École nationale supérieure de
l'électronique et de ses applications

Rapport de projet : Causal Feature Selection in large Multivariate Time Series (MTS) for Forecasting Problems

Élèves :
EL MAAZOUZ AMINE
Barrie Mathys
Tadjouri Imad-eddine

Enseignant :
Vassilis CHRISTOPHIDES
Etienne VAREILLE

17 décembre 2023

Table des matières

1 Description générale du projet

1.1 Mise en contexte

Les séries temporelles multivariées (MTS) sont omniprésentes dans de nombreux domaines scientifiques et d'ingénierie, notamment la santé, la finance, la météorologie. Une MTS est composée de plusieurs variables dépendantes du temps qui peuvent dépendre non seulement de leurs valeurs passées, mais aussi d'autres variables.

Lorsqu'on tente de prévoir (prédire) les pas de temps futurs d'une MTS en fonction des observations passées, le nombre de covariables (c'est-à-dire de variables dépendantes du temps mesurées simultanément) peut être trop élevé par rapport au nombre de points de référence (pas de temps), entraînant une modélisation imprécise en raison, paradoxalement, d'un manque de données.

1.2 Objectif du projet

Notre objectif est d'amener des éléments de réponse à 3 questions :

- Quelle est la meilleure combinaison d'algorithmes de sélection de variables et de prédiction ?
- Quelles sont les meilleures variables à sélectionner ? Comment se comportent-elles vis-à-vis du graphe causal ?
- Quel est l'influence des hyperparamètres sur les performances de nos algorithmes ?

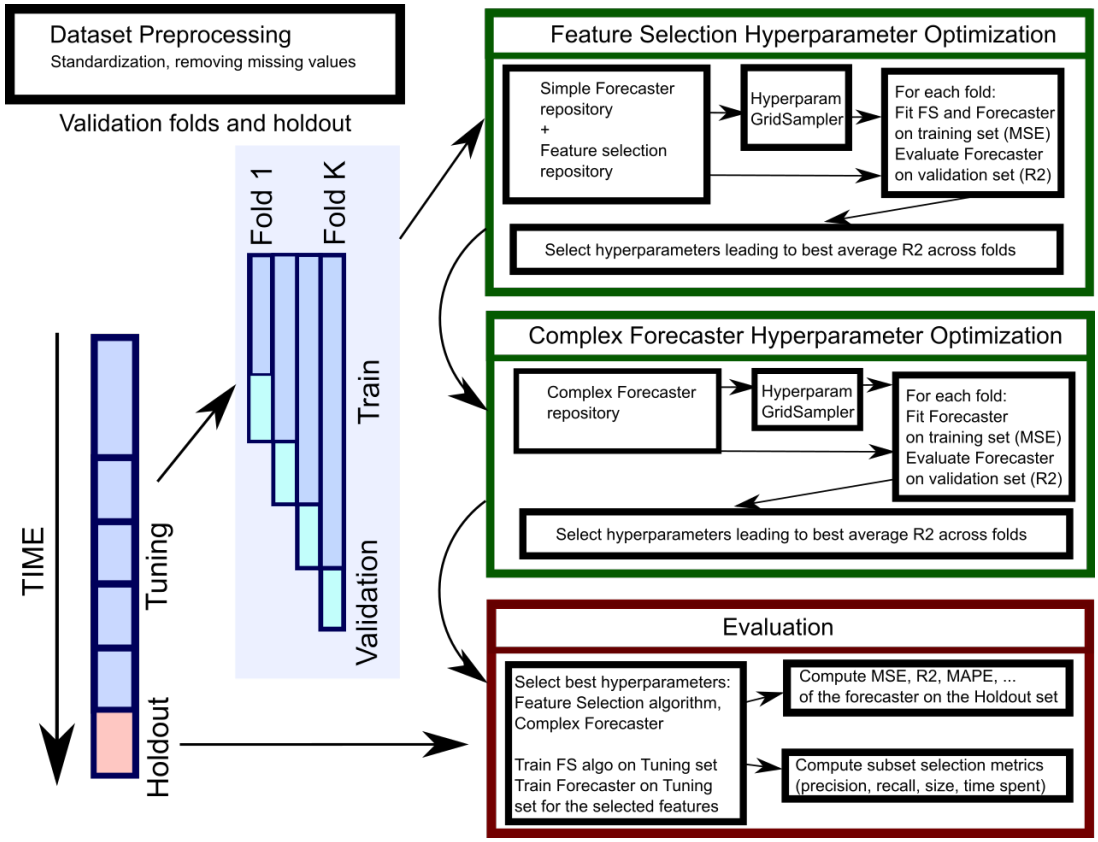
Pour cela, nous allons nous appuyer sur des résultats expérimentaux sur différents jeux de données.

1.3 Plan du projet

Notre projet va donc se répartir en deux parties :

- Une première partie focalisée sur l'étude des algorithmes de sélection de variables, leur compréhension, ainsi que leur implémentation. Nous aurons ensuite à analyser nos résultats expérimentaux, les expliquer puis les mettre en relation avec les algorithmes de prédiction.
- Une seconde partie qui consistera à étudier les algorithmes de prédiction, mettre en évidence le besoin de la sélection de variable, évaluer les résultats de la première partie.

1.4 Pipeline utilisé



1.5 Les bases de données utilisées

— Monotonic : 1000 variables, 500 horodatages, non linéaire.
— Varsmall : 10 variables, 3500 horodatages, linéaire.
— FMRI : forte corrélation, 5 variables, non linéaire

2 Etude des algorithmes de sélection de variables

2.1 Introduction

La première phase de notre projet se consacre à une analyse approfondie des algorithmes de sélection de variables appliqués aux séries temporelles multivariées (MTS). Cette étape fondamentale vise à établir une compréhension approfondie des méthodes de sélection de variables et à explorer spécifiquement deux classes d'algorithmes : Rrelieff et MRMR.

L'étude débutera par une présentation exhaustive des différentes classes d'algorithmes de sélection de variables. Cette phase introductive jettera les bases conceptuelles en mettant en lumière les diverses approches utilisées pour identifier les variables les plus pertinentes dans un contexte de séries temporelles multivariées.

Le premier volet de notre exploration se concentrera sur l'algorithme Rrelieff. Ce dernier, initialement développé pour la classification, a été adapté pour la sélection de variables dans le contexte des séries temporelles. Nous examinerons en détail son mécanisme, ses avantages et ses limitations, ainsi que sa pertinence dans la sélection des caractéristiques temporelles cruciales.

Le deuxième algorithme mis en avant est MRMR, qui se distingue par sa capacité à établir un équilibre entre la réduction de la redondance entre variables et la maximisation de leur pertinence pour la prédiction. Nous explorerons son cadre conceptuel, son application spécifique aux séries temporelles multivariées, et son impact sur la qualité des modèles prédictifs.

Cette première partie constitue un socle essentiel pour la suite du projet, fournissant une compréhension approfondie des outils de sélection de variables spécifiquement adaptés aux défis posés par les MTS. En se concentrant sur Relief et MRMR, nous jetterons les bases nécessaires pour évaluer leur efficacité dans le contexte complexe des séries temporelles multivariées.

2.2 Présentation de quelques classes d'algrotihme de sélection de variables

2.3 Notre premier algorithme de sélection de variable : Rrelieff

2.3.1 Introduction

RReliefF est conçu pour évaluer les caractéristiques en fonction de leur pertinence vis-à-vis d'une variable cible. En se basant sur les différences entre les valeurs des caractéristiques, cet algorithme attribue des poids adaptés pour déterminer quelles caractéristiques sont les plus informatives pour la prédiction.

2.3.2 Explication de l'algorithme

Le filtre est fondé sur une approche itérative et suit les étapes suivantes :

— On commence déjà par définir : W_{dy} poids lié à la présence de valeurs différentes pour la variable réponse y , W_{dj} poids lié à la présence de valeurs différentes pour le prédicteur Fj et finalement $W_{dy \wedge dj}$ poids lié à la présence de valeurs différentes pour la variable réponse y et pour le prédicteur Fj en même temps.
— Les poids sont tous fixés à zéro
— On choisit aléatoirement une observation X_r
— Pour chaque voisin X_q parmi les k-voisins de X_r on met à jour les poids selon les formules suivantes :
— $W_{dy}^{i} = W_{dy}^{i-1} + \Delta_y(X_r, X_q).d_{rq}$ avec d_{rq} désigne la distance euclidienne entre r et q
— $W_{dj}^{i} = W_{dj}^{i-1} + \Delta_j(X_r, X_q).d_{rq}$
— $W_{dy \wedge dj} = W_{dydj}^{i-1} + \Delta_y(X_r, X_q).d_{rq}.\Delta_j(X_r, X_q)$

Avec $\Delta_y(X_r, X_q) = \frac{|y_r - y_q|}{max(y) - min(y)}$ signifie la différence normalisée entre les valeurs de la variable de réponse continue y pour deux observations spécifiques, $X_r et X_q$ avec $y_r et y_q$ qui représentent réspectivement la valeur de la variable de réponse pour l'obsérvation $X_r et X_q$

2.3.3 Implémentation

Pour appliquer l'algorithme RReliefF à nos données, nous avons utilisé la bibliothèque sklearn_relief, mon filtre comporte pas mal d'hyper paramètre mais on a tenu compte que de n qui représente le nombre voisin proche. Nous avons donc créé une fonction spécifique pour cette tâche, appelée apply_Rrelieff. Cette fonction prend en compte les étapes suivantes :

Apply Rrelieff

[415]:
```
def apply_Rrelieff(data,i,target):
    X,y,_=prepare_data_vectorize(data,config['lags'],target)
    X_df=pd.DataFrame(X,columns=[f'lag_{i}' for i in range(1,X.shape[1]+1)])
    num_features_to_select=config['num_features_to_select']
    rrelief_instance = sr.RReliefF(n_features=i)
    rrelief_instance.fit(X,y)
    selected_feature_indices = transform(rrelief_instance, X_df)
    vector_mask = [name in selected_feature_indices for name in range(len(X_df.columns))]
    selected_feature_names=vector_mask_to_columns(vector_mask,data)
    return selected_feature_names
```

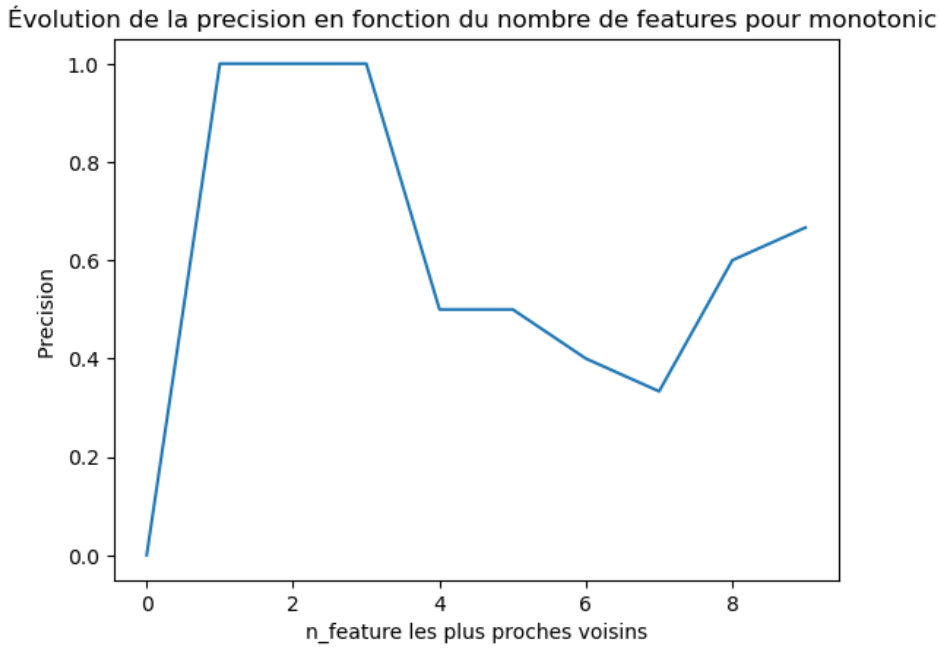
Cette fonction prend en entrée les données data, un paramètre i spécifiant le nombre de caractéristiques à sélectionner, et la variable cible target. Elle utilise la méthode fit de l'instance RReliefF pour ajuster le modèle aux données d'entraînement, puis utilise la méthode transform pour extraire les indices des caractéristiques les mieux classées en utilisant les poids calculés par l'algorithme RreliefF. Enfin, elle renvoie les noms des caractéristiques sélectionnées à partir des indices obtenus.

5

2.3.4 Résultats

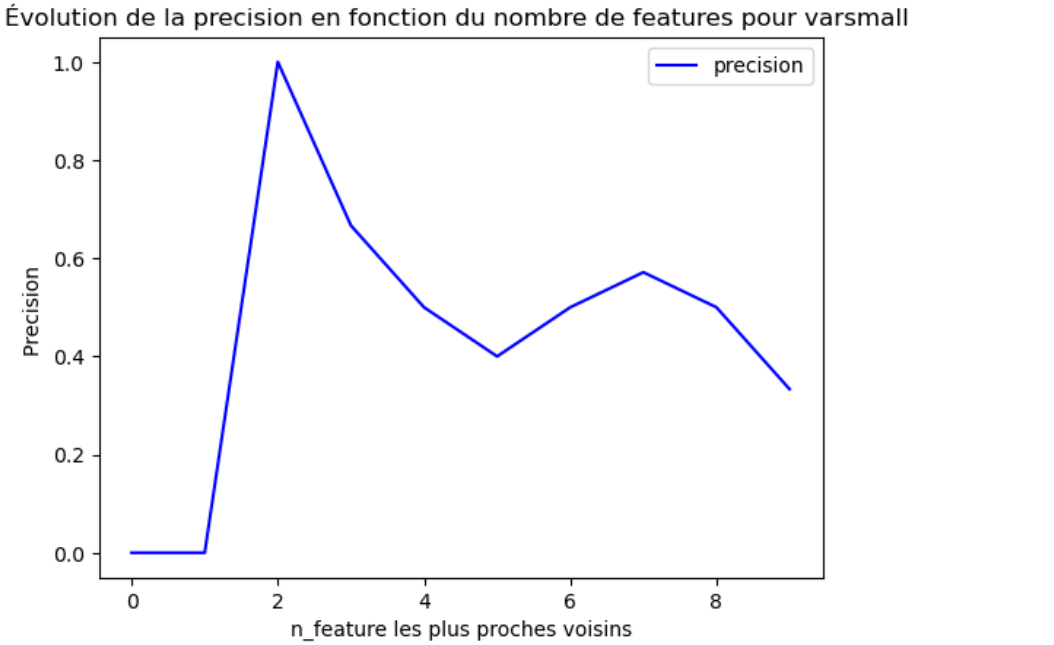
Pour la dataset Monotonic je trouve les résultats suivants :

```
Pour la target "3"
Text(0.5, 1.0, 'Évolution de la precision en fonction du nombre de features pour monotonic ')
```



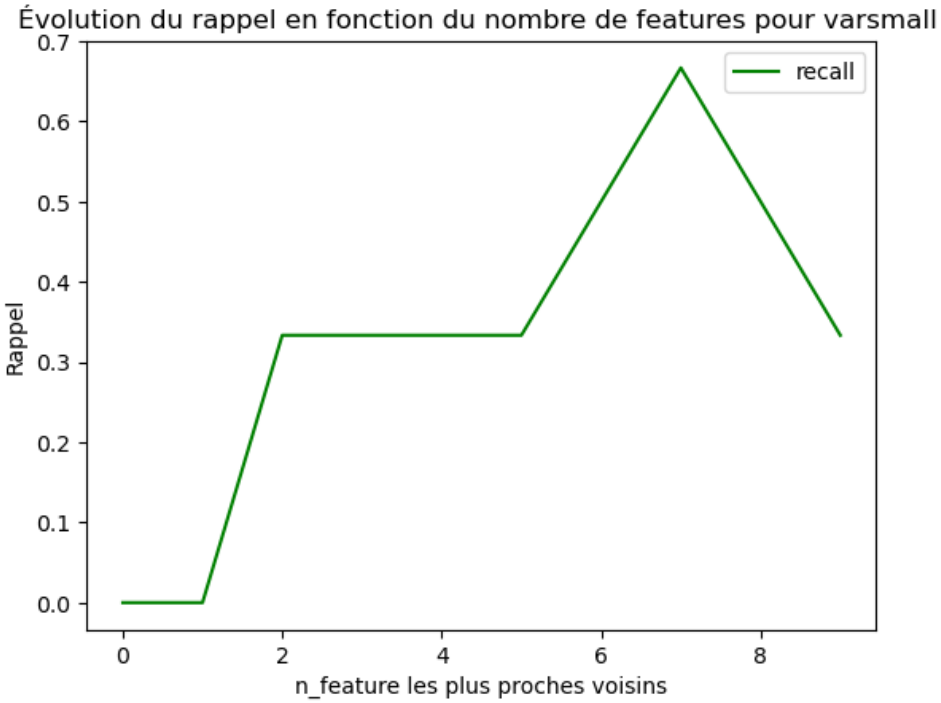
Pour la dataset Varsmall je trouve les résultats suivants :

```
Pour la target "2"
Text(0.5, 1.0, 'Évolution de la precision en fonction du nombre de features pour varsmall ')
```



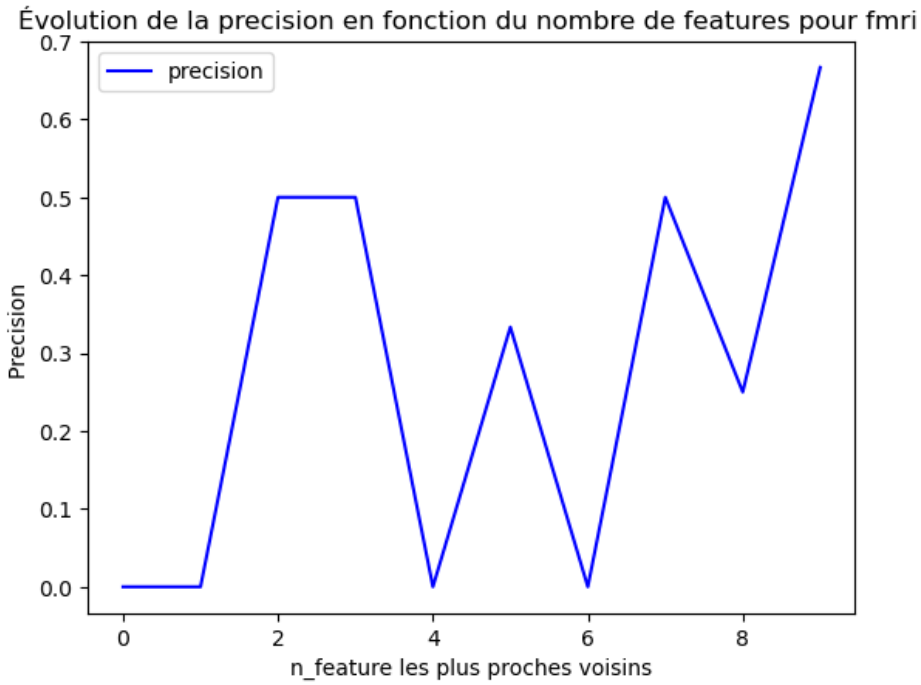
6

```
Pour la target "2"
Text(0.5, 1.0, 'Évolution du rappel en fonction du nombre de features pour varsmall ')
```

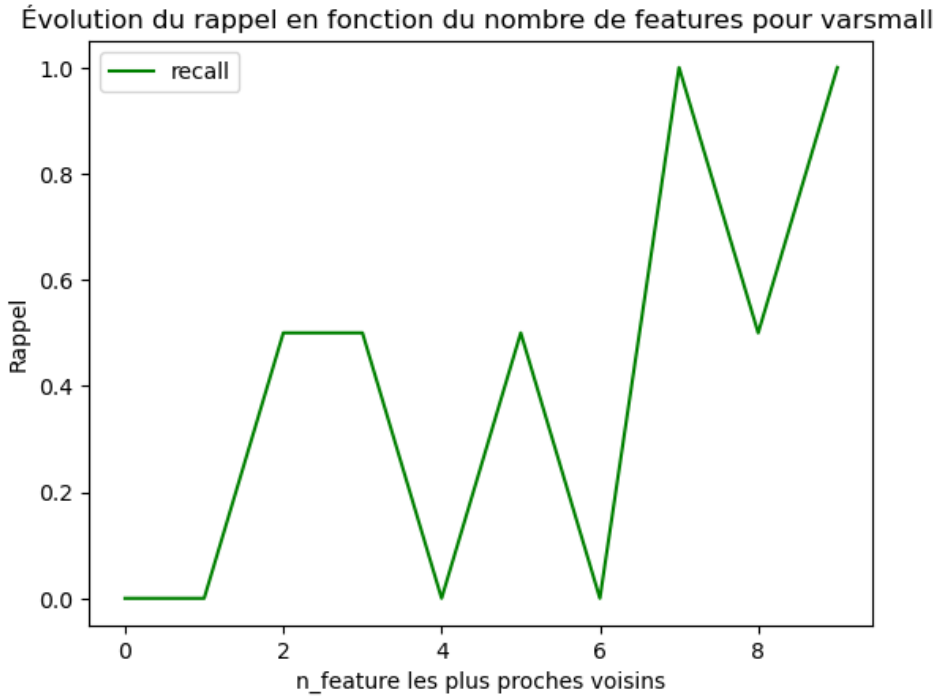


Pour la dataset FMRI je trouve les résultats suivants :

```
Pour la target "1"
Text(0.5, 1.0, 'Évolution de la precision en fonction du nombre de features pour fmri ')
```



```
Pour la target "1"
Text(0.5, 1.0, 'Évolution du rappel en fonction du nombre de features pour varsmall ')
```



2.4 Notre deuxième algorithme de sélection de variable : MRMR

2.4.1 Introduction

Dans le cadre de notre projet, nous explorons plusieurs méthodes de Feature Selection pour optimiser les pertinences des variables à utiliser dans l'algorithme de Forecasting. L'une des méthodes qui a été choisie est l'algorithme MRMR (Maximum Relevance Minimum Redundancy), une technique de Feature Selection qui cherche à identifier un ensemble de variables qui maximise la pertinence avec la variable cible tout en minimisant la redondance entre les variables sélectionnées. Nous allons établir les principes fondamentaux de cet algorithme, ses avantages ainsi que les résultats obtenus vis à vis des dataset définis.

2.4.2 Explication de l'algorithme

On peut décrire les étapes de l'algorithme MRMR ainsi :

- Pour chaque caractéristiques on calcule la mesure de pertinence avec la variable cible, la mesure de la pertinence va évaluer à quel point chaque caractéristique est individuellement liée à la variable cible. Ces mesures de pertinences se font selon des métriques comme la mesure de corrélation, l'information mutuelle...
- On classe les caractéristiques par rapport à leurs mesures de pertinences en ordre décroissant, les caractéristiques les plus pertinentes par rapport à la variable cible sont en tête de liste.
- On créer deux listes, une qui stocke les caractéristiques sélectionnée (S) et une autre pour les caractéristiques restantes (R).
- On choisit maintenant les caractéristiques de R qui ont la mesure de pertinence maximale avec la variable cible la plus élevée, on les place donc dans S.

— Pour chaque caractéristique dans S, on calcule la mesure de redondance entre la caractéristique sélectionnée et celle déjà présente dans S. La mesure de la redondance évalue à quel point une caractéristique apporte une information similaire à celle déjà sélectionnée. On peut mesurer la redondance avec des métriques comme le coefficient de détermination (R^2), l'information mutuelle ou la corrélation.
— On choisit les caractéristiques de R qui minimise la redondance avec les caractéristiques déjà sélectionnées dans S, on les ajoute à S et les retire de R.
— On répète le processus des deux dernières étapes jusqu'à avoir une taille de S désirée ou bien l'épuisement de R.

2.4.3 Implémentation

Pour implémenter l'algorithme MRMR, il fallait importer la fonction mrmr_regression depuis la bibliothèque mrmr installée sur l'IDE :

La fonction qui applique à proprement parler la sélection de variable selon MRMR est apply_mrmr.

Cette fonction possède comme arguments les données que l'on utilise comme support, le nombre de caractéristiques que l'on souhaite sélectionner et la variable cible que l'on choisit parmi le jeu de données. La fonction ajuste le modèle aux données et vectorise les colonnes du jeu de données, sur ces colonnes on applique la fonction mrmr_regression qui sélectionne les variables selon la méthode définit précédemment. Finalement, les caractéristiques sont renvoyées selon le nom qu'elle possède dans le jeu de données.

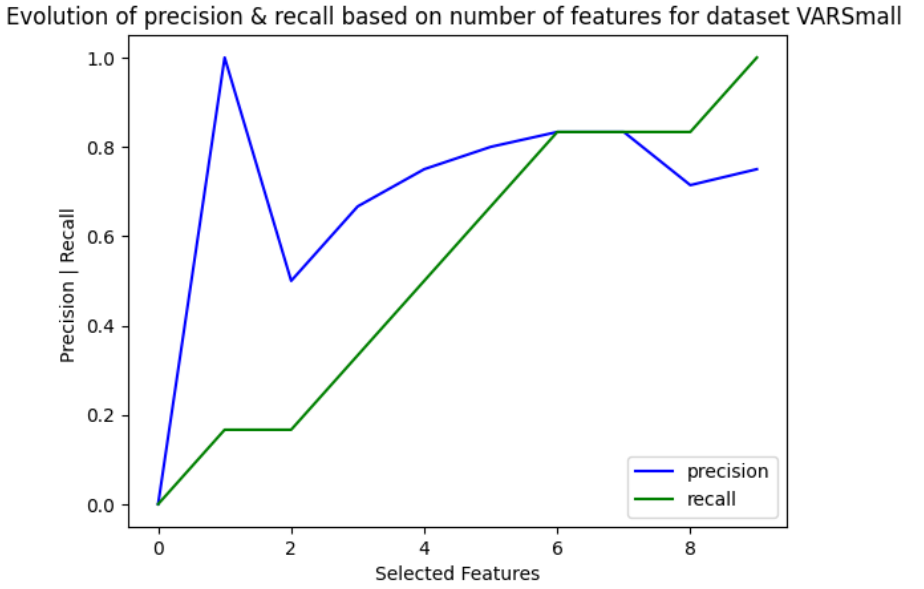
2.4.4 Résultats

Concernant les résultats, nous avons pût pour l'algorithme MRMR tester la sélection de variables sur les dataset que nous avons définis précedemment.

L'algorithme sélectionne dix caractéristiques et calcule les métriques de précision et rappel (recall) liée à celle-ci. Notre objectif étant de trouver le meilleur compromis entre ces deux métriques pour évaluer dans un premier temps la performance de notre algorithme.

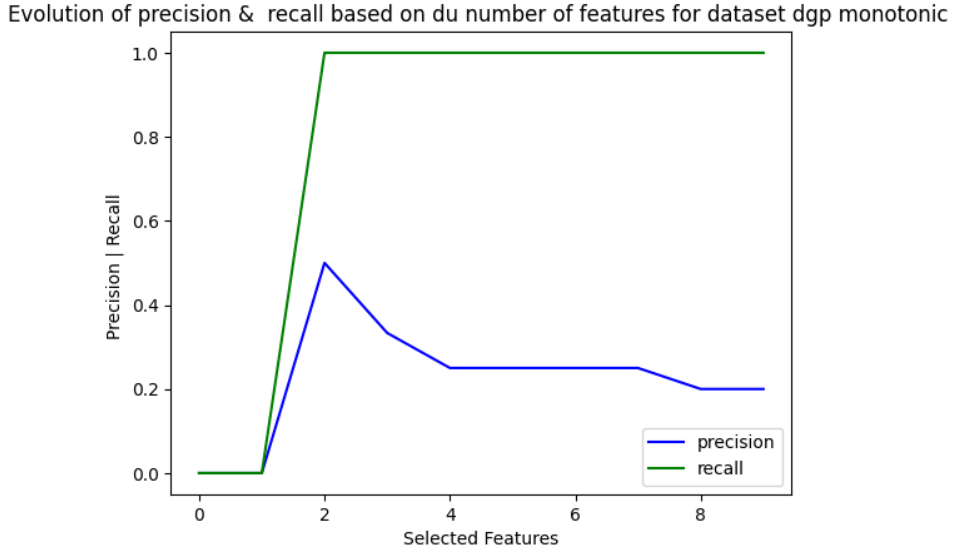
Pour le dataset VARSmall, nous obtenons ceci :

```
Pour la target "2"
Text(0.5, 1.0, 'Evolution of precision & recall based on number of features for dataset VARSmall ')
```



Pour le dataset dgp/monotonic, nous obtenons ceci :

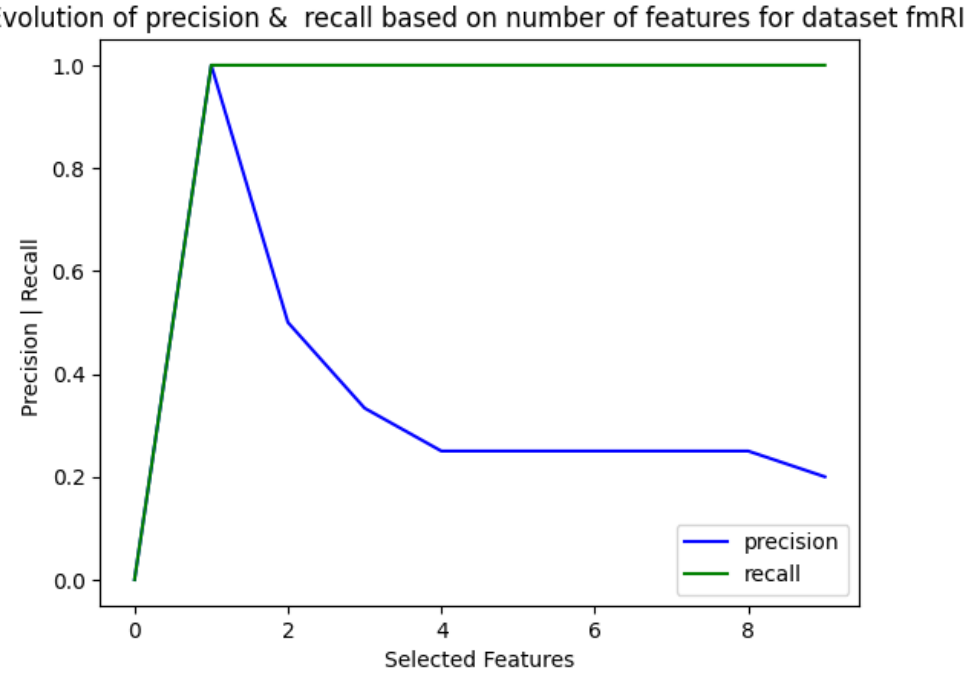
```
Pour la target "X3"
Text(0.5, 1.0, 'Evolution of precision &  recall based on du number of features for dataset dgp monotonic ')
```



10

Pour le dataset FMRI, nous obtenons ceci :

```
Pour la target "4"
Text(0.5, 1.0, 'Evolution of precision &  recall based on number of features for dataset fmRI ')
```



3 Etude des algorithmes de prédiction

3.1 Introduction

La seconde phase de notre projet marque une transition cruciale vers l'évaluation des algorithmes de sélection de variables spécifiquement adaptés aux séries temporelles multivariées (MTS). Cette étape est structurée pour fournir une compréhension approfondie des mécanismes de sélection de caractéristiques dans le contexte dynamique des séries temporelles complexes.

3.2 Présentation des algorithmes de prédiction étudiés

Nous avons étudié principalement 3 algorithmes : le LSTM (Long-Short Term Memory, le TFT (Temporal Fusion Transformer) et dans une moindre mesure le DeepAR.

Les réseaux LSTM (Long Short-Term Memory) sont une catégorie de réseaux de neurones récurrents (RNN) conçus pour capturer les schémas de répétition dans le temps. Les LSTM intègrent des mécanismes de porte (portes d'oubli, portes d'entrée et portes de sortie) qui permettent de réguler le flux d'information à travers le réseau, facilitant ainsi la mémorisation de schémas complexes sur de longues séries temporelles. Ils sont largement utilisés dans les applications de séries temporelles pour leur capacité à traiter des données avec des schémas de répétition complexes.

Le Temporal Fusion Transformer est un modèle de prédiction de séries temporelles qui s'appuie sur l'architecture Transformer. Contrairement aux modèles basés sur les réseaux de neurones récurrents, le TFT utilise des mécanismes d'attention pour capturer les relations temporelles entre les différentes variables, ce qui en fait une option puissante pour les séries temporelles multivariées.

DeepAR est un modèle de séries temporelles probabiliste développé par Amazon. Il utilise une architecture de RNN pour capturer les motifs temporels complexes. DeepAR est particulièrement adapté à la prédiction probabiliste, générant des distributions de probabilité pour les prédictions futures.

3.3 Notre algorithme de prédiction principal : Temporal Fusion Transforme(TFT)

3.3.1 Explication de l'algorithme

Le Temporal Fusion Transformer (TFT) utilise l'architecture Transformer pour prédire des séries temporelles multivariées. Le processus commence par encoder chaque série temporelle à l'aide d'embeddings temporels, préservant la structure temporelle. Les mécanismes d'attention sont employés pour mettre en évidence les relations importantes.

La fusion temporelle combine ces informations avec les poids d'attention, créant une représentation globale des séries temporelles.

Le TFT peut générer des prédictions sur plusieurs horizons temporels simultanément, anticipant les valeurs futures pour différentes variables. Cependant, nous allons l'utiliser pour prédire une seule étape future.

L'optimisation utilise la rétropropagation du gradient pour minimiser une fonction de perte (dans notre cas la RMSE), ajustant ainsi les paramètres du modèle.

En résumé, le TFT exploite les mécanismes d'attention de l'architecture Transformer pour capturer les relations temporelles dans les séries temporelles multivariées.

3.3.2 Implémentation

Cette sous-partie plonge dans les aspects pratiques de l'implémentation de l'algorithme, décrivant les étapes cruciales de sa mise en œuvre dans le contexte des séries temporelles multivariées.

Nous avons choisi pour l'implémentation du TFT, la librairie PyTorch Forecasting qui est particulièrement adaptée aux problèmes de prédiction des séries temporelles multivariées.

L'implémentation a été assez difficile car la documentation disponible et les exemples d'utilisation ne sont pas très riches. Cependant, nous avons réussi à faire fonctionner une première version du TFT basique, sans optimisation des hyperparamètres, qui nous permet d'avoir des premiers résultats.

3.3.3 Résultats

Vis-à-vis du diagramme de Gant, de l'avancée de notre projet et de la difficulté à implémenter une première version du TFT, nous n'avons pu réaliser qu'une expérience. Néanmoins cette expérience fournit des résultats concluants quant à la nécessité d'une sélection de variable avant la phase de prédiction.
Description de l'expérience :

- Dataset utilisé : VARSmall, un dataset de 10 variables, toutes les variables peuvent être des cibles, les données du dataset sont linéaires. La RMSE minimal donné par l'équation de simulation est RMSE = 1
- Pour éviter les résultats bruités, nous avons décidé de comparer pour chaque cible de comparer les performances (RMSE et R2 Score) du TFT sur l'ensemble du dataset et sur un dataset comportant seulement les variables proposées par le ground truth. Nous aurons donc à la fin de cette expérience, une moyenne des métriques du TFT entraîné sur tout le dataset et une moyenne des métriques du TFT entraîné sur les variables du ground truth. On s'attend à avoir une RMSE plus proche de 1 et un R2 Score plus grand pour le deuxième cas.

Paramètre de l'expérience :

- Nombre d'époques : maximum 50
- learning rate : 0.03

Résultat de l'expérience : On trouve un R2 Score moyen de 0.46 avec les variables du ground truth comparé à 0.41 sans sélection de variables. On note donc une amélioration des performances sur le R2 Score de 12%

On trouve une RMSE moyenne de 1.064 avec les variables du ground truth comparé à 1.144 sans sélection de variables. On note donc une amélioration des performances sur la RMSE de 7%
Remarques :

- Pour la RMSE, la différence n'est pas vraiment significative, on peut assimiler cette différence à du bruit
- On peut supposer qu'on aurait pu obtenir de meilleurs résultats sur la performance en optimisant les hyperparamètres pour le TFT (deuxième partie du projet)

Conclusion de l'expérience : On a réussi à mettre en évidence la nécessité de mettre en œuvre une sélection de variables au préalable de la prédiction

4 Conclusion

Nous avons donc réussi à implémenter une première version de nos algorithmes, réaliser nos premières expériences et mettre en évidence l'importance de la sélection de variable sur un premier dataset.
Notre objectif dans la fin du projet sera d'explorer les autres datasets et s'adapter à leur specifites. Nous allons également maintenant rentrer plus en détail dans l'optimisation des hyperparametres ce qui nous permettra d'évaluer l'influence de ces derniers.