Rapport – Maintenance sur compresseurs d'air

Quentin Gense

Louis Soyez

M1 Data & IA

# 1. Identification du problème

Dans le secteur industriel, les compresseurs d’air sont des éléments essentiels mais sujets à des défaillances variées (bruit anormal, encrassement, pression instable, etc.). Les procédures actuelles de maintenance sont souvent réactives, intervenant uniquement après l’apparition de symptômes critiques. Cela entraîne des interruptions de service coûteuses.

L’objectif du projet est donc de développer un système de maintenance prédictive à l’aide de techniques d’apprentissage automatique, afin d’identifier ces défaillances à partir de mesures captées en temps réel par des capteurs intégrés.

# 2. Données utilisées

Le dataset provient de la plateforme Kaggle, sous le nom “Predictive Maintenance Dataset Air Compressor”. Il contient des lectures de capteurs (pression, température, débit, bruit, etc.), des indicateurs qualitatifs d’état (propre/sale, bruyant/silencieux) et des identifiants de machines.

Une étape importante a été la création d’une variable synthétique `diagnostic`, permettant de qualifier une machine comme “Défaillante” ou “Sans problème” à partir de la combinaison des indicateurs (radiator, bearings, wpump, exvalve).

# 3. Analyse exploratoire (EDA)

Une analyse détaillée a été menée : vérification de la complétude des données, visualisation des distributions de variables continues, inspection de la répartition des valeurs catégorielles, et exploration de la variable `diagnostic`. Une corrélation entre certaines variables (comme le régime moteur `rpm`) et la santé du système a été observée.

Aucune valeur manquante n’a été observé et il n’y a donc pas eu de besoin de modification à ce niveau. Néanmoins des colonnes comme ‘id’ et ‘acmotor’ ont été supprimé à cause de leur manque de pertinence dans ce jeu de données.

Des clusters de machines ont été identifiés à l’aide de l’algorithme KMeans sur `rpm`, suggérant des modèles de fonctionnement différents entre les groupes. Ce regroupement a permis d’adapter la normalisation des variables par cluster.

# 4. Prétraitement

Le prétraitement a comporté plusieurs phases : nettoyage, suppression des colonnes redondantes, conversion typée, normalisation avec `MinMaxScaler` par cluster. Les variables numériques ont été standardisées tandis que les étiquettes ont été conservées telles quelles.

# 5. Modélisation : Apprentissage automatique

Nous avons testé différents modèles et avons utiliser du fine-tuning sur certains d’entre eux afin de maximiser nos résultats. Beaucoup des modèles rentraient en surapprentissage, bien que l’entrainement n’étais pas affecté, on observait très clairement que les données de train étaient apprises par cœur par le modèle. Il était donc important de modifier les hyperparamètre afin de rendre le modèle plus flexible et l’empêcher d’apprendre par cœur nos données. Ce problème était notamment dû au faible jeu de donnée dont nous disposions (1000 lignes – 26 colonnes).

1. RandomForestClassifier

Le Random Forest est un algorithme d'ensemble basé sur la technique de bagging. Il s'appuie sur la création d'une multitude d'arbres de décision, chacun entraîné sur un sous-échantillon différent des données d'entraînement, avec un sous-ensemble aléatoire de caractéristiques.

Cette méthode permet de réduire la variance du modèle et améliore sa robustesse face au surapprentissage. Chaque arbre vote pour une classe, et la classe majoritaire est retenue comme prédiction finale.

Random Forest est particulièrement utile dans les jeux de données avec des variables bruyantes ou de nombreuses interactions non linéaires. Il est aussi apprécié pour sa capacité à estimer l'importance relative des variables, ce qui aide à l’interprétation.

1. LinearSVC

LinearSVC est une version linéaire du Support Vector Machine (SVM). Il cherche à séparer les classes par un hyperplan en maximisant la marge entre les points les plus proches des deux classes opposées (appelés vecteurs de support).

Contrairement à la version non linéaire de SVM, LinearSVC ne peut capturer que des frontières de décision linéaires. Il est cependant extrêmement rapide et efficace, surtout dans les cas où les données sont linéairement séparables ou faiblement bruitées.

LinearSVC fonctionne bien pour les données à grande dimension, mais nécessite une normalisation préalable des variables pour de bonnes performances.

1. SVC

SVC (Support Vector Classifier) est la version plus flexible du SVM. Elle permet l’utilisation de noyaux (kernels) pour projeter les données dans un espace de plus grande dimension, où une séparation linéaire est possible.

Cette technique rend le SVC très puissant dans des cas complexes où les frontières de décision sont non linéaires, ce qui est souvent le cas dans des problèmes industriels ou biomédicaux.

L’inconvénient majeur est le coût computationnel élevé pour les grands ensembles de données, et une complexité de paramétrage accrue (choix du noyau, réglage du paramètre C, etc.).

1. LogisticRegression

La régression logistique est un modèle linéaire probabiliste. Elle estime la probabilité d’appartenance à une classe à l’aide d’une fonction logistique (sigmoïde).

Elle est simple, rapide à entraîner, et facilement interprétable via les coefficients associés à chaque variable. C’est un excellent point de départ pour tout projet de classification.

Cependant, elle montre rapidement ses limites dans des contextes où la relation entre les variables et la classe cible est non linéaire ou implique des interactions complexes.

1. KNeighborsClassifier

Le classifieur K plus proches voisins (KNN) est une méthode dite paresseuse, qui ne construit pas de modèle explicite. Lors d’une prédiction, il compare l’exemple aux données d’entraînement et sélectionne les k plus proches voisins selon une distance (souvent euclidienne).

Ce modèle est intuitif et fonctionne bien pour des jeux de données simples, mais devient inefficace dès que la dimension augmente (phénomène de la malédiction de la dimensionnalité).

Il est également très sensible au bruit et au choix du paramètre k. Une normalisation des données est impérative pour garantir de bonnes performances.

1. DecisionTreeClassifier

L’arbre de décision est un algorithme qui divise récursivement l’espace de décision en fonction de règles simples (seuils sur les variables). Il construit une hiérarchie de décisions jusqu’à atteindre un critère d’arrêt (pureté d’un nœud, profondeur maximale, etc.).

Facile à comprendre et à interpréter, ce modèle est très utilisé en phase exploratoire ou lorsqu’on souhaite expliquer les prédictions. Il peut aussi gérer des données mixtes (numériques et catégorielles) sans transformation particulière.

Cependant, l’arbre seul est très instable (variation des données = changement de structure) et sujet au surapprentissage, surtout s’il n’est pas limité en profondeur.

1. ExtraTreeClassifier

L’Extra Tree Classifier (ou Extremely Randomized Tree) est une version plus aléatoire de l’arbre de décision. Il choisit les points de division complètement au hasard, contrairement à un arbre classique qui sélectionne la meilleure séparation selon un critère (comme l’entropie).

Cette approche accélère fortement l’apprentissage et réduit la variance, au prix d’un léger accroissement du biais. Seul, ce modèle est souvent peu performant, mais il excelle dans des structures en ensemble (comme ExtraTreesClassifier).

Il permet aussi de diminuer la corrélation entre les arbres d’une forêt, ce qui améliore globalement la généralisation.

1. XGBClassifier

XGBoost (Extreme Gradient Boosting) est une méthode d’ensemble très performante basée sur le principe du boosting. Elle ajoute séquentiellement des arbres faibles (peu profonds) en corrigeant les erreurs des modèles précédents à chaque itération.

XGBoost intègre de nombreuses optimisations : régularisation, gestion des valeurs manquantes, parallélisation, et early stopping. Il est souvent utilisé comme modèle de référence dans les compétitions de data science.

Il excelle sur les données tabulaires et gère efficacement les relations complexes entre variables. Toutefois, il nécessite un tuning rigoureux pour en tirer le meilleur parti, et reste relativement opaque à interpréter sans outils spécialisés.

Les résultats se valaient tous bien que le DecisionTreeClassifier et le XGBClassifier se sont démarquer des autres avec une précision de presque 100%.

Une spécificité notable du projet est l’approche en multilabelling. Au lieu de prédire une seule étiquette, plusieurs sorties binaires ont été apprises simultanément pour différents composants (bearings, radiator, wpump, exvalve). Cela reflète la réalité industrielle où plusieurs défaillances peuvent coexister.

# 6. Modélisation : Mise en place d’Ensemble-Learning (Stacking)

L’ensemble learning repose sur l’idée que plusieurs modèles (souvent appelés apprenants faibles) combinés peuvent produire une prédiction plus robuste et précise qu’un seul modèle. Les techniques d’ensemble les plus connues sont le Bagging, le Boosting et le Stacking.

Le stacking consiste a organizer les modèles d’apprentissage en niveaux :

- Niveau 0 : On entraîne plusieurs modèles de base séparément sur les mêmes données d’entré.

- Niveau 1 : On entraîne un méta-modèle sur les sorties des modèles de base. Il apprend alors à combiner les prédictions de ces modèles pour produire une sortie finale.

Les avantages du stacking est qu’il combien les forces de différents modèles (linéaire, non linéaire, simples, complexes…), il est très performant et ce surtout sur des jeux de données riches et complexes et il peut capturer des interactions entre les prédictions des modèles de base. Néanmoins il est plus complexe à mettre en œuvre, il risque de surapprendre si le méta-modèle est trop puissant ou si les modèles de base ne sont pas diversifiés et il a un besoin de techniques rigoureuses comme la cross-validation imbriqué pour éviter les fuites de données entre niveaux.

Dans notre modèle nous avons utilisé une stratégie de stacking mult-label pour conserver cette prédiction sur différents types de problèmes. Nous utilisons ici 4 modèles :

* RandomForestClassifier

SVC

LogisticRegression

KNeighborsClassifier

Par la suite nous utilisons une méthode de Cross-Validation afin de vérifier si notre modèle n’entre pas dans un état de surapprentissage, nous obtenons des résultats positifs démontrant que les résultats du modèle sont constants et qu’il ne surapprend pas.

# 7. Évaluation des résultats

Les résultats des modèle DecisionTreeClassifier et XGBClassifier ont montré une bonne capacité de classification sur les différents labels. Les métriques typiques telles que l’accuracy, la précision, le rappel, la F1-score ont été utilisées.

Les performances sont globalement bonnes, mais certaines catégories sont plus difficiles à prédire, il est possible que ces classes ne se démarque pas spécialement par rapport aux autres défauts. Le bruit dans les données ou la redondance entre certaines variables pourrait également perturber l’apprentissage.

# 8. Limites et améliorations

Le modèle pourrait être amélioré via :

- la création de nouvelles variables (ratios, agrégats temporels),

- le traitement du déséquilibre de classes avec des techniques comme SMOTE,

- l’augmentation du volume de données pour utiliser des modèles plus complexes,

- l’exploration d’architectures plus complexes comme des réseaux de neurones ou des modèles séquentiels (LSTM) si les données temporelles sont disponibles.

# 9. Conclusion

Le projet illustre avec précision comment une approche de machine learning (modèles classiques et ensemble learning + multilabelling) peut être appliquée pour détecter des défaillances sur des compresseurs d’air. Le diagnostic automatisé, nourri par des capteurs internes, permettrait d’anticiper les pannes et de rationaliser les opérations de maintenance. Cette solution est transférable à d’autres équipements industriels intelligents.