

AI Tech Labs 0 \Rightarrow 1

Zhenhua He

03/11/2022



High Performance
Research Computing
DIVISION OF RESEARCH

Original slides created by Dr. Jian Tao

AI Tech Labs

Lab I. JupyterLab (30 mins)

We will set up a Python virtual environment and run JupyterLab on the HPRC Grace Portal.

Lab II. Data Exploration (30 mins)

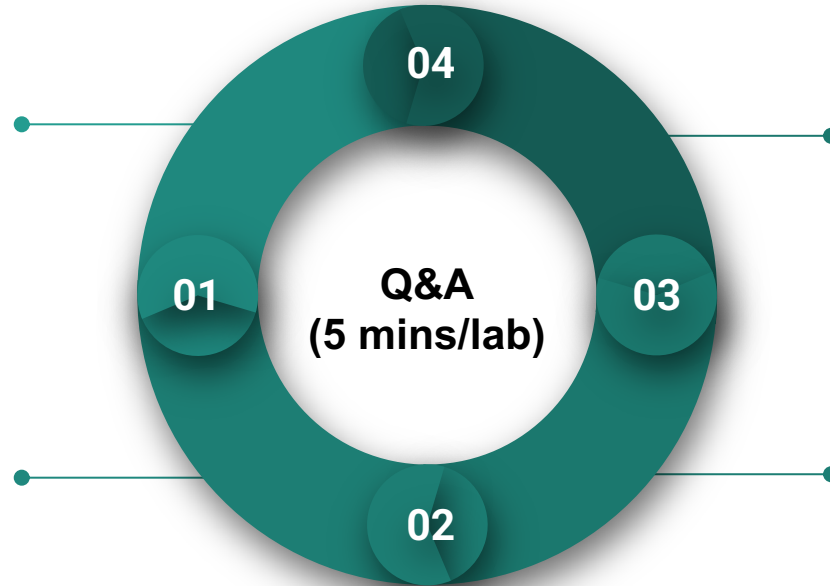
We will go through some examples with two popular Python libraries: Pandas and Matplotlib for data exploration.

Lab IV. Deep Learning (30 minutes)

We will learn how to use Keras to build and train a simple image classification model with deep neural network (DNN).

Lab III Machine Learning (30 minutes)

We will learn to use scikit-learn library for linear regression and classification applications.



Lab I. JupyterLab



File Edit View Run Kernel Tabs Settings Help

Files

- notebooks
- Data.ipynb (an hour ago)
- Fasta.ipynb (a day ago)
- Julia.ipynb (a day ago)
- Lorenz.ipynb (seconds ago)**
- R.ipynb (a day ago)
- iris.csv (a day ago)
- lightning.json (9 days ago)
- lorenz.py (3 minutes ago)

Running

Commands

Cell Tools

Output View

lorenz.py

In this Notebook we explore the Lorenz system of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

```
In [4]: from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

sigma 10.00
beta 2.67
rho 28.00

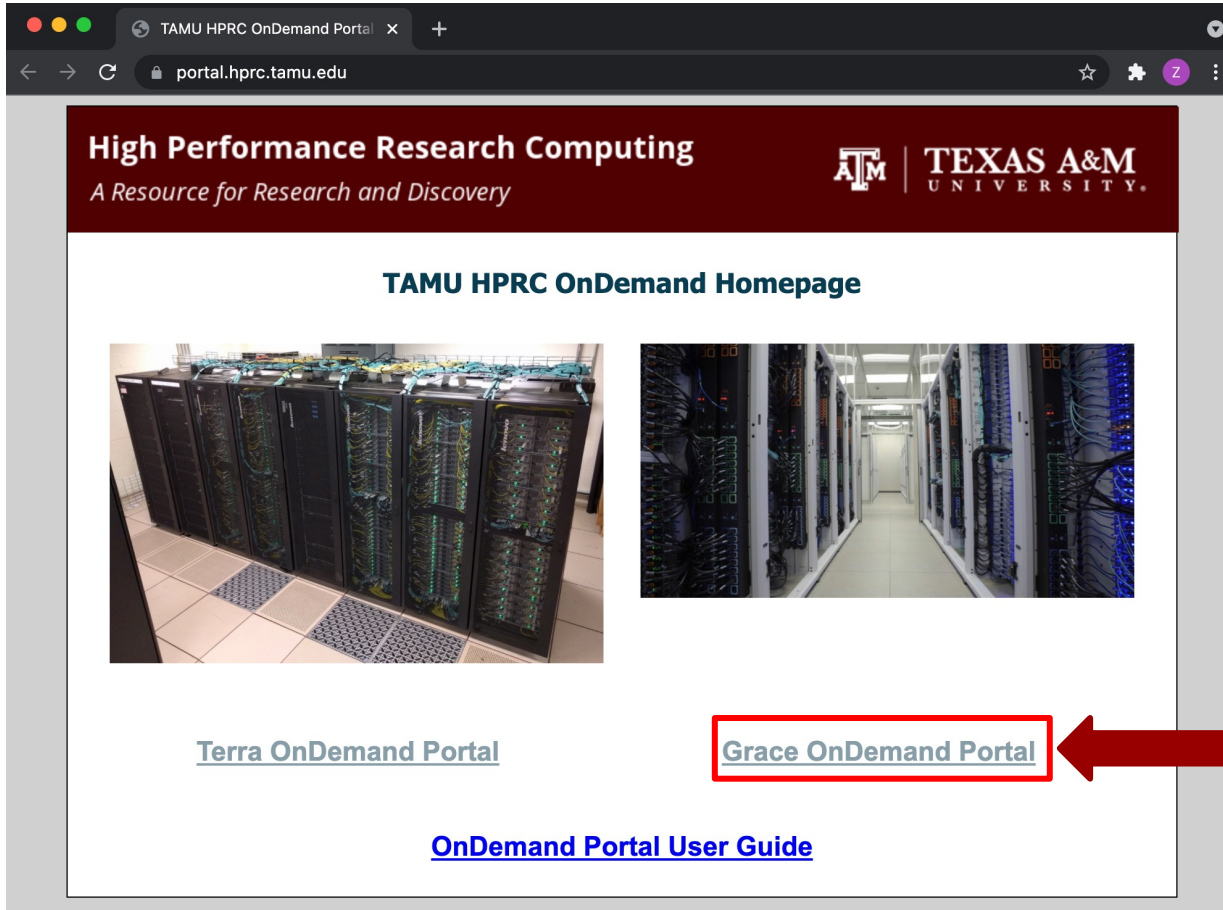
A 3D plot of the Lorenz attractor, showing a complex, swirling trajectory in a 3D space. The axes are labeled x, y, and z. The trajectory starts near the origin and spirals outwards, forming a shape that resembles a butterfly or a pair of wings. The plot is rendered with a semi-transparent surface, allowing the internal structure to be visible.

```
9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
19
20     def lorenz_deriv(x,y,z, t0, sigma=sigma, beta=beta, rho=rho):
21         """Compute the time-derivative of a Lorenz system."""
22         x, y, z = x,y,z
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
24
25     # Choose random starting points, uniformly distributed from -15 to 15
26     np.random.seed(1)
27     x0 = -15 + 30 * np.random.random((N, 3))
28
```

L1 - Resources

- Texas A&M High Performance Research Computing (HPRC)
- Grace Quick Start Guide
- HPRC Portal
- HPRC YouTube Channel
- Jupyter Project
- help@hprc.tamu.edu

Login HPRC Portal



The image shows a browser window displaying the TAMU HPRC OnDemand Portal homepage. The browser's address bar shows the URL `portal.hprc.tamu.edu`. The page header features the text "High Performance Research Computing" and "A Resource for Research and Discovery" on the left, and the Texas A&M University logo on the right. The main content area is titled "TAMU HPRC OnDemand Homepage" and contains two photographs of server racks. Below the photos are two links: "Terra OnDemand Portal" and "Grace OnDemand Portal". The "Grace OnDemand Portal" link is highlighted with a red box and a red arrow pointing to it from the right. At the bottom of the page is a link for "OnDemand Portal User Guide".

TAMU HPRC OnDemand Portal

portal.hprc.tamu.edu

High Performance Research Computing
A Resource for Research and Discovery

ATM | TEXAS A&M UNIVERSITY

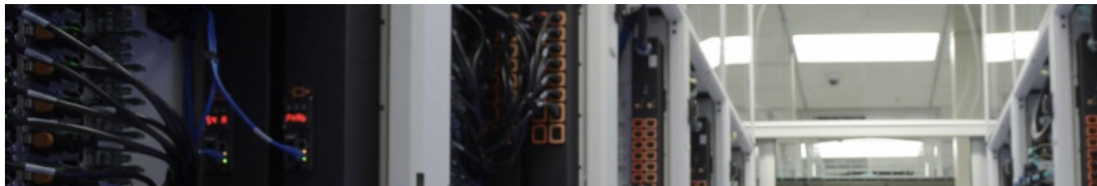
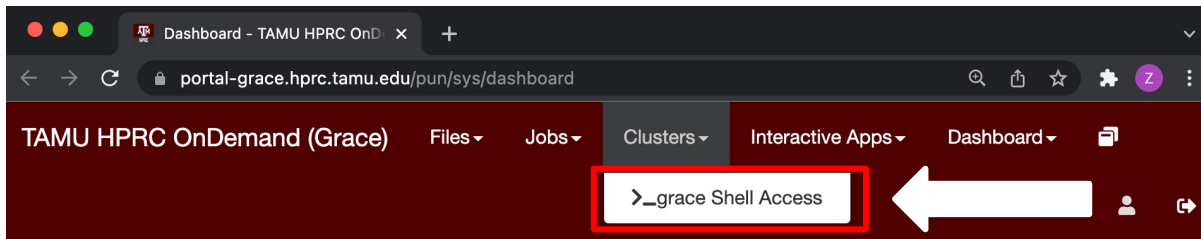
TAMU HPRC OnDemand Homepage

[Terra OnDemand Portal](#)

[Grace OnDemand Portal](#)

[OnDemand Portal User Guide](#)

Shell Access - I



OnDemand provides an integrated, single access point for all of your HPC resources.

Message of the Day

Grace Cluster Status, January 30 - February 04

UPDATE (8:09pm 02/04/2022): The Grace storage has been recovered from the Jan. 30th, Feb. 1st, and Feb. 3rd incidents involving storage software defects. The Grace maintenance has also been completed. Quota enforcement has also been restored for the Grace storage. Some users may have storage usage that is drastically over their intended home and scratch directory quotas. Please archive and remove any unneeded data if possible. Our apologies again for the extended unavailability of the Grace cluster.

IMPORTANT POLICY INFORMATION

Unauthorized use of HPRC resources is prohibited and subject to criminal prosecution.

Shell Access - II

```
happidence1@login1:~
portal-grace.hprc.tamu.edu/pun/sys/shell/ssh/grace.hprc.tamu.edu

*****
This computer system and the data herein are available only for authorized
purposes by authorized users. Use for any other purpose is prohibited and may
result in disciplinary actions or criminal prosecution against the user. Usage
may be subject to security testing and monitoring. There is no expectation of
privacy on this system except as otherwise provided by applicable privacy laws.
Refer to University SAP 29.01.03.M0.02 Acceptable Use for more information.
*****

Password:
Duo two-factor login for happidence1

Enter a passcode or select one of the following options:

1. Duo Push to XXX-XXX-9472
2. Phone call to XXX-XXX-9472
3. SMS passcodes to XXX-XXX-9472 (next code starts with: 1)

Passcode or option (1-3): 1
Success. Logging you in...
Last login: Mon Feb 28 21:09:35 2022 from 128.194.35.38

=====
|               Texas A&M University High Performance Research Computing               |
|-----|
| Website:           https://hprc.tamu.edu |
| Consulting:        help@hprc.tamu.edu (preferred) or (979) 845-0219 |
| Grace Documentation: https://hprc.tamu.edu/wiki/Grace |
| Terra Documentation: https://hprc.tamu.edu/wiki/Terra |
| YouTube Channel:   https://www.youtube.com/texasamhprc |
|-----|
=====

*****
*               === IMPORTANT POLICY INFORMATION ===               *
*****
```

Python Virtual Environment (VENV)

Load Modules

Create a VENV

Activate the VENV

Install Python Modules

Deactivate (when not used)

```
# clean up and load Anaconda  
cd $SCRATCH  
module purge  
module load Anaconda3/2021.11
```

```
# create a Python virtual environment  
conda create -n mylab
```

```
# activate the virtual environment  
conda activate mylab
```

```
# install required packages to be used in the portal  
conda install scikit-learn  
conda install tensorflow
```

```
# deactivate the virtual environment  
# conda deactivate
```


Common Anaconda Commands

```
# Conda virtual environment
```

```
conda info
```

```
conda create -n VENV
```

```
conda create -n VENV python=3.8
```

```
conda env list
```

```
# show Conda installation
```

```
# create a virtual environment
```

```
# create a venv with a py version
```

```
# list installed venv
```

```
# Conda package management
```

```
conda list
```

```
conda search PACKAGENAME
```

```
conda install PACKAGENAME
```

```
conda update PACKAGENAME
```

```
conda remove PACKAGENAME
```

```
# list all installed packages
```

```
# search a Conda package
```

```
# install a Conda package
```

```
# update a Conda package
```

```
# remove a Conda package
```

Check out Exercises

happidence1/AILabs

github.com/happidence1/AILabs

Search or jump to... Pull requests Issues Marketplace Explore

happidence1 / AILabs Public

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

Go to file Add file Code

happidence1	Add files via upload
data	Credit to Dr. Jian Tao at TAMU
images	Credit to Dr. Jian Tao at TAMU
01_jupyterlab.ipynb	Add files via upload
02_data_exploration_pandas.ipynb	Add files via upload

Clone

HTTPS SSH GitHub CLI

<https://github.com/happidence1/AILabs>

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

About

No description, website, or topics provided.

Readme

Releases

No releases published

[Create a new release](#)

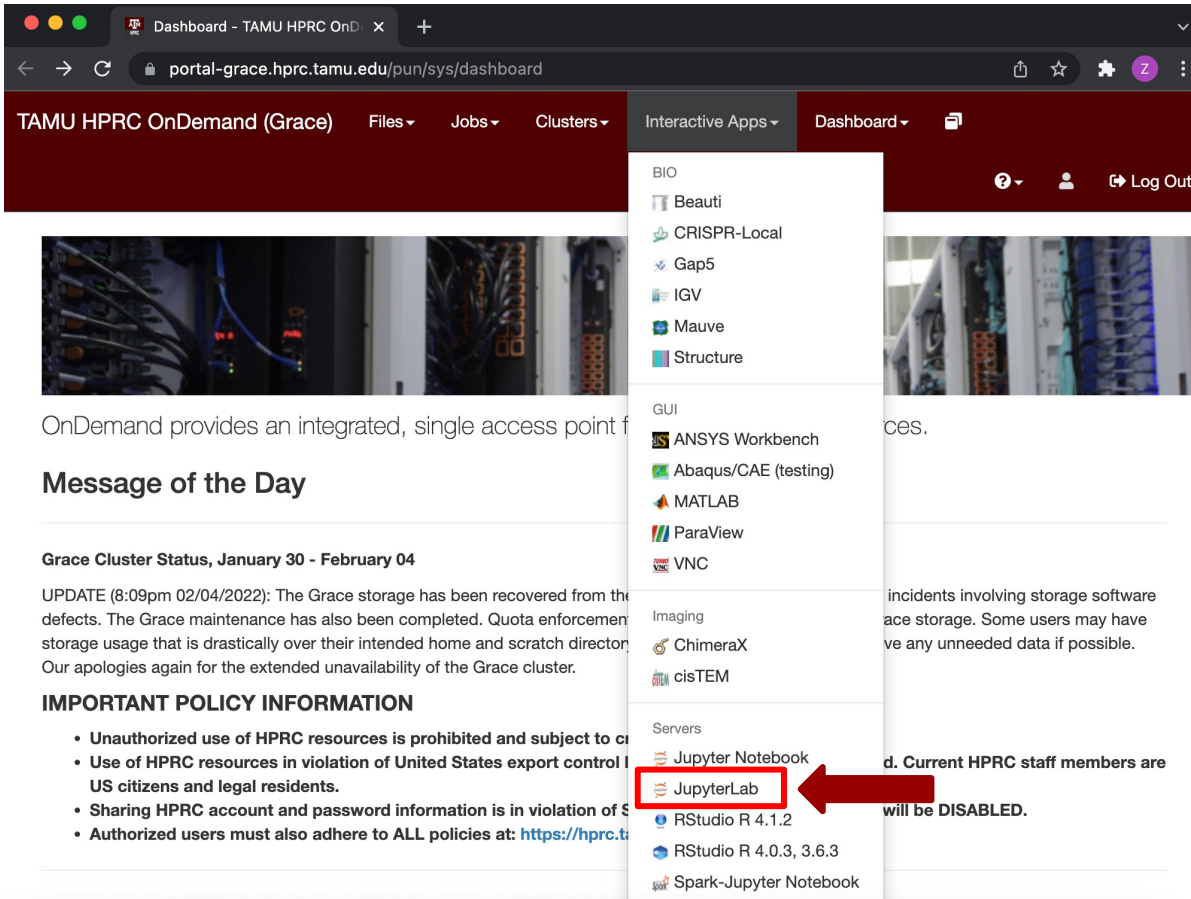
```
# git clone (check out) the Jupyter notebooks for the labs
git clone https://github.com/happidence1/AILabs.git
```

README.md

Languages

Jupyter Notebook 100.0%

Go to JupyterLab Page



The screenshot shows a web browser window with the URL `portal-grace.hprc.tamu.edu/pun/sys/dashboard`. The dashboard header includes navigation tabs: TAMU HPRC OnDemand (Grace), Files, Jobs, Clusters, Interactive Apps, and Dashboard. The 'Interactive Apps' menu is open, displaying a list of applications categorized by type:

- BIO
 - Beauti
 - CRISPR-Local
 - Gap5
 - IGV
 - Mauve
 - Structure
- GUI
 - ANSYS Workbench
 - Abaqus/CAE (testing)
 - MATLAB
 - ParaView
 - VNC
- Imaging
 - ChimeraX
 - cisTEM
- Servers
 - Jupyter Notebook
 - JupyterLab** (highlighted with a red box and arrow)
 - RStudio R 4.1.2
 - RStudio R 4.0.3, 3.6.3
 - Spark-Jupyter Notebook

The background of the dashboard shows a server rack and a 'Message of the Day' section with the following text:

OnDemand provides an integrated, single access point for...
Message of the Day
Grace Cluster Status, January 30 - February 04
UPDATE (8:09pm 02/04/2022): The Grace storage has been recovered from the...
IMPORTANT POLICY INFORMATION

- Unauthorized use of HPRC resources is prohibited and subject to...
- Use of HPRC resources in violation of United States export control... US citizens and legal residents.
- Sharing HPRC account and password information is in violation of S...
- Authorized users must also adhere to ALL policies at: <https://hprc.tamu.edu>

At the bottom of the page, a red warning message reads: **!! WARNING: THERE ARE ONLY NIGHTLY BACKUPS OF USER HOME DIRECTORIES. !!**

Set Virtual Environment

JupyterLab - TAMU HPRC OnD

portal-grace.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sys/jupyterlab/session_contexts/n...

TAMU HPRC OnDemand (Grace) Files Jobs Clusters Interactive Apps Dashboard Help Logged in as happidence1 Log Out

Home / My Interactive Sessions / JupyterLab

Interactive Apps

- BIO
- Beauti
- CRISPR-Local
- Gap5
- IGV
- Mauve
- Structure
- GUI
- ANSYS Workbench
- Abaqus/CAE (testing)
- MATLAB
- ParaView
- VNC
- Imaging
- ChimeraX
- cisTEM
- Servers
- Jupyter Notebook
- JupyterLab**

JupyterLab

This app will launch a [JupyterLab](#) server on the [Grace](#) cluster.

Module

Anaconda3/2021.11

Anaconda/3-x.x.x.x and Anaconda3 use Python3

Optional Environment to be activated

mylab

Enter the name of the environment to be activated. (Optional)

The default virtualenvs for Anaconda3/2021.11 and Python/3.8.2 have jupytermod which enables loading Imod modules.

Leave blank to use the [default](#) environment for the selected Module.

Your optional conda environment must have been previously built with one of the Anaconda or Python modules listed in the Module option above. See [instructions](#).

Select if you are part of the iHESP group and want to launch jupyterlab from your /ihesp directory

Number of hours

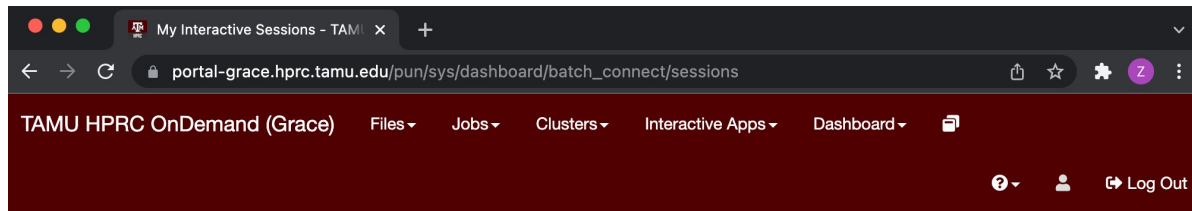
3

Number of cores:

1

Specify the number of cores. [1, 40], allocated on a node from the [Grace](#)

Connect to JupyterLab



Session was successfully created. x

Home / My Interactive Sessions

Interactive Apps

BIO

Beauti

CRISPR-Local

Gap5

IGV

Mauve

Structure

GUI

ANSYS Workbench

Abaqus/CAE (testing)

MATLAB

JupyterLab (3652078)

1 node | 1 core | Running

Host: >_c623

Delete

Created at: 2022-03-06 21:30:38 CST

Time Remaining: 2 hours and 58 minutes

Session ID: d99fc1ef-ecf-4be7-b563-d542008b719c

Connect to JupyterLab

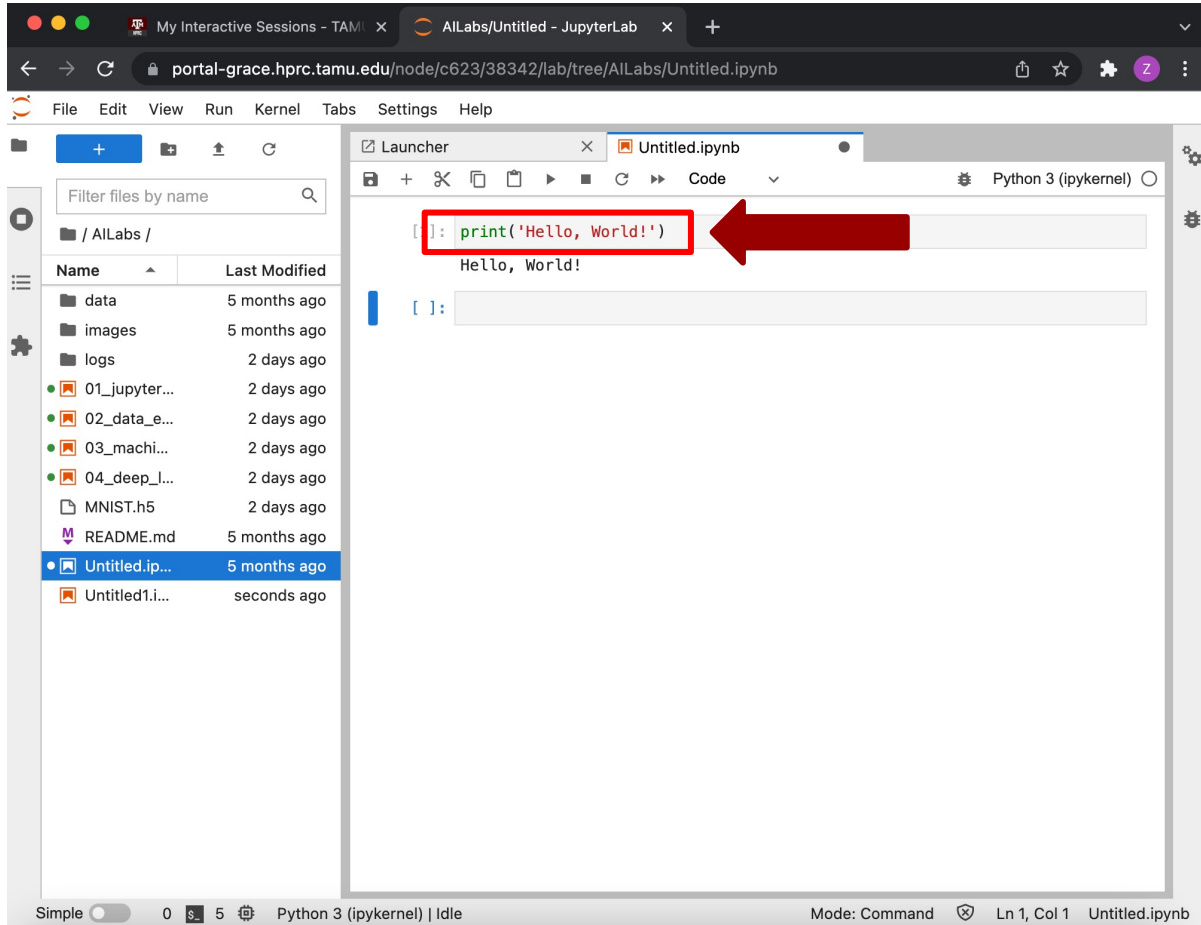


Create a Jupyter Notebook

The screenshot shows the JupyterLab interface in a web browser. The browser address bar displays the URL: `portal-grace.hprc.tamu.edu/node/c623/38342/lab/tree/AllLabs`. The JupyterLab interface includes a top menu bar with options: File, Edit, View, Run, Kernel, Tabs, Settings, and Help. On the left, there is a file browser pane with a search bar labeled "Filter files by name" and a list of files and folders. The main area is titled "Launcher" and shows a section for "AllLabs". Under "AllLabs", there is a "Notebook" section with a red box around the "Python 3 (ipykernel)" icon, and a red arrow pointing to it. Below the "Notebook" section is a "Console" section with a "Python 3 (ipykernel)" icon. At the bottom, there is an "Other" section with icons for Terminal, Text File, Markdown File, and Python File. The status bar at the bottom left shows "Simple" and "0 5". The status bar at the bottom right shows "Launcher".

Name	Last Modified
data	5 months ago
images	5 months ago
logs	2 days ago
01_jupyter...	2 days ago
02_data_e...	2 days ago
03_machi...	2 days ago
04_deep_l...	2 days ago
MNIST.h5	2 days ago
README.md	5 months ago
Untitled.ip...	5 months ago

Test JupyterLab



The screenshot displays the JupyterLab web interface. The browser address bar shows the URL `portal-grace.hprc.tamu.edu/node/c623/38342/lab/tree/AllLabs/Untitled.ipynb`. The interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a file browser on the left. The file browser shows a directory structure with files like `data`, `images`, `logs`, and `Untitled.ip...`. The main workspace shows a code cell with the following content:

```
[ ]: print('Hello, World!')  
Hello, World!  
[ ]:
```

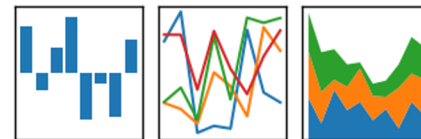
A red rectangular box highlights the `print('Hello, World!')` line, and a red arrow points from the right towards this line. The status bar at the bottom indicates the current mode is 'Command' and the cursor is at 'Ln 1, Col 1' in the 'Untitled.ipynb' file.

Lab II. Data Exploration

matplotlib 

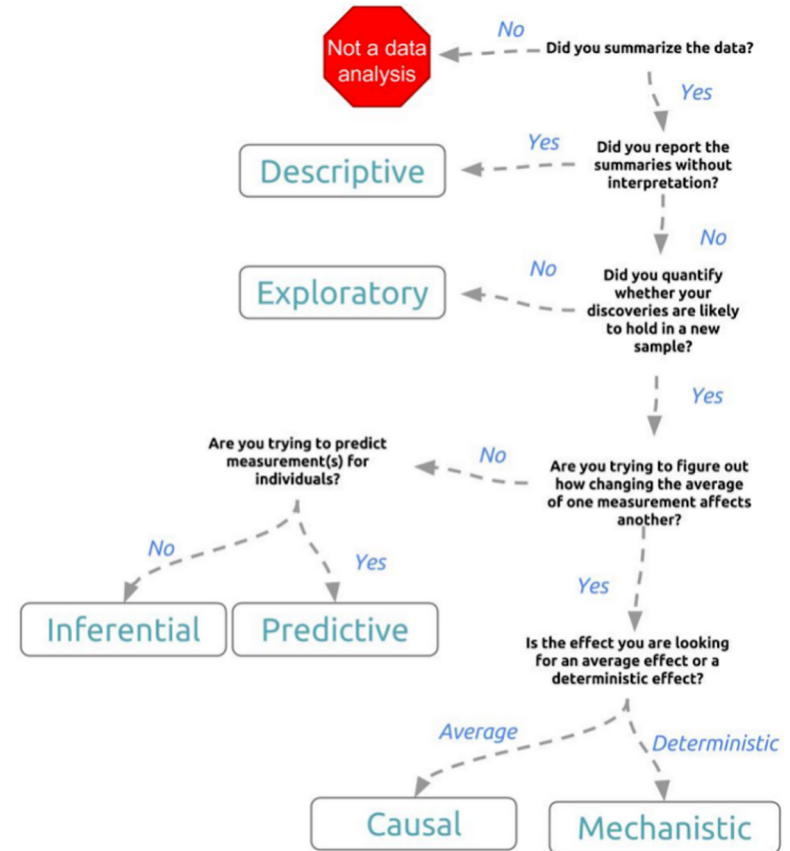
pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Types of Data Science Problems

- **Descriptive** (summaries, e.g., census)
- **Exploratory** (search for unknowns, e.g., four-planet solar system)
- **Inferential** (find correlations, e.g., many social studies)
- **Predictive** (make predictions, e.g., Face ID, Echo, Siri)
- **Causal** (explore causation, e.g., smoking versus lung cancer)
- **Mechanistic** (determine governing principles, e.g., experimental science)



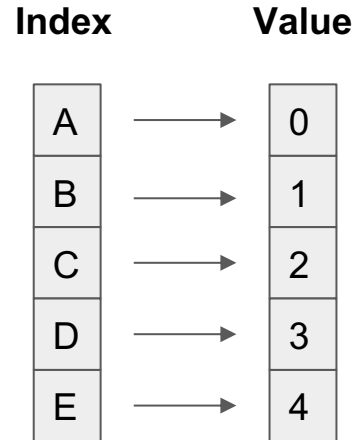
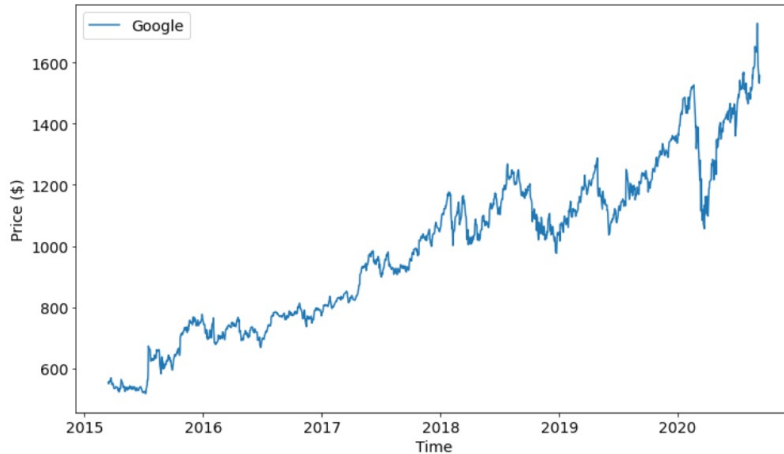
Data Structures

Pandas has two data structures that are descriptive and optimized for data with different dimensions.

- **Series:** 1D labeled array
- **DataFrame:** General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed columns

Series in pandas

- One-dimensional labeled array
- Capable of holding any data type (integers, strings, floating point numbers, etc.)
- Example: time-series stock price data



DataFrame in pandas

- Primary Pandas data structure
- A dict-like container for Series objects
- Two-dimensional size-mutable
- Heterogeneous tabular data structure

A	B	C	D	E	F	G	H
id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
7129300520	20141013T00	221900	3	1	1180	5650	1
6414100192	20141209T00	538000	3	2.25	2570	7242	2
5631500400	20150225T00	180000	2	1	770	10000	1
2487200875	20141209T00	604000	4	3	1960	5000	1
1954400510	20150218T00	510000	3	2	1680	8080	1
7237550310	20140512T00	1.23E+06	4	4.5	5420	101930	1
1321400060	20140627T00	257500	3	2.25	1715	6819	2
2008000270	20150115T00	291850	3	1.5	1060	9711	1
2414600126	20150415T00	229500	3	1	1780	7470	1

Columns

Index	C1	C2	C3	C4
A	0	x	0.1	True
B	1	y	2.4	False
C	2	z	1.9	True
D	NA	w	8.3	False
E	9	a	6.8	False

Pandas Learning Objectives

After this lesson, you will know how to:

- Create a DataFrame
- Drop Entries
- Index, Select, and Filter data
- Sort data
- Input and Output



[JupyterLab Exercises](#)

Pandas Cheat Sheet

Data Wrangling
with pandas
Cheat Sheet
<http://pandas.pydata.org>

Syntax – Creating DataFrames

a	b	c
1	4	7
2	5	8
3	6	9

```
df = pd.DataFrame(
    {"a": [4, 5, 6],
     "b": [7, 8, 9],
     "c": [10, 11, 12]},
    index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Specify values for each row.

a	b	c
1	4	7
2	5	8
3	6	9

```
df = pd.DataFrame(
    {"a": [4, 5, 6],
     "b": [7, 8, 9],
     "c": [10, 11, 12]},
    index = pd.MultiIndex.from_tuples(
        [(1, 'a'), (1, 'b'), (1, 'c')],
        names=['n', 'v']))
```

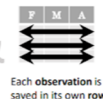
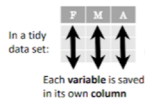
Create DataFrame with a MultiIndex

Method Chaining

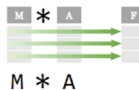
Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
     .rename(columns={
         'variable': 'var',
         'value': 'val'})
     .query('val > 200'))
```

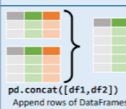
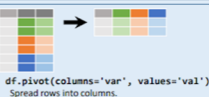
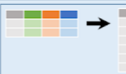
Tidy Data – A foundation for wrangling in pandas



Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.



Reshaping Data – Change the layout of a data set



```
df.sort_values('mpg')
Order rows by values of a column (low to high).

df.sort_values('mpg', ascending=False)
Order rows by values of a column (high to low).

df.rename(columns = {'y': 'year'})
Rename the columns of a DataFrame

df.sort_index()
Sort the index of a DataFrame

df.reset_index()
Reset index of DataFrame to row numbers, moving index to columns.

df.drop(columns=['Length', 'Height'])
Drop columns from DataFrame
```

Subset Observations (Rows)



```
df[df.Length > 7]
Extract rows that meet logical criteria.

df.drop_duplicates()
Remove duplicate rows (only considers columns).

df.head(n)
Select first n rows.

df.tail(n)
Select last n rows.
```

```
df.sample(frac=0.5)
Randomly select fraction of rows.

df.sample(n=10)
Randomly select n rows.

df.iloc[10:20]
Select rows by position.

df.nlargest(n, 'value')
Select and order top n entries.

df.nsmallest(n, 'value')
Select and order bottom n entries.
```

Logic in Python (and pandas)		
<	!=	Not equal to
>	df.column.isin(values)	Group membership
==	pd.isnull(obj)	is NaN
<=	pd.notnull(obj)	is not NaN
>=	df[['a', 'b']].all()	logical and, or, not, not, any, all

Subset Variables (Columns)



```
df[['width', 'length', 'species']]
Select multiple columns with specific names.

df['width'] or df.width
Select single column with specific name.

df.filter(regex='regex')
Select columns whose name matches regular expression regex.
```

regex (Regular Expressions) Examples	
"/L/"	Matches strings containing a period '.'
"/Length\$"	Matches strings ending with word 'Length'
"/^Sepal"	Matches strings beginning with the word 'Sepal'
"/^1-5\$/"	Matches strings beginning with '1' and ending with 1,2,3,4,5
"/^(?!Species)\$/"	Matches strings except the string 'Species'

```
df.loc[:, 'x2': 'x4']
Select all columns between x2 and x4 (inclusive).

df.iloc[:, [1, 2, 5]]
Select columns in positions 1, 2 and 5 (first column is 0).

df.loc[df['a'] > 10, ['a', 'c']]
Select rows meeting logical condition, and only the specific columns.
```

Summarize Data

```
df['v'].value_counts()
Count number of rows with each unique value of variable

len(df)
```

Handling Missing Data

```
df.dropna()
Drop rows with any column having NA/null data.

df.fillna(value)
```

Combine Data Sets

```
adf + bdf
a1 x2 + a1 x3 =
```


Matplotlib Learning Objectives

After this lesson, you will know how to:

- Scatter plot and Line plot
- Subplots
- Color map
- Contour figures
- 3D figures
 - Surface plots
 - Wire-frame plot
 - Contour plots with projections



[JupyterLab Exercises](#)

Matplotlib Cheat Sheet

Python For Data Science Cheat Sheet

Matplotlib

Learn Python interactively at [www.DataCamp.com](https://www.datacamp.com)

Matplotlib
Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

1 Prepare The Data

1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> Y, X = np.mgrid[:3:100, :3:100]
>>> U = 1 + X**2 + Y
>>> from matplotlib.colors import get_sample_data
>>> img = np.load(get_sample_data('axvh_grid/visdata_normal.npy'))
```

2 Create Plot

```
>>> import matplotlib.pyplot as plt
```

Figure

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

Axes

All plotting is done with respect to an Axes. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) # row=col=num
>>> ax2 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows=2, ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

3 Plotting Routines

1D Data

```
>>> lines = ax.plot(x,y)
>>> ax.scatter(x,y)
>>> axes[0,0].bar([1,2,3],[3,4,5])
>>> axes[1,0].barh([0.5,1.5],[1,1,2])
>>> axes[1,1].axhline(0.45)
>>> axes[0,1].axvline(0.45)
>>> ax.fill(x,y,color='blue')
>>> ax.fill_between(x,y,color='yellow')
```

2D Data or Images

```
>>> fig, ax = plt.subplots()
>>> im = ax.imshow(img,
                  cmap='gist_march',
                  interpolation='nearest',
                  vmin=0,
                  vmax=2)
```

Draw points with lines or markers connecting them
Draw unconnected points, scaled or colored
Plot vertical rectangles (constant width)
Plot horizontal rectangles (constant height)
Draw a horizontal line across axes
Draw a vertical line across axes
Draw filled polygons
Fill between y-values and x

Colormapped or RGB arrays

Vector Fields

```
>>> axes[0,1].arrow([0,0.5,0.5])
>>> axes[1,1].quiver(y,z)
>>> axes[2,1].streamplot(X,Y,U,V)
```

Add an arrow to the axes
Plot a 2D field of arrows
Plot 2D vector fields

Data Distributions

```
>>> ax1.hist(y)
>>> ax1.boxplot(y)
>>> ax3.violinplot(z)
```

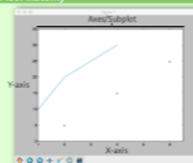
Plot a histogram
Make a box and whisker plot
Make a violin plot

Pseudocolor plot of 2D array
Pseudocolor plot of 2D array
Plot contours
Plot filled contours
Label a contour plot

```
>>> axes[0].pcolor(data2)
>>> axes[0].pcolorersh(data)
>>> C2 = plt.contourf(x,z)
>>> axes[2].contour(data1)
>>> axes[2] = ax.clabel(C2)
```

Plot Anatomy & Workflow

Plot Anatomy



Workflow

The basic steps to creating plots with matplotlib are:

- 1 Prepare data
- 2 Create plot
- 3 Plot
- 4 Customize plot
- 5 Save plot
- 6 Show plot

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,4]
>>> y = [10,20,30]
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111)
>>> ax.plot(x, y, color='lightblue', linewidth=3)
>>> ax.scatter(2,4,4,
             [15,25],
             color='darkgreen',
             marker='*')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig("foo.png")
>>> plt.show()
```

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha=0.4)
>>> ax.plot(x, y, c='k')
>>> fig.colorbar(mappable=horizontal)
```

Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker='*')
>>> ax.plot(x,y,marker='o')
```

Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,linestyle='solid')
>>> plt.plot(x,y,linestyle='-')
>>> plt.plot(x,y,linestyle='--',color='r',linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(
    -2, 1,
    "Example Graph",
    style='italic',
    style='italic')
>>> ax.annotate("time",
              xy=(8, 0),
              xycoords='data',
              xytext=(10.5, 0),
              textcoords='data',
              arrowprops=dict(arrowstyle='->',
                              connectionstyle='arc3',))
```

MathText

```
>>> plt.title(r'$\sigma_i=150$', fontsize=20)
```

Limits, Legends & Layouts

Limits & Autocasting

```
>>> ax.margins(x=0.5,y=0.1)
>>> ax.axis('equal')
>>> ax.set(xlim=[0,10],ylim=[-1.5,1.5])
>>> ax.set_xlim(0,10)
```

Legends

```
>>> ax.set(title="An Example Axes",
          ylabel="Y-Axis",
          xlabel="X-Axis")
>>> ax.legend(loc='best')
```

Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),
               ticklabels=[3,100,-12,"foo"])
>>> ax.tick_params(axis='y',
                  direction='inout',
                  length=10)
```

Subplot Spacing

```
>>> fig.subplots_adjust(wspace=0.5,
                       hspace=0.3,
                       left=0.15,
                       right=0.9,
                       top=0.9,
                       bottom=0.1)
```

```
>>> fig.tight_layout()
>>> ax.spines["top"].set_visible(False)
>>> ax.spines["bottom"].set_position(("outward",10))
```

Set the aspect ratio of the plot to 1
Set limits for x-and y-axis
Set limits for x-axis
Set a title and x-and y-axis labels
No overlapping plot elements
Manually set x-ticks
Make y-ticks longer and go in and out
Adjust the spacing between subplots
Fit subplot(s) in to the figure area
Make the top axis line for a plot invisible
Move the bottom axis line outward

5 Save Plot

```
>>> plt.savefig("foo.png")
>>> plt.savefig("foo.png", transparent=True)
```

Save figures
Save transparent figures

6 Show Plot

```
>>> plt.show()
```

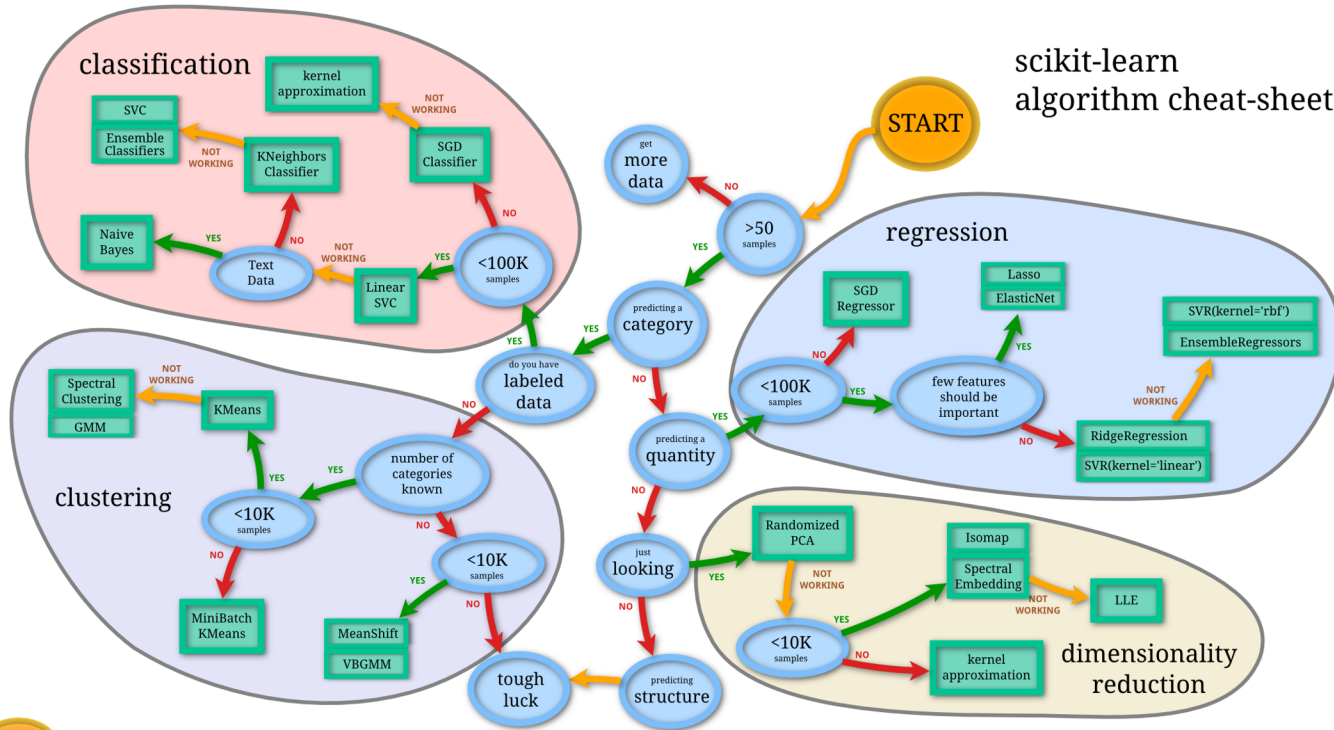
Close & Clear

```
>>> plt.cla()
>>> plt.clf()
>>> plt.close()
```

Clear an axis
Clear the entire figure
Close a window

DataCamp
Learn Python for Data Science interactively

Lab III. Machine Learning



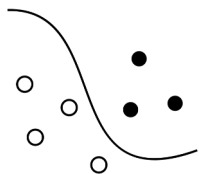
Main Features of scikit-learn



Classification

Identifying category of an object

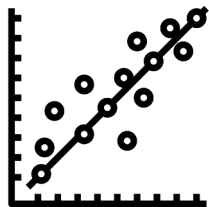
Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a attribute for an object

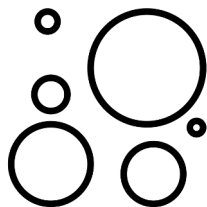
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Clustering

Grouping similar objects into sets

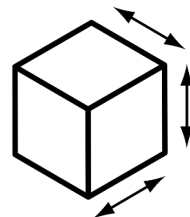
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...



Dimension Reduction

Reducing the number of dimensions

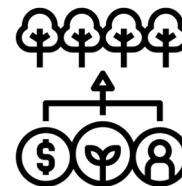
Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



Model Selection

Selecting models with parameter search

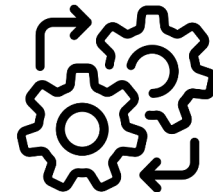
Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...



Preprocessing

Preprocessing data to prepare for modeling

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



Lab IV. Deep Learning

Deep Learning

by Ian Goodfellow, Yoshua Bengio, and Aaron Courville

<http://www.deeplearningbook.org/>

Animation of Neutron Networks

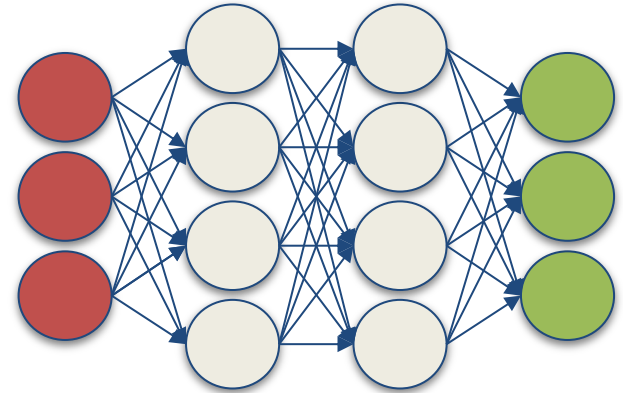
by Grant Sanderson

<https://www.3blue1brown.com/>

Visualization of CNN

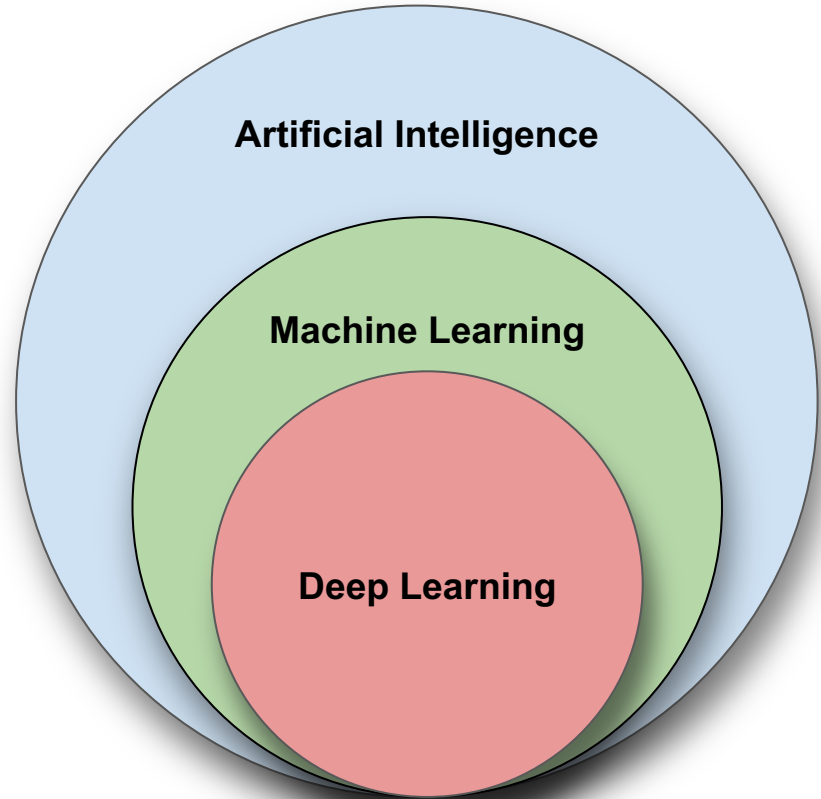
by Adam Harley

<https://www.cs.ryerson.ca/~aharley/vis/conv/>



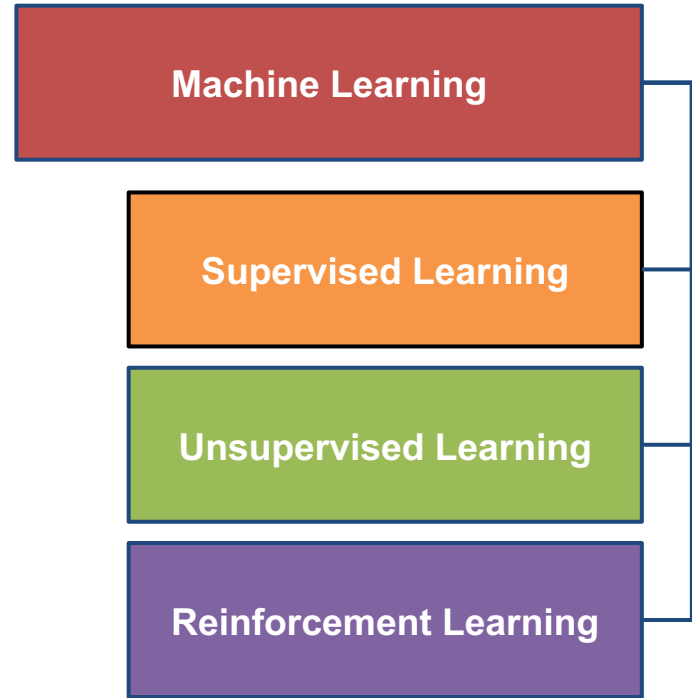
Relationship of AI, ML, and DL

- **Artificial Intelligence (AI)** is anything about man-made intelligence exhibited by machines.
- **Machine Learning (ML)** is an approach to achieve **AI**.
- **Deep Learning (DL)** is one technique to implement **ML**.

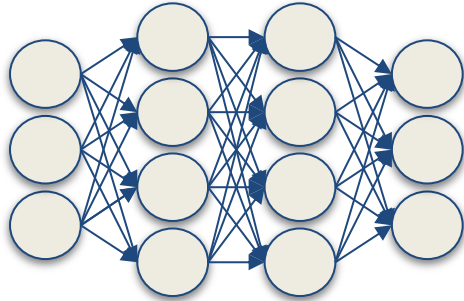
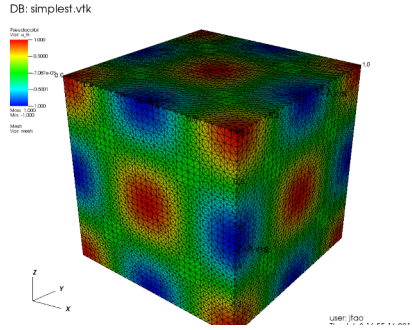


Types of ML Algorithms

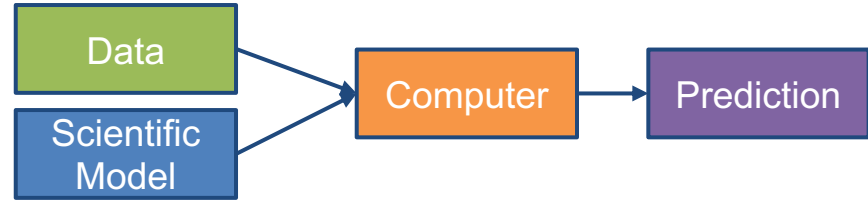
- **Supervised Learning**
 - trained with labeled data; including regression and classification problems
- **Unsupervised Learning**
 - trained with unlabeled data; clustering and association rule learning problems.
- **Reinforcement Learning**
 - no training data; stochastic Markov decision process; robotics and self-driving cars.



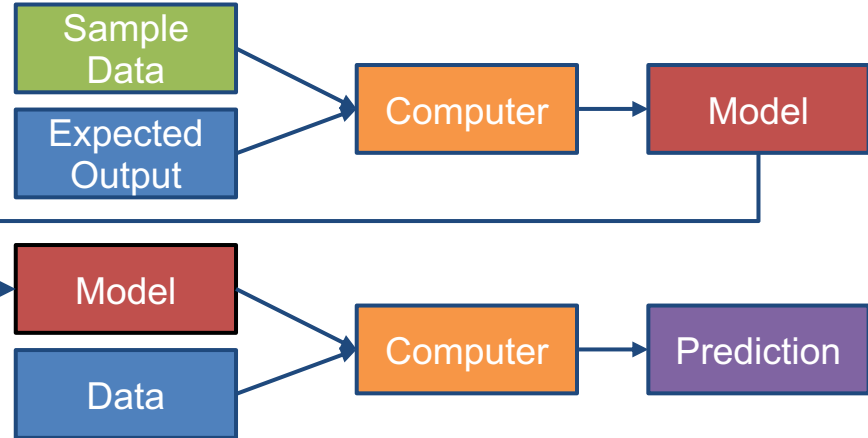
Machine Learning



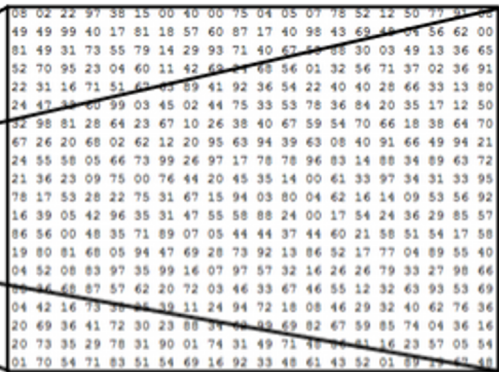
Traditional Modeling



Machine Learning (Supervised Learning)



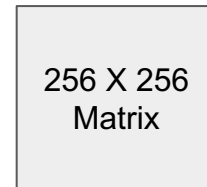
Inputs and Outputs



What the computer sees

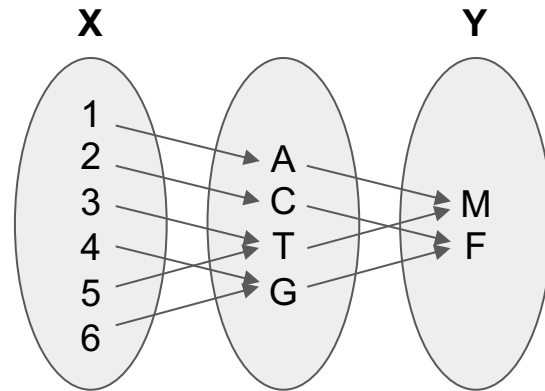
image classification → 82% cat
15% dog
2% hat
1% mug

Image from the [Stanford CS231 Course](#)



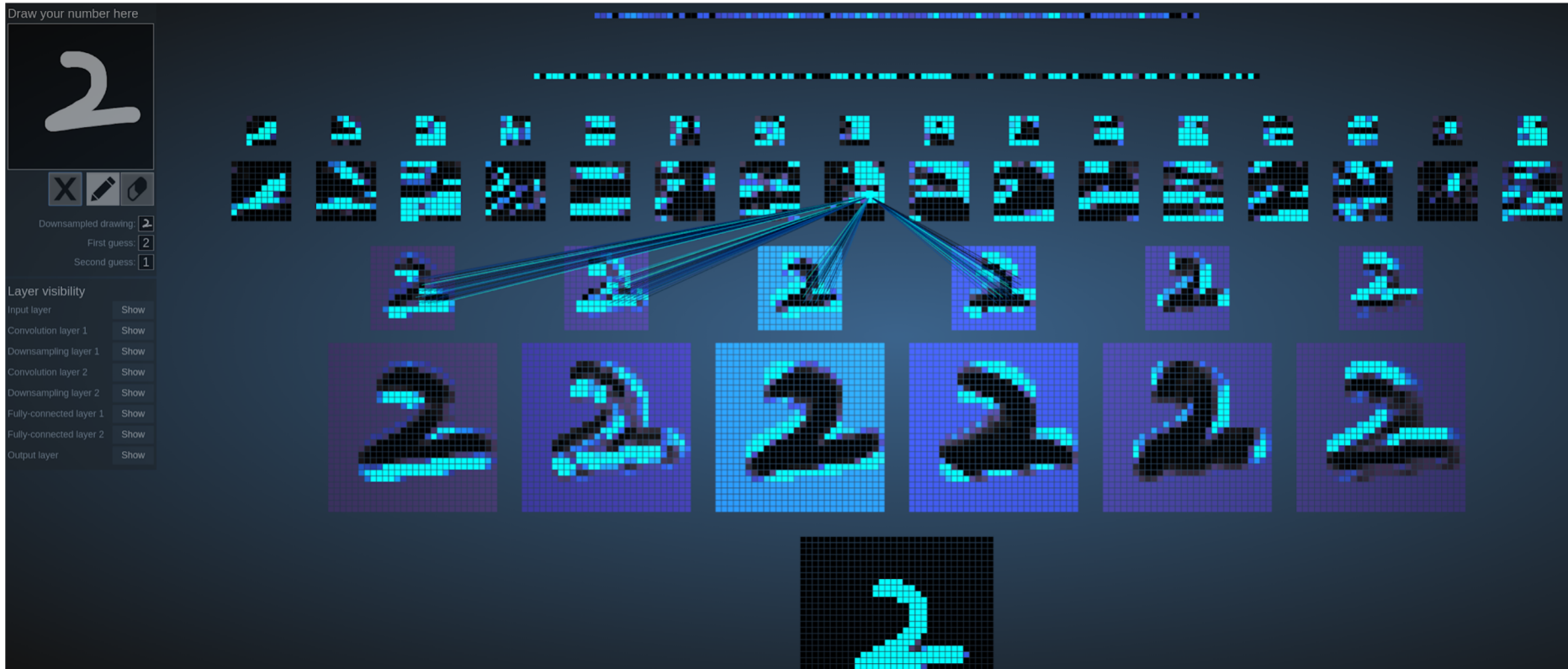
DL model

4-Element Vector



With deep learning, we are searching for a **surjective** (or **onto**) function f from a set X to a set Y .

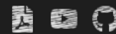
MNIST - CNN Visualization



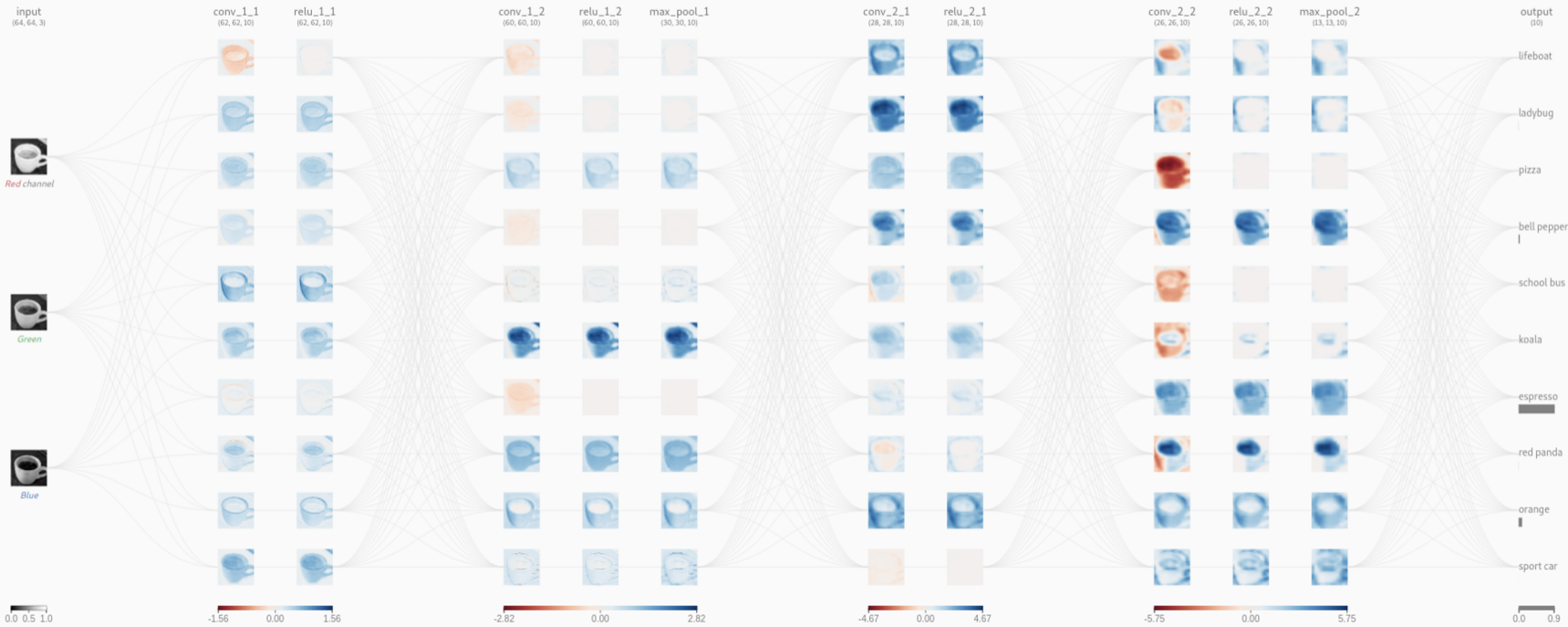
(Image Credit: <http://scs.ryerson.ca/~aharley/vis/>)

CNN Explainer

CNN EXPLAINER Learn Convolutional Neural Network (CNN) in your browser!

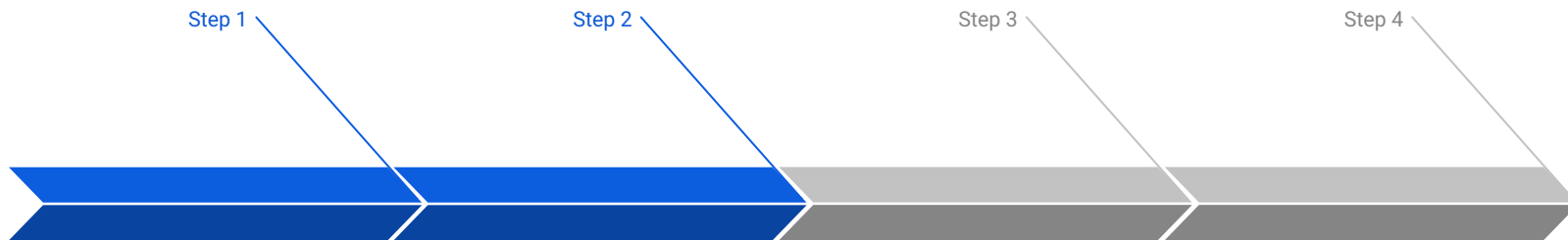


Show detail Unit



(Image Credit: <https://poloclub.github.io/cnn-explainer/>)

Machine Learning Workflow with Keras



Prepare Train Data

The preprocessed data set needs to be shuffled and splitted into training and testing data.

Define Model

A model could be defined with Keras Sequential model for a linear stack of layers or Keras functional API for complex network.

Training Configuration

The configuration of the training process requires the specification of an optimizer, a loss function, and a list of metrics.

Train Model

The training begins by calling the fit function. The number of epochs and batch size need to be set. The measurement metrics need to be evaluated.