Maëlis COLLOMB
Arthur BUISSON
Cédric YOGANATHAN
Groupe A
Promo 2025

Rapport projet Apprentissage Machine

Rostom KACHOURI

08/05/2023

Introduction :

Dans le cadre de notre cours électif « Introduction à l'apprentissage Machine », nous avons réalisé un projet qui a pour but de modéliser la variation journalière du prix de l'électricité en France et en Allemagne. Nous avions une grande base de données de variation qui a comme variables par exemple la pluie, la température, la consommation totale d'électricité ou encore la production d'énergie journalière nucléaire. Chaque variable est respectivement mesurée en France et en Allemagne.

Ce projet va nous permettre d'interpréter des données afin de comprendre la variation du prix de l'électricité pour chaque jour dans les deux pays.

Nous avons trois bases de données : Data_X, qui sont les données sur lesquelles on a travaillé, Data_Y, qui sont les données en sortie et enfin DataNew_X qui sont les données sur lesquelles nous allons tester notre modèle définitif.

Durant ce projet, nous avons préparé les données, puis nous les avons analysé, modélisé et enfin nous avons élaboré des modèles.

Ce rapport décrira notre travail en détail pour chacune de ces étapes.

Préparation des données :

On a utilisé de nombreuses librairies pour la réalisation de ce projet :

- Numpy pour les calculs mathématique logique et de matrices
- Pandas pour manipuler les données
- Seaborn et matplotlib pour visualiser les données à l'aide de graphiques
- La fonction MinMaxScaler de Scikit-learn pour normaliser les données
- La fonction train_test_split pour les tests de données
- Sklearn.cluster pour le clustering de données.

On importe les données avec la commande de Pandas pd.read_csv().

On affiche les statistiques de nos données grâce à describe() de la librairie pandas, c'est-à-dire des informations sur la moyenne, la médiane, le minimum...

On affiche le nombre de valeurs nulles par colonnes, c'est-à-dire pour chaque variables grâce à la fonction isnull() puis on fait la somme pour avoir le nombre de valeurs avec sum(). Afin de ne pas garder des valeurs nulles dans nos données, on les remplace par la moyenne des valeurs de chaque colonne correspondantes avec fillna() et mean() pour la moyenne.

Par la suite, on normalise les données avec la fonction MinMaxScaler() et une boucle for qui traite chaque données.

On supprime enfin les données inutiles avec la fonction drop() de pandas. On expliquera dans la partie suivante pourquoi nous avons choisi ces données.

Analyse exploratoire des données :

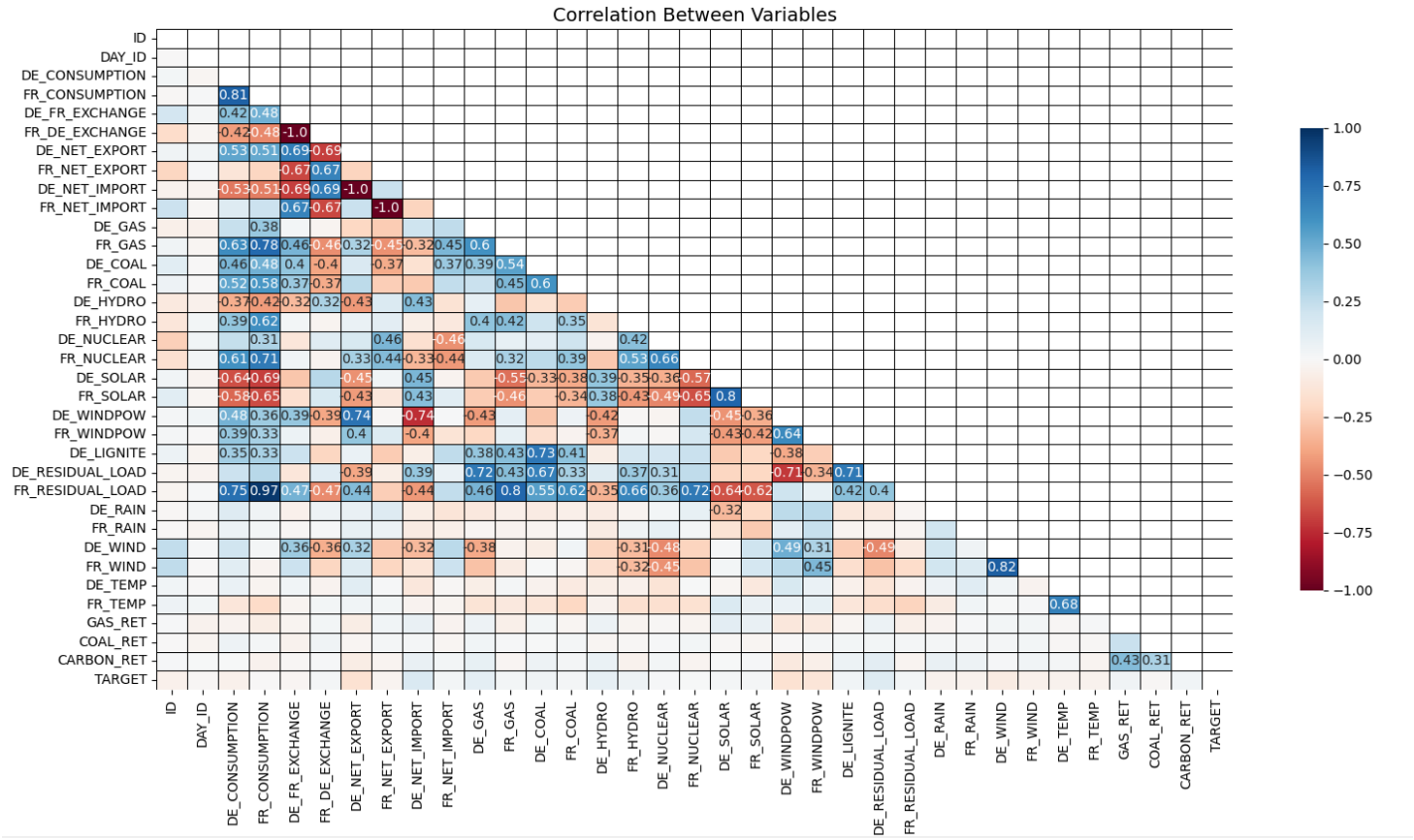
La variable à trouver dans notre projet et la variation par jour du prix de l'électricité, ce qui correspond à la colonne TARGET dans notre base de données Data_Y.

On réutilise la fonction describe() pour avoir les nouvelles statistiques sur les données normalisées, ainsi que la fonction info() pour avoir le type de variable de nos données.

Afin de mieux interpréter nos données et les relations qui les unissent, on fait une matrice de corrélation de Data_X. On utilise la fonction corr() qu'on assigne à une nouvelle variable. La matrice étant symétrique, on génère un masque qui nous permettra d'afficher uniquement le triangle inférieur ainsi que la diagonale. On créée un label pour les corrélations comprises entre -0.3 et 0.3 afin de ne pas les afficher car elle ne nous sont pas utiles. On donne le titre « Correlation Between Variables » à notre matrice.

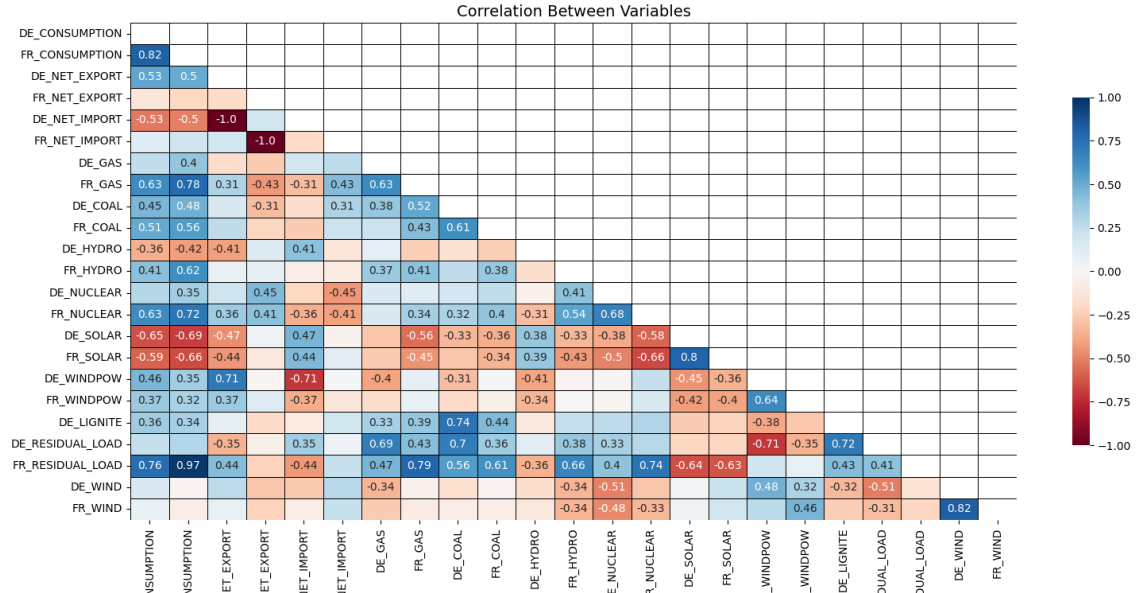
Enfin, on utilise la librairie Matplotlib afin de pouvoir afficher notre matrice. Tout d'abord, on créée une figure avec la fonction figure(). Ensuite, on crée une

heatmap afin d'afficher la corrélation entre nos différentes variables. Puis on l'affiche avec la fonction show().

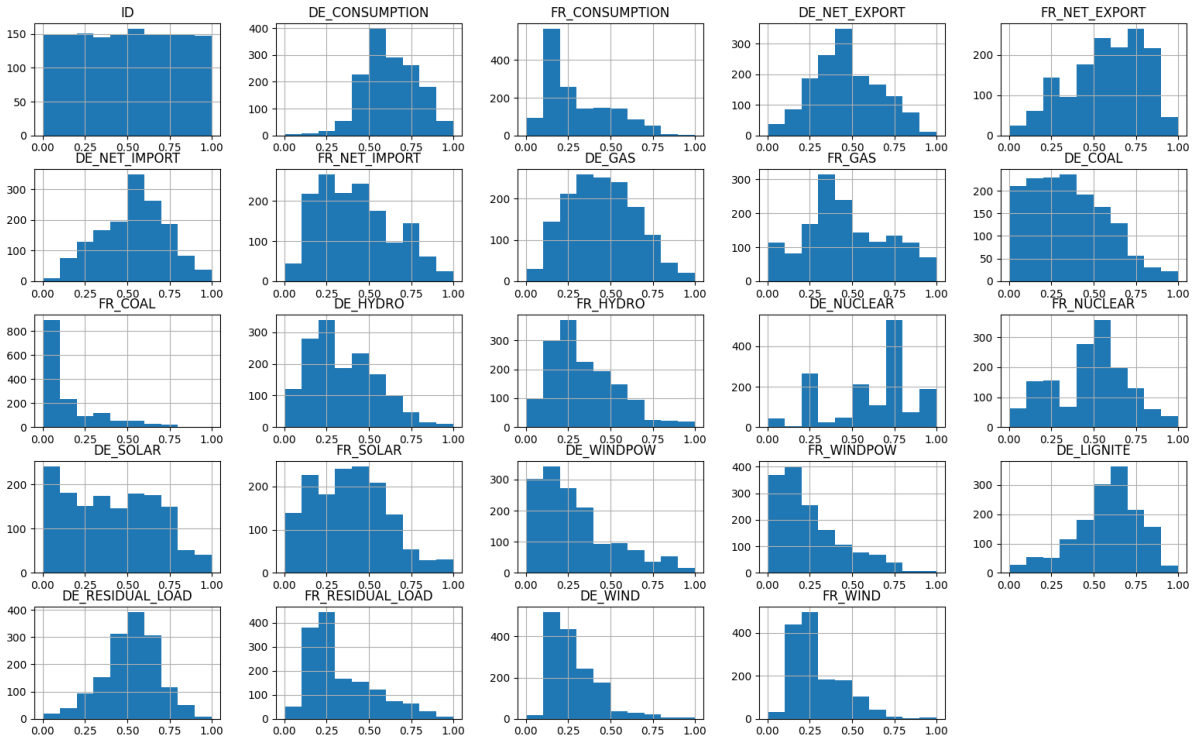


On peut remarquer que certaines variables sont moins utiles que d'autres pour notre projet. Par exemple la variable correspondant à la variation de la pluie en France et en Allemagne respectivement FR_RAIN et DE_RAIN ne sont pas importantes ici pour déterminer le prix de l'électricité car elles ont des coefficients de corrélations très bas avec les autres variables. Ces variables ont été supprimés dans la préparation des données comme expliqué précédemment. Nous avons fait de même avec COUNTRY, DAY_ID, FR_TEMP, DE_TEMP, GAS_RET, COAL_RET, CARBON_RET, DE_FR_EXCHANGE et FR_DE_EXCHANGE.

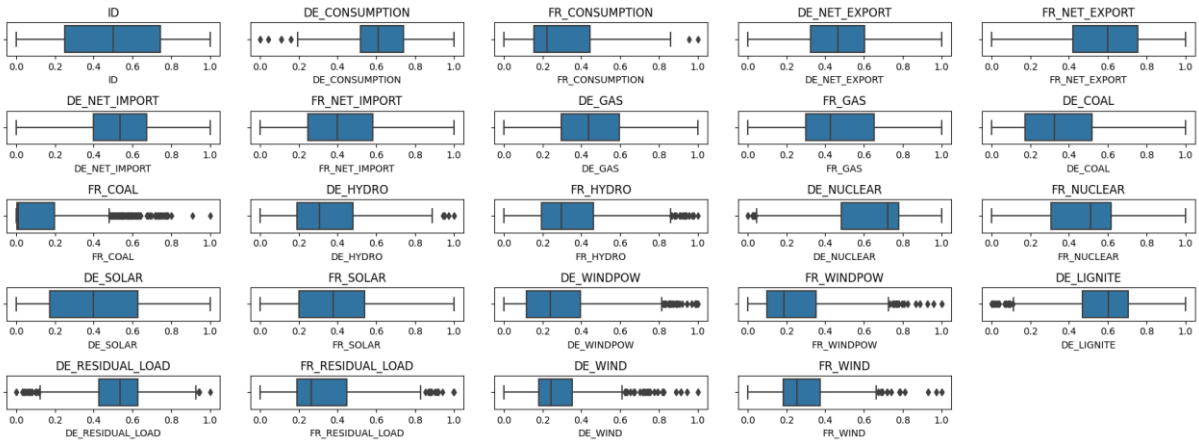
Voici donc la nouvelle matrice de corrélation sans les variables que l'on a jugé sans intérêt dans notre cas :



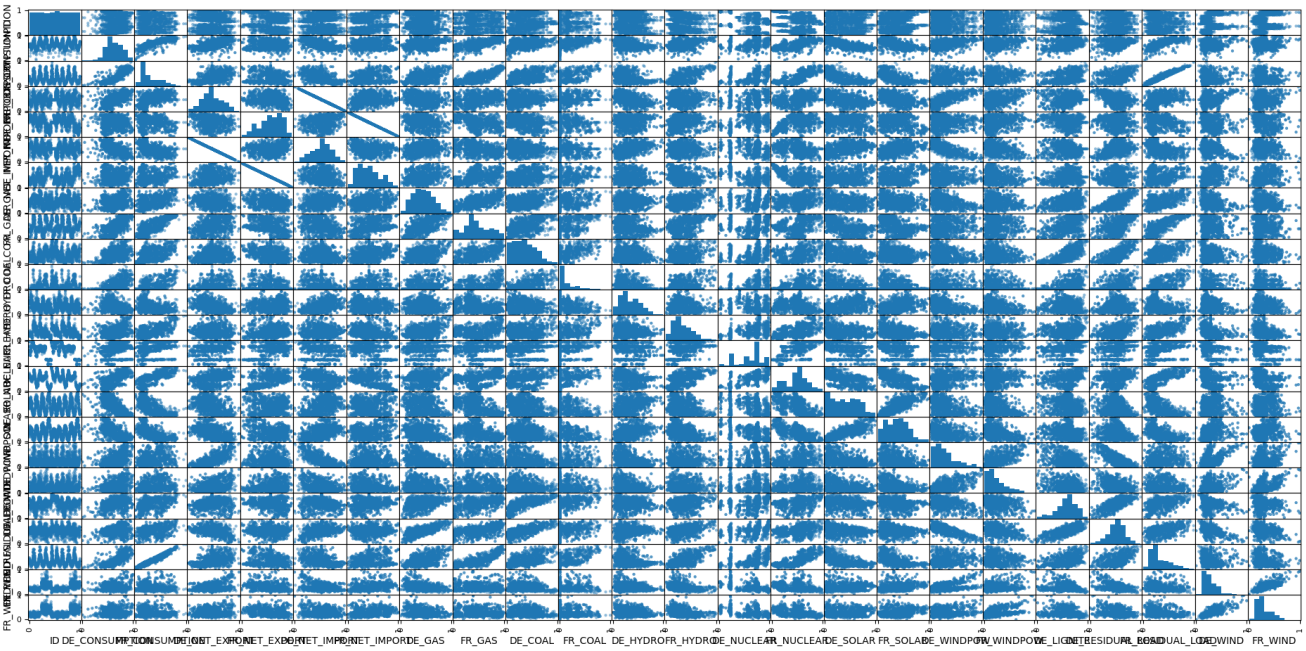
Nous avons également réaliser d'autre graphiques pour analyser nos données comme un histogramme avec la fonction hist() :



Ainsi qu'un diagramme en boîte de 7 lignes et 5 colonnes avec subplots() :



Et un diagramme de dispersion avec scatter_matrix(). Les voici :



Modélisation des données :

On importe toutes les fonctions dont nous avons besoin pour les six modèles différents de la librairie sklearn.
On crée nos variables de test avec la fonction split(). On sépare nos données en deux parties : 70% pour l'entraînement et 30% pour le test.

Chaque modèle fonctionne de la même manière : on crée un objet pour le modèle choisi, on entraîne le modèle avec les données d'entraînement puis on fait les prédictions avec les données de test et on affiche le résultat.

On commence avec la régression linéaire simple. On crée un objet de modèle de regréssion linéaire avec LinearRegression(). On entraine le modèle créé avec fit() et nos variables d'entraînement, puis on fait les prédictions sur les données de test et on les affiche.

Pour la régression ridge et lasso, on crée l'objet du modèle de régression ridge avec Ridge() et lasso avec Lasso() puis on suit le même processus que pour la régression linéaire simple.
Aussi, la régression ridge permet de restreindre les coefficients à des valeurs plus petites, ce qui peut aider à éviter le surajustement en réduisant la variance.
La régression lasso elle permet de conduire certains coefficients à zéro, ce qui peut aider à éviter le surajustement en produisant des modèles plus simples avec moins de variables explicatives.

On utilise KNeighborsClassifier() pour crée l'objet pour la méthode KNN, puis on entraîne et on fait les prédictions.

Enfin, pour notre dernier modèle, on utilise DecisionTreeRegressor() pour l'objet.

Evaluation des méthodes :

Pour chaque modèle, on calcule l'erreur quadratique moyenne (RMSE) en calculant la racine de la moyenne pour chaque résultat de modèle avec les résultats sur les données de test. Le RMSE est une mesure de l'erreur de prédiction d'un modèle, où des valeurs plus faibles indiquent une meilleure performance.

Ensuite, on calcule les coefficient de détermination grâce à la fonction r2_score() pour chaque modèle. Le R2 mesure à quel point les données s'ajustent au modèle. Un R2 plus élevé indique une meilleure performance du modèle

Enfin, on calcule la corrélation de Spearman pour chaque modèle grâce à spearmanr() avec les résultats trouvé sur les données de test. Le coefficient de corrélation de Spearman est utilisé pour mesurer la corrélation entre deux variables, sans supposer qu'elles suivent une relation linéaire.

Voici les résultats pour les différents algorithmes :

```
RMSE pour la régression linéaire :  0.7157877791448191
RMSE pour la régression Ridge :  0.8890984243199416
RMSE pour la régression Lasso :  0.7145680974929007
RMSE pour la méthode des k-NN :  135.53584235738606
RMSE pour l'arbre de décision :  28.550079463638298

R2 pour la régression linéaire :  0.5020587000387935
R2 pour la régression Ridge :  0.5048628511522337
R2 pour la régression Lasso :  0.5037542093747169
R2 pour la méthode des k-NN :  0.44010194505412886
R2 pour l'arbre de décision :  0.4989967983409367

Spearman pour la régression linéaire :
rho :
 [[ 1.         -0.03674053  1.         -0.03675615]
 [-0.03674053  1.         -0.03674053  0.16826835]
 [ 1.         -0.03674053  1.         -0.03675615]
 [-0.03675615  0.16826835 -0.03675615  1.        ]]
pval :
 [[0.00000000e+00 4.37388896e-01 0.00000000e+00 4.37193948e-01]
 [4.37388896e-01 0.00000000e+00 4.37388896e-01 3.42058639e-04]
 [0.00000000e+00 4.37388896e-01 0.00000000e+00 4.37193948e-01]
 [4.37193948e-01 3.42058639e-04 4.37193948e-01 0.00000000e+00]]

Spearman pour la régression Ridge :
rho :
 [[ 1.         -0.03674053  1.         -0.04284057]
 [-0.03674053  1.         -0.03674053  0.18280861]
 [ 1.         -0.03674053  1.         -0.04284057]
 [-0.04284057  0.18280861 -0.04284057  1.        ]]
pval :
 [[0.00000000e+00 4.37388896e-01 0.00000000e+00 3.65115405e-01]
 [4.37388896e-01 0.00000000e+00 4.37388896e-01 9.79058060e-05]
 [0.00000000e+00 4.37388896e-01 0.00000000e+00 3.65115405e-01]
 [3.65115405e-01 9.79058060e-05 3.65115405e-01 0.00000000e+00]]
```

```
Spearman pour la régression Lasso :
rho :
 [[ 1.         -0.03674053  1.         -0.04136534]
 [-0.03674053  1.         -0.03674053  0.1769135 ]
 [ 1.         -0.03674053  1.         -0.04136534]
 [-0.04136534  0.1769135  -0.04136534  1.        ]]
pval :
 [[0.00000000e+00 4.37388896e-01 0.00000000e+00 3.81875008e-01]
 [4.37388896e-01 0.00000000e+00 4.37388896e-01 1.64532523e-04]
 [0.00000000e+00 4.37388896e-01 0.00000000e+00 3.81875008e-01]
 [3.81875008e-01 1.64532523e-04 3.81875008e-01 0.00000000e+00]]

Spearman pour la méthode des k-NN :
rho :
 [[ 1.         -0.03674053  0.96621361 -0.15986332]
 [-0.03674053  1.         -0.02425511  0.09081603]
 [ 0.96621361 -0.02425511  1.         -0.17828231]
 [-0.15986332  0.09081603 -0.17828231  1.        ]]
pval :
 [[0.00000000e+000 4.37388896e-001 2.46411022e-265 6.74127704e-004]
 [4.37388896e-001 0.00000000e+000 6.08231659e-001 5.44840491e-002]
 [2.46411022e-265 6.08231659e-001 0.00000000e+000 1.46062073e-004]
 [6.74127704e-004 5.44840491e-002 1.46062073e-004 0.00000000e+000]]

Spearman pour l'arbre de décision :
rho :
 [[ 1.         -0.03674053  0.9978017  -0.24093544]
 [-0.03674053  1.         -0.03723275  0.04353811]
 [ 0.9978017  -0.03723275  1.         -0.24146626]
 [-0.24093544  0.04353811 -0.24146626  1.        ]]
pval
:  [[0.00000000e+00 4.37388896e-01 0.00000000e+00 2.37352343e-07]
 [4.37388896e-01 0.00000000e+00 4.31269253e-01 3.57352868e-01]
 [0.00000000e+00 4.31269253e-01 0.00000000e+00 2.22900000e-07]
 [2.37352343e-07 3.57352868e-01 2.22900000e-07 0.00000000e+00]]
```

On voit donc la régression Lasso a le RMSE le plus faible de toutes les méthodes testées, suivie de la régression Linéaire simple, de la régression Ridge, de l'arbre de décision et de la méthode des k-NN.

La régression Lasso, elle a le R2 le plus élevé, suivi de près de la régression Ridge, de la régression Linéaire simple, de l'arbre de décision et de la méthode des k-NN.

Pour le coefficient de Spearman, il est difficile de déterminer un algorithme supérieur car les résultats sont très similaires pour tous les algorithmes. Les valeurs de rho pour chaque paire de variables sont proches de 1 ou de -1, indiquant une forte corrélation positive ou négative. Les p-values sont également très faibles, indiquant que ces corrélations sont statistiquement significatives.

En résumé, on peut dire que la régression Lasso a les performances les plus élevées selon les trois mesures utilisées, tandis que la méthode des k-NN et l'arbre de décision ont des performances relativement faibles.

Voici donc le classement des différents algorithmes :

1. Régression Lasso
2. Régression Linéaire simple
3. Régression Ridge
4. Arbre de décision
5. Méthodes K-NN

Conclusion :

Finalement, ce projet nous a permis d'approfondir nos compétences en Apprentissage Machine, et de nous entrainer sur un cas concret.

Nous avons désormais de meilleures connaissances sur les librairies, l'analyse de grande base de données et les modèles utiles dans ce domaine.

Enfin, nous avons aussi pu renforcer nos compétences d'esprit d'équipe et d'organisation.