What is Machine Learning (ML) ?

- prediction

What is Machine Learning (ML) ?

- **informed** prediction
    - prediction is better than random guess
    - method: learn from existing data
    - goal: Generalization. Predicting on new, unseen data

Where is ML used ?

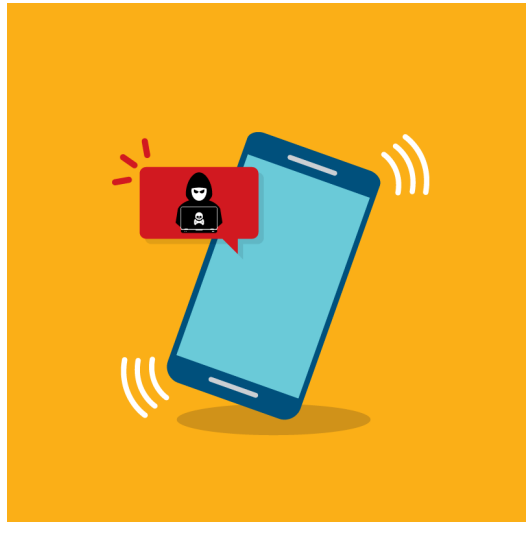
Everywhere !!

- targeted advertising
    - Why does Facebook seem to know what I'm thinking ?

Where is ML used ?

Spam detection

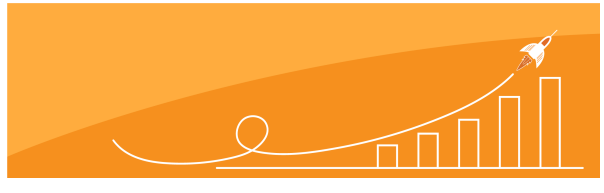
- You are a winner !



Where is ML used ?

Forecasting

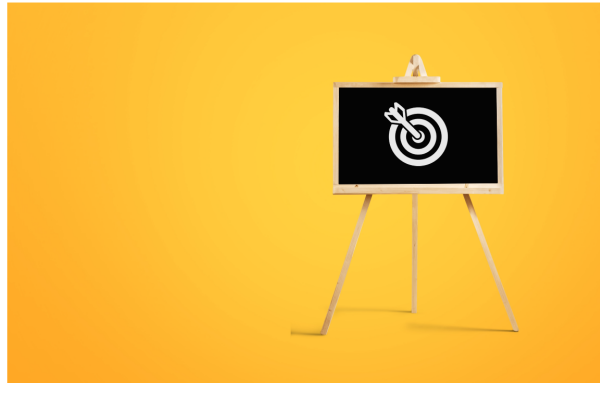
- Sales
- Logistics
- Where's my Uber ?



Where is ML used ?

Anomaly detection

- Credit card fraud



Uses in Finance

- Model prices, risk
    - hedging
- Trading signals
- forecast sales
- Predict defaults, prepayments

Not just numeric data !

- Images
    - Satellite:
        - Counting cars in a parking lot to forecast sales
        - How full is that oil tank ?

- Did the CFO really mean what he said ? facial signals for confidence/evasiveness
- Text

    - Twitter sentiment as a signal ?
    - SEC filings
    - Derive industry groups by clustering press releases

What you need to succeed: An inquiring mind

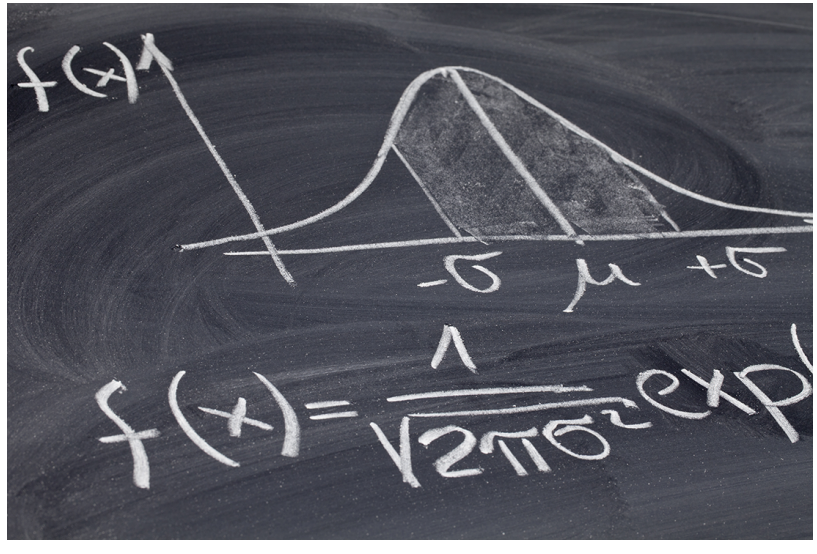
- Approach this topic like a Scientist
    - Find a problem, gather data, formulate a hypothesis, test.
    - Repeat.

What you need to succeed: Technical skills

You are engineers !

What you need to succeed: Solid programming skills

What you need to succeed: Some math/statistics



To be a successful data scientist, you need to understand the machinery.

It is not enough to know an API.

What you need to succeed Self-motivation and energy

- Willingness to pick up tools/skills outside of lectures
- You are engineers, nothing is too hard !

Technical prerequisites

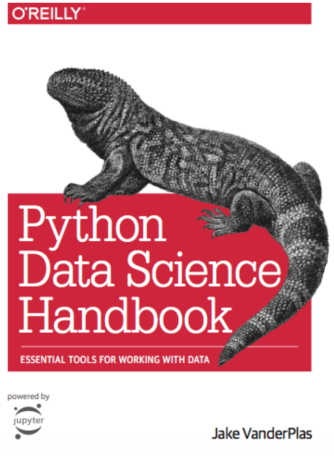
- Python
    - Object oriented (OO) Python
    - Numpy
    - Pandas
    - Matplotlib
- Some statistics (e.g., regression)
- Some math
    - comfort with Matrix/Vector notation

Textbooks

Python Data Science Handbook

VanderPlas (https://jakevdp.github.io/PythonDataScienceHandbook/)

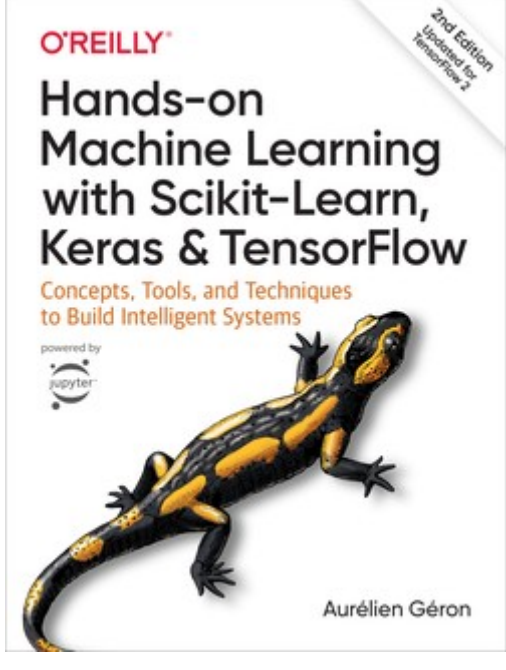
- Online !
- Solid foundation to acquire pre-requisites
    - Jupyter
    - Numpy
    - Pandas
    - Matplotlib
    - Quick view of models



Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition

Geron (https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/)

- Assumes you know the pre-requisites
    - More detailed chapters on various models



Notebooks

The real learning comes from active "doing" (playing with notebooks) rather than passive "reading".

The course material and both textbooks have code repositories (including notebooks) on GitHub

- The VanderPlas "book" is actually a notebook !

Here's how to download a repository.

First: create a directory in which to store the notebooks

- Our custom is to have a folder called `Notebooks` in the home directory, but you are free to change it
- The following command will create the directory and switch to it

```
mkdir ~/Notebooks; cd ~/Notebooks
```

Accessing a repository on Github

If the `git` command is not available on your machine you can install it via the command

```
conda install git
```

`git` provides a convenient way to download a repo to your local machine (and tools to keep it updated).

Or: you can download a repo as a ZIP file.

Get a repository via git

Use the following commands

- To get the course material

```
git clone https://github.com/kenperry-public/ML_Fall_2020.git
```

- To get the repositories associated with the recommended textbooks:

```
git clone https://github.com/jakevdp/PythonDataScienceHandbook.git
git clone https://github.com/ageron/handson-ml2.git
```

Get a repository as a ZIP file

Using a web browser: visit the URL in each of the `git clone` commands above

- Click on the

  button
- Choose "Download ZIP"

Machine Learning *using* Scikit-Learn (sklearn), TensorFlow (TF)

sklearn is a popular library for Machine Learning. We will be using for the Classical ML part of the series.

TensorFlow is another popular library that we will use in the second part of the course.

We are learning **Machine Learning**, not sklearn/TensorFlow !

Tools are a means, not an ends

- Goal is to understand ML independent of the toolset
- You can be an expert in sklearn/TensorFlow and still not understand ML

Teaching method

Iterative: visit the problem many times, at increasing levels of focus

- Top-down vs bottom-up

- Motivate: very high level view
    - Know WHAT we are trying to achieve

- Understand: medium level view

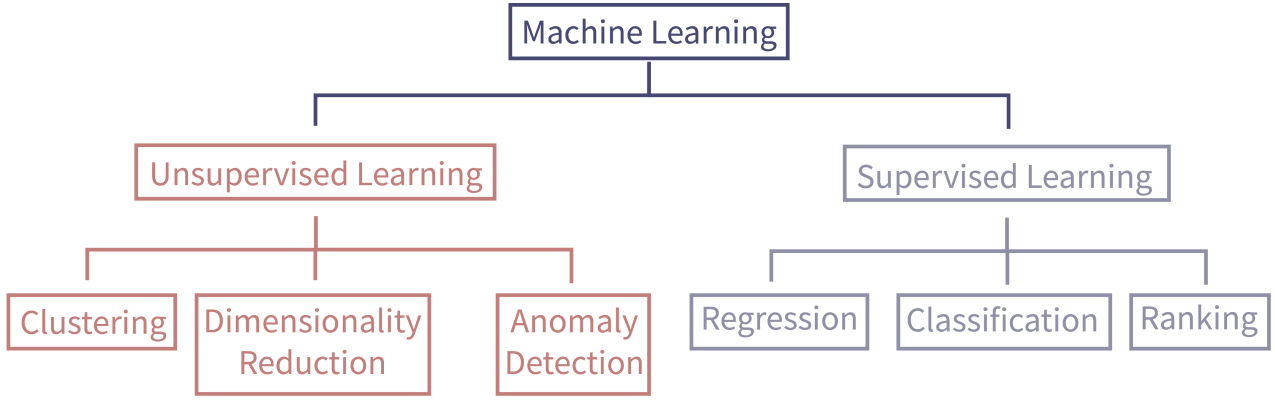
- Deep understanding
    - math, statistics

Bonus: advice from a practitioner

Dimensions of ML

Types of learning

- Supervised
- Unsupervised


Machine Learning
Unsupervised Learning
Supervised Learning
Clustering
Dimensionality Reduction
Anomaly Detection
Regression
Classification
Ranking

</td></tr></table>

Types of targets

- Continuous
- Discrete/Categorical

Types of features

- numeric
- image
- text

Challenges of ML

- You need data to train, often a lot of it
    - Not always easy to get
        - supervised: needs to be labeled
    - Quality issues
    - Is the training data representative of "the real world" for which you are designing ?

- Overfitting and Underfitting
    - Overfit: good training accuracy, poor generalization
    - Underfit: lost opportunity

- Engineering meaningful features is key
    - Data transformations
        - Create features that aid prediction
        - Art and science
    - Deep Learning may view feature engineering as part of the problem, not the solution !

- Testing and validation
    - An honest test uses held-out data
    - Training data is a precious resource; painful to hold some out

ML in one slide

Observations:
features
features/labels

$x,^1\ x,^2 \ldots, x^N$

$y,^1\ y,^2 \ldots, y^N$

Create a predictor (estimator, hypothesis) $y^* = h(x^*)$

- Predicts an outcome when presented with features x^*
- Prediction based on the subset of $x,^1\ x,^2 \ldots, x^N$
- That looks like x^*

Predict

Scikit Learn

```
from sklearn import mod_class
model = mod_class.Model()

model.fit(x,y)
```

```
y_star =
model.predict(x_star)
```

In [3]:
```
print("Done")
```

```
Done
```