Projet de Fin d'études
CentraleSupélec - ArcelorMittal

Analyse d'une ligne de fabrication d'acier très haute résistance (THR) par des méthodes de Machine Learning pour la prédiction des qualités mécaniques

Guillaume Barrée - Antoine Pagneux

CentraleSupélec

ArcelorMittal

10 avril 2022

Plan

Introduction

Contexte

Contexte :

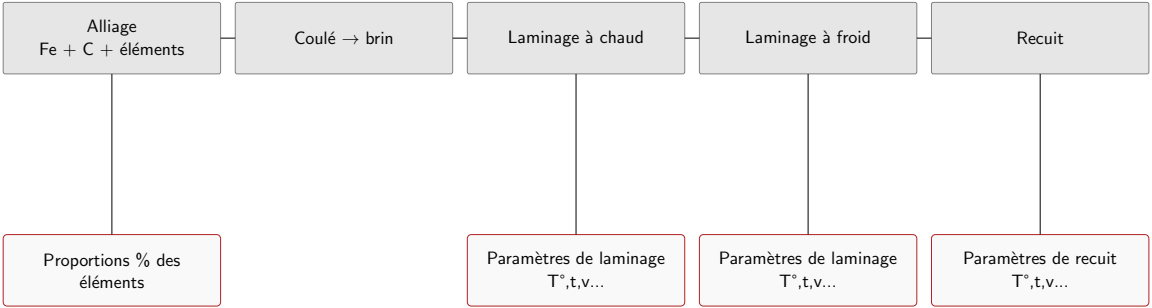
- Acier Très Haute Résistance $\rightarrow$ Propriétés mécaniques très intéressantes pour l'industrie :
  - Ductilité ($\rightarrow A_{\%}$).
  - Résistante Mécanique ($\rightarrow R_m$).
- Aciers très appréciés dans le domaine du transport routier : ($\searrow$ Épaisseur des pièces $\Rightarrow \searrow$ Masse $\Rightarrow \searrow$ Consommation de carburant $\Rightarrow \searrow$ Émissions de GES).
- Acier est un matériau recyclable.

Objectifs :

- Analyser une ligne de fabrication d'aciers THR par méthodes de Machine Learning pour la prédiction de leurs propriétés mécaniques.

Introduction

Fabrication des aciers à très haute résistance



Introduction

Enjeux

À l'heure actuelle :

- ArcelorMittal dispose de Modèles physiques pour certains aciers.
- Mais les aciers THR sont assez complexes à modéliser physiquement.
- $\Rightarrow$ Utilisation de modèles de Machine Learning.

Introduction

Cadre de l'étude

Prédire :

- La **résistance mécanique** R_m.
- L'**allongement à la rupture** $A_{\%}$.
- La **limite conventionnelle d'élasticité** $R_{e,0.2\%}$.

En fonction de :

- La **composition chimique** de l'alliage (proportion des éléments : C, Mn, Si, Cr, Mo...).
- Les **paramètres des traitements thermiques** (T° chauffe, T° refroidissement, temps de maintien...).
- Les **paramètres des traitements mécaniques** (Pression exercée par les laminoires...)

Plan

Analyse des données

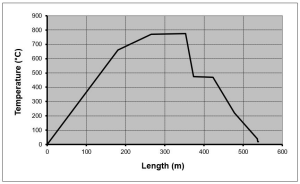
Présentation des données brutes

Données conservées pour nos modèles

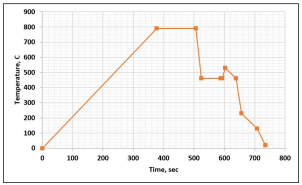
- Coilnr
- Date
- Re02 MPa
- Rm MPa
- A80%
- Direction
- Type
- Th mm
- C ppm
- Mn ppm
- Si (ppm)
- P (ppm)
- S (ppm)
- Al (ppm)
- Ti (ppm)
- Cr (ppm)
- Nb (ppm)
- B (ppm)
- Mo (ppm)
- Linespeed (m/min)
- SKP elongation (%)
- Heating (C/s)
- t_soaking_hot (s)
- Cooling (C/s)
- t_soaking_cold (s)
- soaking_hot (C)
- soaking_cold (C)

Traitement des données

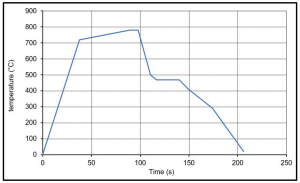
Sélection des lignes



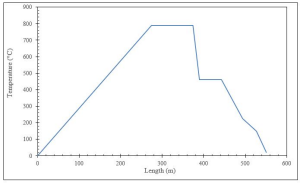
Cycle SDG3



Cycle Galma



Cycle EKO1



Cycle Sagunto

Numéro	Nom des fichiers
all	Tous les fichiers
0	Galma
1	SDG3-v2
2	SDG3.5
3	EKO1
4	SDG3
5	Sagunto

Figure – Correspondance numéro-nom de fichier

Traitement des données

Sélection des données

Sélection de certains échantillons :

Direction : T ou L

Type : I20 ou JI5

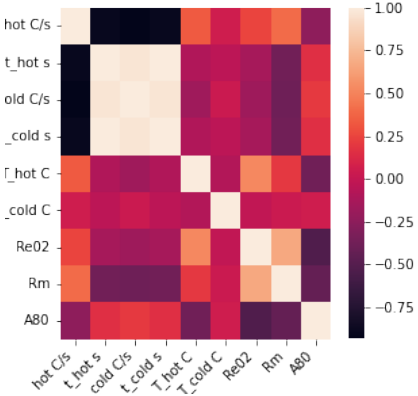
Sélection de caractéristiques :

Il est possible de supprimer certaines caractéristiques de données d'entraînement

- Si une caractéristique est absente d'un fichier
- Si une caractéristique n'apporte rien par rapport à la sortie

Analyse des données

Corrélation entrées-sorties



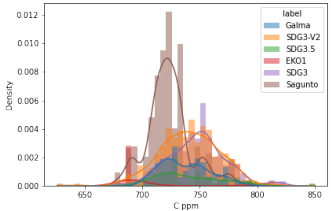
SDG3

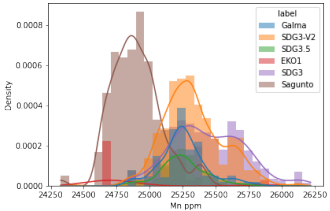
Analyse

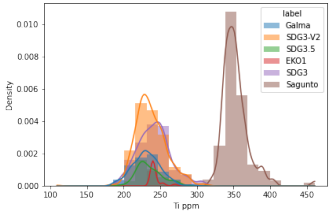
- La vitesse de chauffe, de refroidissement, le temps de maintien à chaud et le temps de maintien à froid sont très corrélés. Ceci est sûrement dû à des règles métiers.
- Il est difficile de trouver un pattern expliquant clairement une corrélation entre nos entrées et nos sorties.

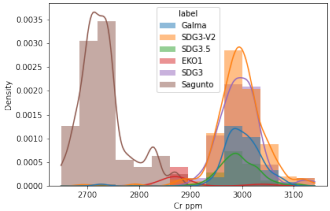
Analyse des données

Distribution des entrées









Analyse

- Différence notable entre Sagunto et les autres lignes
- Distribution quasiment Gaussienne pour les caractéristiques
- Information utile pour trouver la cause de caractéristiques mécaniques hors norme

Plan

Critères d'apprentissage

Formalisation du problème

Définitions mathématiques des espaces :

- $\mathcal{X}$: Ensemble des valeurs que peut prendre un vecteur x_i pour une observation des *features* du problème.
- $\mathcal{Y}$: Espace d'observations d'une des *targets*. (e.g. $\mathcal{Y}$ peut représenter l'ensemble des valeurs que peuvent prendre R_m, $R_{e,02\%}$ ou encore $A_{\%}$).

Définition d'une observation :

- Un couple $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.
- Avec
  - n *features* et 3 *targets* : R_m, $R_{e,02\%}$ et $A_{\%}$.
  - m observations de ces variables.

On note $X \in \mathcal{M}_{m,n}(\mathbb{R})$ la matrice des observations et $Y = \left(Y_{R_m}, Y_{R_e}, Y_{A_{\%}}\right) \in \mathcal{M}_{m,3}(\mathbb{R})$.

Définition d'un estimateur :

Une fonction $h : \mathcal{X} \to \mathcal{Y}$ qui pour un vecteur x_i cherche à approcher la valeur vraie y_i correspondante.

Critères d'apprentissage

Fonction de coût *MSE*

Définition d'une fonction de coût :

$$L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^{+} \tag{1}$$

Dans le cadre de problème de régression :

On utilise la Mean Square Error :

$$MSE = \frac{1}{m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right)^2 \tag{2}$$

Critères d'apprentissage

Métriques *RMSE* et R^2

Root Mean Square Error :

$$RMSE = \sqrt{MSE} \tag{3}$$

Coefficient de détermination (ou de Pearson) :

$$R^2 = 1 - \frac{\sum_{i=1}^{m} (y_i - h(x_i))^2}{\sum_{i=1}^{m} (y_i - \overline{y})^2} \tag{4}$$

où

$$\overline{y} = \frac{1}{m} \sum_{i=1}^{m} y_i$$

Plan

Modèles de régression :

3 catégories de modèles de régression :

- Les méthodes d'ensemble.
- Les Séparateurs à Vastes Marges[a] (non présentés ici).
- Les réseaux de neurones.

a. ou Machine à Vecteur de support (SVM)

Méthode d'ensemble

Idée principale : Arbre de décision

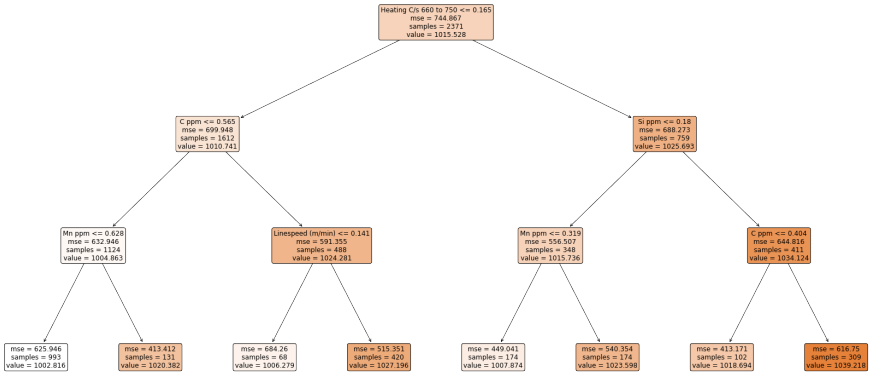


Figure – Exemple d'un arbre de décision d'une profondeur de 3

Méthode d'ensemble

Idée principale : Arbre de décision

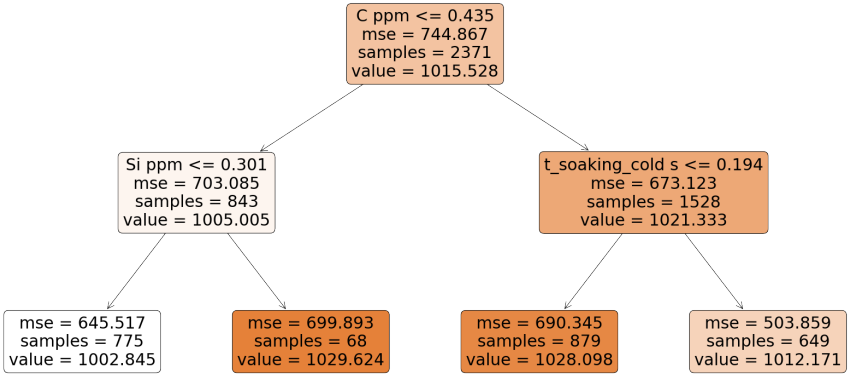


Figure – Exemple d'un arbre de décision d'une profondeur de 2

Méthodes d'ensemble

Forêts aléatoires (Random Forest)

Idées principales :

- Utilisation d'un certain nombre d'estimateurs (Arbres de décision) relativement simple vis-à-vis de la complexité du problème.
- Assemblage pour pouvoir effectuer une prédiction beaucoup plus robuste qu'un simple estimateur.
- Problèmes de régression : Résultat = Moyenne des résultats de chaque arbre.

Méthodes d'ensemble

Forêts aléatoires (Random Forest)

"Forêts" Aléatoires :

- Estimateurs sont des arbres de décision.
- Les arbres de décisions sont assemblés $\rightarrow$ on parle naturellement de *forêt*.

Forêts "Aléatoires" :

- *Tree bagging* : Pour n variables (features) et m données d'apprentissage, la construction de k arbres de décisions pour $1 \leq k \leq m$ s'effectue comme ceci :
  1. Tirage aléatoire, avec remplacement, de k échantillons de données.
  2. Entraînement de l'arbre de décision sur les k données d'apprentissage.
  3. Application de l'inférence sur chacun des arbres, puis calcul de la moyenne des sorties.
- *Feature sampling* : Il y a tirage aléatoire de l variables parmi les n variables du problème.

Méthodes d'ensemble

Forêts aléatoires (Random Forest)

Principaux hyper-paramètres :

- *bootstrap* : Booléen permet de choisir si l'ensemble des données d'apprentissage est utilisé pour construire chaque arbre de décision.
- *criterion* : Critère à prendre en compte pour mesurer la qualité d'une valeur prédite par rapport à la valeur vraie. Par défaut, l'erreur quadratique moyenne est utilisée.
- *max_depth* : Entier déterminant la profondeur maximale de chaque arbre de décision. Par défaut, toutes les divisions possibles sont effectuées.
- *max_features* : Entier pour le nombre de variables (*features*) à prendre en compte lors de la recherche des meilleurs divisions.
- *min_samples_split* : Entier pour le nombre minimum d'échantillons de données requis pour diviser un nœud interne.
- *n_estimators* : Entier pour le nombre d'arbres de décision (i.e. le nombre d'estimateurs) dans la forêt.
- *n_jobs* : Entier pour le nombre de travaux à effectuer en parallèle.

Méthodes d'ensemble

Forêts aléatoires (Random Forest)

Réglage des hyper-paramètres :

- *GridSearchCV* :
  - Méthode de recherche exhaustive sur des valeurs de paramètres spécifiées pour les hyper-paramètres.
  - Trouver la combinaison qui apprend le mieux.
  - Cette méthode utilise la validation croisée (*cross-validation*).
- *BayesSearchCV* :
  - Les paramètres de l'estimateur utilisés pour appliquer ces méthodes sont optimisés par une recherche à validation croisée sur les paramètres.
  - Contrairement à GridSearchCV, toutes les valeurs de paramètres ne sont pas testées, mais un nombre fixe de paramètres est échantillonné à partir des distributions spécifiées.

Méthodes d'ensemble

Forêts aléatoires (Random Forest) - Implémentation dans scikit-learn

```
from sklearn.ensemble import RandomForestRegressor

## Instance of RFR and definie its hyperparameters
rfr = RandomForestRegressor(bootstrap=False,
                            max_depth=21
                            max_features=10,
                            min_samples_split=6,
                            n_estimators=100,
                            n_jobs=-1)

## Train the model on train-set
rfr.fit(X_train, Y_train_rm)

## Apply inference on test-set
Y_pred_rfr = rfr.predict(X_test)
```

Figure – Régression par Random Forest

Méthodes d'ensemble

Extra Trees

Différences avec le Random Forest :

- *Extra Trees* :
  - Utilise toujours l'ensemble des données d'entraînement, et non pas un échantillon comme *Random forest*.
  - $\Rightarrow$ Pas d'hyper-parametres *bootstrap*.
- La seconde différence provient de la sélection du point de coupe afin de diviser les noeuds des arbres de décision.
  - Dans *Extra Trees* le point de coupe de la division est choisi de manière de manière aléatoire.
  - *Extra Trees* permet ainsi d'ajouter de l'aléatoire mais conserve tout de même de l'optimisation.

Les hyper-paramètres sont identiques.

Méthodes d'ensemble

Extra Trees - Implémentation dans scikit-learn

```python
from sklearn.ensemble import ExtraTreesRegressor

## Instance of ETR and definie its hyperparameters
etr = ExtraTreesRegressor(  max_depth=23,
                            max_features=11,
                            min_samples_split=6,
                            n_estimators=105,
                            n_jobs=-1)

## Train the model on train-set
etr.fit(X_train, Y_train_rm)

## Apply inference on test-set
Y_pred_rfr = etr.predict(X_test)
```

Figure – Régression par Extra Trees

Méthode d'ensemble

Gradient Boosting

Méthode de Boosting - Différence avec les autres méthodes d'ensembles :

Les méthodes de *Boosting* construisent un nombre k d'arbres en **séries** et non pas en **parallèles**.

Le $k+1$ arbre a alors accès à son prédécesseur et à son erreur commise. Ce $k+1$ arbre va alors chercher à corriger l'erreur commise par son prédécesseur.

Méthode d'ensemble

Gradient Boosting

Principe du Gradient Boosting :

- *Gradient Boosting* applique de la descente de gradient lors de la construction itérative des arbres.
- Comparaison des résultats obtenus pour chaque nouvel estimateur h_i, $1 \leq i \leq k$, à l'erreur laissée par h_{i-1}. La construction se fait donc itérativement et la descente de gradient est applicable.

Hyper-paramètres :

- *learning_rate* : Paramétrage du taux d'apprentissage α lors de la descente de gradient.

Méthodes d'ensemble

Gradient Boosting - Implémentation dans scikit-learn

```
from sklearn.ensemble import GradientBoostingRegressor

## Instance of GBR and definie its hyperparameters
gbr = GradientBoostingRegressor( learning_rate=0.1,
                                 max_depth=23,
                                 max_features=11,
                                 min_samples_split=6,
                                 n_estimators=110,
                                 )

## Train the model on train-set
gbr.fit(X_train, Y_train_rm)

## Apply inference on test-set
Y_pred_rfr = gbr.predict(X_test)
```

Figure – Régression par Gradient Boosting

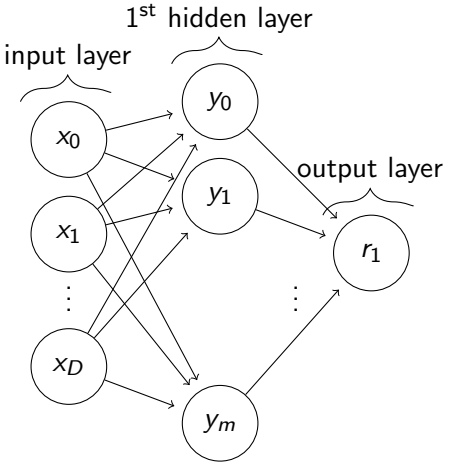
Réseau de neurones

Introduction

Le Perceptron :

Introduit dans par Rosenblatt, 1962. Il est constitué d'une couche d'entrée, d'une couche cachée et d'une couche de sortie.

$$\forall i, r_i = \sigma\left(\sum_j w_{j,i}\mathbf{x} + \mathbf{b_i}\right) \quad (5)$$

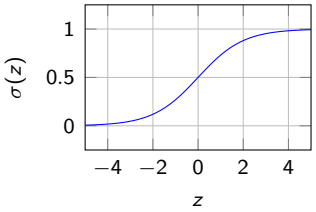


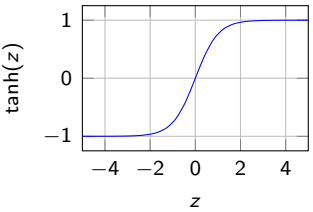
Réseau de neurones

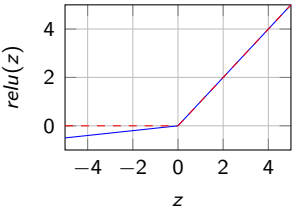
Fonction d'activation

Fonction d'activation :

Dans le domaine des réseaux de neurones artificiels, la fonction d'activation est une fonction mathématique appliquée à un signal en sortie d'un neurone artificiel (σ). La fonction d'activation est souvent une fonction non linéaire.





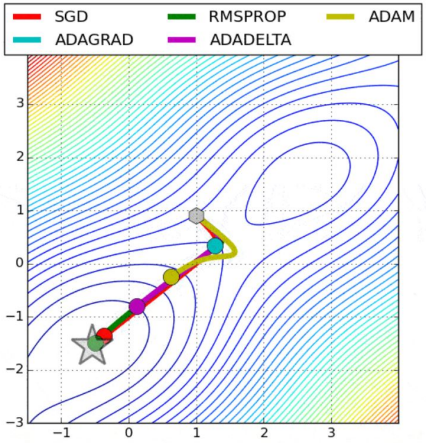


Réseau de neurones

Optimizer

Le Perceptron :

- Descente de gradient
- Descente de gradient stochastique
- Descente de gradient par mini-batch
- Adagrad
- AdaDelta
- Adam



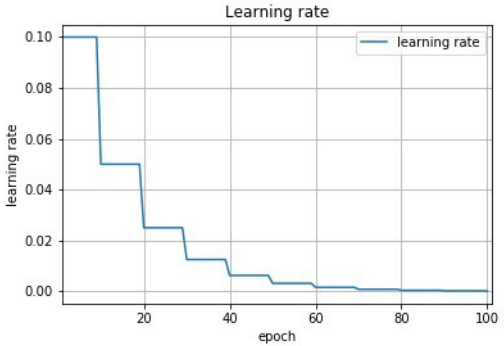
Réseau de neurones

Learning Rate Scheduler

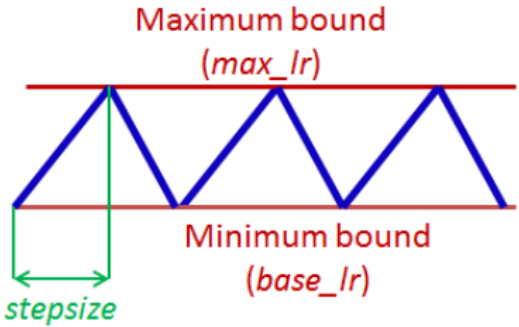
Figure – Différents cas pour le learning rate

Réseau de neurones

Learning Rate Scheduler



(a) ReduceOnPlateau

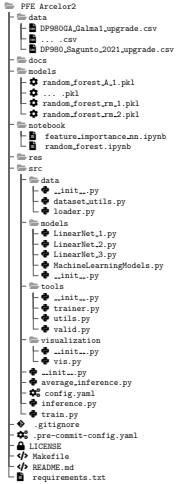


(b) CyclicLR

Plan

Architecture de dépôt GitLab

Arborescence



Éléments de l'arborescence :

- `./data` : Fichiers de données, fichiers *.csv* correspondants aux données de chaque ligne.
- `./docs` : Fichiers de documentation du code.
- `./models` : Modèles développés entraînés. Répertoire de sortie après entraînement des modèles.
- `./notebooks` : Notebooks pour déboguer et tester certaines implémentations.
- `./src` : Scripts python du projet.

Paramétrage du code

YAML file

```yaml
DATASET:
  DATA_FORMAT: csv
  PREPROCESSING:
    MERGE_FILES:
      WHICH: "all"  # Either "all" or list of indexes.
      # ex : [2] or [0, 1, 2] or [2, 0] ...
    NORMALIZE:
      TYPE: "StandardScaler"  # "MinMaxScalar"
    REMOVE_FEATURES:
      ACTIVE: True
      WHICH: ["Linespeed (m/min)", "B ppm"]
    REMOVE_SAMPLES:
      ACTIVE: False
      WHICH:
        #Direction: "T"  # ["T", "L"]
        Type: "JI5"  # ["JI5", "I2O"]
    TARGET: "rm"  # ["rm", "re02", "A80"]
  BATCH_SIZE: 32
  TEST_VALID_RATIO: [0.1, 0.2]
  VERBOSITY: True
  NUM_THREADS: 4
```

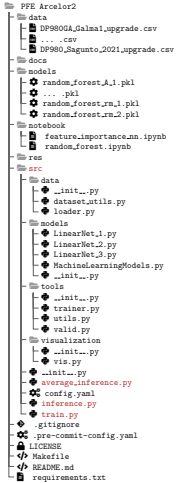
Figure – Extrait YAML configuration du dataset

Ce qui est paramétrable :

- Dataset :
  - Retirer des variables de la base d'apprentissage.
  - Retirer des lignes de données de la base d'apprentissage.
  - Choix de la/des lignes de fabrication.
  - ...
- Choix et Paramètres des modèles.
- Paramètres pour l'entraînement.
- Paramètrage pour l'inférence.
- ...

Exécution du code

Arborescence



Exécution des scripts python :

- Entraînement d'un modèle de machine learning ou d'un réseau de neurones :
  `python3 train.py --path_to_config ./config.yaml`
- Inférence :
  `python3 inference.py --path_to_config ./config.yaml`
- Inférence avec une moyenne de modèles :
  `python3 average_inference.py --path_to_config ./config.yaml`

Plan

Résultats - Modèles de Machine Learning - Méthodes d'ensemble

Prédiction de R_m par Random Forest sur 6 lignes

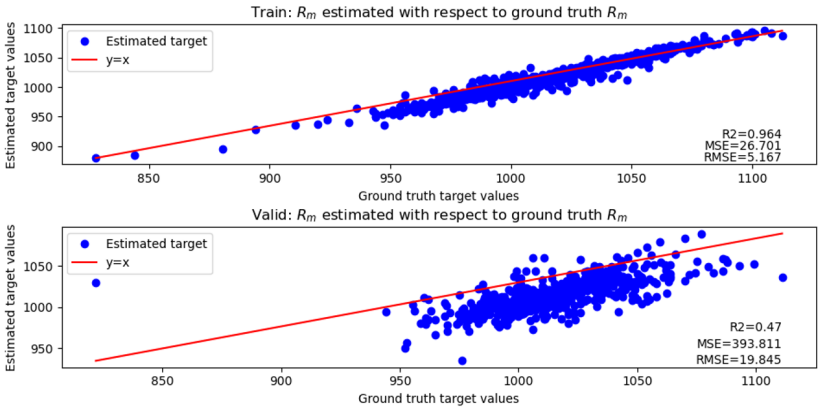


Figure – Prédiction de R_m en fonction de la valeur réelle

Résultats - Modèles de Machine Learning - Méthodes d'ensemble

Features importances de R_m par Random Forest sur 6 lignes

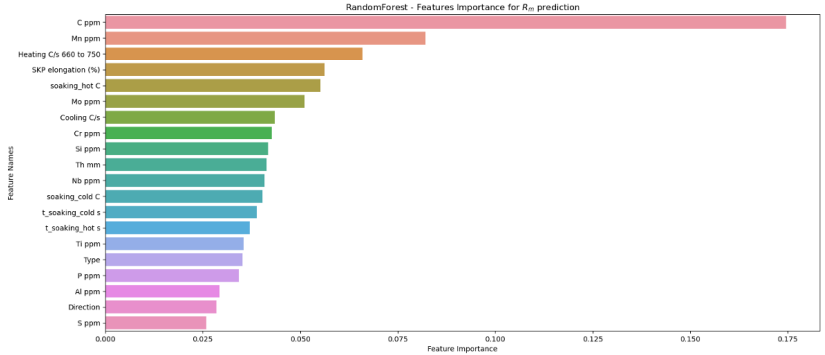


Figure – Features importances de R_m en fonction de la valeur réelle

Résultats - Modèles de Machine Learning - Méthodes d'ensemble

Résultats sur 6 lignes

6 lignes		ML			
Rm		Méthodes d'ensemble			
		RF	ET	GB	Average
RMSE	Train	5,17	7,47	3,62	4,91
	Valid	19,84	19,21	20,14	19,55
	Test				17,85

6 lignes		ML			
Re02		Ensemble			
		RF	ET	GB	Average
RMSE	Train	8,51	11,2	7,17	8,31
	Valid	28,76	26,77	29,97	27,88
	Test				27,87

6 lignes		ML			
A80		Méthode d'ensemble			
		RF	ET	GB	Average
RMSE	Train	3,63	6,75	3,36	4,1
	Valid	13,86	13,35	14,37	13,78
	Test				14,01

Résultats - Modèles de Machine Learning - Méthodes d'ensemble

Features importances sur 6 lignes

Résistance Mécanique - R_m			
Modèle	Feature 1	Feature 2	Feature 3
Random Forest	C (17%)	Mn (8%)	Heating C/s (7%)
Extra Trees	C (15%)	Mn (9%)	Direction (5%)
Gradient Boosting	C (17%)	Heating C/s (8%)	Mn (8%)

Résistance Élastique - $R_{e,0.2}$			
Modèle	Feature 1	Feature 2	Feature 3
Random Forest	soaking hot C (24%)	Type (21%)	SKP elongagion (14%)
Extra Trees	Type (25%)	soaking hot C (16%)	Direction (11%)
Gradient Boosting	soaking hot C (24%)	Type (19%)	SKP elongagion (14%)

Allongement à la Rupture - $A_{80\%}$			
Modèle	Feature 1	Feature 2	Feature 3
Random Forest	Type (28%)	soaking hot C (13%)	Th (6%)
Extra Trees	Type (37%)	soaking hot C (9%)	Direction (6%)
Gradient Boosting	Type (27%)	soaking hot C (13%)	SKP elongagion (7%)

Résultats - Réseau de neurones

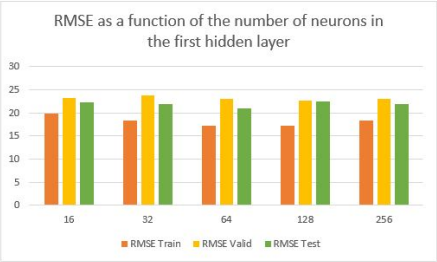
Résultats 6 lignes

6 lignes		NN								
Rm		1 Layer				2 Layers				
		32	64	128	256	16	32	64	128	256
RMSE	Train	21,2	24,9	23,6	22,3	19,8	18,4	17,3	17,3	18,3
	Valid	23,1	25,9	24,2	22,8	23,2	23,9	23	22,7	23
	Test	23,5	26,6	25,1	23,2	22,2	22	20,9	22,5	21,9

6 lignes		NN								
Re02		1 Layer				2 Layers				
		32	64	128	256	16	32	64	128	256
RMSE	Train	42,9	42,8	44,5	41,9	42	39,6	39,5	40,4	42
	Valid	42,8	41,8	43,2	40,8	43	43,1	43,7	43,8	43,6
	Test	44,1	43,3	44,1	43	46,7	44,5	44,8	46,3	47,9

6 lignes		NN								
A80		1 Layer				2 Layers				
		32	64	128	256	16	32	64	128	256
RMSE	Train									7,13
	Valid									9,92
	Test									11,2

(a) Performances NN sur les six lignes



(b) Évolution des performances sur des réseaux à deux couches

Résultats - Réseau de neurones

Résultats sur les différentes lignes - Résistance mécanique

Dataset	RMSE	R2
Train	11.5	0.74
Valid	18.05	0.15
Test	20.30	-0.08

(a) Galma

RMSE	R2
8.75	0.85
18.52	0.25
16.30	0.44

(b) SDG3

RMSE	R2
21.09	0.85
33.16	-3.19
50.09	0.67

(c) EKO1

Dataset	RMSE	R2
Train	12.48	0.75
Valid	22.37	0.31
Test	20.88	0.40

(d) Sagunto

RMSE	R2
12.02	0.83
26.55	0.33
19.59	0.52

(e) SDG3 V2

RMSE	R2
9.31	0.82
24.29	-0.29
32.81	-1.32

(f) SDG3.5

Résultats - Réseau de neurones

Résultats sur les différentes lignes - Résistance élastique

Dataset	RMSE	R2
Train	15.44	0.79
Valid	26.56	0.44
Test	28.82	0.28

(a) Galma

RMSE	R2
21.72	0.74
40.30	0.07
37.47	0.21

(b) SDG3

RMSE	R2
16.58	0.94
28.64	0.48
35.04	0.90

(c) EKO1

Dataset	RMSE	R2
Train	12.03	0.79
Valid	21.74	0.21
Test	16.62	0.60

(d) Sagunto

RMSE	R2
19.76	0.90
33.03	0.75
27.38	0.79

(e) SDG3 V2

RMSE	R2
8.80	0.92
33.09	-0.09
26.63	0.18

(f) SDG3.5

Résultats - Réseau de neurones

Résultats sur les différentes lignes - Allongement à 80%

Dataset	RMSE	R2
Train	7.20	0.81
Valid	8.59	0.69
Test	10.49	0.56

(a) Galma

RMSE	R2
9.84	0.56
15.36	-0.08
12.91	0.12

(b) SDG3

RMSE	R2
9.17	0.79
17.38	-0.16
12.77	0.75

(c) EKO1

Dataset	RMSE	R2
Train	11.54	0.25
Valid	15.48	-0.22
Test	14.92	-0.55

(d) Sagunto

RMSE	R2
12.71	0.65
17.99	0.14
19.18	0.25

(e) SDG3 V2

RMSE	R2
7.28	0.70
14.74	-1.09
9.71	-0.95

(f) SDG3.5

Résultats - Réseau de neurones

Importances des caractéristiques dans la prédiction

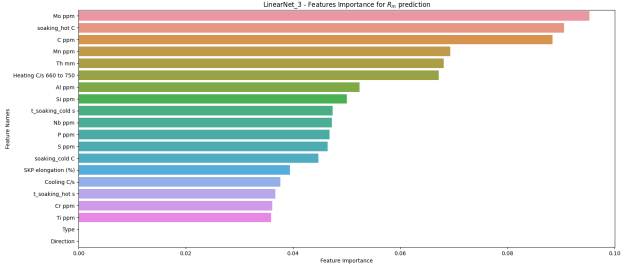


Figure – Importance des caractéristiques

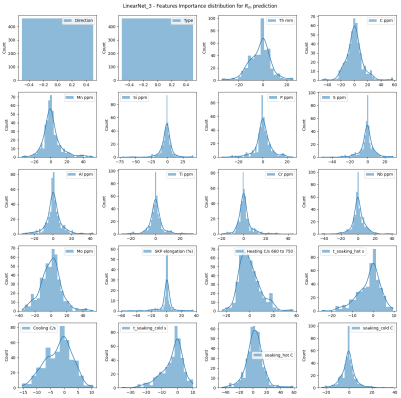


Figure – Distribution de l'importance des caractéristiques

Résultats - Réseau de neurones

Importances des caractéristiques dans la prédiction

rm	feature_1	feature_2	feature_3
Galma	Heating (C/s)	SKP elongation (%)	C (ppm)
SDG3 V2	Direction	soaking hot (C)	C (ppm)
SDG3.5	Heating (C/s)	C (ppm)	Ti (ppm)
EKO1	Heating (C/s)	soaking cold (C)	soaking hot (C)
SDG3	Mo (ppm)	soaking hot (C)	C (ppm)
Sagunto	Th (mm)	C (ppm)	Si (ppm)
All	Coolling (C/s)	C (ppm)	Th (mm)

Résultats - Réseau de neurones

Importances des caractéristiques dans la prédiction

re02	feature_1	feature_2	feature_3
Galma	Heating (C/s)	Cooling (C/s)	soaking hot (C)
SDG3 V2	soaking hot (C)	SKP elongation (%)	Th (mm)
SDG3.5	soaking cold (C)	Th (mm)	Mo (ppm)
EKO1	Heating (C/s)	Th (mm)	soaking cold (C)
SDG3	soaking hot (C)	Th (mm)	t_soaking_hot (s)
Sagunto	Th (mm)	t_soaking_hot (s)	t_soaking_cold (s)
All	Th (mm)	Heating (C/s)	Cooling (C/s)

Résultats - Réseau de neurones

Importances des caractéristiques dans la prédiction

A80%	feature_1	feature_2	feature_3
Galma	Type	Heating (C/s)	Cooling (C/s)
SDG3 V2	soaking hot (C)	Th (mm)	Mn (ppm)
SDG3.5	Cooling (C/s)	t_soaking_cold (s)	soaking hot (C)
EKO1	Heating (C/s)	Cooling (C/s)	Th (mm)
SDG3	Th (mm)	soaking hot (C)	t_soaking_cold (s)
Sagunto	Heating (C/s)	Cooling (C/s)	t_soaking_hot (s)
All	Heating (C/s)	t_soaking_cold (s)	Th (mm)

Plan

Conclusion

Points à retenir

- Suite à nos études, il semble possible de pouvoir prédire les caractéristiques mécaniques des aciers à partir de la composition chimique, des paramètres des traitements thermiques et des paramètres des traitements mécaniques ;
- Avec si peu de données, il est compliqué pour les algorithmes de généraliser. La variation de la target avec des inputs égaux n'aident pas ;
- Après l'étude des données, il semble possible de déterminer la cause de potentielles anomalies ;
- Avec plus de données et des lignes "proches", il sera possible d'inférer des résultats sur des lignes non vues lors de l'entraînement.