Virtual Environments & Virtual Machines

CS107/AC207

Pavlos Protopapas
Institute for Applied Computational Science, Harvard

Outline

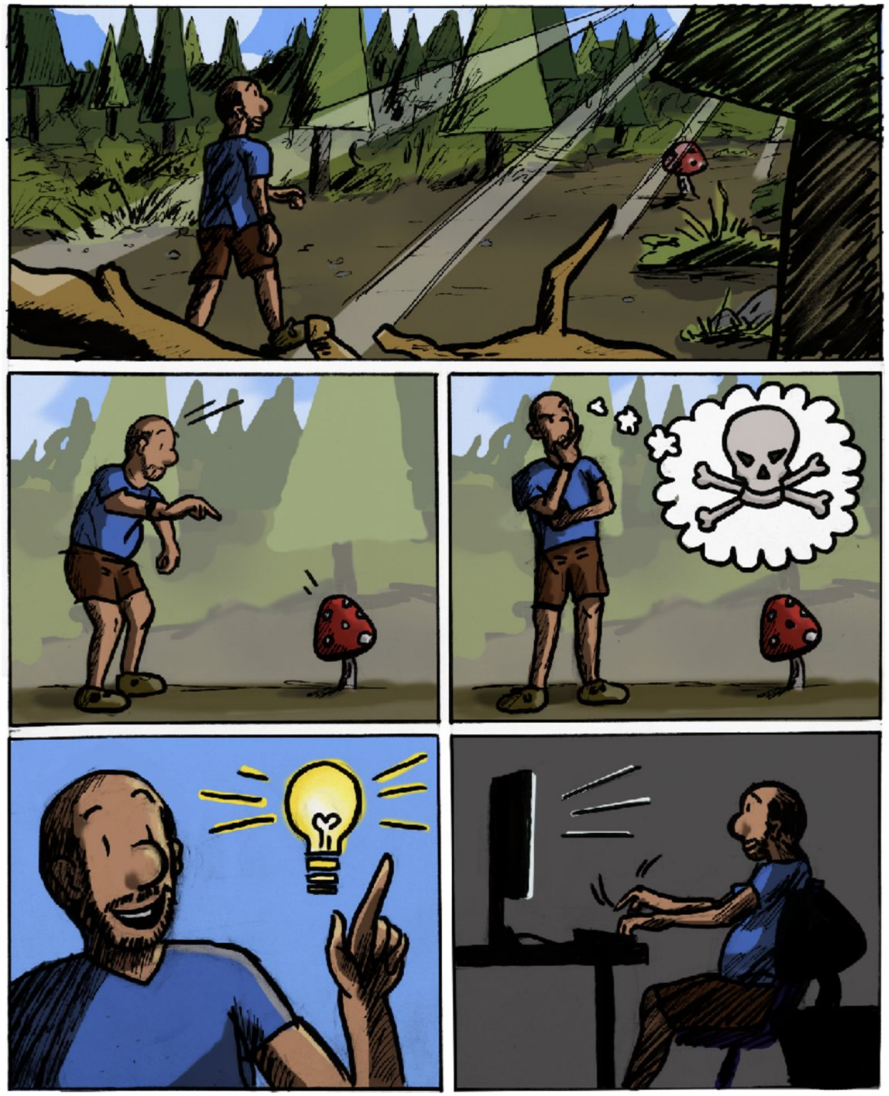
1. Motivation
2. Virtual Machines
3. Virtual Environments

Outline

1. **Motivation**
2. Virtual Machines
3. Virtual Environments

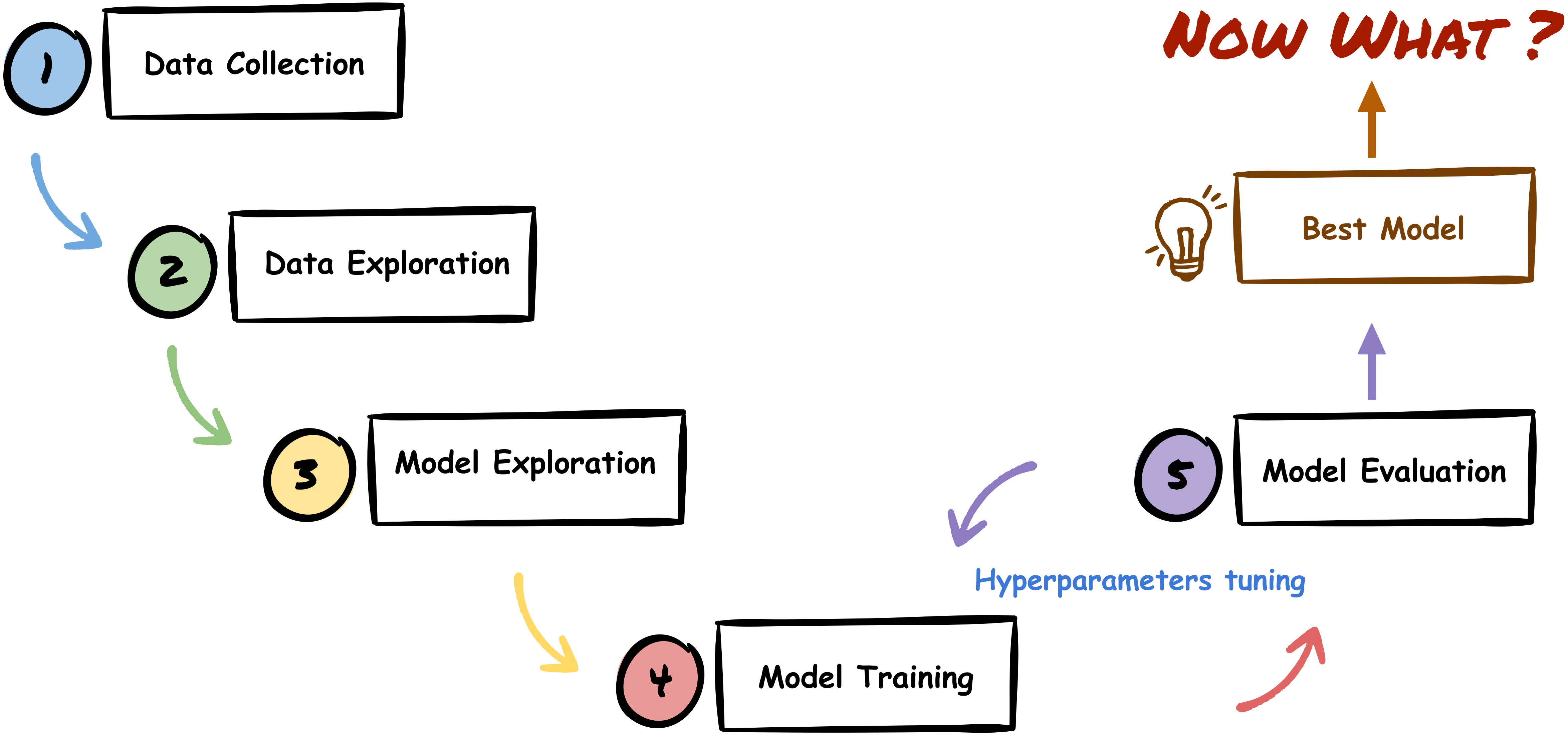
We want to build a 🍄 Mushroom Finder App

- Pavlos likes to go the forest for mushroom picking
- Some mushrooms can be poisonous
- Help build an app to identify mushroom type and if poisonous or not



Credit: Nikolas Protopapas

ML/Deep Learning Flow



1 Data Collection

2 Data Exploration

3 Model Exploration

4 Model Training

5 Model Evaluation

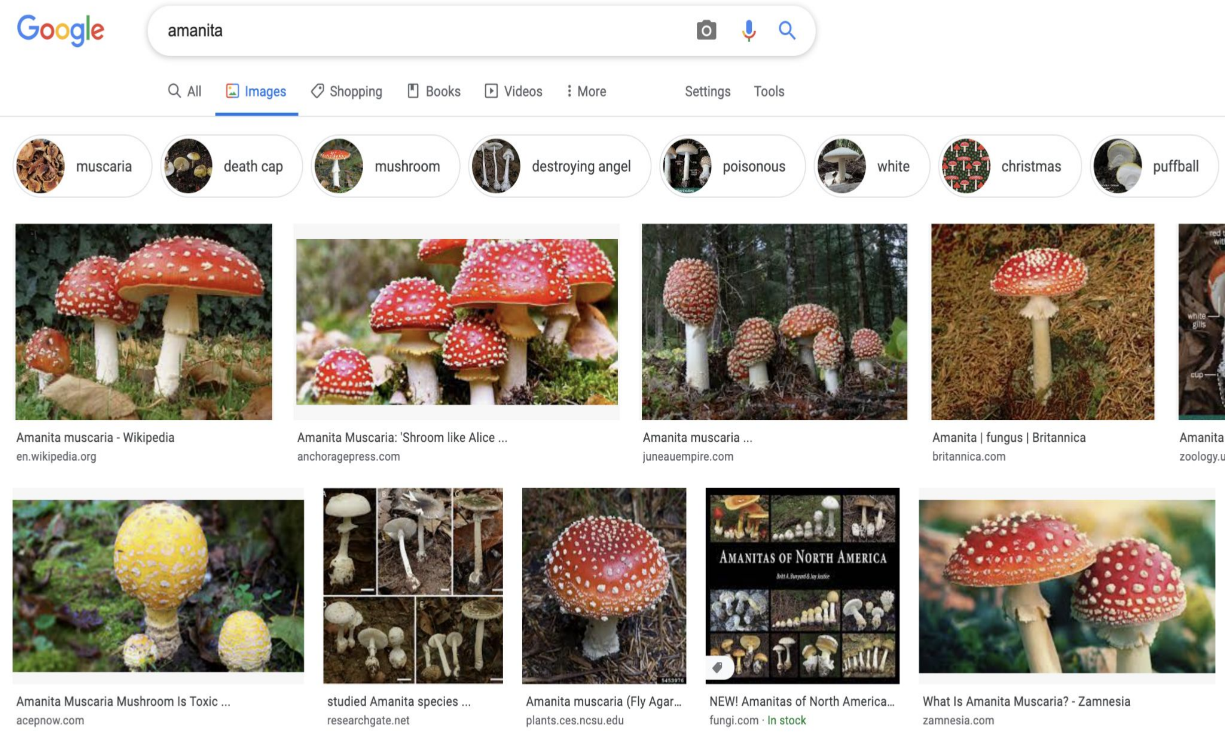
Hyperparameters tuning

Best Model

NOW WHAT ?

🍄 Mushroom App: Data Collection

- Collect images from Google
- For our demo we downloaded images for mushrooms oyster, crimini, **amanita (Poisonous)**



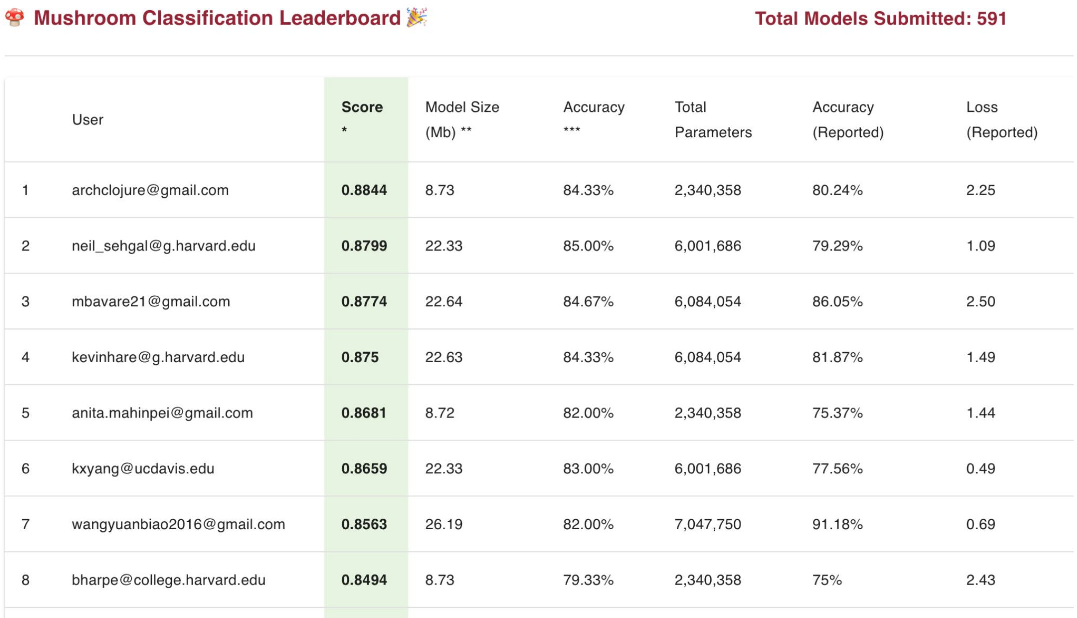
amanita

Python Script

🍄 Mushroom App: Models

- Identify our problem task
- Try various model architectures
- Transfer Learning
- Hyperparameters tuning

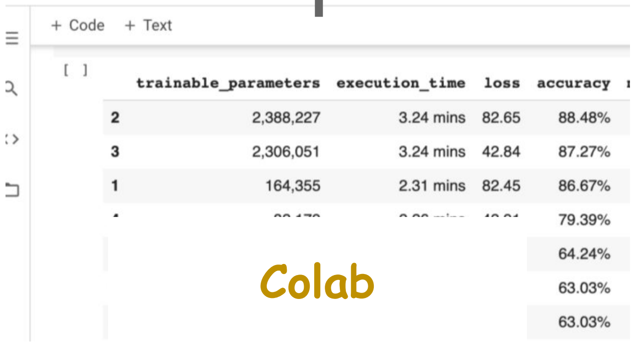
AC215



🍄 **Mushroom Classification Leaderboard** 🎉

Total Models Submitted: 591

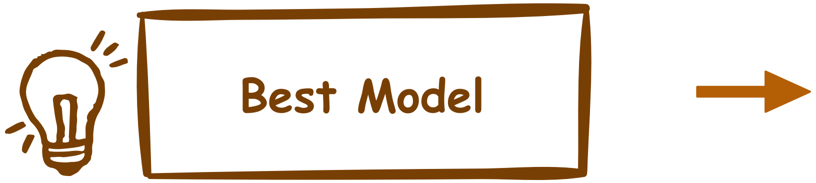
	User	**Score** *	Model Size (Mb) **	Accuracy ***	Total Parameters	Accuracy (Reported)	Loss (Reported)
1	archclojure@gmail.com	**0.8844**	8.73	84.33%	2,340,358	80.24%	2.25
2	neil_sehgal@g.harvard.edu	**0.8799**	22.33	85.00%	6,001,686	79.29%	1.09
3	mbavare21@gmail.com	**0.8774**	22.64	84.67%	6,084,054	86.05%	2.50
4	kevinhare@g.harvard.edu	**0.875**	22.63	84.33%	6,084,054	81.87%	1.49
5	anita.mahinpei@gmail.com	**0.8681**	8.72	82.00%	2,340,358	75.37%	1.44
6	kxyang@ucdavis.edu	**0.8659**	22.33	83.00%	6,001,686	77.56%	0.49
7	wangyuanbiao2016@gmail.com	**0.8563**	26.19	82.00%	7,047,750	91.18%	0.69
8	bharpe@college.harvard.edu	**0.8494**	8.73	79.33%	2,340,358	75%	2.43



	trainable_parameters	execution_time	loss	accuracy
2	2,388,227	3.24 mins	82.65	88.48%
3	2,306,051	3.24 mins	42.84	87.27%
1	164,355	2.31 mins	82.45	86.67%
[illegible]	[illegible]	[illegible]	[illegible]	79.39%
				64.24%
				63.03%
				63.03%

Colab

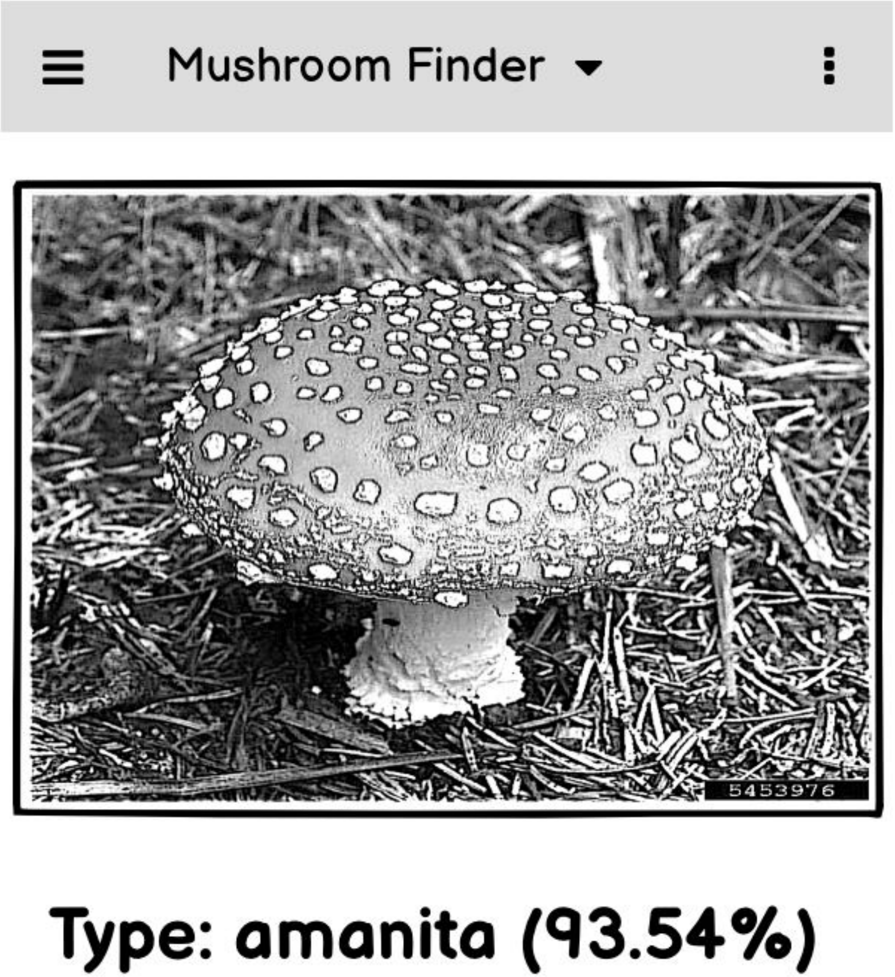
🍄 Mushroom App: Best Model



NOW WHAT ?

🍄 Mushroom App

- We want to build an app to take a photo of a mushroom and it helps us identify the type of mushroom
- How do we build the app?



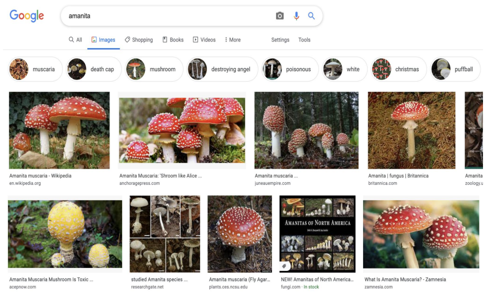
Type: amanita (93.54%)

How do we build an App?

- Collaborate with team to design and develop
- Expose best model as an API
- Build a frontend using HTML & javascript
- Deploy app to a cloud provider
- http://awesome-mushroom-app.com [Go live]

How do we build an App?

Data Collection



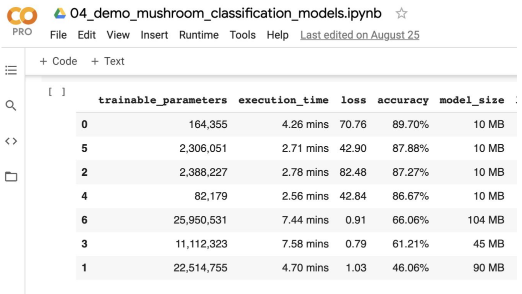
Python Script

Data Exploration

Model Exploration

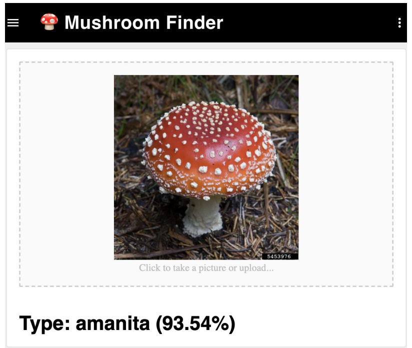
Model Training

Model Evaluation



	trainable_parameters	execution_time	loss	accuracy	model_size
0	164,355	4.26 mins	70.76	89.70%	10 MB
5	2,306,051	2.71 mins	42.90	87.88%	10 MB
2	2,388,227	2.78 mins	82.48	87.27%	10 MB
4	82,179	2.56 mins	42.84	86.67%	10 MB
6	25,950,531	7.44 mins	0.91	66.06%	104 MB
3	11,112,323	7.58 mins	0.79	61.21%	45 MB
1	22,514,755	4.70 mins	1.03	46.06%	90 MB

Colab



Type: amanita (93.54%)

Rest API

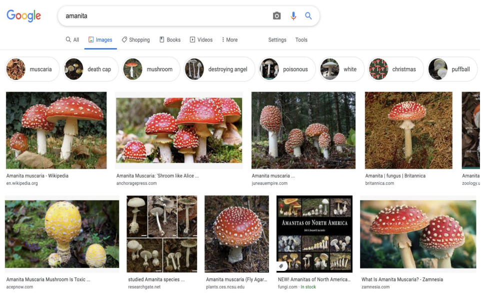
Best Model

Type: amanita (93.54%)

IDE / Code Editor

How do we build an App?

Data Collection



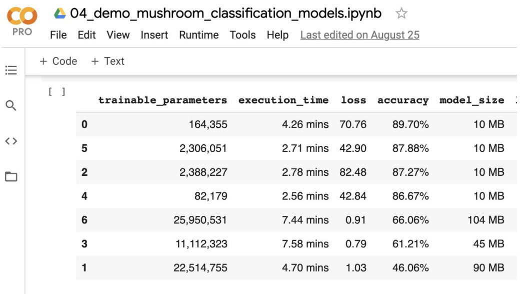
Python Script

Data Exploration

Model Exploration

Model Training

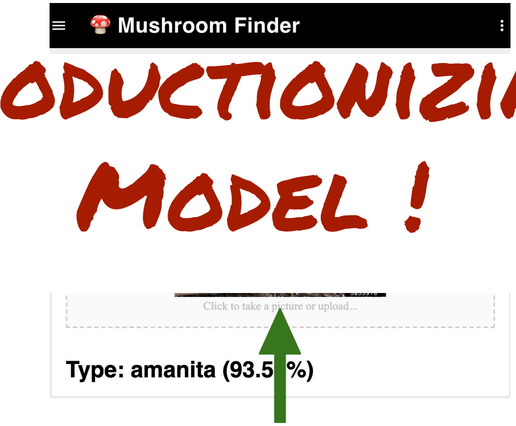
Model Evaluation



	trainable_parameters	execution_time	loss	accuracy	model_size
0	164,355	4.26 mins	70.76	89.70%	10 MB
5	2,306,051	2.71 mins	42.90	87.88%	10 MB
2	2,388,227	2.78 mins	82.48	87.27%	10 MB
4	82,179	2.56 mins	42.84	86.67%	10 MB
6	25,950,531	7.44 mins	0.91	66.06%	104 MB
3	11,112,323	7.58 mins	0.79	61.21%	45 MB
1	22,514,755	4.70 mins	1.03	46.06%	90 MB

Colab

PRODUCTIONIZING MODEL !



Rest API

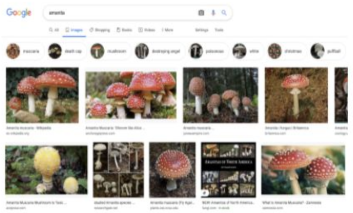
Best Model

Type: amanita (93.54%)

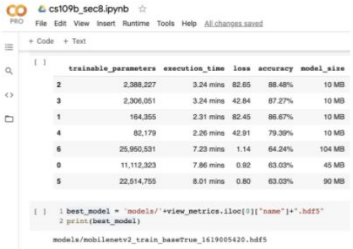
IDE / Code Editor

Challenges

Development



Python Script



	trainable_parameters	execution_time	loss	accuracy	model_size
2	2,388,227	3.24 mins	82.65	88.48%	10 MB
3	2,306,051	3.24 mins	42.84	87.27%	10 MB
1	164,355	2.31 mins	82.45	86.67%	10 MB
4	82,179	2.26 mins	42.91	79.39%	10 MB
6	25,950,531	7.23 mins	1.14	64.24%	104 MB
0	11,112,323	7.86 mins	0.92	63.03%	45 MB
5	22,514,755	8.01 mins	0.80	63.03%	90 MB

```
best_model = 'models/'+view_metrics.iloc[0]["name"]+".hdf5"
print(best_model)
```

```
models/mobilenetv2_train_baseTrue_1619005420.hdf5
```

Colab

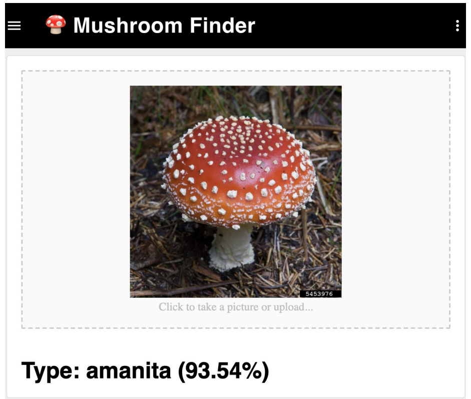
Rest API

Best Model

Type: amanita (93.54%)

IDE / Code Editor

Deployment



Type: amanita (93.54%)

Python: pipenv
Chromium: Mac install
OS: Mac

Python: Colab provided env
OS: Linux

Python: pipenv
OS: Mac

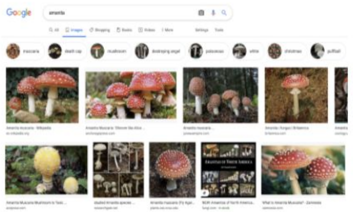
Python: pipenv
OS: Linux

Server

**One developer
Using a Macbook**

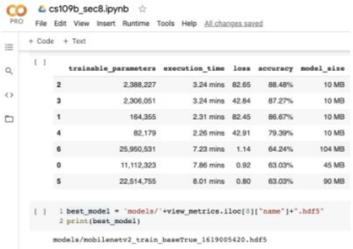
Challenges - Multiple Developers

Development



Python Script

Python: pipenv
Chromium: Mac install, Windows install
OS: Mac, Windows



cs109b_sec8.ipynb

	trainable_parameters	execution_time	loss	accuracy	model_size
2	2,388,227	3.24 mins	82.65	88.48%	10 MB
3	2,306,051	3.24 mins	42.84	87.27%	10 MB
1	164,355	2.31 mins	82.45	86.67%	10 MB
4	82,179	2.26 mins	42.91	79.39%	10 MB
6	25,950,531	7.23 mins	1.14	64.24%	104 MB
0	11,112,323	7.86 mins	0.92	63.03%	45 MB
5	22,514,755	8.01 mins	0.80	63.03%	90 MB

```
best_model = 'models/'+view_metrics.iloc[0]["name"]+".hdf5"
print(best_model)
```

```
models/mobilenetv2_train_baseTrue_1619005420.hdf5
```

Colab

Python: Colab provided env
OS: Linux

Rest API

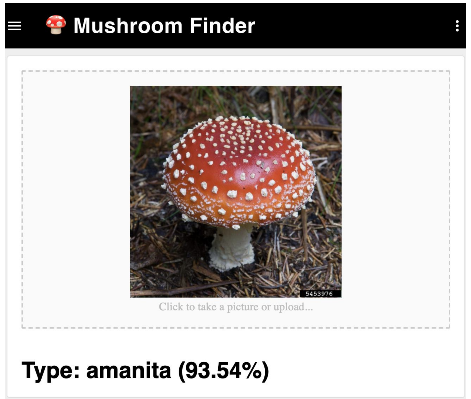
Best Model

Type: amanita (93.54%)

IDE / Code Editor

Python: pipenv
OS: Mac, Windows

Deployment



Type: amanita (93.54%)

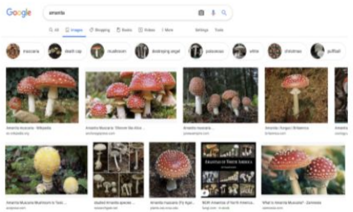
Python: pipenv
OS: Linux

Server

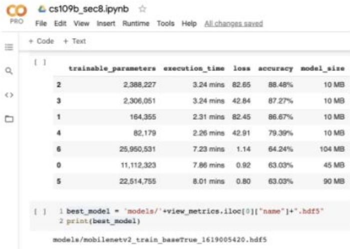
Multiple developers, Using Mac and Windows OS

Challenges - Multiple Developers + Automation

Development



Python Script



cs109b_sec8.ipynb

	trainable_parameters	execution_time	loss	accuracy	model_size
2	2,388,227	3.24 mins	82.65	88.48%	10 MB
3	2,306,051	3.24 mins	42.84	87.27%	10 MB
1	164,355	2.31 mins	82.45	86.67%	10 MB
4	82,179	2.26 mins	42.91	79.39%	10 MB
6	25,950,531	7.23 mins	1.14	64.24%	104 MB
0	11,112,323	7.86 mins	0.92	63.03%	45 MB
5	22,514,755	8.01 mins	0.80	63.03%	90 MB

```
best_model = 'models/'+view_metrics.iloc[0]["name"]+".hdf5"
print(best_model)
```

```
models/mobilenetv2_train_baseTrue_1619005420.hdf5
```

Colab

Rest API

Best Model

Type: amanita (93.54%)

IDE / Code Editor

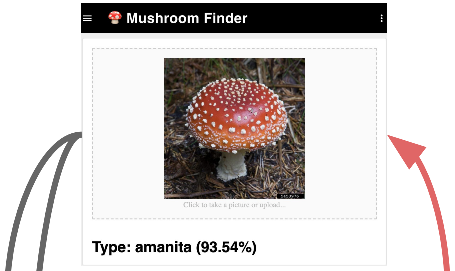
Python: pipenv
Chromium: Mac install, Windows install
OS: Mac, Windows

Python: Colab provided env
OS: Linux

Python: pipenv
OS: Mac, Windows

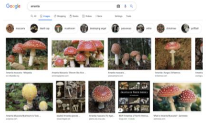
Multiple developers, Using Mac and Windows OS

Deployment



Type: amanita (93.54%)

Model Training



Python Script

Python: pipenv
OS: Linux

Server

Challenges / Solutions

Challenges:

- OS specific installations required
- How to collaborate code/frameworks?
- Automate data collection / model training
- New team member onboarding
- “It works on my machine” ¯_(ツ)_/¯

Solutions:

- Isolate development into environments that can be shared
- Develop in an common OS regardless of developers host OS
- Track software/framework installs

Tools

- Virtual Machines
- Virtual Environments
- Containers

Outline

1. Motivation
2. **Virtual Machines**
3. Virtual Environments

Virtual Machines Demo

https://github.com/dlops-io/simple-translate

Why should we use virtual machines?

Motivation

- All team members want an identical machine with same OS
- Should be easy create and delete instances
- All softwares installations need to be same across team members
- Team members need to run the same experiments in Isolation!

Virtual Machines!

Why should we use virtual machines?

Advantages

- Full autonomy: it works like a separate computer system; it is like running a computer within a computer.
- **Very secure**: the software inside the virtual machine cannot affect the actual computer.
- Lower costs: buy one machine and run multiple operating systems.
- Used by all Cloud providers for on demand server instances.

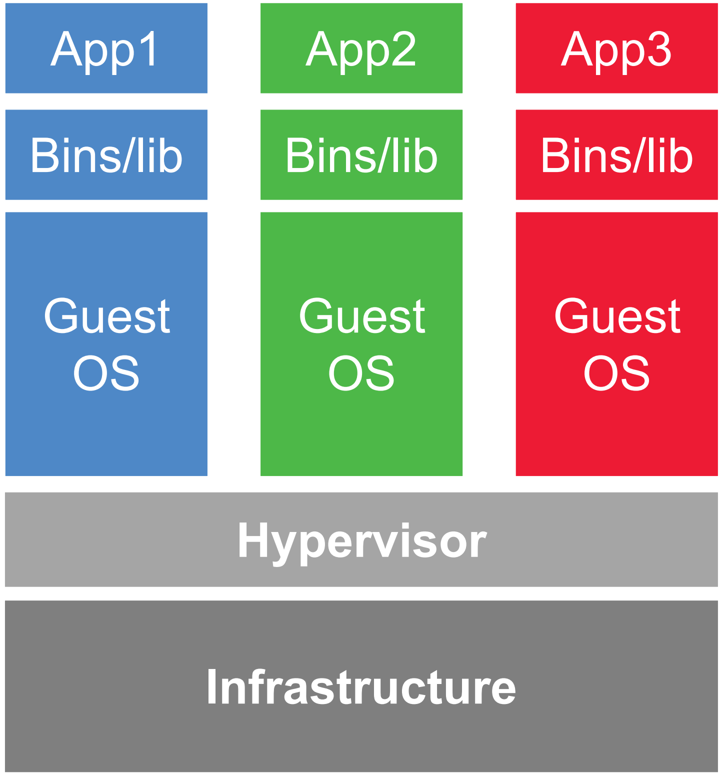
Softwares used for Virtualization

- VirtualBox
- VMWare
- Parallels

Why should we use virtual machines?

Advantages

- virtual machines have their own virtual hardware: CPUs, memory, hard drives, etc.
- you need a hypervisor that manages different virtual machines on server
- hypervisor can run as many virtual machines as we wish
- operating system is called the "host" while those running in a virtual machine are called "guest“
- You can install a completely different operating system on this virtual machine



Machine Virtualization

Why should we use virtual machines?

Limitations

- Uses hardware in local machine
- Not very portable since size of VMs are large
- There is an overhead associated with virtual machines
    - Guest is not as fast as the host system
    - Takes a long time to start up
    - It may not have the same graphics capabilities

Outline

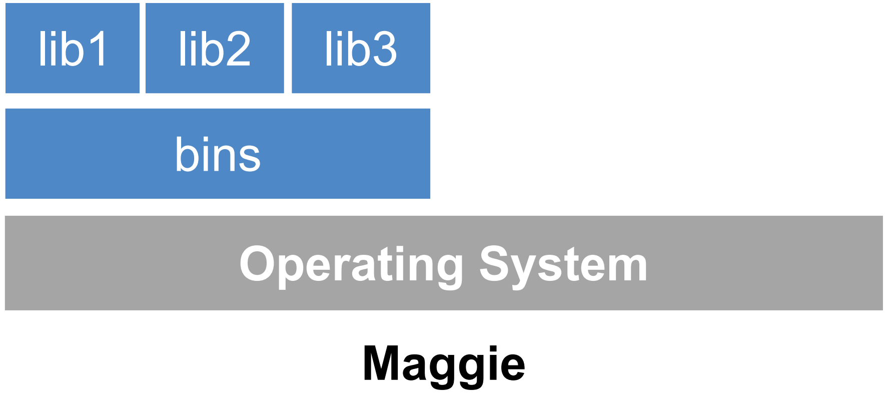
1. Motivation
2. Virtual Machines
3. **Virtual Environments**

Why should we use virtual environment?

- Virtual environments help to make development and use of code more streamlined.
- Virtual environments keep dependencies in separate “sandboxes” so you can switch between both applications easily and get them running.
- Given an operating system and hardware, we can get the exact code environment set up using different technologies. This is key to understand the trade off among the different technologies presented in this class.

Why should we use virtual environment?

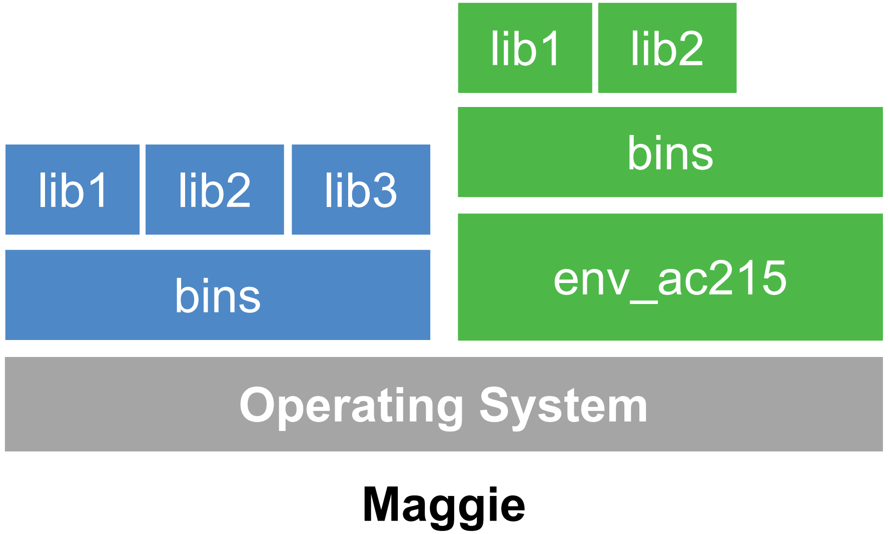
- Maggie took CS109, she used to run her Jupyter notebooks from anaconda prompt. Every time she installed a module it was placed in the either of `bin`, `lib`, `share`, `include` folders and she could import it in and used it without any issue.



```
$ which python
/c/Users/maggie/Anaconda3/python
```

Why should we use virtual environment?

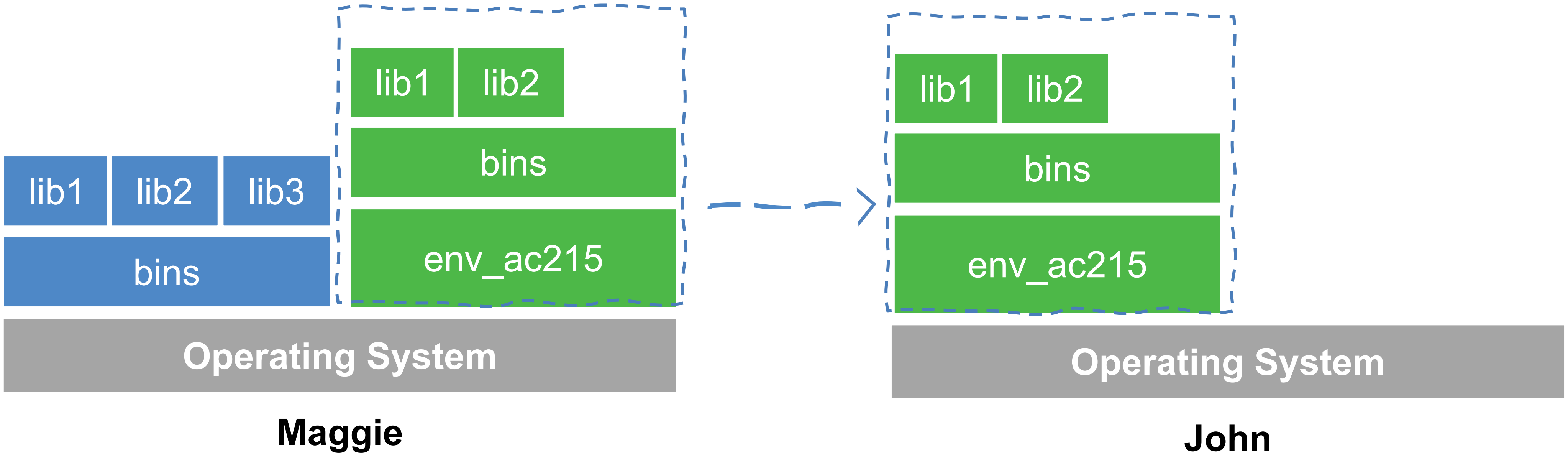
- Maggie starts taking AC215, and she thinks that it would be good to isolate the new environment from the previous environments avoiding any conflict with the installed packages. She adds a layer of abstraction called virtual environment that helps her keep the modules organized and avoid misbehaviors while developing a new project.



```
$ which python
/c/Users/maggie/Anaconda3/envs/env_ac215/python
```

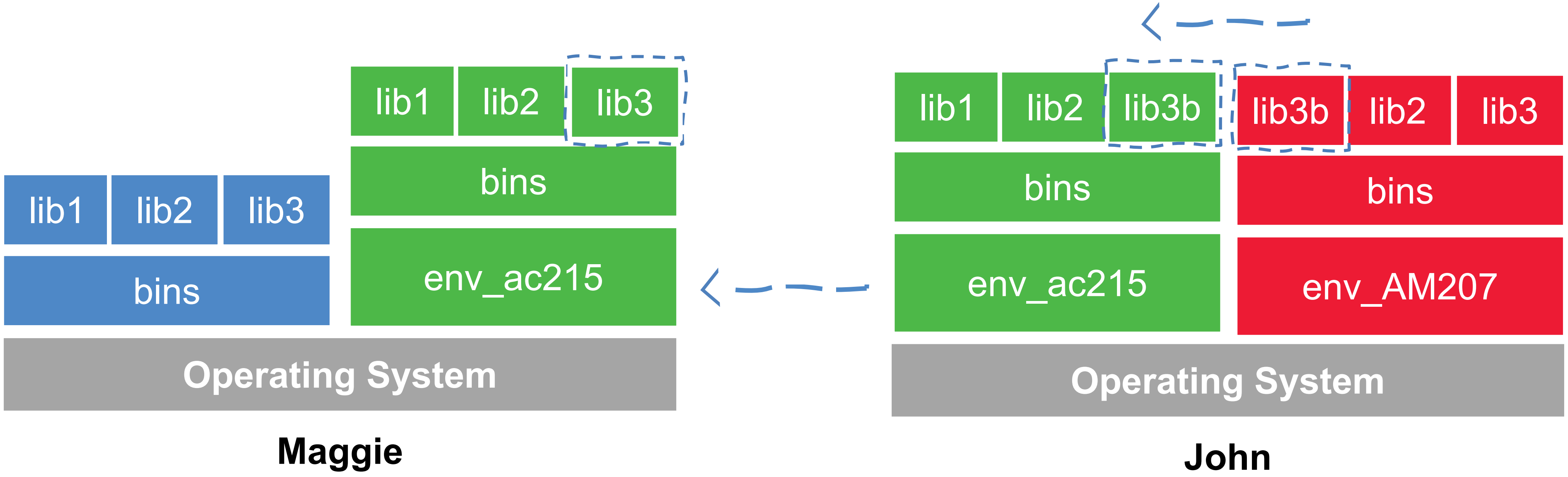
Why should we use virtual environment?

- Maggie collaborates with John for the final project and shares the environment she is working on through .yml file (for conda env).



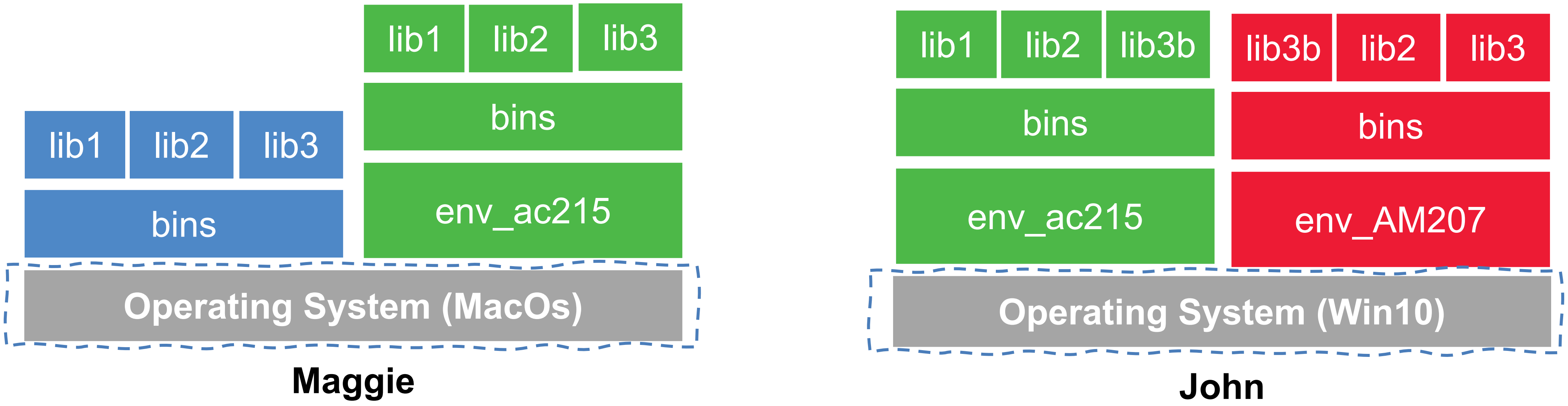
Why should we use virtual environment?

- John experiments with a new method he learned in another class and adds a new library to the working environment. After seeing tremendous improvements, he sends Maggie back his code and a new .yml file (for conda env). She can now update her environment and replicate the experiment.



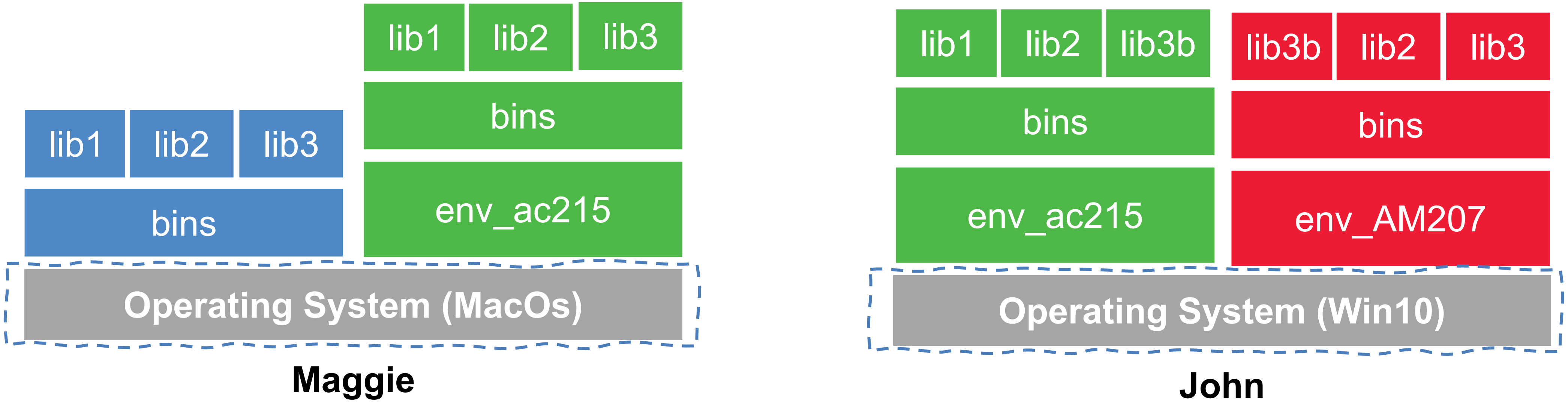
Why should we use virtual environment?

- What could go wrong?
- Unfortunately, Maggie and John reproduce different results, and they think the issue relates to their operating systems. Indeed while Maggie has a MacOS, John uses a Win10.



Why should we use virtual environment?

- What could go wrong?



Virtual environments

Pros

- Reproducible research
- Explicit dependencies
- Improved engineering collaboration

Cons

- Difficulty setting up your environment
- Not isolation
- Does not always work across different OS

What are virtual environments then?

- A virtual environment is an isolated python environment where the interpreter can run libraries and scripts independently from other virtual environments
- Think of a virtual environment as a directory with the following components:
  - *site_packages/* directory where third-party libraries are installed
  - *links* [really symlinks] to the executables on your system
  - some *scripts* that ensure that the code uses the interpreter and site packages in the virtual environment

Creating Virtual Environments

- **VirtualEnv**

  The default way to create virtual environments in python

- **Conda**

  Is a package manager and environment manager for Data Scientists

- **PipEnv**

  Production-ready tool that aims to bring the best of all packaging worlds to the Python world

VirtualEnv

- Virtual environments manager embedded in Python
- Incorporated into broader tools such as pipenv
- Allow to install modules using pip package manager

VirtualEnv

How to use it:

- create an environment within your project folder `virtualenv your_env_name`
- it will add a folder called environment_name in your project directory
- activate environment: `source env/bin/activate`
- install requirements using: `pip install package_name=version`
- deactivate environment once done: `deactivate`

Conda

- Virtual environments manager embedded in Anaconda
- Allow to use both conda and pip to manage and install packages
- Virtual environments comes pre-installed with various engineering and data science packages

Conda

How to use it:

- create an environment

  ```
  conda create --name your_env_name python=3.7
  ```

- it will add a folder located within your anaconda installation

  ```
  /Users/your_username /anaconda3/envs/your_env_name
  ```

- activate environment `conda activate your_env_name` (should appear in your shell)
- install requirements using `conda install package_name=version`
- deactivate environment once done `conda deactivate`
- duplicate your environment using YAML file `conda env export > my_environment.yml`
- to recreate the environment now use `conda env create -f environment.yml`

Conda

How to use it:

- find which environment you are using

```
conda env list
```

- create an environment

```
conda create --name your_env_name python=3.7
```

- it will add a folder located within your anaconda installation

```
/Users/your_username/[opt]/anaconda3/envs/your_env_name
```

- activate environment

  `conda activate your_env_name` (should appear in your shell)

- install requirements using

```
conda install package_name=version
```

- deactivate environment once done

```
conda deactivate
```

- duplicate your environment using YAML file `conda env export > my_environment.yml`
- to recreate the environment now use `conda env create -f environment.yml`

PipEnv

- Built on top of *VirtualEnv*
- Fixes many shortcomings of *VirtualEnv*
- Distinguish development vs. production environments
- Automatically keeps track of packages and package dependencies using a `Pipfile` & `Pipfile.lock`

PipEnv

How to use it:

- Need to `pip install pipenv`
- To create a new environment run `pipenv install`
- Activate the environment by `pipenv shell`
- To install a new package `pipenv install numpy` or `pip install numpy` (this will not lock the package automatically)
- To sync from an existing Pipfile: `pipenv sync`

More on Virtual environments

Further readings

- Pipenv: Python Dev Workflow for Humans

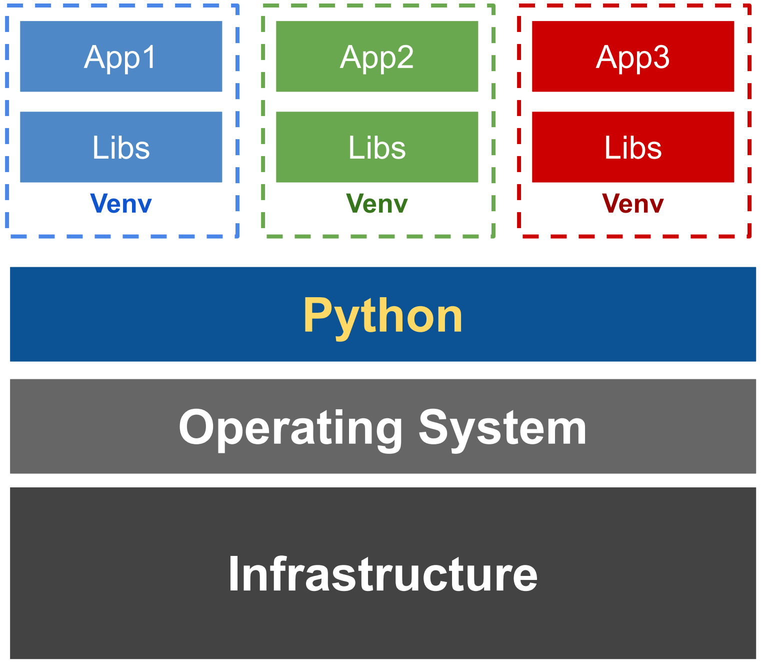
  https://pipenv.pypa.io/en/latest/
- For detailed discussions on similarities and differences among virtualenv and conda

  https://jakevdp.github.io/blog/2016/08/25/conda-myths-and-misconceptions/
- More on venv and conda environments

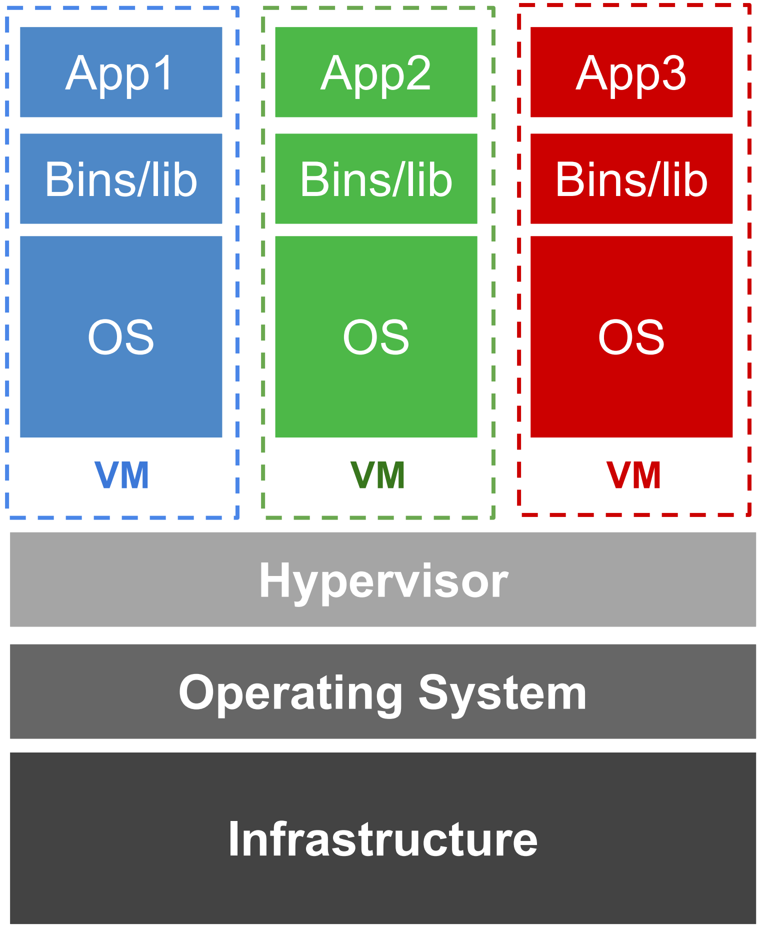
  https://towardsdatascience.com/virtual-environments-104c62d48c54

  https://towardsdatascience.com/getting-started-with-python-environments-using-conda-32e9f2779307

Virtual Environments vs Virtual Machine



Virtual Environments



Virtualization

THANK YOU