



Ecole Polytechnique de l'Université de Tours

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

**Projet Recherche & Développement
2021-2022**

Interprétabilité des IA (XAI)

Application à l'analyse de séries temporelles

Tuteur académique
Nicolas RAGOT

Étudiant
Joshua ROUSSEAU (DI5)

4 avril 2022



Liste des intervenants

Nom	Email	Qualité
Joshua ROUSSEAU	joshua.rousseau@etu.univ-tours.fr	Étudiant DI5
Nicolas RAGOT	nicolas.ragot@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Joshua ROUSSEAU surnommé l'auteur.

L'Ecole Polytechnique de l'Université de Tours est représentée par Nicolas RAGOT surnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Joshua ROUSSEAU, *Interprétabilité des IA (XAI): Application à l'analyse de séries temporelles*, Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2021-2022.

```
@mastersthesis{
  author={ROUSSEAU, Joshua},
  title={Interprétabilité des IA (XAI): Application à l'analyse de séries temporelles},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2021-2022}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	v
1 Introduction	1
1 Acteurs, enjeux et contexte	1
Acteurs	1
Enjeux et contexte	1
2 Objectifs	2
3 Hypothèses	2
4 Bases méthodologiques	2
2 Description générale	3
1 Environnement du projet	3
1.1 La méthode	3
1.2 Les données	4
2 Caractéristiques des utilisateurs	4
3 Fonctionnalités du système	4
4 Structure générale du système	4
3 État de l'art	7
1 Interprétabilité des IA	7

	Interprétabilité et Explicabilité.....	7
2	Avancée des travaux sur l'interprétabilité des séries temporelles	8
	Définition des méthodes d'interprétabilité.....	8
3	Détail des méthodes étudiées	9
3.1	Méthodes appliquées aux CNN	9
3.1.1	Méthode 1 : CAM	9
3.1.2	Méthode 2 : Gradient*Input	9
3.1.3	Méthode 3 : ConvTimeNet	10
3.2	Méthodes appliquées aux RNN	10
3.2.1	Méthode 1 : Transformers.....	10
4	Conclusion.....	12
4	Analyse et conception	13
1	Un point sur la méthode	13
2	"PyTorch Forecasting".....	13
5	Mise en oeuvre	16
1	Outils et librairies utilisées.....	16
1.1	Outils	16
	Anaconda + IDE Spyder	16
	Google Colaboratory	17
1.2	Librairies.....	17
2	Éléments d'implémentation, choix techniques	17
2.1	Structuration du programme.....	17
2.2	Étude et pré-traitement des données	18
3	Analyse des résultats, évaluation, qualité	18
6	Bilan et conclusion	20
1	Bilan du semestre 9	20
	Recherche.....	20
	Conception.....	20
	Restant à faire	20
2	Bilan du semestre 10.....	20
	Pré-traitement des données.....	20
	Conception du modèle.....	20
	Analyse des résultats.....	21
	Restant à faire	21
3	Bilan sur la qualité	21
4	Bilan auto-critique et sur la gestion du projet	21

Annexes	22
A Planification, gestion de projet	23
1 Évolution du projet - Semestre 9.....	23
2 Évolution du projet - Semestre 10.....	24
B Description des interfaces	26
1 Interfaces Matérielles/Logicielles.....	26
2 Interfaces Homme/Machine.....	26
3 Interfaces Logiciel/Logiciel.....	26
C Cahier de Spécifications	27
1 Spécifications Fonctionnelles.....	27
1.1 Définition de la fonction 1 : Pré-Traitement des données.....	27
1.2 Définition de la fonction 2 : Configuration et apprentissage du modèle.....	27
1.3 Définition de la fonction 3 : Evaluation des performances du modèle.....	28
1.4 Définition de la fonction 4 : Affichage de l'interprétabilité du modèle.....	28
2 Spécifications non fonctionnelles.....	29
2.1 Contraintes de développement et conception.....	29
Contraintes technologiques.....	29
Contraintes matérielles.....	29
Logiciels et bibliothèques à utiliser.....	29
Environnement.....	29
Protocoles de communication.....	29
2.2 Contraintes de fonctionnement et d'exploitation.....	30
2.2.1 Performances.....	30
2.2.2 Capacités.....	30
2.2.3 Contrôlabilité.....	30
2.2.4 Sécurité.....	30
2.2.5 Maintenance et évolution du système.....	30
D Cahier du développeur	31
1 Introduction.....	31
2 Descriptions détaillées de données exploitées.....	31
3 Descriptions détaillées des classes, modules, réalisations.....	31
E Document d'installation	32
1 Installation de Google Colab.....	32
2 Installation de Anaconda + Spyder.....	33
F Document d'utilisation	34

G	Documents supplémentaires produits	35
1	Résumé de l'étude du survey : "Explainable Artificial Intelligence (XAI) on Time Series"	35
H	Webographie	51
I	Bibliographie	52
J	Acronymes	53

Table des figures

1	Introduction	
1.1	Exemple de modèle en cascade.....	2
2	Description générale	
2.1	Diagramme prévisionnel des cas d'utilisation de l'application	6
3	État de l'art	
3.1	Utilisation de la méthode « Gradient*Input » pour identifier la contribution des données brutes d'entrée lors de la classification des séries temporelles.....	10
3.2	Architecture des Transformers tirée du papier "Attention is all you need" [3]	11
4	Analyse et conception	
4.1	Attention relevée sur l'ensemble des variables étudiées	14
4.2	Importance des variables statiques étudiées (en pourcentage)	14
4.3	Importance relevée sur les variables utilisées dans la partie Encodeur du transformer	15
4.4	Importance relevée sur les variables utilisées dans la partie Décodeur du transformer	15
5	Mise en oeuvre	
5.1	Prédictions obtenues en moyenne pour chaque valeur présente dans le jeu de validation.....	19
A	Planification, gestion de projet	
A.1	Diagramme de Gantt du travail effectué au semestre 9	23
A.2	Diagramme de Gantt prévisionnel du semestre 10	24

A.3	Diagramme de Gantt au jour 1 du semestre 10.....	24
A.4	Diagramme de Gantt final du semestre 10.....	25

1

Introduction

1 Acteurs, enjeux et contexte

Acteurs

Les acteurs de ce projet sont Nicolas Ragot, enseignant chercheur travaillant dans l'équipe RFAI, qui est l'encadrant de ce projet, et enfin Joshua Rousseau, élève ingénieur en DI5, qui a réalisé ce projet.

Enjeux et contexte

Les séries temporelles sont par nature omniprésentes dans la vie de tous les jours, car elles peuvent représenter n'importe quelle variable variant au cours du temps. Elles le sont d'autant plus notamment dans l'industrie ou dans des domaines critiques tels que la santé, l'automobile, l'environnement, ...

Ainsi, l'utilisation de méthodes d'apprentissage automatique (machine learning) effectué sur ces séries temporelles est devenu pour notre société actuelle un enjeu majeur, étant donné les nombreuses utilisations possibles permettant de nous faciliter grandement la réalisation de tâches, considérées comme étant complexes auparavant, et ce dans de nombreux domaines.

Bien que d'énormes progrès scientifiques aient été faits en apprentissage automatique sur les séries temporelles, on a rapidement fait face à un inconvénient majeur dans l'utilisation de ces méthodes : leur manque d'interprétabilité. Il s'agit d'un inconvénient majeur car plusieurs applications de ces séries temporelles se retrouvent par exemple dans le domaine médical, ou dans la conduite de voitures autonomes, où il y a justement un grand besoin d'interprétabilité.

C'est ainsi qu'un nouveau champ d'investigation a vu le jour pour faire face à ce problème : l'XAI (eXplainable Artificial Intelligence). Son objectif est de proposer des méthodes permettant d'interpréter ou d'expliquer le fonctionnement des modèles de machine learning utilisés et de conserver une part de décision humaine dans l'utilisation de ces modèles ou de systèmes étant entièrement automatisés.

Tous ces aspects seront développés dans la section consacrée à l'état de l'art (Chapitre 3).

2 Objectifs

Ce PRD se base sur les enjeux et le contexte évoqués précédemment, et est divisé en plusieurs objectifs. Dans un premier temps un état de l'art sera réalisé afin de répertorier les différentes méthodes d'interprétabilité existantes sur l'apprentissage des séries temporelles. Ensuite, une de ces méthodes sera choisie afin d'être appliquée dans un cas concret pour vérifier son bon fonctionnement et sa capacité à fournir une explication. Enfin, si les deux objectifs précédents sont remplis, un dernier objectif serait d'utiliser une autre méthode que celle choisie afin de comparer les résultats obtenus avec les deux méthodes.

3 Hypothèses

Après avoir effectué un état de l'art sur les différentes méthodes d'interprétabilité existantes (voir le chapitre 3), il a été décidé de prendre la méthode des « Transformers », du fait de la grande popularité qu'elle suscite sur les applications de séries temporelles, mais également sur les résultats prometteurs qu'elle fournit. Dans le cas éventuel où le choix de cette solution produirait des résultats peu satisfaisants, il sera alors possible de se rabattre sur une des autres méthodes évoquées dans l'état de l'art.

4 Bases méthodologiques

La réalisation de ce projet se déroulera sur deux phases :

- Une phase d'analyse de l'existant et des besoins (état de l'art, ...), et de conception de l'application sur tout le long du semestre 9
- Une phase de développement / mise en œuvre de cette application au semestre 10

Étant donné que nous travaillons ici sur un projet orienté recherche, la gestion de ce projet s'organisera sous un modèle en cascade. Ce modèle semble approprié pour ce type de projet, et suivra les deux parties présentées ci-dessus.

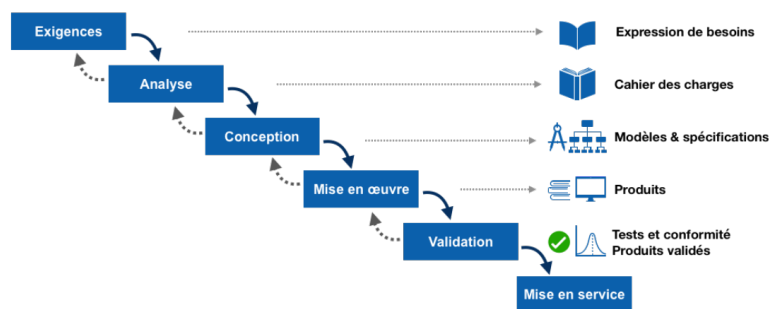


Figure 1.1 – Exemple de modèle en cascade

La planification de ce projet sera effectuée sur un diagramme de Gantt créé sous l'application GanttProject.

Quant au développement de l'application, il se fera sous le langage de programmation Python.

2

Description générale

1 Environnement du projet

Ce PRD fait suite à deux précédents PRD ayant été réalisés sur le sujet de l'interprétabilité des IA en général. Cependant, là où mes prédécesseurs ont plutôt travaillé sur de l'analyse des images et des vidéos, le travail de ce projet portera surtout sur de l'analyse de séries temporelles. Il s'agit donc ici d'une nouvelle problématique.

Le but de ce PRD est de concevoir et de développer une application utilisant une des méthodes existantes en matière d'interprétabilité des IA sur les séries temporelles, et de l'appliquer sur des données afin de pouvoir obtenir des résultats prédictifs sur ces données et en tirer des conclusions sur l'interprétabilité de cette méthode.

On dispose donc pour ce projet d'une étude[2] qui répertorie l'ensemble des méthodes d'interprétabilité existantes pour les modèles de machine learning sur des séries temporelles. En fonction de cela, nous choisirons une méthode qui nous paraît être intéressante à étudier, et qui servira donc au développement du programme.

Le développement de cette application dépendra donc de deux aspects :

- La méthode choisie en conclusion de l'état de l'art
- Le jeu de données / la base de données choisie pour tester notre méthode

1.1 La méthode

La méthode choisie pour développer notre application est celle dite des « Transformers ». Il s'agit d'une méthode assez récente (datant de 2017) utilisée principalement pour du traitement du langage naturel, mais s'avérant également être très efficace sur les séries temporelles.

Comme pour les RNN (réseaux de neurones récurrents), elle est basée sur des mécanismes d'attention qui vont donc analyser des séquences de données et attribuer des états en fonction de la significativité de ces séquences. La différence majeure avec les RNN est qu'il n'y a pas d'ordre pour passer en revue les données, par conséquent le temps alloué à l'entraînement sur ces données sera bien plus réduit que sur un RNN classique. Cette technologie a rendu obsolète les modèles LSTM (Long Short Term Memory), auparavant très utilisés dans ce domaine.

On pourra alors prétendre par le biais de cette méthode à obtenir une classification des variables significatives se trouvant dans un jeu de données fourni, et donc par conséquent à une interprétabilité du modèle étudié.

1.2 Les données

— **Données en entrée :**

Les données qui seront utilisées dans ce programme proviendront principalement de jeux de données et/ou de bases de données fournis au préalable par l'encadrant. Ces jeux de données seront principalement représentés par des tableaux individus / variables. Cela peut être pour prendre un exemple une liste de personnes caractérisées par des attributs comme leur taille, leur âge, ...

— **Résultats attendus en sortie :**

En sortie de notre programme, les résultats fournis seront une classification des variables pertinentes des données fournies, ainsi que l'interprétabilité du modèle étudié. L'ensemble de ces résultats seront visualisables sous la forme de graphiques.

2 Caractéristiques des utilisateurs

L'utilisation de cette application sera principalement réservée aux chercheurs en informatique travaillant dans ce domaine précis. Étant un projet très orienté sur de la recherche, l'application résultante de celui-ci ne sera pas destinée à une utilisation tout public.

3 Fonctionnalités du système

L'application doit pouvoir être en mesure de prendre en entrée des données (fournies en ligne de commande) choisies par l'utilisateur, d'appliquer le modèle choisi durant l'état de l'art sur ces données afin de pouvoir générer en sortie une interprétabilité de l'algorithme. Cette interprétabilité prendra la forme de graphiques montrant une classification des variables / caractéristiques significatives parmi les données fournies en entrée. Ces graphiques pourront être ensuite enregistrés dans un dossier spécifié ou non par l'utilisateur, pour pouvoir stocker les résultats obtenus et ainsi les consulter.

4 Structure générale du système

L'application disposera de plusieurs fonctions qui interagiront les unes avec les autres afin de produire le résultat espéré.

On dispose pour cela d'une librairie, découverte grâce à l'état de l'art effectué sur les méthodes d'interprétabilité des IA sur les séries temporelles : « pytorch forecasting ». Cette librairie fournit un modèle déjà entraîné sur des séries temporelles et fournit un certain nombre de méthodes / fonctions permettant de faciliter les prévisions de séries temporelles.

D'après ce qui a été évoqué précédemment dans ce rapport, on est en mesure de pouvoir définir plusieurs fonctions phares de cette application.

- **Une fonction sur le pré-traitement des données**

Cette première fonction va être en mesure de récupérer les données rentrées dans le programme pour les transformer dans un format qui nous permettra, par la suite, de les rentrer dans le modèle sans erreurs.

- **Une fonction pour la configuration et l'apprentissage du modèle d'interprétabilité**

Une fois les données formatées cette fonction permettra de les intégrer dans le modèle des transformers fourni par la librairie "pytorch-forecasting".

Nous avons pu tester (voir partie Analyse et Conception (Chapitre 4)) une première fois le modèle fourni par la librairie "pytorch-forecasting". Cependant il était déjà entraîné de base sur des données fournies par cette même librairie. L'ajout de cette fonction va donc nous permettre également de pouvoir réapprendre le modèle afin de pouvoir le tester sur des données externes (fournies par le laboratoire).

- **Une fonction d'évaluation des performances du modèle**

Cette fonction permettra, une fois le modèle exécuté sur les données, de recenser l'ensemble des résultats obtenus en sortie du modèle, que cela soit les prédictions obtenues en général ou bien les prédictions obtenues en fonction des variables pertinentes, puis de les afficher sous forme de graphiques pour que l'utilisateur puisse interpréter ces résultats.

- **Une fonction d'affichage de l'explicabilité du modèle**

Cette fonction se concentrera sur l'explicabilité du modèle et fournira, comme pour la fonction précédente, des graphiques montrant la classification des variables pertinentes et une explication du modèle pour permettre à l'utilisateur d'interpréter ces résultats.

Tout cela est représenté dans le diagramme des cas d'utilisation de l'application suivant :

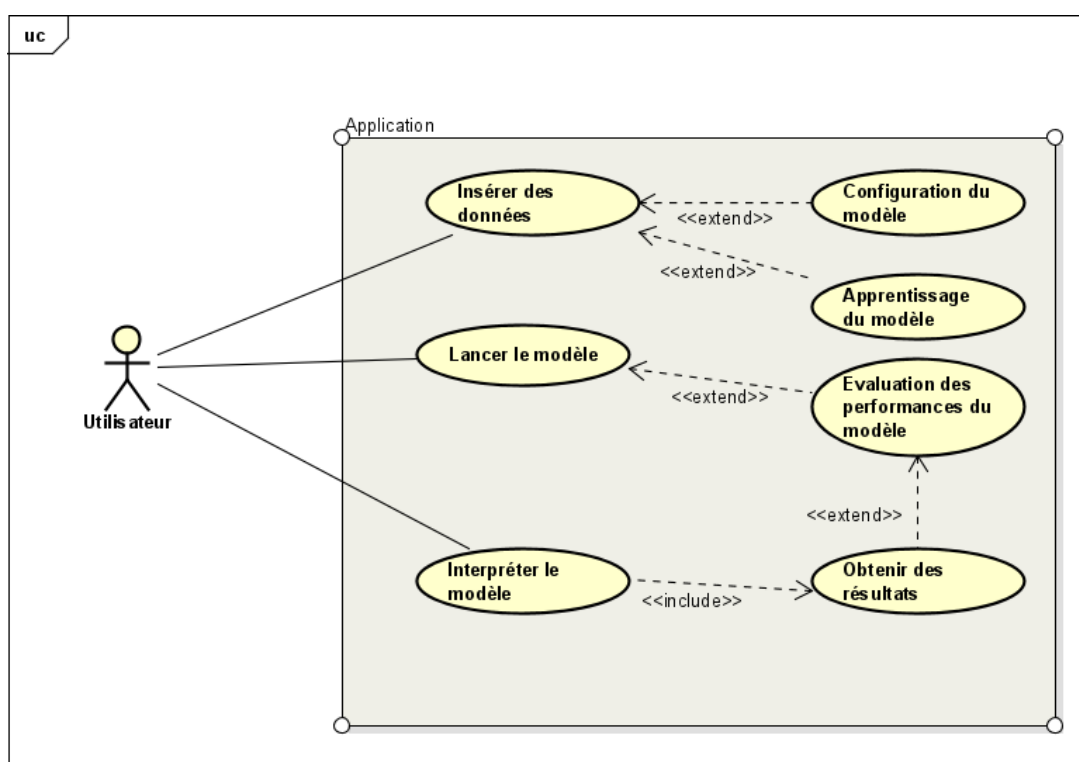


Figure 2.1 – Diagramme prévisionnel des cas d'utilisation de l'application

3

État de l'art

Nous allons maintenant voir un peu plus en détail ce qu'est l'interprétabilité des IA dans les séries temporelles et quelles sont les applications mises en œuvre aujourd'hui.

1 Interprétabilité des IA

Les informations présentes dans cette première partie ont été principalement tirées de l'étude "Explainable Artificial Intelligence on Time Series Data" [2].

Comme évoqué dans le contexte et les enjeux du projet, l'apprentissage automatique appliqué aux séries temporelles revêt d'une importance particulière, car il a de nombreuses applications potentielles dans divers domaines.

Plusieurs tâches peuvent être effectuées grâce à l'apprentissage automatique des séries temporelles, les principales sont plutôt basées sur de la classification, la prévision ou bien le regroupement de ces séries. Pour effectuer toutes ces tâches, les algorithmes de deep learning sont depuis plusieurs années des algorithmes de pointe pour les problèmes avec des séries temporelles en entrée.

Cependant l'un des inconvénients majeurs de ces algorithmes est le manque d'interprétabilité, et ce à cause de leur grande complexité. L'Explainable Artificial Intelligence (XAI) est donc une grande préoccupation pour les séries temporelles, car la plupart des algorithmes utilisés ne sont pas interprétables. De plus, plusieurs applications des séries temporelles dans le monde réel, dans des domaines tels que le médical ou la conduite autonome, sont d'une importance fondamentale et ont donc besoin de cette interprétabilité.

Interprétabilité et Explicabilité

Deux termes sont évoqués fréquemment lorsque l'on parle de machine learning : "Interprétabilité" et "Explicabilité". L'XAI a permis de mettre au point des méthodes qui, une fois appliquées sur des modèles de deep learning, permettent de fournir soit une explicabilité, soit une interprétabilité sur le fonctionnement de l'algorithme. La différence entre ces deux concepts se situe dans le niveau de détails qu'ils génèrent.

Une explicabilité consiste à fournir des éléments permettant de mettre en relation les valeurs prises par certaines variables (les caractéristiques) et leurs conséquences sur, par exemple, la prévision d'un score, et ainsi sur la décision.

A contrario une interprétabilité va fournir des éléments liés directement aux caractéristiques ou variables qui participent le plus à la décision, voire même d'en quantifier leur importance.

La différence entre ces deux termes est assez subtile, mais on pourrait la résumer ainsi : l'interprétabilité consiste à être capable de discerner la mécanique derrière l'algorithme sans forcément savoir pourquoi, alors que l'explicabilité va expliquer comment ça fonctionne.

Dans le cadre de notre projet, on se focalisera principalement sur la recherche d'une interprétabilité du modèle que l'on va étudier.

2 Avancée des travaux sur l'interprétabilité des séries temporelles

De grandes avancées en matière d'interprétabilité ont été faites dans les domaines de la vision par ordinateur (par extension des images et des vidéos), ainsi que dans le traitement du langage naturel (**Natural Language Processing (NLP)**). Ce n'est pas le cas malheureusement des séries temporelles, et en particulier pour une raison.

Lorsqu'un être humain regarde une photo ou lit un texte il peut comprendre instinctivement, et de manière intuitive, les informations qui lui sont montrées. Cela n'est pas le cas pour les séries temporelles, car bien qu'elles soient omniprésentes dans la nature, nous n'avons pas l'habitude de représenter ces données temporelles directement sous la forme d'un signal qui varie au cours du temps.

Par conséquent, la plupart des méthodes disponibles aujourd'hui pour l'interprétabilité des modèles appliqués sur les séries temporelles sont à la base issues de méthodes développées pour la vision par ordinateur et le traitement du langage naturel principalement.

Nous allons maintenant introduire divers termes pour définir ces méthodes d'interprétabilité.

Définition des méthodes d'interprétabilité

Il existe deux types de méthodes pour interpréter un modèle de machine learning.

La première catégorie de méthodes consiste à l'intégrer avant le lancement de l'algorithme, pour qu'elle puisse se charger ensuite d'apprendre le fonctionnement de celui-ci afin de fournir directement en sortie l'interprétabilité. En d'autres termes l'interprétabilité est directement incorporée dans le modèle de machine learning : ce type de méthode est dit "Ante-hoc", du fait qu'on l'applique avant le lancement de l'algorithme.

La seconde catégorie de méthodes est directement l'opposée de cette première. La méthode va être utilisée dès que l'algorithme de machine learning a fini d'être exécuté et va traiter les résultats obtenus (classification des variables significatives présentes dans les données notamment) pour fournir une interprétabilité du modèle. Ce type de méthode est dit "Post-Hoc", du fait qu'on l'applique une fois le modèle exécuté.

Les méthodes post-hoc existantes pour le traitement des séries temporelles présentées dans l'étude[2] sont toutes adaptées uniquement sur des réseaux de neurones convolutifs (CNN).

Enfin il existe parmi les méthodes post-hoc deux autres catégories :

- Les méthodes dites "Model-Agnostic"
Ces méthodes ont la spécificité de pouvoir être appliquées sur tous les types de modèles existants.
- Les méthodes dites "Model-Specific"
Ces méthodes sont applicables uniquement à un seul type de modèle.
Maintenant que nous avons une connaissance des différents termes liés aux méthodes d'interprétabilité en IA, voyons les différentes méthodes existantes sur l'application des séries temporelles.

3 Détail des méthodes étudiées

Cette dernière partie présente en détail les diverses méthodes présentes dans l'étude [2]. Cette liste nous permettra de définir sur quelle méthode nous nous focaliserons pour la phase de développement du projet. Un diaporama répertoriant ces méthodes et résumant de manière plus globale l'étude est également disponible en fin de rapport pour plus de détails.

3.1 Méthodes appliquées aux CNN

3.1.1 Méthode 1 : CAM

La méthode CAM (Class Activation Mapping) est une méthode post-hoc créée à la base pour le traitement des images, mais elle a pu montrer des résultats dans le traitement des séries temporelles.

Le principe de cette méthode consiste à mettre en avant certaines sous-séquences de données qu'elle juge significatives sur l'ensemble du jeu de données fournies en entrée du modèle. Elle est très utile pour comprendre comment un réseau neuronal convolutif arrive à produire une décision de classification.

Plus précisément elle s'appuie sur la présence de couches de moyennes globales de mise en commun à l'extrémité des couches convolutives. La couche de mise en commun globale prend N canaux et renvoie ses valeurs moyennes. Les canaux avec des activations plus élevées ont ainsi des signaux plus élevés.

Il s'agit cependant d'une méthode assez complexe à mettre en oeuvre car le modèle utilisé doit avoir une architecture spécifique à ce type de cas : il doit disposer d'une couche de mise en commune de moyennes globales située après les couches convolutives.

3.1.2 Méthode 2 : Gradient*Input

La méthode du Gradient*Input est ce que l'on appelle une méthode d'attribution. L'attribution est calculée en prenant les dérivées partielles de la sortie par rapport à l'entrée et en les multipliant par l'entrée elle-même.

Plus spécifiquement elle va interpréter les décisions des réseaux de neurones en extrayant les nœuds hautement actifs dans un canal et en visualisant les sous-séquences des données d'entrée qui contribuent aux nœuds hautement activés. Pour cela on utilise des CPHAP (Clustered Pattern of Highly Activated Period) qui sont un axe moyen dans le temps d'un cluster, soit une liste de séquences temporelles qui activent des nœuds.

Un exemple peut être représenté avec l'image suivante :

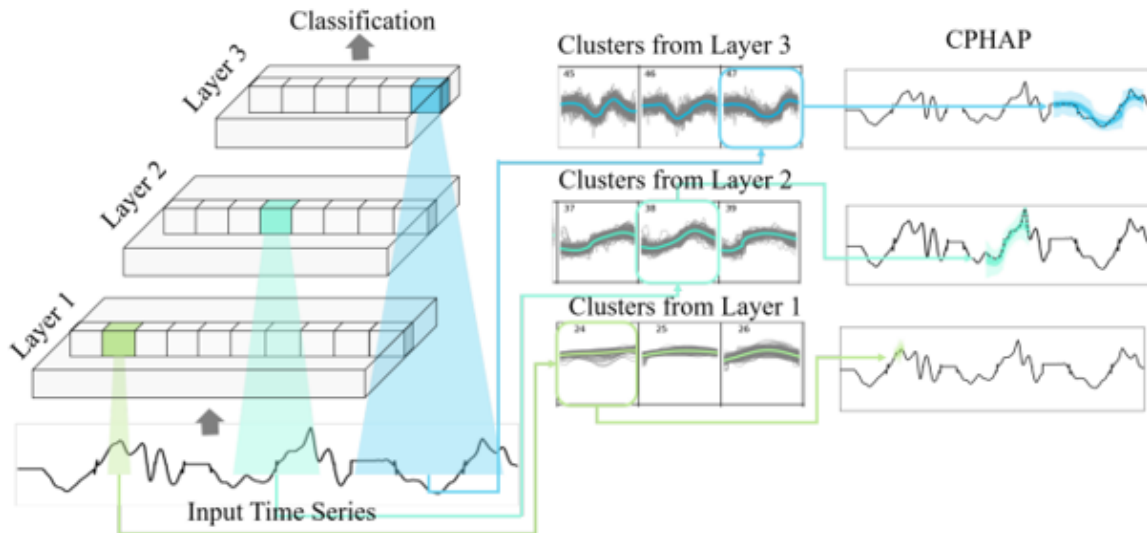


Figure 3.1 – Utilisation de la méthode « Gradient*Input » pour identifier la contribution des données brutes d'entrée lors de la classification des séries temporelles

On visualise les sous-séquences d'entrée qui contribuent aux nœuds hautement activés. Enfin chaque sous-séquence extraite est affectée à un groupe de motifs similaires.

3.1.3 Méthode 3 : ConvTimeNet

La méthode ConvTimeNet (CTN) utilise la méthode de sensibilité à l'occlusion qui va occulter des parties de la série temporelle et calculer la différence de probabilité pour la classe prédite. Ainsi plus la différence sera élevée, plus l'importance de cette partie sera importante dans la classification finale des variables.

Plus précisément ce type de méthode est basé sur de la perturbation. Elle calcule directement la contribution des entités d'entrée en les supprimant, en les masquant ou en les modifiant, en exécutant une passe avant sur la nouvelle entrée et en mesurant la différence avec l'entrée d'origine. Plus la différence est élevée, plus la contribution de la sous-séquence d'entrée qui a été modifiée est élevée.

3.2 Méthodes appliquées aux RNN

3.2.1 Méthode 1 : Transformers

Cette méthode est l'une des plus récentes à ce jour (2017), cependant elle a suscité beaucoup d'intérêt de par les résultats prometteurs qu'elle a pu produire dans le domaine du traitement du langage naturel. A tel point que les méthodes utilisées avant dans ce domaine (des méthodes comme le LSTM (Long Short Term Memory) par exemple) sont presque devenues obsolètes ou n'ont plus grandement d'intérêt d'être utilisées.

L'architecture de cette méthode est basée sur du Sequence-to-Sequence (ou Seq2Seq), ce qui peut se résumer par une architecture de type encodeur-décodeur.

Voici une représentation de cette architecture, tirée du célèbre papier "Attention is all you need" [3] :

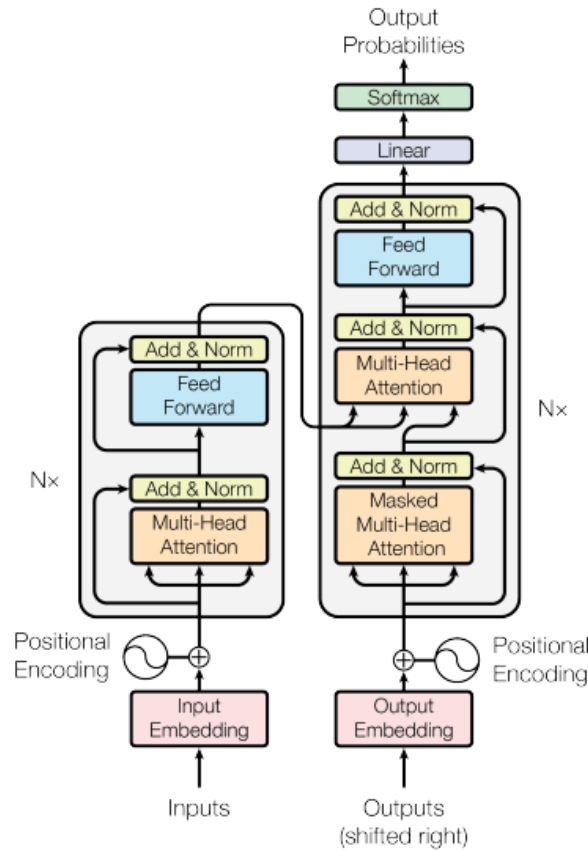


Figure 3.2 – Architecture des Transformers tirée du papier "Attention is all you need" [3]

Basiquement ce type d'architecture va transformer une séquence donnée d'éléments en une autre séquence d'éléments, ce qui en fait une très bonne architecture pour la traduction d'une séquence de mots d'une langue vers une autre.

Le fonctionnement de cette architecture est basé sur ce que l'on appelle les mécanismes d'attention. L'encodeur va prendre en compte toutes ses entrées et décider lesquelles sont les plus importantes en leur attribuant un poids à chacune d'elles. Dans le cadre de cette architecture il s'agit précisément des "Multi-Head Attention". Cette spécificité du transformer va lui permettre de se focaliser sur plusieurs informations en même temps, plutôt que sur information à la fois comme on pourrait le voir sur des mécanismes d'attention plus "basiques" comme les mécanismes de "Self Attention".

Jusqu'à présent les réseaux de neurones récurrents (RNN) étaient le meilleur moyen de capturer des dépendances opportunes dans les séquences d'entrée. Cependant cette nouvelle architecture a prouvé que l'utilisation unique des mécanismes d'attention permettait de fournir de meilleurs résultats en traitement du langage naturel, mais également dans d'autres tâches comme les séries temporelles. Le principal problème avec les RNN réside dans leur incapacité à fournir une parallélisation lors du traitement. Le traitement d'un RNN est séquentiel, c'est-à-dire que nous ne pouvons pas calculer la valeur du prochain pas de temps à moins d'avoir accès à la sortie du précédent. Cela rend les approches basées sur les RNN lentes. Or dans cette nouvelle architecture il n'y a pas d'ordre de passage en revue des séquences de données, ce qui réduit drastiquement le temps d'entraînement du modèle, et fournit par la même occasion de meilleurs résultats qu'avec un RNN classique.

4 Conclusion

Pour conclure sur cet état de l'art, il existe aujourd'hui encore très peu de méthodes fiables pour interpréter un modèle de traitement de séries temporelles. Mais cela risque de changer, car de nouvelles méthodes sont mises au point de plus en plus rapidement aujourd'hui, et avec l'avènement des "transformers", il est tout à fait possible que de nouvelles méthodes puissent voir le jour pour le domaine des séries temporelles.

Pour la suite du projet, et en concertation avec mon encadrant, nous avons décidé de nous focaliser sur la méthode des transformers, car fournissant des résultats prometteurs même dans des tâches liées aux séries temporelles.

4

Analyse et conception

Cette partie va plus se focaliser sur la méthode choisie en conclusion de l'état de l'art (les "Transformers") et va détailler le déroulement de l'implémentation de cette méthode dans un futur programme qui sera la conclusion de la partie développement de ce projet.

1 Un point sur la méthode

L'objectif de ce projet n'est pas de rentrer trop dans les détails sur le fonctionnement de cette méthode, mais plutôt de pouvoir implémenter une méthode d'interprétation qui puisse s'adapter à un modèle, afin de montrer simplement son bon fonctionnement.

Le choix de cette méthode s'est donc effectué sur l'effet de mode qu'elle a connu lors de son dévoilement, et même si elle a été conçue à la base pour traiter des tâches liées au traitement du langage naturel, certaines applications sur des séries temporelles ont révélé des résultats assez intéressants.

2 "PyTorch Forecasting"

Pour pouvoir développer notre application il a fallu tout d'abord commencer par rechercher des modèles existants afin que, dans l'idéal, on puisse se servir de ce modèle comme une base à notre projet afin de pouvoir implémenter notre méthode et la tester dessus.

Grâce à des recherches complémentaires menées dans cette direction, nous avons pu trouver l'existence d'un package python open source nommé "Pytorch Forecasting". Ce package a été développé dans une optique de faciliter la prévision de séries temporelles en fournissant des classes de modèles adaptées ainsi qu'un large ensemble de classes visant à simplifier beaucoup d'étapes qui seraient vues auparavant comme fastidieuses (notamment au niveau du pré-traitement du jeu de données fourni).

Le modèle sur lequel nous allons travailler est donc contenu dans une classe nommée "TemporalFusionTransformer" fournie par ce package. Il s'agit d'une application directe provenant de l'article [1], pour plus de détails.

En fin de semestre 9 nous avons eu l'occasion de tester le fonctionnement de ce modèle à partir d'un tutoriel fourni par le package [WWW] qui présente, sous la forme d'un exemple,

toutes les étapes nécessaires à l'interprétation de la méthode des transformers, et ce, du pré-traitement des données fournies en entrée du programme à l'interprétabilité fournie par le modèle "TemporalFusionTransformer". Ce tutoriel nous aura permis de parfaitement configurer notre machine personnelle afin que, lorsque le développement de l'application commencera, tout l'environnement du projet soit fonctionnel.

Cet exemple s'est basé sur l'étude d'un jeu de données d'un peu plus de 20 000 échantillons sur des ventes de bières d'une entreprise [WWW2], l'objectif étant de pouvoir effectuer une prévision sur ces ventes. En plus de ces données on a également à notre disposition des informations sur le prix de vente, l'emplacement des agences vendant ces produits, les jours spéciaux tels que les jours fériés et le volume vendu dans l'ensemble de l'industrie.

Si l'on regarde les résultats possibles à obtenir sur ces informations, en terme d'interprétabilité, on peut prétendre à ce type de résultats :

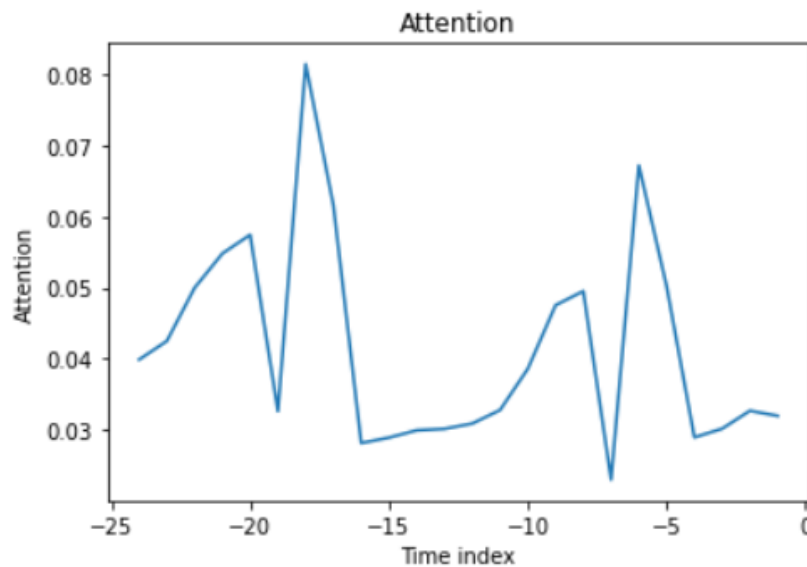


Figure 4.1 – Attention relevée sur l'ensemble des variables étudiées

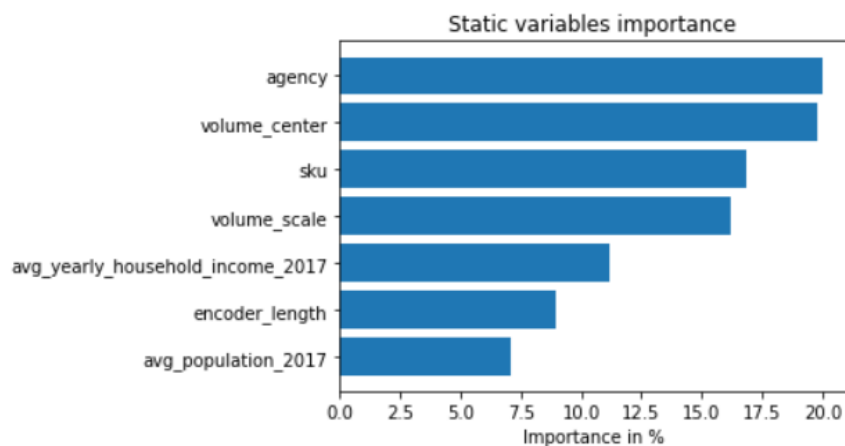


Figure 4.2 – Importance des variables statiques étudiées (en pourcentage)

Ces résultats sont obtenables grâce au modèle "TemporalFusionTransformer", qui contient des fonctions permettant de générer ces graphiques à partir des prédictions faites par celui-ci. On a ainsi accès à l'importance accordée aux différentes variables étudiées, et cela nous donne une idée globale de ce que le modèle a pu prioriser lors de son fonctionnement.

Les caractéristiques de volume observées dans le passé figurent parmi les principaux prédicteurs

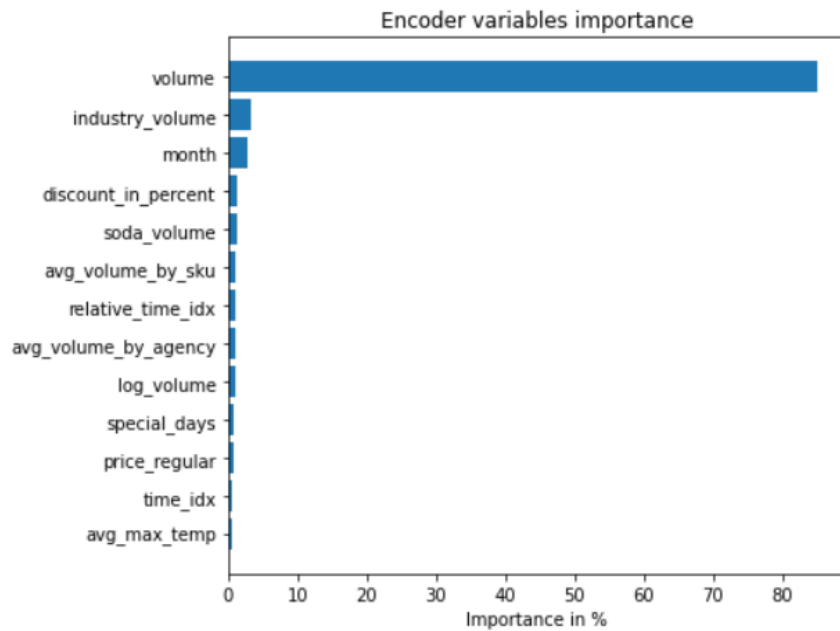


Figure 4.3 – Importance relevée sur les variables utilisées dans la partie Encodeur du transformer

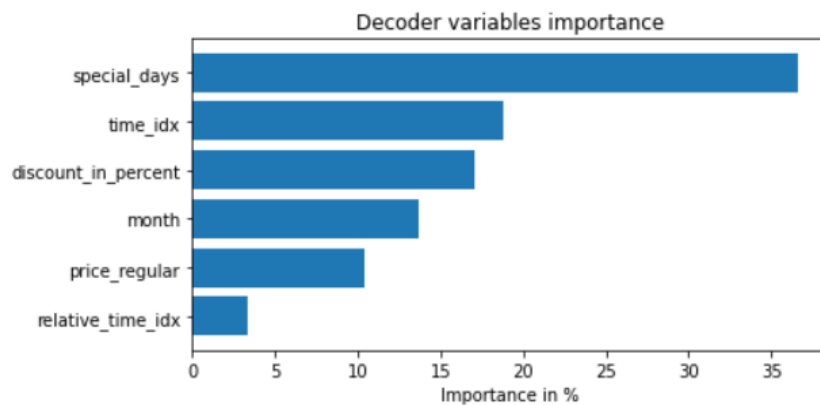


Figure 4.4 – Importance relevée sur les variables utilisées dans la partie Décodeur du transformer

de l'encodeur. Tandis que dans la partie décodeur, ce sont les variables liées au prix qui figurent parmi les principaux prédicteurs.

Nous verrons plus en détail dans la suite du projet si cette approche que nous avons choisie portera ses fruits sur un jeu de données différent.

5

Mise en oeuvre

1 Outils et librairies utilisées

Le développement fut prévu à la base pour se dérouler sur deux environnements différents, un sur une machine locale avec un environnement Anaconda comprenant l'IDE Spyder, ainsi que sur un environnement hébergé en ligne sur Google Colaboratory (abrégé en Google Colab). Bien que le déroulé du développement se soit consacré majoritairement sur Google Colab, il existe une version du programme fournie pour être exécuté en local sur Spyder.

1.1 Outils

Nous allons évoquer rapidement les deux outils utilisés pour voir leurs avantages et inconvénients et à quelle situation privilégier l'un plutôt que l'autre.

Anaconda + IDE Spyder

Anaconda est une des plateformes de distribution Python les plus populaires dans le monde, elle intègre plusieurs applications permettant de travailler notamment sur de la data science ou du machine learning. Parmi ces applications on retrouve l'IDE Spyder, qui permet d'exécuter directement du code Python par exemple.

Parmi les avantages liés à ce choix d'environnement, on retrouve la possibilité de disposer en local du code et de l'exécuter sans avoir une connexion à internet. On dispose également de fonctionnalités intéressantes comme l'accès à un débogueur pour pouvoir poser des points d'arrêt où l'on souhaite pour traiter l'exécution d'une ligne de code en détails.

L'inconvénient majeur de ce choix d'environnement est qu'il peut nécessiter (et plus particulièrement dans notre cas) de ressources matérielles plus ou moins importantes en fonction du code exécuté. Or, dans notre cas, on est amenés ici à développer un modèle de prédiction et d'interprétabilité sur des séries temporelles, ce qui engendre une grande quantité de calculs à effectuer. Malheureusement, si votre machine locale n'est pas suffisamment puissante (plus de 8 Go de RAM au processeur), le temps d'exécution du programme va s'allonger et les résultats potentiels devenir presque faussés. L'utilisation de cet environnement est alors réservée aux utilisateurs disposant d'une machine locale suffisamment performante.

Google Colaboratory

Google Colab est un environnement d'exécution à distance permettant d'exécuter du code en python, il est très adapté notamment à l'analyse de données et au machine learning.

Il a l'avantage de pouvoir être disponible à distance, et ainsi de pouvoir disposer du code sur n'importe quelle machine. La seule condition pour y accéder est d'avoir un compte Google et une connexion internet. Étant un environnement hébergé, il ne nécessite pas de disposer d'une machine personnelle performante, étant donné que tous les calculs seront effectués à distance. On pourra alors être sûrs d'obtenir des résultats plus prometteurs qu'avec une machine locale plus ou moins performante sur l'environnement décrit précédemment.

Le code sur Google Colab se présente sous la forme de Notebooks (dans la même forme que Jupyter par exemple) où l'on peut créer des cellules pour y ajouter du code ou bien du texte pour argumenter par exemple sur ce que fait le code. Il s'agit d'un format très pratique pour la création de programmes d'exemples ou de tutoriels, cependant ce format n'est pas très adapté pour développer une application complète. En effet, comme le code est séparé en cellules, il est moins évident de créer des fonctions ou des classes sans devoir toutes les créer dans des cellules séparées.

1.2 Librairies

Un certain nombre de librairies ont été requises au développement de ce programme. Parmi elles nous avons Pandas pour le pré-traitement du jeu de données, Pytorch et Pytorch Lightning pour la création de l'entraîneur de notre modèle.

La librairie la plus importante utilisée ici est Pytorch Forecasting. Il s'agit ici de la librairie essentielle pour la réalisation de notre programme. Car en effet, elle nous fournit le modèle de Transformers sur lequel appliquer nos données et dispose de méthodes, liées au modèle, qui nous permettent de simplifier grandement la génération de prédictions ou bien nous fournir une interprétabilité du fonctionnement du modèle.

En fonction de l'environnement choisi (en local ou via Colab), les étapes d'installation de ces librairies sont totalement différentes. Pour avoir plus de détails sur l'installation de ces librairies, ou même plus globalement des environnements spécifiés ci-dessus, veuillez vous référer à cette partie du rapport (Chapitre E).

2 Éléments d'implémentation, choix techniques

2.1 Structuration du programme

Le programme se décompose en 4 grandes parties, chaque partie (fonction) représentant une cellule (sur Google Colab uniquement) :

- Data-preprocessing : Fonction permettant de préparer les données pour la prédiction,
- Model-training : Fonction utilisée pour créer le modèle de Transformers ainsi que l'entraîneur du modèle, et l'entraîner sur les données préproduites via la fonction précédente,
- Evaluate : Fonction utilisée pour évaluer le modèle et générer des prédictions
- Interpret : Fonction utilisée pour générer une interprétation du fonctionnement du modèle

2.2 Étude et pré-traitement des données

Nous avons choisi, pour le développement de notre programme, de travailler sur un jeu de données provenant de la plateforme en ligne Kaggle.

Il s'agit d'une plateforme créée à la base pour l'organisation de compétitions dans le domaine de la science des données, où des entreprises proposent des problèmes à résoudre, et la communauté les résout contre un prix réservés aux plus performants d'entre eux. Ce site propose également par conséquent une grande base de datasets (jeu de données) à étudier, et c'est justement l'un d'eux qui nous a intéressé.

Ce dataset [WWW1] comporte des relevés météorologiques qui ont été effectués toutes les heures pendant environ 5 ans sur 30 villes des États-Unis et du Canada (de 2012 à 2017). Parmi ces relevés on retrouve des données liées à la température, l'humidité, la pression, la direction et la vitesse du vent, etc. Afin de tester ces données sur notre modèle, il a été choisi de se focaliser dans un premier temps sur les données liées à la température, afin que cela serve de base à de futurs ajouts (par exemple montrer l'impact que peut avoir l'humidité ou la pression sur la température d'une ville). La ville que nous avons choisie pour étude est la ville de Portland. Ce choix n'a eu aucune importance sur notre étude, l'idée ici étant simplement de pouvoir disposer d'un jeu de données sur lequel travailler.

La ville de Portland dispose, au total, de plus de 40 000 relevés de températures, ce qui en fait un jeu de données assez conséquent. Pour éviter de se lancer dans une complexité d'étude trop élevée nous avons choisi de réduire drastiquement ce nombre de relevés, en se focalisant uniquement sur ceux effectués durant les mois de Juillet et Août de chaque année étudiée, soit un total d'environ 7400 relevés. Ainsi, nous avons gardé uniquement les relevés allant du 1er Juillet au 31 Août de chaque année. Enfin, les températures qui furent fournies étaient en degré Kelvin (soit une différence de 273.15 pour une température en degré Celcius). Dans notre pré-traitement de données nous avons choisi de rajouter une colonne pour préciser les températures en degré Celcius pour permettre une meilleure lisibilité pour la suite.

L'ensemble des choix et modifications montrées ici sont présents dans le code et documentés pour une meilleure compréhension du pré-traitement.

3 Analyse des résultats, évaluation, qualité

Voici les résultats que nous avons pu obtenir à la suite de l'exécution de notre modèle et de son évaluation :

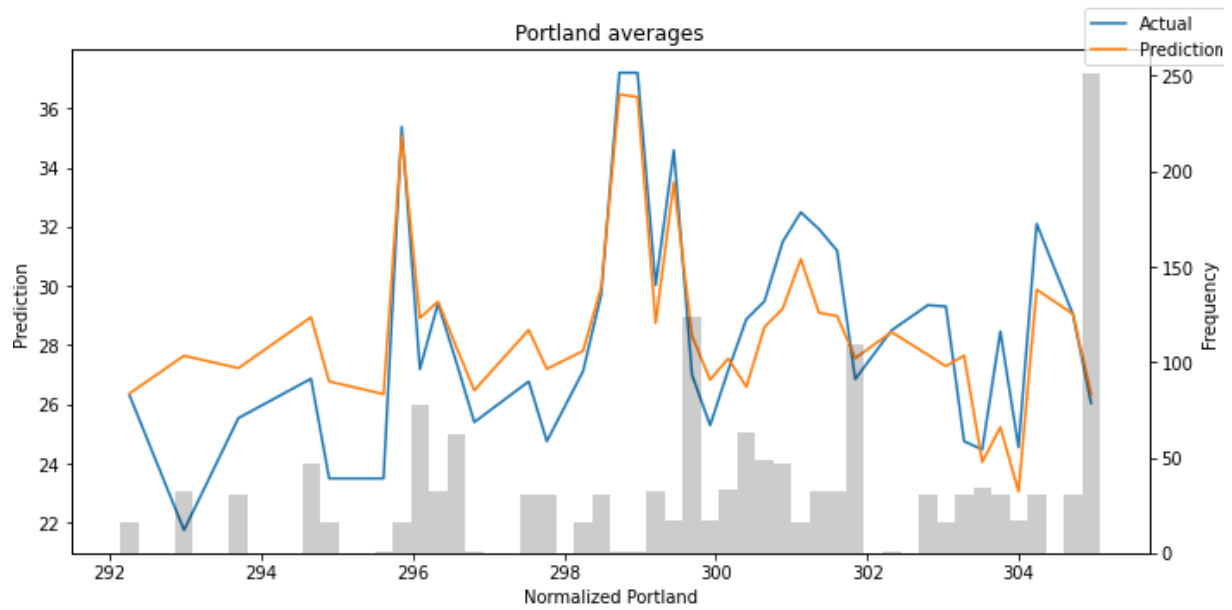


Figure 5.1 – *Prédictions obtenues en moyenne pour chaque valeur présente dans le jeu de validation*

6

Bilan et conclusion

1 Bilan du semestre 9

Ce premier semestre aura été consacré à la réalisation de la partie recherche et la partie conception du projet. Voici en détail ce qui a pu être fait au cours de ce semestre.

Recherche

Cette partie recherche aura consisté en l'étude d'un survey [2], qui nous aura amené à engranger un maximum d'informations et de connaissances sur l'interprétabilité des IA afin de choisir une méthode d'interprétabilité à étudier pour le prochain semestre.

Conception

Cette partie aura principalement été consacrée à la rédaction de documents liés au projet tels que l'état de l'art ou bien le cahier des spécifications. Nous avons également pu tester la méthode choisie lors de la partie recherche pour étudier son fonctionnement et ainsi prévoir sereinement la phase développement du projet.

Restant à faire

Aucun retard n'a été généré sur les tâches effectuées au cours de ce premier semestre. Par conséquent tout ce qui a été prévu a pu être terminé dans les temps.

2 Bilan du semestre 10

Ce second semestre aura été consacré à la réalisation du développement du programme et a été scindé en plusieurs parties :

Pré-traitement des données

Cette partie du début du semestre aura consisté en l'étude approfondie du jeu de données fourni et des multiples transformations à effectuer dessus dans le programme

Conception du modèle

Cette seconde partie aura été consacrée au développement du modèle ainsi que de l'entraîneur de celui-ci pour que le modèle puisse bien s'ajuster sur nos données.

Analyse des résultats

Cette dernière partie, celle ayant duré le plus longtemps durant ce semestre, fut d'analyser les résultats obtenus grâce au modèle et revenir sur les paramètres affectés aux modèles dans le cas où les résultats n'étaient pas convaincants.

Restant à faire

La phase de développement et d'analyse, ayant pris beaucoup plus de temps que prévu, m'a empêchée de pouvoir travailler sérieusement sur l'aspect qualité et tests du projet. Une phase d'amélioration de la qualité serait donc restante à faire.

3 Bilan sur la qualité

Concernant la qualité, l'ensemble du programme a été commenté pour expliquer le fonctionnement de chaque partie du code, et plusieurs guides (installation, développement, utilisation) ont été édités pour permettre une potentielle reprise du projet.

4 Bilan auto-critique et sur la gestion du projet

La gestion du projet à l'aide du logiciel GanttProject s'est globalement bien déroulée, malgré l'allongement du développement du programme, ce qui aura empiété sur l'analyse des résultats et poussé à modifier certaines tâches se déroulant à la base sur la fin du projet.

Malgré ce point noir, la durée prévue des autres tâches aura été globalement très respectée.

Annexes

A

Planification, gestion de projet

1 Évolution du projet - Semestre 9

Pour la gestion de ce projet un diagramme de Gantt a été créé au tout début de celui-ci afin de pouvoir planifier chaque tâche qui serait effectuée au cours de ce premier semestre. Il aura globalement été suivi à la lettre.

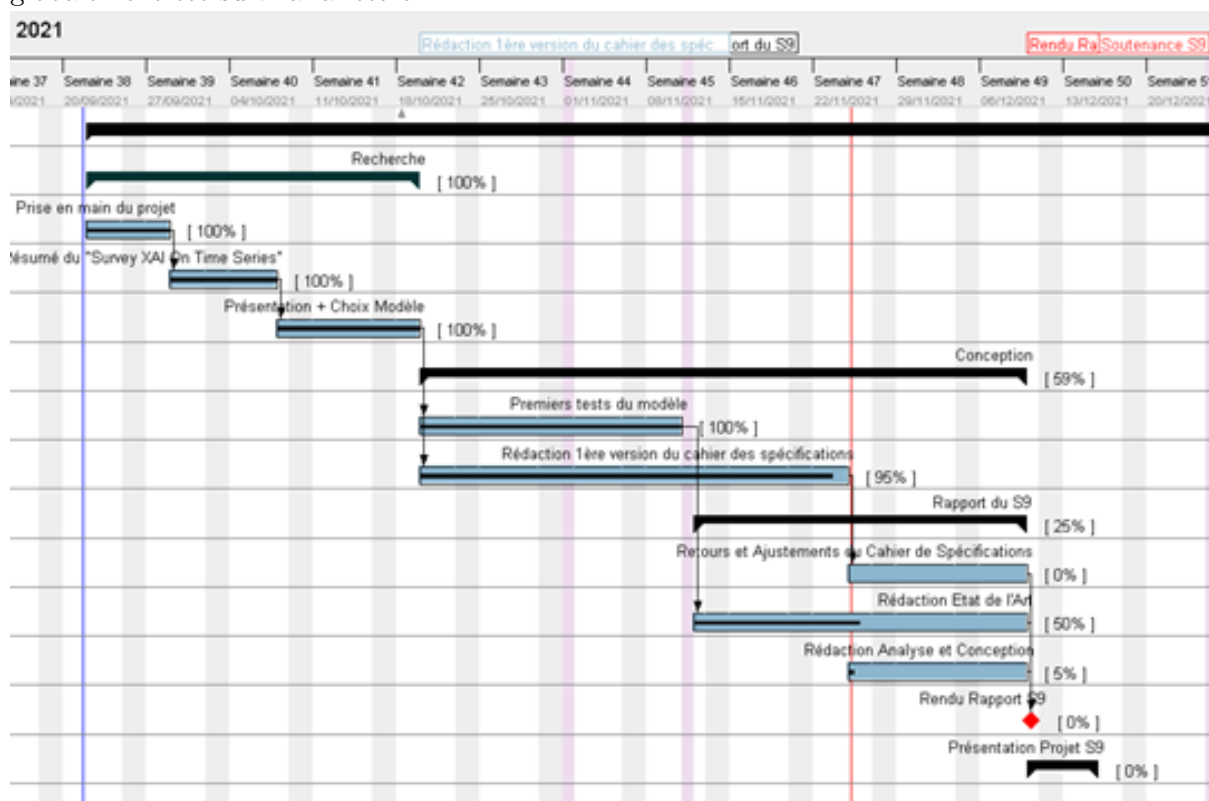


Figure A.1 – Diagramme de Gantt du travail effectué au semestre 9

A la fin du semestre 9 un planning prévisionnel a été défini sur ce même diagramme concernant les futures tâches arrivant au semestre 10 :

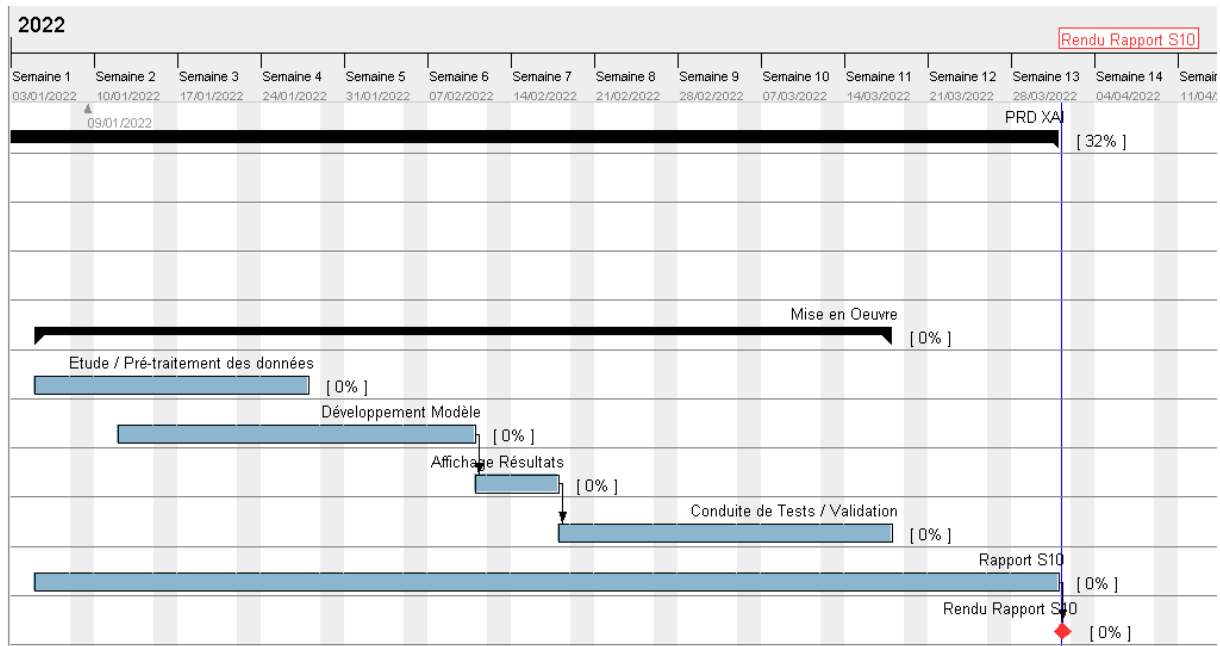


Figure A.2 – Diagramme de Gantt prévisionnel du semestre 10

Si des changements opèrent sur la réalisation et la durée de ces tâches, ce diagramme prévisionnel sera amené alors à être modifié en conséquence tout au long du semestre 10.

2 Évolution du projet - Semestre 10

Au tout début de ce semestre le diagramme de Gantt prévisionnel a été revu pour rajouter et réajuster des tâches qui seraient trop longues ou trop courtes, voici donc le nouveau diagramme de Gantt prévisionnel lié au tout début du semestre 10 :

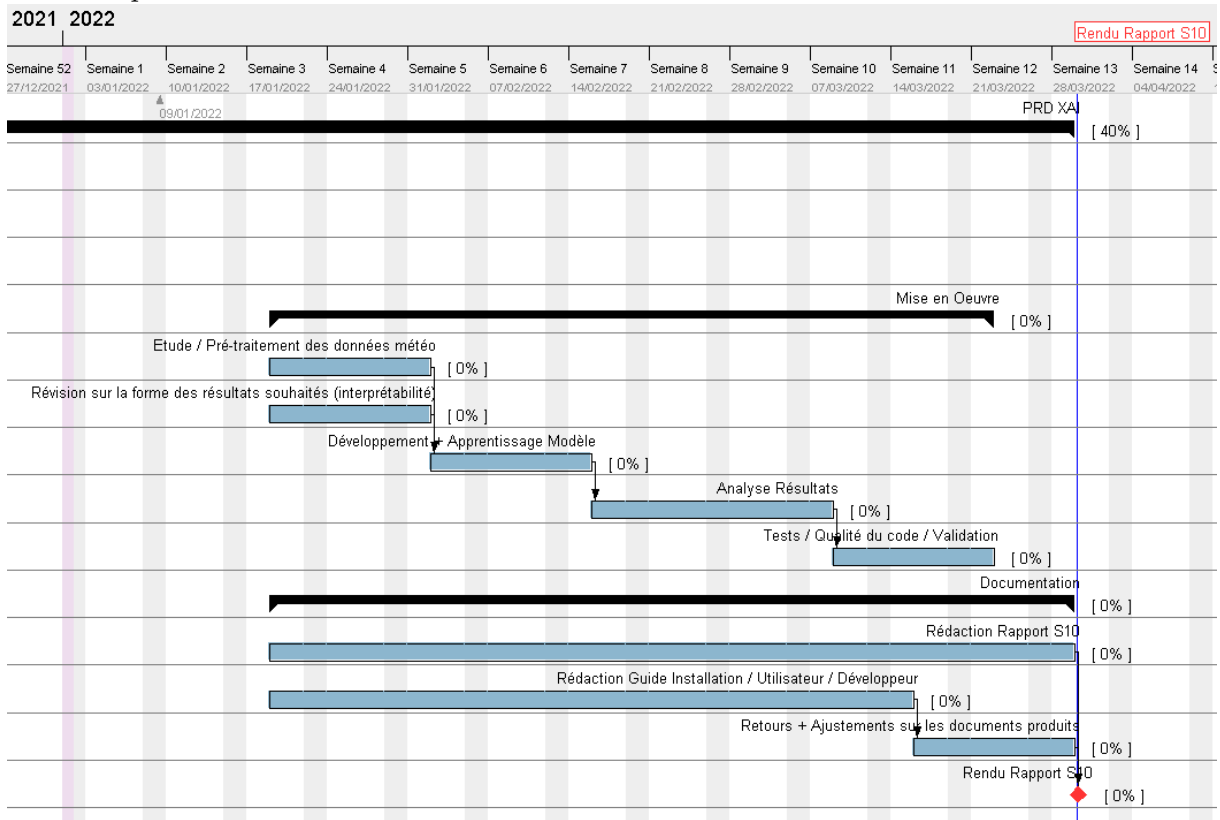


Figure A.3 – Diagramme de Gantt au jour 1 du semestre 10

A la fin de ce semestre ce diagramme aura pu évoluer au fur et à mesure de l'avancement dans les différentes tâches, et aura été marqué notamment par un allongement de la durée du développement, ce qui nous aura forcé à modifier la durée de tâches se déroulant normalement sur la fin du projet. Voici le diagramme de Gantt final montrant les modifications effectuées :

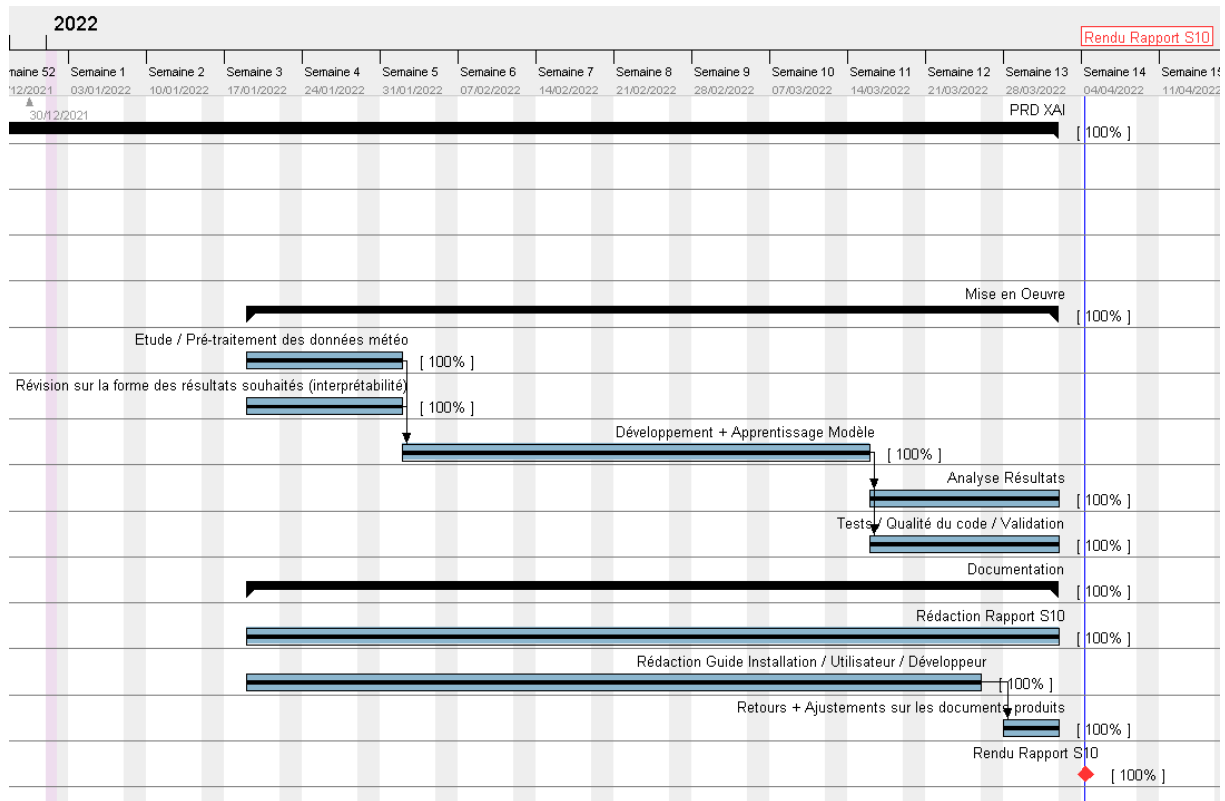


Figure A.4 – Diagramme de Gantt final du semestre 10

B

Description des interfaces

1 Interfaces Matérielles/Logicielles

L'exécution du programme sur une machine physique personnelle nécessite qu'elle ait un bon GPU et suffisamment de RAM pour limiter la durée d'exécution de celui-ci. L'utilisation de Google Colab sera alors préconisée si la machine n'est pas assez performante.

2 Interfaces Homme/Machine

L'application ne disposera pas d'interface graphique lors de son exécution, tout se déroulera dans la console de développement, ou bien tout s'affichera automatiquement pour Google Colab. Les résultats obtenus en sortie du programme, tels que les graphiques par exemple, pourront être stockés en local ou sur Google Drive, en fonction de la plateforme utilisée.

3 Interfaces Logiciel/Logiciel

Le code Python sera composé de plusieurs fonctions qui pourront communiquer entre elles et utiliser les bibliothèques mentionnées dans les contraintes de développement. Il n'y aura pas d'interface spécifique pour chacune de ces fonctions, étant donné que nous travaillerons sur un seul langage de programmation.



Cahier de Spécifications

1 Spécifications Fonctionnelles

Les spécifications présentées ci-dessous répondent au besoin mentionné dans ce rapport, qui est de pouvoir entraîner un modèle de Transformers existant sur des jeux de données pour en ressortir des résultats visuels (graphiques, ...) et une interprétabilité du fonctionnement du modèle.

1.1 Définition de la fonction 1 : Pré-Traitement des données

Format de la fonction :

Entrée : Un jeu de données fourni en entrée par l'utilisateur au format CSV

Sortie : Un objet de la classe `TimeSeriesDataSet` contenant toutes les données d'entrée formatées de sorte à pouvoir être utilisées par le modèle

Pour traiter nos données d'entrée nous devons utiliser la classe `TimeSeriesDataSet` fournie par la librairie `pytorch forecasting` qui permet d'ajuster nos données de sorte à ce qu'elles soient utilisables par le modèle, lui même provenant de cette librairie, ce qui nous oblige à utiliser cette méthode afin de simplifier la programmation de cette partie.

Un premier formatage aura tout de même lieu avant le passage dans la classe `TimeSeriesDataSet`. En effet, cette classe n'accepte que les `DataFrame` (de la librairie python `pandas`).

Contraintes : En fonction de comment le jeu de données fourni en entrée est formaté, un traitement plus ou moins lourd pourrait avoir lieu pour transformer ce jeu de données en `DataFrame pandas`.

1.2 Définition de la fonction 2 : Configuration et apprentissage du modèle

Format de la fonction :

Entrée : L'objet `TimeSeriesDataSet` fourni par la fonction précédente

Sortie : Un modèle de Transformer fonctionnel

Cette fonction va consister à faire fonctionner le modèle grâce aux méthodes associées à la classe « `TemporalFusionTransformer` » fournie par la bibliothèque .

Dans le même temps cette fonction permettra également de pouvoir faire réapprendre ce modèle sur les nouvelles données provenant du laboratoire.

L'objectif est qu'en sortie de cette méthode on puisse obtenir un modèle fonctionnel pour pouvoir y extraire les résultats souhaités par la suite.

Contraintes : Pas de contraintes particulières pour cette fonction.

1.3 Définition de la fonction 3 : Evaluation des performances du modèle

Format de la fonction :

Entrée : Le modèle généré par la fonction précédente

Sortie : Des graphiques représentant la classification effectuée sur le jeu de données

Cette fonction permettra de générer une classification des variables significatives en fonction du modèle obtenu grâce à la fonction décrite précédemment.

Cette classification prendra la forme d'un graphique qui sera affiché dans la console de commandes pour que l'utilisateur puisse en prendre connaissance. IL sera ensuite sauvegardé dans un dossier se situant dans le répertoire du programme pour une consultation ultérieure.

Contraintes : Pas de contraintes particulières pour cette fonction.

1.4 Définition de la fonction 4 : Affichage de l'interprétabilité du modèle

Format de la fonction :

Entrée : Le modèle généré et exécuté précédemment

Sortie : Des graphiques représentant la classification des variables pertinentes du dataset et d'autres éléments permettant à l'utilisateur d'interpréter le modèle généré.

Cette fonction se focalisera sur la génération d'une interprétabilité du modèle. L'interprétation que l'utilisateur pourra se faire du modèle se décomposera sur l'affichage :

- D'une classification des variables pertinentes
- De la répartition de l'attention allouée à ces variables

Contraintes : Pas de contraintes spécifiques à cette fonction.

2 Spécifications non fonctionnelles

2.1 Contraintes de développement et conception

Contraintes technologiques

La spécificité d'un tel projet nécessite, avant de se lancer dans une phase de développement, de réaliser une veille technologique afin de bien lister l'ensemble des méthodes à notre disposition et de bien établir un contexte tout autour de ce projet.

Contraintes matérielles

Dans le cas où l'on utiliserait Google Colab pour l'exécution de notre programme, la seule contrainte ici serait d'avoir une bonne connexion internet, pour pouvoir accéder facilement au site. Il n'y a donc pas de contraintes matérielles autre (utilisation d'un GPU puissant, ...) dans ce cas de figure. S'il y a besoin d'entraîner un modèle spécifique au point de ne pas pouvoir faire marcher le programme sur Google Colab, un ordinateur puissant disposant de suffisamment de RAM et d'un bon GPU sera nécessaire, afin de limiter majoritairement la durée d'exécution du programme.

Logiciels et bibliothèques à utiliser

L'utilisation de librairies spécifiques à notre cas (majoritairement des librairies de deep learning) ainsi que d'un modèle déjà entraîné ne nous laisse pas le choix sur le langage de programmation à utiliser : Python.

Les principales bibliothèques qui seront utilisées pour notre programme sont des librairies de deep learning comme "PyTorch" ainsi que "pytorch-forecasting", une librairie contenant notamment notre modèle de transformer pré-entraîné (plus de détails donnés dans les sections "Etat de l'art" et "Analyse et Conception" de ce rapport). De plus, nous utiliserons notamment NumPy pour le prétraitement des données d'entrée.

Environnement

Le développement de l'application se fera sur la plateforme en ligne Google Colab. Cette plateforme permet d'écrire et d'exécuter un programme, le tout hébergé sur une machine à distance. Le choix de cette plateforme s'est fait dans une optique d'obtenir de meilleurs résultats. En effet les machines sur lesquelles sont exécutées notre code sont conçues pour contenir de puissantes cartes graphiques et beaucoup de RAM, ainsi le temps d'exécution du code en est réduit et les résultats meilleurs.

Dans le cas où l'on viendrait à effectuer l'apprentissage de notre modèle sur notre machine personnelle, on utilisera alors Anaconda, qui permet de créer un environnement virtuel simplifié afin de facilement gérer les bibliothèques utilisées. Quant à l'IDE on utilisera Spyder.

Protocoles de communication

Dans la mesure où notre programme n'utilise pas de contenu externe à celui-ci, ou ne nécessite pas d'appel à un serveur spécifique, il n'y a pas de problématique de communication liée à ce projet.

2.2 Contraintes de fonctionnement et d'exploitation

2.2.1 Performances

L'exécution du programme pourra prendre de plusieurs secondes à plusieurs minutes, en fonction de la taille des données passées en entrée de celui-ci. La phase la plus longue sera lors de l'entraînement du modèle sur ces données. Il n'y a cependant pas de contraintes concernant la durée d'exécution, ce qui n'empêche pas donc de devoir attendre plusieurs minutes pour obtenir des résultats.

2.2.2 Capacités

Notre modèle doit pouvoir travailler sur des séries de données temporelles, et en fonction de la taille de ces données d'entrée la mémoire RAM peut être plus ou moins sollicitée. Cela est vrai uniquement dans le cas où l'on utiliserait une machine personnelle pour faire fonctionner notre application. Dans ce cas de figure, et si la taille des données étudiées est vraiment importante, une machine contenant plus de 8 Go de RAM est conseillée.

Si l'exécution se fait depuis Google Colab, il n'y a aucune contrainte de capacité étant donné que tout se fait depuis une machine hébergée en externe.

2.2.3 Contrôlabilité

Sur Google Colab le code sera divisé en plusieurs blocs et devront donc être exécutés à la suite pour faire fonctionner correctement l'entièreté du code. Pour chaque exécution de bloc des messages apparaîtront pour informer l'utilisateur du processus en cours et si des problèmes surviennent. En fonction des problèmes relevés (s'il y en a), l'utilisateur pourra revenir sur le bloc de code qui pose un problème et y apporter des modifications.

Sur Spyder le code sera exécuté sous la forme d'un seul bloc, mais les informations liées au fonctionnement de l'application apparaîtront tout de même dans la console du logiciel pour guider l'utilisateur.

2.2.4 Sécurité

L'application développée ne nécessitera pas de connexion à un serveur, donc pas d'identifiants de connexion ou de mots de passe. Ainsi tout se passera donc en local. De plus il n'y a aucune gestion de plusieurs utilisateurs dans notre cas. Il n'y aura donc pas de contrainte de sécurité liée au fonctionnement du programme.

2.2.5 Maintenance et évolution du système

En parallèle de l'application un cahier développeur sera rédigé et rendu disponible une fois ce projet finalisé, afin qu'il puisse être repris (si jamais il devait l'être) dans les meilleures conditions possibles.

De la documentation et des guides utilisateurs seront également disponibles pour l'installation des librairies, logiciels, etc.

Une évolution possible à ce projet serait d'utiliser des jeux de données différents pour tester un peu plus en détail le fonctionnement de l'apprentissage de la méthode.

D

Cahier du développeur

1 Introduction

2 Descriptions détaillées de données exploitées

3 Descriptions détaillées des classes, modules, réalisations

E

Document d'installation

Il existe deux possibilités pour installer le projet, le choix de ces possibilités se portera principalement sur la capacité de votre machine actuelle à faire tourner des programmes assez lourds.

En effet, l'exécution de ce programme nécessite une puissance de calcul assez élevée qui peut entraîner, sur des machines moins performantes, des baisses de performances lors de l'exécution du programme, et peut amener à la fin à obtenir des résultats très éloignés de ce que l'on pourrait obtenir avec une machine plus performante.

Ainsi, vous avez les deux possibilités suivantes pour installer le projet :

- L'utilisation de Google Colab, dont l'exécution du programme pourra se faire sur un serveur dédié et ainsi palier au problème d'une machine personnelle peu performante
- L'utilisation d'un environnement Anaconda avec l'IDE Spyder, réservé aux machines performantes

Nous détaillerons dans ce guide chacune des étapes vous permettant d'installer le projet pour chaque possibilité.

1 Installation de Google Colab

Si le choix s'est porté sur Google Colab, voici les différentes étapes nécessaires pour pouvoir exécuter sereinement le programme. Dans un premier temps, il vous faut ouvrir Google Colab via une page web puis importer le fichier au format .ipynb (Importer -> Choisir le fichier -> XAI_TimeSeries.ipynb) sur Google Colab. Une fois le fichier chargé, il vous faut cliquer sur le bouton « Connecter » afin d'établir une connexion sur une machine hébergée par Google qui va servir d'intermédiaire pour exécuter toutes les cellules de notre programme. Ce programme comme on peut le constater est divisé en plusieurs cellules incluant soit du code Python, soit simplement du texte décrivant globalement ce que fait la cellule de code suivante. Pour exécuter une cellule, chacune d'entre elles disposent d'un bouton « Exécuter la cellule » qui apparaît à gauche de la première ligne de code lorsque l'on passe la souris sur la cellule. Cliquez simplement dessus, et l'ensemble du code présent dans la cellule sera exécuté. La première cellule à exécuter va permettre d'installer toutes les librairies nécessaires au bon fonctionnement du programme sur la machine. Cependant attention sur ce point, la première exécution de ce code ne fonctionnera pas complètement du premier coup à cause de problèmes de dépendances liés à l'installation du

package « pytorch-forecasting » via pip. Une fois la première exécution faite, un bouton « Restart Runtime » devrait apparaître pour corriger le problème de dépendance, cliquez dessus. Une fois l'environnement d'exécution redémarré, re-exécuter la cellule pour installer définitivement toutes les librairies. Cette manipulation sera à faire obligatoirement à chaque fois qu'une connexion sur une nouvelle machine hébergée sera effectuée. Enfin, comme il est nécessaire de disposer d'un espace de stockage pour récupérer ou stocker nos fichiers d'entrées ou de sortie de notre programme, un compte Google Drive est obligatoire pour pouvoir servir d'espace de stockage à nos fichiers. Il nous faut donc pour cela monter notre compte Drive, et la cellule de code suivant celle sur l'installation des librairies permet cela. Lors de l'exécution de cette cellule, une fenêtre Google Accounts apparaîtra afin de pouvoir choisir le compte Google Drive où vous souhaitez stocker / utiliser les fichiers nécessaires au bon fonctionnement du programme. Pour l'exemple utilisé dans notre programme il est nécessaire également de disposer du fichier 'temperatures.csv' sur le Drive mentionné précédemment, afin que le programme puisse lire ces données. Votre environnement est désormais prêt à exécuter le programme.

2 Installation de Anaconda + Spyder

Si le choix s'est porté sur l'environnement Anaconda et l'IDE Spyder, voici les différentes étapes à suivre pour faire fonctionner le projet.

Dans un premier temps il vous faut installer Anaconda sur votre machine. Il s'agit de la plateforme de distribution Python la plus populaire dans le monde et intègre un certain nombre d'applications liées au traitement de la data science, dont l'IDE Spyder.

Une fois Anaconda installé, ouvrez l'application « Anaconda Navigator », afin d'accéder aux applications ainsi qu'à votre environnement virtuel. Parmi la liste d'applications disponibles, Spyder devrait être présent et s'il n'est pas marqué comme étant installé, installez-le.

L'installation d'Anaconda vous fournit déjà un environnement de base sur lequel vous pouvez travailler. Néanmoins si vous souhaitez installer les librairies requises au projet sur un autre environnement virtuel il est possible d'en créer un directement dans Anaconda Navigator (dans l'onglet Environments -> Create).

Une fois votre environnement favori sélectionné, passez sur la liste des packages / librairies disponibles et cliquez en haut sur « Update index » afin de charger la liste des librairies disponibles à l'installation. Enfin, à partir de l'outil de recherche de librairies, cherchez puis installez les librairies suivantes :

- Pandas (version utilisée : 1.3.4)
- Pytorch (version utilisée : 1.10.0)
- Pytorch-Forecasting (version utilisée : 0.9.1)
- Pytorch-Lightning (version utilisée 1.3.8)
- Tensorflow (version utilisée : 2.3.0)

Bien que les versions mentionnées soient assez spécifiques, il ne devrait pas y avoir de contraintes spéciales à faire fonctionner le programme avec une librairie ayant une version plus ancienne ou plus récente.

Attention : il est possible que certaines de ces librairies ne s'affichent pas à l'installation, pour résoudre ce problème il suffit d'ouvrir « Anaconda Prompt » (l'invite de commandes d'Anaconda) en mode administrateur, de se connecter à son environnement virtuel (par défaut il sera connecté à l'environnement de base) puis d'exécuter la commande suivante : « conda install -c conda-forge <nom_de_la_librairie> »

Une fois toutes ces étapes faites vous pouvez désormais ouvrir l'IDE Spyder, ouvrir le fichier « XAI_TimeSeries.py », et faire fonctionner le script à votre guise.

F

Document d'utilisation

G

Documents
produits

supplémentaires

1

Résumé de l'étude du survey : "Explainable Artificial Intelligence (XAI) on Time Series"



Résumé de l'enquête

Explainable Artificial Intelligence (XAI) on Time Series Data : A Survey




Introduction - Contexte

- Modèles de machine learning devenus trop complexes à comprendre / interpréter
- Utilisation de ces modèles dans de nombreux domaines « critiques »
 - Santé
 - Automobile
 - Environnement, ...
- Besoin urgent de comprendre ce qu'il se passe dans l'exécution d'algorithmes d'apprentissage automatique

Terminologies de l'XAI



Caractéristiques d'un modèle « interprétable »



Comprendre les modèles de machine learning

- Deux possibilités :
 - On s'assure dès le départ que l'algo d'apprentissage automatique peut être interprété (« model-based » ou méthodes « ante-hoc »)
 - L'algo est trop complexe
 - On fait alors appel à diverses approches / algos pour mieux comprendre son fonctionnement
 - Méthodes « post-hoc »
 - Approches de « boîte noire » et de « boîte blanche »
- Les différents types de méthodes utilisables sont distinguées en fonction de plusieurs paramètres



Catégorisation des méthodes d'interprétabilité

- Agnosticité
 - Model-agnostic : Applicable à tous les types de modèles
 - Model-specific : Applicable à un seul type de modèle
- Portée (Scope)
 - Approche Locale : Expliquer une seule partie du modèle
 - Approche Globale : Expliquer l'ensemble du modèle



Catégorisation des méthodes d'interprétabilité

- Types de données étudiées
 - Texte
 - Tableaux
 - Images / vidéos
- Types d'explications possibles
 - Représentations graphiques (courbes de corrélation, ...)
 - Informations sur l'importance de caractéristiques (features) du modèle
 - Modèles de substitution (Surrogate models) : Création d'un modèle simplifié à l'intérieur du modèle complexe de base



Méthodes présentées dans le Survey pour les séries temporelles

- Interpréter / expliquer des réseaux de neurones
 - CNN (Convolutional Neural Network)
 - RNN (Recurrent Neural Network)
- Ainsi que des méthodes pour du data mining



Réseaux de neurones convolutifs - CNN

- Deux catégories de méthodes pour interpréter les CNN
 - Les méthodes pour de la rétropropagation
 - Les méthodes pour de la perturbation
- Uniquement des méthodes « post-hoc »
- Méthodes originellement appliquées dans d'autres domaines que les séries temporelles



Réseaux de neurones convolutifs - CNN

- Rétropropagation

- Utilisation de l'algorithme CAM (Class Activation Mapping)

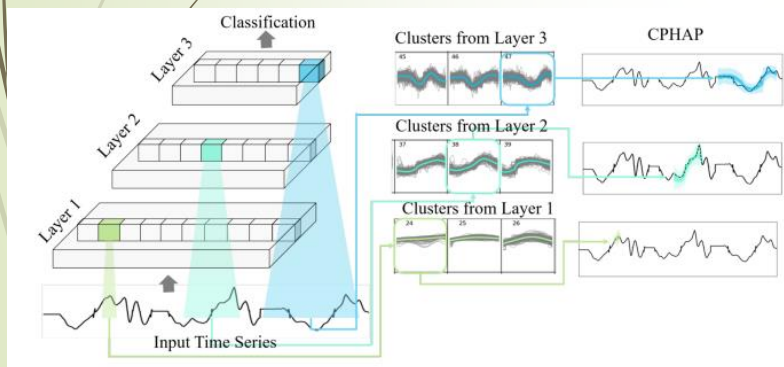
- Possibilité de mettre en évidence les sous-séquences de la série temporelle d'entrée qui sont au maximum représentatives d'une classe particulière.
 - Inconvénient : Une couche de mise en commun globale moyenne doit être ajoutée après les couches convolutives.

Réseaux de neurones convolutifs - CNN

■ Rétropropagation

■ Utilisation de la méthode Gradient*Input

- Calcul de l'activation des neurones et des filtres par rapport à une instance précise.
- Les sous-séquences d'entrée traitées ayant les filtres les plus élevés ont la plus forte contribution à la prédiction.



- Extraction des nœuds hautement actifs dans un canal
- Visualisation des sous-séquences d'entrée qui contribuent aux nœuds hautement activés.
- Enfin chaque sous-séquence extraite est affectée à un groupe de motifs similaires.



Réseaux de neurones convolutifs - CNN

■ Perturbations

- Utilisation de **ConvTimeNet** (CTN) (applications sur des **classifications** uniquement)
 - Il s'agit d'un CNN déjà entraîné sur des classifications de séries temporelles
 - Utilisation d'une méthode de sensibilité à l'occlusion
 - Trouver des régions de la série temporelle qui vont affecter la classification finale
 - Existence d'un survey lié uniquement au CTN qui présente un peu plus en détail cette méthode



Réseaux de neurones récurrents - RNN

- Rapide mention de l'algorithme SHAP (Shaplet Additive Explanation)
 - Méthode commune d'attribution de caractéristiques (model-agnostic)
- Combinaison d'un CNN comme extracteur de caractéristiques avec un modèle LSTM (Long Short Term Memory)
 - Afin d'apprendre les dépendances temporelles de la série



Réseaux de neurones récurrents - RNN

- Utilisation des « Transformers »
 - Utilisant les mécanismes d'attention
 - Méthode très populaire aujourd'hui dans l'interprétabilité des séries temporelles



Data Mining

- Utilisation de la méthode SAX (Symbolic Aggregation Approximation)
 - Transforme la série temporelle en une représentation en PAA (Piecewise Aggregate Approximation)
 - Attribution d'un symbole à chaque segment représentant la contribution de celui-ci dans la série temporelle initiale.

Résumé des méthodes XAI applicables aux séries temporelles

Model	Ante-hoc/ Post-hoc	Methodology	Model Specific/ Model Agnostic	Scope	Target audience	Explanation evaluation
<i>FCN</i> , Wang et al. [26]	Post-hoc	Backpropagation-based	Specific	Local	Developer	No
<i>ESS-CNN</i> , Fawaz et al. [27]	Post-hoc	Backpropagation-based	Specific	Local	User/DM	Yes/Qualitative
<i>XrayD-DNN</i> , Oviedo et al. [28]	Post-hoc	Backpropagation-based	Specific	Local/Global	Developer	Yes/Qualitative
<i>CY-EDL</i> , Wolanin et al. [85]	Post-hoc	Backpropagation-based	Specific	Local/Global	User/DM	Yes/Qualitative
<i>ConvtimeNet</i> , Kashiparekh et al. [40]	Post-hoc	Perturbation-based	Agnostic	Local	Developer	No
<i>FFC</i> , Tonekaboni et al. [39]	Post-hoc	Perturbation-based	Agnostic	Local/Global	Developer	Yes/Quantitative
<i>MI-FCNN</i> , Strodthoff et al. [32]	Post-hoc	Backpropagation-based	Specific	Local	User/DM	No
<i>Tsviz</i> , Siddiqui et al. [1]	Post-hoc	Backpropagation-based	Specific	Local/Global	Developer	Yes/Qualitative
<i>CHAP</i> , Cho et al. [33]	Post-hoc	Backpropagation-based	Specific	Local/Global	Developer	Yes/Quantitative
<i>CA-SFCN</i> , Hao et al. [86]	Ante-hoc	Attention Mechanism	Specific	Local	Developer	No
<i>ALSTM-FCN</i> , Karim et al. [43]	Ante-hoc	Attention Mechanism	Specific	Local	Developer	No
<i>AM-LSTM</i> , Schockaert et al. [44]	Ante-hoc	Attention Mechanism	Specific	Local/Global	Developer	No
<i>TFT</i> , Lim et al. [50]	Ante-hoc	Attention Mechanism	Specific	Local/Global	Developer	No
<i>Tsinsight</i> , Siddiqui et al. [87]	Ante-hoc	Attention Mechanism	Specific	Local/Global	Developer	No
<i>Sax-vsm</i> , Senin et al. [56]	Ante-hoc	SAX	Specific	Global	Developer	No
<i>SAX-SFA-SEQ</i> , Le Nguyen et al. [57]	Ante-hoc	SAX	Specific	Global	Developer	No
<i>AI-PR-CNN</i> , Wang et al. [81]	Ante-hoc	Shapelets	Specific	Global	Developer	No
<i>BSPCOVER</i> , Li et al. [83]	Ante-hoc	Shapelets	Specific	Global	Developer	No
<i>GST</i> , Kidger et al. [80]	Ante-hoc	Shapelets	Specific	Global	Developer	No
<i>eUA-CRNN</i> , Tan et al. [88]	Ante-hoc	Attention Mechanism	Specific	Local	Developer	No
<i>IDH-DSC-ERNN</i> , Choi et al. [45]	Ante-hoc	Attention Mechanism	Specific	Local	Developer	No
<i>ETNODE</i> , Gao et al. [89]	Ante-hoc	Attention Mechanism	Specific	Local/Global	Developer	No
<i>Tsxpain</i> , Munir et al. [90]	Post-hoc	Backpropagation-based	Agnostic	Local	Developer	Yes/Quantitative
<i>TCL</i> , Vinayavekhin et al. [46]	Ante-hoc	Attention Mechanism	Specific	Local	Developer	Yes/Quantitative
<i>ICU-LSTM</i> , Ge et al. [47]	Ante-hoc	Attention Mechanism	Specific	Global	Developer	No
<i>Series saliency</i> , Pan et al. [91]	Post-hoc	Perturbation-based	Agnostic	Local	Developer	Yes/Qualitative
<i>Mtex-cnn</i> , Assaf et al. [92]	Post-hoc	Backpropagation-based	Specific	Local	Developer	Yes/Quantitative
<i>RATIO</i> , Augustin et al. [93]	Post-hoc	Counterfactuals	Agnostic	Local	Developer	No
<i>PDL</i> , Gee et al. [94]	Ante-hoc	Prototypes	Specific	Global	Developer	Yes/Qualitative
<i>FAHP</i> , El-Sappagh et al. [68]	Ante-hoc	Fuzzy logic	Specific	Global	User	No
<i>Deep-FCM</i> , Wang et al. [69]	Ante-hoc	Fuzzy logic	Specific	Global	Developer	Yes/Qualitative

TABLE I: Summary of XAI methods applied to time series. Abbreviations: SAX: Symbolic Aggregate Approximation; DM: Decision Maker;

H

Webographie

- [WWW1] David BENIAGUEV. *Historical Hourly Weather Data 2012-2017*. URL : <https://www.kaggle.com/datasets/selfishgene/historical-hourly-weather-data?select=temperature.csv>.
- [WWW2] Utathya GHOSH. *Volume Forecasting - SKU future volume analysis and prediction*. URL : <https://www.kaggle.com/utathya/future-volume-prediction>.
- [WWW3] *PyTorch Forecasting : Demand forecasting with the Temporal Fusion Transformer*. URL : <https://pytorch-forecasting.readthedocs.io/en/latest/tutorials/stallion.html>.

I

Bibliographie

- [1] Bryan LIM, Sercan O. ARIK, Nicolas LOEFF et Tomas PFISTER. « Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting ». In : *CoRR* abs/1912.09363 (2020). arXiv : 1912.09363. URL : <https://arxiv.org/abs/1912.09363>.
- [2] Thomas ROJAT, Raphael PUGET, David FILLIAT, Javier Del SER, Rodolphe GELIN et Natalia DIAZ-RODRIGUEZ. « Explainable Artificial Intelligence (XAI) on Time Series Data : A Survey ». In : *CoRR* abs/2104.00950 (2021). arXiv : 2104.00950. URL : <https://arxiv.org/pdf/2104.00950.pdf>.
- [3] Ashish VASWANI, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT, Llion JONES, Aidan N. GOMEZ, Łukasz KAISER et Illia POLOSUKHIN. « Attention is all you need ». In : *CoRR* abs/1706.03762 (2017). arXiv : 1706.03762. URL : <https://arxiv.org/pdf/1706.03762.pdf>.

J

Acronymes

NLP Natural Language Processing. 8

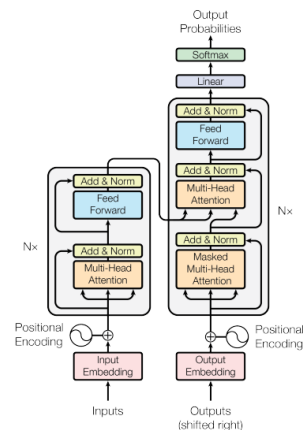
Interprétabilité des IA (XAI) : Application à l'analyse de séries temporelles

Joshua ROUSSEAU

Encadrement : Nicolas RAGOT

Objectifs

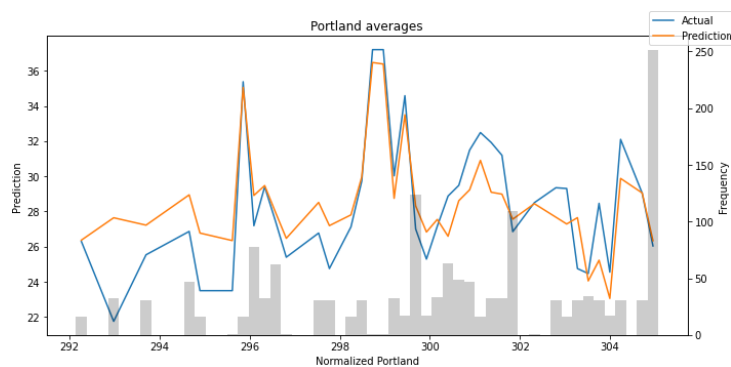
L'objectif de mon PRD fut de répertorier l'ensemble des méthodes d'interprétation déjà existantes sur le traitement des séries temporelles, et d'en appliquer une sur un cas concret. La méthode qui a été choisie est celle dite des Transformers.



Architecture d'un Transformer

Mise en œuvre

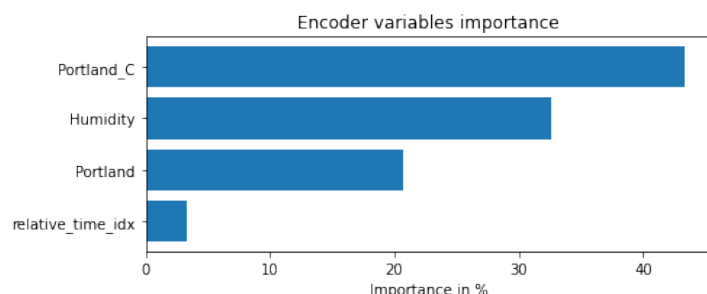
Cette méthode a été appliquée sur un jeu de données lié à des relevés météorologiques, l'objectif de cette mise en œuvre était de pouvoir fournir dans un premier temps une prédiction de ces données, mais également une interprétation sur le fonctionnement du modèle de transformer étudié.



Prédictions obtenues en moyenne sur l'ensemble du jeu de données étudié

Résultats attendus

L'utilisation de cette méthode basée sur les transformers a permis de constater les performances du modèle ainsi que des éléments sur l'interprétabilité et les choix faits par le modèle.



Joshua ROUSSEAU

Encadrement : Nicolas RAGOT

Objectifs

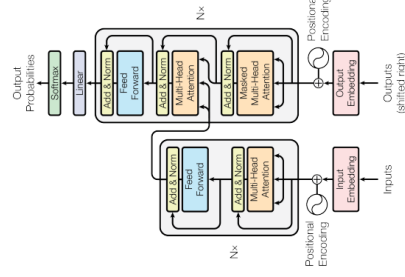
L'objectif de mon PRD fut de répertorier l'ensemble des méthodes d'interprétation déjà existantes sur le traitement des séries temporelles, et d'en appliquer une sur un cas concret. La méthode qui a été choisie est celle dite des Transformers.

Mise en œuvre

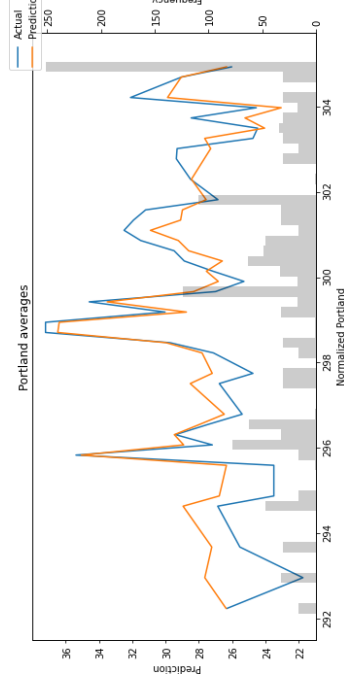
Cette méthode a été appliquée sur un jeu de données lié à des relevés météorologiques, l'objectif de cette mise en œuvre était de pouvoir fournir dans un premier temps une prédiction de ces données, mais également une interprétation sur le fonctionnement du modèle de transformer étudié.

Résultats attendus

L'utilisation de cette méthode basée sur les transformers a permis de constater les performances du modèle ainsi que des éléments sur l'interprétabilité et les choix faits par le modèle.



Architecture d'un Transformer



Prédictions obtenues en moyenne sur l'ensemble du jeu de données étudié

Interprétabilité des IA (XAI)

Application à l'analyse de séries temporelles

Résumé

Ce document est un rapport de Projet de Recherche et de Développement axé sur l'interprétabilité des intelligences artificielles sur les séries temporelles. Il fait l'état des méthodes existantes dans le domaine, puis présente l'implémentation de l'une de ces méthodes : les "Transformers" et son application sur les séries temporelles.

Mots-clés

Interprétabilité, Explicabilité, IA, XAI, Séries Temporelles, Transformers

Abstract

This document is a Research and Development Project report focused on the interpretability of artificial intelligences over time series. It reviews the existing methods in the field, then presents the implementation of one of these methods : the "Transformers" and its application on time series.

Keywords

Interpretability, Explainability, AI, XAI, Time Series, Transformers