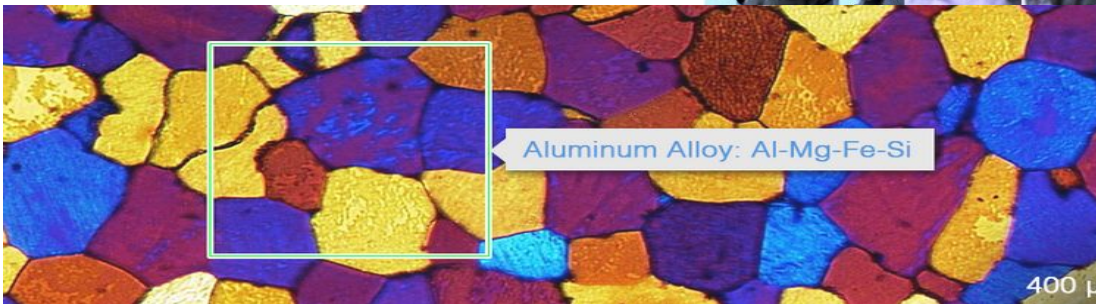
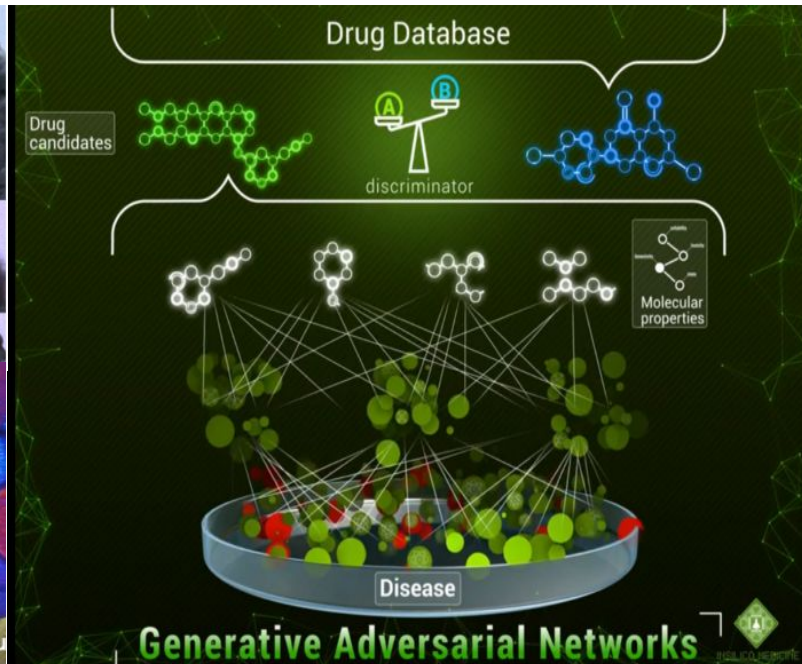
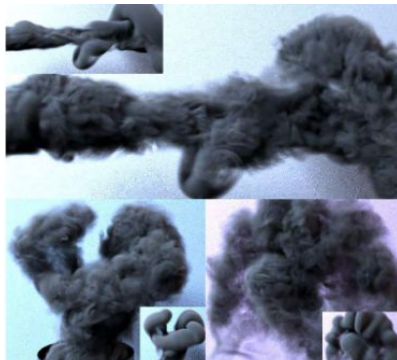
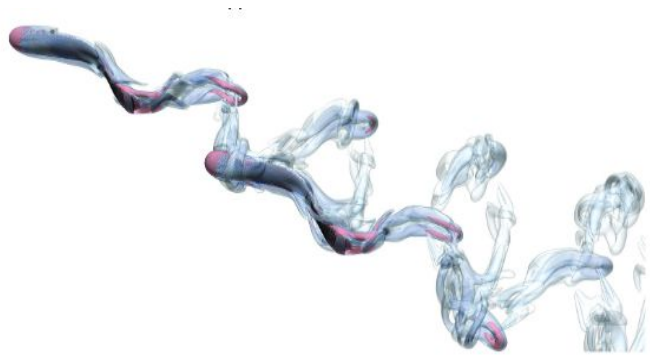


# ENM 360: Introduction to Data-driven Modeling

## Lecture #1: Introduction, motivation and course logistics

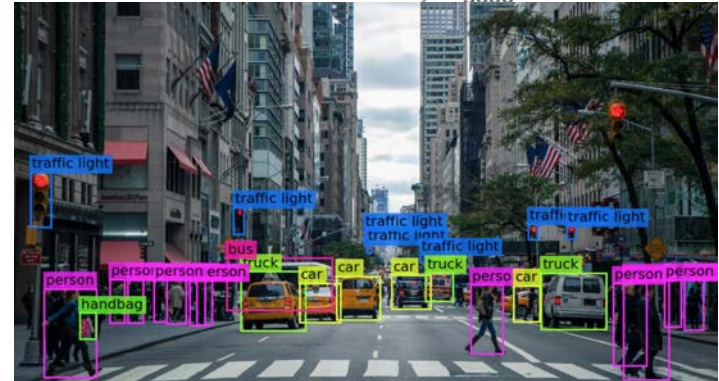
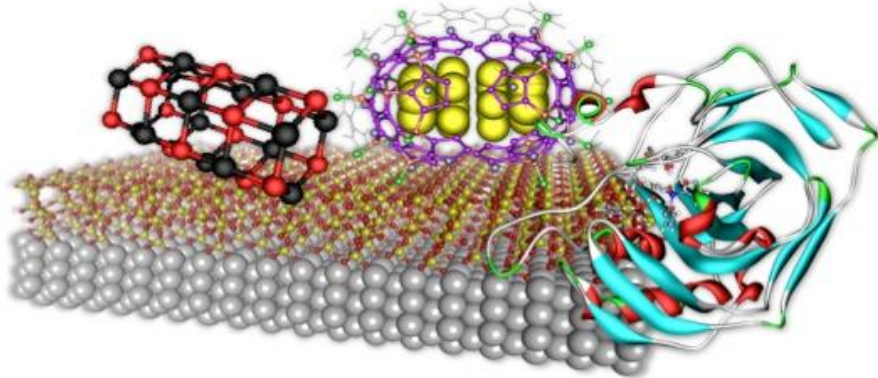
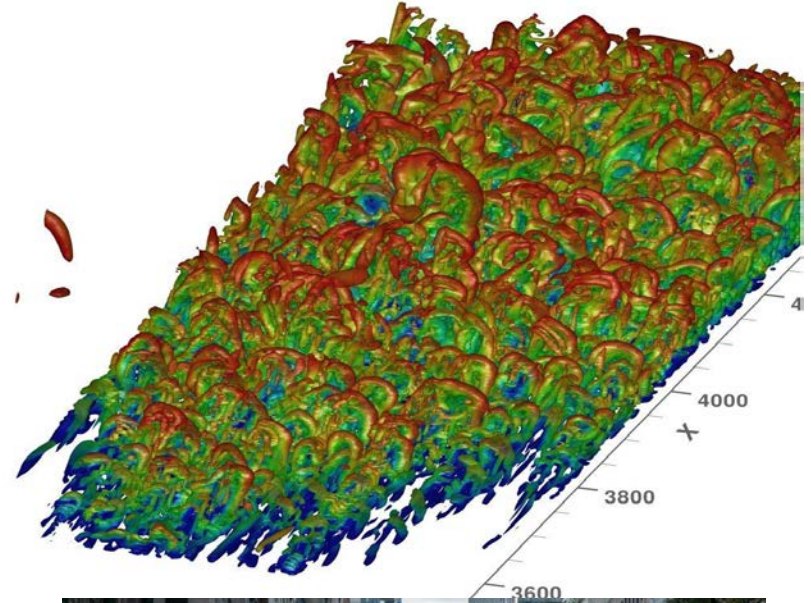
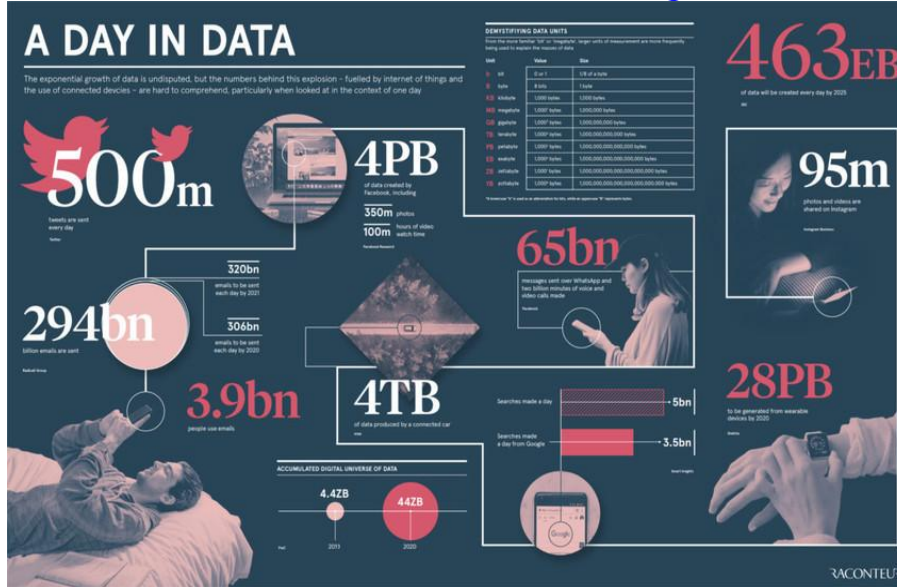
Paris Perdikaris  
September 1, 2020





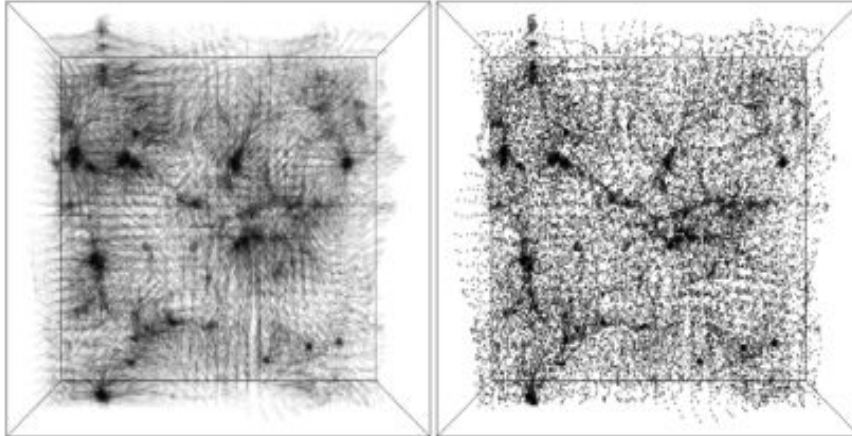
*From recognizing voice, text or images to designing more efficient airplane wings and discovering new drugs, machine learning is introducing a transformative set of tools in data analysis with increasing impact across engineering, sciences, and commercial applications. In this course, you will learn about principles and algorithms for extracting patterns from data and making effective automated predictions. We will cover concepts such as regression, classification, density estimation, feature extraction, sampling, and probabilistic modeling, and provide a formal understanding of how, why, and when these methods work in the context of analyzing physical, biological, and engineering systems.*

# Living in the Age of Big Data





# Successes of Deep Learning in Different Fields



```
python /tf_files/label_image.py /tf_files/diagnose/leg005.jpg  
python /tf_files/label_image.py /tf_files/diagnose/leg009.jpg
```

10-m

Use  
Deep  
Learning  
to Help  
Diagnose  
Injury...



GraphDef version 9. Use tf.nn.brokenleg (score = 0.91144)  
healthyleg (score = 0.08856)

gender ♂  
age 23  
HR 95 bpm  
120/60  
ECHO D  
CO PWR<500  
Frg 2.0 MHz  
1600 mm  
AO 100%



GraphDef version 9. Use tf.nn.brokenleg (score = 0.76665)  
healthyleg (score = 0.23335)

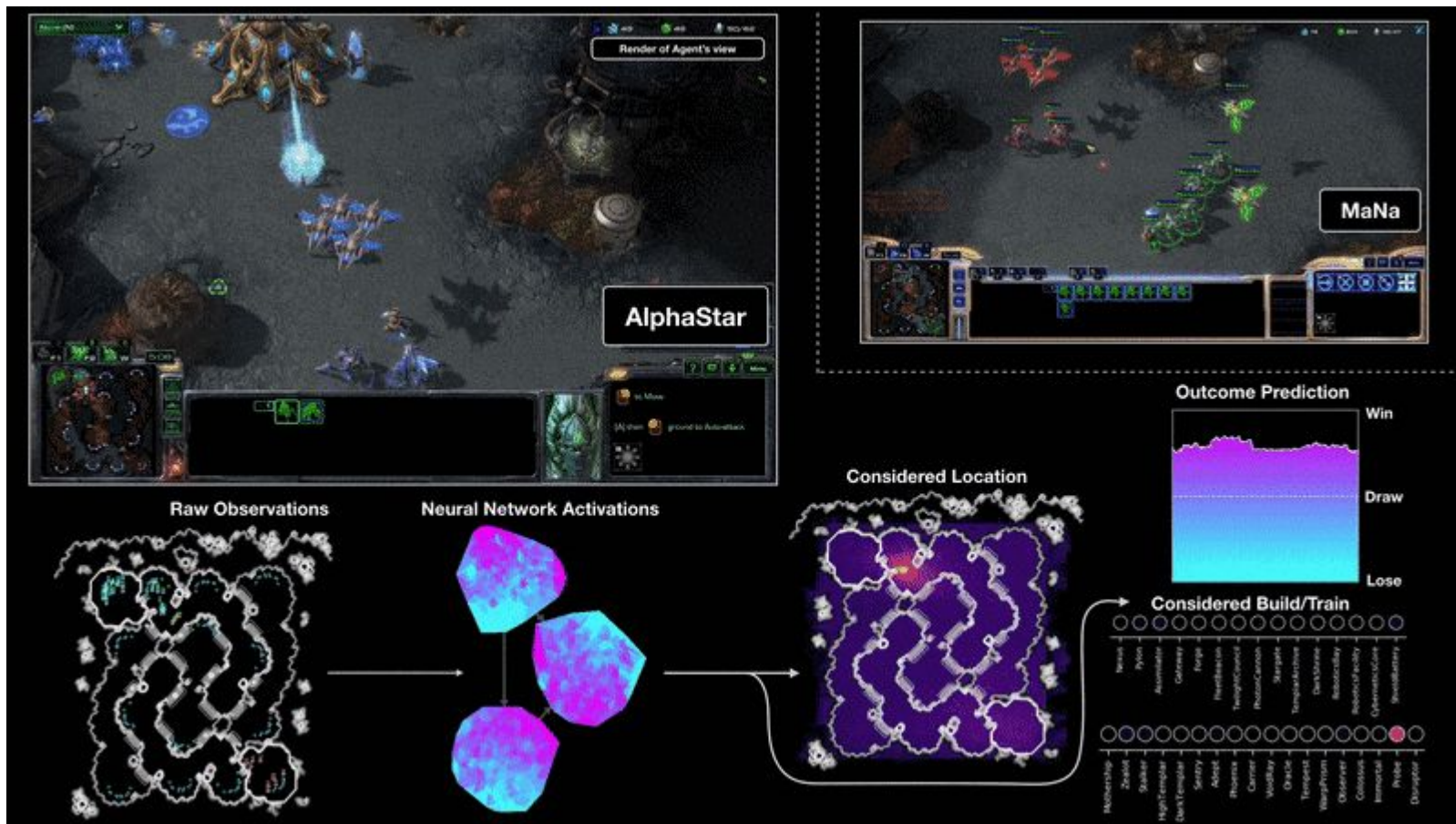
Human View



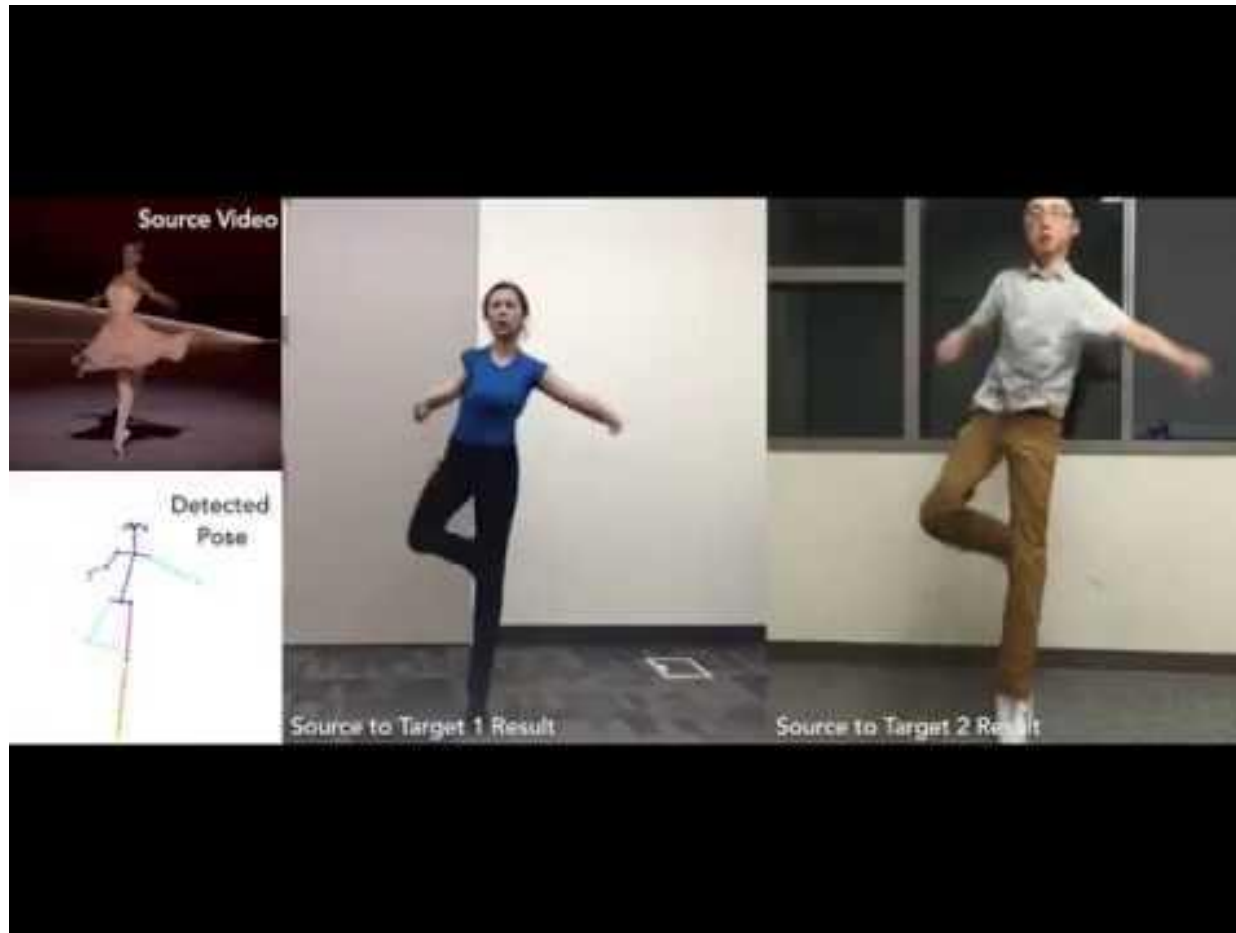
AI View

3.006	-1.386	-0.4695	0.883	1	0.84
-0.3154	-0.5425	-0.5	0.866	0	0.82
3.11	-1.36	-0.9336	0.3584	1	0.78
-2.324	2.863	0.9746	0.225	0	0.86
3.037	-1.361	-0.7773	0.6294	1	0.82
-1.387	2.951	0.988	0.1565	0	0.74
3.023	-0.9395	0.05234	-0.9985	0	0.66
2.951	-0.5747	0.01746	1	0	0.72
2.963	-1.303	0.3906	0.9204	0	0.68
2.834	-3.164	0.01746	-1	0	0.68
3.127	-1.368	0.6562	0.755	1	0.55
3.088	-1.366	0.4695	0.883	0	0.55
2.984	-1.398	-0.225	0.9746	1	0.55
3.037	-1.391	0.788	0.6157	0	0.55
3.076	-1.438	0.883	0.4695	0	0.55
-2.412	2.846	0.996	0.08716	1	0.3

## Working with high dimensional data



# Working with high dimensional data

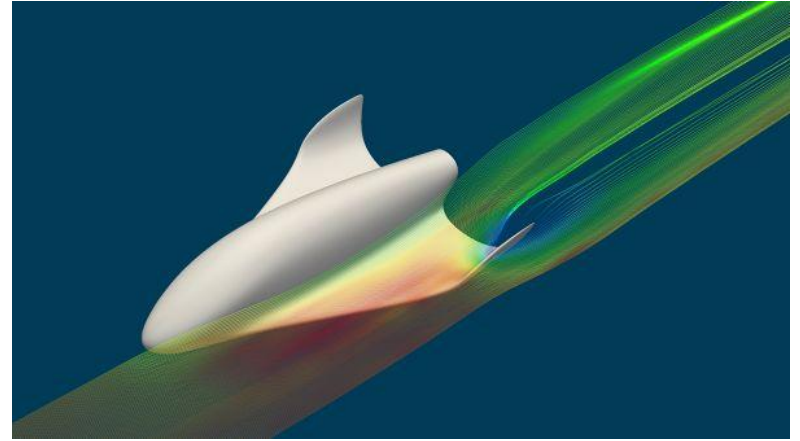
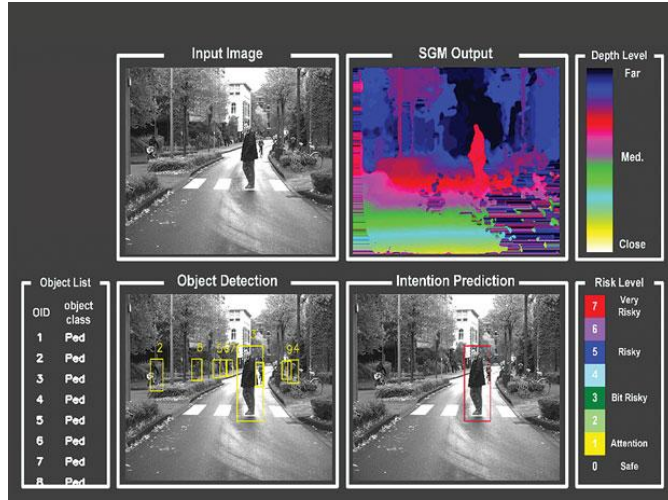




## Working with high dimensional data

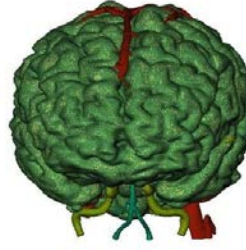
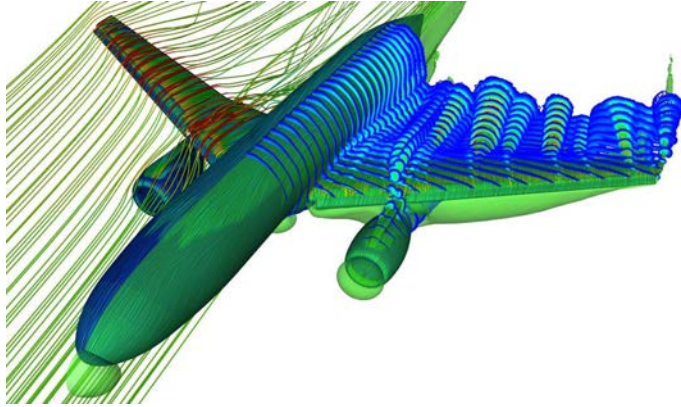


# Motivation and Addressing open challenges

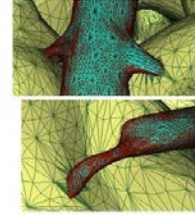




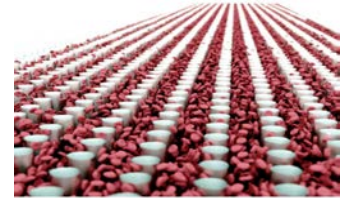
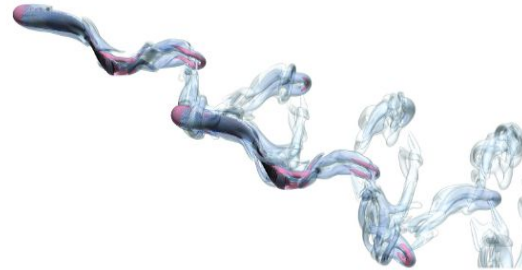
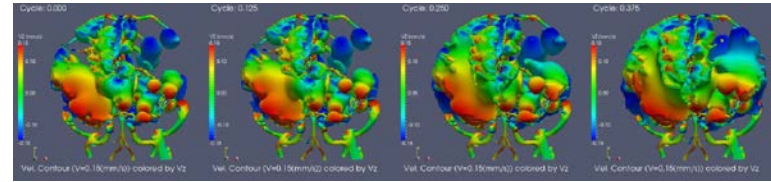
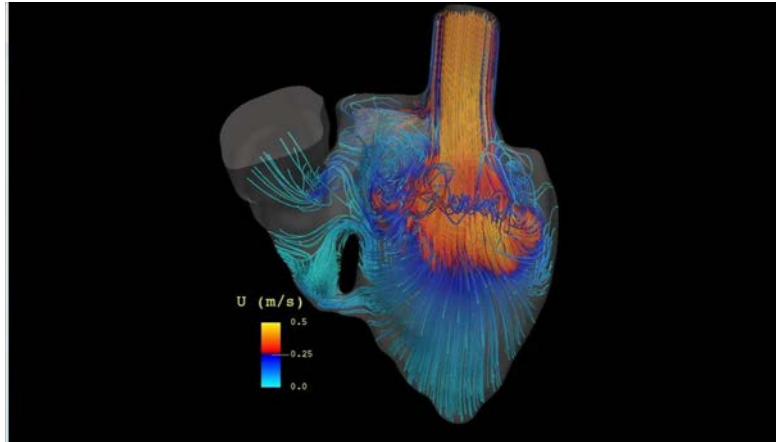
# Computational Science and Engineering

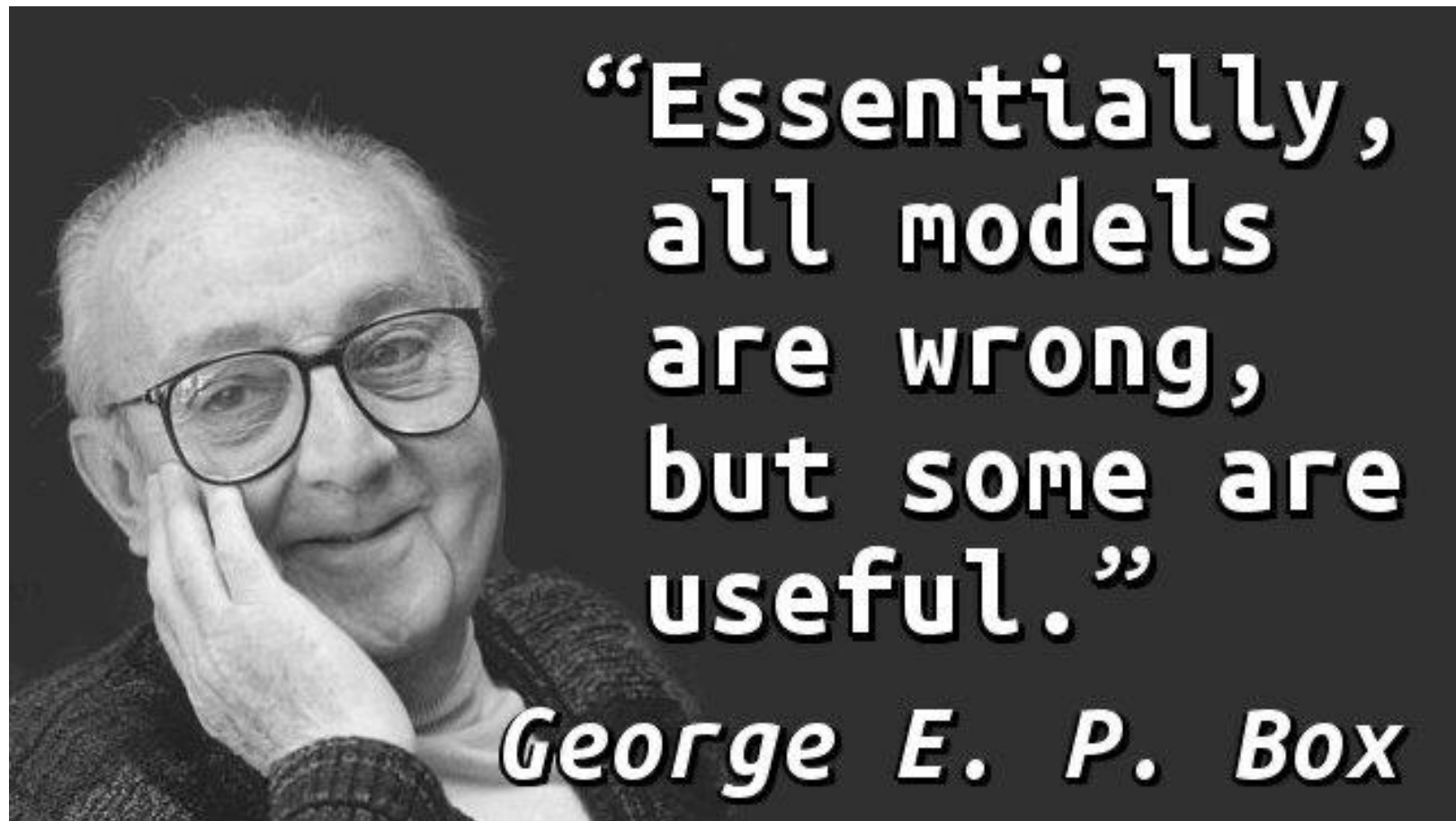


(a) Triangular Meshes  
(overall view)



(b) Triangular meshes  
(enlarged view)

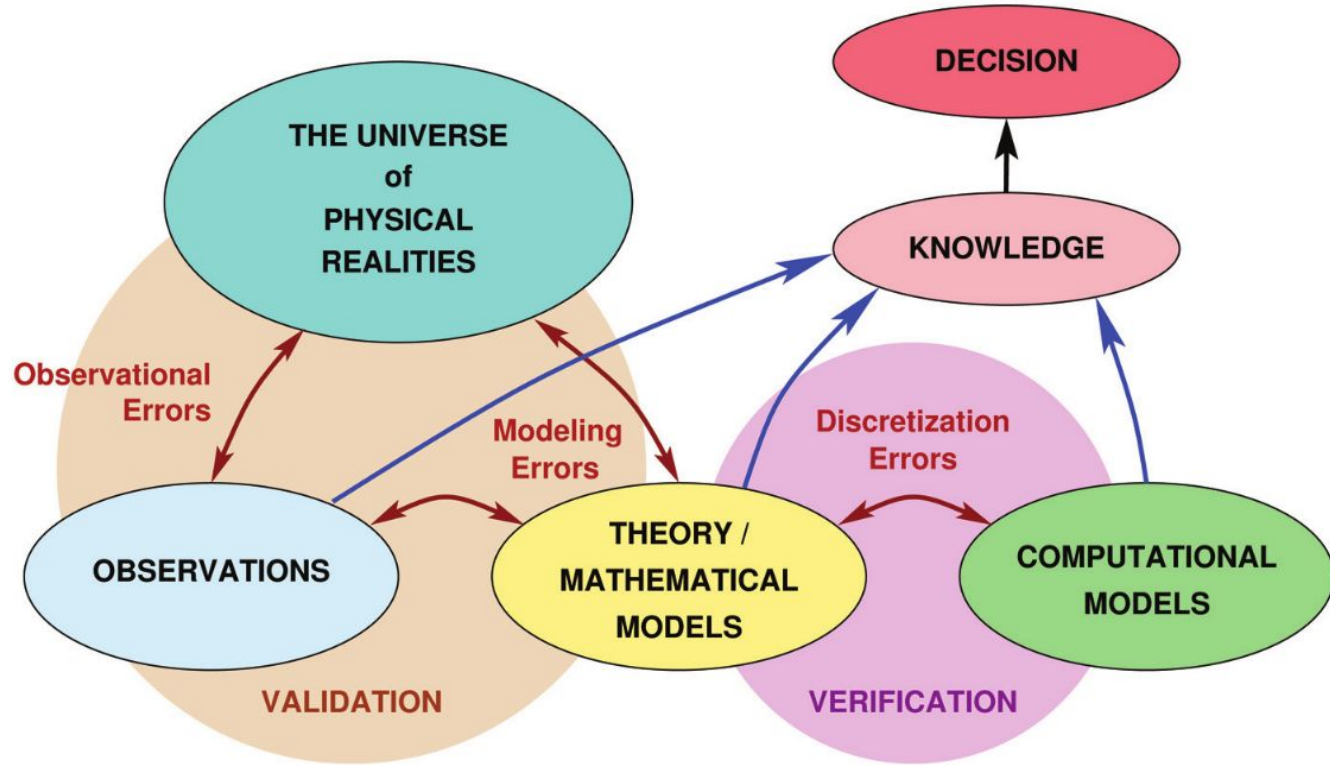




**“Essentially,  
all models  
are wrong,  
but some are  
useful.”**

***George E. P. Box***

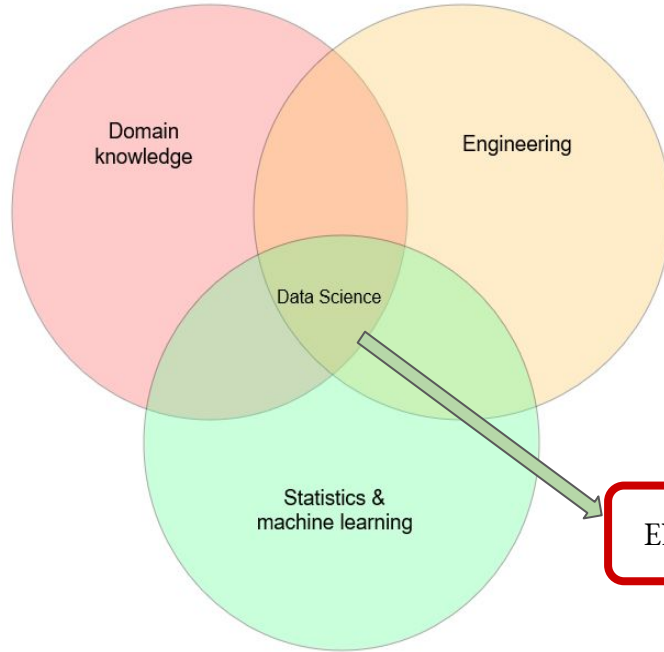
# Predictive Science and Modeling



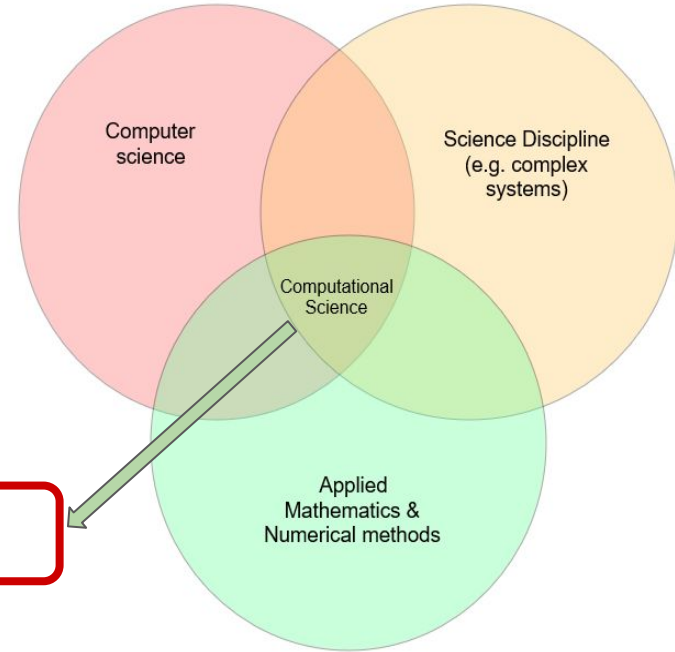


# Data Science / Computational Science

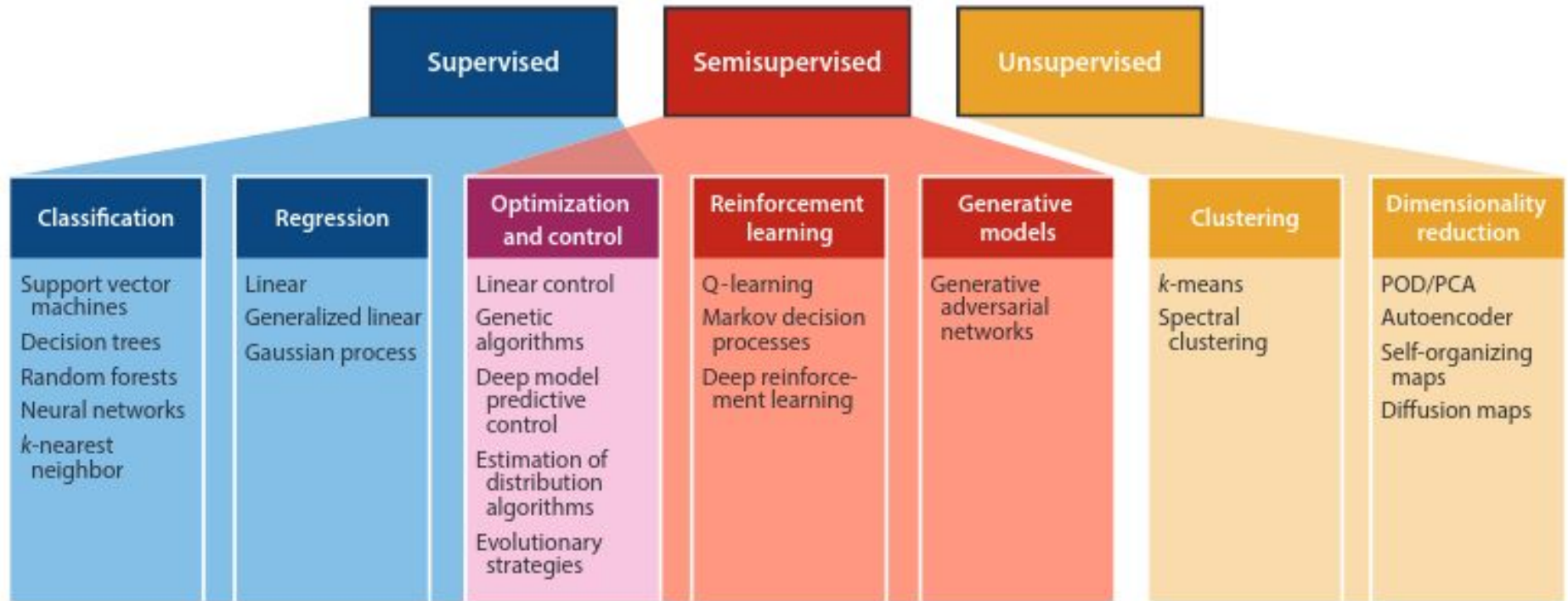
Data Science



Computational science

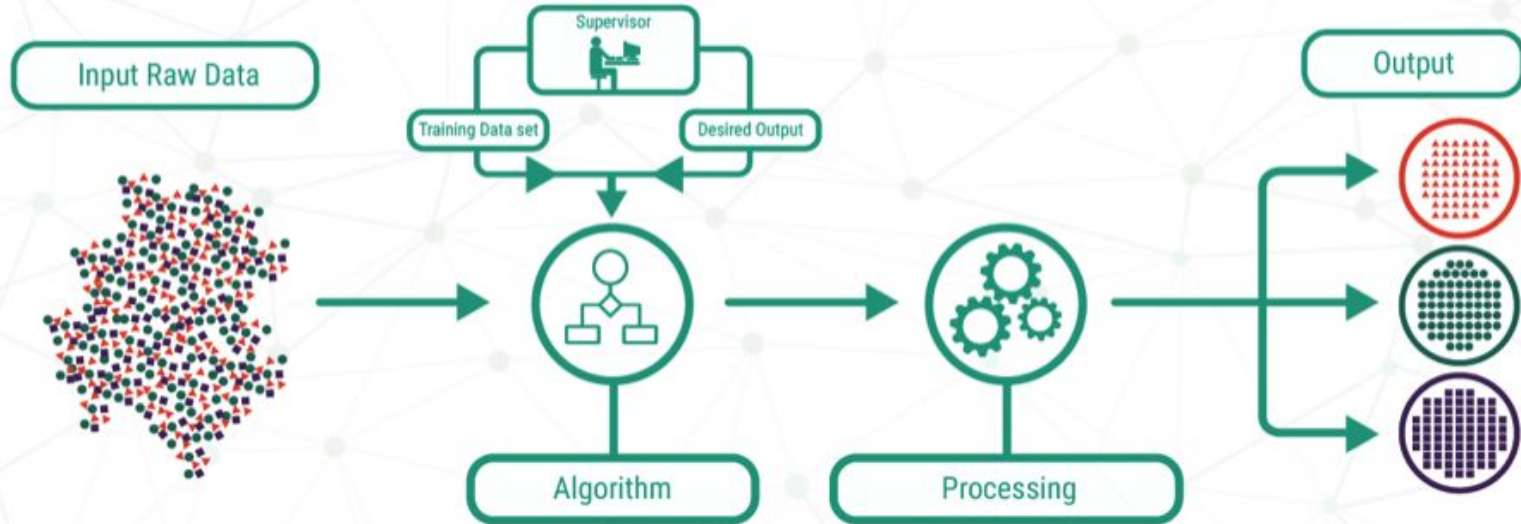


# Machine Learning Categories



# Supervised Learning

## SUPERVISED LEARNING

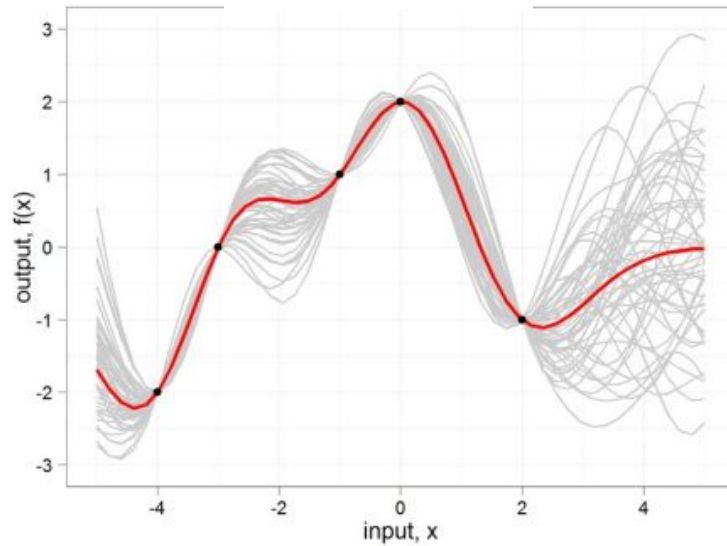




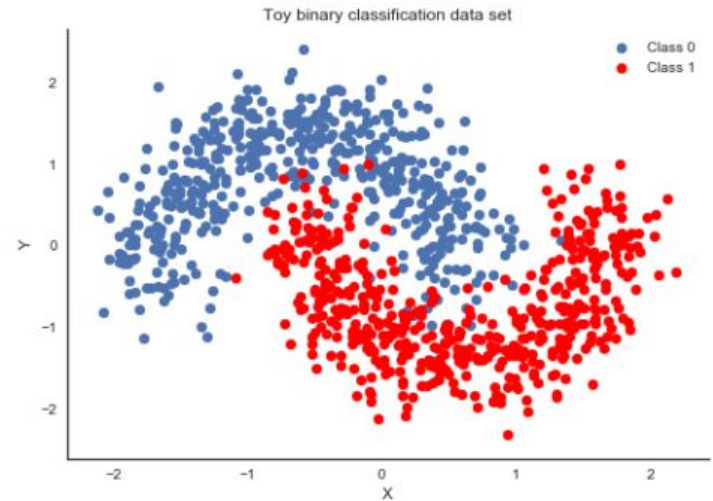
# Supervised Learning

## Supervised Learning

### Regression

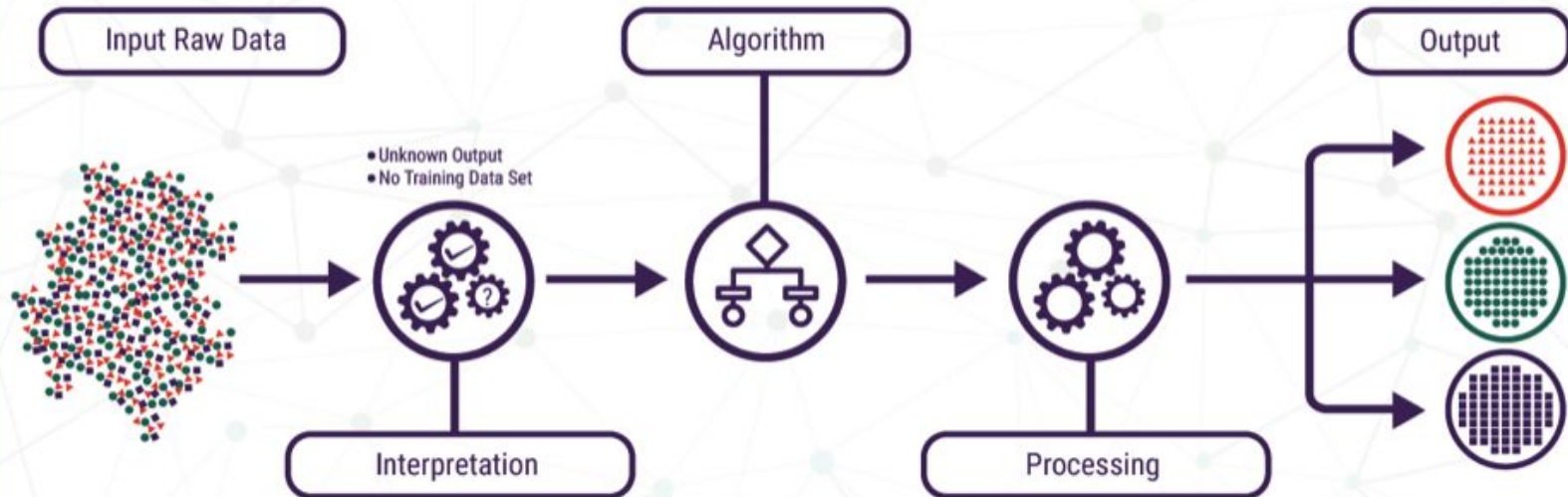


### Classification



# Unsupervised Learning

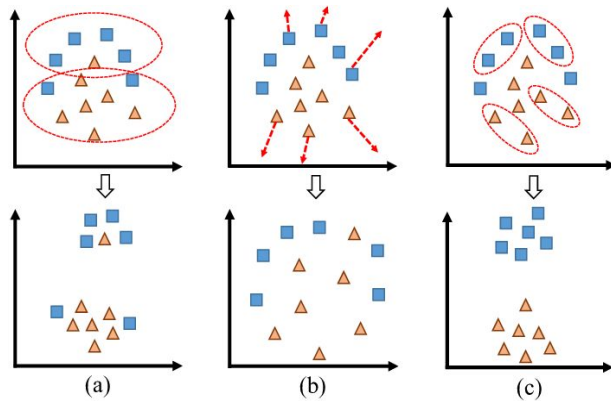
## UNSUPERVISED LEARNING



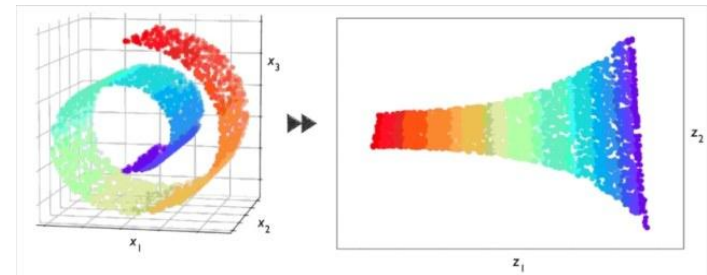
# Unsupervised Learning

## Unsupervised Learning

### Clustering

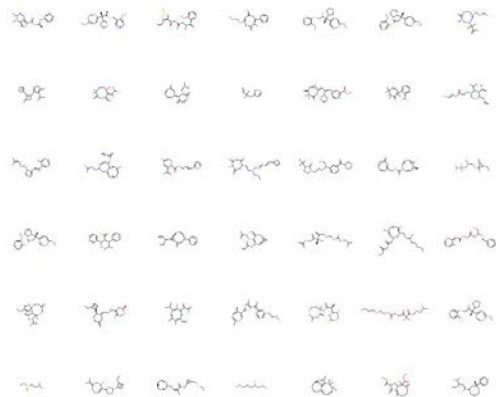
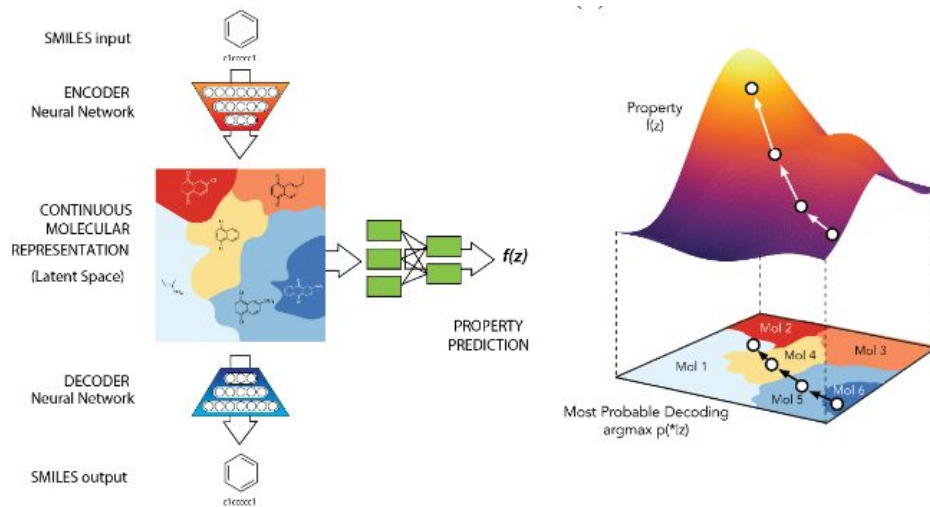


### Dimensionality Reduction

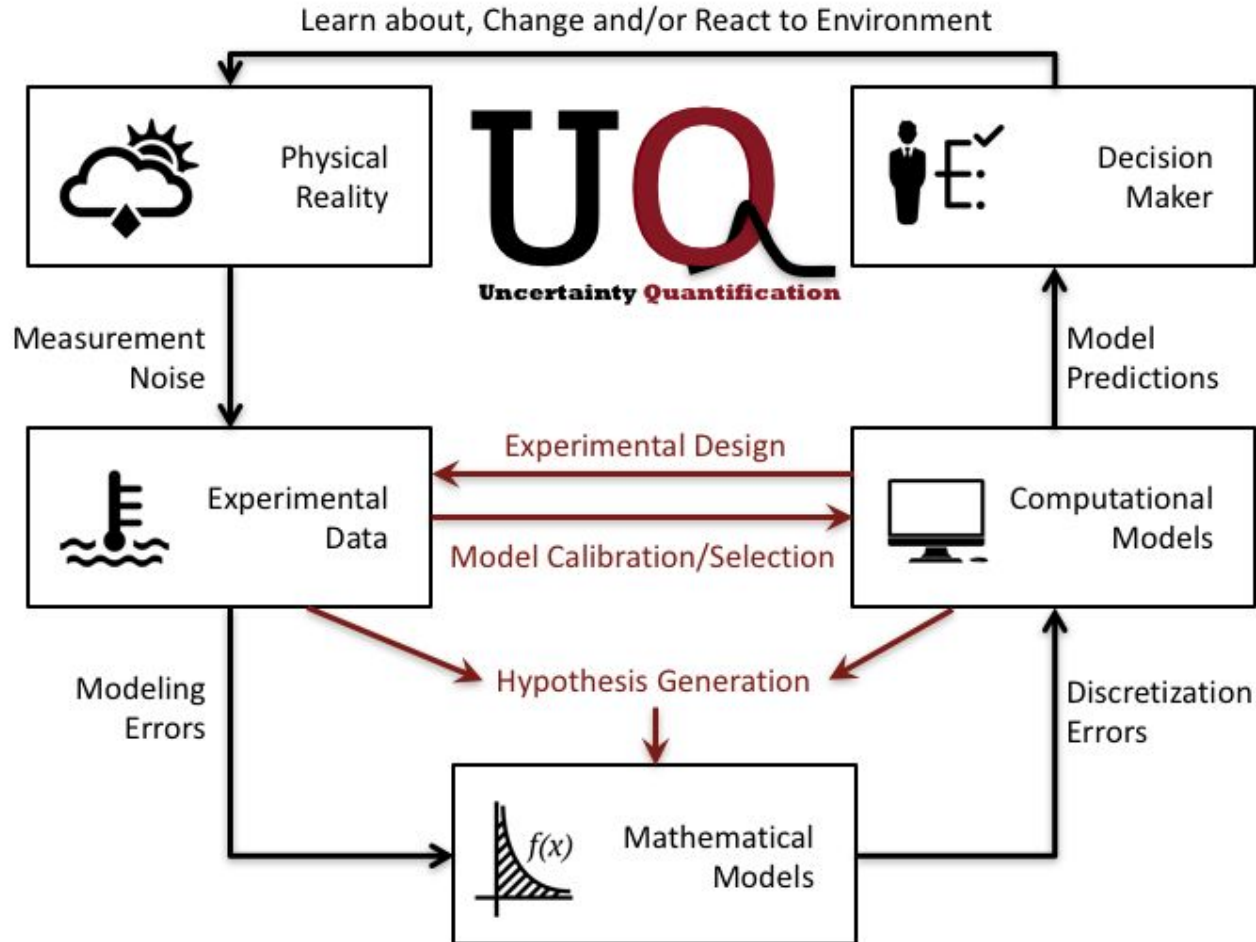




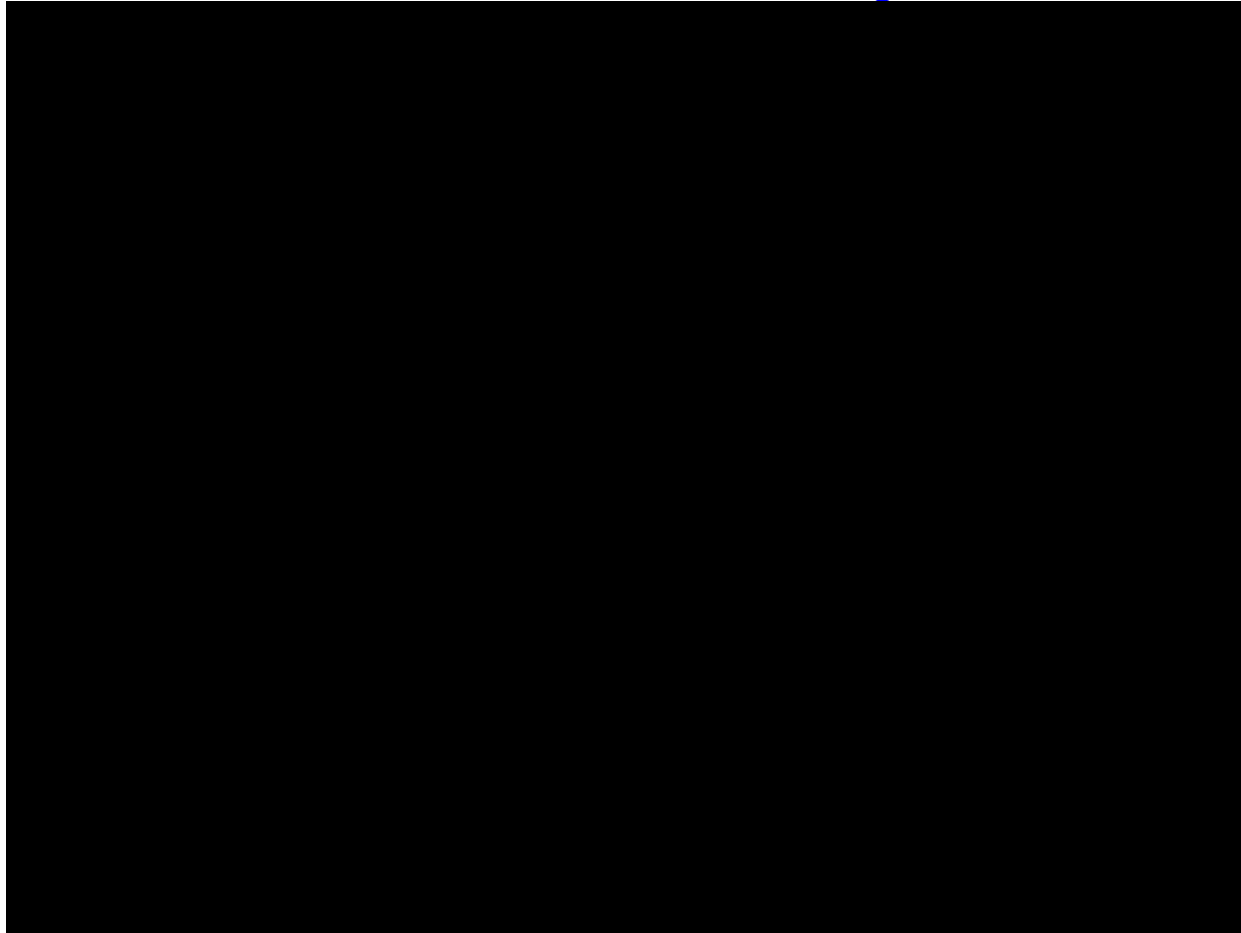
# Semi-supervised Learning (Generative Models)



# Uncertainty Quantification



## A motivational example



# Course goals

- Learning how to analyze and synthesize data towards enhancing their understanding and ability to model physical, biological, and engineering systems.
- Hands-on skills on contemporary machine learning tools enabling them to construct prediction models, extract patterns and characterize the statistical properties of data.
- Applications of these tools spanning a diverse set of engineering disciplines, including fluid dynamics, heat transfer, mechanical design, and biomedical engineering.

## **Key motifs**

- Representation
- Approximation
- Optimization
- Control



# Course Logistics

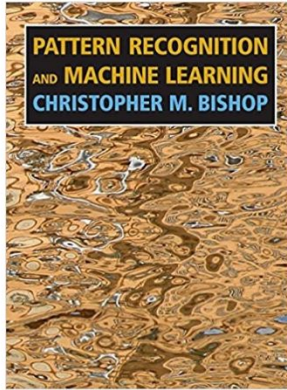
- Duration: 14 Weeks
- Time: Tuesdays and Thursdays, 3:00pm to 4:30pm.
- Dates: September 1 to December 10, 2020. No classes on Thursday, November 26.
- Office Hours: TBA (Thursdays, 4:30pm to 6:30pm)
- Zoom Lectures: <https://seas-upenn.zoom.us/j/99268155643>
- Zoom Office Hours: <https://seas-upenn.zoom.us/j/96811397184>
- Course Website (Detailed Description of the Course): <https://www.seas.upenn.edu/~enm360/>
- Github page (For course material, slides and codes):  
<https://github.com/PredictiveIntelligenceLab/ENM360>

# Course Outline

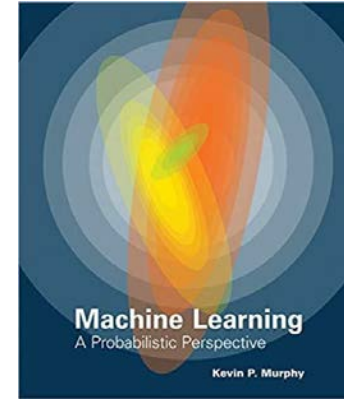
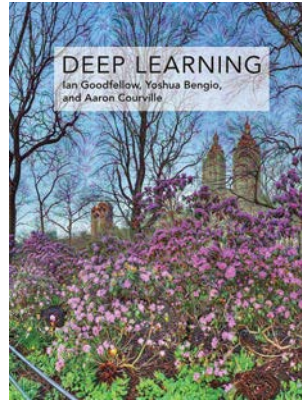
- Week 1: Course introduction, Python tutorial.
- Week 2: Primer on scientific computing: Linear systems, curve fitting, numerical differentiation and integration.
- Week 3: Primer on probability: random variables, distributions and moments, independence, Bayes rule.
- Week 4: Linear regression.
- Week 5: Logistic regression.
- Week 6: Density estimation.
- Week 7: Dimensionality reduction: Principal Component Analysis and Dynamic Mode Decomposition.
- Week 8: Sparse regression and compressive sensing.
- Week 9: Kalman filtering and state-space models (If time permits) .
- Week 10: Sampling and Monte Carlo inference.
- Week 11: Neural networks.
- Week 12: Kernel methods.
- Week 13: Bayesian optimization and active learning.
- Week 14: Final project presentations.

# Resources

## Textbook



Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.



# Reading

- The Emergence of Predictive Computational Science:
  - Computer predictions with quantified uncertainty ([Oden, Moser, & Ghattas, 2010](#))
  - [Lecture](#) by J.T Oden.
- Review papers on recent advances in machine learning:
  - Probabilistic machine learning and artificial intelligence ([Ghahramani, 2015](#))
  - Deep learning ([LeCun, Bengio, & Hinton, 2015](#))
  - Machine learning: Trends, perspectives, and prospects ([Jordan & Mitchell, 2015](#))
- Scientific computing in Python:
  - [Lectures and code](#) by Robert Johansson.

1. Oden, T., Moser, R., & Ghattas, O. (2010). Computer predictions with quantified uncertainty, part I. *SIAM News*, 43(9), 1–3.
2. Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553), 452–459.
3. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
4. Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255–260.

# Software

- A python 3 distribution configured for scientific computing. You can set this up either by [Anaconda](#) or [Virtual Environment](#).
- [JAX](#). Machine learning libraries from Google. Very close to Numpy library syntax, but very powerful.
- [Autograd](#). Efficient computed derivatives of Numpy code.
- [Jupyter Notebook](#). You will need this in order to follow some in-class tutorials.
- [Git](#). You will need this in order to download and stay in sync with the latest code we will develop in class
- [Spyder](#). An useful IDE for programming in Python.



# Python Interpreter

In order to use the Python programming language you will need to write a Python script and then invoke the Python interpreter. After you are done with your program, you can pass it as an argument to the interpreter, who will read and execute your set of commands.

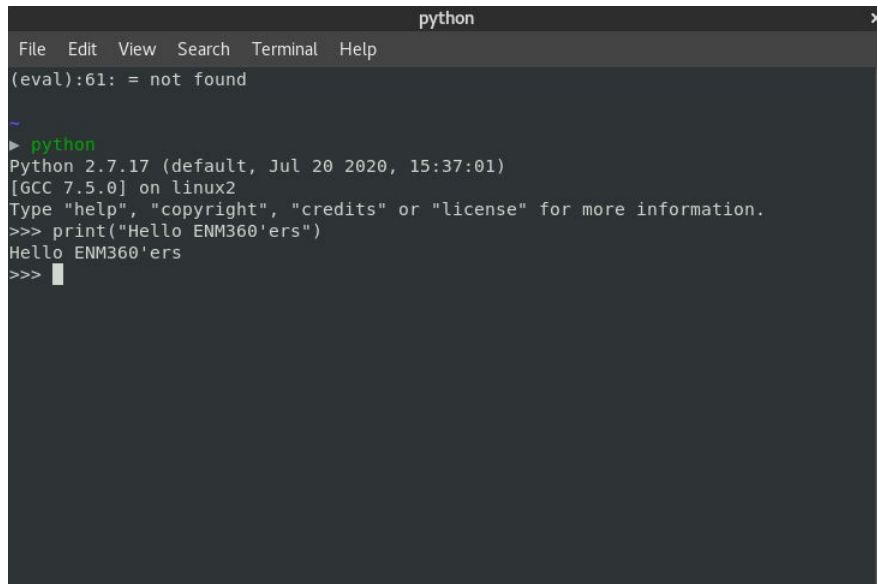
The Python interpreter uses whitespace indentation to determine which pieces of code are grouped together in a special way—e.g., as part of a function, loop, or class. How much space is used is not typically important, as long as it is consistent. If two spaces are used to indent the first time, two spaces should be used to indent subsequently.

Run the your program *my-program.py* in the terminal using the interpreter:

```
python my-program.py
```

Or start the interpreter in the interactive terminal mode:

NOT ALWAYS USEFUL TO USE THE  
INTERPRETER IN SUCH WAY!!!

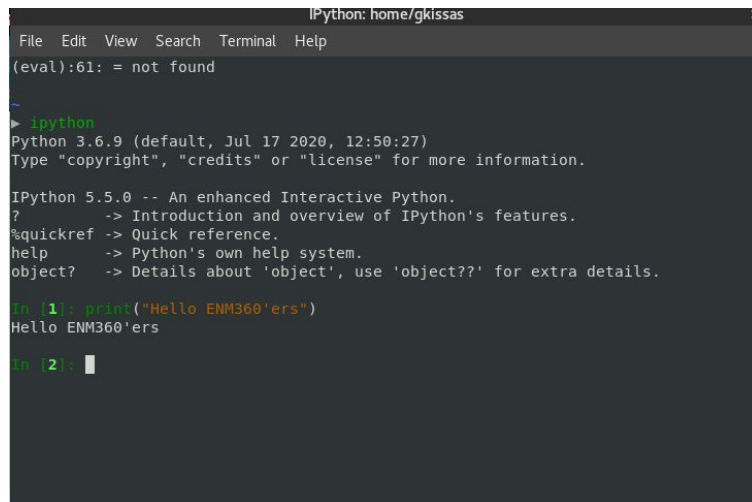


```
python
File Edit View Search Terminal Help
(eval):61: = not found

~
> python
Python 2.7.17 (default, Jul 20 2020, 15:37:01)
[GCC 7.5.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello ENM360'ers")
Hello ENM360'ers
>>> █
```

# IPython

IPython is an interactive Python shell that provides great user friendly futures. IPython shell is often considered as a work-horse in scientific code development.



```
IPython: home/gkissas
File Edit View Search Terminal Help
(eval):61: = not found

~
> ipython
Python 3.6.9 (default, Jul 17 2020, 12:50:27)
Type "copyright", "credits" or "license" for more information.

IPython 5.5.0 -- An enhanced Interactive Python.
?      -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help    -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]: print("Hello ENM360'ers")
Hello ENM360'ers

In [2]:
```

Some of the useful features are the following:

- Command history, which can be browsed with the up and down arrow keys.
- Tab auto-completion.
- In-line code editing.
- Object introspection and automatic extract of documentation strings from Python objects like classes and functions.
- Good interaction with operating system shell.
- Support for multiple parallel back-end processes, that can run on computing clusters or cloud services.

# Jupyter Notebook

jupyter jupyter\_tutorial Last Checkpoint: a few seconds ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

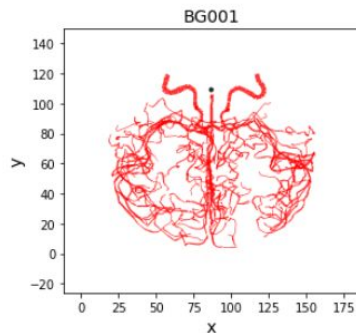
Python 3.6.10 64-bit ('brain': conda)

Code

```
In [1]: import neurom as nm
        from neurom.core import Tree
        from neurom import morphmath as mm
        from neurom import viewer
```

```
In [2]: nrn = nm.load_neuron('BG001.swc')
        fig, ax = viewer.draw(nrn)
        fig.show()
```

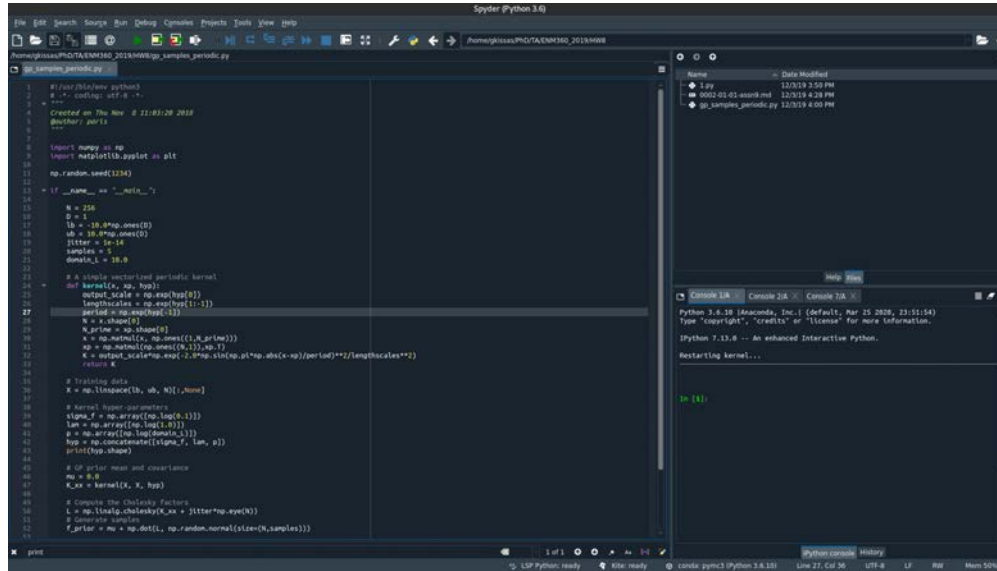
/home/gkissas/anaconda3/envs/brain/lib/python3.6/site-packages/ipykernel\_launcher.py:3: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend\_inline, which is a non-GUI backend, so cannot show the figure.  
This is separate from the ipykernel package so we can avoid doing imports until



In [ ]:

# Spyder

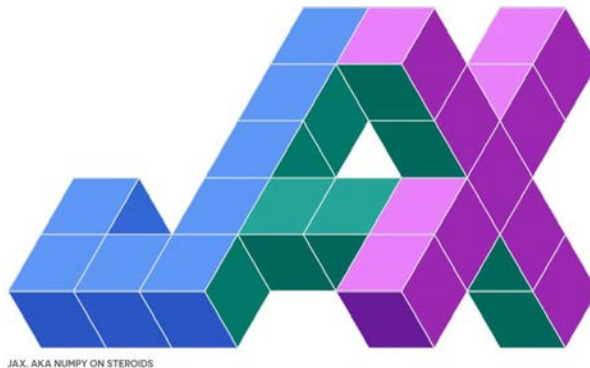
[Spyder](#) is a MATLAB-like IDE for scientific computing for Python, written in Python. It has the advantage that code developing, execution and debugging is carried out in a single environment, and work on different calculations can be organized in the IDE environment.



Some of the useful features of Spyder:

- Syntax highlighting, dynamic code introspection and integration with the python debugger.
- Variable explorer, IPython command prompt.
- Integrated documentation and help.





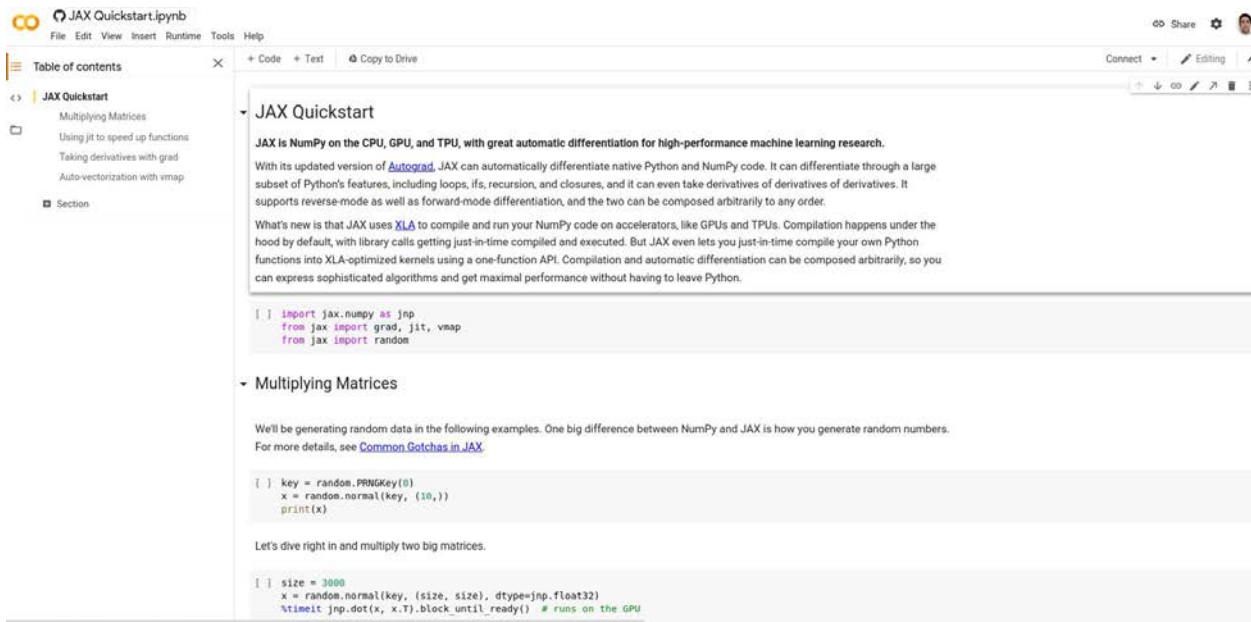
## **Resources:**

<https://colab.research.google.com/github/google/jax/blob/master/docs/notebooks/quickstart.ipynb>

<https://colinraffel.com/blog/you-don-t-know-jax.html>

<https://iaml.it/blog/jax-intro-english>

# GOOGLE COLAB



The screenshot shows a Google Colab notebook interface. The top bar includes the Google Colab logo, the notebook title 'JAX Quickstart.ipynb', and menu options: File, Edit, View, Insert, Runtime, Tools, and Help. On the right, there are icons for sharing, settings, and a user profile. Below the top bar, a 'Table of contents' sidebar is visible on the left, listing sections like 'JAX Quickstart', 'Multiplying Matrices', 'Using jit to speed up functions', 'Taking derivatives with grad', 'Auto-vectorization with vmap', and 'Section'. The main content area displays the 'JAX Quickstart' section, which includes introductory text about JAX's capabilities and two code blocks. The first code block imports JAX libraries, and the second code block demonstrates matrix multiplication with timing and GPU acceleration.

JAX Quickstart

JAX is NumPy on the CPU, GPU, and TPU, with great automatic differentiation for high-performance machine learning research.

With its updated version of [Autograd](#), JAX can automatically differentiate native Python and NumPy code. It can differentiate through a large subset of Python's features, including loops, ifs, recursion, and closures, and it can even take derivatives of derivatives of derivatives. It supports reverse-mode as well as forward-mode differentiation, and the two can be composed arbitrarily to any order.

What's new is that JAX uses [XLA](#) to compile and run your NumPy code on accelerators, like GPUs and TPUs. Compilation happens under the hood by default, with library calls getting just-in-time compiled and executed. But JAX even lets you just-in-time compile your own Python functions into XLA-optimized kernels using a one-function API. Compilation and automatic differentiation can be composed arbitrarily, so you can express sophisticated algorithms and get maximal performance without having to leave Python.

```
[ ] import jax.numpy as jnp
    from jax import grad, jit, vmap
    from jax import random
```

Multiplying Matrices

We'll be generating random data in the following examples. One big difference between NumPy and JAX is how you generate random numbers. For more details, see [Common Gotchas in JAX](#).

```
[ ] key = random.PRNGKey(0)
    x = random.normal(key, (10,))
    print(x)
```

Let's dive right in and multiply two big matrices.

```
[ ] size = 1000
    x = random.normal(key, (size, size), dtype=jnp.float32)
    %timeit jnp.dot(x, x.T).block_until_ready() # runs on the GPU
```

## Homework submission:

- Submit your homework on Google Colab .
- You will have to include Python scripts, figures, latex text with reasoning and explanations of your answers.
- JAX does not work on Windows!!! Get familiar with Colab!!!!



**WELCOME TO ENM360!!!**