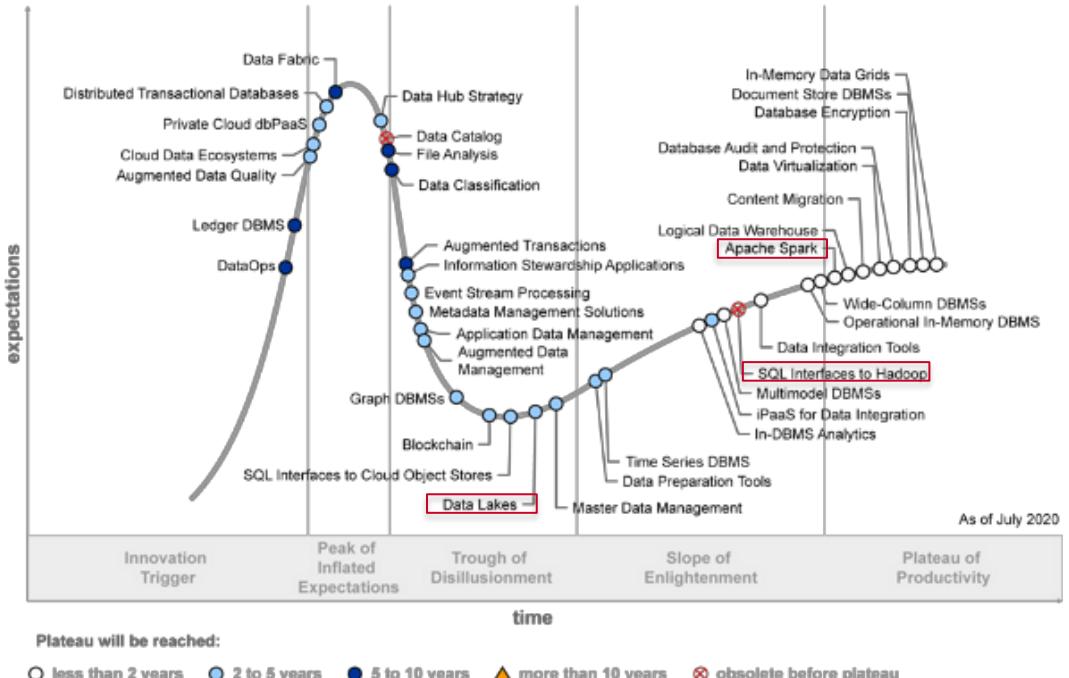
## Gartner 2019: Hype Cycle for Data Management

(source:

<https://www.gartner.com/en/documents/3955768/hype-cycle-for-data-management-2019>)

# Hadoop

## Hype Cycle for Data Management, 2020



## Gartner 2020: Hype Cycle for Data Management

(source: <https://www.denodo.com/en/document/analyst-report/gartner-hype-cycle-2020>)

# HDFS

---

## ➤ Hadoop Distributed File System (HDFS)

- Remember: **distributed storage** requires a **distributed file system**
  - That is what **HDFS** is
- Main idea: HDFS breaks down large files into file blocks and spread them across multiple computers in a cluster
- Storage layer
  - Foundation for most tools in the Hadoop ecosystem
  - Scalable: one can easily add more nodes to increase total storage space
  - Reliable: fault tolerant (more on this soon)

# HDFS

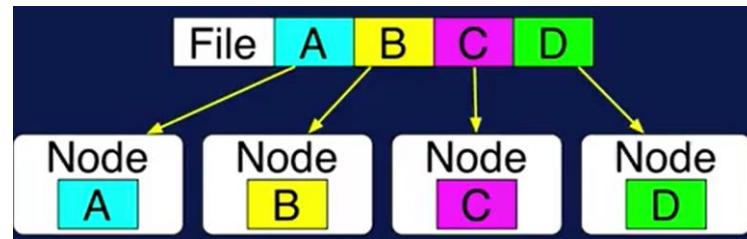
---

- Think about a very large data file (gigabytes or petabytes)
  - Example: Kaggle's Data Science Bowl 2017
    - Goal: improve lung cancer detection
    - Prize: \$1,000,000
    - Data size: 67+ GB
- HDFS breaks such files into chunks (blocks) and spread them over a computer cluster

# HDFS

---

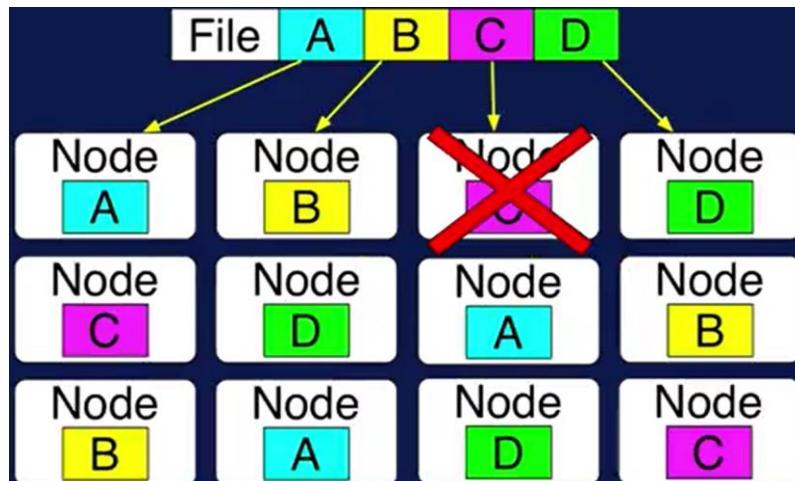
- Example: suppose a person uploads a 256-MB file to HDFS
  - Default block (piece) size: 64 MB
    - Configurable
  - That file is split into four parts ( $256/64 = 4$ ) and the resulting blocks are spread across the cluster



- In the above example, what happens if Node C fails?
  - E.g., power outage

# HDFS

- HDFS is designed for fault tolerance
  - Replication: HDFS makes copies of blocks on different nodes to prevent data loss
    - Default: 3 copies (configurable)



Example: if a node containing block C fails, two other nodes can still provide block C for a requesting application

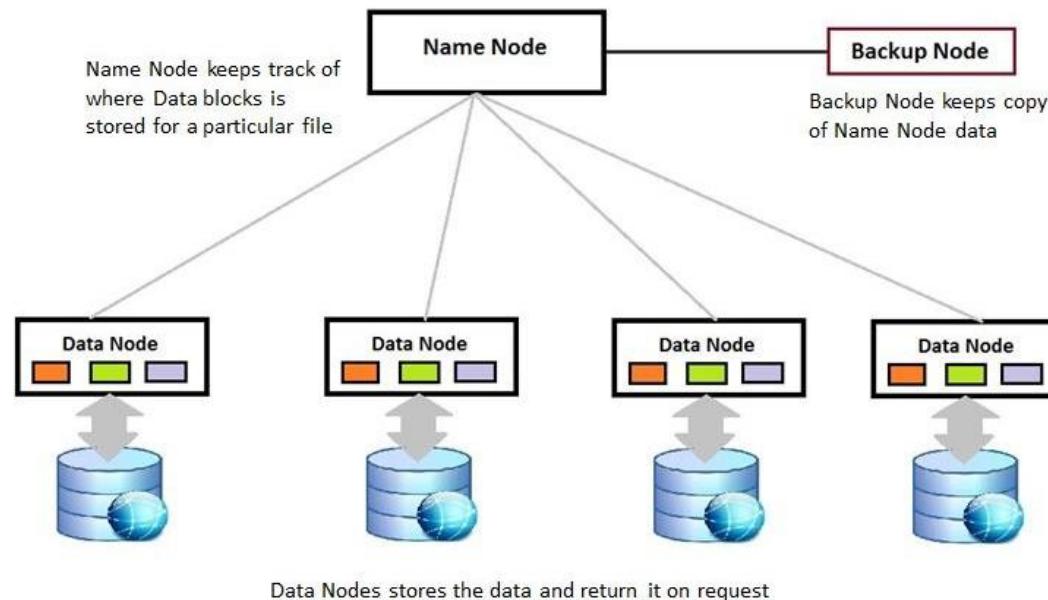
# HDFS

---

- HDFS vs Relational Databases Vs MongoDB
  - HDFS stores files (virtually any type)
    - Structured (e.g., CSV) and unstructured (e.g., images) data are inside files
  - Relational databases store data inside tables
    - Highly structured data
    - One must design a database model first
      - Not very flexible
  - MongoDB stores data using the JSON format
    - Very flexible when handling textual data
      - No need for a predefined model

# HDFS

## ➤ Technical aspects: bird's-eye view



# HDFS

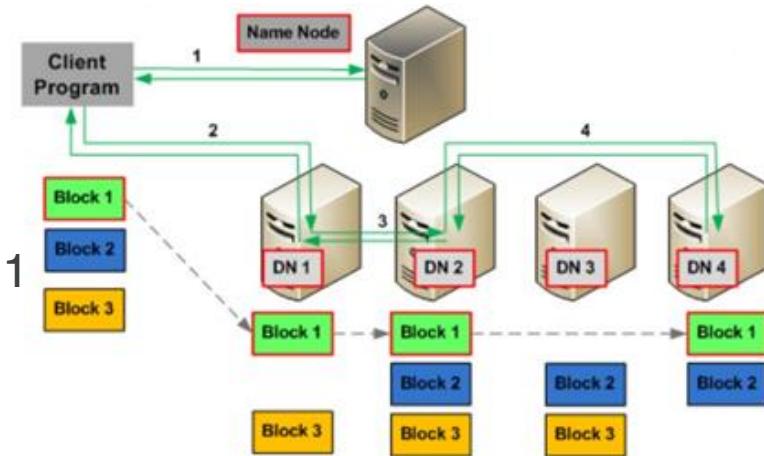
---

- Technical aspects: two key components
  - **Name Node** for metadata
    - Handles metadata (data about the cluster)
    - Manager of the HDFS cluster
      - Keeps track of file names, location of blocks in directories/computers, etc.
    - Usually one per cluster, but there might exist a secondary, backup node
  - **Data Node** for block storage
    - Nodes store data (file blocks)
    - Usually, one Data Node refers to one machine
      - So, there are often many Data Nodes per cluster
    - Listens to commands from the Name Node
      - Block creation, deletion, replication

# HDFS

- Simplified example: suppose a task (client) wants to write a big file to a Hadoop cluster

1. Client makes a request to store Block 1;  
Name node informs the client the locations (e.g., IP addresses) where Block 1 must be stored
  - First location: Data Node 1 (DN 1)
2. Client sends Block 1 and a list of locations to DN 1
3. DN 1 stores Block 1 and sends Block 1 plus a list of locations to DN 2
4. DN 2 stores Block 1 and sends Block 1 plus a list of locations to DN 4



# HDFS

---

- HDFS is all about storing files in a distributed manner
  - We will soon observe how HDFS works in practice
  - Before doing do, let's review some relevant concepts
    - **Data warehouse**
    - **Data lake**

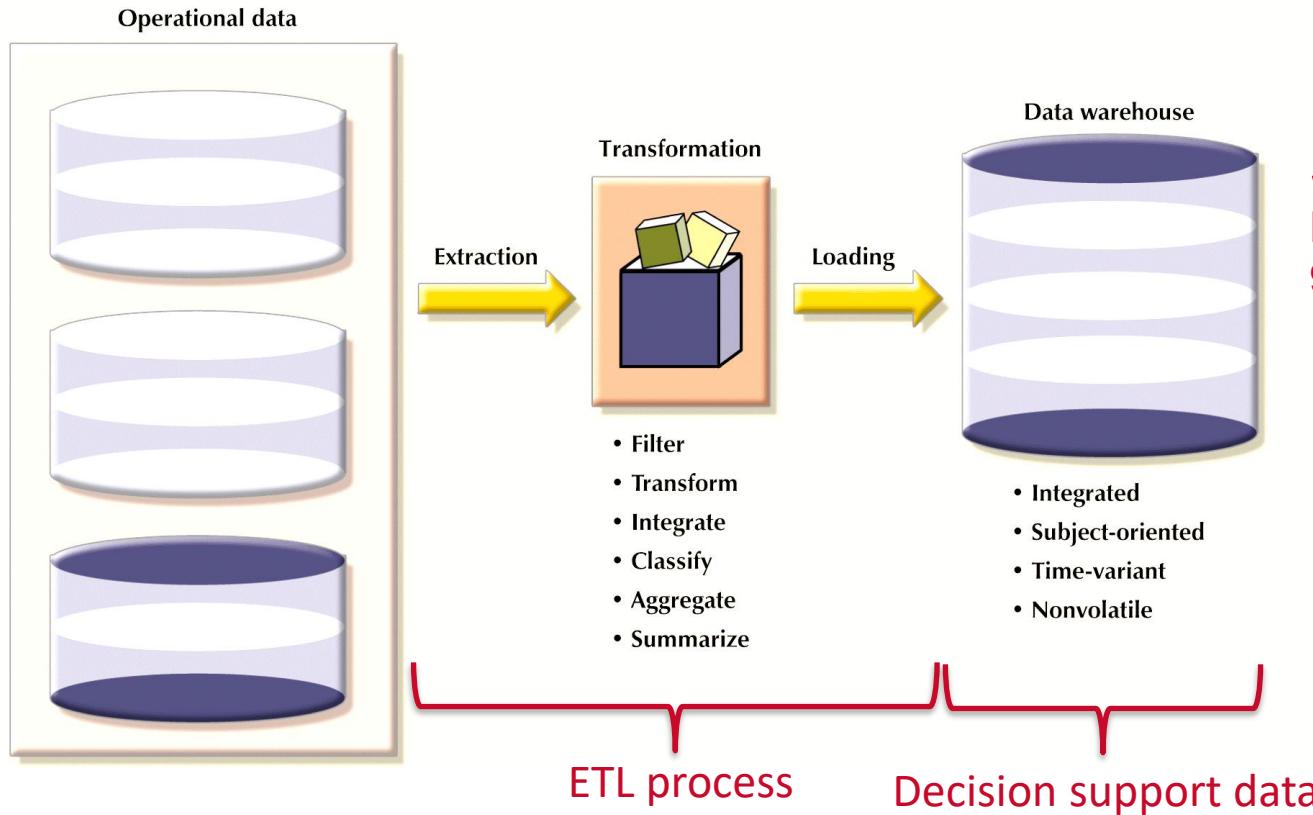
# Data Warehouse

---

## ➤ Data Warehouse

- Business intelligence (BI) technique to integrate and analyze data
- Oftentimes, it is about an enormous collection of data
  - *Subject oriented* – designed around key entities (concepts)
  - *Integrated* – consistency in naming convention, encoding, translation, ...
  - *Time-variant* – data are organized by time periods
  - *Non-volatile* – data are updated in batches, rather than as transactions occur

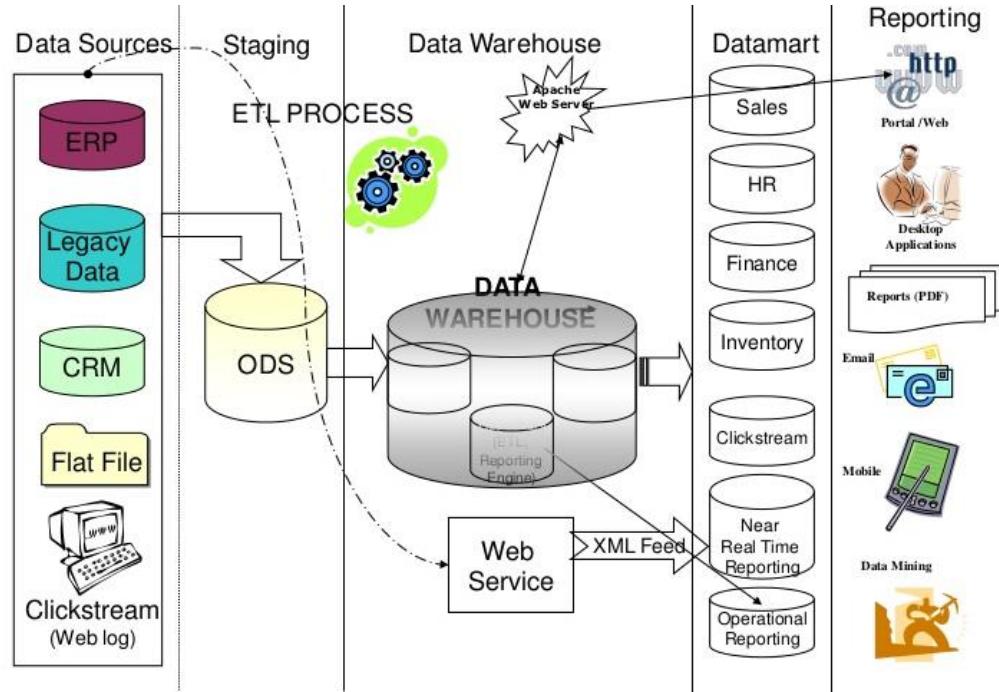
# Data Warehouse



Source:  
Database Systems,  
9th Edition

# Data Warehouse

➤ A more realistic view



# Data Warehouse

---

## ➤ Operational data

- Mostly stored in relational databases
- Optimized to support transactions representing daily operations

## ➤ Decision support data

- Derived from operational data
- Stored in a database optimized for data analysis and query processing

# Data Warehouse

---

- **ETL** - key processes for deriving decision support data from operational data
  - **EXTRACT**
    - The process of extracting operational data, *e.g.*, query a relational database
    - Can be very time-consuming
      - Often happens when the operational databases are not heavily used
    - Data can be stored in distributed databases
    - Potentially different database vendors (*e.g.*, Oracle, MySQL) or representations (*e.g.*, relational model, network model, hierarchical model)

# Data Warehouse

---

- ETL - key processes of deriving decision support data from operational data
  - **TRANSFORM**
    - Standardizing different pieces of data
      - Example: convert “male”/“female” to 0/1, inches to centimeters, date format from “dd/mm/yy” to “yyyy/mm/dd”
    - Cleaning data
      - Same record stored in different databases
      - Misuse of data entry fields
        - *E.g.*, age = 200, SSN = 0000000000

# Data Warehouse

---

- **ETL** - key processes of deriving decision support data from operational data
  - **LOAD**
    - The process of loading the data into the warehouse
    - Occur in batches, rather than after each transaction involving operational data

# Data Warehouse

---

- Is HDFS a data warehouse technology?
  - Technically, no!
  
- Unlike HDFS, a data warehouse:
  1. Follows a pre-defined architecture
    - *E.g.*, a star schema having dimensions, facts, and attributes
  2. Follows a strong data quality assurance process (*i.e.*, ETL)

# Data Lake

---

- So, what is HDFS?
  - Answer: a ***Data Lake***
    - Centralized repository of data
    - Stores all kinds of data in raw format (*i.e.*, files)
      - Images, texts, audio, ...
    - Data are not necessarily curated before being dumped into a data lake
      - One of the main criticism of Hadoop HDFS

# Data Lake

---

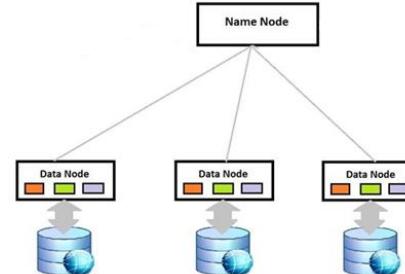
- So, what is Hadoop HDFS?
  - Answer: a *Data Lake*
    - Hadoop was “the next big thing” about 10 years ago
    - Common approach followed by many companies during Hadoop’s hype peak time:

*“We see customers creating big data graveyards, dumping everything into HDFS [Hadoop Distributed File System] and hoping to do something with it down the road. But then they just lose track of what’s there. The main challenge is not creating a data lake, but taking advantage of the opportunities it presents”* (Sean Martin, CTO of Cambridge Semantics)

# HDFS

---

- Let's visualize some of the previous concepts
  - Recall that HDFS is a back-end technology
    - “Boring,” not visually appealing
  - Cluster created on IBM Cloud
    - PaaS called *Analytics Engine*
      - See our previous class for instructions on how to set up this service
    - 1 Name Node (4 cores), 3 Data Nodes (12 cores)
      - \$2.8 per hour
    - Topology



# HDFS

- Cluster administrator's perspective
  - Ambari software

The image shows two side-by-side screenshots. On the left is the 'IBM Analytics Engine' cluster configuration interface. It displays cluster details like 'Compute nodes' (3), 'Task nodes' (0), and components like Apache Spark 2.3.2, Hadoop 3.1.1, etc. A red box highlights the 'Launch Console' button at the top right. Below it, a section for 'Price estimate' shows 'USD 2.80/hour'. A red arrow points from the 'Launch Console' area to the 'Ambari' interface on the right. On the right is the 'Ambari Metrics' dashboard. It features a sidebar with services like HDFS, YARN, MapReduce2, Tez, Hive, Pig, ZooKeeper, Ambari Metrics, Knox, and Spark2. The main area shows metrics: 'NameNode Heap' at 14%, 'HDFS Disk Usage' at 0%, 'NameNode RPC' at 0 ms, and 'Memory Usage' showing a graph with values around 9.3 GB.

# HDFS

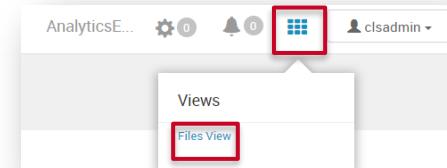
## ➤ Cluster administrator's perspective

- Ambari software
  - Services: Hadoop services running in the cluster
  - *Services -> HDFS -> DATANODES*: Info about data nodes

| Hosts                    |                    |              |               |       |         |                                                            |          |             |
|--------------------------|--------------------|--------------|---------------|-------|---------|------------------------------------------------------------|----------|-------------|
|                          | Name               | IP Address   | Rack          | Cores | RAM     | Disk Usage                                                 | Load Avg | Versions    |
| <input type="checkbox"/> | chphc-732-dn001... | 172.16.172.2 | /default-rack | 4 (4) | 15.51GB | <div style="width: 20%; background-color: #0072bc;"></div> | 0.20     | HDP-3.1.4.0 |
| <input type="checkbox"/> | chphc-732-dn002... | 172.16.172.3 | /default-rack | 4 (4) | 15.51GB | <div style="width: 25%; background-color: #0072bc;"></div> | 0.23     | HDP-3.1.4.0 |
| <input type="checkbox"/> | chphc-732-dn003... | 172.16.172.4 | /default-rack | 4 (4) | 15.51GB | <div style="width: 23%; background-color: #0072bc;"></div> | 0.29     | HDP-3.1.4.0 |

Items per page: 10 ▾ 1 - 3 of 3 <>

- *Views -> “Files Views”*: Upload data to HDFS



# HDFS

---

- There are 2 files inside my cluster
  - */user/clsadmin/Data.csv* (1.2 MB)
  - */user/clsadmin/ Docklands.jmp* (450 MB)
- What happened behind the scenes when I uploaded these files?
  - Tip: IBM uses blocks of size 128 MB; replication factor = 3
- Hacking time
  - I will connect to the Name Node using SSH
  - Issue two commands to get info about the above files

```
hdfs fsck /user/clsadmin/Data.csv -files -locations -blocks
```

```
hdfs fsck /user/clsadmin/Docklands.jmp -files -locations -blocks
```

# HDFS

---

## File #1: Data.csv (1.2 MB)

```
FSCK started by clsadmin (auth:SIMPLE) from /172.16.122.135 for path /user/clsadmin/Data.csv at Tue Mar 09 15:48:52 UTC 2021
/user/clsadmin/Data.csv 1230849 bytes, replicated: replication=3, 1 block(s): OK
0. BP-2089008039-172.16.122.134-1615302408196:blk_1073743602_2778 len=1230849 Live_repl=3 [DatanodeInfoWithStorage[172.16.122.131:50010, DS-f4d957e2
-1c71-42e9-8310-c5d0aec12cb7,DISK], DatanodeInfoWithStorage[172.16.122.132:50010, DS-df4aac5a-b829-4b47-adbe-b5d79cbc75de,DISK], DatanodeInfoWithStor
age[172.16.122.130:50010, DS-126c24f2-174c-46ca-b9bb-7ba343d48a2c,DISK]]
```

```
Status: HEALTHY
Number of data-nodes: 3
Number of racks: 1
Total dirs: 0
Total symlinks: 0
```

```
Replicated Blocks:
Total size: 1230849 B
Total files: 1
Total blocks (validated): 1 (avg. block size 1230849 B)
Minimally replicated blocks: 1 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 3.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
```

# HDFS

---

## File #2: Docklands.jmp (450 MB)

```
[FSCK started by cladmin (auth:SIMPLE) from /172.16.122.135 for path /user/cladmin/Docklands.jmp at Tue Mar 09 15:52:12 UTC 2021
/usr/cladmin/Docklands.jmp 450200813 bytes, replicated: replication=3_4 block(s): OK
[0] BP-2089008039-172.16.122.134-1615302408196:blk_1073743598_2774 [len=134217728] Live_repl=3 [DatanodeInfoWithStorage[172.16.122.130:50010,DS-126c24
f2-174c-46ca-b9bb-7ba343d48a2c,DISK], DatanodeInfoWithStorage[172.16.122.132:50010,DS-df4aac5a-b829-4b47-adbe-b5d79cbc75de,DISK], DatanodeInfoWithSt
orage[172.16.122.131:50010,DS-f4d957e2-1c71-42e9-8310-c5d0aec12cb7,DISK]]
[1] BP-2089008039-172.16.122.134-1615302408196:blk_1073743599_2775 [len=134217728] Live_repl=3 [DatanodeInfoWithStorage[172.16.122.132:50010,DS-4ef651
36-5875-4096-a8df-b7d5644a133e,DISK], DatanodeInfoWithStorage[172.16.122.130:50010,DS-7440a047-980e-405f-aa45-d69fc6aa5396,DISK], DatanodeInfoWithSt
orage[172.16.122.131:50010,DS-d4df355f-be8b-4ca6-95b6-75b1a3549cb7,DISK]
[2] BP-2089008039-172.16.122.134-1615302408196:blk_1073743600_2776 [len=134217728] Live_repl=3 [DatanodeInfoWithStorage[172.16.122.130:50010,DS-126c24
f2-174c-46ca-b9bb-7ba343d48a2c,DISK], DatanodeInfoWithStorage[172.16.122.131:50010,DS-f4d957e2-1c71-42e9-8310-c5d0aec12cb7,DISK], DatanodeInfoWithSt
orage[172.16.122.132:50010,DS-df4aac5a-b829-4b47-adbe-b5d79cbc75de,DISK]]
[3] BP-2089008039-172.16.122.134-1615302408196:blk_1073743601_2777 [len=47547629] Live_repl=3 [DatanodeInfoWithStorage[172.16.122.130:50010,DS-7440a04
7-980e-405f-aa45-d69fc6aa5396,DISK], DatanodeInfoWithStorage[172.16.122.131:50010,DS-d4df355f-be8b-4ca6-95b6-75b1a3549cb7,DISK], DatanodeInfoWithSt
orage[172.16.122.132:50010,DS-4ef65136-5875-4096-a8df-b7d5644a133e,DISK]]

Status: HEALTHY
Number of data-nodes: 3
Number of racks: 1
Total dirs: 0
Total symlinks: 0

Replicated Blocks:
Total size: 450200813 B → 134,217,728 * 3 + 47,547,629 = 450,200,813 bytes (450 MB)
Total files: 1
Total blocks (validated): 4 (avg. block size 112550203 B)
Minimally replicated blocks: 4 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 3.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
```

# HDFS

---

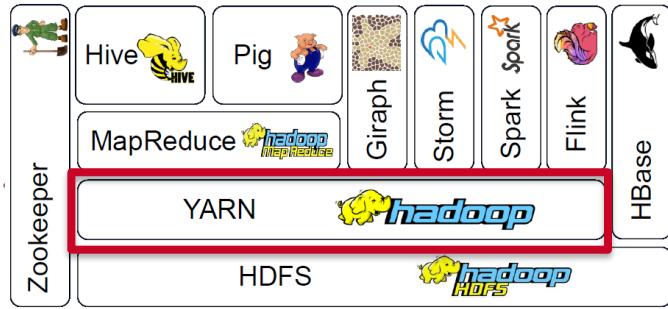
- All the technicalities related to HDFS (configuration, load balancing, installation, etc.) are often handled by IT professionals
  - Data scientists use HDFS to request/send files (data) to/from Hadoop
- Analogy: accessing data/storing data from/in a relational database
  - Database administrator = IT professional who takes care of the technical aspects of running/maintaining the database

# HDFS

---

- The Name Node can tell an application/task where each piece of data is stored
  - The relevant computations are then executed in the Data Node
    - “Moving computation to data”
    - Moving data around is costly (bandwidth wise)
- Unanswered questions
  - If a task wants to access the resources from a node (CPU, memory, disk, ...), how do we know whether that node is busy
    - Busy = dealing with another task
    - We need some sort of resource manager/negotiator!

# YARN



- Yet Another Resource Negotiator
  - Layer on top of HDFS
  - Schedule resources to be used by different applications
    - Enables running multiple applications over HDFS in parallel
- In practice, it is rare to deal with YARN directly
  - In this course, we will use Python to connect to high-level services (such as Spark), which in turn may use YARN behind the scenes

# Hadoop

---

- When to use Hadoop HDFS
  - High data volume
  - High data variety
  - Multiple or parallel computations using the same data
- When not to use Hadoop
  - “Small data” processing
  - To store very well-structured data
    - Relational databases will likely do a better job
  - To store textual data only
    - MongoDB will likely do a better job

# Summary

---

- We learned about a basic service in Hadoop
  - HDFS
    - Distributed file system
- Homework 10 is available on Canvas
- Next lecture
  - Spark

# ISA 414 – Managing Big Data

## Lecture 22 – Introduction to Hadoop

*(Part II)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



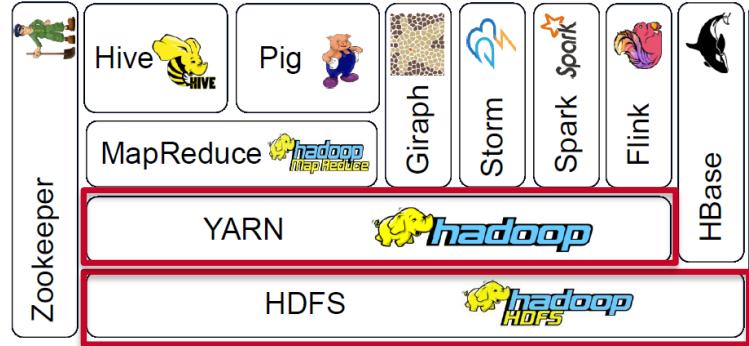
MIAMI UNIVERSITY

Copyright © Arthur Carvalho

# Lecture Objectives

---

- Understand the inner workings of Hadoop
  - HDFS



# Lecture Instructions

---

- There are several new concepts today
  - Suggestion: actively take notes
  - Important keywords are highlighted in the slides

# Hadoop

---

- Summary of previous lecture
  - Handling massive amounts of data is one of the biggest challenges brought by big data
    - How to store massive amounts of data?
      - Cheap solution: **commodity clusters** and **distributed storage**
        - Individual files are stored in different computers in a cluster
        - Massive files are broken down into chunks of data, which are stored in different computers (nodes) in a cluster
        - A **distributed file system** tracks where each piece of data is located

# Hadoop

---

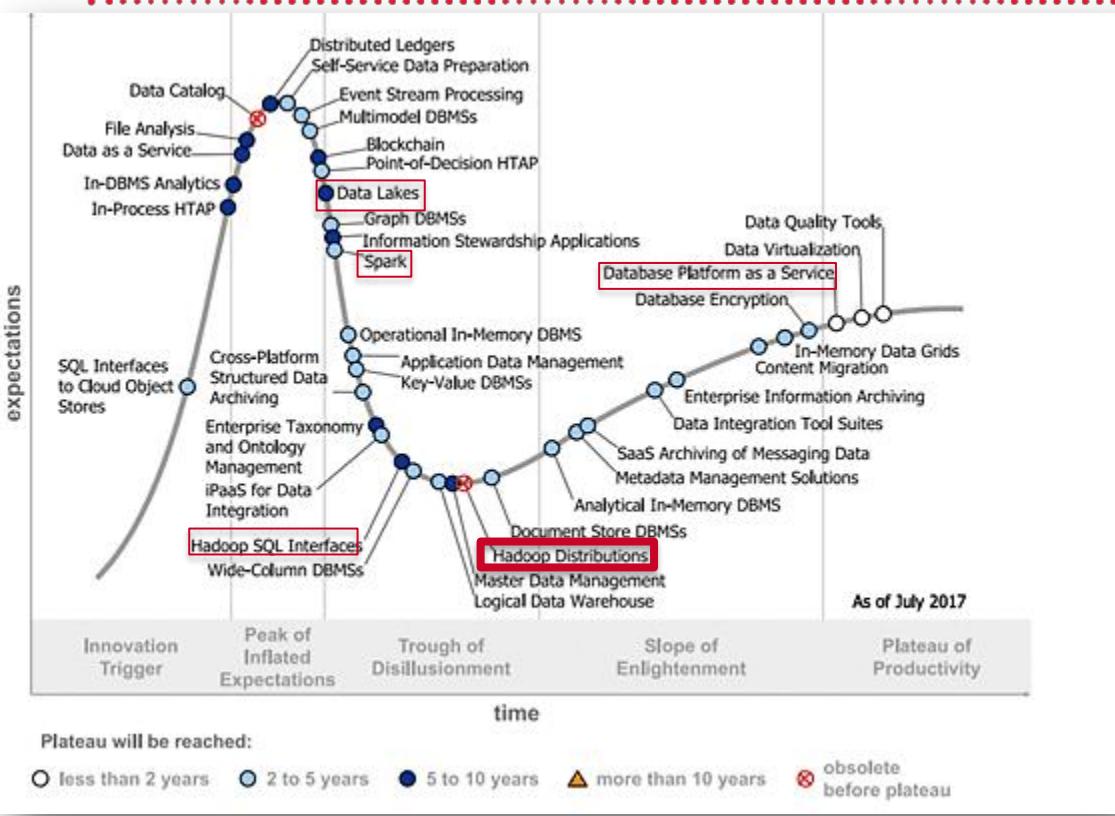
- Summary of previous lecture
  - Handling massive amounts of data is one of the biggest challenges brought by big data
    - How to analyze (perform computations using) big data?
      - Cheap solution: **commodity clusters** and **distributed computation**
        - Paradigm: “**move computation to data**”

# Hadoop

---

- Summary of previous lecture
  - **Hadoop**: framework used for distributed storage and computing
    - *I.e.*, a tool that manages commodity clusters
  - A collection of technologies
    - From data storage to data collection and analysis, these technologies might have drastically different roles
  - Many of the Hadoop technologies are still part of Gartner's Hype Cycle
    - [https://blogs.gartner.com/andrew\\_white/2020/08/07/data-and-analytics-hype-cycles-for-2020-just-published/](https://blogs.gartner.com/andrew_white/2020/08/07/data-and-analytics-hype-cycles-for-2020-just-published/)

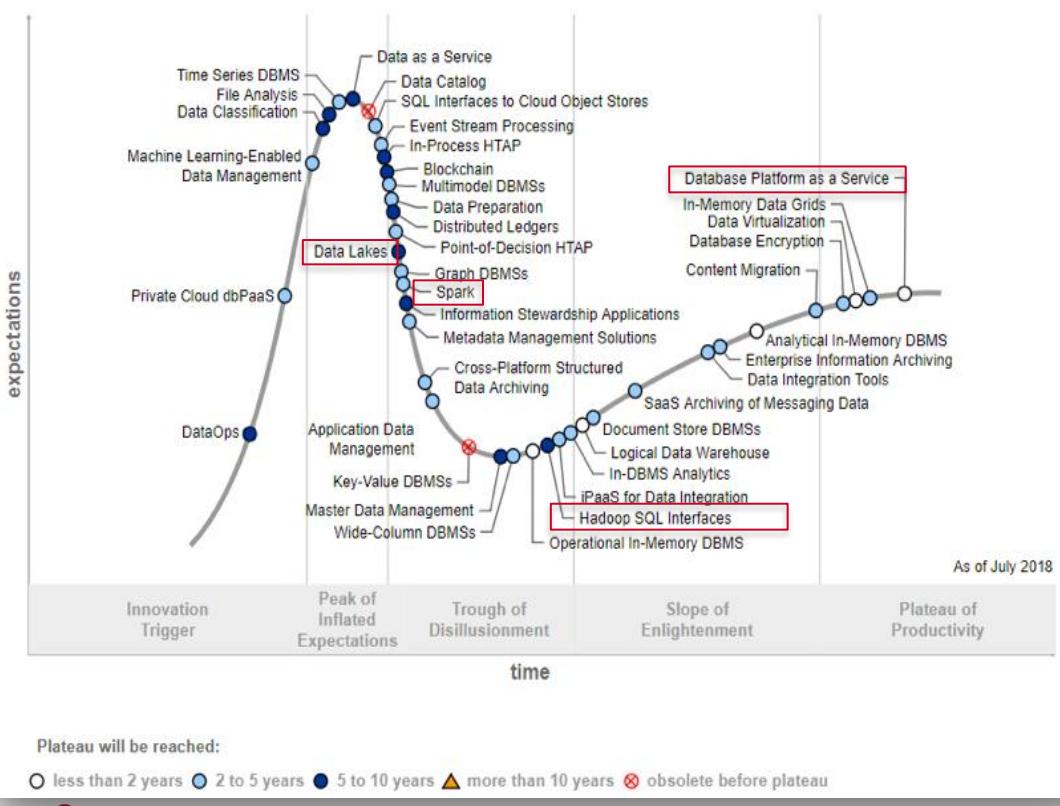
# Hadoop



## Gartner 2017: Hype Cycle for Data Management

(source: <https://www.gartner.com/newsroom/id/3809163>)

# Hadoop

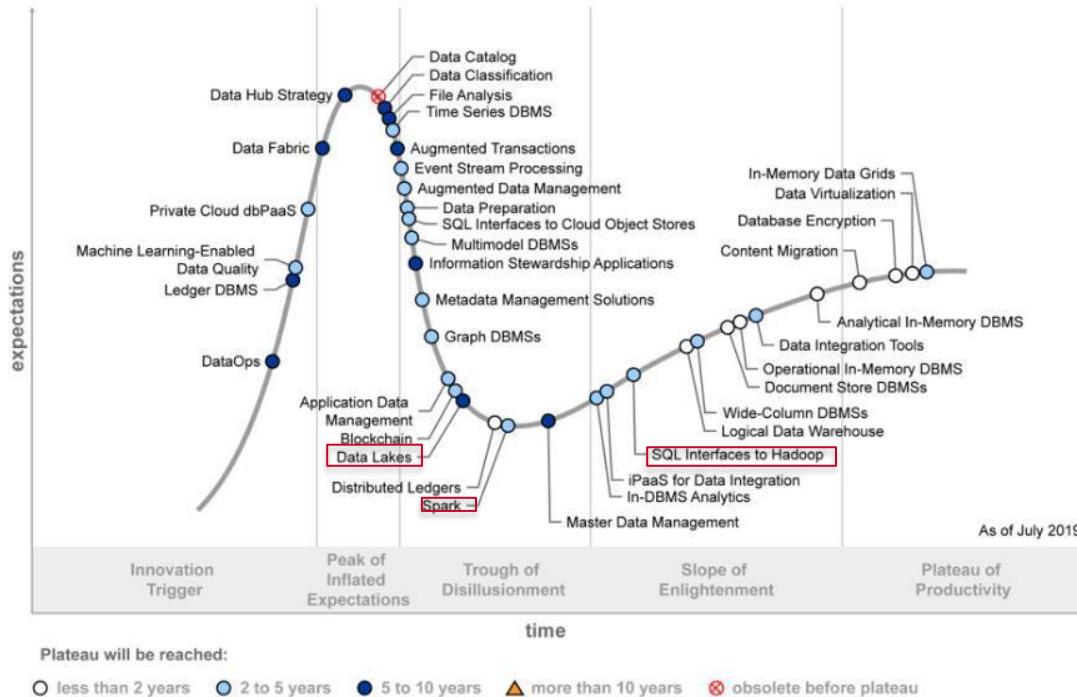


## Gartner 2018: Hype Cycle for Data Management

(source: <https://www.gartner.com/en/newsroom/press-releases/2018-09-11-gartner-hype-cycle-for-data-management-positions-three-technologies-in-the-innovation-trigger-phase-in-2018>)

# Hadoop

## Hype Cycle for Data Management, 2019



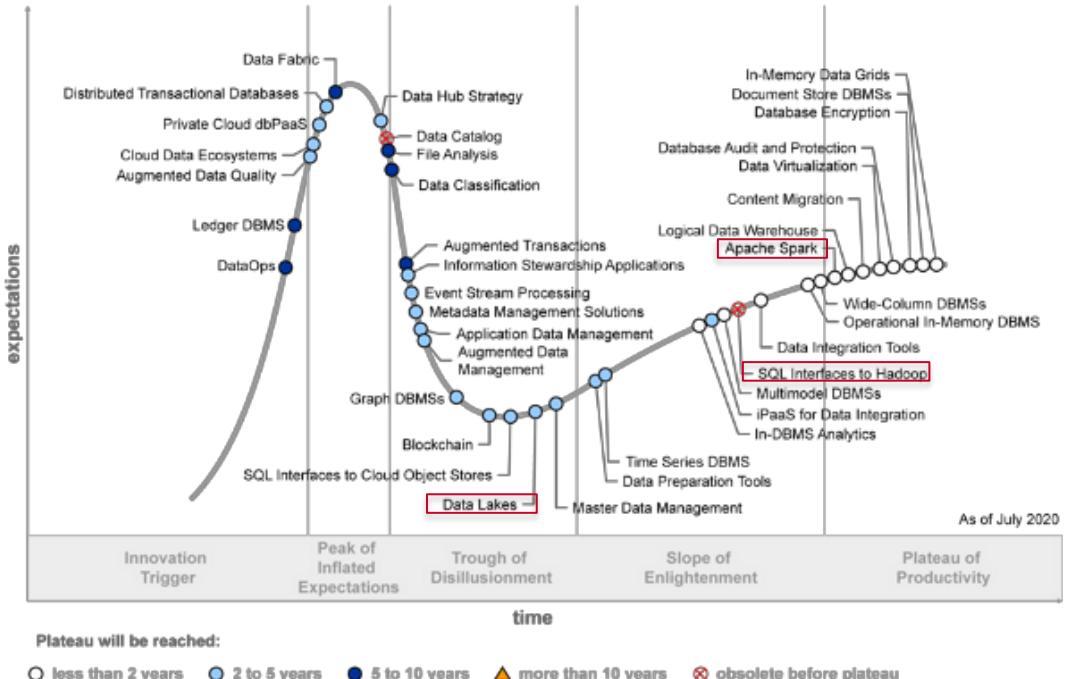
## Gartner 2019: Hype Cycle for Data Management

(source:

<https://www.gartner.com/en/documents/3955768/hype-cycle-for-data-management-2019>)

# Hadoop

## Hype Cycle for Data Management, 2020



## Gartner 2020: Hype Cycle for Data Management

(source: <https://www.denodo.com/en/document/analyst-report/gartner-hype-cycle-2020>)

# HDFS

---

## ➤ Hadoop Distributed File System (HDFS)

- Remember: **distributed storage** requires a **distributed file system**
  - That is what **HDFS** is
- Main idea: HDFS breaks down large files into file blocks and spread them across multiple computers in a cluster
- Storage layer
  - Foundation for most tools in the Hadoop ecosystem
  - Scalable: one can easily add more nodes to increase total storage space
  - Reliable: fault tolerant (more on this soon)

# HDFS

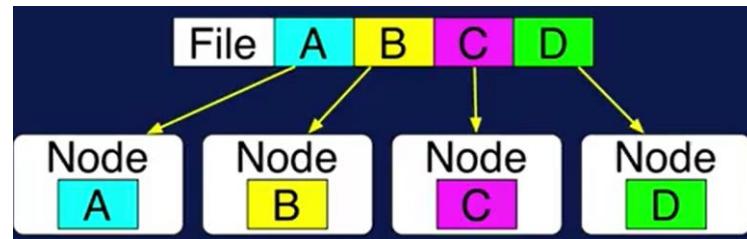
---

- Think about a very large data file (gigabytes or petabytes)
  - Example: Kaggle's Data Science Bowl 2017
    - Goal: improve lung cancer detection
    - Prize: \$1,000,000
    - Data size: 67+ GB
- HDFS breaks such files into chunks (blocks) and spread them over a computer cluster

# HDFS

---

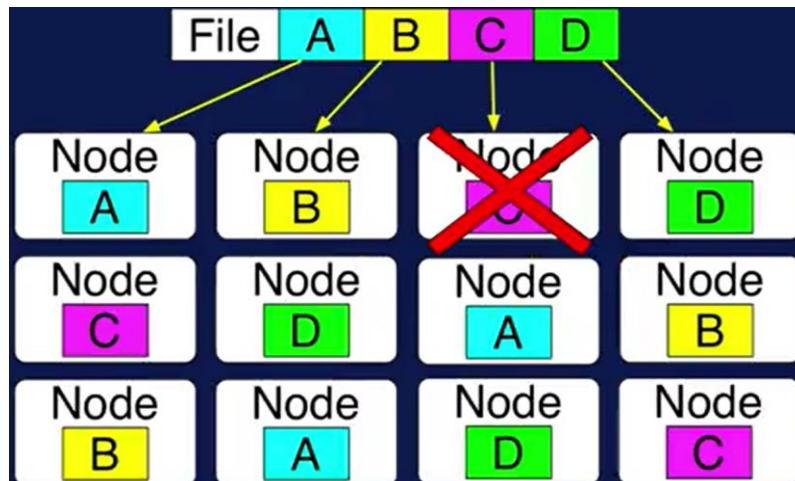
- Example: suppose a person uploads a 256-MB file to HDFS
  - Default block (piece) size: 64 MB
    - Configurable
  - That file is split into four parts ( $256/64 = 4$ ) and the resulting blocks are spread across the cluster



- In the above example, what happens if Node C fails?
  - E.g., power outage

# HDFS

- HDFS is designed for fault tolerance
  - Replication: HDFS makes copies of blocks on different nodes to prevent data loss
    - Default: 3 copies (configurable)



Example: if a node containing block C fails, two other nodes can still provide block C for a requesting application

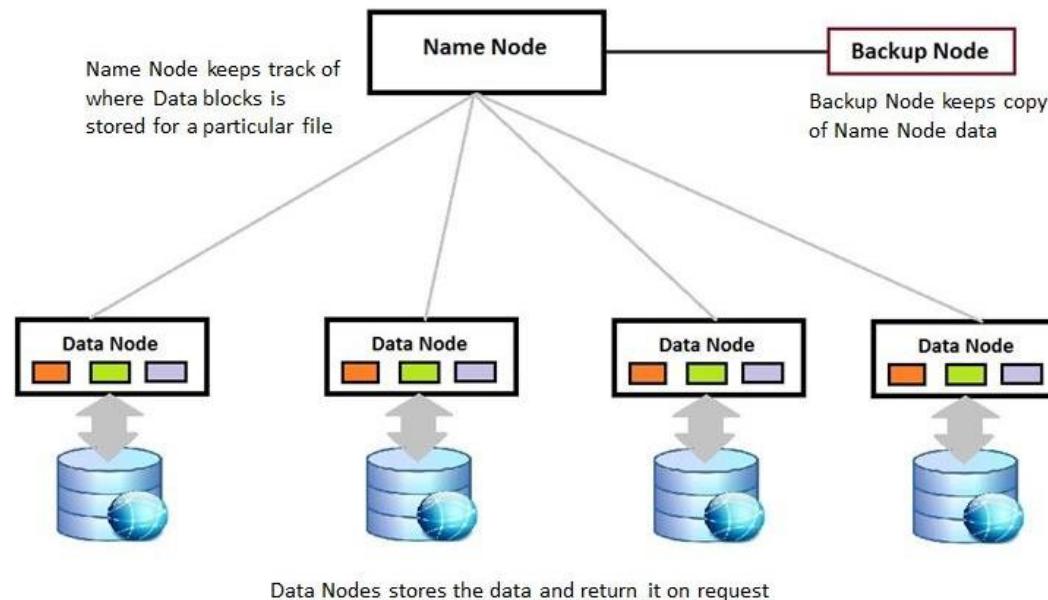
# HDFS

---

- HDFS vs Relational Databases Vs MongoDB
  - HDFS stores files (virtually any type)
    - Structured (e.g., CSV) and unstructured (e.g., images) data are inside files
  - Relational databases store data inside tables
    - Highly structured data
    - One must design a database model first
      - Not very flexible
  - MongoDB stores data using the JSON format
    - Very flexible when handling textual data
      - No need for a predefined model

# HDFS

## ➤ Technical aspects: bird's-eye view



# HDFS

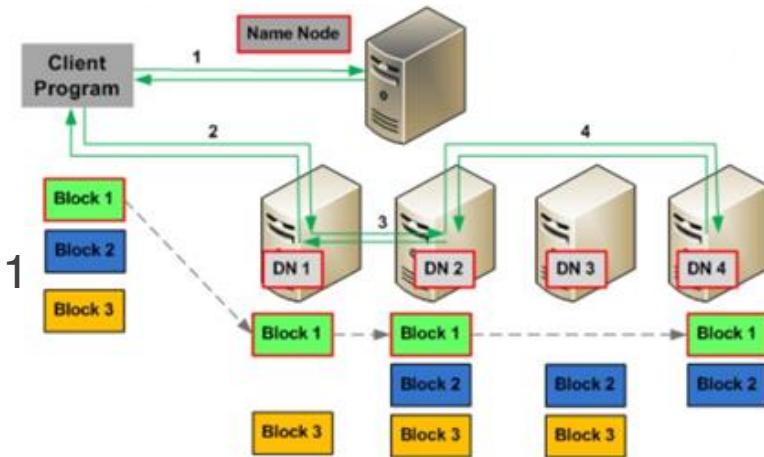
---

- Technical aspects: two key components
  - **Name Node** for metadata
    - Handles metadata (data about the cluster)
    - Manager of the HDFS cluster
      - Keeps track of file names, location of blocks in directories/computers, etc.
    - Usually one per cluster, but there might exist a secondary, backup node
  - **Data Node** for block storage
    - Nodes store data (file blocks)
    - Usually, one Data Node refers to one machine
      - So, there are often many Data Nodes per cluster
    - Listens to commands from the Name Node
      - Block creation, deletion, replication

# HDFS

- Simplified example: suppose a task (client) wants to write a big file to a Hadoop cluster

1. Client makes a request to store Block 1;  
Name node informs the client the locations (e.g., IP addresses) where Block 1 must be stored
  - First location: Data Node 1 (DN 1)
2. Client sends Block 1 and a list of locations to DN 1
3. DN 1 stores Block 1 and sends Block 1 plus a list of locations to DN 2
4. DN 2 stores Block 1 and sends Block 1 plus a list of locations to DN 4



# HDFS

---

- HDFS is all about storing files in a distributed manner
  - We will soon observe how HDFS works in practice
  - Before doing do, let's review some relevant concepts
    - **Data warehouse**
    - **Data lake**

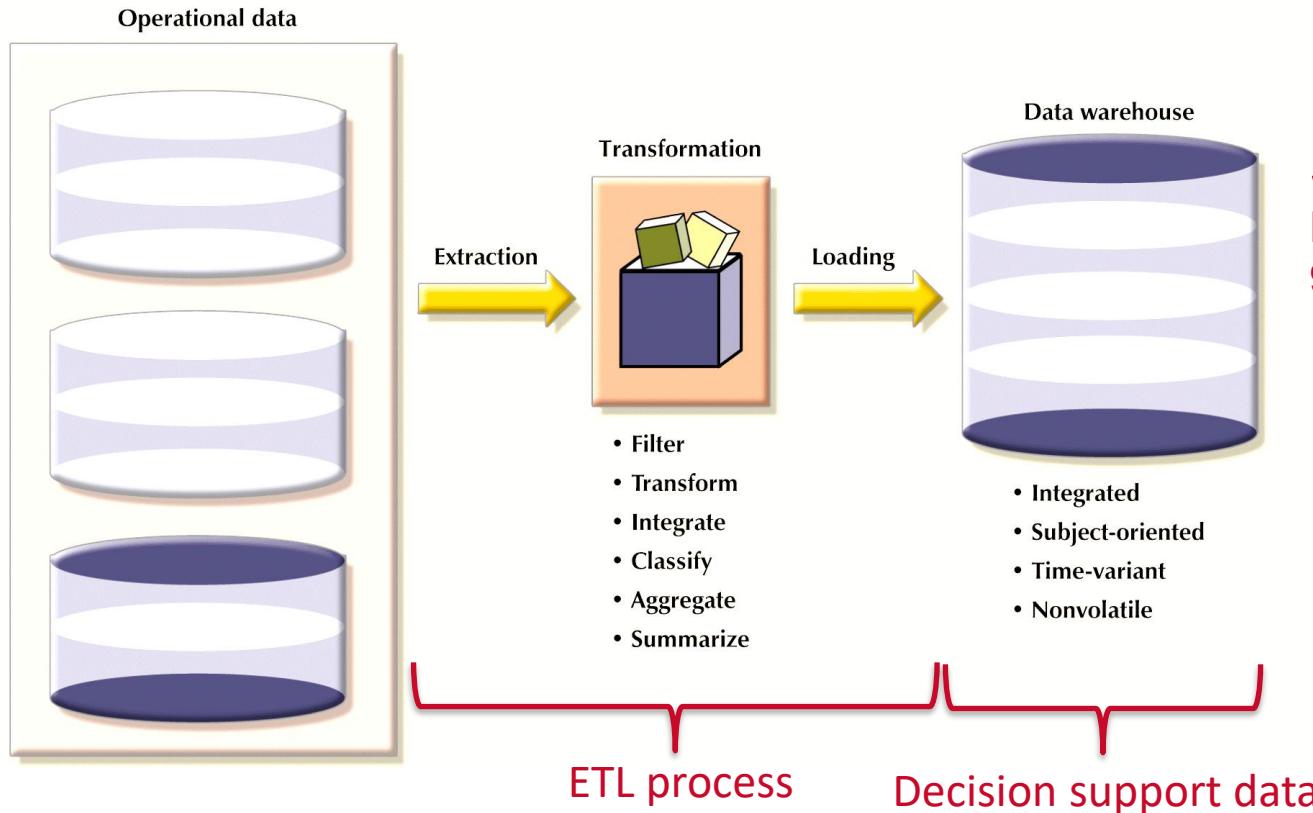
# Data Warehouse

---

## ➤ Data Warehouse

- Business intelligence (BI) technique to integrate and analyze data
- Oftentimes, it is about an enormous collection of data
  - *Subject oriented* – designed around key entities (concepts)
  - *Integrated* – consistency in naming convention, encoding, translation, ...
  - *Time-variant* – data are organized by time periods
  - *Non-volatile* – data are updated in batches, rather than as transactions occur

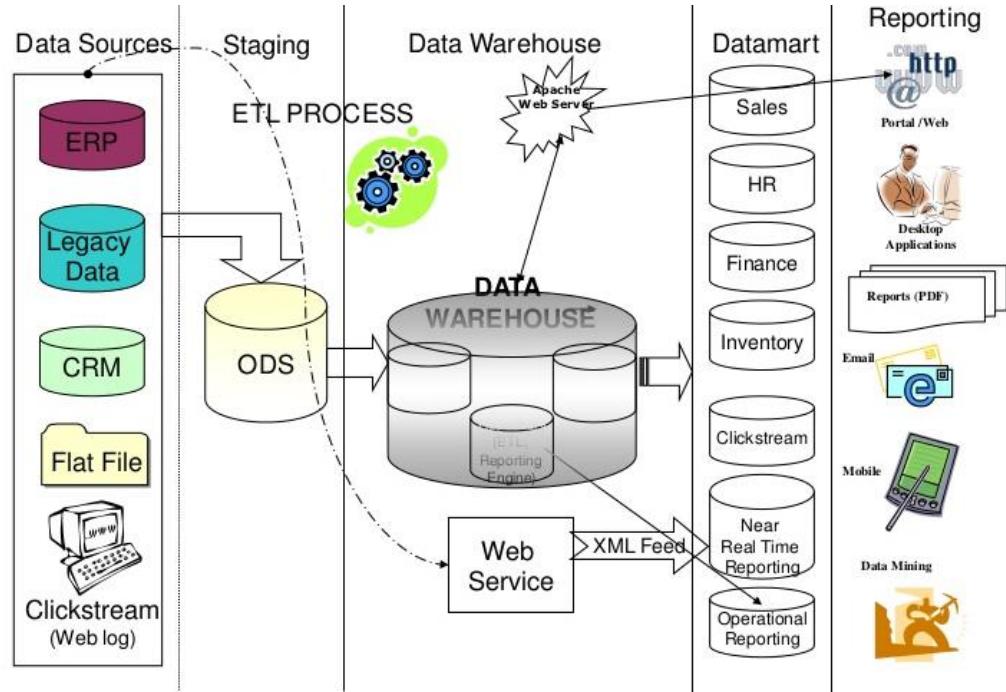
# Data Warehouse



Source:  
Database Systems,  
9th Edition

# Data Warehouse

➤ A more realistic view



# Data Warehouse

---

## ➤ Operational data

- Mostly stored in relational databases
- Optimized to support transactions representing daily operations

## ➤ Decision support data

- Derived from operational data
- Stored in a database optimized for data analysis and query processing

# Data Warehouse

---

- **ETL** - key processes for deriving decision support data from operational data
  - **EXTRACT**
    - The process of extracting operational data, *e.g.*, query a relational database
    - Can be very time-consuming
      - Often happens when the operational databases are not heavily used
    - Data can be stored in distributed databases
    - Potentially different database vendors (*e.g.*, Oracle, MySQL) or representations (*e.g.*, relational model, network model, hierarchical model)

# Data Warehouse

---

- ETL - key processes of deriving decision support data from operational data
  - **TRANSFORM**
    - Standardizing different pieces of data
      - Example: convert “male”/“female” to 0/1, inches to centimeters, date format from “dd/mm/yy” to “yyyy/mm/dd”
    - Cleaning data
      - Same record stored in different databases
      - Misuse of data entry fields
        - *E.g.*, age = 200, SSN = 0000000000

# Data Warehouse

---

- **ETL** - key processes of deriving decision support data from operational data
  - **LOAD**
    - The process of loading the data into the warehouse
    - Occur in batches, rather than after each transaction involving operational data

# Data Warehouse

---

- Is HDFS a data warehouse technology?
  - Technically, no!
- Unlike HDFS, a data warehouse:
  1. Follows a pre-defined architecture
    - *E.g.*, a star schema having dimensions, facts, and attributes
  2. Follows a strong data quality assurance process (*i.e.*, ETL)

# Data Lake

---

- So, what is HDFS?
  - Answer: a ***Data Lake***
    - Centralized repository of data
    - Stores all kinds of data in raw format (*i.e.*, files)
      - Images, texts, audio, ...
    - Data are not necessarily curated before being dumped into a data lake
      - One of the main criticism of Hadoop HDFS

# Data Lake

---

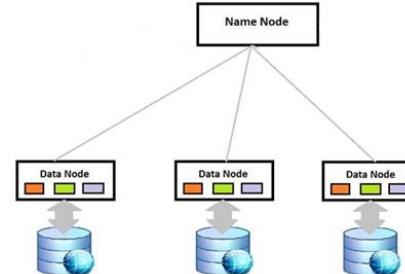
- So, what is Hadoop HDFS?
  - Answer: a *Data Lake*
    - Hadoop was “the next big thing” about 10 years ago
    - Common approach followed by many companies during Hadoop’s hype peak time:

*“We see customers creating big data graveyards, dumping everything into HDFS [Hadoop Distributed File System] and hoping to do something with it down the road. But then they just lose track of what’s there. The main challenge is not creating a data lake, but taking advantage of the opportunities it presents”* (Sean Martin, CTO of Cambridge Semantics)

# HDFS

---

- Let's visualize some of the previous concepts
  - Recall that HDFS is a back-end technology
    - “Boring,” not visually appealing
  - Cluster created on IBM Cloud
    - PaaS called *Analytics Engine*
      - See our previous class for instructions on how to set up this service
    - 1 Name Node (4 cores), 3 Data Nodes (12 cores)
      - \$2.8 per hour
    - Topology



# HDFS

- Cluster administrator's perspective
  - Ambari software

The image shows two side-by-side screenshots. On the left is the 'IBM Analytics Engine' cluster configuration interface. It displays cluster details like 'Compute nodes' (3), 'Task nodes' (0), and a 'Software package' (AE 1.2 Spark and Hadoop). A red box highlights the 'Launch Console' button at the top right. Below it, a 'Price estimate' section shows 'USD 2.80/hour' for the 'United States'. A red arrow points from the 'Launch Console' button to the Ambari Metrics dashboard on the right. The right screenshot is the 'Ambari Metrics' dashboard. It features a sidebar with services like HDFS, YARN, and MapReduce2, and a main area with four circular metrics: 'NameNode Heap' (14%), 'HDFS Disk Usage' (0%), 'NameNode RPC' (0 ms), and 'Memory Usage' (9.3 GB).

# HDFS

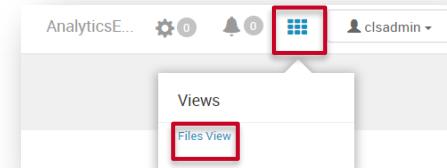
## ➤ Cluster administrator's perspective

- Ambari software
  - Services: Hadoop services running in the cluster
  - *Services -> HDFS -> DATANODES*: Info about data nodes

| Hosts                    |                    |              |               |       |         |                                                            |          |             |
|--------------------------|--------------------|--------------|---------------|-------|---------|------------------------------------------------------------|----------|-------------|
|                          | Name               | IP Address   | Rack          | Cores | RAM     | Disk Usage                                                 | Load Avg | Versions    |
| <input type="checkbox"/> | chphc-732-dn001... | 172.16.172.2 | /default-rack | 4 (4) | 15.51GB | <div style="width: 20%; background-color: #0072bc;"></div> | 0.20     | HDP-3.1.4.0 |
| <input type="checkbox"/> | chphc-732-dn002... | 172.16.172.3 | /default-rack | 4 (4) | 15.51GB | <div style="width: 25%; background-color: #0072bc;"></div> | 0.23     | HDP-3.1.4.0 |
| <input type="checkbox"/> | chphc-732-dn003... | 172.16.172.4 | /default-rack | 4 (4) | 15.51GB | <div style="width: 23%; background-color: #0072bc;"></div> | 0.29     | HDP-3.1.4.0 |

Items per page: 10 ▾ 1 - 3 of 3 <>

- *Views -> “Files Views”*: Upload data to HDFS



# HDFS

---

- There are 2 files inside my cluster
  - */user/clsadmin/Data.csv* (1.2 MB)
  - */user/clsadmin/ Docklands.jmp* (450 MB)
- What happened behind the scenes when I uploaded these files?
  - Tip: IBM uses blocks of size 128 MB; replication factor = 3
- Hacking time
  - I will connect to the Name Node using SSH
  - Issue two commands to get info about the above files

```
hdfs fsck /user/clsadmin/Data.csv -files -locations -blocks
```

```
hdfs fsck /user/clsadmin/Docklands.jmp -files -locations -blocks
```

# HDFS

---

## File #1: Data.csv (1.2 MB)

```
FSCK started by clsadmin (auth:SIMPLE) from /172.16.122.135 for path /user/clsadmin/Data.csv at Tue Mar 09 15:48:52 UTC 2021
/user/clsadmin/Data.csv 1230849 bytes, replicated: replication=3, 1 block(s): OK
0. BP-2089008039-172.16.122.134-1615302408196:blk_1073743602_2778 len=1230849 Live_repl=3 [DatanodeInfoWithStorage[172.16.122.131:50010, DS-f4d957e2
-1c71-42e9-8310-c5d0aec12cb7,DISK], DatanodeInfoWithStorage[172.16.122.132:50010, DS-df4aac5a-b829-4b47-adbe-b5d79cbc75de,DISK], DatanodeInfoWithStor
age[172.16.122.130:50010, DS-126c24f2-174c-46ca-b9bb-7ba343d48a2c,DISK]]
```

```
Status: HEALTHY
Number of data-nodes: 3
Number of racks: 1
Total dirs: 0
Total symlinks: 0
```

```
Replicated Blocks:
Total size: 1230849 B
Total files: 1
Total blocks (validated): 1 (avg. block size 1230849 B)
Minimally replicated blocks: 1 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 3.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
```

# HDFS

---

## File #2: Docklands.jmp (450 MB)

```
[FSCK started by cladmin (auth:SIMPLE) from /172.16.122.135 for path /user/cladmin/Docklands.jmp at Tue Mar 09 15:52:12 UTC 2021
/usr/cladmin/Docklands.jmp 450200813 bytes, replicated: replication=3_4 block(s): OK
[0] BP-2089008039-172.16.122.134-1615302408196:blk_1073743598_2774 [len=134217728] Live_repl=3 [DatanodeInfoWithStorage[172.16.122.130:50010,DS-126c24
f2-174c-46ca-b9bb-7ba343d48a2c,DISK], DatanodeInfoWithStorage[172.16.122.132:50010,DS-df4aac5a-b829-4b47-adbe-b5d79cbc75de,DISK], DatanodeInfoWithSt
orage[172.16.122.131:50010,DS-f4d957e2-1c71-42e9-8310-c5d0aec12cb7,DISK]]
[1] BP-2089008039-172.16.122.134-1615302408196:blk_1073743599_2775 [len=134217728] Live_repl=3 [DatanodeInfoWithStorage[172.16.122.132:50010,DS-4ef651
36-5875-4096-a8df-b7d5644a133e,DISK], DatanodeInfoWithStorage[172.16.122.130:50010,DS-7440a047-980e-405f-aa45-d69fc6aa5396,DISK], DatanodeInfoWithSt
orage[172.16.122.131:50010,DS-d4df355f-be8b-4ca6-95b6-75b1a3549cb7,DISK]
[2] BP-2089008039-172.16.122.134-1615302408196:blk_1073743600_2776 [len=134217728] Live_repl=3 [DatanodeInfoWithStorage[172.16.122.130:50010,DS-126c24
f2-174c-46ca-b9bb-7ba343d48a2c,DISK], DatanodeInfoWithStorage[172.16.122.131:50010,DS-f4d957e2-1c71-42e9-8310-c5d0aec12cb7,DISK], DatanodeInfoWithSt
orage[172.16.122.132:50010,DS-df4aac5a-b829-4b47-adbe-b5d79cbc75de,DISK]]
[3] BP-2089008039-172.16.122.134-1615302408196:blk_1073743601_2777 [len=47547629] Live_repl=3 [DatanodeInfoWithStorage[172.16.122.130:50010,DS-7440a04
7-980e-405f-aa45-d69fc6aa5396,DISK], DatanodeInfoWithStorage[172.16.122.131:50010,DS-d4df355f-be8b-4ca6-95b6-75b1a3549cb7,DISK], DatanodeInfoWithSt
orage[172.16.122.132:50010,DS-4ef65136-5875-4096-a8df-b7d5644a133e,DISK]]]

Status: HEALTHY
Number of data-nodes: 3
Number of racks: 1
Total dirs: 0
Total symlinks: 0

Replicated Blocks:
Total size: 450200813 B → 134,217,728 * 3 + 47,547,629 = 450,200,813 bytes (450 MB)
Total files: 1
Total blocks (validated): 4 (avg. block size 112550203 B)
Minimally replicated blocks: 4 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 3
Average block replication: 3.0
Missing blocks: 0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
```

# HDFS

---

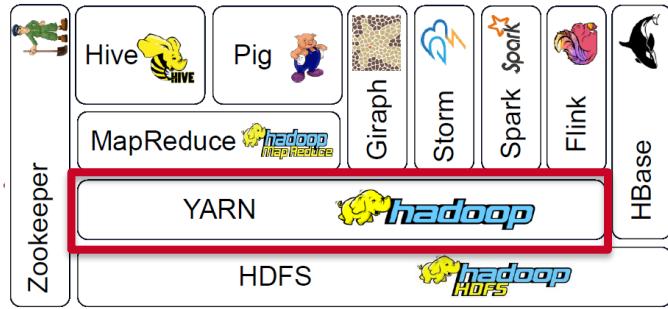
- All the technicalities related to HDFS (configuration, load balancing, installation, etc.) are often handled by IT professionals
  - Data scientists use HDFS to request/send files (data) to/from Hadoop
- Analogy: accessing data/storing data from/in a relational database
  - Database administrator = IT professional who takes care of the technical aspects of running/maintaining the database

# HDFS

---

- The Name Node can tell an application/task where each piece of data is stored
  - The relevant computations are then executed in the Data Node
    - “Moving computation to data”
    - Moving data around is costly (bandwidth wise)
- Unanswered questions
  - If a task wants to access the resources from a node (CPU, memory, disk, ...), how do we know whether that node is busy
    - Busy = dealing with another task
    - We need some sort of resource manager/negotiator!

# YARN



- Yet Another Resource Negotiator
  - Layer on top of HDFS
  - Schedule resources to be used by different applications
    - Enables running multiple applications over HDFS in parallel
- In practice, it is rare to deal with YARN directly
  - In this course, we will use Python to connect to high-level services (such as Spark), which in turn may use YARN behind the scenes

# Hadoop

---

- When to use Hadoop HDFS
  - High data volume
  - High data variety
  - Multiple or parallel computations using the same data
- When not to use Hadoop
  - “Small data” processing
  - To store very well-structured data
    - Relational databases will likely do a better job
  - To store textual data only
    - MongoDB will likely do a better job

# Summary

---

- We learned about a basic service in Hadoop
  - HDFS
    - Distributed file system
- Homework 10 is available on Canvas
- Next lecture
  - Spark

# ISA 414 – Managing Big Data

## Lecture 23 – Introduction to Spark

*(Part I)*

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Announcements

---

- Assignment 4 is now on Canvas
  - Deadline: Sunday Nov. 14<sup>th</sup> before 11:59 pm
- Final-project groups
  - You will present your preliminary ideas on Tuesday and Thursday next week
  - It is now time to form the groups
    - **Send me your preferences by the end of Wednesday, Nov 10<sup>th</sup>**

# Lecture Objectives

---

- Review of Homework 10
- Understand the basic components of traditional computer architectures
  - Difference between main memory and secondary storage
- Learn about Spark
  - RDD, Transformations, Actions, Libraries
- Prepare the Databricks environment

# Lecture Instructions

---

- Download the files *mobydick.txt* and *Lecture 23.ipynb* from Canvas

# Lecture Instructions

---

Choose a cloud provider

 Amazon Web Services

 Microsoft Azure

 Google Cloud Platform

Get started

By clicking "Get started", you agree to the [Privacy Policy](#) and [Terms of Service](#)

Don't have a cloud account?  
Community Edition is a limited Databricks environment for personal use and training.

[Get started with Community Edition](#)

By clicking "Get started with Community Edition", you agree to the [Privacy Policy](#) and [Community Edition Terms of Service](#)

# Lecture Instructions

---

- You should get to this screen

## Welcome to databricks

[Explore the Quickstart Tutorial](#)

Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

[Import & Explore Data](#)

Quickly import data, preview its schema, create a table, and query it in a notebook.

[Create a Blank Notebook](#)

Create a notebook to start querying, visualizing, and modeling your data.

[Common Tasks](#)

-  [New Notebook](#)
-  [Create Table](#)
-  [New Cluster](#)
-  [New Job](#)
-  [New MLflow Experiment](#)
-  [Import Library](#)
-  [Read Documentation](#)

[Recents](#)

Recent files appear here as you work.

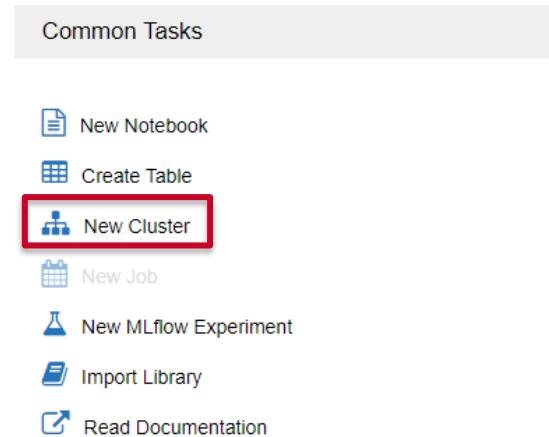
[What's new in v3.56](#)

Databricks Status  
[View latest release notes](#)

# Lecture Instructions

---

- Let's start by creating a cluster
  - Go to “Common Tasks” -> “New Cluster”



# Lecture Instructions

---

- Configure your cluster and click on “Create Cluster”

Create Cluster

New Cluster | [Cancel](#) | [Create Cluster](#) | 0 Workers: 0 GB Memory, 0 Cores, 0 DBU  
1 Driver: 15.3 GB Memory, 2 Cores, 1 DBU [?](#)

Cluster Name: ISA414

Databricks Runtime Version [?](#): Runtime: 10.0 (Scala 2.12, Spark 3.2.0) | [▼](#)

**Note:** Databricks Runtime 8.x and later use Delta Lake as the default table format. [Learn more](#)

Instance: Free 15 GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please [upgrade your Databricks subscription](#).

Instances [Spark](#)

Availability Zone [?](#): auto | [▼](#)

# Computer Architecture

---

- Computers have 3 major components
  - CPU (Central Unit Processing)
    - Might have one or more “cores”
      - Each core processes standard instructions (such arithmetic operations) independently
      - Allows for parallel computing inside a single computer

# Computer Architecture

---

- Computers have 3 major components
  - Main (primary) memory
    - Operates at very high speed
    - Low capacity (storage space)
    - Expensive
    - Electricity based (volatile)
      - All data is lost after a computer is turned off
    - Technologies: RAM, DRAM, SRAM, ...

# Computer Architecture

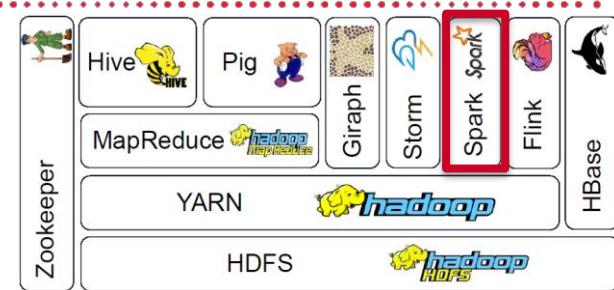
---

- Computers have 3 major components
  - Auxiliary storage (secondary memory)
    - Sometimes referred to as the “disk”
    - Slow to access information
    - High capacity (storage space)
    - Cheap
    - Non-volatile
      - Retain stored data even when a computer is powered off
  - Technologies
    - Hard disk (mechanical, magnetic storage)
    - Solid-state disk (SSD – no mechanical components)
    - ...

# The Hadoop Ecosystem

## ➤ Overview

- HDFS
  - Hadoop Distributed File System
  - Scalable and reliable storage
- Yarn
  - Schedule jobs/task over HDFS storage
- **Spark**
  - **Built for real-time, in-memory processing of data**



# Spark

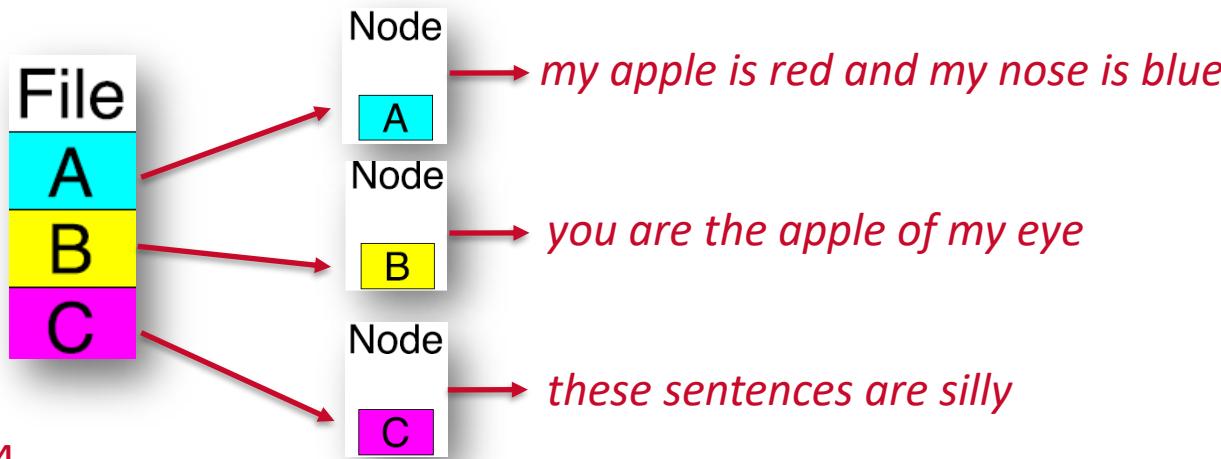
---

- The current “big thing” in predictive analytics
  - Originally developed by a PhD student at UC Berkeley in 2009
    - Currently managed by the Apache foundation
      - Initial release: 2014
  - Allows for distributed computation
    - More flexible than MapReduce
  - Easy to use
    - Many predefined distributed operations
      - Joins, filters, merge, ...
    - Many predefined machine learning algorithms
      - Decision trees, random forests, linear regression,...

# Quick Intro to MapReduce

---

- Suppose a very large textual data set is stored in a commodity cluster
  - We will define a MapReduce program to calculate the frequency of words in the data



# Quick Intro to MapReduce

## ➤ Map operation

- Executed in the node where the data block is stored
  - Moving computation to data
- Example
  - Key = word
  - Value = 1

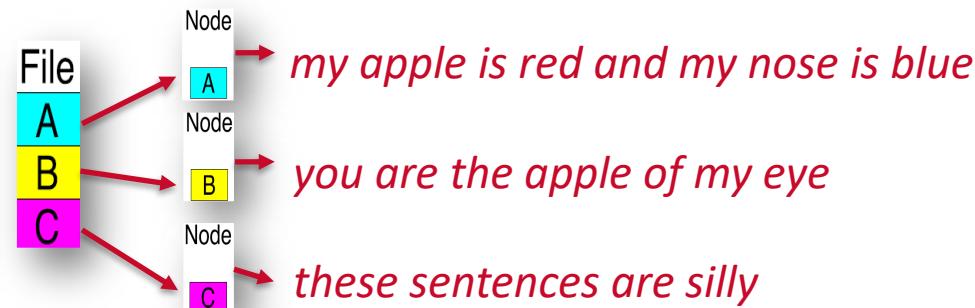
Node A → *my apple is red and my nose is blue*  
Node B → *you are the apple of my eye*  
Node C → *these sentences are silly*

|  |  | Outputs |       |         |       |
|--|--|---------|-------|---------|-------|
|  |  | Map (A) |       | Map (B) |       |
|  |  | Key     | Value | Key     | Value |
|  |  | my      | 1     | you     | 1     |
|  |  | apple   | 1     | are     | 1     |
|  |  | is      | 1     | the     | 1     |
|  |  | red     | 1     | apple   | 1     |
|  |  | and     | 1     | of      | 1     |
|  |  | my      | 1     | my      | 1     |
|  |  | nose    | 1     | eye     | 1     |
|  |  | is      | 1     |         |       |
|  |  | blue    | 1     |         |       |

# Quick Intro to MapReduce

## ➤ Sort and shuffle operation

- Nodes exchange data among themselves
  - Key-value pairs with the same key stay in the same node



| Node A |         | Node B |       | Node C    |       |
|--------|---------|--------|-------|-----------|-------|
| Key    | Value   | Key    | Value | Key       | Value |
| my     | (1,1,1) | you    | 1     | these     | 1     |
| apple  | (1,1)   | red    | 1     | sentences | 1     |
| is     | (1,1)   | are    | (1,1) | silly     | 1     |
|        |         | of     | 1     | and       | 1     |
|        |         | the    | 1     | nose      | 1     |
|        |         | eye    | 1     | blue      | 1     |
|        |         |        |       |           |       |

# Quick Intro to MapReduce

## ➤ Reduce operation

- Values with similar keys are aggregated
  - Aggregation technique must be defined by the code
- Outputs are saved back to HDFS (keys become unique)
  - A client can later request the aggregate results from HDFS
- Example:
  - Reduce = sum

| HDFS   |       |        |       |           |       |
|--------|-------|--------|-------|-----------|-------|
| Node A |       | Node B |       | Node C    |       |
| Key    | Value | Key    | Value | Key       | Value |
| my     | 3     | you    | 1     | these     | 1     |
| apple  | 2     | red    | 1     | sentences | 1     |
| is     | 2     | are    | 2     | silly     | 1     |
|        |       | of     | 1     | and       | 1     |
|        |       | the    | 1     | nose      | 1     |
|        |       | eye    | 1     | blue      | 1     |

client  
request

# Quick Intro to MapReduce

---

- Parallelization during MapReduce
  - Map: function applied to individual blocks of data in different nodes
  - Shuffle and sort: parallelization during sorting
  - Reduce: parallelization to aggregate individual results
- MapReduce is language independent
  - In theory, it can be implemented using virtually any programming language

# Back to Spark

---

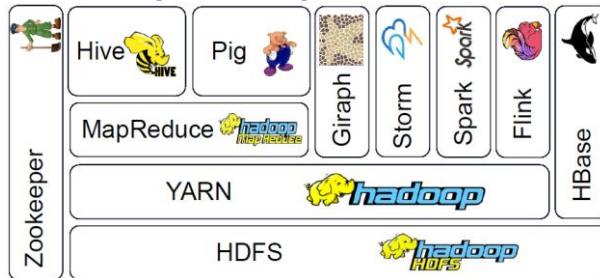
## ➤ Key benefits

- No need to explicitly define map and reduce tasks like in MapReduce
- In-memory caching of the data
  - Data are loaded into the nodes' main memory and often stay there until a task is done
    - Oftentimes, complex tasks are executed 10x to 100x faster than in the MapReduce framework
- Spark has a native programming language: Scala
  - Many programming language interfaces: Python, Java, R

# Spark

---

- Spark requires a job manager and a distributed storage system
  - Hadoop setup: YARN + HDFS



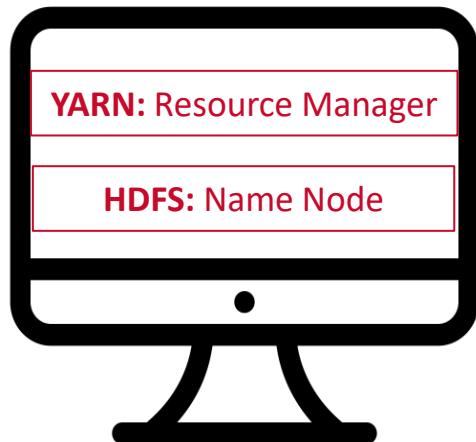
- Alternatives:
  - Job manager: native Spark cluster, Apache Mesos
  - Distributed storage: MapR, Cassandra, Amazon S3, Kudu

# Spark

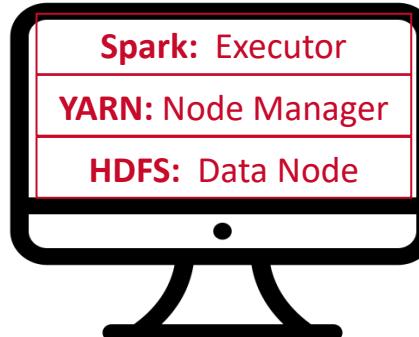
---

- Spark will run as another service inside nodes
  - The nodes running Spark form a “Spark cluster”

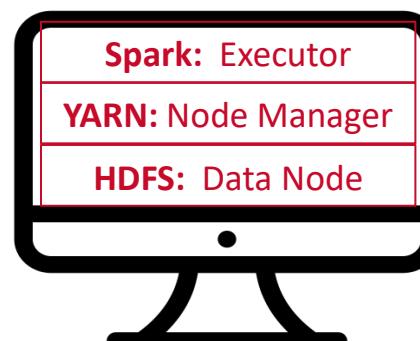
Master Node



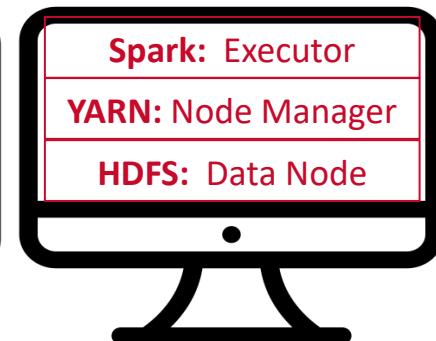
Worker Node 1



Worker Node 2



Worker Node 3



# Spark

---

## ➤ Key definitions

- Consider you have a data set distributed across a cluster of machines
- The data set is represented by a structure called *Resilient Distributed Dataset* (RDD) once loaded into the Spark cluster
- RDD's main characteristics:
  - *Distributed*: blocks of data are distributed across nodes in a cluster
  - *In-memory*: data inside RDDs are loaded and possibly kept into the main memory of the data nodes for rapid reuse
  - *Immutable*: RDDs cannot be changed
- RDD supports two operations: actions and transformations

# Spark

---

## ➤ Data pipeline



- Transformations: operations that return another RDD
  - *E.g.*, data manipulation (joins, filter, map, groupBy)
- Actions: operations that trigger a computation and return values
  - *E.g.*, count, max, min, reduce

# Spark

---

## ➤ Transformations

- Functions that take an RDD as the input and produce one or many RDDs as the output



# Spark

---

## ➤ Transformations

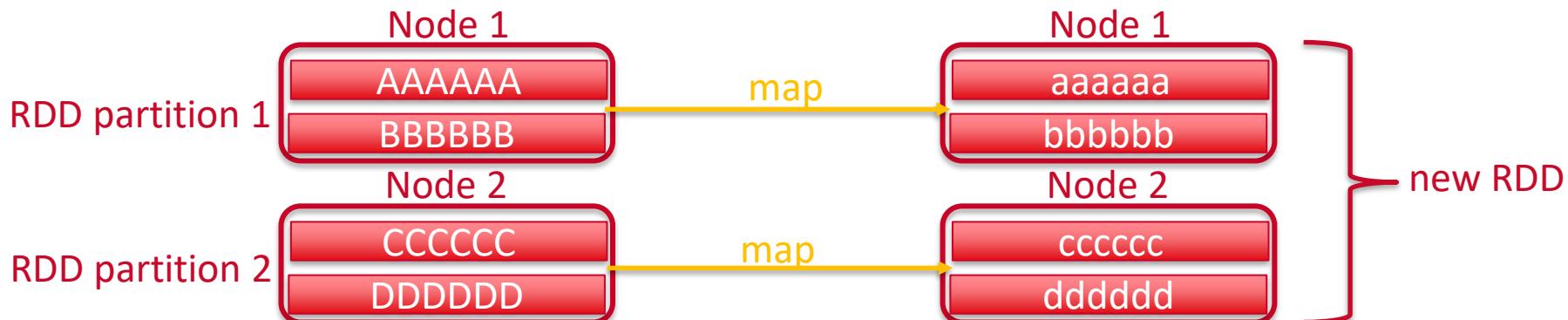
- Many transformations can be chained together
  - Each one produces an intermediate RDD



# Spark

## ➤ Transformations

- Example #1: **map** transformation = apply a function to each partition of an RDD
  - Suppose one has a massive textual file in HDFS and that file is loaded into a Spark cluster (the file becomes an RDD)
  - The analyst wants to transform all characters to lower case using the map transformation



# Spark

---

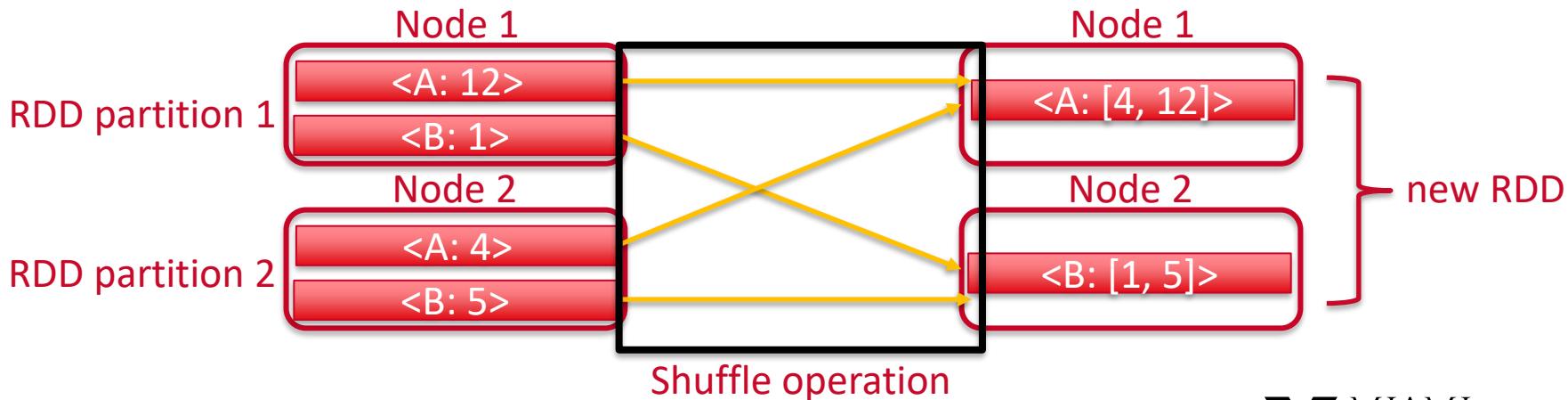
## ➤ Transformations

- Note that the **map** function is completely local
  - Each node's computations are independent of other nodes' computations
    - Computations are processed locally
  - These are called ***narrow transformations***
    - No transferring of data through the network
  - Transformations can also be ***wide***
    - Involve the transferring of data through the network
    - Consume more resources

# Spark

## ➤ Transformations

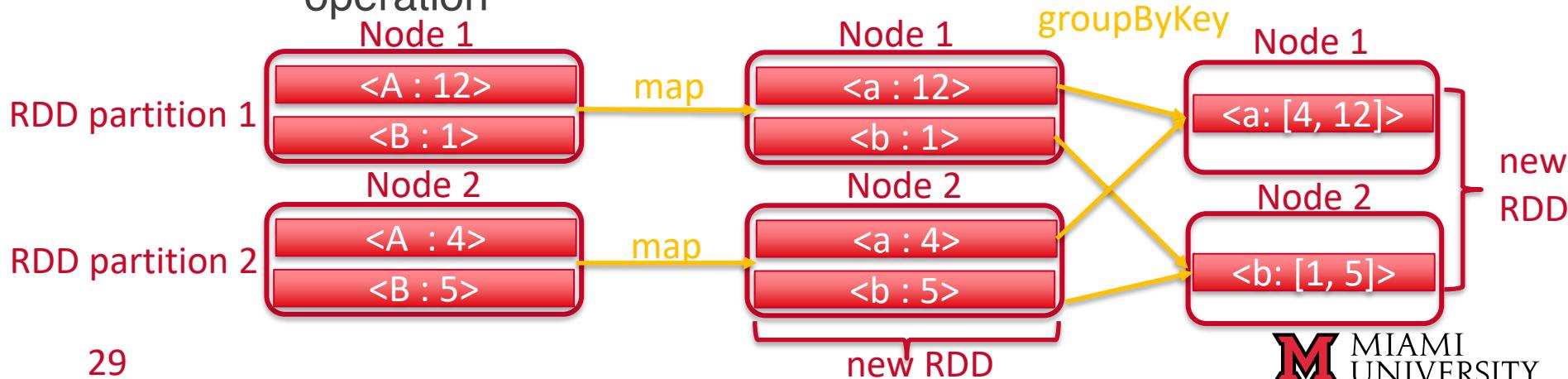
- Example #2: **groupByKey** transformation
  - Input: an RDD of key-value pairs
  - Output: transfers all the values that have the same key to the same partition
  - Example:



# Spark

## ➤ Transformations

- Clearly, many transformations can be chained together
  - Example #3: a map (to lowercase) followed by a groupByKey operation



# Spark

---

## ➤ Transformations

- There are many different transformations in Spark
  - The following list is not exhaustive
    - *map, filter, flatMap, mapPartitions, mapPartitionsWithIndex, sample, union, intersection, distinct, groupByKey, reduceByKey, aggregateByKey, sortByKey, join, cogroup, cartesian, pipe, coalesce, repartition, repartitionAndSortWithinPartitions*
- One great thing about Spark transformations is that their inputs/outputs are not necessarily key-value pairs

# Spark

---

- Transformations are *lazy*
  - They are not performed right away
  - When an action is called, Spark looks at the whole chain of transformations and creates an optimal execution plan
- Actions
  - Last step of the data workflow
    - Spark creates an execution plan and sends the tasks to the nodes
  - Produces a result/outcome

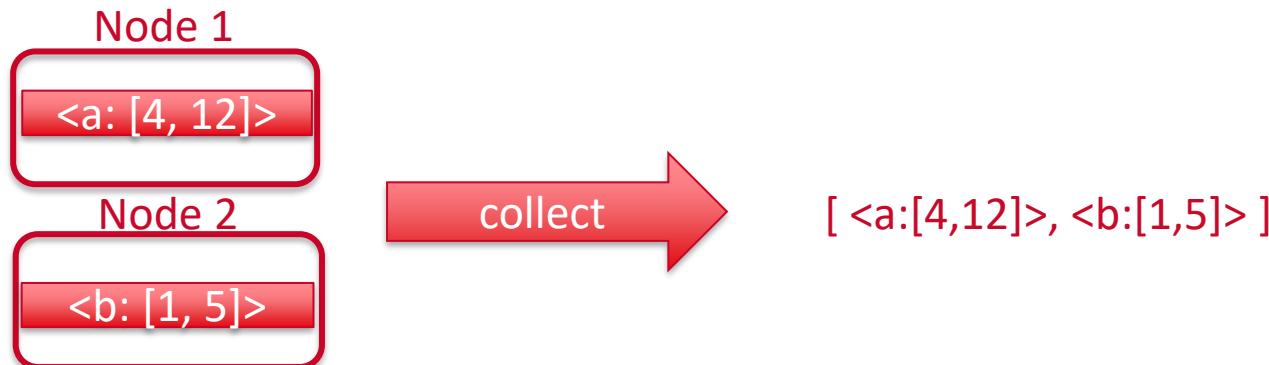
# Spark

---

## ➤ Actions

- Example: **collect**

- Returns the content of each RDD partition to an app/user



# Spark

---

## ➤ Actions

- There are many different actions in Spark
  - The following list is not exhaustive
    - *reduce, collect, count, first, take, takeSample, takeOrdered, saveAsTextFile, saveAsSequenceFile, saveAsObjectFile, countByKey, foreach*

# Spark

---

- One can perform distributed computations in Spark by calling transformations and actions
  - RDD programming model
  - Example in Scala: counting the number of occurrences of the word “spark” in a file

```
val data = spark.read.textFile("spark_test.txt").rdd
val mapFile = data.flatMap(lines => lines.split(" ")).filter(value => value=="spark")
println(mapFile.count())
```
- In practice, it is more likely that one will use one of the four *Spark libraries* built on top of the previously discussed concepts
  - They are incredibly easy to use
  - SQL, ML, Streaming, GraphX

# Spark

---

## ➤ Spark Libraries

- Spark SQL: implements relational queries on Spark
- Spark Streaming: implements incremental stream processing using a model called “discretized streams”
  - Transformations and actions are applied to small batches of data, such as every 200 milliseconds
- Spark GraphX: provides a graph computation interface ideal for social network analysis
- Spark MLlib: implements more than 50 common machine learning algorithms for distributed model training
  - Summary statistics, correlations, hypothesis testing, classification, regression, cluster analysis, dimensionality reduction, ...

# Databricks

---

➤ Let's go back to Databricks

- “*An open and unified data analytics platform*”
  - PaaS
  - Collaborative environment where analytics teams can work together
  - From the creators of Spark
  - Heavily used in industry
    - We face many limitations because we are using the free version
- One can easily create:
  - Clusters of machines running on top of Azure, AWS, Google Cloud
  - Notebooks and run popular ML frameworks (SK-learn, TensorFlow, Keras, Spark, ...)

# Databricks

---

- Your cluster should be ready by now
  - Clusters in the free version are deleted after a couple hours of inactivity

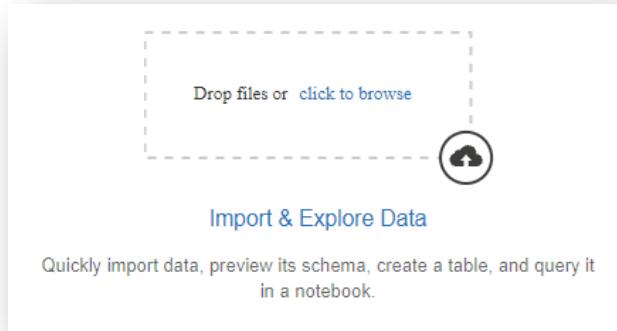
The screenshot shows the Databricks Cluster Overview page for a cluster named 'ISA414'. The cluster status is green. The top navigation bar includes 'Edit', 'Clone', 'Restart', 'Terminate', and 'Delete' buttons. Below the cluster name, tabs for 'Configuration', 'Notebooks', 'Libraries', 'Event Log', 'Spark UI', 'Driver Logs', 'Metrics', 'Apps', and 'Spark Cluster UI - Master' are visible. The 'Configuration' tab is selected. Key configuration details shown include:

- Databricks Runtime Version:** 10.0 (includes Apache Spark 3.2.0, Scala 2.12)
- Driver Type:** Community Optimized (15.3 GB Memory, 2 Cores, 1 DBU)
- Instance:** A message states: "Free 15 GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription."
- Instances Tab:** Shows tabs for 'Instances' (selected), 'Spark', 'JDBC/ODBC', and 'Permissions'.
- Availability Zone:** us-west-2c

# Databricks

## Let's upload a local file to our cluster

- Click on the Databricks logo on top-left 
- Drag and drop the file *moby dick.txt* to upload the file to the cluster
  - Copy the path to the file



Create New Table

Data source   S3 DBFS Other Data Sources Partner Integrations

DBFS Target Directory  /FileStore/tables/

Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)

Files 

|               |                                                                                     |
|---------------|-------------------------------------------------------------------------------------|
| moby dick.txt |  |
| 1.3 MB        | <input type="button" value="Remove file"/>                                          |

✓ File uploaded to /FileStore/tables/moby dick.txt



# Databricks

---

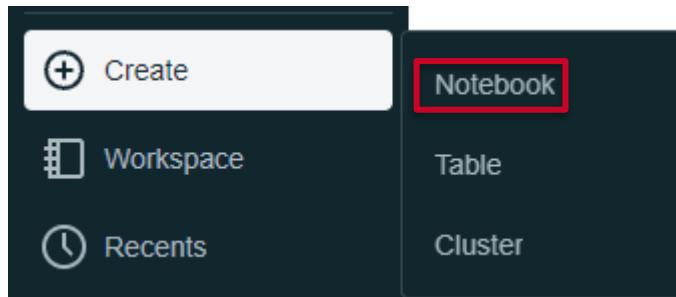
➤ So, are we using Hadoop HDFS here?

- No! we are using a similar technology called DBFS  
*“Databricks File System”*
  - Data lake
- Such a technology can retrieve data from any major cloud provider
  - Meaning that the data may be stored by Amazon, Microsoft, Google, etc.

# Databricks

---

- Let's create a Jupyter notebook running on our cluster
  - Click on the side menu-item “*Create*” -> “*Notebook*”
  - Fill in the notebook form



Create Notebook

Name

Default Language

Cluster

# Databricks

---

- Voila! We are a Jupyter notebook on top of a cluster
  - Spark is already configured for us
- Let's run the last test
  - Load textual data, run transformations and actions to count the number of times each word occurs in the book Moby-Dick
  - Ignore the code
    - We learn how to use Spark via its libraries, instead of transformations and actions

# Databricks

---

1. Testing Spark
  - Open the file *Lecture 23.ipynb* locally with VS code
2. Create four Python cells on Databricks
3. Copy and paste each Python cell from VS Code into Databricks
4. Click on “Run All” to run all cells on Databricks
  - Any errors?

# Summary

---

- We learned about Spark
  - Transformations and Actions
- We learned about Databricks
  - Great integration of Jupyter notebooks and Spark
  - Required for Assignment 4
- Next lecture: Spark (part II)
  - Libraries: Machine Learning & SQL

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent

# ISA 414 – Managing Big Data

## Lecture 24 – Introduction to Spark

(Part II)

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Agenda

- 11/16 and 11/18: presentation of (preliminary) project ideas
  - Goals:
    1. Strengthen your project ideas
    2. Soft skills: presentation
  - Duration
    - ISA 414: 15 minutes
    - ISA 514: 20 minutes
  - Presentation structure
    - Business background
    - Problem definition
    - Proposed solution (bird's-eye view)
    - Potential pros and cons

## MODERN DATA SCIENTIST

Data Scientist, the sexiest job of the 21th century, requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.

### MATH & STATISTICS

- ★ Machine learning
- ★ Statistical modeling
- ★ Experiment design
- ★ Bayesian inference
- ★ Supervised learning: decision trees, random forests, logistic regression
- ★ Unsupervised learning: clustering, dimensionality reduction
- ★ Optimization: gradient descent and variants

### PROGRAMMING & DATABASE

- ★ Computer science fundamentals
- ★ Scripting language e.g. Python
- ★ Statistical computing packages, e.g. R
- ★ Databases: SQL and NoSQL
- ★ Relational algebra
- ★ Parallel databases and parallel query processing
- ★ MapReduce concepts
- ★ Hadoop and Hive/Pig
- ★ Custom reducers
- ★ Experience with xaaS like AWS



### DOMAIN KNOWLEDGE & SOFT SKILLS

- ★ Passionate about the business
- ★ Curious about data
- ★ Influence without authority
- ★ Hacker mindset
- ★ Problem solver
- ★ Strategic, proactive, creative, innovative and collaborative

### COMMUNICATION & VISUALIZATION

- ★ Able to engage with senior management
- ★ Story telling skills
- ★ Translate data-driven insights into decisions and actions
- ★ Visual art design
- ★ R packages like ggplot or lattice
- ★ Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau

# Agenda

- 11/16 and 11/18: presentation of (preliminary) project ideas
  - Your project grade will also reflect this presentation
    - Hint: take it seriously and come prepared
  - Non-presenters
    - Your behavior during the presentations will be reflected in your individual, final grade
  - The presentation slides must be sent to the instructor by email 24h before the day of the presentation

## MODERN DATA SCIENTIST

Data Scientist, the sexiest job of the 21th century, requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.

### MATH & STATISTICS

- ★ Machine learning
- ★ Statistical modeling
- ★ Experiment design
- ★ Bayesian inference
- ★ Supervised learning: decision trees, random forests, logistic regression
- ★ Unsupervised learning: clustering, dimensionality reduction
- ★ Optimization: gradient descent and variants

### PROGRAMMING & DATABASE

- ★ Computer science fundamentals
- ★ Scripting language e.g. Python
- ★ Statistical computing packages, e.g. R
- ★ Databases: SQL and NoSQL
- ★ Relational algebra
- ★ Parallel databases and parallel query processing
- ★ MapReduce concepts
- ★ Hadoop and Hive/Pig
- ★ Custom reducers
- ★ Experience with xaaS like AWS



### DOMAIN KNOWLEDGE & SOFT SKILLS

- ★ Passionate about the business
- ★ Curious about data
- ★ Influence without authority
- ★ Hacker mindset
- ★ Problem solver
- ★ Strategic, proactive, creative, innovative and collaborative

### COMMUNICATION & VISUALIZATION

- ★ Able to engage with senior management
- ★ Story telling skills
- ★ Translate data-driven insights into decisions and actions
- ★ Visual art design
- ★ R packages like ggplot or lattice
- ★ Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau

# Agenda

---

- Presentation order (**Tuesday, 11/16**)
  - **ISA 514** (10:05 am – 11:25 am)
    - Group D: Alexander Amata, Jack Boyd, Will Dunn, Ben Keeley
    - Group A: Brendan Byrnes, Griffin Lester, Vish Nalagatla
  - **ISA 414** (1:15 pm – 2:35 pm)
    - Group D: Sierra Hessinger, Elianna Pecha, Rebekah Poth, Jacob Steele
    - Group A: John Doll, Abby Larson, Aubrey Liu, Spencer Townes
    - Group H: Hannah Le, Matt Madias, Carly Schechtman, Anthony Troiano
    - Group C: Charlie Fox, Ava Kunar, Tara Morrison, Nick Telerico

# Agenda

---

- Presentation order (**Thursday, 11/18**)
  - **ISA 514** (10:05 am – 11:25 am)
    - Group B: Maddie Banyas, Amara Cummins, Ashley Lonsinger, Meghana Muvva
    - Group C: Arnav Damodhar, Abay Ismailov, Jason Lantz
  - **ISA 414** (1:15 pm – 2:35 pm)
    - Group F: Natalie Day, Sam Groth, Aidan McGaughy, Andre Su
    - Group G: Benjamin Boczulak, Ben Cawley, Caroline Davis, Jack Laux
    - Group B: Cecilia Dauer, Julia Edelman, Macayla Temple
    - Group I: Nick Cimarusti, Mitch Gray, Thomas Hemsworth, Brendan Keck
    - Group E: Nina Gollapudy, Nicholas Hesselgesser, Esha Kallam, Chau Vu

# Agenda

---

- 11/23 and 11/30: in-class group work
- 12/02: review session

# Lecture Objectives

---

- Review Homework 10
- Discuss Assignment 4
- Continue to learn about Spark and its libraries
  - Learn how to use machine learning algorithms in Spark
    - ML Library
  - Learn how to use SQL to obtain data from RDDs in tabular format
    - SQL Library

# Lecture Instructions

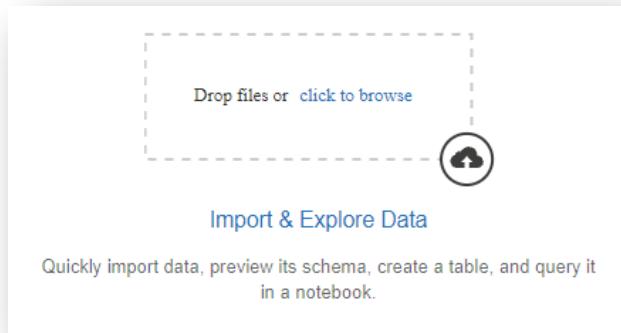
---

- Download the file *Lecture 24.ipynb* from Canvas
- Download the data set *noshow.csv* available on Canvas

# Lecture Instructions

---

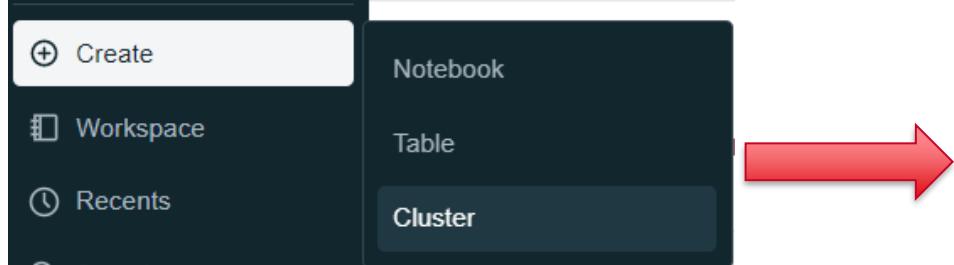
- Go to <https://community.cloud.databricks.com/>
  - Sign in
- Upload the file noshow.csv to Databricks
  - Copy the path to the file



A screenshot of the Databricks "Create New Table" interface. At the top, it says "Create New Table". Below that, "Data source" is set to "Upload File". The "DBFS Target Directory" field contains "/FileStore/tables/" with "(optional)" next to it. A "Select" button is to the right. Below these fields, a message states: "Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)". Under the "Files" section, there is a list with one item: "noshow.csv" with a green checkmark, "25.1 MB", and a "Remove file" link. At the bottom, a message says "✓ File uploaded to /FileStore/tables/noshow.csv". There are two buttons at the bottom: "Create Table with UI" and "Create Table in Notebook".

# Lecture Instructions

- Next, create a cluster
  - Go to the left panel, “Create” -> “Cluster”



Create Cluster

New Cluster

Cancel Create Cluster 0 Workers:0 GB Memory, 0 Cores, 0 DBU  
1 Driver:15.3 GB Memory, 2 Cores, 1 DBU ⓘ

Cluster Name: ISA514

Databricks Runtime Version ⓘ  
Runtime: 10.0 (Scala 2.12, Spark 3.2.0)

Note: Databricks Runtime 8.x and later use Delta Lake as the default table format. [Learn more](#)

Instance

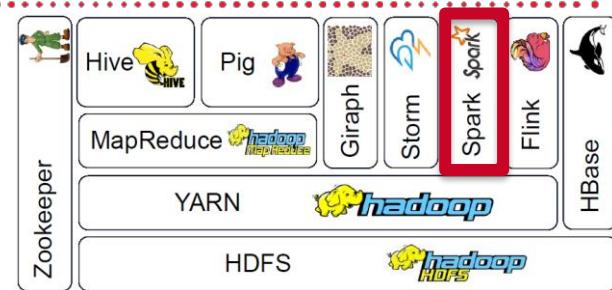
Free 15 GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.

Instances Spark

Availability Zone ⓘ  
auto

# The Hadoop Ecosystem

- Overview and agenda
  - HDFS
    - Hadoop Distributed File System
    - Scalable and reliable storage
  - Yarn
    - Schedule jobs/task over HDFS storage
  - MapReduce
    - Programming model that simplifies parallel/distributed computations
    - Two functions: Map (apply) and Reduce(summarize)
  - **Spark**
    - Built for real-time, in memory processing of data

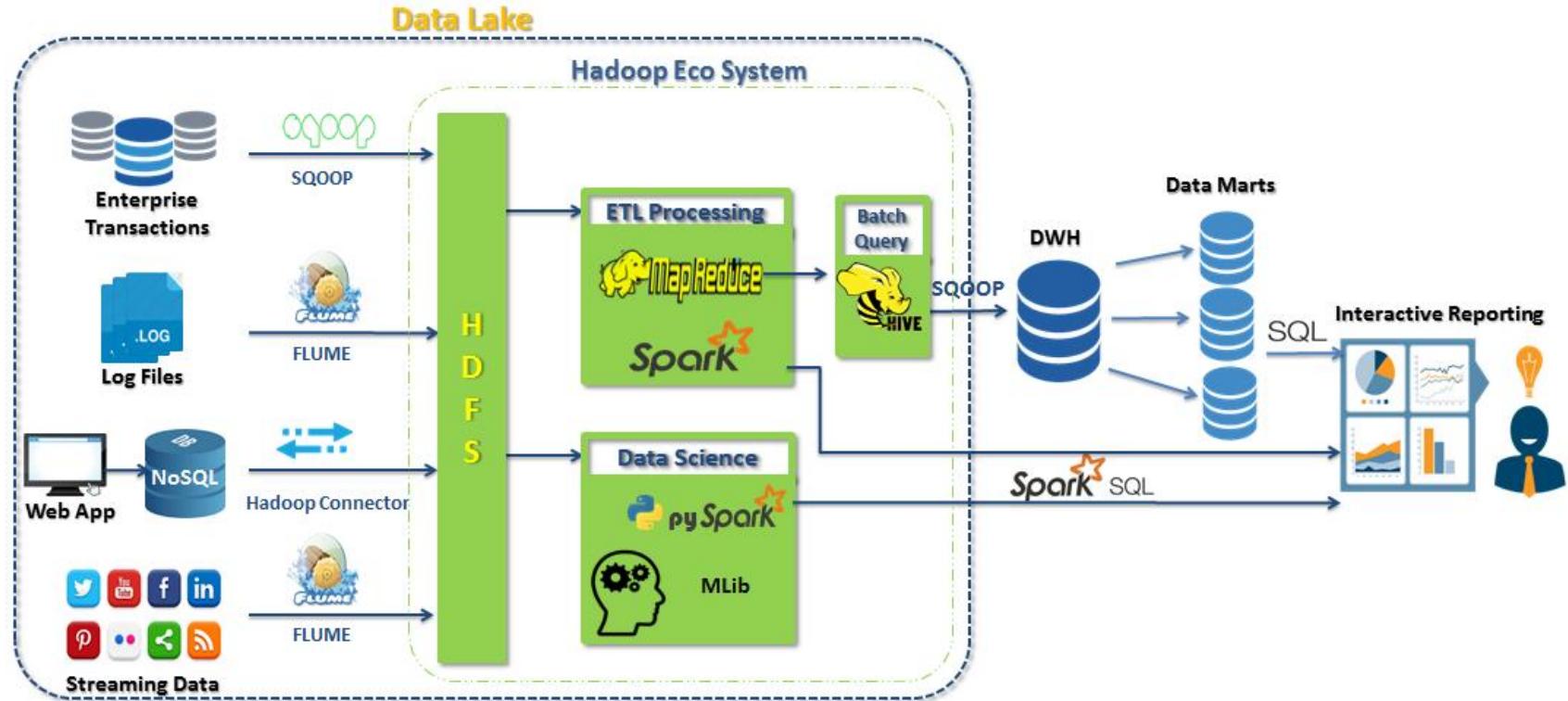


# Data Lake

---

- How does data get into Hadoop?
  - Standard approach
    - Hadoop services automatically extract and send data to HDFS
    - Examples:
      - *Sqoop*: transfers bulk data from relational databases to HDFS
      - *Flume*: transfers log data from web servers to HDFS
      - ...

# Data Lake



# Spark

---

## ➤ Spark

- Core definitions: RDD, transformations, and actions
- Libraries: Streaming, SQL, ML, GraphX
  - Our focus today: ML and SQL



# Spark

---

## ➤ Spark

- Always keep the following perspective in mind



We can simply use **Spark libraries** to send commands to a spark cluster

# PySpark

---

- Documentation
  - All what you need to know about PySpark:
    - <https://spark.apache.org/docs/latest/api/python/reference/index.html>
  - We shall see some of these functions in class today

# Spark

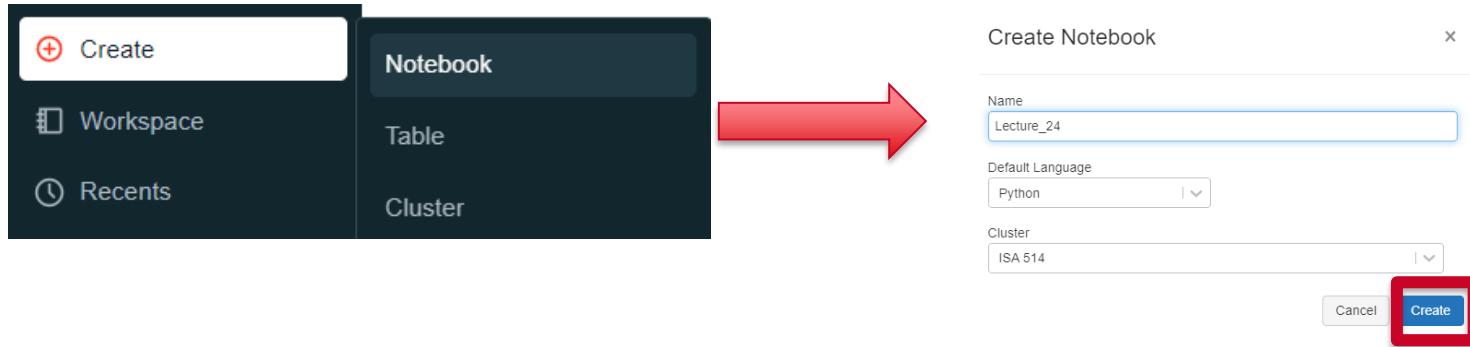
---

- No-show case study using Spark
  - Question: will a certain person who booked a doctor's appointment show up?
  - Background: real-life data from Brazil
    - Government-sponsored healthcare: "*I will not pay anything if I book an appointment and don't show up*"
    - No shows take spots from people who need the service
  - Idea: if we can predict who will show up, we can estimate the number of no-shows
    - Classification problem: dependent variable *Status*
    - Overbooking: good policy
      - That is what some airlines do

# Spark

---

- Let's connect to Spark
  - Create a notebook: Go to the left side panel, “Create” -> “Notebook”



- As we progress, copy the code from “*Lecture 24.ipynb*” into Databricks

# Spark

---

- Spark heavily relies on the concepts of *pipes* (or pipelines)
  - Many chained commands
    - The output of one command is the input to another one
  - Consequences
    - Shorter code
    - Fewer variables (lower memory consumption)

# Spark

---

- Loading a CSV file from a cluster to the main memory

```
# File location and type
```

```
file_location = "/FileStore/tables/noshow.csv"
```

```
# inferSchema := detect data types
```

```
# header := whether first row contains column names
```

```
raw_data = spark.read.load(file_location, format="csv", inferSchema="true", header="true")
```

- Retrieving and showing the first 20 rows in our data frame

```
raw_data.show()
```

# Spark

---

## ➤ Let's do some data preprocessing now

- Documentation:

<https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.html>

- Check variable types: `df.dtypes`

- Is the data set unbalanced?

```
df.select("Status").groupBy("Status").count().show()
```

# Spark

---

- Let's do some data preprocessing now
  - Let's reduce the number of values for the variable **DayOfTheWeek**
    - replace function: <https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.replace.html#pyspark.sql.DataFrame.replace>

```
df = df.replace(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"],  
                "WeekDay", "DayOfTheWeek")
```

```
df = df.replace(["Saturday", "Sunday"], "WeekEnd", "DayOfTheWeek")
```

# Spark

---

## ➤ Let's do some data preprocessing now

- Basic statistics for the variable **Age**

- **summary()** function: <https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.summary.html#pyspark.sql.DataFrame.summary>

```
df.select("Age").summary().show()
```

- Filtering negative **Age**

- **filter()** function: <https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.filter.html#pyspark.sql.DataFrame.filter>

```
df = df.filter(df["age"] >= 0 )
```

# ML Library

---

- It is time to build a predictive model
  - Like with sklearn, we have to transform qualitative variables to numeric values
    - Qualitative values to indexes: `StringIndexer` function
    - Qualitative values to 3+ dummies: `OneHotEncoder` function

```
from pyspark.ml.feature import StringIndexer
```

```
stringToIndex = StringIndexer(inputCol = 'Gender', outputCol = 'GenderIndex')  
df = stringToIndex.fit(df).transform(df)
```

```
stringToIndex = StringIndexer(inputCol = 'DayOfTheWeek', outputCol = 'DayOfTheWeekIndex')  
df = stringToIndex.fit(df).transform(df)
```

```
stringToIndex = StringIndexer(inputCol = 'Status', outputCol = 'StatusIndex')  
df = stringToIndex.fit(df).transform(df)
```

# ML Library

---

- It is time to build a predictive model
  - PySpark ML Library requires that all the predictors must be combined into a single vector

```
from pyspark.ml.feature import VectorAssembler
```

```
predictors = ['Age', 'GenderIndex', 'DayOfTheWeekIndex', 'Diabetes',
              'Alcoolism', "HiperTension", "Handcap", "Smokes",
              "Scholarship", "Tuberculosis", "Sms_Reminder",
              "AwaitingTime"]
```

```
assembler = VectorAssembler(inputCols=predictors, outputCol="predictors")
df = assembler.transform(df)
```

# ML Library

---

➤ It is time to build a predictive model

- Remember the discussion: training vs test sets

```
training_set, test_set = df.select(["StatusIndex","predictors"]).randomSplit([0.75,0.25])
```

- Training a random forest with 100 trees

```
from pyspark.ml.classification import RandomForestClassifier
```

```
model = RandomForestClassifier(numTrees=100,  
                               featuresCol= "predictors",  
                               labelCol='StatusIndex')
```

```
model = model.fit(training_set)
```

# ML Library

---

## ➤ Evaluation

- Predictions on the test set

```
predictions = model.transform(test_set)
```

- Model's overall accuracy

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
evaluator = MulticlassClassificationEvaluator(labelCol="StatusIndex",  
                                              predictionCol="prediction",  
                                              metricName = "accuracy")
```

```
accuracy = evaluator.evaluate(predictions)
```

```
print("Accuracy = %s" % (accuracy))
```

# SQL Library

---

- Hadoop has services that allow one to use SQL to retrieve data from structured files inside HDFS, such as CSV files
  - Spark SQL
  - Hive (not covered in this course)
  - Impala (not covered in this course)

# Spark

---

## ➤ Spark SQL

- Once a data set is loaded into a Spark cluster, SQL commands can be used to retrieve data from the RDD  
`raw_data.createOrReplaceTempView("TABLE")`
- SQL commands are translated into transformations and actions
  - Examples

`spark.sql("SELECT AGE FROM TABLE").show()`

`spark.sql("SELECT Status, COUNT(Status) FROM TABLE GROUP BY Status").show()`

`spark.sql("SELECT * FROM TABLE WHERE AGE < 0").show()`

# Spark

---

- You might be asking yourself: “*is this whole thing worth it?*”
  - “*I could do the same thing with fewer lines of code and running quicker without Spark*”
- Always keep in mind that Spark is used when there are tons of data and many available nodes
  - Only then, you can see how Spark speeds up data processing
  - Think about a data set having millions or even billions of appointments

# Summary

---

- We learned about Spark ML and SQL libraries
  - To use Spark in Python, one simply needs to learn about a bunch of predefined functions
  
- Next lecture: project (idea) presentations

Copyright 2021 Arthur Carvalho. All rights reserved. This material may not be published, broadcast, rewritten, or redistributed without explicit written consent

# ISA 414 – Managing Big Data

## Lecture 29 – Review Session

Dr. Arthur Carvalho

[arthur.carvalho@miamioh.edu](mailto:arthur.carvalho@miamioh.edu)



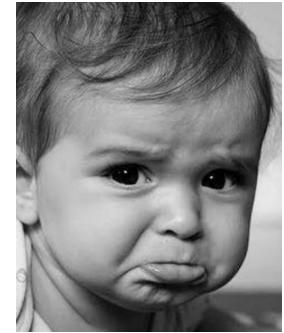
MIAMI UNIVERSITY

Copyright © 2021 Arthur Carvalho

# Announcements

---

- Last lecture :~(
  - Stay in touch:
    - Arthur: <https://www.linkedin.com/in/arthurgcarvalho/>



# Announcements

---

## ➤ Project

- Deadline: Sunday, December 5<sup>th</sup>, 11:59 pm
  - Report + code + raw data set(s)
  - I must be able to run your code on the provided data sets and reproduce all of your results
- Remember to fill in the “*Peer and Self Evaluation Form*” available on Canvas
  - You will only receive your individual project grade after submitting the evaluation form
  - Final, individual grades might differ from group grades
    - Group members who contributed to the project will have their grades raised
    - Group members who did not contribute as much to the project will have their grades lowered

# Announcements

---

## ➤ Conceptual final exam

- Available from Monday, December 6th (12 a.m.) to Saturday, December 11<sup>th</sup> (11:59 p.m.)
- No need to come to campus
  - Please, let me know if you would like to use the PC lab
- More on the format of the exam later in this lecture

# Announcements

---

## ➤ Course evaluations

- It is time to get some feedback from you
- Changes based on previous course evaluations:
  - Course difficulty
    - Yes, the course is easier now than it was before
  - Hands-on activities with IBM Cloud
  - Final exam format
  - No more classes on relational database
  - More time for the final project

# Announcements

---

## ➤ Course evaluations

- What worked and what could be improved?
- Please, let me know your thoughts (5-minute survey)
  1. Go to [mymiami.miamioh.edu](http://mymiami.miamioh.edu)
  2. In the “My Courses” tab, click on “Course Evaluations” for the course **ISA 414/514 – Managing Big Data**

# Lecture Objectives

---

- Review Assignment 4
- Introduction to the format of the final exam
- Review the major concepts learned in the course
  - Focus on the final exam

# Final Exam

---

- Online
  - Available from December 6<sup>th</sup> (12 a.m.) to December 11<sup>th</sup> (11:59 p.m.)
- Duration: 2h
- You are allowed to use any search engine, slides, notes, ...
  - Be very careful with the amount of time you spend on search engines or looking at notes
- No cell phones, second monitors
- No access to websites that allow for collaboration
- No more than one person in the room

# Final Exam

---

- Format: similar to previous quizzes
  - Multiple-choice questions
  - Some questions have more than one correct answer
- 25 questions in total, 4 points each
  - Randomly drawn from a database containing 100 questions
  - Covers every single topic we discussed in class
    - No Python code

# Final Exam

---

- Security measures (please, pay attention to this)
  1. We will use Proctorio for proctoring
  2. The orders of questions and answers are random
  3. You have only one shot
    - Your exam grade is final

.....

# REVIEW SESSION

# Big Data

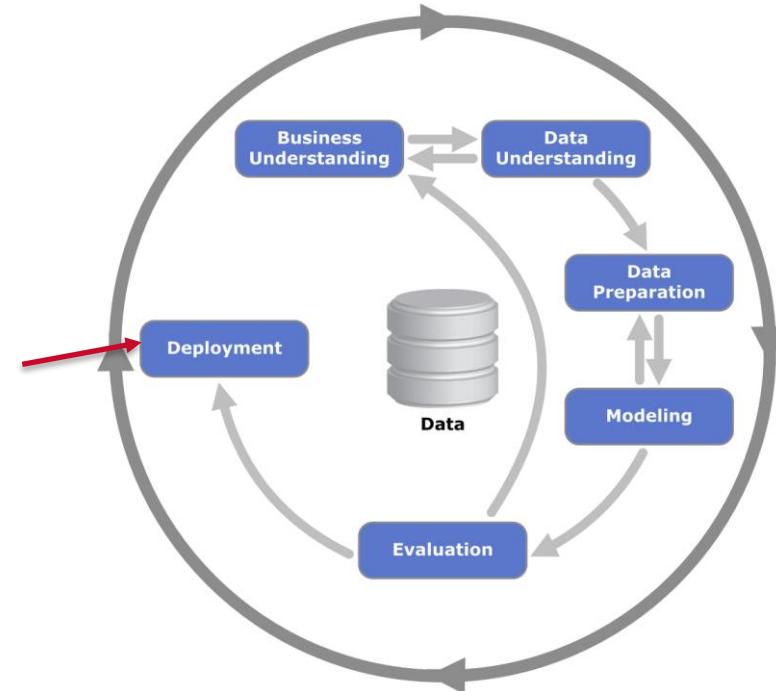
---

- Managing big data
  - Big data
    - Defined by a number of “Vs”
      - Volume: vast amount of data
      - Variety: different forms of data (text, images, voice, geospatial, etc.)
      - Velocity: speed at which data is generated/analyzed
      - Veracity: biases, noise, abnormalities, uncertainties, truthfulness, trustworthiness of the data

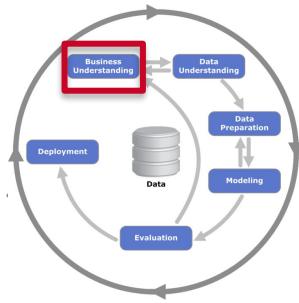
# CRISP-DM

- Key concept: CRISP-DM cycle

not covered in  
this course



# Business Understanding



- Can we translate the business problem into a data analytics task (or many subtasks)?
- Supervised vs unsupervised learning techniques
  - Supervised means there is a clear target we use when training models
- Occasionally, we do not need statistical models
  - Descriptive analytics is enough to solve a business problem

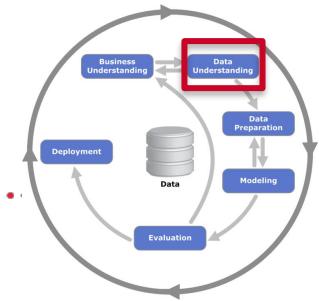
covered  
in this  
course

→

| Problem                | Supervised | Unsupervised |
|------------------------|------------|--------------|
| Classification         | X          |              |
| Regression             | X          |              |
| Similarity Matching    | X          |              |
| Clustering             |            | X            |
| Co-occurrence grouping |            | X            |
| Profiling              |            | X            |

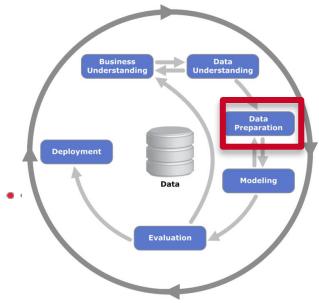
# Data Understanding and Collection

- We learned about:
  - Web & log scraping
    - Regular expressions
  - API
    - Request: REST
    - Response: JSON and XML
  - Database queries
    - Document-oriented databases (JSON)



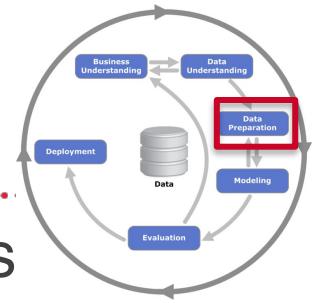
# Data Preparation/Preprocessing

- Raw data is cleaned/prepared for analysis
- Traditional steps:
  - Remove/impute invalid/missing values
  - Merge duplicate observations and/or different data sets
  - Transform/remove outliers
- Feature selection
  - Remove highly correlated predictors
  - Scaling
  - Dimensionality reduction



} not covered in this course

# Data Preparation/Preprocessing

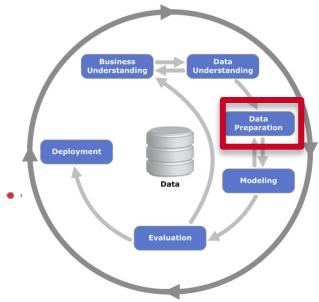


- We have focused on preparing textual data for analysis
  - Bag of words: each word is considered individually (“token”)
  - DTM: document-term matrix
    - Rows = documents
    - Columns = words
    - Cells = counts
      - TF: measures the popularity of a word inside a document
      - IDF: measures the popularity of a word in a corpus
      - TFIDF: measures the popularity of a word in a document and across the whole corpus

# Data Preparation/Preprocessing

## ➤ Other common techniques

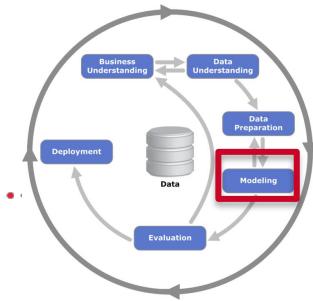
- Remove stop words
  - Remove numbers
  - Stemming
  - Sparsity reduction
  - N-grams
- } dimensionality reduction
- } dimensionality increase



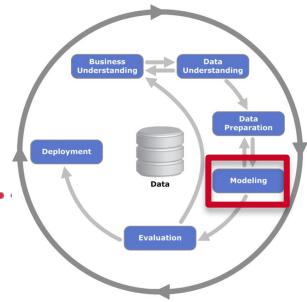
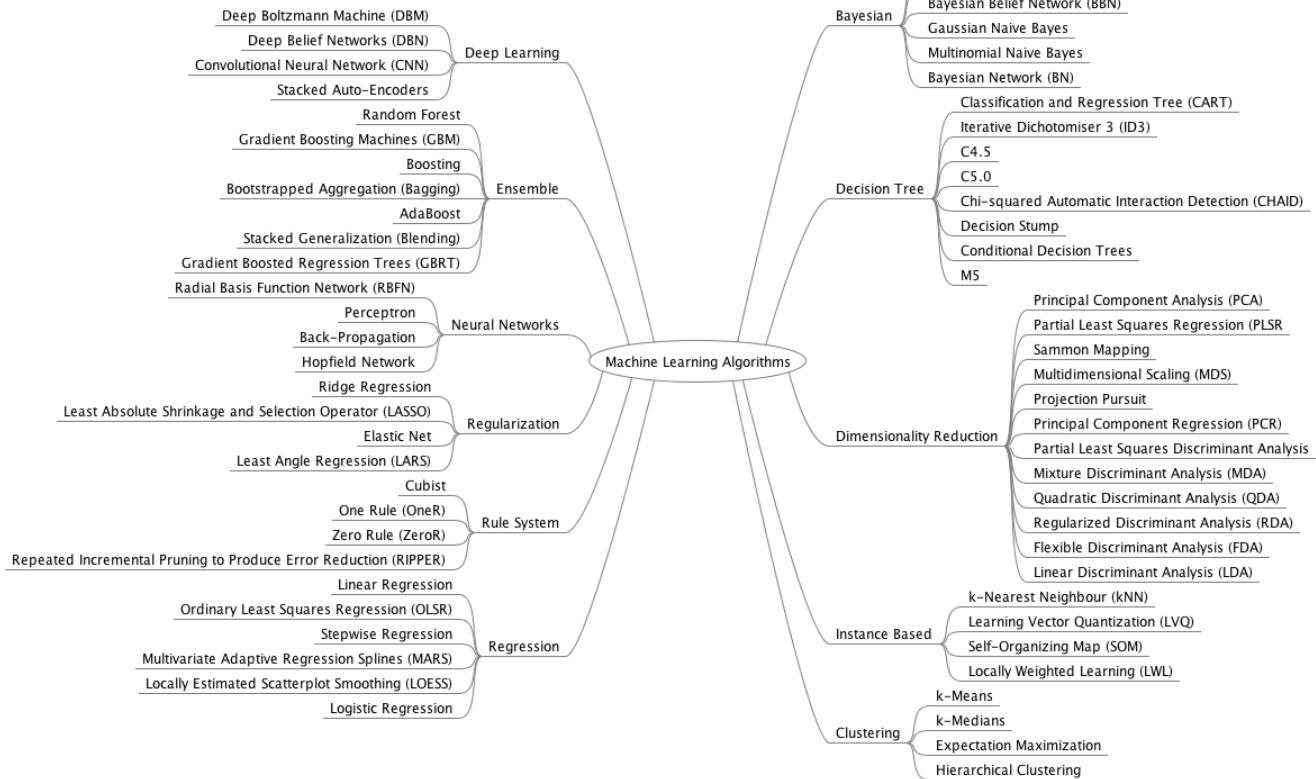
# Data Modeling/Analysis

- We have learned a few techniques to analyze data

- Textual data
  - Sentiment analysis
  - LDA
- Statistical modeling
  - Classification: decision trees and random forests
  - Regression: regression trees and random forest
- Keep in mind that there are many other techniques that are beyond the scope of this course



# Data Modeling/Analysis



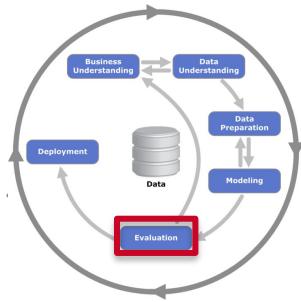
data

# machine learning algorithm



# Evaluation

---



- Many different ways of evaluating a data analytics solution
  - Internal/external experts
  - Statistical analysis of errors when making predictions
    - Build the model using a training set
    - Evaluate the model using a test (holdout) set
    - Different metrics
      - Classification: overall accuracy, specificity, sensitivity, ROC area
      - Regression: MSE, RMSE, MAE, MAPE ...

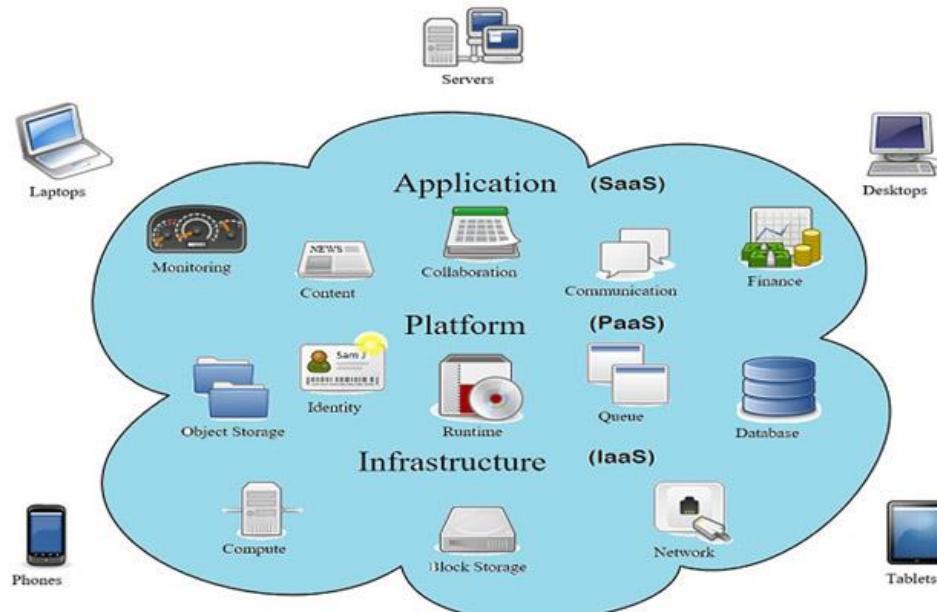
# Review

---

- Big data brings new, complex challenges
  - Volume, variety, velocity, veracity
- Second part of the course: two big-data analytics enablers
  - Cloud computing and storage
    - IBM Cloud, Microsoft Azure, Google Cloud Platform, Amazon Web Services, ...
  - Distributed storage and computing
    - Hadoop environment and Spark

# Cloud Technologies

- Cloud = rental agreement
  - Different levels of engagement and servicing



# Cloud Technologies

---

## ➤ Things to remember

- Definitions
  - IaaS: Infrastructure as a Service
  - PaaS: Platform as a Service
  - SaaS: Software as a Service
  - XaaS: Everything as a Service
- In-house vs cloud infrastructure
  - Considerations: Costs, Demand, Skills, Security

# Cloud Technologies

---

- **Q1:** consider a company whose e-commerce sales grow massively during holiday season (e.g., Best Buy). The company's in-house database system suffers tremendously from such an increase in online traffic, which causes the company's website to run slowly. When it comes to the IT infrastructure, what would likely be the most cost-effective and efficient approach to handle the extra online traffic during the holiday season?
- a) Sign an IaaS agreement: the company should move its whole infrastructure to the cloud
  - b) Sign a PaaS agreement: the company should move its whole infrastructure to the cloud
  - c) Sign a SaaS agreement: the company should follow a hybrid approach where the database workload is distributed between the in-house and cloud infrastructure
  - d) Sign a PaaS agreement: the company should follow a hybrid approach where the database workload is distributed between the in-house and cloud infrastructure

# Cloud Technologies

---

- **Q2:** consider a company that relies on data-analytics sporadically. As such, the company has no dedicated data scientists and no relevant skill set. Suppose that company is now planning a predictive-analytics project. Luckily, the data set to be used was already used in previous projects, meaning that it is ready for analysis. What would likely be the most cost-effective approach(es) for that company?
- a) Sign a PaaS agreement: the company should obtain a complete data-science platform, including IDE and distributed storage/computation frameworks
  - b) Hire several seasoned data scientists to complete the task
  - c) Sign an XaaS agreement that automates the development and deployment of statistical models

# Hadoop

---

- Cluster of commodity computers

- **Distributed storage**

- Individual files are stored in different computers in a cluster
    - Massive files are broken down into chunks of data, which are stored in different computers (nodes) in a cluster
    - Sharing ‘disks’ (secondary storage)

- **Distributed computation**

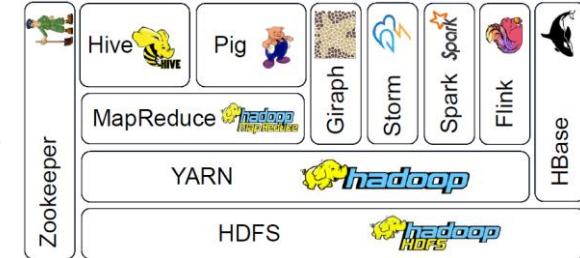
- Complex, time-consuming computations are distributed across different nodes
    - Sharing CPUs/cores and main memory
    - Paradigm: “move computation to data”
      - Reduce network traffic

# Hadoop

---

➤ **Hadoop:** framework used for distributed storage and computing

- A tool that manages commodity clusters
- A collection of technologies
  - HDFS: Hadoop Distributed File System: distributed storage
  - Yarn: schedule jobs/task over HDFS storage
  - MapReduce: programming model that simplifies parallel/distributed computing
  - Spark: built for real-time, in memory processing of data



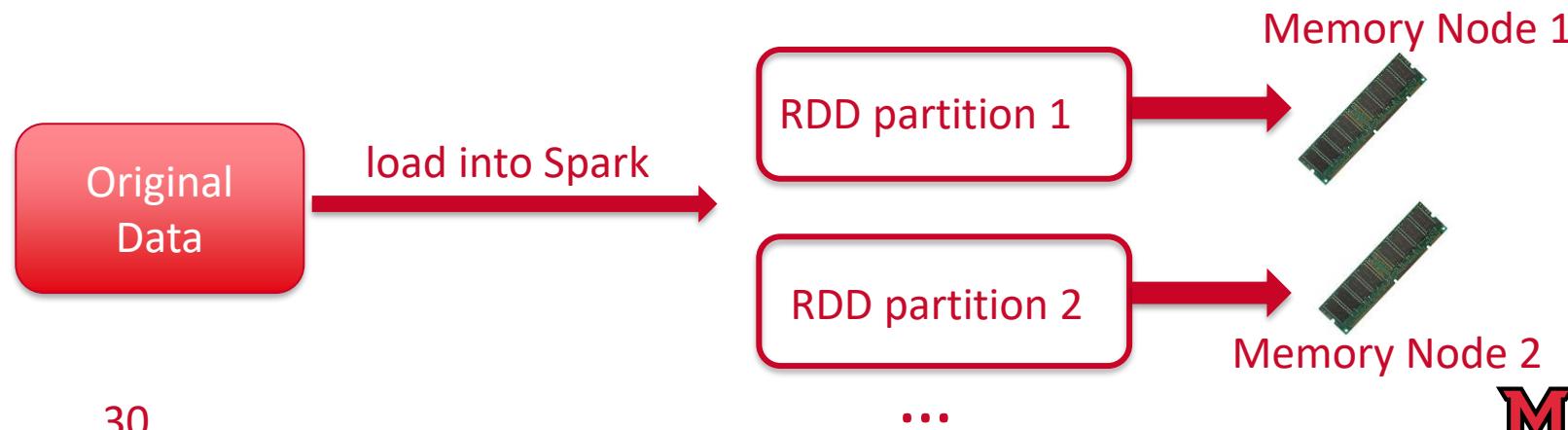
# MapReduce

---

- Things to remember: a computational model to perform distributed computations
- Consists of three main steps: Mapping, Sort & Shuffling, and Reduce operations
  - The programmer only defines the mapping and reduce steps
    - Sort & shuffling is done behind the scenes
    - Literally, two functions: Map() and Reduce()
      - Map = computation to be executed in each block
        - **Uses key-value pairs**
      - Reduce = summary/grouping of the outputs produced by the nodes
        - **Uses key-value pairs**

# Spark

- The current “big thing” in data analytics
  - Allows for distributed computation
    - More flexible than MapReduce
    - In-memory processing



# Spark

---

- In practice, it is very common to use Spark libraries, which are built on top of transformations/actions, in conjunction with programming languages such as R, Python, and Java



# Spark

---

- Similar to MapReduce, one can only see the true power of Spark when working with big data sets and many nodes
- **Q3:** what are the reasons for the current trend where many companies are replacing MapReduce with Spark?
  - a) Unlike MapReduce, Spark is free
  - b) Spark tends to be 10x to 100x faster than MapReduce due to being an in-memory technology
  - c) Spark does not require the underlying code to process data as key-value pairs
  - d) Besides HDFS, Spark works well with other distributed-storage technologies

# Final Considerations

- Analytics is not equal to statistics or computer science
  - Understanding technologies and business processes also matter
  - **Things we experienced in this course**
    - Make sure you share your experience with recruiters
    - Yes, there is a lot more to learn
  - Remember: the perfect data scientist is like a unicorn
    - The most important skill: **be passionate about learning**
      - Hacker mindset

# MODERN DATA SCIENTIST

Data Scientist, the sexiest job of the 21th century, requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.

## MATH & STATISTICS

- ★ Machine learning
- ★ Statistical modeling
- ★ Experiment design
- ★ Bayesian inference
- ★ Supervised learning: decision trees, random forests, logistic regression
- ★ Unsupervised learning: clustering, dimensionality reduction
- ★ Optimization: gradient descent and variants



## PROGRAMMING & DATABASE

- ★ Computer science fundamentals
- ★ Scripting language e.g. Python
- ★ Statistical computing packages, e.g., R
- ★ Databases: SQL and NoSQL
- ★ Relational algebra
- ★ Parallel databases and parallel query processing
- ★ MapReduce concepts
- ★ Hadoop and Hive/Pig
- ★ Custom reducers
- ★ Experience with xaaS like AWS

## DOMAIN KNOWLEDGE & SOFT SKILLS

- ★ Passionate about the business
- ★ Curious about data
- ★ Influence without authority
- ★ Hacker mindset
- ★ Problem solver
- ★ Strategic, proactive, creative, innovative and collaborative

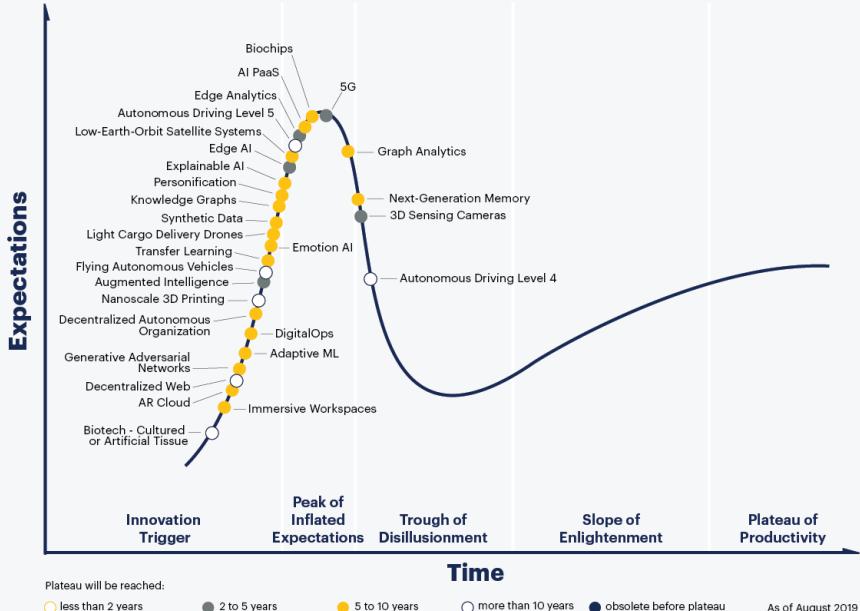
## COMMUNICATION & VISUALIZATION

- ★ Able to engage with senior management
- ★ Story telling skills
- ★ Translate data-driven insights into decisions and actions
- ★ Visual art design
- ★ R packages like ggplot or lattice
- ★ Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau

# Final Considerations

- Always be familiar with the newest technologies and trends

## Gartner Hype Cycle for Emerging Technologies, 2019



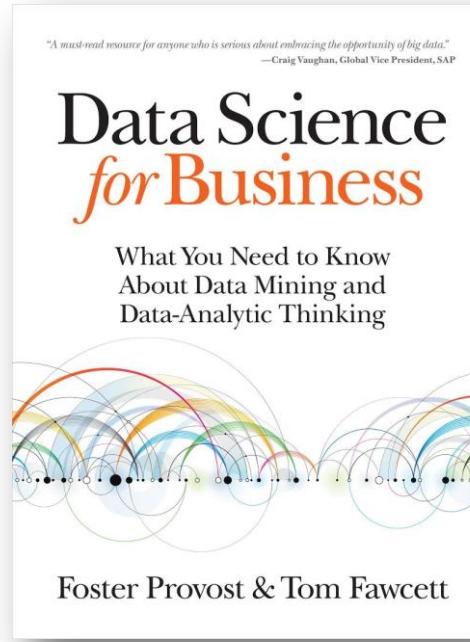
[gartner.com/SmarterWithGartner](http://gartner.com/SmarterWithGartner)

Source: Gartner  
© 2019 Gartner, Inc. and/or its affiliates. All rights reserved.

# Final Considerations

---

- Winter project idea:
  - Read the following book from cover to cover
    - Easy to follow: no mathematics
    - Several examples of how to apply data science to real-life business problems
    - Authors are very famous analytics researchers
    - Book I would use if I was teaching MBA-level analytics courses



# Final Considerations

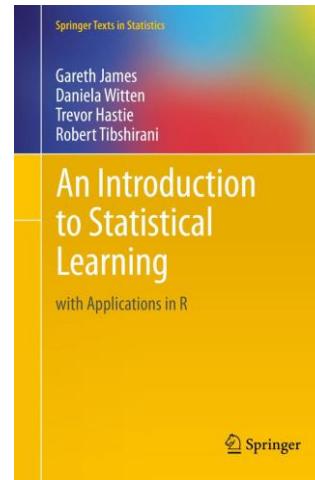
---

➤ Winter project idea:

- Read the following book from cover to cover
  - Free (and legal) pdf copy available at:

[https://web.stanford.edu/~hastie/ISLR2/ISLRv2\\_website.pdf](https://web.stanford.edu/~hastie/ISLR2/ISLRv2_website.pdf)

- Easy to follow: not a lot of mathematics
- Several examples (in R)
- Authors are very famous statisticians
- Book I would use if I was teaching ISA 491



# Final Considerations

---

