

# AI Tech Labs 0⇒1

Jian Tao

jtao@tamu.edu

HPRC Short Course

10/30/2020



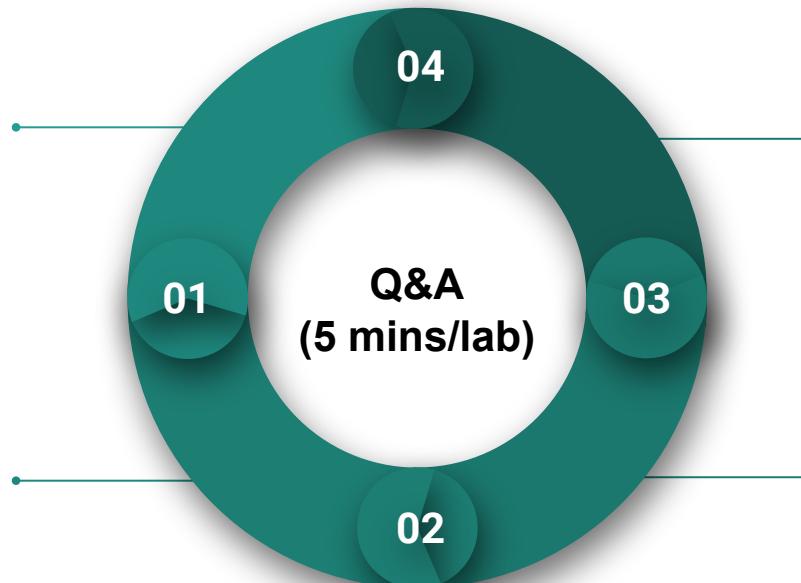
# AI Tech Labs

## Lab I. JupyterLab (15 mins)

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

## Lab II. Data Exploration (30 mins)

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



## Lab IV. Deep Learning (30 minutes)

We will learn how to use Keras to create 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 for linear regression and classification applications.

# Lab I. JupyterLab



Screenshot of JupyterLab interface showing a notebook titled "Lorenz.ipynb".

The interface includes:

- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Left Sidebar:** Files, Running, Commands, Cell Tools, Tabs.
- Central Area:** Notebook tabs (Lorenz.ipynb, Terminal 1, Console 1, Data.ipynb, README.md), a text editor, and an Output View.
- Output View:** Displays a Lorenz attractor plot generated by the code in the notebook.
- Code Editor:** Shows Python code for solving the Lorenz equations and plotting the attractor.

Code in the code editor:

```
from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)

def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=.3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random((N, 3))
```

# L1 - Resources

- [Texas A&M High Performance Research Computing \(HPRC\)](#)
- [Ada Quick Start Guide](#)
- [HPRC Portal](#)
- [HPRC YouTube Channel](#)
- [Jupyter Project](#)

# L1 - Login HPRC Portal

TAMU HPRC OnDemand Port. X +

portal.hprc.tamu.edu

High Performance Research Computing  
*A Resource for Research and Discovery*

TAMU HPRC OnDemand Homepage



[Ada OnDemand Portal](#)

[OnDemand Portal](#)

[OnDemand Portal User Guide](#)

A red arrow points from the "OnDemand Portal" link to the "Ada OnDemand Portal" link.

# L1 - Shell Access - I

The screenshot shows a web browser window for the TAMU HPRC OnDemand Ada dashboard at [portal-ada.hprc.tamu.edu/pun/sys/dashboard](https://portal-ada.hprc.tamu.edu/pun/sys/dashboard). The interface includes a top navigation bar with links for Dashboard, Files, Jobs, Clusters (which is currently selected), Interactive Apps, and other system icons. A dropdown menu is open under the Clusters button, listing three options: >\_ Ada Shell Access, >\_ Terra Shell Access, and >\_ Curie Shell Access. The first option, >\_ Ada Shell Access, is highlighted with a red box and a red arrow pointing to it from the right side of the image.

TAMU HPRC OnDemand (Ada)    Files ▾    Jobs ▾    Clusters ▾    Interactive Apps ▾

>\_ Ada Shell Access  
>\_ Terra Shell Access  
>\_ Curie Shell Access

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

**Message of the Day**

**\*\* Ada Cluster Maintenance, September 29 \*\***

The Ada cluster will be unavailable from 9am to 6pm on Tuesday, September 29th. Software and hardware maintenance will be performed during this downtime. Jobs will not be scheduled if they will overlap with this maintenance window.

**IMPORTANT POLICY INFORMATION**

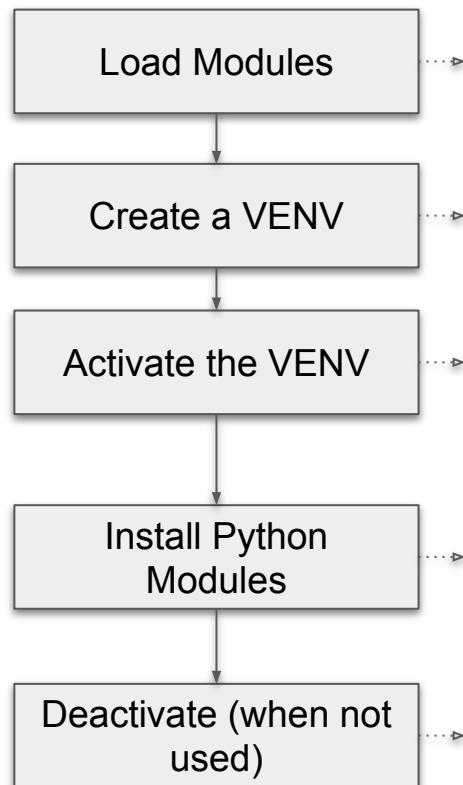
<https://portal-ada.hprc.tamu.edu/pun/sys/dashboard>

# L1 - Shell Access - II

A screenshot of a terminal window titled "jtao@login8:~". The window is connected to the URL "portal-ada.hprc.tamu.edu/pun/sys/shell/ssh/ada.tamu.edu". The terminal displays a security notice and a password prompt:

```
*****  
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 administrative/disciplinary actions or criminal prosecution against  
the user. Usage may be subject to security testing and monitoring to ensure  
compliance with the policies of Texas A&M University, Texas A&M University  
System, and the State of Texas. There is no expectation of privacy on this  
system except as otherwise provided by applicable privacy laws. Users should  
refer to Texas A&M University Standard Administrative Procedure 29.01.03.M0.02,  
Rules for Responsible Computing, for guidance on the appropriate use of Texas  
A&M University information resources.  
*****  
Password:  
Duo two-factor login for jtao  
Enter a passcode or select one of the following options:  
1. Duo Push to iPhone (iOS)  
2. Duo Push to iPad (iOS)  
Passcode or option (1-2): 1  
Success. Logging you in...  
Last login: Fri May 1 22:10:51 2020 from connect-172-31-38-197.vpn.tamu.edu  
=====| Texas A&M University High Performance Research Computing |  
| Website: https://hprc.tamu.edu |  
| Consulting: help@hprc.tamu.edu (preferred) or (979) 845-0219 |  
| Ada Documentation: https://hprc.tamu.edu/wiki/Ada |  
| Curie Documentation: https://hprc.tamu.edu/wiki/Curie |
```

# L1 - Python Virtual Environment (VENV)



```
# clean up and load Anaconda
cd $SCRATCH
module purge
module load Anaconda/3-5.0.0.1

# create a Python virtual environment
conda create -n mylab

# activate the virtual environment
source activate mylab

# install required package to be used in the portal
conda install jupyterlab=1.2.2
conda install pandas matplotlib
conda install scikit-learn
conda install tensorflow

# deactivate the virtual environment
# deactivate
```

# L1 - Common Anaconda Commands

```
# Conda virtual environment
conda info                                # show Conda installation
conda create -n VENV                      # create a virtual environment
conda create -n VENV python=3.4            # create a venv with a py version
conda env list                            # list installed venv

# Conda package management
conda list                                 # list all installed packages
conda search  PACKAGENAME                 # search a Conda package
conda install  PACKAGENAME                # install a Conda package
conda update   PACKAGENAME                # update a Conda package
conda remove   PACKAGENAME                # remove a Conda package

# install required package to be used in the portal
conda install jupyterlab=1.2.2
conda install pandas matplotlib
conda install scikit-learn
conda install tensorflow
```

# L1 - Check out Exercises

The screenshot shows a GitHub repository page for 'jtao/ailabs'. The repository has 1 branch and 0 tags. A 'Clone' modal is open, displaying the HTTPS URL: <https://github.com/jtao/ailabs.git>. The modal also includes options to 'Download ZIP'.

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

**Lab I. JupyterLab (15 mins)**  
We will set up a Python virtual environment and run JupyterLab on the HPRC Portal.

**04**

**Lab IV. Deep Learning (30 minutes)**  
We will learn how to use Keras to create and train a simple image classification.

Publish your first package

# L1 - Go to JupyterLab Page

The screenshot shows the TAMU HPRC OnDemand Ada dashboard. The top navigation bar includes links for TAMU HPRC OnDemand (Ada), Files, Jobs, Clusters, Interactive Apps, a user profile for jtao, and Log Out. The main content area features a large image of server racks and a message: "OnDemand provides an integrated, single access point to all cluster resources". Below this is a "Message of the Day" section about Ada Cluster Maintenance on September 29, stating that the cluster will be unavailable from 9am to 6pm. A red box highlights the "JupyterLab" option under the "Interactive Apps" menu, which is preceded by a red arrow pointing to it.

TAMU HPRC OnDemand (Ada)    Files ▾    Jobs ▾    Clusters ▾    Interactive Apps ▾    jtao    Log Out

OnDemand provides an integrated, single access point to all cluster resources.

**Message of the Day**

\*\* Ada Cluster Maintenance, September 29 \*\*

The Ada cluster will be unavailable from 9am to 6pm on Tuesday, September 29. There will be no downtime. Jobs will not be scheduled if they will overlap with this maintenance.

**IMPORTANT POLICY INFORMATION**

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

BIO

- IGV
- Mauve
- Structure

GUI

- ANSYS Workbench
- Abaqus/CAE
- LS-PREPOST
- MATLAB
- ParaView
- VNC

Galaxy

- Galaxy (maroon)
- Galaxy (reveille)

Servers

- Jupyter Notebook
- JupyterLab** (highlighted with a red box and a red arrow pointing to it)
- RStudio Server with R 3.4.3 (Singularity)
- DStudio Server with D 2.6.1 (Singularity)

# L1 - Set Virtual Environment

The screenshot shows a web browser window for 'JupyterLab' on the 'TAMU HPRC OnDemand (Ada)' cluster. The URL is [https://portal-ada.hprc.tamu.edu/pun/sys/dashboard/batch\\_connect/sys/jupyterlab/session\\_contexts/new](https://portal-ada.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sys/jupyterlab/session_contexts/new). The page title is 'JupyterLab'. The left sidebar lists various interactive applications: BIO, IGV, Mauve, Structure, GUI, ANSYS Workbench, Abaqus/CAE, LS-PREPOST, MATLAB, ParaView, VNC, and Servers. The main content area is titled 'JupyterLab' and describes launching a JupyterLab server on the Ada cluster. It shows a dropdown menu set to 'Anaconda/3-5.0.0.1' and a text input field for 'JupyterLab Environment to be activated' containing 'mylab'. A large red arrow points to the 'mylab' input field.

JupyterLab

This app will launch a JupyterLab server on the Ada cluster.

**Module**

Anaconda/3-5.0.0.1

Anaconda/3- is Python3

**JupyterLab Environment to be activated**

mylab

Enter the name of environment to be activated. Changing this field is optional.

Use the default jupyterlab\_v1.2.2 unless you have installed your own JupyterLab conda Environment.

Your optional conda environment must have been previously built with one of the Anaconda modules listed in the Module option above. See instructions.

# L1 - Connect to JupyterLab

The screenshot shows a web browser window for the TAMU HPRC OnDemand (Ada) system. The URL in the address bar is `portal-ada.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sessions`. The page title is "My Interactive Sessions". The main content area displays a success message: "Session was successfully created." Below this, the navigation bar includes links for Home, My Interactive Sessions, Files, Jobs, Clusters, Interactive Apps, and user information (jtao, Log Out). A sidebar on the left lists various interactive applications: BIO, IGV, Mauve, Structure, GUI, ANSYS Workbench, and Abaqus/CAE. The central panel shows a "JupyterLab (12695677)" session card. The card details include: Host: nxt1735, Created at: 2020-09-19 19:10:05 CDT, Time Used: 5 minutes, and Session ID: e18d47a1-3d4f-4f15-b46d-bff5fa09174a. A red arrow points to a blue button labeled "Connect to JupyterLab" which is highlighted with a red border.

Session was successfully created.

TAMU HPRC OnDemand (Ada)

Home / My Interactive Sessions

Interactive Apps

- BIO
- IGV
- Mauve
- Structure
- GUI
- ANSYS Workbench
- Abaqus/CAE

JupyterLab (12695677)

1 node | 1 core | Running

Host: nxt1735

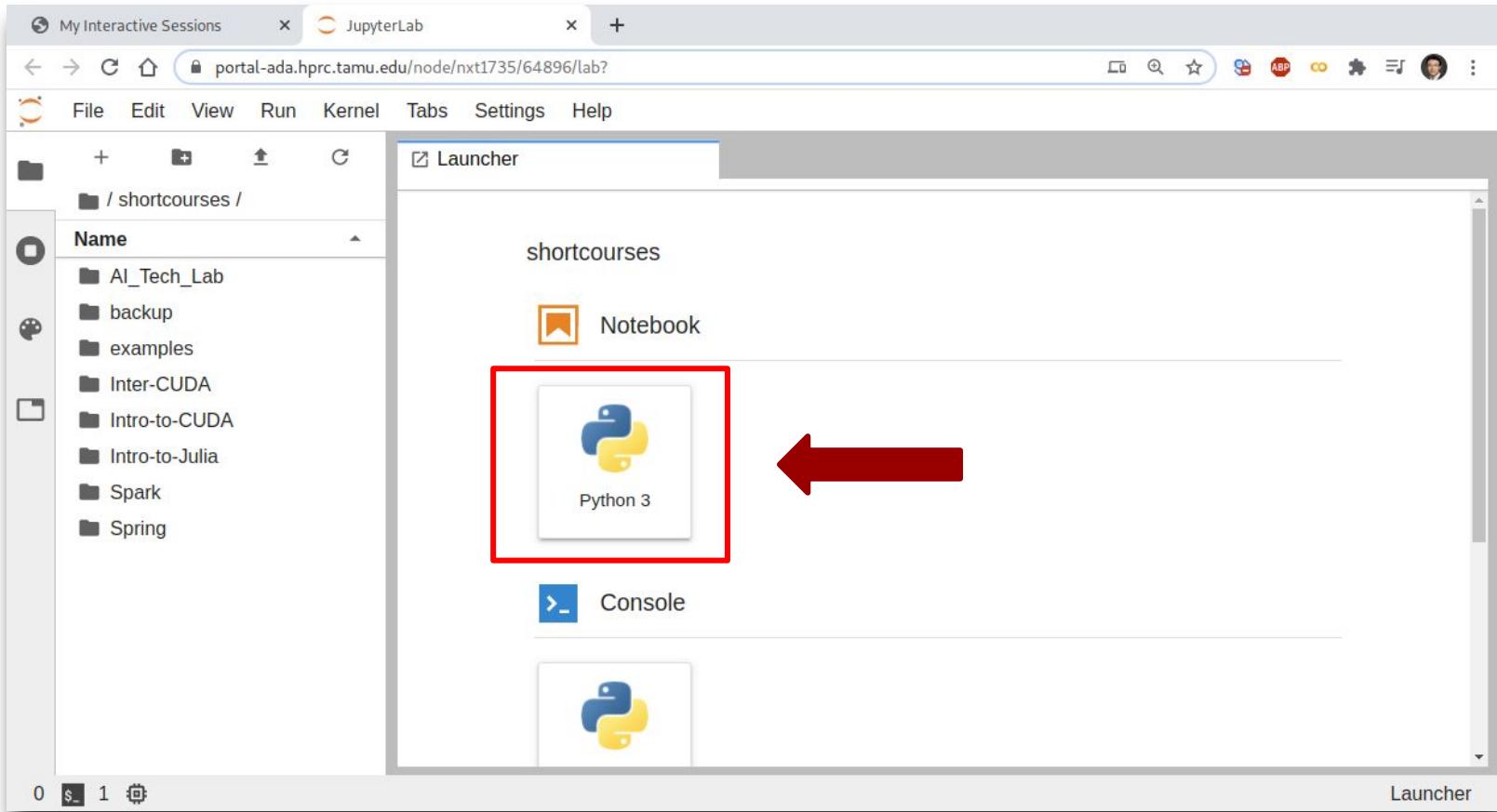
Created at: 2020-09-19 19:10:05 CDT

Time Used: 5 minutes

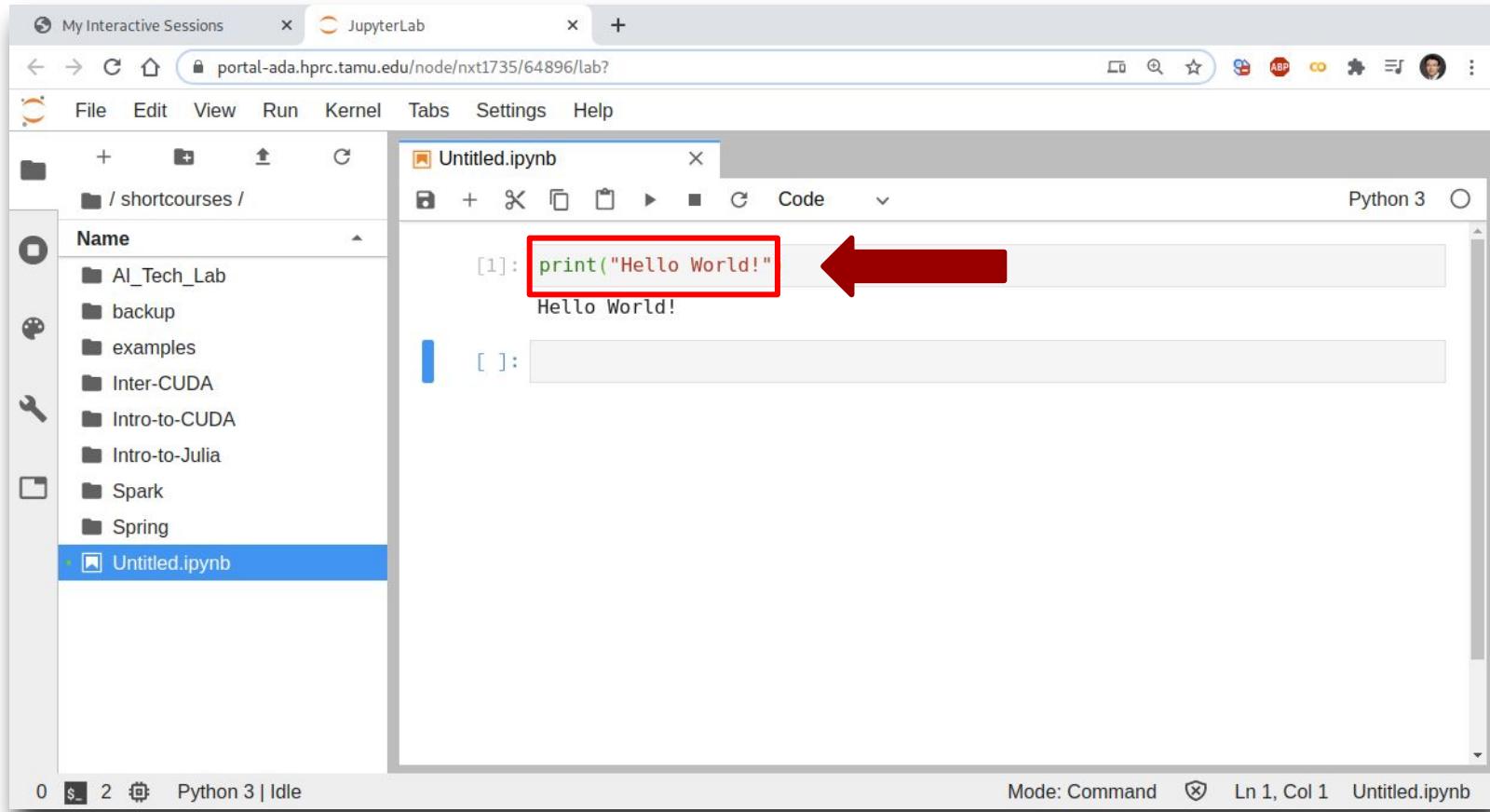
Session ID: e18d47a1-3d4f-4f15-b46d-bff5fa09174a

Connect to JupyterLab

# L1 - Create a Jupyter Notebook



# L1 - Test JupyterLab

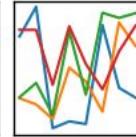


# Lab II. Data Exploration



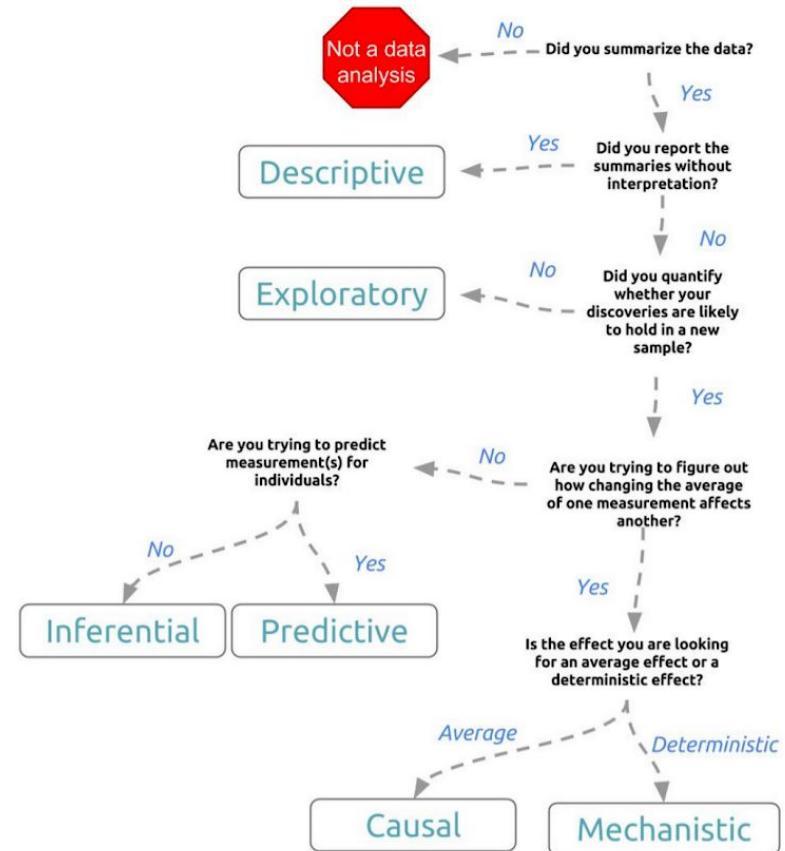
pandas

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



# Types of Data Science Problems

- **Descriptive** (summaries, e.g., census)
- **Exploratory** (search for unknowns, e.g., SETI@home, Einstein@home)
- **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)



Credit: Jeff Leek - The Elements of Data Analytic Style

# 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** Also see Lists & NumPy

**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))
>>> data[3, 3] = np.random.random((10, 10))
>>> x = np.linspace(-3.1415926, -3.1415926)
>>> U = -1 + x**2 / 4
>>> V = 1 - x - y**2
>>> from matplotlib import image
>>> img = np.load('data/mnist/mnist.npz')
>>> img = np.load('data/elec_grid/Electricity_normal.npz')
```

**2 Create Plot**

**Figure**

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=fig.get_size_inches() * 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_subplot()
>>> fig.add_subplot(221) # row-col-num
>>> ax3 = 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)
>>> ax.plot([1,2,3],[3,4,5])
>>> ax.polygon([[0,0,1,1,0],[0,1,1,0,0]])
>>> ax.patches([0,0,1,1,0,0,1,1])
>>> ax.patches([0,0,1,1,0,0,1,1])
>>> ax.patches([0,0,1,1,0,0,1,1])
>>> ax.fill_between(x,y,color='yellow')
```

**2D Data or Images**

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

**Colormapped or RGB arrays**

**Plot Anatomy & Workflow**

**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 = [1,2,3,4]
>>> y = [1,4,9,16]
>>> fig = plt.figure() # Step 1
>>> ax = fig.add_subplot(111) # Step 2
>>> ax.plot(x, y, color='lightblue', linewidth=3) # Step 3
>>> ax.scatter([1,2,3], [1,4,9], color='darkgreen', marker='*') # Step 4
>>> ax.set_xlim(1, 6.5) # Step 5
>>> plt.savefig('foo.png') # Step 6
>>> plt.show()
```

**4 Customize Plot**

**Colors, Color Bars & Color Maps**

```
>>> plt.plot(x, y, x**2, x**3)
>>> ax.set_color_cycle(['red','blue'])
>>> ax.set_color_cycle(['red','blue'])
>>> im = ax.imshow(img, cmap='seismic')
```

**Markers**

```
>>> fig, ax = plt.subplots()
>>> ax.plot(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='dashed')
>>> plt.plot(x,y,'--',x**2,y**2,'-')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

**Text & Annotations**

```
>>> ax.text(2,1,
>>>         'Example Graph',
>>>         style='italic')
>>> ax.annotate('Size',
>>>             xy=(1,1),
>>>             xycoords='data',
>>>             xytext=(10.5, 8),
>>>             textcoords='offset pixels',
>>>             arrowprops=dict(arrowsize=2,
>>>                           connectionstyle="arc3"))
```

**Vector Fields**

```
>>> ax.cla()
>>> ax.arrow(0,0,0.5,0.5)
>>> ax.cla()
>>> ax.quiver(x,y,U,V)
>>> ax.cla()
>>> streamplot(X,Y,U,V)
```

**Data Distributions**

```
>>> ax.hist(y)
>>> ax.hist(y, bins=10)
>>> ax.violinplot(z)
```

**5 Save Plot**

**Save Figures**

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

**6 Show Plot**

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

**Close & Clear**

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

**DataCamp**  
Learn Python for Data Science interactively

[https://s3.amazonaws.com/assets.datacamp.com/blog\\_assets/Python\\_Matplotlib\\_Cheat\\_Sheet.pdf](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Matplotlib_Cheat_Sheet.pdf)

# Key Plotting Concepts in Matplotlib

- **Matplotlib: Figure**

Figure is the object that keeps the whole image output.

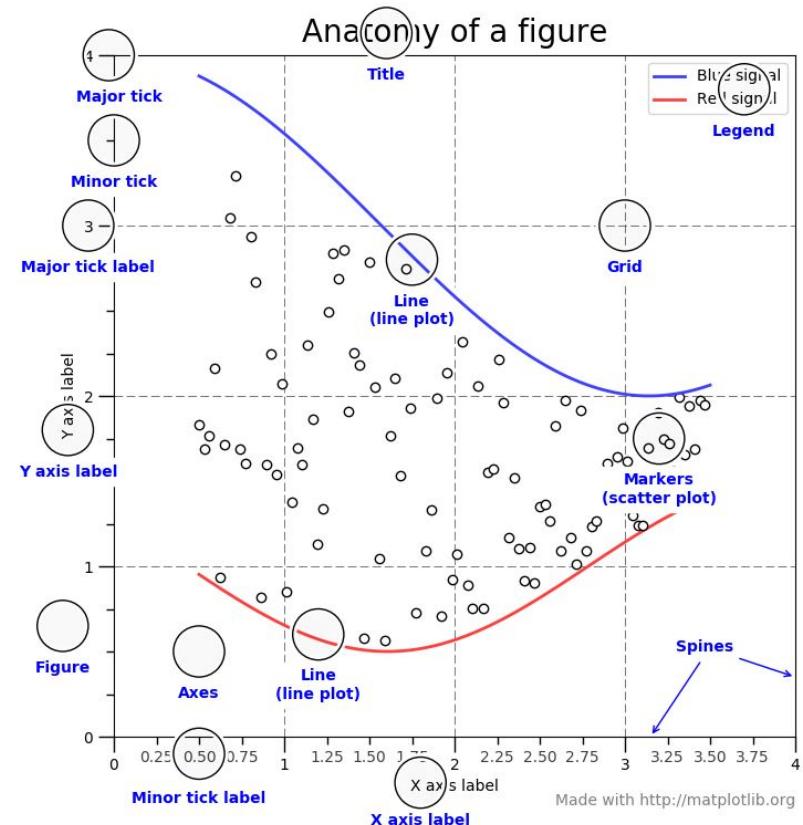
Adjustable parameters include:

1. Image size (`set_size_inches()`)
2. Whether to use `tight_layout()` (`set_tight_layout()`)

- **Matplotlib: Axes**

Axes object represents the pair of axis that contain a single plot (x-axis and y-axis). The Axes object also has more adjustable parameters:

1. The plot frame (`set_frame_on()` or `set_frame_off()`)
2. X-axis and Y-axis limits (`set_xlim()` and `set_ylim()`)
3. X-axis and Y-axis Labels (`set_xlabel()` and `set_ylabel()`)
4. The plot title (`set_title()`)



(Credit: [matplotlib.org](http://matplotlib.org))

# Data Structures

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

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

# Series in pandas

"Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the index." - [pandas site](#)

```
In [3]: s = pd.Series(np.random.randn(5),  
                   index=['a', 'b', 'c', 'd', 'e'])
```

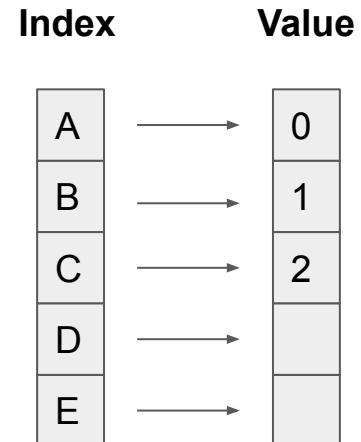
```
In [5]: s.index
```

```
In [6]: pd.Series(np.random.randn(5))
```

```
In [7]: d = {'b': 1, 'a': 0, 'c': 2}
```

```
In [8]: pd.Series(d)
```

```
In [12]: pd.Series(5., index=['a', 'b', 'c', 'd', 'e'])
```



# DataFrame in pandas

"Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure." - [pandas site](#)

```
In [2]: d = {'col1': [1, 2], 'col2': [3, 4]}
```

```
In [3]: df = pd.DataFrame(data=d)
```

```
In [5]: df.index
```

```
In [6]: df = pd.DataFrame(  
    np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]),  
    columns=['a', 'b', 'c'])
```

Index	Columns			
	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 Cheat Sheet

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

### Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	9	9	12

```
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
a	4	7	10
b	5	8	11
c	9	9	12

```
df = pd.DataFrame(  
    {"a": [4, 5, 6],  
     "b": [7, 8, 9],  
     "c": [10, 11, 12]},  
    index = pd.MultiIndex.from_tuples(  
        [('d',1),('d',2),('e',2)],  
        names=['n','v']))
```

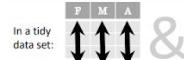
Create DataFrame with a MultiIndex.

### Method Chaining

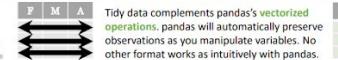
Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

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

### Tidy Data – A foundation for wrangling in pandas



In a tidy data set:



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



M \* A

### Reshaping Data – Change the layout of a data set



pd.melt(df)

Gather columns into rows.



df.pivot(columns='var', values='val')

Spread rows into columns.



pd.concat([df1, df2])

Append rows of DataFrames



pd.concat([df1, df2], axis=1)

Append columns of DataFrames

```
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 > 10]
```

Extract rows that meet logical criteria.

```
df.duplicated()
```

Remove duplicate rows (only considers columns).

```
df.head(n)
```

Select first n rows.

```
df.tail(n)
```

Select last n rows.

### 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

'\.' Matches strings containing a period.'

'Length\$' Matches strings ending with 'Length'

'Sepal' Matches strings beginning with the word 'Sepal'

'x|1|2|3\$' Matches strings beginning with 'x' 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[:, 'a'] > 10, ['a', 'c']
```

Select rows meeting logical condition, and only the specific columns.

### Summarize Data

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

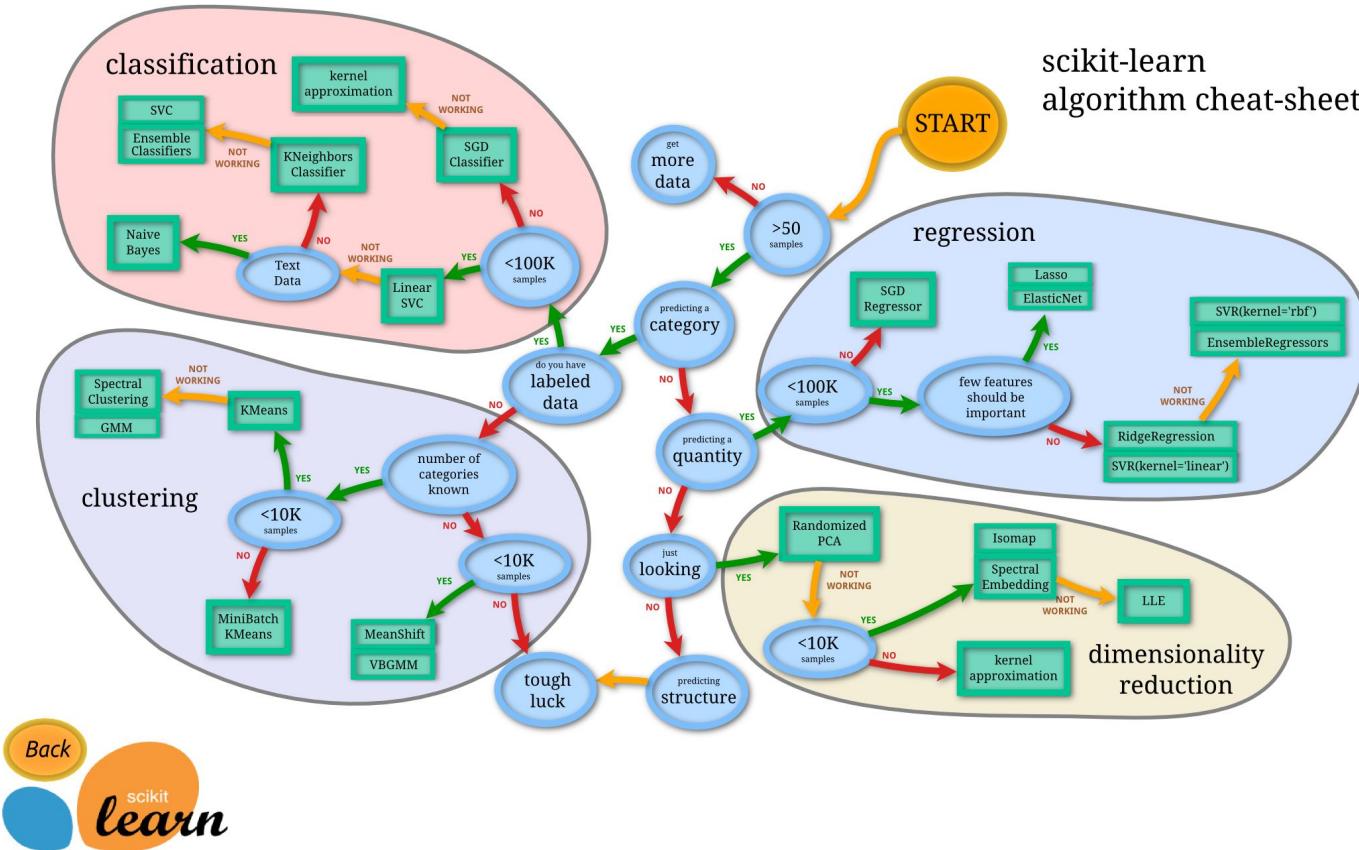
### Handling Missing Data

```
df.dropna()  
Drop rows with any column having Na/null data.  
df.fillna(value)
```

### Combine Data Sets

```
adf + bdf  
adf x2 bdf  
A 1 + B 1  
A T + B T  
A 1 + B 1 = C 1
```

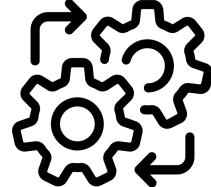
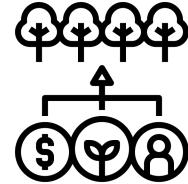
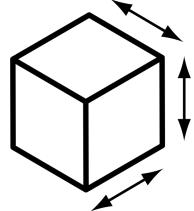
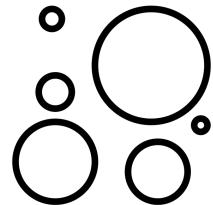
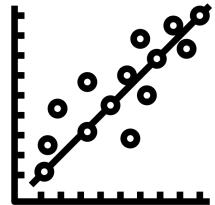
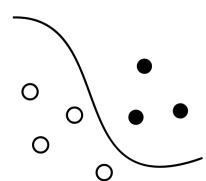
# Lab III. Machine Learning



# Main Features of scikit-learn



Classification	Regression	Clustering	Dimension Reduction	Model Selection	Preprocessing
<p><b>Identifying category of an object</b></p> <p><b>Applications:</b> Spam detection, image recognition.</p> <p><b>Algorithms:</b> SVM, nearest neighbors, random forest, and more...</p>	<p><b>Predicting a attribute for an object</b></p> <p><b>Applications:</b> Drug response, Stock prices.</p> <p><b>Algorithms:</b> SVR, nearest neighbors, random forest, and more...</p>	<p><b>Grouping similar objects into sets</b></p> <p><b>Applications:</b> Customer segmentation, Grouping experiment outcomes</p> <p><b>Algorithms:</b> k-Means, spectral clustering, mean-shift, and more...</p>	<p><b>Reducing the number of dimensions</b></p> <p><b>Applications:</b> Visualization, Increased efficiency</p> <p><b>Algorithms:</b> k-Means, feature selection, non-negative matrix factorization, and more...</p>	<p><b>Selecting models with parameter search</b></p> <p><b>Applications:</b> Improved accuracy via parameter tuning</p> <p><b>Algorithms:</b> grid search, cross validation, metrics, and more...</p>	<p><b>Preprocessing data to prepare for modeling</b></p> <p><b>Applications:</b> Transforming input data such as text for use with machine learning algorithms.</p> <p><b>Algorithms:</b> preprocessing, feature extraction, and more...</p>



# Lab IV. Deep Learning

## *Deep Learning*

by Ian Goodfellow, Yoshua Bengio, and Aaron Courville

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

## *Animation of Neuron Networks*

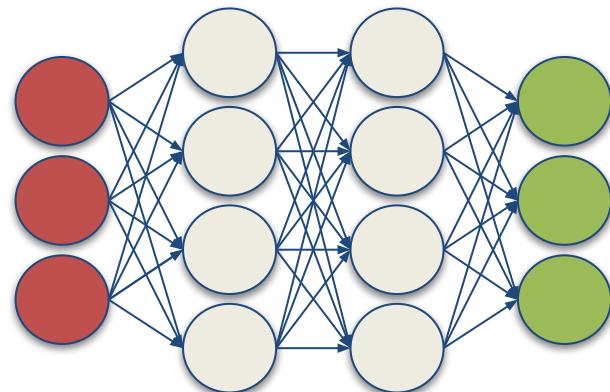
by Grant Sanderson

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

## *Visualization of CNN*

by Adam Harley

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



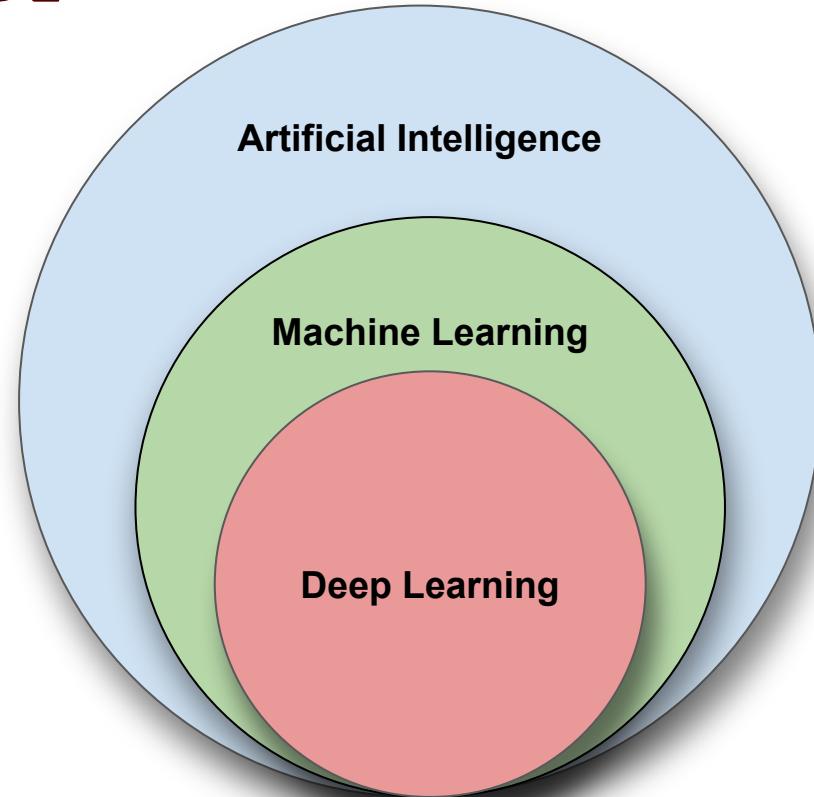
TensorFlow  
2.0



Keras

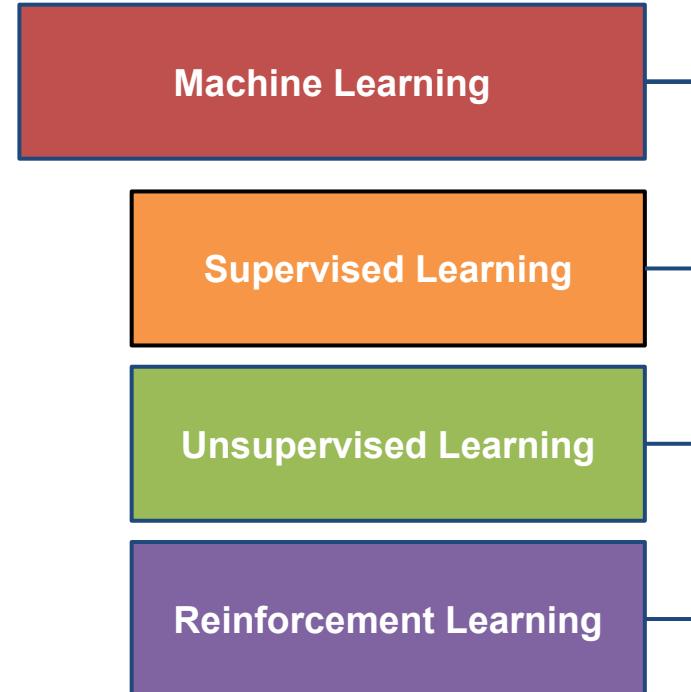
# Relationship of AI, ML, and DL

- **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**.

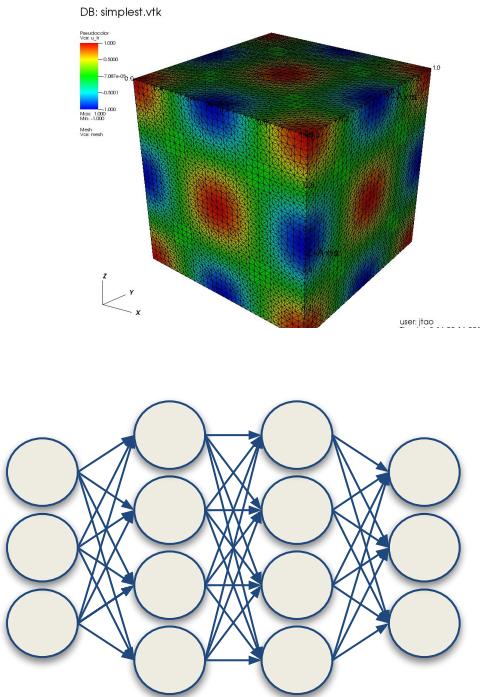


# Types of ML Algorithms

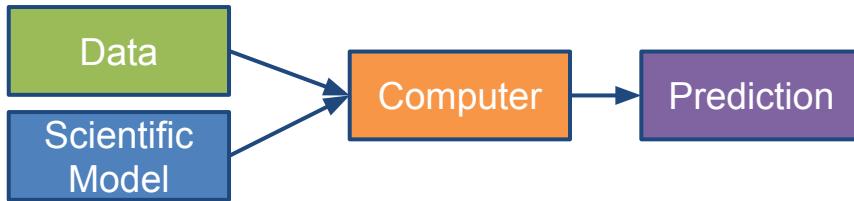
- **Supervised Learning**
  - trained with labeled data; including regression and classification problems
- **Unsupervised Learning**
  - trained with unlabeled data; clustering and association rule learning problems.
- **Reinforcement Learning**
  - no training data; stochastic Markov decision process;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



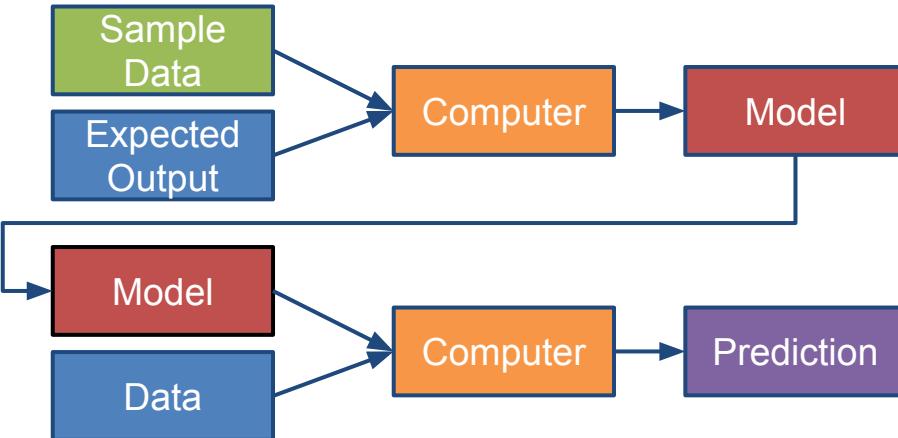
# Machine Learning



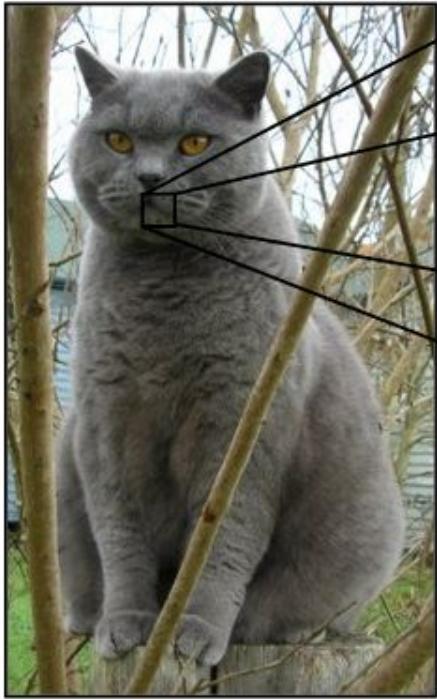
## Traditional Modeling



## Machine Learning (Supervised Learning)



# Inputs and Outputs

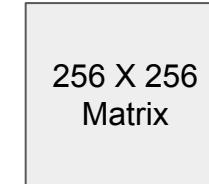


05	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	00	
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	53	58	30	08	49	13	36	65
52	70	95	23	04	60	11	42	69	51	68	56	01	32	56	71	37	02	36	91
22	31	16	71	51	63	03	69	41	92	36	54	22	40	40	28	66	33	13	80
24	47	34	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
52	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	50	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	32	08	03	97	35	99	16	07	97	57	32	16	26	26	79	35	27	98	66
59	34	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	55	36	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	37	03	69	82	67	59	85	74	04	36	16	
20	73	35	29	78	31	90	01	74	31	49	71	49	55	51	16	23	57	05	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	19	67	48

What the computer sees

image classification →

82% cat  
15% dog  
2% hat  
1% mug



DL model

4-Element Vector

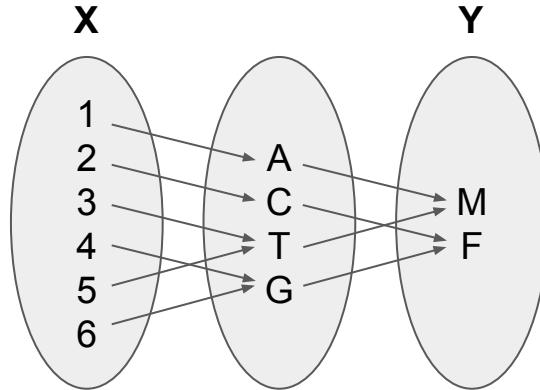


Image from the [Stanford CS231 Course](#)

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

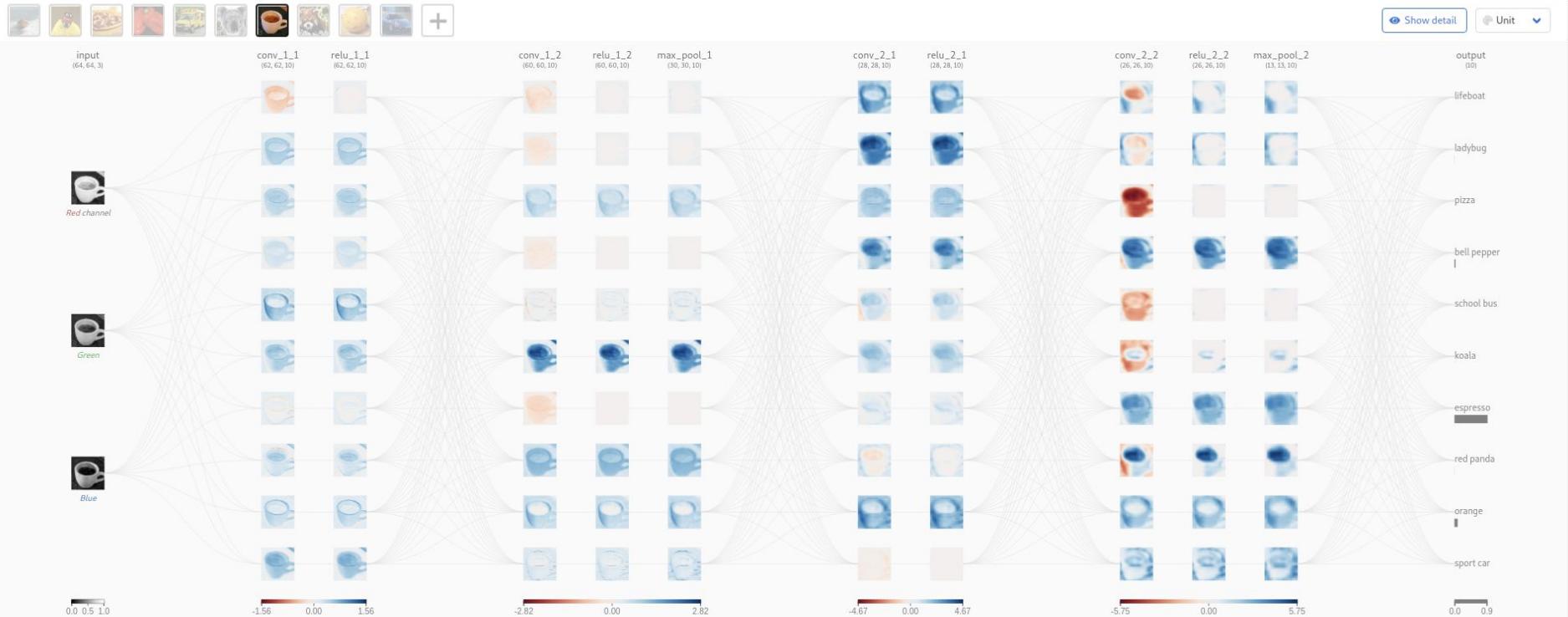
# MNIST - CNN Visualization



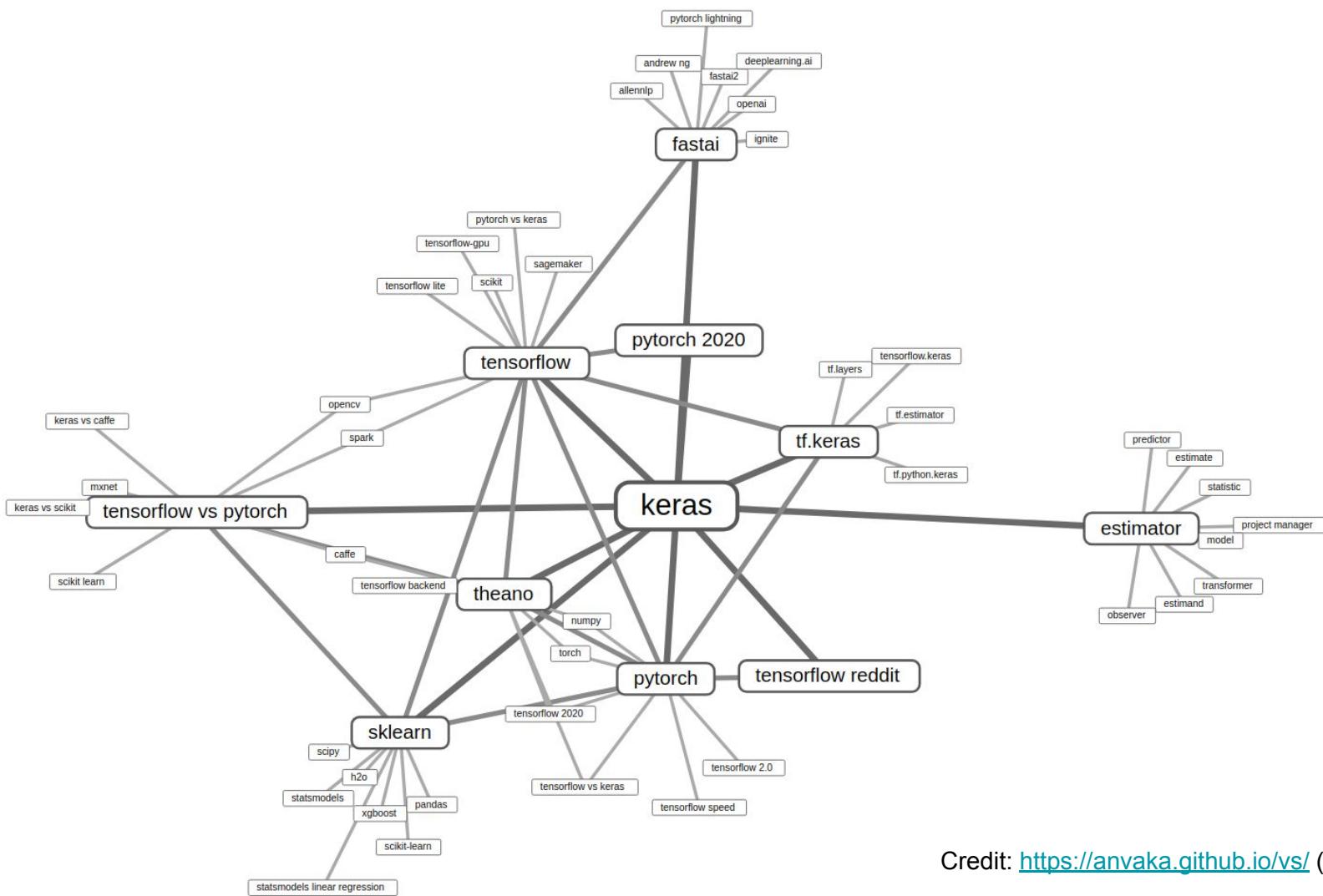
(Image Credit: <http://scs.ryerson.ca/~aharley/vis/>)

# CNN Explainer

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

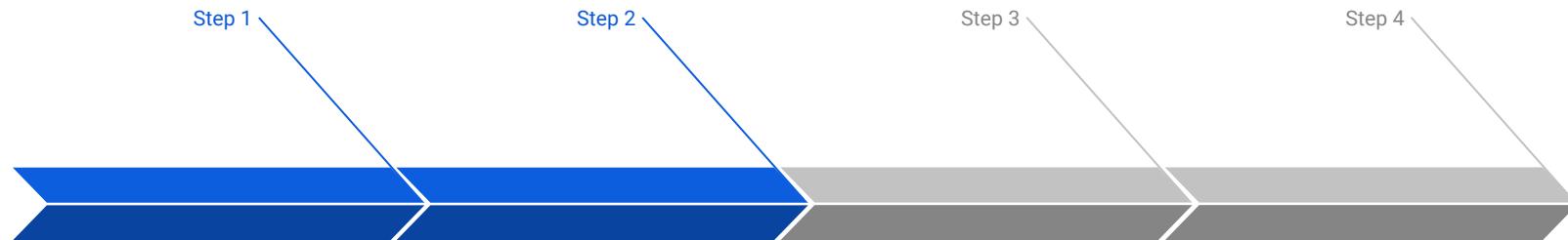


(Image Credit: <https://poloclub.github.io/cnn-explainer/>)



Credit: <https://anvaka.github.io/vs/> ([source](#))

# Machine Learning Workflow with Keras



## Step 1 Prepare Train Data

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

## Step 2 Define Model

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

## Step 3 Training Configuration

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

## Step 4 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.