RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ 20 AOUT 1955 - SKIKDA
FACULTÉ DE SCIENCE
DÉPARTEMENT De L'INFORMATIQUE

Mémoire de fin d'études

Pour l'obtention du diplôme de :

Master en **Informatique**

Spécialité : **Réseaux et Systèmes Distribués**

THEME

Prédiction des ventes de médicaments par l'Apprentissage Automatique

Réalisé par :

- **Bilal LEKAMECHE**
- **Zahra NEGHRA**

Encadré par :

M. Laïdi FOUGHALI

Année universitaire : 2023 /2024

Dédicaces

À mes parents,

Votre soutien inébranlable, vos encouragements constants et votre amour inconditionnel m'ont donné la force et la détermination nécessaires pour atteindre ce jalon.

À ma femme et mes enfants,

Votre amour et votre compréhension ont été mon refuge et ma motivation. Ma chère **Sihem**, ta patience, ton soutien et ta confiance en moi ont été des piliers essentiels durant ce parcours exigeant. Tes sacrifices et ton dévouement ont rendu possible l'achèvement de ce mémoire.

À mes enfants, **Maria** et **Koussai**, vos sourires et vos câlins ont été des rayons de lumière, même dans les moments les plus sombres. Vous êtes ma plus grande source d'inspiration et de bonheur.

À ma fille qui va bientôt naître, si Dieu le veut, nous attendons ton arrivée avec impatience.

À mes professeurs,

Merci pour votre guidance, votre expertise et votre dévouement. Vous m'avez non seulement transmis des connaissances précieuses, mais vous m'avez également inspiré à toujours viser l'excellence. Vos conseils avisés et votre soutien tout au long de mon parcours académique ont été inestimables.

À mes amis et collègues,

Votre camaraderie, votre aide précieuse et vos encouragements ont rendu ce voyage d'apprentissage plus agréable et supportable. Merci pour les moments de partage, de soutien mutuel et de motivation. Vous avez été une source de force et de réconfort durant les périodes de stress et de doute.

À tous ceux qui ont cru en moi et m'ont soutenu de près ou de loin,

Votre confiance et votre soutien ont été essentiels pour me permettre d'atteindre cet objectif. Je vous en suis éternellement reconnaissant.

Avec toute ma gratitude et mon affection,

L. Bilal

Dédicaces

Du profond de mon cœur, je dédie ce travail à tous qui me sont chers.

À mes chers parents

Aucune dédicace ne saurait exprimer mon respect. Mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être.

Mon cher père décédé, merci beaucoup pour tous vos sacrifices, je prie Dieu de vous placer au paradis éternel

Je vous remercie pour tous le soutien et l'amour que vous me portez depuis mon enfance et j'espère que votre bénédiction m'accompagne toujours
À mon paradis, Mama, reposez votre cœur, votre rêve est devenu réalité. Enfin votre petite fille est devenue diplôme comme tu a tant imaginée et désirée.

À mes merveilleux frères et ma sœur

A mes cher frère « adel » et « Yassin »et ma belle-sœur « Bouteina » en témoignage de l'attachement, de l'amour et l'affectation que je porte pour vous, vous êtes toujours dans mon cœur .je vous dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.

À mes professeurs

Je dédie mes professeurs pour leur enseignement précieux et leurs conseils avisés.

À mes amis et collègues,

À mes amis et collègues « Rayene » et « Maissa » merci mes chers proches vous m'avez soutenu dans mon parcours universitaire.

À tous ma famille, merci beaucoup pour vous remercier de m'avoir soutenu tout au long de ce parcours.

N. Zahra

Remerciement

Nous souhaitons exprimer notre plus profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce mémoire de master. Leur soutien, sous diverses formes, a été essentiel et a grandement facilité l'aboutissement de ce travail.

Nous tenons à remercier tout particulièrement notre encadreur, **M. Laïdi FOUGHALI**. Sa patience, ses conseils avisés et son expertise ont été déterminants pour la réussite de ce mémoire. Sa disponibilité et son implication ont été d'un soutien précieux, et c'est en grande partie grâce à lui que ce projet a pu voir le jour.

Enfin, nous n'oublions pas nos familles et nos amis pour leur soutien constant et leurs encouragements tout au long de cette aventure académique. Leurs encouragements et leur soutien moral ont été inestimables.

Merci à tous pour votre aide précieuse.

Résume

L’industrie pharmaceutique est un secteur clé pour la santé publique et l’économie nationale. En effet, les ventes de médicaments sont un indicateur essentiel de la demande en produits de santé et la prédiction de ces ventes aide diverses parties prenantes, notamment les fabricants, les distributeurs et les régulateurs. De nombreuses recherches ont proposé des méthodes de prédiction des ventes en utilisant des techniques traditionnelles de régression. Avec l’arrivée du big data et les avancées de l’apprentissage automatique, il est désormais possible de construire des modèles plus sophistiqués afin d’exploiter les informations contenues dans les vastes ensembles de données pharmaceutiques.

Dans le cadre de ce mémoire, nous avons dans un premier temps présenté les bases de l'apprentissage automatique. Cette introduction vise à fournir les connaissances nécessaires pour comprendre comment ces techniques peuvent être appliquées à la prédiction des ventes. Dans un second temps, nous avons présenté les spécificités du marché des médicaments. En comprenant ces particularités, on peut mieux appréhender les défis et les variables à considérer pour prédire les ventes de médicaments de manière précise. Ensuite, nous implémentons les modèles proposés basent sur la bibliothèque « Tensorflow », en commençant par préparer la collection de données choisie. Pour finir, nous évaluons les résultats obtenus par nos modèles.

Mots clés : Apprentissage Automatique, prédiction des ventes, ventes de médicaments.

Abstract

The pharmaceutical industry is a key sector for public health and the national economy. Indeed, drug sales are a critical indicator of demand for healthcare products, and predicting these sales helps various stakeholders, including manufacturers, distributors, and regulators. Much research has proposed methods for predicting sales using traditional regression techniques. With the arrival of big data and advances in machine learning, it is now possible to build more sophisticated models to exploit the information contained in large pharmaceutical datasets.

As part of this dissertation, we first presented the basics of machine learning. This introduction aims to provide the knowledge necessary to understand how these techniques can be applied to sales prediction. Secondly, we presented the specificities of the medicines market. By understanding these particularities, we can better comprehend the challenges and variables to consider in accurately predicting drug sales. Then, we implemented the proposed models, based on the “Tensorflow” library, starting by preparing the chosen data collection. Finally, we evaluated the obtained results by our models.

Keywords: Machine Learning, sales prediction, drug sales.

Table des matières

Liste des figures....4

Liste des tableaux....5

Introduction générale....6

Chapitre I : L'apprentissage automatique....8

1. Introduction :....8

2. Un peu d'histoire :....8

3. Types d'apprentissage automatique :....9

3.1. Apprentissage supervisé....9

3.2. Apprentissage non supervisé....10

3.3. Apprentissage par renforcement....10

4. Applications de l'apprentissage automatique dans le monde réel....11

4.1. L'apprentissage automatique dans les finances....11

4.2. L'apprentissage automatique dans la santé....11

4.3. L'apprentissage automatique dans les véhicules autonomes....11

4.4. L'apprentissage automatique dans le commerce électronique....12

4.5. L'apprentissage automatique dans la gestion des ressources humaines....12

5. Conclusion :....12

Chapitre II : Les ventes de médicaments et leurs particularités....13

1. Introduction :....13

2. Nature des Ventes de Médicaments :....13

2.1. Saisonnière et Cyclique :....13

2.2. Politique de Santé et Réglementation :....13

2.3. Innovation et Demande :....13

2.4. Influence des Médias et des Campagnes de Marketing :....14

3. Catégorisation et Généricité des Médicaments :....14

3.1. Catégorisation des Médicaments : 14

3.2. Médicaments Génériques : 14

4. Attribution de Groupe de Médicaments : 15

5. Avantages financiers du marché algérien des médicaments : 15

5.1. Fixation des prix : 16

5.2. Tendance Croissante des Médicaments Génériques : 16

5.3. Investissements et Innovation : 16

6. Etat de l'art : 16

7. Conclusion : 17

Chapitre III : Modèle proposé 18

1. Introduction 18

2. Objectif et motivation 18

3. Collection, Sélection, Préparation et normalisation des données : 19

3.1. Collection des données 19

3.2. Sélection des données : 19

3.3. Préparation et normalisation des données : 20

4. Modèles à implémenter : 20

4.1. Modèle 1: LSTM (Long Short-Term Memory) 20

4.2. Modèle 2 : MLP (MultiLayer Perceptron) 21

4.3. Modèle 3 : Régression Polynomiale 23

5. Conclusion 23

Chapitre IV : Implémentation 24

1. Introduction : 24

2. Préparation de l'environnement de développement : 24

3. Bibliothèques utilisées : 25

3.1. Numpy : 25

3.2. Pandas : 25

3.3. Scikit-learn :.....25
3.4. Tensorflow :.....26
3.5. Keras :.....26
3.6. Matplotlib :.....26
3.7. Firebird:.....27
3.8. Installation.....27
4. Modèle général d'un algorithme basé sur tensorflow :.....27
5. Préparation des données – phase 1 :.....29
6. Métriques d'évaluation :.....31
7. Création, compilation et entrainement des modèles :.....31
7.1. Préparation des données – phase 2 :.....31
7.2. Création des modèles :.....33
7.3. Compilation et Entrainement des modèles :.....35
8. Résultat obtenu :.....37
8.1. LSTM (Long Short-Term Memory) :.....37
8.2. MLP (MultiLayer Perceptron) :.....40
8.3. Régression polynomiale :.....43
9. Comparaison et évaluation des résultats.....46
9.1. Comparaison :.....46
9.2. Évaluation :.....48
10. Conclusion :.....48
Conclusion générale.....49
Bibliographie.....50
Annexe.....I
1. LSTM :.....I
2. MLP :.....IV
3. Régression polynomiale :.....VII

Liste des figures

Figure 01 : Chronologie de l'apprentissage Automatique 8
Figure 02 : Types d'apprentissage automatique 9
Figure 03: Exemple de l'apprentissage supervisé 10
Figure 04 Exemple de l'apprentissage supervisé 10
Figure 05: Exemple de l'apprentissage par renforcement 11
Figure 06: Demande du marché pharmaceutique en Afrique par pays 15
Figure 07: Cellule d'un réseau LSTM 21
Figure 08: Diagramme d'un perceptron multicouche 22
Figure 09: Squelette d'un algorithme basé sur la librairie tensorflow 28
Figure 10: Représentation des données brute (Cas : Paracétamol 500MG Comprimé) 29
Figure 11: Représentation des données après traitement des valeurs manquants (Cas : Paracétamol 500MG Comprimé) .. 29
Figure 12: Représentation des données après lissage (Cas : Paracétamol 500MG Comprimé) 30
Figure 13: Représentation des données des ventes hebdomadaires (Cas : Paracétamol 500MG Comprimé) 30
Figure 14 : Résultat d'entrainement - Modèle LSTM - Produit : Paracétamol - 500mg – Comprimé 37
Figure 15 : Résultat d'entrainement - Modèle LSTM - Produit : Paracétamol - 1000mg - Comprimé 38
Figure 16 : Résultat d'entrainement - Modèle LSTM - Produit : Diclofénac – 1g/100g – gél 38
Figure 17 Résultat d'entrainement - Modèle LSTM - Produit : Oméprazole – 20mg 39
Figure 18 : Résultat d'entrainement - Modèle LSTM - Produit : Amoxicilline - 250g/5ml - Flacon 40
Figure 19 : Résultat d'entrainement - Modèle MLP - Produit : Paracétamol - 500mg – Comprimé 40
Figure 20 : Résultat d'entrainement - Modèle MLP - Produit : Paracétamol - 1000mg – Comprimé 41
Figure 21 : Résultat d'entrainement - Modèle MLP - Produit : Diclofénac – 1g/100g – gél 41
Figure 22 : Résultat d'entrainement - Modèle MLP - Produit : Oméprazole - 20mg 42
Figure 23 : Résultat d'entrainement - Modèle MLP - Produit : Amoxicilline - 250mg/5ml - Flacon 43
Figure 24 : Résultat d'entrainement – Modèle : Régression polynomiale - Produit : Paracétamol - 500mg – Comprimé .. 43
Figure 25 : Résultat d'entrainement – Modèle : Régression polynomiale - Produit : Paracétamol - 1000mg – Comprimé . 44
Figure 26 : Résultat d'entrainement – Modèle : Régression polynomiale - Produit : Diclofénac – 1g/100g – gél 44
Figure 27 : Résultat d'entrainement – Modèle : Régression polynomiale - Produit : Oméprazole – 20mg 45
Figure 28 : Résultat d'entrainement – Modèle : Régression polynomiale - Produit : Amoxicilline – 250mg/5ml – Flacon .. 45
Figure 29 : Comparaison des résultats de MAE 46
Figure 30 : Résultats du métrique "MAE" 46
Figure 31 : Résultats du métrique "MSE" 46
Figure 32 : Résultats du métrique "RMSE" 47
Figure 33 : Résultats du métrique "MAPE" 47
Figure 34 : Résultats du métrique "R^2" 48

Liste des tableaux

Tableau 01 : Résultat de sélection des produits..... 19

Introduction générale

Vue la transformation numérique accélérée du secteur pharmaceutique. Les entreprises de ce domaine sont confrontées à des défis complexes, notamment la gestion efficace des stocks, la prévision de la demande, et la minimisation des pertes dues à des médicaments périmés ou non vendus. L'intégration de l'apprentissage automatique offre une opportunité révolutionnaire pour répondre à ces enjeux en fournissant des modèles prédictifs capables d'anticiper les tendances de vente avec une plus grande précision. Notre étude cherche à exposer quelques techniques de Machine Learning basées sur l'incroyable bibliothèque TensorFlow, d'analyser leur pertinence et efficacité dans le contexte spécifique de la vente de médicaments, et d'apporter des solutions innovantes pour optimiser les stratégies commerciales des entreprises pharmaceutiques.

Notre mémoire aborde la problématique de l'estimation de la valeur des ventes de médicaments. Pour résoudre cette problématique, nous avons créé trois modèles. Le premier modèle est basé sur les réseaux de neurones récurrents (RNN) en raison de leur capacité à traiter des données séquentielles et à capturer les dépendances temporelles, c'est LSTM (Long Short-Term Memory). Le second modèle utilise des perceptrons multicouches (MLP) pour leur capacité à modéliser des relations non linéaires complexes dans les données. Les MLP sont particulièrement efficaces pour les tâches de classification et de régression où les données peuvent être considérées comme indépendantes et identiquement distribuées. Dans le troisième et dernier modèle, nous avons opté pour l'utilisation de la régression polynomiale, une méthode de modélisation qui étend la régression linéaire en utilisant des polynômes pour modéliser les relations entre les variables indépendantes et dépendantes.

Notre contribution se situe à plusieurs niveaux :

• Premièrement, nous présentons une vue d'ensemble de l'apprentissage automatique, de ses différents types et de ses applications dans le monde réel.

• Deuxièmement, nous abordons les ventes de médicaments et leurs particularités.

• Ensuite, nous présentons les trois modèles d'apprentissage automatique sélectionnés.

• Enfin, nous décrivons la création, l'entraînement et le test de ces trois modèles sur la collection de données choisies afin de comparer leurs précisions.

Ce mémoire est structuré en quatre chapitres, en plus de l'introduction et de la conclusion. Les deux premiers chapitres sont théoriques, tandis que les deux derniers décrivent les travaux effectués pour la réalisation des trois modèles proposés.

Le premier chapitre propose une présentation globale de l'apprentissage automatique, en commençant par un bref historique, puis en abordant les types d'apprentissage et leurs applications dans le monde réel. Le second chapitre se concentre sur les ventes de médicaments et leurs particularités.

Le troisième chapitre présente en détail les trois modèles proposés pour la prédiction des ventes de médicaments. Enfin, le quatrième chapitre expose l'implémentation et les résultats des trois modèles, en décrivant d'abord les différentes bibliothèques utilisées.

Chapitre I : L'apprentissage automatique

1. Introduction :

Le Machine Learning, en français L'apprentissage automatique, situé à l'intersection de l'informatique et des statistiques, permet aux ordinateurs de s'améliorer grâce à l'expérience.

Dans ce chapitre, nous présentons une définition, suivie par un bref historique pour comprendre l'évolution de cette technologie, en suite en explore les types d'apprentissage automatique (supervisé, non supervisé, et par renforcement), Enfin, nous avons examiné les multiples applications de l'apprentissage automatique dans le monde réel. L'objectif est de fournir une compréhension fondamentale des principes et des pratiques essentielles de l'apprentissage automatique.

2. C'est quoi l'apprentissage automatique :

L'apprentissage automatique, ou Machine Learning en anglais, est un sous-domaine de l'intelligence artificielle qui se concentre sur la conception et le développement d'algorithmes permettant aux ordinateurs de « apprendre » à partir de données. Plutôt que d'être explicitement programmés pour effectuer une tâche, les algorithmes d'apprentissage automatique utilisent des techniques statistiques pour détecter des modèles dans les données et améliorer progressivement leur performance sur une tâche spécifique.

3. Un peu d'histoire :

L'histoire de l'apprentissage automatique présenter par la **figure 01** est intimement liée à celle de l'intelligence artificielle, remontant aux années 1950 avec Alan Turing et son test de Turing. En 1958, Frank Rosenblatt a présenté le Perceptron, premier algorithme d'apprentissage supervisé. En 1959, Arthur Samuel popularise le terme "apprentissage machine" avec un jeu de dames capable d'apprendre.

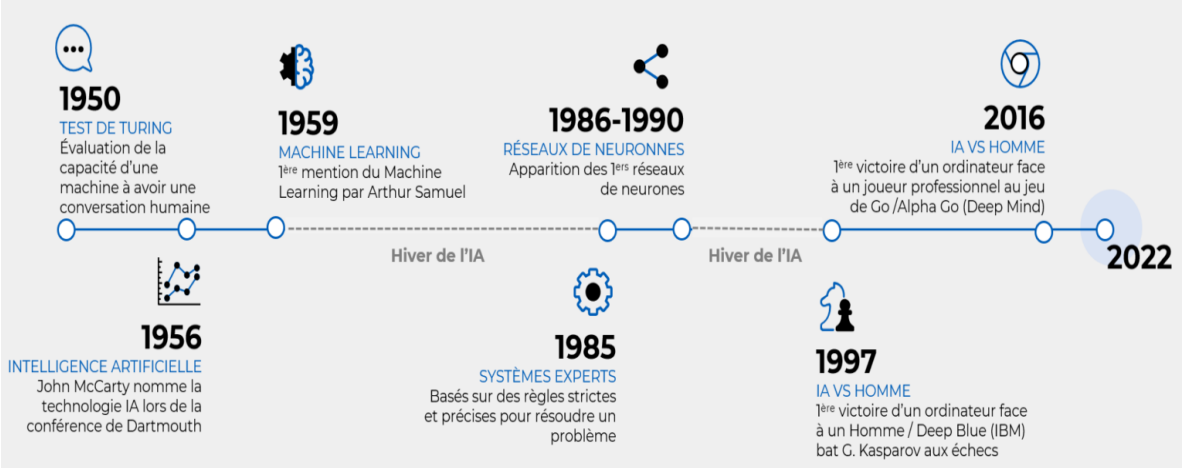

Figure 01 : Chronologie de l'apprentissage Automatique

La découverte majeure de la rétro propagation par David Rumelhart, Geoffrey Hinton et Ronald Williams dans les années 1980 a révolutionné le domaine.

En 2012, Geoffrey Hinton et ses étudiants ont développé un algorithme de reconnaissance d'images performant. Aujourd'hui, des entreprises et universités collaborent pour démocratiser l'apprentissage machine avec des ressources open source comme TensorFlow et PyTorch. [1]

4. Types d'apprentissage automatique :

L'apprentissage automatique représente un ensemble étendu d'algorithmes et d'instruments de modélisation employés pour analyser des données dans une diversité de domaines. Trois catégories principales d'apprentissage automatique sont reconnues, chacune étant conçue pour des objectifs distincts.

La **figure 02** illustre les différents types d'apprentissage automatique.

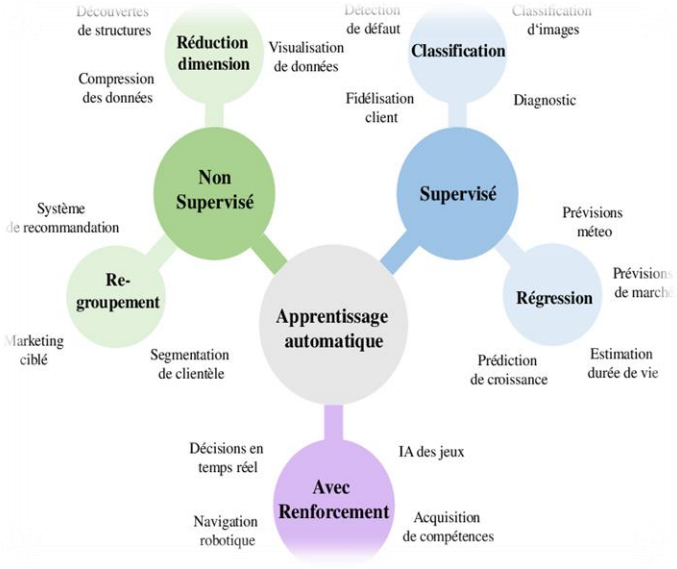

Figure 02 : Types d'apprentissage automatique

4.1. Apprentissage supervisé

On parle d'apprentissage supervisé lorsque l'on dispose de données d'entraînement étiquetées, où chaque instance de formation est appariée avec un résultat ou une étiquette spécifique, La **figure 03** présente un exemple simple d'apprentissage supervisé.

Le but est de développer un modèle apte à prédire l'étiquette de nouvelles entrées inédites. L'apprentissage supervisé est fréquemment appliqué à des tâches de classification et de régression.

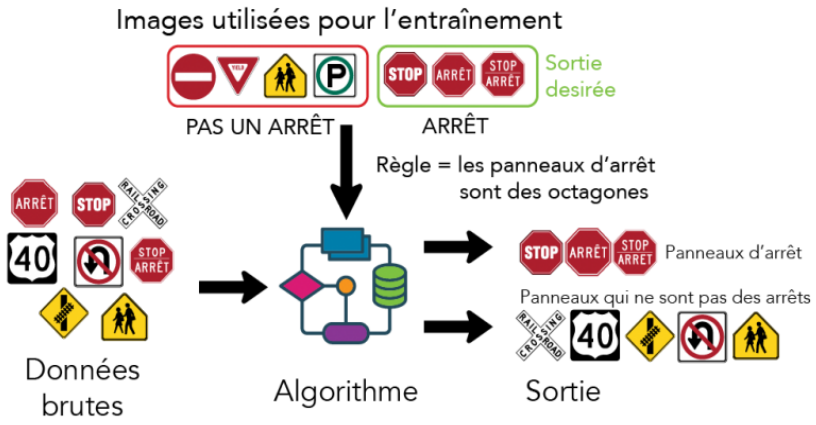

Figure 03: Exemple de l'apprentissage supervisé

Généralement utilisé pour la régression lorsque la sortie à prédire peut prendre des valeurs continues, ou pour la classification qui est une tâche consistant à choisir une classe (valeur) parmi toutes celles possibles.

4.2. Apprentissage non supervisé

À l'opposé de l'apprentissage supervisé, cette approche utilise des données non étiquetées pour l'entraînement. Les algorithmes visent à découvrir des structures ou des motifs cachés au sein des données, tels que le regroupement (clustering) ou la réduction de la dimensionnalité, La **figure 04** présente un exemple simple d'apprentissage non supervisé

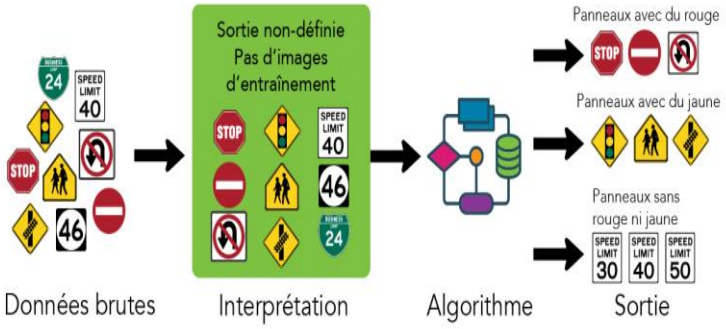

Figure 04 Exemple de l'apprentissage non supervisé

4.3. Apprentissage par renforcement

Dans ce modèle, Un agent apprend à effectuer des choix en interagissant avec un environnement. Il reçoit des récompenses ou des sanctions en fonction de ses actions comme il est illustré sur la **figure 05**, avec comme finalité la maximisation de la somme des récompenses accumulées au fil du temps.

Chacune de ces méthodologies dispose de ses propres algorithmes et techniques, adaptés aux problématiques qu'elles cherchent à résoudre. Par exemple, dans le cadre de l'apprentissage supervisé, des algorithmes tels que les réseaux de neurones, les machines à vecteurs de support, et les arbres de décision sont couramment utilisés. Les principaux défis

résident dans la sélection de l'algorithme approprié, le réglage des paramètres, ainsi que dans l'évaluation de la performance des modèles via des méthodes de test et de validation, afin d'assurer leur capacité à se généraliser à de nouvelles données. [2]

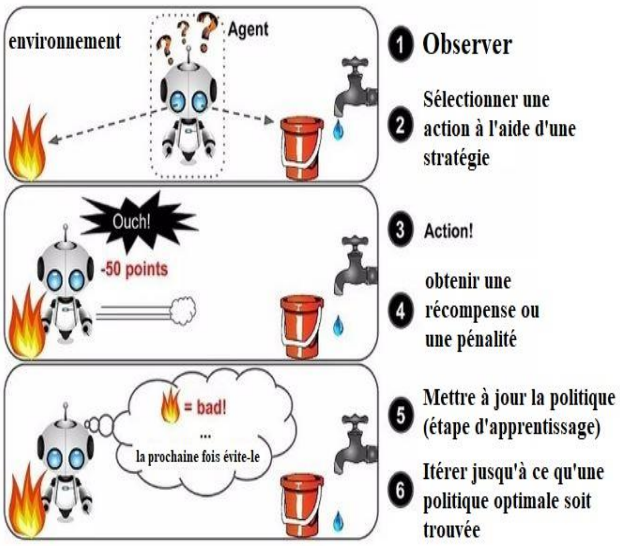

Figure 05: Exemple de l'apprentissage par renforcement

5. Applications de l'apprentissage automatique dans le monde réel

5.1. L'apprentissage automatique dans les finances

L'industrie financière a largement adopté l'apprentissage automatique pour l'analyse de données et la prédiction de tendances. Les algorithmes d'apprentissage automatique sont utilisés pour prévoir les fluctuations des marchés boursiers, évaluer les risques de crédit, détecter la fraude financière et optimiser les portefeuilles d'investissement.

5.2. L'apprentissage automatique dans la santé

Dans le domaine de la santé, l'apprentissage automatique a révolutionné la médecine en aidant les professionnels de la santé à diagnostiquer des maladies, à prédire les résultats des traitements et à détecter les anomalies dans les images médicales. L'analyse des données médicales massives permet également de découvrir de nouveaux médicaments et de personnaliser les traitements.

5.3. L'apprentissage automatique dans les véhicules autonomes

Les véhicules autonomes sont devenus une réalité grâce à l'apprentissage automatique. Les systèmes de conduite automatisée utilisent des capteurs, des caméras et des radars pour recueillir des données en temps réel sur l'environnement et les véhicules autour d'eux. Ces données sont ensuite traitées par des algorithmes d'apprentissage automatique pour prendre des décisions en temps réel, assurant ainsi une conduite sûre et autonome.

5.4. L'apprentissage automatique dans le commerce électronique

Les entreprises de commerce électronique exploitent l'apprentissage automatique pour personnaliser les recommandations de produits, améliorer les systèmes de recommandation, optimiser les prix en fonction de la demande et détecter les comportements frauduleux des utilisateurs.

5.5. L'apprentissage automatique dans la gestion des ressources humaines

L'apprentissage automatique est utilisé pour optimiser les processus de recrutement, en analysant les candidatures et en prédisant les candidats les mieux adaptés à un poste. De plus, il peut être employé pour prédire les départs de collaborateurs et améliorer la rétention des talents.

6. Conclusion :

En résumé, ce chapitre nous a permis d'explorer l'univers de l'apprentissage automatique. Nous avons commencé par définir ce qu'est l'apprentissage automatique. Ensuite, nous avons parcouru un bref historique pour comprendre l'évolution de cette technologie et comment elle a transformé divers secteurs au fil du temps.

Nous avons ensuite catégorisé les différents types d'apprentissage automatique, en expliquant les particularités de chaque catégorie.

Enfin, nous avons examiné les multiples applications de l'apprentissage automatique dans le monde réel. Que ce soit dans les domaines de la santé, ...

Nous savons maintenant quelles sont les stratégies nécessaires pour appliquer ces techniques au modèle prédictif. De plus, en regardant les cas d'utilisation du Machine Learning, nous pouvons obtenir de l'inspiration de notre travail et extraire les meilleures pratiques à prendre pour le traitement des données de vente, l'amélioration des modèles de prévision et l'optimisation des performances de nos prévisions dans l'industrie pharmaceutique.

Le chapitre suivant sera « Vente de médicaments et caractéristiques des médicaments », où nous serons capables de saisir le domaine d'activité que nous souhaiterions préciser.

Chapitre II : Les ventes de médicaments et leurs particularités

1. Introduction :

Les médicaments jouent un rôle essentiel dans le domaine de la santé en soignant et en guérissant les patients. Le système de distribution de médicaments est un processus dynamique complexe qui est profondément façonné par de nombreuses variables économiques, sociales et réglementaires. Pour ce faire, nous allons essayer de parler des nombreuses particularités des ventes de médicaments, de la classification des médicaments et du concept de génériques, ainsi que des réalités spécifiques du marché algérien, puisque tous ces éléments sont nécessaires pour établir le cadre dans lequel nous allons construire de bons modèles de prédiction via l'apprentissage automatique.

2. Nature des Ventes de Médicaments :

2.1. Saisonnière et Cyclique :

Les ventes de médicaments sont plutôt saisonnières. Par exemple, certaines ventes d'analgésiques sont susceptibles d'être plus élevées en hiver en raison de l'apparition d'un rhume ou de la grippe, tandis que des antihistaminiques se vendront probablement bien au printemps en raison des allergies. Un autre exemple est qu'un certain groupe de médicaments peut suivre un cycle de ventes en fonction de l'apparition d'une épidémie ou d'une pandémie.

2.2. Politique de Santé et Réglementation :

Les réglementations strictes en matière de santé publique et les politiques de remboursement sont les déterminants les plus importants des ventes de médicaments. Les décisions du gouvernement concernant les politiques de remboursement, l'approbation de nouveaux médicaments et les campagnes de vaccination affectent directement les tendances de demande.

2.3. Innovation et Demande :

L'innovation dans le domaine pharmaceutique et médical est le facteur principal influençant la variation des ventes. Une nouvelle drogue, une nouvelle thérapie révolutionnaire ou toute percée dans la recherche scientifique peut déclencher des poussées de demande. De plus, la fin des brevets de médicaments entraîne des fluctuations significatives des ventes, car la durée de vie limitée des brevets provoque des sauts importants dans les ventes de médicaments génériques.

2.4. Influence des Médias et des Campagnes de Marketing :

Les campagnes de marketing, la publicité et la couverture médiatique ont un effet énorme sur les ventes de médicaments. Une bonne publicité ou une bonne couverture médiatique peut faire monter les ventes en flèche, tandis que les commentaires négatifs ou les mises en garde de rappel peuvent les faire plonger.

3. Catégorisation et Généricité des Médicaments :

3.1. Catégorisation des Médicaments :

Les médicaments peuvent être catégorisés de plusieurs façons, que ce soit par le mécanisme qu'ils utilisent, leur application thérapeutique, leur composition chimique ou leur statut réglementaire. Les catégories prépondérantes sont :

- **Antibiotiques** : Médicaments utilisés pour traiter les infections bactériennes.
- **Antiviraux** : Médicaments prescrits pour traiter les infections virales.
- **Antifongiques** : Médicaments utilisés contre les infections fongiques.
- **Antihistaminiques** : Utilisés dans le traitement des allergies.
- **Analgésiques** : Utilisés pour l'analgésie.

Ce sont les catégories les plus spécifiques qui non seulement aident à comprendre les ventes catégorielles, mais aident également à prédire les tendances associées à des types particuliers de médicaments.

3.2. Médicaments Génériques :

Ce sont des médicaments qui sont génériques, tout comme les médicaments de marque, mais les brevets sont arrivés à expiration. Ils ont la même force, la même dose et la même forme galénique que les médicaments de marque. Parfois, la forme galénique générique d'un médicament est moins chère, ce qui peut avoir un impact sur le comportement des consommateurs et même les politiques d'approvisionnement des organismes de santé.

Attributs des Médicaments Génériques :

- **Prix Bas** : Les médicaments génériques sont peu coûteux par rapport aux médicaments de marque, ce qui les rend plus susceptibles d'être utilisés et de fonctionner sur le marché.
- **Interchangeabilité** : Les médicaments génériques sont interchangeables par rapport aux médicaments de marque et n'entraînent aucune perte d'efficacité d'après la loi locale.
- **Effet sur les Ventes de Médicaments de Marque** : L'un des effets indésirables de la commercialisation des médicaments génériques est qu'elle réduit considérablement les ventes des médicaments de marque correspondants.

4. Attribution de Groupe de Médicaments :

Les groupes de médicaments peuvent être attribués en termes de classe thérapeutique, ce qui est donc plus pratique pour surveiller et analyser les ventes de médicaments. Cela permet de :

- **Améliorer les Stratégies de Marketing :** En mettant en évidence les bons groupes de médicaments.

- **Améliorer la Précision des Modèles de Prédiction :** En trouvant les caractéristiques communes des groupes.

- **Faciliter la Surveillance Réglementaire :** En facilitant une surveillance plus précise des groupes de médicaments.

5. Avantages financiers du marché algérien des médicaments :

Le marché algérien des médicaments est façonné par des opportunités et des défis particuliers qui affectent l'avantage financier des entreprises pharmaceutiques. Les variables telles que les moteurs économiques, les politiques de santé publique et la préférence des consommateurs déterminent le comportement du marché. [3]

La **figure 06** montre la croissance attendue du marché pharmaceutique en Algérie dans les années prochain par rapport au marché africain.

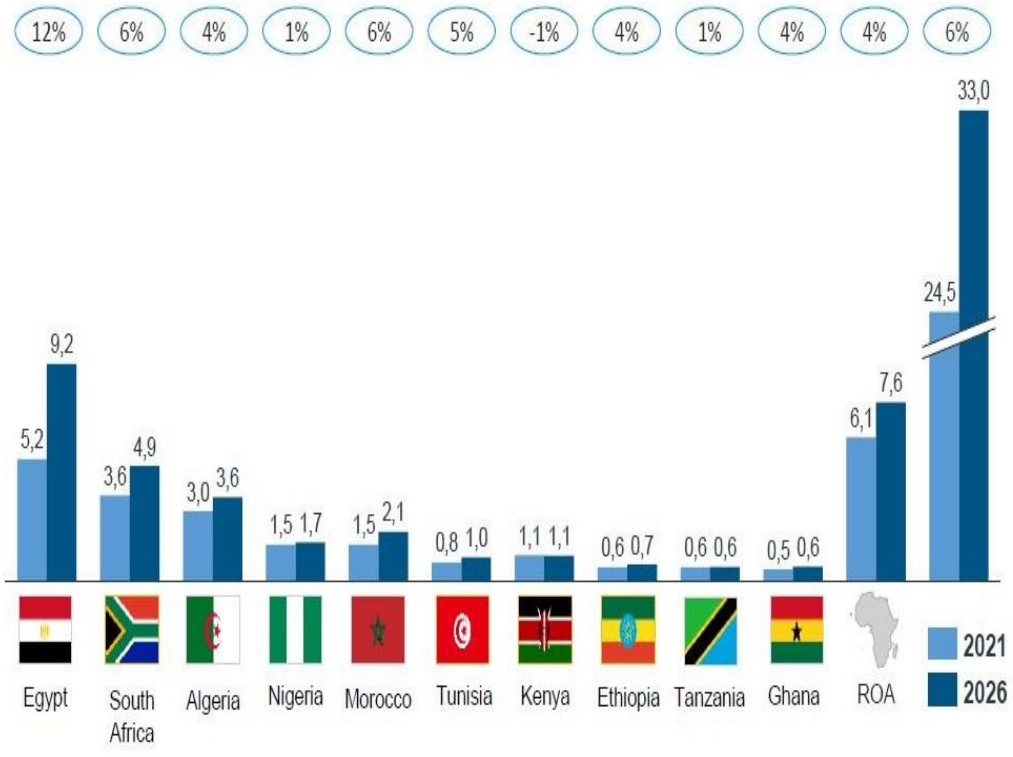

Figure 06: Demande du marché pharmaceutique en Afrique par pays

En cite les avantages suivants :

5.1. Fixation des prix :

La fixation des prix des médicaments est très réglementée par le gouvernement, ce qui peut réduire les marges bénéficiaires des entreprises. D'un autre côté, les réglementations assurent que les médicaments sont abordables pour un grand nombre de consommateurs, augmentant ainsi le volume des ventes.

5.2. Tendance Croissante des Médicaments Génériques :

La tendance croissante des médicaments génériques en Algérie est causée par le fait qu'ils sont moins chers que les médicaments de marque. Les médicaments génériques offrent une opportunité de gain pécuniaire énorme car ils permettent aux entreprises de satisfaire une demande énorme tout en maintenant les coûts de fonctionnement le plus bas possible.

5.3. Investissements et Innovation :

C'est une grande récompense pour les entreprises qui investissent dans le développement de nouvelles drogues et l'innovation. Le marché de l'Algérie est très affamé de produits innovants répondant à des besoins de santé spécifiques, par exemple les besoins en solutions thérapeutiques pour les maladies chroniques.

6. Etat de l’art :

La revue systématique de la littérature concernant les méthodes de prédiction des ventes dans le secteur pharmaceutique met en exergue l’évolution et la diversité des approches analytiques employées. Les techniques traditionnelles, comme la régression linéaire et l’analyse des séries temporelles, ont été longtemps appréciées pour leur simplicité d’application et leur efficacité dans le traitement de données stables. Néanmoins, elles montrent leurs limites face à des jeux de données caractérisés par des non-linéarités ou des motifs complexes.

À 1’ère contemporaine, l’intérêt s’est porté vers l’adoption de méthodes analytiques avancées, incluant l’apprentissage automatique et les modèles basés sur les réseaux de neurones, afin de pallier les insuffisances des approches conventionnelles. L’étude menée par Dutta et al. (2022) examine l’application de divers algorithmes d’apprentissage automatique pour la prédiction des ventes pharmaceutiques, soulignant une performance optimale de la régression linéaire dans leur cas spécifique. Meng et al. (2021), quant à eux, évaluent l’efficacité comparative du modèle Prophet et des réseaux de neurones LSTM, avec une préférence marquée pour la précision supérieure offerte par les LSTM dans la prévision des ventes de médicaments.

L'intégration de variables promotionnelles dans un modèle SVR par Liu et al. (2021) représente une autre innovation notable, offrant une amélioration significative des capacités prédictives par rapport aux modèles traditionnels de séries temporelles. Ces contributions académiques témoignent de la transition des méthodes de prévision de ventes dans l'industrie pharmaceutique, depuis les techniques statistiques basiques vers des modèles d'apprentissage automatique plus élaborés, capables de traiter des données d'une complexité accrue et d'affiner la précision des prévisions.

7. Conclusion :

Dans ce chapitre, nous avons étudié la nature des ventes de médicaments en termes de sauts saisonniers et cycliques, d'incertitude due à la règlementation et aux politiques de santé, d'innovation et de dépendance à la demande, et de la publicité et de la commercialisation. Nous avons ensuite examiné ce que sont les médicaments génériques et comment diviser les médicaments, suivis de l'analyse des spécificités du marché algérien des médicaments. Finalement nous avons discuté des approches actuelles et des modèles utilisés dans ce domaine.

Ce sont les dynamiques et les caractéristiques qu'il faut bien comprendre pour des modèles de prédiction des ventes robustes et efficaces que nous allons présenter dans le chapitre suivant.

Chapitre III : Modèle proposé

1. Introduction

L'un des défis majeurs de la prédiction des ventes de médicaments consiste à trouver des modèles d'apprentissage automatique capables de fournir des prévisions précises. Pour atteindre cet objectif, il est essentiel de préparer soigneusement les données d'entrée, en traitant les problèmes courants tels que les données manquantes, les attributs catégoriels et textuels, et les échelles différentes des attributs numériques.

Ce chapitre présente les motivations et objectifs de l'étude les méthodes de prétraitement des données, et les trois modèles d'apprentissage automatique sélectionnés : LSTM, MLP, et régression polynomiale.

L'objectif est de développer des outils prédictifs robustes pour améliorer la prédiction des ventes de médicaments et répondre efficacement aux besoins du marché pharmaceutique en Algérie.

2. Objectif et motivation

L'objectif de ce mémoire est de développer des modèles prédictifs basés sur l'apprentissage automatique pour estimer les ventes de médicaments en Algérie. La dynamique et l'importance du marché pharmaceutique algérien, marqué par une croissance rapide et des besoins croissants en matière de santé, nécessitent des outils de prévision efficaces pour assurer une gestion optimale des stocks et des ressources. En réponse à ces défis, ce travail propose l'application de trois modèles d'apprentissage automatique : les réseaux de neurones à mémoire à long terme (LSTM), les perceptrons multicouches (MLP) et la régression polynomiale.

Dans le contexte spécifique de notre étude, les données disponibles présentent certaines limitations, notamment en termes de paramètres et de variables, en se basant principalement sur les quantités historiques de ventes. Malgré ces contraintes, l'application des modèles LSTM, MLP et de régression polynomiale permet d'explorer différentes approches pour capturer les tendances et les motifs complexes présents dans les données historiques. La motivation derrière cette étude est de fournir aux acteurs du secteur pharmaceutique des outils avancés pour anticiper les fluctuations du marché, optimiser les chaînes d'approvisionnement et répondre de manière proactive aux besoins des patients.

3. Collection, Sélection, Préparation et normalisation des données :

3.1. Collection des données

Nous nous sommes basés sur une base de données d'une pharmacie active à Constantine, contenant plus de 8000 références de produits et plus de 4 millions de lignes de ventes, allant de l'année 2002 jusqu'à 2023. Cela nous permet d'entraîner et d'évaluer les modèles proposés de manière efficace.

Plusieurs autres facteurs peuvent influencer les ventes de médicaments. Parmi eux, on trouve la saisonnalité, les campagnes publicitaires, les tendances des maladies, les changements réglementaires, et les prix des concurrents. Cependant, après une analyse approfondie des données de notre base de données (BDD), nous avons constaté que seule la quantité des ventes est significative pour notre cas spécifique. En se concentrant sur cette variable clé, nous essayons d'améliorer la précision de nos prévisions.

3.2. Sélection des données :

Nous avons sélectionné les médicaments qui figurent en tête des ventes par classe thérapeutique afin de mettre en œuvre nos algorithmes. De plus, comme chaque classe thérapeutique est représentative d'un groupe de médicaments, le passage à la généralisation devrait être évident. Cependant, l'entraînement des modèles a été effectué grâce à une analyse approfondie de la base de données des ventes.

Les résultats de sélection se figure sur le tableau suivant :

N°	Code	Médicament représentant la classe thérapeutique	Nombre des ventes
1	03B005	Paracétamol 500MG – Comprimé – Boite – 20	99 007
2	03B081	Paracétamol 1000MG – Comprimé – Boite – 10	41 783
3	10A001	Oméprazole – 20MG – Boite – 14	36 826
4	13G043	Amoxicilline – 250MG – 5ML – Flacon – 60ML	25 471
5	21A004	Diclofénac 1G/100G – gélule – Tube – 50G	24 716

Tableau 01 : Résultat de sélection des produits

3.3. Préparation et normalisation des données :

La préparation des données est une étape importante pour les modèles d'apprentissage automatique car elle détermine directement la qualité et la précision des prédictions du modèle. Des données bien préparées, nettoyées et normalisées permettent d'éviter les biais, de réduire le bruit et d'améliorer la performance globale du modèle.

Dans notre cas la préparation des données était basée sur les points suivants :

a- Nettoyage des données :

- **Traitement des valeurs manquantes** : Pour remplir les valeurs manquantes, l'une des méthodes bien connues est la moyenne mobile. Il s'agit de remplacer les valeurs manquantes par la moyenne des valeurs entourant afin de lisser les données tout en conservant la cohérence et la tendance des données.

- **Gestion des outliers (aberrantes) :** Les anomalies sont des valeurs qui s'écartent significativement du reste des données et peuvent fausser les résultats de l'analyse. On a utilisé Une méthode courante pour la détection et le retraitement des valeurs aberrantes l'Écart Interquartile (IE).

b- Normalisation des données :

Dans notre cas nous avons utilisé **Min-Max Scaling** qui nous permettant des comparaisons uniformes, et transforme les valeurs de sorte qu'elles se situent entre 0 et 1.

4. Modèles à implémenter :

4.1. Modèle 1 : LSTM (Long Short-Term Memory)

Pour notre premier modèle, nous allons utiliser les réseaux de neurones à mémoire à long terme (LSTM), une variante des réseaux de neurones récurrents (RNN) particulièrement efficace pour traiter les données séquentielles et les dépendances temporelles à long terme. Les LSTM ont été développés pour surmonter les limitations des RNN traditionnels, notamment le problème de la disparition du gradient, qui rend difficile l'apprentissage des dépendances temporelles lointaines.

Les LSTM possèdent une structure unique représenter par la **figure 07,** de cellules mémoire et de portes (porte d'entrée, porte d'oubli et porte de sortie) qui régulent le flux d'informations à travers le réseau, permettant ainsi de conserver et de rappeler des informations sur de longues périodes de temps. Cette architecture permet aux LSTM de modéliser efficacement des séries temporelles complexes et des motifs non linéaires présents dans les données. [4]

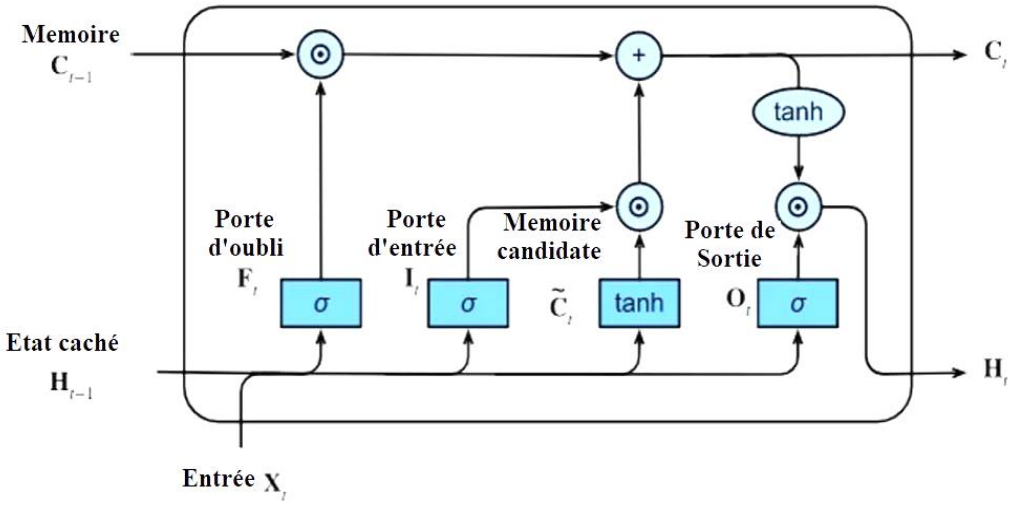

Figure 07: Cellule d'un réseau LSTM

En appliquant les LSTM à la prédiction des ventes de médicaments, nous cherchons à capturer les tendances saisonnières, les effets promotionnels, et d'autres variations temporelles importantes. Les LSTM sont particulièrement adaptés à cette tâche en raison de leur capacité à gérer les séquences de données longues et à apprendre les relations temporelles complexes sans nécessiter une ingénierie manuelle extensive des caractéristiques.

4.2. Modèle 2 : MLP (MultiLayer Perceptron)

Pour ce deuxième modèle, nous allons utiliser les perceptrons multicouches (MLP), qui sont des réseaux de neurones artificiels constitués de plusieurs couches de neurones entièrement connectées comme le montre la **figure 08**. Les MLP sont parmi les architectures les plus fondamentales et les plus polyvalentes en apprentissage automatique supervisé, capables de modéliser des relations complexes entre les variables d'entrée et de sortie.

Les MLP sont composés d'une couche d'entrée, de plusieurs couches cachées et d'une couche de sortie. Chaque neurone de ces couches est connecté à tous les neurones de la couche précédente, et les connexions sont pondérées par des coefficients appris durant le processus d'entraînement. L'entraînement des MLP se fait généralement par rétro propagation, une méthode qui ajuste les poids en minimisant l'erreur entre les prédictions du modèle et les valeurs réelles à l'aide d'un algorithme de descente de gradient.

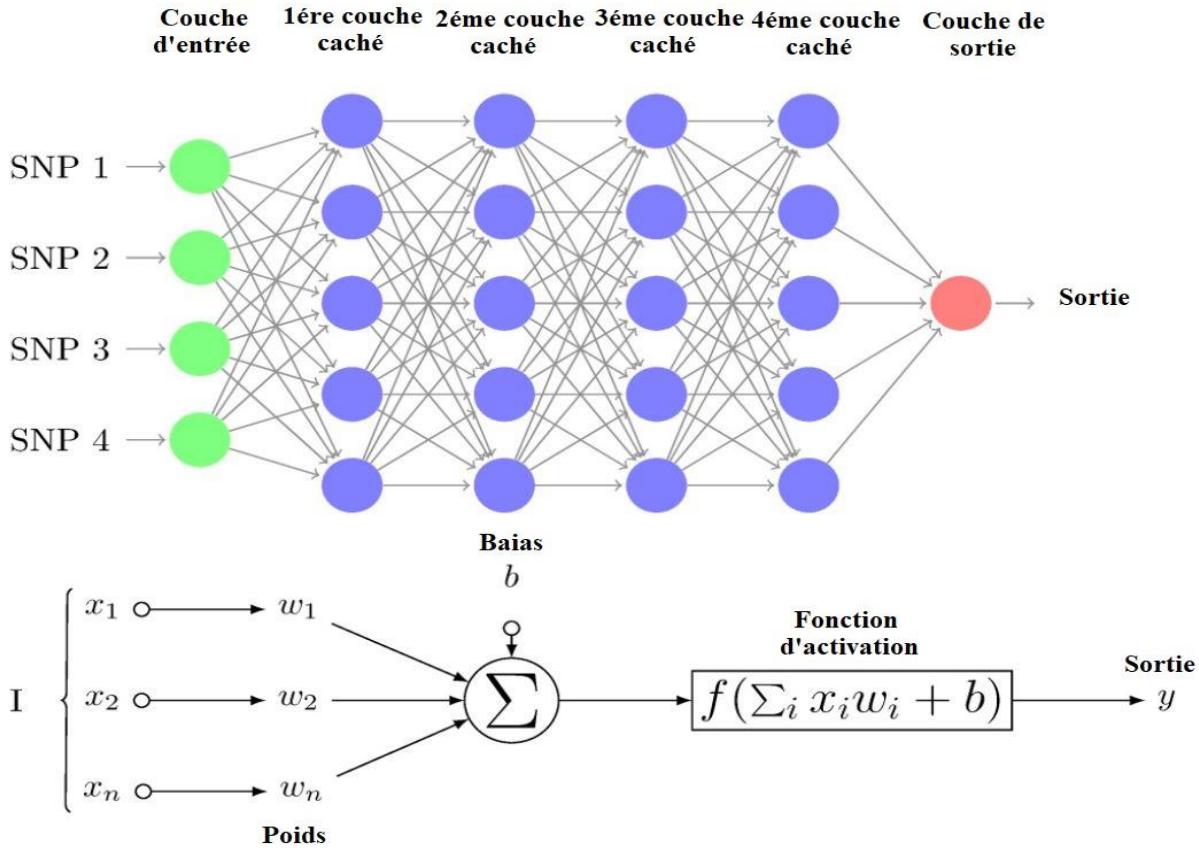

Figure 08: Diagramme d'un perceptron multicouche

L'utilisation des MLP pour la prédiction des ventes de médicaments permet de capturer les interactions non linéaires entre les différentes caractéristiques des données, comme les tendances saisonnières, les effets promotionnels, et d'autres facteurs contextuels. Les MLP sont particulièrement utiles lorsque les relations entre les variables ne sont pas simplement linéaires et nécessitent des transformations complexes pour être correctement modélisées.

En appliquant les MLP à notre étude, nous bénéficions de leur capacité à apprendre directement à partir des données, sans nécessiter une ingénierie manuelle extensive des caractéristiques. Cette approche a été soutenue par diverses études, comme celle de Dutta et al. (2022), qui a démontré l'efficacité des MLP dans divers contextes de prédiction des ventes. En intégrant les MLP dans notre cadre de prévision, nous visons à améliorer la précision et la robustesse des prédictions des ventes de médicaments en Algérie, en exploitant pleinement la capacité des MLP à modéliser des relations complexes et non linéaires dans les données historiques. [5]

4.3. Modèle 3 : Régression Polynomiale

Pour ce troisième modèle, nous allons utiliser la régression polynomiale, une extension de la régression linéaire qui permet de modéliser des relations non linéaires entre les variables indépendantes et dépendantes. Contrairement à la régression linéaire, qui cherche à ajuster une ligne droite aux données, la régression polynomiale ajuste une courbe polynomiale, offrant ainsi une plus grande flexibilité pour capturer des motifs complexes dans les données.

La régression polynomiale fonctionne en introduisant des termes supplémentaires (les puissances des variables indépendantes) dans le modèle de régression, permettant ainsi de capturer les effets des interactions non linéaires. Par exemple, un modèle de régression polynomiale de degré 2 inclura non seulement les termes linéaires mais aussi les carrés des variables indépendantes, permettant de modéliser des relations quadratiques.

En appliquant la régression polynomiale à la prédiction des ventes de médicaments, nous cherchons à capturer des tendances non linéaires dans les données historiques, telles que les variations saisonnières et les effets des campagnes promotionnelles. Cette méthode est particulièrement utile lorsque les données montrent des relations courbées plutôt que des lignes droites, ce qui est souvent le cas dans les séries temporelles de ventes. [6]

5. Conclusion

Ce chapitre a fourni une structure détaillée pour la prédiction des ventes de médicaments par l'apprentissage automatique, en abordant chaque étape essentielle du processus. Nous avons commencé par une introduction qui a mis en lumière les défis spécifiques du domaine. Ensuite, nous avons exposé les objectifs et les motivations de notre étude, justifiant l'importance de modèles prédictifs précis pour le marché pharmaceutique algérien.

Nous avons également présenté un état de l'art des techniques actuelles de prévision, mettant en évidence les avancées et les limites des approches traditionnelles et modernes. La section suivante a détaillé la collecte, la sélection, la préparation et la normalisation des données, soulignant l'importance d'un prétraitement rigoureux pour garantir la qualité des données d'entrée.

Enfin, nous avons décrit les trois modèles d'apprentissage automatique implémentés : les réseaux de neurones à mémoire à long terme (LSTM), les perceptrons multicouches (MLP) et la régression polynomiale. Chaque modèle a été choisi pour ses capacités uniques à capturer des motifs complexes dans les données de ventes de médicaments.

Chapitre IV : Implémentation

1. Introduction :

Nous abordons dans ce chapitre l'implémentation de trois modèles de prévision LSTM, MLP et la régression polynomiale. La section est consacrée à l'analyse des performances des modèles en termes d'exactitude de la prédiction et de la robustesse des résultats.

En premier temps en va entamer la préparation des données. Ensuite, nous allons introduire et construire un par un tous les modèles. Pour chacun des modèles, nous les testerons sur un sous-ensemble des données correspondant à un médicament spécifique pour évaluer leurs performances. Les résultats que nous obtenons seront calculés et comparés pour enfin décider du modèle qui donnera les meilleures prédictions.

Nous concluons cette section en comparant les performances obtenues par les trois modèles. Basé sur le résultat de la comparaison, on peut facilement comprendre quel est le meilleur des trois modèles qui peut prédire les ventes de médicaments en prenant en considération les détails des données et les exigences du domaine pharmaceutique. Ce chapitre est important dans notre recherche car il discute en réalité des opportunités et des limites des différentes techniques d'apprentissage automatique utilisées dans les prévisions des ventes de médicaments.

2. Préparation de l'environnement de développement :

Pour le développement des modèles souhaités, nous avons choisi Python et PyCharm en raison de leur flexibilité, puissance, et popularité dans la communauté des data scientists et développeurs. Python offre une vaste gamme de bibliothèques spécialisées en machine learning, tandis que PyCharm fournit un environnement de développement intégré (IDE) robuste et convivial. Cette combinaison permet d'accélérer le développement, d'améliorer la productivité et de garantir un code de haute qualité.

La préparation de l'environnement consiste en plusieurs étapes. Dont en voici la liste :

- Téléchargement et installation de la version 3.11 de Python.
- Téléchargement et installation de la version 17.0.10 de PyCharm.
- Installez les bibliothèques essentielles pour la machine learning.

3. Bibliothèques utilisées :

Pour l'implémentation des modèles d'apprentissage automatique, nous avons utilisé plusieurs bibliothèques robustes et largement adoptées dans la communauté scientifique.

3.1. Numpy :

(Numerical Python) est une bibliothèque python open source indispensable et fondamental pour le calcul scientifique, utilisées pour l'analyse et la manipulation des données numérique, Il ajoute à Python de puissantes structures de données qui garantissent des calculs efficaces avec des tableaux et des matrices multidimensionnelles et fournit un énorme nombre de fonctions mathématiques de haut niveau qui opèrent sur ces tableaux et matrices.

Il fournit ndarray, un objet tableau homogène à n dimensions, et prend entièrement en charge une approche orientée objet, permettant au programmeur de coder selon le paradigme qu'il préfère. [7]

3.2. Pandas :

Pandas est un package Python fournissant des structures de données rapides, flexibles et expressives conçues pour faciliter le travail avec des données « relationnelles » ou « étiquetées » à la fois simples et intuitives.

Il vise à constituer l'élément fondamental de haut niveau pour faire analyse de données pratique et réelle en Python. De plus, son objectif plus large est de devenir l'outil le plus puissant, et flexible d'analyse/manipulation de données open source disponible dans n'importe quelle langue.

Pandas est bien adapté à de nombreux types de données :

- Données tabulaires avec des colonnes de type hétérogène, comme dans un tableau SQL ou une feuille de calcul Excel.
- Données de séries chronologiques ordonnées et non ordonnées (pas nécessairement à fréquence fixe).
- Données matricielles arbitraires (typées de manière homogène ou hétérogène) avec des étiquettes de lignes et de colonnes
- Toute autre forme d'ensembles de données observationnelles/statistiques. [8]

3.3. Scikit-learn :

Est une bibliothèque open source polyvalente et robuste pour l'analyse de données écrite en python.

Basé sur d'autres bibliothèques python : NumPy, SciPy et matplotlib, et fournit une sélection d'outils efficaces pour l'apprentissage automatique et la modélisation statistique, notamment la

classification, la régression, le clustering et la réduction de dimensionnalité via une interface de cohérence en Python.

3.4. Tensorflow :

Est une bibliothèque open-source développée par Google pour le calcul numérique et l'apprentissage automatique. Elle nous permet de définir, d'entraîner et d'évaluer des réseaux de neurones.

La version 2.0 de TensorFlow apporte des améliorations significatives en termes de flexibilité et de facilité d'utilisation, notamment en simplifiant l'API et en favorisant l'utilisation de Keras comme interface de haut niveau. Cette version permet aux développeurs de tirer pleinement parti des différentes architectures matérielles telles que les CPU, les GPU et les TPU. TensorFlow 2.0 facilite l'entraînement et l'exécution des modèles sur ces plateformes grâce à une intégration transparente et des optimisations spécifiques. Les utilisateurs peuvent ainsi développer et déployer leurs modèles d'apprentissage automatique avec une performance accrue, que ce soit sur des machines locales ou sur des infrastructures de cloud computing. Cette flexibilité permet une meilleure adaptation aux besoins spécifiques des projets, en maximisant l'efficacité et en réduisant les temps de calcul. [9]

3.5. Keras :

Keras est une bibliothèque de haut niveau pour les réseaux de neurones, écrite en Python, capable de fonctionner avec plusieurs frameworks de calcul comme TensorFlow, Microsoft Cognitive Toolkit (CNTK), et Theano. Conçue pour faciliter l'expérimentation rapide avec des réseaux de neurones profonds, Keras est particulièrement adaptée à la recherche et au développement. Parmi ses caractéristiques clés, on trouve :

- Interface simple et cohérente.
- Flexibilité et modularité.
- Support de plusieurs backends.
- Compatibilité avec TensorFlow.
- Bénéficie d'une large communauté de développeurs.

3.6. Matplotlib :

Matplotlib est une bibliothèque permettant de créer des tracés 2D de tableaux en Python. Bien que Matplotlib soit écrit principalement en Python pur, il fait un usage intensif de NumPy et autre code d'extension pour fournir de bonnes performances même pour les grands tableaux.

Matplotlib est conçu pour être capable de créer des tracés simples avec seulement quelques lignes de commandes.

3.7. Firebird:

Firebird est un package de bibliothèque Python open source qui implémente les bases de données relationnelle Firebird.

En plus de l'ensemble minimal de fonctionnalités de l'API Python DB standard, FDB expose également l'intégralité de l'API client native (ancien style) du moteur de base de données et le nombre d'extensions et d'améliorations supplémentaires pour une utilisation pratique de Firebird.

3.8. Installation

```
c:\Users\bilal>pip install pandas numpy tensorflow keras matplotlib fdb
c:\Users\bilal>
```

4. Modèle général d'un algorithme basé sur tensorflow :

La **figure 09** montre un exemple commenté de la création et de l'entraînement d'un réseau de neurones multicouches (MLP) en utilisant TensorFlow et Keras pour simplifier la compréhension des modèles que nous allons proposer, surtout pour quelqu'un qui n'est pas familier avec les bibliothèques précédentes.

Explication des étapes :

1. Importation des bibliothèques nécessaires :

   - tensorflow et keras pour la construction et l'entraînement du modèle.

   - numpy pour la création et la manipulation des données.

   - scikit-learn pour la normalisation des données et la division en ensembles d'entraînement et de test.

2. Préparation des données :

   - Création de données fictives pour l'exemple.

   - Division des données en ensembles d'entraînement (80%) et de test (20%).

   - Normalisation des données pour améliorer la convergence du modèle.

3. Construction du modèle :

   - Création d'un modèle séquentiel avec trois couches : une couche d'entrée et deux couches cachées.

   - Utilisation de la fonction d'activation ReLU pour les couches cachées et de la fonction sigmoïde pour la couche de sortie (adaptée à une tâche de classification binaire).

```python
# 1. Importation des bibliothèques nécessaires
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np

# 2. Préparation des données
X, y = np.random.rand(1000, 20), np.random.randint(2, size=1000)

X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 3. Construction du modèle
model = Sequential()

model.add(Dense(64, activation='relu', input_shape=(20,)))

model.add(Dense( units: 32, activation='relu'))

model.add(Dense( units: 1, activation='sigmoid'))

# 4. Compilation du modèle
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

# 6. Évaluation du modèle
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test accuracy: {test_acc}")

# 7. Utilisation du modèle pour des prédictions
new_data = np.random.rand(5, 20)
new_data = scaler.transform(new_data)
predictions = model.predict(new_data)
print(predictions)
```

Figure 09: Squelette d'un algorithme basé sur la librairie tensorflow

4. Compilation du modèle :

- Spécification de l'optimiseur (adam), de la fonction de perte (binary_crossentropy) et des métriques (accuracy).

5. Entraînement du modèle :

- Entraînement du modèle sur les données d'entraînement avec validation sur les données de test.

- Le modèle s'entraîne pendant 20 époques avec une taille de lot de 32.

6. Évaluation du modèle :

- Évaluation de la performance du modèle sur les données de test.

7. Utilisation du modèle pour des prédictions :

- Prédiction sur de nouvelles données normalisées pour voir les résultats.

5. Préparation des données – phase 1 :

Afin d'implémenter et évaluer un modèle d'apprentissage automatique pour la prédiction des ventes, en prépare les données par le traitement des valeurs manquant, ensuite en ajuste les outliers (aberrantes), est à la fin regroupe les ventes hebdomadaires.

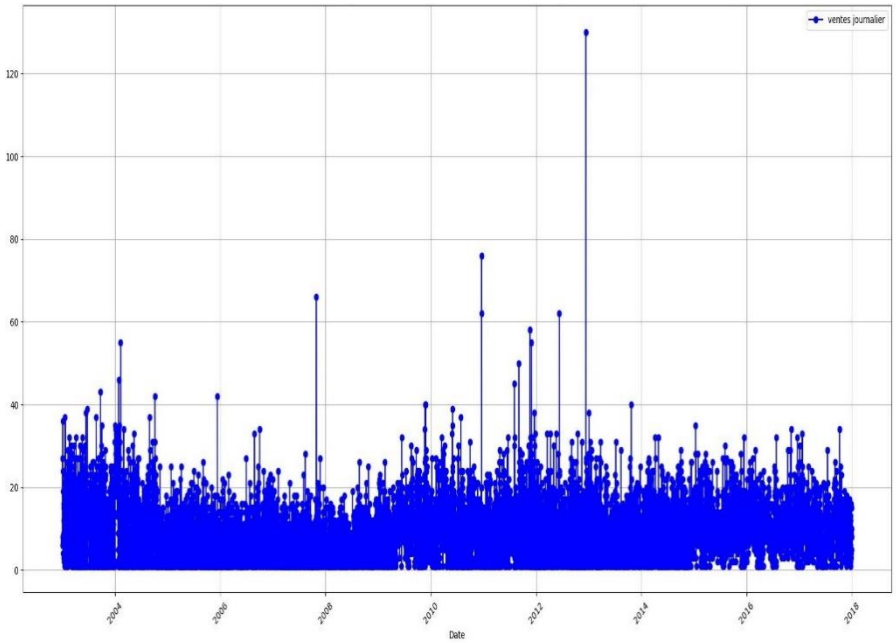

Figure 10: Représentation des données brute (Cas : Paracétamol 500MG Comprimé)

La **figure 10** présente les données brutes pour le paracétamol 500 mg sous forme de comprimés. Sur l'axe des X, les dates sont échelonnées, couvrant une période allant de janvier 2004 à décembre 2018. L'axe des Y montre les quantités de comprimés vendus par jours, mesurées en unités.

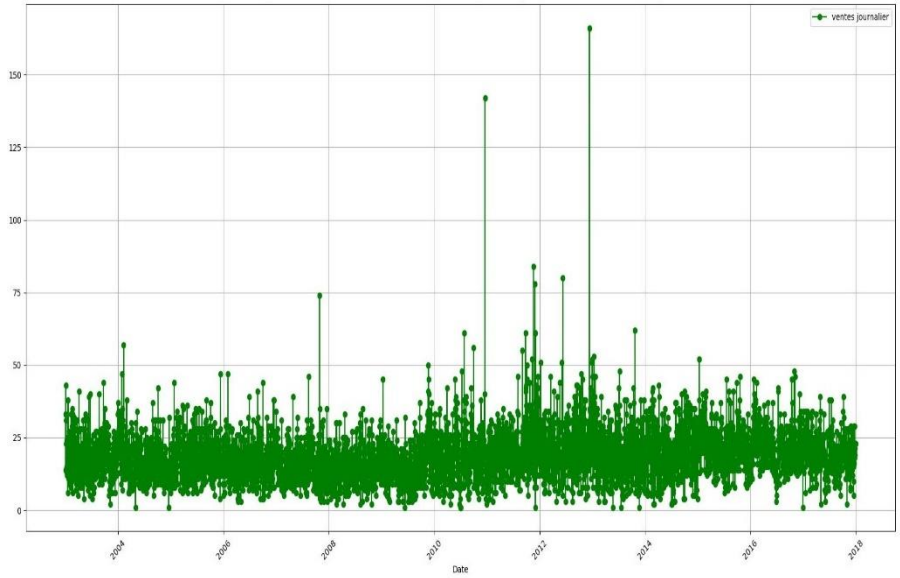

Figure 11: Représentation des données après traitement des valeurs manquants (Cas : Paracétamol 500MG Comprimé)

La **figure 11** présente les données après le traitement des valeurs manquants en utilisant la moyenne mobile pour le paracétamol 500 mg sous forme de comprimés.

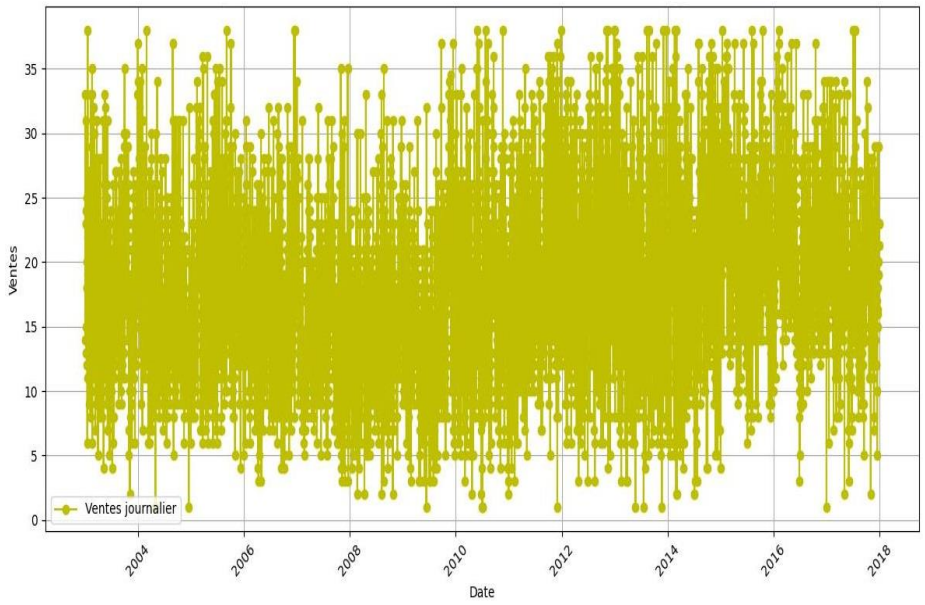

Figure 12: Représentation des données après lissage (Cas : Paracétamol 500MG Comprimé)

La **figure 12** présente les données après le lissage des valeurs aberrantes en utilisant l'écart interquartile (IE) pour le paracétamol 500 mg sous forme de comprimés.

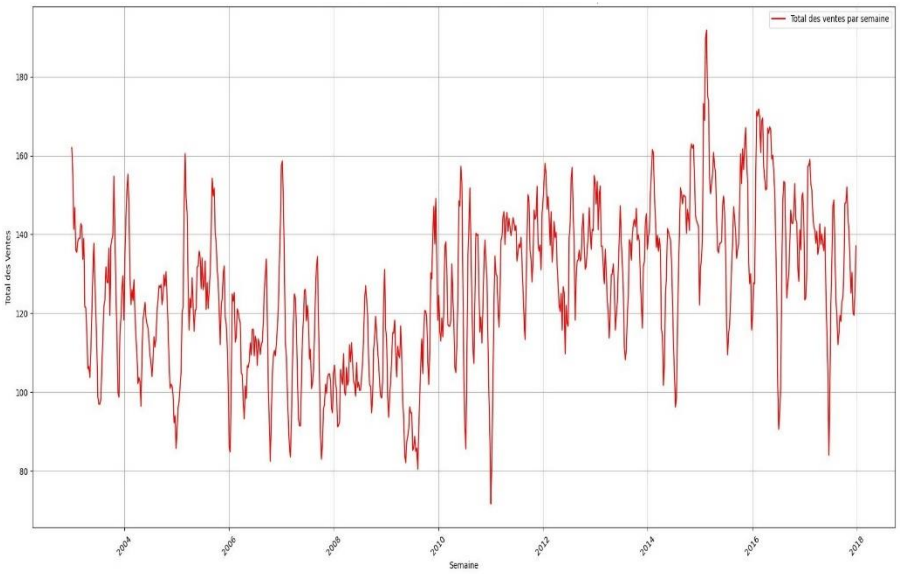

Figure 13: Représentation des données des ventes hebdomadaires (Cas : Paracétamol 500MG Comprimé)

La **figure 13** présente les données après le regroupement des ventes journalier en hebdomadaires pour le paracétamol 500 mg sous forme de comprimés.

6. Métriques d'évaluation :

Ce sont des outils essentiels pour évaluer les performances des modèles [10]

- **Erreur Absolue Moyenne (Mean Absolute Error, MAE) :** mesure la moyenne des erreurs absolues entre les prédictions et les valeurs réelles.

Un modèle avec une MAE plus faible peut être meilleur en raison de sa capacité à capturer des tendances ou des variations importantes.

- **Erreur Quadratique Moyenne (Mean Squared Error, MSE) :** mesure la moyenne des carrés des erreurs entre les prédictions et les valeurs réelles. Elle amplifie les grandes erreurs en les élevant au carré, ce qui peut être utile si de grandes erreurs sont particulièrement indésirables.

Une MSE plus faible indique généralement que le modèle est meilleur pour ajuster les données.

- **Erreur Quadratique Moyenne Racine (Root Mean Squared Error, RMSE) :** C'est la racine carrée de la MSE. Elle fournit une mesure de l'erreur qui est dans la même unité que la variable de réponse, ce qui facilite l'interprétation.

Des valeurs plus basses de RMSE indiquent une meilleure précision du modèle. RMSE est souvent préféré lorsque les erreurs de prédiction plus importantes sont plus critiques, car il pénalise davantage les erreurs plus importantes en raison de la racine carrée dans son calcul.

- **Erreur Absolue Moyenne en Pourcentage (Mean Absolute Percentage Error, MAPE) :** mesure la moyenne des erreurs absolues en pourcentage des valeurs réelles. Elle est utile pour comprendre l'erreur moyenne en termes relatifs, particulièrement dans des contextes où les échelles des données varient.

Une MAPE plus faible indique une meilleure précision du modèle.

- **Coefficient de Détermination (R^2) :** mesure la proportion de la variance de la variable dépendante qui est prédite par le modèle. Il varie entre 0 et 1, où 1 indique une parfaite correspondance entre les prédictions et les valeurs réelles.

7. Création, compilation et entrainement des modèles :

Dans ce qui suit nous allons créer les trois modèles, chacun d'eux sera entrainé puis tester sur les collections d'entrainement et de test respectivement.

Afin d'évaluer la précision des trois modèles, nous calculerons différentes mesures de performances après l'entrainement pour qu'à la fin nous puissions les comparer.

7.1. Préparation des données – phase 2 :

Avant de créer, compiler et entrainer chaque modèle on réalise une 2éme phase de préparation qui consiste à :

7.1.1. Définir la longueur de la séquence d'entrée :

A l'aide de l'instruction « **sequence_length** » ont déterminé la longueur des séquences d'entrée qui seront utilisées pour entraîner le modèle.

- **LSTM :** sequence_length =12
- **MLP :** sequence_length = 10
- **Régression polynomiale :** sequence_length = 10

7.1.2. Création des séquences :

Pour préparer les données pour un modèle de prédiction séquentielle. Chaque séquence générée contient une série d'éléments à partir desquels le modèle apprend à prédire le prochain élément. Cela est particulièrement utile dans les séries temporelles où la valeur suivante dépend des valeurs précédentes.

La création des séquences est assurée par la fonction « `create_sequences` » :

```
97  def create_sequences(data, sequence_length):
98      sequences = []
99      for i in range(len(data) - sequence_length):
100         sequences.append(data[i:i + sequence_length + 1])
101     return np.array(sequences)
```

Ou :

- Une liste vide `sequences` est initialisée pour stocker les séquences générées.

- La boucle for parcourt les indices de data de 0 à len(data) - sequence_length – 1 ; Pour chaque indice i, un sous-ensemble de data est extrait allant de i à i + sequence_length + 1 (inclusif) ; Ce sous-ensemble, qui contient sequence_length + 1 éléments, est ajouté à la liste sequences.

- La liste `sequences` est convertie en un tableau NumPy est retourné par la fonction.

7.1.3. Division des données :

```
107 train_size = int(len(sequences) * 0.8)
108 val_size = int(len(sequences) * 0.1)
109
110 train_sequences = sequences[:train_size]
111 val_sequences = sequences[train_size:train_size + val_size]
112 test_sequences = sequences[train_size + val_size:]
```

Les séquences sont divisées en trois ensembles : train, val, et test avec des proportions de 80%, 10%, et 10% respectivement.

7.1.4. Préparation des données pour chaque modèle :

```
X_train = train_sequences[:, :-1]
y_train = train_sequences[:, -1]
X_val = val_sequences[:, :-1]
y_val = val_sequences[:, -1]
X_test = test_sequences[:, :-1]
y_test = test_sequences[:, -1]
```

- X_train, X_val, et X_test contiennent les séquences d'entrée (toutes les valeurs sauf la dernière de chaque séquence).
- y_train, y_val, et y_test contiennent les valeurs à prédire (la dernière valeur de chaque séquence).

7.2. Création des modèles :

7.2.1. LSTM (Long Short-Term Memory)

```
model = tf.keras.Sequential([
    tf.keras.layers.LSTM( units: 256, return_sequences=True, input_shape=(sequence_length, 1)),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.LSTM( units: 256, return_sequences=True),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.LSTM( units: 128, return_sequences=True),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.LSTM(128),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense( units: 128, activation='swish'),
    tf.keras.layers.Dense(1)
])
```

- **Définition du modèle séquentiel :** Le modèle `Sequential` est une pile linéaire de couches. Il est approprié pour un modèle où chaque couche a exactement un tenseur d'entrée et un tenseur de sortie.

- **Couches LSTM** :

  - `LSTM(256)` : Une couche LSTM avec 256 unités de mémoire.
  - `return_sequences=True`: Indique que la couche doit renvoyer les séquences complètes au lieu de seulement le dernier état caché.
  - `input_shape=(sequence_length, 1)`: La forme de l'entrée attendue est une séquence de longueur `sequence_length` avec une seule caractéristique par pas de temps.

- **Couches Dropout :** La couche Dropout est une technique de régularisation utilisée dans les réseaux de neurones pour prévenir le surapprentissage (overfitting). Le surapprentissage se

produit lorsque le modèle s'ajuste trop étroitement aux données d'entraînement, ce qui entraîne une mauvaise généralisation aux nouvelles données.

La première couche a un taux de dropout de 10%. Cela signifie que 10% des unités de sortie seront mises à zéro de manière aléatoire pendant l'entraînement pour prévenir le surapprentissage (overfitting).

- **Couches Dense** : (ou couche entièrement connectée) est l'une des couches de base les plus utilisées dans les réseaux de neurones artificiels.

La premier couche Dense entièrement connectée avec 128 unités, Utilise la fonction d'activation `swish`, une activation non linéaire qui est définie comme `x * sigmoid(x)`.

7.2.2. MLP (MultiLayer Perceptron)

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(sequence_length,)),
    tf.keras.layers.Dense( units: 256, activation='swish'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense( units: 256, activation='swish'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense( units: 128, activation='swish'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense( units: 128, activation='swish'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(1)
])
```

Le modèle précédent est composé de couches denses (Dense) entrecoupées de couches Dropout pour régulariser le modèle. Les couches Dense permettent d'apprendre des représentations complexes des données grâce à leurs nombreux neurones et la fonction d'activation swish, tandis que les couches Dropout réduisent le risque de surapprentissage en désactivant aléatoirement une fraction des neurones pendant l'entraînement. La couche finale produit une sortie unique.

- **Couche Flatten :** Cette couche aplatit l'entrée. Si l'entrée est une séquence de dimensions (sequence_length,), elle est transformée en un vecteur de taille sequence_length. Cela est nécessaire pour passer d'une entrée en forme de séquence à une entrée compatible avec les couches Dense suivantes.

7.2.3. Régression polynomiale

```
degree = 2
poly = PolynomialFeatures(degree)
X_train_poly = poly.fit_transform(X_train)
X_val_poly = poly.transform(X_val)
X_test_poly = poly.transform(X_test)
model = Ridge(alpha=1.0)
```

- **Définir le degré du polynôme « 2 » :** pour inclure les termes quadratiques (x^2) ainsi que les termes linéaires (x) et les termes d'interaction dans nos nouvelles caractéristiques.

- **Créer l'objet PolynomialFeatures** : Cet objet va être utilisé pour transformer les caractéristiques d'entrée en leurs versions polynomiales.

- **Transformer les ensembles de données d'entraînement, de validation et de test** : pour ajuster le modèle PolynomialFeatures aux données d'entraînement X_train et les transformer en même temps. Cela crée de nouvelles caractéristiques basées sur les termes polynomiaux.
Est aussi transformer les ensembles de validation X_val et de test X_test en utilisant les mêmes paramètres ajustés aux données d'entraînement.

- **Créer un modèle de régression Ridge** : Ridge est un modèle de régression linéaire avec une régularisation L2. Le paramètre alpha contrôle la force de la régularisation. Un alpha plus grand signifie plus de régularisation.

7.3. Compilation et Entrainement des modèles :

7.3.1. LSTM (Long Short-Term Memory)

```
169  model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0005), loss='mean_squared_error')
170  early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
171  history = model.fit(X_train, y_train, epochs=50, batch_size=2, validation_data=(X_val, y_val), callbacks=[early_stopping])
172  predictions = model.predict(X_test)
```

- Compilation du modèle avec l'algorithme d'optimisation « Adam », et « 0.0005 » qui représente le taux d'apprentissage, qui détermine la taille des étapes que l'optimiseur prend pour minimiser la fonction de perte. Ainsi de mean_squared_error (erreur quadratique moyenne) qui représente la fonction de perte, Elle calcule la moyenne des carrés des différences entre les prédictions et les vraies valeurs cibles. Cette fonction de perte pénalise les grandes erreurs de manière plus sévère que les petites erreurs.

- **EarlyStopping** : Pour surveiller la perte sur les données de validation, arrêtez l'entraînement si la perte de validation ne s'améliore pas pendant 10 époques consécutives. Cette technique aide à prévenir le surapprentissage et restaure les poids du modèle à l'état où la perte de validation était la plus basse. Ainsi, vous êtes assuré d'obtenir le meilleur modèle observé durant l'entraînement.

Entraînement du Modèle :

- X_train et y_train sont les données d'entraînement et les étiquettes correspondantes.

- epochs=50 fixe le nombre maximum d'époques (passes complètes sur les données d'entraînement). Cependant, l'entraînement peut s'arrêter plus tôt si la condition d'Early Stopping est satisfaite.
- batch_size=2 fixe la taille du lot (nombre d'échantillons utilisés pour mettre à jour les poids). Un petit batch size comme 2 est inhabituelle et peut ralentir l'entraînement, mais peut également permettre une meilleure convergence dans certains cas.
- validation_data=(X_val, y_val) utilise les données de validation pour évaluer la perte de validation à la fin de chaque époque.
- callbacks=[early_stopping] spécifie que le callback EarlyStopping doit être utilisé pendant l'entraînement.

- **Prédiction :**

- X_test est le jeu de données de test sur lequel vous voulez effectuer des prédictions.
- « Predictions » contient les prédictions du modèle pour chaque échantillon dans X_test.

7.3.2. MLP (MultiLayer Perceptron)

```
137 model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0005), loss='mean_squared_error')
138 early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
139 history = model.fit(X_train, y_train, epochs=50, batch_size=2, validation_data=(X_val, y_val), callbacks=[early_stopping])
140 predictions = model.predict(X_test)
```

Similaire à celle de LSTM précédant.

7.3.3. Régression polynomiale

```
131 model.fit(X_train_poly, y_train)
132 predictions = model.predict(X_test_poly)
```

- **Entraînement :** La méthode fit ajustée les paramètres du modèle (poids et biais) aux données d'entraînement. La régression Ridge minimisera la fonction de coût avec une pénalité de régularisation L2 sur les poids pour éviter le surapprentissage (overfitting).

- **Prédiction :** « `X_test_poly` » Ce sont les caractéristiques de test transformées en polynômes. Vous avez créé ces caractéristiques en utilisant `PolynomialFeatures` pour les transformer de la même manière que les données d'entraînement.

La méthode predict utilise le modèle entraîné pour prédire les valeurs cibles des données de test. Le modèle appliquera les poids ajustés pendant l'entraînement aux nouvelles données pour générer les prédictions.

8. Résultat obtenu :

8.1. LSTM (Long Short-Term Memory) :

8.1.1. Produit « Paracétamol - 500mg – Comprimé »

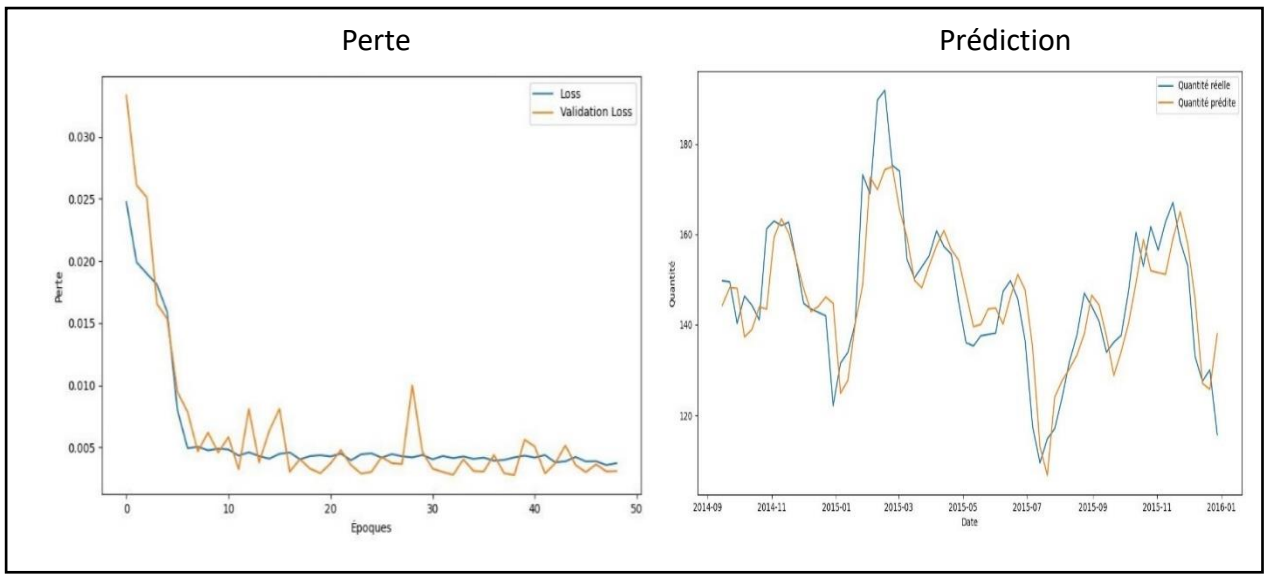

Figure 14 : Résultat d'entrainement - Modèle LSTM - Produit : Paracétamol - 500mg – Comprimé

On analyse les résultats représentés dans la **figure 14**, on peut tirer les conclusions suivantes :

- Initialement, la fonction de perte est élevée, ce qui est attendu car le modèle commence avec des poids aléatoires. Au fur et à mesure de l'entraînement, la perte d'entraînement diminue rapidement, ce qui montre que le modèle ajuste ses poids pour mieux prédire les valeurs cibles.
- La fonction de perte de validation suit une tendance similaire non stable, bien qu'elle soit légèrement plus bas que la perte d'entraînement, ce qui est typique et montre que le modèle généralise raisonnablement bien aux nouvelles données.
- Le modèle montré une capacité raisonnable à capturer les tendances et les motifs présents dans les données pour ce produit, comme en témoignent les prédictions proches des valeurs réelles dans la majorité des cas.

8.1.2. Produit « Paracétamol - 1000mg – Comprimé »

On analyse les résultats représentés dans la **figure 15**, on peut en tirer :

- La fonction de perte de validation ne suit pas une tendance similaire, bien qu'elle soit légèrement plus élèves que la perte d'entraînement.
- Le modèle montré une capacité raisonnable à capturer les tendances et les motifs présents dans les données, mais certaines divergences entre les prédictions et les valeurs réelles ont été observées.

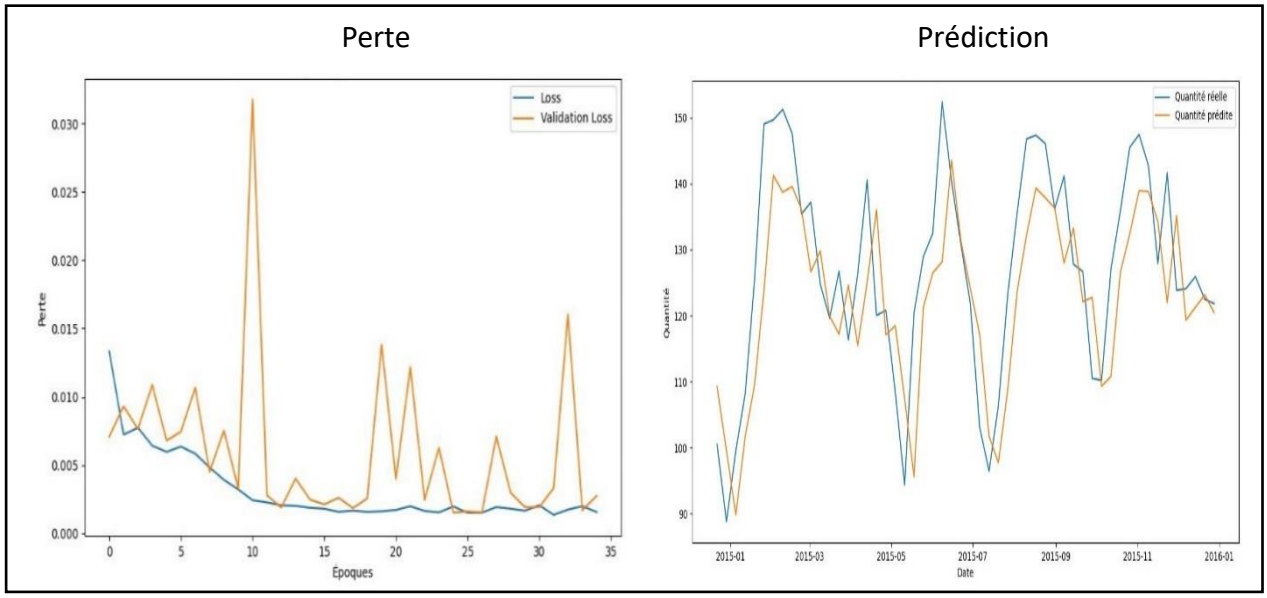

Figure 15 : Résultat d'entrainement - Modèle LSTM - Produit : Paracétamol - 1000mg - Comprimé

- La fonction de perte de validation ne suit pas une tendance similaire, bien qu'elle soit légèrement plus élèves que la perte d'entraînement.
- Le modèle montré une capacité raisonnable à capturer les tendances et les motifs présents dans les données, mais certaines divergences entre les prédictions et les valeurs réelles ont été observées.

8.1.3. Produit « Diclofénac – 1g/100g – gél »

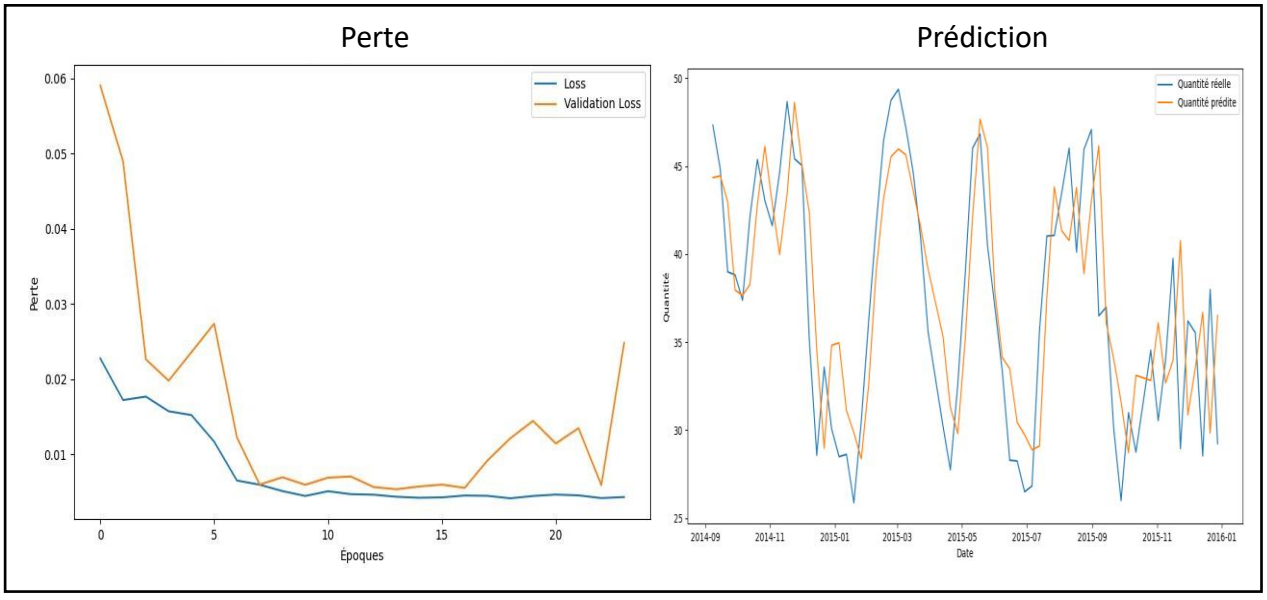

Figure 16 : Résultat d'entrainement - Modèle LSTM - Produit : Diclofénac – 1g/100g – gél

D'après la **figure 16** :

- La fonction de perte de validation remonte à la fin de l'entraînement indiquer que le modèle commence à surapprendre les données d'entraînement.

- Le modèle montré une certaine divergence entre les prédictions et les valeurs réelles à la fin.

8.1.4. Produit « Oméprazole – 20mg »

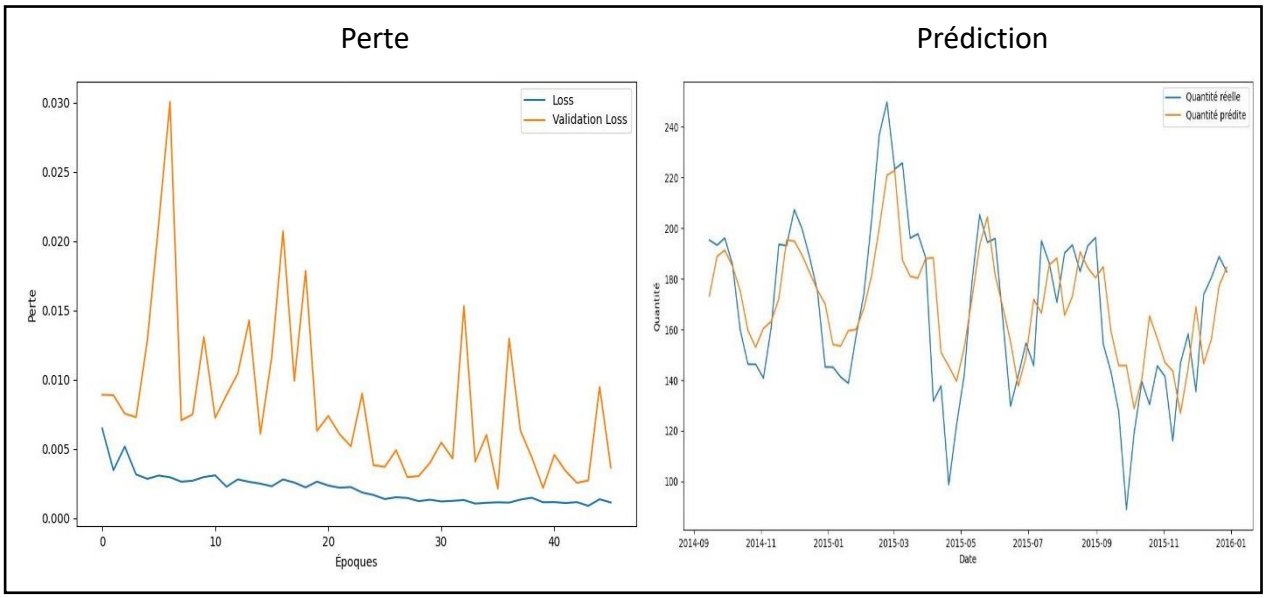

Figure 17 Résultat d'entrainement - Modèle LSTM - Produit : Oméprazole – 20mg

D'après la **figure 17** :

- La fonction de perte de validation s'écarte de la fonction de perte d'entraînement et présente des fluctuations irrégulières.
- Des écarts entre les valeurs prédites et réelles, particulièrement lorsque les valeurs sont très grandes ou très petites. Ce qui montre une sensibilité du modèle aux valeurs extrêmes.

8.1.5. Produit « Amoxicilline – 250mg/5ml – Flacon »

D'après la figure 18 :

- Initialement, la fonction de perte est élevée, ce qui est attendu car le modèle commence avec des poids aléatoires. Au fur et à mesure de l'entraînement, la perte d'entraînement diminue rapidement, ce qui montre que le modèle ajuste ses poids pour mieux prédire les valeurs cibles.
- La fonction perte de validation est inférieure à la fonction de perte, ce qui montre que le modèle s'ajuste trop aux données d'entraînement spécifiques et perd légèrement ces capacités de généralisation sur de nouvelles données.
- Le modèle montré une capacité raisonnable à capturer les tendances et les motifs présents dans les données, mais certaines divergences entre les prédictions et les valeurs réelles ont été observées.

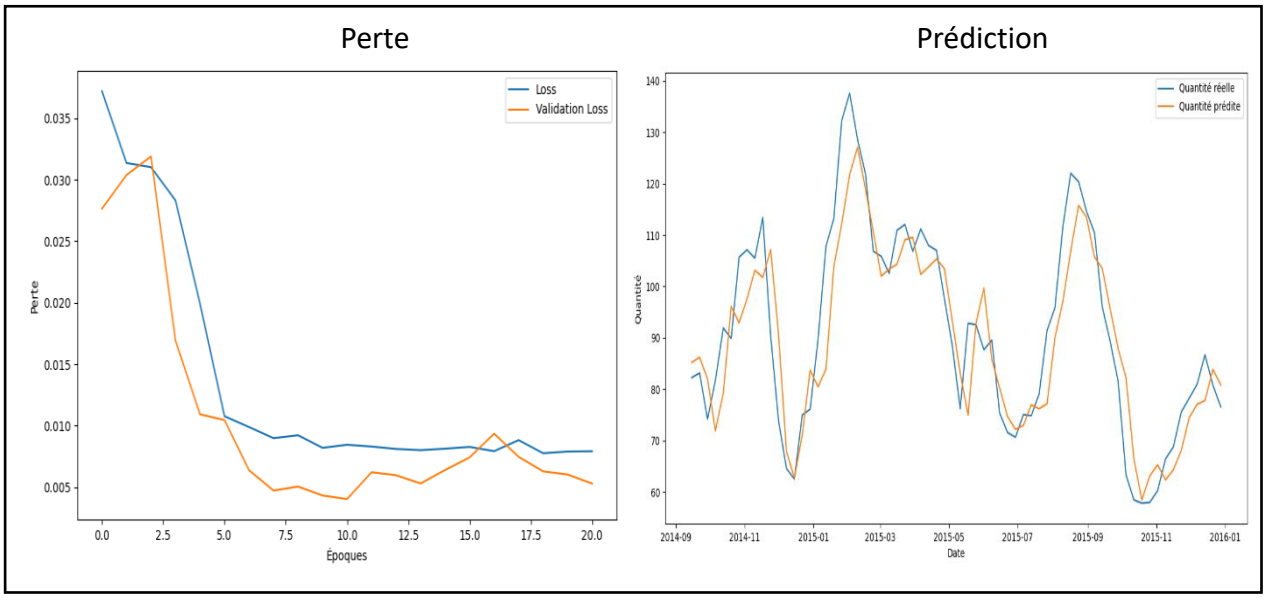

Figure 18 : Résultat d'entrainement - Modèle LSTM - Produit : Amoxicilline - 250g/5ml - Flacon

8.2. MLP (MultiLayer Perceptron) :

8.2.1. Produit « Paracétamol - 500mg – Comprimé »

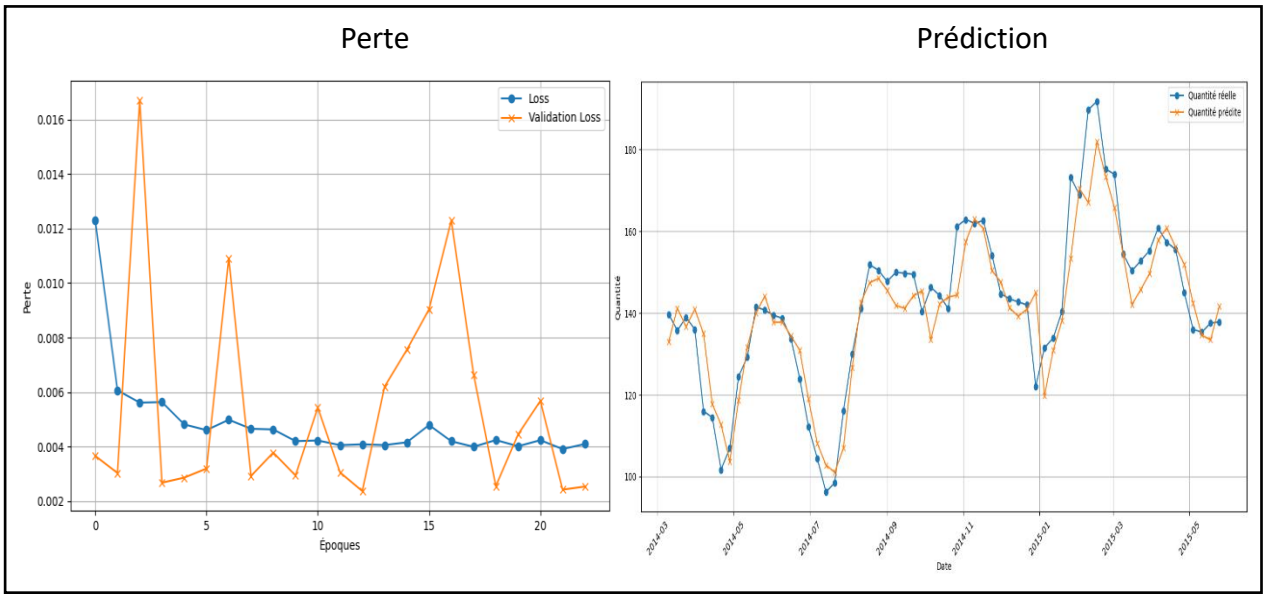

Figure 19 : Résultat d'entrainement - Modèle MLP - Produit : Paracétamol - 500mg – Comprimé

D'après la **figure 19** :

- La fonction de perte de validation est inférieure de la fonction de perte d'entraînement et présente des fluctuations irrégulières.
- Le modèle montré une capacité raisonnable à capturer les tendances et les motifs présents dans les données.

8.2.2. Produit « Paracétamol - 1000mg – Comprimé »

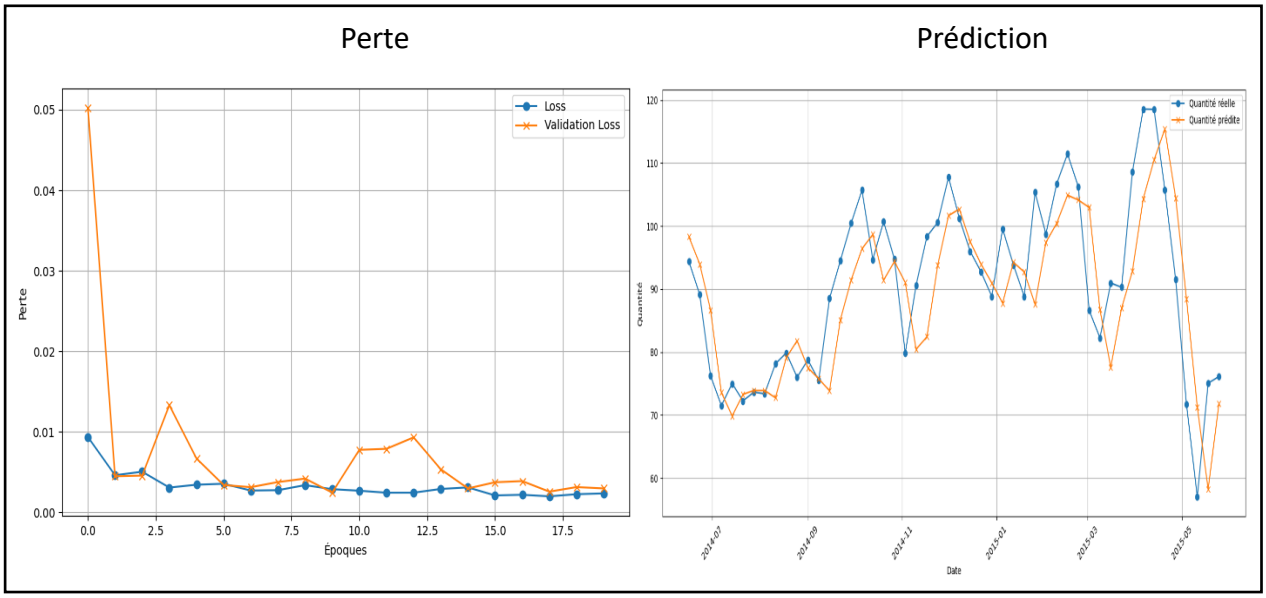

Figure 20 : Résultat d'entrainement - Modèle MLP - Produit : Paracétamol - 1000mg – Comprimé

D'après la **figure 20** :

- La fonction de perte de validation suit une tendance similaire non stable, bien qu'elle soit légèrement plus bas que la perte d'entraînement, ce qui est typique et montre que le modèle généralise raisonnablement bien aux nouvelles données.
- Le modèle montré une capacité raisonnable à capturer les tendances et les motifs présents dans les données, mais certaines divergences entre les prédictions et les valeurs réelles ont été observées.

8.2.3. Produit « Diclofénac – 1g/100g – gél »

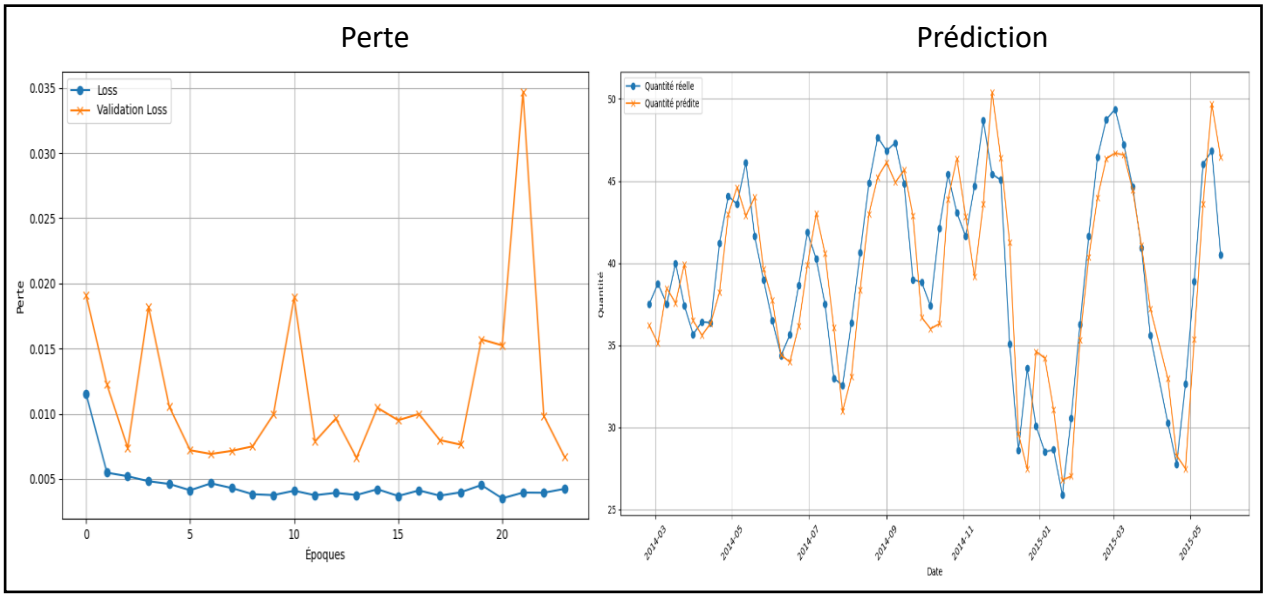

Figure 21 : Résultat d'entrainement - Modèle MLP - Produit : Diclofénac – 1g/100g – gél

D'après la **figure 21** :

- La fonction de perte de validation s'écarte de la fonction de perte d'entraînement et présente des fluctuations irrégulières.
- Le modèle montré une capacité raisonnable à capturer les tendances et les motifs présents dans les données.

8.2.4. Produit « Oméprazole – 20mg »

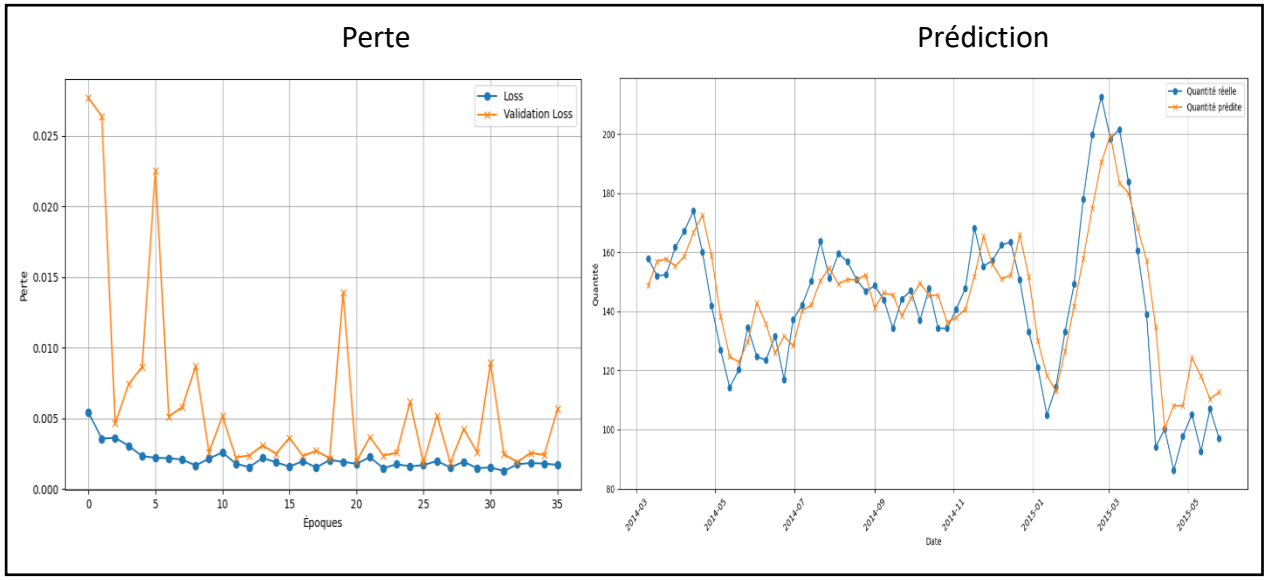

Figure 22 : Résultat d'entrainement - Modèle MLP - Produit : Oméprazole - 20mg

D'après la **figure 22** :

- La fonction de perte de validation s'écarte de la fonction de perte d'entraînement et présente des fluctuations irrégulières.
- Le modèle montré une capacité raisonnable à capturer les tendances et les motifs présents dans les données.

8.2.5. Produit « Amoxicilline – 250mg/5ml – Flacon »

D'après la **figure 23** :

- La fonction de perte de validation est inférieure de la fonction de perte d'entraînement et présente des fluctuations irrégulières.
- Le modèle montré une capacité raisonnable à capturer les tendances et les motifs présents dans les données.

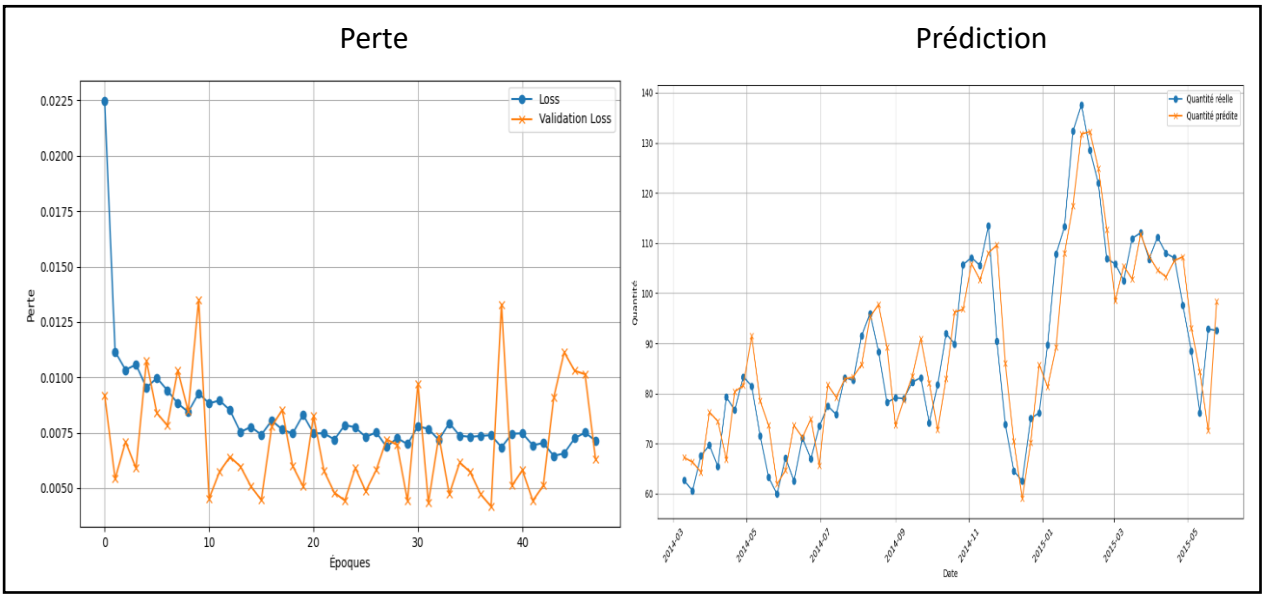

Figure 23 : Résultat d'entrainement - Modèle MLP - Produit : Amoxicilline - 250mg/5ml - Flacon

8.3. Régression polynomiale :

8.3.1. Produit « Paracétamol - 500mg – Comprimé »

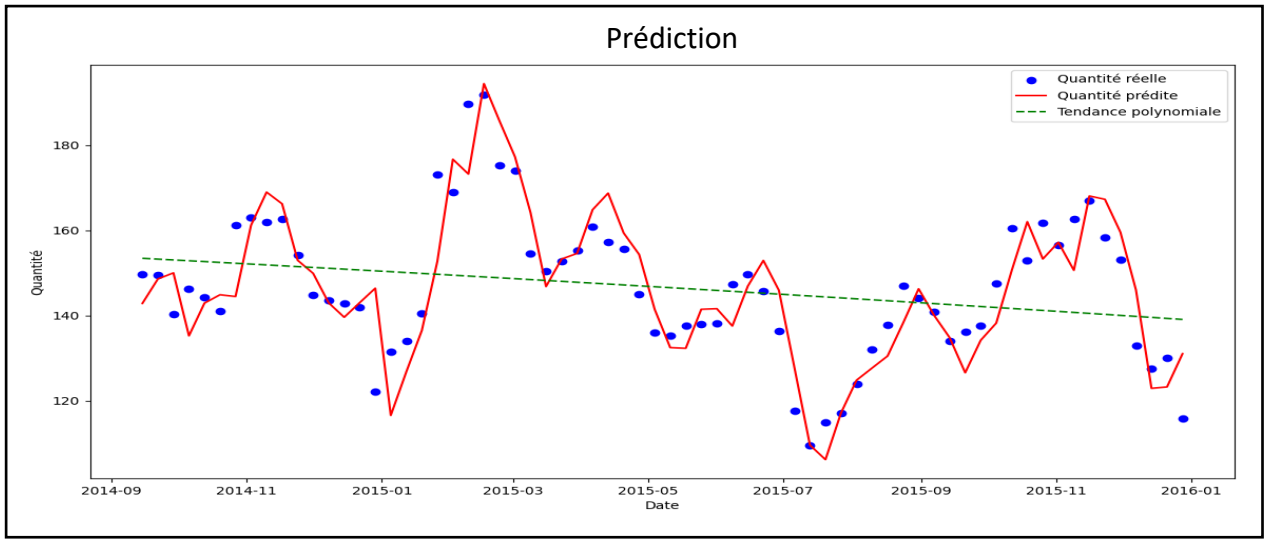

Figure 24 : Résultat d'entrainement – Modèle : Régression polynomiale - Produit : Paracétamol - 500mg – Comprimé

- La **figure 24** montre des écarts remarquables entre les valeurs et la tendance polynomial, particulièrement lorsque les valeurs sont très grandes ou très petites.

8.3.2. Produit « Paracétamol - 1000mg – Comprimé »

- La **figure 25** montre des écarts remarquables entre les valeurs prédites et réelles, particulièrement lorsque les valeurs sont très grandes ou très petites.

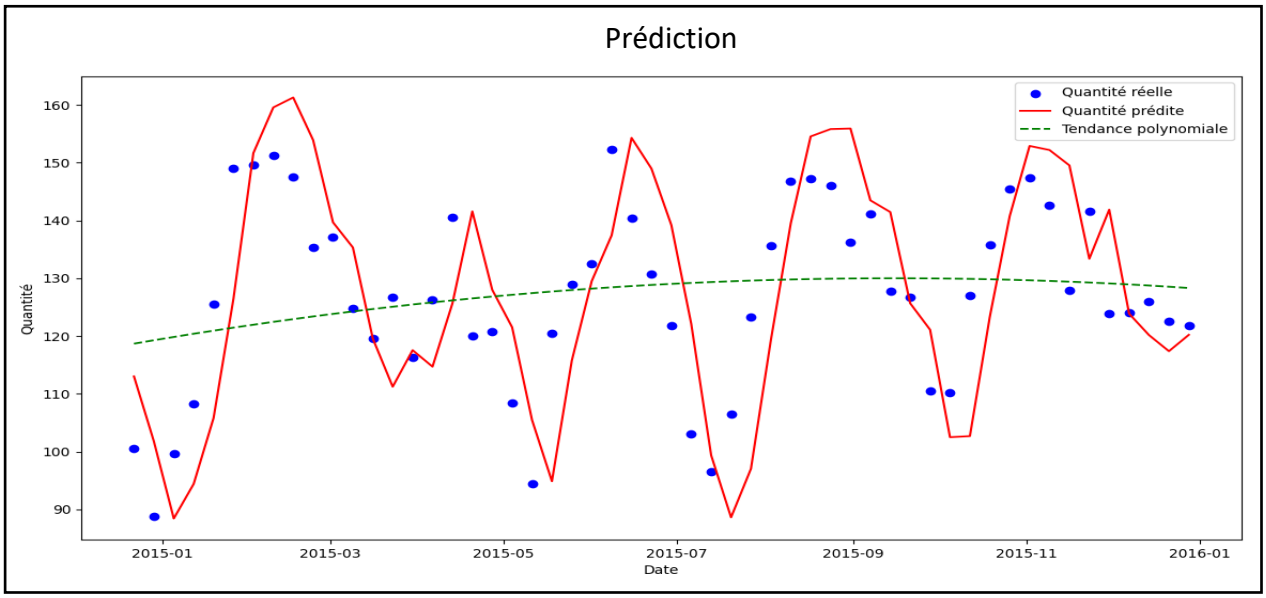

Figure 25 : Résultat d'entrainement – Modèle : Régression polynomiale - Produit : Paracétamol - 1000mg – Comprimé

8.3.3. Produit « Diclofénac – 1g/100g – gél »

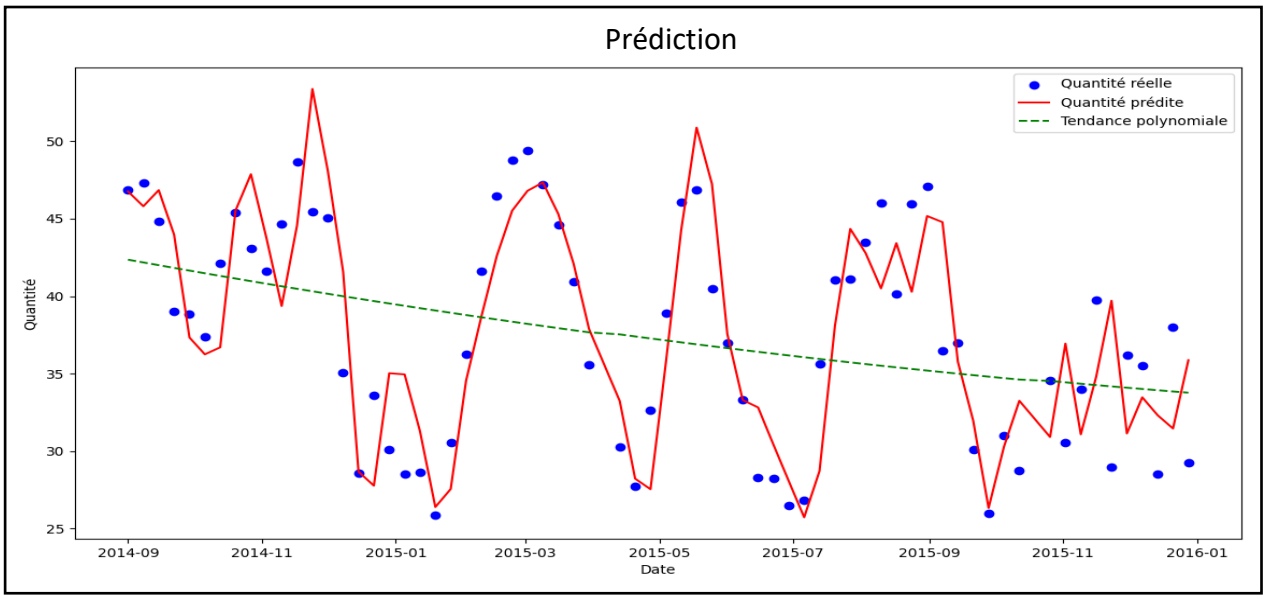

Figure 26 : Résultat d'entrainement – Modèle : Régression polynomiale - Produit : Diclofénac – 1g/100g – gél

- La **figure 26** montre aussi des écarts remarquables entre les valeurs prédites et réelles, particulièrement lorsque les valeurs sont très grandes ou très petites.

8.3.4. Produit « Oméprazole – 20mg »

- Selon la **figure 27** on voit des écarts remarquables entre les valeurs prédites et réelles, particulièrement lorsque les valeurs sont très grandes ou très petites.

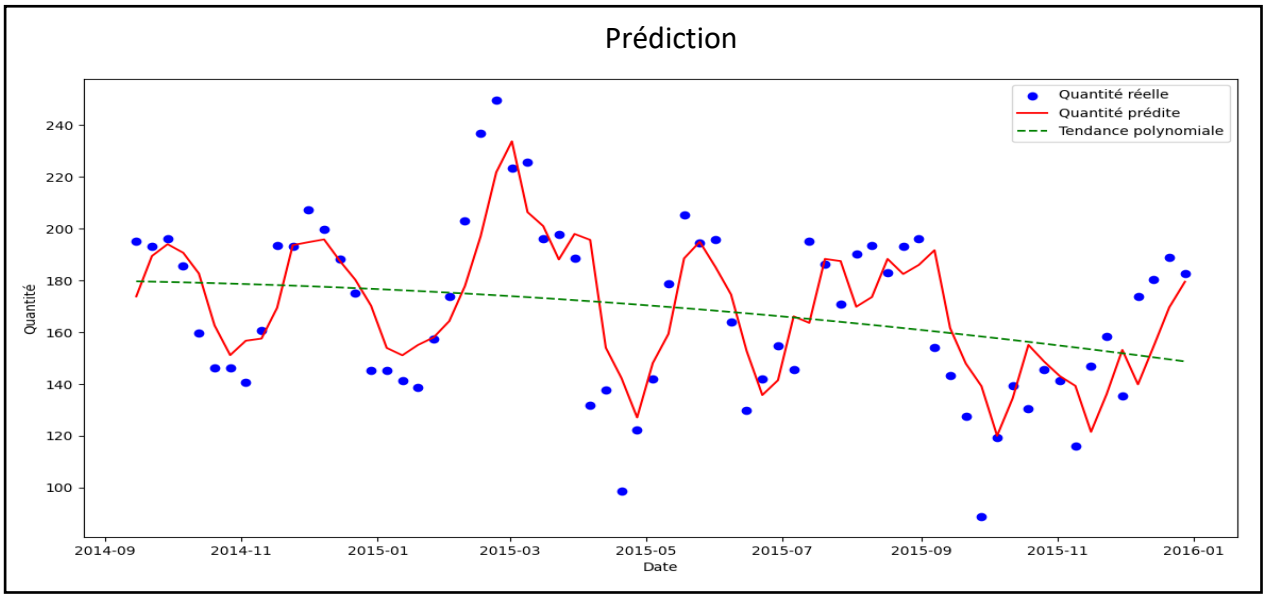

Figure 27 : Résultat d'entrainement – Modèle : Régression polynomiale - Produit : Oméprazole – 20mg

8.3.5. Produit « Amoxicilline – 250mg/5ml – Flacon »

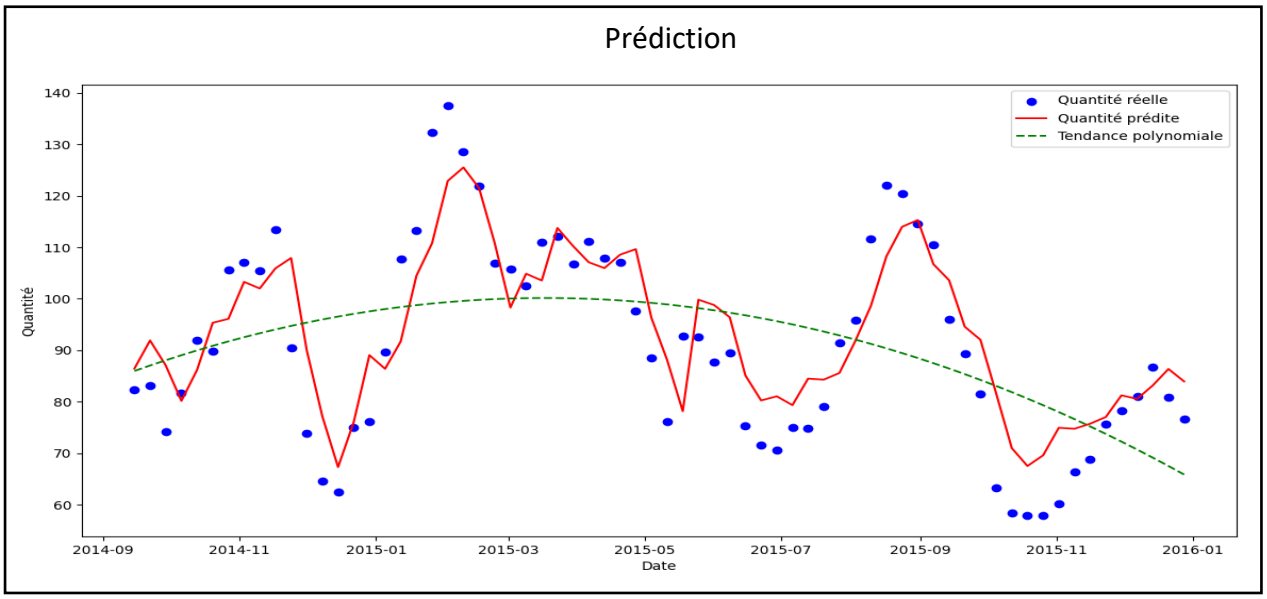

Figure 28 : Résultat d'entrainement – Modèle : Régression polynomiale - Produit : Amoxicilline – 250mg/5ml – Flacon

- D'après la **figure 28** on voit écarts remarquables entre les valeurs prédites et réelles, particulièrement lorsque les valeurs sont très grandes ou très petites.

9. Comparaison et évaluation des résultats

9.1. Comparaison :

9.1.1. Erreur Absolue Moyenne (Mean Absolute Error, MAE) :

D'après la comparaison des résultats du métrique « MAE » représenter par la **figure 30**, on peut voir que les valeurs du modèle **MLP (MultiLayer Perceptron)** sont les plus faible.

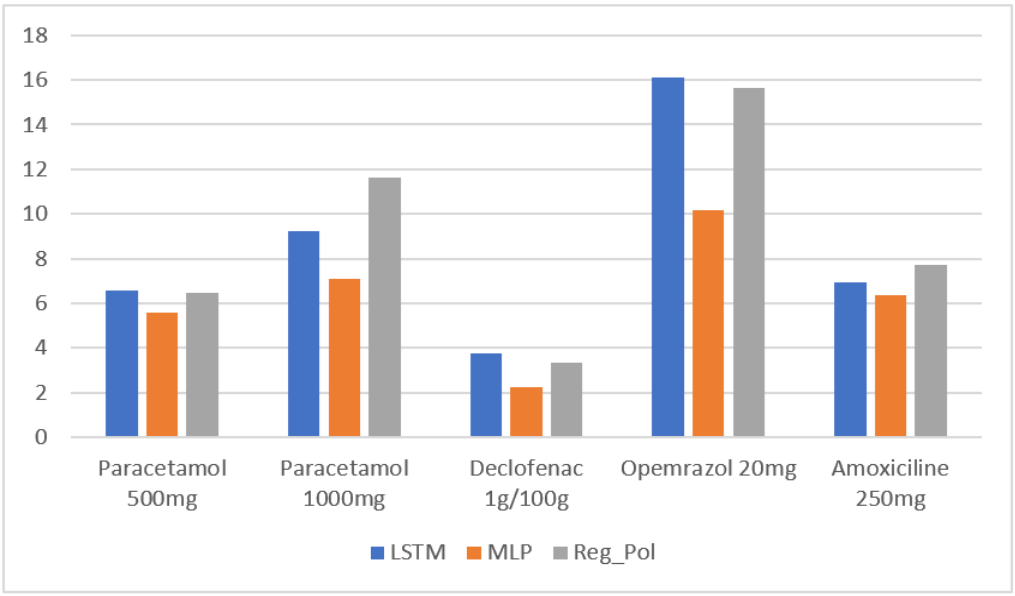

Figure 30 : Résultats du métrique "MAE"

9.1.2. Erreur Quadratique Moyenne (Mean Squared Error, MSE) :

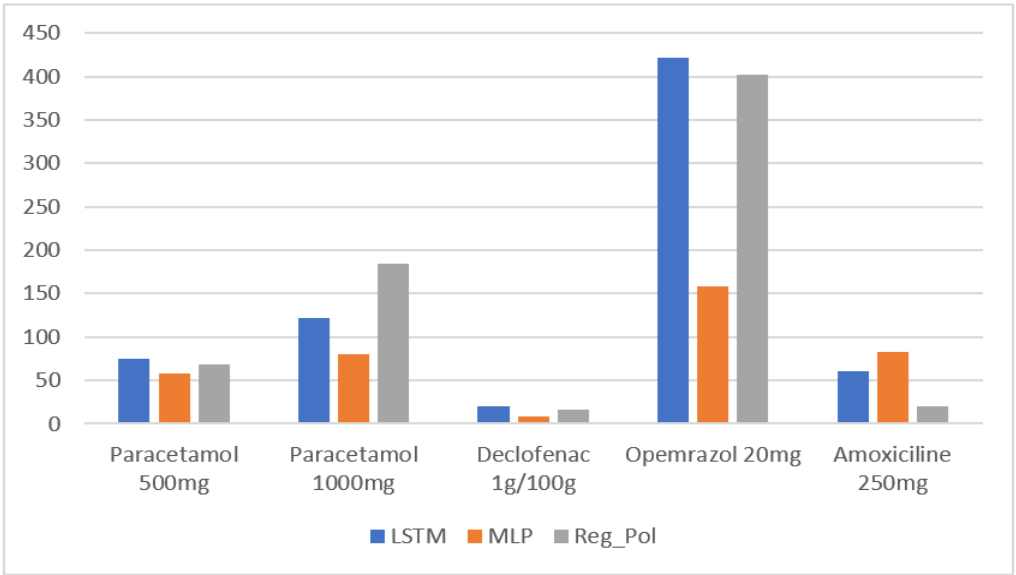

Figure 31 : Résultats du métrique "MSE"

D'après la comparaison des résultats du métrique « MES » représenter par la **figure 31**, on peut voir que les valeurs du modèle **MLP (MultiLayer Perceptron)** sont généralement les plus faible.

9.1.3. Erreur Quadratique Moyenne Racine (Root Mean Squared Error, RMSE) :

D'après la comparaison des résultats du métrique « RMSE » représenter par la **figure 32**, on peut voir que les valeurs du modèle **MLP (MultiLayer Perceptron)** sont les plus faible.

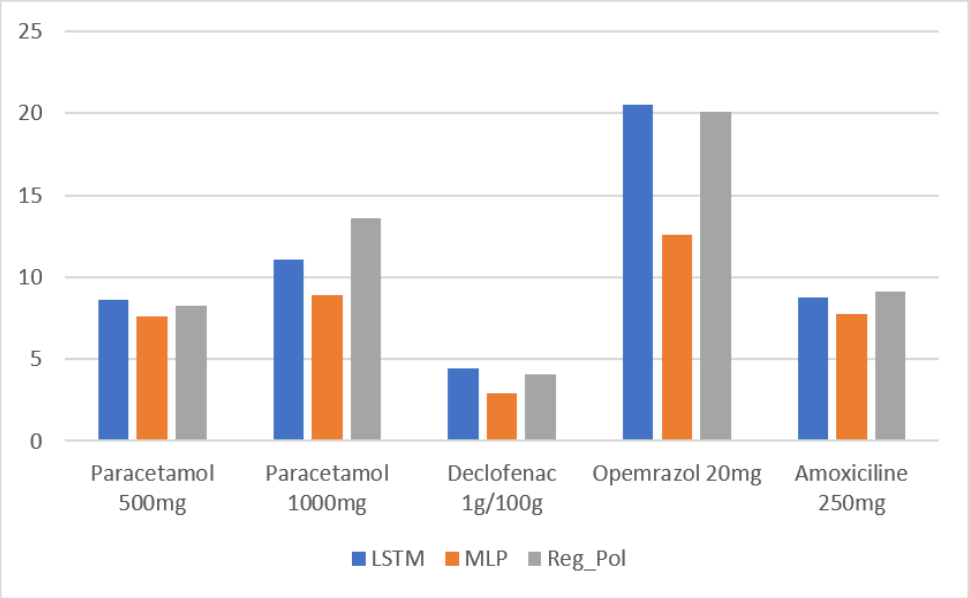

Figure 32 : Résultats du métrique "RMSE"

9.1.4. Erreur Absolue Moyenne en Pourcentage (Mean Absolute Percentage Error, MAPE) :

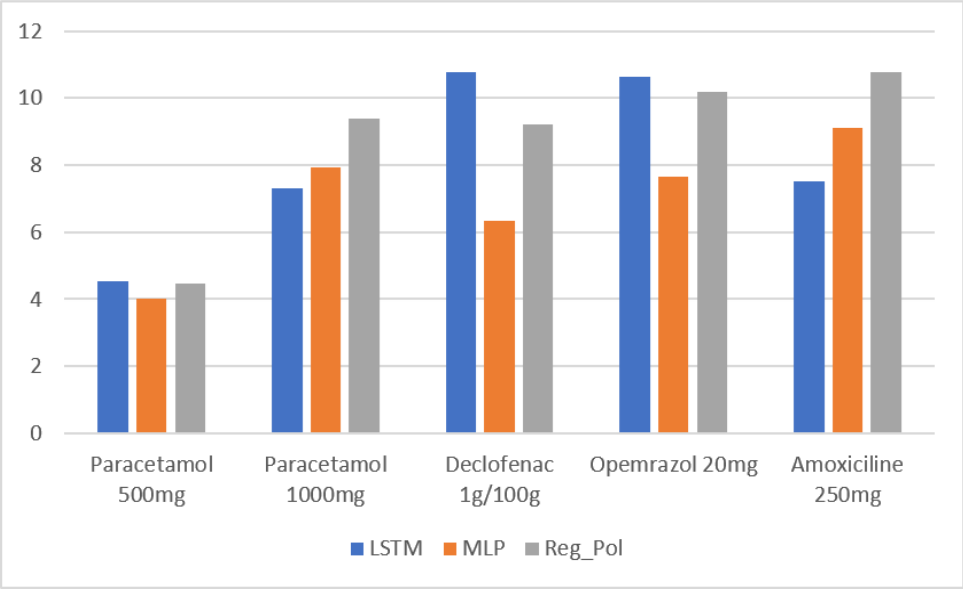

Figure 33 : Résultats du métrique "MAPE"

D'après la comparaison des résultats du métrique « MAPE » représenter par la **figure 33** on peut voir que les valeurs du modèle **MLP (MultiLayer Perceptron)** sont généralement les plus faible.

9.1.5. Coefficient de Détermination (R^2) :

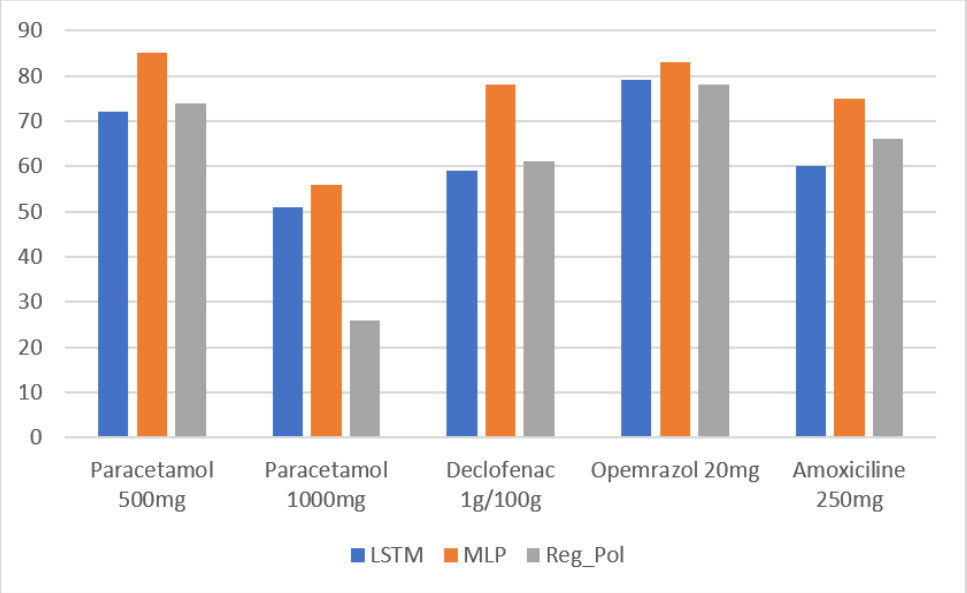

Figure 34 : Résultats du métrique "R^2"

D'après la comparaison des résultats du métrique « R^2 » représenter par la **figure 34**on peut voir que les valeurs du modèle **MLP (MultiLayer Perceptron)** sont les plus élevé.

9.2. Évaluation :

D'après les résultats obtenus après l'entrainement de chaque modèle on a constaté :

- Les résultats de prédiction pour le modèle 2 (MLP) sont plus proche aux valeurs réelles

- Pour les résultats des métriques MAE, MSE, RMSE, et MAPE, le 2éme modèle (MLP) donne 18/20 valeur faible.

- Pour le résultat du coefficient de détermination (R^2) MLP donne la valeur prédit les plus proche aux valeurs réelles

Donc, on peut conclure que « **Multilayer Perceptron (MLP)** » est le modèle le plus adapté pour prévoir les ventes de la pharmacie étudiée, en tenant compte des caractéristiques des données disponibles, à partir desquelles nous nous sommes appuyés principalement sur la quantité des ventes.

10. Conclusion :

Dans ce dernier chapitre, nous avons présenté dans un premier temps la préparation des données, en suite on a vu la préparation de l'environnement de développement et aussi les différentes bibliothèques utiliser.

Par la suite, nous avons créé les trois modèles proposés que nous avons entrainé puis tester.

Enfin, nous avons effectué une comparaison et évaluation des résultats de ces trois modèles.

Conclusion générale

L'apprentissage automatique s'enracine et prouve son efficacité dans divers domaines. Les travaux présentés dans ce mémoire explorent l'application de cette branche de l'informatique à la prédiction des ventes de médicaments. Nous avons souligné l'importance de l'industrie pharmaceutique, un secteur essentiel pour la santé publique et le développement économique national. Il est donc important de tester différentes techniques pour construire des modèles plus performants, capables de fournir des prévisions précises et fiables.

Pour bien cerner notre thème, nous avons structuré notre travail en quatre chapitres. Le premier chapitre présente les bases de l'apprentissage automatique. Le second chapitre se consacre à la présentation des particularités du marché des médicaments, afin de mieux appréhender les défis et les variables à considérer pour une prédiction précise, en termine par un état de l'art des techniques de prédiction des ventes.

Le troisième chapitre est structuré autour de la proposition de notre modèle de prédiction. Nous commençons par la présentation de l'objectif et motivation, ensuite on passe a une description des méthodes de prétraitement des données, et enfin, nous détaillons les modèles que nous avons développés pour cette étude. Le quatrième chapitre est dédié à l'implémentation et à l'évaluation des modèles proposés. Nous y présentons les différentes bibliothèques Python utilisées, les prétraitements appliqués aux données pharmaceutiques, ainsi que la création, l'entraînement et le test des modèles afin d'obtenir des résultats concluants.

La comparaison des résultats nous permet de conclure que le modèle base sur le multilayer perceptron fournit des prédictions plus précises par rapport aux deux autre modèle (LSTM et régression polynomiale) tout en tenant compte en particularité des données. Nous avons également observé que les modèles de régression polynomiale à moins efficace dans notre cas. Ces résultats soulignent l'importance de continuer à développer et affiner des modèles sophistiqués pour améliorer la prédiction des ventes de médicaments, contribuant ainsi à l'efficacité et à la régulation de l'industrie pharmaceutique.

Bibliographie

[1] J. Friedman, T. Hastie, and R. Tibshirani, The elements of statistical learning. Springer series.

[2] J. Brownlee, Deep Learning for Time Series Forecasting, 2019.

[3] L. ZIANI, «ZIANI, L. (2021). L'industrie du Médicament en Algérie: Etat des lieux et Contraintes,» *Revue Abaad Iktissadia,* vol. 11, n° %101, pp. 419-443, 2021.

[4] S. Alex, «SHERSTINSKY, Alex. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network,» *Physica D: Nonlinear Phenomena,* p. 404, 2020.

[5] H. e. M. J.-F. TAUD, Multilayer perceptron (MLP). Geomatic approaches for modeling land change scenarios, 2018, pp. 451-455.

[6] A. PECKOV, A machine learning approach to polynomial regression, Ljubljana, Slovenia, 2012.

[7] T. E. e. a. OLIPHANT, Guide to numpy, USA: Trelgol Publishing, 2006.

[8] W. M. a. t. P. D. Team, pandas: powerful Python data analysis, 2020.

[9] «TensorFlow,» 10 03 2024. [En ligne]. Available: https://www.tensorflow.org/.

[10] R. XU, Algorithmes et métriques d'évaluation pour la confiance dans les systèmes utilisant l'apprentissage machine: application à la reconnaissance visuelle d'objets, Thèse de doctorat: Université Grenoble Alpes, 2023.

[11] 05 06 2024. [En ligne]. Available: https://numpy.org/doc/stable/index.html.

Annexe

1. LSTM :

```python
import fdb
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
con = fdb.connect(dsn='localhost/3050:E:/BDD/db_2.fdb', user='SYSDBA', password='masterkey')
cur = con.cursor()
query = """
SELECT
    datevente, somme_ventes
FROM
    BEST_BDD
WHERE
    code_dci = '136043' and datevente<='2015/12/31'
ORDER BY
    datevente;
"""
cur.execute(query)
rows = cur.fetchall()
con.close()
df = pd.DataFrame(rows, columns=['datevente', 'somme_ventes'])
df['datevente'] = pd.to_datetime(df['datevente'])
df = df.groupby('datevente').sum().reset_index()
date_range = pd.date_range(start=df['datevente'].min(), end=df['datevente'].max())
df = df.set_index('datevente').reindex(date_range).rename_axis('datevente').reset_index()
window_size = 7
df['somme_ventes'] = df['somme_ventes'].fillna(df['somme_ventes'].rolling(window=window_size, min_periods=1).mean())
DF_1 = df
Q1 = df['somme_ventes'].quantile(0.25)
Q3 = df['somme_ventes'].quantile(0.75)
IQR = Q3 - Q1
borne_inf = Q1 - 1.5 * IQR
borne_sup = Q3 + 1.5 * IQR
df_clean = df[(df['somme_ventes'] >= borne_inf) & (df['somme_ventes'] <= borne_sup)]
df_clean.loc[:, 'datevente'] = pd.to_datetime(df_clean.loc[:, 'datevente'], errors='coerce')
df_clean.loc[:, 'annee'] = df_clean['datevente'].dt.year
df_clean.loc[:, 'semaine'] = df_clean['datevente'].dt.isocalendar().week
df_weekly = df_clean.groupby(['annee', 'semaine'])['somme_ventes'].sum().reset_index()
window_size = 4
```

```
window_size = 4
df_weekly['somme_ventes_lisse'] = df_weekly['somme_ventes'].rolling(window=window_size, min_periods=1).mean()
df_weekly['date_rep'] = pd.to_datetime(df_weekly['annee'].astype(str) + df_weekly['semaine'].astype(str) + '1', format='%G%V%u')
df_weekly = df_weekly.set_index('date_rep')
df_weekly = df_weekly.drop(['annee', 'semaine', 'somme_ventes'], axis=1)
df = df_weekly
scaler = MinMaxScaler()
df['quantite_normalized'] = scaler.fit_transform(df[['somme_ventes_lisse']])
sequence_length = 12
def create_sequences(data, sequence_length):
    sequences = []
    for i in range(len(data) - sequence_length):
        sequences.append(data[i:i + sequence_length + 1])
    return np.array(sequences)
all_quantities_normalized = df['quantite_normalized'].values
sequences = create_sequences(all_quantities_normalized, sequence_length)
train_size = int(len(sequences) * 0.8)
val_size = int(len(sequences) * 0.1)
train_sequences = sequences[:train_size]
val_sequences = sequences[train_size:train_size + val_size]
test_sequences = sequences[train_size + val_size:]
X_train = train_sequences[:, :-1]
y_train = train_sequences[:, -1]
X_val = val_sequences[:, :-1]
y_val = val_sequences[:, -1]
X_test = test_sequences[:, :-1]
y_test = test_sequences[:, -1]
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_val = X_val.reshape((X_val.shape[0], X_val.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
model = tf.keras.Sequential([
    tf.keras.layers.LSTM( units: 256, return_sequences=True, input_shape=(sequence_length, 1)),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.LSTM( units: 256, return_sequences=True),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.LSTM( units: 128, return_sequences=True),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.LSTM(128),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense( units: 128, activation='swish'),
    tf.keras.layers.Dense(1)    ])
```

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0005), loss='mean_squared_error')
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=50, batch_size=2, validation_data=(X_val, y_val), callbacks=[early_stopping])
predictions = model.predict(X_test)
y_test_real = scaler.inverse_transform(y_test.reshape(-1, 1))
predictions_real = scaler.inverse_transform(predictions)
results_df = pd.DataFrame({'Date': df.index[-len(predictions_real):],
                           'Quantité réelle': y_test_real.flatten(),
                           'Quantité prédite': predictions_real.flatten()})
print(results_df)
X_next_month = np.array(df['quantite_normalized'].iloc[-sequence_length:].values).reshape(1, -1, 1)
next_month_prediction = model.predict(X_next_month)
next_month_prediction_real = scaler.inverse_transform(next_month_prediction)
print("Prédiction du mois suivant:", next_month_prediction_real[0][0])
mae = mean_absolute_error(y_test_real, predictions_real)
mse = mean_squared_error(y_test_real, predictions_real)
rmse = np.sqrt(mse)
mape = np.mean(np.abs((y_test_real - predictions_real) / y_test_real)) * 100
r2 = r2_score(y_test_real, predictions_real)
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("Mean Absolute Percentage Error (MAPE):", mape)
print("R² Score:", r2)
plt.figure(figsize=(14, 7))
plt.plot( *args: results_df['Date'], results_df['Quantité réelle'], label='Quantité réelle')
plt.plot( *args: results_df['Date'], results_df['Quantité prédite'], label='Quantité prédite')
plt.xlabel('Date')
plt.ylabel('Quantité')
plt.title('Quantités réelles vs prédites')
plt.legend()
plt.show()
plt.figure(figsize=(10, 6))
plt.plot( *args: history.history['loss'], label='Loss')
plt.plot( *args: history.history['val_loss'], label='Validation Loss')
plt.title('Fonction de perte pendant l\'entraînement')
plt.xlabel('Époques')
plt.ylabel('Perte')
plt.legend()
plt.show()
```

2. MLP :

```
import fdb
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

con = fdb.connect(dsn='localhost/3050:E:/BDD/db_2.fdb', user='SYSDBA', password='masterkey')
cur = con.cursor()
query = """
SELECT
    datevente, somme_ventes
FROM
    BEST_BDD
WHERE
    code_dci = '136043' and datevente<='2015/05/31'
ORDER BY
    datevente;
"""

cur.execute(query)
rows = cur.fetchall()
con.close()
df = pd.DataFrame(rows, columns=['datevente', 'somme_ventes'])
df['datevente'] = pd.to_datetime(df['datevente'])
df = df.groupby('datevente').sum().reset_index()
date_range = pd.date_range(start=df['datevente'].min(), end=df['datevente'].max())
df = df.set_index('datevente').reindex(date_range).rename_axis('datevente').reset_index()
window_size = 7
df['somme_ventes'] = df['somme_ventes'].fillna(df['somme_ventes'].rolling(window=window_size, min_periods=1).mean())
Q1 = df['somme_ventes'].quantile(0.25)
Q3 = df['somme_ventes'].quantile(0.75)
IQR = Q3 - Q1
borne_inf = Q1 - 1.5 * IQR
borne_sup = Q3 + 1.5 * IQR
df_clean = df[(df['somme_ventes'] >= borne_inf) & (df['somme_ventes'] <= borne_sup)]
df_clean['annee'] = df_clean['datevente'].dt.year
df_clean['semaine'] = df_clean['datevente'].dt.isocalendar().week
df_weekly = df_clean.groupby(['annee', 'semaine'])['somme_ventes'].sum().reset_index()
window_size = 4
```

```
df_weekly['somme_ventes_lisse'] = df_weekly['somme_ventes'].rolling(window=window_size, min_periods=1).mean()
df_weekly['date_rep'] = pd.to_datetime(df_weekly['annee'].astype(str) + df_weekly['semaine'].astype(str) + '1', format='%G%V%u')
df_weekly = df_weekly.set_index('date_rep')
if 'somme_ventes_lisse' in df_weekly.columns:
    df_weekly = df_weekly.drop(['annee', 'semaine', 'somme_ventes'], axis=1)
scaler = MinMaxScaler()
df_weekly['quantite_normalized'] = scaler.fit_transform(df_weekly[['somme_ventes_lisse']])
sequence_length = 10
def create_sequences(data, sequence_length):
    sequences = []
    for i in range(len(data) - sequence_length):
        sequences.append(data[i:i + sequence_length + 1])
    return np.array(sequences)
all_quantities_normalized = df_weekly['quantite_normalized'].values
sequences = create_sequences(all_quantities_normalized, sequence_length)
train_size = int(len(sequences) * 0.8)
val_size = int(len(sequences) * 0.1)
train_sequences = sequences[:train_size]
val_sequences = sequences[train_size:train_size + val_size]
test_sequences = sequences[train_size + val_size:]
X_train = train_sequences[:, :-1]
y_train = train_sequences[:, -1]
X_val = val_sequences[:, :-1]
y_val = val_sequences[:, -1]
X_test = test_sequences[:, :-1]
y_test = test_sequences[:, -1]
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(sequence_length,)),
    tf.keras.layers.Dense( units: 256, activation='swish'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense( units: 256, activation='swish'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense( units: 128, activation='swish'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense( units: 128, activation='swish'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(1)     ])
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0005), loss='mean_squared_error')
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=50, batch_size=2, validation_data=(X_val, y_val), callbacks=[early_stopping])
predictions = model.predict(X_test)
```

```
y_test_real = scaler.inverse_transform(y_test.reshape(-1, 1))
predictions_real = scaler.inverse_transform(predictions)
results_df = pd.DataFrame({'Date': df_weekly.index[-len(predictions_real):],
                           'Quantité réelle': y_test_real.flatten(),
                           'Quantité prédite': predictions_real.flatten()})
print(results_df)
X_next_month = np.array(df_weekly['quantite_normalized'].iloc[-sequence_length:].values).reshape(1, -1)
next_month_prediction = model.predict(X_next_month)
next_month_prediction_real = scaler.inverse_transform(next_month_prediction)
print("Prédiction du mois suivant:", next_month_prediction_real[0][0])
mae = mean_absolute_error(y_test_real, predictions_real)
mse = mean_squared_error(y_test_real, predictions_real)
rmse = np.sqrt(mse)
mape = np.mean(np.abs((y_test_real - predictions_real) / y_test_real)) * 100
r2 = r2_score(y_test_real, predictions_real)
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("Mean Absolute Percentage Error (MAPE):", mape)
print("R² Score:", r2)
plt.figure(figsize=(14, 7))
plt.plot( *args: results_df['Date'], results_df['Quantité réelle'], label='Quantité réelle', marker='o')
plt.plot( *args: results_df['Date'], results_df['Quantité prédite'], label='Quantité prédite', marker='x')
plt.xlabel('Date')
plt.ylabel('Quantité')
plt.title('Quantités réelles vs prédites')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
plt.figure(figsize=(10, 6))
plt.plot( *args: history.history['loss'], label='Loss', marker='o')
plt.plot( *args: history.history['val_loss'], label='Validation Loss', marker='x')
plt.title('Fonction de perte pendant l\'entraînement')
plt.xlabel('Époques')
plt.ylabel('Perte')
plt.legend()
plt.grid(True)
plt.show()
```

3. Régression polynomiale :

```
import fdb
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

con = fdb.connect(dsn='localhost/3050:E:/BDD/db_2.fdb', user='SYSDBA', password='masterkey')
cur = con.cursor()
query = """
SELECT
    datevente,
    somme_ventes
FROM
    BEST_BDD
WHERE
    code_dci = '136043' and datevente<='2015/12/31'
ORDER BY
    datevente;
"""
cur.execute(query)
rows = cur.fetchall()
con.close()
df = pd.DataFrame(rows, columns=['datevente', 'somme_ventes'])
df['datevente'] = pd.to_datetime(df['datevente'])
df = df.groupby('datevente').sum().reset_index()
date_range = pd.date_range(start=df['datevente'].min(), end=df['datevente'].max())
df = df.set_index('datevente').reindex(date_range).rename_axis('datevente').reset_index()
window_size = 7
df['somme_ventes'] = df['somme_ventes'].fillna(df['somme_ventes'].rolling(window=window_size, min_periods=1).mean())
Q1 = df['somme_ventes'].quantile(0.25)
Q3 = df['somme_ventes'].quantile(0.75)
IQR = Q3 - Q1
borne_inf = Q1 - 1.5 * IQR
borne_sup = Q3 + 1.5 * IQR
df_clean = df[(df['somme_ventes'] >= borne_inf) & (df['somme_ventes'] <= borne_sup)]
df_clean['annee'] = df_clean['datevente'].dt.year
df_clean['semaine'] = df_clean['datevente'].dt.isocalendar().week
df_weekly = df_clean.groupby(['annee', 'semaine'])['somme_ventes'].sum().reset_index()
```

```
window_size = 4
df_weekly['somme_ventes_lisse'] = df_weekly['somme_ventes'].rolling(window=window_size, min_periods=1).mean()
df_weekly['date_rep'] = pd.to_datetime(df_weekly['annee'].astype(str) + df_weekly['semaine'].astype(str) + '1', format='%G%V%u')
df_weekly = df_weekly.set_index('date_rep')
df_weekly = df_weekly.drop(['annee', 'semaine', 'somme_ventes'], axis=1)
scaler = MinMaxScaler()
df_weekly['quantite_normalized'] = scaler.fit_transform(df_weekly[['somme_ventes_lisse']])
print(df_weekly)
sequence_length = 10
def create_sequences(data, sequence_length):
    sequences = []
    for i in range(len(data) - sequence_length):
        sequences.append(data[i:i + sequence_length + 1])
    return np.array(sequences)
all_quantities_normalized = df_weekly['quantite_normalized'].values
sequences = create_sequences(all_quantities_normalized, sequence_length)
train_size = int(len(sequences) * 0.8)
val_size = int(len(sequences) * 0.1)
train_sequences = sequences[:train_size]
val_sequences = sequences[train_size:train_size + val_size]
test_sequences = sequences[train_size + val_size:]
X_train = train_sequences[:, :-1]
y_train = train_sequences[:, -1]
X_val = val_sequences[:, :-1]
y_val = val_sequences[:, -1]
X_test = test_sequences[:, :-1]
y_test = test_sequences[:, -1]
degree = 2
poly = PolynomialFeatures(degree)
X_train_poly = poly.fit_transform(X_train)
X_val_poly = poly.transform(X_val)
X_test_poly = poly.transform(X_test)
model = Ridge(alpha=1.0)
model.fit(X_train_poly, y_train)
predictions = model.predict(X_test_poly)
y_test_real = scaler.inverse_transform(y_test.reshape(-1, 1))
predictions_real = scaler.inverse_transform(predictions.reshape(-1, 1))
results_df = pd.DataFrame({'Date': df_weekly.index[-len(predictions_real):],
                           'Quantité réelle': y_test_real.flatten(),
                           'Quantité prédite': predictions_real.flatten()})
```

```
print(results_df)
X_next_month = poly.transform(np.array(df_weekly['quantite_normalized'].iloc[-sequence_length:].values).reshape(1, -1))
next_month_prediction = model.predict(X_next_month)
next_month_prediction_real = scaler.inverse_transform(next_month_prediction.reshape(-1, 1))
print("Prédiction du mois suivant:", next_month_prediction_real[0][0])
mae = mean_absolute_error(y_test_real, predictions_real)
mse = mean_squared_error(y_test_real, predictions_real)
rmse = np.sqrt(mse)
mape = np.mean(np.abs((y_test_real - predictions_real) / y_test_real)) * 100
r2 = r2_score(y_test_real, predictions_real)
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("Mean Absolute Percentage Error (MAPE):", mape)
print("R² Score:", r2)
plt.figure(figsize=(14, 7))
plt.scatter(results_df['Date'], results_df['Quantité réelle'], label='Quantité réelle', color='blue')
plt.plot( *args: results_df['Date'], results_df['Quantité prédite'], label='Quantité prédite', color='red')
coefs = np.polyfit(range(len(results_df['Date'])), results_df['Quantité réelle'], deg: 2)
poly_trend = np.poly1d(coefs)
trendline = poly_trend(range(len(results_df['Date'])))
plt.plot( *args: results_df['Date'], trendline, label='Tendance polynomiale', color='green', linestyle='--')
plt.xlabel('Date')
plt.ylabel('Quantité')
plt.title('Quantités réelles vs prédites avec tendance polynomiale')
plt.legend()
plt.show()
```