PYTHON DATA SCIENCE

THE BIBLE

THE ULTIMATE BEGINNER'S GUIDE TO LEARN DATA ANALYSIS, FROM THE BASICS TO ADVANCE CONTENT! (PYTHON PROGRAMMING, PYTHON CRASH COURSE, CODING MADE EASY BOOK)

MARK SOLOMON BROWN

PYTHON DATA SCIENCE

THE BIBLE

THE ULTIMATE BEGINNER'S GUIDE TO LEARN DATA ANALYSIS, FROM THE BASICS TO ADVANCE CONTENT! (PYTHON PROGRAMMING, PYTHON CRASH COURSE, CODING MADE EASY BOOK)

MARK SOLOMON BROWN

Python Data Science

The Bible.
The Ultimate Beginner's Guide to Learn Data Analysis, from the Basics and Essentials, to Advance Content! (Python Programming, Python Crash Course, Coding Made Easy Book)

Mark Solomon Brown

Table Of Contents

Introduction

Dear reader,

Inside this technology eBook guide related to Python programming language, you can learn and find incredibly useful information about one of the most popular and the most used programming language in the IT world - Python. You will read more about the history of Python programming language and its usage, the most important information about the founder and short brief history for the essence of programming languages.

There will be included the most relevant tips and tricks about how easily you can find solutions and be introduced in today's data science thanks to this programming language. You will make your knowledge deeper than before thanks to the most relevant and understandable information and details for why Python is a first-class tool mainly, its libraries for storing, manipulating, and gaining insight from data.

These professional details, tips and tricks and so much more information about Python programming language, you will get instantly in this eBook from the first hand. Be sure that after finishing with your reading, you will be more introduced and more familiar with using of this programming language for data science and researching, thanks to this awesome tech guide understandable and acceptable for everyone who want to know something new in their life.

Wish you easy and peaceful reading,

Mark Solomon Brown

The history of Python and programming languages

Do you know that you can “communicate” with your computer and it can understand you really clearly? Do you ever imagine what kind of language can you use to give an order to your computer?

That type of language that you can use to communicate mechanically to your computer and give him orders, is called programming language or computer language…

A programming language is a formal language, which is built of set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms, in software engineering for building various software and programs, in digital development for creating websites, web applications, mobile applications and similar things.

There are programmable machines that use a set of specific instructions, rather than general programming languages. Since the early 1800s, programs have been used to direct the behavior of machines such as Jacquard looms, music boxes and player pianos. The programs for these machines did not produce different behavior in response to different inputs or conditions.

Thousands of different programming languages have been created till today and more are being created every year. Many programming languages are written in an imperative form while others languages use the declarative form.

At the University of Manchester, Alick Glennie developed Autocode in the early 1950s. As a programming language, it was powered with compiler to automatically convert the language into machine code. The first code and compiler was developed in 1952 for the Mark 1 computer that will be located at the University of Manchester and it is considered to be the first compiled high-level programming language.

In 1954, FORTRAN was invented at IBM by John Backus. It was the first worldwide high-level general purpose programming language which had a functional implementation as opposed to just paper design. It is still a popular language for high-performance computing and it is used for programs that benchmark and rank the world's fastest supercomputers with extremely high prices and large memory in petabytes.

As the programming and computer’s world started with the huge rise, the programming languages started to be intended more and more in various development sectors and industries around the world. So, one of these well-

known and popular programming languages is Python.

Python is a high-level, general-purpose programming language. Created by Guido van Rossum in 1991, Python is designed with philosophy emphasizes code readability with its notable use of significant whitespace. Python construction and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum, the founder of Python programming language

Python 2.0 was released on 16th of October 2000 with many new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3rd of December 2008. It was a major revision of the language that is not completely backward-compatible.

Due to concern about the amount of code written for Python 2, support for Python 2.7 was extended to 2020. Language developer Guido van Rossum sold the responsibility for the project until July 2018, but now shares his leadership as a member of a five-person steering council.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization named as the Python Software Foundation, manages and directs resources for Python, CPython and IPython development.

Introduction to Python programming language

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed with emphasis on code readability and its syntax form allows programmers to express concepts in fewer lines of code. That save the programmer's time and keep their ideas fresh and tidy.

Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions- Python 2 and Python 3. Both are quite different.

Before starting with Python programming, you need to pick an interpreter to interpret and run the programs properly. There are many interpreters available for free to run Python scripts like Integrated Development Environment or in short IDLE, which is installed when you install the Python software on your computer.

If you are a Linux operating system user, Python programming language and its interpreter comes bundled with the operating system package and its distribution.

There are no separate compilation and execution steps in Python programming, like C and C++. This programming language is directly run the program from the source code. Python converts the source code into an intermediate form called "bytecodes" which is then translated into native language of specific computer to run it. You don't need to initialize the libraries before you start with coding because they are managed and linked by themselves.

Python programs can be developed and executed on multiple operating system platforms like Linux, Windows, Mac, Solaris and so on. It is free and open source programming language and that means that you can manage and organized for your needs and wishes.

During your Python programming, you don't need to take care about tiny details such as managing the memory used by the program and similar things. Python is developed as a high level programming language and everything is managed by the language itself.

This programming syntax form is closer to English language and everyone can learn coding thanks to this easy to learn high-level programming language which gives more emphasis on the solution to the problem rather than the syntax

writing.

Also, Python can be used within C or C++ program to give scripting capabilities for the program's users without managing the library before. Known as the "batteries included" philosophy, it can help the programmer to do various things involving regular expressions, documentation, unit testing, threading, databases, web browsers, email, XML, HTML, cryptography, GUI and many more.

Besides the standard library, there are other various high-quality libraries such as the Python Imaging Library which is an amazingly simple image manipulation library.

Python has been successfully embedded in a number of software products as a scripting language. GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers.

Python has also been used in artificial intelligence and it is often used for natural language processing tasks and data science including researching.

A number of Linux distributions use installers written in Python programming language, for example Ubuntu has the Ubiquity Python with extensive use in the information security industry and exploit development.

Raspberry Pi is a single board computer that also uses Python as its principal user-programming language. This programming language is using in Game development industry and represent the most used and the most popular game developing programming language. Most of the companies are now looking for candidates who know Python programming and having the knowledge of Python may have more chances of impressing the interviewing panel. So, beginners should start learning Python and excel in it because of its high level of understanding and easy to use syntax.

Python is already quite popular and it has hundreds of different libraries and frameworks that can be used by developers. These libraries and frameworks are really useful in saving time which in turn makes Python even more popular and used worldwide.

Some of the popular libraries of Python are NumPy and SciPy for data scientific computing, Django for web development, BeautifulSoup for XML and HTML parsing, scikit-learn for machine learning applications, nltk for natural language processing etc…

Corporate support is a big part of Python's popularity and usage. Many top companies such as Google, Facebook, Amazon, Quora and similar use Python for their products. In fact, Google has practically adopted Python for many of its

platforms and applications. It also provides various guides and tutorials for working with Python in Google's Python Class.

Big data and machine learning are the hottest trends in modern times and Python is used for much of the research and development in these fields. Python is also crucial in related fields such as Artificial Intelligence and data science that stands to reason that this would be a huge factor in the rapid growth of Python. There are many Python tools for analytics and data science such as Scikit-Learn, Theano and much more.

Also, Python is used with big data using tools such as Pandas, PySpark and similar. This programming language is very popular in today's web development sector. It is easy enough to learn and capable of powering some of the most popular sites in the world like Spotify, Instagram, Pinterest, Mozilla, Yelp and others. There are many famous web frameworks in Python that can be used according to the project requirements. If the project is complicated with multiple features, it is better to use a full-stack framework such as Django, Pyramid, TurboGears, etc. In case, the project is comparatively simpler, a micro framework such as Flask, Bottle, CherryPy is a better option. Python is such a fundamental part of the programming world currently that it is even taught in schools and colleges as a core language requirement. This is because Python is in trend with its huge number of uses in data science, machine learning, deep learning, artificial intelligence, etc. So many of the current students are learning Python and it is obvious that its importance will increase even more in the future.

Python comparison with other programming languages

Python is often compared to other interpreted languages such as Java, JavaScript, Perl, Tcl or Smalltalk. Comparisons to C++, Common Lisp and Scheme can also be enlightening. In practice, the choice of a programming language is often dictated by other real-world constraints such as cost, availability, training and prior investment or even emotional attachment.

Python vs. Java

Python programs are generally expected to run slower than Java programs, but they also take much less time to develop. Python programs are typically 3-5 times shorter than equivalent Java programs. This difference can be attributed to Python's built-in high-level data types and its dynamic typing.

For example, a Python programmer wastes no time declaring the types of arguments, variables or libraries and Python's powerful polymorphic list and dictionary types for which rich syntactic support is built straight into the language, find a use in almost every Python program. Because of the run-time typing, Python's run time must work harder than Java's.

For example, when evaluating some expression, it must first inspect the objects to find out their type, which is not known at compile time. It invokes the appropriate addition operation, which may be an overloaded user-defined method. Java, on the other hand, can perform an efficient integer or floating point addition, but requires variable declarations and does not allow overloading of the operator for instances of user-defined classes.

For these reasons, Python is much better suited as a "glue" language, while Java is better characterized as a low-level implementation language. In fact, the two together make an excellent combination. Components can be developed in Java and combined to form applications in Python.

Python can also be used to prototype components until their design can be "hardened" in a Java implementation. To support this type of development, a Python implementation written in Java is under development, which allows calling Python code from Java and vice versa. In this implementation, Python source code is translated to Java bytecode.

Python vs. Java Script

Python is object-based subset and roughly equivalent to JavaScript. Like and JavaScript, Python supports a programming style that uses simple functions and variables without engaging in class definitions. However, for JavaScript, that is all there.

Python, on the other hand, supports writing much larger programs and better code re-use through a true object-oriented programming style, where classes and inheritance play an important role.

Python vs, Perl

Python and Perl come from a similar background - Unix scripting, which both have long outgrown and spot many similar features but have a different philosophy. Perl emphasizes support common application-oriented tasks, for example by having built-in regular expressions, file scanning and report generating features.

Python emphasizes support common programming methodologies such as data structure design and object-oriented programming and encourages programmers to write readable and maintainable code by providing an elegant, but not overly cryptic notation. As a consequence, Python comes close to Perl but beats it in its original application domain. However, Python is a well applied programming language beyond Perl's niche.

Python vs. Tcl

Like the Python programming language, Tcl is usable as an application extension language, as well as a stand-alone programming language. Tcl programming language, which traditionally stores all data as strings, is weak on data structures and executes typical code much slower than Python.

Tcl also lacks features needed for writing large programs, such as modular namespaces. While a typical large application using Tcl usually contains Tcl extensions written in C or C++ that are specific to that application, an equivalent Python application can often be written in "pure” Python. Of course, pure Python development is much quicker than having to write and debug a C or C++ component.

Python has adopted an interface to Toolkit as its standard GUI component library.

Tcl 8.0 locates the speed issuse by providing a bytecode compiler with limited data type support and adds namespaces. However, it is still a much more weird programming language.

Python vs. Smalltalk

Perhaps the biggest difference between Python and Smalltalk is Python's more mainstream syntax, which gives it a leg up on programmer training. Like Smalltalk, Python has dynamic typing and binding, and everything in Python is an object. However, Python distinguishes built-in object types from user-defined classes, and currently doesn't allow inheritance from built-in types. Smalltalk's standard library of collection data types is more refined, while Python's library has more facilities for dealing with Internet and WWW realities such as email, HTML and FTP.

Python has a different philosophy regarding the development environment and distribution of code. Where Smalltalk traditionally has a monolithic "system image" which comprises both the environment and the user's program, Python stores both standard modules and user modules in individual files which can easily be re-arranged or distributed outside of the system.

Python vs. C++

Almost everything said for Java also applies for C++, where Python code is typically 3-5 times shorter than equivalent Java code, it is often 5-10 times shorter than equivalent C++ code. Anecdotal evidence suggests that one Python programmer can finish in two months what two C++ programmers can't complete in a year. Python shines as a glue language, used to combine components written in C++.

Python vs. Lisp and Scheme

These languages are close to Python in their dynamic semantics, but so different in their approach to syntax that a comparison becomes almost a religious argument. It should be noted that Python has introspective capabilities similar to those of Lisp and Python programs can construct and execute program fragments on the fly. Usually, real-world properties are decisive where Common Lisp is big in every sense and the Scheme world is fragmented between many incompatible versions, where Python has a single, free, compact implementation.

Python vs. PHP

From the development point of view, PHP is a web-oriented language. A PHP application is more like a set of individual scripts maybe even with a single semantic entry point.

Python, on the other hand, is a versatile language which can be also applied for web development. A web application based on Python is a full-fledged

application loaded into memory with its internal state, saved from the query to the request.

Choosing between Python or PHP for web applications pay attention to these characteristics:

- Popularity
- Frameworks
- Learning

Trends and popularity mean a lot in today's IT business. Some customers and product owners want to use only the most popular and hyped technologies for their projects. In such a case you can be a very skilled developer with no customers and jobs at all. That is why before learning anything make sure that the subject you want to know will be famous and not forgotten in 1, 5 or 10 years.

However, PHP and Python are among the most popular programming languages in the world and you have nothing to worry about.

PHP was used to build such giant websites as Wikipedia, Yahoo, WordPress, Friendster, MailChimp, Flickr and many others. But don't think that Python is less used. The Python technology was used to build YouTube, Instagram, a desktop version of Dropbox, Reddit, Bitbucket, Quora, Spotify, Pinterest, internal services of Facebook and a part of PayPal system.

The variety of tools is also extremely important while you are choosing your technology. It defines the simplicity and convenience of your work. If a technology offers multiple instruments for different tasks a programmer can be sure that he won't have to do everything from scratch. The most popular frameworks for PHP are Laravel, Symfony, CodeIgniter, Yii 1 and 2, Phalcon and many others.

These tools can help you create powerful and good looking applications. Python, on the other hand, can't boast of such a number of frameworks. The most usable are Django and Flask but, assuring that it will change soon due to the growing community of Python programming language. It is usually the first question which a student wants to find out. The easier education process is, the faster he can start working and making money.

The Python syntax is easier and simple to learn. Unfortunately, there is not the same about PHP. PHP programming language requires much more time and efforts to master. Python lets you make minor mistakes without code breakages.

It gives newbies some confidence to move on with studying.

From the beginners' perspective, you may want to choose something easier, something more flexible and it is Python. Python allows you to create secure applications while PHP requires additional tools for these purposes. But don't forget that PHP was created specifically for web development and it is used here more often.

Python vs. C#

In terms of simplicity, Python was originally made to look like English speech, so many expressions in it are very easy to read, especially if you use suitable variable names. In addition, due to simple syntax, there are no complicated constructions such as syntactic brackets, a huge number of word-modifiers, various C-like constructions and different ways to initialize variables. It all makes the code written in Python easy for understanding and for learning.

At the same time, C#, due to the language heredity, has lots of things from C++ and Java, which is initially expressed in C-like syntax.

What is more, C# syntax makes it necessary to follow certain rules when writing your own methods or inheriting classes, which are accompanied by another stream of word-modifiers.

One doesn't need to forget about blocks of code which should be 'wrapped' in braces. Python doesn't have it everything, but it uses shifts which also make the code look neat.

As for the script writing, it is probably worth to mention that programs which Python calls scripts are actually scripts, they are just files with code that can be easily executed by the interpreter. One can open them in any editor, work with them and then immediately run again. It is a huge advantage if there is no IDE or a compiler at hand. Moreover, with Python it is much easier to write cross-platform scripts which do not even need to be recompiled.

As a cons of Python, can be highlight the need for an installed interpreter on the machine with scripts. Well, or at least an interpreter packed together with the script in one package or executable file. As a result, it will increase the size of the script from a couple of kilobytes to a dozen megabytes.

In return, C # requires IDE for normal programming. As a plus of C#, it has a very strong support for various components of the Windows system when you are writing scripts for Windows. For example, there are built-in tools for working with the registry, WMI, the network and so on. C# allows you to use Win Forms, which makes it very easy to create a graphical interface if it is

suddenly needed after all.

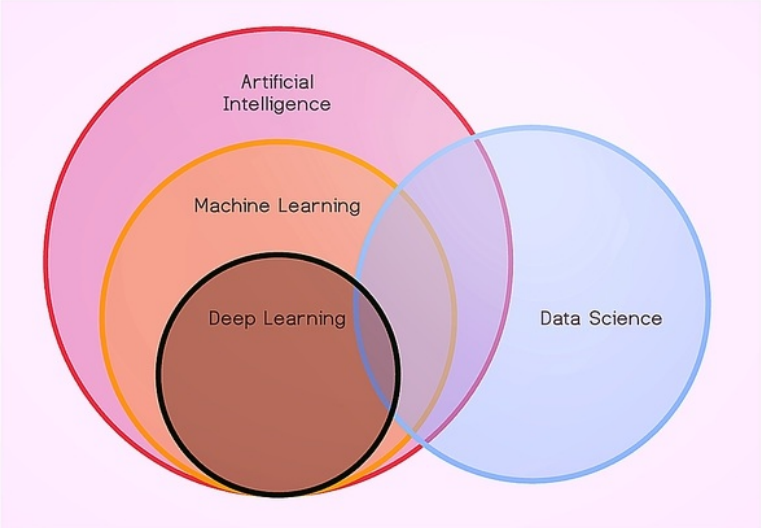
Python is easier to learn, it has many more open source libraries compared to C#. So, the standard library of C# is better than Python's. On the other hand, C# has more features and its performance is higher and it evolves really fast.

Understanding of Data Science

Data Science is not a certain or a single one the kingdom, it is like a combination of various disciplines that are focusing on analyzing data and finding the best solutions based on them. Initially, those tasks were held by math or statistics specialists, but then data-experts began to use machine learning and artificial intelligence, which added optimization and computer science as a method for analyzing data. This new approach turned out to be much faster and effective, and so extremely popular.

So, the popularity of Data Science lies in the fact that it encompasses the collection of large arrays of structured and unstructured data and their conversion into human-readable format, including visualization, work with statistics and analytical methods, machine and deep learning, probability analysis and predictive models, neural networks and their application for solving actual problems.

Artificial Intelligence, machine learning, deep learning and data science undoubtedly, are the most popular technology sectors today. Although, they are somehow related, they are not the same.



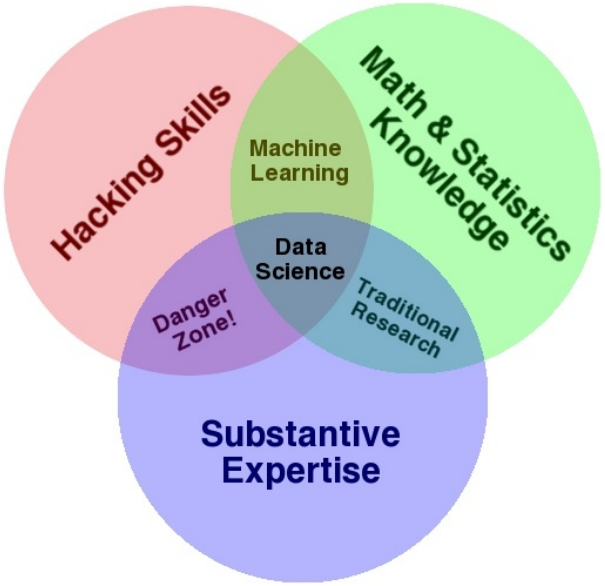
Artificial Intelligence or short AI is the kingdom which, is focusing on the creation of intelligent machines that work and react like a humans. Artificial Intelligence as a study dates back to 1936 when Alan Turing built first AI-powered machines. Despite quite a long history, today Artificial Intelligence in most areas is not able to completely replace a human. The competition of AI with humans in chess and data encryption, are two sides of the same thing.

Machine learning or short ML is a creating tool for extracting knowledge from data. In ML models can be trained on data independently or in stages, like training with a teacher, having human-prepared data or training without a teacher, working with spontaneous, noisy data and similar things.

Deep learning is the creation of multi-layer neural networks in areas where more advanced or fast analysis is needed and traditional machine learning cannot cope. “Depth” provides more than one hidden layer of neurons in the network that conducts mathematical calculations.

Big Data work with huge amounts of often unstructured data. The specifics of the sphere are tools and systems capable of withstanding high loads.

Data Science is the addition of meaning to arrays of data, visualization, collection of insights, and making decisions based on these data. The field specialists use some methods of machine learning and Big Data cloud computing which represent the tools for creating a virtual development environment and much more.



The data scientist is a person who is able to do different actions related to the data, for example, detection of anomalies like abnormal customer behavior, fraud and similar.

The data scientist can also do personalized marketing, including personal e-mail newsletters, retargeting, recommendation systems;

Metric forecasts and performance indicators, quality of advertising campaigns and other activities are included in the data scientist’s work and abilities.

Scoring systems, ability to process large amounts of data and help to make a

decision, for example granting a loan and basic interaction with the client, like standard answers in chat rooms, voice assistants, sorting letters into folders.

To be able to do all of this stuffs and activities, the data scientist have to do:

- Collection Search for channels to collect data
- Check and validation, pruning anomalies that do not affect the result and confuse with further analysis.
- Analysis. The study of data, confirmation of assumptions, conclusions.
- Visualization. Presentation in a form that will be simple and understandable for perception by a person in graphs, diagrams.
- Act. Making decisions based on the analyzed data, for example about changing the marketing strategy, increasing the budget for any activity of the company.

Data Scientist is a person who is better at statistics than any programmer and better at programming than any statistician.

Talking in general about Data Science for serious understanding, you need a fundamental course in probability theory, linear algebra and of course, mathematical statistics. Fundamental mathematical knowledge is important in order to be able to analyze the results of applying data processing algorithms. Despite the abundance of mathematical formulations and evidence, all methods are accompanied by practical examples and exercises.

In fact, a great advantage would be to immediately get acquainted with the basics of programming. Start learning one language and focus on all the shades of programming through the syntax of that language.

For example, Python programming language is an excellent chance to you to enter the data scientist’s world. Firstly, it is perfect for beginners to learn and it has a relatively simple syntax. Secondly, Python combines the demand for specialists and is multifunctional.

Time is a precious resource, so it’s better not to disintegrate at once and not just waste it. After you learn the basics of Python, you need to spend time getting to know the main libraries.

Machine learning allows you to train computers to act independently so that we do not have to write detailed instructions for performing certain tasks. For this reason, machine learning is of great value for almost any area, but first of all, it will work well where there is Data Science.

Supervised Learning is now the most developed form of Machine Learning. The idea here is that you have historical data with some notion of the output variable. Output Variable is meant for recognizing how you can a good combination of several input variables and corresponding output values as historical data presented to you and then based on that you try to come up with a function which is able to predict an output given any input. So, the key idea is that historical data is labeled. Labeled means that you have a specific output value for every row of data, that is presented to it.

Unsupervised learning doesn't have the luxury of having labeled historical data input-output. It allows you to identify what is known as patterns in the historical input data and interesting insights from the overall perspective. So, the output here is absent and all you need to understand is that is there a pattern being visible in the unsupervised set of input. The beauty of unsupervised learning is that it lends itself to numerous combinations of patterns, that's why unsupervised algorithms are harder.

Reinforcement learning or short RL occurs when you present the algorithm with examples that lack labels, as in unsupervised learning. However, you can accompany an example with positive or negative feedback according to the solution the algorithm proposes. RL is connected to applications for which the algorithm must make decisions, and the decisions bear consequences. It is just like learning by trial and error. An interesting example of RL occurs when computers learn to play video games by themselves.

Data Mining is an important analytic process designed to explore data. It is the process of analyzing hidden patterns of data according to different perspectives for categorization into useful information, which is collected and assembled in common areas, such as data warehouses, for efficient analysis, data mining algorithms, facilitating business decision making and other information requirements to ultimately cut costs and increase revenue.

Data Visualization is a general term that describes an effort to help people understand the significance of data by placing it in a visual context. After you have studied everything you need to analyze the data and try your hand at open tasks and contests, then start looking for a job. Of course, you will say only good things, but you have the right to doubt your words.

Today anyone can become Data Scientist. There is everything you need for this in the public domain, like: online-courses, books, competitions for gaining practical experience and so on. It is a good for the first glance, but you shouldn't learn it just because of hype. All we hear about Data Science it is unbelievably

cool and it is the most requested job of the 21st century. If these things are the main motivation for you, nothing ever will work.

Becoming a self-taught Data Scientist is possible. However, the key to your success is a high motivation to regularly find time to study data analysis and its practical application. Most importantly, you have to learn to get satisfaction in the process of learning and working. Think about it.

Getting Started with Python for Data Scientists

In short, understanding Python is one of the valuable skills needed for a data science career. Though it hasn't always been, Python is the programming language of choice for data science. Here is a brief history:

- In 2016, it overtook R on Kaggle, the premier platform for data science competitions.
- In 2017, it overtook R on KDNuggets's annual poll of data scientists' most used tools.
- In 2018, 66% of data scientists reported using Python daily, making it the number one tool for analytics professionals.

Data science experts expect this trend to continue with increasing development in the Python ecosystem. While your journey to learn Python programming may be just beginning, it is nice to know that employment opportunities are abundant and growing fast as well. According to today's marketplace analytics, the average salary for a Data Scientist is $127,900.

So, the future is bright for data science, and Python is just one piece of the proverbial pie. Fortunately, learning Python and other programming fundamentals is as attainable as ever. If you apply yourself and dedicate meaningful time to learning Python, you have the potential to not only pick up a new skill, but potentially bring your career to a new level.

In addition to learning Python in a course setting, your journey to becoming a data scientist should also include soft skills.

The fundamentals

This first step is where you will learn Python programming basics. You will also want an introduction to data science.

One of the important tools you should start using early in your journey is Jupyter Notebook, which comes prepackaged with Python libraries to help you learn these two things.

By joining a community, you will put yourself around like-minded people and increase your opportunities for employment. According to the Society for Human Resource Management, employee referrals account for 30% of all hires.

The Command Line Interface (CLI) lets you run scripts more quickly, allowing you to test programs faster and work with more data.

You may be surprised by how soon you'll be ready to build small Python projects. Try programming things like calculators for an online game, or a program that fetches the weather from Google in your city. Building mini projects like these will help you learn Python. Programming projects like these are standard for all languages and a great way to increase your understanding of the basics.

You should start to build your experience with APIs and begin web scraping. Beyond helping you learn Python programming, web scraping will be useful for you in gathering data later.

Enhance your coursework and find answers to the Python programming challenges you encounter. Read guidebooks, blog posts and even other people's open source code to learn Python and data science best practices and get new ideas.

SQL is used to talk to databases to alter, edit, and reorganize information. SQL is a staple in the data science community, as 40% of data scientists report consistently using it.

Unlike some other programming languages, in Python, there is generally a best way of doing something. The three best and most important Python libraries for data science are NumPy, Pandas, and Matplotlib.

NumPy and Pandas are great for exploring and playing with data. Matplotlib is a data visualization library that makes graphs like you would find in Excel or Google Sheets.

Python has a rich community of experts who are eager to help you learn Python. Resources like Quora, Stack Overflow, and Dataquest's Slack are full of people excited to share their knowledge and help you learn Python programming. We also have an FAQ for each mission to help with questions you encounter throughout your programming courses with Dataquest.

Git is a popular tool that helps you keep track of changes made to your code, which makes it much easier to correct mistakes, experiment, and collaborate with others.

For aspiring data scientists, a portfolio is a must. These projects should include several different datasets and should leave readers with interesting insights that you have gleaned. Your portfolio doesn't need a particular theme. You can find datasets that interest you then come up with a way to put them together.

One of the nice things about data science is that your portfolio doubles as a resume while highlighting the skills you have learned, like Python programming.

While learning Python for data science, you will also want to get a solid background in statistics. Understanding statistics will give you the mindset you need to focus on the right things, so you will find valuable insights rather than just executing code.

Your data science journey will be full of constant learning, but there are advanced courses you can complete to ensure you have covered all the bases.

You will need to be comfortable with regression, classification, and k-means clustering models. You can also step into machine learning and bootstrapping models and creating neural networks using scikit-learn.

At this point, programming projects can include creating models using live data feeds. Machine learning models of this kind adjust their predictions over time. Data science is an ever-growing field that spans numerous industries.

At the rate that demand is increasing, there are exponential opportunities to learn. Continue reading, collaborating, and conversing with others, and you are sure to maintain interest and a competitive edge over time.

There are a lot of estimates for the time it takes to learn Python. For data science specifically, estimates a range from 3 months to a year of consistent practice. Really, it all depends on your desired timeline, free time that you can dedicate to learn Python programming and the pace at which you learn.

Dataquest’s courses are created for you to go at your own speed. Each path is full of missions, hands-on learning and opportunities to ask questions so that you get can an in-depth mastery of data science fundamentals.

Python programming beginnings

Some programming languages live in the heart of data science. Python is one of those languages. It is an integral ingredient for Data Science and vice versa. And actually, it would take prominently long to explain why.

Python provides great functionality to deal with mathematics, statistics and scientific function. When it comes to data science application, it provides extensive libraries to deal with. Not to mention it is open-source, interpreted, high-level tool.

Most importantly, Python is widely used in the scientific and research communities because of its ease of use and its simple syntax makes it easy to adapt for people who even do not have an engineering background.

Python is a comparatively simple language and has an actually pretty handy syntax. It properly supports programmers to program without prepared code and

instantly recognize weak spots.

The best distribution for installation available for Linux, Windows, and Mac OS X is Anaconda distribution. The reason for this is that it contains all necessary libraries for data analysis.

The simplest directive or a language construct that specifies how a compiler should process its input in Python is the “print” directive. It simply prints out a line and also includes a newline, unlike in C, for example: print(“Printing the string.”)

There are lots of specific details concerning syntax you need to learn and this article is no way enough to amply fulfill this task.

Python is a strongly typed language and a strongly-typed language means that it is one in which each type of data is predefined as part of language and all constants or variables defined for a given program must be described with one of the data types, but at the same time it is dynamically typed. There is no declaration of a variable, just an assignment statement.

Python is case-sensitive language and object-oriented language. Everything in Python is an object: numbers, dictionaries, user-defined, and built-in classes.

This programming language has no mandatory operator completion characters, block boundaries are defined by indents. The indent starts a new block, the lack of indent finishes it. Expressions that are waiting for a new indent end with a colon character ‘:’. Single-line comments begin with a pound symbol (hashtag) ‘#’ and for multi-line comments, string literals are used, enclosed in triple apostrophes or triple quotes.

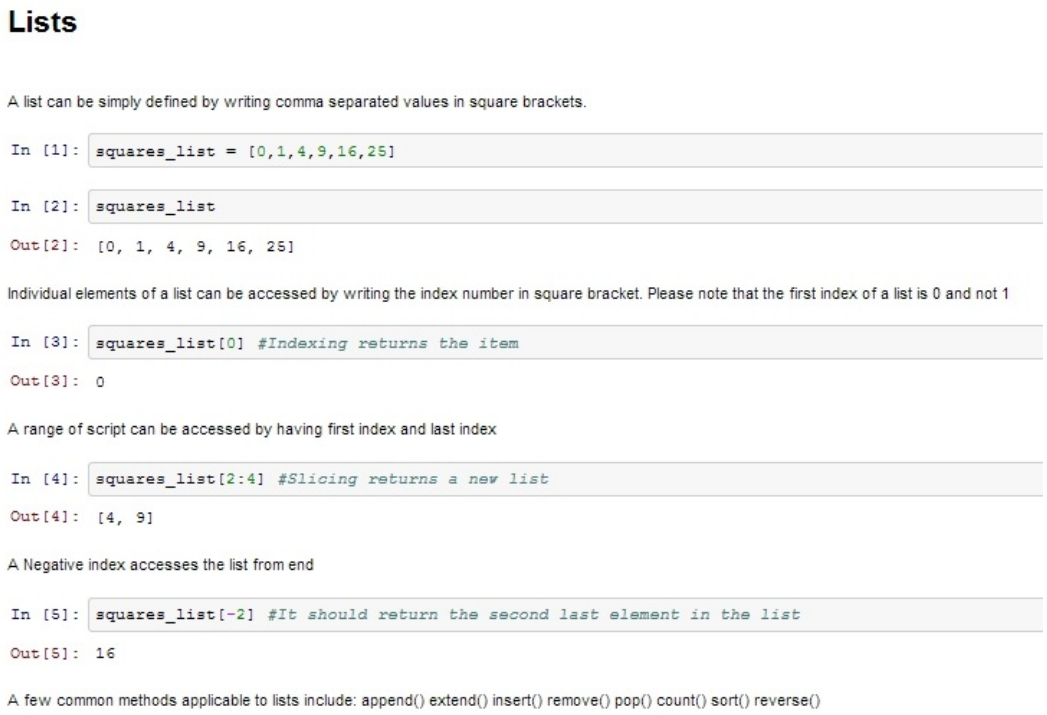
Values are assigned using an equal sign “=” and equality is checked using two equal signs “==”. You can increase or decrease the values using the + = and — = operators, respectively, by the value specified to the right of the operator. This works for many data types, including strings.

In Python, there are many data structures: lists, tuples, and dictionaries. Sets are also available but only in Python 2.5 and later versions. Lists are like one-dimensional arrays but, you can also create lists of other lists and get a multidimensional array, dictionaries are associative arrays and tuples are unchangeable one-dimensional arrays. In Python, “arrays” can be of any type, so you can mix, for example, integers, strings, etc... The index of the first element in arrays of all types is 0 and the last element can be obtained by index - 1.

You can work only with a part of the array elements using a colon ‘:’. In this case, the index before the colon indicates the first element of the used part of the

array and the index after the colon indicates the element after the last element of the used part of the array. If the first index is not specified, the first element of the array is used. If the second is not specified, the last element will be the last element of the array.

A **List** is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item. Just as strings are defined as characters between quotes, lists are defined by having values between square brackets. Here is an example picture of Python lists:



Lists

A list can be simply defined by writing comma separated values in square brackets.

```
In [1]: squares_list = [0,1,4,9,16,25]

In [2]: squares_list
Out[2]: [0, 1, 4, 9, 16, 25]
```

Individual elements of a list can be accessed by writing the index number in square bracket. Please note that the first index of a list is 0 and not 1

```
In [3]: squares_list[0] #Indexing returns the item
Out[3]: 0
```

A range of script can be accessed by having first index and last index

```
In [4]: squares_list[2:4] #Slicing returns a new list
Out[4]: [4, 9]
```

A Negative index accesses the list from end

```
In [5]: squares_list[-2] #It should return the second last element in the list
Out[5]: 16
```

A few common methods applicable to lists include: append() extend() insert() remove() pop() count() sort() reverse()

Strings can simply be defined by the use of single (‘), double (”) or triple (”’) inverted commas. Strings enclosed in tripe quotes (”’) can span over multiple lines and are used frequently in doc strings that represent the Python’s way of documenting functions. The sign slash ‘\’ is used as an escape character. Please note that Python strings are immutable, so you cannot change part of strings.

The example of Python strings:

Strings

A string can be simply defined by using single ('), double (") or triple (""") quotation

```
In [6]: greeting = 'Hello'
        print greeting[1]          # Return character on the index 1
        print len(greeting)        # Prints length of string
        print greeting + 'World'   # String Concatenation

        e
        5
        HelloWorld
```

Raw strings can be used to pass on string as is. Python interpretter does not alter the string, if you specify a string to be raw. Raw strings can be defined by adding r to the string

```
In [8]: stmt = r'\n is a newline character by default.'
        print stmt

        \n is a newline character by default.
```

Python strings are immutable and hence can be changed. Doing so will result in an error

```
In [9]: greeting[1:] = 'i' #Trying to change Hello to Hi. Should result in an error

        ---------------------------------------------------------------------------
        TypeError                                 Traceback (most recent call last)
        <ipython-input-9-2da872e33998> in <module>()
        ----> 1 greeting[1:] = 'i' #Trying to change Hello to Hi. Should result in an error

        TypeError: 'str' object does not support item assignment
```

Common string methods include lower(), upper(), strip(), isdigit(), isspace(), find(), replace(), split() and join(). These are usually very helpful when you need to perform data manipulations or cleaning on text fields.

What are libraries in programming? It is a collection of precompiled routines that a program can use. The routines, sometimes called modules, are stored in object format. Libraries are particularly useful for storing frequently used routines because you do not need to explicitly link them to every program that uses them. Libraries save time because you do not need to construct functions from scratch.

- Read general information
- Open Jupyter Notebook and load the library.
- See how the library works, using the instructions for working with the library.
- Allocate 30 minutes for the study of reference information.

Using this learning path, you will master the library enough to start using it in your work.

First, start learning NumPy as it is the fundamental package for scientific computing with Python. A good understanding of Numpy will help you use tools like Pandas more effectively.

The things you should learn about Numpy:

- basic concepts of Numpy

- The most frequently performed operations in Numpy
- Indexing and slicing of arrays
- Indexing using integer arrays
- Transposing an array
- Universal functions,
- Data processing using arrays
- Frequently used statistical methods

Pandas contain high-level data structures and manipulation tools to make data analysis fast and easy in Python. Work with this library is built on top of NumPy.

Things you shoul learn about Pandas:

- Series
- Data frames
- Dropping entries from an axis
- Working with missing values

For two-dimensional or three-dimensional data visualization, you can learn more about Matplotlib. It is cumbersome yet powerful tool. With Matplotlib you can quickly generate line graphs, histograms, pie charts, and much more.

Things you should learn about Matplotlib are creating different types of visualizations depending on the message you want to convey and learning how to build complex and customized plots based on real data.

There are so many additional Python libraries, here are the most used:

Scipy is a Python module for linear algebra, integration, optimization, statistics and other frequently used tasks in data science. It is highly user-friendly and provides for fast and convenient N-dimensional array manipulation.

PyTorch is based on Torch, is an open-source Machine Learning library that was primarily built for Facebook’s artificial intelligence research group. While it is a great tool for natural language processing and deep learning, it can also be leveraged effectively for data science.

Scikit-learn is a module focused on Machine Learning that is built on top of SciPy. The library provides a common set of ML algorithms through its consistent interface and helps users quickly implement popular algorithms on data sets. It also has all the standard tools for common ML tasks like

classification, clustering, and regression.

Practice makes perfect and especially when it comes to data science.

Now the ball is in the court of practice and starts analytic work with Python. For me personally, the most effective solutions at this stage is to participate in Kaggle contests, to invent and solve a problem yourself, to complete a practical course on data analysis in Python.

Kaggle often holds data analysis contests. The first one need to participate in contests without prize because they are the easiest and more beginners-friendly. With time you can move to more complex tasks.

Let’s imagine a marketer who is tired of staying up late at work due to the fact that he has to manually collect and process data and make visual reports based on them. To simplify his work and return home on time, he sets the task to automate this process using Python, and solves it.

In a similar way, you should find something that makes it hard for you to work with. Then your task is to think about how to fix it. The only thing that can prevent you is ignorance of the sequence of actions. From this, you can skip the necessary steps and fail. Either get stuck in the middle, not knowing how to proceed.

One of the easiest mistakes you can make when mastering Python is attempting to learn too many things especially libraries at once. When you try to learn like that, you spend too much time switching between different concepts, getting frustrated and move on to something else.

Data Analysis and Libraries in Python

Python language is already assisting developers in creating PC games, mobile apps and other enterprise applications. Python with more than 137,000 libraries helps in various ways. In this data-centric world, where consumers demand relevant information in their buying journey, companies also require data scientists to avail valuable insights by processing massive data sets.

This information lead them in critical decision making, streamlining business operations and thousands of other tasks which require valuable information to accomplish efficiently. With this increased demand for data scientists, beginners and professionals are looking for resources to learn this art of analyzing and representing data. There are a few Python libraries which are very helpful in the whole data science-related operations…

1. NumPy

NumPy is the first choice among developers and data scientists who are aware of the technologies which are dealing with data-oriented stuff. It is a Python package available for performing scientific computations. It is registered under the BSD license. Through NumPy, you can leverage n-dimensional array objects, C, C++, Fortran program based integration tools, functions for performing complex mathematical operations like Fourier transformation, linear algebra, random number etc.

One can also use NumPy as a multi-dimensional container to treat generic data. Thus, you can effectively integrate your database by choosing varieties of operations to perform with. NumPy is installed under the Tensor Flow and other complex machine learning platforms empowering their operations internally.

Since it is an Array interface, it allows us multiple options to reshape large datasets. It can be used for treating images, sound waves representations, and other binary operations.

If you have just marked your presence in this data science or ML field, you must have a great understanding of NumPy to process your real-world data sets.

2. Theano

Theano is another useful Python library assists data scientists in performing large multi-dimensional arrays related computing operations. It is more like TensorFlow but the only difference is, it is not that efficient.

It is getting used for distributed and parallel computing based tasks. Through it,

you can optimize, express or evaluate you array-enabled mathematical operations. It is tightly coupled with NumPy powered by implemented numpy.ndarray function.

Due to GPU based infrastructure, it holds the capability to process operations in faster ways than CPU. It stands fit for speed and stability optimizations delivering us the expected outcomes.

For faster evaluation, its dynamic C code generator is popular among data scientists. Here, they can perform unit-testing to identify flaws in the whole model.

3. Keras

Keras is one of the most powerful Python libraries which allow high-level neural networks APIs for integration. Theses APIs execute over the top of TensorFlow, Theano and CNTK. Keras was created for reducing challenges faced in complex researches allowing them to compute faster. For one who is using deep learning libraries for their work, Keras is the best option.

It allows fast prototyping, supports recurrent and convolution networks individually and also their combination, execution over GPU and CPU.

Keras provides a user-friendly environment reducing your effort in cognitive load with simple APIs giving us the required results. Due to its modular nature, one can use varieties of modules from neural layers, optimizers, activation functions etc.., for developing a new model.

It is an open source library written in Python. For data scientists having trouble adding new modules, Keras is a good option where they can simply add a new module as classes and functions.

4. PyTorch

PyTorch is considered one of the largest machine learning libraries for data scientists and researchers. It helps them in dynamic computational graphs design, fast tensor computations accelerated through GPUs and various other complex tasks. In neural network algorithms, PyTorch APIs plays an effective role.

The hybrid front-end PyTorch platform is very easy to use allows us transitioning in graph mode for optimizations. For achieving accurate results in asynchronous collective operations and establishing a peer to peer communication it provides a native supports to the users.

With native Open Neural Network Exchange or in short ONNX support, one can

export models to leverage visualizers, platforms, run-times, and various other resources. The best part of PyTorch it enables a cloud-based environment for easy scaling of resources used in deployment or testing.

It is developed on the concept of another Machine Learning library called as Torch. Since the past few years, PyTorch is getting more popular among data scientists due to trending data-centric demands.

5. SciPy

SciPy is another Python library for researchers, developers and data scientists. Do not get confused with the SciPy stack and library. It provides statistics, optimizations, integration and linear algebra packages for computation. It is based on NumPy concept to deal with complex mathematical problems.

It provides numerical routines for optimization and integration. It inherits varieties of sub-modules to choose from. If you have just started your data science career, SciPy can be very helpful to guide you throughout the whole numerical computations thing.

We can see how Python programming is assisting data scientists in crunching and analyzing large and unstructured data sets. Other libraries like TensorFlow, SciKit-Learn, Eli5 are also available to assist them throughout this journey.

6. Pandas

Pandas referred as a Python Data Analysis Library. This library is another open source Python library for availing high-performance data structures and analysis tools. It is developed over the Numpy package. It contains DataFrame as its main data structure.

With DataFrame you can store and manage data from tables by performing manipulation over rows and columns. Methods like square bracket notations reduce person’s effort in data analysis tasks like square bracket notations. Here, you will get tools for accessing data in-memory data structures performing read and write tasks even if they are in multiple formats such as CSV, SQL, HDFS or excel etc.

```
In [6]: super_tree.head()
```

Out[6]:

	day	my_date	user_id	event_type
0	day_1	2017-12-01	1000007	sent_a_super_tree
1	day_1	2017-12-01	1000010	sent_a_super_tree
2	day_1	2017-12-01	1000011	sent_a_super_tree
3	day_1	2017-12-01	1000019	sent_a_super_tree
4	day_1	2017-12-01	1000022	sent_a_super_tree

7. PyBrain

PyBrain is another powerful modular machine learning library available in Python. PyBrain stands for Python Based Reinforcement Learning, Artificial Intelligence, and Neural Network Library. For entry-level data scientists, it offers flexible modules and algorithms for advanced research. It has varieties for algorithms for evolution, neural networks, supervised and unsupervised learning. For real-life tasks, it has emerged as the best tool which is developed across the neural network in the kernel.

8. SciKit-Learn

Scikit-Learn is a simple tool for data analysis and mining-related tasks. It is open-source and licensed under the BSD. Anyone can access or reuse it in various contexts. SciKit is developed over the Numpy, Scipy, and Matplotlib. It is being used for classification, regression and clustering o manage spam, image recognition, drug response, stock pricing, customer segmentation etc. It also allows dimensionality reduction, model selection and pre-processing.

9. Matplotlib

This 2D plotting library of Python is very famous among data scientists for designing varieties of figures in multiple formats which is compatible across their respected platforms. One can easily use it in their Python code, IPython shells or Jupyter notebook, application servers. With Matplotlib, you can make histograms, plots, bar charts, scatter plots etc.

10. Bokeh

Bokeh is one more visualization library for designing interactive plots. Just like the last one, it is also developed on Matplotlib. Due to the used data-driven documents support, it presents interactive designs in the web browser.

11. Tensorflow

This open source library was designed by Google to compute data low graphs with the empowered machine learning algorithms. It was designed to fulfill high demand for the training neural networks work. It is not just limited to the scientific computations performed by Google rater it is widely being used in the popular real-world application.

Due to its high performance and flexible architecture the deployment for all CPUs, GPUs or TPUs becomes easy task performing PC server clustering to the edge devices.

12. Seaborn

Seaborn was designed to visualize the complex statistical models. It has the potential to deliver accurate graphs such as heat maps. Seaborn was created on the concept of Matplotlib and somehow it is highly dependent on that. Minor to minor data distributions can be easily visualized through this library which is why it has become familiar among data scientists and developers.

13. Plotly

Let’s talk about the Plotly which is one of the most famous web-based frameworks for data scientists. This toolbox offers designing of visualization models with varieties of APIs supported by multiple programing languages including Python. You can easily use interactive graphics and numerous robust accessible through its main website plot.ly. For using Plotly in your working model you need to set up available API keys properly. The accessible graphics are processed on the server side and once successfully executed they will appear on your browser screen.

14. NLTK

NLTK is pronounced as the Natural Language ToolKit. As per its name, this library is very helpful for accomplishing Natural language processing tasks. Initially, it was developed to promote the teaching models and other NLP enabled research such as the cognitive theory of artificial intelligence and linguistic models and similar, which has become a successful resource in its field driving the real world innovations from artificial intelligence.

With NLTK one can perform operations like text tagging, stemming, classifications, regression, tokenization, corpus tree creation, name entities recognition, semantic reasoning, and various other complex AI tasks. Now challenging works requiring large building blocks like semantic analysis and automation or summarization has become an easier task which can be easily

completed with NLTK.

15. Gensim

Gensim is an open source Python-based library which allows topic modeling and space vector computations with the implemented varieties of tools. It is compatible with the large texts making efficient operations and their in-memory processing. It uses the NumPy and SciPy modules for providing efficient and easy to handle the environment.

It uses the unstructured digital texts and processes them with the inbuilt algorithms like word2vec, hierarchical Dirichlet processes (HDP), latent Dirichlet allocation (LDA) and latent semantic analysis (LSA).

16. Scrapy

Scrapy is also pronounced as the spider bots. This library is responsible for crawling programs and retrieving of the structured data from the web applications. This open source library is written in Python. As per the name it was designed for scraping.

It is the complete framework with the potential to collect data through APIs and act like a crawler. Through it, one can write codes, reuse universal programs and create scalable crawlers for their application. Scrapy is created across the Spider class which contains the instructions for a crawler.

17. Statsmodels

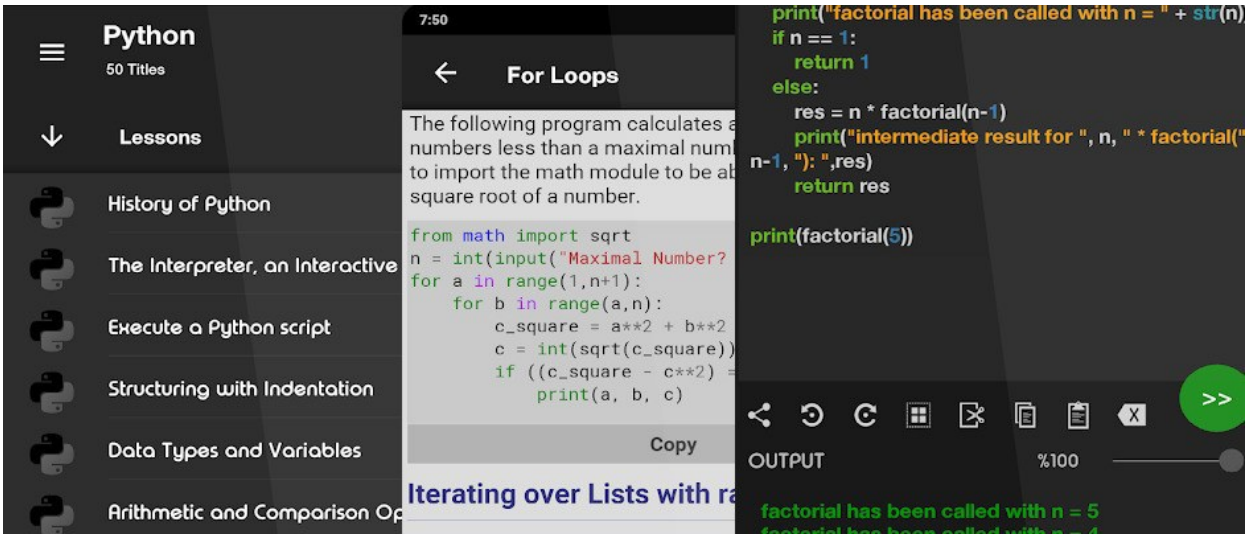
This Python library is responsible for providing the data exploration modules with multiple methods to perform statistical analysis and assertions. The use of regression techniques, robust linear models, analysis models, time series and discrete choice model makes it popular among other data science libraries. It has the plotting function for statistical analysis to achieve high-performance outcomes while processing large statistical data sets.

18. Kivy

This open-source Python library provides a natural user interface which can be easily accessed over the Android, iOS, Linux or Windows. It is licensed open source under MIT. The library is very helpful in building mobile apps and multi-touch applications.

Initially, it was developed for Kivy iOS. It avails the elements like the graphics library, extensive support to hardware such as the mouse, keyboard and wide range of widgets. One can also use it as an intermediate language to create

custom widgets.



19. PyQt

PyQt is a Python binding toolkit for cross-platform GUI. It is implemented as a Python plugin. PyQt is a free application which is licensed under the GNU General Public License. PyQt have almost 440 classes and more than 6000 functions to make a user's journey easier. It includes classes for accessing SQL databases, an XML parser, active X controller classes, SVG support, and many more useful resources to reduce user's challenges.

20. OpenCV

OpenCV is designed for driving growth of the real-time computing application development. It was created by Intel. This open-source platform is licensed under BSD and free to use for anyone. It includes 2D and 3D feature toolkits, object identification algorithms, mobile robotics, face recognition, gesture recognition, motion tracking, segmentation, SFM, AR, boosting, gradient boosting trees, Naive Bayes classifier and many other useful packages.

Even if OpenCV is written in the C++, it provides bindings in Python, Java, and Octave. This application is supported on Windows, Linux, iOS, FreeBSD.

Descriptive statistics

The field of statistics is often misunderstood, but it plays an essential role in our everyday lives. Statistics, done correctly, allows us to extract knowledge from the vague, complex and difficult real world. Wielded incorrectly, statistics can be used to harm and mislead. A clear understanding of statistics and the meanings of various statistical measures is important to distinguishing between truth and misdirection.

The data itself comes from a scraper that scoured one data site. You found some interesting data set and you would like to compare and contrast different data. You will use statistics to describe that data in the data set and derive some insights for yourself. We will perform statistics on some random data and you can use this code to follow along on your own computer during your practice.

```
import csv
with open("the_name.csv", "r", encoding="latin-1") as f:
    wines = list(csv.reader(f))
```

What precisely are statistics? This question is deceptively difficult. Statistics is many, many things, so trying to daylight it into a brief summary would undoubtedly obscure some details. As an entire field, statistics can be thought of as a scientific framework for handling data. This definition includes all the tasks involved with collecting, analyzing and interpretation of data. Statistics can also refer to individual measures that represent summaries or aspects of the data itself. Throughout the article, we will do our best to distinguish between the field and the actual measurements.

What is data? Luckily for us, data is simpler to define. Data is a general collection of observations of the world and can be widely varied in nature, ranging from qualitative to quantitative. Researchers gather data from experiments, entrepreneurs gather data from their users and game companies gather data on their player behavior.

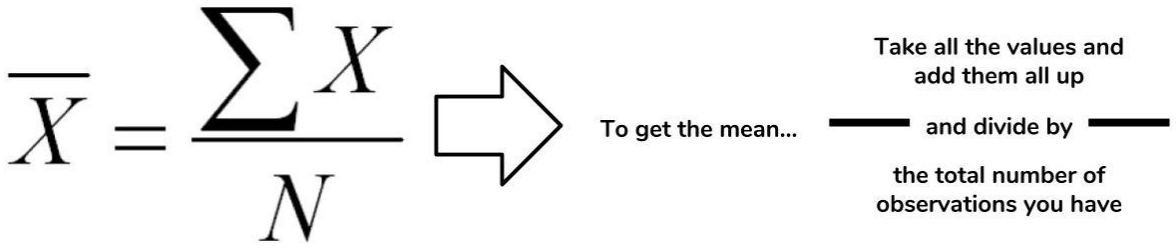
When we have a set of observations, it is useful to summarize features of our data into a single statement called a descriptive statistic. As their name suggests, descriptive statistics describe a particular quality of the data they summarize. These statistics fall into two general categories: the measures of central tendency and the measures of spread.

The measures of central tendency are metrics that represent an answer to the following question: “What does the middle of our data look like?” The word

middle is unclear because there are multiple definitions we can use to represent the middle.

Mean

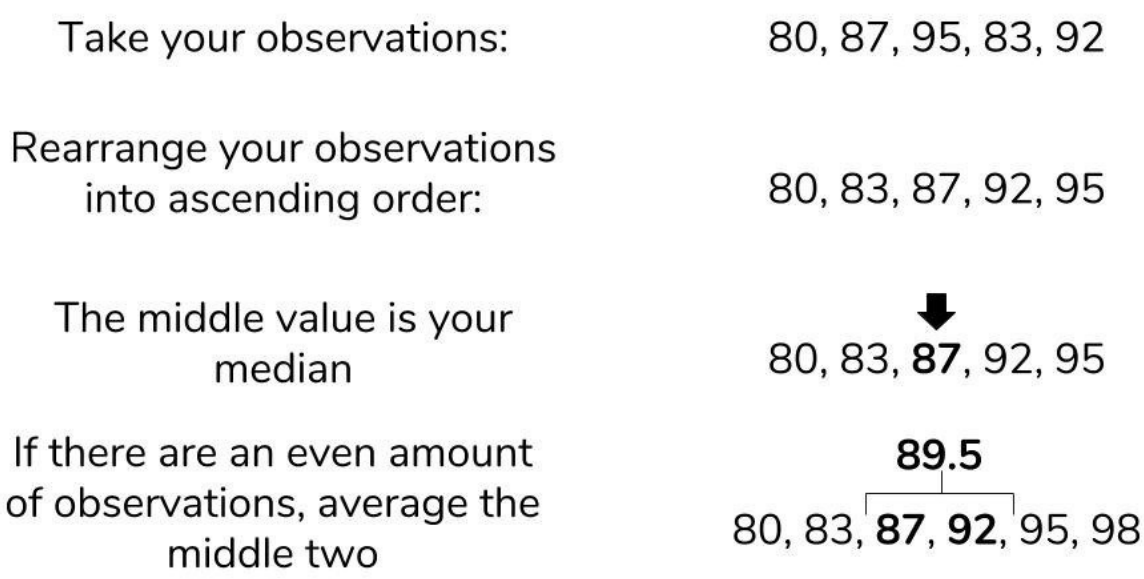
The **mean** is a descriptive statistic that looks at the average value of a data set. While mean is the technical word, most people will understand it as just the average. How is this mean calculated? The picture below is excellent example and takes the actual equation and breaks down the calculation components into simpler terms.



In the case of the mean, the “middle” of the data set refers to this typical value. The mean represents a typical observation in our data set. If we were to pick one of our observations at random, then we are likely to get a value that is close to the mean. The calculation of the mean is a simple task in Python.

Median

The next measure of central tendency is the median. The median also attempts to define a typical value in the data set, but unlike mean, does not require calculation. To find the median, we first need to reorganize our data set in ascending order. Then the median is the value that coincides with the middle of the data set. If there are an even amount of items, then we take the average of the two values that would “surround” the middle.



The example code would be: sum(prices)/len(prices)

Remember that the mean is calculated by summing up all the values we want and dividing by the number of items, while the median is found by simply rearranging items. If we have outliers in our data, items that are much higher or lower than the other values, it can have an adverse effect on the mean. That is to say, the mean is not robust to outliers. The median, not having to look at outliers, is robust to them.

Mode

The last measure of central tendency that we will discuss is the mode. The mode is defined as the value that appears the most frequently in our data. The intuition of the mode as the “middle” is not as immediate as mean or median, but there is a clear rationale. If a value appears repeatedly throughout the data, we also know it will influence the average towards the modal value. The more a value appears, the more it will influence the mean. A mode represents the highest weighted contributing factor to our mean.

The measures of spread or also known as dispersion, answer the question, “How much does my data vary?” There are few things in the world that stay the same every time we observe it. We all know someone who has lamented a slight change in body weight that is due to natural fluctuation rather than outright weight gain. This variability makes the world fuzzy and uncertain, so it is useful to have metrics that summarize this “fuzziness.”

Range

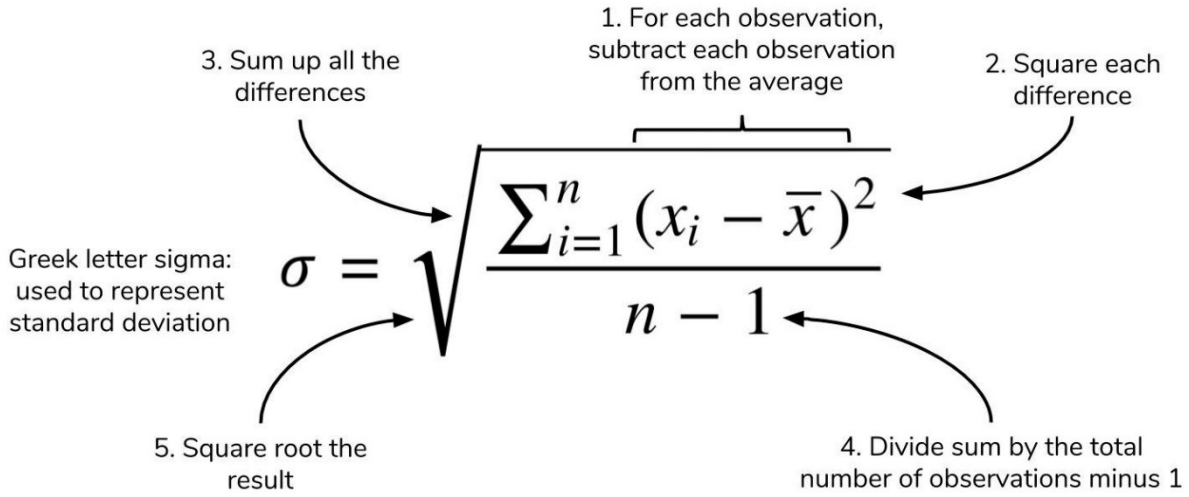
The first measure of spread we will cover is range. Range is the simplest to compute of the measures we'll see: just subtract the smallest value of your data set from the largest value in the data.

The example code would be:

```
price_range = max_price - min_price
print(price_range)
```

Standard deviation

The standard deviation is also a measure of the spread of your observations, but is a statement of how much your data deviates from a typical data point. That is to say, the standard deviation summarizes how much your data differs from the mean. This relationship to the mean is apparent in standard deviation's calculation.



The structure of the equation merits some discussion. Recall that the mean is calculated by summing up all of your observations and dividing it by the number of observations. The standard deviation equation is similar but seeks to calculate the average deviation from the mean, in addition to an extra square root operation. You may see elsewhere that n is the denominator instead of n-1. The specifics of this details is outside the scope of this article, but know that using n-1 is generally considered to be more correct. A link to an explanation is at the end of this article.

We like to calculate the standard deviation to better characterize our data and

scores, so it will need to be created a dedicated function for this. Calculating a cumulative sum of numbers is cumbersome by hand, but Python's for loops make this trivial. We are making our own function to demonstrate that Python makes it easy to perform these statistics, but it is also good to know that the NumPy library also implements standard deviation under std.

Variance

Often, standard deviation and variance are lumped together for good reason.

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n - 1}$$

Variance and standard deviation are almost the exact same thing. Variance is just the square of the standard deviation. Likewise, variance and standard deviation represent the same thing, a measure of spread, but it is worth noting that the units are different. Whatever units your data are in, standard deviation will be the same, and variation will be in that units-squared. A question that many statistics starters ask is, "But why do we square the deviation? While avoiding negative values in the sum is a reason for the squaring operation, it is not the only one.

Like the mean, variance and standard deviation are affected by outliers. Many times, outliers are also points of interest in our data set, so squaring the difference from the mean allows us to point out this significance. If you are familiar with calculus, you will see that having an exponential term allows us to find out where the point of minimum deviation is. More often than not, any statistical analyses you do will require just the mean and standard deviation, but the variance still has significance in other academic areas. The measures of central tendency and spread allow us to summarize key aspects of our data set, and we can build on these summaries to glean more insights from our data.

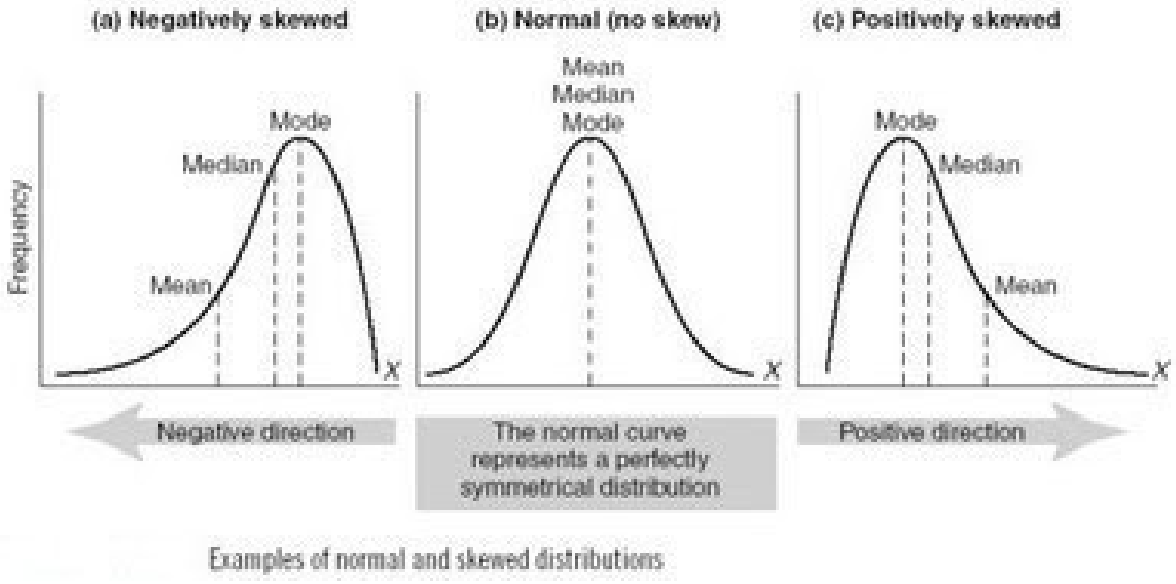
Dispersion

As the measure of central value gives out the central value of the distribution, measures of dispersion describe the spread of data or the variation of data around the central value. Two different distributions can have the same mean or median but different levels of variability.

Measures that describe shape of distribution

Measures of center and spread, tells us just about the central values and spread. How do we describe the shape of the distribution?

Histogram will give us a general idea, but two numerical measures of shape will help us with the precise evaluation of the shape of the distribution. Skewness is the asymmetry in the distribution because of which the curve appears distorted or skewed either to left or right of the normal distribution in a dataset. In other words skewness is the extent to which a distribution differs from a normal distribution.



Examples of normal and skewed distributions

Skewed distributions can be:

- Positively skewed
- Negatively skewed

Positively skewed are the most frequent values and low and tail is towards high values.

Negatively skewed are the most frequent values and high and tail is towards low values.

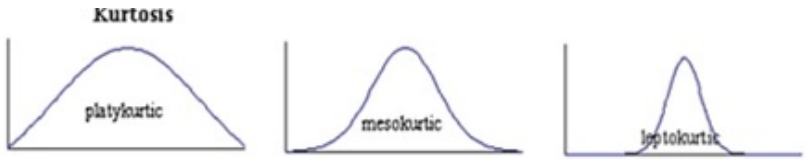
Measures of central tendency can also be used to detect skewness in a distribution. Central tendency values will not be the same if the distribution is skewed.

For example:

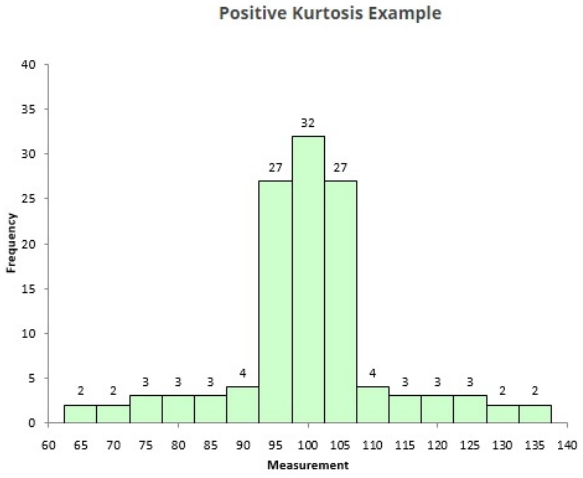
If Mode< Median< Mean then the distribution is positively skewed.

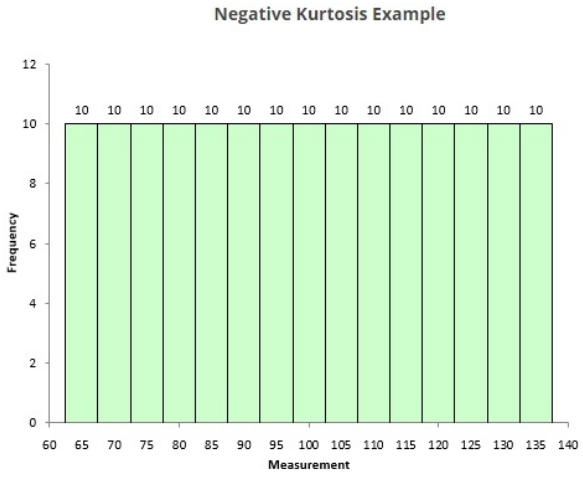
If Mode> Median> Mean then the distribution is negatively skewed.

Kurtosis is the measure of the combined weight of the tails of the distribution relative to the center of the distribution. When a normal distribution is represented via histogram, it shows a bell curve with + and — standard deviations from mean. However, when kurtosis is present then the tails further extend farther than the + and — standard deviations of the normal bell-curved distribution. Kurtosis makes the data to look flatter or less flat as compared to normal distribution. The standard normal distribution has kurtosis of 3.



When the distribution of the data is similar to the normal distribution or the kurtosis of the distribution is 3, it is called as Mesokurtic distribution. Any distribution which has kurtosis more than Normal distribution (K>3), it is called as leptokurtic distribution. This type distribution has positive kurtosis. Distribution which has kurtosis less than Normal distribution (K < 3), it is called as platykurtic(flat) distribution. This type distribution has negative kurtosis.





Data Analysis with Pandas

Pandas, probably is the most popular library for data analysis in Python programming language. This library is a high-level abstraction over low-level NumPy which is written in pure C. It is quite high level, so you don't have to muck about with low level details, unless you really want to.

If you are dealing with complicated or large datasets, seriously consider Pandas. It is based on numpy or scipy, sort of a superset of it. But it gives you a lot of powerful features, like read directly from a Microsoft Excel file, do data joins and some of which features we will go over.

Pandas has two basic data structures: Series and Dataframes.

Series is like numpy's array or dictionary, though it comes with a lot of extra features. Dataframes is a two dimensional data structure that contains both column and row information, like the fields of an Excel file. Remember an Excel file has rows and columns, and an optional header field. All of this data can be represented in a Dataframe.

Pandas is a Python library that provides extensive means for data analysis. Data scientists often work with data stored in table formats like .csv, .tsv, or .xlsx. Pandas makes it very convenient to load, process, and analyze such tabular data using SQL-like queries. In conjunction with Matplotlib and Seaborn, Pandas provides a wide range of opportunities for visual analysis of tabular data.

The main data structures in Pandas are implemented with Series and DataFrame classes. The former is a one-dimensional indexed array of some fixed data type. The latter is a two dimensional data structure "a table" where each column contains data of the same type. You can see it as a dictionary of Series instances. DataFrames are great for representing real data: rows correspond to instances and columns correspond to features of these instances.

Data can be:

- Scalar value which can be integerValue, string
- Python Dictionary which can be Key, Value pair
- Ndarray

Python is a great language for doing data analysis, primarily because of the fantastic ecosystem of data centric Python packages. Pandas is one of those packages, and makes importing and analyzing data much easier.

The most important thing in Data Analysis is comparing values and selecting data accordingly. The “==” operator works for multiple values in a Pandas Data frame too.

In the following example, a data frame is made from a csv file. In the Gender Column, there are only 3 types of values (“Male”, “Female” or NaN). Every row of Gender column is compared to “Male” and a boolean series is returned after that.

```
# importing pandas package
import pandas as pd
# making data frame from csv file
data = pd.read_csv("employees.csv")
# storing boolean series in new
new = data["Gender"] == "Male"
# inserting new series in data frame
data["New"]= new
# display
Data
```

As show in the output image, for Gender= “Male”, the value in New Column is True and for “Female” and NaN values it is False.

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team	New
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing	True
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN	True
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance	False
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance	True
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services	True
5	Dennis	Male	4/18/1987	1:35 AM	115163	10.125	False	Legal	True
6	Ruby	Female	8/17/1987	4:20 PM	65476	10.012	True	Product	False
7	NaN	Female	7/20/2015	10:43 AM	45906	11.598	NaN	Finance	False
8	Angela	Female	11/22/2005	6:29 AM	95570	18.523	True	Engineering	False
9	Frances	Female	8/8/2002	6:51 AM	139852	7.524	True	Business Development	False

In the following example, the boolean series is passed to the data and only Rows having Gender=”Male” are returned.

```
# importing pandas package
import pandas as pd
# making data frame from csv file
```

- data = pd.read_csv("employees.csv")
- # storing boolean series in new
- new = data["Gender"] != "Female"
- # inserting new series in data frame
- data["New"]= new
- # display
- data[new]
- # OR
- # data[data["Gender"]=="Male"]
- # Both are the same

As shown in the output image, Data frame having Gender=”Male” is returned.

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team	New
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing	True
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN	True
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance	True
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services	True
5	Dennis	Male	4/18/1987	1:35 AM	115163	10.125	False	Legal	True
12	Brandon	Male	12/1/1980	1:08 AM	112807	17.492	True	Human Resources	True
13	Gary	Male	1/27/2008	11:40 PM	109831	5.831	False	Sales	True
16	Jeremy	Male	9/21/2010	5:56 AM	90370	7.369	False	Human Resources	True
17	Shawn	Male	12/7/1986	7:45 PM	111737	6.414	False	Product	True
21	Matthew	Male	9/5/1995	2:12 AM	100612	13.645	False	Marketing	True
23	NaN	Male	6/14/2012	4:19 PM	125792	5.042	NaN	NaN	True
24	John	Male	7/1/1992	10:08 PM	97950	13.873	False	Client Services	True

Pandas Index is an immutable ndarray implementing an ordered, sliceable set. It is the basic object which stores the axis labels for all pandas objects. Pandas Index.data attribute return the data pointer of the underlying data of the given Index object.

Use Index.data attribute to find the memory address of the underlying data of the given Index object. Here is the code example:

- # importing pandas as pd
- import pandas as pd
- # Creating the index
- idx = pd.Index(['Jan', 'Feb', 'Mar', 'Apr', 'May'])
- # Print the index

- print(idx)

Now we will use Index.data attribute to find the memory address of the underlying data of the given series object.

- # return memory address
- result = idx.data
- # Print the result
- print(result)

Pandas dataframe.pow() function calculates the exponential power of dataframe and other, element-wise. This function is essentially same as the dataframe ** other but with a support to fill the missing values in one of the input data.

Use pow() function to find the power of each element in the dataframe. Raise each element in a row to a different power using a series. For example:

- # importing pandas as pd
- import pandas as pd
- # Creating the dataframe
- df1 = pd.DataFrame({"A":[14, 4, 5, 4, 1],

"B":[5, 2, 54, 3, 2],

"C":[20, 20, 7, 3, 8],

"D":[14, 3, 6, 2, 6]})

- # Print the dataframe
- Df

	A	B	C	D
0	14	5	20	14
1	4	2	20	3
2	5	54	7	6
3	4	3	3	2
4	1	2	8	6

Python Series.mod() is used to return the remainder after division of two numbers. In the following examples, the data frame used contains data of some NBA players. The image of data frame before any operations is showed below.

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
5	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	NaN	12000000.0
6	Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
7	Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
8	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
9	Marcus Smart	Boston Celtics	36.0	PG	22.0	6-4	220.0	Oklahoma State	3431040.0
10	Jared Sullinger	Boston Celtics	7.0	C	24.0	6-9	260.0	Ohio State	2569260.0
11	Isaiah Thomas	Boston Celtics	4.0	PG	27.0	5-9	185.0	Washington	6912869.0
12	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0
13	James Young	Boston Celtics	13.0	SG	20.0	6-6	215.0	Kentucky	1749840.0

5 rows of data frame are extracted using head() method. A series is created from a Python list using Pandas Series() method. The mod() method is called on new short data frame and the created list is passed as other parameter. The code example:

- # importing pandas module
- import pandas as pd
- # reading csv file from url

- data = pd.read_csv("https://media.geeksforgeeks.org/wp-content/uploads/nba.csv")
- # creating short data of 5 rows
- short_data = data.head()
- # creating list with 5 values
- list =[1, 2, 3, 4, 3]
- # finding remainder
- # creating new column
- short_data["Remainder"]= short_data["Salary"].mod(list)
- # display
- short_data

Remainder after division of values at same index in caller series and other series was returned. Since nothing was passed to fill_value parameter, the Null values are returned as it is. Here is the picture for example:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary	Added values
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0	0.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0	1.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0	0.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0	2.0

pandas.map() is used to map values from two series having one column same. For mapping two series, the last column of the first series should be same as index column of the second series, also the values should be unique.

In the following example, two series are made from same data. pokemon_names column and pokemon_types index column are same and hence Pandas.map() matches the rest of two columns and returns a new series.

- import pandas as pd
- #reading csv files
- pokemon_names = pd.read_csv("pokemon.csv", usecols = ["Pokemon"], squeeze = True)
- #usecol is used to use selected columns
- #index_col is used to make passed column as index

- pokemon_types = pd.read_csv("pokemon.csv", index_col = "Pokemon", squeeze = True)
- #using pandas map function
- new=pokemon_names.map(pokemon_types)
- print (new)

```
0        Grass
1        Grass
2        Grass
3         Fire
4         Fire
5         Fire
6        Water
7        Water
8        Water
9          Bug
10         Bug
11         Bug
12         Bug
13         Bug
14         Bug
15      Normal
16      Normal
17      Normal
```

This function works only with Series. Passing a data frame would give an Attribute error. Passing series with different length will give the output series of length same as the caller.

```
In [7]: data2.map(data)
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-7-d750a707f3b6> in <module>()
----> 1 data2.map(data)

c:\python36-32\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   4370             if self._info_axis._can_hold_identifiers_and_holds_name(name):
   4371                 return self[name]
-> 4372             return object.__getattribute__(self, name)
   4373
   4374     def __setattr__(self, name, value):

AttributeError: 'DataFrame' object has no attribute 'map'
```

Data Visualization

Scroll through the Python Package Index and you will find libraries for practically every data visualization need, from GazeParser for eye movement research to pastalog for realtime visualizations of neural network training. And while many of these libraries are intensely focused on accomplishing a specific task, some can be used no matter what your field.

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has an excellent library for you.

Matplotlib

matplotlib is the O.G. of Python data visualization libraries. Despite being over a decade old, it's still the most widely used library for plotting in the Python community. It was designed to closely resemble MATLAB, a proprietary programming language developed in the 1980s.

Because matplotlib was the first Python data visualization library, many other libraries are built on top of it or designed to work in tandem with it during analysis. Some libraries like pandas and Seaborn are “wrappers” over matplotlib. They allow you to access a number of matplotlib’s methods with less code.

While matplotlib is good for getting a sense of the data, it's not very useful for creating publication-quality charts quickly and easily. As Chris Moffitt points out in his overview of Python visualization tools, matplotlib “is extremely powerful but with that power comes complexity.” matplotlib has long been criticized for its default styles, which have a distinct 1990s feel. The upcoming release of matplotlib 2.0 promises many new style changes to address this problem.

Seaborn

Seaborn harnesses the power of matplotlib to create beautiful charts in a few lines of code. The key difference is Seaborn's default styles and color palettes, which are designed to be more aesthetically pleasing and modern. Since Seaborn is built on top of matplotlib, you'll need to know matplotlib to tweak Seaborn's defaults.

Ggplot

ggplot is based on ggplot2, an R plotting system, and concepts from The Grammar of Graphics. ggplot operates differently than matplotlib. It lets you layer components to create a complete plot. For instance, you can start with axes, then add points, then a line, a trendline, etc. Although The grammar of graphics has been praised as an “intuitive” method for plotting, seasoned matplotlib users might need time to adjust to this new mindset.

According to the creator, ggplot is not designed for creating highly customized graphics. It sacrifices complexity for a simpler method of plotting.

ggplot is tightly integrated with pandas, so it is best to store your data in a DataFrame when using ggplot.

Like ggplot, Bokeh is based on The Grammar of Graphics, but unlike ggplot, it's native to Python, not ported over from R. Its strength lies in the ability to create interactive, web-ready plots, which can be easily output as JSON objects, HTML documents, or interactive web applications. Bokeh also supports streaming and real-time data.

Bokeh

Bokeh provides three interfaces with varying levels of control to accommodate different user types. The highest level is for creating charts quickly. It includes methods for creating common charts such as bar plots, box plots, and histograms. The middle level has the same specificity as matplotlib and allows you to control the basic building blocks of each chart. The lowest level is geared toward developers and software engineers. It has no pre-set defaults and requires you to define every element of the chart.

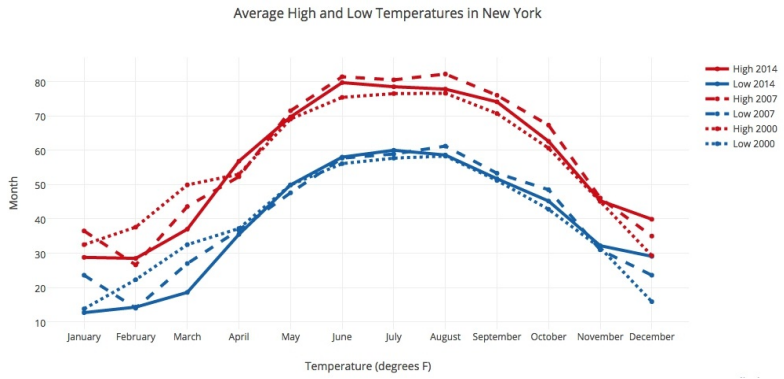
Pygal

Like Bokeh and Plotly, pygal offers interactive plots that can be embedded in the web browser. Its prime differentiator is the ability to output charts as SVGs. As long as you're working with smaller datasets, SVGs will do you just fine. But if you're making charts with hundreds of thousands of data points, they'll have trouble rendering and become sluggish. Since each chart type is packaged into a method and the built-in styles are pretty, it is easy to create a nice-looking chart in a few lines of code.

Plotly

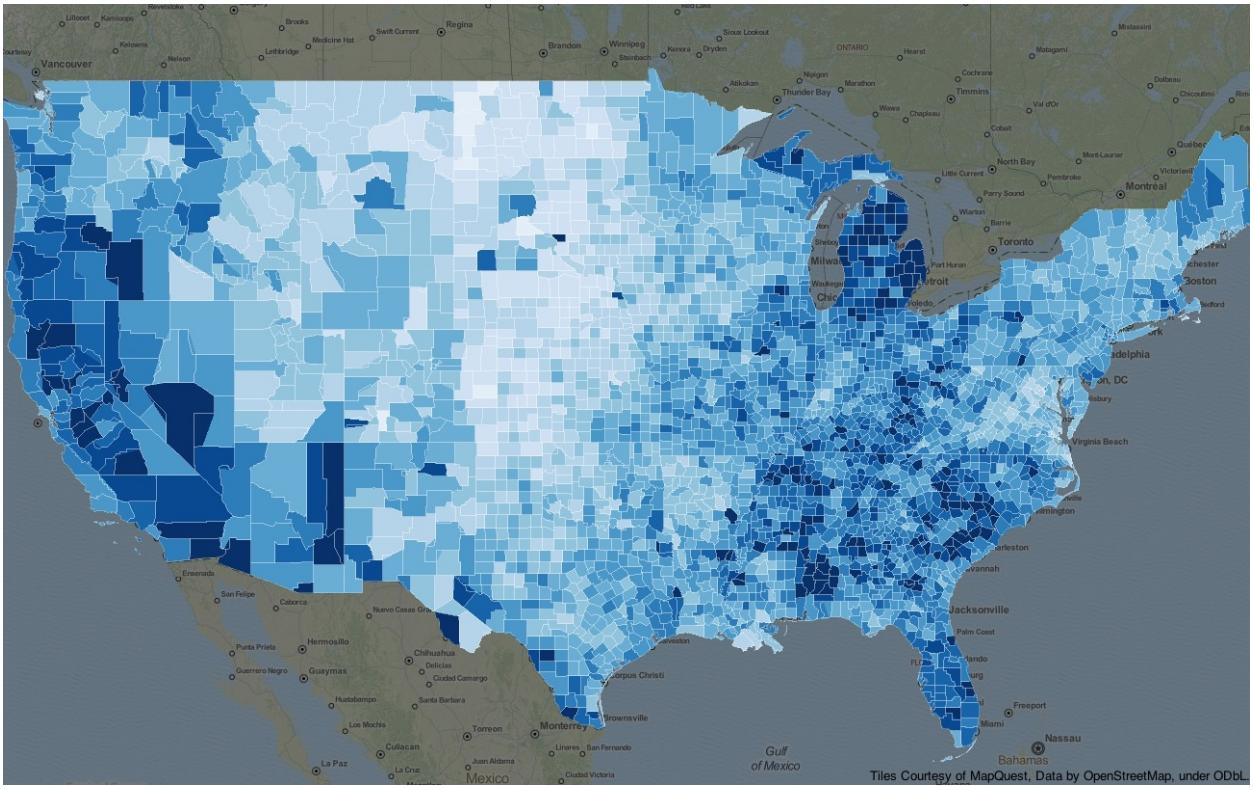
You might know Plotly as an online platform for data visualization, but did you also know you can access its capabilities from a Python notebook? Like Bokeh,

Plotly's forte is making interactive plots, but it offers some charts you won't find in most libraries, like contour plots, dendograms, and 3D charts.



Geoplotlib

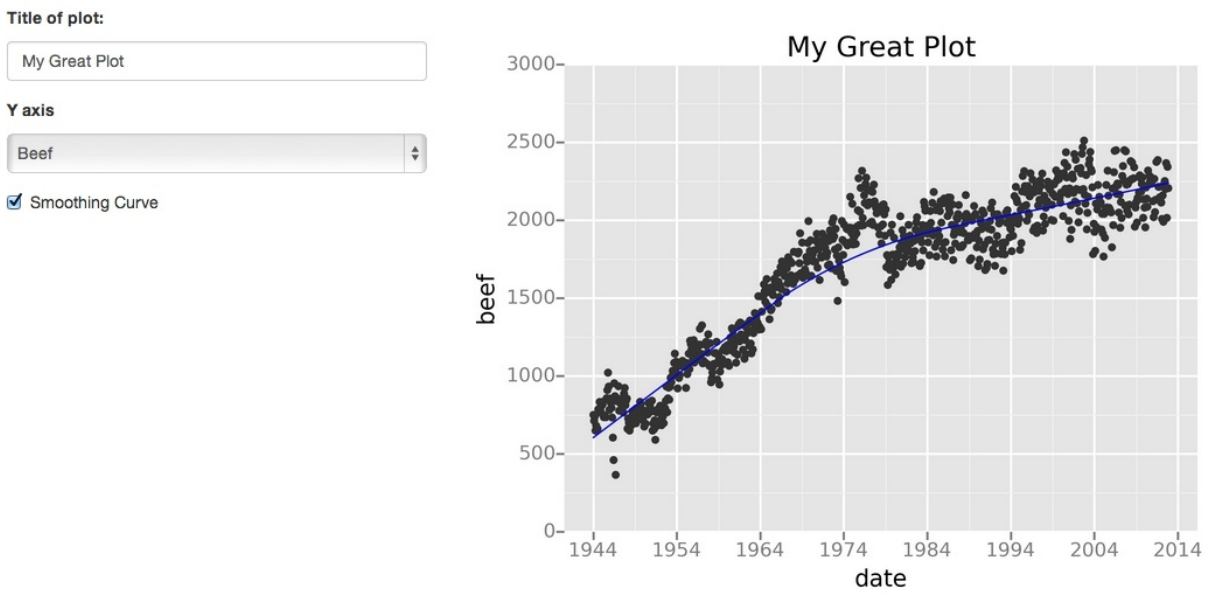
geoplotlib is a toolbox for creating maps and plotting geographical data. You can use it to create a variety of map-types, like choropleths, heatmaps, and dot density maps. You must have Pyglet, an object-oriented programming interface installed to use geoplotlib. Nonetheless, since most Python data visualization libraries don't offer maps, it is nice to have a library dedicated solely to them.



Gleam

Gleam is inspired by R's Shiny package. It allows you to turn analyses into

interactive web apps using only Python scripts, so you don't have to know any other languages like HTML, CSS, or JavaScript. Gleam works with any Python data visualization library. Once you've created a plot, you can build fields on top of it so users can filter and sort data.

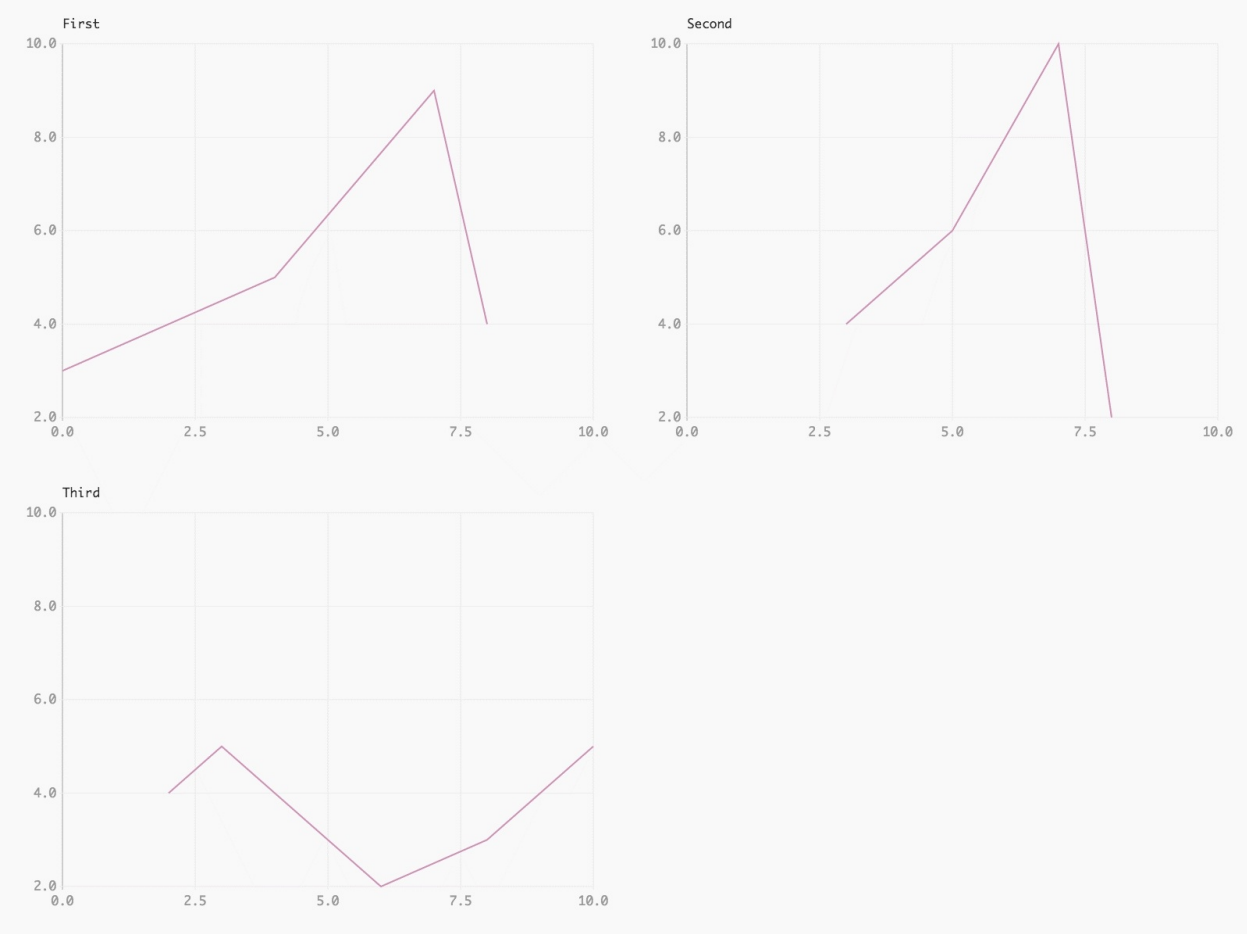


Missingano

Dealing with missing data is a pain. missingno allows you to quickly gauge the completeness of a dataset with a visual summary, instead of trudging through a table. You can filter and sort data based on completion or spot correlations with a heatmap or a dendrogram.

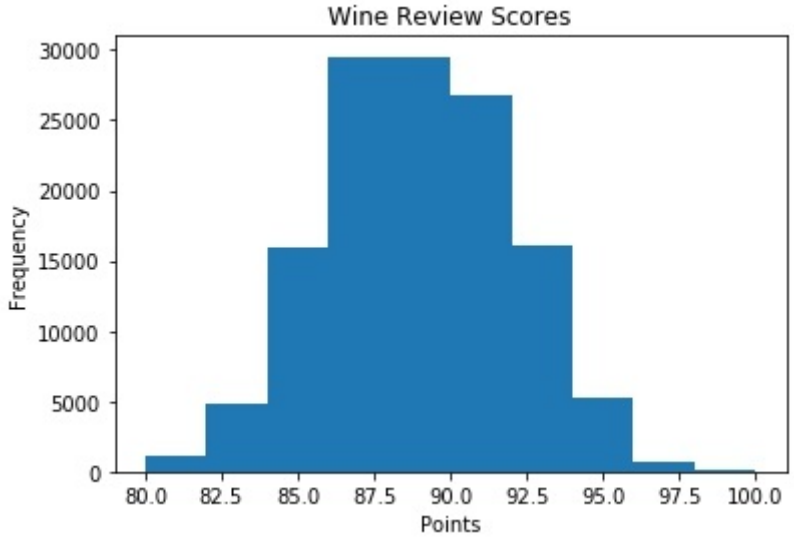
Leather

Leather's creator, Christopher Groskopf, puts it best: “Leather is the Python charting library for those who need charts now and don’t care if they are perfect.” It is designed to work with all data types and produces charts as SVGs, so you can scale them without losing image quality. Since this library is relatively new, some of the documentation is still in progress. The charts you can make are pretty basic, but that is the intention.



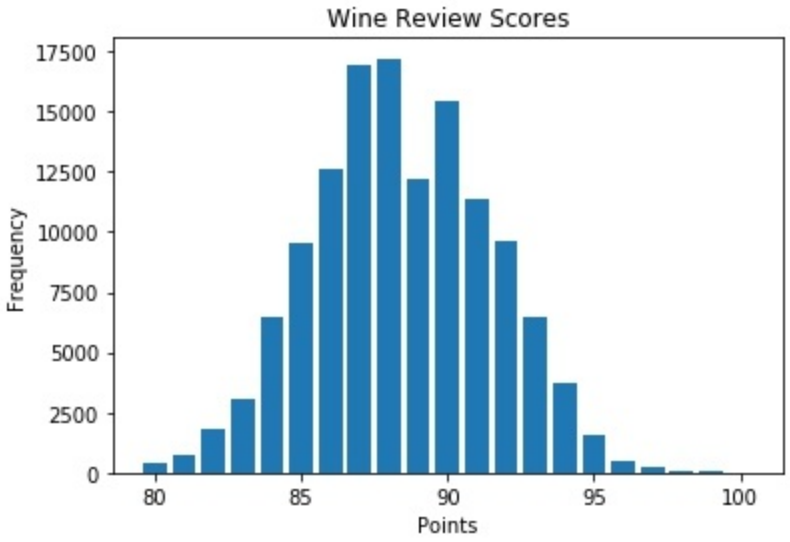
Histogram

In Matplotlib we can create a Histogram using the hist method. If we pass it categorical data like the points column from the wine-review dataset it will automatically calculate how often each class occurs.



Bar Chart

A bar chart can be created using the bar method. The bar-chart isn't automatically calculating the frequency of a category so we are going to use pandas value_counts function to do this. The bar-chart is useful for categorical data that doesn't have a lot of different categories (less than 30) because else it can get quite messy.



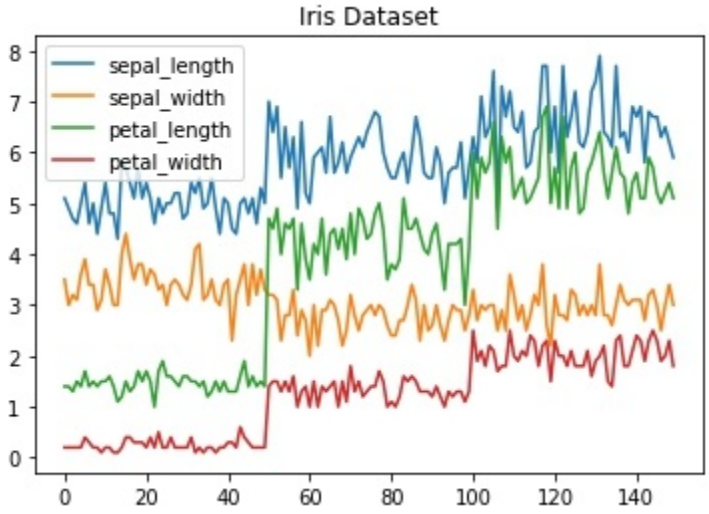
Pandas Visualization

Pandas is an open source high-performance, easy-to-use library providing data structures, such as dataframes, and data analysis tools like the visualization tools we will use in this article.

Pandas Visualization makes it really easy to create plots out of a pandas dataframe and series. It also has a higher level API than Matplotlib and therefore we need less code for the same results.

Line Chart

To create a line-chart in Pandas we can call <dataframe>.plot.line(). Whilst in Matplotlib we needed to loop-through each column we wanted to plot, in Pandas we don't need to do this because it automatically plots all available numeric columns.


Iris Dataset
sepal_length
sepal_width
petal_length
petal_width
0
1
2
3
4
5
6
7
8
0
20
40
60
80
100
120
140

Data Mining

Data mining is the process of discovering predictive information from the analysis of large databases. For a data scientist, data mining can be a vague and daunting task and it requires a diverse set of skills and knowledge of many data mining techniques to take raw data and successfully get insights from it. You will want to understand the foundations of statistics and different programming languages that can help you with data mining at scale.

The desired outcome from data mining is to create a model from a given data set that can have its insights generalized to similar data sets. A real-world example of a successful data mining application can be seen in automatic fraud detection from banks and credit institutions.

Your bank likely has a policy to alert you if they detect any suspicious activity on your account, such as repeated ATM withdrawals or large purchases in a state outside of your registered residence. How does this relate to data mining? Data scientists created this system by applying algorithms to classify and predict whether a transaction is fraudulent by comparing it against a historical pattern of fraudulent and non-fraudulent charges.

That is just one of a number of the powerful applications of data mining. Other applications of data mining include genomic sequencing, social network analysis or crime imaging, but the most common use case is for analyzing aspects of the consumer life cycle. Companies use data mining to discover consumer preferences, classify different consumers based on their purchasing activity and determine what makes for a well-paying customer and also information that can have profound effects on improving revenue streams and cutting costs.

There are multiple ways to build predictive models from data sets and a data scientist should understand the concepts behind these techniques, as well as how to use code to produce similar models and visualizations. These techniques include:

- Regression
- Classification
- Cluster Analysis
- Association and Correlation Analysis
- Outlier analysis

Regression represents the estimating of the relationships between variables by

optimizing the reduction of error.

Classification represents the identifying what category an object belongs to. An example is classifying email as spam or legitimate or looking at a person's credit score and approving or denying a loan request.

Cluster Analysis is finding natural groupings of data objects based upon the known characteristics of that data. An example could be seen in marketing, where analysis can reveal customer groupings with unique behavior which could be applied in business strategy decisions.

Association and Correlation Analysis represent the looking to see if there are unique relationships between variables that are not immediately obvious. An example would be the famous case of beer and diapers. Men who bought diapers at the end of the week were much more likely to buy beer, so stores placed them close to each other to increase sales.

Outlier analysis is examining outliers to examine potential causes and reasons for said outliers. An example of which is the use of outlier analysis in fraud detection and trying to determine if a pattern of behavior outside the norm is fraud or not.

Data mining for business is often performed with a transactional and live database that allows easy use of data mining tools for analysis. One example of which would be an Online Analytical Processing server or OLAP, which allows users to produce multi-dimensional analysis within the data server. OLAPs allow for business to query and analyze data without having to download static data files, which is helpful in situations where your database is growing on a daily basis. However, for someone looking to learn data mining and practicing on their own, an iPython notebook will be perfectly suited to handle most data mining tasks.

Let's walk through how to use Python to perform data mining using two of the data mining algorithms described above regression and clustering.

First things first, if you want to follow along, install Jupyter on your desktop. It is a free platform that provides what is essentially a processer for iPython notebooks that is extremely intuitive to use. Follow these instructions for installation. Everything I do here will be completed in a Python file in Jupyter.

We will be using the Pandas module of Python to clean and restructure our data. Pandas is an open-source module for working with data structures and analysis, one that is ubiquitous for data scientists who use Python. It allows for data scientists to upload data in any format, and provides a simple platform organize, sort, and manipulate that data.

Numpy is a necessary package for scientific computation. It includes an incredibly versatile structure for working with arrays, which are the primary data format that scikit-learn uses for input data.

Matplotlib represent the fundamental package for data visualization in Python. This module allows for the creation of everything from simple scatter plots to 3-dimensional contour plots. Note that from matplotlib we install pyplot, which is the highest order state-machine environment in the modules hierarchy. If that is meaningless to you don't worry about it, just make sure you get it imported to your notebook.

Scipy is a collection of tools for statistics in Python. Stats are the scipy module that imports regression analysis functions.

Having the regression summary output is important for checking the accuracy of the regression model and data to be used for estimation and prediction, but visualizing the regression is an important step to take to communicate the results of the regression in a more digestible format.

Visualizing linear relationships using Seaborn represent the documentation that gives specific examples and show how to modify you regression plots and display new features that you might not know how to code yourself. It also teaches you how to fit different kinds of models, such as quadratic or logistic models.

Classification is one of the largest uses of data mining, both in practical use and in research. As before, we have a set of samples that represents objects or things we are interested in classifying. We also have a new array, the class values. These class values give us a categorization of the samples. There are some examples:

- Determining the species of a plant by looking at its measurements. The class value here would be which species is this?.
- Determining if an image contains a dog. The class would be Is there a dog in this image?.
- Determining if a patient has cancer based on the test results. The class would be does this patient have cancer?.

While many of the examples above are binary (yes/no) questions, they do not have to be, as in the case of plant species classification in this section.The goal of classification applications is to train a model on a set of samples with known classes, and then apply that model to new unseen samples with unknown classes. For example, you want to train a spam classifier on your past e-mails, which you

have labeled as spam or not spam. You want to use that classifier to determine whether your next email is spam, without needing to classify it yourself. Data mining encompasses a number of predictive modeling techniques and you can use a variety of data mining software. To learn to apply these techniques using Python is difficult and it will take practice and diligence to apply these on your own data set. Early on you will run into innumerable bugs, error messages and roadblocks. Stay persistent and diligent in your data mining attempts.

OneR is a simple algorithm that simply predicts the class of a sample by finding the most frequent class for the feature values. OneR is a short hand for One Rule, indicating we only use a single rule for this classification by choosing the feature with the best performance. While some of the later algorithms are significantly more complex, this simple algorithm has been shown to have good performance in a number of real-world datasets.

The algorithm starts by iterating over every value of every feature. For that value, count the number of samples from each class that have that feature value. Record the most frequent class for the feature value, and the error of that prediction. For example, if a feature has two values, 0 and 1, we first check all samples that have the value 0. For that value, we may have 20 in class A, 60 in class B, and a further 20 in class C. The most frequent class for this value is B, and there are 40 instances that have difference classes.

The prediction for this feature value is B with an error of 40, as there are 40 samples that have a different class from the prediction. We then do the same procedure for the value 1 for this feature, and then for all other feature value combinations.

Overfitting is the problem of creating a model that classifies our training dataset very well, but performs poorly on new samples. The solution is quite simple. Never use training data to test your algorithm.

Classification with Scikit-learn

Scikit-learn is an easy-to-use, general-purpose toolbox for machine learning in Python. Scikit-learn is a library that provides a variety of both supervised and unsupervised machine learning techniques. Supervised machine learning refers to the problem of inferring a function from labeled training data and it comprises both regression and classification. Unsupervised machine learning, on the other hand, refers to the problem of finding interesting patterns or structure in the data and it comprises techniques such as clustering and dimensionality reduction. In addition to statistical learning techniques, scikit-learn provides utilities for common tasks such as model selection, feature extraction, and feature selection.

Scikit-Learn package is python's core machine learning package that has most of the necessary modules to support a basic machine learning project. The library provides a unified API or in short Application Programming Interface for practitioners to ease the use of machine learning algorithms with only writing a few lines to accomplish the predictive or classification task. One of the few libraries in python which has kept to the promise of maintaining the algorithm and interface layer simple and not complicating it to cover the entire machine learning feature landscape. The package is written heavily in python, and it incorporates C++ libraries like LibSVM and LibLinear for support vector machines and generalized linear model implementation. The package depends on Pandas, numpy and scipy.

The scikit-learn is useful mainly because of its project vision. Code quality and proper documentation form the core vision. Robust implementation takes priority over as many feature inclusion as possible for a given algorithm and also the implementation is strongly backed by unit tests.

Scikit-learn provides an object-oriented interface centered around the concept of an Estimator. It may be a classification, regression or clustering algorithm or a transformer that extracts or filters useful features from raw data. The example code:

- class Estimator(object):
- def fit(self, X, y=None):
- """Fits estimator to data. """
- # set state of ``self``
- return self
- def predict(self, X):

- """Predict response of ``X``. """
- # compute predictions ``pred``
- return pred

During the fitting process, the state of the estimator is stored in instance attributes that have a trailing underscore ('_'). For example, the coefficients of a LinearRegression estimator are stored in the attribute.

Estimators that can generate predictions provide an Estimator.predict method. In the case of regression, Estimator.predict will return the predicted regression values. It will return the corresponding class labels in the case of classification. Classifiers that can predict the probability of class membership have a method Estimator.predict_place that returns a two-dimensional numpy array of shape where the classes are lexicographically ordered.

Finally, there is a special type of Estimator called Transformer which transforms the input data, for example selects a subset of the features or extracts new features based on the original ones.

Usually, a Transformer does not provide a predict method, but in some cases it may. One transformer that we will use in this posting is sklearn.preprocessing.StandardScaler. Although regression and classification appear to be very different they are in fact similar problems.

In regression our predictions for the response are real-valued numbers. On the other hand, in classification the response is a mutually exclusive class label such as "Is the email spam/ham?" or "Is the credit card transaction fraudulent?" If the number of classes is equal to two, then we call it a binary classification problem; if there are more than two classes, then we call it a multiclass classification problem. In the following we will assume binary classification because it's the more general case, and we can always represent a multiclass problem as a sequence of binary classification problems.

We can also think of classification as a function estimation problem where the function that we want to estimate separates the two classes.

Different classification techniques can often be compared using the type of decision surface they can learn. The decision surfaces describe for what values of the predictors the model changes its predictions and it can take several different shapes: piece-wise constant, linear, quadratic, vornoi tessellation etc…

This section will introduce three popular classification techniques: Logistic Regression, Discriminant Analysis, and Nearest Neighbor. We will investigate

what their strengths and weaknesses are by looking at the decision boundaries they can model.

Logistic regression can be viewed as an extension of linear regression to classification problems. One of the limitations of linear regression is that it cannot provide class probability estimates. This is often useful, for example, when we want to inspect manually the most fraudulent cases.

Linear discriminant Analysis or in short LDA is another popular technique which shares some similarities with LogisticRegression. LDA too finds linear boundary between the two classes where points on side are classified as one class and those on the other as classified as the other class.

The major difference between LDA and LogisticRegression is the way each picks the linear decision boundary: Linear Discriminant Analysis models the decision boundary by making distributional assumptions about the data generating process while Logistic Regression models the probability of a sample being member of a class given its feature values.

The Nearest Neighbor differs fundamentally from the above models in that it is a so-called non-parametric technique. The number of parameters of the model can grow infinitely as the size of the training data grows. Furthermore, it can model non-linear decision boundaries, something that is important for the first two datasets, moons and circles.

In Supervised Learning, we have a dataset consisting of both features and labels. The task is to construct an estimator which is able to predict the label of an object given the set of features. Some more complicated examples are:

- given a multicolor image of an object through a telescope, determine whether that object is a star, a quasar, or a galaxy.
- given a photograph of a person, identify the person in the photo.
- given a list of movies a person has watched and their personal rating of the movie, recommend a list of movies they would like (So-called recommender systems: a famous example is the Netflix Prize).

Supervised learning is further broken down into two categories, classification and regression. In classification, the label is discrete, while in regression, the label is continuous. For example, in astronomy, the task of determining whether an object is a star, a galaxy or a quasar is a classification problem. The label is from three distinct categories. On the other hand, we might wish to estimate the age of an object based on such observations. This would be a regression problem, because the label is a continuous quantity.

Most of the machine learning algorithms will require more labeled observations compared to the toy datasets and the package has inbuild sample generator routines to generate a labeled dataset with the required number of observations. The sample generator make_moons take two key parameters n_samples and noise. the practitioner can supply the routine with a number of samples to generate and the noise added to the input features.

After loading the dataset it has to split into training and testing set to start with the algorithm training. The package has a routine to break a pandas dataframe or numpy array into train and test set. The method takes input features, target array, the size of the test set and stratification array. Stratification is a handy option as the target class proportions are represented the same across training and test set. The target distribution will be the same across training and test dataset

It is mandatory to scale all continuous numeric input features so that not a single feature influences the model performance and input feature. A might range in millions and B in hundreds, and if not scaled to a standard scale the model will not learn the variance in feature B. The package comes with minmax scaled between 0 and 1 and standard scaler where the scale output will include negative values.

The model parameter tuning is a daunting task, and multiple iterations have to be logged in with their performance metrics until one reaches the best set of parameters. Parameter tuning is mostly simplified in Scikit-learn by the GridSearchCV routine. Given a list of model parameter combinations, the method runs all possible combinations and returns the best model parameter along with the best estimator. The method also performs cross-validation, so the best estimator does not overfit the training data.

Practitioners can code their custom estimators. The custom estimator can be part of a pipeline. A pipeline takes multiple estimators and executes them in sequence. It will pass the output of the previous estimator as the input to the next estimator in the list. An entire model process can be designed using the pipeline, and it can be directly fit to a dataset. This routine is a great help in simplifying the model production deployment.

Any machine learning model will require numeric input features continuous or categorical and text features does not integrate well. Scikit-learn has a pill for that too. Use label encoder or one-hot encoder.

IPython and Jupyter

1. iPython

IPython or Interactive Python is a command shell for interactive computing in multiple programming languages, originally developed for the Python programming language, that offers introspection, rich media, shell syntax, tab completion, and history.

IPython provides some features, for example:

- Interactive shells
- A browser-based notebook interface with support for code, text, mathematical expressions, inline plots and other media.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into one's own projects.
- Tools for parallel computing.

IPython is based on an architecture that provides parallel and distributed computing. IPython enables parallel applications to be developed, executed, debugged and monitored interactively. This architecture abstracts out parallelism, enabling IPython to support many different styles of parallelism including:

- Single program, multiple data (SPMD) parallelism
- Multiple program, multiple data (MIMD) parallelism
- Message passing using MPI
- Task parallelism
- Data parallelism
- Combinations of these approaches
- Custom user defined approaches

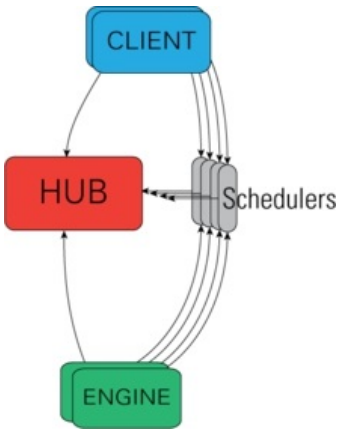
With the release of IPython 4.0, the parallel computing capabilities were made optional and released under the ipyparallel python package.

IPython frequently draws from SciPy stack libraries like NumPy and SciPy, often installed alongside one of many Scientific Python distributions. IPython provides integration with some libraries of the SciPy stack, notably matplotlib, producing inline graphs when in used with the Jupyter notebook. Python

libraries can implement IPython specific hooks to customize rich object display. SymPy for example implements rendering of mathematical expressions as rendered LaTeX when used within IPython context.

IPython allows non-blocking interaction with Tkinter, PyGTK, PySide and wxPython. IPython can interactively manage parallel computing clusters using asynchronous status callbacks and MPI. IPython can also be used as a system shell replacement. Its default behavior is largely similar to Unix shells, but it allows customization and the flexibility of executing code in a live Python environment. Using IPython as a shell replacement is less common and it is now recommended to use Xonsh which provide most of the IPython feature with better shell integrations.

In 2014, Fernando Pérez announced a spin-off project from IPython called Project Jupyter. IPython continued to exist as a Python shell and kernel for Jupyter, but the notebook interface and other language-agnostic parts of IPython were moved under the Jupyter name. Jupyter is language agnostic and its name is a reference to core programming languages supported by Jupyter, which are Julia, Python, and R.



2. Jupyter

Jupyter Notebook, formerly IPython Notebooks, is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input or output cells which can contain code, text, mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output

formats through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell.

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

- IPython
- OMQ
- Tornado
- jQuery
- Bootstrap
- MathJax

A Jupyter kernel is a program responsible for handling various types of request, like code execution, code completions, inspection and providing a reply. Kernels talk to the other components of Jupyter using ZeroMQ over the network, and thus can be on the same or remote machines.

Unlike many other Notebook-like interfaces in Jupyter, kernels are not aware that they are attached to a specific document, and can be connected to many clients at once. Usually kernels allow execution of only a single language, but there are a couple of exceptions.

JupyterHub is a multi-user server for Jupyter Notebooks. It is designed to support many users by spawning, managing, and proxying many singular Jupyter Notebook servers. While JupyterHub requires managing servers, third-party services like Jupyo provide an alternative to JupyterHub by hosting and managing multi-user Jupyter notebooks in the cloud.

JupyterLab is the next-generation user interface for Project Jupyter. It offers all the familiar building blocks of the classic Jupyter Notebook in a flexible and powerful user interface. The first stable release was announced on February 20, 2018.

For learners as well as for more advanced data scientists, the Jupyter Notebook is one of the popular data science tools out there. The interactive environment is not only ideal to teach and learn with, and to share your work with peers, but also ensures reproducible research. Yet, as you are discovering how to work with

this notebook, you will often bump into IPython.

Many computational notebooks were created after the Maple and Mathematics notebooks. This section, however, will focus on the notebooks that have contributed to the rise of data science notebooks.

The Sage notebook as a browser-based system was first released mid-2000s and then in 2007, a new version was released that was more powerful, that had user accounts and could be used to make documents public. It resembled the Google Docs UI design since the layout of the Sage notebook was based on the layout of Google notebooks.

In late 2001, roughly twenty years after Guido van Rossum began to work on Python at the National Research Institute for Mathematics and Computer Science in the Netherlands, Fernando Pérez started developing IPython. The project was heavily influenced by the Mathematica notebooks and Maple worksheets, just like the Sage notebook and many other projects that followed.

The name “Jupyter” was inspired by thinking about the leading open languages for science and even though it might seem like any other acronym, it never meant that other languages were not welcome.

R Markdown and the Jupyter Notebook share the delivering of a reproducible workflow, the weaving of code, output, and text together in a single document, supporting interactive widgets and outputting to multiple formats.

However, the two also differ, like the former focuses on reproducible batch execution, plain text representation, version control, production output and offers the same editor and tools that you use for R scripts. Notebooks have an emphasis on an interactive execution model. They don’t use a plain text representation, but a structured data representation, such as JSON.

Of course, there are even more notebooks that you can consider when you are getting into data science. In recent years, a lot of new alternatives have found their way to data scientists and data science enthusiasts. Think not only of Beaker Notebook, Apache Zeppelin, Spark Notebook, DataBricks Cloud, etc., but also of other tools such as the Rodeo IDE or nteract which also make your data science analyses interactive and reproducible. A good thing to note here is that nteract differs from the other tools that are mentioned here because it makes use of the Jupyter architecture, such as the protocols and the formats.

And notebooks seem to be here to stay. Recently, the next generation of the Jupyter Notebook has been introduced to the community: JupyterLab. The Notebook application includes not only support for Notebooks but also a file manager, a text editor, a terminal emulator, a monitor for running Jupyter

processes, an IPython cluster manager and a pager to display help.

You might think now that this is nothing new. The Jupyter Notebook also has all these things. JupyterLab, however, enables you to make use of all these building blocks of interactive computing in novel ways.

It is possible to adapt IPython for system shell usage with the shell escape. Lines that start with '!' are passed directly to the system shell. For example, ' !ls ' will run ls in the current directory. You can assign the result of a system command to a Python variable with the syntax myfiles=!ls. However, if you want to get the result of an ls function explicitly printed out as a list with strings, without assigning it to a variable, use two exclamation marks (!!ls) or the %sx magic command without an assignment.

Note that !! commands cannot be assigned to a variable, but that the result of a magic (as long as it returns a value) can be assigned to a variable.

IPython also allows you to expand the value of Python variables when making system calls: just wrap your variables or expressions in braces ({}). Also, in a shell command with ! or !!, any Python variable prefixed with $ is expanded. In the code chunk below, you'll see that you echo the argv attribute of the sys variable. Note that you can also use the $/$$ syntaxes to Python variables from system output, which you can later use for further scripting.

To pass a literal $ to the shell, use a double $$. You'll need this literal $ if you want to access the shell and environment variables like $PATH.

Note that besides IPython, there are also other kernels that have special syntax to make sure that lines are executed as shell commands. These aren't really "magics" in the strictest sense of the word because they may implement it with any name they want and it can be completely different than the IPython magics.

The IPython kernel uses, as you might already know, the % syntax element because it's not a valid unary operator in Python. However, lines that begin with %% signal a cell magic: they take as arguments not only the rest of the current line, but all lines below them as well, in the current execution block. Cell magics can in fact make arbitrary modifications to the input they receive, which need not even be valid Python code at all. They receive the whole block as a single string.

Magics are specific to and provided by kernels and are designed to make your work and experience within Jupyter Notebook a lot more interactive. Whether magic commands are available in a certain kernel depends on the kernel developer and on kernel per kernel.

One major feature of the IPython kernel is the ability to display plots that are the output of running code cells. The kernel is designed to work seamlessly with the matplotlib data visualization library to provide this functionality. To make use of it, use the magic command %matplotlib.

As such, your plot will be displayed in a separate window by default. Additionally, you can also specify a backend, such as inline or qt, the output of the plotting commands will be shown inline or through a different GUI backend.

Next, you can also use magics to call up a Python debugger %pdb every time there is an uncaught exception. This will direct you through the part of the code that triggered the exception, which will make it possible rapidly find the source of a bug.

You can also use the %run magic command with the 'd' option to run scripts under the Python debugger's control. It will automatically set up initial breakpoints for you. Lastly, you can also use the %debug magic for even easier debugger access.

You can use the %load_ext magic to load an IPython extension by its module name. IPython extensions are Python modules that modify the behaviour of the shell. Extensions can register magics, define variables, and generally modify the user namespace to provide new features for use within code cells. Here are some examples:

- Use %load_ext oct2py.ipython to seamlessly call M-files and Octave functions from Python,
- Use %load_ext rpy2.ipython to use an interface to R running embedded in a Python process,
- Use %load_ext Cython to use a Python to C compiler,
- Use sympy.init_printing() to pretty print Sympy Basic objects automatically

If you are working with another kernel and you wonder if you can make use of magic commands, it might be handy to know that there are some kernels that build on the metakernel project and that will use, in most cases, the same magics that you will also find in the IPython kernel. You can find a list of the metakernel magics here. The metakernel is a Jupyter or IPython kernel template which includes core magic functions.

Some examples:

- The MATLAB kernel matlab_kernel,

- The Octave kernel octave_kernel,
- The Java9 kernel java9_kernel,
- The Wolfram kernel wolfram_kernel,
- The SAS kernel sas_kernel. … And many more!

Converting and formatting notebooks are features that you will find in the Jupyter ecosystem. Two tools that you'll typically find for these tasks are nbconvert and nbformat.

You can use the former to convert notebooks to various other formats to present information in familiar formats, to publish research and to embed notebooks in papers, to collaborate with others and to share content with a larger audience.

The latter basically contains the Jupyter Notebook format and is the key to understanding that notebook files are simple JSON documents that contain metadata the version of the notebook format and the cells in which all text, code is stored.

Saving and loading notebooks is a feature of the Jupyter Notebook Application. You can load notebooks, saved as files with an .ipynb filename extension, which other people have created by downloading and opening up the file in the Jupyter application. More specifically, you can create a new notebook and then choose to open the file by clicking on the "File" tab, clicking "Open" and selecting your downloaded notebook.

The parallel computing network was part of the IPython project, but as of 4.0, it is a standalone package called ipyparallel. The package is basically a collection of CLI scripts for controlling clusters for Jupyter.

Even though it is split off, it is still a powerful component of the IPython ecosystem that is generally overlooked. It is so powerful because instead of running a single Python kernel and it allows you to start many distributed kernels over many machines.

Typical use cases for ipyparallel are, for example, cases in which you need to run models many different times to estimate the distributions of its outputs or how they vary with input parameters. When the runs of the model are independent, you can speed up the process by running them in parallel across multiple computers in a cluster.

This feature is one that is part of the Jupyter ecosystem. You have the Jupyter Console and a Jupyter terminal application. However, IPython has been used to indicate the original, interactive command-line terminal for Python. It offers an

enhanced readable print loop environment particularly well adapted to scientific computing. This was the standard before 2011 when the Notebook tool was introduced and started offering a modern and powerful web interface to Python.

Next, you also had the IPython console, which started two processes. The original IPython terminal shell and the default profile or kernel which gets started if not otherwise noted. By default, this was Python. The IPython console is now deprecated and if you want to start it, you will need to use the Jupyter Console, which is a terminal-based console frontend for Jupyter kernels. This code is based on the single-process IPython terminal. The Jupyter Console provides the interactive client-side experience of IPython at the terminal, but with the ability to connect to any Jupyter kernel instead of only to IPython. This lets you test any Jupyter Kernel you may have installed at the terminal, without needing to fire up a full-blown Notebook for it. The Console allows for console-based interaction with other Jupyter kernels such as IJulia, IRKernel.

Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython and others. There is also a procedural "pylab" interface based on a state machine like OpenGL. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and further joined by Thomas Caswell. Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement.

Pyplot is a Matplotlib module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python and the advantage of being free and open-source. Here is the code example:

- import matplotlib.pyplot as plt
- import numpy as np
- a = np.linspace(0, 10, 100)
- b = np.exp(-a)
- plt.plot(a, b)
- plt.show()

However, matplotlib is also a massive library, and getting a plot to look just right is often achieved through trial and error. Using one-liners to generate basic plots in matplotlib is fairly simple, but skillfully commanding the remaining 98% of the library can be daunting.

Learning matplotlib can be a frustrating process at times. The problem is not that matplotlib’s documentation is lacking: the documentation is actually extensive. But the following issues can cause some solution-finding activities:

- The library itself is huge, at something like 70,000 total lines of code.
- Matplotlib is home to several different interfaces and capable of interacting with a handful of different backends.
- While it is comprehensive, some of matplotlib’s own public documentation is seriously out-of-date. The library is still evolving,

and many older examples floating around online may take 70% fewer lines of code in their modern version.

A Figure object is the outermost container for a matplotlib graphic, which can contain multiple Axes objects. One source of confusion is the name… An Axes actually translates into what we think of as an individual plot or graph.

You can think of the Figure object as a box-like container holding one or more Axes. Below the Axes in the hierarchy are smaller objects such as tick marks, individual lines, legends, and text boxes. Almost every "element" of a chart is its own manipulable Python object, all the way down to the ticks and labels. Here is the code example:

- >>> fig, _ = plt.subplots()
- >>> type(fig)
- <class 'matplotlib.figure.Figure'>

Alright, we need one more chunk of theory before we can get around to the shiny visualizations, the difference between the stateful and stateless or object-oriented interfaces.

Almost all functions from pyplot, such as plt.plot(), are implicitly either referring to an existing current Figure and current Axes, or creating them anew if none exist.

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

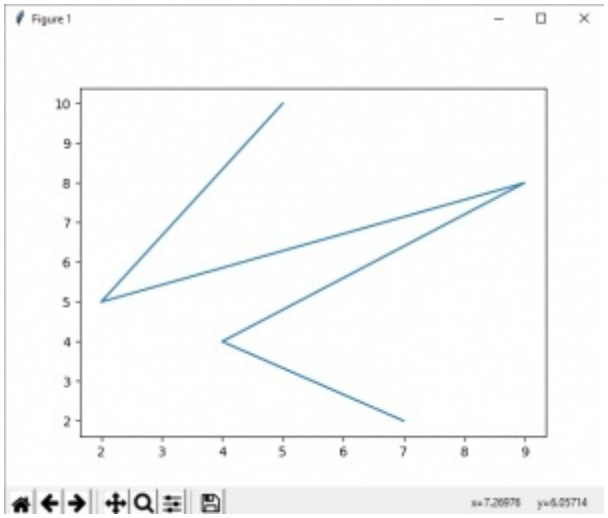
Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information.

Example code:

- # importing matplotlib module
- from matplotlib import pyplot as plt
- # x-axis values
- x = [5, 2, 9, 4, 7]

- # Y-axis values
- y = [10, 5, 8, 4, 2]
- # Function to plot
- plt.plot(x,y)
- # function to show the plot
- plt.show()

The output:

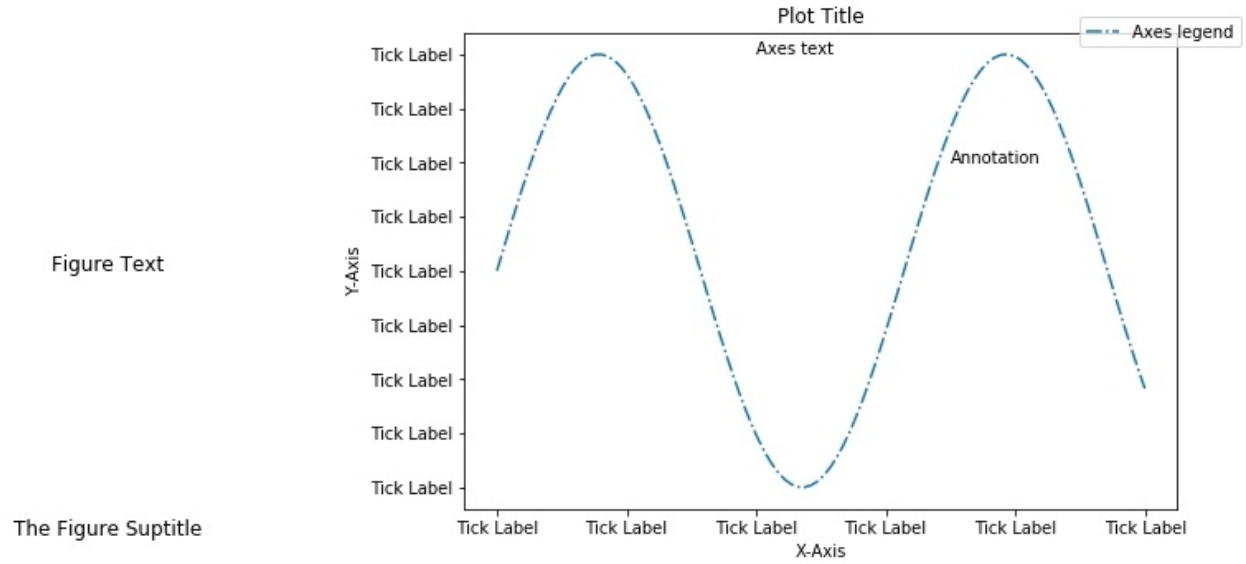


At first sight, it will seem that there are quite some components to consider when you start plotting with this Python data visualization library. You will probably agree with me that it’s confusing and sometimes even discouraging seeing the amount of code that is necessary for some plots, not knowing where to start yourself and which components you should use.

Luckily, this library is very flexible and has a lot of handy, built-in defaults that will help you out tremendously. As such, you don’t need much to get started: you need to make the necessary imports, prepare some data, and you can start plotting with the help of the plot() function.

What you can’t see on the surface is that you have maybe unconsciously made use of the built-in defaults that take care of the creation of the underlying components, such as the Figure and the Axes.

For now, you will understand that working with matplotlib will already become a lot easier when you understand how the underlying components are instantiated. Or, in other words, what the anatomy of a matplotlib plot looks like:



The Figure is the overall window or page that everything is drawn on. It's the top-level component of all the ones that you will consider in the following points. You can create multiple independent Figures. A Figure can have several other things in it, such as a suptitle, which is a centered title to the figure. You'll also find that you can add a legend and color bar.

To the figure you add Axes. The Axes is the area on which the data is plotted with functions such as plot() and scatter() and that can have ticks, labels, etc. associated with it. This explains why Figures can contain multiple Axes.

You already know Matplotlib by now. When you talk about "Matplotlib", you talk about the whole Python data visualization package.

Secondly, pyplot is a module in the matplotlib package. That's why you often see matplotlib.pyplot in code. The module provides an interface that allows you to implicitly and automatically create figures and axes to achieve the desired plot.

This is especially handy when you want to quickly plot something without instantiating any Figures or Axes, as you saw in the example in the first section of this tutorial. You see, you haven't explicitly specified these components, yet you manage to output a plot that you have even customized. The defaults are initialized and any customizations that you do, will be done with the current Figure and Axes in mind.

Lastly, pylab is another module, but it gets installed alongside the matplotlib package. It bulk imports pyplot and the numpy library and was generally recommended when you were working with arrays, doing mathematics interactively and wanted access to plotting features.

You might still see this popping up in older tutorials and examples of matplotlib, but its use is no longer recommended, especially not when you're using the IPython kernel in your Jupyter notebook.

As a solution, you can best use %matplotlib magic in combination with the right backend, such as inline, qt, etc. Most of the times, you will want to use inline, as this will make sure that the plots are embedded inside the notebook.

As you have read in one of the previous sections, Matplotlib is often used to visualize analyses or calcuations. That's why the first step that you have to take in order to start plotting in Python yourself is to consider revising NumPy, the Python library for scientific computing.

Scientific computing might not really seem of much interest, but when you're doing data science you will find yourself working a lot with data that is stored in arrays. You will need to perform operations on them, inspect your arrays and manipulate them so that you're working with the data that is interesting for your analysis and that is in the right format, etc.

In short, you will find NumPy extremely handy when you're working with this data visualization library.

You have seen all components of a plot and you have initialized your first figure and Axes, but to make things a bit more complicated, you will sometimes see subplots pop up in code.

You use subplots to set up and place your Axes on a regular grid. So that means that in most cases, Axes and subplot are synonymous, they will designate the same thing. When you do call subplot to add Axes to your figure, do so with the add_subplots() function. There is, however, a difference between the add_axes() and the add_subplots() function.

The difference between fig.add_axes() and fig.add_subplot() doesn't lie in the result. They both return an Axes object. However, they do differ in the mechanism that is used to add the axes.

In contrast, the add_subplot() function doesn't provide the option to put the axes at a certain position. It does, however, allow the axes to be situated according to a subplot grid. In most cases, you will use add_subplot() to create axes. Only in cases where the positioning matters, you'll resort to add_axes(). Alternatively, you can also use subplots() if you want to get one or more subplots at the same time.

Giving Computers the Ability to Learn from Data

Machine learning gives computers the ability to learn without being explicitly programmed. It is a subfield of computer science.

The idea came from work in artificial intelligence. Machine learning explores the study and construction of algorithms which can learn and make predictions on data. Such algorithms follow programmed instructions, but can also make predictions or decisions based on data.

Machine learning is done where designing and programming explicit algorithms cannot be done. Examples include spam filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), search engines and computer vision.

Machine learning, the application and science of algorithms that make sense of data, is the most exciting field of all the computer sciences. We are living in an age where data comes in abundance, using self-learning algorithms from the field of machine learning, we can turn this data into knowledge. Thanks to the many powerful open source libraries that have been developed in recent years, there has probably never been a better time to break into the machine learning field and learn how to utilize powerful algorithms to spot patterns in data.

ML is being used increasingly and there are many reasons for this…

- The machine will eventually improve itself from its own mistakes.
- Faster than people, so it saves time.
- This took several years by working on it until it gets it right by many parameters.
- It is widely used in face recognition and also captioning photos. It is used to capture thieves, or anyone involved in a crime scene from the camera.

Machine learning is a field of computer science. It is also a type of Artificial Intelligence that enables the programmers to write programs in a more simple way. It focuses more on developing programs that teach computers to change when exposed to new data and to grow. Its goal is to understand and follow the methods by using algorithms to do that task automatically without any human assistance. In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed".

ML algorithms are increasingly being used, and there are many reasons for this.

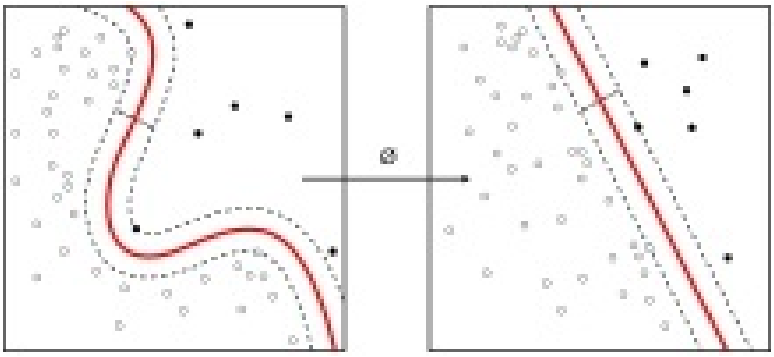
The main reason is because machine learnign has a system that is trained on some datasets that will eventually learn and improve if given a certain task. Machine Learnign is so faster than human beings, it invests time and teaches itself from the data that is given to the machine or from the mistakes it does. This took several years of working on it until it gets it right by many parameters. In present, ML is used to discover the features of relevant data in disordered datasets.

Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Machine learning tasks are classified into several broad categories. In supervised learning, the algorithm builds a mathematical model from a set of data that contains both the inputs and the desired outputs. For example, if the task were determining whether an image contained a certain object, the training data for a supervised learning algorithm would include images with and without that object, and each image would have a label designating whether it contained the object. In special cases, the input may be only partially available or restricted to special feedback. Semi-supervised learning algorithms develop mathematical models from incomplete training data, where a portion of the sample input doesn't have labels.

Classification algorithms and regression algorithms are types of supervised learning. Classification algorithms are used when the outputs are restricted to a limited set of values. For a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email. For an algorithm that identifies spam emails, the output would be the prediction of either "spam" or "not spam", represented by the Boolean values true and false. Regression algorithms are named for their continuous outputs, meaning they may have any value within a range. Examples of a continuous value are the temperature, length, or price of an object.

Active learning algorithms access the desired outputs for a limited set of inputs based on a budget, and optimize the choice of inputs for which it will acquire training labels. When used interactively, these can be presented to a human user for labeling.



Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on known properties learned from the training data, data mining focuses on the discovery of (previously) unknown properties in the data

Data mining uses many machine learning methods, but with different goals. On the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy.

Much of the confusion between these two research communities comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously unknown knowledge. Evaluated with respect to known knowledge, an uninformed method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

Machine learning also has intimate ties to optimization. Many learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances. For example, in classification, one wants to assign a label to instances and models are trained to correctly predict the pre-assigned labels of a set of examples. The difference between the two fields arises from the goal of generalization. While optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples.

There are a lot of ways to simulate human intelligence, and some methods are more intelligent than others.

Artificial Intelligence can be a pile of if-then statements or a complex statistical model mapping raw sensory data to symbolic categories. The if-then statements are simply rules explicitly programmed by a human hand. Taken together, these

if-then statements are sometimes called rules engines, expert systems, knowledge graphs or symbolic AI. Collectively, these are known as Good, Old-Fashioned AI (GOFAI).

Machine learning is a subset of AI. That is, all machine learning counts as AI, but not all AI counts as machine learning. For example, symbolic logic rules engines, expert systems and knowledge graphs – could all be described as AI, and none of them are machine learning.

Deep learning is a subset of machine learning. Usually, when people use the term deep learning, they are referring to deep artificial neural networks, and somewhat less frequently to deep reinforcement learning.

Deep artificial neural networks are a set of algorithms that have set new records in accuracy for many important problems, such as image recognition, sound recognition, recommender systems, natural language processing etc. For example, deep learning is part of DeepMind's well-known AlphaGo algorithm, which beat the former world champion Lee Sedol at Go in early 2016, and the current world champion Ke Jie in early 2017.

This means that AI is changing faster than its history can be written, so predictions about its future quickly become obsolete as well. Are we chasing a breakthrough like nuclear fission or are attempts to wring intelligence from silicon more like trying to turn lead into gold.

Given that the power of AI CO advances in AI. On a purely algorithmic level, most of the astonishing results produced by labs such as DeepMind come from combining different approaches to AI, much as AlphaGo combines deep learning and reinforcement learning. Combining deep learning with symbolic reasoning, analogical reasoning and evolutionary methods all show promise.

Training Machine Learning Algorithms

Machine learning algorithms are programs that can learn from data and improve from experience, without human intervention. Learning tasks may include learning the function that maps the input to the output, learning the hidden structure in unlabeled data or 'instance-based learning', where a class label is produced for a new instance by comparing the new instance to instances from the training data, which were stored in memory. 'Instance-based learning' does not create an abstraction from specific instances.

There are 3 types of machine learning (ML) algorithms:

- Supervised Learning Algorithms
- Unsupervised Learning Algorithms
- Reinforcement learning

Supervised learning uses labeled training data to learn the mapping function that turns input variables (X) into the output variable (Y). In other words, it solves for f in the following example: Y = f (X)

Classification is used to predict the outcome of a given sample when the output variable is in the form of categories. A classification model might look at the input data and try to predict labels like "sick" or "healthy."

Regression is used to predict the outcome of a given sample when the output variable is in the form of real values. For example, a regression model might process input data to predict the amount of rainfall, the height of a person, etc.

Ensembling is another type of supervised learning. It means combining the predictions of multiple machine learning models that are individually weak to produce a more accurate prediction on a new sample. Bagging with Random Forests, Boosting with XGBoost are examples of ensemble techniques.

Unsupervised learning models are used when we only have the input variables (X) and no corresponding output variables. They use unlabeled training data to model the underlying structure of the data.

Association is used to discover the probability of the co-occurrence of items in a collection. It is extensively used in market-basket analysis. For example, an association model might be used to discover that if a customer purchases bread, he. she is 80% likely to also purchase eggs.

Clustering is used to group samples such that objects within the same cluster are

more similar to each other than to the objects from another cluster.

Dimensionality Reduction is used to reduce the number of variables of a data set while ensuring that important information is still conveyed. Dimensionality Reduction can be done using Feature Extraction methods and Feature Selection methods. Feature Selection selects a subset of the original variables. Feature Extraction performs data transformation from a high-dimensional space to a low-dimensional space. Example: PCA algorithm is a Feature Extraction approach.

Reinforcement learning is a type of machine learning algorithm that allows an agent to decide the best next action based on its current state by learning behaviors that will maximize a reward.

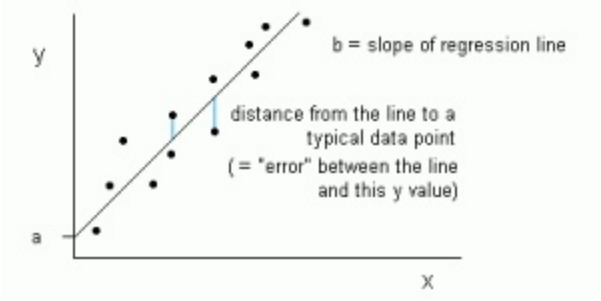
Reinforcement algorithms usually learn optimal actions through trial and error. Imagine, for example, a video game in which the player needs to move to certain places at certain times to earn points. A reinforcement algorithm playing that game would start by moving randomly but, over time through trial and error, it would learn where and when it needed to move the in-game character to maximize its point total.

The top 5 Machine Learning Algorithms for beginners are:

1. Linear Regression

In machine learning, we have a set of input variables (x) that are used to determine an output variable (y). A relationship exists between the input variables and the output variable. The goal of ML is to quantify this relationship.

In Linear Regression, the relationship between the input variables (x) and output variable (y) is expressed as an equation of the form y = a + bx. The goal of linear regression is to find out the values of coefficients a and b. Here, a is the intercept and b is the slope of the line.



2. Logistic Regression

Linear regression predictions are continuous values, logistic regression predictions are discrete values after applying a transformation function. In

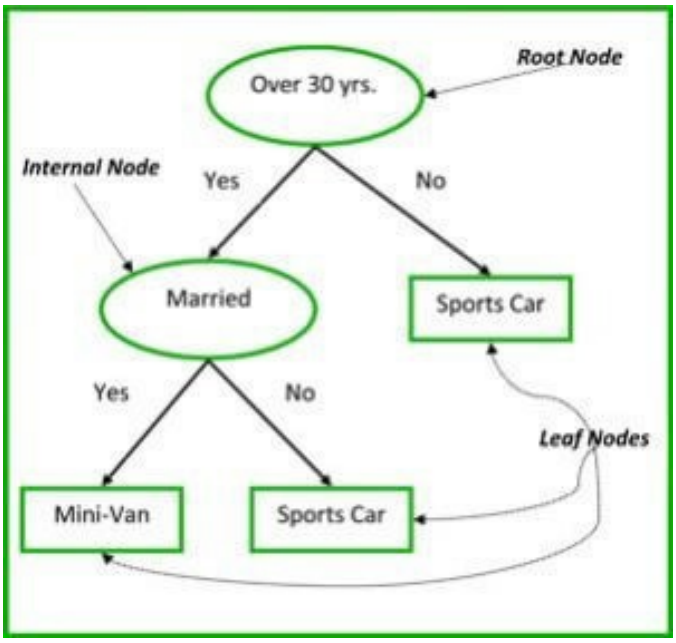
logistic regression, the output takes the form of probabilities of the default class (unlike linear regression, where the output is directly produced). As it is a probability, the output lies in the range of 0-1. So, for example, if we're trying to predict whether patients are sick, we already know that sick patients are denoted as 1, so if our algorithm assigns the score of 0.98 to a patient, it thinks that patient is quite likely to be sick.

3. CART

Classification and Regression Trees (CART) are one implementation of Decision Trees.

The non-terminal nodes of Classification and Regression Trees are the root node and the internal node. The terminal nodes are the leaf nodes. Each non-terminal node represents a single input variable (x) and a splitting point on that variable; the leaf nodes represent the output variable (y). The model is used as follows to make predictions: walk the splits of the tree to arrive at a leaf node and output the value present at the leaf node.

The decision tree in Figure 3 below classifies whether a person will buy a sports car or a minivan depending on their age and marital status. If the person is over 30 years and is not married, we walk the tree as follows : 'over 30 years?' -> yes -> 'married?' -> no.



4. Naïve Bayes

To calculate the probability that an event will occur, given that another event has

already occurred, we use Bayes's Theorem. To calculate the probability of hypothesis(h) being true, given our prior knowledge(d), we use Bayes's Theorem as follows:

P(h|d)= (P(d|h) P(h)) / P(d)

where:

- P(h|d) = Posterior probability. The probability of hypothesis h being true, given the data d, where P(h|d)= P(d1| h) P(d2| h)….P(dn| h) P(d)
- P(d|h) = Likelihood. The probability of data d given that the hypothesis h was true.
- P(h) = Class prior probability. The probability of hypothesis h being true (irrespective of the data)
- P(d) = Predictor prior probability. Probability of the data (irrespective of the hypothesis)

This algorithm is called 'naive' because it assumes that all the variables are independent of each other, which is a naive assumption to make in real-world examples.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

5. KNN

The K-Nearest Neighbors algorithm uses the entire data set as the training set, rather than splitting the data set into a training set and test set.

When an outcome is required for a new data instance, the KNN algorithm goes through the entire data set to find the k-nearest instances to the new instance, or the k number of instances most similar to the new record, and then outputs the mean of the outcomes (for a regression problem) or the mode (most frequent class) for a classification problem. The value of k is user-specified.

The similarity between instances is calculated using measures such as Euclidean distance and Hamming distance.

We have covered some of the the most important machine learning algorithms for data science:

- 5 supervised learning techniques- Linear Regression, Logistic Regression, CART, Naïve Bayes, KNN.
- 3 unsupervised learning techniques- Apriori, K-means, PCA.
- 2 ensembling techniques- Bagging with Random Forests, Boosting with XGBoost.

Conclusion

And we come to the end of this Python data science journey. With all these interesting topics and incredibly useful information, tips and tricks, I hope that you learnt something new about Python programming language and data science and now you will work properly on your projects and data researching.

Here in this awesome and easy understanding guide about the most used data programming language Python, you had the chance to get into data science, artificial intelligent and machine learning world.

Now, I hope you are well introduced in this programming language and you gain more knowledge about Python programming basics, the beginning advices and the tips how to start with programming using this data science programming language.

Also, you had the chance to read about Python data analysis and programming libraries that are necessary during your programming projects and researching activities. Data mining and visualization are also the well covered topics in this tech guide, where you can find relevant information about data collecting and manipulating thanks to the Python programming language.

And then, this Python guide is a great step-by-step tutorial how to start with building your career as a Python programmer and get properly into today’s data science and machine learning world. Catch and use every opportunity related to Python programming and make your life much easier and happier.

Lastly, I hope with all my heart that you have enjoyed this guide, and that it has been useful to you! And if you have come to this point of the book, thank you also for the time you have devoted to reading! I'm sure you've understood that it took a lot of effort to write it, and I'd be extremely grateful if you'd leave me with a 5-star review!

I wish you every success with all my heart!

Mark Solomon Brown

Recommended Readings …

LINUX FOR BEGINNERS

THE BIBLE

THE ULTIMATE BEGINNER'S GUIDE TO LEARN AND EXECUTE LINUX PROGRAMMING, FROM THE BASICS TO ADVANCED CONTENT! (LINUX PROGRAMMING, LINUX CRASH COURSE, CODING MADE EASY BOOK)

MARK SOLOMON BROWN