

# Predictive Analytics

## Optional Lecture

UCI Data Analytics Bootcamp  
Tuesday January 22, 2019  
Peter Kim

Available as Google Slides:

<https://docs.google.com/presentation/d/16EILCeRyTzn4GNiWGuGRNmIVpP1KI-dXJBT1rnw-K5M/edit#slide=id.p>

# Optional review session on predictive analytics.

Based on the Slack anonymous poll to students, “predictive analytics” was a popular requested topic for review.

The term “predictive analytics” is often used synonymously with “machine learning,” which was previously covered over past few weeks of lectures and exercises.

I will share some thoughts and material that wasn’t already covered during previous lectures. We deliberately skipped math, but a little more background and context will go a long way.

Prior lectures extensively covered the scikit-learn workflow, so I’ll try to add new material. But the prior natural language processing (NLP) lecture focused on pyspark, so I can discuss NLP implementation on scikit-learn.

# The same thing called different names.

Predictive Analytics: this phrase tends to come from people with a business background. (or statistics background).

Machine Learning: this phrase tends to come from people with a computer science background.

## It's important to know your audience.

- If a job posting is “analytics”, focus more on revenue increases, cost savings, customer retention, etc. (less focus on code and math).
- If a job posting is “machine learning”, focus more on programming, libraries, and research.

## Some might argue for subtle differences:

- Analytics tends to focus on structured (i.e. tabular) data, while Machine Learning can also include unstructured data such as images and text.
- Machine Learning may involve software and algorithms that use graphics chips (GPU) and/or Big Data (data fits on multiple computers)

# Cultural divide in the statistics world.

Leo Breiman was a Professor of Statistics at UC Berkeley (also instrumental in development of Random Forest algorithm). He wrote a very important paper in 2001 called “Statistical Modeling”. If you come from a statistics background, and want to understand the relationship between (classical) statistics, and (what Breiman calls) algorithmic models, this paper is a good introduction.

[https://projecteuclid.org/download/pdf\\_1/euclid.ss/1009213726](https://projecteuclid.org/download/pdf_1/euclid.ss/1009213726)

[https://en.wikipedia.org/wiki/Leo\\_Breiman](https://en.wikipedia.org/wiki/Leo_Breiman)

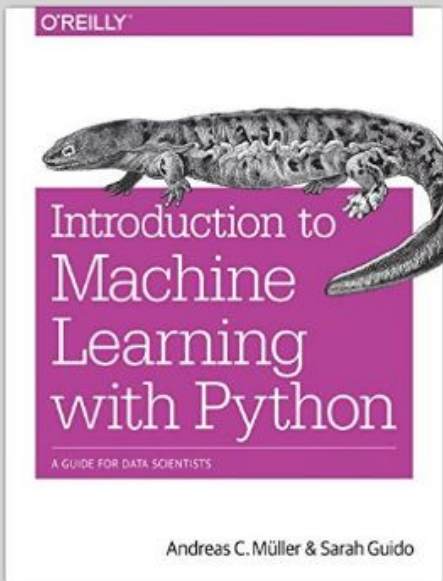
# Good starting place.

<http://amueller.github.io/>

<http://shop.oreilly.com/product/0636920030515.do>

## INTRODUCTION TO MACHINE LEARNING WITH PYTHON

---



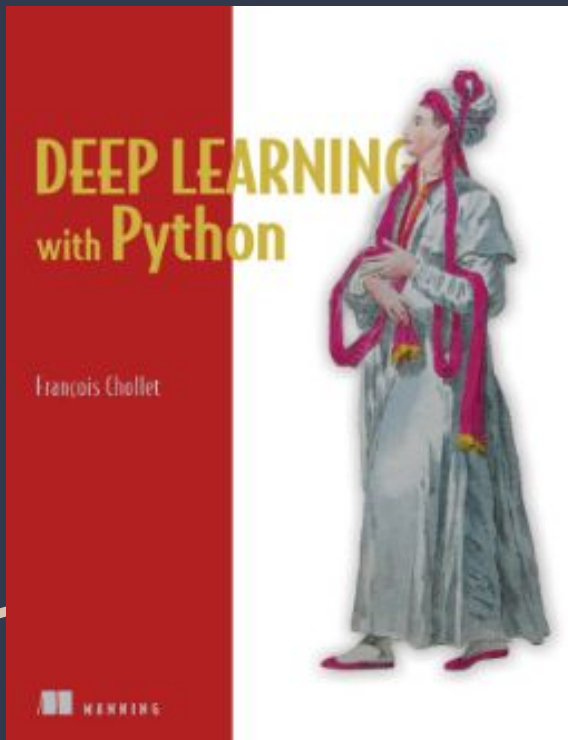
Introduction to Machine Learning with Python provides a practical view of engineering machine learning systems in Python. The premise of the book is to enable people to learn the basics of machine learning without requiring a lot of mathematics. We therefore keep the amount of formulas to a minimum, and instead rely on code and illustrations to bring across the driving principles behind applying machine learning. We heavily focus on the use of the scikit-learn machine learning library, and give a detailed tour of its main modules and how to piece them together to a successful machine learning pipeline.

Book website >

Github repository with all code >

Buy on Amazon >

# History of machine learning (Chollet).



## Probabilistic modeling

- Naive bayes (1950s)
- Logistic regression

## Early neural networks

- Early investigations (1950s)
- Yann LeCun, Bell Labs (1990s)

## Kernel methods

- Support vector machines (1990s)

## Decision trees

- Random forest
- Kaggle launched in 2010
- XGBoost (2014)

## Deep learning (neural networks with GPU)

- Keras (2015)
- Tensorflow (2015)

# GPU vs. CPU.

## What does the graphics processor have to do with machine learning?

What is the big deal about graphics processors (GPU) in machine learning?

Semiconductors (aka “chips”) are the brains that make the computer run. **The most important chip in a computer is called a “CPU”.** It runs the operating system, and coordinates most activities (including the other chips). **The problem is that starting in early 2000s, for the first time since chips were invented (1950s and 1960s), the CPU was not becoming significantly more powerful with each new generation (“Moore’s Law”).**

The reasons for this are somewhat complicated, and require background in physics and electrical engineering. One solution was to build multiple “cores” onto a CPU (e.g. 8 cores), but most programs are not necessarily designed to use the cores in parallel.

Unlike CPUs, the graphics processors (GPUs) continue to get more and more powerful with each generation. **GPUs were originally built for accelerating computer games, so they are designed with a parallel architecture for graphics, using hundreds (or thousands) of cores that each do simple activities (math).** Around 2007, Nvidia released a language (CUDA) that allows developers to program the GPU to perform activities other than graphics. This new approach had clunky name: GPGPU (general purpose GPU) or GPU Compute. **GPUs are really good at certain type of math, called matrix multiplication, because they run hundreds of calculations in parallel.**

**Machine learning libraries take advantage of the GPU architecture (e.g. Tensorflow, Theano, XGBoost), by offloading the math operations onto the GPU** (if your computer has a GPU) instead of the CPU. This can speed up the calculations by 10x or 100x (or more) depending on the nature of the workload. (BTW, this is also why crypto-currency-miners were using GPUs – although not anymore).

Bill Dally wrote an article about Moore’s Law in Forbes Magazine in April 2010.  
<https://www.forbes.com/2010/04/29/moores-law-computing-processing-opinions-contributors-bill-dally.html#30eba3f32a86>

For more background: <https://en.wikipedia.org/wiki/CUDA>

# Accessibility of AI became an explicit goal of keras.

Chollet talks about the accessibility and democratization of artificial intelligence. He notes that accessibility became an explicit goal of keras.

In fact, Chollet's book avoids mathematical notation.

Much of the teaching of machine learning focuses on the math. But what if someone is interested in machine learning but doesn't have background in math. It's still possible to get started with scikit-learn and keras.

If you are interested in learning the math, there are many great (and free!) resources available online. But it's going to take hard work, perseverance, and strong desire to learn.


<https://machinelearningmastery.com/basics-mathematical-notation-machine-learning/>



# What is a tensor?

## Linear Algebra Primer.

These look like a Python “list”  
or a numpy “array”.



```
[ 1 ]
```

This is a “**scalar**” (0 dimensions)  
Variable uses lower-case (“x”)

```
[ 1,  
  2,  
  3 ]
```

This is a “**vector**” (1 dimension)  
Variable uses lower-case (“x”)

```
[ 1, 4,  
  2, 5,  
  3, 6 ]
```

This is a “**matrix**” (2 dimension)  
Variable uses upper-case (“X”)

```
[[ 1, 4,  
   2, 5,  
   3, 6 ],
```

This is a “**tensor**” (3+ dimension)  
Variable uses upper-case (“X”)  
Examples: image, video

```
[ 7, 10,  
  8, 11,  
  9, 12 ]]
```

# What is matrix multiplication?

## Multiplying a Matrix by Another Matrix

But to multiply a matrix **by another matrix** we need to do the "[dot product](#)" of rows and columns ... what does that mean? Let us see with an example:

To work out the answer for the **1st row** and **1st column**:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & \\ & \end{bmatrix}$$

The "Dot Product" is where we **multiply matching members**, then sum up:

$$(1, 2, 3) \cdot (7, 9, 11) = 1 \times 7 + 2 \times 9 + 3 \times 11 \\ = 58$$

We match the 1st members (1 and 7), multiply them, likewise for the 2nd members (2 and 9) and the 3rd members (3 and 11), and finally sum them up.

$$\begin{matrix} \textbf{A} & * & \textbf{B} & = & \textbf{C} \\ \begin{bmatrix} 1, & 2, & 3, \\ 4, & 5, & 6 \end{bmatrix} & & \begin{bmatrix} 7, & 8, \\ 9, & 10, \\ 11, & 12 \end{bmatrix} & & \begin{bmatrix} 58, & 64, \\ 139, & 154, \end{bmatrix} \end{matrix}$$

(2x3 matrix)

(3x2 matrix)

(2x2 matrix)

```
In [1]: 1 import numpy as np
```

```
In [2]: 1 A = np.array([[ 1, 2, 3],  
2                  [ 4, 5, 6]])
```

```
In [3]: 1 A
```

```
Out[3]: array([[1, 2, 3],  
              [4, 5, 6]])
```

```
In [4]: 1 B = np.array([[ 7, 8],  
2                  [ 9, 10],  
3                  [11, 12]])
```

```
In [5]: 1 B
```

```
Out[5]: array([[ 7,  8],  
              [ 9, 10],  
              [11, 12]])
```

```
In [6]: 1 C = np.dot(A, B)  
2      C
```

```
Out[6]: array([[ 58,  64],  
              [139, 154]])
```

# YouTube Spam

Alberto, T.C., Lochter J.V., Almeida, T.A. TubeSpam: Comment Spam Filtering on YouTube. Proceedings of the 14th IEEE International Conference on Machine Learning and Applications (ICMLA'15), 1-6, Miami, FL, USA, December, 2015. (preprint).

- <http://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection>

A useful reference is Kevin Markham's tutorial on NLP using scikit-learn.

- <https://github.com/justmarkham/pycon-2016-tutorial>
- <https://www.youtube.com/watch?v=WHocRqT-KkU>

Lastly, review the Andreas Mueller book for examples:

- <http://shop.oreilly.com/product/0636920030515.do>

# What does the data look like?

448 rows, 5 columns  
Column "CLASS" has (0, 1)  
0: regular comment ("ham")  
1: fake comment ("spam")  
0: 203 records  
1: 245 records

```
In [1]: 1 import pandas as pd
        2 from pandas import DataFrame, Series
```

```
In [2]: 1 df_text = pd.read_csv('Youtube04-Eminem.csv', dtype=str)
```

```
In [3]: 1 df_text.shape
```

```
Out[3]: (448, 5)
```

```
In [4]: 1 df_text.head()
```

```
Out[4]:
```

	COMMENT_ID	AUTHOR	DATE	CONTENT	CLASS
0	z12rwnyrbsefonb232i5ehdxzkjzs2	Lisa Wellas	NaN	+447935454150 lovely girl talk to me xxx	1
1	z130wpnwmyuetxcn23xf5k5ynmkdpjrj04	jason graham	2015-05-29T02:26:10.652000	I always end up coming back to this song 	0
2	z13vsfqirtavju0t22ezrgzyorwxhpf3	Ajkal Khan	NaN	my sister jst received over 6,500 new <a rel=...	1
3	z12wjzc4epnrnja4304cgbbizuved35wxcs	Dakota Taylor	2015-05-29T02:13:07.810000	Cool	0
4	z13xjfr42z3uxdz2223gx5rrzs3dt5hna	Jihad Naser	NaN	Hello I&#39;am from Palastine	1

```
In [5]: 1 df_text['CLASS'].value_counts()
```

```
Out[5]: 1    245
        0    203
        Name: CLASS, dtype: int64
```

# Test-Train Split

Train dataset is for machine learning.

Test dataset is to see how well it worked.

YouTube Dataset	Input (aka X)	Output (aka y)
Train (~80% of data)	358 records ['CONTENT']	358 records ['CLASS']
Test (~20% of data)	90 records ['CONTENT']	90 records ['CLASS']
<b>Total (100% of data)</b>	<b>448 records ['CONTENT']</b>	<b>448 records ['CLASS']</b>

# See Jupyter Notebook.

UCI-YouTube-Spam.ipynb

# Appendix

Back-up slides to follow

(Supplemental material)

# Climb a large mountain just to get started.

1. Learn how to code
  - a. Install Python
  - b. Anaconda
  - c. Core Python
2. How to use third party libraries
  - a. pip and conda
  - b. github
  - c. Read official documentation
  - d. Troubleshoot problems using stackoverflow
  - e. Leveraging online tutorials (including YouTube)
3. Data
  - a. Pandas / Numpy
  - b. Accessing data in databases and APIs
4. Machine learning concepts
  - a. Math approach
  - b. Code approach
    - i. Mueller
    - ii. Chollet



# Books.

Introduction to Machine Learning with Python: A Guide for Data Scientists by Andreas C. Müller (Author), Sarah Guido

Deep Learning with Python by Francois Chollet

Python Data Science Handbook: Essential Tools for Working with Data by Jake VanderPlas  
(free online version available)

Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython by Wes McKinney

“Essentially,  
Python has just  
become the lingua  
franca of nearly all  
the neural network  
toolkits.”

Professor Christopher Manning, Stanford University,  
“Natural Language Processing with Deep Learning -  
Lecture 1” (Uploaded to YouTube in 2017). (34:45 out  
of 1:11:40).

[https://www.youtube.com/watch?v=QQQ-W\\_63UgQ&list=PL3FW7Lu3i5Jsnh1rnUwq\\_TcylNr7EkRe6](https://www.youtube.com/watch?v=QQQ-W_63UgQ&list=PL3FW7Lu3i5Jsnh1rnUwq_TcylNr7EkRe6)

# Outline of Topics

Historical context.

Drew Conway Model of Data Science:

- Coding. Programming language. Reading data, cleaning data, moving data around. Working with third-party libraries.
- Math. Univariate and multivariate statistics. Probability. Linear algebra. Calculus.
- Subject matter expertise. Application of “coding + math” to solve a problem and accomplish a task.

Jupyter Notebook. Natural Language Processing using Scikit-Learn.