MULTI-LABEL CLASSIFICATION FOR PREDICTIVE MAINTENANCE FAILURE MODE DETECTION
AI4I 2020 DATASET

Rayane GALLEZE & Julien PILLIS & Oussama ZILI
Université de Technologie de Compiègne
AI28 - Machine learning pour l'ingénieur

ABSTRACT

Cette étude propose une approche de prédiction des types de panne machine à partir du jeu de données synthétique AI4I 2020, représentatif des scénarios industriels de maintenance prédictive. En exploitant des techniques d'apprentissage supervisé, nous modélisons la relation entre les paramètres opérationnels (températures, vitesse de rotation, couple, usure d'outil, etc.) et les occurrences de pannes machines. L'objectif est de prédire avec précision l'étiquette machine failure , résultant de cinq types de défaillances. Plusieurs modèles sont évalués (logistic regression, random forest, gradient boosting), avec une attention particulière portée à l'interprétabilité des résultats. Les performances obtenues démontrent la faisabilité d'une détection précoce des pannes, ouvrant la voie à une maintenance proactive et à une réduction des arrêts non planifiés.

Notebook accessible sur GitLab :

`https://gitlab.utc.fr/ziliouss/projet_ai28`

1 ANALYSE EXPLORATOIRE DE DONNÉES

Avant toute modélisation, il est essentiel de procéder à une analyse exploratoire des données (AED) afin de mieux comprendre la structure, la distribution et les relations potentielles entre les variables. Cette étape permet d'identifier d'éventuelles anomalies, des déséquilibres de classes, des corrélations significatives ou encore des valeurs manquantes susceptibles d'influencer les performances des modèles. Dans notre cas, l'AED a été particulièrement utile pour visualiser la séparation entre comportements normaux et anomalies, ainsi que pour évaluer la pertinence des différentes features dans la détection des labels cibles. Elle constitue donc un socle nécessaire à la sélection des variables, au choix des modèles et à la définition des stratégies de traitement des données.

1.1 COMPRÉHENSION DES VARIABLES

Table 1: Description des variables du jeu de données de production

Variable	Type	Description
UDI	Entier	Identifiant unique allant de 1 à 10 000.
Product ID	Chaîne	Lettre représentant la qualité (L, M, H) suivie d'un numéro de série spécifique. Répartition : 50% L, 30% M, 20% H.
Type	Catégorie	Identique à la première lettre du Product ID : L, M ou H.
Air temperature [K]	Float	Température de l'air (marche aléatoire), normalisée à $\sigma = 2\,K$ autour de 300 K.
Process temperature [K]	Float	Température de procédé = température de l'air + 10 K + bruit ($\sigma = 1\,K$).
Rotational speed [rpm]	Float	Calculée à partir de 2860 W, bruit normal ajouté.
Torque [Nm]	Float	Moyenne de 40 Nm, écart-type de 10 Nm, valeurs négatives exclues.
Tool wear [min]	Entier	Usure de l'outil, avec ajout de 2 (L), 3 (M) ou 5 (H) minutes selon la qualité.
Machine failure	Booléen	Indique une panne de machine (voir types ci-dessous).
TWF	Booléen	*Tool Wear Failure* : panne par usure de l'outil entre 200–240 min.
HDF	Booléen	*Heat Dissipation Failure* : échec si $\Delta T < 8.6\,K$ et vitesse < 1380 rpm.
PWF	Booléen	*Power Failure* : échec si puissance (couple × vitesse rad/s) < 3500 W ou > 9000 W.
OSF	Booléen	*Overstrain Failure* : panne si (usure × couple) > 11000 (L), 12000 (M), 13000 (H) minNm.
RNF	Booléen	*Random Failure* : échec aléatoire avec une probabilité de 0.1%.

Comprendre la structure et les caractéristiques du jeu de données AI4I 2020 est essentiel pour mener cette étude. Le jeu de données contient 10 000 observations et 14 variables, incluant des caractéristiques opérationnelles, des identifiants produits, et des étiquettes de défaillance.

Le jeu de données ai4i2020.csv contient des informations sur le fonctionnement de machines industrielles et leurs défaillances.

- Dimensions : L'ensemble de données comprend 10 000 entrées (lignes) et 14 variables (colonnes)
- Types de Données : Il se compose de variables numériques (int64, float64) telles que les températures, la vitesse de rotation, le couple et l'usure de l'outil, ainsi que de variables de type object pour l'ID du produit et le type de machine.
- Données Manquantes : Une vérification approfondie a confirmé l'absence totale de valeurs manquantes dans l'ensemble des colonnes, ce qui simplifie grandement la phase de prétraitement des données.

1.2 STATISTIQUES DESCRIPTIVES ET DISTRIBUTION DES VARIABLES

Distribution de la variable cible Machine failure La variable Machine failure agrège toutes les pannes. Son analyse révèle un déséquilibre prononcé :

- Répartition : Seulement 3.39% des entrées correspondent à une défaillance machine (Machine failure = 1), tandis que 96.61% n'en présentent pas (Machine failure = 0).

Cette forte disparité indique que nous sommes face à un problème de détection d'anomalie ou de classification hautement déséquilibrée. Des techniques spécifiques (suréchantillonnage, sous-échantillonnage, ajustement des poids de classe, métriques adaptées comme la précision ou le rappel) devraient être envisagées lors de la phase de modélisation pour gérer cet déséquilibre.

L'examen des histogrammes, boîtes à moustaches (boxplots) et Q-Q plots pour les variables numériques (Air temperature [K], Process temperature [K], Rotational speed [rpm], Torque [Nm], Tool wear [min]) révèle les points suivants:

- Températures (Air & Process) : Présentent des distributions relativement normales avec peu d'outliers significatifs.
- Vitesse de Rotation & Couple : Leurs distributions sont également assez symétriques, avec quelques valeurs extrêmes détectées par les boîtes à moustaches.
- Usure de l'Outil : Montre une distribution plus hétérogène, avec une concentration de valeurs basses et une queue vers la droite, indiquant une usure progressive avec des pics occasionnels. Des outliers sont présents, suggérant des événements d'usure rapide ou des anomalies de capteur.

Ces observations seront importantes pour le choix des méthodes de standardisation ou de normalisation des données avant l'entraînement des modèles.

1.3 Analyse de Corrélation

Une matrice de corrélation a été calculée pour évaluer les relations linéaires entre les variables numériques :

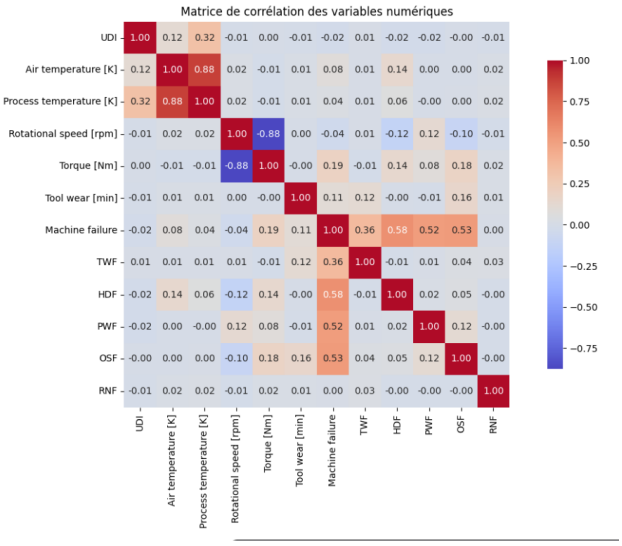


Figure 1: Matrice de corrélation

Une forte corrélation négative est observée entre la Rotational speed [rpm] et le Torque [Nm] (-0.88). Ceci est physiquement cohérent : une machine augmentant son couple tend à réduire sa vitesse de rotation et vice-versa. Une corrélation positive modérée est également visible entre Air temperature [K] et Process temperature [K] (0.88).

Les variables de défaillance spécifiques (TWF, HDF, PWF, OSF, RNF) montrent des corrélations faibles à modérées avec les autres variables numériques. Cela suggère que ces défaillances ne sont pas simplement et fortement liées de manière linéaire à une seule variable opérationnelle, mais peuvent dépendre de combinaisons ou de relations plus complexes.

Cependant, (TWF, HDF, PWF, OSF, RNF) montrent une corrélation plus positive avec la variable cible, à savoir la défaillance de la machine (ce qui est logique, Machine failure vaut 1 si on rencontre une des pannes)

.

1.4 Analyse Approfondie des Cibles Multi-Labels

Pour mieux comprendre les relations entre les variables numériques prédictives (Air temperature [K], Process temperature [K], Rotational speed [rpm], Torque [Nm], Tool wear [min]) et chacun des 5 types de défaillance (OSF, HDF, PWF, RNF, TWF), nous avons utilisé trois mesures de dépendance :

- Corrélation de Pearson : Mesure la force et la direction d'une relation linéaire simple entre deux variables.
- Corrélation de Spearman : Évalue la force et la direction d'une relation monotone (croissante ou décroissante), qu'elle soit linéaire ou non. Elle est basée sur les rangs des données.
- Information Mutuelle (MI) : Quantifie la dépendance générale entre deux variables, capturant à la fois les relations linéaires et non linéaires, y compris les interactions complexes. Une valeur de MI élevée indique une forte dépendance.

L'analyse par cible a révélé les spécificités suivantes :

Table 2: Analyse des corrélations et de l'information mutuelle par type de défaillance

Défaillance	**Corrélations (Pearson / Spearman)**	**Information Mutuelle**	**Interprétation**
HDF	Faibles (< 0.15)	Modérée	HDF peut-être difficile (plus que OSF qui a des correlations linéaires plus fortes) à prédire par un simple seuil ou un arbre unique.
OSF	Élevées (surtout avec le couple)	> 0.01	OSF a les corrélations de pearson les plus élevées malgré des MI élevés. Cela peut indiquer que les dépendances peuvent être complexes mais facile à prédire.
PWF	Faibles à modérées	Faibles	Relations simples ou variables manquantes; modélisation possible mais limitée
TWF	Faibles à modérées	Faibles	Peut être prévisible via l'usure; relations relativement simples
RNF	Très faibles	Très faibles	Aléatoire ; pratiquement impossible à prédire à partir des données mesurées

En complément, nous pouvons tracer les pairplots des variables numériques selon les erreurs :

(a) TWF (b) HDF (c) PWF

(d) OSF (e) RNF

Figure 2: Comparaison des pairplots pour les cibles : TWF, HDF, PWF, OSF et RNF

On remarque que les pannes PWF et OSF se distinguent des autres pannes en étant davantage sur les extrémités. HDF est concentré par zone, mais davantage au milieu de l'ensemble des données. TWF semble plus éparse, avec tout de même un concentration dans une certaine zone lorsque la valeur Tool Wear est élevée. RNF est quant à elle difficile à comprendre, mais cela semble normal, puisque ces pannes sont générées aléatoirement. Globalement, les algorithmes à base d'arbres de décision (DecisionTreeClassfier, RandomForest, XGBoost, LightGBM...) semblent donc les plus adaptés à notre problème. Nous tenterons de vérifier cela lors de la construction du modèle.

Si nous nous intéressons davantage à la répartition des pannes, nous pouvons nous apercevoir que nous n'avons que très peu de données positives pour les pannes TWF et RNF. Cela sera à prendre en compte pour la suite du problème, et nous indique qu'il sera probablement difficile de les prédire correctement. Selon la classification Google[1], le degré de désequilibre est "extrême" (moins d'1% des données ont cette anomalie).

[1] https://developers.google.com/machine-learning/crash-course/overfitting/imbalanced-datasets?hl=fr

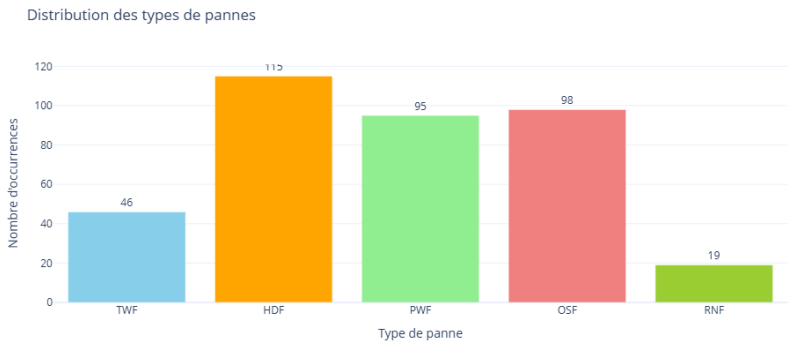


Figure 3: Répartition des pannes

1.5 Importance de la feature : Qualité

Maintenant, nous pouvons nous intéresser à la variable catégorielle ”Qualité”. Une tendance semble émerger en étudiant la proportion des pannes par qualité :

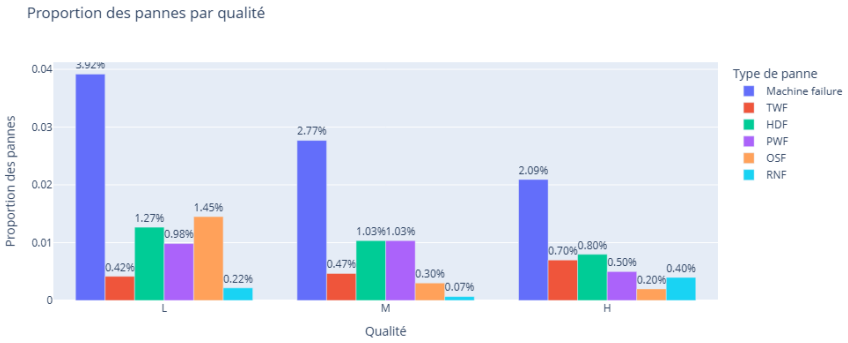


Figure 4: Proportion des pannes par qualité

On remarque que la qualité semble impacter fortement la proportion de pannes de type OSF. On peut s’attendre à ce que la qualité ait une importance pour la détection d’OSF. De plus, nous pouvons nous apercevoir que la qualité du produit a un impact sur le nombre d’occurrences des pannes. En effet, les produits de qualité ‘L‘ (Low) ont tendance à avoir plus de pannes, tandis que ceux de qualité ‘H‘ (High) en ont moins. Cela peut être dû à une meilleure conception et fabrication des produits de haute qualité.

2 Prétraitement des données

Avant de procéder à la modélisation, un travail de prétraitement a été réalisé afin de nettoyer et enrichir les données. Ces étapes visent à préparer un jeu de données cohérent, pertinent et représentatif pour l’apprentissage automatique.

2.1 Extraction et nettoyage des identifiants de produits

La première lettre de la colonne `Product ID` encode une information sur la qualité du produit. Cette lettre a été extraite dans une nouvelle colonne nommée `Quality`, puis supprimée de `Product ID`. Par la suite, la colonne `Quality` a été repositionnée en deuxième position pour une meilleure lisibilité.

2.2 Vérification de la cohérence qualité/type

Une vérification a été effectuée pour identifier les éventuelles incohérences entre les colonnes `Quality` et `Type`. Les échantillons présentant une discordance ont été comptabilisés, ce qui a permis de confirmer que la colonne `Quality` pouvait remplacer avantageusement la colonne `Type`, cette dernière étant dès lors supprimée.

2.3 Suppression des colonnes inutiles

Plusieurs colonnes ont été supprimées :

- `Machine failure` : variable dépendante redondante avec les pannes spécifiques (`TWF`, `HDF`, etc.).
- `Product ID` : identifiant sans valeur explicative directe.
- `UDI` : identifiant unique sans intérêt pour l'apprentissage.

2.4 Encodage des labels

La colonne `Quality` a été encodée numériquement avec les valeurs :

- `L` (Low) $\rightarrow 0$
- `M` (Medium) $\rightarrow 1$
- `H` (High) $\rightarrow 2$

2.5 Création de variables dérivées

Trois nouvelles variables dérivées ont été calculées pour refléter des conditions physiques importantes :

- `delta_temp` : différence entre la température du processus et la température ambiante.
- `power_W` : puissance mécanique approximée, produit du couple et de la vitesse de rotation.
- `usure_couple` : variable combinant l'usure de l'outil et le couple, pour représenter une forme de contrainte mécanique cumulée.

2.6 Séparation du jeu d'entraînement et de test

Un découpage stratifié multilabel a été appliqué à l'aide de la méthode `MultilabelStratifiedShuffleSplit` afin de préserver les proportions de chaque panne dans les ensembles d'apprentissage et de test. Le jeu a été divisé en 10 splits avec 20% des données en test.

3 Elaboration du modèle de prédiction (pour PWF et OSF)

Maintenant que nous avons un aperçu général de nos données, et que nous avons pu les faire parler, nous pouvons essayer d'élaborer le modèle de prédiction.

La première chose à laquelle nous devons nous intéresser est le score à maximiser. En détection d'anomalies, rater une anomalie peut avoir des conséquences graves (sécurité, santé, économique...).

- Si on veut éviter de rater des pannes (pannes non prédites) : Il faut maximiser le recall (rappel). Cela permet de détecter un maximum de vraies pannes, même si cela génère plus de fausses alertes.
- Si on veut éviter les fausses alertes (interventions inutiles) : Il faut maximiser la précision. Cela réduit le nombre de fausses alertes, mais on risque de rater certaines pannes.

Le F1 pourrait être intéressant, mais c'est un compromis : il peut masquer des rappels faibles, cela signifie que l'on peut quand même avoir un F1-score moyen acceptable (rappel faible (beaucoup d'anomalies manquées), mais une précision élevée (peu de fausses alertes)). Cela masque le vrai problème : les anomalies non détectées.

Nous allons supposer que dans, notre contexte, le coût d'un faux négatif est supérieur à celui d'un faux positif (sinon la prédiction serait inutile).

Ainsi, mieux vaut détecter trop d'anomalies (avec quelques fausses alertes)... que de ne pas en détecter du tout. On accepte donc de sacrifier un peu la précision, en cherchant à maximiser le recall. Dans le cadre d'un problème de détection d'anomalies multilabel, notre objectif principal est de ne manquer aucune anomalie. C'est pourquoi, par la suite, nous choisissons de maximiser le rappel (recall) plutôt que le score (ou la précision).

3.1 ÉVALUATION COMPARATIVE DE MODÈLES MULTI-LABEL

Afin d'évaluer les performances de différents algorithmes de classification dans un contexte multi-label, nous avons mis en place une procédure d'évaluation systématique sur 10 splits (train/test) indépendants.

1. Fonction d'évaluation Nous avons défini une fonction `evaluate_model` permettant de :

- Entraîner un modèle donné sur les données d'entraînement ;
- Générer les prédictions sur les données de test ;
- Calculer le score de rappel (*recall*) macro-moyenné ;
- Générer un rapport de classification détaillé pour chaque label ;
- Extraire les matrices de confusion multi-label.

2. Comparaison de plusieurs modèles Une seconde fonction, `evaluate_models`, permet de comparer un ensemble de modèles. Chaque modèle est encapsulé dans un `MultiOutputClassifier`, afin de gérer la nature multi-label du problème. La liste suivante de modèles a été évaluée :

- Extra Trees Classifier
- AdaBoost Classifier
- Random Forest Classifier
- Bagging Classifier
- Gradient Boosting Classifier
- Ridge Classifier
- SGD Classifier
- Logistic Regression
- Support Vector Classifier (SVC)
- Gaussian Naive Bayes
- Decision Tree Classifier
- XGBoost Classifier
- LightGBM Classifier

3. Évaluation sur plusieurs splits Les performances de chaque modèle sont évaluées sur les 10 splits distincts de la base de données. Pour chaque split :

1. Les données d'entraînement et de test sont extraites ;
2. Tous les modèles sont entraînés puis évalués ;
3. Les résultats (score de rappel, rapport de classification, matrices de confusion) sont stockés.

Le recall moyen (de tous les splits) est ensuite calculé pour chaque modèle :

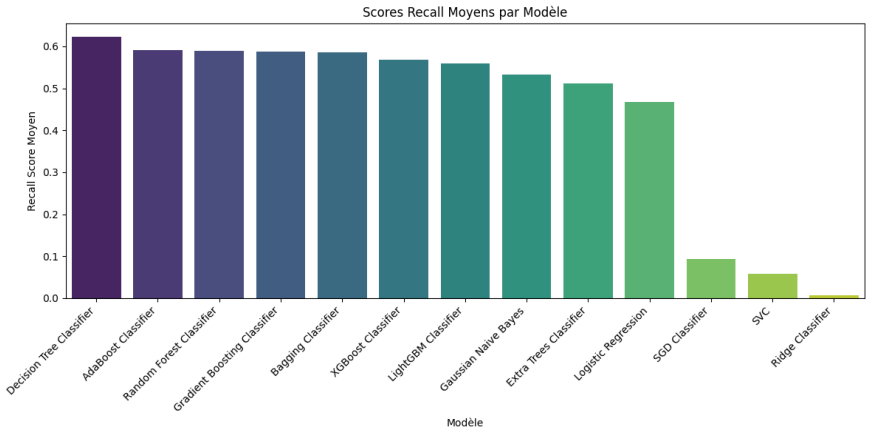


Figure 5: Recall moyen par modèle

En sortie, le Decision Tree Classifier ressort comme le modèle ayant le meilleur recall moyen (0.6236).

Ce résultat était attendu. Le classifieur basé sur les arbres de décision (Decision Tree) présente des performances intéressantes dans notre contexte, comme nous l'avons vu lors de l'analyse exploratoire des données (AED). En particulier, les visualisations par pairplots ont révélé que certaines classes, comme `PWF` et `OSF`, présentent des regroupements de points bien distincts dans des zones éloignées des distributions normales. Ces séparations visuelles suggèrent que les arbres de décision, capables de partitionner efficacement l'espace des caractéristiques, sont bien adaptés à ce type de structure.

3.2 Justification de l'utilisation de la courbe PRC

Afin d'évaluer plus finement les performances de notre modèle, nous nous focalisons sur la courbe de précision-rappel (PRC) pour le split 0, plutôt que sur la courbe ROC.

Dans le cas de classifications multilabels et fortement déséquilibrées la courbe ROC peut s'avérer trompeuse. En effet, elle prend en compte les vrais négatifs, qui sont surreprésentés lorsque la classe positive (anomalie) est rare. Cela peut conduire à des AUC (aire sous la courbe) élevées même si le modèle ne parvient pas à identifier efficacement les anomalies.

En revanche, la courbe PRC se concentre exclusivement sur la performance vis-à-vis de la classe positive. Elle met en lumière le compromis entre la précision et le rappel.

Cette courbe offre donc une évaluation plus fiable dans des contextes à forte asymétrie de classes, en particulier lorsqu'on ne doit pas manquer d'événements rares comme des défaillances.

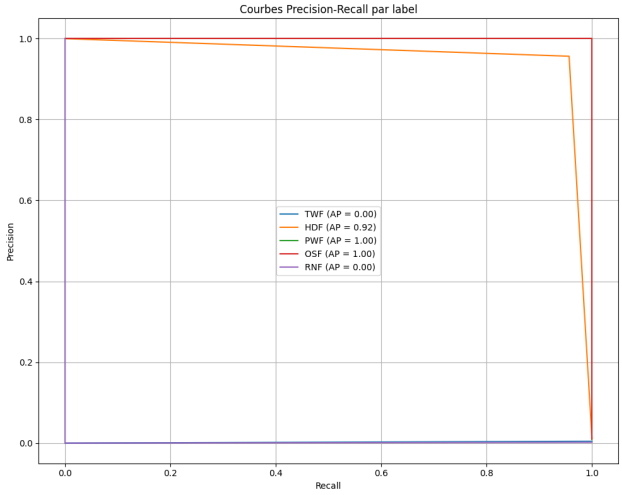


Figure 6: PRC pour le Decision Tree Classifier

La PRC est très bonne pour les pannes PWF et OSF. La détection d'anomalie est parfaite pour ces deux pannes (pour ce split). Elle est un peu moins pour HDF. Concernant RNF et TWF, la PRC est très mauvaise, ce qui est logique car ces pannes sont rares dans les données. Pour le moment nous allons écarter WTF car elles sont très rares. RNF étant générée aléatoirement, elle n'a pas de sens dans le contexte de l'analyse. Nous ne la garderons pas par la suite.

3.3 Optimisation des hyperparamètres pour PWF, HDF et OSF

Nous allons maintenant essayé d'optimiser les paramètres du modèle. Notre approche consiste à effectuer plusieurs essais indépendants (50 par exemple), chacun correspondant à une combinaison différente d'hyperparamètres. Pour chaque combinaison, nous procédons à une évaluation sur l'ensemble des splits disponibles (`df_train_list`, `df_test_list`) afin d'estimer la robustesse du modèle sur des données variées.

Nous utiliseront la librairie Optuna pour cela.

- Lors de chaque essai, un classifieur `DecisionTreeClassifier` est entraîné dans un `MultiOutputClassifier`, ce qui permet de traiter simultanément plusieurs étiquettes.
- L'évaluation du modèle se fait via le score de rappel moyen (`macro-recall`), qui mesure la capacité du modèle à détecter correctement les classes positives (pannes) tout en traitant chaque classe de manière équilibrée, indépendamment de leur fréquence.
- Le score obtenu est ensuite stocké et associé aux hyperparamètres utilisés.

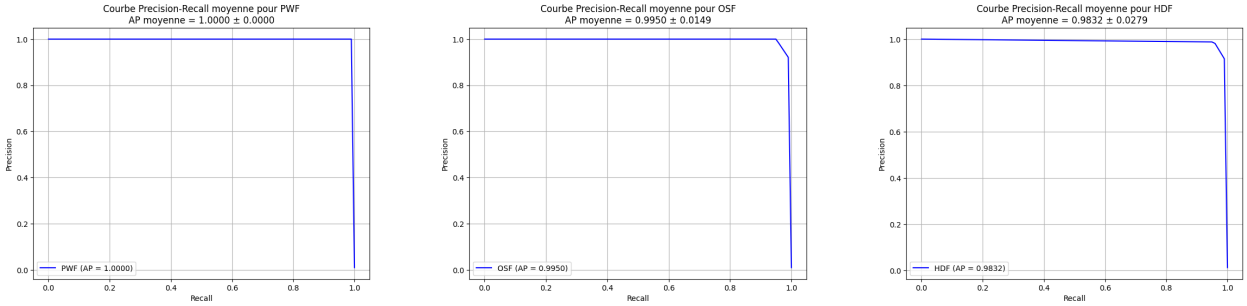
Ce processus permet d'identifier, parmi toutes les configurations testées, celle qui maximise le score de rappel moyen sur l'ensemble des splits. Cette stratégie garantit que le modèle est performant de manière globale, et non uniquement sur un sous-ensemble des données.

Les hyperparamètres optimisés sont :

- `max_depth` : profondeur maximale de l'arbre,
- `min_samples_split` : nombre minimal d'échantillons requis pour diviser un nœud,
- `min_samples_leaf` : nombre minimal d'échantillons dans une feuille.

L'optimisation nous donne un score recall moyen de 0.9833

Cependant, en traçant les courbes PRC moyenne (de tous les splits) :



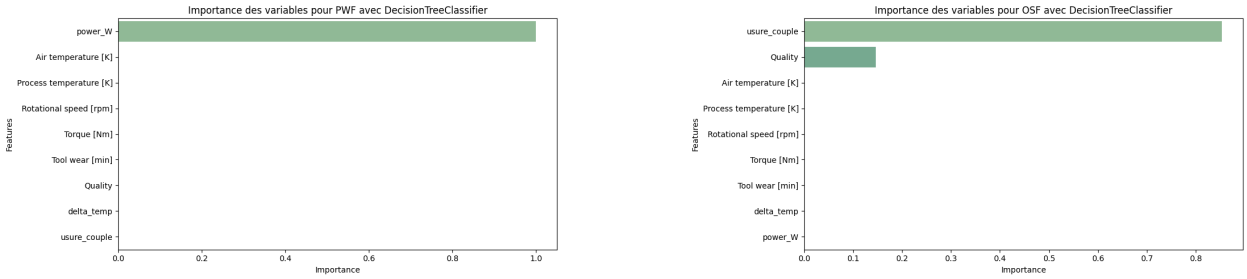
(a) PWF (b) OSF (c) HDF

Figure 7: PRC de PWF, OSF et HDF avec Decision Tree Classifier

Il y a beaucoup de splits avec des prédictions d'anomalies ratées pour HDF. On le remaque également par un AP (average precision) plus faible que pour OSF et PWF. On va donc essayer d'établir un modèle spécifique pour HDF, plus performant. On peut cependant reprendre ce modèle pour PWF, et OSF qui semble suffisament performant. En évaluant ce modèle sur tous les splits, nous obtenons un recall de 1 pour tous les splits, sauf le 10ème qui obtient un recall de 0.9750 (une anomalie râtée pour OSF).

Nous avons jugé ce modèle suffisamment performant.

L'analyse de l'importance des variables montrent que nos intuitions étaient correctes. La puissance définit la panne PWF, tandis que l'OSF est définie selon la qualité et la puissance :



(a) PWF (b) OSF

Figure 8: Importance des features du Decision Tree pour PWF et OSF.

3.4 Pourquoi le DecisionTreeClassifier fonctionne-il bien pour PWF et OSF ?

Le `DecisionTreeClassifier` s'est révélé particulièrement performant dans notre tâche de classification multi-étiquette,pour la détection des anomalies `PWF` et `OSF`. Cette efficacité peut être attribuée à plusieurs propriétés structurelles et statistiques du classifieur, en lien étroit avec la nature de nos données.

3.4.1 Capacité à modéliser des relations non linéaires et des interactions de variables

Un arbre de décision partitionne l'espace des caractéristiques $\mathcal{X} \subseteq \mathbb{R}^d$ en régions disjointes à l'aide de règles logiques simples :

$$x_j \leq t \quad \text{ou} \quad x_j > t,$$

où x_j est une variable d'entrée et t un seuil optimisé. Ce mécanisme permet de modéliser des frontières complexes et non linéaires, sans recourir à des transformations explicites des variables. Il est ainsi particulièrement adapté à des données où les clusters correspondant aux anomalies sont bien séparés visuellement, comme cela a été observé pour les étiquettes `PWF` et `OSF` dans les *pairplots* de l'analyse exploratoire (AED).

3.4.2 SEGMENTATION EFFICACE DES CLASSES MINORITAIRES

Lors de l'apprentissage, les arbres maximisent la réduction d'impureté dans les nœuds, selon par exemple l'indice de Gini :

$$G(S) = 1 - \sum_{k=1}^{K} p_k^2,$$

où p_k est la proportion d'éléments de la classe k dans le nœud S. L'algorithme sélectionne les divisions qui maximisent le gain d'impureté :

$$\Delta G = G(S) - \left(\frac{|S_L|}{|S|} G(S_L) + \frac{|S_R|}{|S|} G(S_R) \right).$$

Ce critère permet de créer des feuilles spécialisées contenant presque exclusivement des observations positives, ce qui favorise un *recall* élevé même en présence d'un fort déséquilibre entre classes.

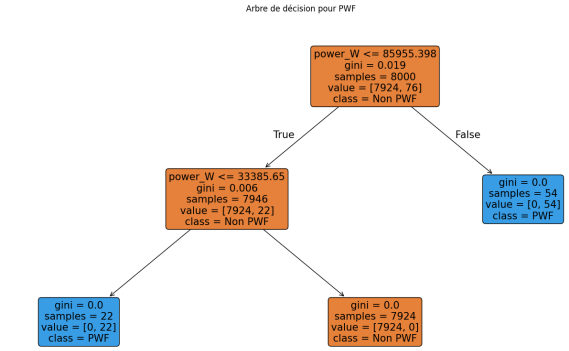
3.4.3 CORRESPONDANCE ENTRE CLUSTERS VISUELS ET RÈGLES DE DÉCISION

Les visualisations ont montré que certaines anomalies comme `PWF` ou `OSF` se situent dans des régions distinctes de l'espace des caractéristiques (vibration, température, etc.). Un arbre peut traduire cela par des règles de type :

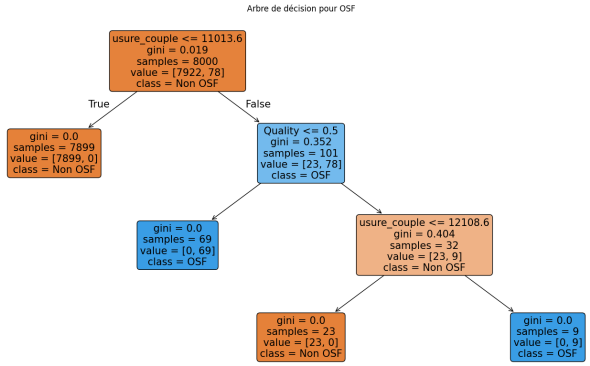
$$\text{Si } \texttt{vibration} > 0.8 \text{ et } \texttt{température} < 50 \Rightarrow \texttt{OSF} = 1.$$

De telles règles sont naturellement apprises par un arbre de décision.

Voici les arbres de décision obtenus pour PWF et OSF:



(a) PWF



(b) OSF

Figure 9: Arbres de décision pour PWF et OSF produits par Decisiton Tree Classifier (premier split)

3.4.4 POURQUOI `HDF` ET `TWF` SONT PLUS DIFFICILES À DÉTECTER

Les anomalies associées aux étiquettes `HDF` et `TWF` sont moins bien détectées pour plusieurs raisons :

- **Proximité avec la distribution normale** : Ces défauts partagent une zone de caractéristiques similaire avec les cas normaux, ce qui rend difficile la séparation par des règles simples.
- **Overlap significatif dans les pairplots** : Contrairement à `OSF` ou `PWF`, aucune séparation très clairement nette vers l'extérieur n'est visible pour `HDF` ou `TWF` dans les projections bivariées, ce qui suggère un manque de signaux discriminants dans les variables actuelles.
- **Faible fréquence d'occurrence** : Leur rareté dans les données d'entraînement peut conduire à un sur-apprentissage ou à un biais de sous-estimation dans les feuilles terminales de l'arbre.

3.4.5 Conclusion

L'arbre de décision s'adapte très bien à notre problème de classification multi-label déséquilibrée pour PWF et OSF, en exploitant sa capacité à apprendre des règles non linéaires, isoler les anomalies visuellement séparables, et tolérer des distributions non standards. Toutefois, son efficacité dépend fortement de la structure des données : les anomalies bien distinctes sont captées efficacement, tandis que les anomalies plus subtiles (comme `HDF` et `TWF`) requièrent potentiellement des approches plus complexes comme nous allons le voir.

3.5 Elaboration d'un modèle de prédiction pour HDF

Comme nous l'avons vu, le modèle optimisé pour PWF et OSF est le Decision Tree Classifier. Nous allons maintenant déterminer un meilleur modèle pour HDF afin d'avoir une meilleur détection d'anomalies (un meilleur AP de la PRC).

Nous avons repris le même processus pour trouver le meilleur modèle pour HDF (cette fois-ci, sans MultiOutput-Classifier). Sans surprise, le meilleur modèle pour HDF est LightGBM, un modèle à base d'arbres de décision et de gradient boosting.

Le score recall moyen obtenu était de 1. Nous avons donc tester pour tous les splits et n'avons eu un recall et une précision de 1 pour tous les jeux.

Nous pouvons alors supposés que ce modèle correspond très bien à la détection de HDF, et qu'il n'y a pas besoin d'optimisation des hyperparamètres. La PRC est également parfaite :

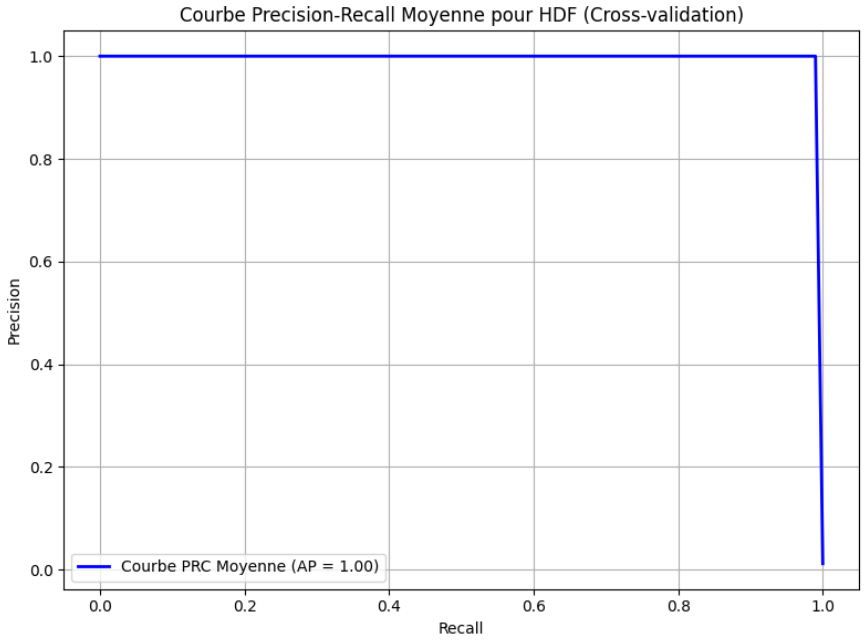


Figure 10: PRC pour HDF avec LightGBM

L'importance des features démontre aussi la nécessité d'avoir un modèle plus développé que le Decision Tree Classifier pour ce type de panne :

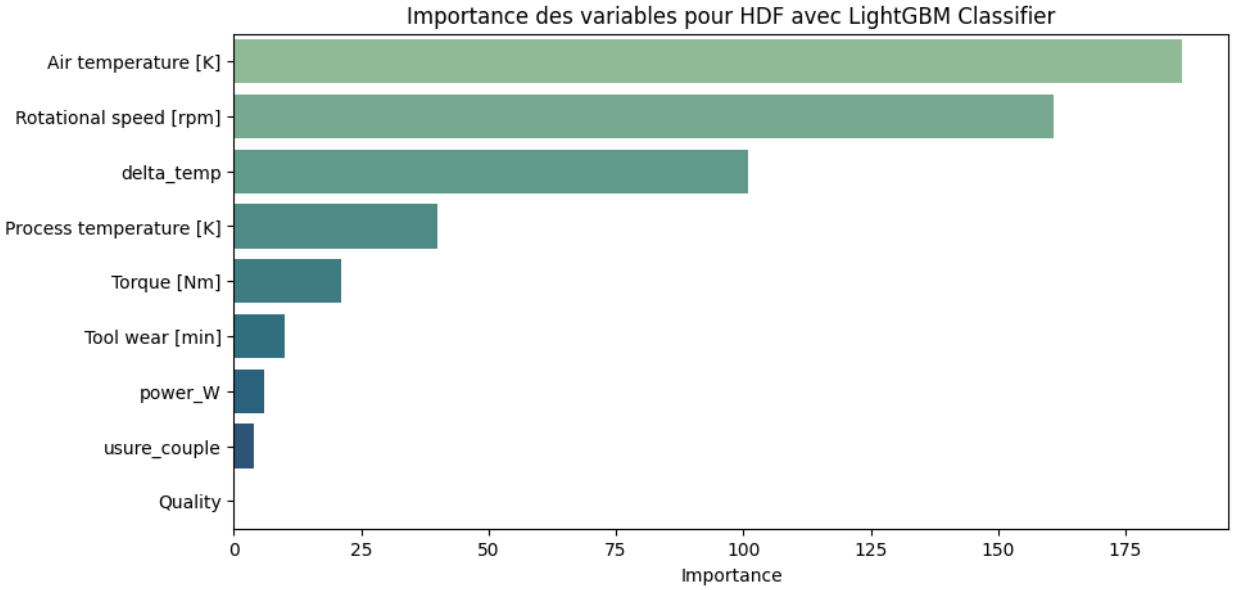


Figure 11: PRC pour HDF avec LightGBM

Comme précédemment énoncé, la température jouent des rôles prépondérants dans la détection de HDF, suivie par la vitesse de rotation et le couple. L'usure de l'outil semble avoir un impact moins significatif.

On remarque que, contraiement à PWF et OSF, plusieurs variables sont importantes pour la prédiction.

LightGBM est particulièrement efficace pour la détection de type de panne car il est capable de capturer des relations complexes et non linéaires entre les variables explicatives, ce qui correspond bien à la nature des données observées en analyse exploratoire. En effet, HDF dépend fortement de combinaisons spécifiques de variables telles que la différence entre la température du process et la température ambiante, ainsi que la vitesse de rotation. Ces conditions ne sont pas toujours séparables simplement par un seuil unique (comme pour la qualité ou la puissance de OSF et PwF), mais plutôt par des interactions multiples entre ces paramètres.

Contrairement à un arbre de décision simple, qui construit une hiérarchie de décisions basée sur des seuils uniques pour séparer les classes (c.f les arbres générés), ce modèle utilise une approche de boosting qui combine plusieurs arbres faibles pour former un modèle fort. Cette méthode permet au modèle de corriger progressivement ses erreurs en s'adaptant aux détails plus subtils des données.

LightGBM semble donc convenir parfaitement pour HDF, avec les paramètres par défaut.

3.6 Pourquoi LightGBM est performant pour la détection de HDF

Dans notre cadre de classification multi-étiquette et déséquilibrée, LightGBM s'est montré plus efficace que d'autres modèles pour détecter l'étiquette HDF, contrairement au DecisionTreeClassifier qui, bien que performant sur des classes comme PWF ou OSF, atteint rapidement ses limites sur des classes plus subtiles comme HDF. L'efficacité de LightGBM repose sur plusieurs aspects théoriques et structurels.

3.6.1 Apprentissage par histogramme et gain de précision

Ce modèle implémente un apprentissage basé sur histogramme, ce qui signifie que les valeurs continues sont discrétisées en bacs (bins), accélérant ainsi le calcul tout en réduisant le bruit. Cela permet de mieux exploiter des petites variations dans des variables continues qui peuvent être discriminantes pour des anomalies moins marquées comme HDF.

3.6.2 Croissance des arbres par feuille (*leaf-wise growth*)

Contrairement à un arbre de décision classique ou à XGBoost qui utilise une croissance *niveau-par-niveau* (level-wise), LightGBM construit ses arbres en ajoutant des feuilles là où le gain est maximal (*leaf-wise*). Cette stratégie

maximise le gain de réduction d'impureté à chaque itération :

$$\text{Gain}(j,t) = \frac{1}{2}\left(\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right) - \gamma,$$

où G et H sont les gradients et hessiennes pour la fonction de perte, et λ, γ sont des régularisations. Cette méthode permet de détecter des patrons fins dans les données, ce qui est crucial pour HDF où les signaux faibles sont souvent masqués dans un bruit de fond important.

3.6.3 Apprentissage en gradient boosting

Le principe du gradient boosting consiste à construire une fonction additive:

$$F(x) = \sum_{m=1}^{M} \alpha_m h_m(x),$$

où chaque h_m est un arbre faible et α_m est son poids. Cela permet à au modèle d'apprendre des corrections successives sur les erreurs précédentes, ce qui est particulièrement utile lorsque la frontière entre les classes est fine, comme dans le cas de HDF, où la première itération ne peut pas suffire à distinguer les anomalies.

3.6.4 Conclusion

LightGBM s'adapte parfaitement à la détection de défauts comme HDF, qui ne sont pas bien séparés linéairement ni visuellement. Sa croissance orientée par le gain d'information, son traitement du déséquilibre, et sa capacité à empiler des corrections successives permettent d'atteindre une performance de rappel significativement supérieure à celle d'autres modèles sur ce type d'étiquette complexe.

4 Détection des pannes TWF : approches, optimisation et analyse

4.1 Contexte et déséquilibre de classe

La panne `TWF` constitue une classe fortement déséquilibrée dans le jeu de données, avec très peu d'occurrences positives comparées aux négatives. Cette rareté complique considérablement l'apprentissage des modèles standards, souvent biaisés en faveur de la classe majoritaire.

4.2 Choix de la métrique d'évaluation et hypothèse de départ

Face à ce déséquilibre, nous avons choisi d'optimiser les modèles à l'aide du score `F1-macro`. Contrairement au `F1-micro`, qui favorise la classe majoritaire en agrégant toutes les prédictions, le `F1-macro` évalue la performance de chaque classe séparément et leur accorde un poids équitable. Cela permet de ne pas négliger la classe `TWF`, pourtant minoritaire mais critique.

Nous avons également formulé une hypothèse de départ guidée par une logique de criticité : il vaut mieux prédire à tort une panne (faux positif) qu'en rater une réelle (faux négatif). Une fausse alerte conduit simplement à une vérification manuelle, tandis qu'une panne non anticipée peut interrompre la chaîne de production, générer des retards, des pertes financières et affecter la planification logistique. Le rappel est donc prioritaire, mais il doit être équilibré par une précision minimale pour conserver la crédibilité du système.

4.3 Comparaison initiale des modèles

Divers modèles (`LogisticRegression`, `SVC`, `SGDClassifier`, etc.) ont été évalués à l'aide des matrices de confusion et des rapports de classification. Aucun ne parvient initialement à bien concilier précision et rappel sur la classe `TWF`. Le `SGDClassifier` offre le meilleur compromis avec un `F1`-score acceptable après ajustement, mais la performance globale reste insuffisante pour une mise en production fiable.

4.4 Optimisation et pénalisation des faux négatifs

Afin de mieux gérer le déséquilibre des classes et d'améliorer la détection des pannes `TWF`, plusieurs techniques ont été mises en œuvre :

- optimisation des hyperparamètres via `Optuna`, en maximisant le `F1-macro` ;
- ajustement du seuil de décision pour influencer le compromis précision/rappel ;
- duplication simple des exemples positifs pour rééquilibrer la distribution ;
- introduction d'une pénalisation explicite des faux négatifs dans la fonction de coût.

Dans les algorithmes supervisés, la fonction de coût guide l'apprentissage en sanctionnant les erreurs. Or, dans un contexte déséquilibré, les erreurs sur la classe minoritaire ont peu d'impact sur la perte globale, ce qui peut conduire à leur négligence. Pour corriger cela, la pénalisation consiste à attribuer un poids plus élevé aux erreurs sur cette classe.

Techniquement, cette pondération est souvent intégrée via les paramètres `class_weight` ou `scale_pos_weight`, qui multiplient l'impact des exemples positifs dans la fonction de perte. Cette méthode s'est montrée particulièrement efficace lorsqu'aucun suréchantillonnage artificiel n'est effectué.

4.5 Suréchantillonnage : random oversampling et SMOTE

Deux techniques de suréchantillonnage ont été testées afin d'atténuer le déséquilibre de classes :

- Le **random oversampling**, qui consiste à dupliquer aléatoirement des exemples de la classe minoritaire. Cette méthode permet d'obtenir des résultats presque parfaits en termes de rappel et de précision sur les données de test. Toutefois, ces performances soulèvent des interrogations : s'agit-il d'une réelle capacité généralisable du modèle, ou bien d'un surapprentissage (*overfitting*) dû à la redondance artificielle des données ?
- **SMOTE** (Synthetic Minority Over-sampling Technique), qui génère de nouveaux exemples synthétiques de la classe minoritaire par interpolation. Cette méthode améliore significativement le rappel, mais ne permet pas de maîtriser le taux de faux positifs, ce qui se traduit par une précision relativement faible. Le score `F1` global reste modéré (`F1` ≈ 0.56).

4.6 Modèle final retenu

Le modèle retenu est un **Gradient Boosting avec pénalisation des faux négatifs**. Il obtient un **F1-macro de 0.5683**, un **rappel de 0.8889** et une **précision de 0.0870**, pour un seuil de décision optimisé à 0.62. Il parvient à détecter la majorité des pannes tout en limitant le nombre de fausses alertes.

Pour renforcer la présence des cas rares sans générer de données artificielles, nous avons également appliqué une duplication simple des lignes positives (`TWF`) dans les données d'apprentissage. Cette stratégie agit comme une forme implicite de pondération, tout en maintenant l'intégrité des données.

Les modèles utilisant des techniques comme `SMOTE` ou le `random oversampling` ont été écartés malgré leurs résultats très élevés, car la création de données artificielles pose des problèmes d'interprétabilité et de validation dans des contextes critiques.

4.7 Fonctionnement du Gradient Boosting

Le *Gradient Boosting* est une méthode d'ensemble reposant sur l'agrégation séquentielle de modèles faibles — souvent des arbres de décision peu profonds — pour construire un modèle fort. À chaque itération, un nouvel arbre est entraîné pour corriger les erreurs des prédictions précédentes.

L'algorithme cherche à minimiser une fonction de perte différentiable $\mathcal{L}(y, F(x))$ en construisant une approximation additive :

$$F_t(x) = F_{t-1}(x) + \gamma_t h_t(x)$$

où :

- $F_t(x)$ est le modèle à l'étape t ;
- $h_t(x)$ est l'arbre ajouté à l'itération t ;
- γ_t est un facteur de régulation optimisé à chaque étape.

Les arbres sont ajustés pour prédire les résidus de la fonction de perte, c'est-à-dire :

$$r_i^{(t)} = - \left. \frac{\partial \mathcal{L}(y_i, F(x_i))}{\partial F(x_i)} \right|_{F(x)=F_{t-1}(x)}$$

Chaque nouvel arbre vient corriger les erreurs des précédents en suivant la direction opposée du gradient local. Ce processus est répété jusqu'à convergence ou jusqu'à un nombre fixe d'itérations. Des hyperparamètres comme le taux d'apprentissage (`learning_rate`), la profondeur maximale (`max_depth`) et le nombre total d'arbres (`n_estimators`) régulent la complexité du modèle.

Le Gradient Boosting peut intégrer directement des poids de classes dans la fonction de perte, ce qui en fait un algorithme particulièrement adapté aux problèmes de classification déséquilibrée.

4.8 Conclusion

L'étude a permis d'identifier un modèle performant pour détecter les pannes `TWF`, malgré un fort déséquilibre de classes. Le Gradient Boosting avec pénalisation des faux négatifs offre le meilleur compromis pour un usage industriel, tout en laissant des pistes d'amélioration au niveau de la précision.

5 Modèles Finaux Retenus par Type de Défaillance

Après une évaluation avancée, des modèles spécifiques ont été choisis pour chaque type de pannes, en fonction de leurs caractéristiques et des compromis entre précision et rappel.

5.1 Modèle pour PWF et OSF : DecisionTreeClassifier

Les pannes `PWF` et `OSF` ont montré des regroupements distincts dans les données, les rendant bien adaptés aux modèles basés sur des arbres.

- Le `DecisionTreeClassifier` s'est avéré le plus performant il a atteint un recall moyen de 0.9833. Les courbes Precision-Recall (PRC) pour `PWF` et `OSF` (*AP* de 1.0000 et 0.9950) sont quasi parfaites.
- La capacité à modéliser des relations non linéaires, à partitionner efficacement l'espace des caractéristiques, et à se concentrer sur les classes minoritaires correspond parfaitement à la nature de ces pannes. L'analyse d'importance des variables a confirmé que la `puissance` est clé pour `PWF`, et la `qualité` et la `puissance` pour `OSF`.

5.2 Modèle pour HDF : LightGBMClassifier

La détection de `HDF` s'est montrée plus complexe en raison de signaux plus subtils et d'un chevauchement avec les données normales.

- Le `LightGBMClassifier` a été le modèle le plus efficace.
- Après optimisation, `LightGBM` a obtenu un recall moyen de 1.0 et une précision de 1.0 pour `HDF` (PRC moyenne avec *AP* de 1.00).
- Sa supériorité est due à ses propriétés avancées : apprentissage par histogramme, croissance d'arbres par feuille (*leaf-wise growth*) et mécanisme de gradient boosting, lui permettant de capturer les relations complexes et non linéaires de `HDF`, où les températures et la vitesse/couple jouent des rôles cruciaux.

5.3 Les cas de TWF et RNF

- **TWF (Tool Wear Failure) :** Cette panne est extrêmement rare et difficile à prédire avec les données actuelles. Malgré les efforts d'optimisation et l'utilisation de `Gradient Boosting` avec pénalisation des faux négatifs (Recall de 0.8889, Précision de 0.0870), la précision reste un défi et sa prédiction est incertaine pour un usage industriel.
- **RNF (Random Failure) :** Étant par définition une défaillance aléatoire avec très peu de lien avec les variables mesurées, sa prédiction n'a pas de sens opérationnel et elle a été écartée.

5.4 Evaluation finale

Le modèle établit nous fournit la PRC (moyenne, sur l'ensemble des splits) suivante :

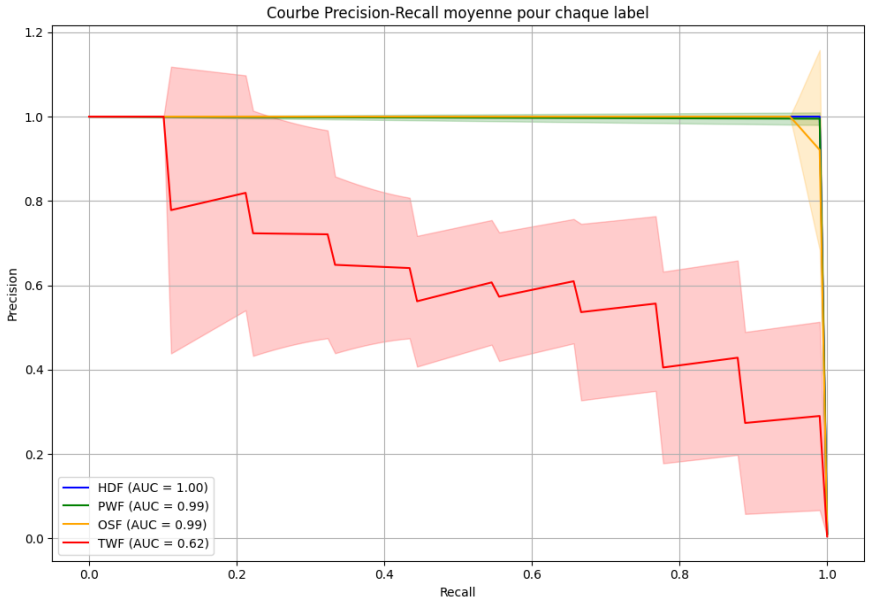


Figure 12: PRC pour HDF avec LightGBM

Le modèle semble très bon pour OSF, PWF et HDF, mais malheureusement moins pour TWF.

6 Conclusion Générale

Cette étude a confirmé la faisabilité de la prédiction des défaillances industrielles grâce à une approche de classification multi-label.

L'AED a été cruciale, révélant la nature distincte et le déséquilibre des différents types de pannes. Pour `PWF` et `OSF`, le `DecisionTreeClassifier` s'est montré très efficace, capitalisant sur leurs séparations claires dans l'espace des caractéristiques. Pour `HDF`, le `LightGBMClassifier` a démontré sa supériorité en modélisant des relations plus complexes et subtiles.

Cependant, il est apparu que `TWF` et `RNF` sont extrêmement difficiles, voire impossibles, à prédire efficacement avec les données et les méthodes actuelles, en raison de leur rareté (`TWF`) ou de leur nature intrinsèquement aléatoire (`RNF`).

En conclusion, ce projet a identifié des modèles performants pour `PWF`, `OSF` et `HDF`, ouvrant la voie à une maintenance prédictive plus proactive et à une réduction significative des arrêts machines non planifiés. Les défis posés par `TWF` soulignent la nécessité d'explorations futures avec des données additionnelles (voire des techniques plus avancées).

Contributions

Ce projet est le fruit d'un travail d'équipe entièrement collaboratif. Nous avons tous les trois activement participé à chaque étape : de l'exploration initiale des données à la construction et l'optimisation des modèles, en passant par l'analyse et l'interprétation des résultats (33% par personne)