Lexicon

Code for the level of various terms:

- A: fundamental, best understood before or during the ML intro course
- B: important, to be understood by the end of the ML intro course
- C1: interesting, but peripheral to the subject (general culture)
- C2: interesting, directly related to the topic, but more complex than the topic (general culture)
- D: advanced, complex and interesting and related to the topic (to give vision and perspective; to "have heard of it")

Data Professions

This is a (non-exhaustive) list of professions, designed to give you an idea of the various roles that are available when dealing with data.

- **Data wrangler (A)**: individual knowing how to create, synthesize, prepare, normalize, and/or homogenize data so that it is coherent and can be processed by algorithms.
- **Data engineer (A)**: individual knowing how to store, distribute, and route data so that it can be processed by algorithms on specific computing machines. They ensure data pipelines are efficient, scalable, and reliable. The data engineers who are tasked with designing data pipelines are often called "data architects".
- **Data analyst/scientist (A)**: individual who knows how to analyze input data, formalize a problem and elaborate hypotheses, choose an algorithmic protocol for data processing, provide visualization tools, and finally evaluate the quality of the predictive model generated by the algorithm. Their work is then used to aid strategic decision-making.
- **Machine learning researcher (A)**: individual who uses mathematical, data science, and engineering principles to invent new machine learning algorithms, or improve existing ones (performance improvement, broadening of the application domain, etc).
- **Machine learning engineer (A)**: individual who uses data science and machine learning principles to implement existing ML algorithms to concrete business cases, e.g., automatic optimal logistics management. They work on selecting algorithms, preprocessing data, and tuning model parameters.
- **Database Administrator (A)**: individual esponsible for managing and maintaining databases, including ensuring data integrity, security, backup, and optimization for efficient data retrieval.

- **AI Ethics Officer (A)**: invidual dedicated to ensuring ethical and responsible development and deployment of artificial intelligence and machine learning systems.
- **Data Governance Manager (C1)**: individual responsible for establishing and implementing data governance policies and procedures to ensure data quality, security, compliance, and ethical use.
- **Data Privacy Officer (C1)**: individual who oversees data protection measures, ensuring compliance with data privacy regulations and policies to safeguard sensitive information.
- **Chief Data Officer (CDO) (C1)**: high-level executives responsible for driving data strategy within an organization, overseeing data management, analytics, and aligning data initiatives with business goals.

Computer Science

Computer Science: Programming

This section describes terms that are fundamental to understand how one can use a computer to automate processes and ideas.

- **Algorithm (A)**: mathematical process capable of transforming inputs (data that is provided) into outputs (data that is generated), according to a determined and structured protocol.
- **Heuristic (B)**: a technique of algorithmic design that allows one to trade precision or correctness of the solution against a faster calculation time. These are often used for very complex problems. Heuristic algorithms are algorithms that provide a "good guess" to a complex problem.
- **Programming language (A)**: a programming language is a way of writing text, usually a hybrid between English and mathematical language, to translate mathematical/algorithmic ideas into a form that an electronic computer can understand and execute.
- **Machine language (A)**: a language based on the binary alphabet (containing series of just 2 symbols, '0' and '1'; or 'the current is blocked' and 'the current is flowing'), allowing a computer to process information automatically.
- **Type (A)**: the category to which a given "real-world concept" belongs in a program. It is one of the most important concepts in computer science, both in practice and in theory. Mathematically, a type corresponds to the mathematical space to which a computer quantity belongs. We can distinguish, for example, the type "Integer" (signed integers), the type "Float" (floating point number, serving as an approximation of real numbers), the type "String" (textual character strings). Things like a "Player", or a

"Color", in a video game are also examples of more complex types. Note that in what follows, the word "type" should be understood to have this specific definition.

- **Code (A)**: code is text written according to the rules of a programming language.

- **Syntax (A)**: the *grammatical structure* of a language. For example, the syntax of a programming language can restrict the programmer to declare the type before a variable (e.g., `Type_A variable_a`) or after the variable (e.g., `variable_a : Type_A`). An instruction ("sentence" of a programming language) that does not respect the syntax of its language will not be executable by the machine, because it is not translatable to machine language.

- **Semantics (A)**: the *meaning* taken by an instruction/phrase in a language. An example of a semantic error would be `Integer my_variable = "bobo"`: "bobo" is not an integer, so the statement, while syntactically correct (in a language like C), is semantically incorrect. Here's an example from "real-world" language by Noam Chomsky: "Colorless green ideas sleep furiously". This is not a grammatical error, but there's clearly an error with the meaning.

- **Variable (computer science) (A)**: in computer science, a variable is a "word" declared through writing, used to store a mathematical value in memory, and then use this mathematical value conceptually. For example, (`Float speed = 5.75` would declare a variable for "speed". In the rest of the code, the programmer would then use "speed" in the code (repeatedly and in the adapted way). This makes code much clearer to read than a "5.75" that hangs around, and is meaningless with a deep understanding of the context.

- **Function (computer science) (A)**: in computer science, function is a series of instructions callable from other places in the code, which can take zero, one, or more inputs and return zero, one, or more outputs. Functions can also affect the state (memory) of a program or computer (side effects). This is for example the case for functions that display information on a screen: they affect the electronics of the computer. A "pure" function is one which does not have side effect: one which is a purely mathematical calculation that can be figured out with pen and paper, given some inputs.

- **Condition (A)**: A condition is a statement that translates a logical calculation (Boolean, a question with a true/false answer) into a conditional redirection (a branching) of the code. Keywords: `if, elif, else, and, or, not, then`.

- **Loop (A)**: A loop is a series of instructions that can be repeated, with minimal configurable changes, as long as a given condition remains true.

- **Turing-completeness (C1)**: roughly speaking, a programming language is said to be "Turing-complete" if it is "as expressive as possible". That means that it is able to execute computations on all types, store information in its memory, execute its computations contextually/conditionally, and redirect its reading head. Turing-completeness is a fundamental notion of theoretical computer science, and for reasons that are long and complex to explain, it is in a way the ultimate "speed limit" to computation. A Turing-complete language is able to execute any (reasonable) function. Almost all programming languages have a Turing-complete level of expressivity.

- **Argument (A)**: an argument is a synonym for a function input.

- **Return (A)**: a return (value) is a synonym for a function output.

- **Signature (A)**: the signature of a function is the declaration of the name of the function, and the types and names of its inputs and outputs. The function's type is defined as "types of inputs -¿ types of outputs", and is a direct consequence of these.

- **Interpreted language, compiled language (C1)**: a language is said to be "interpreted" if a software called "the interpreter" must be launched to read the code line by line, as the code runs, so that it executes. A language is said to be "compiled" if a software called the "compiler" must read the whole code and produce an executable ahead of time, before the program can be executed. Python is an example of an interpreted language. C is a compiled language.

- **Static typing, dynamic typing (C1)**: A "typing" is how the type system for a programming language was designed. It is said to be static if it is necessary at the level of the programming language's syntax, or if it can be inferred before the code is run. A typing is said to be dynamic if the interpreter or software is only able to infer the type of a computer value from its context, at runtime (and thus the declaration of the type of a variable or the arguments of a function is not necessary). Python is a dynamically typed language. C is a statically typed language. Statically typed languages are generally more restrictive in their syntax, but also generally more rigorous and reliable.

- **Paradigm (of a programming language) (C1)**: way to design a programming language. The 3 main paradigms are: imperative, object-oriented, and functional. They are not necessarily mutually exclusive. Imperative languages are very close to how the machine works, giving orders as concrete instructions (typical example: C). Object-oriented languages structure their components in "classes", types that contain both data (state: nouns or adjectives) and functions (actions: verbs). Most modern languages are inspired by object-oriented design; Python included, among others. Functional languages are very close to mathematics and

are inspired by lambda calculus; they have the advantage of producing very solid code because they are close to a mathematical proof, when they are handled by the interpreter or the compiler.

- **Version control / Git / GitHub (B)**: Technology for managing project data and archiving code, used in software development. A git "repository" contains all the archives and changes in a project's data since its creation. This means that you can use it to find anything that was saved at any time in a project's history. Platforms like GitHub allow sharing and collaboration of code projects online. Using git daily is a ***fundamental*** practice when programming.

Computer Science: Hardware

This section describes the important electronic components that allow an electronic computer to work.

- **Computer (C1)**: There are two major definitions of computer: one is a mathematical model for "how to automatically do math" (see Turing-completeness); another is a physical (generally electronic) machine (see Von Neumann architecture).
- **Central Processing Unit (CPU) (C1)**: the fundamental calculation unit of a computer.
- **Graphics Processing Unit (GPU) (C2)**: a computing unit specialized in parallelized computing. Modern GPUs are efficient in performing tasks like simultaneous scalar products, making them suitable for tasks involving linear algebra.
- **Random Access Memory (RAM) (C1)**: a memory unit that contains data and software currently running on the computer. It is cleared as soon as the computer turns off. It is generally in form of electronic flat stick.
- **Read-only memory (ROM) (C1)**: a memory unit that contains inactive but saved data and software of the machine. It can be in the form of a rotating hard disk or a flash memory drive.
- **Motherboard (C1)**: The control unit that contains most of the software necessary for the computer to start. It serves as an interface to various hardware components.
- **Power supply unit (C1)**: A unit that provides power to the computer, ensuring its proper operation.

Computer Science: Data Types

This section describes various data types that are fundamental to algorithmics in general, and data engineering/science in particular.

- **Array/List (A)**: an array, or list, is a series of values, usually of the same type, stored (usually) contiguously in memory, and accessible by index. In the vast majority of languages, the indexing of an array starts at 0. For example, if you have the following array of integers `Array<Integer> my_array = [1, 4, 6, 10, 0]`, then `array[2]` returns 6.

- **Tensor (computer science) (B)**: a tensor is an array of arrays of arrays of arrays... of elements usually of the same type. The "rank" of a tensor is its nesting level. A simple value is a tensor of rank 0. A simple array is a tensor of rank 1. An array of arrays is a rank 2 tensor. An array of arrays of arrays is a rank 3 tensor. Etc. An Excel spreadsheet with only number is an example of a rank 2 tensor.

- **Graph (A)**: a graph is a set of points (usually of the same type, called nodes), linked together 2-by-2 (called edges). A very rudimentary social network, for example, can model its users as "one node per person" and "one edge between two people if they are friends".

- **Data point cloud / Data frame / Data point space (A)**: A data frame is two-dimensional table, with "individuals" as rows, and "attributes" as columns. When its data is visualized geometrically, it corresponds to a point cloud in a mathematical coordinate frame, where each column of the table defines a "dimension", i.e. an axis of the frame. Each individual corresponds precisely to a single point in this frame. Most data tables have too many dimensions for the human brain to visualize (since we are limited to 3 spatial dimensions, 1 temporal dimension, and possibly 1 or 2 color gradients, for a total of 6, which is in practice very rarely attainable in an understandable way).

- **(Relational) database (C2)**: a database is a set of (usually massive) data tables, possibly logically linked together by 2-column tables called "junction tables". These have been the most important data structure for a long time. Relational databases have an advanced mathematical formalism, called "relational algebra", developed by a certain Edgar Codd. There are programming language specific to interactions with databases, including the SQL family of languages. Databases are so frequently relational, that anything else is generally referred to under the umbrella term "NoSQL".

- **Vectorization (C2)**: Vectorization is the expression of computations on arrays in such a way that the computations can take place simultaneously (this is called "parallelism"). This works according to the SIMD (Single Instruction, Multiple Data) principle: you run exactly the same operations on lots of data of the same type, at the same time. It is this principle that allows GPUs to do graphical computations or computation for data or neural networks faster than CPUs.

Computer Science: Miscellaneous

This section has some vocabulary which useful for general culture of computer science, such as network engineering, or systems engineering.

- **Server (C1)**: a server is a piece of software that allows other computers to interact with it. You can think of a "server" as a program "at whose door you can knock".
- **Client (C1)**: a client is a piece of software that can make requests to another computer. Web browsers are the typical "client" software that most people know about. You can think of a client as a program "that goes to knock on other people's door".
- **Peer (C1)**: a peer is a piece of software that is both a client and a server.
- **Operating System (OS) (C1)**: an operating system is a piece of software that allows the management and operation of other software. It is generally the first piece of software that you want to run when turning on a computer. Examples of such architectures include Mac OS, Windows, Linux-Debian, Linux-RedHat, iOS, Android, Raspberry Pi, Microsoft Azure, and more.
- **Cross-platform (C1)**: Refers to software or code that can run on various operating systems.

Computer Science: Programming for Data Science and Machine Learning

This section presents the various tools that programmers who work in data science and machine learning use (most of the time). We explicitly ignore the R programming language, since the goal is to prepare the student to Machine Learning.

- **Python (A)**: Python is a programming language designed to look a lot like plain English, to be simple to learn, to be dynamically typed, to be interpreted, and to have a richness of syntactic expression. This makes it a very good language for discovery through code and experimentation/prototyping. However, this language is sometimes quite slow at runtime, not easy to develop cross-platform, and dynamically typed. This makes it often avoided in production-grade environment, unless special care is taken. Python is above all the language of research, learning, scripting, and prototyping.
- **Jupyter (A)**: Technology used to launch a local server which makes one capable of running Python in a browser. It is very useful for easy Python development, given the problems of cross-platform distribution of Python, as well as those in the production of Python executables.

- **Numpy (B)**: Standard linear algebra library in Python.
- **Scipy (B)**: Standard scientific computing library in Python.
- **SciKit-Learn (B)**: Standard Machine Learning library in Python.
- **PyTorch (B)**: A sort of combination of Numpy and SciKit-Learn which has in recent years become the new standard for Machine Learning development.
- **Pandas (B)**: Library for managing dataframes (data tables) in Python. Very useful, quite standard in data science.
- **Matplotlib (B)**: Complete, but not very pretty, data visualization library in Python. A bit complex to get into at first, but essential to know.
- **Seaborn (B)**: Data visualization library specialized in data science. Does every visualization task that is very "classical" quite well, simply, and beautifully. It is however sometimes difficult or impossible to do data visualizations with it, if these visualizations are too custom.
- **SQL (C1)**: Fundamental database query language. Replaced since by languages that descend from it often (PostgreSQL, GraphQL), even if many databases still work with SQL.

Math for Data Science

Math for Data Science: Abstract Algebra

This section describes the fundamentals that one tends to see during the first weeks of an undergrad of mathematics. While mastery is not essential for machine learning, understanding the following concepts makes the learning of all the mathematics related to data science much, much easier.

- **Set (A)**: a set is a collection of mathematical objects. For example, the binary alphabet is the set containing the symbols 0 and 1, noted $\{0, 1\}$. The natural numbers form another set, noted $\mathbb{N} = \{0, 1, 2, 3, \dots\}$. You can think of a set as a deck of cards are unique, and there might be an infinite amount of cards.
- **Space / algebraic structure (A)**: a space is the combination of a set and with some rules/laws/properties and/or operators on its elements. For example, the '+' operator on the binary language (set of all combinations of symbols of the binary alphabet noted $\{0, 1\}^* = \{$"","0","1","00","01","10","11",...$\}$) is concatenation, and glues the "words" together (e.g., `"1010" + "111" = "1010111"`). The '+' operator in natural numbers is addition (e.g., $5 + 7 = 7 + 5 = 12$). An example of a property is commutativity: $a + b = b + a$. This property is verified in the natural integers but not in the binary language. You can think of properties as the rules you want to choose to play with your chosen deck of cards.

- **Function (mathematics) (A)**: a function is an association of the elements of a set of inputs (called a domain) with a set of outputs (called a codomain), where each input has at most one output image, so that the result of a function, given a fixed input, is deterministic. The square root is an example of a "1 input, 1 output" function. The classical operators (like addition) are examples of "2 input, 1 output" functions (for example, written in this way $+(5, 7) = 5 + 7 = 12$, this fact becomes clearer).
- **Application / Map**: You can consider that "application"/"map" and "function" are synonymous terms in mathematics (by abuse of language, even if there is a technical distinction between the two in French).
- **Image**: the image y by a function f of an element x of the domain is the unique element $y = f(x)$ of the codomain. Put more simply, the "image" is the result value of applying the function, for a given input/output pair.
- **Antecedent**: an antecedent x of an element y of the codomain of a function f is an element of the domain such that $y = f(x)$. Put more simply, the "antecedent" is the "origin" of some result of the function, for a given input/output pair.
- **Variable (mathematics) (A)**: a variable is a value which is named and typed (declared to be a member of a certain space) but not specified. This makes it so that this named value can act as a placeholder, to represent "any value" from the space in which it exists.
- **Parameter (A)**: a parameter is a variable which is given a specific value, for experimentation's sake, but which could just as easily be changed.
- **Function composition**: if f is a function from A to B and g is a function from B to C, then there exists a function $h = g \circ f$ from A to C, such that h is the concatenation of f and then g (ie, first applying f, then g), where the return value of the function f is given as an argument to g.
- **Continuous function (A)**: a function is said to be continuous if its graphical representation (linking its inputs and outputs) has no "break".
- **Derivable function (A)**: a derivable function is a continuous function which does not have any "sharp corners" in its graphical representation.
- **Derivative (A)**: the derivative of a function is another function, representing the slope of rise or fall at each point of this function as a numerical value. A rise for an input point corresponds to a positive value of the derivative at that same input point; a fall corresponds to a negative value at that point.
- **Exponential function (A)**: the exponential function is the "fundamental" function whose derivative is the exponential function itself. Among its many properties, we have $e^{a+b} = e^a \cdot e^b$, i.e., it transforms an addition of inputs into a multiplication of outputs.

- **Logarithmic function / Neperian logarithm (A)**: the reciprocal of the exponential function. The derivative of the neperian logarithm is the function ($x \to \frac{1}{x}$). Also, $\ln(a \cdot b) = \ln(a) + \ln(b)$, i.e., the neperian logarithm turns a product into a sum.

- **Logistic function (B)**: The logistic function is a continuous, strictly increasing function, varying from 0 to 1 (in its outputs) from $-\infty$ to $+\infty$ (in its inputs).

- **Integral (C1)**: the integral is the "ruler" (in the sense of a tool to measure) of mathematics. It is also the "anti-derivative".

- **Symbolic notation of iterated sums and products (B)**: A large Σ represents an iterated sum, a large Π represents an iterated product. Example: $\sum_{i=0}^{i=5} 2^i = 2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 = 1 + 2 + 4 + 8 + 16 + 32 = 63$.

Math for Data Science: Linear Algebra

This is the meat of data science as a subject. All of machine learning, statistics, data science, most of physics, economics, etc, rely on linear algebra in some form or another.

- **Vector space (B)**: a vector space E is an algebraic structure with commutative addition and subtraction (elements of the vector space, called **vectors**, behave like an "abelian group" under addition). It is always accompanied by another structure K called "field", itself having usual addition, subtraction, multiplication and division (except by 0): this is the usual arithmetic that you've seen throughout school. Elements of K, called **scalars** allow one to change (scale) the size of vectors. This structure is coupled with a series of laws that allow the combined operation of K and E in a correct way.

- **Vector addition (A)**: addition of $E \times E \to E$ (2 inputs in E, one output in E).

- **Vector scaling (A)**: multiplication of $K \times E \to E$ (1 input in K, 1 input in E, 1 output in E).

- **Scalar (A)**: a scalar is an element of K. In general, we choose $K = \mathbb{R}$, the field of real numbers. A scalar is a tensor of rank 0.

- **Vector (A)**: a vector is an element of E. In general, a vector will be an array of n elements. The number n is then shared by all elements of E and is called the "dimension" of E. E is then denoted $\mathbb{R}^n$ (for the vector space of real numbers of dimension n). The space of functions from $\mathbb{R}$ to $\mathbb{R}$ are also an example of vectors, because they respect the same laws (addition and scaling). The space $(\mathbb{R} \to \mathbb{R})$ is a vector space. A vector is a tensor of rank 1.

- **Linear combination (B)**: any possible arrangement of scalings and additions of a set of vectors. For example, if we have 3 vectors u, v, and w, then $72 \cdot u + \frac{1}{3} \cdot v - 4.5 \cdot w$ is a linear combination of vectors u, v, and w. We can always simplify a linear combination back to a form where each vector is multiplied by a single scalar, and the list of these scalar-vector products is summed (as is used in the example).

- **Linear independence (A)**: a set of n vectors is linearly independent if the only way to get the zero vector (the zero of E, the element that changes nothing by vector addition) by linear combination is to have the scalar factor in front of each scalar-vector pair be zero. Intuitively, this means that each vector in the set contributes to creating a new dimension, independent of those created by the other vectors: there is no way to express one of the vectors as a linear combination of the others.

- **Basis (A)**: a set of linearly independent vectors that can describe any point in a vector space. There are always precisely as many elements in a basis as there are dimensions in a vector space.

- **Dimension (A)**: number of elements in any basis of a vector space; number of axes needed to represent a vector space.

- **Colinearity (A)**: two vectors are said to be colinear if their direction describes the same line passing through the origin. This means that one can be scaled into the other.

- **Linearity / Linear application (A)**: Said of a function from E into F, where E and F are vector spaces, that sends the origin of E (its zero vector) to the origin of F, and keeps all parallel lines in E parallel in F. Algebraically, a function/application is said to be linear if and only if $f(0_E) = 0_F$ and $f(ku + v) = kf(u) + f(v)$, for all k in K and all u and v in E. Geometrically something "linear" is something straight (a line, a plane, etc); something non-linear is something curved (a sphere, a parabola, etc). "Linear" often also refers to polynomials of degree 1 (which have straight shapes), "quadratic" refers to polynomials of degree 2, "cubic" to polynomials of degree 3, etc.

- **Matrix (A)**: a matrix is a rectangle of scalars. A matrix of size $m \times n$ (m rows, n columns) represents a linear map of $\mathbb{R}^n \to \mathbb{R}^m$ (i.e., a function with as input a vector of dimension n, and as output a vector of dimension m). A matrix is a tensor of rank 2.

- **Matrix multiplication (A)**: matrix multiplication is the function composition of linear maps.

- **Dot product (A)**: the dot product is a product of $E \times E \to K$ (2 vectors of E as input, one scalar of K as output). It is computed as the sum of the term-to-term product of each coordinate of its two inputs. It is denoted $\langle u, v \rangle$, and is also equal to $\|u\| \cdot \|v\| \cdot \cos(u, v)$. Algebraically, for

two vectors u and v of dimension n with coordinates u_i and v_i, the dot product is defined as: $\langle u, v\rangle = \sum_{i=1}^{i=n} u_i \cdot v_i$. The dot product algebraically encodes two pieces of geometric information (one about angles, and one about lengths) into a single number. In practice, the dot product of two vectors of norm 1 is equal to the angle between these two vectors; the dot product of two collinear vectors is equal to the multiplication of their norm.

- **Norm (A)**: The norm of a vector is the distance of displacement that this vector represents. It is usually defined using the dot product, as $\|u\| = \sqrt{\langle u, u\rangle}$ (Euclidean norm, also called the 2-norm, the one from the Pythagorean theorem). There are different norms (useful in ML), like the "supremum norm/∞-norm" or the "Manhattan norm/1-norm".

- **Quadratic norm (A)**: the quadratic norm is the norm of a vector squared. It is usually defined as $\|u\|^2 = \langle u, u\rangle$.

- **Cosine (A)**: value used to define the angle between 2 vectors. Algebraically, $\cos(u, v) = \frac{\langle u,v\rangle}{\|u\|\cdot\|v\|}$.

- **Distance (metric) (B)**: function to define a measure of the distance between two vectors. If the vector space is normed (has a norm function), a distance can always be defined from it as $d(u, v) = \|u - v\|$.

- **Span (generated vector subspace) (B)**: the vector subspace generated by a family (set) of vectors is the set of points that are reachable by linear combinations of the vectors of this family.

- **Normalization (linear algebra) (A)**: vector scaling of a vector u by the value $\frac{1}{\|u\|}$, in order to find a vector $\hat{u}$ of norm 1 and the same direction as u.

- **Tensor (B)**: A tensor is an arrangement of scalars. A tensor of rank zero is a point of numbers (a single scalar). A tensor of rank one is a line of numbers: either a vector (column vector) or a covector (row vector). A tensor of rank two is a rectanle of numbers (a matrix). A tensor of rank three is a cube of numbers (hypermatrix). Etc.

- **Linear form / covector (C1)**: A linear form is a linear map of $E \to K$. Linear forms are row vectors.

- **Vector subspace (B)**: A vector subspace is a vector space contained in another vector space. A vector line, or a vector plane (i.e. passing through the origin) in a vector space of dimension 3 are examples of a vector subspace. Any space is a vector subspace of itself.

- **Hyperplane (B)**: A hyperplane is a vector subspace of dimension $n - 1$ in a vector space of dimension n. The 2D planes are the hyperplanes of 3D space. The 1D lines are the hyperplanes of 2D space. The role

of a hyperplane is to separate a vector space into exactly 2 pieces. For example, any "mirror" symmetry is done with respect to a hyperplane, whatever the dimension n of the enclosing space.

- **Eigenvectors/Eigenvalues (C2)**: Any linear map has vector subspaces that are stable (i.e. if an input is in that special stable subspace, then so is its output by the linear map in question). The are called eigenspaces. Linear maps in 1D eigenspaces amount to dilations (since both input and output are colinear). The vectors serving as the basis for these vector subspaces are called eigenvectors; their respective dilation coefficient is called an eigenvalue.

- **Euclidean space (B)**: vector space along with a dot product operator, allowing it to define a norm (the Euclidean norm) and therefore a metric (the Euclidean metric). This is the usual kind of space in which we do math and physics in high school.

- **Normed vector space (B)**: vector space along with a norm operator over vectors, allowing it to define a metric.

- **Metric space (B)**: vector space having a notion of distance between vectors (though not necessarily a notion of size for vectors).

Math for Data Science: Probability Theory

This section describes the math of probability theory, which forms a lot of the backbone of statistics. Statistics is basically the combination of probability and linear algebra.

- **Universe of discourse (A)**: set describing the collection of possible outcomes of a random experiment (some situation with a random outcome). For example, the universe of discourse for a single roll of a regular (six-sided) die (also called a "1d6") is the set $\Omega = \{1, 2, 3, 4, 5, 6\}$.

- **Probability (A)**: A probability is a function over a universe of discourse that returns an outcome between 0 (ie, 0% chance) and 1 (ie, 100% chance). This universe of discourse is geometrically an object of measure 1 (a length of 1 in dimension 1, an area of 1 in dimension 2, a volume of 1 in dimension 3, an n-hypervolume of 1 in dimension n). The probability function assigns a measure between 0 and 1 to all collections of outcomes. For example, in the universe of discourse described above (the 1d6), the event "an even number is rolled" corresponds to the set $e = \{2, 4, 6\}$ and is precisely one half of the full space (measure 0.5), which corresponds to a 50% chance.

- **Conditional probability (Bayesian probabilities) (A)**: A conditional probability is a probability considering/knowing that an event e is necessarily true. This is geometrically equivalent to restricting oneself to the

section of the universe of discourse where e is true, and considering that this new geometric object is now of measure 1 (ie, certain). For example, take a 1d6, $e_1 = \{1, 2, 3, 4\}$, ie, "rolling 4 or less", and $e_2 = \{2, 4, 6\}$, ie, "rolling an even number". Then, the probability of "rolling an even number *knowing that* we rolled 4 or less" is 0.5, since the set $e_3 = \{2, 4\}$ is half the measure of the the set $e_1 = \{1, 2, 3, 4\}$ which is our new, "knowing that", universe. This should not be confused with the probability of "rolling an even number *and* rolling 4 or less", which would be 1/3.

- **Random variable (A)**: probabilistic experiment where the result is assigned a "success" value (in general a real number). Algebraically, a random variable is a function of the universe of discourse in (in general) the set of real numbers, usually denoted $X : \Omega \to \mathbb{R}$. For example, let's say I win 0.50€ per point on a 1d6 die, but if I roll a 6, I instead lose 3€. Our function looks like $X = \{1 \to 0.5; 2 \to 1; 3 \to 1.5; 4 \to 2; 5 \to 2.5; 6 \to -3\}$.

- **Random vector (B)**: probabilistic experiment where the outcome is assigned multiple values (usually n real numbers). For example, if I refer to the value of two normal throws of the dice in order (2d6) as "x" for the first throw, and "y" for the second, then the example of a random vector where I win 2x candies but lose 0.5xy€ on each pair of throws can be defined as $X = (x, y) \to (2x, -0.5xy)$. Models in data science are usually random vectors that try to match the shape and properties of a cloud of data points, if the probabilistic experiment is performed repeatedly on different points.

- **Expectation (A)**: average of the results of a random variable over a universe, weighted by their probability. It corresponds to the "average return that can be expected in the long run, if we repeat the experiment an infinite amount of times".

- **Variance (A)**: Average of the squared deviations of each of the outcomes from the expectation. This gives an idea of the "spread" of the values of the experiment away from the average. The variance can also be understood as the covariance of a random variable (or vector) with itself, i.e. the quadratic norm of a random variable. In the case of a random vector, the variance takes the form of a symmetric, positive semidefinite matrix, called the variance-covariance matrix.

- **Standard deviation (A)**: the standard deviation is the square root of the variance of a variable or a random vector. It is therefore the norm of a random vector, noted σ_X.

- **Covariance (B)**: The covariance is a measure of how much two random variables vary together. It is the dot product of spaces of random vectors. It is noted $\text{cov}(X, Y)$.

- **Pearson's correlation coefficient (B)**: a measure of how closely two events are correlated. It is the cosine of the spaces of random vectors. It is defined as $\text{cor}(X,Y) = \frac{\text{cov}(X,Y)}{\text{stddev}(X)\cdot\text{stddev}(Y)}$.
- **Bias (A)**: difference between the mean/expectation given by the model, and the one given by the experiment (the real data).

Math for Data Science: Differential Geometry

This is a more advanced mathematical subject, but it is necessary for more advanced topics in data science, such as the explanation for several advanced algorithms, explainability in ML, fields of data science like topological data analysis, etc.

- **Scalar function (B)**: A scalar function is a function from $\mathbb{R}^n \to \mathbb{R}$ that is not necessarily linear (unlike linear forms). Ex: $f : \mathbb{R}^3 \to \mathbb{R}$, such that $(x, y, z) \to xy + z^2 + \cos(x)$.
- **Partial derivative**: derivative of a single coordinate of scalar function. Each coordinate is obtained with the standard derivative, considering all coordinates constant, except one. E.g., going with the previous function, here is the partial derivative in the x coordinate: $\frac{\partial f}{\partial x} = y - \sin(x)$.
- **Differential / gradient (B)**: The differential / gradient is the "fundamental" derivative of a scalar function. The differential gradient indicates the direction in which to move to get the largest increase in the output value. The differential / gradient is calculated as the vector of all partial derivatives. The only difference between the two is that the differential (noted df) is a row vector (covector) and the gradient (noted ∇f) is a column vector (i.e., their input/output space are reversed). E.g., going with the previous function:

$$df = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \end{bmatrix} = \begin{bmatrix} y - \sin(x) & x & 2z \end{bmatrix}$$

- **Multivariate function**: a multivariate function is a function from $\mathbb{R}^n \to \mathbb{R}^m$, which is not necessarily linear. E.g.: $f : \mathbb{R}^3 \to \mathbb{R}^2$ such that $(x, y, z) \to (xy + \cos(x), xyz + z^2)$. Note that we can write $f(x, y, z) = (f_1(x, y, z), f_2(x, y, z))$, where $f_1(x, y, z) = xy + \cos(x)$ and $f_2(x, y, z) = xyz + z^2$ are both scalar functions.
- **Jacobian**: the "fundamental" derivative of a multivariate function. The Jacobian is a matrix of size $m \times n$, where the i-th row corresponds to the differential of the i-scalar function. To caricature, "we stack the differentials". Note that the Jacobian is itself a function from $\mathbb{R}^n \to (\mathbb{R}^n \to \mathbb{R}^m)$; that is, your Jacobian gives you a different matrix for each point in your space where you compute it. Geometrically, the Jacobian gives you the

best linear approximation at a point (x, y, z) of its return in the output space.

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \end{bmatrix} = \begin{bmatrix} y - \sin(y) & x & 0 \\ yz & xz & xy + 2z \end{bmatrix}$$

- **Manifold (C2)**: a manifold is an object which is "locally euclidean". This means that, if you zoom in enough on your object, it looks flat. It's a kind of object on which we can define a geometry through which differentiable functions can exist. Any vector space, as well as any space of functions between vector spaces, is a manifold. The sphere, the torus, the Klein bottle, the n-dimensional Cartesian plane ($\mathbb{R}^n$), are all examples of manifolds.

- **Topology (C2)**: a branch of mathematics that studies the local and global structures of manifolds. Also, a term for a "fabric of reality / choice of mathematical universe" based on the language of sets, on such spaces, in order to define notions of distance, measure, continuity, derivability, etc.

- **Measure theory (C2)**: extension of topology to build a more powerful version of the integral, and to define a way to compute lengths, areas, volumes, whatever the dimension. Probabilities are geometrically understood as "n-volumes relative to a global space of n-volume 1", and measure theory is thus the foundation of all modern probability theory.

- **Topological data analysis (D)**: a technique that aims to study the geometric form (the manifold) on which data points exist in an encompassing statistical vector space, in order to drastically reduce its dimension or to make its logical structure explicit.

- **Differential Geometry (D)**: theory of derivation on manifolds.

- **Information Geometry (D)**: mathematical theory relying on probability, statistics, information theory, measure theory and differential geometry to establish statistical spaces (random vector spaces; data point cloud spaces) and dynamics on general manifolds.

Data science & Machine Learning

General vocabulary of statistics, data science and ML

Here we define terms that are useful all around for scientific/technical data science and ML.

- **Regression**: a type of model learning that seeks to predict numerical values for fictitious input data, close to the real output data in relation to its input data.

- **Classification**: a type of model learning that seeks to predict the categorization of some output based on an input data point.
- **Model (A)**: a model is a mathematical function used as a hypothesis to generate a virtual data point from a chosen input. The goal of a model is to predict the actual data as closely as possible.
- **"Garbage in, garbage out"**: a saying emphasizing the importance of quality data for accurate results in data science and machine learning.
- **R-squared (B)**: one of the measures of the validity of a model.
- **ML fields of study**: Natural Language Processing (NLP); Computer Vision; Decision-making; Business analytics; Ranking; Darwinian algorithms/Reinforcement learning, etc.
- **Neural network (B)**: a neural network is an architecture using linear algebra to build a graph of computational cells (neurons) and run an optimization protocol (part darwinian, part search for an optimum of a hard mathematical function) to allow an algorithm to become efficient for a given task in an unsupervised way.
- **Curse of dimensionality (B)**: the growth of data points required for accurate representation grows exponentially with the number of attributes, making many algorithms inefficient for high-dimensional spaces.
- **Dimensionality reduction (C2)**: techniques used to reduce the dimension of high-dimensional spaces while preserving information. Used for visualization and to make data tractable for algorithms.

Machine Learning: Learning

Here, we define the various terms that make up the "learning" part of machine learning.

- **Supervised Learning (A)**: type of machine learning, where the data is all labeled and specified by a human being, so that the algorithm can use this as a foundation to know if it is right or wrong.
- **Unsupervised Learning / Self-Supervised Learning(A)**: learning where the algorithm itself is supposed to classify, measure or label the data on its own.
- **Semi-Supervised Learning (C2)**: A learning approach where a model is trained on a mix of labeled and unlabeled data. It aims to leverage the unlabeled data to improve model performance.
- **Reinforcement Learning (A)**: learning where the algorithm itself perceives its environment, and learns according to it in a pseudo-darwinian way to perform a task more and more efficiently, based on some fitness/cost function.

- **Transfer Learning (C1)**: A technique where a pre-trained model on one task is fine-tuned on a new, related task. This approach can significantly speed up the training process and improve performance, especially when labeled data is limited. Large Language Models, like GPT-3, are pre-trained on massive amounts of text data and then fine-tuned for specific tasks. This transfer learning approach allows them to perform well on a wide range of language-related tasks.

- **Fitness/Cost function (A)**: a fitness/cost function is a function that, for a whole series of inputs, calculates the distance between the output of this input passed through the model function and the actual result in the data point cloud. The sum of these distances is the result of our cost function and is a measure of the accuracy of our model. Machine learning algorithms try to improve fitness / minimize cost, to become better at their given problem.

- **Hyperparameters (B)**: the parameters that allow one to manage the learning rate or direction of an algorithm.

- **AutoML (C1)**: Automated Machine Learning refers to the process of automating the end-to-end process of applying machine learning to real-world problems, including data preprocessing, feature selection, model selection, and hyperparameter tuning.

- **Human-in-the-Loop (C2)**: Incorporating human reviewers to guide and oversee generated content (often for GANs or LLMs in particular) which helps ensure higher quality and ethical standards.

Data Science & Machine Learning techniques

Here we describe various techniques or meta-approaches that are common in data science and machine learning.

- **Normalization (data science) (B)**: a change of benchmark used to rescale the distances between the points of a data point cloud, so as to express the same relative information between these data points, but so that a specific algorithm has better results.

- **Gradient descent (B)**: an algorithm using iterative gradient calculation to find an extremum of a complex mathematical function (typically, the minimization of a cost function).

- **Principal Components Analysis (C2)**: This is a dimensionality reduction technique (probably the best known and most widely used). Its principle is to express the underlying frame of reference of the data space in its most "expressive" form, i.e. that maximizes the variance of the data projected on the axes of the new frame of reference. This allows having a maximum of information on the data cloud with a minimum of dimensions

(axes). The origin of this new frame is the mean of the data points, and the axes are computed by Singular Value Decomposition (SVD) of the covariance matrix. Projecting the data points onto the main axes allows visualizing an approximate but rather accurate version of a much more complex high-dimensional data space.

- **Decision Trees, Random Forests (C2)**: machine learning techniques that build decision trees (conditional "if" trees) from data, used for classification and regression. They are easy for humans to interpret.
- **Kernel Methods / Integral Kernel / Window functions / Statistical Kernel / RKHS (C2/D)**: methods using covectors of vector spaces of functions, or modified dot products, to transform the shape of data point clouds into something easier to handle (e.g., by linear methods) and achieve better results.
- **Gradient Boosting (B)**: An ensemble learning technique that combines multiple weak learners (typically decision trees) to create a strong predictive model. It builds the model in a stage-wise manner, focusing on correcting the errors of previous iterations. XGBoost is a popular open-source implementation of gradient boosting that is known for its efficiency and performance in machine learning competitions. LightGBM is another gradient boosting framework that is designed to be memory-efficient and fast, making it suitable for large datasets. CatBoost is a gradient boosting algorithm that handles categorical features automatically, eliminating the need for extensive preprocessing.
- **Attention Mechanisms (C2)**: Techniques that allow models to focus on specific parts of input data when making predictions. Attention mechanisms have significantly improved the performance of NLP models.
- **Neural Style Transfer (C2)**: A technique that combines the content of one image with the artistic style of another image using neural networks.
- **Anomaly Detection (C2)**: A field of machine learning focused on identifying rare events or outliers in data. It has applications in fraud detection, network security, and more.

Data Science & Machine Learning algorithms

- **Linear regression (B)**: a technique that allows establishing a model (here, an approximation of a cloud of data points) in the form of a vector subspace minimizing the distance to a set of points.
- **Logistic regression (B)**: form of regression using the logistic function to obtain a probabilistic model of binary classification (true/false).

- **K-means clustering (C2)**: A classification algorithm that seeks to partition a group of n observations (an n-point data point cloud) into k "clusters" where each point belongs to the cluster with the closest mean. The algorithm organizes the points of the initial cloud by iteratively readjusting hyperplanes to converge to a local optimum.
- **K-nearest neighbors (B)**: algorithm for both classification and regression, assigning to each point either the category of its k nearest neighbors (classification) or the average of the values assigned to these k nearest neighbors (regression). It is based on a choice of distance and requires normalization of distances.
- **Support Vector Machines (SVM) (C2)**: supervised learning algorithm for linear classification using hyperplanes. Extensions exist for regression and non-linear classification (using kernel methods).
- **Naive Bayes Classifiers (C2)**: family of algorithms based on conditional probabilities to perform classification.
- **Multiplayer Perceptron (MLP) (B)**: a fundamental neural network architecture that takes input, processes through layers, and produces an output. Learning is done using the backpropagation algorithm.
- **Convolutional Neural Network (CNN) (C2)**: specialized neural network for image processing, using convolution cells to extract local shapes.
- **Recurrent Neural Networks (RNN) (C2)**: neural network architecture with linked non-neighboring layers that can affect each other. It is designed to handle sequences of data, making them suitable for tasks like time series prediction, natural language processing, and more.
- **Long Short Term Memory Network (LSTM) (C2)**: a type of complex Recurrent Neural Network, capable of learning and remembering longer sequences of data, used for NLP and speech recognition.
- **Large Language Model (LLM) (C2)**: Large Language Models (LLMs) are advanced artificial intelligence systems designed to understand and generate human-like text based on vast amounts of training data. LLMs, such as OpenAI's GPT-3, utilize deep learning techniques to learn the statistical patterns and structures of language. These models have the capacity to generate coherent and contextually relevant text, making them invaluable tools for natural language processing tasks. LLMs can perform a wide range of language-related tasks, including text generation, language translation, sentiment analysis, chatbot interactions, content summarization, and more. Their capabilities stem from the complex neural architectures they employ, which consist of multiple layers of interconnected processing units. LLMs have garnered significant attention for their potential to revolutionize various industries and applications by enhancing

human-computer interactions and enabling sophisticated language-related tasks. [This definition was generated using ChatGPT with GPT 3.5]

- **Gated Recurrent Unit (GRU) (C2)**: A variation of the LSTM architecture with fewer parameters, often used for similar tasks as LSTMs.
- **Generative Adversarial Networks (GAN) (C2)**: neural network architecture with two agents, a generator (which acts like forger) and a discriminator (which acts like an inspector), that improve by competing with each other. Used to produce realistic data in image, video or audio format.
- **Transformer Architecture (C2)**: A neural network architecture designed for sequence-to-sequence tasks, such as machine translation. Transformers have revolutionized NLP and are the foundation of models like BERT, GPT, and T5.
- **Q-learning (C2)**: Model-free reinforcement learning algorithm widely used for AIs, generally by having them "play video games" (react and evolve in a given simulated environment).

Other: Design

- **Reverse engineering (B)**: The process of studying the functioning of a product or algorithm to understand its design and production. This can involve reproducing or improving upon the original design.
- **Divergent thinking (B)**: The ability to think creatively and generate a wide range of possible solutions to a design problem. It involves thinking outside the box and considering unconventional ideas.
- **Convergent thinking (B)**: The ability to systematically filter and evaluate possible solutions to a problem based on a defined set of constraints. It is similar to the scientific method (aka Cartesian method).
- **Design thinking (B)**: A mental protocol for creative problem-solving that combines both approaches of divergent and convergent thinking.
- **Algorithm design (B)**: The process of inventing protocols for automated data processing, involving the creation of step-by-step instructions for solving specific problems.