Projet d'évaluation de performance sur la fiabilité des trains

Aurélie DEMURE, Nicolas FERNANDEZ

Mai 2024

Figure 1: Illustration d'un train de filtrage d'eaux industriel

1 Introduction

L'objectif de ce projet d'évaluation de performance est d'étudier des trains de filtrage d'eau industriel. Il s'agit ici de comprendre les jeux de données mis à disposition, les traiter et les analyser pour estimer leur indicateur de santé, leu fiabilité et leur durée de vie résiduelle.

Pour se faire, nous allons procéder de la façon suivante : tout d'abord, nous allons appliquer des pré-traitements aux données. Ensuite, il s'agira d'étudier plusieurs modèles choisis tels que la régression linéaire, la "random forest" et enfin un réseau de neurones. Nous pourrons ensuite conclure sur l'analyse des données.

Sommaire

2 Pré-traitements

2.1 Analyse des données

Concernant le pré-traitement des données, nous avons des informations sur quatorze trains. Les données s'étendent sur cinq ans, de 2015 à 2020. Nous pouvons commencer par analyser les différentes colonnes mises à disposition :

1. Date : la date à laquelle les informations sur les trains ont été prises.
2. Day : compte les jours depuis le début des tests en commençant au jour 1 et en finissant au 1637.
3. weeks : fais le même décompte que Day mais en semaines en commençant à 0,14.
4. Train Status Code : donne un code qui renseigne sur le statut du train concerné chaque jour.
5. Normalized DP (bars) : renseigne sur la variation de pression en bars dans le train. Cet indicateur est identifié comme l'indicateur de performance (paramètre en sortie) dans le sujet.
6. Feed volume (m3) : correspond au volume donner pour alimenter le train.
7. Brine volume (m3) : est le volume de saumure, ou d'eau salée, dans le train.
8. Product volume : le volume de produit dans le train.
9. Recovery : est une donnée en pourcentage (%) correspondant sûrement à la capacité de rétablissement du train au fil du temps.
10. K
11. Train Status : peut prendre différentes valeurs : Operation (opérationnel), Standby (en attente), Shutdown (arrêt, coupure), Master Fault (défaut maître) ou encore (De)Pressurization (dépressurisation)

2.2 Nettoyage des données

Pour effectuer le nettoyage des données, nous commençons par lire les feuilles sur les trains et retirer les colonnes dont le nom est inconnu pour ne garder que celles contenant des informations qui nous semblent utiles. Ensuite, nous avons retirer les lignes dont les pressions ne sont pas renseignées, i.e. de la forme NaN. En effet, comme il s'agit du paramètre de sortir, nous avons besoin de valeurs. Nous pouvions soit retirer les lignes correspondantes, soit remplacer NaN par la valeur moyenne de la colonne pour ce train. Nous avons fait le choix de supprimer les lignes, ayant déjà énormément de données et ne voulant pas fausser les résultats sur l'évolution journalière.

```
def traitement_donnees():
    df = pd.read_excel('Dataset-D.xlsx', sheet_name="Train 2")
    df = df.loc[:,~df.columns.str.contains('^Unnamed')]
    df = df.drop(columns=['Day','K','Date','Train Status','Train Status Code','Feed
                                            Volume (m3)','Brine Volume (m3)','
                                            Product Volume (m3)','Recovery (%)'])
    df = df.dropna(subset=['Normalized DP (bars)'])
    return df[:][:300]
```

3 Régression linéaire

Nous avons commencé par réaliser une régression linéaire du paramètre de santé *Normalized DP* sur le paramètre de *weeks* Dans chacun des cas, 70% des données ont été utilisées pour l'entraînement du modèle et les 30% restantes pour les tests. Deux situations se dessinent :

3.1 Sur un train complet

Nous avons simplement réalisé une régression linéaire sur l'entièreté des données dans un premier temps.

```
X = data.loc[:,data.columns != 'Normalized DP (bars)']
Y = data.loc[:,'Normalized DP (bars)']

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = LinearRegression()
model.fit(X_train,Y_train)

Y_pred = model.predict(X_test)
```

X correspond aux données dont on souhaite réaliser la régression et Y corresponds aux autres paramètres (ici weeks)

Les données sont découpées comme expliqué précédemment entre données d'entraînement et de test puis normalisées.

Pour cette régression linéaire, on obtient en moyenne les métriques de performances suivantes:

- R2 = 0.33073
- RMSE = 0.39579

On remarque que nos métriques de performances sont plutôt faibles. On peut donc chercher à tracer notre régression:

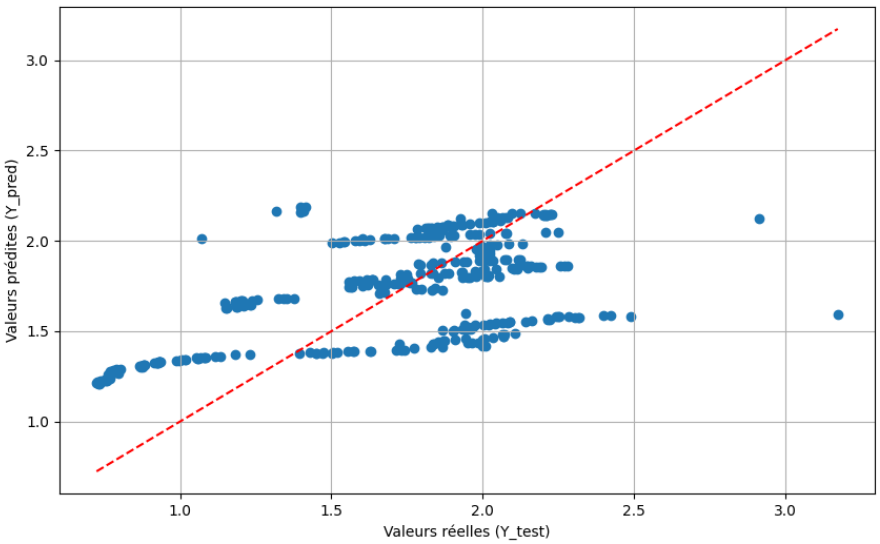

Figure 2: Régression linéaire

Ces métriques et ce graphique suggère que nos données ne sont pas adaptées pour un modèle de régression linéaire. Cependant, il arrive parfois qu'un ensemble de données puisse accepter un modèle de régression linéaire si l'on ne considère qu'une partie de ce jeu de données. Étudions cette situation.

3.2 Sur un échantillon

On peut découper notre échantillon de données et voir si le modèle de régression linéaire peut quand même être intéressant pour faire des estimations sur une partie de ces données. Ainsi, si l'on prend les 300 premières données de nos trains, on se retrouve avec cette régression:

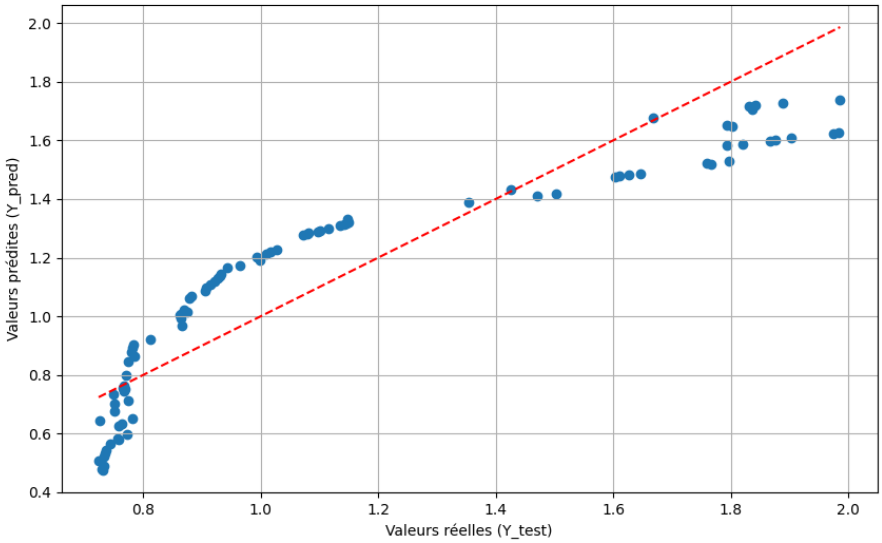

Figure 3: Régression linéaire sur une partie des données

Cette régression semble déjà bien plus convaincante, et ce constat est renforcé par l'étude des métriques de performance qui ont pour moyennes:

- R2 = 0.82445
- RMSE = 0.17617

4 Random forest

Après avoir étudié des régressions linéaires, nous avons essayé un nouveau modèle d'apprentissage : la forêt aléatoire. Nous commençons par importer le dataframe depuis le traitement de données. Puis nous divisons les données comme préciser dans l'énoncé selon deux catégories : les données d'entraînement à 70% et les données de test à 30%. Nous utilisons alors la fonction

```
model = RandomForestRegressor(n_estimators=100, random_state=42)
```

au lieu de

```
model = LinearRegression()
```

utilisé dans la régression linéaire.

Il s'agit ensuite de prédire les valeurs de test et de l'évaluer avec son score R2 et son indice RMSE.

Voici la figure obtenue pour le Train 2 :

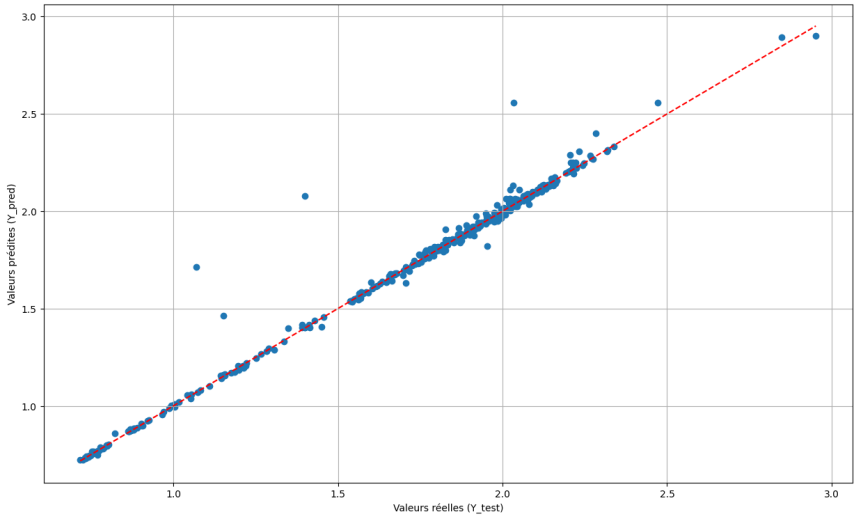

Figure 4: Modèle d'apprentissage à base d'une forêt aléatoire

Nous pouvons donc observer que la régression obtient de bien meilleures valeurs que pour la régression linéaire et suit mieux le modèle.

L'évaluation va également dans ce sens. En effet, pour cette forêt aléatoire, on obtient en moyenne les métriques de performances suivantes:

- R2 = 0.98
- RMSE = 0.062

Cette approche est donc préférable pour modéliser le comportement du système.

5 Réseau de neurones

Le dernier modèle que l'on a choisi est un réseau de neurones. Les features choisies (donc les variables indépendantes) sont représentées par X et la target (variable dépendante) qui est la sortie que le modèle essaie de prédire est représentée par Y:

```
X = data.loc[:,data.columns != 'Normalized DP (bars)']
#'Feed Volume (m3)','Brine Volume (m3)','Product Volume (m3)','Recovery (%)'
Y = data.loc[:,'Normalized DP (bars)']
```

Après avoir normalisé les données, on entraîne notre modèle sur plusieurs couches:

```
model = keras.Sequential(
    [
        keras.layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
        keras.layers.Dense(128, activation='relu'),
        keras.layers.Dense(1)
    ]
)
```

Nous avons choisi d'utiliser un modèle séquentiel avec des couches "Dense" dans lesquelles chaque neurone reçoit une connexion de tous les neurones de la couche précédente. On a choisi d'utiliser 128 neurones dans nos couches et la fonction d'activation *'relu'* qui permet d'introduire de la non-linéarité dans le modèle. On précise l'input_shape dans la première couche afin de spécifier au modèle la dimension des entrées. On a enfin une couche de sortie classique avec 1 neurone.

Avec ce modèle, on obtient en moyenne les métriques de performances suivantes:

- R2: 0.823
- RMSE: 0.201

On obtient le graphique de dispersion comparant les valeurs réelles aux valeurs prédites suivant:

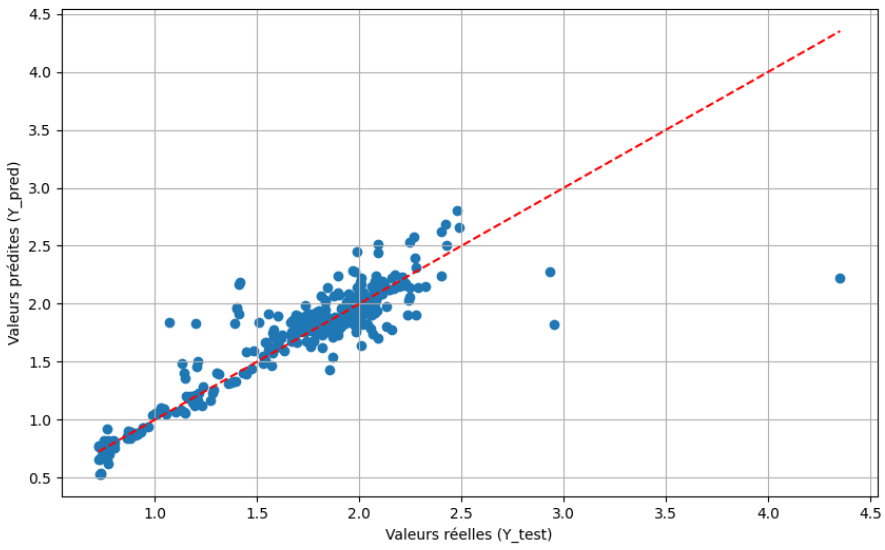

Figure 5: Modèle d'apprentissage à base d'un réseau de neurones

Enfin, nous pouvons étudier l'évolution de l'erreur au fur et à mesure de l'entraînement du modèle. La ligne 'loss' représente l'évolution de l'erreur du modèle tandis que la ligne 'val_loss' représente l'évolution de l'erreur du modèle sur les données qu'il n'a pas vues pendant l'entraînement.

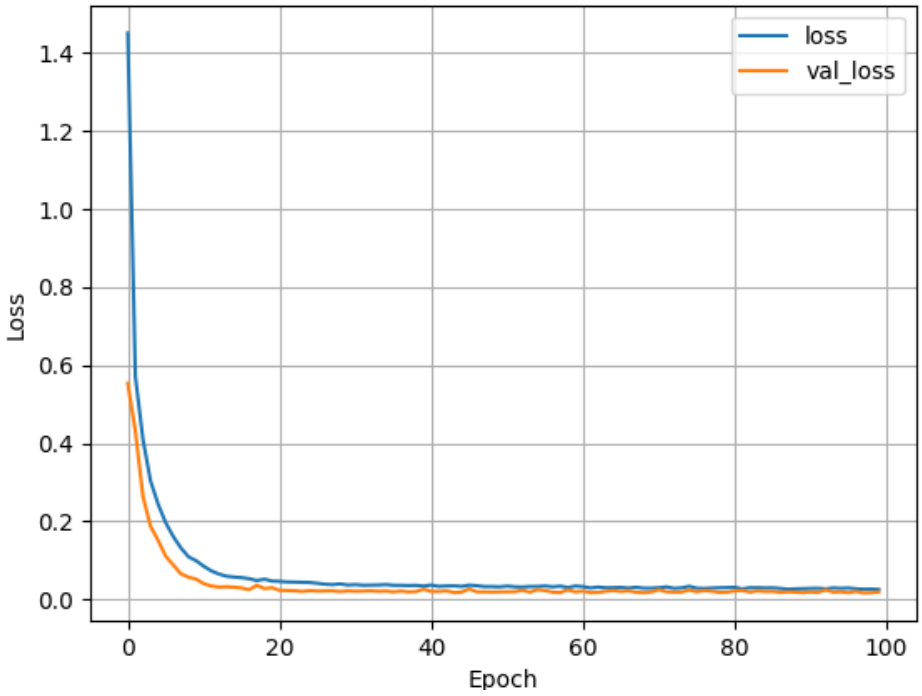

Figure 6: Evolution de l'erreur au fur et à mesure de l'entraînement du modèle

6 Durée de vie

Enfin nous avons cherché à exprimer la durée de vie résiduelle de nos trains de filtrage. Nous avons choisi la métrique de Kaplan-Meier pour estimer la fiabilité de nos trains, puisque cette dernière se prête bien à ce genre de système. On considère qu'il y a échec du système (donc défaillance) lorsque le train n'est pas en *"Operation"*, autrement dit lorsque qu'il est dans l'un des autres états (Shutdown, Master Fault, etc). La fonction de fiabilité/survie prend donc cette forme:

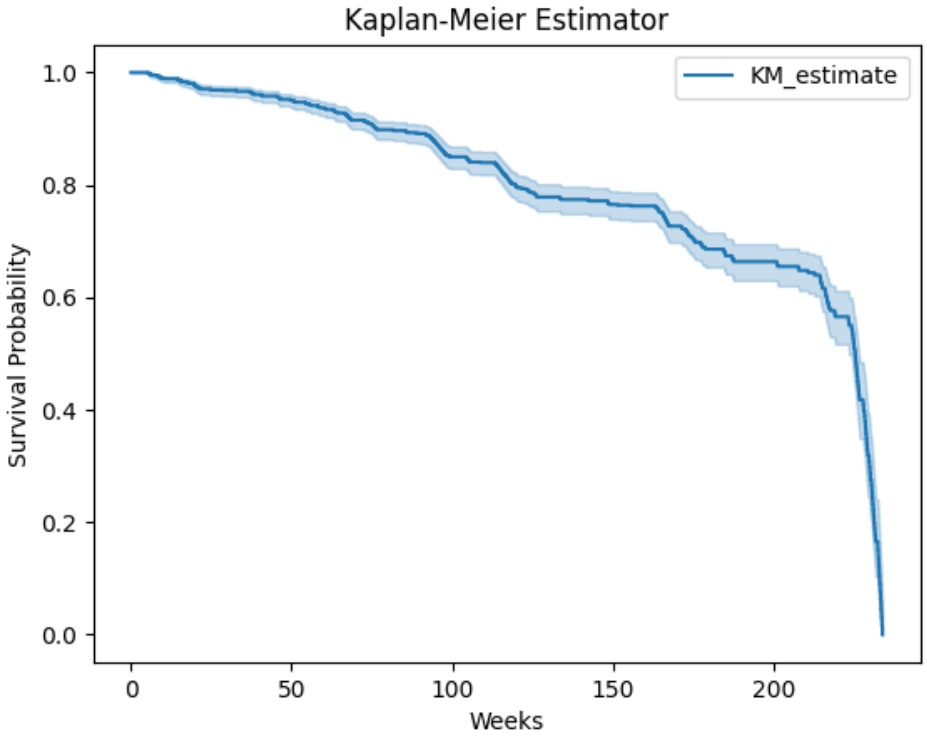

Figure 7: Durée de vie des trains de filtrage

On peut ainsi déterminer la MRL (Durée de vie résiduelle moyenne) pour notre système. On sait que :

$$MRL(t) = \frac{1}{R(T)} \int_t^{\infty} R(x)dx$$

que l'on a calculé de cette manière:

```
def MRL(t):
r_t = kmf.predict(t)
somme = []
for i in range (len(event)-t):
    somme.append(kmf.predict(t+i))
mrl_t = np.sum(somme)/r_t
return mrl_t
```

On a alors les valeurs suivantes:

- MRL(0) = 188.34
- MRL(1) = 110.70
- MRL(2) = 27.77

- MRL(3) = 0

Ces résultats sont cohérents avec nos observations.

7 Conclusion

Nous avons identifié comme indicateur de santé le paramètre 'Normalized DP'. C'est de cet indicateur que nous avons cherché à établir un modèle en relation avec les autres paramètres des données. Nous avons proposé plusieurs approches pour modéliser le comportement du système:

1. Une régression linéaire (d'abord sur l'intégralité des données puis sur les données partielles).
2. Un modèle de Forêt d'arbres décisionnels (Random Forest).
3. Un modèle de réseau de neurones.

De ces 3 modèles, c'est la forêt d'arbres décisionnels qui semble être le plus pertinent pour modéliser notre système de train de filtrage industriel, avec des métriques de performances très satisfaisantes (R2 = 0.98 donc très proche de 1 et RMSE = 0.062 donc très proche de 0).

Enfin, nous avons cherché à modéliser la dégradation du système en étudiant sa durée de vie résiduelle. Après avoir choisi un modèle pertinent, nous avons pu calculer la MRL de notre système pour obtenir des valeurs intéressantes et estimer sa durée de vie.