**Slide 1**

Le MLOPs, c’est quoi ? C’est tout ce qui va permettre d’aider et d’accompagner le fait de mettre en production des algorithmes d'intelligence artificielles, ce ne sont pas des outils ni des technologies ; c’est majoritairement une question de de culture et de bonne pratique.

Bonjour, je m’appelle Guillaume chervet je suis ML ingénieur chez axa France alors je ne sais pas exactement ce que ça veut dire mais en gros je définis mon rôle comme le fait d’accompagner les les data scientist, mettre en production les algorithmes d’intelligence artificielle.

Je travaille dans une équipe qui travaille essentiellement sur des projets de lecture automatique de document en temps réel. Généralement cela va mettre en œuvre principalement un ou plusieurs algorithmes de Deep learning.

Aujourd’hui, quasiment 95% des projets échoue à l’étape mise en production. Ce chiffre commence à diminuer. Cela ne m’étonne pas car sur ce type de projet au top du top de la complexité dans tous les domaines, toutes les clés sont réunies pour que ça échoue.

**Slide 2 : trio intro**

La première problématique et une problématique humaine alors aujourd’hui la plupart des applications on a des devs et des Ops et ce n’est déjà pas facile de faire travailler tout le monde dans une équipe unie. Là on va rajouter encore des spécialités dans les métiers de la data. Des experts de la data.

Aujourd’hui, il est quasiment impossible d’être expert dans les 3 domaines. Ce type de projet est au top du top de la complexité dans tous les domaines d’expertise technique. Cela nécessite par conséquent d’accepter les contraintes des autres et de leur faire confiance.

**Slide 3 : coûte chère intro**

Ajouter au fait qu’il y a plus d’acteurs : on est sur des projets qui généralement coûtent extrêmement chers.

Alors là c’est un projet de classification d’email qui a très bien fonctionné. Chez nous c’est un peu un projet exemple. Aujourd’hui le coût de fonctionnement du projet par mois est environ de 9000€, ce qui fait par an un coup de projet à un peu plus de 100000€. Je pense que l’on peut encore diviser en les coûts par 2. Aujourd’hui sur ce projet on classifie environ 10000 mails par jour et ce qui fait un coût de classification d'un email a 3 centimes d’euros. Si par exemple, on réalise un réentraînement par an, par exemple on va avoir des coûts de projet qui font qu’aujourd’hui le cout de classification d’un mail sera environ de 4 centimes d’euros et là encore je ne parle pas de la mise en place du projet qui elle a coûté un peu plus d’1000000 d’euros.

**Slide 4: coûte chère intro**

Ici un exemple fonctionnel de classification d’un mail.

Pour information, toutes les images que vous voyez là où il y a un petit texte en dessous en style python, ce sont des images générées avec des les algorithmes d'intelligence artificielle.

**Slide 5 : sommaire**

Comme vous pouvez le constater dans ces projets au top du top de la complexité technique, toutes les clés sont réunies pour que ça échoue sont réuni.

Cette présentation est découpée en 4 parties.

Dans la première partie on va introduire le Deep learning et ces problématiques.

Dans la 2nde partie on va expliquer ce qu’on attend d’un un projet en production chez nous chez AXA France.

Ensuite dans la 3ème partie on va s'amuser à faire une recette. Recette de ce qu’il faut faire pour faire échouer un projet. L’idée c'est que on a eu des expériences qui nous aide à nous améliorer aujourd’hui.

Ainsi, dans la 4e partie, nous avons tirer des leçons de ces expériences qui vont nous permettre d’être sur le chemin pour faire qu’un projet se déroule bien.

**Slide 7**

Une des sous-ensembles de l'intelligence artificielle est le machine learning et en un des sous-ensembles du machine learning est le deep learning.

**Slide 8**

Vous entendrez parler souvent de données structurées en général être représenté par un fichier Excel. En général pour ce type de de de de problématique où il y a un nombre d'entrées sorties limitée. On va utiliser des algorithmes de machine learning classique.

**Slide 9**

Quand on a des données de type son, image, vidéo, le nombre d’entrées sortie possible est en général beaucoup plus élevé. Par consequent, on va généralement utiliser des algorithmes de Deep learning qui utilise un véritable réseau de neurones plus complexes.

**Slide 10**

Alors ça c’est un schéma qui est sorti du livre de François Chollet qui est le créateur de Keras que je vous conseille vraiment de de de lire.

Quand on fait du code classique en général on va avoir des règles plus des données d’entrées et on va écrire un algorithme qui va nous permettre de générer des réponses.

En machine learning c'est un peu différent, Il va nous falloir de la donnée d’entrée et aussi beaucoup de réponses. Pas qu’un peu, vraiment beaucoup, des milliers d'exemples 10000, 100 000 voire plus, ce qui va permettre d’écrire du code qui va pouvoir générer les règles : une intelligence artificielle, on va parler en techniquement de « modèle ».

**Slide 11**

En production le code classique ou l’ia s’utilise de la même façon.

On reçoit une entrée, on exécute, puis on récupère le résultat.

**Slide 12**

Dans le passé on parlait de projet : modèle centrique, parce que on n’était pas sûr d’arriver à faire une ia adapté. Aujourd’hui la plupart des points côté ia : en tout cas pour nos besoins chez AXA France ont déjà des solutions viables. On sait qu'on va arriver à réaliser une IA pour nos cas d’usages. Par contre le cœur de la problématique, vraiment le truc qui fait qu’un un projet est faisable ou non : c’est la donnée. Si on a de la bonne donnée, de bonne qualité, des entrées et réponses, beaucoup, c’est vraiment ça qui va faire la clé du succès d’un projet. La donnée, c’est votre trésor qui fera la différence.

**Slide13**

Alors maintenant on va expliquer comment se déroule un peu un projet. Si on reprend notre projet de classification des emails et si on veut commencer le projet. La première chose à faire c'est récupérer de la donnée : des entrer et des réponses. Sur ce projet cela tombe bien on avait déjà des emails et des réponses car en production on a des humains qui classifient les emails manuellement. On a déjà la donnée et les réponses c'est parfait ça pour constituer un set de données.

**Slide14 et 15 et 16**

Ensuite avec cette donnée on va pouvoir Itérer pour créer une IA et construire une API que l’on pourra mettre en production. Notre application en production pourra être consommer par des application cliente.

**Slide 17**

En production il est extrêmement important de mettre en place ce que l'on appelle une boucle de feedback. Une boucle vertueuse. En cas d’erreur de classifications les application cliente doivent permettre de corriger l’erreur manuellement. On va pouvoir au fil de l’eau constituer de nouveau set de données de meilleure qualité et ainsi améliorer nos Intelligences artificielles artificielle.

**Slide 18 Demo intro**

On va faire une petite démo de comment se déroule un projet en se basant un dataset chats et chiens bien connu de Kaggle.

**Slide 19 Demo Data**

Avant de de commencer la petite démo. Imaginons que l’on a 20 000 fichiers à disposition. On va utiliser 17000 fichiers pour l’entrainement du modèle. Puis on va garder 3000 fichiers de côté pour la validation, pour tester sur notre modèle sur des données qu’il n’a jamais vues. Ces données de validation nous serviront aussi à tester les environnements de production.

**Demo Time**

**Slide 20 After Demo Datadrift**

Si on donne à manger à nos models des données différentes, des données sur lesquels elles ont apprises. Les prédictions risque se dégrader.

Pour notre projet de classification d’emails, par exemple si le constat automobile change de template, il est possible que l’IA ne reconnaisse moins bien les sinistres automobiles. Dans ce cas il faut réeentrainer puis re-livrer le modèle en production.

Reprenons la demo avec nos chats et chiens. Maintenant imaginons que nos utilisateurs commencent à utiliser des photos de peluche de chien et chat ce qui n’était pas prévu au début, les prédictions ne fonctionnent plus correctement. Autre exemple avec des dessins de chiens et chats.

**Slide 21 Le Graal**

Le Graal pour nous ; ce serait de pouvoir re-livrer en production sans quasiment aucune action manuelle

**Slide 22 100% code**

Ça c’est un slide Google, un projet de Deep learning c’est un peu moins de 5% data science, par contre c’est 100% de code !

**Slide 23 : Sommaire 2ème partie**

Maintenant je vais vous expliquer ce qu’on attend d’un projet en production chez AXA France

**Slide 24 : Besoins**

Le plus important dans un projet est de répondre aux besoins utilisateurs et en général ce que veut l'utilisateur.

Il veut une bonne qualité de prédiction, que la production ne soit pas chère rapide. Nous chez AXA France, on se donne des contraintes de temps de réponses qui doivent être inférieures à 10 secondes pour la lecture automatique des documents.

La sécurité ça n’est pas négociable.

On veut que les projets soit monitoré. L’idée c’est que si on a un drift de données cela risque d’entrainer un incident de production. Si on est sur un projet majeur qui peut bloquer la boite, on ne va pas attendre l’incident et bloquer la boîte pendant des semaines.

On arrive au dernier point, il faut que l’on puisse être en mesure de ré entraîner et redéployer rapidement. Si on a un incident majeur qui bloque la boite, il faut pouvoir relivrer le plus vite possible et ne pas attendre des mois pour débloquer la situation.

**Slide 25 Schéma fonctionnelle**

Alors là, c'est le schéma fonctionnel du workflow qui explique comment on lit un permis de conduire Français. On lit aussi les anciens permis de conduire Français, mais pour simplifier on ne le pas affiché ici. L'idée c'est que on va recevoir une image ou 1 PDF en entrée et on va d'abord réaliser une première étape où l’on va découper ce PDF en une ou plusieurs pages. Sur chaque page on va exécuter le même algorithme de zoning pour détecter un les rectos et les versos. Après ça on va rentrer dans des algorithmes spécifiques au recto et au verso. Prenons le recto en bas, on va d'abord remettre droite l’image du recto avec un algorithme spécifique au recto. Puis exécuter un autre algorithme pour zoner les champs spécifiques à ce recto pour finalement exécuter un OCR pour extraire finement le texte des champs.

Si on hébergeait une API qui expose ce service en mode monolithe. On aurait plusieurs problèmes, Le premier c’est le temps de réponse c’est-à-dire que quand on va appeler cette API en monolithe ce qui va se passer c’est que chaque algorithme va consommer toutes les ressources CPU de la machine. La conséquence est que le pipeline va s’exécuter en séquentiel, le temps de réponse est supérieure à 30 secondes. Autre problématique c'est que pendant ces 30 secondes : si un autre appel arrive à l’API ; cet appel va devoir attendre la fin du premier traitement pour commencer à traiter le nouveau traitement. Alors vous me direz que face à cela on peut rajouter des machines ; mais autre problématique et si on héberge cela en mode monolithe, il nous faut des machines d’environ 32 gigaoctet de RAM qui coute très chère et qui être vont longue à démarrer l’application. Cela va très compliqué de pouvoir scaler rapidement et s’adapter à la charge utilisateurs.

La solution c’est de mettre en place des microservices, on va héberger chaque algorithme sur des ressources différentes ce qui nous permettent de finement pouvoir configurer la RAM et le CPU nécessaire. On va pouvoir jouer les traitements en parallèle et scaler rapidement.

**Slide 26 : Function**

Comment ça fonctionne techniquement ? On utilise ce qu'on appelle des fonctions. Chaque algorithme est hébergé sur une fonction. Cette fonction écoute une Queue en entrée et dès qu’elle a du temps libre, elle va pouvoir dépiler les messages qui arrive dans la Queue et les traiter.

Par exemple prenons le SPLITTER qui extrait les pages d’un PDF, la fonction va récupérer un message de la Queue, les fichiers ne sont pas stockés dans le Queue mais elles sont stockées dans un REDIS on va recevoir uniquement l’identifiant du fichier en entrée ; on va aller chercher l’image dans REDIS pour exécuter nos algorithmes et par exemple si on a un PDF avec 2 pages on va extraire les 2 pages puis générer 2 identifiant ;pousser les images dans REDIS avec l’identifiant et finalement ajouter 2 message dans la QUEUE en aval pour pouvoir appeler le service suivants, dans ce cas le Zoning de détection des permis.

S**lide 27 : Scale**

L’avantage de fonctionner ainsi est l’on peut scaler rapidement et supporter les piques de charge. Imaginer que tout d’un coup on va recevoir 500 permis, le nombre de fonction va augmenter pour pouvoir dépiler plus vite les messages. Il y a aussi un avantage financier car on ne consomme que ce dont on a besoin.

**Slide 28 : Architecture**

Alors ça c'est le schéma d'architecture. Si vous le trouver compliqué, c’est normal car c’est compliqué.

Je ne vais pas décrire tout le schéma en entier mais, en en résumé on utilise une architecture asynchrone. On va recevoir en haut à gauche une image en PDF le premier POD va le recevoir le fichier le pousser directement fichier dans redis puis avec l’identifiant du fichier ajouter un message dans la QUEUE en amont du splitter. Ensuite le Splitter va récupérer le fichier le traiter, etc. etc.

Cette architecture fonctionne pour le temps réel et aussi en mode Batch.

**Slide 29 : Difficultées**

**Ces projets ne sont pas sans difficultés.**

Je vais vous présenter les 2 plus importantes que l'on rencontre

**Slide 30 : Triptique**

Tout d'abord il y a un vrai triptyque de choix. Un curseur à positionner entre la qualité de prédiction, le temps de réponse et le coût des infrastructures en production.

C'est à dire que si on veut avoir de la meilleure qualité de prédiction on peut accepter de payer plus chère ou de perdre du temps de réponse.

A l'inverse si on veut que on puisse répondre plus rapidement, il faut pouvoir accepter soit de payer plus chère ou de perdre de la qualité de prédiction.

Et à l'inverse si on veut que le projet coûte moins cher en production, il faut soit accepté de de rallonger le temps de traitement des réponses ou alors de diminuer la qualité de prédiction.

**Slide 31 : Trouver les différences ?**

Alors est-ce que vous voyez les différences ? Les différences dans ces images, nous on ne peut pas les voir, cependant l’IA si ! J'ai eu une première expérience de ce problème sur notre tout premier projet sur lequel j'ai travaillé. En bon développeur, il y avait une librairie Pillow qui sert à redimensionner les images et qui n’était plus à jour. Comme on doit maintenir toujours les librairies. Je l’ai mis à jour. Ce qui s'est passé ensuite c’est que les prédictions ne marchent quasiment plus. Alors j'ai regardé les images. Images identiques à mes yeux, impossible à comprendre. Et en fait, la particularité c'est que c’est vraiment un problème pour l’IA qui elle ne reconnait plus les images et est sensible à l’évolution des algorithmes. La conséquence de ce problème est que si en production on n'utilise pas les mêmes versions de python, ni les mêmes versions de chaque librairie et ni même le même OS on n’est quasiment sûr que la production ne va pas fonctionner comme attendu.

Revenons à la démo. Si on revient avec mes chats et mes chiens l'algorithme IA utilisé ici reçoit en entrer des images redimensionnées en 200\*200 pixels via la librairie Pillow. Elle a aussi été entrainé avec en entrer des images redimensionnées avec Pillow en 200\*200 pixels.

Sur le projet permis de conduire, on a remplacé les redimensionnements d’images par la librairie OpenCV ce qui nous a permis de gagner un peu moins de 2 seconds totaux sur le traitement complet du pipeline.

Dans cette démo j’ai aussi implémenté une version ou les images ici en production en production redimensionné avec OpenCV avant d’être donnée au model IA.

Par conséquent, ce qui se passe c’est que si je réalise des prédictions avec un redimensionnement via Pillow puis avec via OpenCV on n’obtient pas les mêmes résultats.

J'ai réalisé un test avec 12500 images et sur 12500 images et on a environ 500 images qui ne prédisent pas la même chose.

**Slide 32 : Sliding impact**

Cela ne fait que 3% de différences de prédiction, mais si vous prenez un pipeline complexe on va enchaîner des algorithmes avec différents IA. Ce qui arrive, c’est que les 3% qui vont se propager tout le long de du pipeline, ce qui fait qu’à la fin le taux de prédiction devient très faible.

**Slide 33 : Sommaire partie 3**

Alors, l’idée, c’est que maintenant pour cette partie on va faire un petit retour d’expérience pour expliquer nos expériences qui nous ont permis de d’apprendre sur ce qu’il ne faut pas faire. Pour le faire de façon rigolote on va ouvrir une pizzeria et on va s'organiser pour servir des pizzas à nos clients. On va mettre en place tout ce qu’il ne faut pas afin de faire couler notre pizzeria.

**Slide 34 : Elisabeth**

Pour commencer on va organiser notre équipe. Dans un projet ce qui se passe, c'est qu'en général il y a 3 phases principales. Dans la première phase d’exploration on va tester si le projet est faisable, ensuite on passe dans une phase d’industrialisation puis enfin une phase de Déploiement et de Run.

Alors là je vous présente Élisabeth qui est data scientist et que l’on va mettre toute seule à travailler sur la phase d’exploration.

**Slide 35 : Elisabeth**

Alors Élisabeth est une excellente cuisinière on va lui demander de de pétrir la pâte à pizza et de bien l'étaler et c'est ce qu'elle fait avec tout son cœur et elle le fait très bien.

**Slide 36 : Elisabeth**

Voilà, maintenant Élisabeth a fait sa pâte à pizza elle l’a étalé.

**Slide 37 : Elisabeth - Hicham**

Je vous présente Hicham, son rôle c’est de répartir les ingrédients sur la pâte à pizza.

**Slide 38 : Elisabeth – Hicham Hop**

Maintenant Élisabeth refourgue la pâte à pizza et les ingrédients à Hicham. Comme vous pouvez voir là à gauche, Élisabeth a oublié de de donner le jambon à Hicham.

**Slide 39 : Hicham**

Alors maintenant, Hicham, c’est à son tour de jouer et c’est à lui d’étaler les ingrédients sur la pâte à pizza, cependant il se rend compte que le jambon a été oublié et il pensait aussi qu’Elisabeth aurait pré-coupé les ingrédients. Malheureusement, il n’a pas de couteau à disposition.

Hicham en bon développeur fait tout ce qu’il peut. Il prend les ingrédients et les étales comme il le peut sur la pâte à pizza.

**Slide 40: Hicham**

Voila

**Slide 41: Hicham - Khalid**

Je vous présente Khalid ; c’est notre Ops est responsable de cuire la pizza pour ensuite la livrer aux clients et ça va être à son tour de jouer.

**Slide 42 : Hicham – Khalid Hop**

Hop, Hicham refourgue la pizza bien garnie avec son cœur à Khalid.

**Slide 43 : Khalid**

Khalid se retrouve avec la pizza bien garnie ; mais mince on a oublié de lui donner les instructions. Il ne sait pas comment la cuire.

**Slide 44 : Khalid**

Alors Khalid fait ce qu’il peut pour cuire la pizza, mais mince c'est une pizza spéciale, il ne le savait pas et maintenant elle est brûlée.

**Slide 45 : Team resumé**

Pour résumer, ici on a organisé notre cuisine de façon à séparer les rôles et à ne pas les faire travailler ensemble et en même temps en « équipes » Elisabeth, Hicham et Khalid. Si vous voulez faire échouer votre projet, c’est la bonne façon de s’organiser.

Il faut savoir que le choix même des librairies dès le début du projet impacte directement sur la production et donc ne pas travailler ensemble, ne pas se soucier de la production dès le début du projet, c'est là une des clés pour le faire échouer.

**Slide 46 : Lilian**

J’allais oublier de parler de Lilian. Lilian est venu ce matin livrer du sel, c'est une des matières premières de nos pizza ; mais il s'est trompé il a livré du sucre ! Elle va être bonne notre pizza.

**Slide 47 : Table**

Maintenant on va dresser la table pour nos clients. Alors si vous voulez faire échouer un projet cela peut être une très bonne idée de ne pas se focaliser sur le besoin utilisateurs.

Premier exemple, sur le projet permis de conduire, on nous a demander de lire les permis en moins de 10 secondes. Quand on a commencé à travailler dessus et prendre le tout premier algorithme, le Splitter PDF pour le mettre en production. On s’est vite rendu compte que sur l’environnement de production, les temps d’extraction des document PDF prenaient environ 15 à 30 secondes ce qui étaient problématiques. Les datascientist s’en sont pas aperçue car ils travaillent sur des machines surpuissantes qui réalise l’opération très rapidement.

Deuxième exemple, quand on était un peu plus loin sur le projet. On s'est rendu compte que certaines lectures étaient très lentes et engorgeait les systèmes. En regardant un petit peu plus en détail les permis qui ne passaient pas. On s'est rendu compte que c'est permis, en général, étaient quasiment illisibles. Pour ce genre de cas, cela aurait été une bonne idée d'aller voir les utilisateurs et de leur demander s’il ne vaut pas mieux tout de suite en 2 secondes remonter que le permis n'est pas assez propre et pas lisible plutôt que d’essayer de le lire « quoi qu'il en coûte » et qu’à la fin, après 40 secondes on ne remonte quasiment aucun champ lu.

**Slide 48 : Apéritif.**

Apéro

Alors là on va enfin servir l'apéro à nos premiers clients. Quand on commence un projet, surtout si on veut faire de l’agile et avoir un Time To Market rapide. Cela peut être une bonne idée de toujours tester toutes les approches de code classique. Plutôt que de tout de suite réaliser du Deep learning.

Mettre en place un algorithme de Deep Learning, c’est un peu une usine à gaz, c’est compliquer, il faut beaucoup de données et cela coute très cher.

Tout d'abord, le code classique va plus vite à mettre en place. Cela permet d’itérer rapidement et en général le temps de traitements est plus rapide et demande moins de ressources CPU.

Le code classique ne me permettra cependant pas d’être aussi bon en prédiction que le Deep learning. Le Deep learning permettra de de résoudre des problématiques très complexe de façon bien meilleure, mais les itérations sont longues.

De manière générale, si vous commencez avec du code classique et que vous n’arrivez pas à atteindre résultats désirez, ce n’est pas du temps perdu, parce que ce code classique va vous permettre de gagner du temps sur la partie annotation. Vous pouvez pré-annoter vos annotations et aussi vous aurez aussi un système de fallback dans le cas où le l’IA n’arrive pas bien à prédire.

**Slide 49 Apéritif 1**

Alors ça c’est un exemple qu’on a eu sur le projet permis de conduire. Pour redresser un recto ou un verso on utilisait à chaque fois 2 IA afin de remettre l’image droite. Le problème, c’est que ces algorithmes étaient très gourmands en ressources et donc coutaient très cher et prenaient trop de temps de traitement. On a pu les remplacer par des algorithmes plus classiques avec la même qualité de résultat, la conséquence : un grand gain de temps de réponse et d’argent.

**Slide 50 Apéritif 2**

On a eu la même problématique sur le projet de classification d'emails. Ce qui se passait c’est qu’on avait un algorithme de Deep learning qui devait remettre droit les documents. Elle ne le faisait pas bien et ça prenait même plus de temps que la lecture du document en lui-même. Du coup on a on a insisté pour chercher des algorithmes et on a réussi à mettre à quasiment toutes les images parfaitement droites sans IA, extrêmement rapidement en moins 0,3 secondes et cela avec très peu de ressources consommées.

Et qui sait, si demain on veut améliorer encore plus les performances de lecture et que l’on a le budget, nous pourrons entrainer une IA.

**Slide 51 Apéritif 3**

Dernier exemple pour notre apéritif, on a essayé de remplacer les algorithmes écrits en Deep Learning sur mesure pour détecter les champs à lire par des algorithmes gratuits trouvés sur internet déjà tout fait. On s'est rendu compte que la qualité de prédiction était quasiment identique. Les seules différences constatées sont que cet algorithme était moins bon pour détecter le texte sur les permis qui sont vraiment complètement illisibles. Si on avait exclu dès le début ces permis illisibles, on aurait pu utiliser cet algorithme et ainsi gagner du Time To Market et gagner beaucoup d’argent.

**Slide 52 : Salad**

Maintenant on a fini l’apéritif, on va passer à l'entrée. Cela peut-être une bonne idée de ne pas industrialiser son code d'entraînement. Si vous n’industrialiser pas votre code d’entrainement. Quand on doit partir sur la phase mise en production on est quasiment obligé de réentraîner les modèles parce qu’on est quasiment obligé de réaliser des adaptions du code. Si vous n’avez pas industrialisé le code d’entraînement c’est-à-dire que vous êtes resté sur Jupyter Notebook sans versionner les données, ni le code. C’est quasiment le temps complet du projet de refaire l’entraînement des modèles.

Alors maintenant, on va pouvoir passer au plat principal et servir notre pizza à nos clients

**Slide 53 : Spaghetti**

En fait non, on s’est trompé, ce n’est pas une pizza ; on a servi un plat de spaghetti. La qualité de code, les tests unitaires aujourd'hui le c'est extrêmement important pour la maintenance du projet. Ne le fait pas si vous ne voulez pas que votre projet réussisse.

Par exemple sur le projet de classification d’emails, on avait une phase d’extraction des données de beaucoup de mails qui prenait environs 2 semaines de temps de traitement. Il y avait 2 choix possible, attendre 2 semaines pour valider une modification de code, ou prendre 20 minutes pour réaliser un test unitaire. Avec cet exemple les Data Scientists on comprit l’intérêt des Test Unitaires. Grace à cela on a pu itérer très rapidement et notre temps de traitement est inférieur à la demi-journée.

**Slide 54 : Code sans test**

Vous voyez cet exemple de code. C’est un code classique que l’on peut généralement voire en python. Sans test unitaire, sans exemple d’entrées et sorties, je suis personnellement incapable de savoir ce que réalise cette fonction.

**Slide 55 : Sel**

Une petite pincée de sel sur notre pizza.

Si vous voulez vraiment faire échouer votre projet, ça peut être aussi une bonne idée de travailler sur des GIT séparés. Un GIT unique, c’est simple et pragmatique, cela permet de travailler en mode équipe via de PullRequest et de se focaliser la production. Cela permet de minimiser le nombre d’action manuel et ainsi simplifie les automatisations. Alors si vous ne voulez pas travailler en équipe et complexifier votre projet déjà complexe, séparer les GIT, c’est une bonne recette pour échouer.

Séparer les GIT c’est aussi super pour avoir des déphasages de code entre l’entrainement et la production et par conséquent que cela ne fonctionne pas.

**Slide 56 : Dessert**

Cerise sur le gâteau pour le dessert !

Ne pas monitorer sa production, ça c'est simple vous avez un projet majeur, vous ne l'avez pas monitoré. Le modèle commence à prédire n'importe quoi et vous n'êtes pas au courant. Vous allez vers l’incident majeur qui va bloquer toute votre boîte et potentiellement la couler.

**Slide 57 : Café**

Petit café

Cela peut être une bonne idée de se focaliser uniquement sur les statistiques. Les statistiques sont un indice bon indicateur. Cependant, d'expérience rien de tel que de regarder de visualiser la donnée. Avec la donnée, on voit exactement ce qui se passe dans un pipeline complexe et cela extrêmement rapidement. Quand il y a un souci il faut toujours revenir au cœur de du problème : là donnée.

**Slide 58 Addition**

Passons à l'addition

Il ne faut pas attendre qu’on vienne vers nous pour commencer à faire attention si le projet coûte trop cher. La planète a des ressources limitées, autant tout de suite penser au fait de de consommer le moins possible de ressources et d'être le plus efficient possible.

Si votre projet coute trop chère, il ne risque pas de vivre longtemps.

**Slide 59 : Sommaire 4ème partie**

Avec ces expériences, nous avons appris et maintenant on peut essayer de se mettre sur le bon chemin et de mettre toutes les chances de succès de notre côté.

**Slide 60 : Expérimentation**

On va repartir de la phase d’explorations. Ce qui est important dès le début dans l’organisation du projet, c’est de faire travailler tous les acteurs ensemble avec un seul objectif : que cela fonctionne en production.

Humainement, de bien comprendre que les problématiques des uns sont les problématiques des autres pour que cela fonctionne. C’est vraiment cela le cœur du succès d’un projet ou non. C’est aussi pour cela que le titre de la présentation s’appelle le ML OPS est une aventure humaine, c'est parce que c'est vraiment une culture de partage associé au fait de se faire confiance.

Cette phase d'exploration va permettre d'estimer si le projet est faisable.

**Slide 61 : Expérimentation Set up**

Chez AXA France nous travaillons pour les problématiques IA directement avec les données de production. Pour sécuriser cela, on va demander à Khalid notre Ops de déployer un environnement de développement. Seulement quelques personnes auront accès sur cet environnement de travaille sur qui est sur AzureML.

**Slide 62 : Expérimentation Élisabeth et Hicham**

Dès le début Élisabeth et Hicham vont travailler ensemble depuis GIT et un DataLake. Ce sera principalement Élisabeth qui sera active sur la première phase mais accompagné par Hicham qui connait les problématiques de prod uction.

**Slide 63 : Expérimentation Annotation**

Assez rapidement on va devoir commencer des phases d’annotation. On va retrouver Lilian. C'est très important de travailler en collaboration avec l’équipe qui va à annoter les données. Pourquoi ? C’est vraiment la données le trésor de votre boite.

Ce que l’on va faire c’est qu’on va d’abord faire des petits lots de données. Qui vont nous servir à affiner les règles d’annotation. Il est extrêmement important que l’annotations soit réalisée de manière identique par tout le monde.

Imaginez que vous devez Zoner des chats. Imaginez que vous avez un chat qui est sur une table et que sa queue passe sous la table coupée sur la photo et redevient visible de l'autre côté. Il se peut qu’un annotateur va Zoner le chat sans la queue à l’autre bout de la table et qu’un autre va sélectionner le chat avec les 2 morceaux de queue inclus.

Avec ses données, L’IA ne pourra pas bien apprendre, il est très important de travailler en équipe et d’affiner les règles d’annotation en fonction de chaque cas spécifique que l’on rencontre.

Une fois les règles bien définies, le volume de données a annoté pourra être augmentée.

**Slide 64 : Expérimentation Model**

Une fois que l’on a les annotés les données, on peut entraîner un modèle en ensuite, pour automatiser et gagner du temps, on va mettre en place une CI d’intégration continue qui va nous permettre de pouvoir rejouer l’entraînement automatiquement quasiment de manière industrielle. Le livrable ; ce n’est pas le modèle. C’est code qui est dans GIT plus la donnée qui est stockée/versionnée dans le DataLake.

**Slide 65 : Expérimentation Livrables**

On peut voir que là on déjà une pâte à pizza bien étalée avec tous nos ingrédients et un couteau prêt à l'emploi donc on a vraiment tout ce qu'il faut.

En sortie de cette étape un on doit tous ensemble être capable d'estimer les métriques.

* C'est à dire la qualité de prédiction
* Estimation du coût de l’infrastructure de production
* Estimation du temps de réponse
* On a déjà un code d'entraînement quasi industriel versionné et les données versionnée, c’est notre livrable

**Slide 66 : Industrialisation**

On va pouvoir entrer dans la phase d'industrialisation ou le but va être de développer et de porter en production des API qui héberge les modèles.

**Slide 68 : Industrialisation Docker**

On va mettre en place CI d’intégration continue qui va nous permettre de construire des images dockers qui vont récupérer les modèles dans le registre de modèle puis construire nos API avec le code d’inférence et les pousser dans un docker registry.

**Slide 69 : Industrialisation GitOps**

Chez AXA France, on déploie nos images dockers sur open shift. On va demander à notre Obs de nous construire les environnements développement, recette. Nous livrons en utilisant le GITOPS, c’est-à-dire que on va avoir 4 branches dans git une pour l’environnement de développement, une pour l’environnement de recettes, 2 autres pour la préproduction et la production.

Ce qui va nous permettre de pouvoir déployer nos applications via des manipulation GIT.

**Slide 70 Industrialisation GitOps ml-cli**

Aujourd’hui on a des outils qui nous permettent de valider le bon fonctionnement de l’application dans les environnements. On vérifier que l’on a la même qualité de prédiction qu’en entrainement. On vérifie aussi le temps de réponse et on configure finement sizing des infrastructures.

Une fois que l’environnement de développement est validé. On peut déployer en recette

Pour cela on réalise une PullRequest sur la branche de recette.

**Slide 71 : Industrialisation AzureML Production**

A ce moment-là on va rentrer dans une autre phase du projet. On va demander à notre Ops de créer un nouvel environnement d’entrainement AzureML toujours sur la production. Sur cet environnement on n’aura pas d’accès en écriture on aura seulement accès en lecture seule.

**Slide 72 : Industrialisation Big CI**

On va mettre en place un pipeline qui va permettre de réentraîner toute la chaîne en automatique afin d’être sûr que l’on a aucun déphasage sur chaque étape et reconstruire chaque modèle dans l’ordre puis construire toutes les images dockers que l’on sauvegarde dans un docker registry.

**Slide 73 : Industrialisation GitOps Dev**

On peut maintenant, via des manipulations GIT déploiement sur l’environnement de développement puis valider l’environement.

**Slide 74 Industrialisation GitOps Rec**

Déployer en recette et valider la recette.

**Slide 75 Industrialisation Ready**

À ce stade on a déjà une pizza quasiment précuite.

**Slide 76 Industrialisation GitOps Pré-production**

Par sécurité, nous n’avons pas les droits de livrer en préproduction. Seul l’OPS peut le faire et il va livrer cette pizza précuite en préproduction. On valide l’environnement.

**Slide 77 Industrialisation GitOps Production**

Puis Khalid déploie en production.

**Slide 78 : Deployment & run**

L’API les modèles sont déployée et maintenant il faut l’application vive.

**Slide79 : Deployment & run Monitoring**

Pour le moment on a mis du monitoring fonctionnel pour monitorer le DataDrift en utilisant Prometheus et grafana en plus du monitoring technique.

Ce monitoring nous permet facilement de relier les métriques a des alertes automatiques qui sont remontées à une autre équipe spécialisée dans le suivi de production.

Par la suite, On travaille sur la mise en place de la boucle de feedback et on va plutôt se brancher sur cette boucle de feedback pour mettre en place des métriques et de l’alerting beaucoup plus fins.

**Slide 80 : End**

Aujourd’hui on a 2 façons de de partager du code je dans un même GIT.

Soit aujourd’hui on positionne un répertoire nommée production. Comme en python tout est module, le code d’entrainement utilise ce module appelé production pour réaliser l’inférence pour le scoring.

**Slide 81 : End**

Autre technique qu'on est en train de tester pour partager du code et travailler ensemble, c’est de faire des packages locaux que l’on réutilise en local.

**Slide 82 : Open source**

Le MLOPS n'est pas centré sur des outils, cependant grâce à nos projets on a pu mettre en place des outils et les open sourcer. C'est un plaisir de vous les partager.

Le daily clean permet d’éteindre automatiquement les environnement Kubernetes.  
Ecotag permet de faire de l’annotation et respecter les règles GDPR.  
mlc-li est notre outil de test d’intégration.

**Slide 83 : Conclusion**

Le MLOPS, c’est avant tout une culture un partage. Aujourd’hui on a des projets qui sont au top du top de la complexité et si on ne met pas tout en place pour que les experts travaillent tous ensemble et comprennent que la problématique de l'un est la problématique de l'autre. Cela ne peut pas marcher. La clé du succès vient dans ce partage et ce travail ensemble.