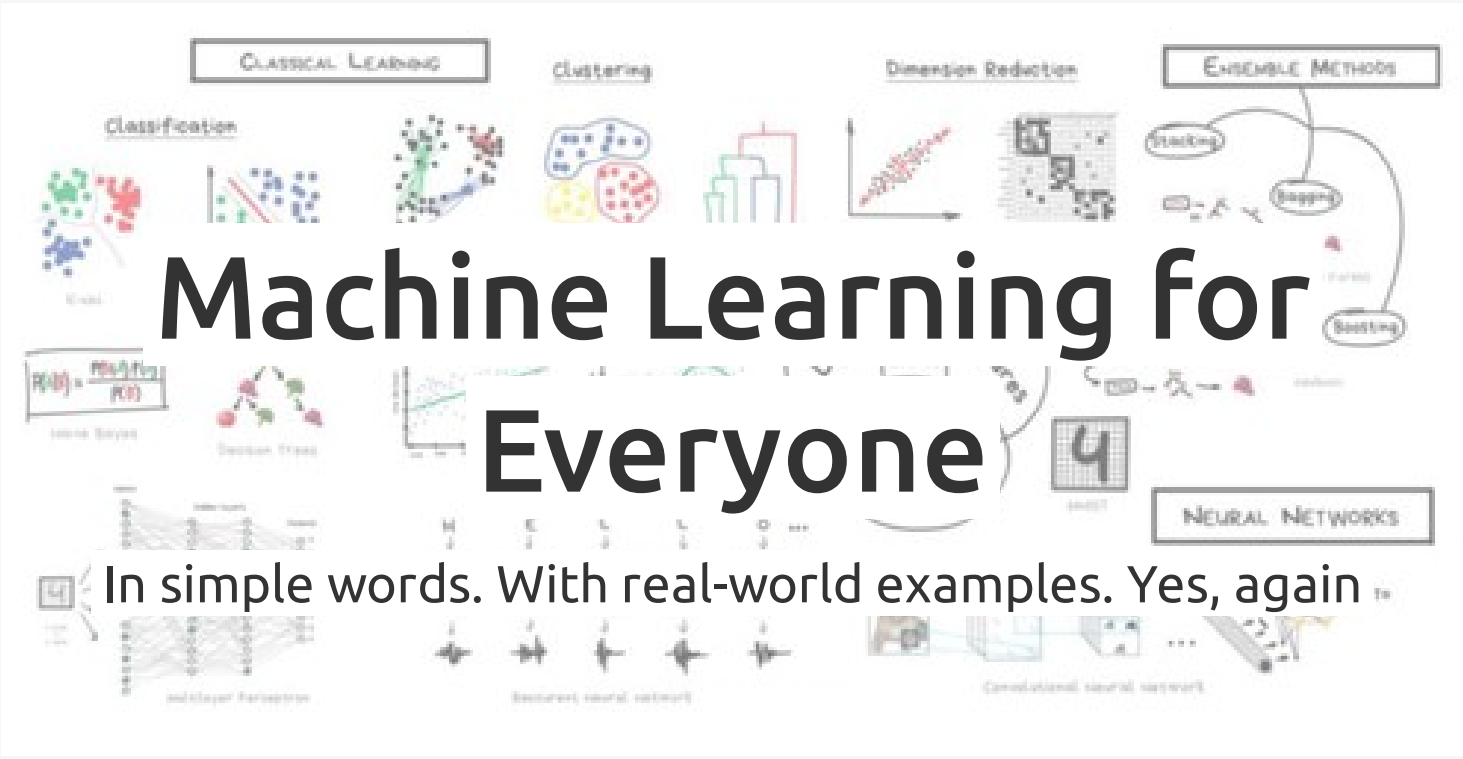


Machine Learning for Everyone

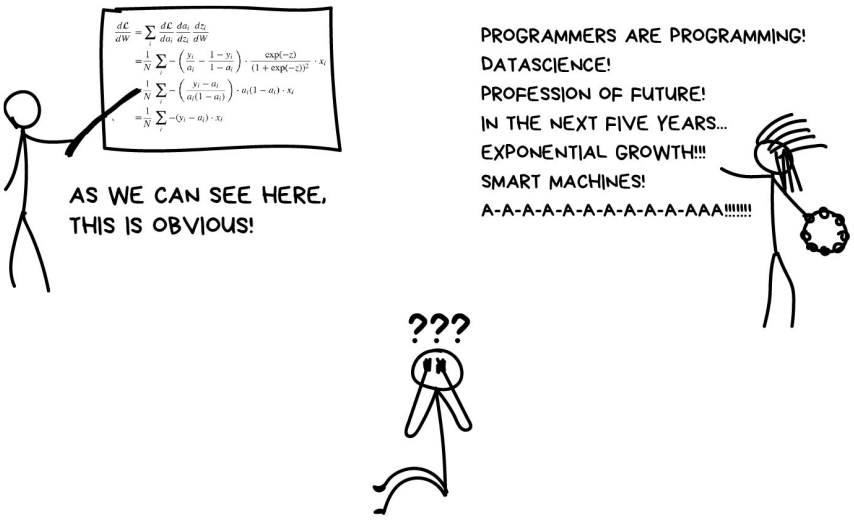
In simple words. With real-world examples. Yes, again

> Other languages: Russian | Portuguese

Machine Learning is like sex in high school. Everyone is talking about it, a few know what to do, and only your teacher is doing it. If you ever tried to read articles about machine learning on the Internet, most likely you stumbled upon two types of them: thick academic trilogies filled with theorems (I couldn't even get through half of one) or fishy fairytales about *artificial intelligence*, *data-science magic*, and *jobs of the future*.

I decided to write a post I've been wishing existed for a long time. A simple introduction for those who always wanted to understand machine learning. Only real-world problems, practical solutions, simple language, and no high-level theorems. One and for everyone. Whether you are a programmer or a manager.

Let's roll.



Why do we want machines to learn?

This is Billy. Billy wants to buy a car. He tries to calculate how much he needs to save monthly for that. He went over dozens of ads on the internet and learned that new cars are around $20,000, used year-old ones are $19,000, 2-year old are $18,000 and so on.

Billy, our brilliant analytic, starts seeing a pattern: *so, the car price depends on its age and drops $1,000 every year, but won't get lower than $10,000.*

In machine learning terms, Billy invented *regression* – he predicted a value (price) based on known historical data. People do it all the time, when trying to estimate a reasonable cost for a used iPhone on eBay or figure out how many ribs to buy for a BBQ party. 200 grams per person? 500?

Yeah, it would be nice to have a simple formula for every problem in the world. Especially, for a BBQ party. Unfortunately, it's impossible.

Let's get back to cars. The problem is, they have different manufacturing dates, dozens of options, technical condition, seasonal demand spikes, and god only knows how many more hidden factors. An average Billy can't keep all that data in his head while calculating the price. Me too.

People are dumb and lazy – we need robots to do the maths for them. So, let's go the computational way here. Let's provide the machine some data and ask it to find all hidden patterns related to price.

Aaaand it works. The most exciting thing is that the machine copes with this task much better than a real person does when carefully analyzing all the dependencies in their mind.

That was the birth of machine learning.

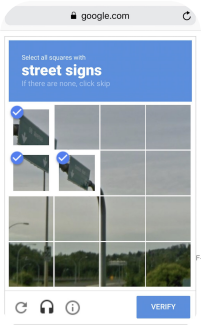
Three components of machine learning

Without all the AI-bullshit, the only goal of machine learning is to predict results based on incoming data. That's it. All ML tasks can be represented this way, or it's not an ML problem from the beginning.

The greater variety in the samples you have, the easier it is to find relevant patterns and predict the result. Therefore, we need three components to teach the machine:

Data Want to detect spam? Get samples of spam messages. Want to forecast stocks? Find the price history. Want to find out user preferences? Parse their activities on Facebook (no, Mark, stop collecting it, enough!). The more diverse the data, the better the result. Tens of thousands of rows is the bare minimum for the desperate ones.

There are two main ways to get the data — manual and automatic. Manually collected data contains far fewer errors but takes more time to collect — that makes it more expensive in general.



Automatic approach is cheaper — you're gathering everything you can find and hope for the best.

Some smart asses like Google use their own customers to label data for them for free. Remember ReCaptcha which forces you to "Select all street signs"? That's exactly what they're doing. Free labour! Nice. In their place, I'd start to show captcha more and more. Oh, wait...

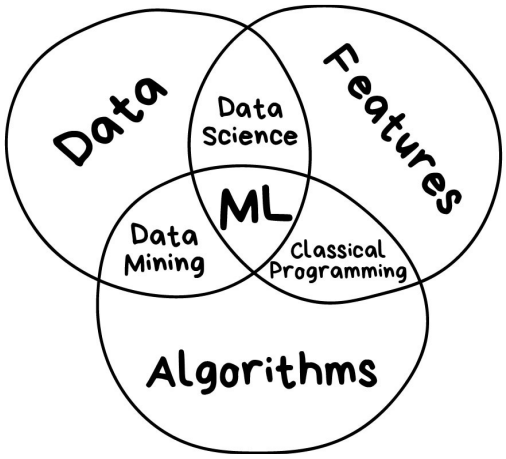
It's extremely tough to collect a good collection of data (usually called a dataset). They are so important that companies may even reveal their

algorithms, but rarely datasets.

Features Also known as parameters or variables. Those could be car mileage, user's gender, stock price, word frequency in the text. In other words, these are the factors for a machine to look at.

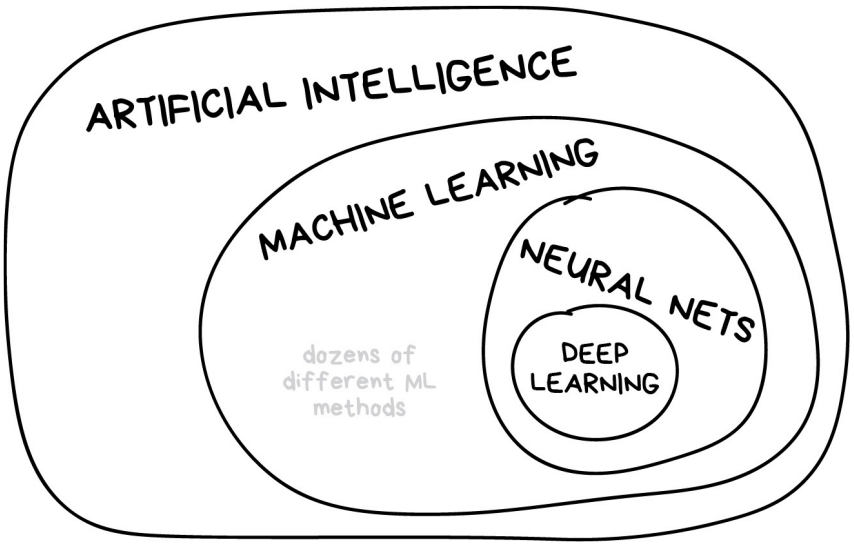
When data stored in tables it's simple — features are column names. But what are they if you have 100 Gb of cat pics? We cannot consider each pixel as a feature. That's why selecting the right features usually takes way longer than all the other ML parts. That's also the main source of errors. Meatbags are always subjective. They choose only features they like or find "more important". Please, avoid being human.

Algorithms Most obvious part. Any problem can be solved differently. The method you choose affects the precision, performance, and size of the final model. There is one important nuance though: if the data is crappy, even the best algorithm won't help. Sometimes it's referred as "garbage in – garbage out". So don't pay too much attention to the percentage of accuracy, try to acquire more data first.



Learning vs Intelligence

Once I saw an article titled *"Will neural networks replace machine learning?"* on some hipster media website. These media guys always call any shitty linear regression at least artificial intelligence, almost SkyNet. Here is a simple picture to show who is who.



Artificial intelligence is the name of a whole knowledge field, similar to biology or chemistry.

Machine Learning is a part of artificial intelligence. An important part, but not the only one.

Neural Networks are one of machine learning types. A popular one, but there are other good guys in the class.

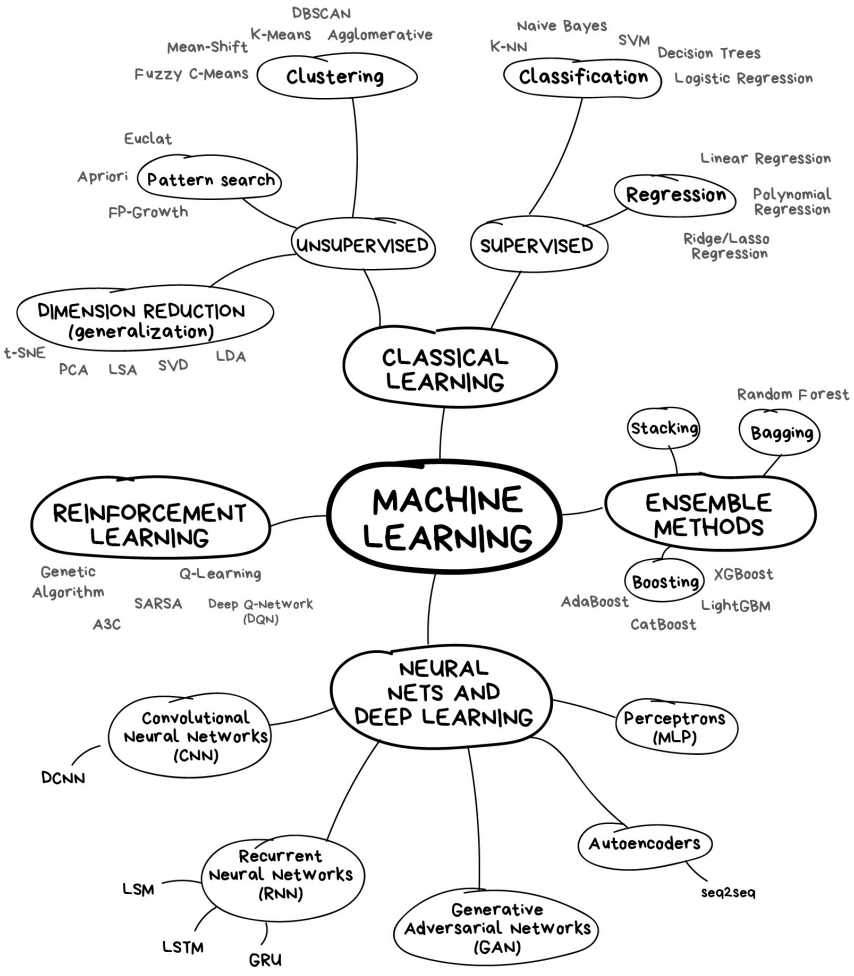
Deep Learning is a modern method of building, training, and using neural networks. Basically, it's a new architecture. Nowadays in practice, no one separates deep learning from the "ordinary networks". We even use the same libraries for them. To not look like a dumbass, it's better just name the type of network and avoid buzzwords.

The general rule is to compare things on the same level. That's why the phrase *"will neural nets replace machine learning"* sounds like *"will the wheels replace cars"*. Dear media, it's compromising your reputation a lot.

Machine can	Machine cannot
Forecast	**Create something new**
Memorize	**Get smart really fast**
Reproduce	**Go beyond their task**
Choose best item	**Kill all humans**

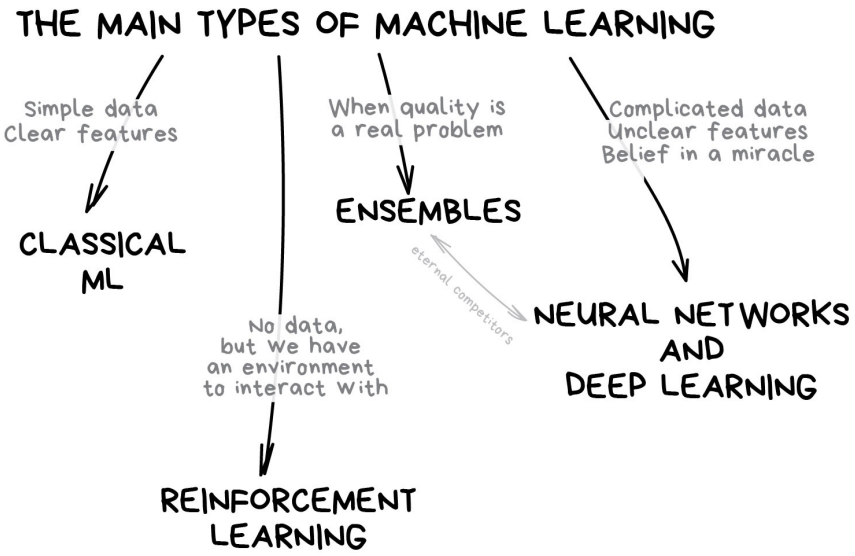
The map of the machine learning world

If you are too lazy for long reads, take a look at the picture below to get some understanding.



Always important to remember — there is never a sole way to solve a problem in the machine learning world. There are always several algorithms that fit, and you have to choose which one fits better. Everything can be solved with a neural network, of course, but who will pay for all these GeForces?

Let's start with a basic overview. Nowadays there are four main directions in machine learning.



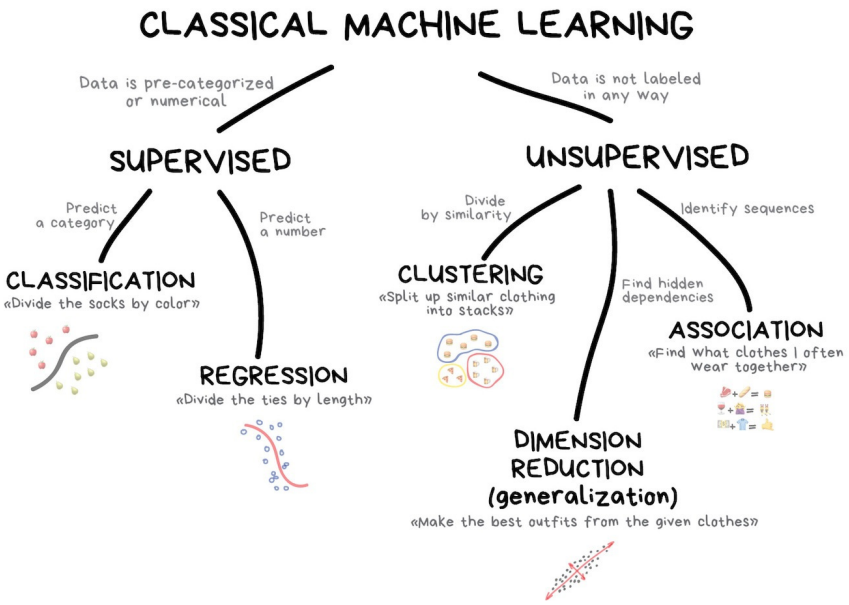
Part 1. Classical Machine Learning

The first methods came from pure statistics in the '50s. They solved formal math tasks — searching for patterns in numbers, evaluating the proximity of data points, and calculating vectors' directions.

Nowadays, half of the Internet is working on these algorithms. When you see a list of articles to "read next" or your bank blocks your card at random gas station in the middle of nowhere, most likely it's the work of one of those little guys.

Big tech companies are huge fans of neural networks. Obviously. For them, 2% accuracy is an additional 2 billion in revenue. But when you are small, it doesn't make sense. I heard stories of the teams spending a year on a new recommendation algorithm for their e-commerce website, before discovering that 99% of traffic came from search engines. Their algorithms were useless. Most users didn't even open the main page.

Despite the popularity, classical approaches are so natural that you could easily explain them to a toddler. They are like basic arithmetic — we use it every day, without even thinking.



1.1 Supervised Learning

Classical machine learning is often divided into two categories – **Supervised** and **Unsupervised Learning**.

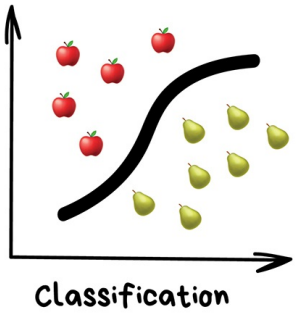
In the first case, the machine has a "supervisor" or a "teacher" who gives the machine all the answers, like whether it's a cat in the picture or a dog. The teacher has already divided (labeled) the data into cats and dogs, and the machine is using these examples to learn. One by one. Dog by cat.

Unsupervised learning means the machine is left on its own with a pile of animal photos and a task to find out who's who. Data is not labeled, there's no teacher, the machine is trying to find any patterns on its own. We'll talk about these methods below.

Clearly, the machine will learn faster with a teacher, so it's more commonly used in real-life tasks. There are two types of such tasks: **classification – an object's category prediction, and regression – prediction of a specific point on a numeric axis**.

Classification

"Splits objects based at one of the attributes known beforehand. Separate socks by based on color, documents based on language, music by genre"



Today used for:
– Spam filtering
– Language detection
– A search of similar documents
– Sentiment analysis
– Recognition of handwritten characters and numbers
– Fraud detection

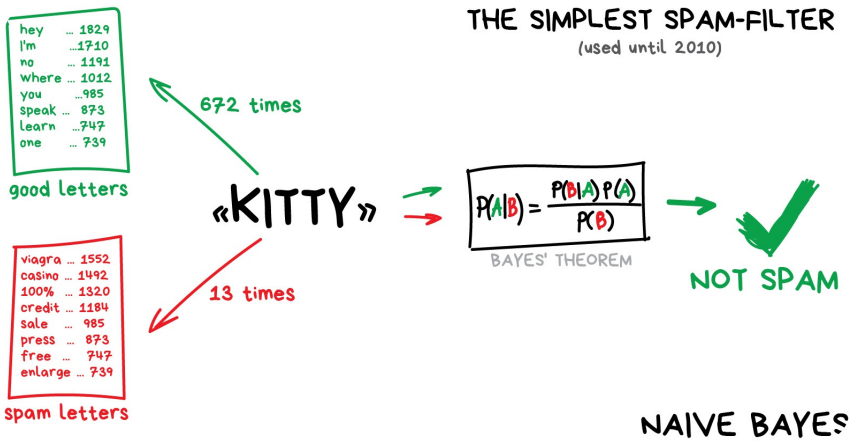
Popular algorithms: Naive Bayes, Decision Tree, Logistic Regression, K-Nearest Neighbours, Support Vector Machine

> From here onward you can comment with additional information for these sections. Feel free to write your examples of tasks. Everything is written here based on my own subjective experience.

Machine learning is about classifying things, mostly. The machine here is like a baby learning to sort toys: here's a robot, here's a car, here's a robo-car... Oh, wait. Error! Error!

In classification, you always need a teacher. The data should be labeled with features so the machine could assign the classes based on them. Everything could be classified — users based on interests (as algorithmic feeds do), articles based on language and topic (that's important for search engines), music based on genre (Spotify playlists), and even your emails.

In spam filtering the Naive Bayes algorithm was widely used. The machine counts the number of "viagra" mentions in spam and normal mail, then it multiplies both probabilities using the Bayes equation, sums the results and yay, we have Machine Learning.



Later, spammers learned how to deal with Bayesian filters by adding lots of "good" words at the end of the email. Ironically, the method was called Bayesian poisoning. Naive Bayes went down in history as the most elegant and first practically useful one, but now other algorithms are used for spam filtering.

Here's another practical example of classification. Let's say you need some money on credit. How will the bank know if you'll pay it back or not? There's no way to know for sure. But the bank has lots of profiles of people who took money before. They have data about age, education, occupation and salary and – most importantly – the fact of paying the money back. Or not.

Using this data, we can teach the machine to find the patterns and get the answer. There's no issue with getting an answer. The issue is that the bank can't blindly trust the machine answer. What if there's a system failure, hacker attack or *a quick fix from a drunk senior*.

To deal with it, we have Decision Trees. All the data automatically divided to yes/no questions. They could sound a bit weird from a human perspective, e.g., *whether the creditor earns more than $128.12?* Though, the machine comes up with such questions to split the data best at each step.

That's how a tree is made. The higher the branch — the broader the question. Any analyst can take it and explain afterward. He may not understand it, but explain easily! (typical analyst)

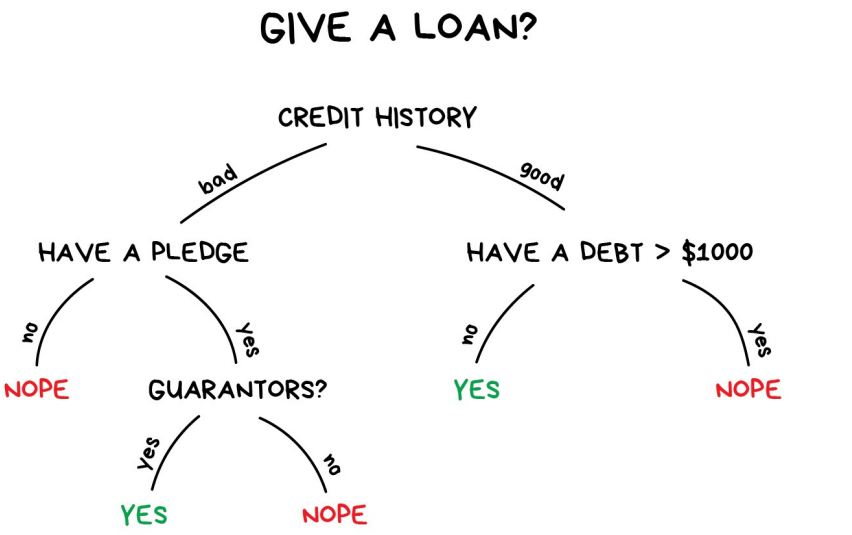
Decision trees are widely used in high responsibility spheres: diagnostics, medicine, and finances.

> The two most popular algorithms for forming the trees are CART and C4.5.

Pure decision trees are rarely used today. However, they often set the basis for large systems, and their ensembles even work better than neural networks. We'll talk about that later.

> When you google something, that's precisely the bunch of dumb trees which are looking for a range of answers for you. Search engines love them because they're

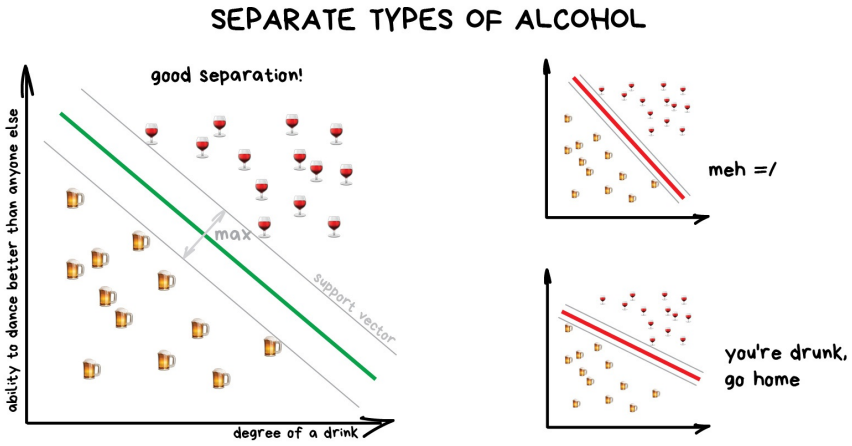
> fast.



DECISION TREE

Support Vector Machines (SVM) is rightfully the most popular method of classical classification. It was used to classify everything in existence: plants by appearance in photos, documents by categories, etc.

The idea behind SVM is simple – it's trying to draw two lines between your data points with the largest margin between them. Look at the picture:



SUPPORT VECTOR MACHINE

There's one very useful side of the classification — anomaly detection. When a feature does not fit any of the classes, we highlight it. Now that's used in medicine — on MRIs, computers highlight all the suspicious areas or deviations of the test. Stock markets use it to detect abnormal behaviour of traders to find the insiders. When teaching the computer the right things, we automatically teach it what things are wrong.

Today, neural networks are more frequently used for classification. Well, that's what they were created for.

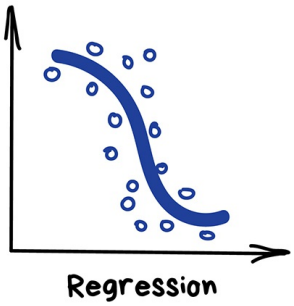
The rule of thumb is the more complex the data, the more complex the algorithm. For text, numbers, and tables, I'd choose the classical approach. The models are smaller there, they learn faster and work more clearly. For pictures, video and all other complicated big data things, I'd definitely look at neural networks.

Just five years ago you could find a face classifier built on SVM. Today it's easier to choose from hundreds of pre-trained networks. Nothing has changed for spam filters, though. They are still written with SVM. And there's no good reason to switch from it anywhere.

Even my website has SVM-based spam detection in comments ¯_(ツ)_/¯

Regression

"Draw a line through these dots. Yep, that's the machine learning"



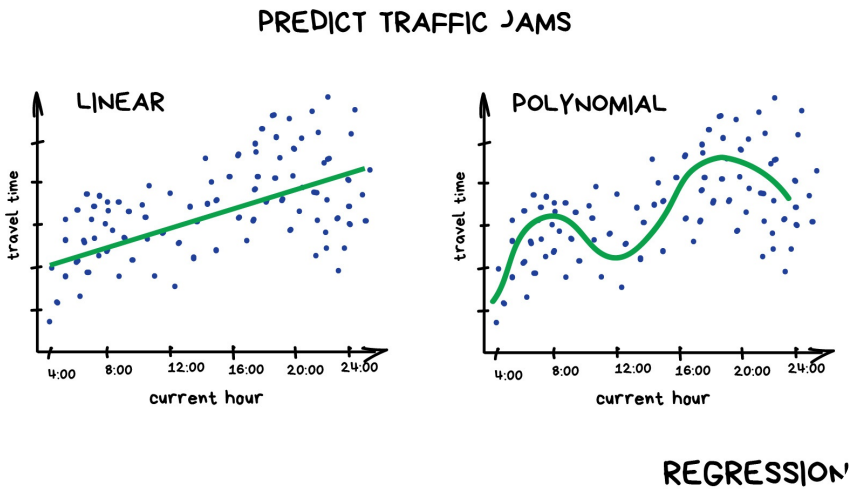
Today this is used for:

- Stock price forecasts
- Demand and sales volume analysis
- Medical diagnosis
- Any number-time correlations

Popular algorithms are Linear and Polynomial regressions.

Regression is basically classification where we forecast a number instead of category. Examples are car price by its mileage, traffic by time of the day, demand volume by growth of the company etc. Regression is perfect when something depends on time.

Everyone who works with finance and analysis loves regression. It's even built-in to Excel. And it's super smooth inside — the machine simply tries to draw a line that indicates average correlation. Though, unlike a person with a pen and a whiteboard, machine does so with mathematical accuracy, calculating the average interval to every dot.



When the line is straight — it's a linear regression, when it's curved – polynomial. These are two major types of regression. The other ones are more exotic. Logistic regression is a black sheep in the flock. Don't let it trick you, as it's a classification method, not regression.

It's okay to mess with regression and classification, though. Many classifiers turn into regression after some tuning. We can not only define the class of the object but memorize how close it is. Here comes a regression.

> If you want to get deeper into this, check these series: Machine Learning for Humans. I really love and recommend it!

1.2 Unsupervised learning

Unsupervised was invented a bit later, in the '90s. It is used less often, but sometimes we simply have no choice.

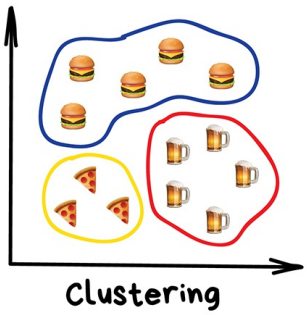
Labeled data is luxury. But what if I want to create, let's say, a bus classifier? Should I manually take photos of million fucking buses on the streets and label each of them? No way, that will take a lifetime, and I still have so many games not played on my Steam account.

There's a little hope for capitalism in this case. Thanks to social stratification, we have millions of cheap workers and services like Mechanical Turk who are ready to complete your task for $0.05. And that's how things usually get done here.

Or you can try to use unsupervised learning. But I can't remember any good practical application for it, though. It's usually useful for exploratory data analysis but not as the main algorithm. Specially trained meatbag with Oxford degree feeds the machine with a ton of garbage and watches it. Are there any clusters? No. Any visible relations? No. Well, continue then. You wanted to work in data science, right?

Clustering

"Divides objects based on unknown features. Machine chooses the best way"



Nowadays used:

- For market segmentation (types of customers, loyalty)
- To merge close points on a map
- For image compression
- To analyze and label new data
- To detect abnormal behavior

Popular algorithms: K-means _clustering, Mean-Shift, DBSCAN

Clustering is a classification with no predefined classes. It’s like dividing socks by color when you don't remember all the colors you have. Clustering algorithm trying to find similar (by some features) objects and merge them in a cluster. Those who have lots of similar features are joined in one class. With some algorithms, you even can specify the exact number of clusters you want.

An excellent example of clustering — markers on web maps. When you're looking for all vegan restaurants around, the clustering engine groups them to blobs with a number. Otherwise, your browser would freeze, trying to draw all three million vegan restaurants in that hipster downtown.

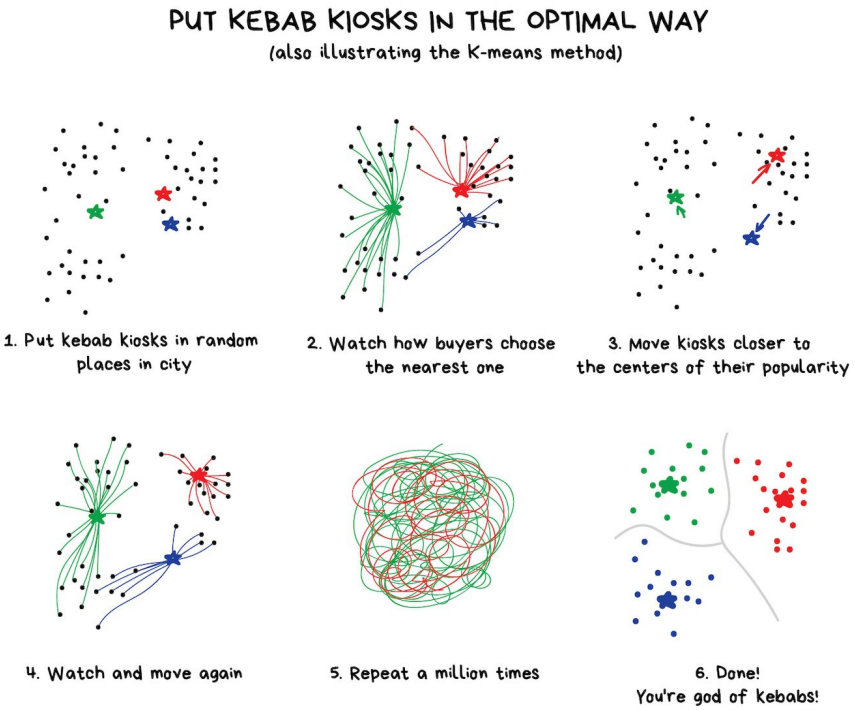
Apple Photos and Google Photos use more complex clustering. They're looking for faces in photos to create albums of your friends. The app doesn't know how many friends you have and how they look, but it's trying to find the common facial features. Typical clustering.

Another popular issue is image compression. When saving the image to PNG you can set the palette, let's say, to 32 colors. It means clustering will find all the "reddish" pixels, calculate the "average red" and set it for all the red pixels. Fewer colors — lower file size — profit!

However, you may have problems with colors like Cyan-like colors. Is it green or blue? Here comes the K-Means algorithm.

It randomly sets 32 color dots in the palette. Now, those are centroids. The remaining points are marked as assigned to the nearest centroid. Thus, we get kind of galaxies around these 32 colors. Then we're moving the centroid to the center of its galaxy and repeat that until centroids stop moving.

All done. Clusters defined, stable, and there are exactly 32 of them. Here is a more real-world explanation:



Searching for the centroids is convenient. Though, in real life clusters not always circles. Let's imagine you're a geologist. And you need to find some similar minerals on the map. In that case, the clusters can be weirdly shaped and even nested. Also, you don't even know how many of them to expect. 10? 100?

K-means does not fit here, but DBSCAN can be helpful. Let's say, our dots are people at the town square. Find any three people standing close to each other and ask them to hold hands. Then, tell them to start grabbing hands of those neighbors they can reach. And so on, and so on until no one else can take anyone's hand. That's our first cluster. Repeat the process until everyone is clustered. Done.

> A nice bonus: a person who has no one to hold hands with — is an anomaly.

It all looks cool in motion:

https://cdn-images-1.medium.com/max/1600/1*tc8UF-honQqUfLC8-ouInQ.gif

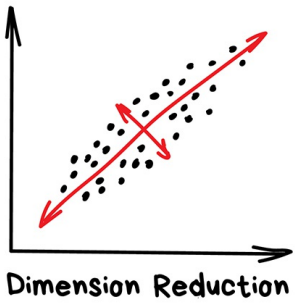
> Interested in clustering? Check out this piece The 5 Clustering Algorithms Data Scientists Need to Know

Just like classification, clustering could be used to detect anomalies. User behaves abnormally after signing up? Let the machine ban him temporarily and create a ticket for the support to check it. Maybe it's a bot. We don't even need to know what "normal behavior" is, we just upload all user actions to our model and let the machine decide if it's a "typical" user or not.

This approach doesn't work that well compared to the classification one, but it never hurts to try.

Dimensionality Reduction (Generalization)

"Assembles specific features into more high-level ones"



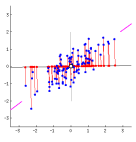
Nowadays is used for:

- Recommender systems (★)
- Beautiful visualizations
- Topic modeling and similar document search
- Fake image analysis
- Risk management

Popular algorithms: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Latent Dirichlet allocation (LDA), Latent Semantic Analysis (LSA, pLSA, GLSA), t-SNE (for visualization)

Previously these methods were used by hardcore data scientists, who had to find "something interesting" in huge piles of numbers. When Excel charts didn't help, they forced machines to do the pattern-finding. That's how they got Dimension Reduction or Feature Learning methods.

It is always more convenient for people to use abstractions, not a bunch of fragmented features. For example, we can merge all dogs with triangle ears, long noses, and big tails to a nice abstraction — "shepherd". Yes, we're losing some information



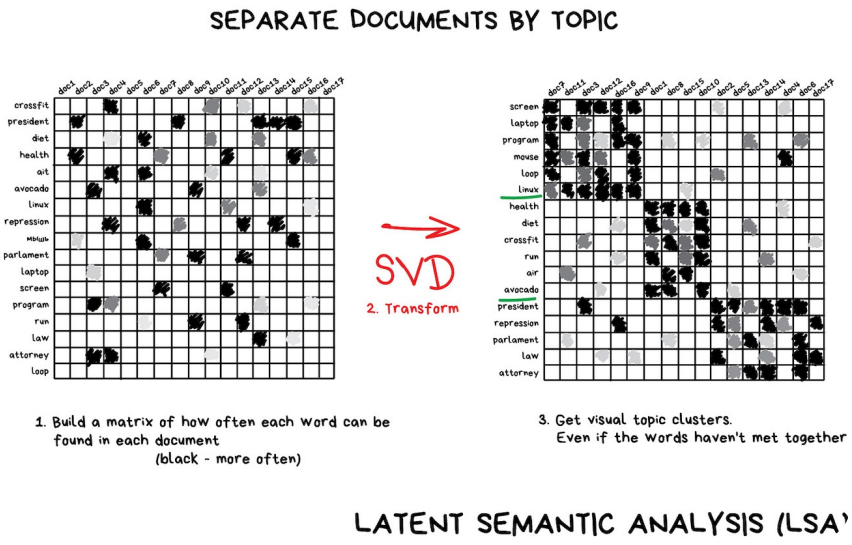
Projecting 2D-data to a line (PCA)

about the specific shepherds, but the
new abstraction is much more useful for naming and explaining purposes. As a bonus, such "abstracted" models learn faster, overfit less and use a lower number of features.

These algorithms became an amazing tool for Topic Modeling. We can abstract from specific words to their meanings. This is what Latent semantic analysis (LSA) does. It is based on how frequently you see the word on the exact topic. Like, there are more tech terms in tech articles, for sure. The names of politicians are mostly found in political news, etc.

Yes, we can just make clusters from all the words at the articles, but we will lose all the important connections (for example the same meaning of *battery* and *accumulator* in different documents). LSA will handle it properly, that's why its called "latent semantic".

So we need to connect the words and documents into one feature to keep these latent connections — it turns out that Singular decomposition (SVD) nails this task, revealing useful topic clusters from seen-together words.



Recommender Systems and Collaborative Filtering is another super-popular use of the dimensionality reduction method. Seems like if you use it to abstract user ratings, you get a great system to recommend movies, music, games and whatever you want.

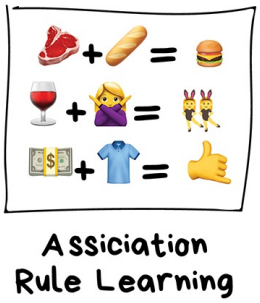
> Here I can recommend my favorite book "Programming Collective Intelligence". It was my bedside book while studying at university!

It's barely possible to fully understand this machine abstraction, but it's possible to see some correlations on a closer look. Some of them correlate with user's age — kids play Minecraft and watch cartoons more; others correlate with movie genre or user hobbies.

Machines get these high-level concepts even without understanding them, based only on knowledge of user ratings. Nicely done, Mr.Computer. Now we can write a thesis on why bearded lumberjacks love My Little Pony.

Association rule learning

"Look for patterns in the orders' stream"



Nowadays is used:

- To forecast sales and discounts
- To analyze goods bought together
- To place the products on the shelves
- To analyze web surfing patterns

Popular algorithms: Apriori, Euclat, FP-growth

This includes all the methods to analyze shopping carts, automate marketing strategy, and other event-related tasks. When you have a sequence of something and want to find patterns in it — try these thingys.

Say, a customer takes a six-pack of beers and goes to the checkout. Should we place peanuts on the way? How often do people buy them together? Yes, it probably works for beer and peanuts, but what other sequences can we predict? Can a small change in the arrangement of goods lead to a significant increase in profits?

Same goes for e-commerce. The task is even more interesting there — what is the customer going to buy next time?

No idea why rule-learning seems to be the least elaborated upon category of machine learning. Classical methods are based on a head-on look through all

the bought goods using trees or sets. Algorithms can only search for patterns, but cannot generalize or reproduce those on new examples.

In the real world, every big retailer builds their own proprietary solution, so nooo revolutions here for you. The highest level of tech here — recommender systems. Though, I may be not aware of a breakthrough in the area. Let me know in the comments if you have something to share.

Part 2. Reinforcement Learning

"Throw a robot into a maze and let it find an exit"

Nowadays used for:

- Self-driving cars
- Robot vacuums
- Games
- Automating trading
- Enterprise resource management

Popular algorithms: Q-Learning, SARSA, DQN, A3C, Genetic algorithm