ENSAM MEKNES

Projet Industrie 4.0 et usine future : Prédiction de TRS

June 22, 2022

Réaliser par:

- AMRANI ALAOUI MOHAMMED
- FAYTOUT ACHRAF

Encadrer par:

- Mme. EL MAZGUALDI CHOUMICHA
- M. KHDOUDI ABDELMOULA

Contents

1 Introduction

Le taux de rendement synthétique est un indicateur très utilisé pour évaluer de manière claire et normée la performance d'un équipement industriel. Il constitue de trois composantes :
Disponibilité de la machine ou de l'équipement, la performance de celle-ci en régime normal et finalement la qualité qu'elle est capable de fournir. Donc le TRS est un indicateur clé, dont l'analyse fournit à la fois la mesure de la performance et les plans d'actions pour l'amélioration (indicateur de pilotage). Est un outil d'investigation efficace.
La mise en œuvre d'une démarche ciblée d'amélioration le TRS permet d'améliorer la productivité et de dégager des capacités supplémentaires.

Notre but dans ce mini projet est de prédire le Taux de Rendement Synthétique (TRS) à partir d'un historique de données brutes, afin de mettre en place un plan d'action convenable à chaque situation.
Pour se faire nous allons suivre les étapes suivantes :

- Conception d'un data-set convenable pour prédire le *TRS*.
- Application du *Data pre-processing* pour le prétraitement des données.
- Application de *Features Selection*.
- Application de *Data visualisation*.
- Construction du modèles Machine Learning, Deep Learning et Benchmarking des modèles.

2 Préparation des données

A partir des tableaux Excel, nous allons fusionner dans un premier temps les tableaux d'Historiques pour avoir un seul Tableau contenant l'historique des commandes produites, Puis nous construisons une colonne 'Id' qui rassemble la machine, le shift et la date dans la dataFrame 'Hist' et aussi pour 'OEE'.

```
oee['id'] = oee['Machine'].map(str) + '/' + oee['Shift Day'].map(str) + '/' + oee['Shift'].map(str)
```

Figure 1: Création de la colonne 'Id' pour la table 'OEE'.

Dans le 2 ème tableau de Historique, on a besoin de faire une petite préparation de la colonne '' Actual Start Date '' pour la rendre sous une forme de date :

```
for i in range(0, hist.shape[0]):
    a = hist.loc[i, "Actual Start Date"]
    m = ""
    for j in a:
        if(j == "."):
            m += "-"
        elif len(m) != len(a) -1 :
            m +=
    hist.loc[i, "Actual Start Date"] = m
```

Figure 2: Traitement du data type du colonne 'Actual Start Date'.

Et après la création de colonne d'heure, il faut crée la colonne "shift" pour qu'on puisse créer l'id :

```
conditions = [
    (hist['hour'] >= 6) & (hist['hour'] < 14),
    (hist['hour'] >= 14) & (hist['hour'] < 22),
    ((hist['hour'] >= 22) & (hist['hour'] <= 23) | (hist['hour'] >= 0) & (hist['hour'] < 6))
]

# create a list of the values we want to assign for each condition
values = ['S1', 'S2', 'S3']

# create a new column and use np.select to assign values to it using our lists as arguments
hist['Shift'] = np.select(conditions, valus)
```

Figure 3: Conception du colonne 'Shift'.

Par la suite, on va regrouper l'information des trois colonnes dans un seul feature appelé 'Id' :

```
hist['id'] = hist['Plan  Workstation'].map(str) + '/' + hist['date'].map(str) + '/' + hist['Shift'].map(str)
```

Figure 4: Création de l'index du tableau 'Hist'.

Après une analyse de données, on a décidé de supprimer quelques colonnes de tableau historiques [**'Pers. No.'**, **'Batch Size (Pcs/Bdl)'**, **'Plan Workstation'**, **'Actual Start Date'** ...], après pour les 4 features [**'Terminal Right'**, **'Terminal Left'**, **'Seal Left'**, **'Seal Right'**] on a remplit les valeurs manquantes par des 0 puisque par exemple dans le cas de *Terminal*, l'information qu'on peut extraire est soit qu'il y a un terminal soit non (binary value) donc par la suit, on va remplir les autres valeurs par des 1.

```
columns = ['Terminal Right', 'Terminal Left', 'Seal Left', 'Seal Right']
for column in columns :
    hist_copy[column].fillna(0, inplace=True)

for column in columns :
    for i in range(0,hist_copy.shape[0]) :
        if hist_copy.loc[i, column] != 0 :
            hist_copy.loc[i, column] = 1
```

Figure 5: Encoding les attributs 'Terminal' et 'Seal'.

Et puisque cette nouvelle étiquette **['id']** n'est pas unique, elle est répétée autant de fois que le nombre de commandes produite le même jour, le même shift et par la même machine.
Nous avons synthétisé les données relatives à toutes ces commandes sur une même ligne en utilisant une fonction d'agrégation avec la spécification de la stratégie de concaténation pour chaque attributs (sum, mean, min, max ...).

```
columns_max = ['Terminal', 'Seal']
columns_mean = ['Prod. Time  (min)','Setup  Time (min)','Wire Left Length','Wire Left
                CrossSection','Stripping Length Left','Stripping Length Right']

df1 = hist_copy.groupby('id')[columns_max].max()
df2 = hist_copy.groupby('id')[columns_mean].mean()
df3 = hist_copy.groupby('id')['Quantity  (Pcs)'].sum()
hist_copy_id = pd.merge(df3, pd.merge(df2, df1, on="id"), on="id")
```

Figure 6: Réduction de l'information par des fonctions d'agrégation.

Finalement on a concatené les deux tableaux par **['Id']**.

```
merged_hist_oee = hist_copy_id.merge(oee,how='inner',on='id')
```

Figure 7: Concatenation des tableaux 'Hist' et 'OEE'.

3 Prétraitement des données

3.1 Traitement des valeurs manquantes

Avant le Data Pre-processing nous avons supprimer toutes les colonnes que nous ne pouvons pas connaitre à l'avance, comme le nombre de pièces bonnes, ce type de variable il ne fallait pas figurer comme input du modèle que nous allons construire.
Les valeurs manquantes de la qualité, sont remplacées par 100 parce que on a supposé que le taux de qualité soit 100%, et nous avons suivi la stratégie de la moyenne pour remplir les valeurs manquantes de OEE, Performance, et Availability.

```
# Filling the missing values of OEE, Availibility and Performance by their mean value
df_hist_oee['Performance'] = df_hist_oee['Performance'].fillna(df_hist_oee['Performance'].mean())
df_hist_oee['OEE'] = df_hist_oee['OEE'].fillna(df_hist_oee['OEE'].mean())
df_hist_oee['Availability'] = df_hist_oee['Availability'].fillna(df_hist_oee['Availability'].mean())
# hypothesis : rate(Quality) = 100 % => then filling the missing values of Quality by 100
v_Quality = 100
df_hist_oee['Quality'] = df_hist_oee['Quality'].fillna(v_Quality)
```

Figure 8: Remplacement des valeurs manquantes.

3.2 Supprimer les outliers

Les valeurs aberrantes sont des points de données qui ne correspondent pas au modèle du reste de l'ensemble de données. La meilleure façon de détecter les valeurs aberrantes dans l'ensemble de données donné est de tracer la boîte à moustaches (box plot) de l'ensemble de données et le point situé à l'extérieur de la boîte dans la boîte à moustaches correspond à toutes les valeurs aberrantes de l'ensemble de données.

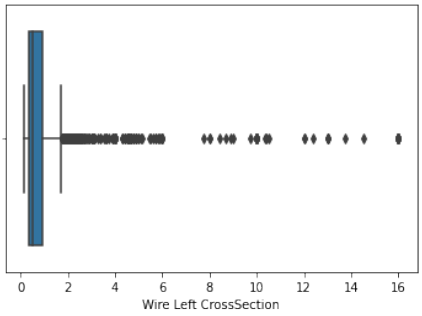

Figure 9: box plot of Wire Left CrossSection

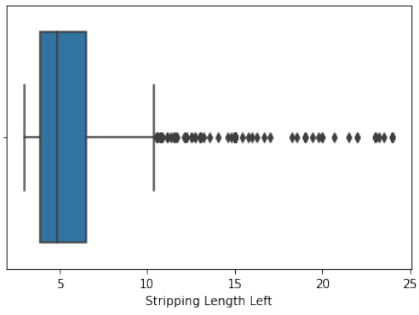

Figure 10: box plot of Stripping Length Left

On a remarqué qu'il y a beaucoup des outliers pour cela on utiliser la technique du Z-score pour éliminer ces derniers.

```
# for each column, if abs((X_f - Mean(X_f))/ Std(X_f)) < 3 then keep it.
df_hist_oee_outlier = df_hist_oee[(np.abs(stats.zscore(df_hist_oee)) < 3).all(axis=1)]
```

Figure 11: Suppression des valeurs aberrantes par la technique z-score.

3.3 Sélection des variables

La sélection de caractéristiques est le processus par lequel un sous-ensemble de caractéristiques pertinentes, ou variables, est sélectionné à partir d'un ensemble de données plus large pour construire des modèles. L'objectif principal de la sélection de caractéristiques est de choisir des caractéristiques qui représentent bien l'ensemble de données en excluant les données redondantes et non pertinentes.
On élimine en premier lieu les variables qui n'ont pas d'effet sur le modèle :

```python
# Drop elements that doesn't effect the model :
list_f = []
for i in range(0,df_hist_oee.nunique().shape[0]) :
    if df_hist_oee.nunique()[i] == 1 :
        list_f.append(df_hist_oee.nunique().index[i])

df_hist_oee = df_hist_oee.drop(list_f, axis = 1)
```

Figure 12: Élimination des attributs non pertinentes.

Par la suite, nous dessinons la matrice de corrélation pour sélectionner les variables corrélées avec un seuil près.

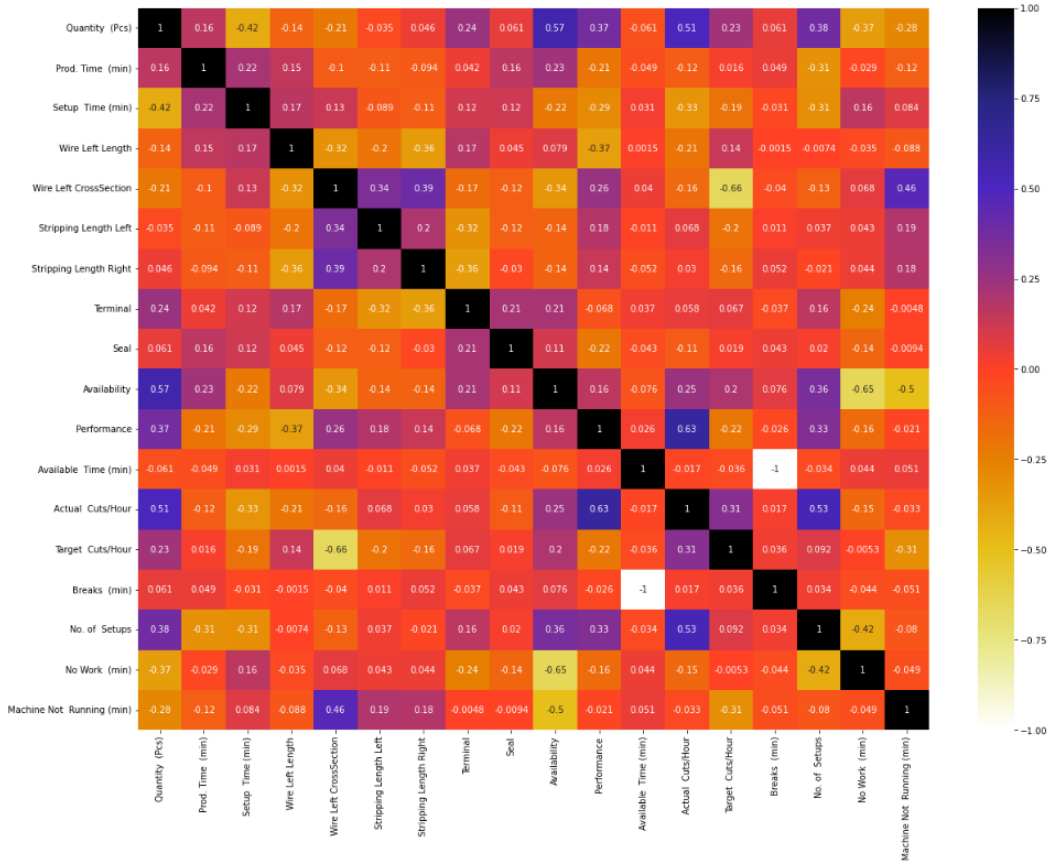

Figure 13: Heatmap de correlation

Dans cette étape, nous allons éliminer les caractéristiques qui sont fortement corrélées. Pour éliminer la dépendance entre variables. La figure suivante illustre le code de cette partie faite sur python, au début, il faut fixer un seuil à partir duquel si le coefficient de corrélation entre deux variables dépasse ce seuil

(80% en générale) on garde qu'un seul et on supprime l'autre, parce qu'elles seront fortement corrélées et représentent presque les mêmes informations pour le modèle. Dans notre cas nous avons 'Breaks (min)' et 'Available time (min)' sont fortement corrélées.

```python
# Select all features with a correlation over the threshold
def correlation(dataset, threshold):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold:
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr
```

Figure 14: Selection des dépendances entre les variables.

On Trouve que ***'Breaks'*** est corrélé avec ***'Available time'***, donc on va l'éliminer.

4 Machine Learning

Le Machine Learning est la science qui permet aux ordinateurs d'apprendre sans être explicitement programmés. Dans son application à travers les problèmes d'affaires, l'apprentissage automatique est également appelé **analyse prédictive**. Dans notre cas, on va utiliser les modèles de machines learning pour faire de l'analyse prédictive. Alors dans cette partie, on va tester différents modèles pour prendre le meilleur entre eux.

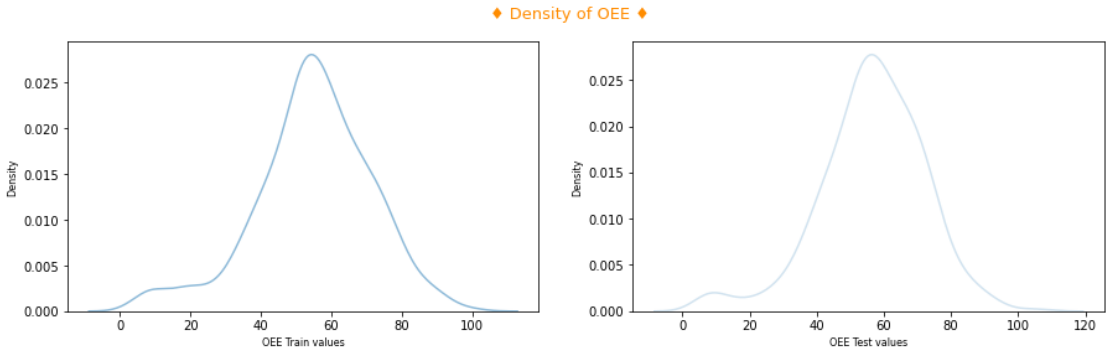

Figure 15: Densité du target 'oee' entre test et train.

4.1 Linear Regression

Nous avons commencé avec ce modèle car c'est un modèle standard et simple, et s'il donne de bons résultats, alors il n'y a pas besoin de s'inquiéter avec des choses compliquées. En effet, la régression linéaire est l'un des algorithmes d'apprentissage automatique supervisé en Python qui observe en permanence les caractéristiques et prédit un résultat, elle donne souvent des bons résultats.

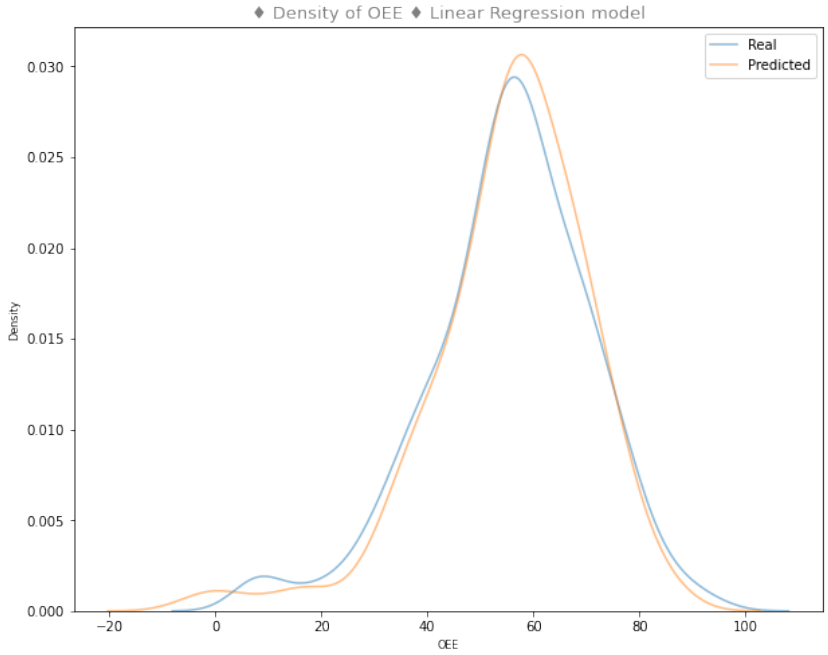

Figure 16: Densité du target 'oee' entre observed et predicted par Linear Regression.

La valeur de $mse = 7.3829$
La valeur de $r^2 = 0.9678$

4.2 Random Forest for Regression

Il fonctionne efficacement sur les grands ensembles de données. Random Forest a une grande précision que les autres algorithmes. En effet, Random Forest Regression utilise la méthode d'apprentissage d'ensemble pour la régression. Méthode d'apprentissage d'ensemble est une technique qui combine les prédictions de multiples algorithmes d'apprentissage automatique pour faire une prédiction plus précise qu'un seul modèle.

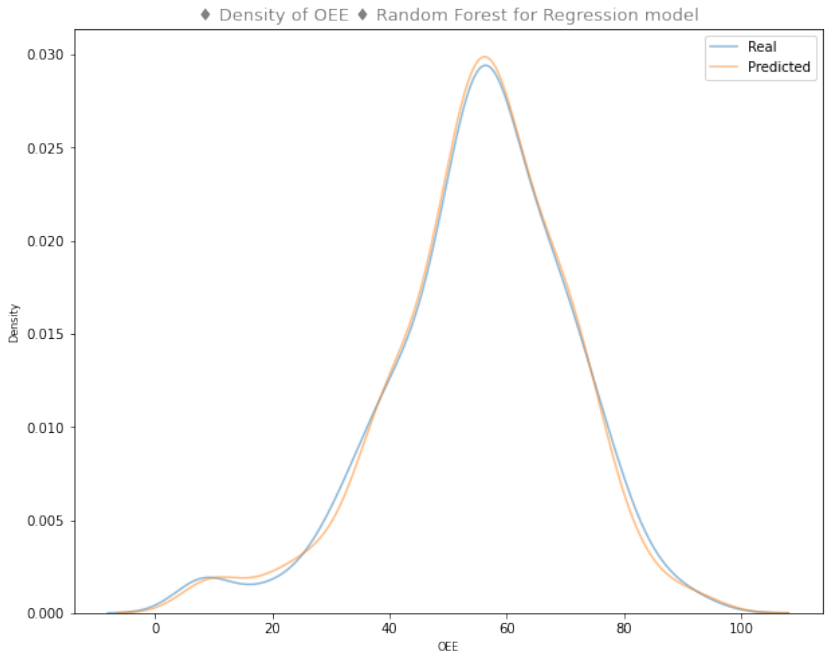

Figure 17: Densité du target 'oee' entre observed et predicted par Random Forest.

La valeur de $mse = 4.4650$
La valeur de $r^2 = 0.9808$

4.3 Gradient Boosting for Regression

le Gradient Boosting Algorithm diminue l'erreur de biais donc c'est intéressant de tester ce modèle aussi. En effet, Gradient boosting Regression calcule la différence entre la prédiction actuelle et la valeur cible correcte connue. Cette différence est appelée résiduelle. Après cela, Gradient boosting Regression entraîne un modèle faible qui mappe les caractéristiques à ce résiduel. Ce résidu prédit par un modèle faible est ajouté à l'entrée du modèle existant et ce processus pousse le modèle vers la bonne cible. Répéter cette étape encore et encore améliore la prédiction globale du modèle.

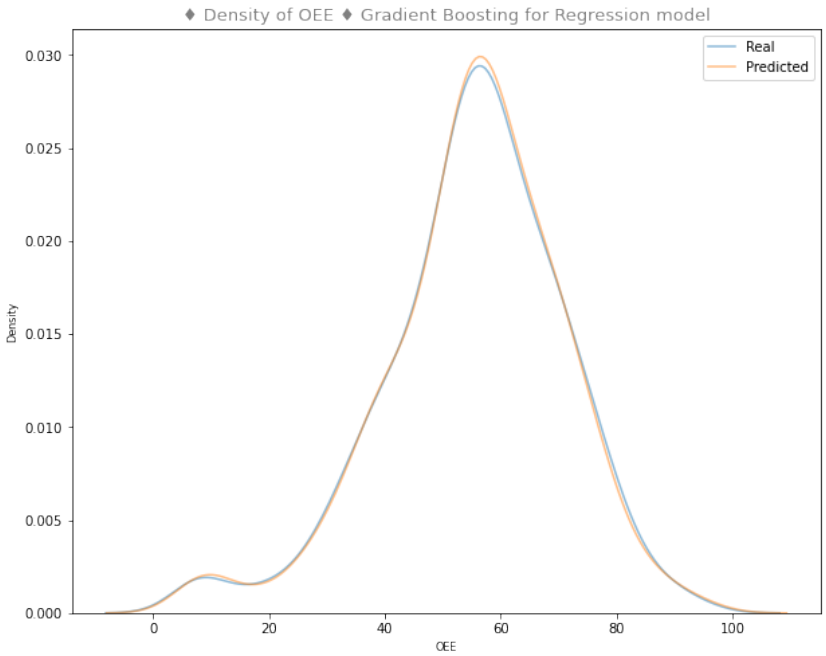

Figure 18: Densité du target 'oee' entre observed et predicted par Gradient Boosting.

La valeur de $mse = 3.2776$
La valeur de $r^2 = 0.9866$

4.4 Deep Learning Model

Le Deep Learning permet aux modèles informatiques d'apprendre en rassemblant les connaissances de l'expérience. Les concepts complexes peuvent être appris par l'approche d'apprentissage profond en raison de sa hiérarchisation. L'apprentissage profond a considérablement bénéficié de l'état de l'art dans de nombreux domaines dans le monde moderne. Le Deep Learning a une immense influence sur des domaines et ça sera très intéressant de l'utiliser dans notre cas.

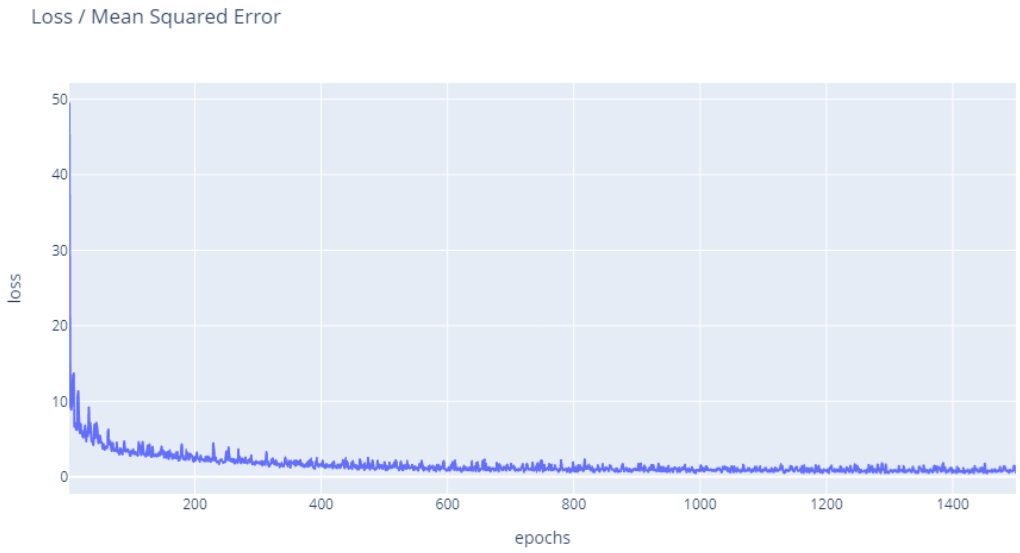

Figure 19: Tracage du Loss function par rapport l'epochs.

La valeur de $mse = 5.3150$
La valeur de $r^2 = 0.9789$

5 Conclusions

D'après les scores qu'on a calculé, on peut dire que Gradient Boosting et Deep Learning donne des bons résultats par rapport aux autres modèles. mais d'apres le r^2 on va choisir le modèle de Gradient Boosting puisque en générale r^2 est plus adapter pour la comparaisone des modèles.

Finalement, dans ce mini projet nous avons construit un modèle de prédiction du Taux de Rendement Synthétique (TRS) à partir d'un historique de données brutes, en passant par data processing et features selection. Cette prédiction va nous permettre de mettre des plans d'action efficace pour anticiper toute situation indésirable.