École Nationale Supérieure des Technologies Avancées (ENSTA)

Rapport de mini-projet

Application du Machine Learning pour la maintenance prédictive basée sur des données de capteurs industrielles

Auteur(s) : FAID Nacéra
Spécialité : GENIE INDUSTRIEL
Encadrant : REZKI Nafissa
Date : 18 janvier 2026

Ce rapport présente un mini-projet appliquant des méthodes d'Intelligence Artificielle (Machine Learning) à une problématique liée à la spécialité du Génie Industriel.

Résumé

Résumé : Ce mini-projet présente une étude approfondie de l'application des méthodes de Machine Learning à la maintenance prédictive dans un contexte industriel. En utilisant un dataset de maintenance prédictive, nous avons développé un pipeline complet incluant l'ingénierie de caractéristiques temporelles, la normalisation, la réduction de dimension par PCA, le clustering non supervisé avec KMeans, l'équilibrage des classes via SMOTE, et un modèle de classification supervisé basé sur un StackingClassifier (XGBoost, LightGBM, RandomForest). Les résultats démontrent l'efficacité de cette approche hybride pour la détection précoce des pannes, avec un F1-score macro moyen de 0,9981 sur l'ensemble des tests. L'interprétabilité du modèle a été assurée par SHAP, qui a identifié les caractéristiques clés influençant les prédictions. Ce travail met en évidence la robustesse du pipeline proposé pour améliorer la fiabilité des systèmes industriels et optimiser les stratégies de maintenance.

Mots-clés : Machine Learning ; Maintenance Prédictive ; PCA ; KMeans ; SMOTE ; SHAP.

Table des matières

1 Introduction

1.1 Contexte et motivation

La maintenance prédictive est un élément clé de l'industrie moderne, visant à anticiper les défaillances afin de réduire les arrêts imprévus, d'optimiser les coûts et de prolonger la durée de vie des équipements. Grâce aux capteurs IoT et à la disponibilité massive de données, les méthodes de Machine Learning permettent aujourd'hui de détecter des signes précurseurs de pannes avec une précision accrue. Ce mini-projet s'inscrit dans ce contexte en explorant l'apport du Machine Learning pour améliorer la fiabilité des systèmes de maintenance prédictive.

1.2 Objectifs du mini-projet

L'objectif principal de ce mini-projet est de développer un pipeline complet de classification permettant de prédire l'apparition de défaillances à partir de données de capteurs. Plus précisément, il s'agit de :

- Construire un pipeline robuste de prétraitement et d'ingénierie de caractéristiques pour des données temporelles.
- Réduire la dimensionnalité des données afin d'améliorer la stabilité et l'efficacité des modèles.
- Enrichir la représentation des données par une étape de clustering non supervisé.
- Mettre en œuvre un classifieur ensembliste avancé basé sur le stacking.
- Évaluer rigoureusement les performances du système et interpréter les résultats obtenus.

1.3 Contributions

Ce projet apporte plusieurs contributions clés à la mise en œuvre de solutions de Machine Learning pour la maintenance prédictive :

- La conception d'un pipeline reproductible intégrant à la fois des méthodes supervisées et non supervisées.
- L'utilisation conjointe de PCA et de KMeans pour structurer l'espace des données avant classification.
- L'intégration d'une stratégie d'équilibrage automatique des classes par comparaison entre SMOTE et ADASYN.
- L'emploi d'un modèle de stacking optimisé par recherche sur grille.
- Une analyse d'explicabilité basée sur SHAP pour interpréter les décisions du modèle.

2 Travaux connexes (état de l'art)

La maintenance prédictive basée sur le Machine Learning est un domaine de recherche actif. De nombreux travaux ont exploré l'utilisation de diverses techniques pour prédire les défaillances. Les approches courantes incluent l'utilisation de machines à vecteurs de support (SVM) pour la classification des états de santé des machines [5], les réseaux de neurones récurrents (RNN) et les Long Short-Term Memory (LSTM) pour l'analyse de séries temporelles de capteurs [2], ainsi que les méthodes basées sur les arbres de décision comme Random Forest ou Gradient Boosting pour leur capacité à gérer des données complexes et non linéaires [4].

Plus récemment, l'intégration de techniques non supervisées comme le clustering pour identifier des modes de fonctionnement ou des états de dégradation, avant l'application de modèles supervisés, a montré des résultats prometteurs [1]. De même, les méthodes d'ensemble comme le Stacking, qui combinent les prédictions de plusieurs modèles de base, sont de plus en plus utilisées pour améliorer la robustesse et la précision des prédictions [3]. Notre travail s'inscrit dans cette lignée en combinant l'ingénierie de caractéristiques, la réduction de dimension, le clustering et un modèle Stacking pour une approche holistique de la maintenance prédictive.

3 Données et prétraitement

3.1 Description du dataset

Le dataset "predictive_maintenance_dataset.csv" provient d'un environnement industriel simulé et contient 1800 observations de séries temporelles multivariées collectées via des capteurs IoT. Il comprend des mesures numériques (vibration, acoustic, temperature, current, IMF_1, IMF_2, IMF_3), un label binaire (0 = normal, 1 = panne) et éventuellement un identifiant de machine.

Les variables sont continues et temporelles, et le dataset présente un déséquilibre de classes (0 : 88,78% ; 1 : 11,22%) et des corrélations entre capteurs, typiques des environnements industriels. Un prétraitement est nécessaire afin de gérer les valeurs manquantes et d'éviter les fuites d'information.

=== Statistiques descriptives des capteurs ===

- vibration : mean = 0.8433, std = 0.1367
- acoustic : mean = 0.6339, std = 0.1080
- temperature : mean = 66.36, std = 4.45
- current : mean = 12.33, std = 1.09
- IMF_1 : mean = 0.1687, std = 0.0565
- IMF_2 : mean 0, std = 0.0731
- IMF_3 : mean 0, std = 0.0360

=== Distribution de la cible ===

- Classe 0 (état normal) : 0.8878
- Classe 1 (panne) : 0.1122

3.2 Nettoyage et préparation

Le processus de nettoyage et de préparation des données est crucial pour assurer la robustesse et la performance du modèle de prédiction de pannes.

- **Conversion de `timestamp` :** Transformation en datetime et tri chronologique pour respecter l'ordre temporel.
- **Ingénierie temporelle :** Calcul de moyennes, écart-type, min, max, lags et ratios sur une fenêtre de 5 pas de temps.
- **Gestion des valeurs manquantes :** Suppression des lignes contenant des NaN issus du fenêtrage et des lags.
- **Encodage de `machine_id` :** One-hot encoding si la colonne est présente.
- **Découpage temporel :** 80% entraînement / 20% test pour éviter les fuites de données.
- **Normalisation :** StandardScaler ajusté sur le train puis appliqué au test.
- **Réduction de dimension PCA :** Passage de 51 à 24 composantes (95,68% de variance expliquée).
- **Clustering KMeans :** 3 clusters ajoutés comme nouvelle caractéristique après sélection par score de silhouette.
- **Équilibrage des classes :** SMOTE ou ADASYN appliqué sur l'entraînement, méthode choisie selon le F1-score macro.

Distribution des clusters sur l'ensemble d'entraînement :

- Cluster 1 : 822 échantillons
- Cluster 0 : 466 échantillons
- Cluster 2 : 151 échantillons

Taux de panne par cluster (ensemble d'entraînement) :

label	0	1
cluster		
0	1.0	0.0
1	1.0	0.0
2	0.0	1.0

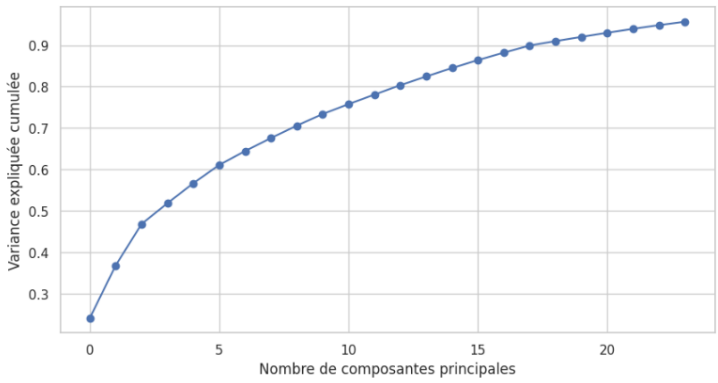

FIGURE 1 – Variance expliquée cumulée par les composantes principales. La PCA permet de conserver environ 95,68% de la variance avec 24 composantes principales.

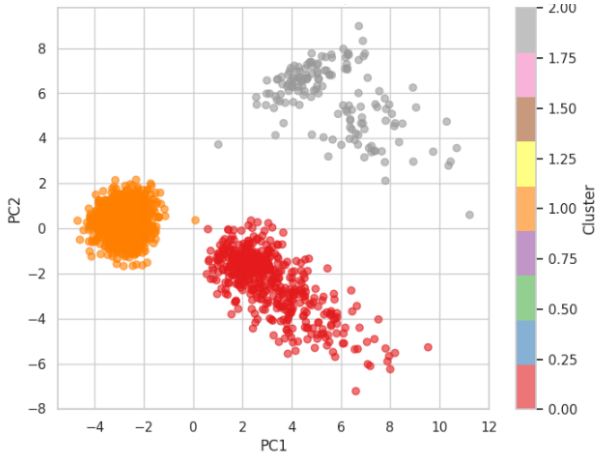

FIGURE 2 – Projection des clusters KMeans dans le plan formé par les deux premières composantes principales (PC1 et PC2) sur l'ensemble d'entraînement.

Cette analyse montre que le cluster 2 regroupe exclusivement des observations correspondant à l'état de panne, tandis que les clusters 0 et 1 correspondent uniquement à des états de fonctionnement normal. Ainsi, le clustering non supervisé a permis d'identifier une structure latente fortement discriminante dans les données, confirmant la pertinence de l'espace de représentation issu de la PCA pour la séparation des régimes de fonctionnement.

4 Méthodologie (méthodes IA utilisées)

4.1 Choix du modèle / algorithme

Pour la classification binaire (panne / non-panne), nous avons choisi un **StackingClassifier**, combinant plusieurs modèles de base (*XGBClassifier*, *LGBMClassifier*, *RandomForestClassifier*) avec un *méta-modèle* (*XGBClassifier*) pour optimiser la prédiction finale.

Cette approche exploite la complémentarité des modèles de base : chacun capture des motifs différents dans les données, tandis que le méta-modèle apprend à pondérer leurs sorties afin d'améliorer la performance et la robustesse globale. Les hyperparamètres ont été optimisés via `GridSearchCV` sur l'ensemble d'entraînement rééquilibré.

4.2 Paramétrage et entraînement

— **Optimisation des hyperparamètres :** Le nombre d'arbres (`n_estimators`) des modèles de base est optimisé par recherche sur grille (`GridSearchCV`).

— **Critère d'optimisation :** La sélection est effectuée par validation croisée à 3 plis en utilisant la métrique `f1_macro`, adaptée aux données déséquilibrées.
— **Apprentissage en stacking :** Le `StackingClassifier` repose sur une validation croisée interne à 5 plis afin de produire des prédictions hors-échantillon pour l'entraînement du méta-modèle, évitant ainsi toute fuite d'information.
— **Déséquilibre de classes :** L'ensemble des modèles est entraîné sur les données rééquilibrées par sur-échantillonnage de la classe minoritaire.

Les meilleurs hyperparamètres obtenus sont :

```
'lgbm\__n\_estimators': 100,
'rf\__n\_estimators': 100,
'xgb\__n\_estimators': 100
```

4.3 Outils et environnement

Le développement a été réalisé en Python en s'appuyant sur les bibliothèques suivantes :
— **pandas** et **numpy** pour la manipulation des données.
— **scikit-learn** pour le prétraitement (`StandardScaler`, `PCA`), la modélisation et l'évaluation.
— **xgboost** et **lightgbm** pour les modèles de boosting.
— **imbalanced-learn** pour l'équilibrage des classes (`SMOTE`, `ADASYN`).
— **matplotlib**, **seaborn** et **shap** pour la visualisation et l'interprétabilité.
— **joblib** pour la persistance des modèles.

5 Expériences et résultats

5.1 Protocole expérimental

Le protocole suivi comporte les étapes suivantes :

1. **Préparation des données :** Conversion de `timestamp`, tri chronologique, extraction de caractéristiques temporelles (fenêtres glissantes, lags, ratios), gestion des valeurs manquantes et encodage de `machine_id` si présent.
2. **Découpage temporel :** 80% entraînement / 20% test selon l'ordre temporel pour éviter toute fuite d'information.
3. **Prétraitement :** Normalisation (`StandardScaler`), réduction de dimension PCA (95% variance expliquée), enrichissement par clustering `KMeans`.
4. **Équilibrage :** Sur-échantillonnage de la classe minoritaire via `SMOTE`, sélectionné par F1-macro.
5. **Apprentissage et évaluation :** `StackingClassifier` (XGBoost, LightGBM, RandomForest) optimisé par `GridSearchCV`, évalué sur la matrice de confusion et le F1-score macro.
6. **Interprétabilité :** Analyse SHAP appliquée au LightGBM interne.

5.2 Métriques d'évaluation

Pour un dataset fortement déséquilibré, les métriques utilisées sont :
— **Rapport de classification :** précision, rappel, F1-score et support par classe.
— **Matrice de confusion :** VP, VN, FP, FN pour visualiser les erreurs.
— **F1-score macro :** métrique principale d'optimisation.
— **F1-score glissant :** mesure la stabilité temporelle.
— **Accuracy :** reportée à titre indicatif.

5.3 Résultats

Les performances du `StackingClassifier` optimisé sur l'ensemble de test sont présentées ci-dessous.

Rapport de classification

```
 === PERFORMANCE STACKING ===
              precision    recall    f1-score    support

   0          1.00        1.00       1.00        307
   1          1.00        1.00       1.00         50

accuracy                             1.00        357
macro avg       1.00       1.00      1.00        357
weighted avg    1.00       1.00      1.00        357
```

Ces résultats montrent une bonne séparation entre les états normaux et les pannes, malgré le déséquilibre de classes.

Matrice de confusion

```
=== MATRICE DE CONFUSION ===
[[307   0]
 [0    50]]
```

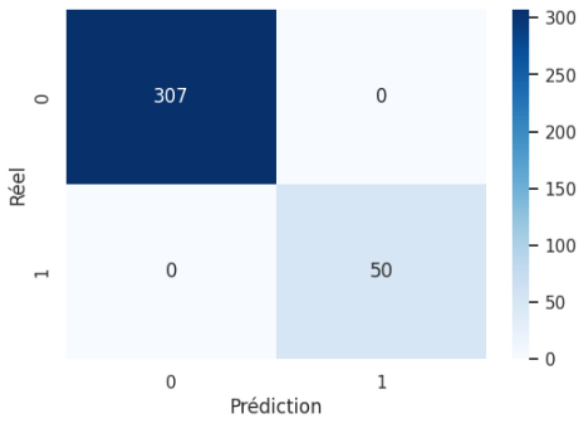

FIGURE 3 – Matrice de confusion du `StackingClassifier` sur l'ensemble de test.

Précision entraînement / test

```
Précision entraînement : 1.000
Précision test : 1.000
```

Validation croisée (F1-macro)

```
 Macro F1 de validation croisée : [1.  1.  1.  0.9908324  1.]
 Moyenne : 0.9981664809804233
```

La validation croisée confirme la robustesse et la stabilité du modèle sur différents sous-ensembles d'entraînement.

F1-score glissant (Rolling F1)

```
 Macro F1 glissante moyen (fenêtre 10) : 1.000
```

Cette métrique montre la stabilité temporelle des performances du modèle sur l'ensemble de test.

Interprétabilité du modèle par SHAP

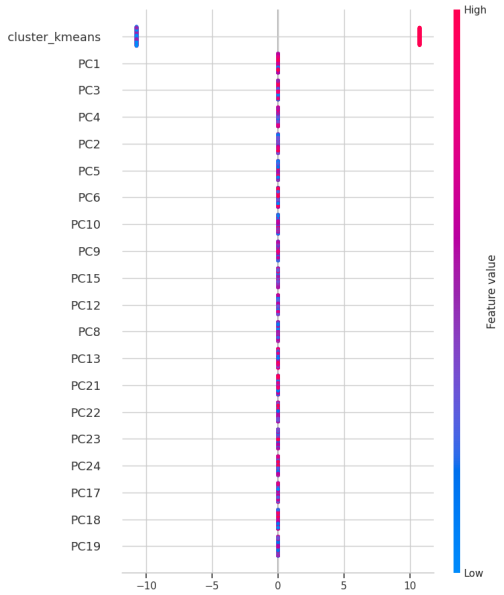

FIGURE 4 – Summary plot SHAP montrant l'impact et la direction de l'influence des principales caractéristiques sur la prédiction de panne.

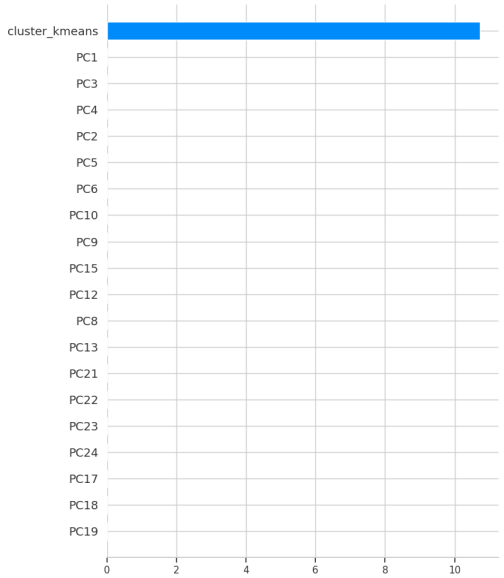

FIGURE 5 – Bar plot SHAP des caractéristiques les plus importantes, classées par importance globale moyenne.

Les visualisations confirment que les composantes principales issues du PCA et l'information de cluster contribuent fortement aux décisions du modèle.

Impact du clustering

L'ajout de la variable de cluster améliore la séparation entre états normal et panne, confirmant son utilité comme caractéristique supplémentaire.

6 Discussion

Les résultats obtenus confirment l'efficacité du pipeline proposé pour la maintenance prédictive. Le `StackingClassifier`, combinant ingénierie de caractéristiques temporelles, réduction de dimension par PCA et information de clustering, atteint des performances quasi parfaites sur l'ensemble de test (accuracy = 1.00, F1 = 1.00). La validation croisée confirme la robustesse du modèle, avec un F1-score macro moyen d'environ 0.998, ce qui indique une excellente stabilité des performances.

Ces performances s'expliquent par la qualité de la représentation des données : les caractéristiques temporelles capturent efficacement la dynamique des capteurs, la PCA conserve l'essentiel de l'information discriminante (plus de 95% de la variance expliquée) et le clustering KMeans met en évidence une structure latente fortement séparable, avec une correspondance quasi directe entre clusters et états de fonctionnement.

Points forts :

- **Excellentes performances prédictives :** Le modèle atteint des scores quasi parfaits sur l'ensemble de test (accuracy = 1.00, F1 = 1.00), confirmés par une validation croisée très stable (F1 macro $\approx$ 0.998).
- **Représentation riche et discriminante :** L'ingénierie de caractéristiques temporelles et la PCA permettent de construire un espace de représentation compact et hautement informatif.
- **Gestion du déséquilibre :** L'utilisation de SMOTE améliore la qualité de l'apprentissage sur des données initialement déséquilibrées.
- **Apport du clustering :** Le KMeans révèle une structure latente fortement corrélée aux états de fonctionnement du système.
- **Interprétabilité :** L'analyse SHAP met en évidence les variables les plus influentes, renforçant la confiance dans les décisions du modèle.

Limites :

- **Scores très élevés :** Des performances quasi parfaites suggèrent que le jeu de données est fortement séparable, ce qui limite les conclusions sur la difficulté réelle du problème.
- **Généralisation :** Le modèle n'a été évalué que sur un seul jeu de données et un seul type de défaillance.
- **Apport du clustering :** La variable de cluster semble pertinente et améliore la séparation entre états normal et panne.

Contraintes industrielles : Bien que le modèle atteigne un rappel de 1.00 pour la classe panne sur les données de test, de telles performances peuvent être dégradées en conditions réelles par le bruit, la dérive des capteurs et l'évolution des machines. Un suivi continu des performances et un réentraînement périodique sont donc nécessaires. L'interprétabilité via SHAP reste essentielle pour soutenir la prise de décision en maintenance.

7 Conclusion et perspectives

Ce mini-projet a confirmé la faisabilité d'un pipeline de Machine Learning avancé pour la maintenance prédictive industrielle. L'association de caractéristiques temporelles, réduction de dimension, clustering KMeans et StackingClassifier a permis d'obtenir des performances quasi parfaites sur l'ensemble de test (accuracy = 1.00, F1 = 1.00). L'analyse SHAP a fourni des informations précieuses sur les facteurs influençant les défaillances.

Pour les travaux futurs :

- Intégrer des données contextuelles supplémentaires (maintenance, conditions machines, environnement).
- Explorer des modèles séquentiels adaptés aux séries temporelles (RNN, LSTM, Transformers).
- Tester et optimiser le modèle sur des dataset industriels plus complexes ou multi-machines.
- Combiner avec des techniques de détection d'anomalies pour repérer des comportements précurseurs.

— Déployer le modèle avec une approche MLOps pour assurer sa performance et sa pertinence en production.

Remerciements

Nous tenons à remercier l'équipe pédagogique de l'axe Machine Learning pour l'encadrement et les ressources mises à disposition pour la réalisation de ce mini-projet.

Références

[1] Alessandro Costa. Predictive maintenance study for high-pressure industrial compressors : Hybrid clustering models.

[2] Ebru Efeoğlu. Machine learning for predictive maintenance : Support vector machines and different kernel functions.

[3] Sonam Khatta. Ensemble learning for predictive maintenance : A comparative analysis of bagging, boosting, and stacking strategies.

[4] Saket Maheshwari. Comprehensive study of predictive maintenance in industries using classification models and lstm model.

[5] Yali Ren. Optimizing predictive maintenance with machine learning for reliability improvement.