Sommaire :

Introduction :

* Présentation du dataset
* Motivation du projet : personnelle , contexte de l’industrie (different type de maintenance)

Data description :

* Explication des variables : qualitative, quantitative
* Analyse : distribution, valeur manquante
* Statistique des variables univariés
* Classes desequilibrés

Data Processing :

* Pipeline de transformation
* Décomposition du dataset ; training, test, validation

Training :

* Recherche des hyperparamètres
* Choix et explication des métriques
* Structure retenu

Explicabilité

* Différents types d’algo

Conclusion

1. **Introduction :**

**L'industrie moderne** est en pleine mutation, avec l'avènement de **l'Industrie 4.0** et l'intégration croissante de technologies numériques. Dans ce contexte, la maintenance des machines devient un enjeu crucial pour garantir la continuité de la production et la sécurité des opérateurs.

Différents types de maintenance existent :

* Maintenance curative : intervention après la panne
* Maintenance préventive : intervention à intervalles réguliers
* Maintenance conditionnelle : intervention basée sur l'état de la machine
* Maintenance prédictive : anticipation des pannes grâce à des modèles prédictifs

Le Deep Learning peut s'avérer être un outil puissant pour la maintenance prédictive, en particulier pour la classification des défauts des machines.

Le dataset "Machine Predictive Maintenance Classification" disponible sur Kaggle est un ensemble de données synthétique qui reflète les situations de maintenance prédictive rencontrées dans l'industrie. Il s'agit d'un dataset binaire de classification de défaillance, ce qui signifie qu'il peut être utilisé pour prédire si une machine est en panne ou non.

Dans cet article, nous explorerons une application du Deep Learning pour la classification de défaut d’une machine.

Les chapitres suivants seront découpés comme suit :

* Description du dataset
* Analyse statistique
* Mise en œuvre du pipeline de transformation
* Structure du réseau de neurones
* Analyse des résultats obtenus
* Explicabilité

1. Description du dataset :

Le dataset est situé à l’adresse suivante : <https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-classification>

ou

https://archive.ics.uci.edu/dataset/601/ai4i+2020+predictive+maintenance+dataset

Il s'agit d'un dataset binaire de classification de défaillance, ce qui signifie qu'il peut être utilisé pour prédire si une machine est en panne ou non et de connaitre le type de défaut.

Les données sont de ce dataset sont des informations a la fois quantitatives (mesures) et qualitatives (défaut, type de produit, type de defaut)

Ce dataset possède deux cibles : ‘Target’ et ‘Failure Type’

‘Target’ : Valeur booléenne indiquant la presence d’un defaut

‘Failure Type’ : Valeur indiquant le type de defaut. Ces valeurs peuvent être :

* ‘No Failure’ : pas de defaut
* ‘Heat dissipation failure’ : Ce défaut est causé par la différence entre les variables ‘Air temperature’ et ‘Process Temperature’, et la vitesse de rotation ‘Rotational Speed’
* ‘Overstrain Failure’ : Ce defaut de surcharge est lié au produit du couple ‘Torque’ et la durée d’utilisation de l’outils ‘Tools Wear’ en fonction du type de produit ‘Type’
* ‘Power Failure’ : Ce defaut apparait lors de depassement de la puissance consommée par le process.

Nous pouvons remarquer que les défauts sont liées soit :

* Directement à des valeurs du dataset
* Indirectement à des valeurs induites du dataset : par exemple l’écart de température ou la puissance du process

Des données complémentaires seront ajoutés au dataset.

La colonne ‘ProductID’ et ‘UDI’ ne représente aucune information utile. Ces données seront donc supprimée.

1. Analyse statisque du dataset

Rentrons enfin dans le vif du sujet :

Ce jeu de donnée contient-il des valeurs manquantes ?

df.isna().sum()

* Pas de valeurs manquantes

Les cibles sont-elles équilibrées ?

sns.countplot(x='Target', data=df)

Une image contenant texte, capture d’écran, Rectangle, affichage

Description générée automatiquement

* Classes fortement déséquilibrés

Les types de défauts sont il équilibrés ?

sns.countplot(x='Failure Type', data=df[df['Target']==1])

Une image contenant texte, capture d’écran, diagramme, Rectangle

Description générée automatiquement

Les données sont elles dupliquées ?

df.duplicated().sum()

* Pas de doublons

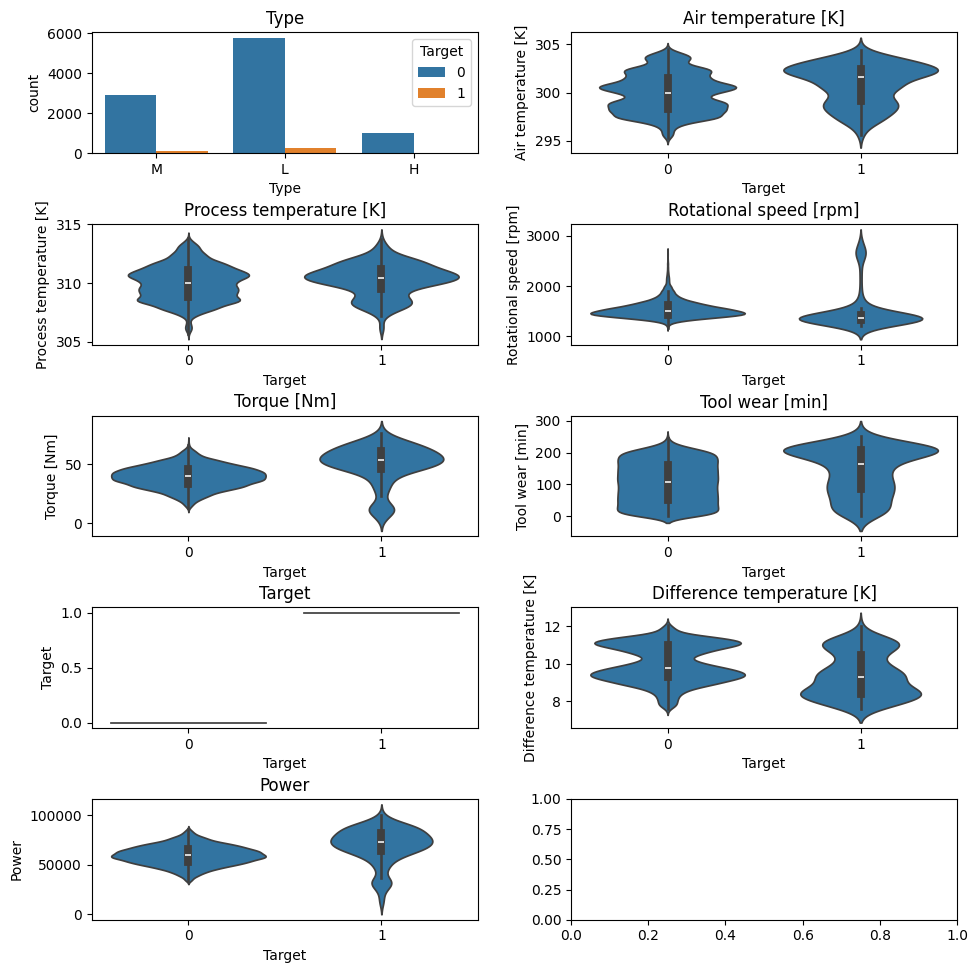
**Premières remarques :**

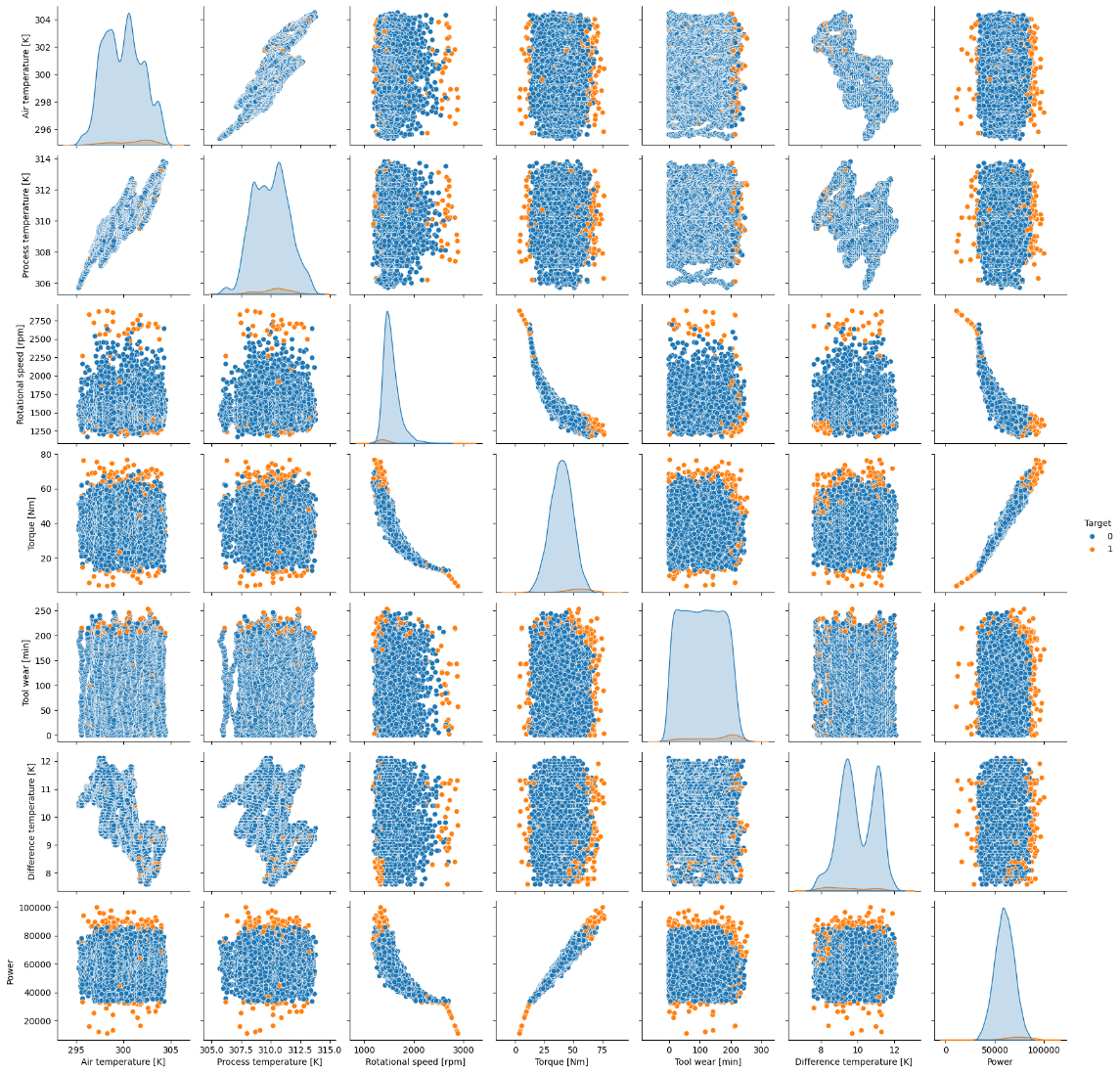
On identifie plusieurs points sur cette première analyse :

* Pas de valeurs manquantes dans les données.
* Pas de doublons
* Les cibles sont déséquilibrés

Analyse statistique :

Ci-dessous on trouvera quelques figures afin d’illustrer cette analyse





Matrice de corrélation

Une image contenant texte, capture d’écran, Parallèle

Description générée automatiquement

Quelques remarques :

* Il n’y a pas de valeurs aberrantes dans les données
* Le défaut apparait sur des valeurs périphériques des variables quantitatives
* Certaines variables ont une distribution normale
* La matrice de corrélation montre des liens entre certaines variables explicatives mais pas de lien avec la cible
* Dans la colonne ‘'Failure Type', on remarque une valeur ‘Random Failure’. Cette valeur n’est représenté que 18 fois sur l’ensemble du dataset. Par son nommage et sa représentation dans le dataset, les lignes concernées seront supprimées
* Des incohérences ont été identifier entre la variable ‘Target’ et ‘Failure type’ . Il a été vu des valeurs dans la variable ‘failure type’ alors que la variable ‘Target’ vaut 0. Ces lignes seront supprimés

1. Transformation des données

Cette étape est cruciale dans le développement de modèles de Deep Learning performants. Il permet d'améliorer la qualité des données et de les rendre plus exploitables par les réseaux de neurones.

Voici les principales étapes du prétraitement des données pour le Deep Learning:

* Nettoyage des données:

Cette étape consiste à la suppression des valeurs aberrantes, au traitement des valeurs manquantes et les corrections des erreurs

Dans ce dataset, nous ne sommes pas concernés par le traitement des valeurs aberrantes ou manquantes.

La correction des erreurs est aussi lié par des incohérences dans des données comme cela a été vu dans la phase d’analyse.

Le jeu de donnée possède ici deux cibles. La premiere cible nous indique si il y a un defaut (True/False) et la seconde indique le type de defaut par une variable catégorielle. Etant donné que nous avons nettoyer les incohérences entre la présence de défaut et le type défaut, nous pouvons utiliser uniquement la variable ‘failure type’

* Normalisation des variables numeriques:

En vue d’une utilisation de ces données pour un réseau de neurones, les données numériques doivent être standardisées (c-a-d moyenne de 0 et variance de 1). Cela permet d’améliorer la convergence du réseau de neurones.

* Encodage des données catégorielles:

Pour ce dataset, il va être utiliser l’encodage One-Hot pour les variables ‘failure type’ et ‘Type’. Cette methode consiste à créer une nouvelle colonne pour chaque categorie.

* Augmentation des données:

Comme il a été vu lors de la phase de l’analyse, le jeu de donnée est fortement déséquilibré. Lors de cette phase de preprocessing, il est possible d’atténuer cette effet en augmentant les données minoritaires ou en diminuant les données majoritaire.

* Séparation des données:

Pour la phase d’entrainement des modèles, il est nécessaire de diviser le jeu de donnée comme suit :

* + Training : ces données sont utilisées pour effectuer l’entrainement du modèle
  + Test : ces données sont utilisées après chaque lot d’apprentissage pour évaluer les performances du modèle.
  + Validation : ces données sont utilisées à la fin du processus d’entrainement et n’est pas mise en œuvre pendant l’entrainement. Cela permet de comparer les performances des modèles selon un jeu de donnée qui n’a pas été vu pendant l’entrainement.

En pratique, il est conseillé de diviser le jeu de données en utilisant les proportions suivantes:

* Ensemble d'entraînement: 60-80%
* Ensemble de validation: 10-20%
* Ensemble de test: 10-20%

1. Recherche des hyperparamètres du réseau

Cette recherche s’effectue en plusieurs étapes :

* Architecture du réseau:
  + - Nombre de couches: Le nombre de couches cachées dans le réseau.
    - Nombre de neurones par couche: Le nombre de neurones dans chaque couche cachée.
    - Fonction d'activation: La fonction d'activation utilisée dans chaque couche (par exemple, ReLU, Sigmoid).
  + Apprentissage:
* Taux d'apprentissage: Le taux d'apprentissage contrôle la vitesse à laquelle le réseau apprend. Un taux d'apprentissage trop élevé peut empêcher le réseau de converger, tandis qu'un taux d'apprentissage trop bas peut ralentir l'apprentissage.
* Nombre d'époques: Le nombre d'époques est le nombre de fois que l'ensemble du jeu de données est utilisé pour entraîner le réseau.
* Fonction de perte: La fonction de perte mesure l'erreur du réseau pendant l'entraînement.
* Optimiseur: L'optimiseur est l'algorithme utilisé pour mettre à jour les poids du réseau pendant l'entraînement.
  + Régularisation:
* L1/L2 regularization: La régularisation L1/L2 permet de réduire le surapprentissage en pénalisant les poids du réseau.
* Dropout: Le dropout est une technique de régularisation qui consiste à désactiver aléatoirement des neurones pendant l'entraînement.
* Batch Normalisation : Cette fonctionnalité normalise les couches intermédiaires du réseau, ce qui permet de réduire la variation des gradients et d’améliorer la stabilité du réseau.

Dans la structure du réseau, le nombre de neurones sur la couche d’entrée est defini par le nombre de variables du jeu de donnée et le nombre de neurones de la couche de sortie est définie par le nombre de modalité de la variable cible. La couche de sortie aura pour fonction d’activation ‘softmax’ afin de prédire la probabilité des chaque modalité.

Concernant la fonction de perte , il sera utilisé ‘Categorical Cross Entropy’ car nous sommes dans un problème multi-classe.

Comme évoqué dans un paragraphe précèdent, le jeu de donné est déséquilibré. Afin d’améliorer les performances du modèle, Il est possible de traiter le problème de plusieurs manières :

* Augmentation de donnée (over sampling): il est possible de rééquilibrer les classes minoritaires. Il est alors possible d’utiliser SMOTE pour cela. Il sera alors générer des données artificielles.
* Diminution des données (under sampling): Cette technique consiste à supprimer les données de la classe majoritaire.
* Pondérer l’importance des différentes classes : le paramètre ‘class\_weight’ de la méthode ‘fit’ offre cette possibilité.

Dans ce projet, il a été mise en œuvre la troisième possibilité.

Afin de suivre les performances du modèle, il a été choisi d’utiliser les métriques suivantes :

* **Categorical\_accuracy** : L'accuracy catégorielle calcule le pourcentage de prédictions correctes parmi toutes les prédictions effectuées par le modèle.
* **AUC**: L'AUC représente la aire sous la courbe ROC (Receiver Operating Characteristic). La courbe ROC représente le taux de vrais positifs (TPR) en fonction du taux de faux positifs (FPR) à différents seuils de prédiction.
* F1-Score : Il s'agit d'une mesure combinée qui prend en compte à la fois la précision et le rappel.
* Kappa : Le Kappa ne se contente pas de mesurer l'accord brut entre les prédictions et les labels, mais corrige cet accord pour tenir compte de l'accord attendu par le hasard.
* Precision : La précision calcule la proportion de prédictions positives correctes parmi toutes les prédictions positives effectuées par le modèle.
* Recall : Le rappel calcule la proportion d'exemples positifs réels correctement identifiés par le modèle parmi tous les exemples positifs réels.

Pour trouver la meilleure structure du model (nombre de couches), vous trouverez ci-après les résultats :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

On trouvera ci-apres la matrice de confusion :

Une image contenant texte, capture d’écran, diagramme, nombre

Description générée automatiquement

1. Explicabilité :

Les réseaux de neurones sont des modèles de type boites noires c’est-à-dire que les paramètres, la structure de celui-ci ne permettent pas de donner une explication de la décision proposée.

Un autre terme est aussi associé à l’explicabilité : l’interprétabilité.

L’explicabilité se veut à montrer le détail d’une décision (locale). L’interprétabilité se veut à aider connaitre les variables qui vont influencer le plus les décisions (globale).

L’explicabilité est de plus en plus nécessaire pour plusieurs raisons :

* **Métiers** : pour une meilleure crédibilité et appréhension par les personnes du métiers. Selon l’enjeu de l’application (médical, bancaire par exemple), il n’est pas concevable de ne pas donner la motivation de la décision proposé par le modèle.
* **Scientifique** : Le DataScientist ou le machine Learning ingénieur pourra mieux comprendre et améliorer le modèle.
* **Règlementaire** : Dans certain domaine d’application, un modèle doit être explicable afin de se conformer à la règlementation (RGPD…)

Les deux librairies les plus connues sont LIME(Local Interpretable Model-agnostic Explainable) et SHAP(Shapley value) .