Utilisation du LSTM dans la maintenance prédictive d'un compresseur à gaz

SOMMAIRE

1) INTRODUCTION

Le contexte relatif à l'élaboration du projet est la promesse d'une révolution par la maintenance prédictive.

Il a été mis en œuvre lors d'une alternance de 6 mois sein de SPIE ICS à Nîmes, dans le service Incubation comptant 7 personnes.
Cette alternance a été effectuée dans le cadre de la formation développeur DATA/Intelligence Artificielle de 2 ans.
Spie ICS, entreprise de services numériques, est une filiale de Spie France. Elle est spécialisée dans le conseil, l'ingénierie, l'intégration, l'infogérance, la maintenance, les services opérés et le Cloud. Elle compte 3000 employés.

Mes missions au sein de l'entreprise étaient d'utiliser l'intelligence artificielle pour de la maintenance prédictive, tester différentes plateformes IA existantes, établir un système de classement de fichiers.
Plus largement cette alternance était pour moi l'occasion de m'immerger dans le milieu informatique et d'appréhender des méthodes de travail, agiles notamment. Elle avait pour objectif personnel de conforter mon envie de travailler dans le secteur DATA/Intelligence Artificielle.
Pour présenter la maintenance prédictive effectuée sur un compresseur à gaz, situé à Saint-Martin-de-Crau, pour le transport de gaz naturel j'ai divisé mon rapport en 5 parties.
Dans un premier temps, je vais aborder la problématique client et la traduction en objectifs.
Dans un deuxième temps, je vais faire un état de l'art sur les compresseurs à gaz, la maintenance prédictive et algorithmes IA associés, exposer un use case.
Dans un troisième temps, j'aborderai la traduction des spécificités fonctionnelles en spécificités techniques
Dans un quatrième temps, j'établirai un bilan et les axes d'amélioration à apporter au projet.
Enfin, j'effectuerai une conclusion sur la cohérence du travail effectué avec la compétence évaluée et l'apport de mon projet dans la problématique client.

2) LA COMPREHENSION DU BESOIN CLIENT

Le client voudrait une solution optimale pour effectuer une maintenance prédictive de son compresseur à gaz.

Il se pose une question, comment choisir le modèle le plus performant pour la maintenance prédictive d'un compresseur à gaz ?

Les objectifs formulés par le client sont :

- Comparer les résultats obtenus par le Random Forest développé par Spie ICS et un LSTM pour une prédiction de panne sur les 12 prochaines heures.
- Comparer les résultats obtenus par le Random Forest développé par Spie ICS et un LSTM pour une prédiction de temps restant avant panne 30 minutes après arrêt du compresseur.

3) L'ETAT DE L'ART

- **Les machines tournantes :**

Par abus de langage le terme compresseur à gaz a été utilisé tout au long de l'alternance. Le terme adéquat est turbine à gaz (ou encore turbine à gaz de combustion). Elle représente une machine tournante thermodynamique appartenant à la famille des moteurs à combustion interne.

Une turbine à gaz utilise le dioxygène de l'air ambiant comme comburant et le comprime. Par combustion du mélange comburant-carburant il se produit une augmentation brutale de la pression et du volume du mélange gazeux mettant en rotation rapide la turbine (1).

Les éléments constituant une turbine à gaz sont : l'entrée, le compresseur, la chambre de combustion, la turbine, la tuyère (figure 1).

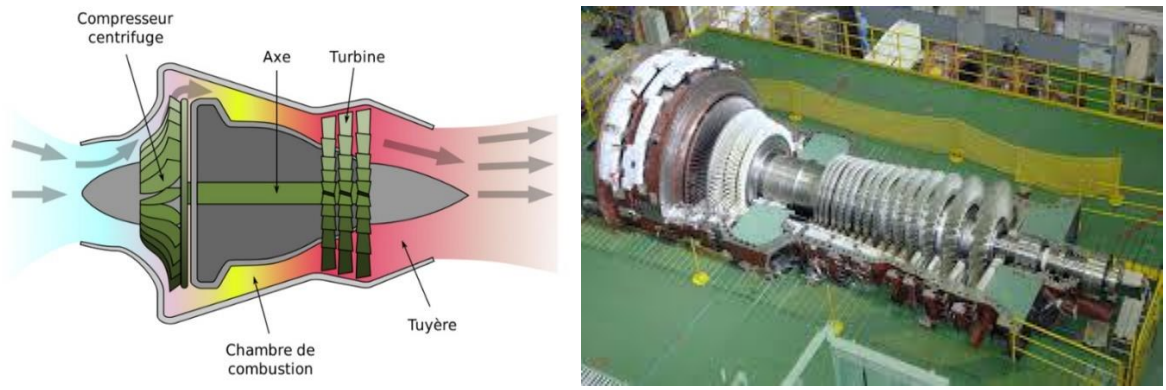


Figure 1 : Images de turbines à gaz

- **La maintenance prédictive :**
  - **Cadre et caractéristiques**

Elle repose sur 4 technologies clés (figure 2).

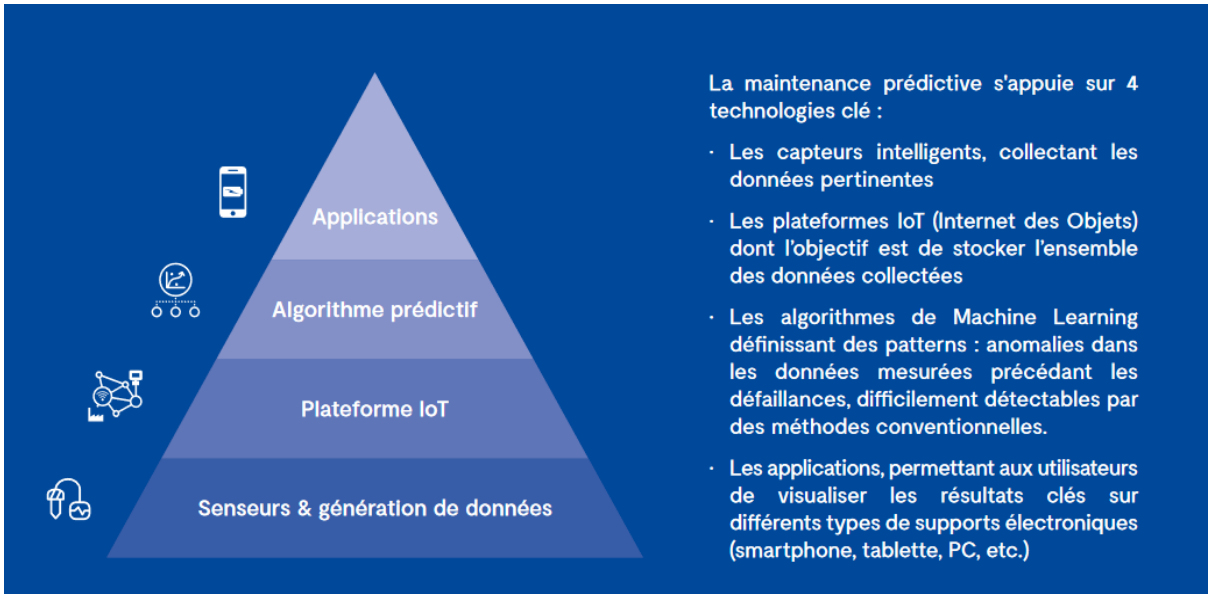


Figure 2 : Les 4 technologies clés de la maintenance prédictive

Le cadre de la maintenance prédictive est décrit dans la figure 3.
Elle utilise des données historiques et effectue une surveillance en temps réel du comportement d'un produit.

	Maintenance prédictive
Quel est son objectif ?	Surveiller les tendances prédisant les circonstances ; prévoir le moment auquel l'équipement pourrait tomber en panne et l'empêcher grâce à des opérations de maintenance
Par qui est-elle effectuée ?	Expert de contrôle, analystes, techniciens de maintenance
A quelle fréquence	Les taches sont effectuées seulement lorsqu'elles sont nécessaires, à des dates prévues
Ressources nécessaires	Equipements pour les méthodes prédictives telles que la thermographie, l'analyse de vibration et autres; outils analytiques (capteurs) et GMAO
Conséquences sur le cycle de production	Un équipement est mis à l'arrêt uniquement avant une panne prédite

Figure 3 : le cadre de la maintenance prédictive (2)

- ## Les enjeux

La maintenance prédictive s'inscrit dans une logique ROIste. En effet, il faut identifier le point limite d'utilisation d'une machine pour rentabiliser au maximum son investissement. Afin d'identifier cette capacité maximale d'utilisation, il faut accepter de laisser tomber en panne des machines. Cela permet aux algorithmes de prédiction d'intégrer le point de rupture de la machine pour mieux l'anticiper par la suite.

Les enjeux économiques sont variables selon le secteur (aéronautique, ferroviaire, automobile, énergétique) mais on peut obtenir selon les études réalisées (3) :

1) 70% de réduction de pannes.
2) 30% de réduction de coûts de maintenance.
3) 50% de réduction des temps d'arrêts non planifiés.

o Le choix des types d'apprentissage automatique et stratégies associées

La maintenance prédictive repose sur des algorithmes supervisés ou non supervisés (figure 4). La principale différence est le besoin d'informations sur les pannes précédentes pour effectuer des prédictions en supervisé contrairement au non supervisé.

	Supervised Learning	Unsupervised Learning
Nécessite des données d'incidents passés	Oui	Non
Capacité à prédire de nouveau types d'incidents	Non	Oui
Difficulté de développement	Modéré	Elevé
Optimisation et maintenance des algorithme	Elevé	Bas

Figure 4 : Comparaison des principales caractéristiques des algorithmes d'apprentissage supervisé et non supervisé

La technique de modélisation choisie dépend du type de problème que l'on essaye de résoudre et des données disponibles (4). La description des stratégies suivantes m'a permis de choisir les plus pertinentes pour la résolution des objectifs clients.

4 stratégies sont possibles :

1) Créer un modèle de régression supervisé pour prédire le temps restant avant la panne (RUL en anglais). Pour cela des données statiques (propriétés mécaniques, usage moyen, environnement machine …) et historiques sont nécessaires.

2) Créer un modèle de classification supervisée pour prédire la panne dans une plage de temps sélectionnée (répond par exemple à la question : Une panne va-t-elle survenir dans 12 heures ?). Pour cela des données statiques et historiques sont nécessaires. Plusieurs types de pannes peuvent être analysés ainsi avec le même algorithme.

3) Signaler un comportement anormal avec un apprentissage non supervisé. Cette stratégie permet d'analyser plusieurs types de pannes mais ne permet pas de savoir si une dégradation de fonctionnement va conduire à une panne, ni dans combien de temps.

Stratégie utile quand les données à analyser sont en faible quantité.

4) Crée un modèle d'analyse de survie pour la prédiction de probabilité de panne dans le temps. Contrairement aux modèles précédents celui-ci n'est pas focalisé sur la prédiction mais sur le processus de dégradation. Ce modèle nécessite des données statiques et la date de survenue de panne. Ce modèle fournit des estimations pour un groupe de machines aux caractéristiques similaires.

o Le choix des modèles d'apprentissage automatique

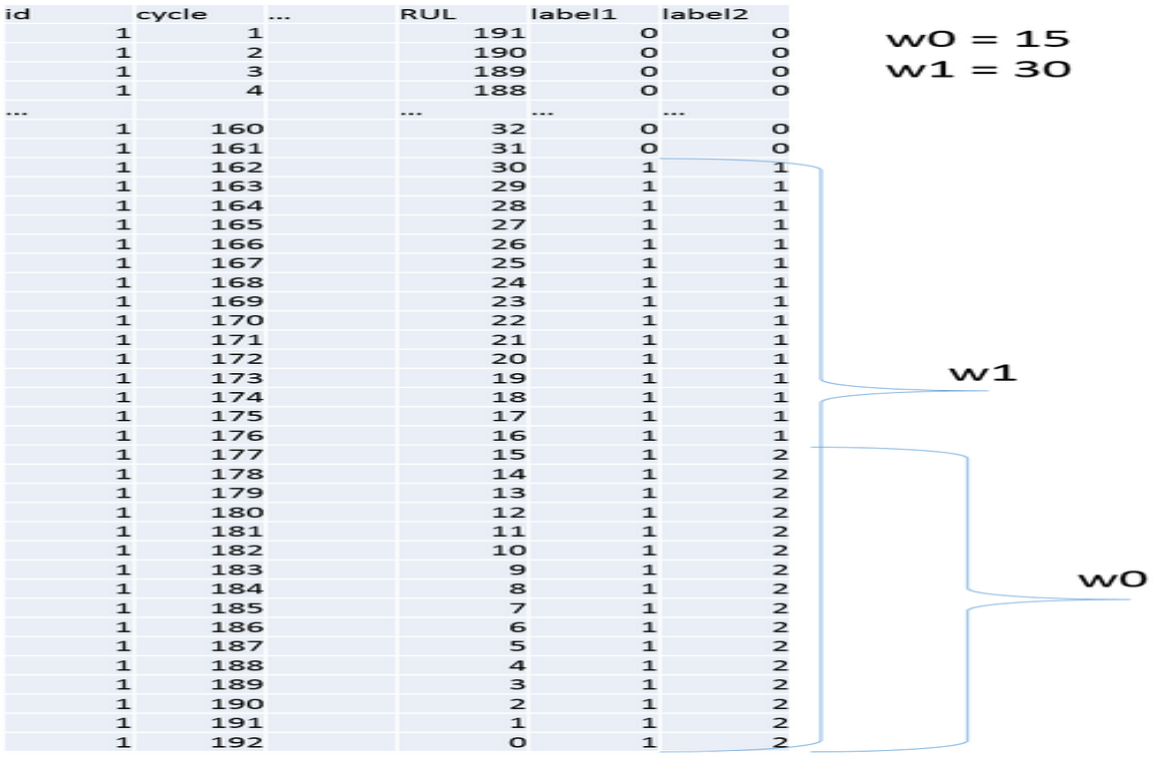
Les résultats de maintenance prédictive de moteurs d'avions à l'aide d'apprentissage automatisé ont été observés (5). En effet, dans un domaine où la précision et la performance sont nécessaires, il est intéressant de connaitre les pratiques. Je me suis donc basé sur les résultats de ce projet pour choisir mes algorithmes. Pour cet use case, plusieurs modèles de régression et classification (binaire ou multiclasses) ont été testés.

La labélisation des données d'entrainement est effectuée selon le protocole suivant (figure 5) :

1) Pour un modèle de régression la cible (target en anglais) est le RUL (Remaining Useful Life). Il reprend le nombre de cycles par ordre décroissant. Le cycle est un pas de temps associé aux enregistrements de données par capteurs. L'unité du cycle est arbitraire, non précisée dans l'use case. Elle peut être l'heure, la minute, la seconde ... Un RUL à 0 signifie « panne ».

2) Pour un modèle de classification binaire, une labélisation 0 ou 1 sur un critère défini à l'avance : une plage temporelle avant panne choisie selon les besoins du client afin de répondre à la question : la machine va-t-elle tomber en panne dans X temps ?

3) Pour un modèle de classification multiclasses, une labélisation 0 à n sur des critères définis à l'avance : des plages temporelles avant panne choisies selon les besoins du client.



id	cycle	...	RUL	label1	label2
1	1		191	0	0
1	2		190	0	0
1	3		189	0	0
1	4		188	0	0
...			...	...	...
1	160		32	0	0
1	161		31	0	0
1	162		30	1	1
1	163		29	1	1
1	164		28	1	1
1	165		27	1	1
1	166		26	1	1
1	167		25	1	1
1	168		24	1	1
1	169		23	1	1
1	170		22	1	1
1	171		21	1	1
1	172		20	1	1
1	173		19	1	1
1	174		18	1	1
1	175		17	1	1
1	176		16	1	1
1	177		15	1	2
1	178		14	1	2
1	179		13	1	2
1	180		12	1	2
1	181		11	1	2
1	182		10	1	2
1	183		9	1	2
1	184		8	1	2
1	185		7	1	2
1	186		6	1	2
1	187		5	1	2
1	188		4	1	2
1	189		3	1	2
1	190		2	1	2
1	191		1	1	2
1	192		0	1	2

- Rul : Remaining useful life (temps restant avant la panne).
- Cycle : pas de temps (heure, minute, seconde …).
- Label 1 : labélisation en classification binaire.
- Rul : Remaining useful life (temps restant avant la panne).
- Cycle : pas de temps (heure, minute, seconde …).
- Label 1 : labélisation en classification binaire.
- Label 2 : Labélisation en classification multiclasses.
- W0 et W1 : plages temporelles.

Figure 5 : labélisation des données en maintenance prédictive pour des algorithmes de classification et régression

Les résultats obtenus sont les suivants :

1) Concernant la régression, les arbres de décision sont les plus performants avec un RMSE (Root Mean Squared Error) d'environ 30 contre 86 pour les réseaux de neurones.

2) Concernant la classification binaire, les réseaux de neurones sont les plus performants avec un F1 score de 0.87 contre 0.81 pour les arbres de décision et la régression logistique.

3) Concernant la classification multiclasses, les réseaux de neurones sont plus performants avec une « accuracy » de 0.92 % contre 0.88% pour la régression logistique.

- *LE LSTM*

Sur la base des informations fournies dans le chapitre précédent le LSTM a été choisi.

C'est un réseau de neurones particulier, un système informatique s'inspirant du fonctionnement du cerveau humain**.**

Le plus simple réseau de neurones est le perceptron. Il est caractérisé par des variables avec leur poids, une fonction de combinaison et une fonction d'activation (figure 6).

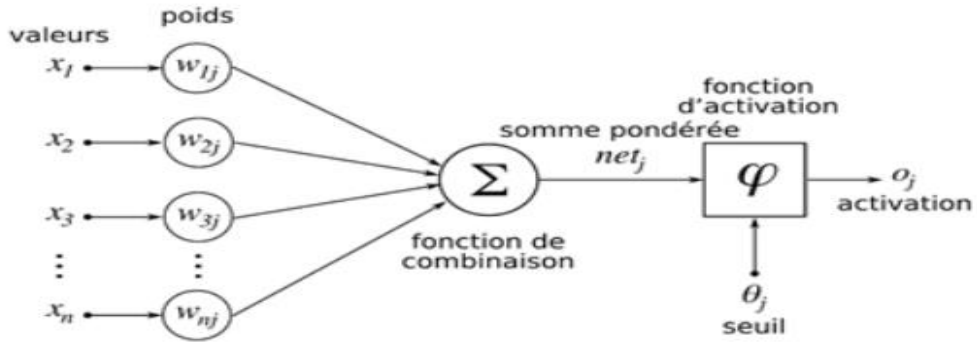


Figure 6 : structure d'un perceptron

Les réseaux de neurones contiennent trois couches (6) :

1) La couche d'entrée avec les données sous forme de tenseur.

2) Les couches cachées qui contiennent les valeurs intermédiaires calculées lors d'un entrainement du réseau.

3) La couche de sortie avec les neurones contenant les résultats d'objectifs fixés au préalable.

Ils utilisent des données d'entrainement pour « apprendre » et faire des prédictions.

Le LSTM est le réseau de neurones récurrent le plus connu. Il permet de gérer des séquences temporelles contrairement au réseau de neurones classique. En effet, sa structure introduit un mécanisme de mémoire des entrées précédentes qui persiste dans les états internes du réseau et peut ainsi impacter toute sortie future.

Un block LSTM contient (figure 7) (7) :

1) Une porte d'oubli.
2) Une porte d'entrée.
3) Une porte de sortie.

La porte d'oubli filtre les informations contenues dans la cellule mémoire précédente.
La porte d'entrée décide quelles nouvelles informations peuvent être mises en mémoire.
La porte de sortie fournit des résultats pour le bloc suivant et pour l'utilisateur en fonction des informations en mémoire et des nouvelles informations entrantes.
Les portes d'entrée utilisent des fonctions d'activation (sigmoïde, tangente hyperbolique) pour faire passer ou non les informations.

Pour effectuer une prédiction de séries temporelles, il faut introduire la chronologie en assemblant plusieurs blocks LSTM.

- Forget gate (porte d'oubli)
- Input gate (porte d'entrée)
- Output gate (porte de sortie)
- Hidden state (état caché)
- Cell state (état de la cellule)

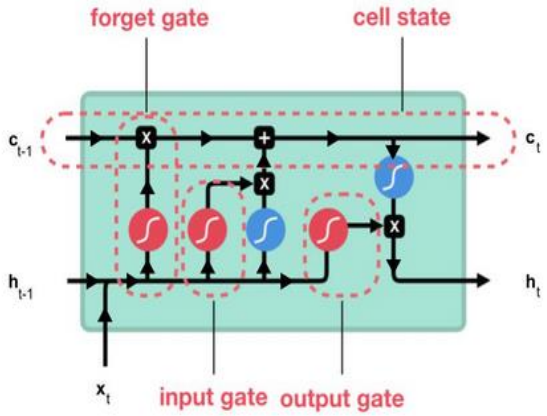


Figure 7 : Structure d'un block LSTM

Les principaux paramètres et hyper paramètres du LSTM sont les suivants :

1) le look back ou nombre de pas de temps (timestep),
2) le « batch size » ou nombre de données entrainées par étape d'entrainement,
3) le nombre d'époch ou nombre d'entrainements de l'algorithme effectué sur les données,
4) le nombre de couches cachées et le nombre de nodes (nombre de neurones cachées) pouvant être sélectionnés par une équation (8),
5) les fonctions de perte et d'activation. La fonction de perte la plus connue est la descente de gradient.
6) Le learning rate contrôlant l'ajustement des poids des neurones du modèle par respect du gradient de perte (« loss gradient »). Le learning rate affecte la vitesse d'atteinte d'un minimum local de perte par back-propagation (9).
7) Le dropout afin d'ignorer certains neurones lors de la phase d'entrainement et prévenir l'overfitting (10).

4) LA TRADUCTION TECHNIQUE ET CHOIX TECHNIQUES DU PROJET

A partir des objectifs client et personnels j'ai établi un workflow en 5 étapes (figure 8).

Pour chaque étape j'ai effectué une traduction des spécifications fonctionnelles en spécifications techniques (démarches et procédés incluant les technologies).

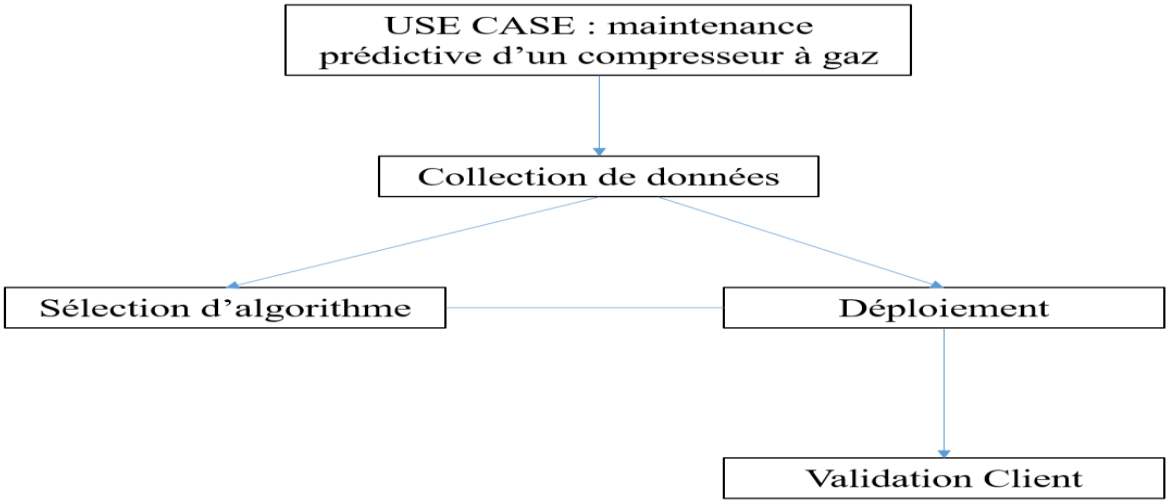


Figure 8 : Workflow du projet

Sur l'ensemble du projet, j'ai codé en Python grâce à Jupiter notebook et Visual Studio 2019. Ce dernier fournit plusieurs fonctionnalités python (interpréteur du langage, la gestion de projets en flask ou django, le switch rapide entre environnement). Python a été choisi parce qu'il est plutôt facile d'utilisation et d'apprentissage (pratiqué en formation) et qu'il possède de nombreuses librairies puissantes pour le machine learning (pandas, Scikit-learn, TensorFlow).

- ## Les spécifications techniques du use case

La première étape consiste à définir le cadre dans lequel s'inscrit la problématique client. Je l'ai l'obtenu en réalisant la veille technologique (chapitre précédent) sur le web et plusieurs réunions client. Après discussion avec le client j'ai fixé la démarche à mettre en place : une « preuve de concept » ou « Proof of conCept » (PoC) en anglais afin de valider la possibilité de pouvoir prédire des pannes de compresseur à gaz et d'identifier les variables corrélées. Puis une démarche de « preuve de valeur » ou « Proof of Value » (PoV) en anglais a été instaurée afin de répondre aux enjeux client et d'atteindre un niveau de performance de prédictions permettant de réduire les coûts de maintenance des compresseurs, d'augmenter les performances et la longévité des équipements.

- ## Les spécifications techniques de la collecte de données

La deuxième étape consiste à collecter les données afin de pouvoir les explorer et les manipuler pour fournir des résultats chiffrés permettant de répondre aux attentes du client.

o La création d'une base de données

Les données sont enregistrées par des capteurs machine et fournies par le client sous forme de fichiers csv (données historiques). J'ai donc décidé de créer une base de données (BDD) en local puis de la stocker sur un cloud dans l'objectif de présenter des données sous forme de dashboard dynamique.
Par ailleurs, les collections peuvent être utiles dans l'optique d'améliorer le projet. En effet, elles sont très utilisées dans le « Big Data » et l'IOT (Internet of Things), pour la collecte de données transportées. De plus, avec l'augmentation de la taille des données récoltées par capteurs, le cloud ou une plateforme personnelle peuvent rapidement devenir une solution de stockage. Enfin, la base de données permettrait de traiter le temps réel (11).

J'ai donc d'abord créé et rempli une base de données relationnelle SQLite à partir des fichiers csv fournis et à l'aide de la bibliothèque SQLite3 et des fonctions associées. J'ai choisi ce système de gestion afin d'obtenir une base de données relationnelle, simple d'utilisation, compatible avec python, ne nécessitant pas l'installation d'un logiciel, permettant la création de grandes BDD (maximum de 140 téraoctets), pour finaliser un projet web plutôt simple sans un accès prévu à plusieurs clients ou sans requêtes complexes.
La BDD a ensuite été convertie en MySQL à l'aide de l'outil « ESF Database Migration Toolkit » afin de pouvoir stocker un duplicat dans un « bucket » sur google cloud storage puis l'exporter dans une instance Google Cloud SQL et la connecter à Google Data Studio.
Par soucis de sécurisation des informations seul les 50000 premiers enregistrements de la base de données ont été dupliqués sur le Cloud.
Les clés étrangères de la BDD ont été définies avec MySQL Workbench. Une relation un à un a été définie entre les tables. Celle-ci est réalisée de la même manière qu'une relation un à plusieurs mais sans définition de hiérarchie parent-enfant. Le modèle Entité-Relation (ou Entité-Association) est visualisable dans la figure 9. Ce schéma traduit la logique suivante : chaque enregistrement est caractérisé par une date et une valeur (numérique ou catégorielle) relevée par les capteurs.

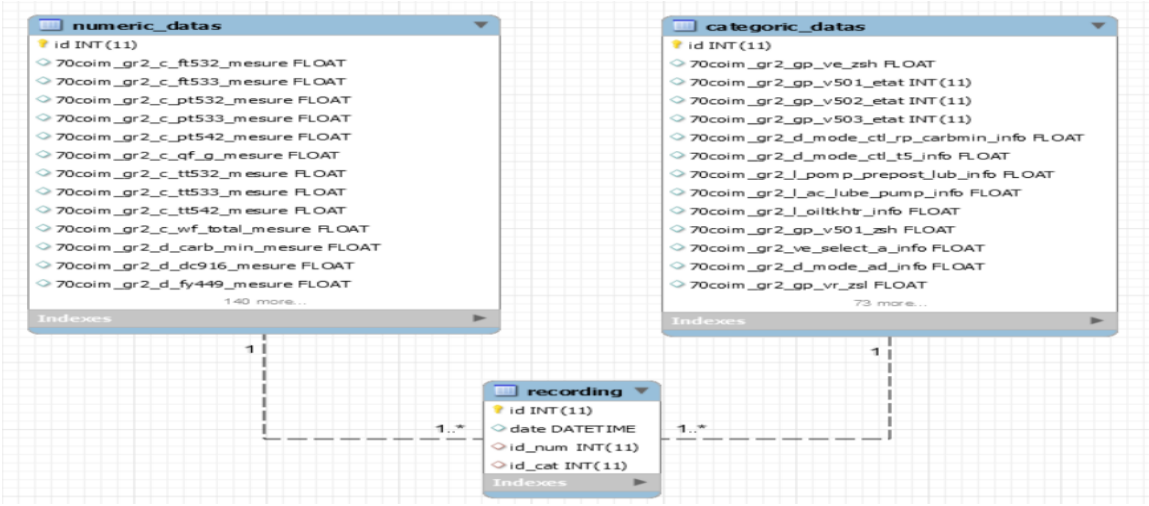


Figure 9 : Schéma Entité-Relation des données du compresseur à gaz obtenu avec MySQL Workbench

o L'exploration, le nettoyage et le prétraitement des données

Les données brutes issues de fichiers CSV stockés en local (239 variables : 153 numériques et 86 booléens*)* sont d'abord importées dans un dataframe sur python grâce à la bibliothèque Pandas. Un algorithme est créé puisque toutes les données Excel ne sont pas au même format et propices à l'importation. Celui-ci est basé sur des fonctions python utilisées sur des dataframes.

Les données pré nettoyées par le client sont ensuite explorées.
L'objectif est de visualiser les plages temporelles de marche-arrêt et les survenues historiques de pannes en utilisant la variable de pression du compresseur identifiée par le client. Pour cela la bibliothèque Matplotlib en python a été utilisée afin d'obtenir une courbe de la mesure d'intérêt (figure 10). A l'arrêt, la pression d'un compresseur devrait rester constante or ici elle diminue jusqu'à 0 bar (panne).

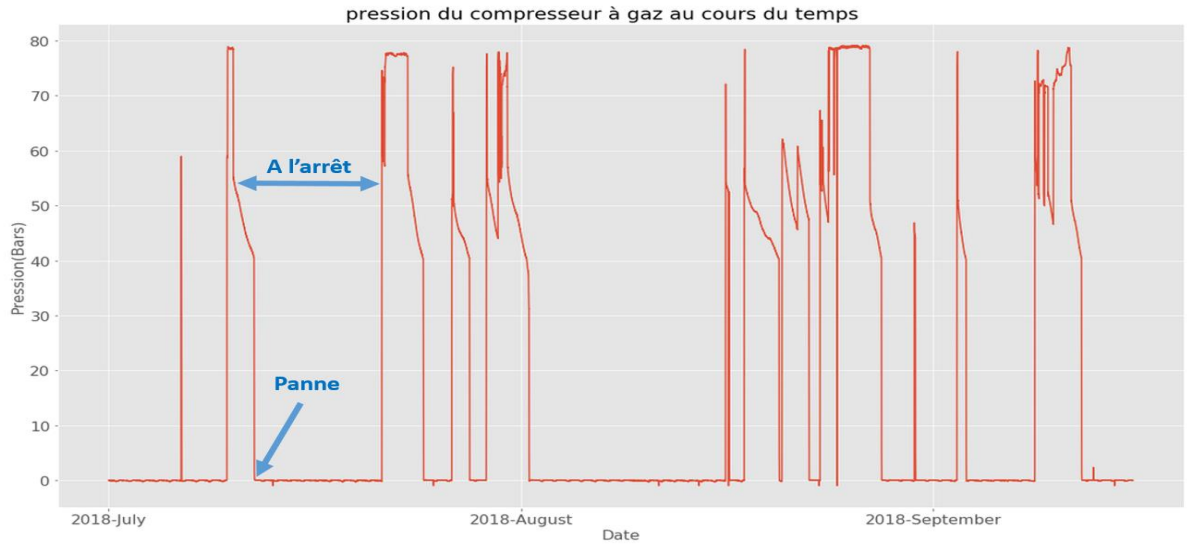


Figure 10 : Visualisation de la variable PT542 (pression du compresseur) au cours du temps

L'obtention de ces informations a permis de découper les données afin d'obtenir plusieurs jeux disponibles pour le machine learning (jeux de données avec ou sans plages temporelles de fonctionnement du compresseur à gaz).
L'obtention de ces informations a également permis de labéliser les données de façons spécifiques et différentes pour des problèmes de classification et régression en maintenance prédictive (voir le chapitre sur « Le choix des modèles d'apprentissage automatique »).
Ces éclaircissements ont également permis de mettre en place une remodélisation automatique des fichiers csv en dataframe contenant seulement les plages d'arrêt machine jusqu'à la dépressurisation du compresseur à gaz. Pour cela un algorithme avec différentes fonctions numpy et pandas a été créé. La création de cette automatisation permettrait la récolte de données en temps réel dans le futur.

Un descriptif statistique des données a également été réalisé.

Les données présentes dans un dataframe ont ensuite été nettoyées. En effet, celles-ci contenaient beaucoup de NaNs.

Dans un premier temps les variables contenant plus de 50 % de NaNs ont été supprimées ou les enregistrements avec des valeurs manquantes pour ces variables ont été effacés.

Les enregistrements liés aux valeurs manquantes des variables avec moins de 5% de NaNs ont également été supprimés. Ces seuils ont été déterminés après une étude des pratiques générales réalisées en machine learning.

La bibliothèque python Missingno a permis de visualiser les variables contenant de 5 à 50% de NaNs (figure 11).

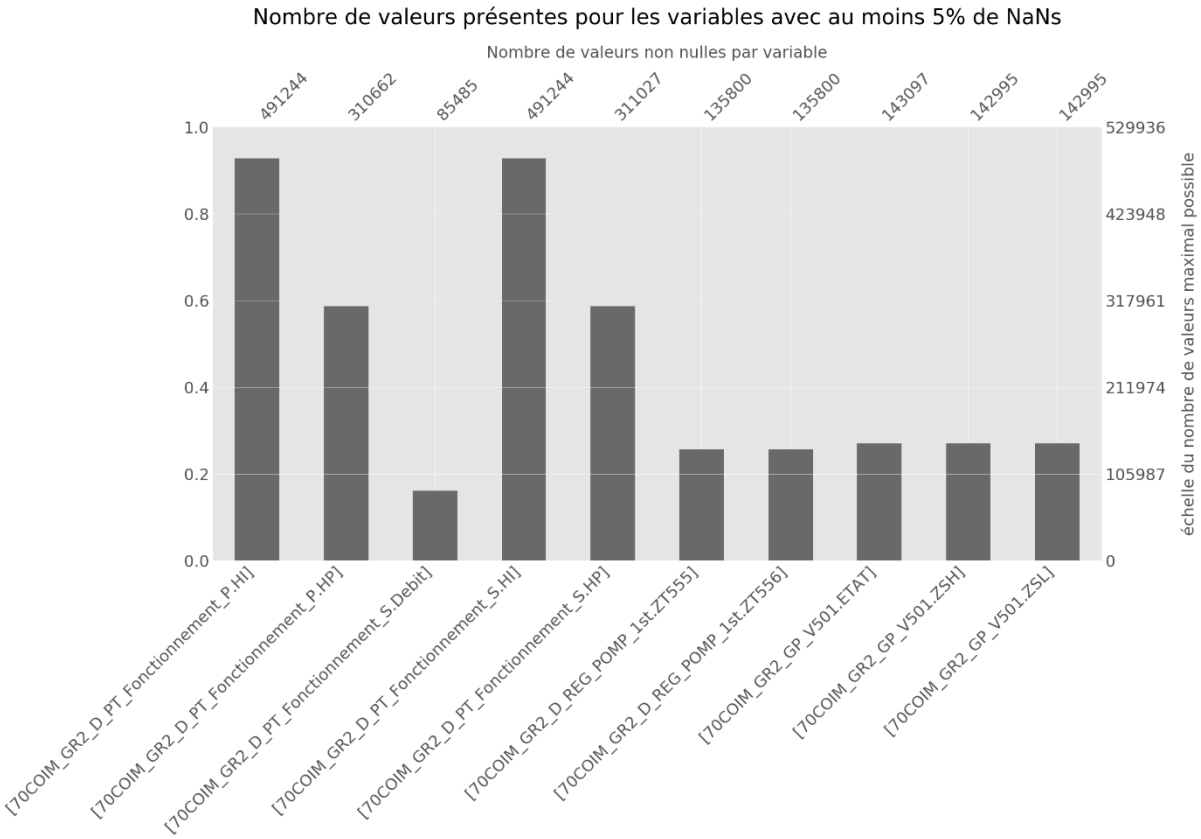


Figure 11 : Nombre de valeurs présentes pour les variables avec au moins 5% de NaNs

Certaines de ces variables ont été supprimées, des enregistrements du Dataframe ont été effacés ou bien les valeurs manquantes ont été remplacées (remplacement par la dernière ou la prochaine valeur non nulle ou bien avec la moyenne des valeurs de la variable ou encore par interpolation linéaire avec les bibliothèques python fillna et interpolate). Le choix final des méthodes et techniques s'est basé sur les résultats de la qualité de prédictions obtenus par machine learning.

La matrice de corrélation de nullité Missigno a permis de visualiser des corrélations entre variables (figure 12). Cette corrélation mesure l'effet de l'impact de la présence ou non d'une valeur d'une variable sur l'apparition de valeurs d'une autre variable. Les variables avec une corrélation proche de 1 ont donc subi un même traitement sur les NaNs. A l'inverse, les variables avec une corrélation proche de 0 ont donc subi un traitement individualisé avec les méthodes et techniques citées précédemment.

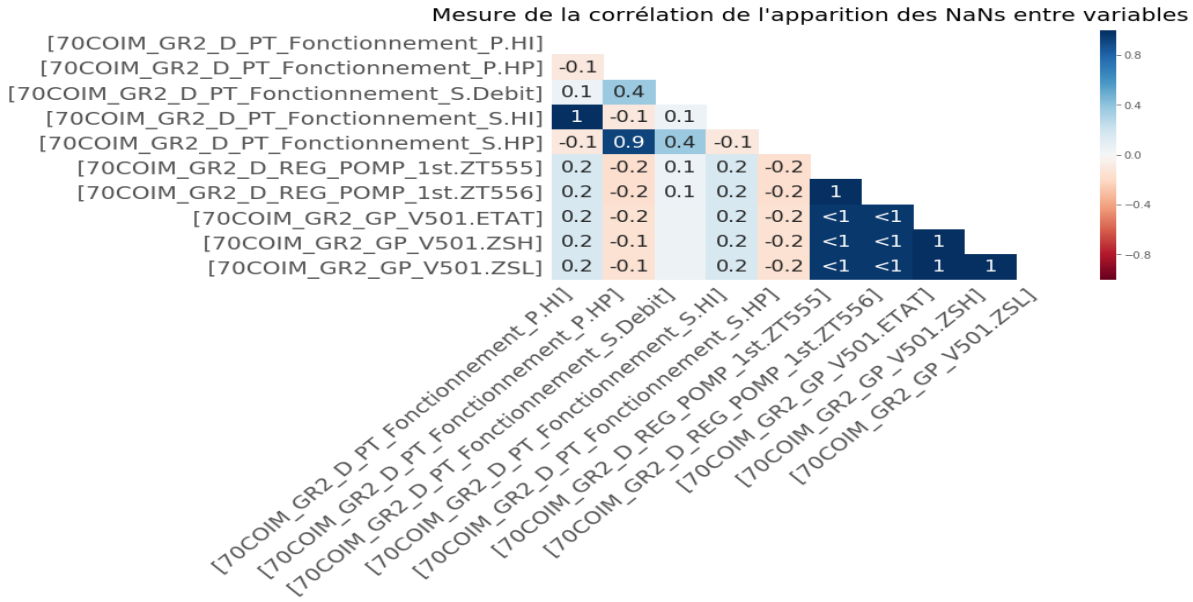


Figure 12 : Matrice de corrélation Missingno mesurant la corrélation d'apparition des NaNs entre variables

Des variables ont ensuite été créées. Travaillant sur des séries temporelles (considération possible, voir « axes d'améliorations »), une colonne « Id » a été créée. Elle s'incrémente à chaque nouvelle plage temporelle arrêt machine-panne. Cette colonne est nécessaire pour intégrer la cyclicité du fonctionnement d'un compresseur et ainsi compléter la préparation de séries temporelles avant utilisation par le LSTM.

D'autres variables combinées ont été fabriquées : la variance et la moyenne mobiles des données brutes. Celles-ci pourraient permettre d'améliorer les prédictions du LSTM en facilitant l'identification de cyclicité ou une tendance d'évolution possible de séries temporelles.

Enfin, tous les traitements sur les données citées au-dessus ont permis d'obtenir des jeux de données différents et de comparer leur apport dans les performances de modèles de machine et deep learning.

- **Les spécifications techniques de la sélection et utilisation d'algorithmes de machine et deep learning**

La troisième étape consiste à sélectionner les algorithmes de prédiction. Pour cela je me suis basé sur les résultats des métriques d'évaluation obtenus par plusieurs algorithmes utilisés en maintenance prédictive (Use case du chapitre état de l'art). Ainsi j'ai choisi le Long Short-Term Memory *(LSTM)* en classification et régression supervisées. En effet deux types de problèmes sont identifiés par le deuxième objectif client et doivent donc faire intervenir différents procédés. Par ailleurs le LSTM est utile pour faire des prédictions de séries temporelles.

o La préparation des données

Tout d'abord, les variables nombreuses ont été sélectionnées par « feature selection » afin de possiblement améliorer les performances de l'algorithme de prédiction. Cette méthode se base entre autre sur les retours métiers. Ici, le client a fourni les 7 variables principales associées à la dépressurisation et un visuel a été établi sur chacune d'elle par l'élaboration de courbes avec Matplotlib. Des méthodes de filtration ont été utilisées avec des bibliothèques python comme chi2, mutual_info_classif, mutual_info_regression et SelectKBest de sklearn.
La réduction de la dimensionnalité des données a également été réalisée par le module PCA (Principal component analysis) de sklearn.

Puis les données issues de « feature selection » ou pas ont été normalisées avec la fonction MinMaxScaler de sklearn. En effet, cette dernière permet de mettre toutes les données à la même échelle, sur un intervalle [0,1] et donc de réduire l'effet des valeurs aberrantes, « outliers » en anglais (12).

Ensuite, le train/test split en classification a été réalisé. Deux méthodes basées sur la sélection de données avec ou sans les phases de marche du compresseur ont été mises en place. Les plages temporaires consécutives avec ou sans arrêt machine contenant 80 % du nombre total de pannes du dataframe d'intérêt ont été sélectionnées pour le train et le reste pour le test. Ce choix s'est basé sur le principe de Pareto[1].
Le train/test split en régression a été fabriqué différemment. Les plages temporaires avec ou sans arrêt machine contenant toutes les observations précédant un temps d'intérêt pour le test se retrouvent dans le train. Le test ne contient donc que quelques observations relatives au temps choisi pour faire la prédiction.

[1] Principe de Pareto, aussi appelé loi de Pareto ou encore loi des 80-20, phénomène empirique constaté dans certains domaines : environ 80% des effets sont le produit de 20% des causes.

Dans la bibliographie, le train/test est unique et correspond à celui réalisé en régression (13). Pour éviter d'obtenir un test avec peu de prédictions en classification et donc avoir un doute sur la fiabilité des résultats des métriques d'évaluation j'ai changé le protocole.

Enfin, la mise en forme des données pour le LSTM a été effectuée. Un algorithme permettant un look back de 30 sur les données de train et de test a été créé et utilisé.
La matrice 3D des données [échantillon, pas de temps, nombre de variables] a ainsi été obtenue (14).
La valeur du look back a été choisi afin de ne surtout pas dépasser la plage temporelle (30min) de données nécessaire pour entrainer un LSTM en régression permettant de répondre à l'objectif client d'une prédiction 30 minutes après arrêt.
L'échantillon correspond aux nombres de mesures par pas de temps.
Pour éviter un problème de mémoire l'algorithme a inclus un générateur « yield ». La fonction génératrice permet d'effectuer en boucle la mise en forme des données [échantillon, pas de temps, nombre de variables] pour chaque plage temporelle début_enregistrement-panne_machine.
La préparation des données d'entrée et le paramétrage du modèle LSTM a suivi en très grande partie un github de référence (13).

Les données de la variable cible en classification ont toujours été balancées. Un Upsampling (entrainement) a toutefois été réalisé en utilisant le module Smotenc de la bibliothèque imblearn.

○ Les paramétrages du LSTM

Le choix des paramètres du réseau de neurones comme le nombre d'époch, le nombre de couches, le nombre de neurones et les fonctions d'activations a été effectué à l'aide du git hub de référence avec cependant quelques différences (figure 13). La fonction sequential de keras a été utilisée afin d'ajouter toutes les couches au modèle. Le batch_size a été chois afin de ne pas trop ralentir les temps de calcul. Le callback a été utilisé afin de pouvoir obtenir une vue interne du modèle durant l'entrainement.

```
# design network
model = Sequential()
model.add(LSTM(100, return_sequences = True, input_shape = (X_train.shape[1], X_train.shape[2]))) #input_shape=(t
model.add(Dropout(0.2))
model.add(LSTM(50, return_sequences = False))
model.add(Dropout(0.2))
#model.add(Dense(y.shape[1]))

model.add(Dense(1)) # number of output = 1.
model.add(Activation('linear'))
model.compile(loss="mean_squared_error", optimizer="adam")
model.summary()
# fit network
history = model.fit(X_train, y_train, epochs=150, batch_size=200, validation_data=(X_test, y_test), verbose=1, sh
                    callbacks = [keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=10, verbo
```

Figure 13 : Paramétrage du LSTM en régression

Le nombre d'époch a été défini afin d'éviter l'overfitting. Pour cela les courbes historiques de la fonction de perte lors de l'entrainement et de la validation ont été visualisées avec matplotlib et grâce au callback (figure 14). En ajoutant un epoch supplémentaire, les courbes de training et validation se croisent permettant de remarquer un overfitting. De manière générale des tests empiriques sur les hyper paramètres ont été effectués sur le LSTM.

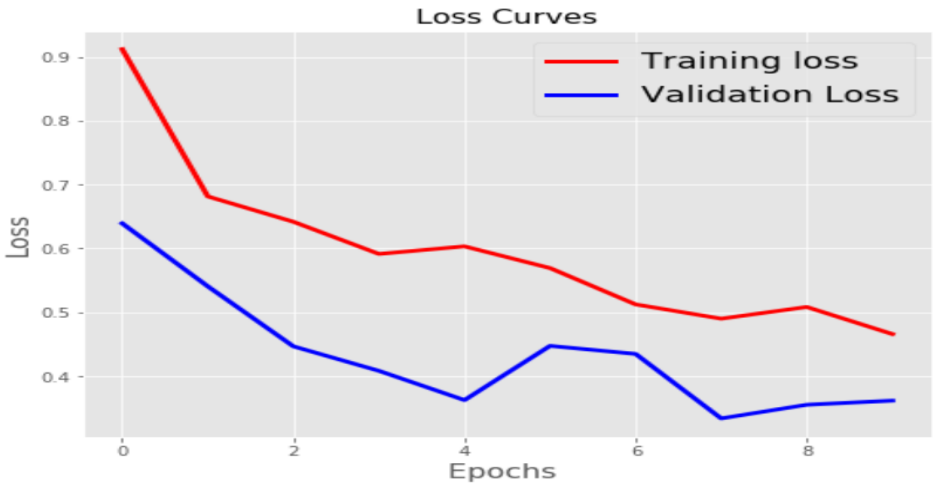


Figure 14 : Evolution de la parte de gradient en fonction du nombre d'epochs pour l'entrainement et la validation d'un LSTM

○ L'évaluation des algorithmes de prédiction

Pour comparer les différents modèles mis en place différentes métriques de scoring ont été utilisées. Pour les algorithmes de classification, la matrice de confusion, la précision et le recall ont été obtenus (figure 15). Ces métriques sont intéressantes pour évaluer un modèle qui retourne des variables binaires. Le rappel (recall) ou sensibilité est le taux de vrais positifs et doit être observé avec la précision (proportion de prédictions correctes) pour évaluer efficacement le modèle. En effet, des prédictions binaires déséquilibrées conduisent à surestimer la valeur d'une des deux métriques et en complément à sous-estimer la valeur de la métrique associée (15). Pour évaluer un compromis entre rappel et précision, j'ai calculé le F1 score aussi appelé F-measure (figure 15).

Precision	$\frac{TP}{TP+FP}$
Rappel	$\frac{TP}{TP+FN}$
F-measure	$\frac{2.precision.rappel}{precision+rappel}$

- TP = True Positive, FP = False Positive, FN = False negative.
- Predicted = valeur prédicte, Acutal = vraie valeur, N = nombre de prédictions.

Figure 15 : métriques d'évaluation utilisées avec les algorithmes de classification

Pour les algorithmes de régression, la métrique utilisée est la racine carrée de l'écart quadratique moyen (RMSE en anglais). Le RMSE comprend à la fois la variance et le biais de l'estimateur. Sa formule est reportée sur la figure 16. Ce dernier a été préféré au MSE afin de ramener la métrique à l'échelle des prédictions effectuées.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Figure 16 : métriques d'évaluation utilisées avec les algorithmes de régression

○ La sélection de features

Comme dans l'un des chapitres précédents la sélection de features a été mise en place, cette fois, par les méthodes « wrapper » et « embedded » (16) et l'utilisation des modules python feature_importances et SelectFromModel. Cette sélection est basée sur les résultats obtenus par les algorithmes d'apprentissage machine à partir de combinaisons avec toutes les variables.

L'intérêt de ces techniques ici est double : améliorer potentiellement les performances de l'algorithme de prédiction et connaitre les variables les plus corrélées à la panne (dépressurisation), aux oscillations anormales de pression afin de répondre aux objectifs clients.

Celles-ci sont plus « accurate » que la méthode par filtration mais prennent plus de temps de calcul et sont donc plus adaptées aux prédictions à partir d'un nombre limité de variables explicatives.

La matrice de corrélation (heatmap) voulue par le client et mesurant la corrélation entre les variables sélectionnées a été créée grâce à la bibliothèque Seaborn. Celle-ci est présentée dans le chapitre suivant : « Le bilan de projet, l'atteinte des objectifs ».

• Les spécifications techniques de la validation client

La dernière étape consiste à faire valider le travail par le client.

Cela a été fait par mesure des métriques des performances des modèles de machine learning utilisées. Ces métriques établissent la fiabilité des prédictions dans un contexte donné.

5) LE BILAN DE PROJET

- **L'atteinte des objectifs**

Les objectifs client ont tous été atteints. En effet :

- Une comparaison de la prédiction de pannes entre le LSTM et le Random Forest en classification 12 heures à l'avance a été obtenue (figure 17). Le LSTM propose une précision de 78% et un recall de 100 % alors que le Random Forest propose une précision de 95% et un recall de 88%. Ce dernier modèle obtient de meilleurs résultats.

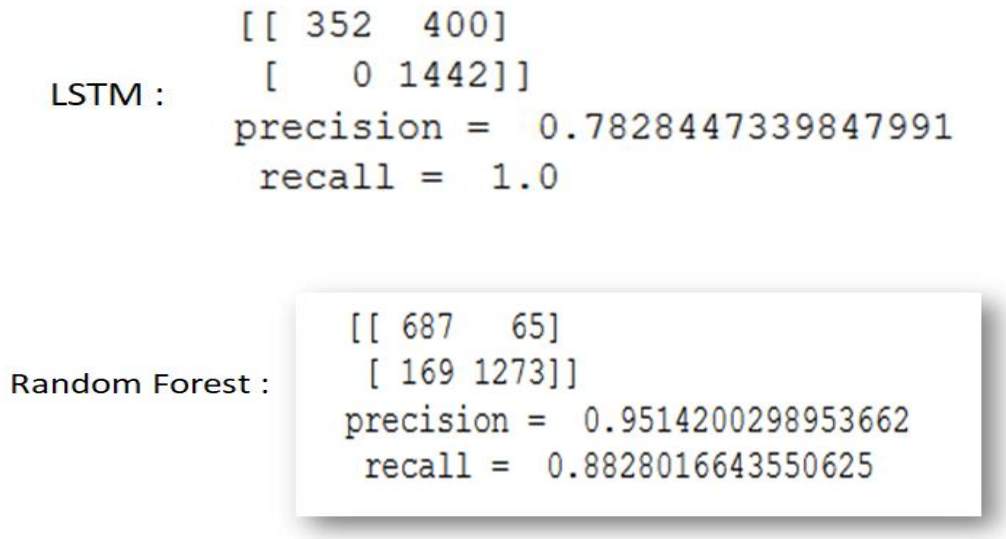


Figure 17 : Matrice de confusion, précision et recall obtenus sur différentes prédictions avec le LSTM et le Random Forest en classification 12 heures à l'avance

- Une comparaison de la prédiction de panne entre le LSTM et le Random Forest en régression 30 min après l'arrêt du compresseur à gaz (figure18). Des valeurs quasiment uniques de prédictions sont toujours obtenues et le RMSE est très élevé avec le LSTM. Le RMSE du Random Forest est meilleur : 203.

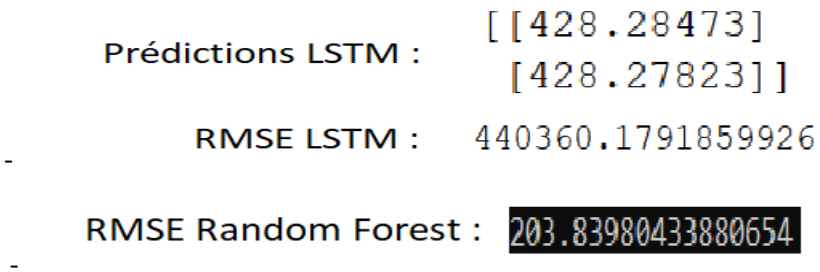


Figure 18 : RMSE obtenus sur différentes prédictions avec le LSTM et le Random Forest en régression 30 minutes après l'arrêt du compresseur

6) LES AXES D'AMELIORATION

Les prédictions de séries temporelles effectuées par le LSTM en régression n'ont pas donné de bons résultats.

Je suppose que cet échec est dû à la suppression de plusieurs variables numériques de l'entrainement de l'algorithme. Ces six variables contenaient un nombre très important de valeurs manquantes (>60%). Cette hypothèse est plausible. En effet, deux variables dans le jeu de données sont nécessaires pour améliorer le modèle de 15% alors qu'elles contiennent environ 30% de NaNs. Une technique de remplissage des valeurs manquantes n'a pas été utilisé et semble pourtant être la plus performante. Il s'agit d'utiliser des modèles de classification ou régression, de les entrainer sur un jeu de données sans trous et de les tester sur un jeu de données avec valeurs manquantes afin de les trouver (17).

Le Random Forest aurait lui fait de meilleures prédictions puisqu'il n'utilise pas de séries temporelles.

7) CONCLUSION

A partir des objectifs client j'ai programmé un LSTM en régression et classification (préparation des données, réglages des hyper paramètres). J'ai également entrainé et utilisé le modèle crée. Les résultats en classification ont été satisfaisants pour le client. Par contre le modèle n'a pas fonctionné en régression.

Afin de faire fonctionner ce modèle une piste est privilégiée. Le remplissage des NaNs du dataset d'origine avec des algorithmes IA. Cela permettrait au modèle de pouvoir alors travailler sur une série temporelle correcte.

Ce travail répond à la problématique client de trouver une méthode de sélection d'un modèle IA performant pour la maintenance prédictive d'un compresseur à gaz.
Pour cela, les modèles de Random Forest et LSTM ont été comparés sur deux stratégies différentes de maintenance prédictive et le meilleur a été choisi sur la base de métriques.

BIBLIOGRAPHIE

Sources internet :

(1) https://fr.wikipedia.org/wiki/Turbine_%C3%A0_gaz

(2) https://www.mobility-work.com/fr/blog/maintenance-predictive-vs-maintenance-preventive-strategie-entreprise

(4) https://medium.com/bigdatarepublic/machine-learning-for-predictive-maintenance-where-to-start-5f3b7586acfb

(5) https://gallery.azure.ai/Experiment/a677f8eececf40eaa158699a2b27e3c8

(6) https://mc.ai/introduction-au-deep-learning-et-aux-reseaux-de-neurones-pour-les-nuls/

(7) https://medium.com/smileinnovation/lstm-intelligence-artificielle-9d302c723eda

(8) https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046

(9) https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10

(10) https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

(11) https://www.decideo.fr/Big-Data-IoT-IA-quelle-gestion-de-la-donnee_a11227.html
https://www.silicon.fr/hub/hpe-intel-hub/collecter-transporter-stocker-et-traiter-les-donnees-iot

(12) https://mrmint.fr/data-preprocessing-feature-scaling-python

(13) https://github.com/Samimust/predictive-maintenance

(14) https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

(15) https://openclassrooms.com/fr/courses/4297211-evaluez-les-performances-dun-modele-de-machine-learning/4308256-evaluez-un-algorithme-de-classification-qui-retourne-des-valeurs-binaires

(16) https://medium.com/@cxu24/common-methods-for-feature-selection-you-should-know-2346847fdf31

(17) https://towardsdatascience.com/the-tale-of-missing-values-in-python-c96beb0e8a9d

Sources documents :

(3) https://www.iacpartners.com/uploads/Publications/Pred%20Maintenance/iac-partners-maintenance-predictive.pdf