Reproductibilité en apprentissage automatique

David Beauchemin

Département d'informatique et de génie logiciel,
Université Laval

david.beauchemin.5@ulaval.ca

30 octobre 2020

Objectifs de la présentation

- Sensibiliser sur les enjeux de la reproductibilité.
- Inciter l'intégration des solutions permettant une meilleure reproductibilité dans vos solutions d'affaires ou académiques.

Mes qualifications

DAVID BEAUCHEMIN
Candidat au doctorat
Département d'informatique et de génie logiciel

- Introduit (informellement) à la recherche reproductible en 2016 (RMarkdown et Git).
- Participation à REPROLANG, visant la reproductibilité d'articles scientifiques ayant mené à la publication d'un article en 2020 (où nous abordons les notions présenté ici) [**?**].
- Membre actif dans le développement de solution d'intégration facilitant la reproductibilité (Poutyne, MLFlow callback).

Plan

C'est quoi la reproductibilité ?

La reproductibilité est le principe qu'on ne peut tirer de conclusions que d'un événement bien décrit, qui est apparu plusieurs fois, provoqué par des **personnes différentes**.
Toutefois, on utilise souvent ce terme pour spécifiquement désigné la **réplicabilité**. Soit la réplication (reproduction) des résultats d'un articles dans des environnements pas (toujours) différents [**?**, **?**].

En somme

- Être capable de répliquer les résultats d'un article/ d'un projet,
- à partir du même jeux de données ou un jeux de données différents (mais proche),
- en utilisant la procédure d'entrainement de l'article ou en utilisant notre procédure d'entrainement et
- en utilisant le code du projet.

Pourquoi s'y intéressé ?

- 70 % des chercheurs en science on échoué dans leur tentative de reproduire un article d'un autre chercheur,
- 50 % n'on pas réussit à reproduire leur **propre** expérimentations [**?**].

Pourquoi s'y intéressé ?

L'informatique ne fais pas exception à cela malgré la simplicité (théorique) de réplication des résultats. Selon une étude, sur 255 articles près de 40 % n'était pas réplicable [**?**].

Pourquoi s'y intéressé ?

La réplicabilité du code et d'un article facilite la réutilisation pour d'autres projets de recherche **et** le transfert vers l'industrie.

Plan

- Non disponibilité du jeux de données ou version (pas clair) du jeux de données,
- mauvaise spécification ou sous-spécification du modèle ou de la procédure de formation,
- manque de disponibilité du code nécessaire pour exécuter les expériences, ou erreurs dans le code,
- configuration du modèle déficiente [**?**] [1].

1. Liste sélective de ceux présenté dans l'article.

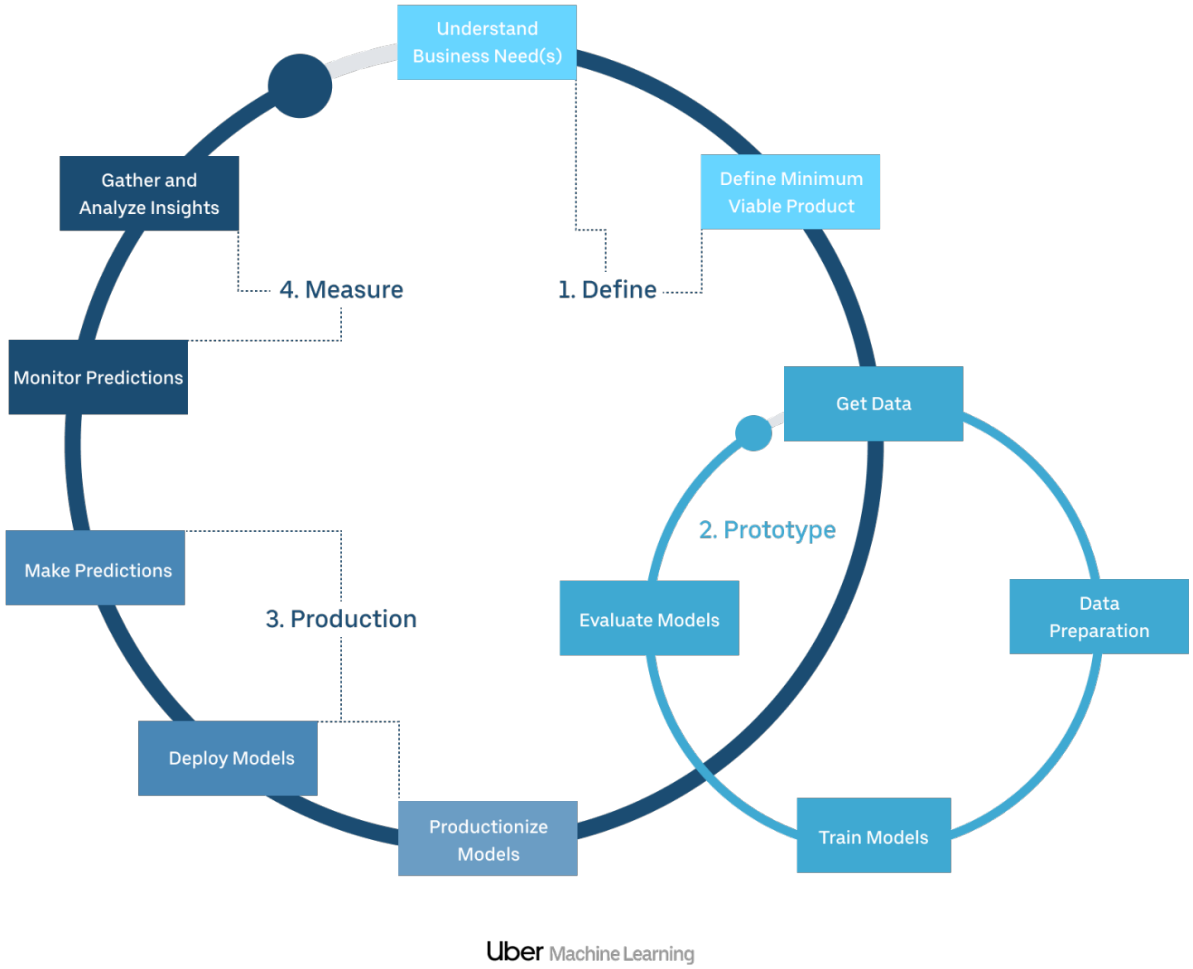

FIGURE – From Uber Engineering [2]

2. https://eng.uber.com/scaling-michelangelo/

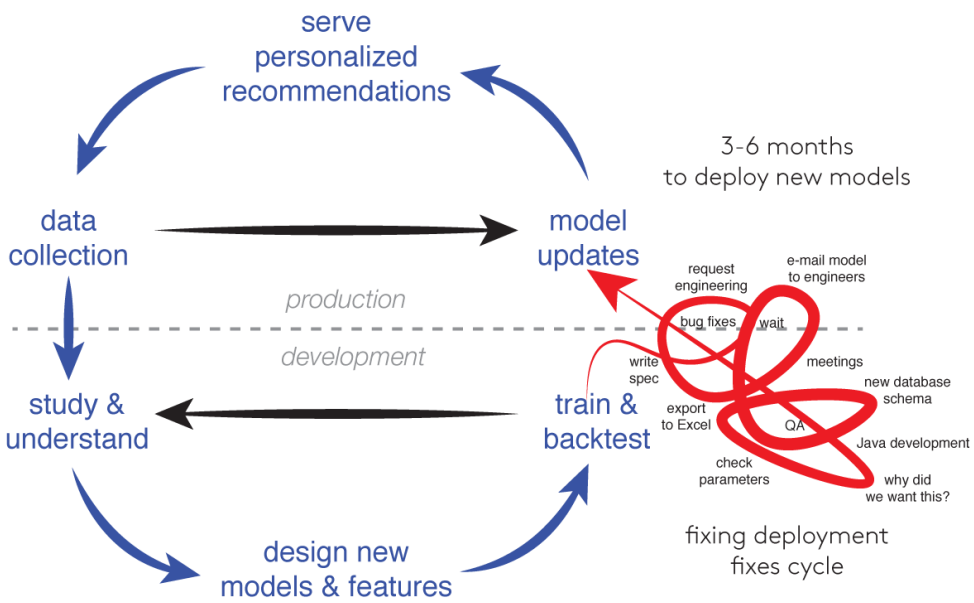

FIGURE – The need for Agile machine learning [3]

3. https://johann.schleier-smith.com/blog/2015/08/09/need-for-agile-machine-learning.html

Plan

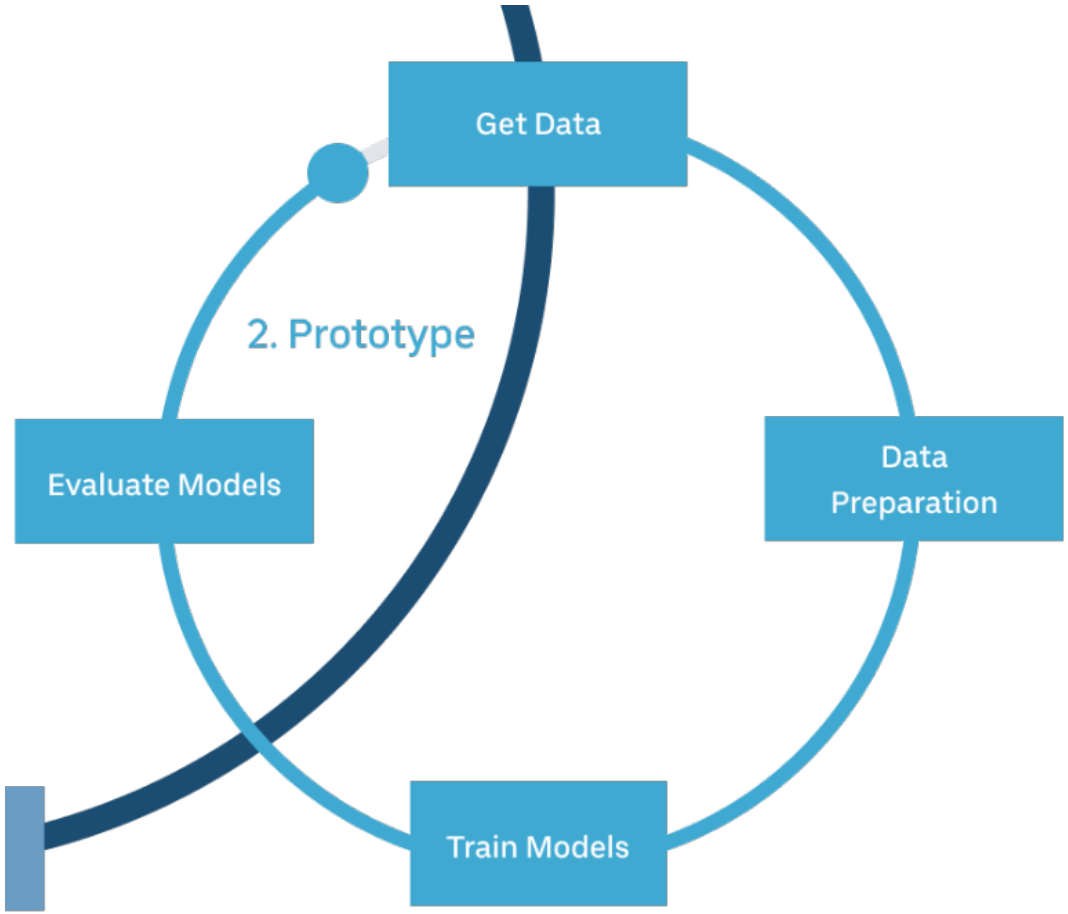

FIGURE – From Uber Engineering [4]

4. https://eng.uber.com/scaling-michelangelo/

Version des données - Étapes de pré-processing

- Avec qu'elle version des données on travail ?
- Comment gérer **facilement** plusieurs version des données ?
- Comment définir **facilement** les étapes de pré-processing des données ?

Il nous faut des *data pipelines*, des tuyaux que nous pouvons raccorder **facilement** à nos modèles pour l'entrainement et la mise en production, par exemple, Data Version Control (DVC) [5].

5. https ://dvc.org/

Version du code

- Avec qu'elle version du code on travail ?
- Comment savoir **rapidement** qu'elle est la différence d'implémentation entre deux versions du modèles ?
- Comment gérer **facilement** les embranchements d'expérimentations ?

Il nous faut un outil nous permettant de visualiser la différence entre des fichiers de code et nous permettant d'avoir plusieurs version du code pouvant exister **en même temps**, par exemple Git [6].

6. https ://git-scm.com/

Développement des modèles

- Ne pas réinventer la roue.
- Simplifier l'écriture de code pour développer des modèles.
- Qui facilite l'entrainement (GPU, multi-GPU/CPU).

Il nous faut des outils nous permettant de simplifier le développement de nos modèles, par exemple, Poutyne [**?**], PyTorch Lightning [**?**], Scikit-Learn [**?**], Gensim [**?**] et Allen NLP [**?**].

Entraînement, configuration et résultats

- Avec qu'elle version du code, du modèle et des données avons-nous fait cet entrainement ?
- Quels sont les résultats ?
- Comment visualiser **rapidement** les résultats et les paramètres de configuration ?

Il nous faut des outils nous permettant de *logger* les paramètres d'entrainement et les résultats, par exemple, MLFlow [**?**] et Sacred [**?**].

Rapport et analyse des résultats

- Comment créer des tableaux de résultats **facilement** (pas à la *mitaine*) ?
- Comment s'assurer **facilement** que les résultats sont à jours ?
- Comment visualiser **rapidement** les résultats et les paramètres de configuration ?

Il nous faut des outils nous permettant de créer des tableaux de résultats à même les résultats, soit de diminuer le plus possible le travail manuel, par exemple, Python2latex [7] et Markdown [8].

7. `https://github.com/jsleb333/python2latex`
8. `https://fr.wikipedia.org/wiki/Markdown`

Dockerisation

- Comment s'assurer que nos modèles fonctionne sur d'autres environnement ?
- Comment faciliter la réutilisation de notre code ?

Docker ! [9]

9. `https://www.docker.com/`

Plan

Développer des processus rigoureux (par essai et erreur) et ne pas prendre tout ce qui a été discuter ici comme l'unique solution.

Pour aller plus loin

- Clean code [10]
- Dagger de projet [11]

10. https://www.oreilly.com/library/view/clean-code-a/9780136083238/
11. https://github.com/davebulaval/cookiecutter-machine-learning-research

References I