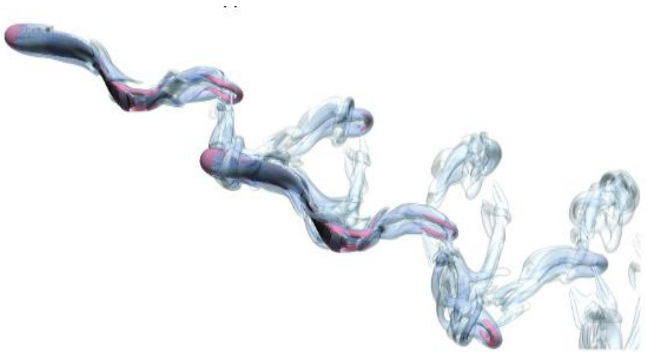
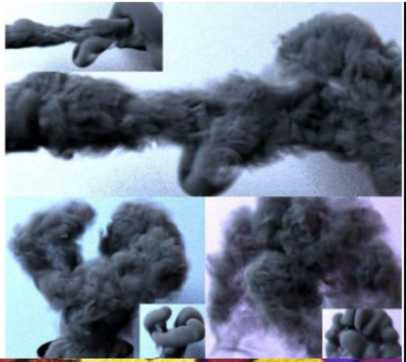
ENM 360: Introduction to Data-driven Modeling

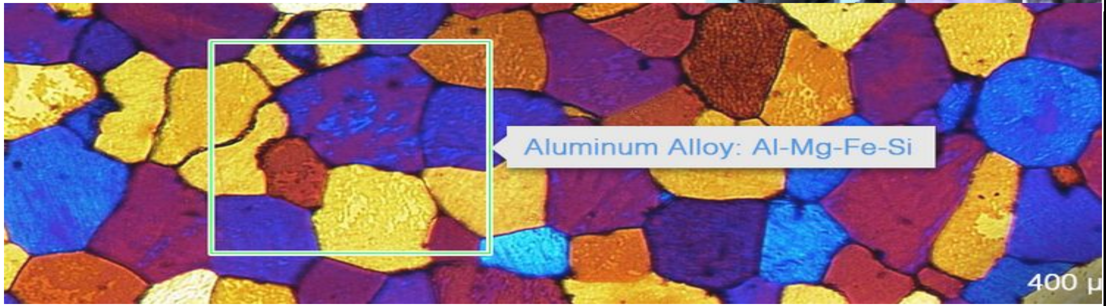
Lecture #1: Introduction, motivation and course logistics

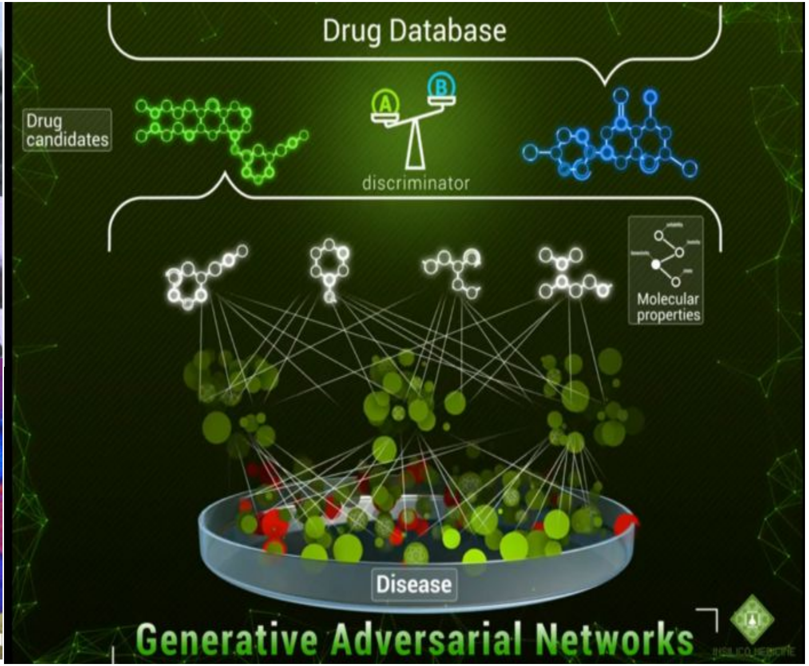
Paris Perdikaris
September 1, 2020

UNIVERSITY of PENNSYLVANIA



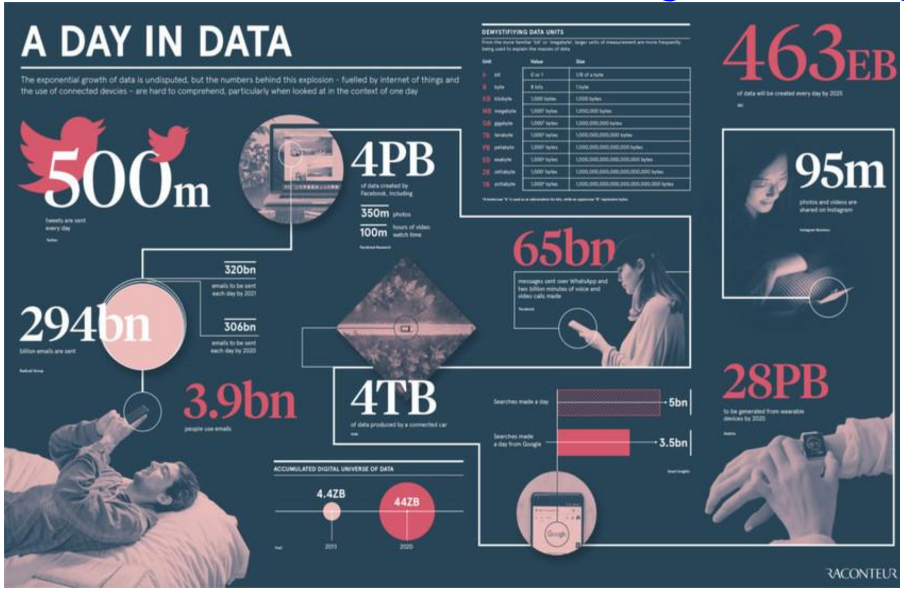


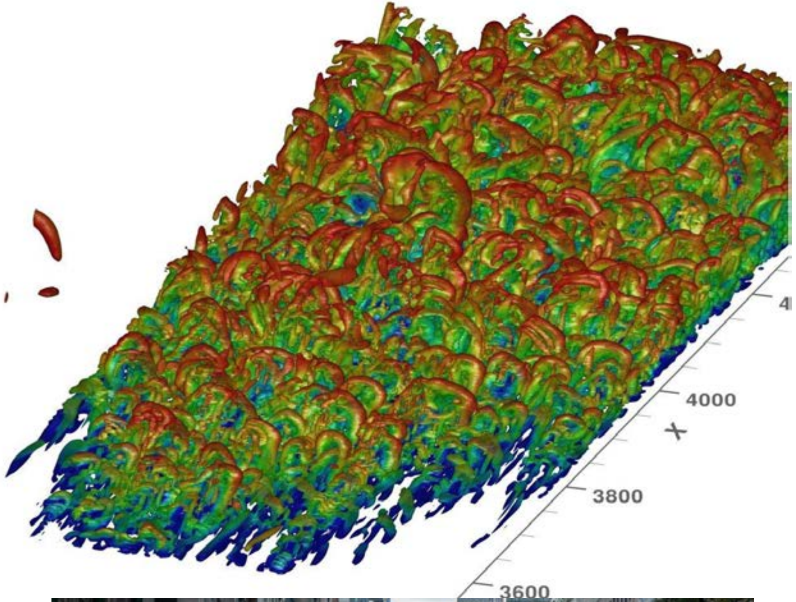


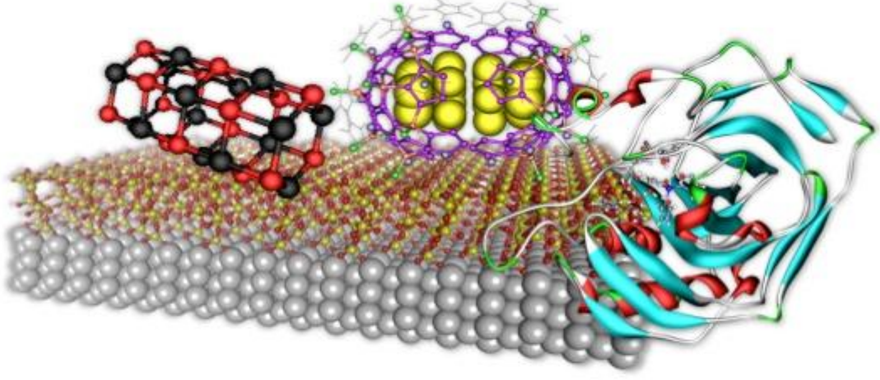


From recognizing voice, text or images to designing more efficient airplane wings and discovering new drugs, machine learning is introducing a transformative set of tools in data analysis with increasing impact across engineering, sciences, and commercial applications. In this course, you will learn about principles and algorithms for extracting patterns from data and 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.

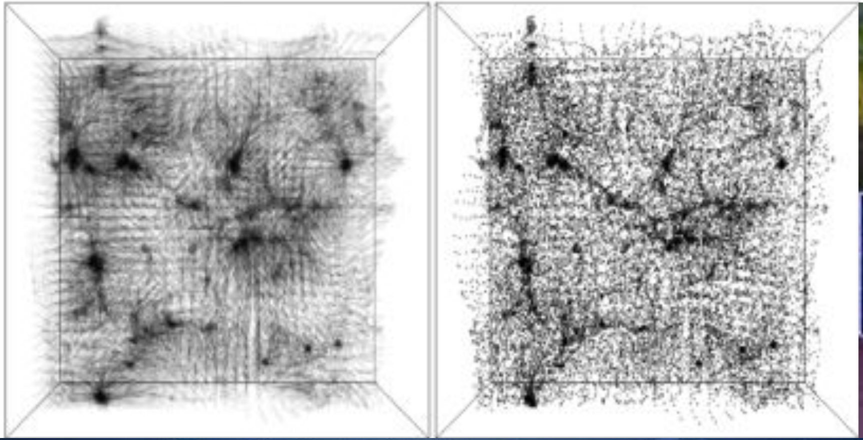
Living in the Age of Big Data

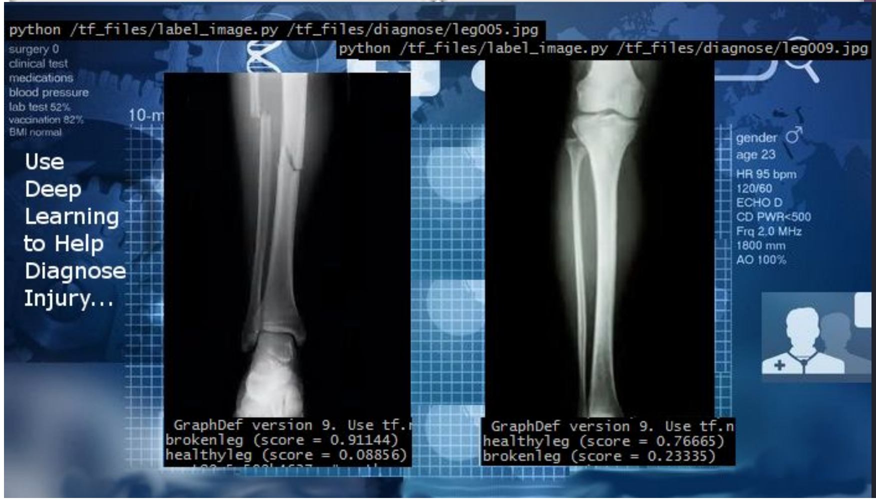


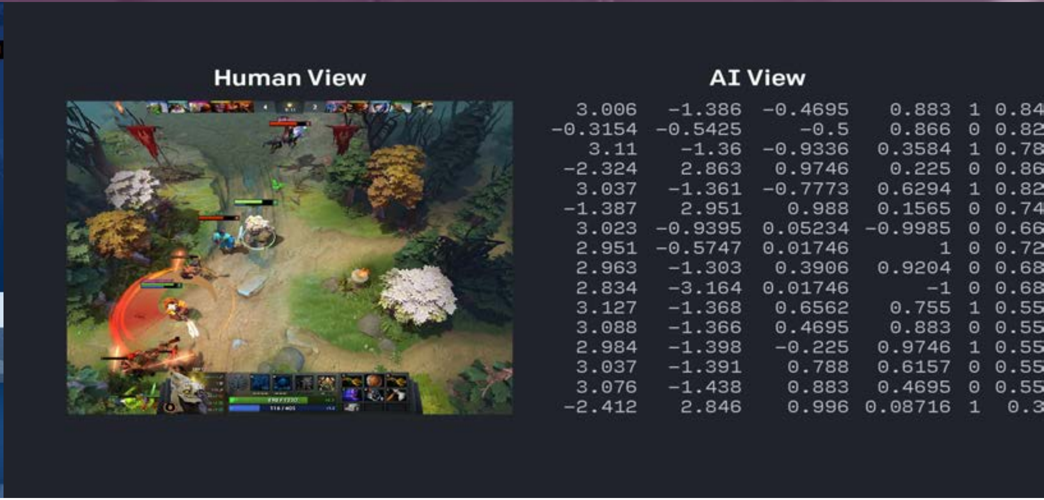




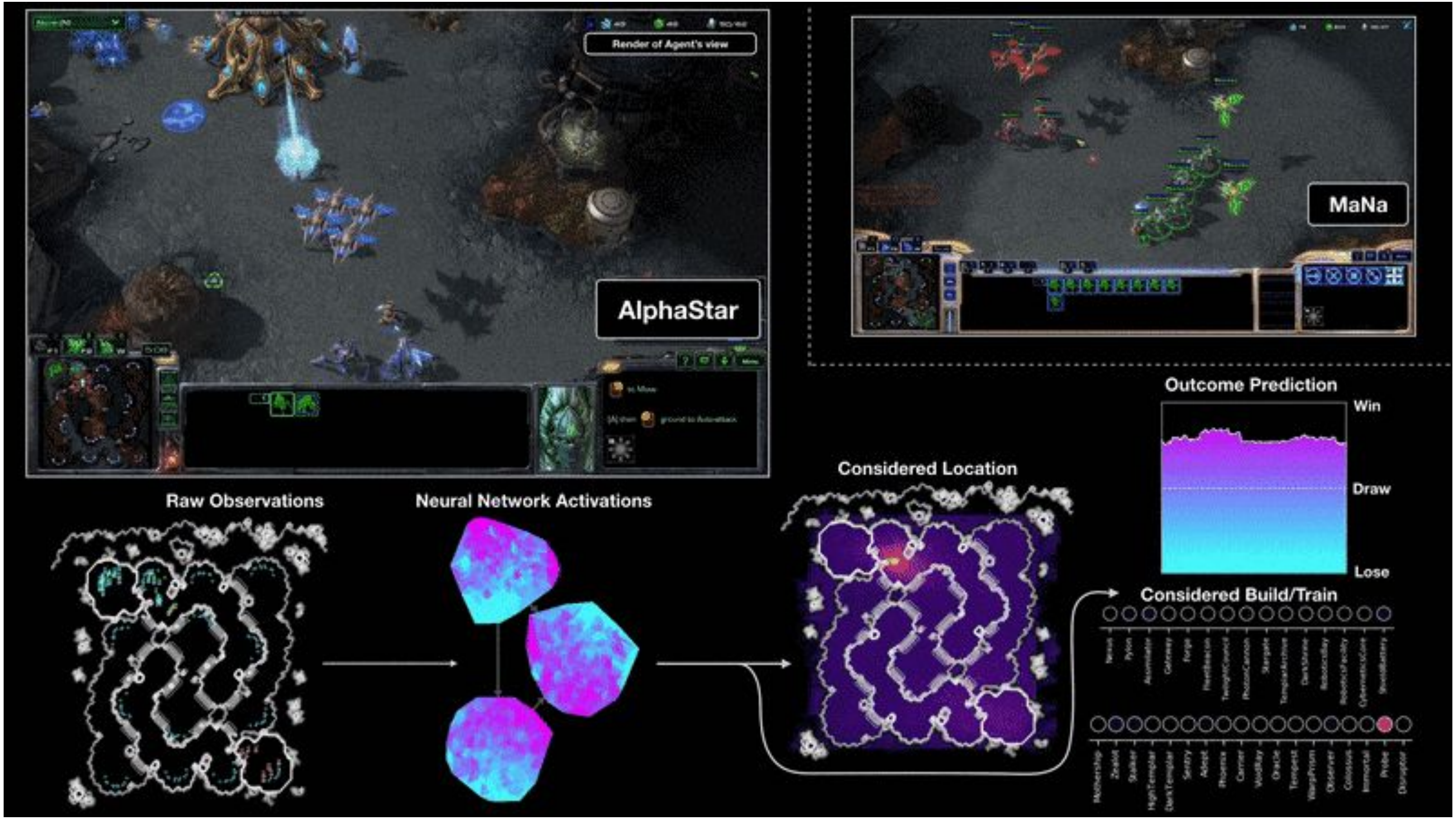
Successes of Deep Learning in Different Fields







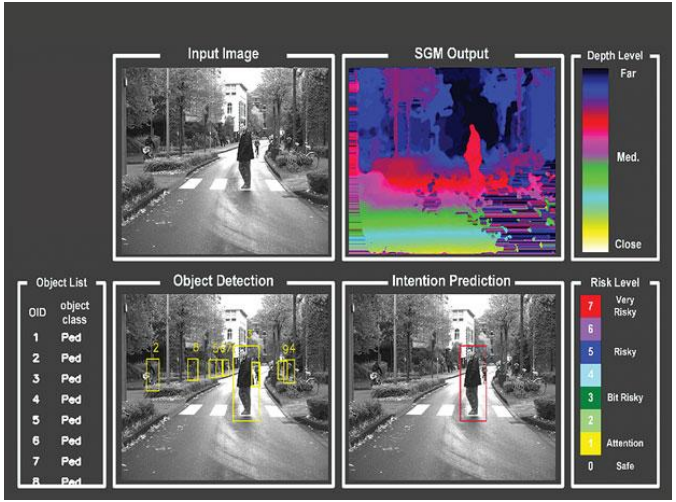
Working with high dimensional data

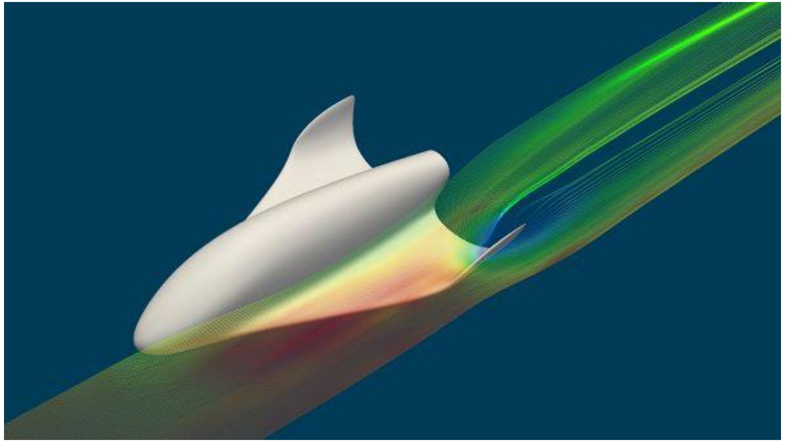


Working with high dimensional data

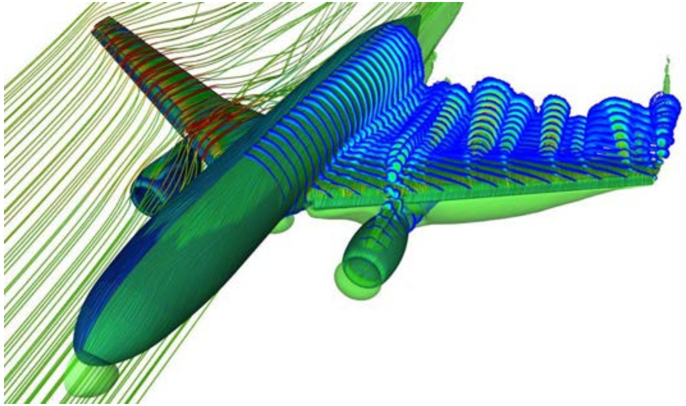
Working with high dimensional data

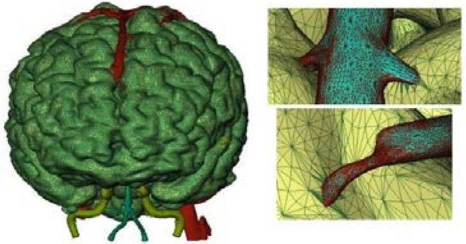
Motivation and Addressing open challenges





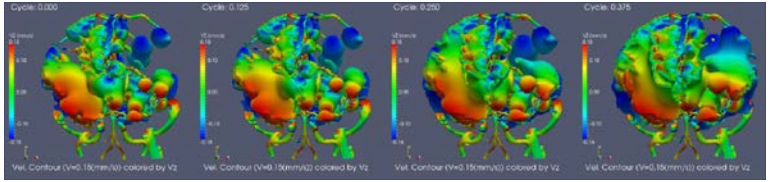
Computational Science and Engineering

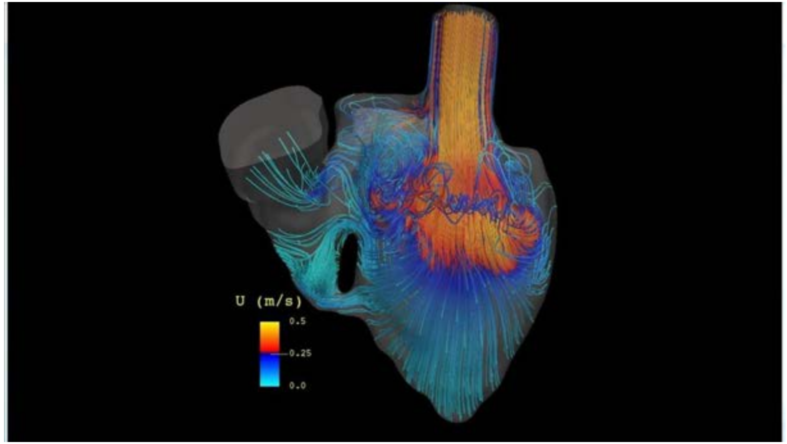


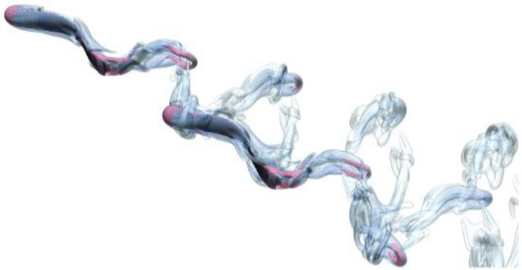


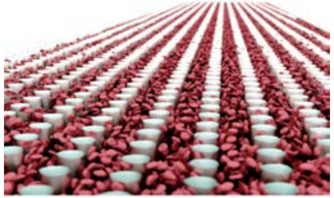
(a) Triangular Meshes
(overall view)

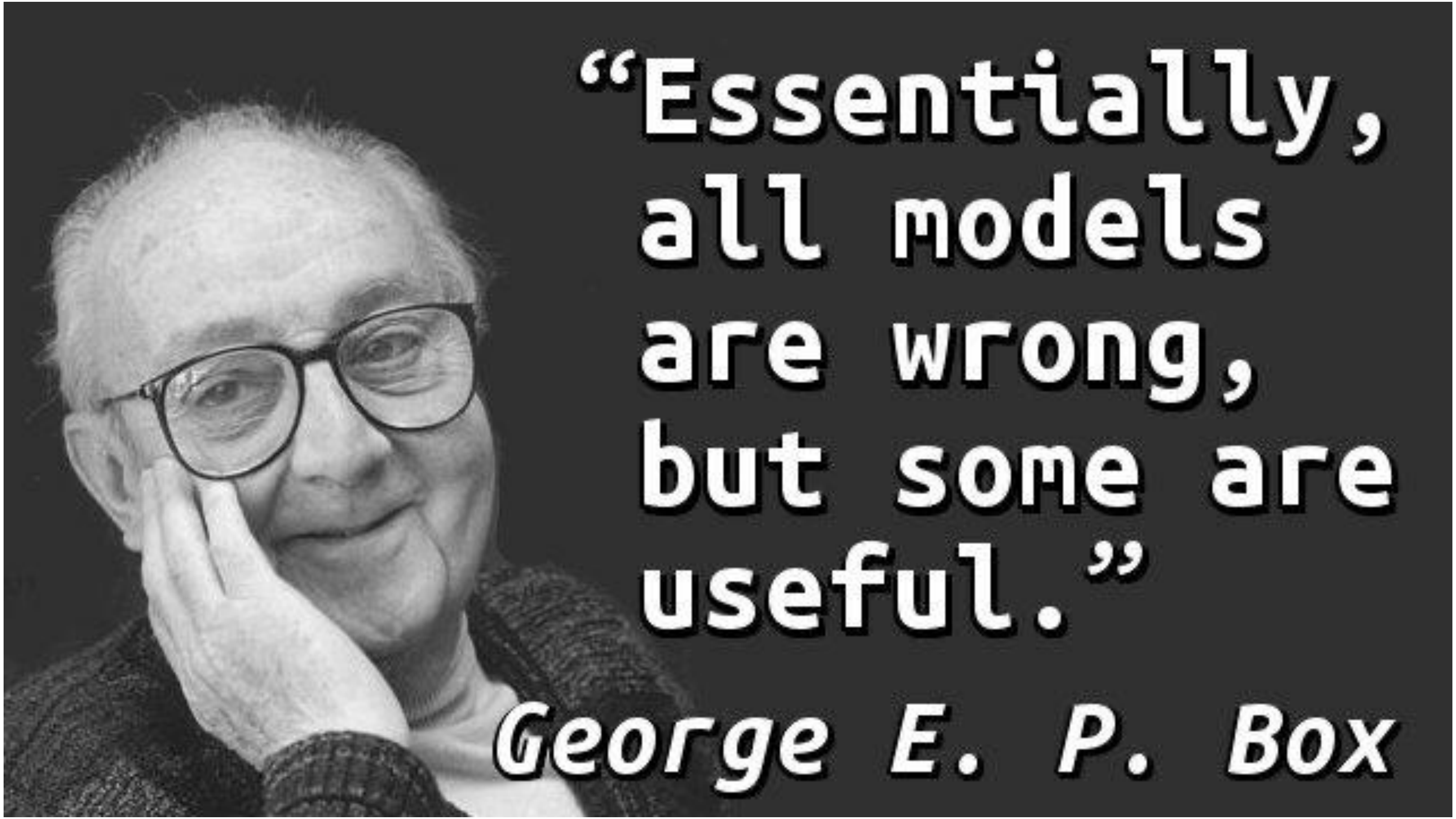
(b) Triangular meshes
(enlarged view)



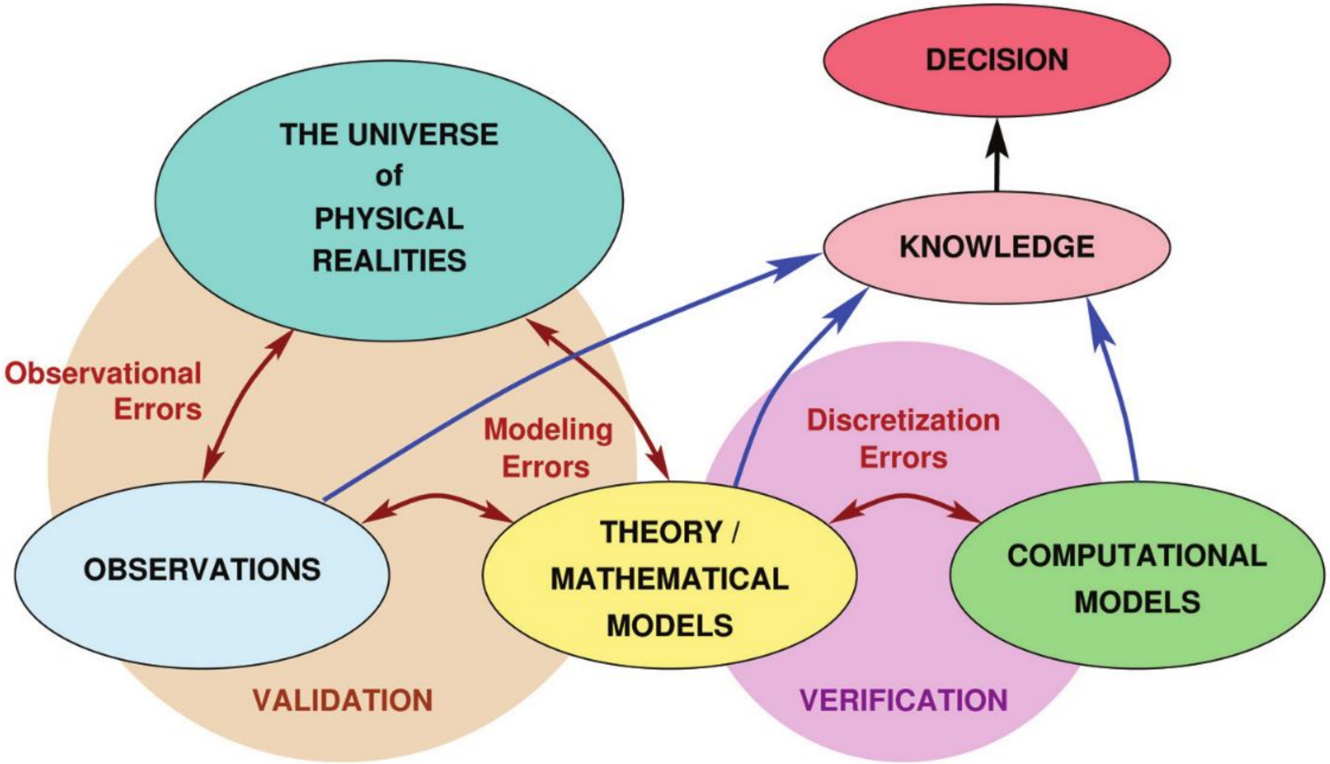




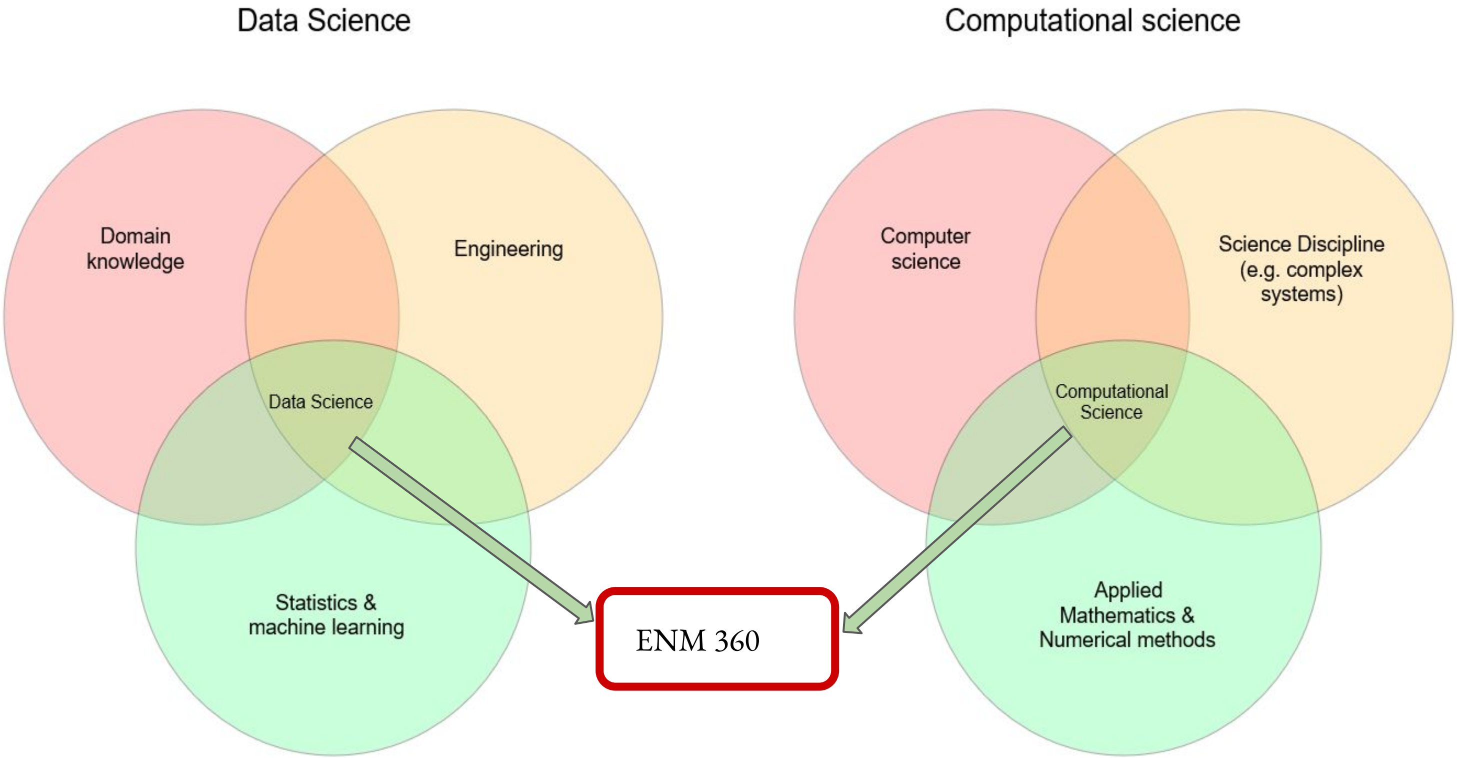



"Essentially, all models are wrong, but some are useful."
George E. P. Box

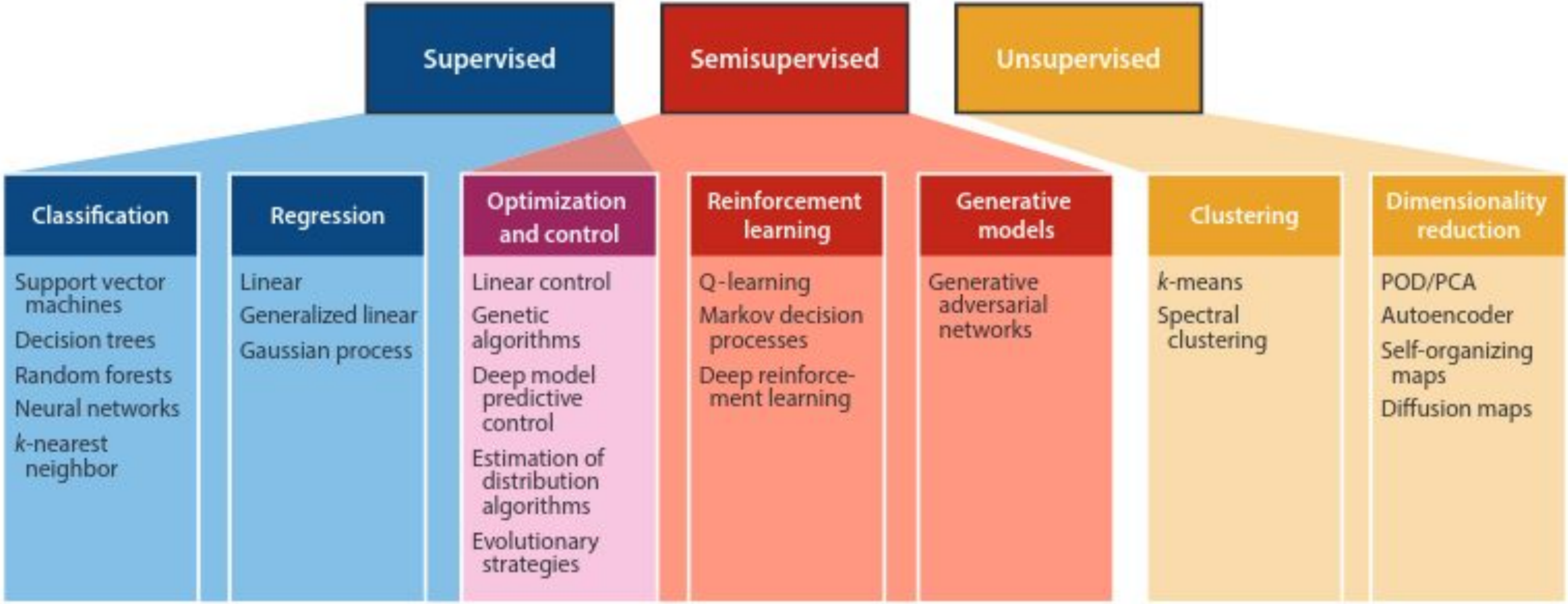
Predictive Science and Modeling



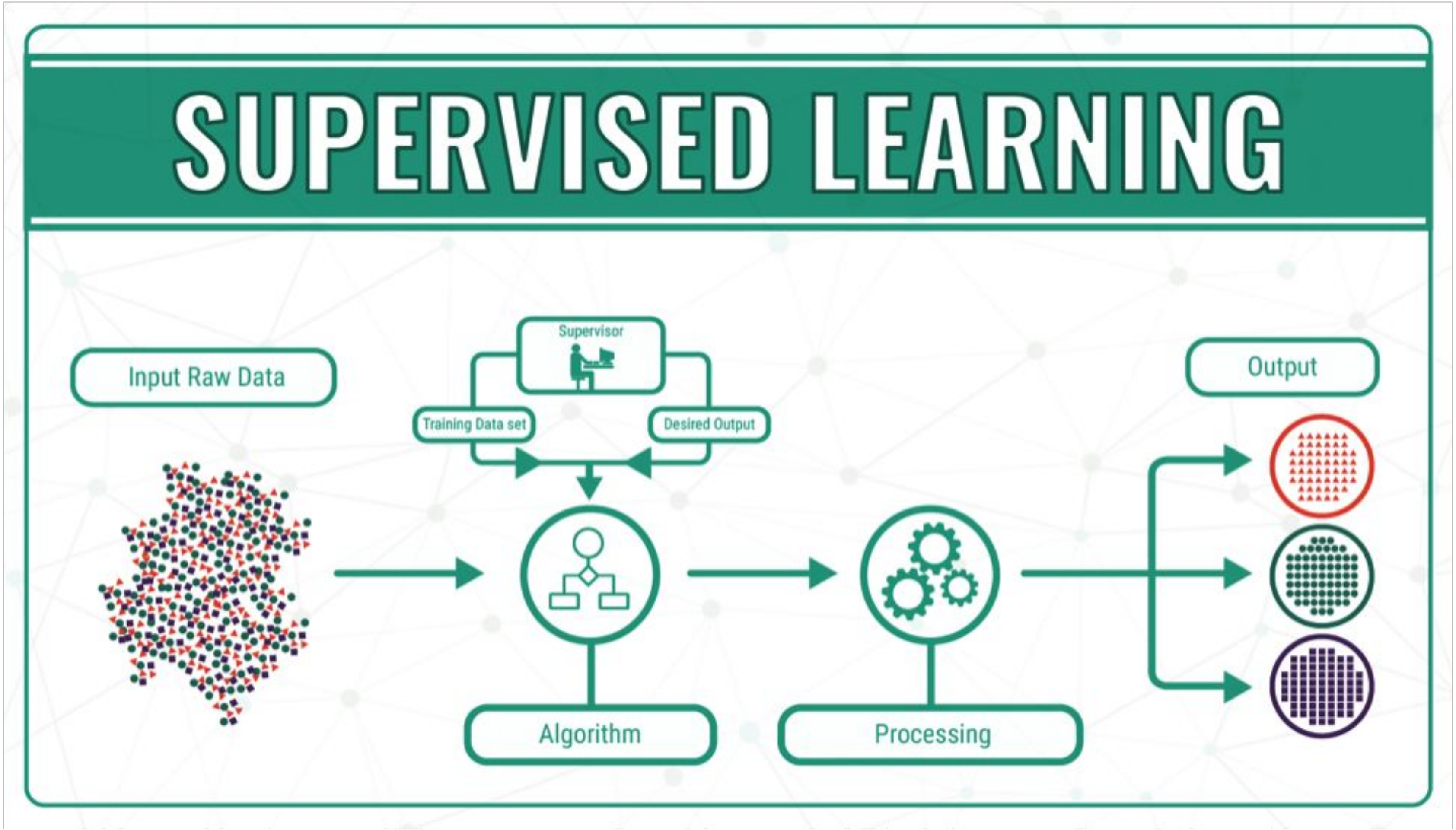
Data Science / Computational Science



Machine Learning Categories



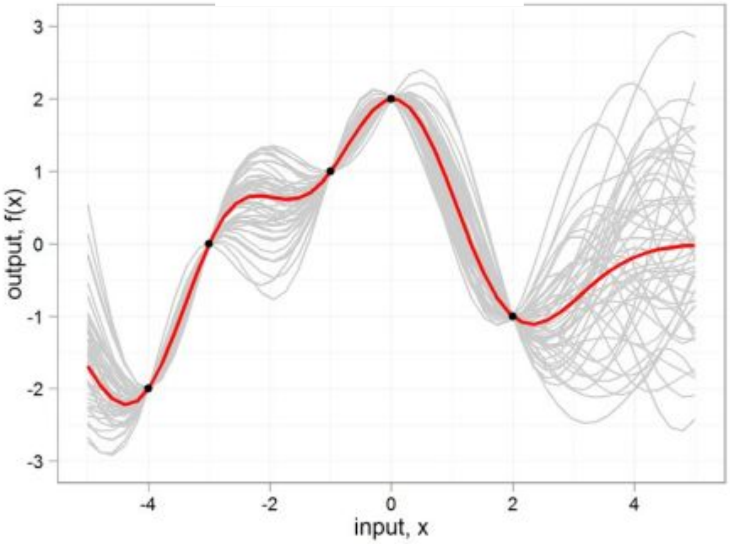
Supervised Learning



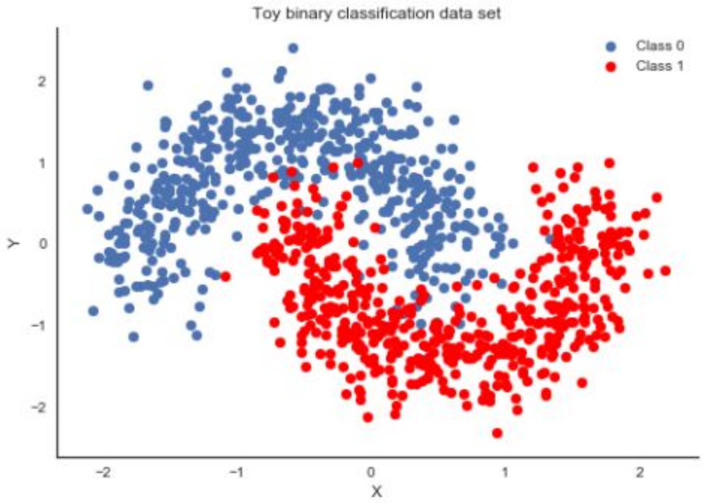
Supervised Learning

Supervised Learning

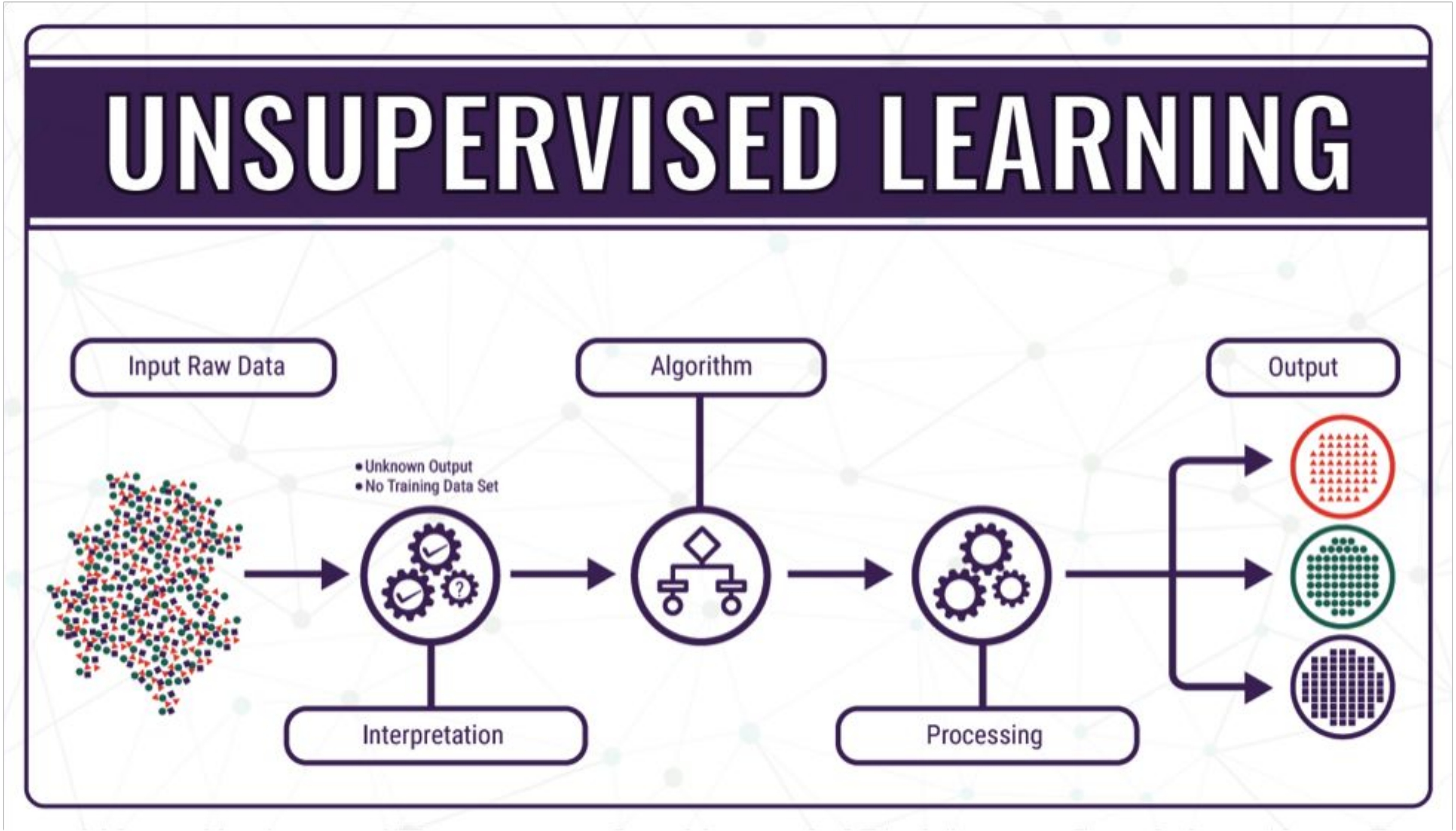
Regression



Classification



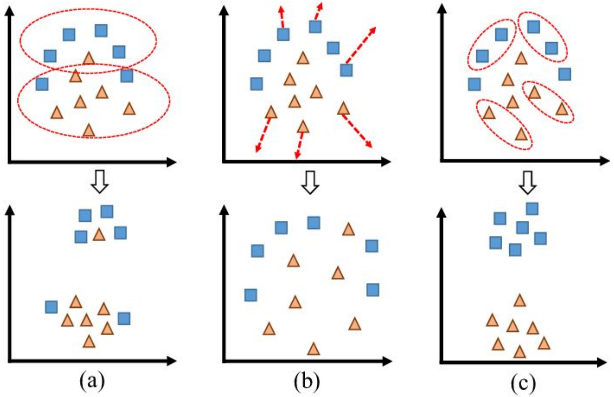
Unsupervised Learning



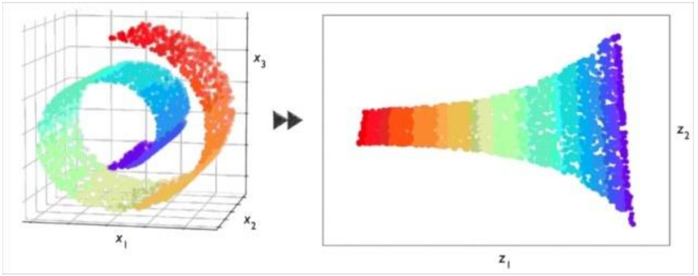
Unsupervised Learning

Unsupervised Learning

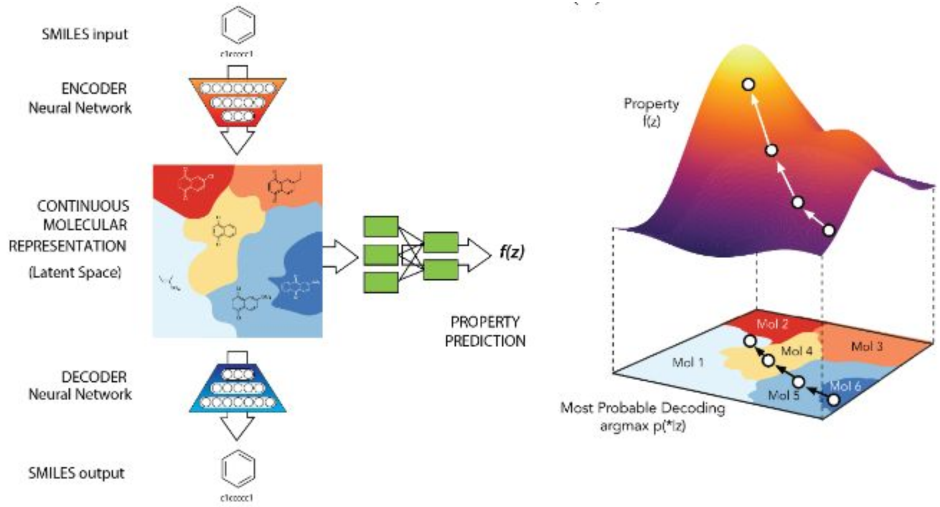
Clustering

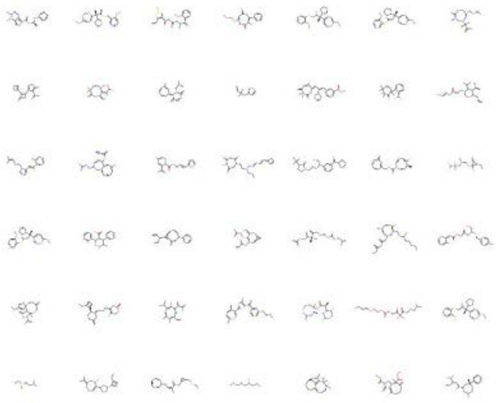


Dimensionality Reduction

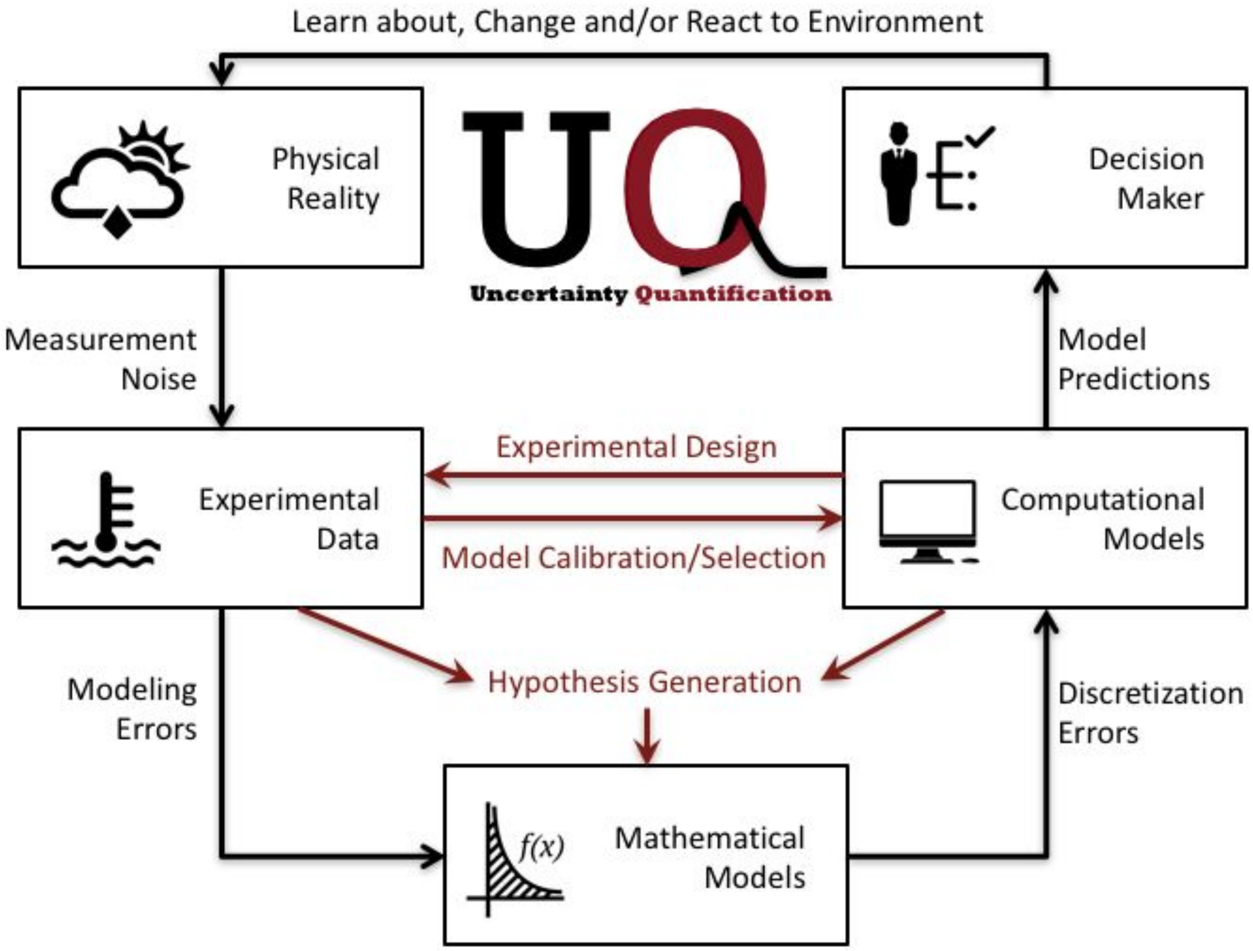


Semi-supervised Learning (Generative Models)





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

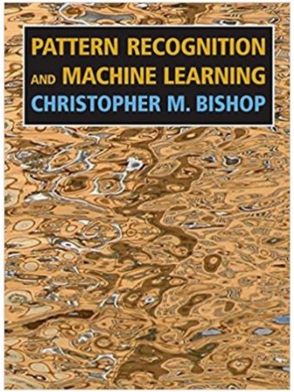
- Duration: 14 Weeks
- Time: Tuesdays and Thursdays, 3:00pm to 4:30pm.
- Dates: September 1 to December 10, 2020. No classes on Thursday, November 26.
- Office Hours: TBA (Thursdays, 4:30pm to 6:30pm)
- Zoom Lectures: https://seas-upenn.zoom.us/j/99268155643
- Zoom Office Hours: https://seas-upenn.zoom.us/j/96811397184
- 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

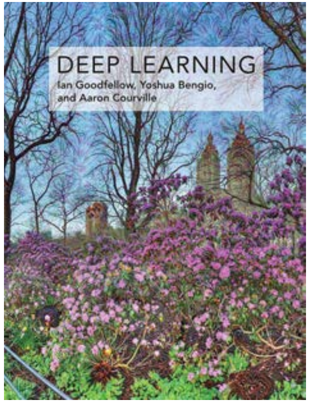
- Week 1: Course introduction, Python tutorial.
- Week 2: Primer on scientific computing: Linear systems, curve fitting, numerical differentiation and integration.
- Week 3: Primer on probability: random variables, distributions and moments, independence,Bayes rule.
- Week 4: Linear regression.
- Week 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.

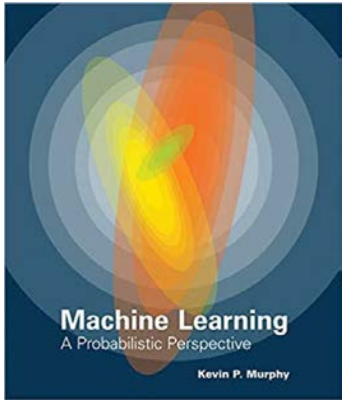
Resources

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

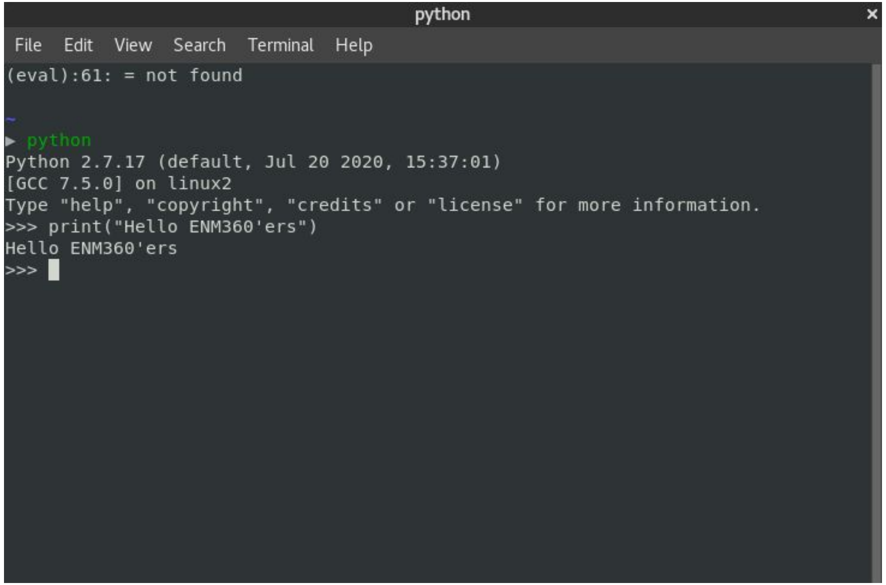
In order to use the Python programming language you will need to write a Python script and then invoke the Python interpreter. After you are done with your program, you can pass it as an argument to the interpreter, who will read and execute your set of commands.

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
```

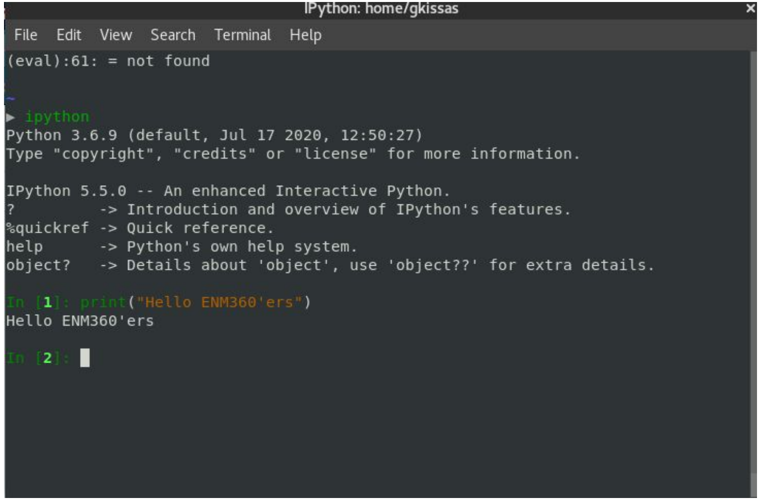
Or start the interpreter in the interactive terminal mode:



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

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.



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

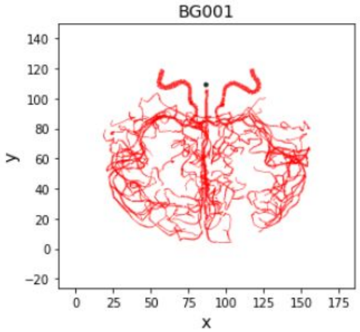
File Edit View Insert Cell Kernel Help Trusted Python 3.6.10 64-bit ('brain': conda)

Run Code

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

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

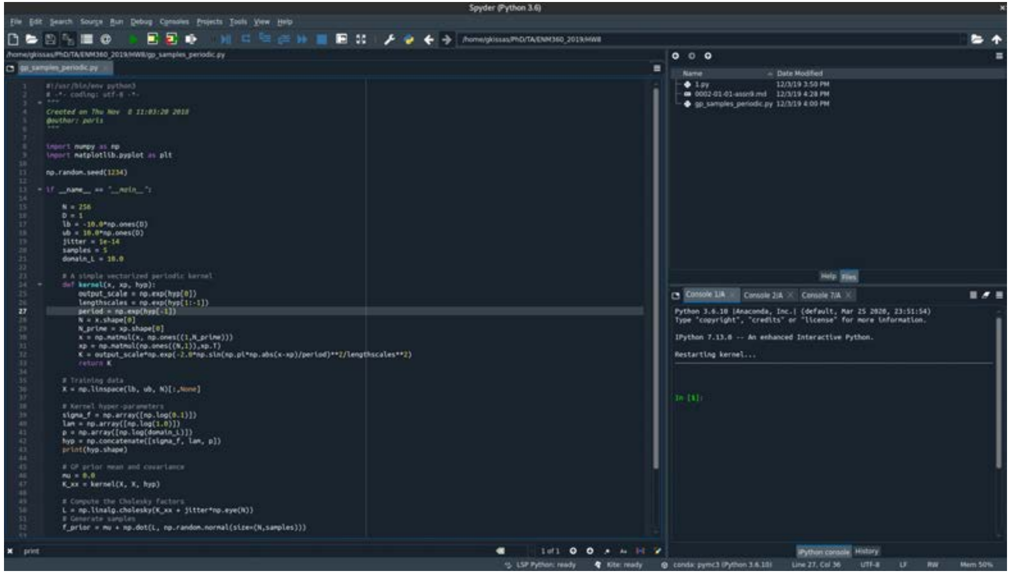
```
/home/gkissas/anaconda3/envs/brain/lib/python3.6/site-packages/ipykernel_launcher.py:3: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
  This is separate from the ipykernel package so we can avoid doing imports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

JAX, AKA NUMPY ON STEROIDS

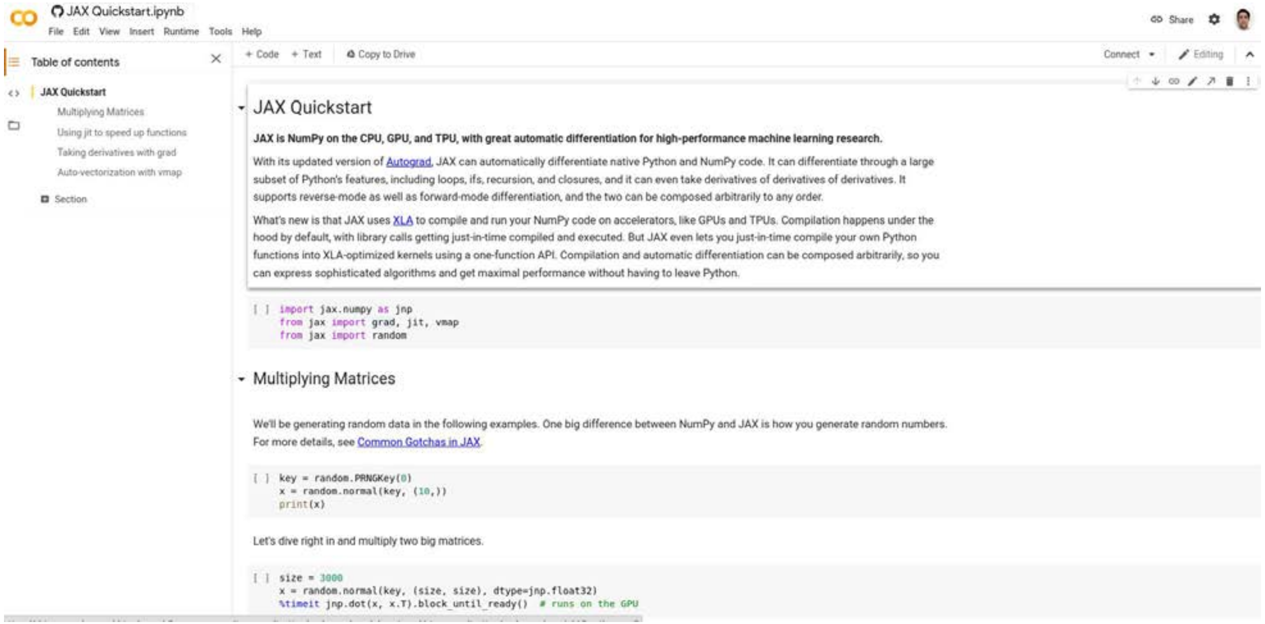
Resources:

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



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!!!