Machine Learning Cheat Sheet

Here is all you need to get started from scratch, including sample projects, data sets, and references. Author: Vincent Granville. Published June 6, 2019.

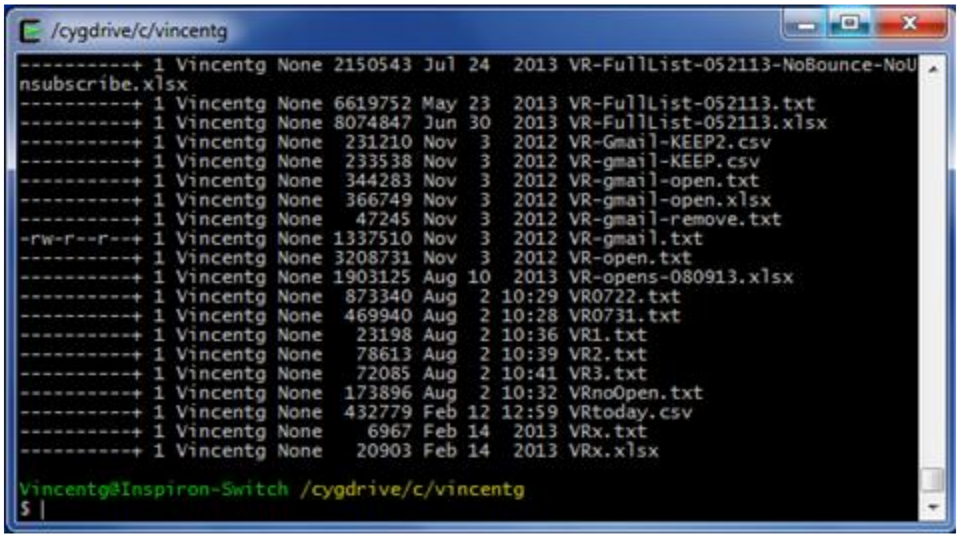
1. Hardware

A laptop is the ideal device. I've been using Windows laptops for years, and I always installed a Linux layer (acting as an operating system on top of Windows), known as Cygwin. This way, you get the benefits of having Windows (Excel, Word, compatibility with clients and employers, many apps such as FileZilla) together with the flexibility and pleasure of working with Linux. Note that Linux is a particular version of UNIX. So the first recommended step (to start your data science journey) is to get a modern Windows laptop and install Cygwin.

Even if you work heavily on the cloud (AWS, or in my case, access to a few remote servers mostly to store data, receive data from clients and backups), your laptop is you core device to connect to all external services (via the Internet). Don't forget to do regular backups of important files, using services such as DropBox.

2. Linux environment on Windows laptop

Once you installed Cygwin, you can type commands or execute programs in the Cygwin console. Here's how the console looks like on my laptop:



```
----------+ 1 Vincentg None 2150543 Jul 24  2013 VR-FullList-052113-NoBounce-NoU
nsubscribe.xlsx
----------+ 1 Vincentg None 6619752 May 23  2013 VR-FullList-052113.txt
----------+ 1 Vincentg None 8074847 Jun 30  2013 VR-FullList-052113.xlsx
----------+ 1 Vincentg None  231210 Nov  3  2012 VR-Gmail-KEEP2.csv
----------+ 1 Vincentg None  233538 Nov  3  2012 VR-gmail-KEEP.csv
----------+ 1 Vincentg None  344283 Nov  3  2012 VR-gmail-open.txt
----------+ 1 Vincentg None  366749 Nov  3  2012 VR-gmail-open.xlsx
----------+ 1 Vincentg None   47245 Nov  3  2012 VR-gmail-remove.txt
-rw-r--r--+ 1 Vincentg None 1337510 Nov  3  2012 VR-gmail.txt
----------+ 1 Vincentg None 3208731 Nov  3  2012 VR-open.txt
----------+ 1 Vincentg None 1903125 Aug 10  2013 VR-opens-080913.xlsx
----------+ 1 Vincentg None  873340 Aug  2 10:29 VR0722.txt
----------+ 1 Vincentg None  469940 Aug  2 10:28 VR0731.txt
----------+ 1 Vincentg None   23198 Aug  2 10:36 VR1.txt
----------+ 1 Vincentg None   78613 Aug  2 10:39 VR2.txt
----------+ 1 Vincentg None   72085 Aug  2 10:41 VR3.txt
----------+ 1 Vincentg None  173896 Aug  2 10:32 VRnoOpen.txt
----------+ 1 Vincentg None  432779 Feb 12 12:59 VRtoday.csv
----------+ 1 Vincentg None    6967 Feb 14  2013 VRx.txt
----------+ 1 Vincentg None   20903 Feb 14  2013 VRx.xlsx

Vincentg@Inspiron-Switch /cygdrive/c/vincentg
$ |
```

Figure 1: Cygwin (Linux) console on Windows laptop

You can open multiple Cygwin windows on your screen(s).

To connect to an external server for file transfers, I use the Windows FileZilla freeware rather than the command-line ftp offered by Cygwin. If you need full privileges on the remote machine, use Putty instead (for Telnet/SSH sessions).
You can run commands in the background using the & operator. For instance,

```
$ notepad VR3.txt &
```

will launch Notepad (the standard Windows text editor) from the Cygwin console, into another window, and open the file `VR3.txt` located in your local directory (if this file exists in that directory). Note the `$` symbol preceding any command (see Figure 1). In addition, the console also displays the username (`Vincent@Inspiron-Switch` in my case) as well as the directory I'm in `(/cygdrive/c/vincentg/` in Linux, corresponding to the `c://vincentg/` pathname under windows).

Basic operations:

- Changing directory is performed with the command cd (examples: `cd subfolder/sub-subfolder`, `cd ..` to go one level above, `cd .` to go to your home directory)
- Listing content of directory is done with command `ls -l` (note that `-l` is a command argument used to specify that want a full, detailed listing; without this option, the listing shown in Figure 1 would be far less detailed).
- If you don't know your local directory, type in the command pwd, it will tell you your location (path)

So far you've learned the following Linux concepts: command line and symbol `$` (sometimes replaced by `>` depending on the Linux version), operator `&` (for background processing), paths, commands `cd`, `pwd`, and `ls`, command options (`-l` for `ls`) and shortcuts (`.` and `..` for the `cd` command).

A few more things about files

Files have an extension that indicates what kind of file it is (text, image,spreadsheet) and what software can open and process them. In Figure 1, `VR3.txt` has the `.txt` extension, meaning it's a text file - the most widespread type of data file. There are two types of files: binary (used by various programs; compressed/encrypted format) and text (can be processed easily by any program or editor). It is important to know the distinction when doing FTP file transfers (FTP clients allow you to specify the type of file, though it's automated and transparent to the user with FileZilla).

Other extensions include

- .csv (comma-separated text file that you can open with Excel or Notepad; it can have more than 1 million rows),
- .xlsx (Excel files limited to 1 million rows, this is a binary file),

- .gz (compressed files, thus binary files),
- .png (best image format, other image formats include .gif, .jpg, .jpeg, and .bmp; these are binary files),
- .docx (Word documents; binary),
- .html (text files representing source code of a web page),
- .sql (text file used to store an SQL query, used as input for some database clients such as Brio),
- .php (PHP code, text format),
- .pl (Perl code, text format),
- .js (Javascript code, text format),
- .r (R code, text format),
- .py (Python code, text format),
- .c (C code, text format),
- .exe (Windows executable),
- .xml (XML, text format for name-value pairs)

Files are not stored exactly the same way in Windows and UNIX. Also, some systems use UNICODE for file encoding, which takes much more space but allow you e.g. to work with Chinese characters (stored using two bytes per character). When processing such a file (they are rather rare fortunately), you'll first need to clean it and standardize it to traditional ASCII (one byte = one character).

Finally, the best text format that you can use is tab separated: each column or field is separated by a TAB, an invisible char represented by \t in some programming languages. The reason is that some fields contain commas, and thus using csv (comma-separated text files) results in broken fields and data that looks like garbage, and is hard to process (requiring a laborious cleaning step first, or talking to your client to receive tab-separated format instead).

When processing data, the first step is to produce a data dictionary. It is easily done using a scripting language.

File management

Filenames should be designed carefully (no space or special char in a filename), especially when you have thousands or millions of files across thousands of directories and sub-directories, and across dozens of servers (the cloud). One of the two core components of Hadoop is actually its file management system, known as HDFS (the other component being the distributed *Map-Reduce* architecture to process tasks).
It's always a good idea to always have a time stamp embedded into the file name, representing the creation date. Note that in Figure 1, the files all start with VR, an abbreviation for Vertical Response, as these files are coming or related to our email service provider, called Vertical Response. File names should be very detailed: keep in

mind that sooner rather than later, you might run scripts to process millions of them. Without proper naming conventions, this task will be impossible.

A final word, if you look at Figure 1, the first column indicates who can read (r), re-write (w) or execute (x) these files, besides me. It's never been an issue on Windows for me, but on a true UNIX operating system (not Cygwin), you might want to set the right protections: for example Perl scripts (despite being text) must be set to Executable, with the UNIX command `chmod 755 filename.pl`, where `filename.pl` is your Perl script. File protections (and locks) are important for organizations where files can be shared by many users, sometimes simultaneously.

3. Basic UNIX commands

You don't need to spend hours learning UNIX and buy 800-page books on the subject. The following commands will get you started, once you have your Cygwin console:

- `cd`, `pwd`, `ls` (see section 2)
- `tail -100, head -150` to extract the last 100 or first 150 rows of a file
- `cp`, `mv`, `mkdir`, `rmdir` respectively copy a file to another location, rename a file, create a new directory or delete a directory (you need to erase all files first)
- sort, uniq respectively sort a file and remove duplicate entries (you can sort alphabetically or numerically depending on the option; default is alphabetical order)
- `gzip`: compress/un-compress files
- `wc`: count number of rows and words in a text file
- `grep`: identify all rows containing a specific string in a text file (it helps to be familiar with regular expression)
- `cat`: display content of text file on your screen
- `chmod`: change file protections, see section 2
- `history`: lists the last commands you used, as it is very common to re-use the same commands all the time.
- `cron`, `crontab`: to automatically schedule tasks (running an executable once a day)

Operators include `>` (to save output to a new file), `>>` to append output to an existing file, `|` (the pipe operator, see examples), `&` (see section 2, used for background or batch mode when executing a command), `*` (see examples) and `!` (see examples.)

Examples

- `sort filename.txt | uni -c > results.txt` (sort `filename.txt` alphabetically - not numerically - then remove duplicates, and for each final entry count number of duplicates with option `-c`; store results in `results.txt`)
- `rm -i test*.txt` (remove all files starting with test and with extension `.txt`; the extension `-i` is to request manual confirmation before each file gets deleted)

- `grep  'abc' test.txt | wc` (extract all rows containing ‘abc’ in `test.txt`, then count these rows with `wc`)
- `!545` (run command #545, after you run the command history to get the lists of previously entered commands)

Check out details for these commands (exact syntax and options).

Miscellaneous

Shell scripts (or batch files) are small programs that execute a list of commands, and can be run in batch mode. For regular expressions, see section 4.

4. Scripting languages

You can get started in data science wth just a few Unix commands, a tool for statistical analyses such as R (unless you write your own algorithms to get more robust and simple tools) and a scripting programming language such as Perl or Python. Python (together with Pandas libraries) is the most popular language for data science. Python and machine learning resources are provided later in this article. This article is a good introduction on Python for data science. This reference has tons of resources about Python for data science.

Here I describe fundamental features of Perl, but they apply to all scripting languages. You can download Perl from ActiveState. Numerous short programs (Perl, but also R), easy to read and understand, can be found here. Perl scripts are text files with a `.pl` extension (say `myprogram.pl`) that you can execute in the Cygwin console with the command line `perl myprogram.pl` once you have installed Perl on your laptop.

Core elements of scripting languages

Some basic stuff that is used in pretty much any programs include

- **Hash tables** are lists of name-value pairs, where insertion or deletion of an element is very fast. They can be descibed as arrays indexed by strings, and constitute a powerful, fundamental data structure. They can be used to produce efficient joins. See our data dictionary article for a simple illustration. Hash tables store data using a syntax such as `$myhash{"Vincent Granville|Data Scientist"} = "yes"`; In this case the index is bi-dimensional and is made up of the name and job title; the value is "yes" or "no". If the name or job title is not in your data, no entry is created (that's why this data structure produces efficient joins). See also this article on feature selection, for a more sophisticated application.

- **Associative arrays** are just hash tables: arrays indexed by strings rather than integers. In Perl, they are declared using `%myhash=()` while regular arrays are declared using `@myarray=()`. Memory allocation for hash tables is automated in Perl. However, you should create a variable `$myhashsize` that is incremented by 1 each time an entry is added to `%myhash` (or decremented by 1 in case of deletion). This way, you know how big your hash tables grow. If your program displays (on the screen) the exact time every 300,000 newly created hash entries, you'll have an idea when you run out of memory: at that moment, your Perl script suddenly starts running 20 times slower. When this happens, it's time to think about optimization using Hadoop or Map-Reduce.

- String processing and regular expressions: the sample code below contains basic strings substitution including special characters (`\n`, `\:`). Many substitutions can be performed in just one tiny bit of code using **regular expressions**, click here or here for details. One of the most widespread operations is to split a text `$text` into elements stored in an array `@myarray`; the syntax is `@myarray = split(/\t/,$text)`; Here we assume that text elements are separated by TABs (the special character `\t`). The number of text elements is stored in the variable `$#myarray`.

The easiest way to learn how to code is to look at simple, well written sample programs of increasing complexity, and become an expert in Google search to find solutions to coding questions - many answers can be found on StackOverflow. I have learned R, SQL C, C++ and Perl that way, without attending any training. If you need training, read the section on training in chapter 2 in my book or check out this list of courses. The following are good examples of code to get you started.

Sample scripts to get you started

Here is some sample code.

- Code to run SQL queries 10 times faster than Brio, Toad etc.
- Source code for our Big Data keyword correlation API (Perl API)
- Source code to compute N-grams (NLP)
- Simulation of stochastic processes
- Simple source code to simulate nice cluster structures
- Ridge regression with bootstrap
- Basic web crawler
- Model-free confidence intervals (see section 2, subsection "Perl code")

Below is a simple script that performs automated dns lookups to extract domain names associated with IP addresses. The input file is a list of IP addresses (`ips.txt`) and the output file is a text file `outip.txt` with two fields, tab-separated: IP address and domain name. A temporary file `titi.txt` is created each time we call the external Cygwin command `'nslookup'`. Note that `$` is used for variables. There's some basic string processing here, for instance: `$ip=~s/\n//g` substitutes each carriage return /

line feed (special character `\n`) by nothing (empty) in the variable `$ip`. Note that the symbol `#` means that what follows (in the line in question) is a comment, not code.

```
`rm titi.txt`;
# $ip="107.2.111.109";
open(IN,"<ips.txt");
open (OUT,">outip.txt");
while ($lu=<IN>) {
  $ip=$lu;
  $n++;
  $ip=~s/\n//g;
  if ($ip eq "") { $ip="na"; }
  `nslookup $ip | grep Name > titi.txt`;
  open(TMP,"<titi.txt");
  $x="n/a";
  while ($i=<TMP>) {
    $n++;
    $i=~s/\n//g;
    $i=~s/Name\://g;
    $x=$i;
  }
  close(TMP);
  print OUT "$ip\t$x\n";
  print "$n> $ip | $x\n";
  sleep(0);
}
close(OUT);
close(IN);
```

Now, you can download logfiles (see free data sets in section 6), extract IP addresses and traffic statistics per IP address, and run the above script (using a distributed architecture, with 20 copies of your script running on your laptop) to extract domain names attached to IP addresses. Then you can write a program to map each IP address to an IP category using the technique described in my article Internet Topology Mapping. And finally, sell or license the final data to clients.

A few more useful concepts:

- Functions in Perl are declared using the `subroutine` reserved keyword. A few examples are found in the sample scripts. Learn how to pass an argument that is a variable, an array or an hash table. Subroutines can return more than one value. Use of global variables is discouraged, but with proper care (naming conventions), you can do it without risks.
- You can write programs that accept command-line arguments. Google 'command-line arguments' for details.
- Libraries (home-made or external) require an inclusion directive, such as require `LWP::UserAgent;` in the web robot sample code (see link above) that uses the LWP library. If a library is not available in your Perl distribution, you can download and add it using the ppm command, or even manually (see my book page 138, where I discuss how to manually install the library `Permutor.pl`).

- Scripts can be automatically run according to a pre-established schedule, say once a day. Google 'cron jobs' for details, and check this article for running cron jobs on Cygwin.

Exercise

Write a script that accesses all the text files on your laptop using two steps:

- recursively using the `ls-l > dir.txt` Cygwin command from within Perl to create directory listings (one for each folder / subfolder) saved as text files and named dir.txt
- accessing each text file from each of these automatically created directory listings dir.txt in each directory

Then count the number of occurrences for each word (broken down per file creation year) across these files, using a hash table. Purpose: identify keyword trends in your data.

5. Python, R, Hadoop, SQL, DataViz

R is a popular language to perform statistical analyses or nice graphics. I would not use it for black-box applications. Large applications such as text clustering involving 20 million keywords are performed in Python or Perl, known as scripting languages. Python libraries for data analysis and machine learning are widely available and discussed in a few O'Reilly books: they offer an alternative to R, for big data processing. Note that R does have an extensive collection of sophisticated statistical functions, too many in my opinion. Finally, R is currently used for exploratory data analysis rather than production-mode development. For more info, read R versus SAS versus Python or R versus Python.

R programming

You can download the open-source R package from The R Project. Installation and running R programs via the GUI, on a Windows laptop, is straightforward. Memory limitations can be bypassed using multiple copies of R on multiples machines, some R packages, or using RHadoop (R + Hadoop). R programs are text files with an `.r` extension. Useful links:

- Producing videos with R
- R libraries
- R cheat sheets
- Sample R code

Also, see here for more references.

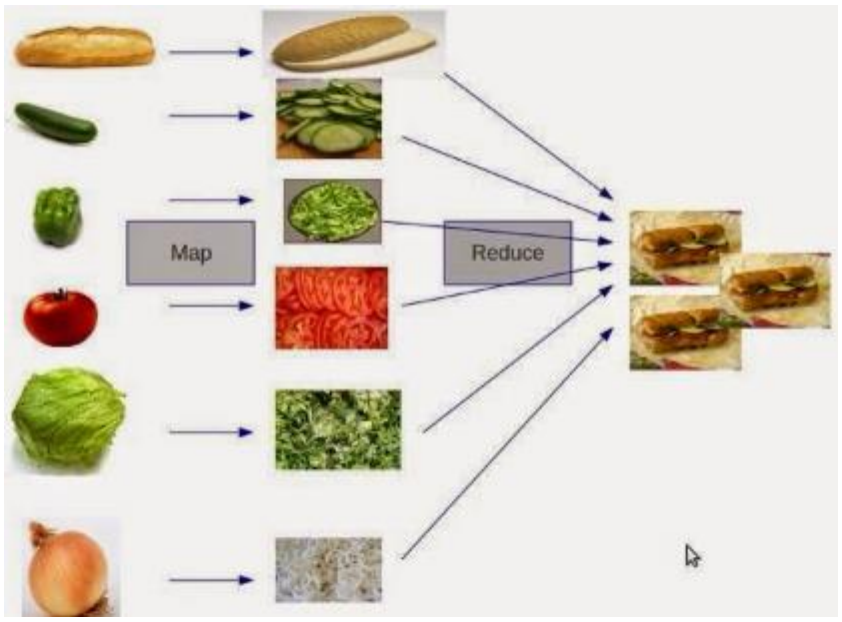
Python programming

Check out these articles about Python for data science, the preferred full-fledged programming language for data scientists. Sample code, cheat sheets, and machine learning libraries for Python, can be found on GitHub.com and in the following articles:

- Python cheat sheets
- Python libraries
- Sample Python code

Hadoop

Hadoop is a file management system used to perform tasks in a distributed environment, across multiple servers if necessary, by spitting files into sub-files, performing the analysis on each sub-file separately, and summarizing the results (by collecting the various outputs associated with each file, and putting it together). This environment uses redundancy to easily and transparently recover from server crashes. It works well for web crawling projects, even for sorting, but not so much for graph databases or real time data (except as a back-end platform). For an example, read this article. For an example of what Map-Reduce (the distributed architecture supporting Hadoop) can't do, follow this link.



Map-Reduce explained in one picture

SQL

Finally, don't forget that SQL is still a widely used language. Learn at least the basics, including to joining multiple tables efficiently, and playing with indexes and keys. The book SQL Essentials is a good starting point. Also, search this PDF document for the keyword *fuzzy joins*. NoSQL databases also exist, in particular graph databases.

Excel

Excel has advanced functions such as Linest (linear regression), Vlookup, percentiles, rank statistics, random numbers, or index (indirect cell referencing.) A large collection of Excel spreadsheets featuring advanced machine learning techniques can be found as attachments in the articles posted here or within this book (search for the keyword *Excel* or *spreadsheet* in this PDF document.) In particular:

- Advanced Machine Learning with Basic Excel
- Black box confidence intervals - See section 2, Source Code
- Model-free confidence intervals

Also, a list of articles about data science with Excel can be found here.

Visualization

Many visualizations can be performed with R (see section on R in this article), Excel, or Python libraries. Specific types of charts (graphs or decision trees) require special functions. The most popular software is Tableau. Birt (by Accenture) is popular for dashboards and Visio for diagrams. Most tools allow you to produce maps, scatterplots and various types of visualizations. For a reference, follow this link or search for visualization cheat sheet. See also here (dataviz with R.) Also, search the web to learn how to avoid creating bad charts.

6. Machine Learning

To understand the difference between machine learning and data science, read this article. A large list of machine learning references can be found here. It covers the following domains:

- Support Vector Machines
- Clustering
- Dimensionality Reduction
- Anomaly Detection
- Recommender Systems
- Collaborative Filtering
- Large Scale Machine Learning

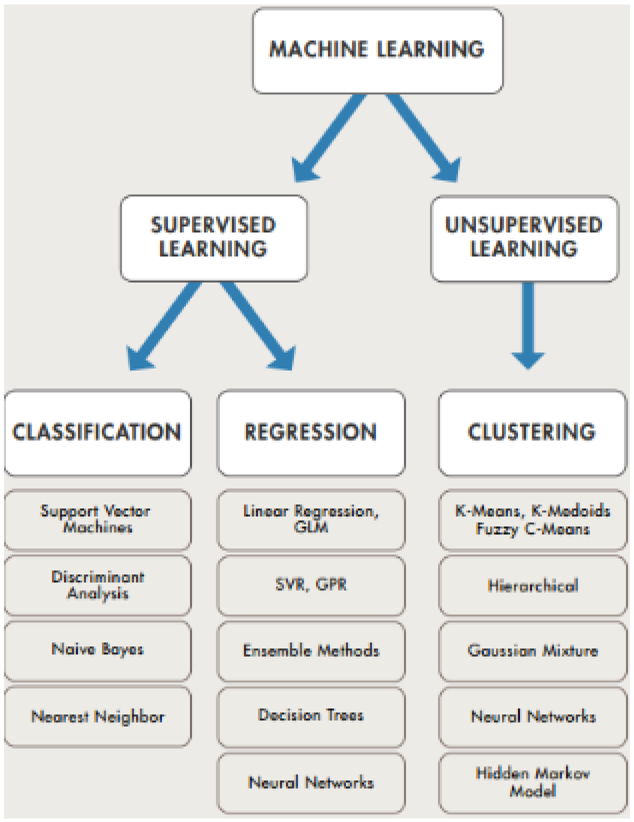
- Deep Learning
- Sparse Coding

Also check out our resources section. Some of these resources include:

- 34 Great Articles and Tutorials on Clustering
- 22 Great Articles and Tutorials on Classification Methods
- 13 Great Articles and Tutorials about Correlation
- 26 Great Articles and Tutorials about Regression Analysis
- 15 Great Articles About Decision Trees
- 27 Great Resources About Logistic Regression
- Four Great Pictures Illustrating Machine Learning Concepts
- 11 Great Hadoop, Spark and Map-Reduce Articles
- 20 Cheat Sheets: Python, ML, Data Science, R, and More
- 25 Great Articles About SQL and NoSQL
- 15 Great Articles about Bayesian Methods and Networks
- 22 Great Articles About Neural Networks
- 21 Great Articles and Tutorials on Time Series
- 15 Deep Learning Tutorials
- 11 Great Articles About Natural Language Processing (NLP)
- Statistical Concepts Explained in Simple English
- Machine Learning Concepts Explained in One Picture

Algorithms

The picture below summarizes the main types of algorithms used in machine learning (ML). Source: MathWorks. See also this list and for more details, follow this link. Deep learning (neural networks with several intermediate layers) is getting more and more popular: click here for a starting point.



Getting started

A great ML cheat sheet can be found here. For a search engine focusing exclusively on ML and related topics, click here. For competitions, visit Kaggle.com. For job interview questions and answers, follow this link or download this document. Glossaries can be found here. For a great, interactive tutorial, check out Ajit Jaokar's book series, here. Many ML projects (and hopefully yours in the future) are hosted on Github.com. To ask questions, one of the best platforms is Stackexchange: see here. For more articles from the author of this book, visit this web page.

ML Applications

The following articles illustrate how ML is used in business.

- Unusual ML application: gaming technology
- 22 tips for better data science
- 21 data science systems used by Amazon to operate its business
- 40 Techniques Used by Data Scientists
- Designing better algorithms: 5 case studies
- Architecture of Data Science Projects
- 33 unusual applications of machine learning
- 24 Uses of Statistical Modeling (Part II) | (Part I)

Data sets and sample projects

Open source data sets can be found here and here. Here is another list featuring 100 data sets. KDNuggets.com also maintains a fairly comprehensive list of data sets. The following articles also feature interesting data sets:

- Source code for our Big Data keyword correlation API
- Great statistical analysis: forecasting meteorite hits
- Fast clustering algorithms for massive datasets
- 53.5 billion clicks dataset available for benchmarking and testing
- Over 5,000,000 financial, economic and social datasets
- New pattern to predict stock prices, multiplies return by factor 5
- 3.5 billion web pages
- Another large data set - 250 million data points - available for do...
- 125 Years of Public Health Data Available for Download
- From the trenches: real data science project (Google Analytics)

You can also start working on the following projects:

- Analyzing 40,000 web pages to **optimize content**: see here. Work on the data to identify the types of articles and other metrics associated with success (and how do you measure success in the first place?), such as identifying great content for our audience, forecasting articles' lifetime and page views based on subject line or category, assessing impact of re-tweets, likes, and sharing on traffic, and detecting factors impacting Google organic traffic. Also, designing a tool to identify new trends and hot keywords. See also here for a related NLP project.

- **Categorization** of data scientists. Also, create a list of top 500 data scientists using public data such as Twitter, and rate them based on number of followers or better criteria. Also identify new stars and trends - note that new stars have fewer followers even though they might be more popular, as it takes time to build a list of followers. Classify top practitioners into a number of categories (unsupervised clustering) based on their expertise (identified by keywords or hashtags in their postings or LinkedIn profile). Filter out automated from real tweets. Finally, create a taxonomy of data scientists: see here for a starting point.

- **Spurious correlations** in big data, how to detect and fix it. You have n = 5,000 variables uniformly distributed on [0,1]. What is the expected number m of correlations that are above p = 0.95? Perform simulations or find theoretical solution. Try with various values of n (from 5,000 to 100,000) and p (from 0.80 to 0.99) and obtain confidence intervals for m (m is a function of n and p). Identify better indicators than correlation to measure whether two time series are really related. The purpose here is twofold: (1) to show that with big data, your strongest correlations are likely to be spurious, and (2) to identify better metrics than correlation in this context. A starting point is my article about the curse of big data, also in my book pages 41-45. Or read my article on strong correlations.

- Perform **simulations** to assess the probability of some extreme events (useful in fraud detection problems, to detect fake or shared profiles or fake reviews). See here, also here (are there too many twin points in this dataset?) and here. Simulations are useful in pattern detection problems. For number theory applications (experimental mathematics involving chaotic sequences) - with several statistical tests being used to assess departure from randomness - check out Appendix B in this book, or in my book on stochastic processes, here.