Rapport Technique d'évaluation

Projet Pynomaly

Promotion Data Scientist Avril 2021

Participants :

Nicolas Gislais – Roméo Hatchi

Contexte

Le projet de détection d'anomalies de machines industrielles et de jouets à partir de prise de son est une application de la Data Science très concrète et globale (peut être utilisée dans toutes les industries).

Contexte d'insertion du projet dans votre métier.

Roméo : Je suis professeur agrégé de mathématiques et docteur en mathématiques appliquées. Je souhaitais travailler dans le deep learning car c'est le domaine le plus « récent » et le plus théorique/abstrait. C'était aussi un défi personnel car c'était le sujet le plus difficile et je voulais me prouver à moi-même que malgré ma surdité, je pouvais néanmoins travailler sur des fichiers audio.

Nicolas : Je suis aussi professeur agrégé de mathématiques. J'avais pour objectif d'utiliser mes connaissances avancées en mathématiques pour résoudre un problème de data science. L'enseignement au lycée m'ayant éloigné du travail mathématique de haut niveau, j'avais envie de me replonger dans l'utilisation d'outils mathématiques complexes (transformée de Fourier, réseaux de neurones, méthodes d'évaluation). Résoudre un problème concret et valorisable auprès d'entreprises est aussi une raison pour laquelle j'ai demandé à travailler sur le sujet.

Du point de vue technique :

Le premier obstacle concret a été de faire appréhender le sens auditif à Roméo : tenter de vulgariser les notions de son et de fréquences entre autres. Pour nous aider à résoudre cela et à comprendre les outils mathématique principaux dans l'analyse de données audio (la transformée de Fourier, et ses variantes rapides et à court terme), nous avons trouvé un article bien utile écrit par Pierre Chainais : *De la transformée de Fourier à l'analyse temps-fréquence bivariée* [5]. Il y est notamment présenté un parallèle entre les partitions musicales et les spectrogrammes.
Une autre problématique technique a été la gestion de mémoire et de stockage des données. Le traitement de données audio aboutit rapidement à l'utilisation de fichiers de plusieurs gigaoctets ou de variables de dimensions conséquentes. Un exemple d'erreurs que l'on a vues :

```
MemoryError: Unable to allocate 6.41 GiB for an array with shape (210000, 32, 128, 1) and data type float64
```

C'est d'ailleurs une problématique que l'on n'a pas pu résoudre définitivement.
Une problématique corollaire est celle de la puissance de calcul. L'entraînement des modèles à plusieurs millions de poids requiert un minimum de puissance de calcul si l'on ne souhaite pas avoir des temps de calculs considérables.

Du point de vue économique :

L'intérêt économique d'un tel projet est facilement décelable : création d'un test qualité, recherche de défauts sur une chaîne de production, etc. L'application industrielle d'une telle technologie reste tout de même encore complexe et n'était pas notre objectif au vu de nos moyens actuels.

Du point de vue scientifique :

La recherche en data science dans le domaine de l'audio se développe beaucoup, au vu de l'intérêt important que peut apporter le machine learning (et a fortiori le deep learning). Nous avons notamment trouvé un article faisant état de l'art : *Trends in audio signal feature extraction methods [6].*

Objectifs

Quels sont les principaux objectifs à atteindre ? Décrivez en quelques lignes.

Notre principal objectif est d'identifier les machines défaillantes, à partir de prises des sons stockées sous la forme de fichiers wav. Pour cela, nous entraînons un modèle à partir de prises de sons sur des machines sans défauts. La raison pour laquelle il fallait entraîner les modèles uniquement sur des sons « normaux » est détaillée dans la partie Projet. Le modèle doit ensuite être capable de réussir à classer un son entre les classes « Normal » et « Anomaly », à travers une analyse numérique des données. L'objectif secondaire est d'arriver à produire un modèle dont les scores de classification sont les plus élevés possibles.

Pour chacun des membres du groupe, préciser le niveau d'expertise autour de la problématique adressée ?

Roméo : je n'avais aucune expertise dans ce domaine.
Nicolas : outre avoir un peu travaillé sur la transformée de Fourier (c'était notamment une partie d'un des sujets de l'agrégation que j'ai passée), je n'avais aucune expertise dans ce domaine.

Êtes vous entré en contact avec des experts métiers pour affiner la problématique et les modèles sous-jacents ? Si oui, détaillez l'apport de ces interactions.

Non.

Avez vous connaissance d'un projet similaire au sein de votre entreprise, ou bien dans votre entourage ? Quel est son état d'avancement ? En quoi vous a-t-il aidé dans la réalisation de votre projet ? En quoi votre projet contribue-t-il à l'améliorer ?

Cette problématique est issue du *DCASE 2020 Challenge Task 2: Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring [3]*. L'article initial, écrit par les organisateurs, présentait les enjeux et leurs propres résultats obtenus. 40 équipes y ont participé et il y a eu au total 117 soumissions. La plupart ont réussi à faire mieux que l'équipe d'organisateurs.
Nous avons en particulier étudié les travaux d'une équipe portugaise de chercheurs de l'université de Minho et de l'institut CCG ZGDV, à Guimaraes. Ils ont écrit un article à ce sujet : *Deep dense and convolutional autoencoders for unsupervised anomaly detection in machine condition sounds [4]*.
Nos codes se sont inspirés des leurs, notamment en ce qui concerne la sélection de features et de la structure générale des modèles de deep learning.

Data

Cadre

Quel(s) jeu(x) de donnée(s) avez vous utilisé pour atteindre les objectifs de votre projet ?

Nous avions des données sonores de 6 machines différentes : Slider, Fan, Pump, ToyCar, ToyConveyor et Valve. Pour l'entraînement, nous n'avions que des données « normales », c'est-à-dire de machines fonctionnelles. Pour l'évaluation des modèles, des données normales étaient mélangées à celles issues de machines défaillantes (labellisées « anomaly »).

Ces données sont elles disponibles librement ? Dans le cas contraire, qui est le propriétaire de la donnée ?

Oui, elles sont disponibles sur Kaggle, à cette adresse :
https://www.kaggle.com/daisukelab/dc2020task2
Elles ont été produites pas des équipes japonaises. Les processus de production des données sont détaillés dans les sources [1] et [2].

Décrivez la volumétrie de votre jeu de données ?

Nous avons 30987 fichiers audio, chacun durant 10 secondes (ou 11 secondes uniquement pour la machine ToyCar), pour un total d'environ 9,5 Go de données brutes.
Pour chaque type de machine, il y a entre 4094 et 6509 fichiers audio, répartis inégalement entre les jeux de d'entraînement et de test.

Pertinence

Avez vous eu à nettoyer et à traiter les données ? Si oui, décrivez votre processus de traitement.

Les équipes qui ont produit les données ont livré un dataset parfaitement référencé et nettoyé. Il n'y a aucun fichier en double ou inutilisable.
Pour pouvoir utiliser facilement ces nombreux fichiers, il a tout de même fallu générer un DataFrame de référencement des chemins d'accès vers chacun des fichiers. Cela a été l'occasion d'y associer des méta-données élémentaires :

- type de machine
- ID de la machine
- jeu de données (train ou test)
- ID de la prise de son
- le label cible (normal ou anomaly)
- la fréquence d'échantillonnage
- le nombre d'échantillons

Pour extraire les deux dernières méta-données, et plus généralement pour manipuler les données brutes, nous avons utilisé la bibliothèque Librosa.

Nous avons d'abord vérifié si les deux méta-données extraites étaient utiles pour la détection d'anomalie. La fréquence d'échantillonnage est la même pour tous les fichiers (16 kHz). Ensuite, nous

nous sommes assurés que conformément à ce qui a été annoncé par les organisateurs, les fichiers avaient tous la même durée. C'est donc avec le nombre d'échantillons que l'on a fait cette vérification. Toutes les machines ont des prises de son longues de 10 secondes, excepté celles de ToyCar, qui durent 11 secondes. Comme ces deux variables étaient identiques pour toutes les données d'un même type de machine, nous n'avons pas pu les utiliser.
Au regard du nombre élevé de données à analyser, nous avons séparé les données d'entraînement en 6 jeux, un par type de machine (la variable Machine_Type dans le DataFrame initial).
Nous avons ensuite fait l'évaluation des modèles au sein d'un jeu suivant la variable Machine_ID. En effet, pour chaque type de machine, il y a plusieurs machines dont les sons ont été acquis, chacune identifiée par leur Machine_ID. Il y en a trois ou quatre pour chaque type. On peut le résumer comme suit :

```
machine_types = ["fan","pump","valve","slider","ToyCar","ToyConveyor"]
IDs = {"fan":[0,2,4,6],
       "pump":[0,2,4,6],
       "valve":[0,2,4,6],
       "slider":[0,2,4,6],
       "ToyCar":[1,2,3,4],
       "ToyConveyor":[1,2,3]}
```

Enfin, nous souhaitions réussir à déterminer le label cible : « normal » ou « anomaly ». Cette information est donnée par la variable 'Status'.
En résumé, lors de l'évaluation, pour chaque type de machine, nous n'utilisons que deux variables, 'Machine_ID' et 'Status'.

Quelles variables vous semblent les plus pertinentes au regard de vos objectifs ?

Mis à part les données brutes audio, rien ne semblait utile pour la détection d'anomalie. C'est par la transformation de ces données que des informations pertinentes allaient être extraites pour les modèles.

Quelle est la variable cible ?

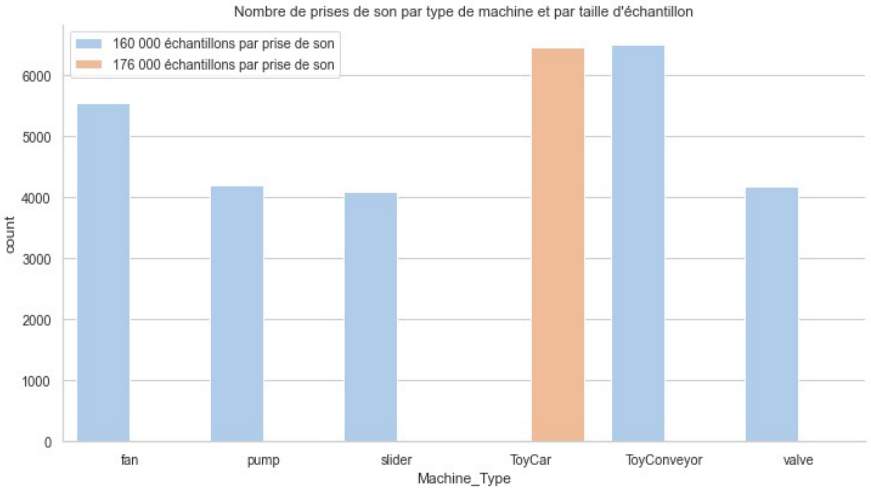
Il est important de faire la différence entre la variable cible de notre projet (et du challenge), et la variable cible des modèles de deep learning que l'on a utilisés.

La variable cible du projet, est la variable 'Status' dans le DataFrame de référencement. Elle traduit si la prise de son observée provient d'une machine sans défaut ou non. Les classes respectives sont « normal » et « anomaly ».

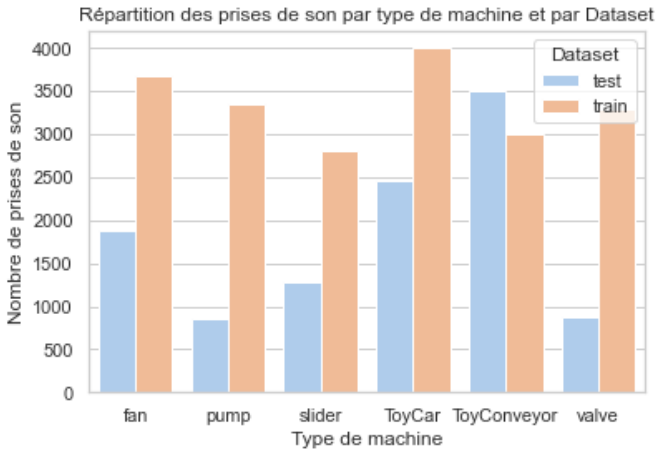
Pour les modèles auto-encoder que nous avons utilisés, leur but est d'avoir un vecteur de sortie le plus identique possible à celui d'entrée. Ainsi, les modèles auto-encodeurs n'utilisent qu'une seule variable, qui est à la fois le « X » (variable explicative) et le « y » (variable cible). Nous détaillerons pourquoi c'est ainsi dans la partie Projet.

Décrivez la distribution de ses valeurs ?

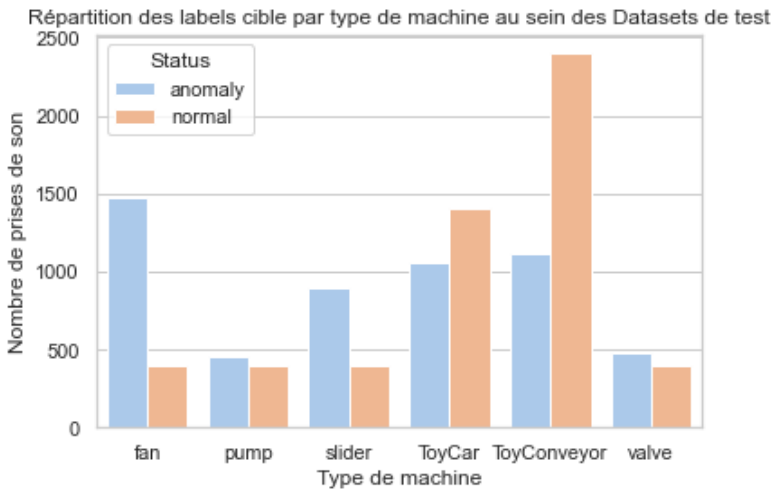
Nous avons six types de machines et les données sont réparties ainsi :



-ToyConveyor : 6509 prises de sons (3000 pour l'entraînement et 3509 pour l'évaluation)
-ToyCar : 6459 prises de sons (4000 pour l'entraînement et 2459 pour l'évaluation)
-fan : 5550 prises de sons (3675 pour l'entraînement et 1875 pour l'évaluation)
-pump : 4205 prises de sons (3349 pour l'entraînement et 856 pour l'évaluation)
-valve : 4170 prises de sons (3291 pour l'entraînement et 879 pour l'évaluation)
-slider : 4094 prises de sons (2804 pour l'entraînement et 1290 pour l'évaluation)



Comme nous l'avons déjà dit, toutes les données d'entraînement sont labellisées normales. Au sein des jeux de test, la répartition entre données normales et anormales est donnée par le graphique suivant :

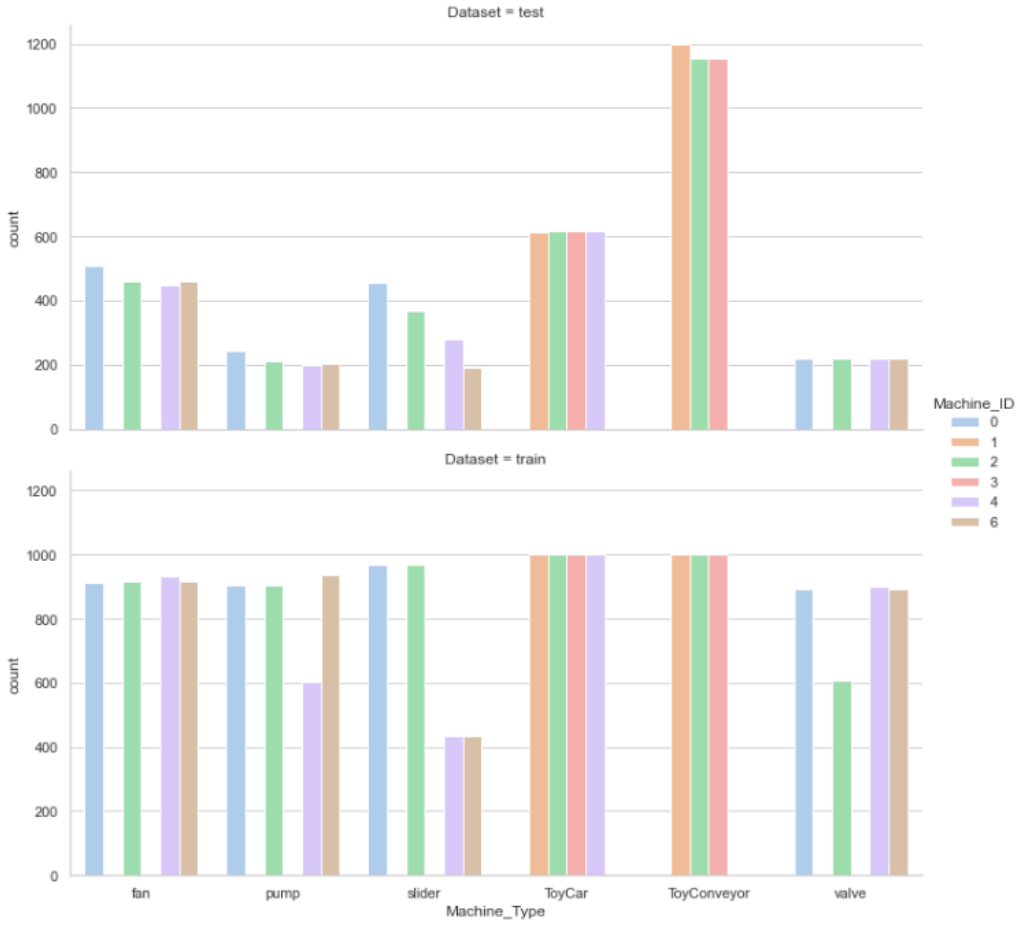


Quelles particularités de votre jeu de données pouvez-vous mettre en avant ?

La répartition du nombre de prises de son est inégale entre :

- chaque type de machines
- chaque machine ID
- les labels cibles
- chaque dataset d'entraînement et de test

L'entraînement et l'évaluation des modèles se fera donc sur un nombre de données assez variable suivant le type de machines.



Projet

Classification du problème

À quel type de problème de machine learning votre projet s'apparente-t-il ? (Classification, régression, clustering ….)

C'est un problème de classification, et plus précisément de la détection d'anomalie : un domaine qui englobe la détection de fraude ou de spam par exemple. Nous devons réussir à reconnaître les données anormales des données normales, afin de déterminer quelles sont les machines défaillantes.

Quelle est la métrique de performance principale utilisée pour comparer vos modèles ?

Nous avons utilisé l'AUC et la pAUC.
L'AUC est l'aire sous la courbe ROC.
La pAUC nous permettait de vérifier que le modèle ne classait pas trop de données normales parmi les anormales et de ne pas donner trop souvent l'alerte à tort. Nous voulions un modèle fiable, en lequel avoir confiance.

Avez vous utilisé d'autres métriques de performances qualitative ou quantitative ? Si oui, détaillez.

Nous avons utilisé le f1-score des 2 classes de la variable cible, en les sommant, pour déterminer le seuil d'erreur à appliquer pour classer les éléments.

Choix du modèle & Optimisation

Quels algorithmes avez vous essayé ?

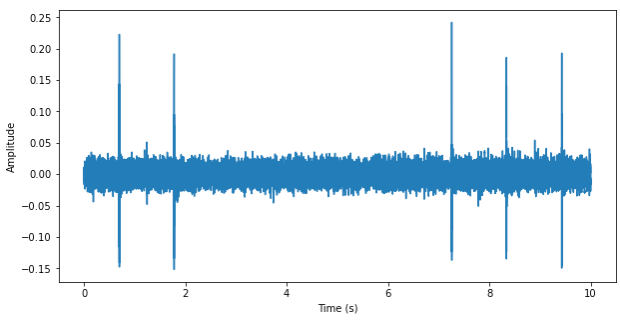
Les modèles classiques d'apprentissage supervisé étaient inadaptés pour notre projet car dans le jeu d'entraînement, nous n'avions que des données d'un même label (normales). Par conséquent, si nous appliquions l'un de ces modèles, il se contentait de classer toutes les données de la même manière, en leur attribuant le label « normal ».
Nous avons néanmoins essayé les algorithmes de régression logistique et de SVM (Support Vector Machine) mais les résultats étaient déplorables et c'était même moins bon que l'aléatoire, dès que l'on avait plus de données anormales que de normales.
Par conséquent, nous devions utiliser des modèles d'apprentissage non-supervisé, par exemple de deep learning. Le choix étant très vaste, nous avons choisi d'utiliser les modèles d'auto-encodeur dense et de convolution proposés dans l'article de l'équipe portugaise [4].

Décrivez celui / ceux que vous avez retenu et pourquoi ?

L'enjeu est de détecter une anomalie de manière *non-supervisée*. En effet, il n'est pas possible de créer un jeu d'entraînement qui englobe toutes les anomalies possibles pour permettre à un modèle d'apprendre à les détecter. On ne peut pas lister toutes les anomalies de sons d'une machine. Cela vient du fait, quasiment évident, que l'on ne peut connaître toutes les anomalies d'une machine, et encore moins leur effet sur le son qu'elles produisent.
Il faut donc entraîner le modèle à produire une représentation du « profil » normal du son d'une machine et ensuite qu'il mette en avant « l'éloignement » d'un son d'une machine défaillante par rapport à ce profil généré.
Ainsi, le travail des modèles de deep learning que l'on a utilisés ne consiste pas à classifier directement les fichiers sons au sein des classes « normal » et « anomaly ».

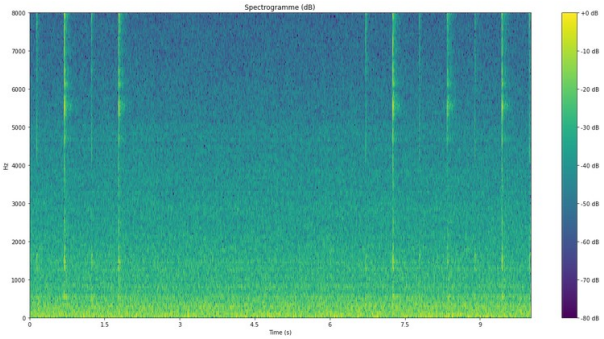
Voici comment se déroule le processus entier :

- Tout d'abord il faut appliquer une série de transformations sur des données brutes, pour obtenir des données numériques exploitables par le modèle. Le preprocessing que l'on a retenu transforme chaque fichier audio en un vecteur de features : un spectrogramme, qui est le résultat de l'application d'une transformée de Fourier à court terme (dans sa version discrète) au signal audio numérisé.



Signal audio numérisé

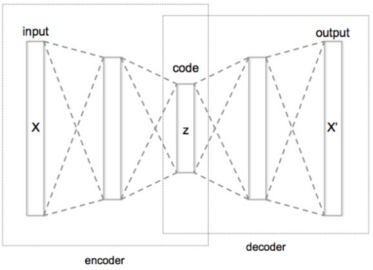
Preprocessing



Spectrogramme

- C'est ensuite au modèle d'entrer en jeu. Les modèles de deep learning que nous avons utilisés sont des auto-encodeurs. Leur principe est d'une part d'encoder le vecteur d'entrée, c'est-à-dire de générer une représentation de ce vecteur (le « compresser » en quelque sorte). Puis,

d’autre part, de le décoder afin de reconstruire le vecteur d’origine.
L’entraînement du modèle se fera sur des vecteurs de features provenant *uniquement* de sons « normaux ». Ainsi, le modèle sera très adapté à reconstruire ce type de vecteurs, et – on l’espère – beaucoup moins pour des vecteurs de features provenant de sons de machines défaillantes.



Structure schématique d'un auto-encodeur

- Enfin, lors d’une prédiction du modèle, on comparera l’erreur entre un vecteur d’entrée et de sortie. Si l’erreur dépasse un certain seuil (à définir), on pourra considérer que le son comporte une anomalie et sera classé en tant que tel.

Nous avons construit deux modèles : Dense (avec principalement des couches Dense) et Convolutif (contenant essentiellement des couches de convolution). Tous les deux consistent en une combinaison d'un réseau d'encodeurs (Encoder) et d'un réseau de décodeurs (Decoder).
Pour chacun de ces modèles, nous avons fait une sélection de features spécifique car les dimensions d’entrée des deux modèles étaient différentes : (640,) pour Dense et (32,128,1) pour Convolutif.

Qu’est ce qui a engendré une amélioration significative de vos performances ?

Une feature selection la plus adaptée possible. Celle qu’a tenté Nicolas n’a pas permis d’obtenir des résultats vraiment plus probants que les modèles naïfs essayés en début de projet.

Avez vous analysé les erreurs de votre modèle ?

Il nous a paru très compliqué d’expliquer la performance d’un modèle à partir de l’analyse de ses erreurs, vu la nature des données.

Description des travaux réalisés

Répartition de l’effort sur la durée et dans l’équipe

Morceler votre projet en un maximum de tâches unitaires. Produisez le diagramme de Gant a posteriori en spécifiant qui s’est occupé de quelle tâche et à quelle moment. (joindre le diagramme en annexe du rapport)

-Lecture d'articles et compréhension des problématique du sujet : Nicolas et Roméo
-Construction du dataframe avec les chemins et les informations principales : Nicolas
-Première exploration des données et analyse statistique : Nicolas et Roméo
-Spectrogrammes : Nicolas
-Construction des modèles de deep learning : Roméo
-Sélection de features : Roméo
(Nicolas avait essayé de son côté pour le modèle Dense, sans succès)
-Mise au propre des fichiers .py et des notebooks : surtout Nicolas
-Entraînements et résultats du modèle Dense : Nicolas
-Entraînements et résultats du modèle Convolutif: Roméo

Bibliographie

Sur quels éléments bibliographiques (articles de recherches, blog, livres, etc...) vous êtes vous appuyé pour réaliser votre projet ?

Rapports de productions des données :
[1] MIMII Dataset: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection – Harsh Purohit, Ryo Tanabe, Kenji Ichige, Takashi Endo,Yuki Nikaido, Kaori Suefusa, and Yohei Kawaguchi – *2019*
[2] ToyADMOS: A Dataset of Miniature-Machine Operating Sounds for Anomalous Sound Detection – Yuma Koizumi, Shoichiro Saito, Hisashi Uematsu, Noboru Harada and Keisuke Imoto – 2019

Sujet du challenge sur lequel se base le projet :
[3] DCASE 2020 Challenge Task 2: Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring

Rapport de production d’une solution au challenge :
[4] Deep Dense and Convolutional Autoencoders for Unsupervised Anomaly Detection in Machine Condition Sounds – Alexandrine Ribeiro Luís Miguel Matos Pedro José Pereira Eduardo C. Nunes – 2020

Sources scientifiques sur l’extraction de features à partir de son :
[5] De la transformée de Fourier à l’analyse temps-fréquence bivariée – Pierre Chainais – 2018
[6] Trends in audio signal feature extraction methods – Garima Sharma, Kartikeyan Umapathy, Sridhar Krishnan – *2020*

Sources pour le développement en Python :
- site officiel de *Python*
- sites officiels des packages utilisés *Scikit-learn*, *Keras*, *Tensorflow*, *Librosa*

Difficultés rencontrées lors du projet

Quel a été le principal verrou scientifique rencontré lors de ce projet ?

La sélection de features a été très compliquée à maîtriser. Comprendre les subtilités du traitement du signal audio et ensuite l'utiliser pour extraire des données suffisamment discriminantes (qui apporteront suffisamment d'informations au modèle pour qu'il puisse faire le bon choix) nous a pris beaucoup de temps.
De même, la construction d'un modèle de deep learning adapté à notre problématique ne s'improvise pas.
Pour surmonter ces deux verrous, nous nous sommes appuyés sur nos sources [4] et [6].

Pour chacun des points suivants, si vous avez rencontré des difficultés, détaillez en quoi elle vous ont ralenti dans la mise en place de votre projet :

Prévisionnel : (tâche qui ont pris plus de temps que prévu etc ….)

Comme dit précédemment, la sélection de features a été très longue à mettre en place. Il fallait d'abord comprendre la théorie qui se cachait derrière le traitement du signal.
De plus, pour Roméo, ce n'était pas aisé à appréhender, compte-tenu de sa surdité. Le son est en effet quelque chose de plutôt abstrait pour lui.
Ensuite, il fallait saisir le principe des MFEC et MFCC et être ensuite en mesure de l'appliquer à nos données.

Nous avons aussi manqué de structuration dans l'avancée du projet. Nous n'avons pas vraiment suivi le calendrier proposé avec les livrables de modélisation et mal équilibré le travail entre les sprints et le projet. Préparer un planning cohérent en début de projet, et s'y tenir, aurait pu nous éviter le crunch très stressant de la dernière semaine avant le rendu.

Jeux de données : (Acquisition, volumétrie, traitement, agrégation etc.…)

Le dataset final d'entraînement, après le preprocessing, est très volumineux. Pendant un temps nous souhaitions le sauvegarder, mais cela générait des fichiers de 10 à 20 Go par type de machine. De quoi saturer rapidement nos disques durs, nous avons donc abandonné cette idée.
De plus, au vu de la taille des données, l'entraînement des modèles et leur évaluation ont été très longs à faire (plusieurs heures par type de machine). Nous avons essayé d'utiliser Google Colab mais la quantité de données à traiter n'a mené qu'à des interruptions de processus.

Compétences technique / théoriques : (Timing d'acquisition des compétences, compétence non proposée en formation etc…)

Ce projet nécessitait des connaissances en deep learning. Or, nous sommes des débutants, qui n'avions aucune compétence en Data Sciences au début de la formation, et nous avons été longtemps perdus devant la complexité du projet. C'est seulement lorsque nous avons pu aborder les derniers modules du parcours pédagogique (Keras, Tensorflow, RNN) que nous avons pu avancer. La masterclass Tensorflow, débloquée pour nous dès le début du projet, était très difficile pour nous et il y avait beaucoup de choses que nous ne pouvions comprendre.
Nous avons également eu des difficultés à répartir les efforts entre les modules à valider et le projet.

Pertinence : (de l’approche, du modèle, des données etc …)

Au début, pendant plus d'un mois, nous n'avions pas les connaissances nécessaires pour travailler sur le projet. Nous travaillions sans orientation claire des efforts, ce qui était assez inefficace et frustrant.

IT : (puissance de stockage, puissance computationnelle, etc….)

Lorsque l'on charge les fichiers audio avec Librosa, on se retrouve avec des vecteurs très grands en taille de longueur 160 000 (ou 176 000 pour ToyCar) !
Ces données étaient difficiles à exploiter en l'état, et l’information contenue dans le signal audio brut n’est pas suffisamment discriminante. Il fallait effectuer une sélection de features. Nous avons construit des modèles qui exigeaient des dimensions précises donc nous devions transformer le vecteur initial en une liste de vecteurs de longueur adaptée.
Par conséquent, si nous souhaitions enregistrer les datasets en local, il fallait prévoir beaucoup de mémoire (plus de 10 Go pour un dataset d’entraînement en général), ce qui a rapidement été impossible.
De par nos configurations hardware et les OS sur lesquels on travaillait, nous n’avons pas pu déléguer les calculs sur nos cartes graphiques. L’entraînement des modèles fut donc extrêmement long, surtout pour le Convolutif. Sur ce modèle, nous n'avons ainsi pu faire que trois epochs pour chaque machine.

Bilan & Suite du projet

En quoi votre projet a-t-il contribué à un accroissement de connaissance scientifique ?

Ce projet nous a permis de travailler tous les aspects du deep learning (sélection des features, construction de modèles, choix des hyperparamètres, analyse des résultats, interprétation graphique...)
Nous sommes désormais plus à l'aise avec les notions de deep learning. Nous avons également appris à chercher en autonomie les informations nécessaires pour avancer dans notre projet.

Pour chacun des objectifs du projet, détaillez en quoi ils ont été atteints ou non.

A partir des travaux de l'équipe portugaise, nous avons réussi à construire deux modèles AutoEncoders performants qui donnent de bons résultats sur la plupart des machine_id. Les performances sont du même ordre que l’équipe. Nous n’avons pas su affiner la feature selection et les modèles pour augmenter les scores AUC et pAUC de manière significative.
Nous avons en revanche manqué de temps pour développer nos propres modèles.

Dans le cas contraire, quelles pistes d’amélioration suggérez-vous pour améliorer les performances de votre modèle ?

Bien que les résultats soient globalement satisfaisants, les modèles sont très longs à entraîner. Nous aurions pu ne sélectionner qu'une partie des jeux d’entraînement et/ou utiliser des outils de Tensorflow (transformation en tenseurs, via la méthode *from_tensor_slices*, utilisation de la classe Dataset, etc.). L’optimisation des calculs est à approfondir.

Nous aurions pu essayer d'autres types de modèles, tels que les GAN (Generative Adversarial Networks) et les AE variationnels mais nous avons manqué de temps.
Nous aurions également pu privilégier une approche de clustering, en déterminant une frontière précise de décision entre les normales et les autres.

Annexes

Diagramme de gant

Voir dans le fichier ‘DiagrammeGantt_Pynomaly.pdf’

Description des fichiers de code

Notre GitHub se trouve à cette adresse : https://github.com/DataScientest/Pynomaly

Bibliothèques nécessaires :

- pandas
- numpy
- matplotlib
- librosa
- tensorflow
- scikit-learn
- sys
- os

DataFrame utilisé pendant tout le projet :
Le fichier .csv.
Le notebook détaillant la construction du dataframe.

Exploration statistique des données et DataViz' :
Le notebook présentant notre exploration et quelques graphiques.

Modèle Dense Auto-encoder :
Le fichier .py contenant toutes les fonctions nécessaires à la modélisation Dense.
Le notebook pour lancer la modélisation et enregistrer tous les résultats.

Modèle Convolutif Auto-encoder :
Le fichier .py contenant toutes les fonctions nécessaires à la modélisation Convolutif.
Le notebook pour lancer la modélisation et enregistrer tous les résultats.

Évaluations :
Le fichier .py contenant les fonctions nécessaires pour afficher les résultats des modélisations.
Le notebook présentant les résultats des évaluations sous forme de tableaux.
Le notebook si on souhaite regarder les résultats détaillés pour chaque Machine_ID d'un type de machine.
Tous les résultats des évaluations sous forme d'un fichier .zip.

Dossier sources :
Tous les articles en .pdf qui nous ont servi pendant ce projet.

Dossier drafts :
Tous nos fichiers d’essais (notebooks, fonctions, …) pour suivre l'évolution de notre travail.

Pour rappel, les fichiers audio sont disponibles sur Kaggle.