

# AI Tech Labs 0⇒1

Jian Tao

jtao@tamu.edu

HPRC Short Course

10/30/2020



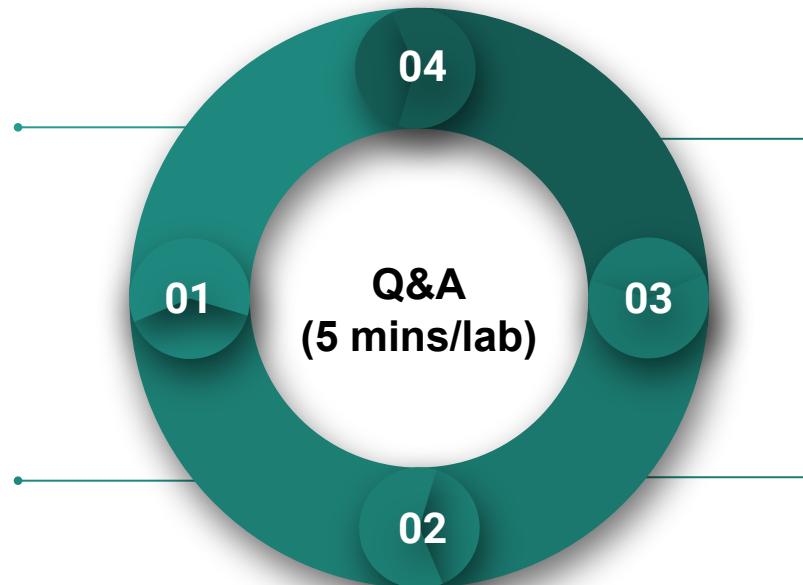
# AI Tech Labs

## 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 a sidebar with tabs for "Files", "Running", "Commands", "Cell Tools", and "Tabs".

The main area shows the notebook content:

```
In this Notebook we explore the Lorenz system of differential equations:  
 $\dot{x} = \sigma(y - x)$   
 $\dot{y} = \rho x - y - xz$   
 $\dot{z} = -\beta z + xy$   
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)
```

The "Output View" tab displays sliders for parameters sigma, beta, and rho, and a 3D plot of the Lorenz attractor.

The "lorenz.py" file is shown in the code editor:

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./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**

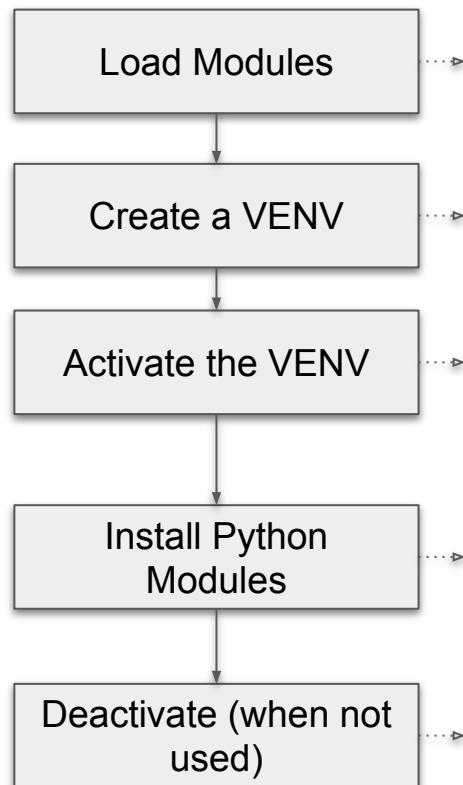
<https://portal-ada.hprc.tamu.edu/pun/sys/dashboard>

# L1 - Shell Access - II

A screenshot of a terminal window titled "jtao@login8:~". The window shows a terminal session on the portal-adam.hprc.tamu.edu system. The session begins with a security notice about the availability of the system for authorized users. It then prompts for a password, indicating a Duo two-factor login for the user "jtao". The user is asked to enter a passcode or select one of two options: "Duo Push to iPhone (iOS)" or "Duo Push to iPad (iOS)". The user enters option 1. The terminal then displays a success message, "Success. Logging you in...", followed by the last login information: "Last login: Fri May 1 22:10:51 2020 from connect-172-31-38-197.vpn.tamu.edu". Below this, there is a footer for Texas A&M University High Performance Research Computing, providing links to the website, consulting services, Ada documentation, and Curie documentation.

```
*****  
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
# source 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 banner 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 arrow points to the "Interactive Apps" menu, which lists several options under BIO (IGV, Mauve, Structure), GUI (ANSYS Workbench, Abaqus/CAE, LS-PREPOST, MATLAB, ParaView, VNC), Galaxy (Galaxy (maroon), Galaxy (reveille)), Servers (Jupyter Notebook, JupyterLab, RStudio Server with R 3.4.3 (Singularity), DStudio Server with Python 3.6.1 (Singularity)), and RC staff members.

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

BIO

- IGV
- Mauve
- Structure

GUI

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

Galaxy

- Galaxy (maroon)
- Galaxy (reveille)

Servers

- Jupyter Notebook
- JupyterLab**
- RStudio Server with R 3.4.3 (Singularity)
- DStudio Server with Python 3.6.1 (Singularity)

RC staff members

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. No jobs will be scheduled during this 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 prohibited. Individuals using HPRC resources 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>

# L1 - Set Virtual Environment

The screenshot shows a web browser window for 'JupyterLab' on the 'portal-ada.hprc.tamu.edu' dashboard. The URL is 'portal-ada.hprc.tamu.edu/pun/sys/dashboard/batch\_connect/sys/jupyterlab/session\_contexts/new'. The page title is 'TAMU HPRC OnDemand (Ada)'. The left sidebar lists 'Interactive Apps' including 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.

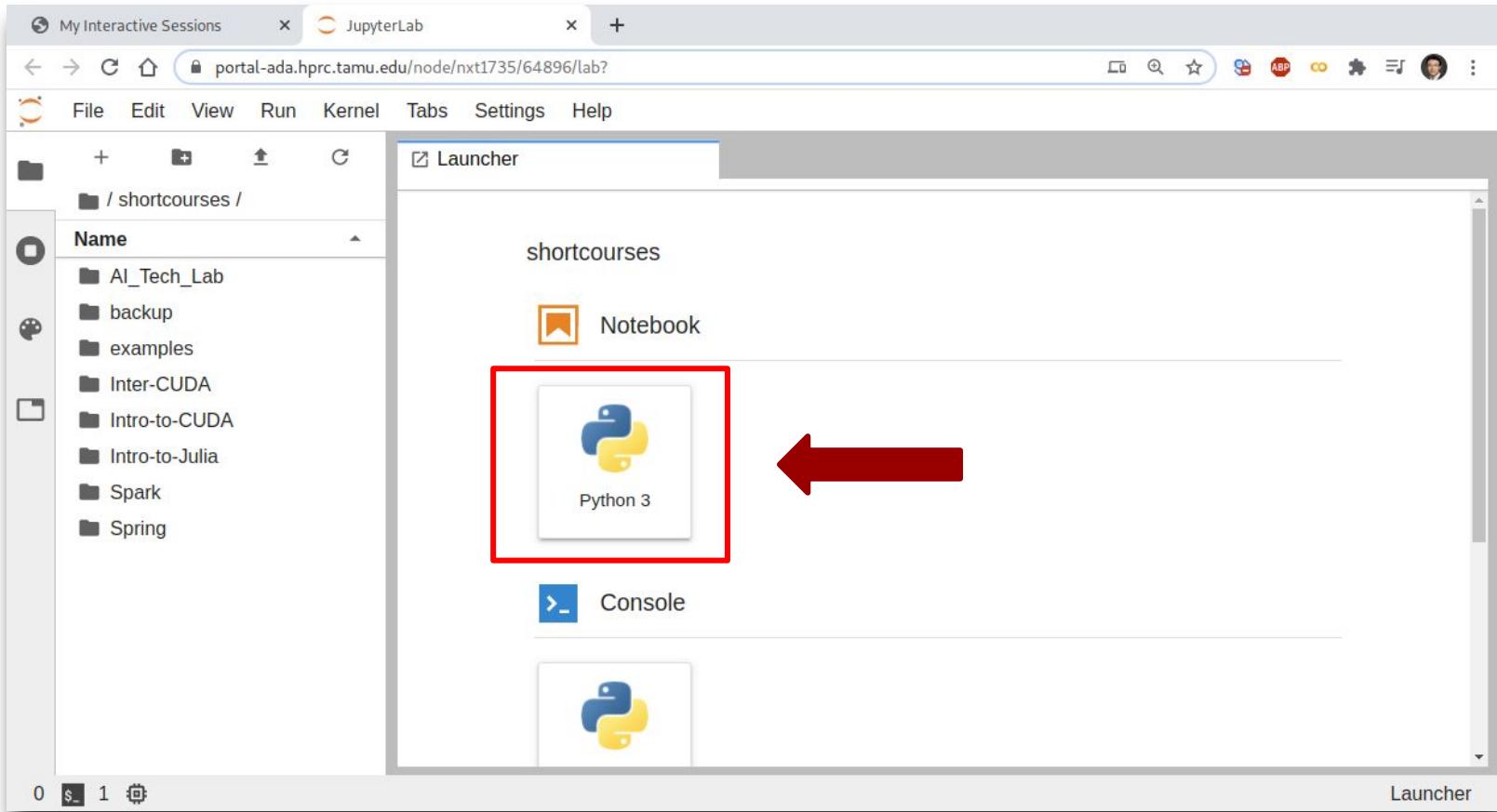
# L1 - Connect to JupyterLab

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 detailed view of a JupyterLab session with the following details:

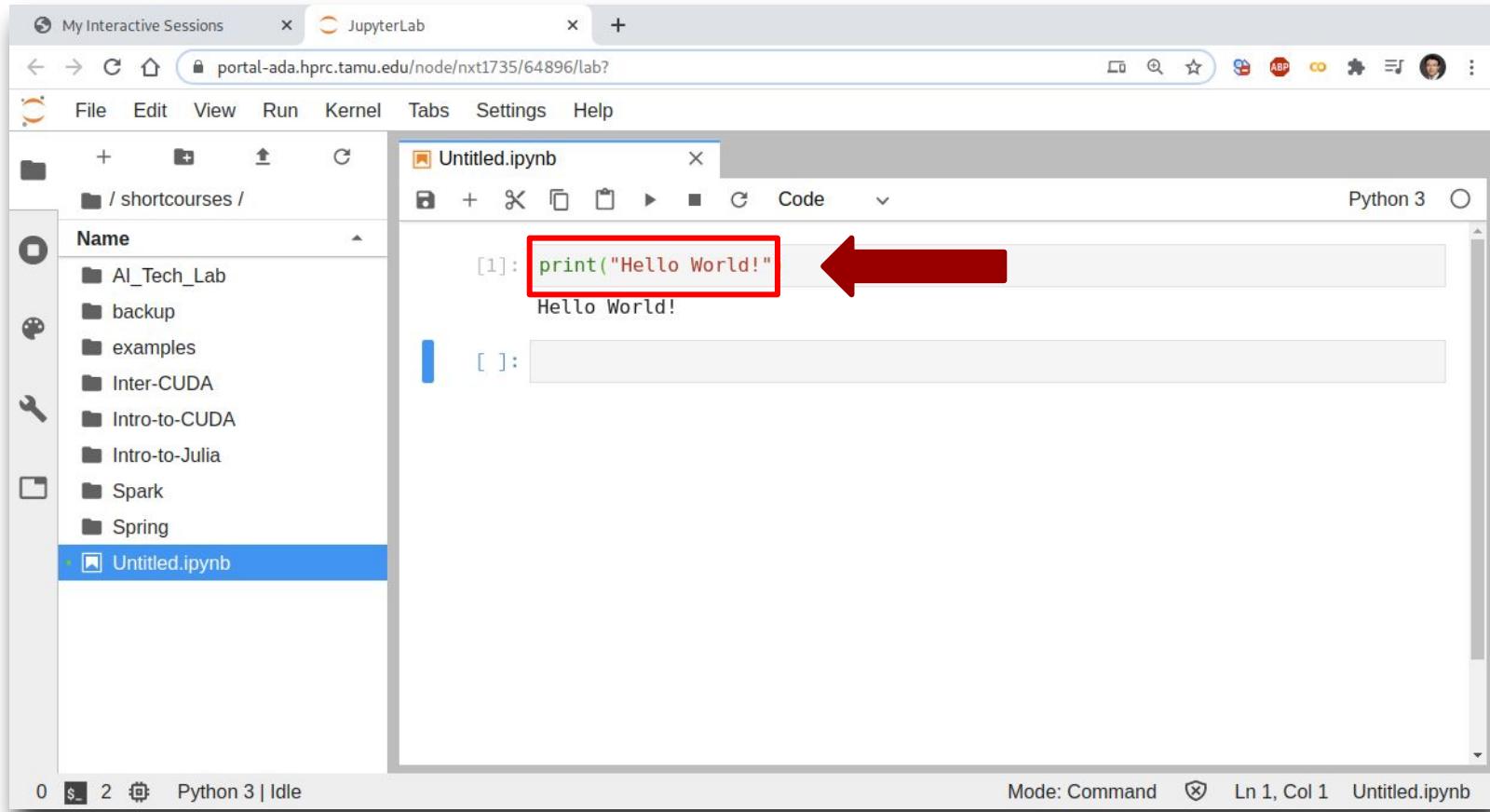
- JupyterLab (12695677)**
- Host:** nxt1735
- Created at:** 2020-09-19 19:10:05 CDT
- Time Used:** 5 minutes
- Session ID:** e18d47a1-3d4f-4f15-b46d-bff5fa09174a
- Status:** 1 node | 1 core | Running
- Action:** Delete (with a red arrow pointing to the "Connect to JupyterLab" button)

A large red arrow points from the bottom right towards the "Connect to JupyterLab" button, which is highlighted with a red border.

# L1 - Create a Jupyter Notebook



# L1 - Test JupyterLab

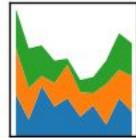
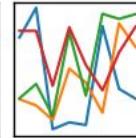


# Lab II. Data Exploration



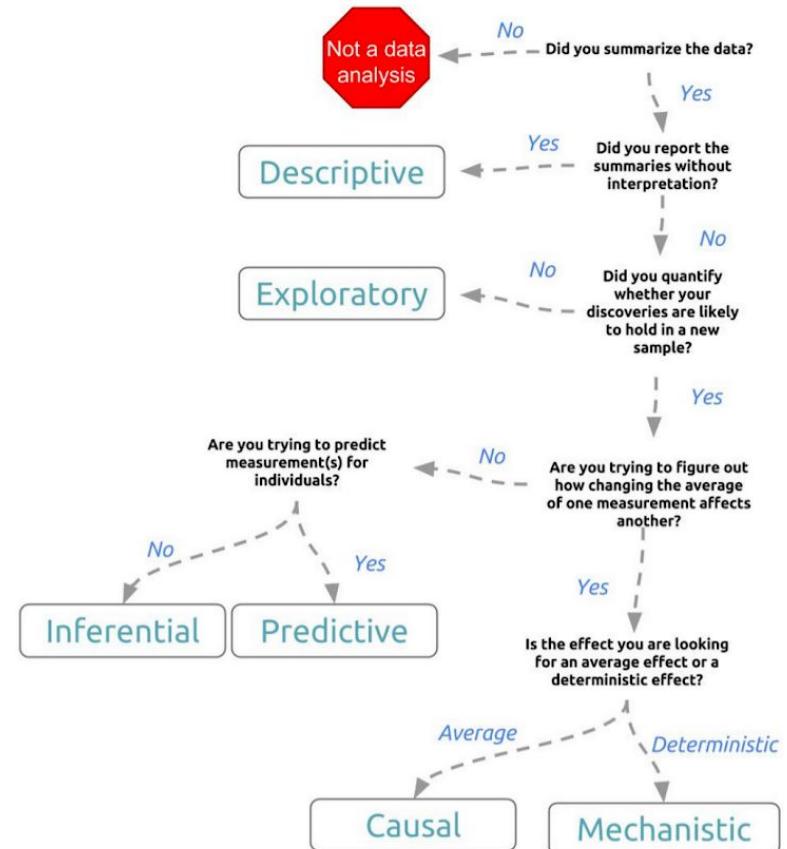
pandas

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



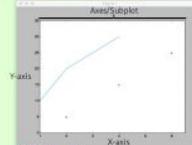
# Types of Data Science Problems

- **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	
<b>Matplotlib</b>	Learn Python interactively at <a href="http://www.DataCamp.com">www.DataCamp.com</a>
	
<b>Matplotlib</b>	
Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.	
<b>1 Prepare The Data</b>	Also see <a href="#">Lists &amp; NumPy</a>
<b>1D Data</b>	
<pre>&gt;&gt;&gt; import numpy as np &gt;&gt;&gt; x = np.linspace(0, 10, 100) &gt;&gt;&gt; y = np.cos(x) &gt;&gt;&gt; z = np.sin(x)</pre>	
<b>2D Data or Images</b>	
<pre>&gt;&gt;&gt; data = 2 * np.random.random((10, 10)) &gt;&gt;&gt; data[2, 3] = np.random.random((10, 10)) &gt;&gt;&gt; V = x * np.pi * np.sin(z) / 1000 &gt;&gt;&gt; V = 1 + x - V**2 &gt;&gt;&gt; from matplotlib.image import get_sample_data &gt;&gt;&gt; img = np.imread(get_sample_data('mnist/mnist_binarize_normal.npy'))</pre>	
<b>2 Create Plot</b>	
<b>Figure</b>	
<pre>&gt;&gt;&gt; import matplotlib.pyplot as plt &gt;&gt;&gt; fig = plt.figure() &gt;&gt;&gt; fig2 = plt.figure(figsize=plt.rcParams['figure.figsize'])</pre>	
<b>Axes</b>	
All plotting is done with respect to an <code>Axes</code> . In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.	
<pre>&gt;&gt;&gt; fig.add_axes() &gt;&gt;&gt; ax1 = fig.add_subplot(221) # row-col-num &gt;&gt;&gt; ax2 = fig.add_subplot(222, sharex=ax1) &gt;&gt;&gt; fig3, axes = plt.subplots(nrows=2, ncols=2) &gt;&gt;&gt; fig4, axes2 = plt.subplots(ncols=3)</pre>	
<b>3 Plotting Routines</b>	
<b>1D Data</b>	
<pre>&gt;&gt;&gt; lines = ax.plot(x,y) &gt;&gt;&gt; ax.scatter(x,y) &gt;&gt;&gt; ax.vlines([1,2,3], [3,4,5]) &gt;&gt;&gt; ax.hlines([1,2,3], [1,2,3]) &gt;&gt;&gt; ax.step(x,y) &gt;&gt;&gt; ax.stem(x,y) &gt;&gt;&gt; ax.fill(x,y,color='blue') &gt;&gt;&gt; ax.fill_between(x,y,color='yellow')</pre>	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 values and 0
<b>2D Data or Images</b>	
<pre>&gt;&gt;&gt; fig, ax = plt.subplots() &gt;&gt;&gt; ax.imshow(img, cmap='gist_earth',              interpolation='nearest',              vmin=0,              vmax=2)</pre>	Colormapped or RGB arrays
<b>Plot Anatomy &amp; Workflow</b>	<b>Workflow</b>
<b>Plot Anatomy</b>	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
	<pre>&gt;&gt;&gt; import matplotlib.pyplot as plt &gt;&gt;&gt; x = [1,2,3,4] &gt;&gt;&gt; y = [15,20,25,30] &gt;&gt;&gt; fig, ax = plt.subplots() &gt;&gt;&gt; ax.plot(x, y, color='lightblue', linewidth=3) # Step 2 &gt;&gt;&gt; ax.scatter([2,4,6], [15,20,25], color='darkgreen', marker='+') # Step 3 &gt;&gt;&gt; ax.set_xlim(1, 6.5) &gt;&gt;&gt; plt.savefig('foo.png') # Step 4 &gt;&gt;&gt; plt.show() # Step 6</pre>
<b>4 Customize Plot</b>	
<b>Colors, Color Bars &amp; Color Maps</b>	
<pre>&gt;&gt;&gt; plt.plot(x, x, x**2, x**3) &gt;&gt;&gt; ax.plot(x, y, alpha=0.4) &gt;&gt;&gt; ax.set_color_cycle(['red','blue']) &gt;&gt;&gt; from matplotlib.colors import LinearSegmentedColormap &gt;&gt;&gt; im = ax.imshow(img, cmap='seismic')</pre>	
<b>Markers</b>	
<pre>&gt;&gt;&gt; fig, ax = plt.subplots() &gt;&gt;&gt; ax.scatter(x,y,marker='*') &gt;&gt;&gt; ax.plot(x,y,marker='o')</pre>	
<b>LineStyles</b>	
<pre>&gt;&gt;&gt; plt.plot(x,y,linewidth=4.0) &gt;&gt;&gt; plt.plot(x,y,ls='solid') &gt;&gt;&gt; plt.plot(x,y,ls='dashed') &gt;&gt;&gt; plt.plot(x,y,ls='dashdot') &gt;&gt;&gt; plt.plot(x,y,ls='dotted') &gt;&gt;&gt; plt.setp(lines,color='r',linewidth=4.0)</pre>	
<b>Text &amp; Annotations</b>	
<pre>&gt;&gt;&gt; ax.text(1, 1, 'Example Graph', style='italic') &gt;&gt;&gt; ax.annotate("A", xy=(8, 0), xytext=(10.5, 0), textcoords="data", arrowprops=dict(facecolor="black", connectionstyle="arc3"), )</pre>	
<b>Vector Fields</b>	
<pre>&gt;&gt;&gt; ax.quiver(x,y,z) &gt;&gt;&gt; ax.streamplot(X,Y,U,V)</pre>	Add an arrow to the axes Plot a 2D field of arrows Plot 2D vector fields
<b>Distribution</b>	
<pre>&gt;&gt;&gt; ax1.hist(y) &gt;&gt;&gt; ax1.boxplot(y) &gt;&gt;&gt; ax1.violinplot(y)</pre>	Plot a histogram Make a box and whisker plot Make a violin plot
<b>3D</b>	
<b>Vector Fields</b>	
<pre>&gt;&gt;&gt; ax2 = fig3.add_subplot(211) &gt;&gt;&gt; ax2.quiver(x,y,z) &gt;&gt;&gt; ax2.set_zlabel('z')</pre>	Pseudocolor plot of zD array Pseudocolor plot of zD array Plot contours Plot filled contours Label a contour plot
<b>Data Distributions</b>	
<pre>&gt;&gt;&gt; ax3 = fig3.add_subplot(212) &gt;&gt;&gt; ax3.set_zlabel('z')</pre>	
<b>Close &amp; Clear</b>	
<b>Close</b>	
<pre>&gt;&gt;&gt; plt.close() &gt;&gt;&gt; plt.cif() &gt;&gt;&gt; plt.close()</pre>	Clear an axis Clear the entire figure Close a window
<b>Show Plot</b>	
<pre>&gt;&gt;&gt; plt.show()</pre>	

[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.

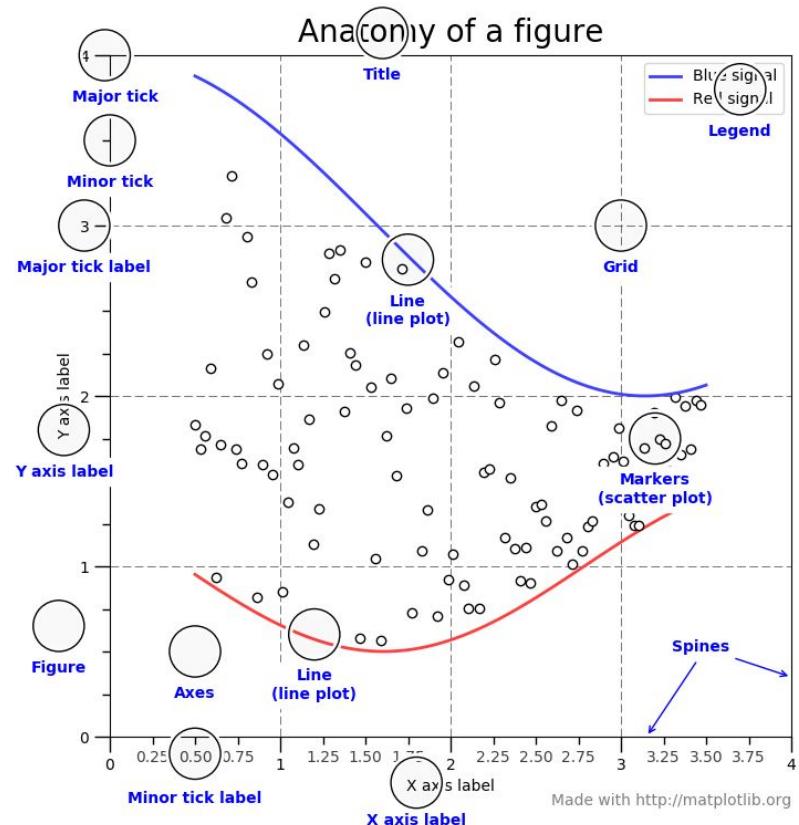
Adjustable parameters include:

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'])
```

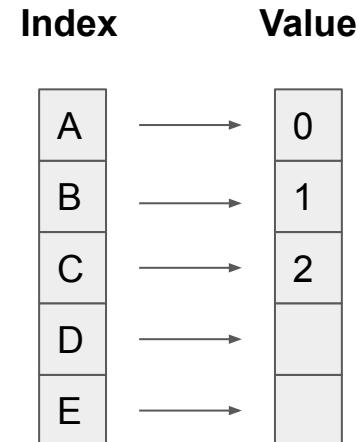
```
In [5]: s.index
```

```
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>

## Tidy Data

A foundation for wrangling in pandas

In a tidy data set:  
Each variable is saved in its own column  
Each observation is saved in its own row

### Syntax – Creating DataFrames

```
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.
```

```
df = pd.DataFrame([
    {"n": 1, "x": 4, "y": 5, "z": 6},
    {"n": 2, "x": 7, "y": 8, "z": 9},
    {"n": 3, "x": 10, "y": 11, "z": 12}],
    index = pd.MultiIndex.from_tuples(
        [(1, 'a'), (2, 'b'), (3, 'c')]),
    names=['n', 'x', 'y', 'z'])
Create DataFrame with a MultiIndex.
```

### 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 > 7]`  
Extract rows that meet logical criteria

`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.

### 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='regrex')`  
Select columns whose name matches regular expression regex.

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

`df.iloc[:, 1, 2, 3]`  
Select columns in positions 1, 2 and 3 (and 1st column is 0).

`df.loc[df['a'] > 1, ['a', 'c']]`  
Select rows meeting logical condition, and only the specific columns.

### 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'))
    )
```

### Logic in Python (and pandas)

<	Less than	=	Not equal to
>	Greater than	!=	df.column.isin(values)
==	Equals	isna	Group membership
!=	Less than or equal	notna	is NaN
	Greater than or equal	any	is not NaN
=	Greater than or equals	all	Logical and; or, not, xor, wxyz_all

<http://pandas.pydata.org> - This cheat sheet created by Rutisha Data Wrangling Cheatsheet (<https://www.rutisha.com/pandas-data-wrangling-cheat-sheet.pdf>) written by Rutisha, Project Coordinator

### 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`

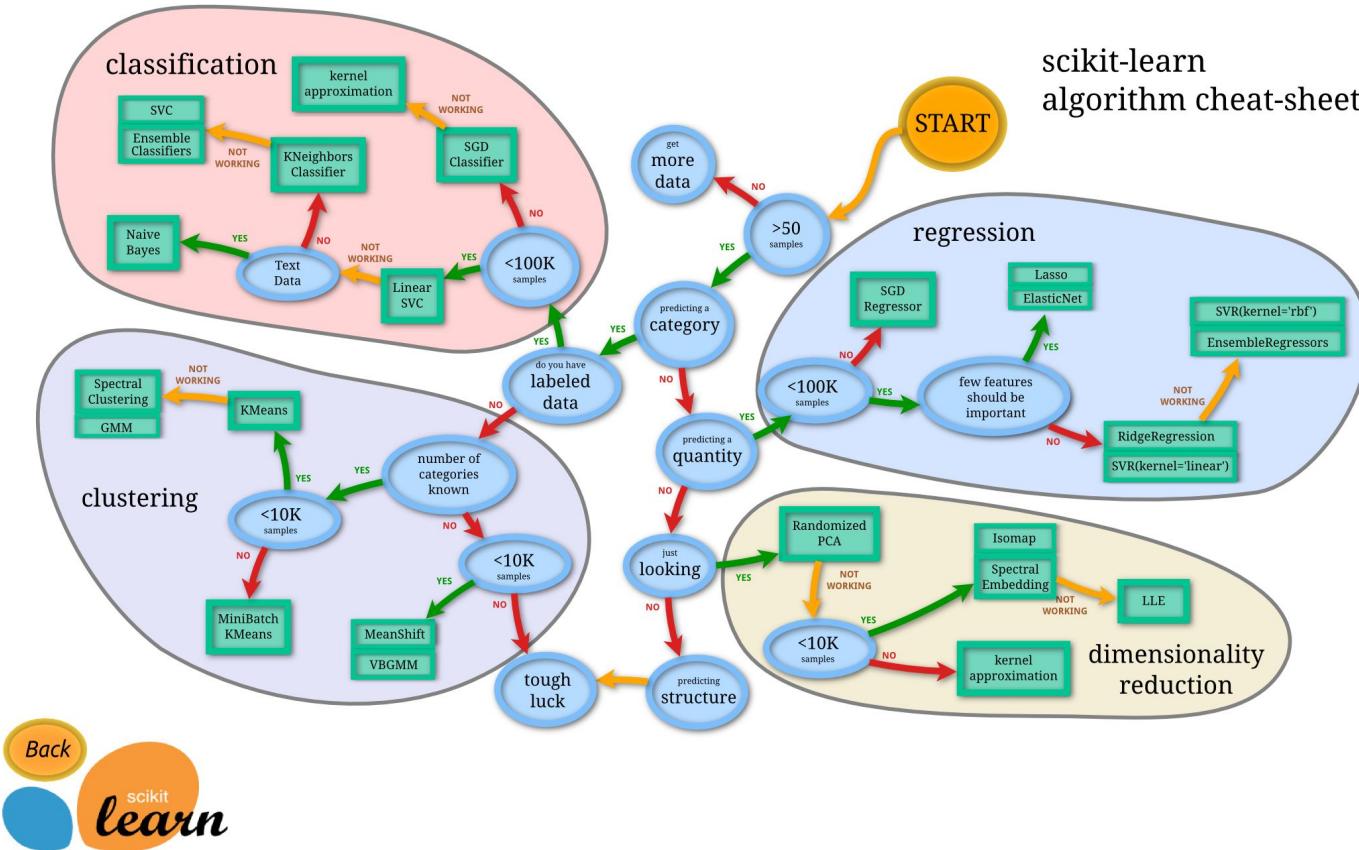
`x1 + x2`

`A + B`

`x1 + x2 + A + B`

[https://pandas.pydata.org/Pandas\\_Cheat\\_Sheet.pdf](https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf)

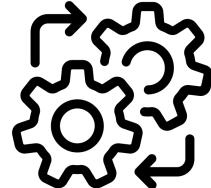
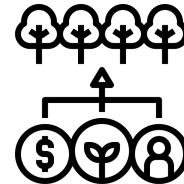
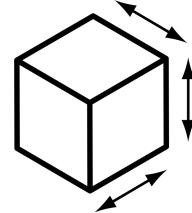
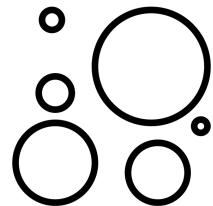
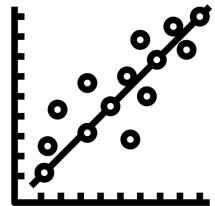
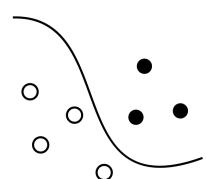
# Lab III. Machine Learning



# 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*

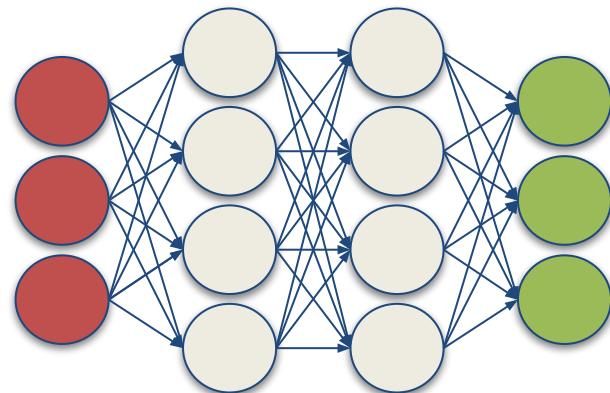
by Grant Sanderson

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

## *Visualization of CNN*

by Adam Harley

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



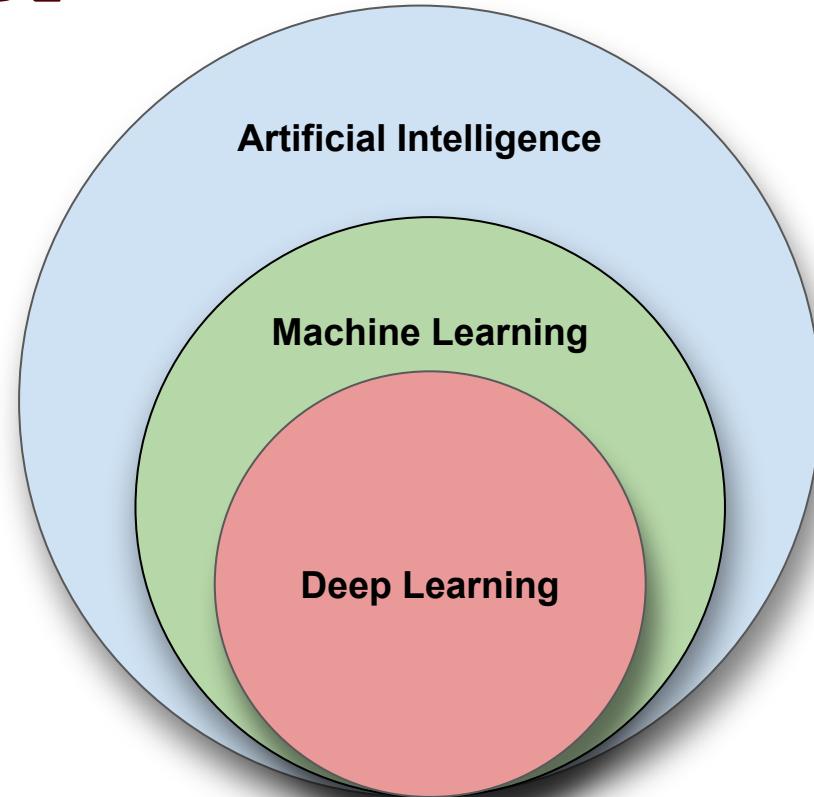
TensorFlow  
2.0



Keras

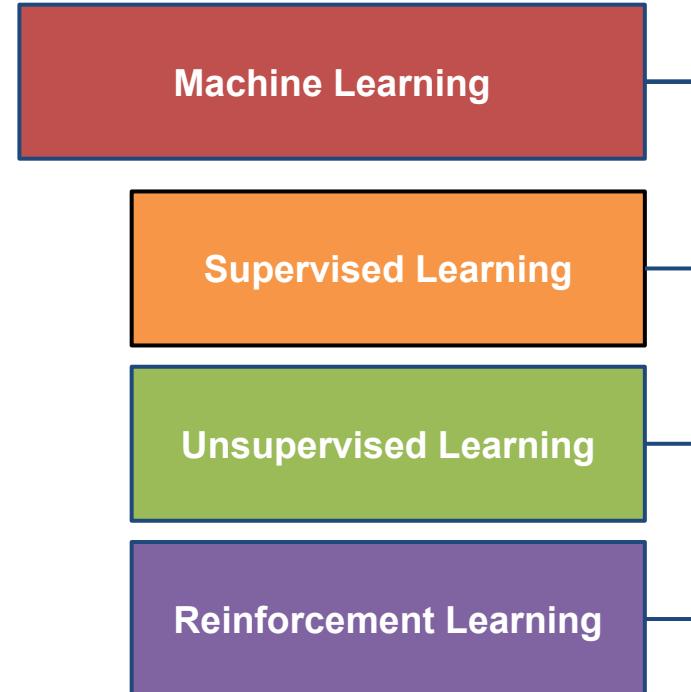
# Relationship of AI, ML, and DL

- **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**.

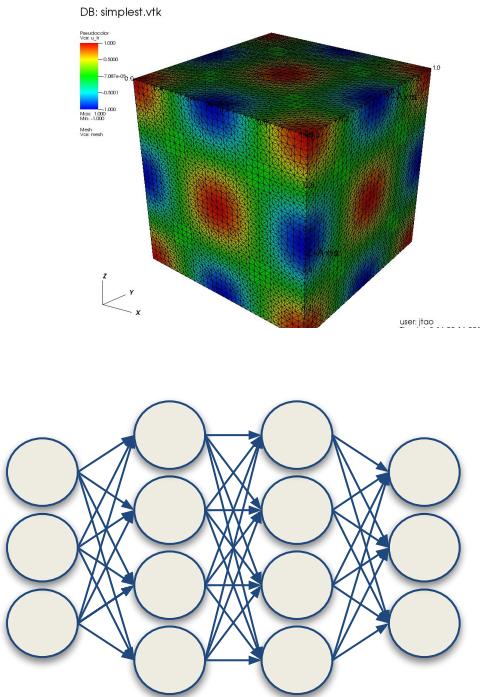


# Types of ML Algorithms

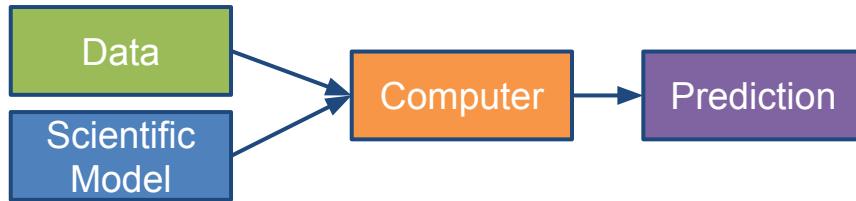
- **Supervised Learning**
  - trained with labeled data; including regression and classification problems
- **Unsupervised Learning**
  - trained with unlabeled data; clustering and association rule learning problems.
- **Reinforcement Learning**
  - no training data; stochastic Markov decision process;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



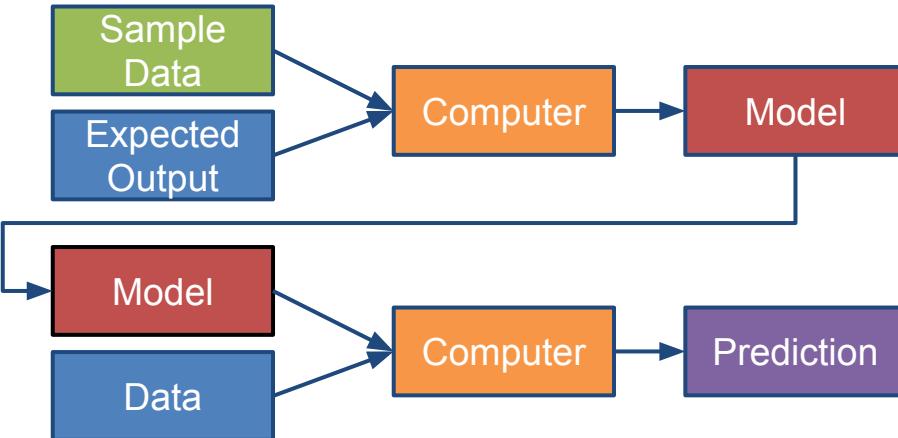
# Machine Learning



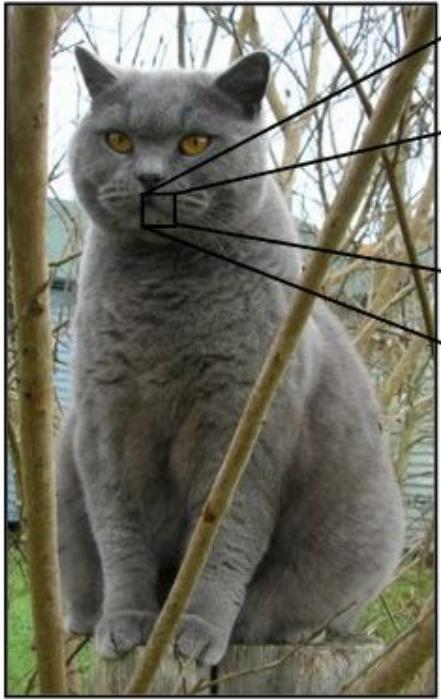
## Traditional Modeling



## Machine Learning (Supervised Learning)



# Inputs and Outputs

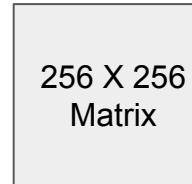


05	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	04
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	04	56	62
81	49	31	73	55	79	14	29	93	71	40	67	53	88	30	08	49	13	36
52	70	95	23	04	60	11	42	69	51	68	56	01	32	56	71	37	02	36
22	31	16	71	51	63	03	69	41	92	36	54	22	40	40	28	66	33	13
24	47	34	60	99	03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

What the computer sees

image classification

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



DL model

4-Element Vector

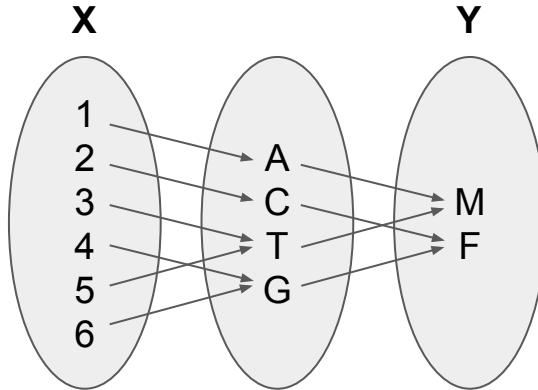
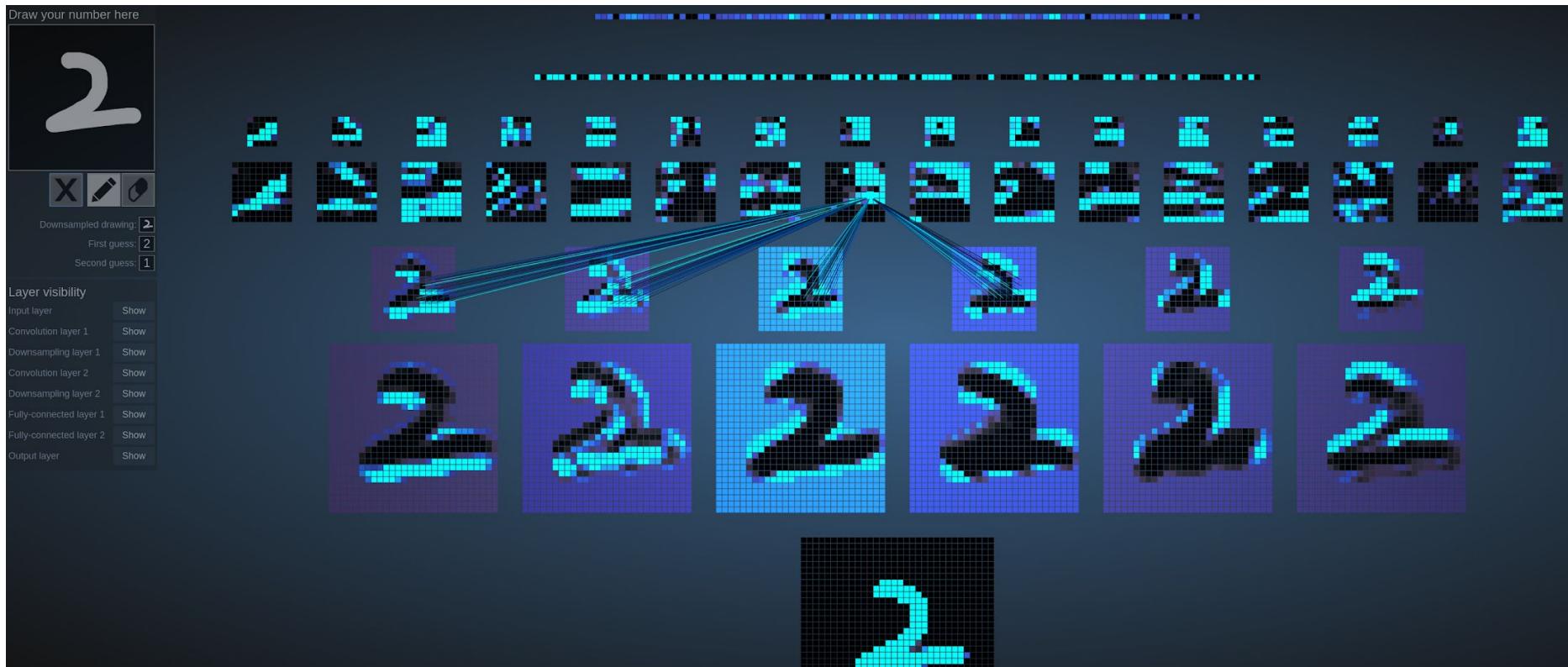


Image from the [Stanford CS231 Course](#)

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

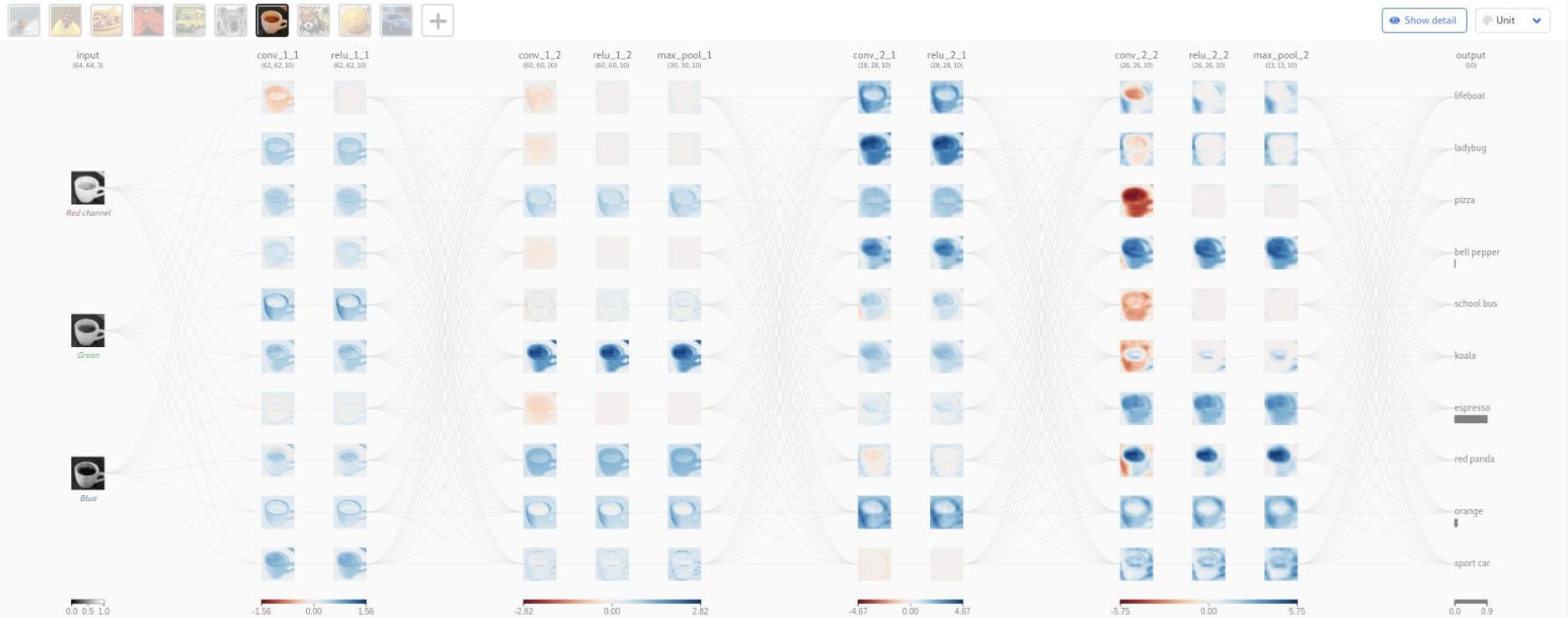
# MNIST - CNN Visualization



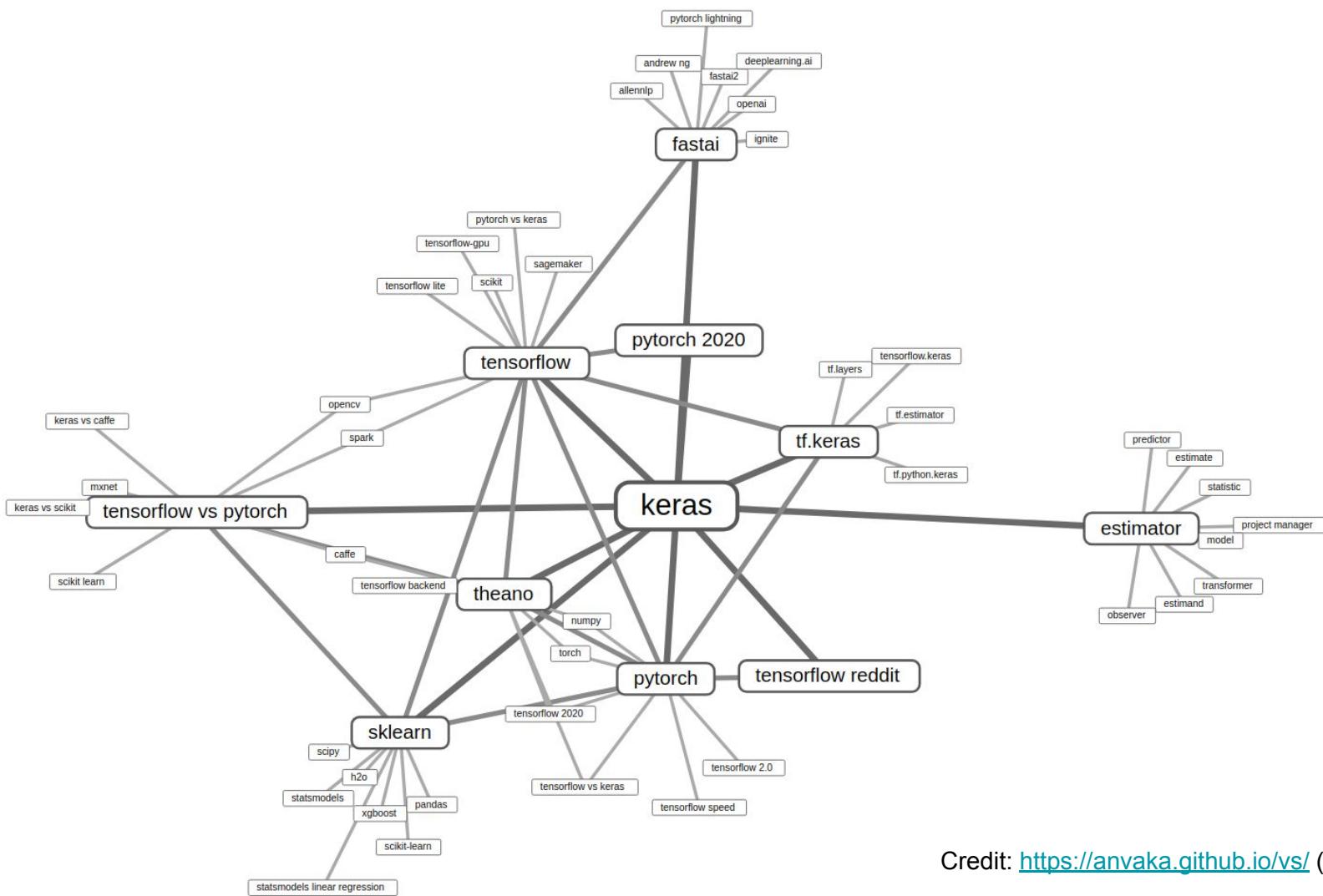
(Image Credit: <http://scs.ryerson.ca/~aharley/vis/>)

# CNN Explainer

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

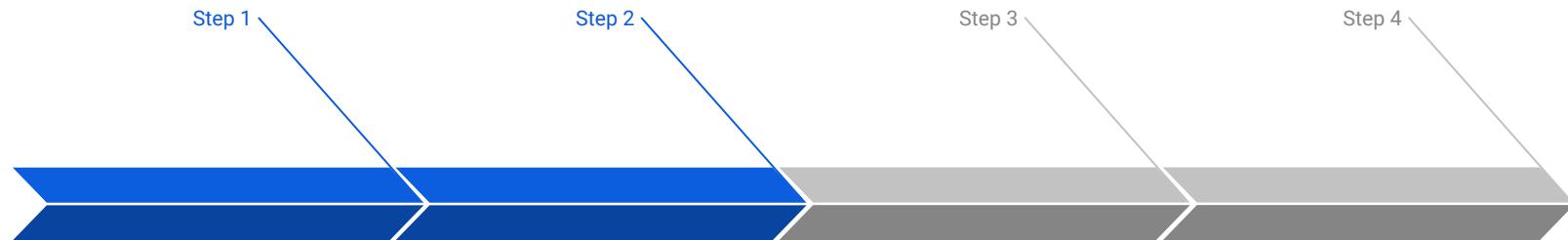


(Image Credit: <https://poloclub.github.io/cnn-explainer/>)



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

# Machine Learning Workflow with Keras



## 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.