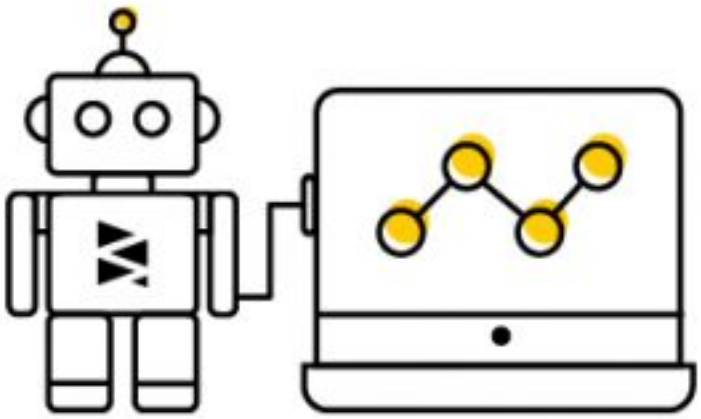
Chapter 1

Contribution



1.1 Introduction

Dans ce chapitre, nous abordons le dernier volet de ce mémoire, qui a pour objectif l'exposition de la phase de réalisation de notre projet. Cette phase est considérée comme étant la concrétisation finale de notre recherche représentée par le développement et l'évaluation d'un modèle de prédiction, de l'activité biologique et finalement d'un ensemble de molécules par l'apprentissage en profondeur. Tout d'abord, nous allons expliqué la démarche et la méthode proposée, ensuite nous détaillons l'implémentation de l'approche proposée ainsi que les outils utilisés pour illustrer le fonctionnement du modèle. Enfin, nous discutons les résultats obtenus par notre modèle et les comparer avec d'autres approches dans la littérature.

1.2 Description de la méthode proposée

Maintenant, nous allons fait le point pour expliquer notre démarche quant va utilisé pour ce faire notre projet et qui va nous permettre de mettre en pratique , avec suggères les étapes suivantes :

1. Importer les bibliothèque quand va utilisé
2. La lecture des données à partir le fichier csv de dataset
3. La transformation de données word2vect
4. Définie le modèle avec leur architecture
5. Mettre le modèle à l'entrainement sur les données
6. L'évaluation de modèle avec le test
7. La prédiction des valeur

1.3 Schéma de la méthode proposé

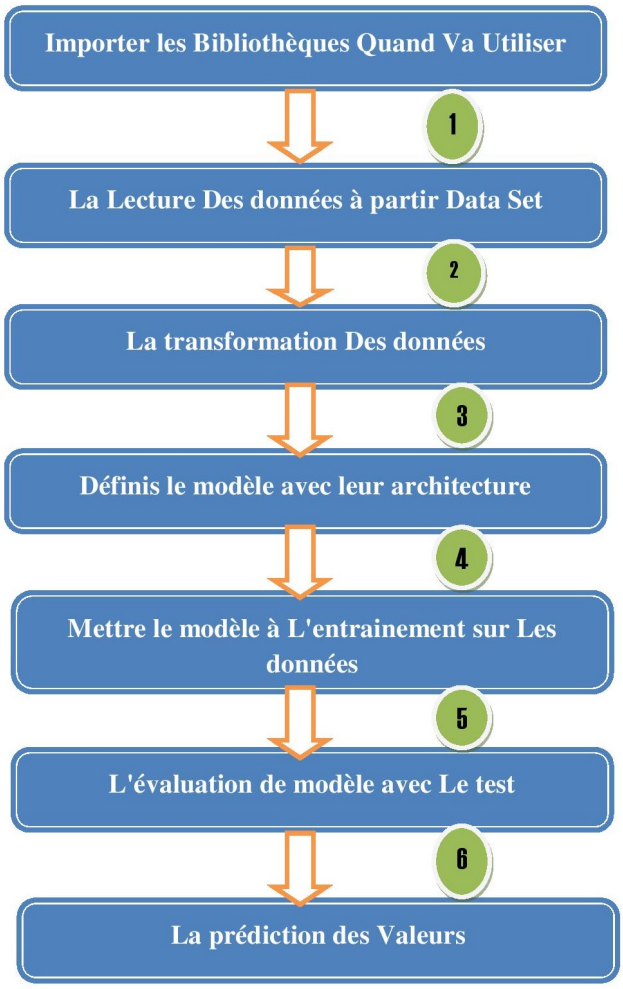


Figure 1.1: Les étapes de la méthode proposé

1.3.1 Différence entre le développement traditionnel et l'apprentissage automatique

Le diagramme suivant illustre la différence entre l'apprentissage automatique et le développement de logiciel traditionnel [5] :

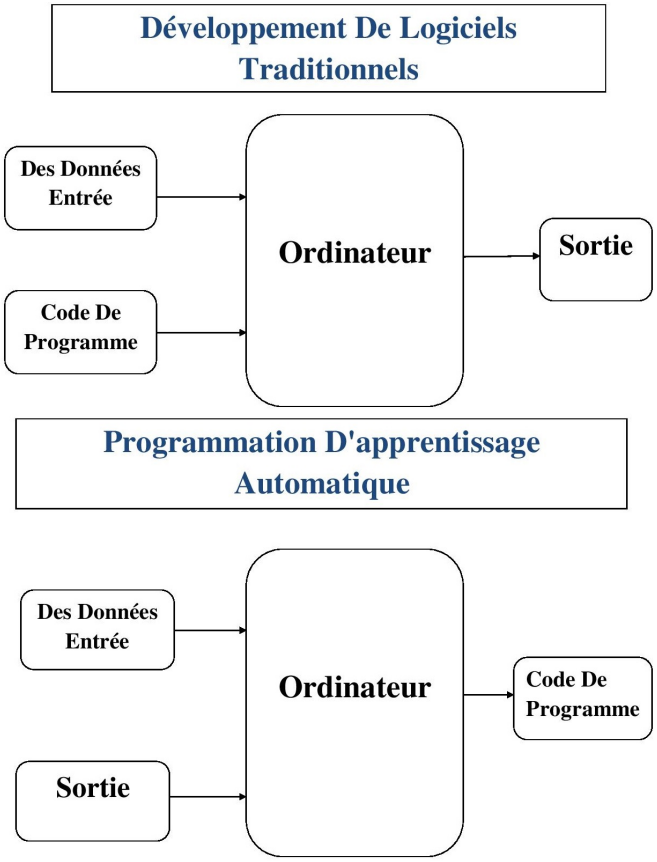


Figure 1.2: le développement traditionnel et l'apprentissage automatique

Développement de logiciel traditionnel

Les programmeurs créent des programmes qui spécifient comment transformer une entrée en sortie souhaitée [5] .

L'apprentissage automatique

Les programmeurs créent des modèles qui peuvent apprendre à produire la sortie souhaitée pour une entrée donnée. Cet apprentissage remplit le rôle traditionnel du programme informatique [5] .

1.3.2 Le langage de programmation choisir

C'est bien beau de vouloir apprendre un langage de programmation, mais il faut savoir lequel choisir. alors pour notre projet nous allons utiliser le langages **python** .

Figure 1.3: Langages de programmation

1.3.3 Python

python est un langage de programmation open source interprété côté serveur et non compilé. Créé par Guido van Rossum [11], il est utilisé pour le développement web, le développement de jeux-vidéos et autres logiciels, ainsi que pour les interfaces utilisateur graphiques. Il a notamment été

utilisé dans la création d'Instagram, de YouTube et de Spotify, et est l'un des langages de programmation officiels de Google.

1.3.4 Pour quoi nous choisir le python

au niveau de notre projet qui représente python est le langage de programmation le plus utilisé dans le domaine du Apprentissage en profondeur et de la Data Science. qui fournit des bibliothèque très puissante. qui permettent de coder des algorithmes d'intelligence artificielle et de machine learning, en particulier des réseaux de neurones .

1.3.5 Les bibliothèques

Avant de commencer à attaquer notre script nous avons beaucoup de **bibliothèques** qui nous permettent et nous aident à travailler pour implémenter ce modèle .

Figure 1.4: librairies

1.3.6 DataSet

Un DataSet, pour résumer, est une représentation d'une base de données sous forme d'objet. Il contient des tables, elles-mêmes contiennent des colonnes et des lignes. On pourrait le schématiser par la figure suivante.

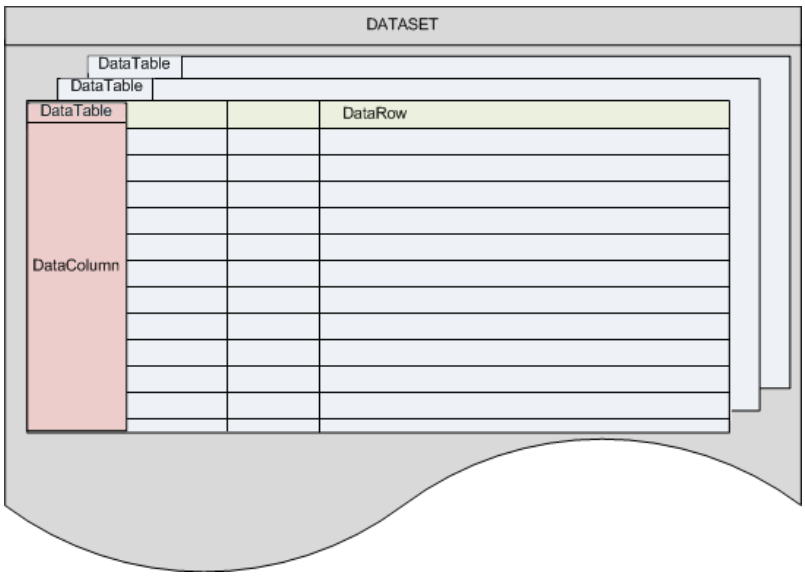


Figure 1.5: Schéma d'un DataSet

1.3.7 La lecture de données

Au niveau de notre projet l'ensemble données (DataSet) représente un fichier CSV qui contient l'ensemble des molécule et descripteur (Smille canonic) avec leur activité biologique sous nom (Observed) ,aussi plusieur autre colonne comme le nom de la molécule et rdkit Mol. Pour apprendre à lire nos données nous allons faisant appel à un bibliothèque spécifiées qui s'appelle panda, Cela vous permet de lire et traiter ce fichier un enregistrement à la fois.

1.3.8 Concaténer les lignes et les colonnes

après la lecture de données on passe par une étape très importante qui va nous faire concaténer les lignes et les colonnes, pour former de nouvelles structure de données [5].

Row	Name	SmilesCanonical	Observed
0	Benzyl	Cc1ccccc1N	-0.36
1	Aniline	NCCc1ccccc1	-0.24
2	Benzane	CCc1cccc(N)c1	0.12
3	Methilany	OCC=Cc1ccccc1	0.28

Table 1.1: Concaténation entre trois colonnes

1.3.9 Transformation des données

Les réseaux de neurones ne fonctionnent pas directement sur les cadres de données Python. Un réseau de neurones nécessite une matrice **numérique** ou format numérique. pour convertir les colonnes qui ne sont pas au format numérique par exemple dans notre dataSet la colonnes de SmilesCanonical est une format de chaine de caractère (String).

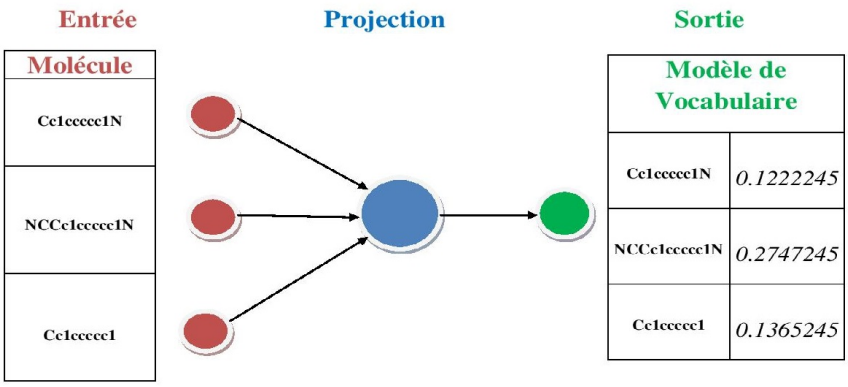


Figure 1.6: Modèle de transformation

Alors on va utilisé des méthodes basés sur Le traitement du langage naturel NLP. pour trouvées un modèle puissante de transformation de format string vers le format numérique.

Traitement automatique du langage naturel NLP

est une branche très importante du Apprentissage automatique et donc de l'intelligence artificielle. Le NLP est la capacité d'un programme à comprendre le langage humain. Certains exemples pratiques de la NLP sont la reconnaissance vocale [9].

Figure 1.7: Langage naturel

Transformation mot à vecteur word2vec

est une structure de réseau de neurones permettant de générer l'intégration de mots en formant le modèle à un problème de classification supervisée. Une telle méthode a été introduite pour la première fois dans le document Estimation efficace des représentations de mots dans l'espace vectoriel comme vous avez déja vue dans la figure de Modèle de transformation [10].

1.3.10 Définis le modèle

Au niveau d'une application d'apprentissage en profondeur ,L'étape la plus importante est de définir **le modèle** et leur **architecture** après de choisir le type de modèle, Alors on a choisi le type de modèle le plus utilisé est le modèle **séquentiel** .
Un excellent moyen d'utiliser l'apprentissage en profondeur consiste à créer un réseau de neurones à convolution (CNN).

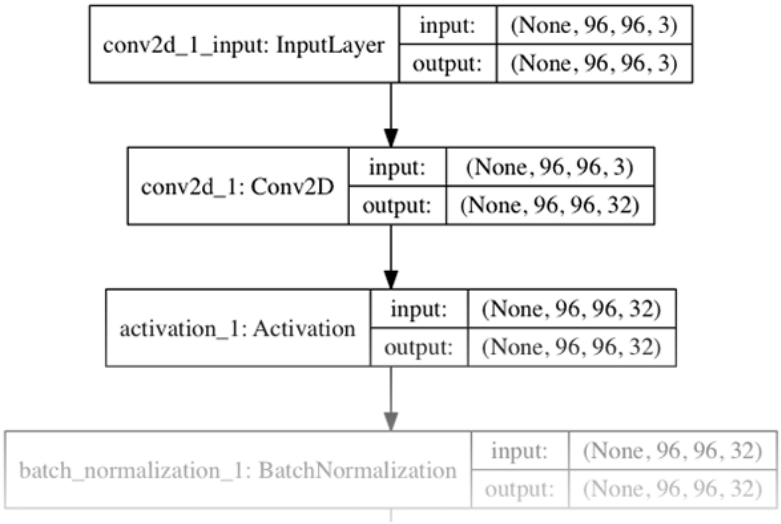


Figure 1.8: Modèle CNN

Le modèle séquentiel

Le modèle doit savoir à quelle forme d'entrée il doit s'attendre. Pour cette raison, la première couche d'un modèle séquentiel doit recevoir des informations sur sa forme en entrée [4].

Spécifier la forme d'entrée

Le modèle doit savoir à quelle forme d'entrée il doit s'attendre. Pour cette raison, la première couche d'un modèle séquentiel (et uniquement la première,

car les couches suivantes peuvent effectuer une inférence de forme automatique) doit recevoir des informations sur la forme en entrée [6] .
cette entrée Passez par un argument **inputshape** à la première couche.

Définis le réseau de neurones

Dans cette étape nous avons choisir le type de réseau de neurones quant va utilise dans notre modèle qui est le CNN. le réseau de neurones de convolution est similaire à un réseau de perceptrons multicouches. Les principales différences sont ce que le réseau apprend, comment il est structuré et à quoi il sert principalement [8] .

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

Figure 1.9: Modèle CNN

Le réseau de neurones de convolution Conv1D

Cette couche crée un noyau de convolution qui est convolutionné avec l'entrée de couche sur une seule dimension spatiale (ou temporelle) pour produire un tenseur de sorties [2] . par exemple :
keras.layers.Conv1D(filters, kernelsize, inputshape, activation='relu') [2]

Le réseau de neurones de convolution Conv2D

Cette couche crée un noyau de convolution qui est convolué avec l'entrée de la couche sur une deux dimension pour produire un tenseur de sorties. dans notre modèle nous avons utilisé le Conv2D parce que notre vecteur de données est avec deux dimension [2] . par exemple :
keras.layers.Conv2D(filters, kernelsize, inputshape, activation='relu') [2]

Les Arguments utilisé aux CNN

généralement les arguments utilisé au niveaux de réseau de neurones de convolution :

1. **filtres :** la dimensionnalité de l'espace de sortie (c'est-à-dire le nombre de filtres de sortie dans la convolution).
2. **kernelsize:** liste d'un seul entier, spécifiant la longueur de la fenêtre de convolution 1D.
3. **Activation:** Les activations peuvent être utilisées via une couche d'activation ou via l'argument d'activation pris en charge par toutes les couches .

1.3.11 Mettre le modèle à l'entrainement sur les données

Avant de passé le modèle à l'entrainement sur l'ensemble de données qui nous avons lire a partir **dataSet** vous devez configurer le processus d'apprentissage, qui s'effectue via la méthode de **compilation** [2] , comme suit :
model.compile(optimizer='rmsprop', loss='mse') [2]
après l'étape de compilation en passe directement à l'étape de l'entrainement sur l'ensemble de données ou qui 'il s'appelle l'ensemble d'apprentissage pour construire et évaluer le modèle de prédiction par une grand partie de notre données et lasse l'autre pour le test et l'évaluation [5]. comme suit :
model.fit(X ,Y,batchsize=1,epochs=100) [2]

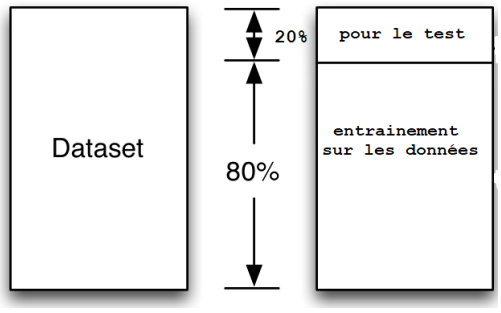


Figure 1.10: L'entrainement et l'évaluation sur les données

1.3.12 La prédiction

Après la phase de test nous atteindrons notre objectif consistant à établir un modèle d'apprentissage en profondeur basé sur un réseau de neurones convolutifs pour utilisé sur la modélisation Qsar pour la prédiction de l'activité biologique,enfin Génère des prédictions de sortie pour les échantillons d'entrée comme suit :
predict(X, batchsize=None, verbose=0) [2]

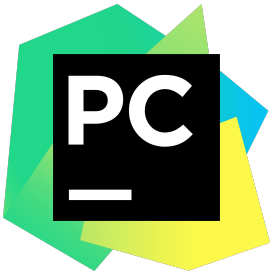
1.4 Implémentation

...

1.5 Les outils utilisés

Au cours de cette étape, nous soulignerons les divers outils que nous avons utilisés pour mener à bien notre projet concerné l'apprentissage en profondeur pour la prédiction de l'activité biologique.

1.5.1 Définition Pycharm



PyCharm est un environnement de développement intégré utilisé pour programmer en Python. Il permet D'analyse de code et contient un dèbogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions [7].

1.5.2 Pourquoi nous avons choisi Pycharm ?

Dans PyCharm, nous pouvez voir où et comment sont définis les symboles, tels que les balises, les classes, les champs, les méthodes ou les fonctions, dans notre projet. À cette fin, l'IDE propose la fenêtre de définition rapide [7].
Vu que l'API Python de TensorFlow est la plus complète, nous l'avons choisi pour travailler et donc c'est pour ça on a choisi Pycharm un environnement de développement dédié au langage Python.

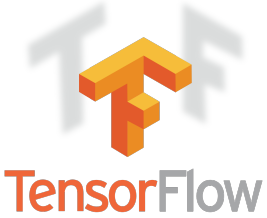
1.5.3 Définition Anaconda

ANACONDA Anaconda est une distribution libre , des langages de programmation Python , appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique) [12].

1.5.4 Pourquoi nous avons choisi Anaconda ?

Anaconda est le moyen le plus simple d'exercer la science des données Python et l'apprentissage automatique sur Windows. Il s'agit du standard de l'industrie en matière de développement, de test et de formation sur un seul ordinateur [3]. une distribution Python libre qui intègre directement un grand nombre de packages.

1.5.5 Plateforme de développement (TensorFlow)



Nous avons utilisé dans notre travail TensorFlow qui est un framework de programmation pour le calcul numérique, développé par l'équipe Google Brain et qui a été rendu Open Source par Google en Novembre 2015. La version 1.0.0 a été lancée en février 2017 [1]. Depuis sa libération, TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des frameworks les plus utilisés pour le Deep Learning, comme le montrent les dernières comparaisons [13].
TensorFlow est très populaire pour ses nombreux avantages qu'on cite ci-dessous:

1. Multi-plateformes (Linux, Mac OS, et même Android et iOS).
2. APIs en Python, C++, Java et Go (l'API Python est plus complète cependant, c'est sur celle-ci que nous allons travailler).
3. Temps de compilation très courts.
4. Supporte les calculs sur CPU, GPU et même le calcul distribué sur cluster.
5. Documentation extrêmement bien fournie avec de nombreux exemples et tutoriels.

TensorFlow est aujourd'hui particulièrement utilisé pour l'apprentissage en profondeur, et donc les réseaux de neurones. Son nom est notamment inspiré du fait que les opérations courantes sur des réseaux de neurones sont principalement faites via des tables de données multi-dimensionnelles, appelées Tenseurs (Tensor). Un Tensor à deux dimensions est l'équivalent d'une matrice) [13].
Aujourd'hui, les principaux produits de Google sont basés sur TensorFlow: Gmail, Google Photos, Reconnaissance de voix,. . . .,etc.

Bibliography

[1] Hong-Yunn Chen et al. Tensorflow–a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] François Chollet. Keras documentation. *keras. io*, 2015.

[3] Gilbert Gede, Dale L Peterson, Angadh S Nanjangud, Jason K Moore, and Mont Hubbard. Constrained multibody dynamics with python: From symbolic equation generation to publication. In *ASME 2013 international design engineering technical conferences and computers and information in engineering conference*, pages V07BT10A051–V07BT10A051. American Society of Mechanical Engineers, 2013.

[4] DE Hardt, E Constantine, and A Wright. A model of the sequential bending process for manufacturing simulation. *Journal of engineering for industry*, 114(2):181–187, 1992.

[5] Jeff Heaton. *Introduction to neural networks with Java*. Heaton Research, Inc., 2008.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Kahina Machouche, Kamel Amroun, et al. *Conception et réalisation d'une application Android pour la gestion des ressources humaines synchronisé avec l'ERP Odoo*. PhD thesis, Université Abderrahmane Mira-Bejaia, 2018.

[8] Konstantinos Makantasis, Konstantinos Karantzalos, Anastasios Doulamis, and Nikolaos Doulamis. Deep supervised learning for hyperspectral data classification through convolutional neural networks. In

2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pages 4959–4962. IEEE, 2015.

[9] Chintan Bimal Maniyar, Chintan M Bhatt, Tejas Nimeshkumar Pandit, and Dewanshi Harishankar Yadav. Cheerbot: A step ahead of conventional chatbot. In *Next-Generation Wireless Networks Meet Advanced Machine Learning Applications*, pages 306–322. IGI Global, 2019.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[11] Guido Van Rossum and Fred L Drake. *The python language reference manual*. Network Theory Ltd., 2011.

[12] Guido van Rossum et al. The python language. *See http://www. python. org*, 1990.

[13] Xenia Salinas Ventalló. Spatio-temporal emotion recognition.