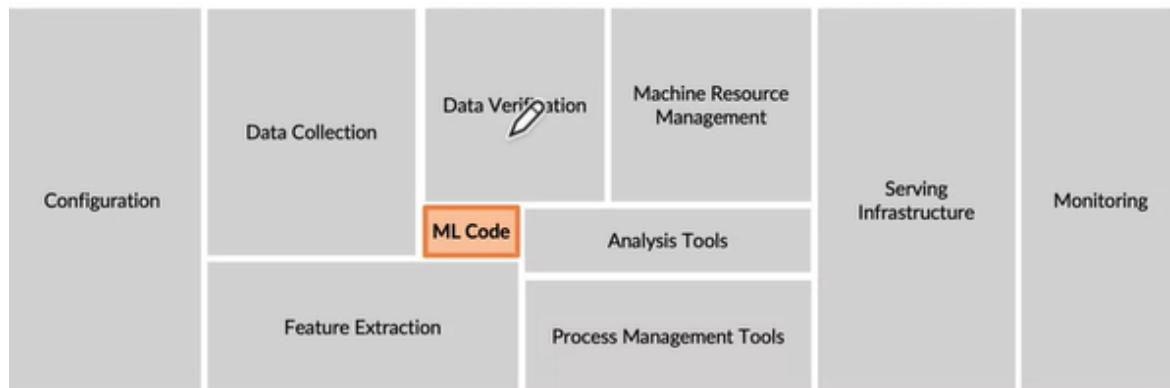


MLOps -> quase que um protocolo para criar um modelo de machine learning que seja usavel em produção, seja útil no mundo real.

The requirements surrounding ML infrastructure



1. Semana 1 - Pipeline de um sistema de ML e deployment

Steps of an ML project

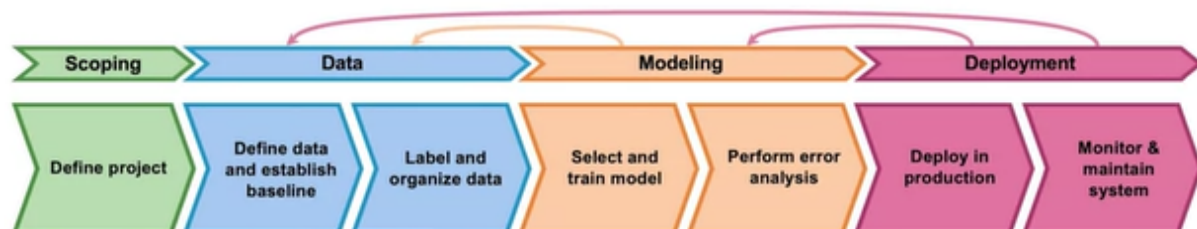
Scoping -> define project, o que queremos descobrir

Data-> definir a data e organizar a label e a data

Modelling -> Selecionar e treinar um modelo, analise de performance e erro

Deployment: -> deploy in production e monitor/manter o sistema

The ML project lifecycle



O curso começa com deployment, depois modelling, depois data e depois scoping

Deployment:

O que torna o deployment difícil? Dois problemas, os estatísticos/modelo e os de problema de software

Tipos de deployment:

- 1 - Novo produto
- 2 - Automate/assist uma tarefa manual
- 3 - Substituir um antigo sistema de ML

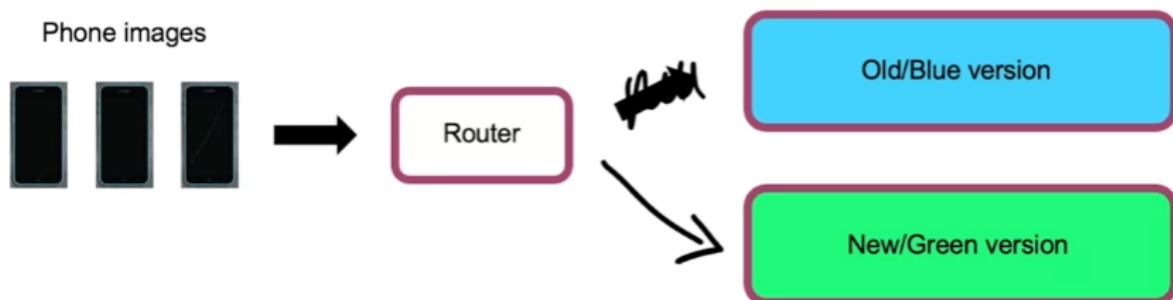
Em todos deve se preocupar com incrementos graduais e possíveis rollback caso algo não esperado aconteça

Shadow mode deployment: inspecionar se o que o algoritmo de aprendizado esta fazendo é certo

Canaray deployment: padrão de deployment em que se coloca 5% por vez e vai aumentando gradativamente. permite detectar erros cedo no desenvolvimento

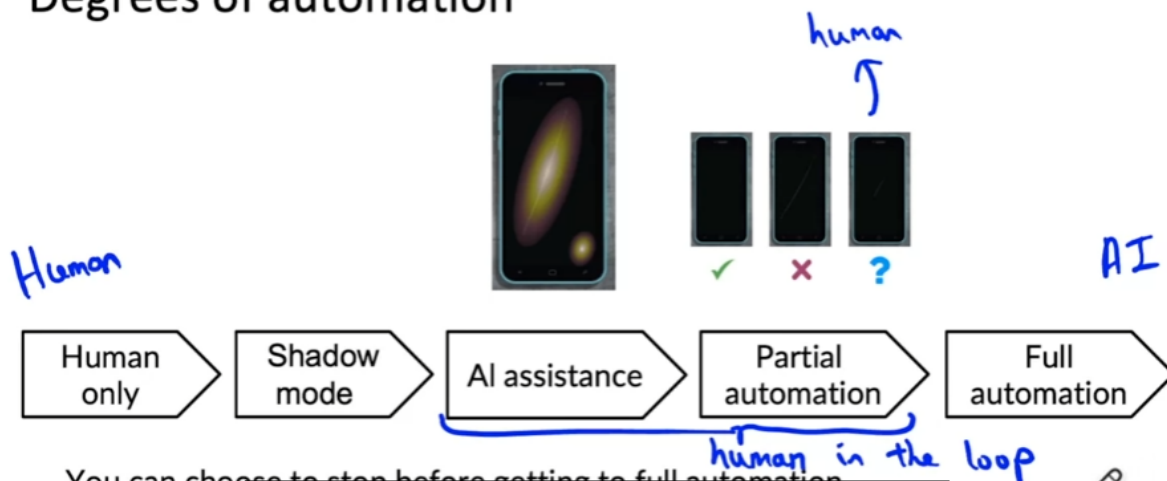
Blue green deployment: quando se muda a versão de azul para verde, simplesmente se muda pra onde os dados estão sendo roteador, uma vantagem é um rollback fácil

Blue green deployment



Graus de automação: É importante a inspeção humana

Degrees of automation



Monitoramento: Prática mais comum é usar uma dashboard, como monitor o server, os inputs/outputs

Algumas práticas interessantes:

- > Brainstorm coisas que podem dar errado
- > Brainstorm algumas estatísticas métricas
- > Implementar várias métricas e descartar com o tempo as que não são tão uteis

Examples of metrics to track

Software metrics:

Memory, compute, latency, throughput, server load

Input metrics:



Avg input length
Avg input volume
Num missing values
Avg image brightness

Output metrics:

times return " " (null)
times user redoes search

Métricas que podem ser usadas:

Software: memória, computação, latência, throughput, server load

Input: Média do tamanho do input, média do volume do input, número de valores faltantes (missing values), média de brilho da imagem...

Output: quantas vezes retorna nulo, quantas vezes refaz uma busca

2. Semana 2

AI Systems = Code + data

Ou seja, um sistema de AI tem uma parte que o modelo é fundamental e outra em que os dados são fundamentais

Desafios no desenvolvimento do modelo::

-> Passos:

- 1 - Ir bem no training set
- 2 - Ir bem no test set
- 3 - Ir bem quanto ao objetivo do negócio/projeto

Não adianta também apenas o erro ser baixo, por exemplo, a performance pode favorecer ou ser pior/melhor em partes do dataset (discriminar por característica específica), ou problemas quanto a não detectar casos raros, por exemplo colocar sempre 0 em um dataset que tem poucos 1's

Estabelecer uma baseline para níveis de performance: Uma forma interessante é perceber o quão longe está as previsões das que um humano faria (HLP - human level performance), as vezes 70% é o melhor que um humano conseguiria num test set também e o melhor seria focar em outros tipos





Speech recognition example:

Type	Accuracy	Human level performance	
Clear Speech	94%	95%	10/0
Car Noise	89%	93%	4/0
People Noise	87%	89%	2/0
→ Low Bandwidth	<u>70%</u>	70%	~0/0

HLP funciona bem em dados não estruturados, como imagens, audio e texto, enquanto dados estruturados, como tabelas gigantes o HLP não é tão útil

Unstructured and structured data

Unstructured data		Structured data			
Image		User ID	Purchase	Number	Price
Audio		3421	Blue shirt	5	\$20
Text	<div>This restaurant was great!</div>	612	Brown shoes	1	\$35
		Product ID	Product name	Inventory	
		385	Football	158	

Outras formas de avaliação, é procurar o estado da arte das implementações presentes, ver a performance de um sistema antigo

Ways to establish a baseline

- Human level performance (HLP)
- Literature search for state-of-the-art/open source
- Quick-and-dirty implementation
- Performance of older system

Baseline helps to indicates what might be possible. In some cases (such as HLP) is also gives a sense of what is irreducible error/Bayes error.

Dicas para começar um projeto de ML:

- Começar com literatura, como cursos, blogs, projetos open source
- Encontrar uma implementação open source para usar de base
- Escolher um algoritmo já consolidado com várias informações disponíveis é muitas vezes mais interessante que escolher um algoritmo novo sem muita informação
- Fazer um teste de sanidade, por exemplo dando overfitting num training set pequeno antes de ir para um maior, ou treinando em uma parte pequena do dataset e depois ir aumentando pra ver se faz sentido

Error analysis:

- Examinar em cada entrada o que pode estar causando aquele erro (por exemplo, num áudio o barulho de carros) e colocar numa tabela.
Manualmente isso pode ser complicado, mas existem bibliotecas no Python para isso, como Landing less para detecção de imagens em computer vision
- Interessante ver a importância desses erros, quantos erros são causados por isso, etc, e o quão possível é melhorar isso
- Ver a quantidade de dados que aquela tag de erro ocupa, pra ver se é importante focar nisso

Prioritizing what to work on

Type	Accuracy	Human level performance	Gap to HLP	% of data
<u>Clean Speech</u>	94%	95%	1%	60% → 0.6%
Car Noise	89%	93%	4%	4% → 0.16%
People Noise	87%	89%	2%	30% → 0.6%
Low Bandwidth	70%	70%	0%	6% → ~0%

-
- Priorizar o room for improvement, a frequência de cada categoria, o quão fácil é melhorar aquela categoria e o quão importante é melhorar aquela categoria

Skewed Datasets:

- Quando o ratio de positivos para negativos é muito longe de 50-50, 99 pra 1, etc
- Interessante fazer uma matriz de confusão, nessa matriz precision e recall devem ser bons, por tanto o f1 score deve ser bom

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

Auditing Framework:

- Checar acurácia, bias, e outros problemas
- Fazer um brainstorm do que pode dar errado e estabelecer métricas desses problemas em partes dos dados
- Receber a avaliação do product owner antes de dar deploy

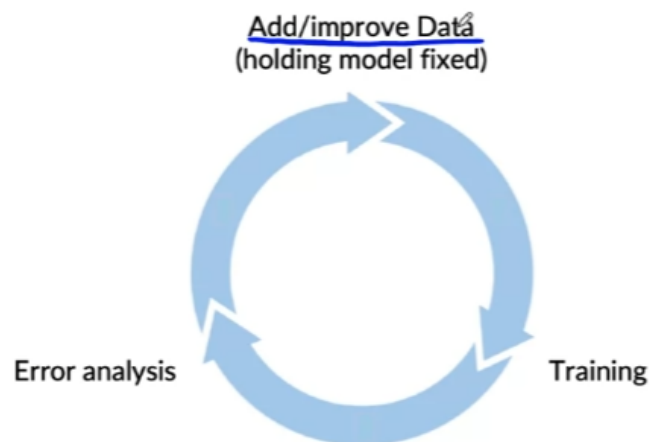
Data centric / Model centric AI development:

- Model centric: pegar os dados que temos e desenvolver um modelo que seja o melhor possível nesses dados, mantem os dados fixos e melhora o código/modelo
- Data centric: Pensamos primeiro na qualidade dos dados, usa ferramentas para melhorar a qualidade dos dados

Data augmentation

- Aumentar a quantidade de dados pode ser interessante para aumentar a proximidade entre o modelo e o que queremos, podendo ter um efeito de aumentar a performance em alguns tipos nos dados e depois ir melhorando nos outros/outras partes do dataset

Data iteration loop



-
- Adicionar dados geralmente não machuca a performance, entretanto o modelo não pode ter um bias alto e também não se deve adicionar muitos dados para um tipo para que isso mude

Adding features:

- Adicionar features pode ser interessante para tratar de erros mais específicos detectados no error analysis
- Mais interessante de se usar em dados estruturados do que em não estruturados (imagens)

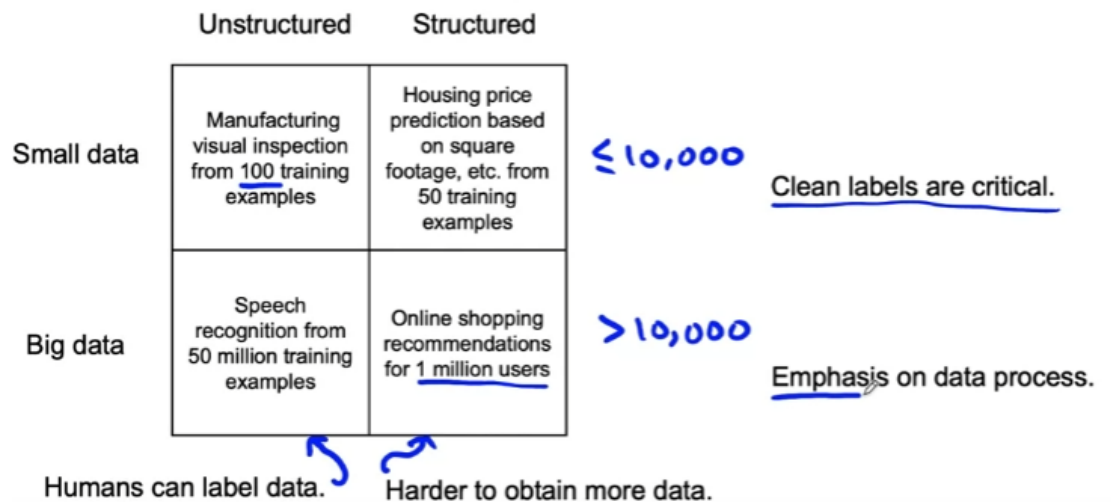
Experiment Tracking:

- trackear: Code versioning, dataset usado, hiperparâmetros
- usar arquivos e texto, spreadsheets, tracking system, etc

3. Semana 3 - Data

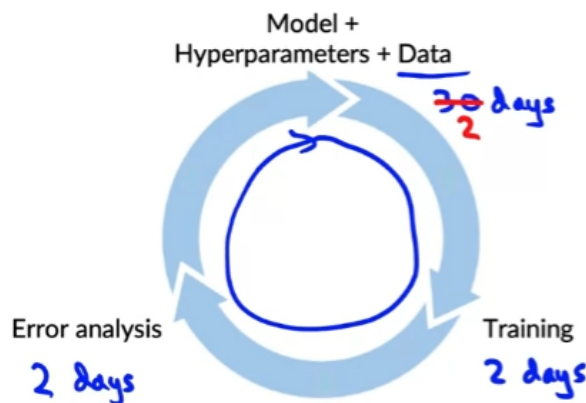
- Há vários tipos de problema de machine learning, separados em se são estruturados ou não, se são grandes ou pequenos, etc
- Há uma criticalidade maior na qualidade dos dados quando se tem um dataset menor, enquanto em um dataset maior há um enfase maior no processo em si.

Major types of data problems



- Data augmentation.
- Em dados não estruturados: existe uma probabilidade maior de ter uma coleção de dados unlabelled, geralmente podemos usar data augmentation injetando imagens por exemplo e isso é mais útil
- Em dados estruturados é mais difícil de se obter mais dados
- Em datasets pequenos, dados limpos são críticos, pode-se manualmente olhar pelo dataset e arrumar labels que já estavam errados
- Em datasets maiores, o foco é maior no processo, não é humanamente possível checar os dados muitas vezes.
- Quando for classificar uma label, se houver descondância no time o ideal seria resolver, mas caso contrário também pode-se criar uma classificação para somente aquela label ambigua, através de votações e tentar criar um grupo que classifique esses casos ambiguos juntos.
- Quanto a tempo, o importante é chegar no loop o mais rapido possível, ao menos que você veja que mais dados são necessários por experiência própria

How long should you spend obtaining data?



- Get into this iteration loop as quickly possible.
- Instead of asking: How long it would take to obtain m examples?
Ask: How much data can we obtain in k days.
- Exception: If you have worked on the problem before and from experience you know you need m examples.

-
- Há vários tipos de como classificar os dados, seja internamente (na empresa) (in-house) por uma fonte externa (outsourced) ou por uma fonte externa da comunidade (crowdsourced). Isso depende muito do dataset, o quão caro pode ser isso, etc.
- Datapipeline / Datacascades: passos que os dados precisam passar antes de chegarem no output final, 2 etapas principais: POC (proof-of-concept) em que o objetivo é decidir se a aplicação é aplicável e pode ir para deploy, deve-se priorizar um protótipo que funcione e está tudo bem os dados precisarem de pré-processamento manual. E a Production Phase que após a utilidade do projeto ser estabelecida se usa ferramentas mais sofisticadas (TensorFlow Transform, etc) para ter certeza que é replicável

