

End-to-end Python Practice on Industrial Data Engineering

葉信和 / Hsin-Ho Yeh

Software Engineer / Funder / CEO @ 信誠金融科技

hsinho.yeh@footprint-ai.com



2022/06/01

2022/06/08

Download Slides

<https://reurl.cc/p1pg8Q>



About me

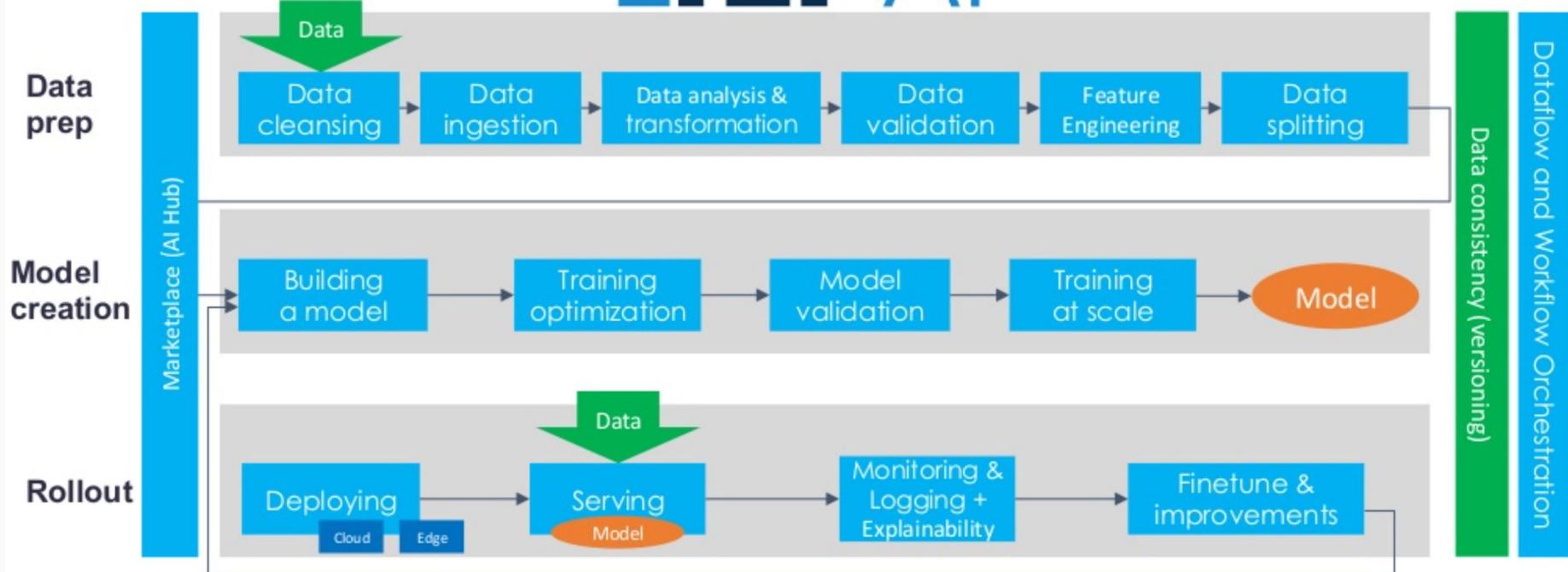
- 2020 - Present at 信誠金融科技
 - Shrimping: A data-sharing platform
 - <https://get-shrimping.footprint-ai.com>
 - Tintin: a machine learning platform for everyone
 - <https://get-tintin.footprint-ai.com>
- 2016 - 2020 at Igloolnsure (16M+ in series A+ 2020)
 - Provide digital insurance for e-economic world
 - Funded in KUL, Headquartered in Singapore
 - First employee/ Engineering Lead / Regional Head/ Chief Engineer
- 2013 - 2016 at Studio Engineering @ hTC
 - Principal Engineer on Cloud Infrastructure Team
- 2009 - 2012 at IIS @ Academia Sinica
 - Computer vision, pattern recognition, and data mining
- CS@CCU, CS@NCKU alumni



Agenda

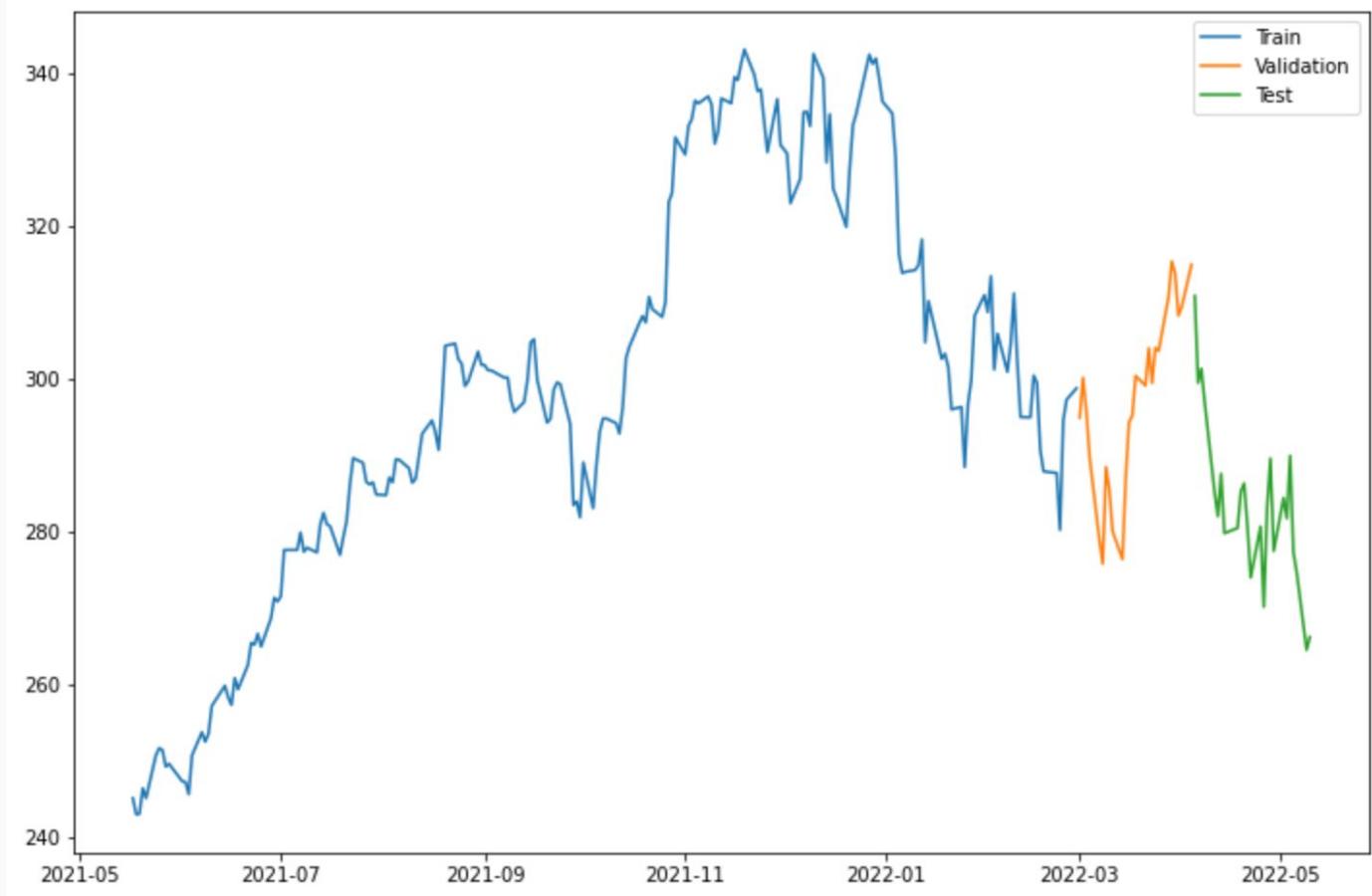
- End-to-end flow on industrial data analysis practice
- How and where do we collect data?
- Web scraping with selenium
- Data Analysis with Pandas
- Build a financial model with Kubeflow
- Q&A

Real-world Machine Learning Application - End-to-End ML LifeCycle



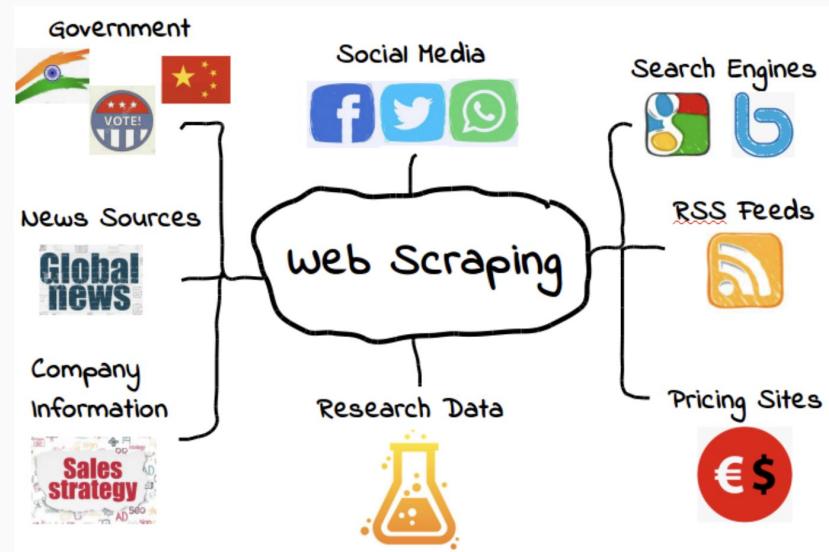
Source: <https://www.slideshare.net/AnimeshSingh/advanced-model-inferencing-leveraging-kubeflow-serving-knative-and-istio-196096385>

How to predict stock price for the next day?



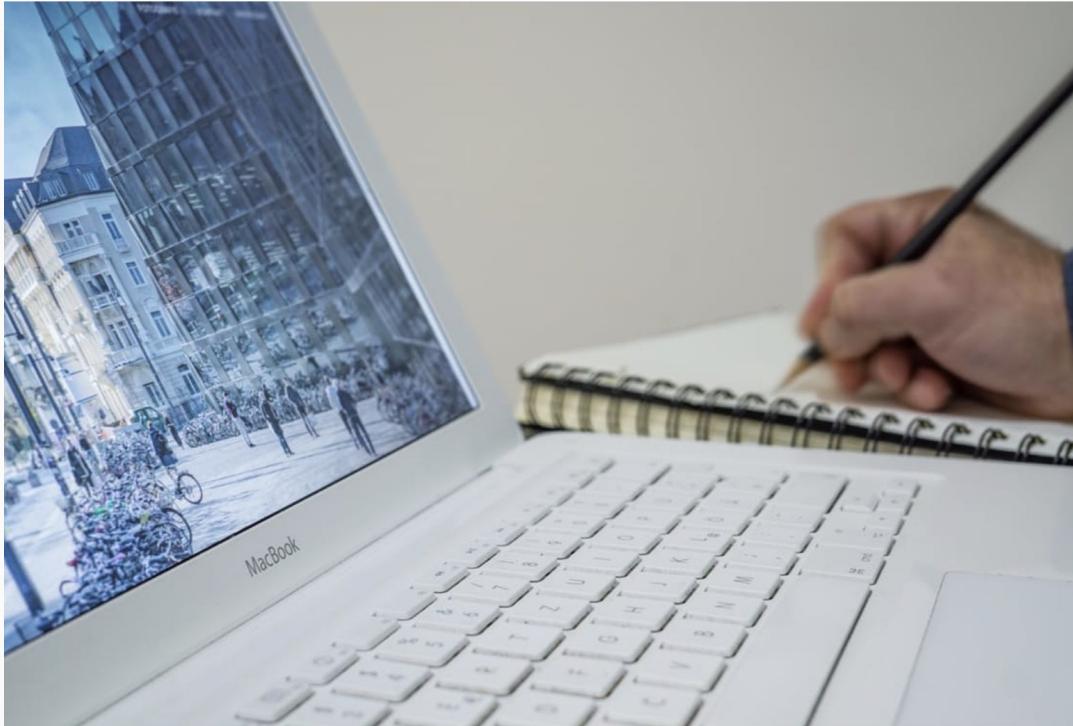
What is Web Scraping?

- Web Scraping is an **automation** process of collecting **structure** data from public websites.
- Common Use Cases including
 - Competitor price monitoring on E-commerce
 - News monitoring from social network



Different ways of collecting data from websites (1/3)

- Manually (slow & slow & slow)



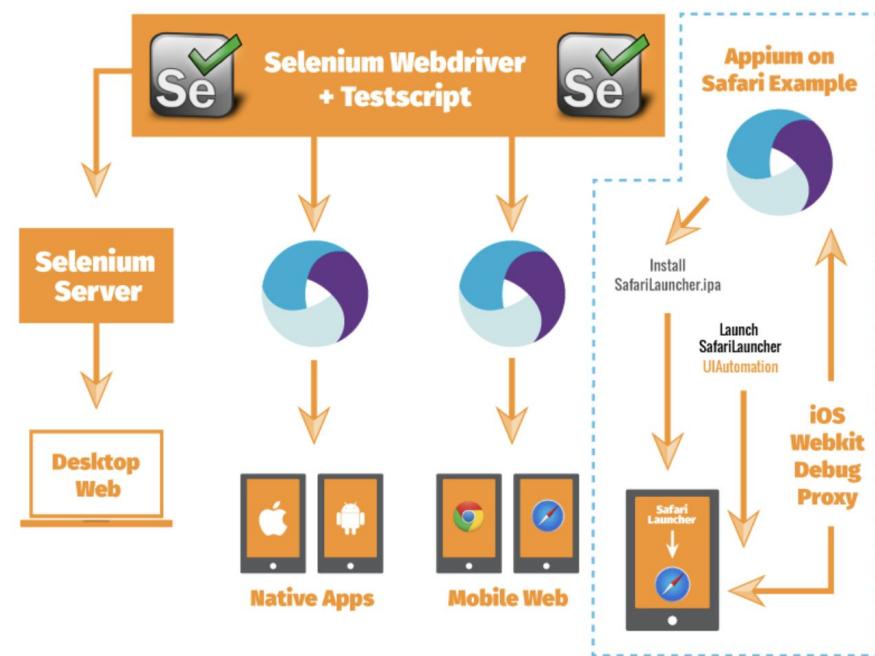
Different ways of collecting data from websites (2/3)

- API Requests with Python(Stable & Fast but not always works...)

```
>>> import requests  
  
>>> r = requests.get('https://www.google.com.tw/')  
  
>>> print(r.status_code)  
200  
  
>>>print(r.text)  
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage"  
lang="zh-TW"><head><meta content="text/html; charset=UTF-8"  
http-equiv="Content-Type"><meta  
content="/images/branding/googleg/1x/googleg_standa....
```

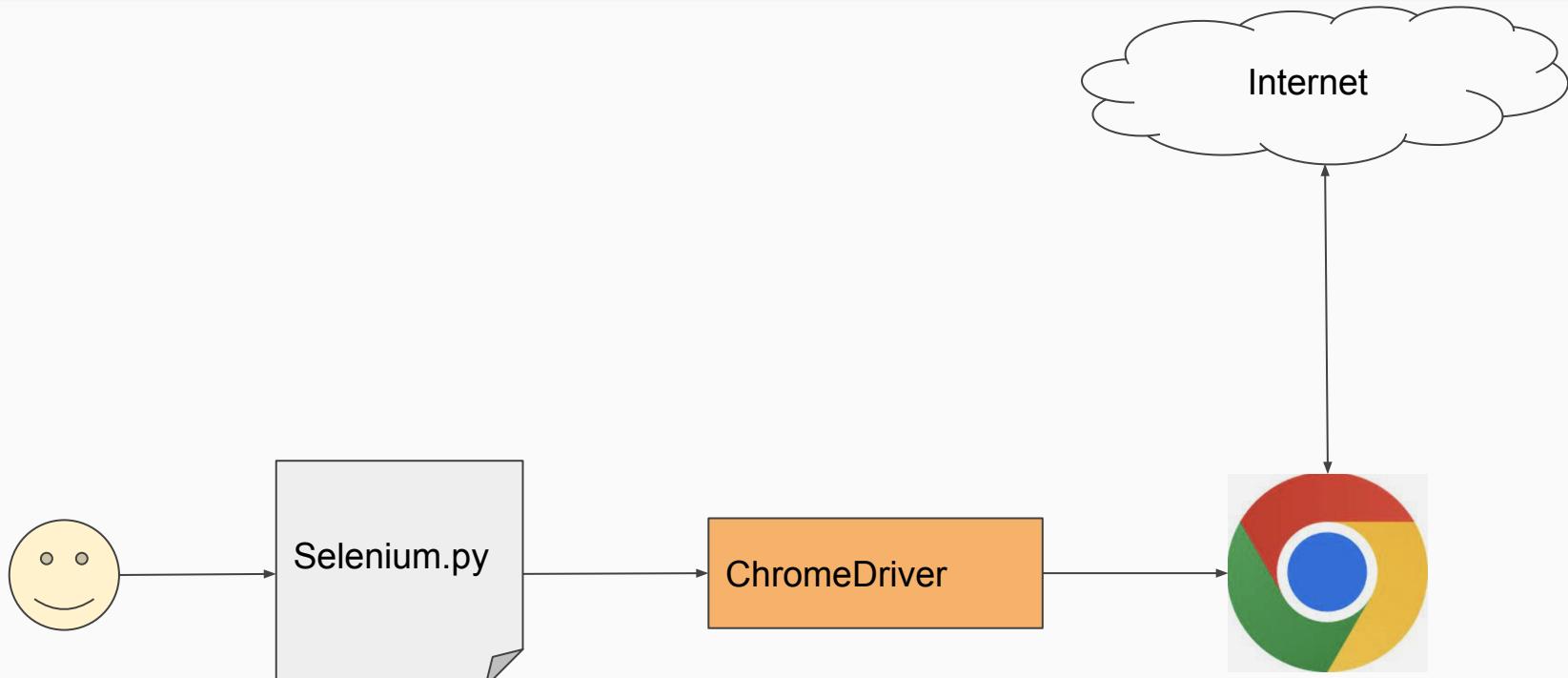
Different ways of collecting data from websites (3/3)

- UI Automation with Selenium
 - Selenium is an web automation tool which allows you to automate UI testing for different browsers via its webdriver.
 - Webdriver is used to control each Browser behaviors making it more like real-human interaction.
- Because of its nature of automation and mimic human behavior, making it a good human-like web-scraping



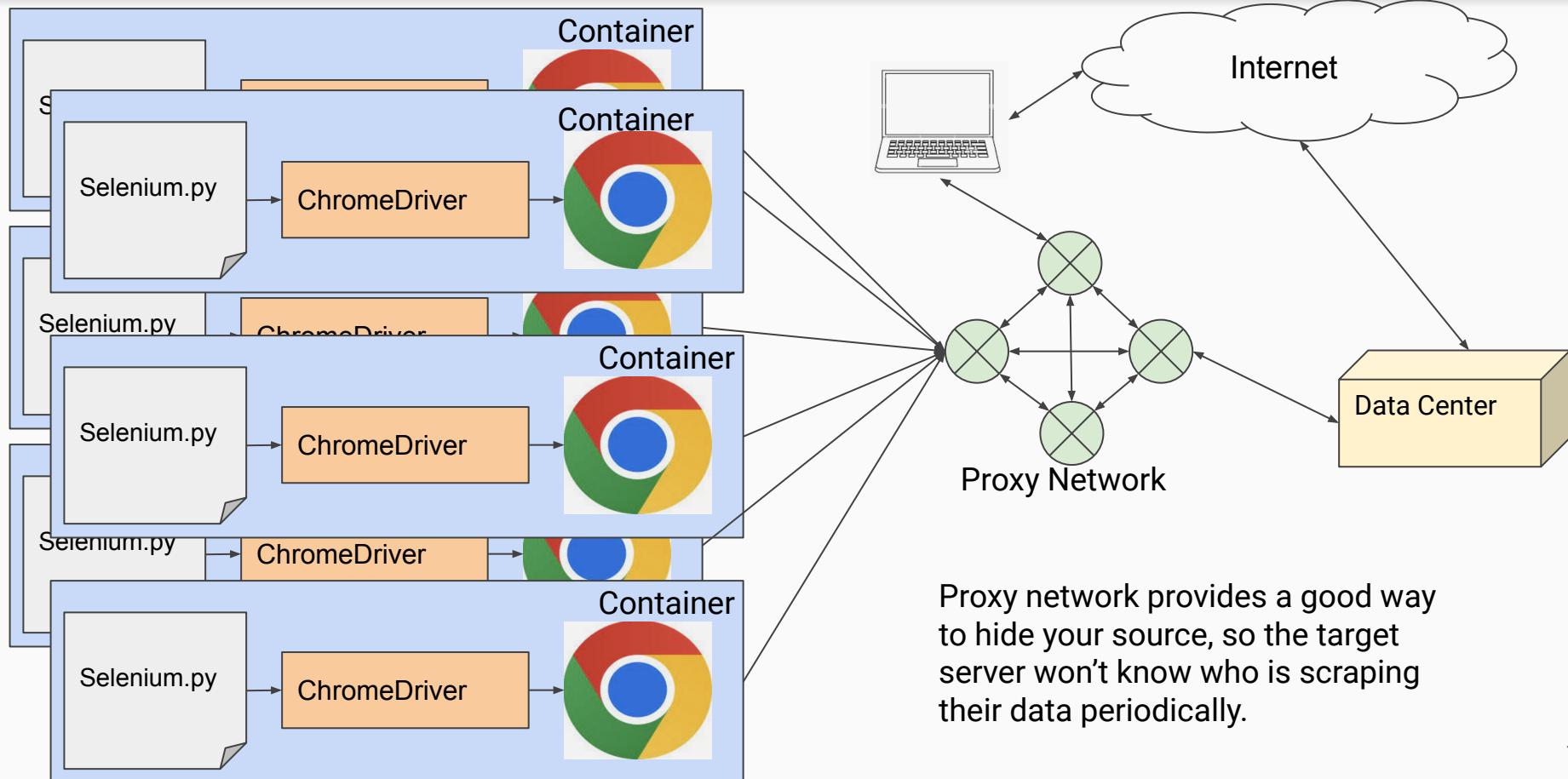
Ref: <https://smartbear.com/blog/selenium-cross-browser-testing-on-mobile-devices/>

How Selenium WebDriver Works?



ChromeDriver relies on the pre-installed chrome package to send the requests and render the HTML tags

How Selenium WebDriver Works In Scale?



Selenium Examples

<https://github.com/FootprintAI/selenium-example>

The screenshot shows a GitHub repository page for 'FootprintAI / selenium-example'. The repository is public and has 1 branch and 0 tags. The main file list includes 'hsinhyoyeh remove', 'drivers', 'examples', 'Dockerfile', 'LICENSE', 'README.md', and 'chromedriver'. On the right side, there is a 'Code' dropdown menu with options for 'Clone' via HTTPS, SSH, or GitHub CLI, and links for 'Open with GitHub Desktop' and 'Download ZIP'.

File	Description
hsinhyoyeh remove	
drivers	add headless as options
examples	remove
Dockerfile	add Dockerfile
LICENSE	Initial commit
README.md	Initial commit
chromedriver	add ig/goodinfo example

Selenium: Basic Example

```
1 from drivers.webdriver import WebDriver
2
3 from selenium.webdriver.common.by import By
4 from selenium.webdriver.common.keys import Keys
5
6 def do(driver: WebDriver, q: str):
7     driver.get("http://www.python.org")
8     assert "Python" in driver.title
9     elem = driver.find_element(By.NAME, "q")
10    elem.clear()
11    elem.send_keys(q)
12    elem.send_keys(Keys.RETURN)
13    assert "No results found." not in driver.page_source
```

By is for element selection
Key is for mapping keyboard

Selenium: Webdriver Inheritance & encapsulation

```
1 import os
2 import platform
3 from selenium.webdriver import Chrome, ChromeOptions
4
5 class WebDriver(Chrome):
6     def __init__(self, headless:bool=False):
7         exec_path = None
8         if platform.system() == "Darwin":
9             exec_path = self.__wrap_abspath('chromedriver.darwin')
10            elif platform.system() == "Windows":
11                exec_path = self.__wrap_abspath('chromedriver.exe')
12                elif platform.system() == "Linux":
13                    exec_path = self.__wrap_abspath('chromedriver.linux')
14
15        chrome_options = ChromeOptions()
16        # comment this to enable guest browser to show up (for debugging)
17        if headless:
18            chrome_options.add_argument('--headless')
19            chrome_options.add_argument('--no-sandbox')
20
21        Chrome.__init__(self,chrome_options=chrome_options, executable_path=exec_path)
22
23    def __wrap_abspath(self, binary)-> str:
24        return os.path.join(os.path.dirname(os.path.abspath(__file__)), binary)
```

inheritance

encapsulation

Selenium: Cookies & Navigation

```
1  from drivers.webdriver import WebDriver
2
3  from time import sleep
4
5  def navigation(driver: WebDriver):
6      driver.get("https://www.google.com")
7      sleep(1)
8      driver.get("https://www.python.org")
9      sleep(1)
10     driver.back()
11     sleep(1)
12     driver.forward()
13
14 def get_cookies(driver: WebDriver):
15     driver.get("https://www.google.com")
16     print(driver.get_cookies())
```

Selenium: Exceptions

```
1 from drivers.webdriver import WebDriver
2
3 def exception_handling(driver: WebDriver):
4     from selenium.common.exceptions import TimeoutException, NoSuchElementException
5     try:
6         driver.get("https://www.google.com")
7     except TimeoutException:
8         print("time out")
9     try:
10         driver.find_element_by_id("you-should-not-see-me")
11     except NoSuchElementException:
12         print("no element found")
```

Selenium: query methods

Tag

```
<input type="text" name="passwd" id="passwd-id" />
```

Attribute

```
// we have four different ways to locate an individual UI element
```

```
element = driver.find_element(By.ID, "passwd-id")
element = driver.find_element(By.NAME, "passwd")
element = driver.find_element(By.XPATH, "//input[@id='passwd-id']")
element = driver.find_element(By.CSS_SELECTOR, "input#passwd-id")
```

DOM(Document Object Model) Tree

An example of the DOM

Let's start with the following simple document:

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <title>About elk</title>
5 </head>
6 <body>
7   The truth about elk.
8 </body>
9 </html>
```

The DOM represents HTML as a tree structure of tags. Here's how it looks:



Ref: <https://javascript.info/dom-nodes>

Example: Instagram Login

```
0
7     driver.get("https://www.instagram.com")
8     sleep(5) # wait for the page loading is finished
9     # find username fields
10    username_field=driver.find_element_by_css_selector("input[name='username']")
11    # find password fields
12    password_field=driver.find_element_by_css_selector("input[name='password']")
13    username_field.clear()
14    password_field.clear()
15    # send email in username field
16    username_field.send_keys(email)
17    # send password in password field
18    password_field.send_keys(password)
19    sleep(5) # sleep 5 seconds
20
21    # click login button
22    driver.find_element_by_css_selector("button[type='submit']").click()
23    __post_login(driver)
24
25 def __post_login(driver: WebDriver):
26
```

Example: Find Stock Price

```
1  from drivers.webdriver import WebDriver
2
3  from selenium.webdriver.common.by import By
4  from selenium.webdriver.support.ui import WebDriverWait
5  from selenium.webdriver.support import expected_conditions as EC
6
7  def do_find(driver: WebDriver, symbol: str):
8      driver.get("https://goodinfo.tw/StockInfo/index.asp")
9      ele = driver.find_element_by_xpath(u"//*[@id='txtStockCode']")
10     ele.clear()
11     ele.send_keys(symbol)
12     driver.find_element_by_xpath(u"//input[@value='股票查詢']").click()
13
14     # the following code will wait until the element is visible
15     try:
16         wait = WebDriverWait(driver, 30)
17         wait.until(EC.visibility_of_element_located((By.XPATH, u"//*[@id='imgKC']")))
18     except:
19         raise Exception("target element is not show up")
```

You have a scraping, we have lots of anti-Scraping tools.

- How to bypass Anti-Scraping Tool
 - Keep Rotate Your IP Address
 - By a well-paid proxy service provider
 - By restricting a certain area of your location
 - Use Real User Agent and valid Referrer
 - UA(User-Agent) are http header which is used identify what browser type you used to visit this website.
 - Keep a list of valid UI and use them randomly.
 - Avoid Periodically Requests
 - Keep random intervals between requests, random delay are helpful.
 - Update your code frequently
 - Anti-scraping tool would change DOM structure frequently, making your script failed to find target elements.
- However, Implementing these guidelines could costly if your application is still below a certain scale.

Shrimping: A data-sharing platform

Shrimping provides a unified way for clients to get human-centric information in a simple, easy, and low cost fashion.

<https://get-shrimping.footprint-ai.com/>

Web scraping scenario for Shrimping.

- UI Validation
 - When working with business partners, it is extremely important that your partner has interpreted your product correctly.
- Collecting sell history on an ecommerce platform
 - Track the sell volume of all products on an eCommerce platform.
 - As the number of products could be big (approximately 100M active products), how to get each product's sell records on daily basis is extremely challenging.
- Scraping and analyzing KOL's feeds on social network platforms
 - Find out a KOL and his/her fans preference for retargeting, reselling, or other marketing strategies.
 - Social network platform always implemented anti-bot mechanism, making it hard to collect in a large scale fashion.

Data Analysis with Panda and Numpy

<https://pandas.pydata.org/docs/>

<https://numpy.org/doc/stable/user/index.html#user>

```
import numpy as np
```

```
import pandas as pd
```

Pandas Series: One-dimensional array of holding **any** data type

```
>>> import numpy as np
>>> a = pd.Series(np.random.randn(5))
>>> a
0    0.427981
1    0.998562
2    0.649786
3    1.341638
4    0.743008
dtype: float64
>>> a[a > a.median()]
1    0.998562
3    1.341638
dtype: float64
>>> a.size
5
>>> a.array
<PandasArray>
[0.4279808645556574, 0.9985615198039794, 0.6497859868724282,
 1.3416378899984038, 0.7430084840615195]
Length: 5, dtype: float64
>>> a.dtype
dtype('float64')
>>> a.to_numpy()
array([0.42798086, 0.99856152, 0.64978599, 1.34163789, 0.74300848])
>>> a2 = a * 2
>>> a2
0    0.855962
1    1.997123
2    1.299572
3    2.683276
4    1.486017
dtype: float64
>>> a3 = a[:-1] + a2[1:]
>>> a3
0        NaN
1    2.995685
2    1.949358
3    4.024914
4        NaN
dtype: float64
```

DataFrame: Think like table, work like table...

- Pandas is a Python library and the de-facto standard for working with **structured** tabular data on Python
- Rich Format Supported including CSV, JSON, Parquet, MS EXCEL, ...

asset	max_training	max_validation	mean_training	mean_validation	med_training	med_validation	min_training	min_validation	std_training	std_validation
AAN	2.685913	5.132591	0.046193	0.029751	0.120043	0.237710	-4.324665	-7.955100	1.356045	2.851172
ACC	1.546164	3.473945	0.024481	-0.191317	0.067734	0.000000	-1.493781	-8.192803	0.662752	2.181609
ACGL	1.225804	3.676009	0.035951	-0.350968	0.072673	0.000000	-1.788876	-6.347304	0.607838	1.949640
ADC	1.343051	1.567481	0.118847	-0.090484	0.152312	0.001203	-1.302874	-4.310633	0.603213	1.131439
AEL	3.002969	5.464704	0.026192	-0.094789	0.076017	0.000000	-3.330113	-7.611560	1.290956	2.659290
AFG	1.095037	3.558017	0.005339	-0.387737	0.072613	-0.060772	-2.050200	-7.330084	0.614609	2.013369
AFL	1.159893	3.050265	0.051159	-0.295960	0.072861	-0.038611	-1.414419	-5.840748	0.514807	1.709449
AGM	2.189914	2.572398	0.119186	-0.179405	0.130402	-0.031308	-2.258110	-5.378916	0.944030	1.627170
AGNC	0.934225	2.371371	-0.016914	-0.196734	0.026154	0.159754	-1.373769	-6.580712	0.481505	1.653974

Example of a simple Pandas table with “assets” as index and various numerical columns

Dataframe Indexing

- `iloc` (position-based indexing) vs `loc` (label-based indexing)
 - Label-based indexing returns with “match” labels
 - Position-based indexing returns elements with its numeric position index

```
>>> f = pd.DataFrame({'a':[1,2,3,4,5], 'b':[10,20,30,40,50], 'c': [9,8,7,6,5]})  
>>> f.head()  
   a   b   c  
0  1  10   9  
1  2  20   8  
2  3  30   7  
3  4  40   6  
4  5  50   5  
>>> f.loc[2:3, 'b']  
2    30  
3    40  
Name: b, dtype: int64  
>>> f.iloc[2:3, 1]  
2    30  
Name: b, dtype: int64
```

Dataframe Join Operation

- Join provides a way to intersect / union two Dataframes
 - Use 'On' to specify a specific join key
 - Use 'How' to specify we want to have inner / outer join

```
>>> f = pd.DataFrame({'a':[1,2,3,4,5], 'b':[10,20,30,40,50], 'c': [9,8,7,6,5]})  
>>> g = pd.DataFrame({'e':['I','II','III']})  
>>> f.join(g, how='outer')  
      a    b    c     e  
0   1   10   9     I  
1   2   20   8     II  
2   3   30   7    III  
3   4   40   6    NaN  
4   5   50   5    NaN  
>>> f.join(g, how='inner')  
      a    b    c     e  
0   1   10   9     I  
1   2   20   8     II  
2   3   30   7    III  
>>> f.join(g, how='inner', on="a")  
      a    b    c     e  
0   1   10   9     II  
1   2   20   8    III
```

Dataframe Multiindex

- Indexing allows you projecting Dataframe, and Multiindex simplify this process

```
>>> adult[adult.columns[:8]].head()
   education  education_num  sex  age  workclass  fnlwgt  marital_status  occupation  relationship  race  capital_gain  capital_loss
0      10th            6  Female  17       ?    304873  Never-married        ?        Own-child  White     34095          0
1      10th            6  Female  17  Private  169658  Never-married  Other-service  Own-child  White          0          0
2      10th            6  Female  17       ?   158762  Never-married        ?        Own-child  White          0          0
3      10th            6  Female  17  Private  329783  Never-married       Sales  Other-relative  White          0          0
4      10th            6  Female  17  Private  139183  Never-married       Sales        Own-child  White          0          0
>>> adult.loc[[' Bachelors'], ['label']]
   education  education_num  sex  age  label
0      Bachelors            13  Female  19  <=50K
1      Bachelors            13  Female  19  <=50K
2      Bachelors            13  Female  20  <=50K
3      Bachelors            13  Female  21  <=50K
4      Bachelors            13  Female  21  <=50K
...                   ...
90      Bachelors            13      Male  90  <=50K
91      Bachelors            13      Male  90   >50K
92      Bachelors            13      Male  90  <=50K
93      Bachelors            13      Male  90  <=50K
94      Bachelors            13      Male  90  <=50K
[5355 rows x 1 columns]
>>> adult.loc[[' Bachelors', 13, ' Female', 19], ['label']]
   education  education_num  sex  age  label
0      Bachelors            13  Female  19  <=50K
1      Bachelors            13  Female  19  <=50K
```

Dataframe Query

- Query is the most used filter to filter rows by its column's value.

```
>>> adult
   education      education_num  sex  age  workclass  fnlwgt  marital_status  occupation  ...  capital_loss  hours_per_week  native_country  label
0    10th            6        Female  17       ?    304873  Never-married          ?  ...  0  32  United-States  <=50K
1    10th            6        Female  17     Private  169658  Never-married  Other-service  ...  0  21  United-States  <=50K
2    10th            6        Female  17       ?    158762  Never-married          ?  ...  0  20  United-States  <=50K
3    10th            6        Female  17     Private  329783  Never-married        Sales  ...  0  10  United-States  <=50K
4    10th            6        Female  17     Private  139183  Never-married        Sales  ...  0  15  United-States  <=50K
...
5  Some-college  10        Male  84  Self-emp-inc  172907  Married-civ-spouse        Sales  ...  0  35  United-States  >50K
6  Some-college  10        Male  90     Private  52386  Never-married  Other-service  ...  0  35  United-States  <=50K
7  Some-college  10        Male  90     Private  52386  Never-married  Other-service  ...  0  35  United-States  <=50K
8  Some-college  10        Male  90  Self-emp-not-inc  282095  Married-civ-spouse  Farming-fishing  ...  0  40  United-States  <=50K
9  Some-college  10        Male  90       ?  225063  Never-married          ?  ...  0  10  South  <=50K
[32561 rows x 11 columns]
```

```
>>> adult.query("workclass == ' Private' ")
   education      education_num  sex  age  workclass  fnlwgt  marital_status  occupation  ...  capital_loss  hours_per_week  native_country  label
0    10th            6        Female  17     Private  169658  Never-married  Other-service  ...  0  21  United-States  <=50K
1    10th            6        Female  17     Private  329783  Never-married        Sales  ...  0  10  United-States  <=50K
2    10th            6        Female  17     Private  139183  Never-married        Sales  ...  0  15  United-States  <=50K
3    10th            6        Female  17     Private  130125  Never-married  Other-service  ...  0  20  United-States  <=50K
4    10th            6        Female  17     Private  27032  Never-married        Sales  ...  0  12  United-States  <=50K
...
5  Some-college  10        Male  79  Private  121318  Married-civ-spouse  Adm-clerical  ...  0  20  United-States  <=50K
6  Some-college  10        Male  80  Private  87518  Never-married  Prof-specialty  ...  1816  60  United-States  <=50K
7  Some-college  10        Male  81  Private  122651  Married-civ-spouse        Sales  ...  0  15  United-States  <=50K
8  Some-college  10        Male  90  Private  52386  Never-married  Other-service  ...  0  35  United-States  <=50K
9  Some-college  10        Male  90  Private  52386  Never-married  Other-service  ...  0  35  United-States  <=50K
[22696 rows x 11 columns]
```

Dataframe Visualization

```
plt.figure(figsize=(15, 6))
plt.subplots_adjust(top=1.25, bottom=1.2)

company_list = [MSFT, AAPL, TSLA, VOO]

for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Adj Close'].plot()
    plt.ylabel('Adj Close')
    plt.xlabel(None)
    plt.title(f"Closing Price of {tech_list[i - 1]}")

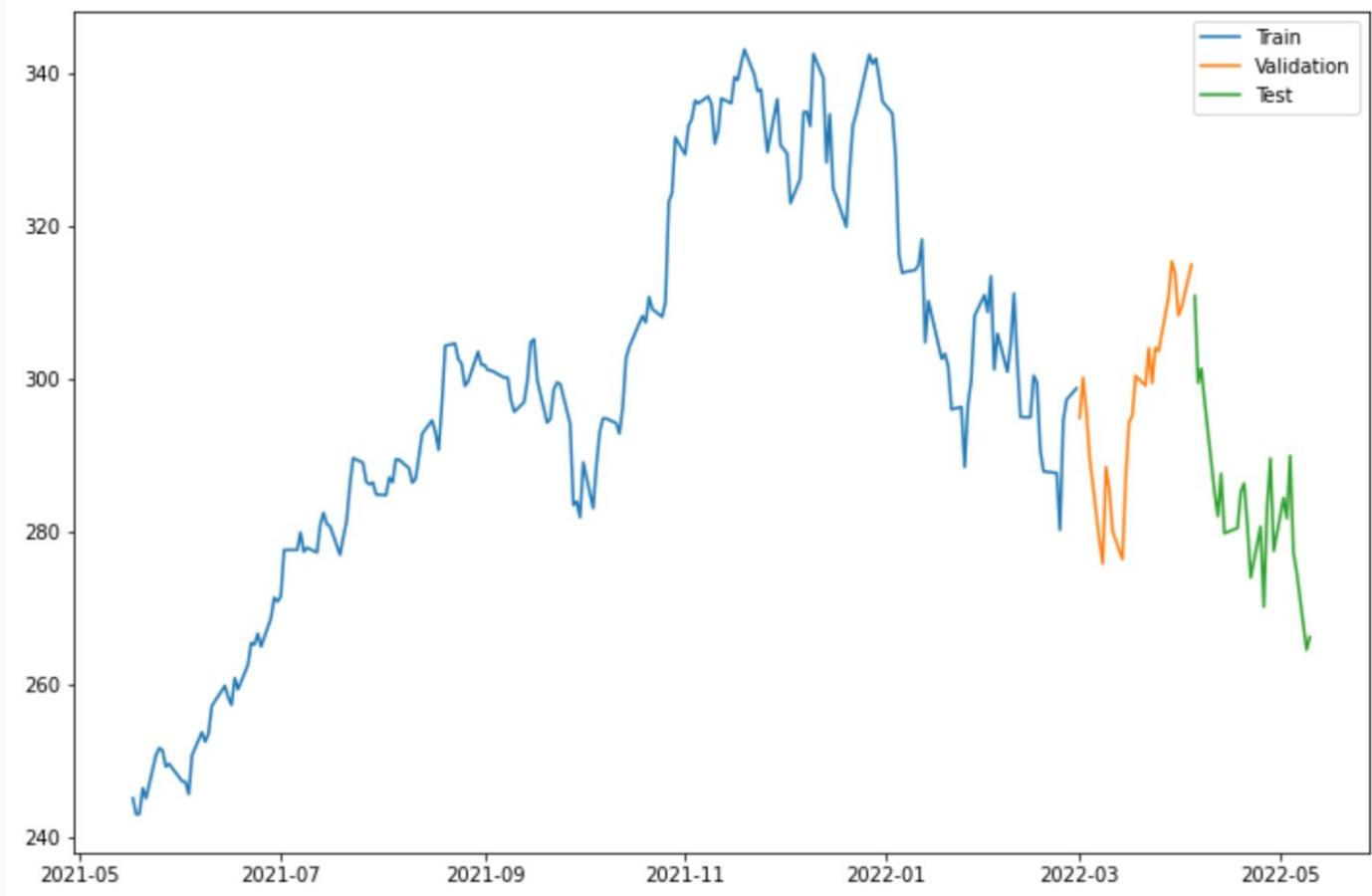
plt.tight_layout()
```



Financial Model Analysis with Kubeflow

Can LSTM model be used to predict
the stock price?

How to predict stock price for the next day?



What sequences means in a series?

Input: [300, 320, 310, 350, 390]

t0	300
t1	320
t2	310
t3	350
t4	390
t5	?

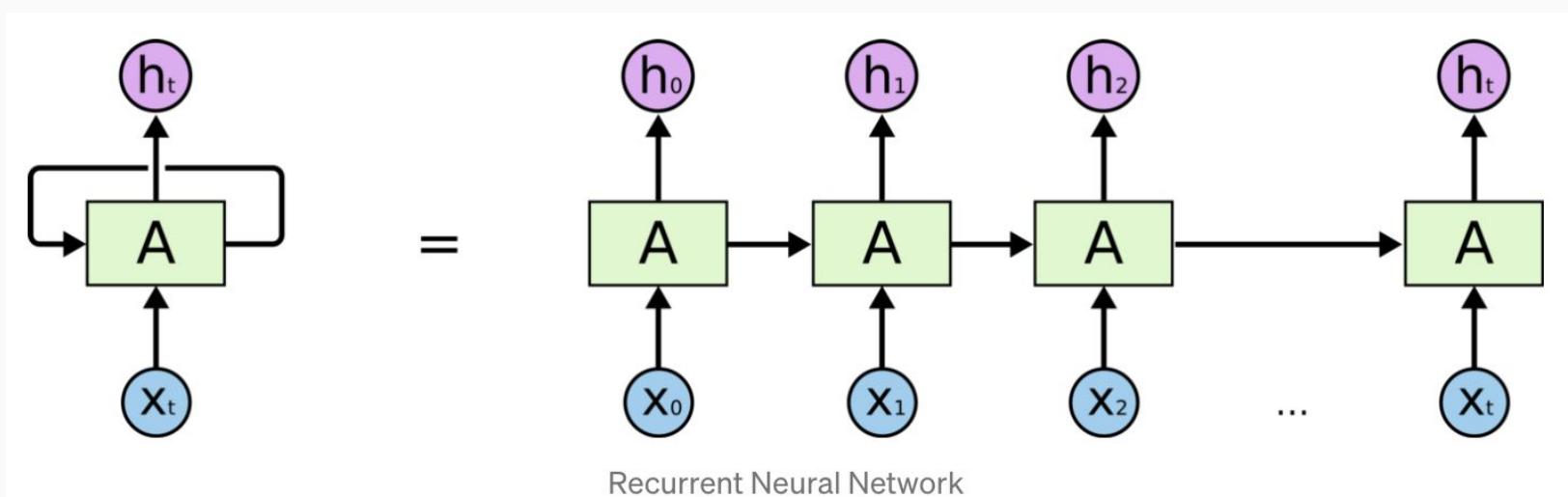
What sequences means in a sentence?

Input: I am a good guy

t0	I
t1	am
t2	a
t3	good
t4	guy
t5	?

What Is RNN (Recurrent Neural Network)?

Recurrent Neural Network (RNN) takes decisions on CURRENT (X_t) and PREVIOUS (X_{t-1}) inputs. Especially useful in topics including machine translation, speech recognition.

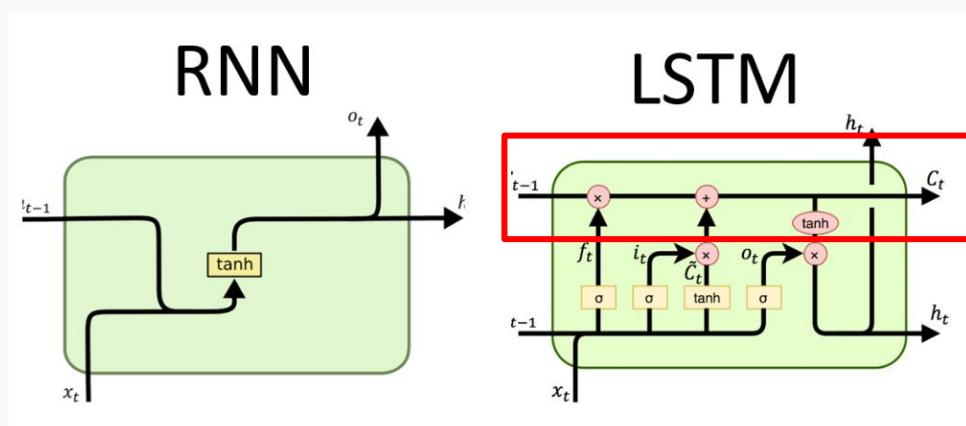


What Is LSTM (Long-short term memory)?

RNN only remember the latest things from X and it didn't remember(no memory) anything before at the beginning.

LSTM provides an information highway to let the neuron to selectively choose

1. forget from its memory (focus on the current inputs)
2. Listens to what information it added into memory (though information highway)



<https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3>

<https://www.quora.com/How-is-LSTM-different-from-RNN-In-a-layman-explanation>

How to formulate a sequence into a trainable dataset?

t0	t1	t2	t3	t4
300	320	310	350	390

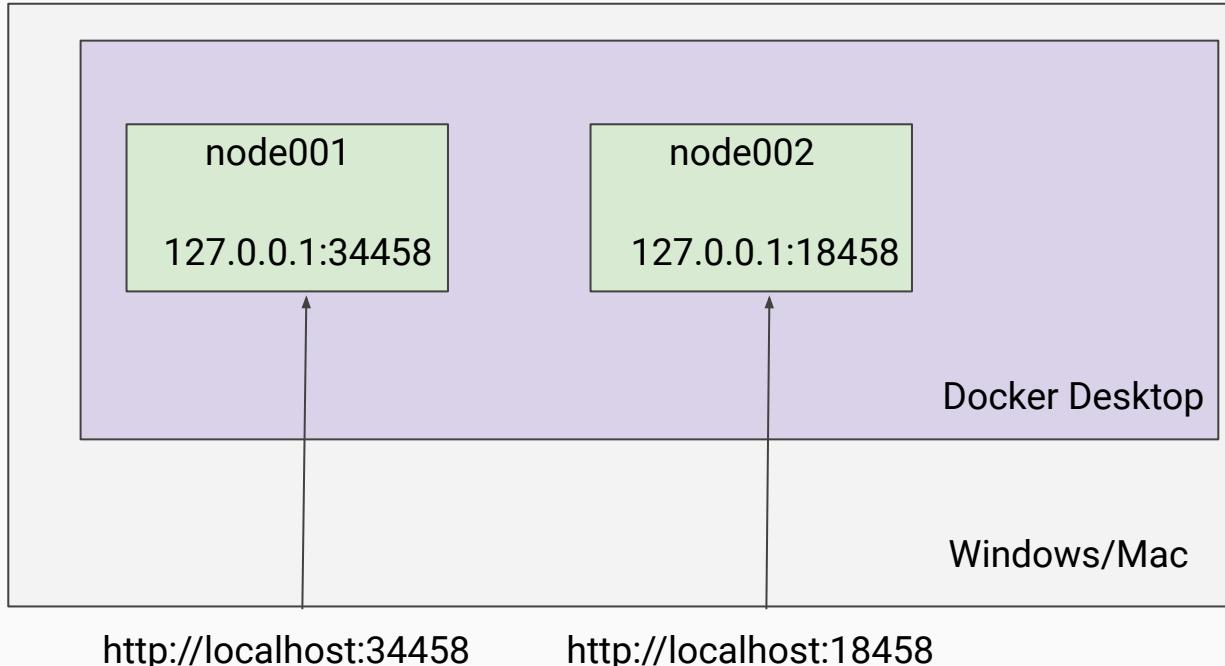
Windows size 3, we looks last three days data

Feature 1	Feature 2	Feature 3	Target
300	320	310	350
320	310	350	390

Environment Setup

Deployment Overview

Provisioning with multikf <https://github.com/FootprintAI/multikf/tree/main/docs>



Deployment Overview: few steps setup

```
// install dockerd (windows)
https://github.com/FootprintAI/kubeflow-workshop/blob/main/install/windows/dockerd.bat.md

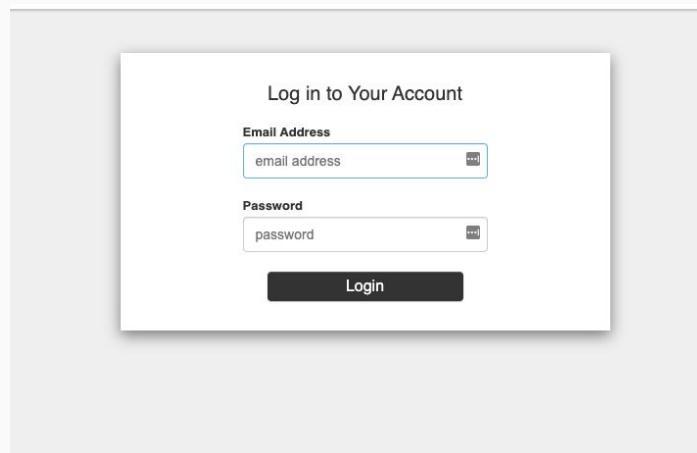
// install multikf (windows)
wget https://github.com/FootprintAI/multikf/raw/main/build/multikf.windows.exe
chmod +x multikf.windows.exe

// add an instances
./multikf.windows.exe add node002

// connect kubeflow
./multikf.windows.exe connect kubeflow node002
```

Wait! 所以我說那個帳號密碼呢?

Account: user@example.com
Password: 12341234



Step1: Use Notebook as Online IDE (1/3)

The screenshot shows the Kubeflow web interface. On the left, a sidebar menu is visible with the following items:

- Home
- Notebooks** (highlighted with a red box)
- Tensorboards
- Volumes
- Models
- Experiments (AutoML)
- Experiments (KFP)
- Pipelines
- Runs
- Recurring Runs
- Artifacts
- Executions

At the bottom of the sidebar, there is a link to "Privacy • Usage Reporting" and a note about the build version: "build version dev_local".

The main content area is titled "Notebooks" and displays a table with the following columns: Status, Name, Type, Age, Image, GPUs, CPUs, Memory, and Volumes. There is no data in the table.

In the top right corner of the main content area, there is a blue button labeled "+ NEW NOTEBOOK". This button and the "Notebooks" menu item are both highlighted with red boxes. A large red arrow points upwards towards the "+ NEW NOTEBOOK" button.

Step1: Use Notebook as Online IDE (2/3)

The screenshot shows the Kubeflow interface for creating a new Notebook Server. A red box highlights the 'Name' field containing 'demo'. Another red box highlights the 'Namespace' field containing 'kubeflow-user-example-com'. A large red arrow points from the text 'Specify a name and resources like CPU and Memory' to the 'CPU / RAM' section, which includes fields for 'Requested CPU' (0.5) and 'Requested Memory in GiB' (1). The 'Image' section is also visible.

Kubeflow

kubeflow-user-example-c...

Specify the name of the Notebook Server and the Namespace it will belong to.

Name: demo

Namespace: kubeflow-user-example-com

Image

A starter Jupyter Docker Image with a baseline deployment of ML packages

Custom Image

jupyterlab 1 2

Image: j1r0q0g6/notebooks/notebook-servers/jupyter-tensorflow-full:v1.4

Advanced Options

CPU / RAM

Specify the total amount of CPU and RAM reserved by your Notebook Server. For CPU-intensive workloads, you can choose more than 1 CPU (e.g. 1.5).

Requested CPU: 0.5

Requested Memory in GiB: 1

Advanced Options

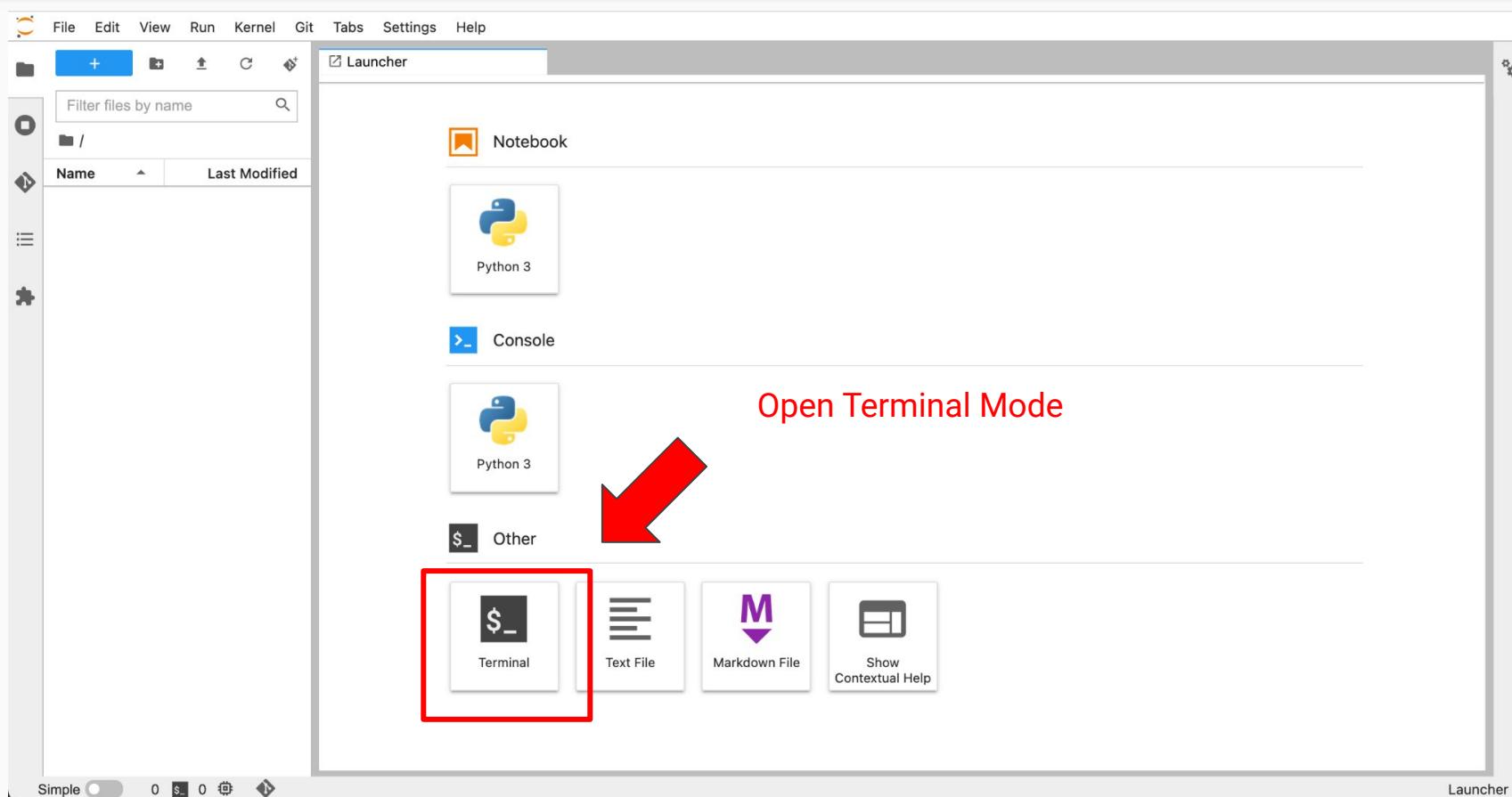
Specify a name and resources like CPU and Memory

Step1: Use Notebook as Online IDE (3/3)

The screenshot shows the Kubeflow interface with the 'Notebooks' section selected in the sidebar. The main area displays a table of notebooks with columns for Status, Name, Type, Age, Image, GPUs, CPUs, Memory, and Volumes. Two notebooks are listed: 'demo1' and 'demo2'. Both are in a 'Running' state (indicated by green checkmarks). The 'demo1' row shows it was created 20 hours ago, uses the 'jupyter-scipy:v1.4' image, and has 0 GPUs, 0.5 CPUs, 1Gi memory, and 1Gi volumes. The 'demo2' row shows it was created 2 hours ago, uses the 'jupyter-tensorflow-full:v1.4' image, and has 0 GPUs, 0.5 CPUs, 1Gi memory, and 1Gi volumes. Each row has a 'CONNECT' button, a copy icon, and a delete icon. A red box highlights the 'CONNECT' button for 'demo2', and a large red arrow points upwards from the bottom of the screen towards this button.

Status	Name	Type	Age	Image	GPUs	CPUs	Memory	Volumes
✓	demo1	jupyter	20 hours ago	jupyter-scipy:v1.4	0	0.5	1Gi	1Gi
✓	demo2	jupyter	2 hours ago	jupyter-tensorflow-full:v1.4	0	0.5	1Gi	1Gi

Step2: Use terminal to download the materials (1/3)



Step2: Use terminal to download the materials (2/3)

The screenshot shows the Jupyter Notebook interface. On the left is a file browser sidebar. In the center is a terminal window titled "Terminal 1" showing the output of a git clone command. A red box highlights the command and its output. A large red arrow points upwards from a code block below towards the terminal window. The code block contains the command: `git clone https://github.com/footprintai/kubeflow-workshop`.

```
groups: cannot find name for group ID 1222
(base) jovyan@dem01-0:~$ git clone https://github.com/footprintai/kubeflow-workshop
Cloning into 'kubeflow-workshop'...
remote: Enumerating objects: 164, done.
remote: Counting objects: 100% (164/164), done.
remote: Compressing objects: 100% (98/98), done.
remote: Total 164 (delta 89), reused 133 (delta 62), pack-reused 0
Receiving objects: 100% (164/164), 1.71 MiB | 8.29 MiB/s, done.
Resolving deltas: 100% (89/89), done.
(base) jovyan@dem01-0:~$
```

```
git clone https://github.com/footprintai/kubeflow-workshop
```

Simple 1 0 Terminal 1 48

Step2: Use terminal to download the materials (3/3)

The screenshot shows a Jupyter Notebook interface. On the left is a file browser pane with a sidebar containing icons for file operations like new file, copy, move, and delete. A search bar at the top of the browser allows filtering by file name. Below the search bar is a list of files under the path `/kubeflow-workshop / pipelines /`. The list includes:

- img (directory)
- 0.helloworld.ipynb
- 1.conditional-flow.ipynb
- 2.persistvolume.ipynb
- 3.calc_metrics.ipynb
- 4.mnist.ipynb
- 5.auto-mnist-with-katib.ipynb
- 6.kfserving.ipynb
- 7.kfserving-canary-rollout.ipynb
- kfp.ipynb
- testdata.jpg
- testdata2.jpg
- testdata3.jpg

A red box highlights the first seven items in the list. To the right of the file browser is a terminal window titled "Terminal 1". It displays the output of a `git clone` command:

```
groups: cannot find name for group ID 1337
(base) jovyan@demo1-0:~$ git clone https://github.com/footprintai/kubeflow-workshop
Cloning into 'kubeflow-workshop'...
remote: Enumerating objects: 164, done.
remote: Counting objects: 100% (164/164), done.
remote: Compressing objects: 100% (98/98), done.
remote: Total 164 (delta 89), reused 133 (delta 62), pack-reused 0
Receiving objects: 100% (164/164), 1.71 MiB | 8.29 MiB/s, done.
Resolving deltas: 100% (89/89), done.
(base) jovyan@demo1-0:~$
```

At the bottom of the terminal window, there is a status bar with the text "Terminal 1".

Financial Model Builder

Step3: Open financial model ipynb under tutorials/stockprice

The screenshot shows a Jupyter Notebook interface. On the left, there is a file browser with a red box highlighting the list of files in the 'tutorials/stockprice/' directory. The list includes '1.visualize-and-build-stock.ipynb' which was modified 'seconds ago'. The main area is a code editor titled '1.visualize-and-build-stock'. It contains Python code for stock price prediction using LSTM. The code includes comments explaining the purpose of the notebook, how to use Yahoo Finance via the yfinance package, and how to partition the data for training and prediction. It also installs required packages like pandas_datareader and yfinance, imports pandas and numpy, and imports matplotlib.pyplot for plotting. The code editor has tabs for 'Code', 'git', and 'Python 3'.

```
[ ]: # This jupyter notebook demonstrates how to predict stock with LSTM(long-short term memory) to
# predict stock price of the next day

# In this tutorial, we are going to use yahoo finance (called yfinance https://github.com/raroussi/yfinance)
# packages to download market data in daily basis.

# For model training and prediction, we partition the whole stock price series into several time-windowed (or
# sequence for input and use tensorflow to build the model.

# reference:
# [1] https://colab.research.google.com/drive/1Bk4zPQwAfzoSHZokKUefKL1s6lqmam6S?usp=sharing#scrollTo=8b-JsTv
# [2] https://www.kaggle.com/code/faressayah/stock-market-analysis-prediction-using-lstm/notebook

[ ]: # Install the required package with script mode
!pip install pandas_datareader yfinance

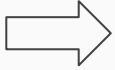
[4]: import pandas as pd
import numpy as np

# For plot
import matplotlib.pyplot as plt

# For reading stock data from yahoo
from pandas_datareader.data import DataReader
import yfinance as yf
```

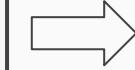
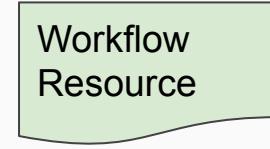
Kubeflow Terms

Kubeflow Terms

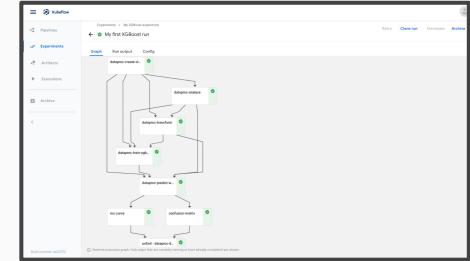


```
1.1: with open('requirements.txt', 'w') as f:
1.2:     f.write("tensorflow==1.8.9\n")
1.3:     pip install -r requirements.txt --upgrade --user
1.4:
1.5: import kfp
1.6: import os
1.7:
1.8: def echo(text):
1.9:     return kfp.components.create_op_component(
2.0:         name="echo",
2.1:         image="library/bash:4.4.23",
2.2:         command=["sh", "-c", "echo $1 | tee $2"],
2.3:         arguments={"text": "$0", "text2": "$1"})
2.4:
2.5: def argo_parallel():
2.6:     return kfp.components.create_op_component(
2.7:         name="argo_parallel",
2.8:         image="library/bash:4.4.23",
2.9:         command=["sh", "-c", "echo $1 | tee $2"],
3.0:         arguments={"text": "$0", "text2": "$1"})
3.1:
3.2: del Pipeline
3.3:
3.4: # run this file to demonstrate execution order management.
3.5: # run `argo_parallel` pipeline to demonstrate parallel execution order.
3.6: # run `echo` pipeline to demonstrate sequential execution order.
3.7: step1_task = echo_op()
3.8: step2_task = argo_parallel()
3.9: step3_task = echo_op()
3.10: step2_task.add_parallel_argo_parallel_task(step3_task)
3.11:
3.12: # generate workflow artifacts in .zip format
3.13: My_pipeline=compile_parallel_compiled_pipeline_parallel_pipeline('yellowworld.zip')
```

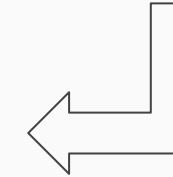
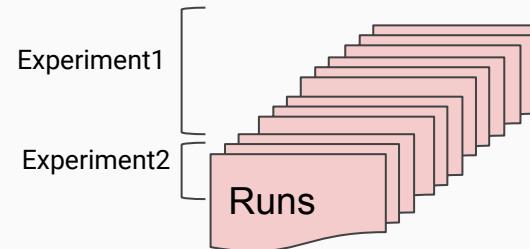
Compiled



Create a Pipeline



Pipeline Code



Create Run

Hello World Example

Step4: Compile helloworld.ipynb (1/2)

The screenshot shows a Jupyter Notebook interface with a terminal window titled "Terminal 1". The terminal displays Python code for generating workflow artifacts. A specific line of code is highlighted with a red border:

```
[ ]: # generate workflow artifacts in .zip format
kfp.compiler.Compiler().compile(execution_order_pipeline, 'helloworld.zip')
```

The rest of the code in the terminal is as follows:

```
[ ]: with open("requirements.txt", "w") as f:
    f.write("kfp==1.8.9\n")

!pip install -r requirements.txt --upgrade --user

[ ]: import kfp
from kfp import dsl

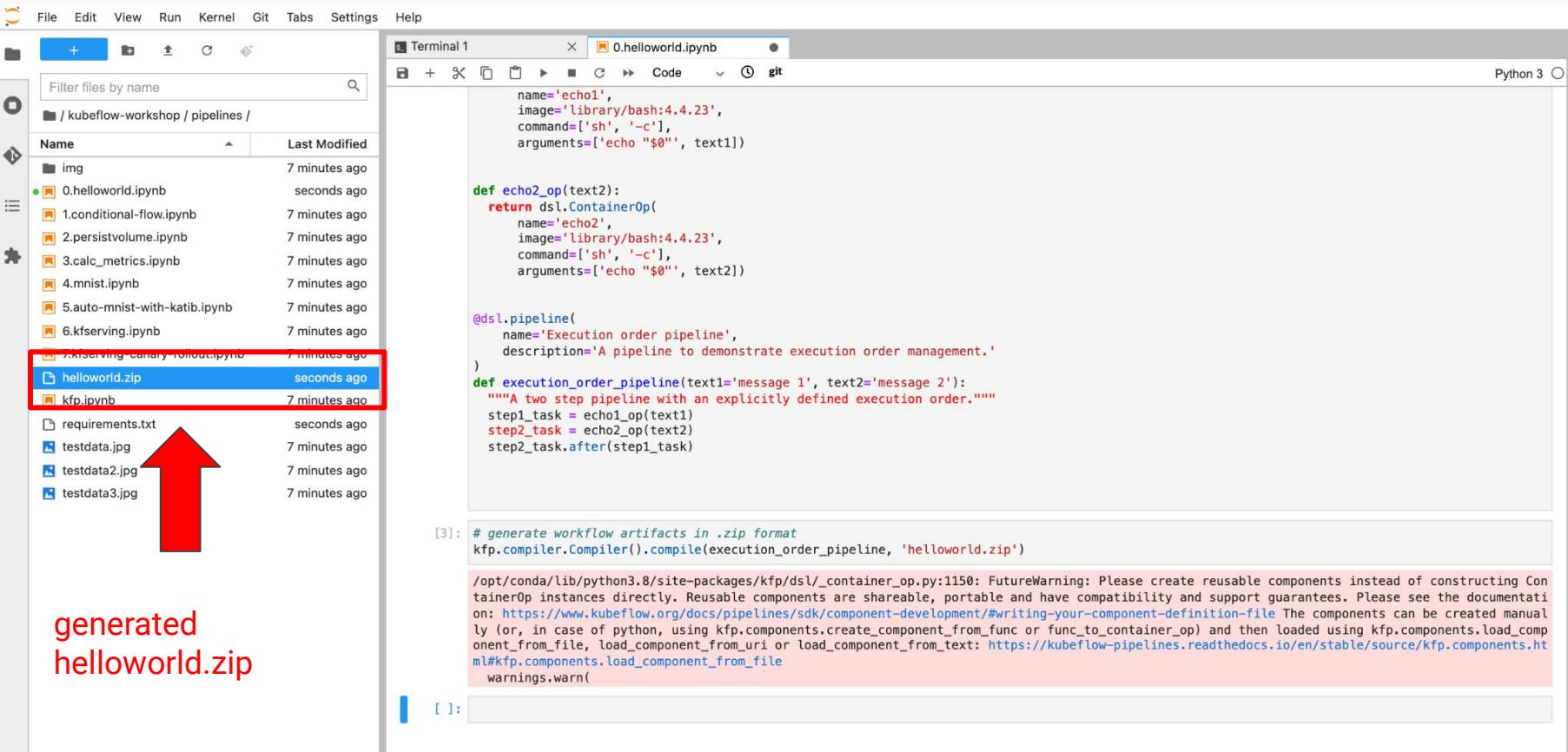
def echo1_op(text1):
    return dsl.ContainerOp(
        name='echo1',
        image='library/bash:4.4.23',
        command=['sh', '-c'],
        arguments=['echo "$0"', text1])

def echo2_op(text2):
    return dsl.ContainerOp(
        name='echo2',
        image='library/bash:4.4.23',
        command=['sh', '-c'],
        arguments=['echo "$0"', text2])

@dsl.pipeline(
    name='Execution order pipeline',
    description='A pipeline to demonstrate execution order management.')
def execution_order_pipeline(text1='message 1', text2='message 2'):
    """A two step pipeline with an explicitly defined execution order."""
    step1_task = echo1_op(text1)
    step2_task = echo2_op(text2)
    step2_task.after(step1_task)
```

The left sidebar shows a file tree with several Jupyter notebooks and other files. The notebook "0.helloworld.ipynb" is currently selected.

Step4: Compile helloworld.ipynb (2/2)



The screenshot shows a Jupyter Notebook interface with a sidebar containing a file tree and a main area with a terminal window.

File Tree (Sidebar):

- / kubeflow-workshop / pipelines /
- Name Last Modified
- img 7 minutes ago
- 0.helloworld.ipynb seconds ago
- 1.conditional-flow.ipynb 7 minutes ago
- 2.persistvolume.ipynb 7 minutes ago
- 3.calc_metrics.ipynb 7 minutes ago
- 4.mnist.ipynb 7 minutes ago
- 5.auto-mnist-with-katib.ipynb 7 minutes ago
- 6.kfserving.ipynb 7 minutes ago
- 7.kfserving_end-to-end.ipynb 7 minutes ago
- helloworld.zip seconds ago** (highlighted with a red box)
- kfp.ipynb 7 minutes ago
- requirements.txt seconds ago
- testdata.jpg 7 minutes ago
- testdata2.jpg 7 minutes ago
- testdata3.jpg 7 minutes ago

Terminal Window:

```
Terminal 1 x 0.helloworld.ipynb
Python 3 ○

name='echo1',
image='library/bash:4.4.23',
command=['sh', '-c'],
arguments=['echo "$0"', text1]

def echo2_op(text2):
    return dsl.ContainerOp(
        name='echo2',
        image='library/bash:4.4.23',
        command=['sh', '-c'],
        arguments=['echo "$0"', text2])

@dsl.pipeline(
    name='Execution order pipeline',
    description='A pipeline to demonstrate execution order management.')
def execution_order_pipeline(text1='message 1', text2='message 2'):
    """A two step pipeline with an explicitly defined execution order."""
    step1_task = echo1_op(text1)
    step2_task = echo2_op(text2)
    step2_task.after(step1_task)

[3]: # generate workflow artifacts in .zip format
kfp.compiler.Compiler().compile(execution_order_pipeline, 'helloworld.zip')

/opt/conda/lib/python3.8/site-packages/kfp/dsl/_container_op.py:1150: FutureWarning: Please create reusable components instead of constructing ContainerOp instances directly. Reusable components are shareable, portable and have compatibility and support guarantees. Please see the documentation on: https://www.kubeflow.org/docs/pipelines/sdk/component-development/#writing-your-component-definition-file The components can be created manually (or, in case of python, using kfp.components.create_component_from_func or func_to_container_op) and then loaded using kfp.components.load_component_from_file, load_component_from_uri or load_component_from_text: https://kubeflow-pipelines.readthedocs.io/en/stable/source/kfp.components.html#kfp.components.load_component_from_file
warnings.warn(
```

A large red arrow points from the highlighted "helloworld.zip" entry in the sidebar up towards the terminal output, indicating the result of the compilation command.

generated helloworld.zip

Step4: Create a Pipeline (1/7)

The screenshot shows the Kubeflow Pipelines interface. On the left, a sidebar menu is visible with the following items:

- Home
- Notebooks
- Tensorboards
- Volumes
- Models
- Experiments (AutoML)
- Experiments (KFP) (highlighted with a red box)
- Pipelines (highlighted with a red box)
- Runs
- Recurring Runs
- Artifacts
- Executions

The main content area is titled "Pipelines" and displays a list of existing pipelines. A red box highlights the "Upload pipeline" button in the top right corner, and a large red arrow points towards it from below. The table lists the following pipelines:

<input type="checkbox"/>	Pipeline name	Description	Upload
<input type="checkbox"/>	[Tutorial] V2 lightweight Python com...	source code Shows different component input and output options for KFP v2 components.	11/30/2021, 1:02:25 PM
<input type="checkbox"/>	[Tutorial] DSL - Control structures	source code Shows how to use conditional execution and exit handlers. This pipeline will randomly fail to demonstr...	11/30/2021, 1:02:24 PM
<input type="checkbox"/>	[Tutorial] Data passing in python co...	source code Shows how to pass data between python components.	11/30/2021, 1:02:23 PM
<input type="checkbox"/>	[Demo] TFX - Taxi tip prediction mod...	source code GCP Permission requirements. Example pipeline that does classification with model analysis based on...	11/30/2021, 1:02:22 PM
<input type="checkbox"/>	[Demo] XGBoost - Iterative model tra...	source code This sample demonstrates iterative training using a train-eval-check recursive loop. The main pipeline ...	11/30/2021, 1:02:21 PM

At the bottom right of the main area, there is a "Rows per page: 10" dropdown and navigation arrows (< >).

Step4: Create a Pipeline (2/7)

Kubeflow

Pipeline Versions

← Upload Pipeline or Pipeline Version

Create a new pipeline Create a new pipeline version under an existing pipeline

Upload pipeline with the specified package.

Pipeline Name*
0.helloworld

Pipeline Description*
0.helloworld

Choose a pipeline package file from your computer, and give the pipeline a unique name.
You can also drag and drop the file here.
For expected file format, refer to [Compile Pipeline Documentation](#).

Upload a file
helloworld (19).zip Import by url
Package Url

Code Source (optional)

Create Cancel

1.Pipeline Name

2.specify zip file location

3.Create

Step4: Create a Pipeline (3/7)

The screenshot shows the Kubeflow Pipelines interface. On the left is a dark sidebar with navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, Artifacts, and Executions. Below this is a "Manage Contributors" section.

The main area is titled "Pipelines" and shows a pipeline named "0.helloworld (0.helloworld)". It displays a "Graph" view of the pipeline, which consists of two nodes: "echo1" at the top and "echo2" below it, connected by a downward arrow. There is also a "YAML" tab. A "Simplify Graph" button is present.

At the top right of the main area are three buttons: "+ Create run", "+ Upload version", and "+ Create experiment". The "+ Create experiment" button is highlighted with a red box and a large red arrow pointing towards it from the bottom right. To the right of the arrow, the text "Create an experiment" is written in red.

A summary panel at the bottom left provides details about the pipeline:

Summary	
ID	6f25028f-01e3-4acd-9389-7ec2031fb04b
Version	0.helloworld
Version source	(no value)

Step4: Create a Pipeline (4/7)

The screenshot shows the Kubeflow interface for creating a new experiment. On the left is a dark sidebar with navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP) (which is expanded to show Pipelines, Runs, Recurring Runs, Artifacts, and Executions), Manage Contributors, and GitHub integration.

The main area is titled "Experiments" and "← New experiment". It has a sub-section titled "Experiment details" with a descriptive text: "Think of an Experiment as a space that contains the history of all pipelines and their associated runs".

A form is displayed for entering experiment details:

- Name:** A text input field containing "0.helloworld.exp" is highlighted with a red border.
- Description (optional):** An empty text input field.
- Buttons:** A row of buttons with "Next" highlighted with a red border and a large red arrow pointing to it from the right. Other buttons include "Cancel".

A red annotation text on the right side of the form reads: "Enter experiment name and press Next button."

Step4: Create a Pipeline (5/7)

The screenshot shows the Kubeflow UI for creating a pipeline run. The left sidebar contains navigation links: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP) (selected), Pipelines, Runs, Recurring Runs, Artifacts, Executions, Manage Contributors, GitHub, Documentation, Privacy + Usage Reporting, and build version dev_local.

The main area is titled "Run details". It includes fields for "Pipeline", "Pipeline Version", "Run name", "Description (optional)", "Experiment", "Service Account (Optional)", "Run Type" (set to "One-off"), and "Run parameters". The "Run parameters" section contains fields for "text1" and "text2". At the bottom are "Start" and "skip this step" buttons.

Two red arrows point from the right towards the "Pipeline" and "Experiment" fields, which are highlighted with red boxes. Red text annotations provide instructions:

1. Run a pipeline and specify its version
2. Add its experiment name

Step4: Create a Pipeline (6/7)

Kubeflow

kubeflow-user-example-c... ▾

Experiments

← 0.helloworld.exp

Recurring run configs 0 active

Experiment description

Manage

Refresh Archive

Runs

+ Create run + Create recurring run Compare runs Clone run Archive

Active Archived

Filter runs

Run name	Status	Duration	Pipeline Version	Recurring Run	Start time
Run of 0.helloworld (1e261)	?	-	0.helloworld	-	12/2/2021, 4:33:10 PM

Rows per page: 10 < >

Run List

Step4: Create a Pipeline (7/7)

The screenshot shows the Kubeflow interface for creating a pipeline. On the left, the sidebar includes options like Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, Artifacts, Executions, Manage Contributors, and GitHub.

The main area displays a pipeline run titled "Run of 0.helloworld (1e261)". The "Graph" tab is selected, showing a flow from "echo1" to "echo2". A red box highlights this graph area. The "Run output" tab is also visible.

A modal window provides detailed information about the execution:

- Input/Output** tab is selected, showing "execution-order-pipeline-fxxfn-4223123588".
- Input parameters**: text1 (message 1)
- Input artifacts**: None
- Output parameters**: None
- Output artifacts**: main-logs (minio://mipeline/artifacts/execution-order-pipeline-qvlt5/2021/11/30/execution-order-pipeline-qvlt5-137025724/main.log, message 1)

A large red arrow points upwards from the "Output artifacts" section towards the "Running outputs" text.

Running outputs

Hyperparameter Example

Step5: Hyperparameter tuning with katib (1/4)

The screenshot shows a Jupyter Notebook interface. On the left, there is a file browser pane with a red border around it, displaying a list of files in the directory `/.../tutorials/stockprice-with-lstm/`. The files listed are:

- 1.visualize-... 32 minutes ago
- 2.optimize... 6 minutes ago
- helloworld... 7 minutes ago
- requireme... 8 hours ago

The main pane displays Python code for hyperparameter tuning using Katib. The code consists of two code cells:

```
[16]: ## import required pkgs
import kfp
import kfp.dsl as dsl
from kfp import components

from kubeflow.katib import ApiClient
from kubeflow.katib import V1beta1ExperimentSpec
from kubeflow.katib import V1beta1AlgorithmSpec
from kubeflow.katib import V1beta1EarlyStoppingSpec
from kubeflow.katib import V1beta1EarlyStoppingSetting
from kubeflow.katib import V1beta1ObjectiveSpec
from kubeflow.katib import V1beta1ParameterSpec
from kubeflow.katib import V1beta1FeasibleSpace
from kubeflow.katib import V1beta1TrialTemplate
from kubeflow.katib import V1beta1TrialParameterSpec

[17]: ## define katib objective, stopping criteria, and parameter spaces
experiment_name = "median-stop-lstm"
experiment_namespace = "kubeflow-user-example-com"

# Trial count specification.
max_trial_count = 4
max_failed_trial_count = 2
parallel_trial_count = 1

# Objective specification.
objective=V1beta1ObjectiveSpec(
    type="minimize",
    goal= 5
```

At the bottom of the interface, there are status indicators: "Simple" (button), "1 \$ 0" (button), "No Kernel | Idle" (text), "Saving completed" (text), "Mode: Command" (text), "Ln 7, Col 20" (text), and the file name "2.optimize-model-wit".

Step5: Hyperparameter tuning with katib (2/4)

The screenshot shows the Kubeflow interface for managing experiments. On the left, a sidebar lists various resources: Home, Notebooks, Tensorboards, Volumes, Models, Experiments (AutoML), Experiments (KFP), Pipelines, Runs, Recurring Runs, and Artifacts. The main area is titled 'Experiments' and shows the details for the experiment 'hello1'. It includes sections for 'Recurring run configs' (0 active) and 'Experiment description'. Below this is a 'Runs' section with tabs for 'Active' (selected) and 'Archived'. A red box highlights the first row of the 'Active' run table, which lists three runs. The columns in the table are: Run name, Status, Duration, Pipeline Version, Recurri..., Start time, quotient, and remainder.

Run name	Status	Duration	Pipeline Version	Recurri...	Start time	quotient	remainder
Run of hello-world_version_1	?	-	hello-world_version_1	-	5/17/2022, 9:42:...		
Run of hello-world_version_1	✓	0:06:19	hello-world_version_1	-	5/17/2022, 9:30:...		
Run of hello-world_version_1	✓	0:01:24	hello-world_version_1	-	5/17/2022, 9:23:...	0.000	6.000

Step5: Hyperparameter tuning with katib (3/4)

The screenshot shows the Kubeflow interface with the sidebar navigation bar on the left and the main 'Experiments' view on the right.

Left Sidebar (Kubeflow Navigation):

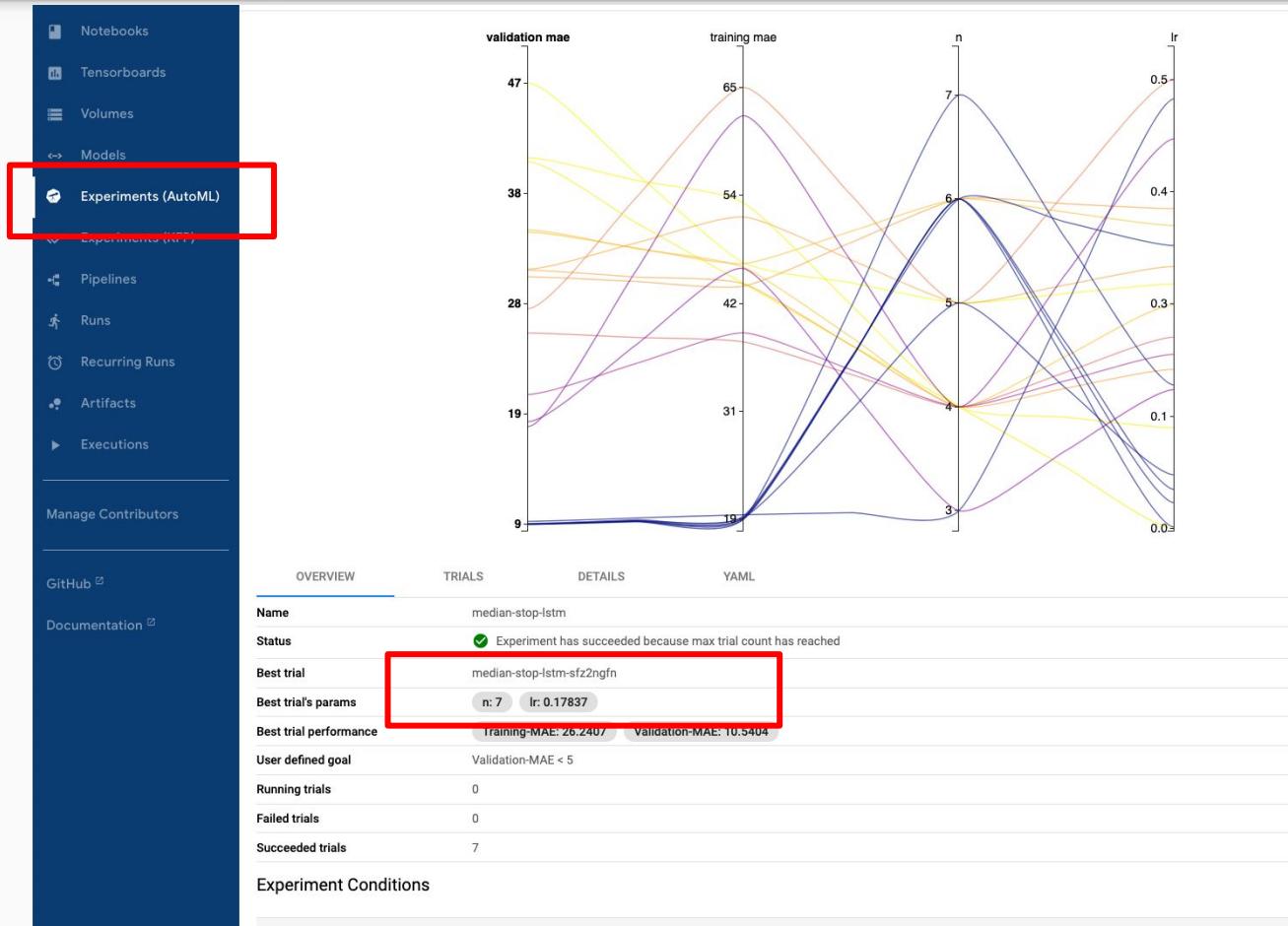
- Home
- Notebooks
- Tensorboards
- Volumes
- Models
- Experiments (AutoML)** (This item is highlighted with a red border.)
- Experiments (KFP)
- Pipelines
- Runs
- Recurring Runs

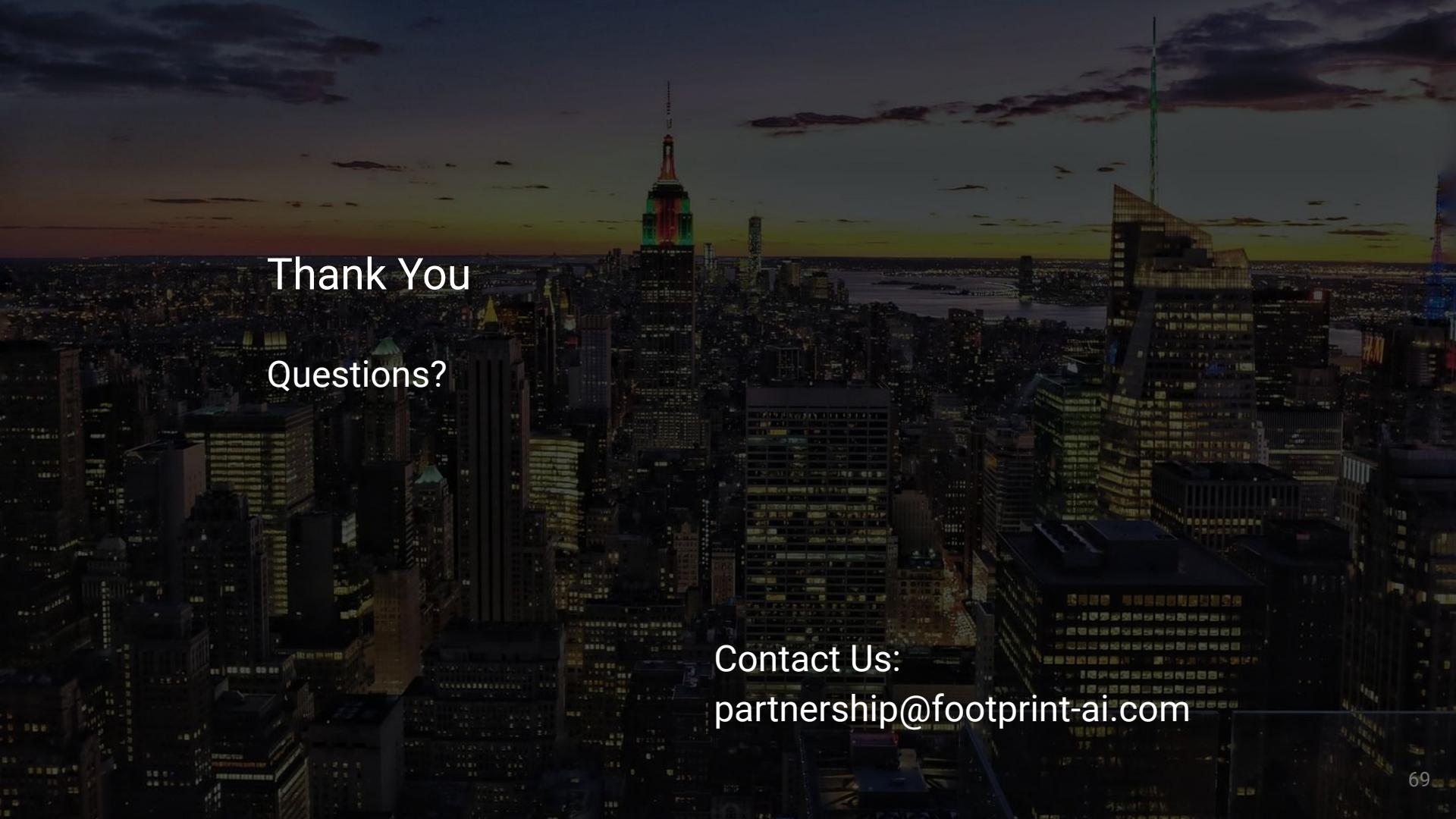
Main View (Experiments):

The main view displays a table of experiments. A new experiment button (+ NEW EXPERIMENT) is located in the top right corner of the table header.

Status	Name	Age	Successful trials	Running trials	Failed trials	Optimal trial
✓	median-stop	10 hours ago	4	0	0	Validation accuracy: 0.92028
✓	median-stop-lstm	7 minutes ago	7	0	0	Training mae: 26.2407

Step5: Hyperparameter tuning with kтив (4/4)



The background of the slide is a photograph of a city skyline at night, likely New York City, with the Empire State Building prominently visible. The sky is a mix of dark blues and warm orange and yellow hues from the setting sun.

Thank You

Questions?

Contact Us:
partnership@footprint-ai.com



***“The Best Engineers
Are Lazy”***

-Ancient Engineering Proverb

Materials

- Slides:
 - <https://github.com/FootprintAI/talks/tree/main/slides>
- Multikf
 - <https://github.com/FootprintAI/multikf>
- Kubeflow Workshop
 - <https://github.com/footprintai/kubeflow-workshop>