Get Started With AI

Artificial Intelligence

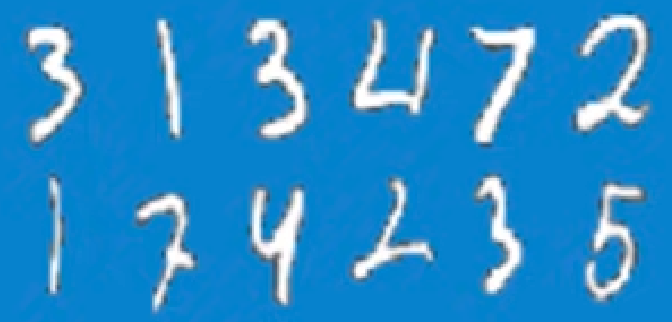
Any technique that enables computers to mimic human behavior

Machine Learning

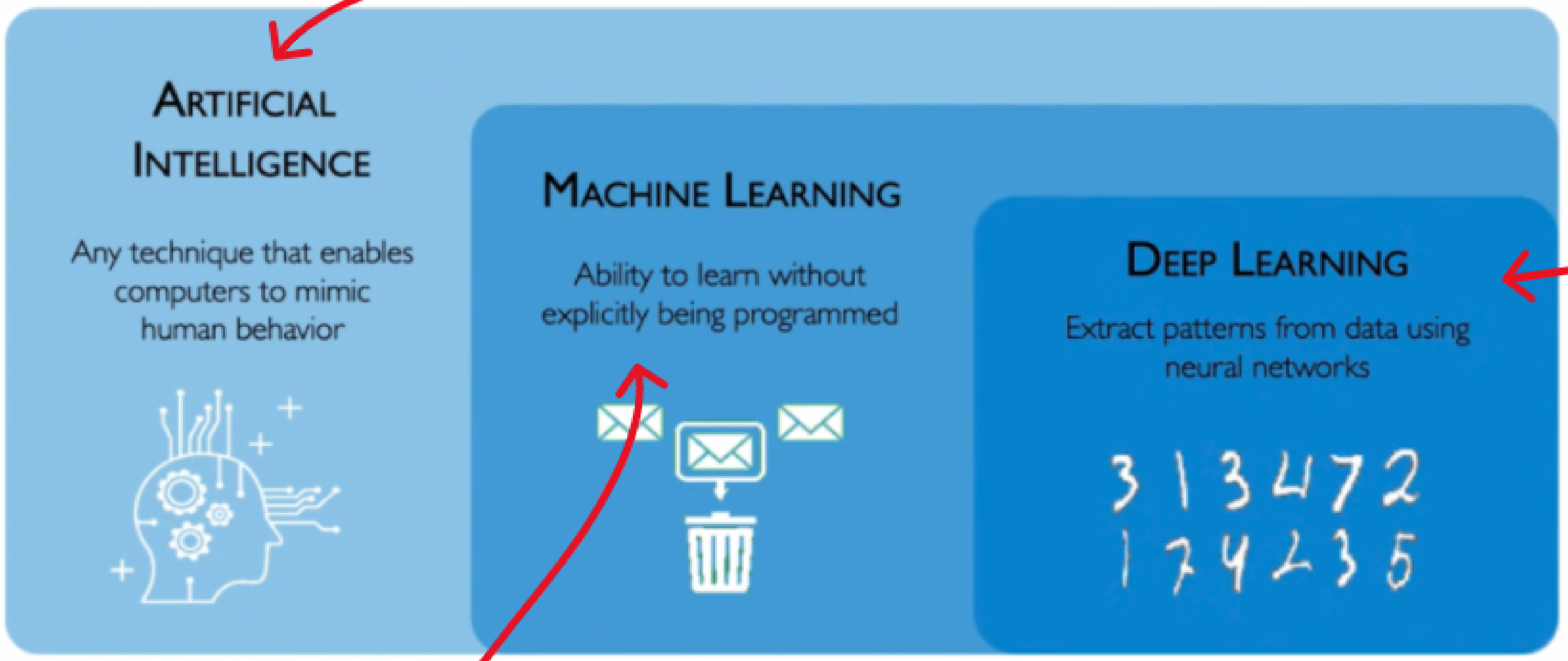
Ability to learn without explicitly being programmed

Deep Learning

Extract patterns from data using neural networks



heurystyki przeszukań, systemy rozmyte, prolog, perceptron, systemy eksperckie, algorytmy genetyczne



złożone, głebokie sieci neuronowe

regresja liniowa, regresja logistyczna, drzewa decyzyjne, random forest, SVM, bagging, boosting, sieci neuronowe

Mat Velloso 🇺🇦
@matvelloso

Difference between machine learning and AI:

If it is written in Python, it's probably machine learning

If it is written in PowerPoint, it's probably AI

Machine Learning vs Data Science

ML

- designing complex, intelligent self-learning systems
- understanding the problem
- figuring out how the to make something "learn"

- the essence is "experimentation"

- you're like an engineer/scientist
- you create new stuff

DS

- extracting useful information from data
- presenting results in easy-to-consume format, often to non-technical people at the company

- the essence is "data analysis"

- you're like the detective
- you only discover insights

ML is an experimental science!

hypothesis -> testing

You try to figure out
which kind of "learning design" would work best for the given problem,
and conduct experiments to verify your hypothesis.

This way you improve your system
repeatedly until you start getting useful results.

Naive assumption of ML project lifecycle

Collect Data

Find large sources of raw data and send it off to MTurk to get it annotated

Create Model

Design a model in TensorFlow/PyTorch for your task

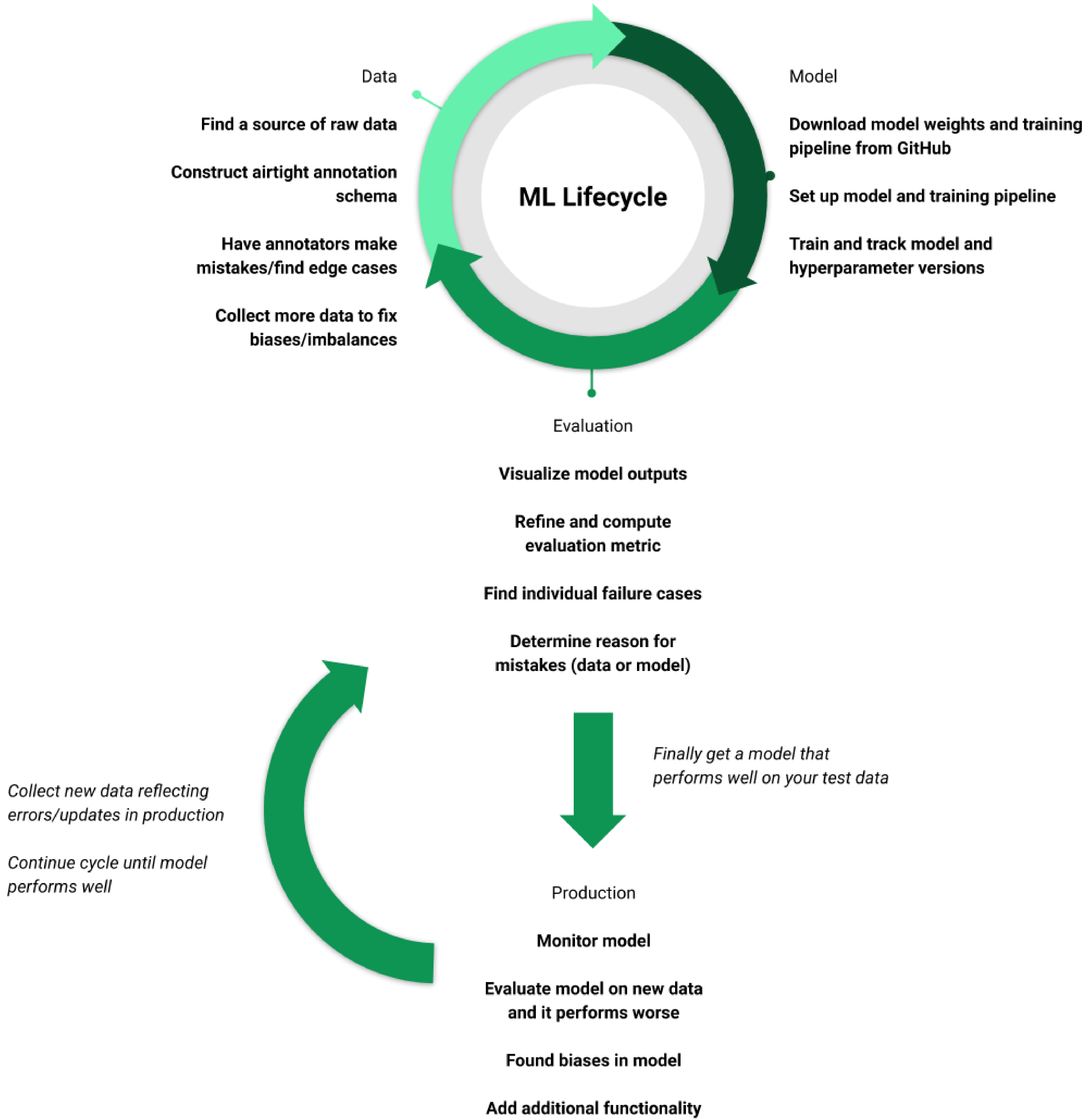
Train Model

Train your model with a few different hyperparameter combos to maximize performance

Deploy

Deploy your best model to production and have it perform well on production data streams indefinitely

More realistic view of ML project lifecycle



Typical Career Paths

What do you enjoy the most?

My subjective distinction of different AI and data-related roles

- Data Engineer / Big Data Engineer
- Data Analyst
- Data Scientist
- Machine Learning Engineer / Specialist
- Deep Learning Engineer
- MLOps
- Research Scientist

Data Engineer / Big Data Engineer

- Makes sure that the infrastructure to collect, transform/process and store data is well-built
- Python, SQL, Spark, Hadoop, DVC, Databases

Data Analyst

- Similar to Data Scientist
- Works closely with business teams
- Excel, Tableau, Power BI

Data Scientist

- Very overused term
- Analyzing, processing, and interpreting data
- Often build simple ML models, like linear regression
- Statistics, SQL, Python/R, Machine Learning

Machine Learning Engineer / Specialist

- Kinda "jack of all trades" who builds ML solutions
- Their work overlaps with that of data engineers, but they focus largely on applying ML models and building the related infrastructure and deployment
- Usually don't need the deep knowledge of deep learning, statistics and math
- Most of the time, they don't come up with any new ideas
- Often uses typical ML algorithms: linear regression, logistic regression, decision trees, bagging, boosting, clusterization, PCA, all kinds of neural networks
- Python, Docker/Kubernetes, Keras/Pytorch/Tensorflow, Cloud, DVC, FastAPI,

Deep Learning Engineer

- Implementing SotA (State of the Art) models from papers
- Model optimization, sophisticated testing, finding bottlenecks
- Sometimes proposing modifications or even coming up with completely novel ideas
- Requires very good knowledge of some deep learning framework like pytorch/tensorflow/jax
- Requires good enough math to read papers fluently
- Python, MLOps, optionally C/C++

Research Scientist

- Companies that focus on bleeding edge technologies often have this role
- Comes up with novel ideas in AI/ML
- Writes papers
- Usually has some specialized knowledge in NLP, computer vision, speech, robotics, etc.
- Requires decent knowledge of some deep learning framework, usually pytorch or jax
- Almost impossible to get in unless you have a PhD
- Few job positions and a lot of people applicants due to inflation of ML PhD graduates from top universities
- At FAANG you would need to have 4+ papers at top ML conferences to even be invited to the interview, and most jobs get distributed through networking either way

MLOps

- Automating and optimizing testing of ML models
- Configuring, maintaining, and building deployment tools
- Leads best-practices in company, for building, testing, and releasing software
- Identifying infrastructure needs and translating them into action
- CI/CD pipelines, cloud solutions, Docker/Kubernetes, DevOps, data drift tracking, strong Linux, API skills

Get started: theory

What math background do you need?

understanding derivatives

+

simple matrix multiplication

and you're good to go

ML Theory Resources

3blue1brown videos

The most intuitive math explanations you will ever find!

- https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

Machine Learning from Stanford by Andrew Ng

Consists of 11 weeks

- Definitely worth doing the first 5 weeks
- Easy to consume
- Completely free
- https://www.coursera.org/learn/machine-learning

Deep Learning Specialisation by Andrew Ng

Consists of 5 very easy-to-follow courses

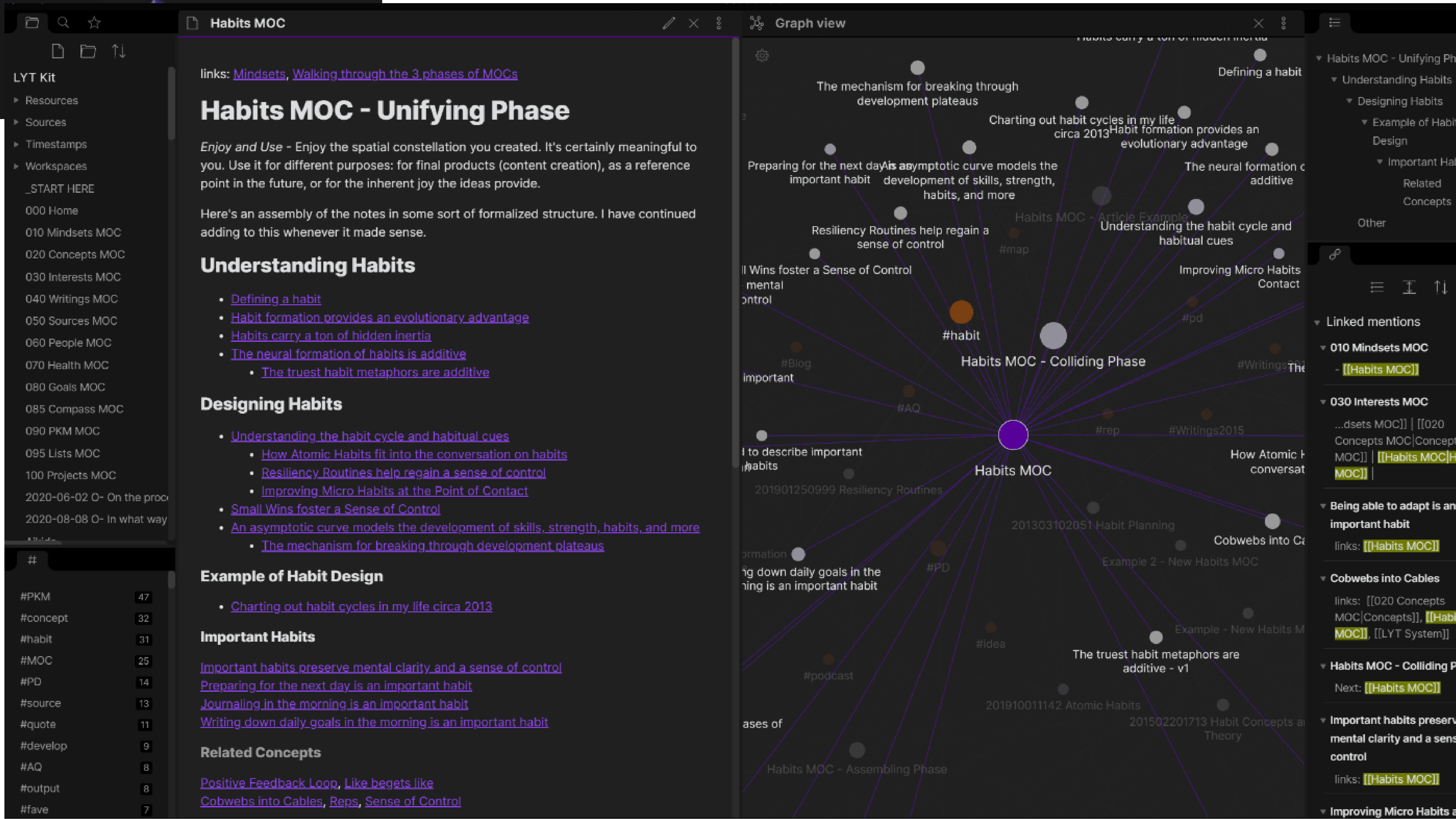
- Very thorough
- Optimization insights, CNNs, RNNs
- https://www.coursera.org/specializations/deep-learning

Remember to take good notes!

Obsidian

A knowledge base that works on local Markdown files.

obsdmd



https://obsidian.md/

Get started: practice

Programming Language?

Just learn Python.

Don't bother with other languages like R if you don't know Python very well.

We have some project propositions on GDSC website
to get you started with Python:
https://gdsc-lodz-university-of-technology.github.io/gdsc-website-template/

ML Framework?

High-level framework

- Keras
- Pytorch Lightning
- FastAI

Low-level framework

- Tensorflow
- Pytorch
- Jax

My recommendations for beginners:

1. Start with Pytorch
2. After that, try Pytorch Lightning (it requires very little additional learning)

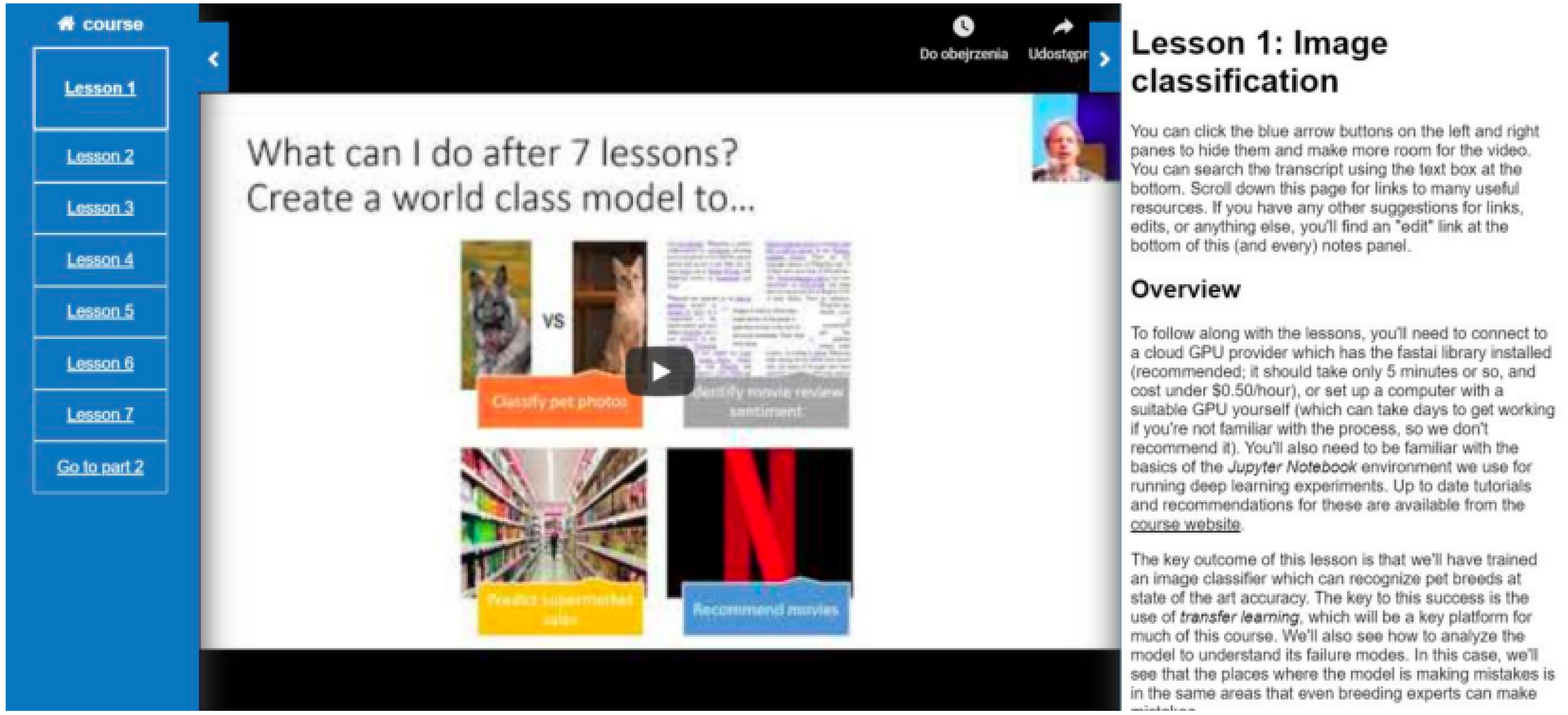
General resource for learning practical skills

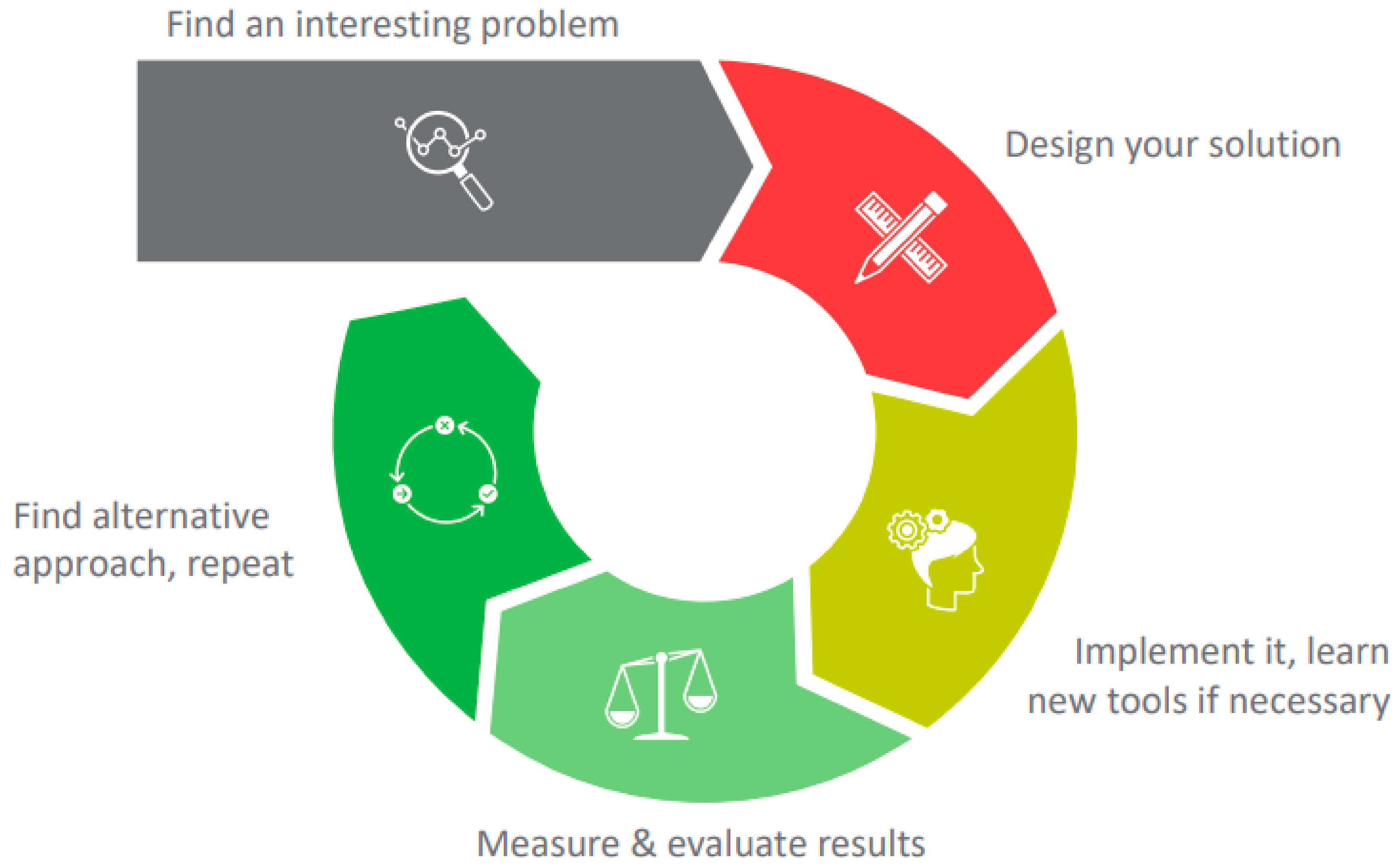
Fast.ai – hack your way into machine learning

website:
https://course.fast.ai

Assumes that you already know how to program (and know basics of Python as well)

- Focused on practical aspects of machine learning
- Most courses focus on deep learning (neural networks)
- You won't learn much theory there
- Plenty of practical tips

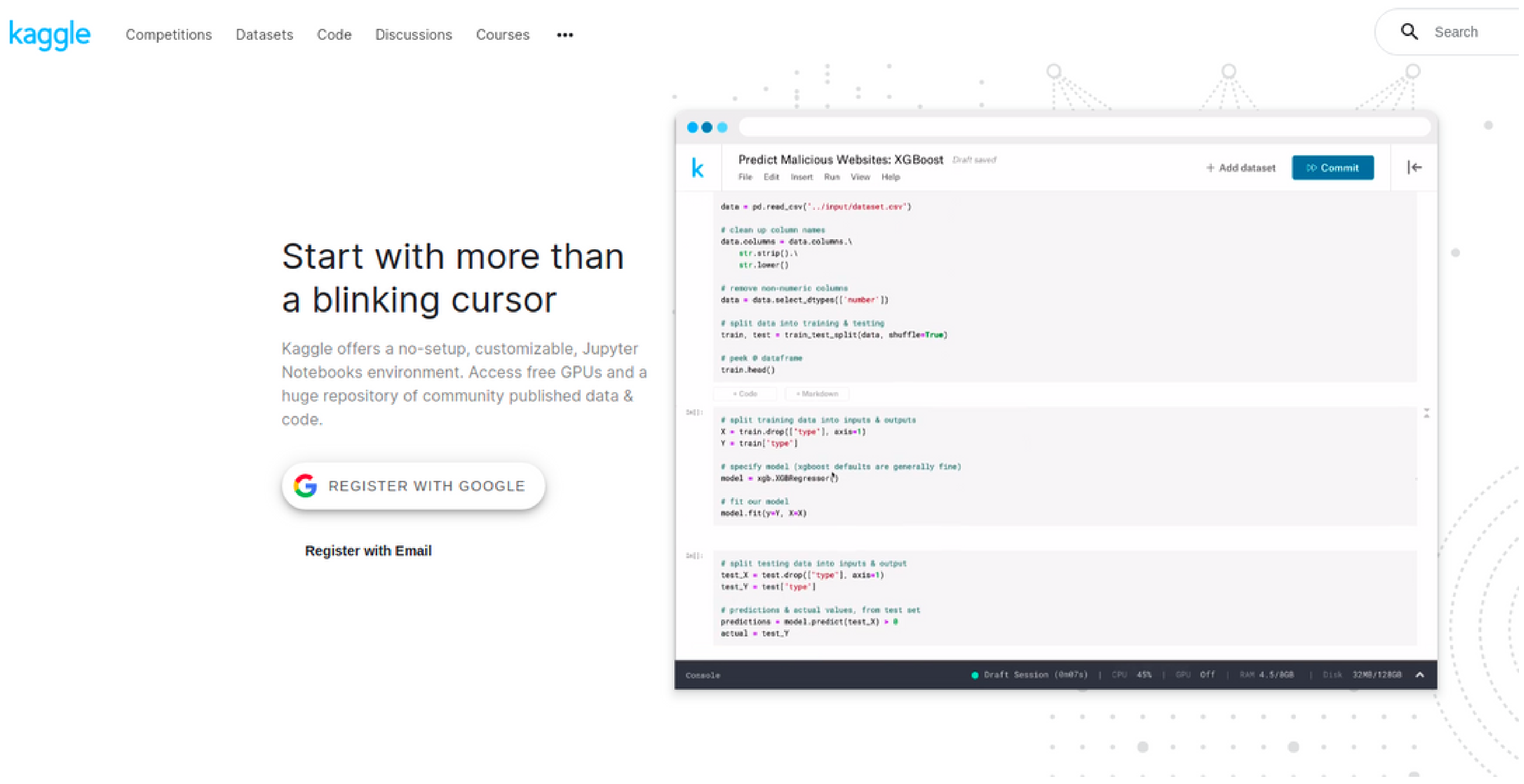




Find an interesting problem
Design your solution
Implement it, learn new tools if necessary
Measure & evaluate results
Find alternative approach, repeat

Source of projects #1

Kaggle



Taking part in data science competitions is often the best way to learn & test skills.

- Competitions – solve problems submitted by real companies, win prizes (or at least get experience)
- Datasets & notebooks
  - Explore solutions of other people shared as public notebooks
  - Use hosted Jupyter Notebook with easy access to datasets & a free GPU
  - Edit other people's notebooks to improve or play with their solutions
- Courses – for a start this is probably where you want to go

Source of projects #2

University: subjects, student societies & research groups

Enroll in a machine learning subject or reach out to fellow students & researchers.

Machine learning subjects at University

- Quality depends heavily on your university
- Can be a good way of getting into ML & AI

Students societies

- Usually ran by people interested in machine learning and willing to share their skills
- Good places to look for competition teammates, project ideas and new knowledge sources

Machine learning research groups

- Researchers often could use free help, and will teach you necessary skills to make it possible
- This is excellent way of getting an interesting topic for your BSc, MSc or PhD

PŁ Opportunities

SIIUM: https://siium.iis.p.lodz.pl

Research Group: PLantation
http://ics.p.lodz.pl/~tomczyk/

Sekcja SI: https://discord.gg/UuAmyXY

AI + other interests

What's hot in AI right now?

https://www.stateof.ai

https://digitalpoland.org/en/publications/download?id=df54480f-f689-4d5d-8629-31277b864611

Key themes in the 2021 Report include:

- **AI is stepping up in more concrete ways**, including being applied to mission critical infrastructure like national electric grids and automated supermarket warehousing optimization during pandemics.
- **AI-first approaches have taken biology by storm** with faster simulations of humans' cellular machinery (proteins and RNA). This has the potential to transform drug discovery and healthcare.
- **Transformers have emerged as a general purpose architecture for machine learning**, beating the state of the art in many domains including NLP, computer vision, and even protein structure prediction.
- Investors have taken notice, with **record funding this year into AI startups**, and two first ever IPOs for AI-first drug discovery companies, as well as blockbuster IPOs for data infrastructure and cybersecurity companies that help enterprises retool for the AI-first era.
- **The under-resourced AI-alignment efforts from key organisations** who are advancing the overall field of AI, as well as concerns about datasets used to train AI models and bias in model evaluation benchmarks, raises important questions about how best to chart the progress of AI systems with rapidly advancing capabilities.
- **AI is now an actual arms race rather than a figurative one.** AI researchers have traditionally seen the AI arms race as a figurative one -- simulated dogfights between competing AI systems carried out in labs -- but that is changing with reports of recent use of autonomous weapons by various militaries.
- Within the US-China rivalry, **China's ascension in research quality and talent training is notable**, with Chinese institutions now beating the most prominent Western ones. The world's dependence on Taiwan's semiconductor industry, which makes AI chips for global tech giants, is a central point of geopolitical tension.
- As with other aspects of the so-called "splinternet", there is an **emergence and nationalisation of large language models**.

Where should I look for a job?

Big Tech (FAANG/MANGA)

- by definiton: highly selective technology companies like Google, Nvidia, Amazon, Apple, Uber, Pinterest, etc.
- give you great credentials for future, even if you don't want to work at google, it's often worth to be "ex-google"
- most skillful people in the industry
- hard and often long recruitment processes
- usually you won't get in if you're just good, you need to be excellent
- still very low presence in Poland but it's changing

Startups

- often can be very selective since startups can't afford for bad hires
- very intense
- often hire people who aren't very specialized, but instead can be "jack of all trades"
- give you shares, but 80% of startups fail either way so you probably won't get rich

Average companies and AI software houses

- decent chance for being accepted
- examples: every company in Łódź, e.g. Accenture, Digica, Ericsson, Comarch, EY, etc.
- they often look for people
- need to be careful to not join a place where the team just lack skills
- generally AI software houses are very good experiences for beginners!

	Big companies	Startups
Pros	• Brand name recognition + good for your resume. • Stability. Google's stock is unlikely to be useless in a few years. • You'll probably have a well-defined role and and won't have to work as much as at a startup. • There'll be a well-defined procedure to go up the ranks -- you can do a reasonable amount of work and you're set. • (Hopefully) a great code review process to help you become a much better software engineer. • Plenty of smart people to work with. • Perks: free food, free massage, free car rentals, free Lyft codes, generous 401(k) matching, etc.	• You can contribute significantly to the product. • You can get to know everyone and the CEO might even listen to you. • You can do multiple things at once -- making your job interesting. • You can grow with the company and get promoted much faster than at a big company. • You can learn A LOT. • There's a chance you'll make a lot of money.
Cons	• Easy to settle into complacency. • You'll probably be just another cog in the system -- your effort or lack of it won't change anything. • You'll probably just work with a tiny piece of code -- your work becomes boring fast. • Unlikely that the management will ask for your opinion about where the company is headed.	• Nobody's heard of the startup.. • The company will probably fail and your 0.5% equity will become useless. • You might have to work a lot and work on everything, even things that you hate. • Terrible code review process. After a year, your code might still be crap. • The startup probably doesn't have as many world-class engineers as Google for you to work with. • Less perks.

https://huyenchip.com/2018/10/08/career-advice-recent-cs-graduates.html

https://huyenchip.com/2021/02/27/why-not-join-a-startup.html

How recruitment processes look like

Some of my experiences

- Comarch
  - 1 written test
- Samsung
  - 1 online test which lasted an hour
  - 1 interview which lasted an hour
- Nvidia
  - initiated by a recruiter through linkd
  - 1 initial talk about company
  - 1 code interview
  - 1 techniqual interview about deep learning in general

At decent companies the interview won't be a "quiz" - it should be a conversation where each question comes from the previous one.

Good companies will focus on testing you about topics you feel very strong about, because ML is too wide as a field and no one knows everything.

Interview1 (krótka wstępna rozmowa)

- czemu jesteś zainteresowany
- O co chodzi w adversarial machine learning?
- Jak zbudowałbyś model który z jednego grafu robi 2 grafy (przewidywanie efektu reakcji chemicznej: substract -> product)

Interview2 (projekty)

- jak transformer może zmierzyć podobieństwo między tekstami, skoro ta architektura po prostu przewiduje sekwencje
- jak zmierzyć jak bardzo dobry jest taki graf semantycznych połączeń z naszej aplikacji
- dużo otwartych pytań typu: jakie widzisz potencjalnie trudności w chemiinformatyce
- skoro transformery są takie dobre to dlaczego nie użyuwać ich zamiast sieci grafowych albo na odwrót

Interview3 (ML teoria)

- wymienić kilka metod supervised/unsupervised
- jakie są możliwe lossy w regresji a jakie do klasyfikacji
- dlaczego SGD się nazywa "stochastic"?
- exploding i vanishing gradient, kieedy, dlaczego, jak przeciwdziałać
- opisać jak optymalizuje się parametry w sieci
- jak inicjalizować parametry modelu, czemu jak zainicjalizujemy takimi samymi wartościami to się nie uczy
- saddle points, jak wpływa na ten problem optimizer
- jak działa backpropagacja
- jak działa momentum
- KL-divergence czym jest do czego się stosuje, po co
- kiedy gradient funkcji aktywacji jest mocny/słaby, jak to się ma do tanh/sigmoida/relu

Interview4 (rozmowa o mnie):

- jak radzisz sobie z niejasnością/niepewnością?
- jakiś przypadek gdzie projekt się nie powiódł
- jak ci się podobało gdzieś tam na stażu?
- gdzie widzisz się za 5 lat?
- czy te problemy są dla ciebiee ciekawe i czemu chcesz nad tym pracować

Interview5:

- jak zmienia się bias i variance w zależności od k w k-fold cross validation
- kiedy używać innego thresholda w f1 niż 0.5
- kiedy rocauc lepszy niż f1, kiedy powinno się go używać, jakie ma zalety
- czasem warto raportować f1 oraz rocauc i porównać zależność między nimi
- Dlaczego model based miało by być lepsze niż model free w generowaniu grafu które zaproponowałem,
- Jaka jest statystyczna inerpretacja biasu i wariancji

Interview 1 (wstępna rozmowa o firmie)

Interview 2 (code interview)

- Count Islands

  https://leetcode.com/problems/count-sub-islands/

Interview 3

- opisz jak działa prosta sieć neuronowa, tak na poziomie szkoły średniej
- napisać wzór na binarną cross-entropie, i jak się on ma do wieloklasowej cross-entropii?
- dlaczego stosujemy logarytm w crossentropii? co nam daje wyciąganie logarytmu z prawdopodobieństwa?
- jak ma się KL-Divergence do cross-entropii? jak przekształcić jedno w drugie?
- wymienic jakąś inną metodę liczenia różnicy między dystrybucjami prawdopodobieństwa niż cross-entropia i KL-Divergence (można np. metodą Hellingera)
- jak to możliwe że sieci się czegoś uczą skoro topologia funkcji kosztu ma wiele lokalnych minimów? (lr, optimizery, stochastyczność algorytmu gradient descent, lokalne minima w wysoko-wymariowych przestrzeniach są mniej prawdopodobne)
- opisać dowolny optimizer (opisałem momentum)
- powiązanie między algorytmem momentum a optymalizowaniem sieci drugą pochodną
- dlaczego nie trenujemy sieci drugą pochodną? (generalnie jest to analitycznie lepsze i potężniejsze, daje lepsze efekty, ale zbyt złożone obliczeniowo w praktyce)
- czym są siodełka (saddle points)? jak wpływają na uczenie?
- jakie warunki muszą być spełnione żeby funkcja kosztu mogła być użyta (musi być różniczkowalna i najlepiej w miare wypukła)
- czy wszystkie operacje w sieciach są różniczkowalne i co się robi w razie jak nie są (relu nie jest w zerze więc przyjmujemy umowną wartość gradientu)
- jak wyliczamy gradient dla max-poolingu? (po prostu ignorujemy wartości neuronów które nie zostały wybrane przez max-pooling, i to wbrew pozorom nadal działa dobrze)**
- jak działa autograd w pytorchu a jak zwykłe Variable** (Variables są już deprecated, autograd automatycznie wspiera optymalizowanie tensorów za pomocą flagi requires_grad=True)
- jak działa backpropagacja, szczegółowo wyjaśnić
- co dokładnie jest trzymane w pamięci podczas forward i backpropagacji (parametry i wartości neuronów dla każdego datapointa, a potem gradienty, wartości neuronów mogą być na bieżąco usuwane w miarę backpropagacji)
- jaka jest złożoność pamięciowa prostej sieci neuronowej podczas forward i backpropagacji (batch_size*num_neurons*num_gradients + num_weights)
- jak ma się do tej złożoności gradient checkpointing
- dostałem kod w pytorchu, miałem 2 minuty na zapoznanie się, a potem musiałem po angielsku wytłumaczyć krok po kroku co ten model robi i narysować na wirtualnym whiteboardzie

Some interview and resume advices

You need to be able to talk a lot about your projects.
Use the STAR teqnique.

- https://www.indeed.com/career-advice/interviewing/how-to-use-the-star-interview-response-technique

Formulate every paragraph in your CV as a : problem->solution->impact.
Be very concise and concrete.
Avoid vague statements like "experienced in...".

Highlight technical accomplishments, using metrics to display your impact (like money/time saved).

Technical interviews are a pain. Prepare for them a least a month in advance.

Best resources

- How to prepare for the interview and great ML interview questions
  - https://huyenchip.com/ml-interviews-book/
- 300 page book with advanced ML questions and answers
  - https://github.com/BoltzmannEntropy/interviews.ai

Some generic advice

- Learn to use LaTeX early.
- Complete some online course once in a couple of months.
- Find an internship as early as possible (best would be first or second year).
- Find people who are at the same level as you are, have strong ambitions, and are willing to collaborate.
- Attend hackathons. Especially AI-focused ones.
- Attend some ML conference at least once. Recommended: MLinPL
- Don't overload yourself with too many insignificant tasks/projects/courses. Focus on 1 very impactful project at a time, where you will be able to present later that you made a huge impact.
- Time management is everything!
- Having strong GitHub is very important. Invest your time to build some decent GitHub repos.
- Having open-source background is very beneficial. Develop experience by making PRs to open-source ML libraries, or writing your own toy libraries. You can even make PRs that simply improve comments in the code or make error messages more clear - devs will be very thankful.
- Your GPA doesn't really matter since you can just not put in on your CV and no one will care.
- Education is only important at the beginning. You don't need a master's degree, but it can help you find a junior-level job. You probably need a PhD if you want to do any kind of research commercially.
- Stop comparing yourself to other people. No matter how good you are, there will always be someone who’s better than you at something. Instead, compare you of today with you of yesterday.
- Know what you’re optimizing for: money, new experiences, prestige, personal growth, or something else?